UNIVERSITATEA POLITEHNICA TIMIȘOARA

MICROSISTEM CU MICROPROCESORUL 8086

PROIECTARE CU MICROPROCESOARE

Tănase Elena-Alexandra

An universitar: 2025 - 2026

TEMA PROIECTULUI

Să se proiecteze un microsistem cu următoarea structură:

- 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 și 8255
- rutinele de emisie / recepție caracter pe interfața serială
- rutina de emisie caracter pe interfață paralelă
- rutina de scanare a minitastaturii
- rutina de aprindere / stingere a unui led
- rutina de afișare a unui caracter hexa pe un rang cu segmente

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

1. UNITATEA CENTRALĂ CU MICROPROCESORUL 8086

Unitatea centrală a microsistemului este compusă din:

- microprocesorul Intel 8086
- generatorul de tact 8284A
- circuitele pentru amplificarea și separarea magistralelor de adrese și date (circuitul amplificator/separator bidirecțional 74LS245, circuitul registru 74LS373)

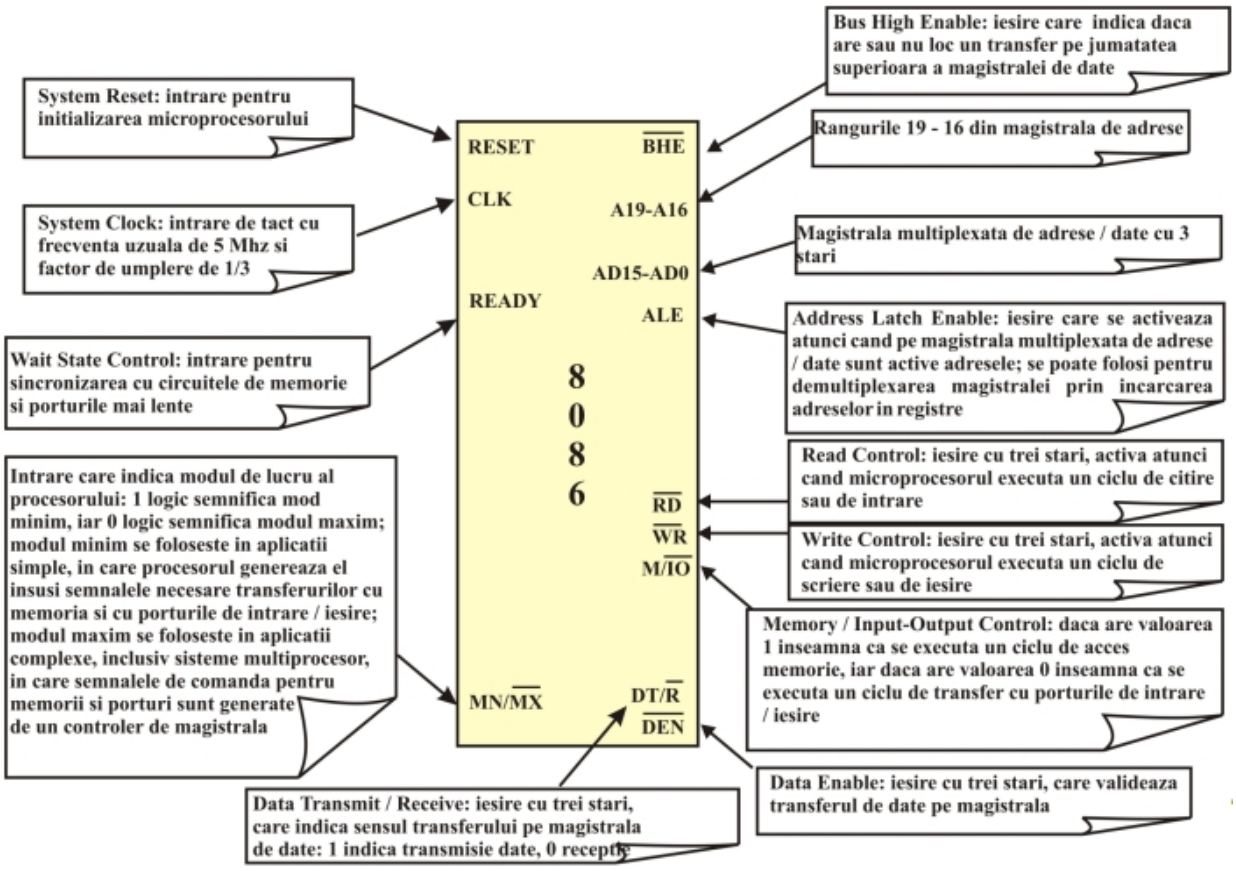
■ Microprocesorul Intel 8086:

Microprocesorul **Intel 8086** este un procesor pe **16 biți** care a stat la baza arhitecturii x86, devenind un reper important în evoluția calculatoarelor. De la lansarea sa din 1978, a fost folosit pe scară largă în PC-urile timpurii și în diverse sisteme industriale tocmai pentru că era fiabil, accesibil și foarte bine documentat.

Deși astăzi este depășit tehnologic, 8086 rămâne extrem de valoros în mediul academic, pentru că oferă un model clar și ușor de urmărit al modului în care funcționează un CPU real: segmentarea memoriei, registrele pe 16 biți și organizarea internă sunt intuitive și bine structurate.

În plus, arhitectura sa împărțită în Bus Interface Unit și Execution Unit ajută la înțelegerea ideii de paralelizare între fetch și execute, un principiu folosit și în procesoarele moderne. Chiar și astăzi, multe concepte din 8086 se regăsesc, sub forme evoluate, în procesoarele actuale compatibile x86.

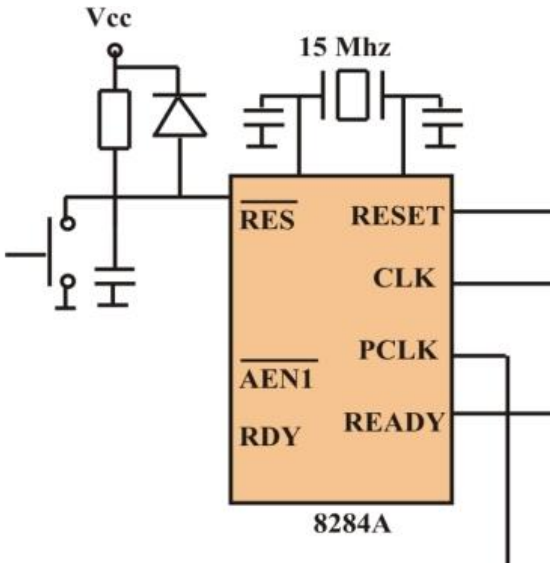
Principalele terminale ale microprocesorului 8086:



• Are registre interne pe 16 biți și o magistrală externă de date tot pe 16 biți, ceea ce îi permite prelucrarea mai rapidă a informațiilor față de procesoarele pe 8 biți din aceeași perioadă.
• Poate adresa direct până la 1 MB de memorie, datorită magistralei de adrese pe 20 de biți.
• Lucrează mai rapid datorită frecvenței de tact
• Magistralele de date și adrese sunt multiplexate, iar unele semnale externe au funcții multiple, ceea ce permite integrarea procesorului într-un pachet cu doar 40 de pini.
• Funcționează cu o singură tensiune de alimentare (+5 V), iar frecvența de lucru variază în funcție de versiune: 4 MHz, 5 MHz sau 8 MHz.

- Circuitul 8284A:

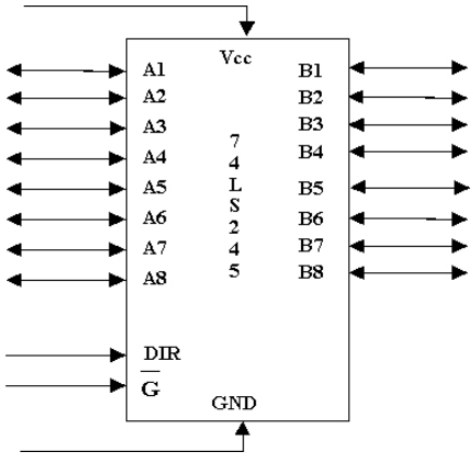
Generatorul de tact are rolul de a furniza semnalul de ceas atât pentru microprocesor, cât și pentru circuitele specializate de interfațare, asigurând funcționarea sincronizată a întregului sistem. Acesta generează și semnalul READY, prin care sincronizează microprocesorul cu perifericele mai lente, permițând extinderea ciclurilor de mașină atunci când este necesar. În plus, tot generatorul de tact produce semnalul de RESET, folosit pentru inițializarea microprocesorului; semnalul este aplicat într-o formă sincronizată cu tactul pentru a garanta pornirea corectă a sistemului. Cristalul de 15 MHz este elementul extern care stabilește frecvența de bază pentru generatorul 8284A. El nu face parte din circuitul intern propriu-zis, ci doar îi furnizează referința de frecvență necesară pentru producerea semnalelor de tact.



- Circuitul amplificator/ separator bidirecțional 74x245:

Este un circuit folosit pentru amplificarea/ separarea magistralelor bidirecționale ale microprocesoarelor

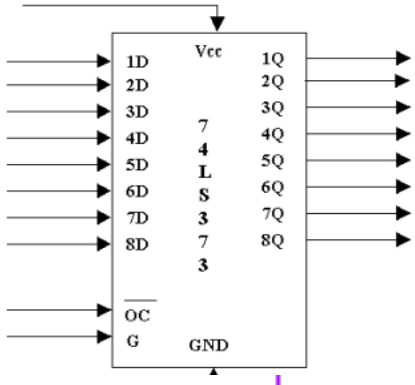
/G	DIR	A8 – A1	B8 – B1
0	0	B8 – B1	Intrări
0	1	Intrări	A8 – A1
1	X	A 3 – a stare	A 3 – a stare

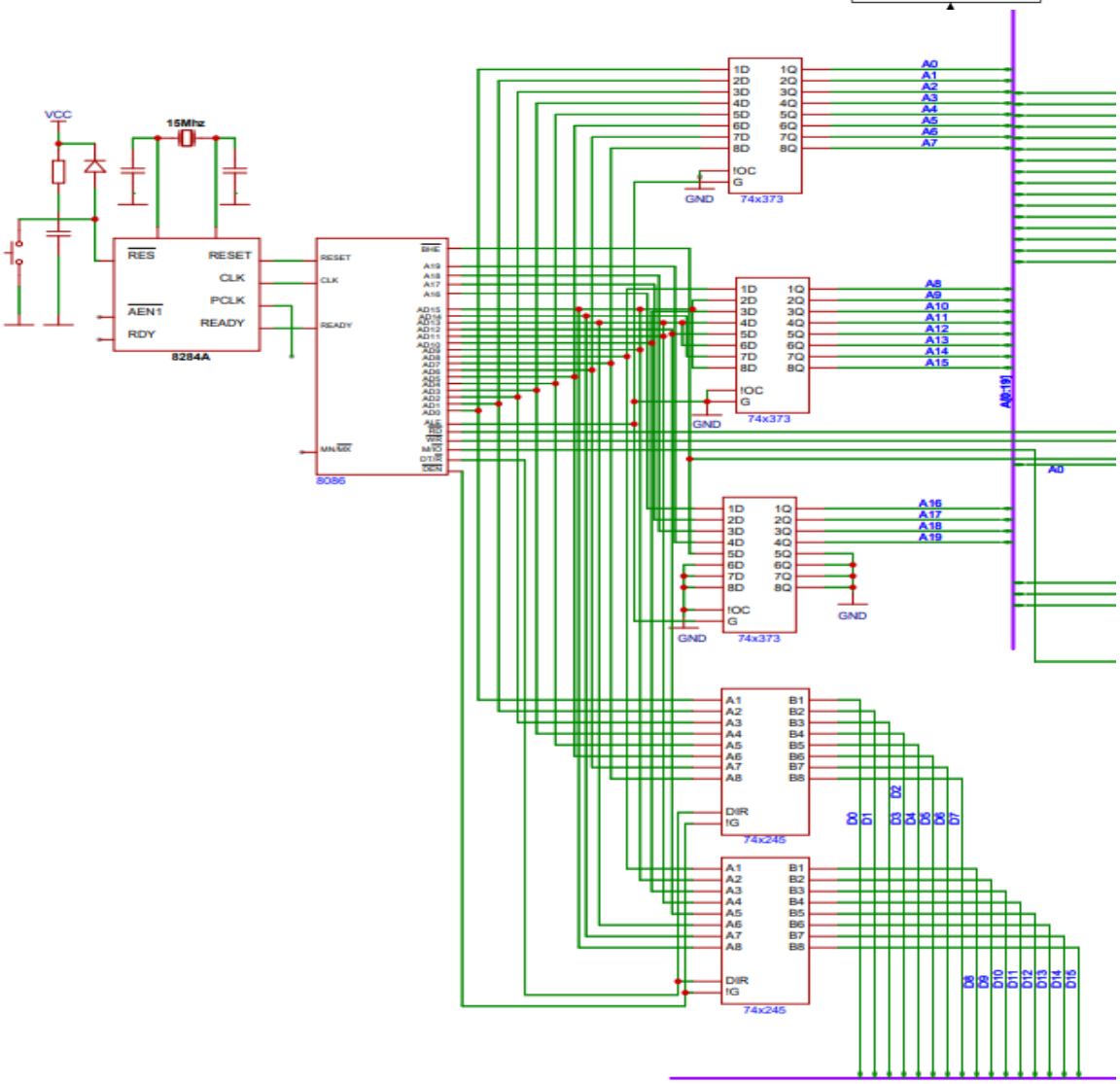


■ Circuitul registru 74x373:

Memoriile și porturile cer ca adresele să rămână stabile toată durata ciclului; demultiplexarea necesită registre. Este un registru cu 8 ranguri și 3 stări:

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





2. CONECTAREA MEMORIILOR

- **Memoria EPROM** (*Erasable Programmable Read-Only Memory*) este un tip de memorie nevolatilă care poate fi programată de utilizator, dar poate fi ștearsă doar prin expunere la lumină ultravioletă. După ștergere, poate fi reprogramată cu noi date.

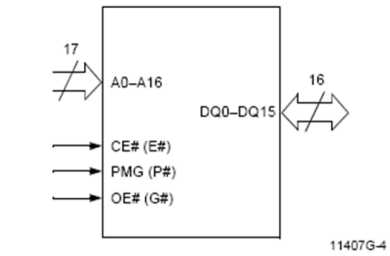
Circuitul 27C2048:

- ❑ 256 KB capacitate
- ❑ Timp de acces: 45 ns;

PIN DESIGNATIONS

A0–A16	=	Address Inputs
CE# (E#)	=	Chip Enable Input
DQ0–DQ15	=	Data Input/Outputs
OE# (G#)	=	Output Enable Input
PGM# (P#)	=	Program Enable Input
V_{CC}	=	V_{CC} Supply Voltage
V_{PP}	=	Program Voltage Input
V_{SS}	=	Ground

LOGIC SYMBOL



- **Memoria SRAM** *(Static Random Access Memory) este* o memorie rapidă și volatilă, care își păstrează datele doar cât timp este alimentată. Nu are nevoie de mecanisme suplimentare pentru menținerea informației, ceea ce îi oferă timpi de acces foarte mici. Este folosită în special pentru cache-uri.

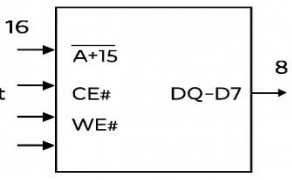
Circuitul 62512:

- ❑ 64 KB capacitate
- ❑ Timp de acces: 70 ns;

PIN DESIGNATIONS

A0–A15	=	Address Inputs
CE#	=	Chip Enable Input
WE#	=	Write Enable Input
OE#	=	Output Enable Input
VCC	=	Power Supply
VSS	=	Ground

LOGIC SYMBOL



Decodificarea memoriilor:

- **EPROM**: 128 KB memorie
  $2^7 * 2^{10} = 2^{17}$ B »» Adresa de început: **00000H**, Adresa de sfârșit: **1FFFFH**

Dat fiind că circuitul 27C2048 are o capacitate de **256KB**, pentru implementarea memoriei de 128KB **vom folosi un circuit de tipul 27C2048**, unde pinul A16 va fi legat la o valoare fixă (GND sau VCC). Astfel, vom folosi doar jumătate din circuit.

- **SRAM**: 64KB memorie

  $2^6 * 2^{10} = 2^{16}$ B »» Adresa de început **20000H**, Adresa de sfârșit: **2FFFFH**
  Dat fiind că circuitul 62512 are o capacitate de **64KB**, pentru implementarea memoriei de 64KB **vom folosi două circuite de tipul 62512**, întrucât acesta are o magistrală de date bidirecțională pe 8 biți, iar noi avem 16 linii de adrese. Pentru implementare, cel mai semnificativ pin (**A15**) va avea valoarea fixă '0' când reprezentăm cel mai nesemnificativ octet (**D0-D7**) și valoarea fixă '1' când reprezentăm cel mai semnificativ octet(**D8-D15**).

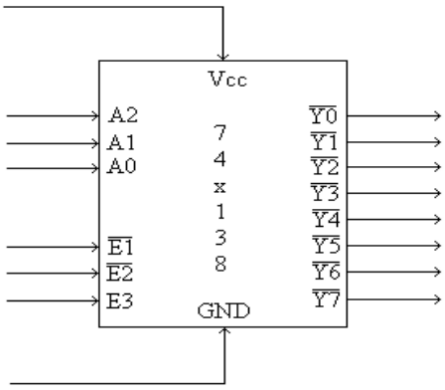
	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
EPROM Start	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EPROM End	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SRAM Start	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SRAM end	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

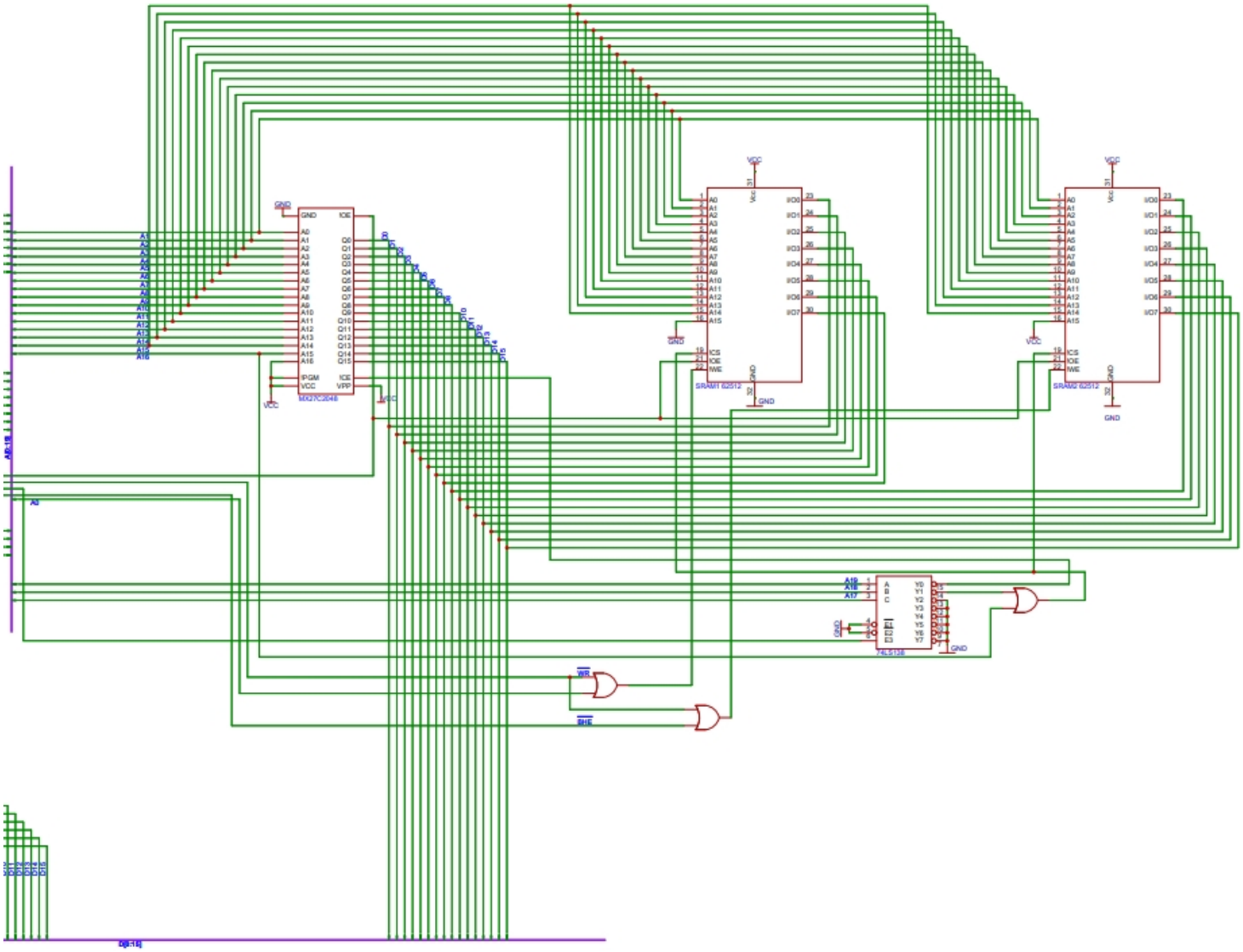
- **<u>Circuitul decodificator 74x138:</u>**

  Rolul unui decodificator de memorii este să genereze semnale de selecție pentru circuitele de memorie din microsistem. În urma decodificării liniilor de adresă a celor două zone de memorie, liniile de adresă A19, A18, A17 și A16 generează semnale de selecție:

SEL_{EPROM} = (!A19)&&(!A18)&&(!A17) »»!(Y0) – va merge în intrarea **!CE**
SEL_{SRAM} = (!A19)&&(!A18)&&(A17)&&(!A16) »»!(Y1)+A16 – va merge în cele două intrări **!CS**

E3	/E2	/E1	A2	A1	A0	/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

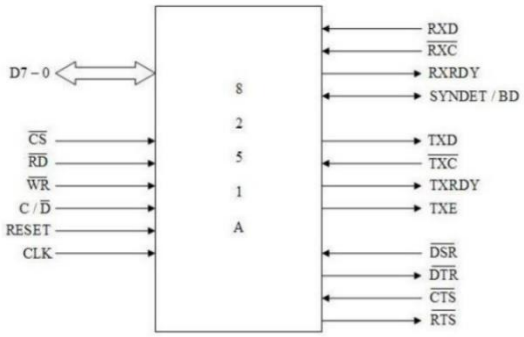




3. INTERFAȚA SERIALĂ ȘI PARALELĂ

- **INTERFAȚA SERIALĂ** reprezintă sistemul de circuite și programe fundamentale care contribuie la comunicarea bit cu bit între unitatea centrală și dispozitivele periferice. Prin intermediul acesteia se pot transmite și recepționa date folosind un număr redus de linii de comunicație, fiind utilizate semnale de control și stare pentru a asigura funcționarea corectă a transferului de date.

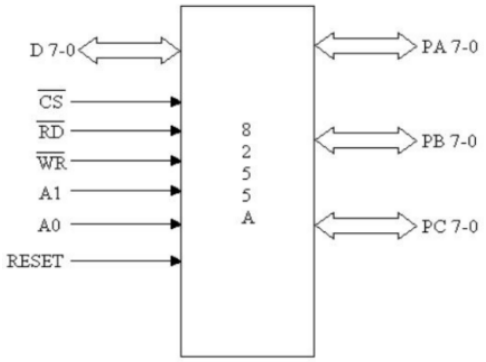
  Circuitul 8251: interfață serială programabilă care realizează comunicația asincronă sau sincronă între CPU și periferice, oferind semnale de stare precum **TxRDY** și **RxRDY** pentru controlul transmisiei și recepției.



/CS	/RD	/WR	C//D	Operaţie
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

- **INTERFAȚA PARALELĂ**. Spre deosebire de transferul serial, unde transferul datelor se face bit după bit, la transferul paralel se transferă 8 biți simultan, iar transferul este însoțit şi de semnale de dialog. Acest tip de transfer permite o viteză mai mare de comunicare, fiind utilizat în special pentru conectarea perifericelor apropiate de unitatea centrală, unde sincronizarea este mai ușor de realizat.

  Circuitul 8255: interfață paralelă programabilă care dispune de porturile A, B și C pentru transfer de date și de un registru de control (RCC) utilizat pentru configurarea modului de lucru și a direcției porturilor.



/CS	/RD	/WR	A1	A0	Operaţia
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 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

■ Conectarea porturilor

❑ În funcție de microcomutatorul S1:

S1=0 --» port de date: 0AF0H, port de comenzi: 0AF2H

S1=1 --» port de date: 0BF0H, port de comenzi: 0BF2H

❑ În funcție de microcomutatorul S2:

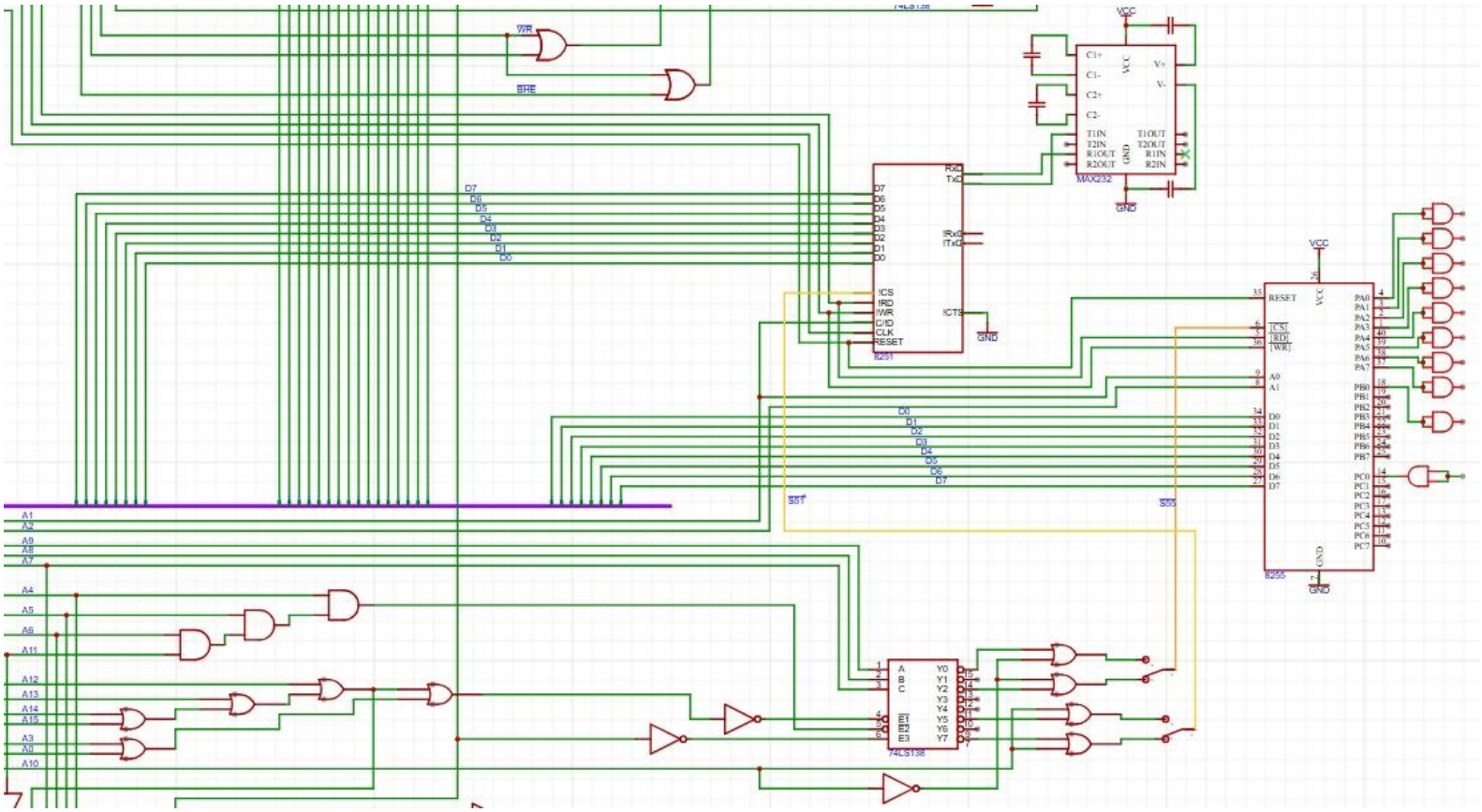
S2=0 --» port A: 0D70H, port B: 0D72H, port C: 0D74H, port RCC:0D76H

S2=1 --» port A: 0C70H, port B: 0C72H, port C: 0C74H, port RCC: 0C76H

		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
S1=0	Date	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0
	State	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0
S1=1	Date	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0
	State	0	0	0	0	1	0	1	1	1	1	1	1	0	0	1	0
S2=0	PortA	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0
	PortB	0	0	0	0	1	1	0	1	0	1	1	1	0	0	1	0
	PortC	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	0
	RCC	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1	0
S2=1	PortA	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0
	PortB	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	0
	PortC	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	0
	RCC	0	0	0	0	1	1	0	0	0	1	1	1	0	1	1	0

- Folosind decodificatorul **74x138,** ecuatiile de enable sunt:
  !E1=A15||A14||A13||A12||A3||A0
  !E2= ~(A11&&A6&&A5&&A4)
  E3 = ~M/IO

- În urma decodificării liniilor de adresă a celor patru zone de memorie, liniile de adresă A9, A8, A7 și A10 generează semnale de selecție:
  !SEL$_{S1=0}$ = !(A9) + A8 + !(A7) = !(Y5) + A10
  !SEL$_{S1=1}$ = !(A9) + !(A8) + !(A7) = !(Y7) + A10
  !SEL$_{S2=0}$ = A9 + !(A8) + A7 = !(Y2) + !(A10)
  !SEL$_{S2=1}$ = A9 + A8 + A7 = !(Y0) + !(A10)



❑ Subrutine necesare

- Rutina de programare a circuitului **8251** (alegem S1=0)

```
;; 8 biţi de date, fără paritate, factor de multiplicare x16, rata 9600bp

DATA_PORT   EQU 0AF0H
STATUS_PORT EQU 0AF2H
MOV DX, STATUS_PORT
MOV AL,0CEH ; cuvânt de mod
OUT DX,AL
MOV AL,15H ; cuvânt de comandă
OUT DX,AL
RET
```

- Rutina de transmisie caracter: (se transmite un caracter din CL)

```
TR: MOV DX, STATUS_PORT
    IN AL,DX ; citire şi testare rang TxRDY din cuvântul de stare
    RCR AL,1 ; rotește cu un bit right prin carry flag
    JNC TR ; dacă CF=0 atunci TxRDY =0
    MOV AL,CL ; se preia data din registrul CL
    MOV DX, DATA_PORT
    OUT DX,AL
    RET
```

- Rutina de recepție caracter: (se recepteaza un caracter în CL)

```
REC:  MOV DX, STATUS_PORT
      IN AL,DX ; citire şi testare rang RxRDY din cuvântul de stare
      RCR AL,2 ; rotește cu 2 biți right prin carry flag
      JNC REC ; dacă CF=0 atunci RxRDY =0,
      MOV DX, DATA_PORT
      IN AL,DX ; se preia data de la 8251
      MOV CL,AL ; se depune data în registrul CL
      RET
```

- Rutina de programare a circuitului **8255** (alegem S2=0)

```
A_PORT EQU 0D70H
B_PORT EQU 0D72H
C_PORT EQU 0D74H
RCC_PORT EQU 0D76H ; registru de control și de comandă
MOV DX, RCC_PORT; adresa cuvântului de comandă
```

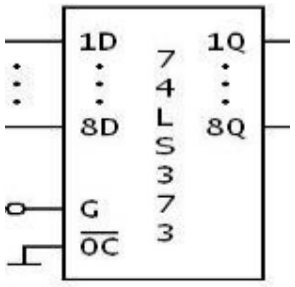
```
MOV AL,81H ; cuvânt de comandă (C=intrare; A,B=ieșire =>mod de lucru 0)
OUT DX,AL ; trimitem configurația
RET

PAR: MOV DX, C_PORT;
    IN AL,DX ; citire valoare de pe portul C (BUSY)
    RCR AL,1 ; bitul 0 ajunge in CF
    JNC PAR ; daca receptorul e 0 așteptăm
    MOV AL,CL ; se preia caracterul din registrul CL
    MOV DX, A_PORT
    OUT DX,AL ; transmisie date către port A
    MOV AL,01H ; STB e inactiv
    MOV DX, B_PORT ; B e port de control
    OUT DX,AL (/STB = 1)
    AND AL,00H
    OUT DX,AL ; /STB = 0
    OR AL,01H
    OUT DX,AL ; /STB = 1
    RET
```

4. CONECTAREA AFIȘAJELOR

A. 6 LED-uri

LED-ul (**Light Emitting Diode**) este o diodă semiconductoare care emite lumină la polarizarea directă a joncțiunii p-n. În configurația utilizată, LED-urile sunt active pe **nivel logic 0**, iar starea lor este menținută cu ajutorul unui registru **74LS373** conectat la magistrala de date.



74LS373 este un **latch** octal transparent (8 biți) folosit pentru memorarea temporară a datelor de pe magistrală. Atunci când semnalul **G** este activ, ieșirile urmăresc intrările; când G devine inactiv, valoarea este memorată și menținută la ieșire.

B. Modul de afișare cu 7 segmente și 6 ranguri

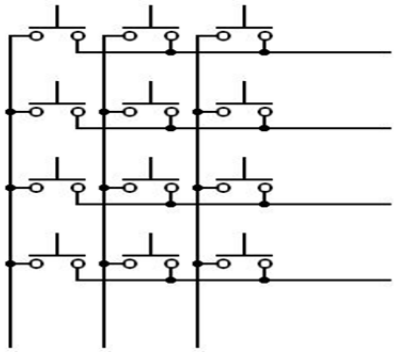
Afișajul cu 7 segmente este un dispozitiv utilizat pentru afișarea cifrelor numerice prin aprinderea combinată a celor șapte segmente luminoase, notate cu literele **a–g**. Fiecare segment este controlat individual, iar prin activarea diferitelor combinații de segmente se pot reprezenta cifrele de la 0 la 9. În funcție de tipul afișajului (anod comun), segmentele sunt activate pe nivel logic 0.

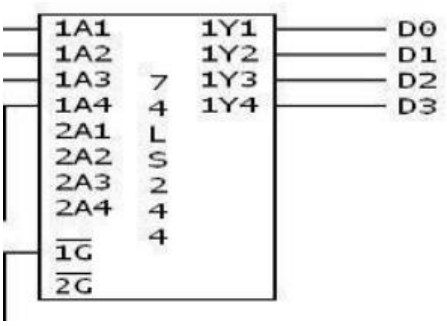
CIFRE	LITERELE FOLOSITE	CODURI p g f e d c b a
0	a,b,c,d,e,f	1 1 0 0 0 0 0 0 » C0H
1	b,c	1 1 1 1 1 0 0 1 » F9H
2	a,b,d,e,g	1 0 1 0 0 1 0 0 » A4H
3	a,b,c,d,g	1 0 1 1 0 0 0 0 » B0H
4	b,c,f,g	1 0 0 1 1 0 0 1 » 99H
5	a,c,d,f,g	1 0 0 1 0 0 1 0 » 92H
6	a,c,d,e,f,g	1 0 0 0 0 0 1 0 » 82H
7	a,b,c	1 1 1 1 1 0 0 0 » F8H
8	a,b,c,d,e,f,g	1 0 0 0 0 0 0 0 » 80H
9	a,b,c,d,f,g	1 0 0 1 0 0 0 0 » 90H

C. Minitastatură cu 12 contacte

Minitastatura este organizată sub forma unei matrici de linii și coloane, la intersecția acestora fiind plasate tastele. Pentru comandă se utilizează un port de ieșire, iar starea tastelor este citită prin intermediul unui port de intrare. Scanarea tastaturii se face prin activarea succesivă a câte unei coloane, aplicând nivel logic 0 pe aceasta și nivel logic 1 pe celelalte, după care se verifică starea liniilor. Detectarea unui nivel logic 0 pe o linie indică apăsarea tastei corespunzătoare.

În implementarea utilizată, tastatura are o structură matricială 3×4.





Pentru interfațare se utilizează un circuit de tip 74LS373, care asigură memorarea valorilor de ieșire, și un circuit 74LS244, folosit ca buffer tri-state pentru citirea semnalelor de intrare.

■ Tabel cu porturi

Port de intrare **LED: SL = 0910H**

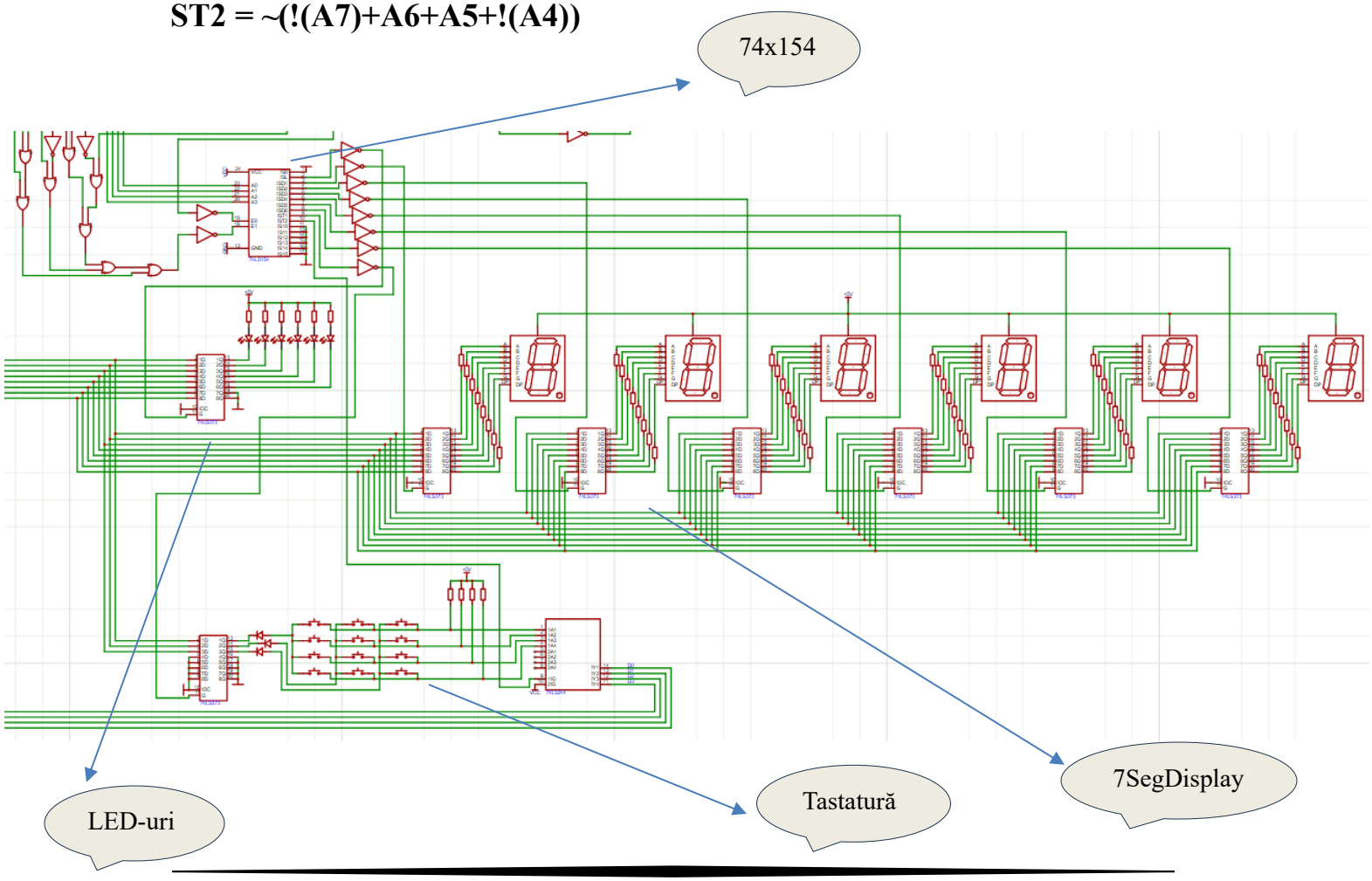
Porturi de intrare **7SegDisplay: SD1=0920H, SD2=0930H, SD3=0940H, SD4=0950H, SD5=0960H, SD6=0970H**

Port de ieșire/intrare **Minitastatură: ST1=0980H (ieșire), ST2=0990H (intrare)**

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
SL	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0910H
SD1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0920H
SD2	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0930H
SD3	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0940H
SD4	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0950H
SD5	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0960H
SD6	0	0	0	0	1	0	0	1	0	1	1	1	0	0	0	0	0970H
ST1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0980H
ST2	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0990H

- Folosind decodificatorul **74x154** (DEC 4X16), ecuațiile de enable sunt:
  E0= ~(M/~IO)
  E1= ~(A15+A14+A13+A12+!(A11)+A10+A9+!(A8)+A3+A2+A1+A0)

- În urma decodificării liniilor de adresă a celor nouă zone de memorie, liniile de adresă A7, A6, A5 și A4 generează semnale de selecție:
  !SL =A7+A6+A5+!(A4)
  !SD1=A7+A6+!(A5)+A4
  !SD2=A7+A6+!(A5)+!(A4)
  !SD3=A7+!(A6)+A5+A4
  !SD4=A7+!(A6)+A5+!(A4)
  !SD5=A7+!(A6)+!(A5)+A4
  !SD6=A7+!(A6)+!(A5)+!(A4)
  !ST1=!(A7)+A6+A5+A4
  ST2 = ~(!(A7)+A6+A5+!(A4))



❑ Subrutine necesare:

▪ Rutina de aprindere a unui LED

;; anod comun » led-ul va fi aprins cand '0' este active

```
LED_PORT EQU 0910H
MOV DX, LED_PORT
MOV AL, XX ;  xx = masca de biți care stabilește care led se aprinde
OUT DX,AL
```

LED-uri	Configurație	Masca XX
LED 1	1111 1110	FEH
LED 2	1111 1101	FDH
LED 3	1111 1011	FBH
LED 4	1111 0111	F7H
LED 5	1110 1111	EFH
LED 6	1101 1111	DFH

▪ Rutina de stingere a unui LED

```
MOV AL, FFH;
OUT DX, AL
```

▪ Rutina de afișare a unui caracter hexa pe un anumit rang

Adresele: 0920H – 0970H

```
.data
;; coduri pentru cifre
CODES db C0H, F9H, A4H, B0H, 9BH, 99H, 82H, F8H, 80H, 90H
;; coduri pentru rang-uri
PORTS dw 0920H, 0930H, 0940H, 0950H, 0960H, 0970H

.code
MOV SI, OFFSET CODES ; Adresa tabelului de coduri pentru cifre
MOV DI, OFFSET PORTS ; Adresa tabelului de porturi
MOV CX, 6 ; Număr de rânduri de afișat (6 ranguri)

7seg_loop:      MOV DX, [DI] ; Ia portul curent din tabelul PORTS
                MOV AL, [SI] ; Ia codul cifrei curente din tabelul CODES
                OUT DX, AL ; Trimite codul cifrei la portul respectiv
```

```
ADD SI, 1 ; Treci la următorul cod de cifră
ADD DI, 2 ; Treci la următorul port
LOOP 7seg_loop; Repetă pentru toate cele 6 rang-uri
```

- **Rutina de programare a minitastaturii**

TASTATURA:

```
IN_PORT  EQU 0980H   ;  ST2 – citire linii tastatură
OUT_PORT EQU 0990H   ; /ST1 – comandă coloane tastatură
```

*; se pune 0 pe a prima coloană să vedem dacă s-au acționat tastele 1,4,7,**

```
MOV AL, FEH ;se pune 0 pe prima coloană (1111 1110 B)
OUT OUT_PORT, AL ;selectez /ST1 (ieșirea tastaturii)

IN AL, IN_PORT ;încarc conținutul aflat la adresa ST2
AND AL, 01H ;verific prima poziție
JZ TASTA1 ;salt la tasta 1 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 02H ;verific a doua poziție
JZ TASTA4 ;salt la tasta 4 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 04H ;verific a treia poziție
JZ TASTA7 ;salt la tasta 7 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 08H ;verific a patra poziție
JZ TASTA* ;salt la tasta * dacă rezultatul este 0
```

; se pune 0 pe a doua coloană să vedem dacă s-au acționat tastele 2,5,8,0

```
MOV AL, FDH ; se pune 0 pe a doua coloană (1111 1101 B)
OUT OUT_PORT, AL

IN AL, IN_PORT
AND AL, 01H ;verific prima poziție
JZ TASTA2 ;salt la tasta 2 dacă rezultatul este 0
```

```
IN AL, IN_PORT
AND AL, 02H ;verific a doua poziție
JZ TASTA5 ;salt la tasta 5 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 04H ;verific a treia poziție
JZ TASTA8 ;salt la tasta 8 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 08H ;verific a patra poziție
JZ TASTA0 ;salt la tasta 0 dacă rezultatul este 0
```

; se pune 0 pe a treia coloana sa vedem daca s-au actionat tastele 3,6,9,#

```
MOV AL, FBH ; se pune 0 pe a treia coloană (1111 1011 B)
OUT OUT_PORT, AL

IN AL, IN_PORT
AND AL, 01H ;verific prima poziție
JZ TASTA3 ;salt la tasta 3 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 02H ;verific a doua poziție
JZ TASTA6 ;salt la tasta 6 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 04H ;verific a treia poziție
JZ TASTA9 ;salt la tasta 9 dacă rezultatul este 0

IN AL, IN_PORT
AND AL, 08H ;verific a patra poziție
JZ TASTA# ;salt la tasta # dacă rezultatul este 0

; se reia baleierea
JMP TASTATURĂ
.....
```

- **Tratarea acționării tastelor**

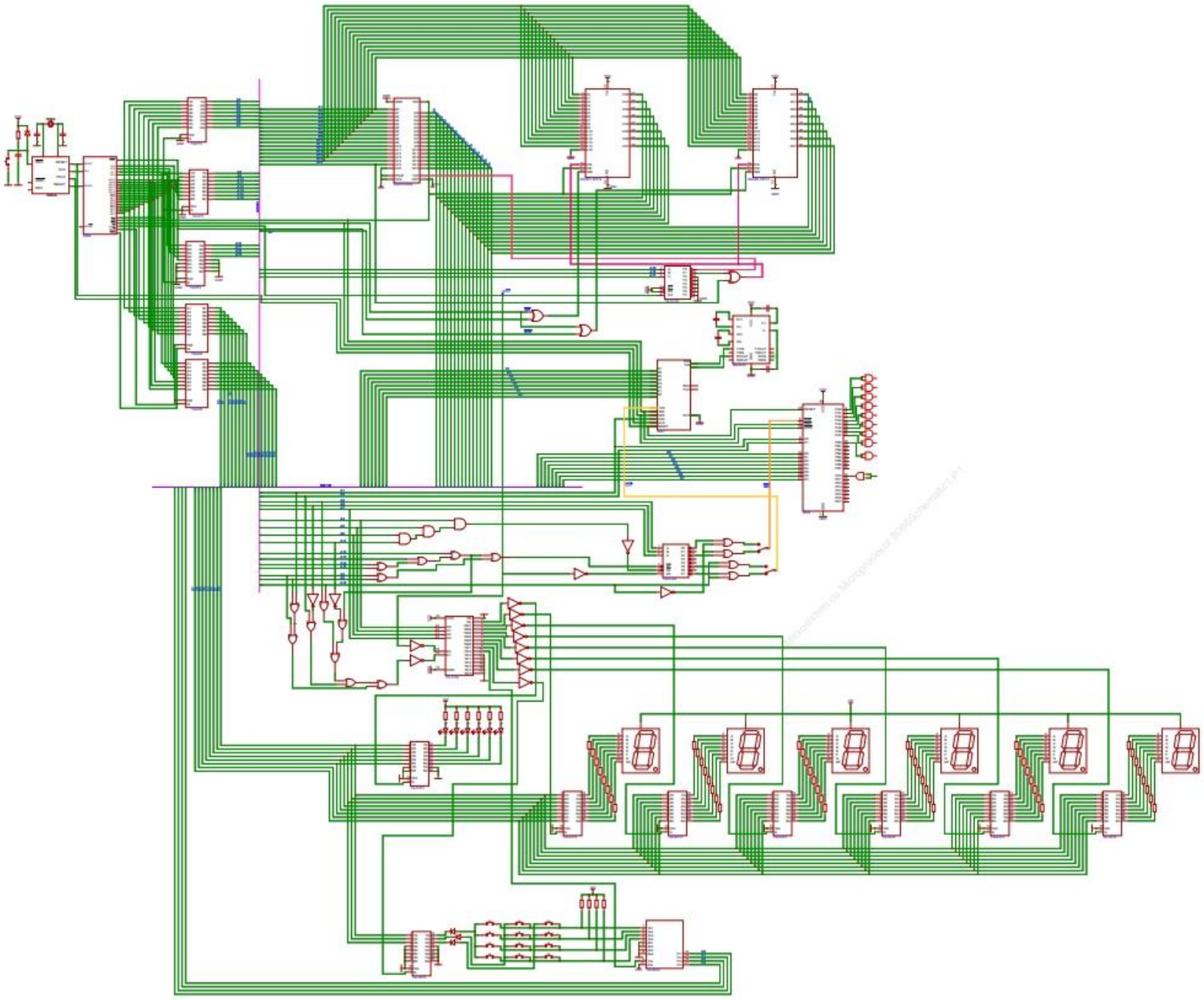
TASTA1:

```
    CALL DELAY ; se aşteaptă stabilirea contactelor
    DISABLE_1: IN AL, IN_PORT ; se citeşte din nou linia şi se aşteaptă
dezactivarea tastei
                AND AL,01H
                JZ DISABLE_1
                CALL DELAY
                MOV CL, 00H
                RET
…….....
```

TASTA#:

```
    CALL DELAY
    DISABLE_#:  INT AL, IN_PORT
                AND AL, 08H
                JZ DISABLE_9
                CALL DELAY
                MOV CL, 00H
                RET
```

Analog pentru restul tastelor



BIBLIOGRAFIE:

[1]Datasheet-ul circuitelor folosite din EasyEda
[2]Materiale de curs și laborator
[3]EasyEDA Quick Tips | EasyEDA Pro – Youtube