Microsistem cu microprocesorul 8086

- ***Numele universității:***

  Universitatea Politehnica Timișoara
- ***Numele facultății:***

  Facultatea de Automatică și Calculatoare
- ***Numele disciplinei:***

  Proiectarea cu Microprocesoare
- ***Numele proiectului:***

  Microsistem cu microprocesorul 8086
- ***Numele autorului:***

  Marian Flavius-Andrei
- ***Anul universitar:***

  Anul 3
- ***Seria și grupa:***

  Tehnologia Informației, Grupa 2.1

Tema proiectului

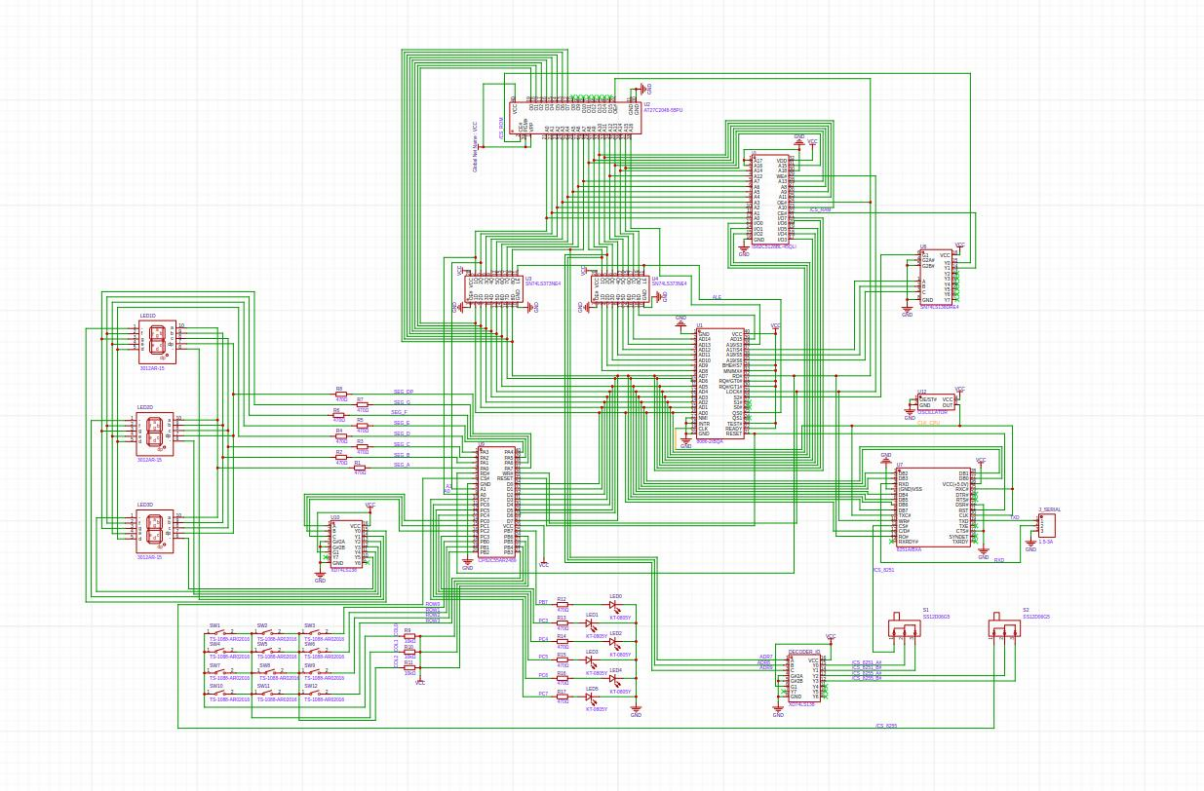
Scopul proiectului este proiectarea unui microsistem bazat pe microprocesorul 8086, care să includă:

- unitate centrală cu microprocesorul 8086
- 128 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
- 6 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:

- rutinele de programare ale circuitelor 8251 şi 8255;
- rutinele de emisie/ recepţie caracter pe interfaţa serială;
- rutina de emisie caracter pe interfaţă paralelă;
- rutina de scanare a minitastaturii;
- rutina de aprindere/ stingere a unui led;
- rutina de afişare a unui caracter hexa pe un rang cu segmente.

Descrierea hardware



Întreaga schemă

Unitatea centrală (CPU 8086)

Ce este și la ce o folosim?

- Microprocesorul 8086 este unitatea centrală de procesare a sistemului. El se ocupă cu execuția programului din memoria EPROM, accesează datele din memoria SRAM și comunică cu perifericele (8251, 8255, tastatura, cele 6 leduri și afișajul cu 7 segmenți). Practic toate celălalte circuite sunt legate în jurul acestuia.

Conexiuni importante (pin cu pin):

- **Alimentarea + clock-ul**
  - VCC - legat la 5V.
  - GND (piniul 1 și 20) - la masa comună.
  - CLK – legat la ieșirea oscilatorului de 1MHz. Practic asigură semnalul de clock pentru sincronizarea tuturor ciclurilor.

- **Magistrala de date**
  - AD0-AD15 – pinii dedicați pentru magistrala multiplexată de date + adresă.
    - Pentru adresă, pe aceste linii apare adresa A0-A15
    - Pentru date, pe aceste linii se transferă date D0-D15

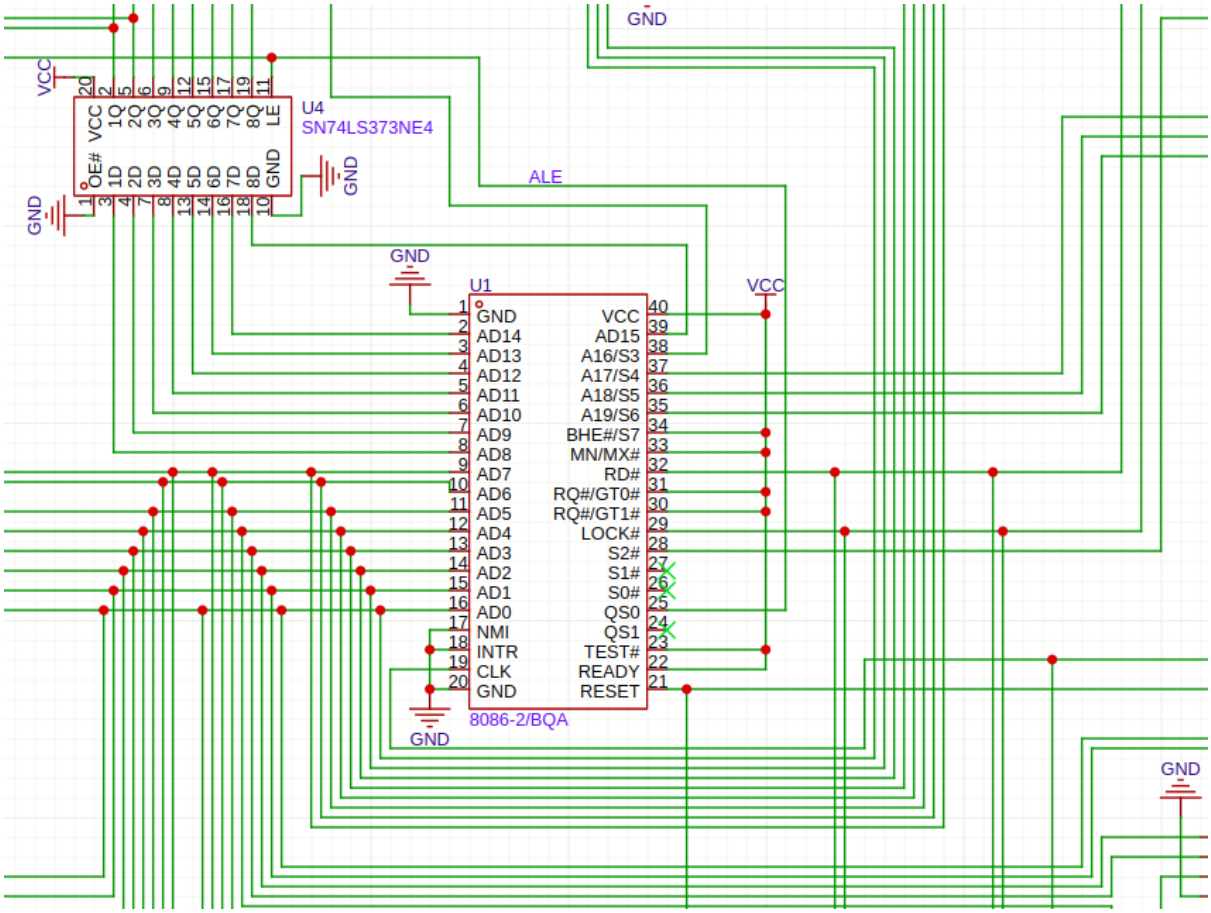
- **Alte adrese**
  - A16/S3-A19/S6S
    - Sunt conectați direct la EPROM și SRAM, ca să putem să adresăm întregrul spațiu de memorie necesar.
    - Tot aici, o parte din pini merg în chip-select.

- **Semnalul ALE + latch-urile de adresă**
  - ALE(Address Latch Enable):
    - Este legat la intrarea LE a celor două latch-uri (74LS373).
    - Când ALE-ul este 1, practic ambele latch-ul rețin adresa A0-A15, astfel adresa rămâne salvată la ieșirea latch-urilor, către memorie și periferice.

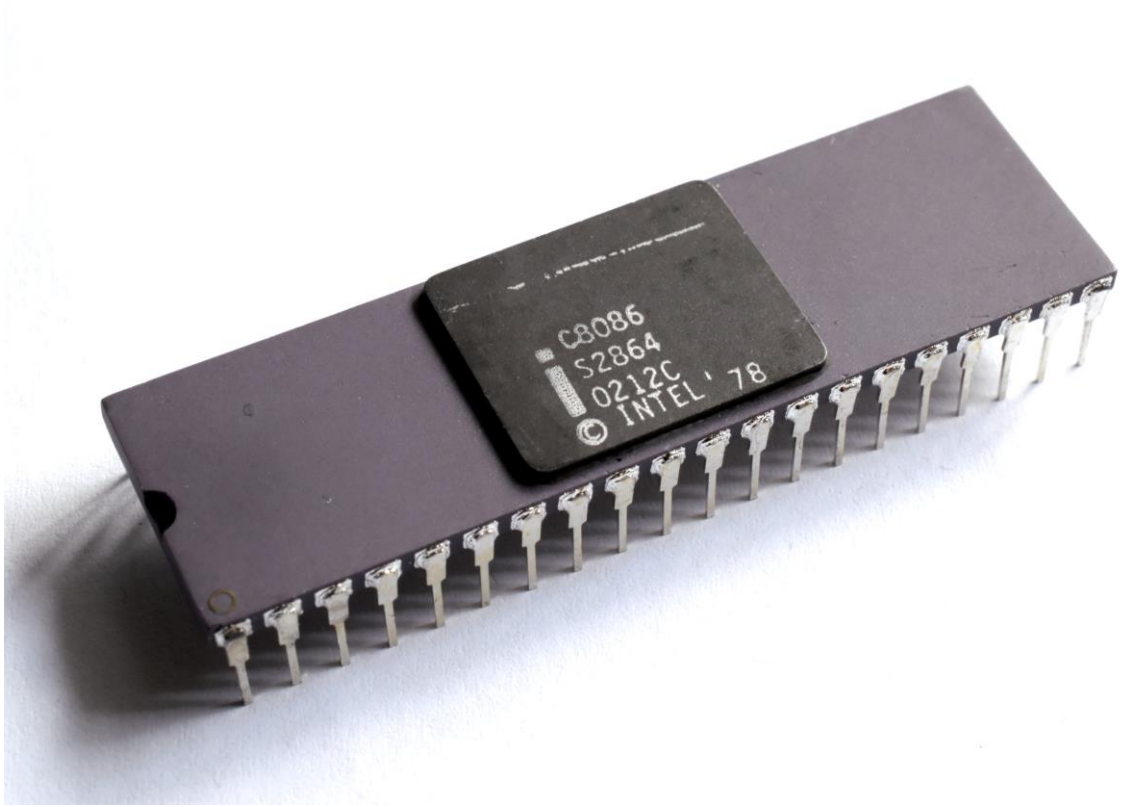
- **Semnale de control pentru memorie și I/O**
  - RD# - reprezintă citirea, merge la /OE# al EPROM/SRAM și la /RD# al perifericelor.
  - WR# - reprezintă scrierea, merge la /WE# al SRAM-ului și la /WR# al 8251/8255.
  - MN/MX# - pentru selectarea locului de lucru. Este conectat la VCC pentru a lucra în modul minim de lucru, unde folosim direct semnalul de /RD și /WR.

- **Reset, întreruperi și handshake:**
  - RESET – este pentru repornirea sau inițializa sistemul.
  - READY – pentru ca 8086 să poată insera stări de așteptare, pinul este legat la VCC.
  - INTR, NMI – nu sunt folosiți cei 2 pini, deci sunt puși pe No Connect Flag.

- **Alți pini:**
  - RQ/GT0#, RQ/GT1#, LOCK#, QS0, QS1, S0#, S1#, S2#, TEST#
    - O parte din pini sunt legați la VCC sau GND sau pus un No Connect Flag.



Microprocesorul 8086 din EasyEda



Microprocesorul 8086

Memoria EPROM (27C2048)

Ce este și la ce o folosim?

- Memoria ROM, conține rutinele de I/O și codul de inițializare (practic boot + rutine). Practic microprocesorul 8086 citește din acesta, fără să-l programeze. Cu memoria ROM putem să vedem aceeași adresă ca a microprocesorului.

Conexiuni importante (pin cu pin):

- A0 – A15:
  - Pinii de intrare, pentru adresa EPROM-ului, care sunt legații la magistrala de adresă a microprocesorului 8086.
  - Astfel cei 16 pini, îi împărțim în 2 latch-uri pentru a reține adresa, când ALE-ul este activ până la următorul ciclu.
  - EPROM-ul vede aceeași adresă ca și microprocesorul nostru 8086.

- O0 – O7:
  - O0 – O7 reprezintă ieșirile de data. Acestea sunt conectate la magistrala de date D0-D7 a sistemului.
  - Când ROM-ul este selectat și pinul RD# este și el activ, pe aceste linii apare octetul citit, pe care 8086 îl ia pe D0–D7.
  - Folosim fix primii 8 pini, pentru că lucrăm pe 8 biți (avem 128KiB).

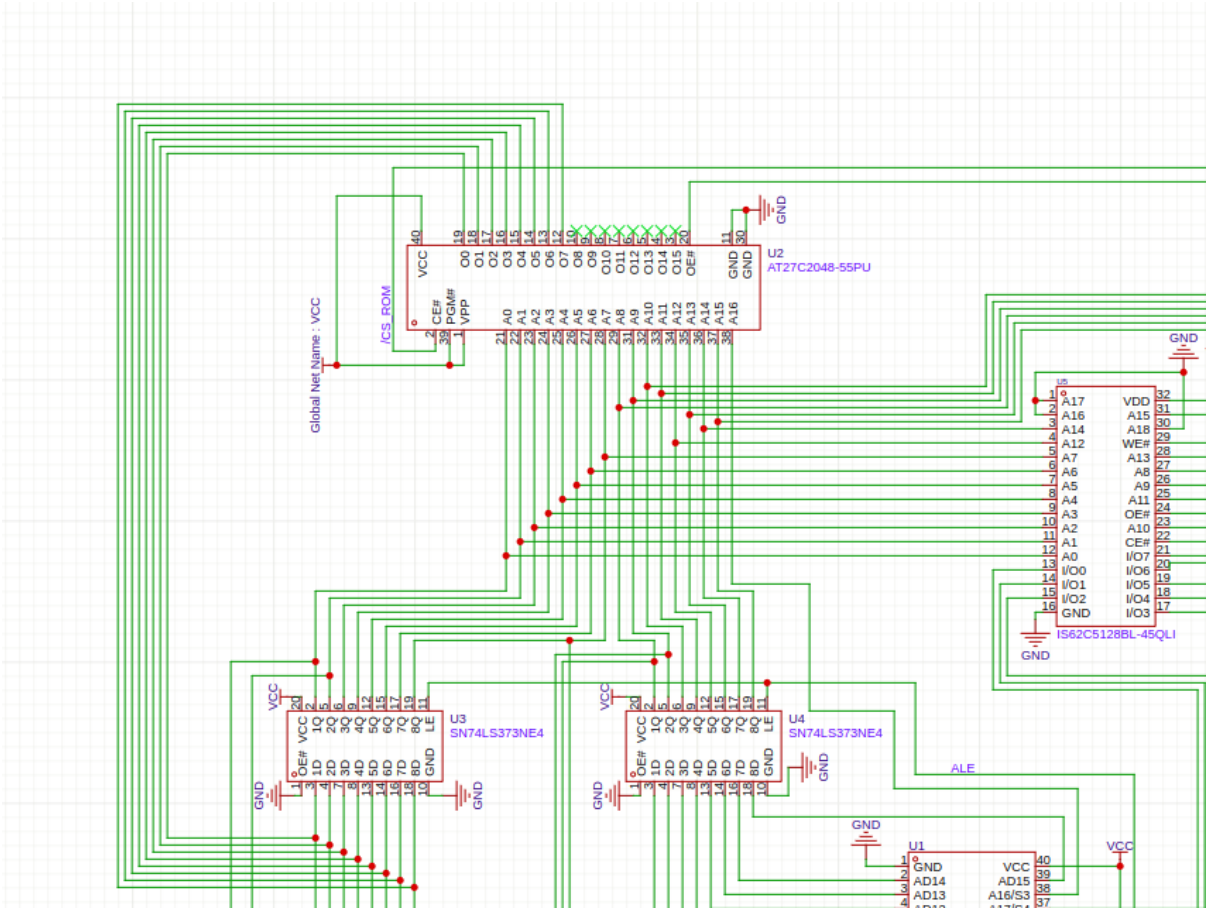
- Chip Enable / CS_ROM(CE#):
  - Pinul CE# este pentru semnalul de chip select, generat de decodorul de adrese.
  - /CS_ROM-ul este activ (0 logic), doar în intervalul de adrese rezervat pentru ROM.

- Output Enable (OE#):
  - OE# este legat la RD# de la 8086, iar când citirea este activă atunci se activează ieșirile EPROM-ului și se pun datele pe D0-D7.

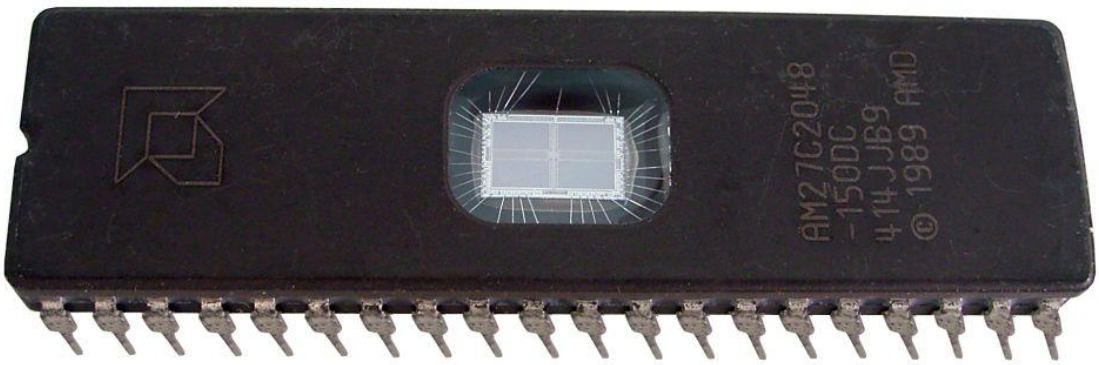
- VPP / PGM#:
  - Pin pentru programare, care este legat la VCC pentru a dezactiva funcționalitatea pentru modul de programare.

- VCC + GND:
  - Pinii pentru alimentare și masă.

- Pini nefolosiți:
  - O8 – O15 acești pinii nu îi folosim, deci vor fi puși cu un flag de tip No Connect Flag, pentru că noi lucrăm doar pe 8 biți



Memoria EPROM din EasyEda



Memori EPROM 27C2048

Memoria SRAM (62512)

Ce este și la ce o folosim?

- Memoria RAM, conține codul de inițializare și eventual rutinele de I/O. Procesorul 8086 doar citește din acest circuit, nu îl și programează. Pe scurt, memoria RAM este practic memoria de lucru, în care se scrie și citește.

Conexiuni importante (pin cu pin):

- A0 – A15:
  - Pinii pentru adrese, care sunt legați la magistrala de adrese. Practic RAM-ul vede, ce vede și 8086.

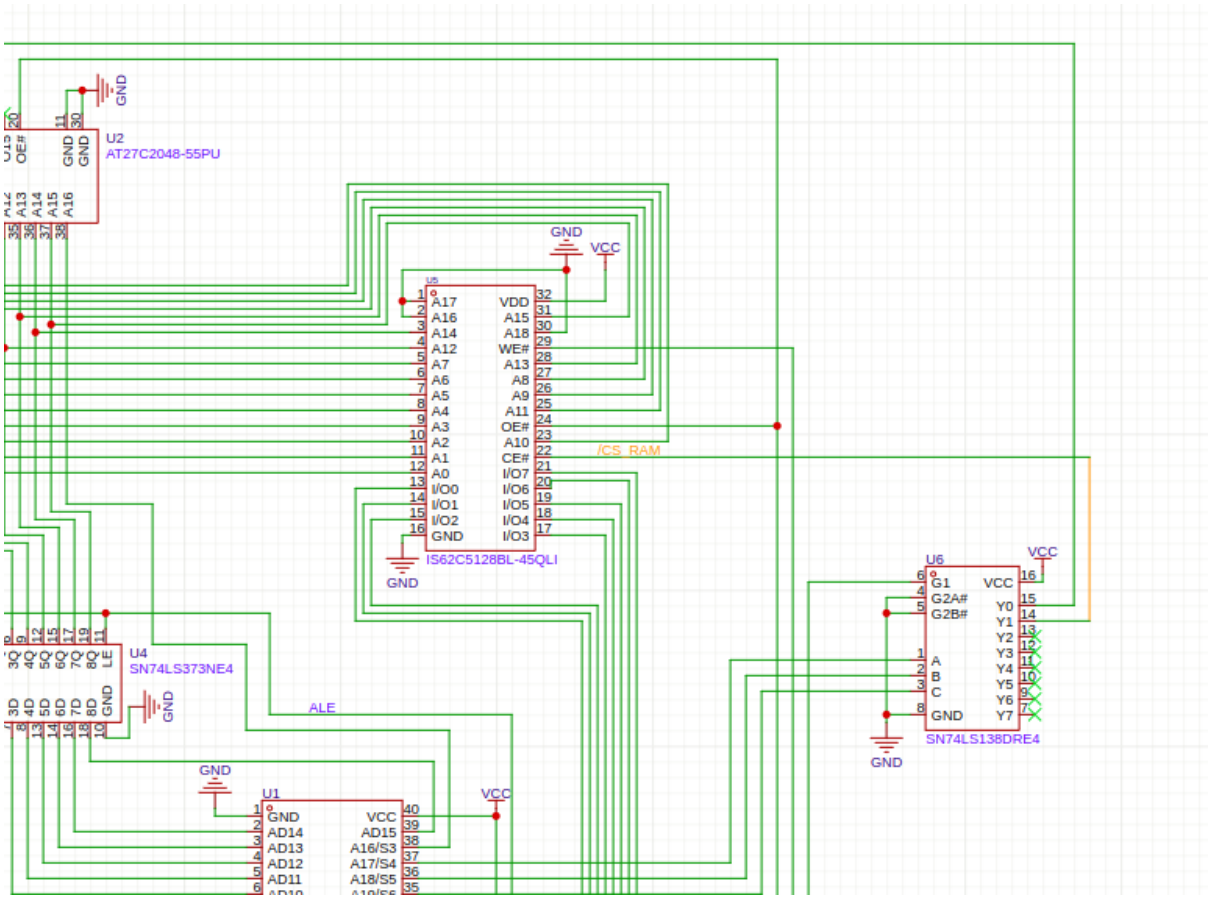
- I/O0 - I/O7:
  - Pinii pentru magistrala de date, care sunt conectați la pinii D0-D7.
  - Pentru citire, pinul MEMR# de la 8086 este activ, iar SRAM-ul pune octetul cerut pe D0-D7.
  - Pentru scriere, pinul MEMW# de la 8086 este activ, iar SRAM-ul ia datele de pe D0-D7 și le memorează la adresa selectată.

- /CS_RAM:
  - Semnalul pentru chip select, generat de decodorul de adrese.
  - Când /CS_RAM este activ (este 0), doar când este în intervalul de adrese rezervate pentru RAM, în rest nu este activ, deci nu interferează cu EPROM-ul sau alte periferice.

- /OE și /WE
  - /OE - Output Enable este legat la MEMR# de la 8086 și permite ieșirea datelor pe pinii D0-D7 doar pentru citire.
  - /WE - Write Enable este egat la MEMW# de la 8086 și permite scrierea în RAM.
  - Astfel combinația /CS, /OE, /WE asigură că SRAM-ul nostru doar la combinația potrivită, adică adresa potrivită + operația potrivită.

- Alimentare, masă și pinii neutilizați:
  - VCC – este alimentarea.
  - GND – masa.

- A16, A17, A18 sunt legați la GND pentru că nu folosim. Noi dorim folosim doar primul block de 64 KiB.



Memoria SRAM din EasyEda

Memorie SRAM 62512

Harta memoriei:

Selecția memoriei ROM, cât și a memoriei RAM, se face cu un decodor de 3 la 8 (74LS138), comandat de biții superiori de pe magistrala de adrese (A19,A18,A17). Ieșirile decodorului sunt active și generează semnalele de chip select pentru RAM sau ROM.

Astfel, biții de intrare pentru decodor (A,B,C) vin de pe biții de adresă A19, A18 și A17, iar G1 este 1 pentru că trebuie să fie activ doar ciclurile de memorie.

Atunci spațiul de 1MB, este împărțit în fix 8 părți de câte 128KiB fiecare.

A19	A18	A17	Ieșire activă (LOW)	Interval adresă (hexa)
0	0	0	Y0 = 0	00000h - 1FFFFh
0	0	1	Y1 = 0	20000h - 3FFFFh
0	1	0	Y2 = 0	40000h - 5FFFFh
0	1	1	Y3 = 0	60000h - 7FFFFh
1	0	0	Y4 = 0	80000h - 9FFFFh
1	0	1	Y5 = 0	A0000h - BFFFFh
1	1	0	Y6 = 0	C0000h - DFFFFh
1	1	1	Y7 = 0	E0000h - FFFFFh

Ecuațiile:

- Y0 = !A19 + !A18 + !A17
- Y1 = !A19 + !A18 + A17
- Y2 = !A19 + A18 + !A17
- Y3 = !A19 + A18 + A17
- Y4 = A19 + !A18 + !A17
- Y5 = A19 + !A18 + A17
- Y6 = A19 + A18 + !A17
- Y7 = A19 + A18 + A17

Decodificarea memoriilor:

EPROM - 128KiB:

- $2^7 + 2^{10} = 2^{17}$ biți .
- Număr de locații în hexa: 20000h (se pune 1 pe poziția 17 – 0010 0000 0000 0000 0000).
- Spațiul de adresare este: 00000h (adresa de început) și 1FFFFh (adresa de final).

SRAM – 64 KiB:

- $2^6 + 2^{10} = 2^{16}$ biți.

- Număr de locații în hexa: 10000h (se pune 1 pe poziția 16 – 0001 0000 0000 0000 0000).
- Spațiul de adresare este: 40000h (adresa de început) și 4FFFFh (adresa de final).

	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	hex
EPROM	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**00000h**
	0	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1FFFFh**
SRAM	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**40000h**
	0	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**4FFFFh**

- CS_ROM = Y0 = !A19 + !A18 + !A17
- CS_RAM = Y1 = !A19 + !A18 + A17

Interfața serială (8251)

Ce este și la ce o folosim?

- Circuitul 8251, este dedicat pentru interfața serială programabilă. În acest proiect, folosirea acestuia are ca și scop comunicarea cu unitatea centrală (8086), prin intermediul unui adaptor USB-TTL sau un alt tip de echipament serial.

Conexiuni importante (pin cu pin):

- Pentru magistrala de date (DB0-DB7):
  - Cei 8 pini sunt legați la D0-D7 ale sistemului, comun cu EPROM, SRAM și 8255 (interfața paralelă).
  - Motivul pentru care sunt legate așa, pentru că trebuie să apară în spațiul de I/O al lui 8086, rezultând că trebuie să fie pe aceeași magistrală de date cu restul perifericelor.
- RD#, WR#, CS#, C/D#
  - RD#, WR# vin din /RD și /WR ale 8086, combinate cu logica pentru I/O.
  - CS# vine de la ieșirea decodorului IO (74LS138), prin microcomutatorul S1.

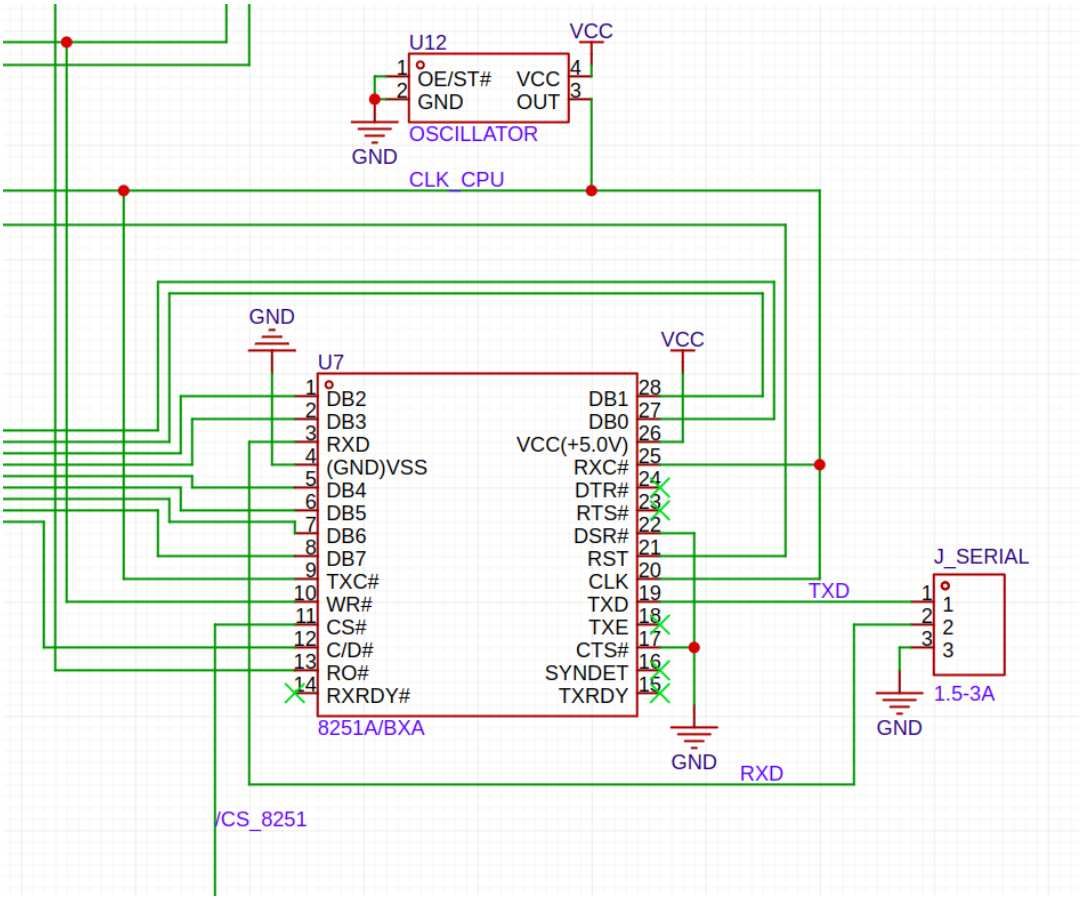
- C/D# selectează între registrul de date și cel de comandă.
- Motivul este că numai când adresa se află în zona 0AF0H – 0AF2H sau 0BF0H – 0BF2H și /WR sau /RD sunt active, 8251 ar trebui să răspundă pe magistrală

- RXD, TXD
  - TXD la pin 1 la conectorul J_SERIAL
  - RXD la pin 2 la conectorul J_SERIAL
  - GND comun la pin 3 J_SERIAL
  - Pentru conectarea mai ușoară a unui adaptor USB-TTL .
- CLOCK
  - CLK_8251 este clock-ul derivat din cel al sistemului principal.
  - 8251 are nevoie de un clock, practic stabil pentru sincronizariile interne.

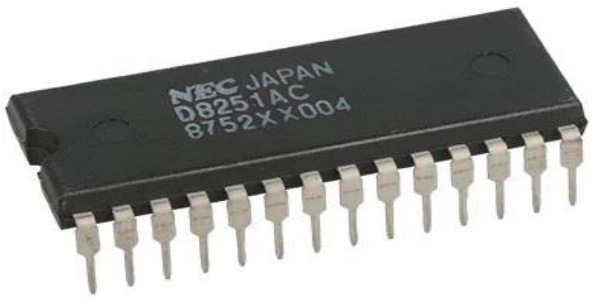
- Restul pinilor:
  - Restul de pini rămași o să fie legați la VCC sau GND sau cu No Connect Flag
  - Am ales metoda aceasta, pentru a simplifica acest proiect.

Legătura între operaţiile realizate de circuit şi starea terminalelor de comandă:

/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	Citere a datei.
0	1	0	1	Scriere a cuvintelor de comandă.
0	1	0	0	Scriere a datei.



Schema circuit 8251 din EasyEda



Circuitul 8251

Interfața paralelă (8255)

Ce este și la ce o folosim?

- Circuitul 8255, este dedicat pentru interfața paralelă. În acest proiect, respectivul circuit are un rol important, deoarece este folosit pentru afișajul cu 7 segmenți, minitastatura de 3x4 și cele 6 led-uri pentru status. Acesta este un circuit cu 3 porturi paralele, fiecare fiind de câte 8 biți (PA,PB,PC).

Conexiuni importante (pin cu pin):

- Pentru magistrala de date (D0-D7):
  - Cei 8 pini sunt legați la D0-D7 ale sistemului, comun cu EPROM, SRAM și 8251 (interfața paralelă).
  - Motivul pentru care sunt legate așa, pentru că trebuie să apară în spațiul de I/O al lui 8086, rezultând că trebuie să fie pe aceeași magistrală de date cu restul perifericelor.
  - Practic este același caz ca și la circuitul 8251 (DB0 – DB7)

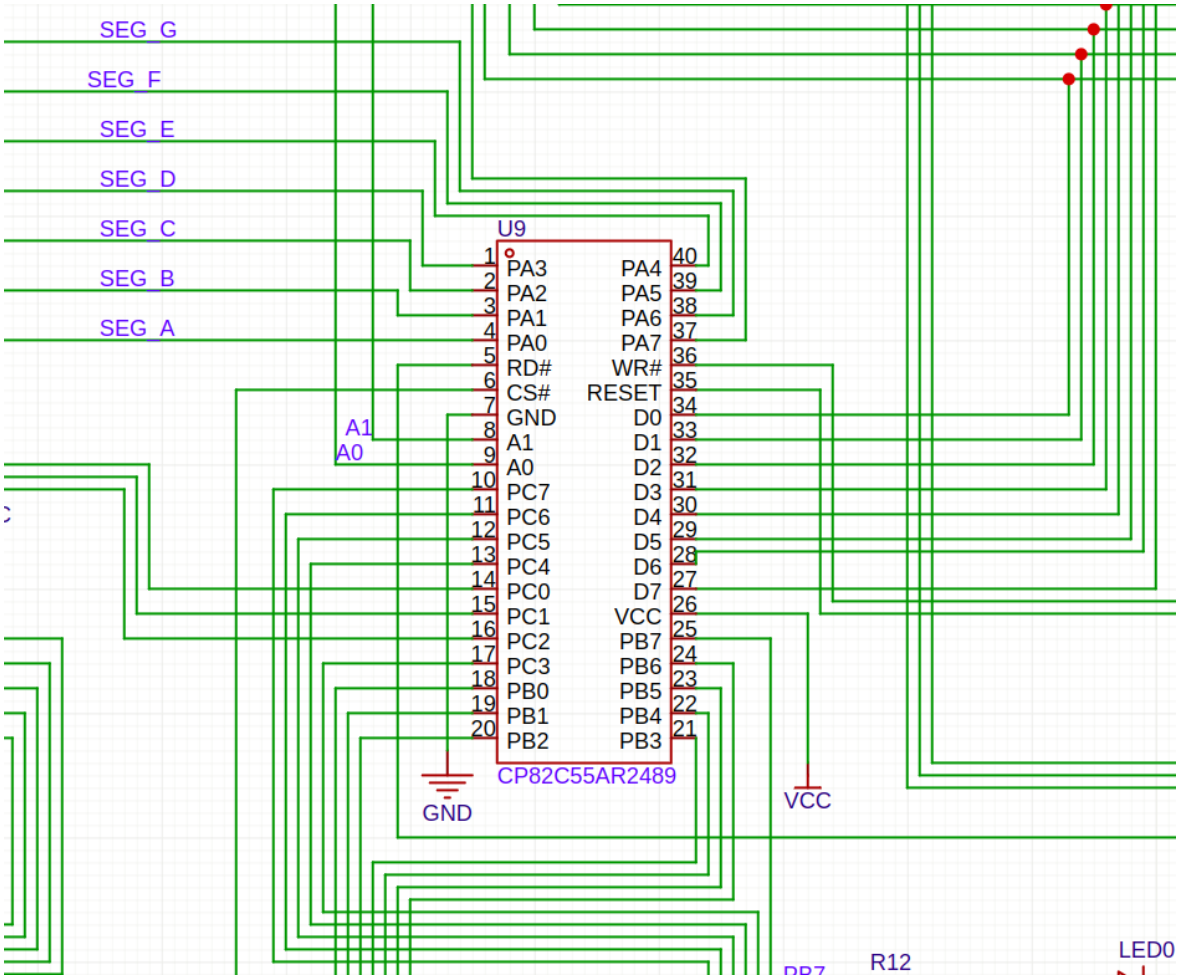
- RD#, WR#, CS#:
  - RD#, WR# vin din /RD și /WR ale 8086, combinate cu logica pentru I/O.
  - CS# vine de la ieșirea decodorului IO (74LS138), prin microcomutatorul S2.
  - Circuitul 8255 trebuie accesat și programat ca și port de I/O.

- A0, A1:
  - A0 și A1 ajută la selectarea registrului:
    - A1 A0 = 00 -> Port A
    - A1 A0 = 01 -> Port B
    - A1 A0 = 10 -> Port C
    - A1 A0 = 11 -> Registrul de control
- Port-ul A/B/C:
  - Dedicat pentru afișajul cu 7 segmenți.
  - PA0 – PA6 sunt segmentele SEG_A – SEG_G pentru fiecare led.
  - Pentru fiecare led, punem câte o rezistență de 470 ohm.
  - PA7 este pentru led-ul punct (SEG_DP).
  - un subset de biți - ROW0…ROW3, COL0…COL2 pentru tastatură.
  - alți biți - intrări A,B,C la 74LS138 (select digit).
  - alți biți - LED-uri (PB7, PC3–PC7).
  - Astfel, distribuția aleasă, permite folosirea unui singur 8255 pentru toate perifericile.

Semnificațiile terminalelor:

/CS	/RD	/WR	A1	A0	Operația
0	1	0	0	0	Scrie în portul A
0	1	0	0	1	Scrie în portul B
0	1	0	1	0	Scrie în portul C
0	1	0	1	1	Scrie 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 operaţie – magistrala de date este în a 3-a stare
0	1	1	X	X	Fără operaţie – magistrala de date este în a 3-a stare
1	X	X	X	X	Magistrala de date este în a 3-a stare

Circuitul 8255



Schemă circuit 8255 din EasyEda

Astfel interfața serială este plasată în zona 0AF0H – 0AF2H sau 0BF0H – 0BF2H și interfața paralelă este plasată în zona 0CF0H – 0CF2H sau 0DF0H – 0DF2H.

Modul	Adresa	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
8251 S1=A	0AF0h	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0
8251 S1=A	0AF1h	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	1
8251 S1=A	0AF2h	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0

8251 S1=B	0BF0h	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0
8251 S1=B	0BF1h	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	1
8251 S1=B	0BF2h	0	0	0	0	1	0	1	1	1	1	1	1	0	0	1	0
8255 S2=A	0C70h	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0
8255 S2=A	0C71h	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1
8255 S2=A	0C72h	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	0
8255 S2=A	0C73h	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1
8255 S2=A	0C74h	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	0
8255 S2=A	0C75h	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	1
8255 S2=A	0C76h	0	0	0	0	1	1	0	0	0	1	1	1	0	1	1	0
8255 S2=B	0D70h	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0
8255 S2=B	0D71h	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	1
8255 S2=B	0D72h	0	0	0	0	1	1	0	1	0	1	1	1	0	0	1	0
8255 S2=B	0D73h	0	0	0	0	1	1	0	1	0	1	1	1	0	0	1	1
8255 S2=B	0D74h	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	0
8255 S2=B	0D75h	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	1

8255 S2= B	0D76 h	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1	0

Ecuații:

- Pentru 8251, cât și pentru 8255 aproape toți pinii au valori egale, doar pinul A8 diferă care este 0 sau 1.
- Pentru 8251 cu A2, A1, A0 selectăm sub-adresa din x0, x1, x2.

  8251 – S1 - A: A11 + !A10 + A9 + **!A8** + A7 + A6 + A5 + A4

  8251 – S1 - B: A11 + !A10 + A9 + **A8** + A7 + A6 + A5 + A4

- Pentru 8255 selecția se face cu ajutorul lui A0 și A1.
  - A1 A0 == 00 - PA
  - A1 A0 == 01 - PB
  - A1 A0 == 10 - PC
  - A1 A0 == 11 - Control

  8255 – S2 - A: A11 + A10 + !A9 + **!A8** + !A7 + A6 + A5 + A4

  8255 – S2 - B: A11 + A10 + !A9 + **A8** + !A7 + A6 + A5 + A4

Afișajul cu 7 segmenți (3 x 3012AR-15 + 74LS138)

Ce este și la ce o folosim?

- Reprezintă un afișaj format din 3 digits cu 7 segmente, multiplexate. Îl folosesc pentru a afișa valori (ex. codul tastat, un număr, un registru).

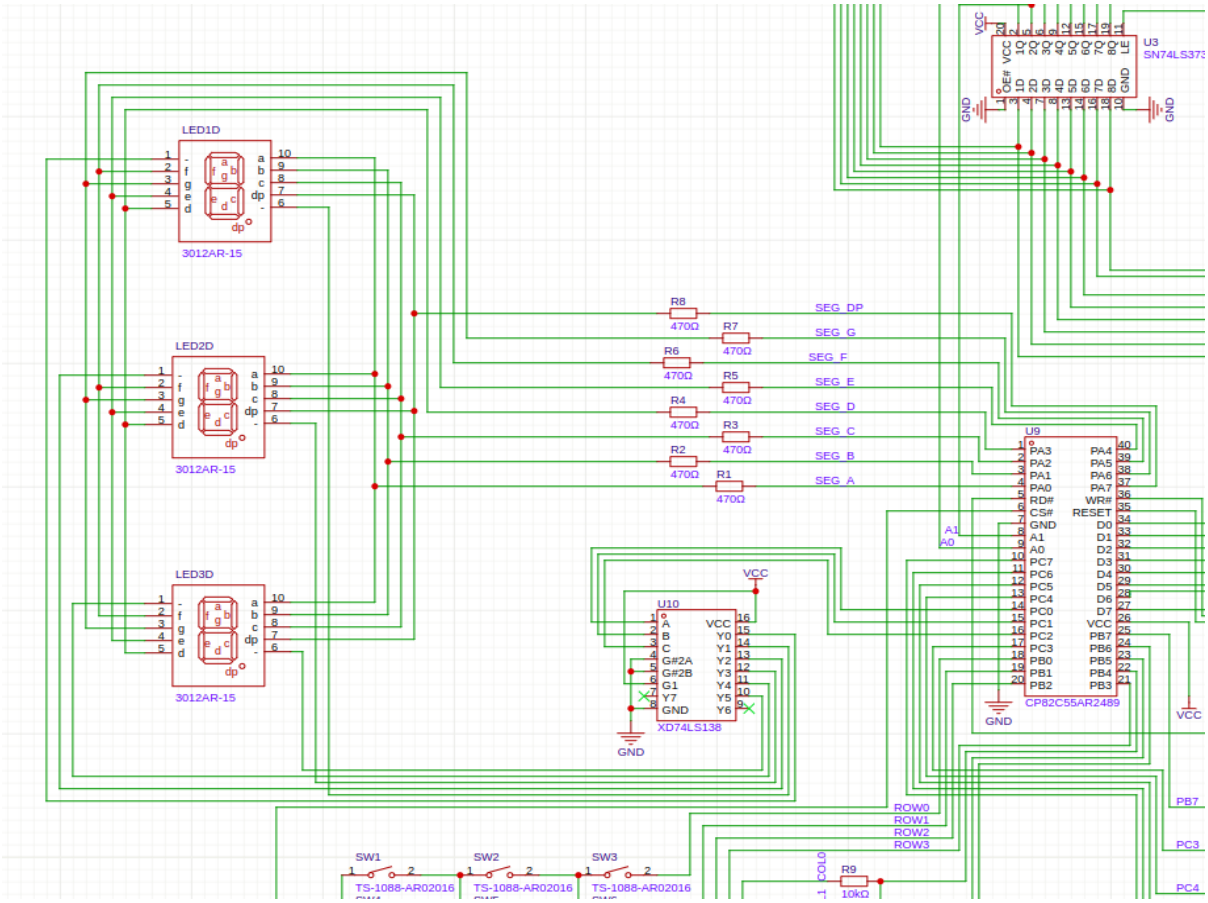
Conexiuni importante (pin cu pin):

- Segmentele a, b, c, d, e, f, g, dp:
  - Toate cele 3 afișaje au segmentele legate la aceleași magistrale(SEG_A – SEG_DP).
  - Pentru fiecare led, punem câte o rezistență de 470 ohm, astfel limităm curentul pentru led-uri.
  - Motivul pentru care dorim să împărțim la comun fiecare segment, pentru fiecare digit este pentru economisirea de pini.

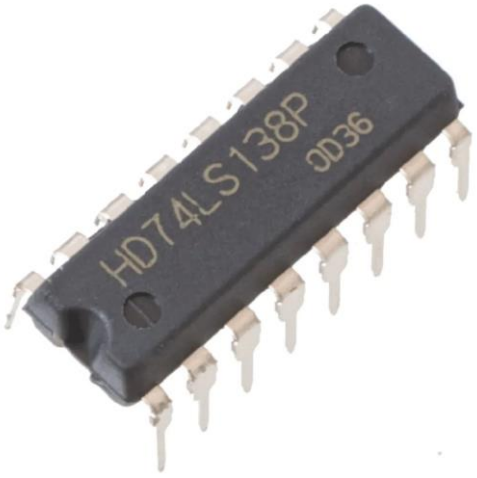
- Selectarea digitului – 74LS138
  - Pentru intrările A, B, C folosim 3 biți de la 8255 (PC0, PC1, PC2).
  - Ieșirile Y0, Y1, Y2 sunt comune celor 3 digits, unul pentru fiecare digit.
  - G1, G2A#, G2B# setate pentru activ permanent (G1 = VCC, G2A#/G2B# = GND).
  - Se folosesc fix 3 biți de la circuitul 8255 pentru a activa pe rând digitul de 0, 1 sau 2.

Pentru a, b, c, d, e, f, g și dp corespund SEG_A SEG_B SEG_C SEG_D SEG_E SEG_F SEG_G SEG_DP. Având un anod comun vom avea astfel: 0 = ON sau 1 = OFF.

Ch	dp	g	f	e	d	c	b	a	Valoare(hex)
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90
A	1	0	0	0	1	0	0	0	88
b	1	0	0	0	0	0	1	1	83
C	1	1	0	0	0	1	1	0	C6
d	1	0	1	0	0	0	0	1	A1
E	1	0	0	0	0	1	1	0	86
F	1	0	0	0	1	1	1	0	8E



Schemă afișaj cu 7 segmente din EasyEda



Circuit 74LS138

Minitastatură 3x4

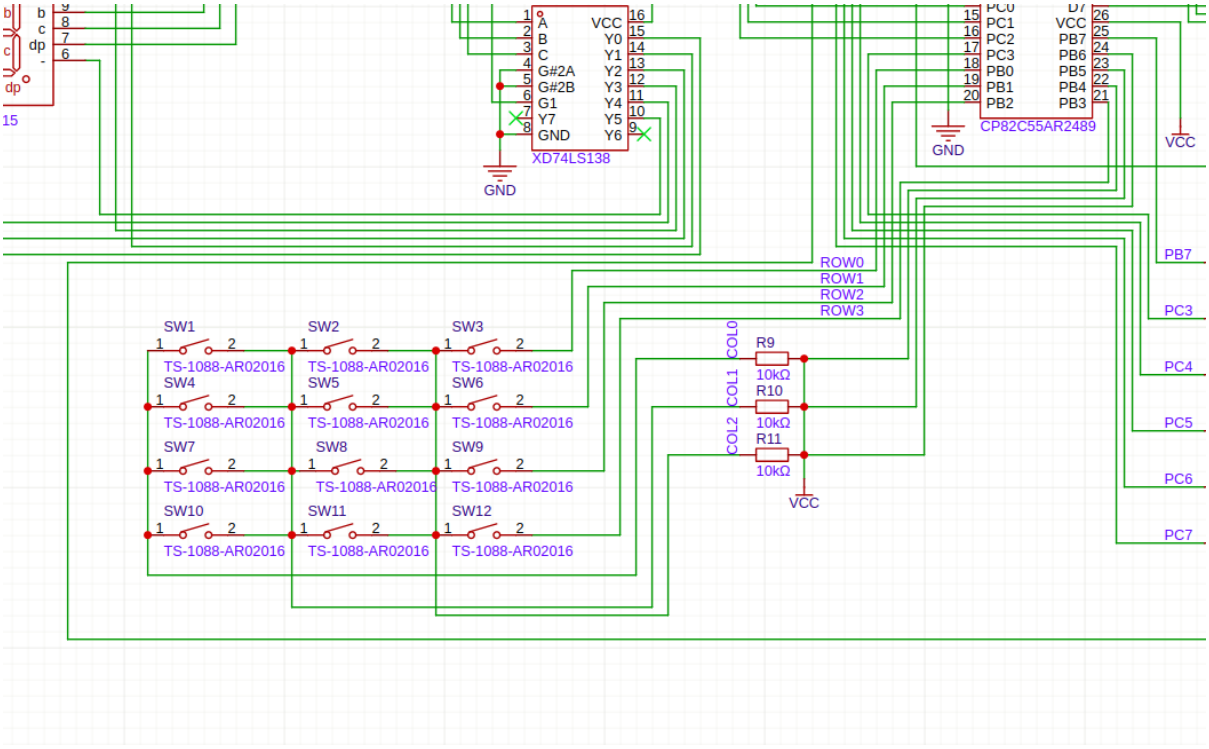
Ce este și la ce o folosim?

- O tastatură sub forma unei matrici, cu 12 butoane, unde avem 3 coloane și 4 rânduri. Folosită pentru introducerea de coduri sau comenzi.

Conexiuni importante (pin cu pin):

- Pinul 1 al fiecărui buton este conectat pe rânduri:
  - SW1, SW2, SW3 - ROW0
  - SW4, SW5, SW6 - ROW1
  - SW7, SW8, SW9 - ROW2
  - SW10, SW11, SW12 - ROW3
- Pinul 2 al butoanelor e conectat pe coloane:
  - coloana 0: SW1, SW4, SW7, SW10 - COL0
  - coloana 1: SW2, SW5, SW8, SW11 - COL1
  - coloana 2: SW3, SW6, SW9, SW12 - COL2
- Rezistențele de 10 kilo - ohm sunt legate la VCC ca să țină coloanele la 1 logic în mod normal, iar apăsarea unei taste permite rândului pus la 0 să aducă coloana la 0 pentru detectare.

	COL0	COL1	COL2
ROW0	SW1	SW2	SW3
ROW1	SW4	SW5	SW6
ROW2	SW7	SW8	SW9
ROW3	SW10	SW11	SW12



Tastatură

6 led-uri

Ce este și la ce o folosim?

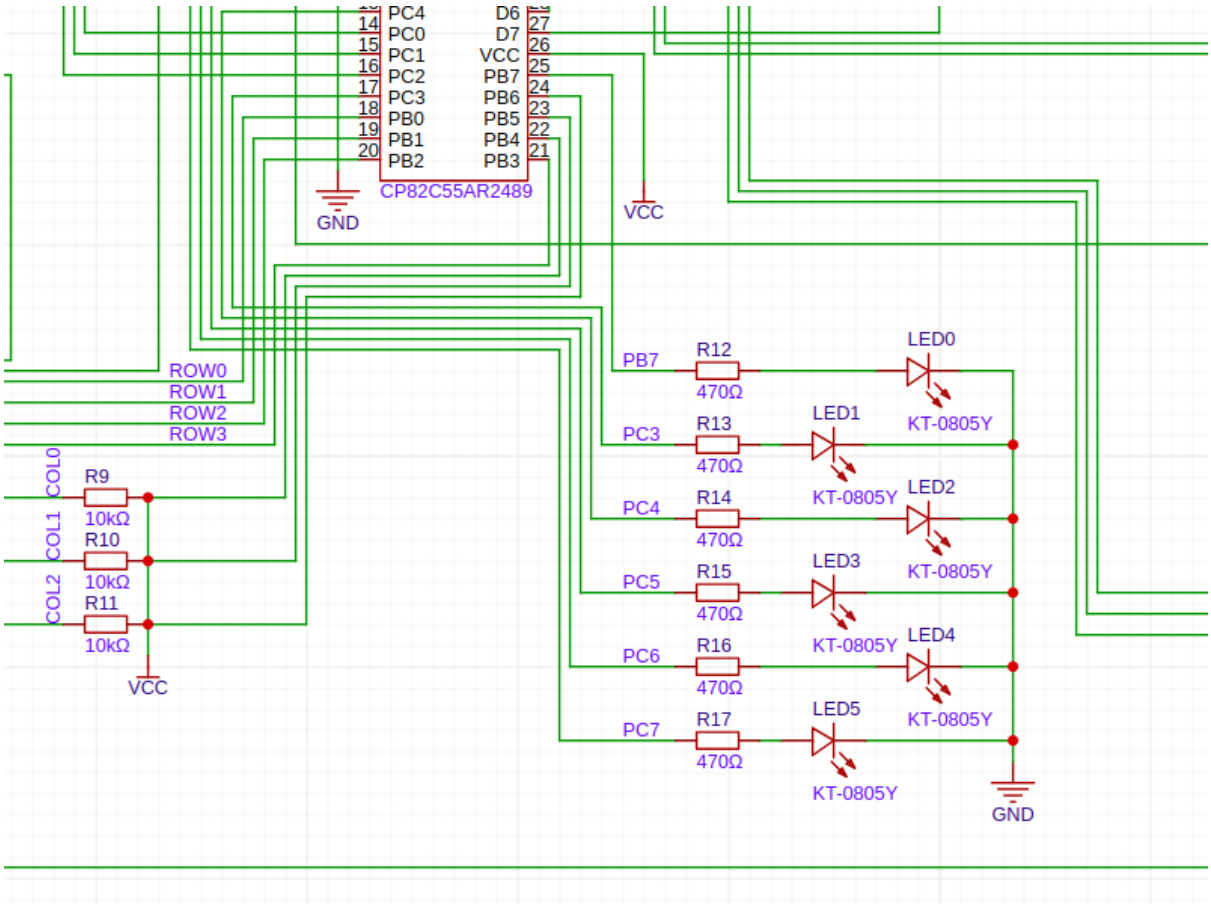
- Sunt 6 LED-uri pentru stare.

Conexiuni importante (pin cu pin):

- PB7, PC3 – PC7 pentru LED0 – LED5:
  - Fiecare linie trece printr-o rezistență de 470 ohm și apoi în anodul LED-ului.
  - Catodul LED-urilor este comun la GND.

Fiecare ieșire trece printr-o rezistență de 470Ω, iar apoi prin anodul LED-ului propriu-zis. Astfel, fiecare catod este legat la același GND. LED-ul se aprinde, atunci când primește 1 logic de pe pinul respectiv.

LED	Pin 8255 (net)	Rezistență	Tip comandă	Stare aprins
LED0	**PB7**	R12 = 470Ω	ieșire	**1**
LED1	**PC3**	R13 = 470Ω	ieșire	**1**
LED2	**PC4**	R14 = 470Ω	ieșire	**1**
LED3	**PC5**	R15 = 470Ω	ieșire	**1**
LED4	**PC6**	R16 = 470Ω	ieșire	**1**
LED5	**PC7**	R17 = 470Ω	ieșire	**1**



Schemă led-uri din EasyEda

Led + componentele sale

Decodor I/O + microcomutatorul S1, S2

Ce sunt și la ce le folosesc?

- Pentru decodor folosim un 74LS138, care generează semnale de chip select pentru 8251, cât și pentru 8255, în funcție de adresele A7–A9, iar S1, respectiv S2, sunt două microcomutatoare care mută perifericele între două ferestre de adresă, conform enunțului (0AF0H – 0AF2H sau 0BF0H – 0BF2H pentru 8251 și 0D70H – 0D76H sau 0C70H – 0C76H pentru 8255).

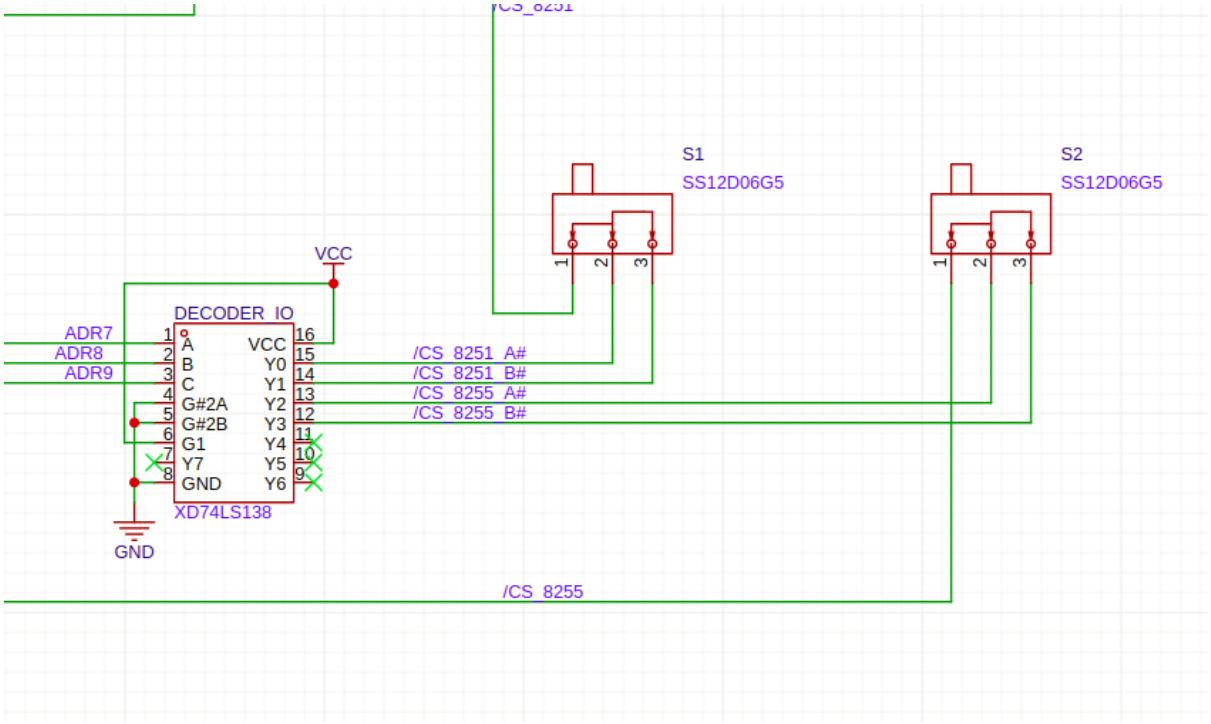
Conexiuni importante (pin cu pin):

- Intrări 74LS138
  - A, B, C <- ADR7, ADR8, ADR9.
  - G1, G2A#, G2B# setate, pentru activare în zona de adresă dorită, fiind în restul timpului ieșirile sunt inactive.

- Ieșirile 74LS138
  - Y0 -> /CS_8251_A# (0AF0H – 0AF2H)
  - Y1 -> /CS_8251_B# (0BF0H – 0BF2H)
  - Y2 -> /CS_8255_A# (0C70H – 0C76H)
  - Y3 -> /CS_8255_B# (0D70H – 0D76H)

- Microcomutatorul S1
  - Pinul 1 pentru intrarea lui /CS_8251
  - Pinul 2 pentru /CS_8251_A# (C70H – 0C76H))
  - Pinul 3 pentru /CS_8251_B# (0BF0H – 0BF2H)

- Microcomutatorul S2
  - Pinul 1 pentru intrarea lui /CS_8255
  - Pinul 2 pentru /CS_8255_A# (0AF0H – 0AF2H)
  - Pinul 3 pentru /CS_8255_B# (0D70H – 0D76H)



Schemă decoder + microcomutator

Descrierea software

1. Rutina de programare pentru circuitul 8251:
   - Datele iniţiale ale transferului: 8 biţi de date, fără paritate, factor de multiplicare 16, rata de transfer 9600 bps;
   - Adresele de port: **05A2H (06C2H)** – comenzi/ stări, **05A0H (06C0H)** date.
   - În DX vom avea adresa de port, pentru cuvintele de comandă/stare.
   - Cuvânt de comandă: 15H.
   - Cuvânt de mod: 0CEH.

```
INIT_8251:
        MOV DX, 05A2H ; port comenzi / stări 8251
        MOV AL, 0CEH ; cuvânt de mod
        OUT DX, AL
        MOV AL, 15H ; cuvânt de comandă:
        OUT DX, AL
        RET
```

2. Rutina pentru transmiterea unui caracter pe interfața serială:

```
TRANSMISIE_CARACTER:
TR: IN AL,DX ; citire şi testare rang TxRDY din cuvântul de stare
        RCR AL,1
        JNC TR
        MOV AL,CL ; se preia data din registrul CL
        MOV DX,05A0H ; portul pt date pt 8251
        OUT DX,AL
        RET
```

3. Rutina pentru recepționarea unui caracter pe interfața serială:

```
RECEPTIE_CARACTER:
REC:
    IN AL,DX ; citire şi testare rang RxRDY din cuvântul de stare
    RCR AL,2
    JNC REC
    MOV DX,05A0H
    IN AL,DX ; se preia data de la 8251
    MOV CL,AL ; se depune data în registrul CL
    RET
```

4. Rutina de programare pentru circuitul 8255:
   - Modurile de lucru:
     - 0 ieşire pentru porturile A şi B
     - 0 intrare pentru portul C inferior;
   - Adresele de port:
     - 0580H (0680H) – portul A
     - 0582H (0682H) – portul B
     - 0584H (0684H) – portul C
     - 0586H (0686H) – registrul de comandă (RCC)

   Selectarea uneia dintre cele două variante de adresare se realizează prin poziţia microcomutatorului din schema hardware.

```
MOV DX, 0586H
MOV AL, 81H
OUT DX, AL
RET
```

5. Rutina pentru transmiterea unui caracter pe interfața paralelă:

Rutina de emisie caracter pe interfața paralelă realizează transmiterea datelor prin scrierea directă a caracterului pe portul A al circuitului 8255. În proiectul realizat nu sunt utilizate semnale de handshake hardware, emisia fiind realizată fără testarea stării perifericului.

```
EMISIE_PARALEL:
        MOV DX, 0580H ; Port A - date paralele (8255)
        MOV AL, CL ;se ia caracterul din registrul CL
        OUT DX, AL ; transmitere caracter
        RET
```

6. Rutina pentru scanarea tastaturii:

```
SCAN_TASTATURA:
        MOV DX, 0582H ; Port B - ROW + COL

        ;=======ROW0======
        MOV AL, 11111110b ; PB0 = 0 - ROW 0 activ
        OUT DX, AL
        IN AL, DX

        TEST AL, 00010000b ; COL0 (PB4) - tasta 1
        JZ T1
        TEST AL, 00100000b ; COL1 (PB5) - tasta 2
        JZ T2
```

```
TEST AL, 01000000b ; COL2 (PB6) - tasta 3
JZ T3

;=======ROW1======
MOV AL, 11111101b ; PB1 = 0 - ROW 1 activ
OUT DX, AL
IN AL, DX

TEST AL, 00010000b ;tasta 4
JZ T4
TEST AL, 00100000b ;tasta 5
JZ T5
TEST AL, 01000000b ;tasta 6
JZ T6

;=======ROW2======
MOV AL, 11111011b ; PB2 = 0 - ROW 2 activ
OUT DX, AL
IN AL, DX

TEST AL, 00010000b ;tasta 7
JZ T7
TEST AL, 00100000b ;tasta 8
JZ T8
TEST AL, 01000000b ;tasta 9
```

```
        JZ T9

        ;=======ROW3======
        MOV AL, 11110111b ; PB3 = 0 - ROW 3 activ
        OUT DX, AL
        IN AL, DX

        TEST AL, 00010000b ;tasta *
        JZ TSTAR
        TEST AL, 00100000b ;tasta 0
        JZ T0
        TEST AL, 01000000b ;tasta #
        JZ THASH

        RET

T1: MOV AL, '1'
        RET
T2: MOV AL, '2'
        RET
T3: MOV AL, '3'
        RET
T4: MOV AL, '4'
        RET
T5: MOV AL, '5'
```

```
        RET
T6: MOV AL, '6'
        RET
T7: MOV AL, '7'
        RET
T8: MOV AL, '8'
        RET
T9: MOV AL, '9'
        RET
TSTAR: MOV AL, '*'
        RET
T0: MOV AL, '0'
        RET
THASH: MOV AL, '#'
        RET
```

7. Rutina pentru aprinderea/stingerea unui LED:

LED	Port	D7	D6	D5	D4	D3	D2	D1	D0	Hex
0	PB7	0	1	1	1	1	1	1	1	07FH
1	PC3	1	1	1	1	0	1	1	1	0F7H
2	PC4	1	1	1	0	1	1	1	1	0EFH
3	PC5	1	1	0	1	1	1	1	1	0DFH
4	PC6	1	0	1	1	1	1	1	1	0BFH
5	PC7	0	1	1	1	1	1	1	1	07FH

- LED-urile sunt conectate la atât la portul B, cât și la portul C ale circuitului 8255. Astfel,aprinderea unui LED se face prin aplicarea unui nivel logic 0 pe bitul dorit, iar stingerea prin nivel logic 1.

- Pentru aprinderea LED-ului:

```
LED0_ON:
        MOV DX, PORT_B
        MOV AL, 07FH ; PB7 = 0
        OUT DX, AL
        RET

LED1_ON:
        MOV DX, PORT_C
        MOV AL, 0F7H ; PC3 = 0
        OUT DX, AL
        RET

LED2_ON:
        MOV DX, PORT_C
        MOV AL, 0EFH ; PC4 = 0
        OUT DX, AL
        RET

LED3_ON:
        MOV DX, PORT_C
        MOV AL, 0DFH ; PC5 = 0
        OUT DX, AL
        RET
```

```
LED4_ON:
        MOV DX, PORT_C
        MOV AL, 0BFH ; PC6 = 0
        OUT DX, AL
        RET

LED5_ON:
        MOV DX, PORT_C
        MOV AL, 07FH ; PC6 = 0
        OUT DX, AL
        RET
```

- Pentru stingerea LED-urilor:

```
LED_OFF_ALL:
        MOV DX, PORT_B
        MOV AL, 0FFH
        OUT DX, AL

        MOV DX, PORT_C
        MOV AL, 0FFH
        OUT DX, AL
        RET
```

8. Rutina pentru afișarea caracterului dorit pe afișajul cu 7 segmenți:

Ch	dp	g	f	e	d	c	b	a	Valoare(hex)
0	1	1	0	0	0	0	0	0	0C0H
1	1	1	1	1	1	0	0	1	0F9H
2	1	0	1	0	0	1	0	0	0A4H
3	1	0	1	1	0	0	0	0	0B0H
4	1	0	0	1	1	0	0	1	099H
5	1	0	0	1	0	0	1	0	092H
6	1	0	0	0	0	0	1	0	082H
7	1	1	1	1	1	0	0	0	0F8H
8	1	0	0	0	0	0	0	0	080H
9	1	0	0	1	0	0	0	0	090H
A	1	0	0	0	1	0	0	0	088H
b	1	0	0	0	0	0	1	1	083H
C	1	1	0	0	0	1	1	0	0C6H
d	1	0	1	0	0	0	0	1	0A1H
E	1	0	0	0	0	1	1	0	086H
F	1	0	0	0	1	1	1	0	08EH

Tabelul:

```
TAB_7SEG DB 0C0H ;0
         DB 0F9H ;1
         DB 0A4H ;2
         DB 0B0H ;3
         DB 099H ;4
         DB 092H ;5
         DB 082H ;6
         DB 0F8H ;7
         DB 080H ;8
         DB 090H ;9
```

```
        DB 088H ;A
        DB 083H ;b
        DB 0C6H ;C
        DB 0A1H ;d
        DB 086H ;E
        DB 08EH ;F
```

Observație: Dacă AL nu se află între 0 și F nu se va afișa nimic.

```
AFISARE_HEX_7SEG:
        AND AL, 0FH ; 0 - F

        CMP AL, 0
        JZ DIG0
        CMP AL, 1
        JZ DIG1
        CMP AL, 2
        JZ DIG2
        CMP AL, 3
        JZ DIG3
        CMP AL, 4
        JZ DIG4
        CMP AL, 5
        JZ DIG5
```

```
CMP AL, 6
JZ DIG6
CMP AL, 7
JZ DIG7
CMP AL, 8
JZ DIG8
CMP AL, 9
JZ DIG9
CMP AL, 0AH
JZ DIGA
CMP AL, 0BH
JZ DIGB
CMP AL, 0CH
JZ DIGC
CMP AL, 0DH
JZ DIGD
CMP AL, 0EH
JZ DIGE
CMP AL, 0FH
JZ DIGF
RET
```

Codurile de afișare:

```
DIG0: MOV AL, 0C0H
      JMP OUT7
DIG1: MOV AL, 0F9H
      JMP OUT7
DIG2: MOV AL, 0A4H
      JMP OUT7
DIG3: MOV AL, 0B0H
      JMP OUT7
DIG4: MOV AL, 099H
      JMP OUT7
DIG5: MOV AL, 092H
      JMP OUT7
DIG6: MOV AL, 082H
      JMP OUT7
DIG7: MOV AL, 0F8H
      JMP OUT7
DIG8: MOV AL, 080H
      JMP OUT7
DIG9: MOV AL, 090H
      JMP OUT7
DIGA: MOV AL, 088H
      JMP OUT7
DIGB: MOV AL, 083H
```

```
        JMP OUT7
DIGC: MOV AL, 0C6H
        JMP OUT7
DIGD: MOV AL, 0A1H
        JMP OUT7
DIGE: MOV AL, 086H
        JMP OUT7
DIGF: MOV AL, 08EH
```

Scrierea pe portul A:

```
OUT7:
        MOV DX, PORT_A
        OUT DX, AL
        RET
```

Exemplu:

```
MOV AL, 8
CALL AFISARE_HEX_7SEG ; va afișa 8

MOV AL, 0FH
CALL AFISARE_HEX_7SEG ; F
```

Bibliografie

- https://easyeda.com/forum
- https://elupu.utcluj.ro/wp-content/uploads/2018/10/L2.pdf
- https://ro.scribd.com/doc/294138257/microprocesorul-8086
- https://www.microchip.com/content/dam/mchp/documents/OTH/ProductDocuments/DataSheets/doc0632.pdf
- https://www.wolfgangrobel.de/electronics/datasheets/eprom/27C2048.pdf
- https://ro.wikipedia.org/wiki/SRAM
- https://susta.cz/fel/74/pdf/74LS373.pdf
- https://www.ti.com/lit/ds/symlink/sn74s138a.pdf?ts=1765910152431&ref_url=https%253A%252F%252Fwww.google.com%252F
- https://www.elprocus.com/3-to-8-line-decoder-74ls138-ic-pin-configuration-features-circuit-and-applications/
- http://discipline.elcom.pub.ro/amp2/curs/8251.htm
- https://map.grauw.nl/resources/midi/intel_8251.pdf
- https://en.wikipedia.org/wiki/Intel_8255
- https://users.utcluj.ro/~apateana/PIO8255.pdf
- https://ro.wikipedia.org/wiki/Intel_8255
- https://sites.google.com/site/bazeleelectronicii/home/circuite-diverse/10-afisaj-cu-7-elemente
- https://en.wikipedia.org/wiki/USB-to-serial_adapter
- https://ro.wikipedia.org/wiki/Afi%C8%99aj_cu_%C8%99apte_segmente
- https://ocw.cs.pub.ro/courses/soc/laboratoare/05
- https://en.wikipedia.org/wiki/Chip_select
- https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds/how-to-use-them