Facultatea de Automatică și Calculatoare

Universitatea Politehnica Timișoara

Implementarea unui Microsistem cu Microprocesorul 8086

Universitatea Politehnica Timișoara

Facultatea de Automatică și Calculatoare

Realizat de: **Mingiuc Bianca Anamaria**

An universitar: **2025-2026**

1. TEMA PROIECTULUI

Să se proiecteze un microsistem cu următoarea structură hardware și software:

- **Unitate centrală:** Microprocesor Intel 8086.
- **Memorie Program (EPROM):** Capacitate 128 KB, utilizând circuite 27C512.
- **Memorie Date (SRAM):** Capacitate 64 KB, utilizând circuite 62256.

Interfață Serială: Circuit 8251, mapat în zona 04D0H – 04D2H sau 05D0H – 05D2H (selectabil prin switch S1).

- **Interfață Paralelă:** Circuit 8255, mapat în zona 0250H – 0256H sau 0A50H – 0A56H (selectabil prin switch S2).
- **Periferice:**
    - Minitastatură matriceală (9 contacte).
    - 10 LED-uri pentru semnalizare.
    - Modul de afișare cu 7 segmente (8 ranguri).
    - Modul LCD alfanumeric (2 linii x 16 caractere).

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.

Structura rutinelor (intrări, secvenţe, ieşiri) va fi stabilită de fiecare student.

2. DESCRIEREA HARDWARE

2.1. Unitatea Centrală (CPU)

Componente Principale

1. Microprocesorul 8086

Primul microprocesor pe 16 biți care a cunoscut o largă utilizare;

Apariția lui a fost urmată la scurt timp de o familie de componente: generatorul de tact 8284, controlerul de magistrală 8288, coprocesorul matematic 8087 și coprocesorul de intrare / ieșire 8089.

Caracteristici:

- registrele interne și magistrala de date externă sunt pe 16 biți;
- posibilitatea de a adresa direct 1 Mo 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.

Semnale:

- RESET = intrare pentru inițializarea microprocesorului
- CLK = intrare de tact cu frecvența uzuală 5 MHz și factor de umplere de 1/ 3

- READY = intrare pentru sincronizarea cu circuitele de memorie și porturile mai lente
- MN/ !MX = intrare ce indică modul de lucru al procesorului: 1 logic (mod minim), 0 logic (mod maxim)
- DT/ !R (Data trasmit/Receive) = ieșire cu trei stări, care indică sensul transferului pe magistrala de date: 1 indică transmisie date, 0 recepție
- !DEN (Data Enable) = ieșire cu trei stări, care validează transferul de date pe magistrală
- !BHE/ S7 (Bus High Enable) = ieșire care indică dacă are sau nu loc un transfer pe jumătatea superioară a magistralei de date
- A19-A16 = rangurile 19-16 din magistrala de adrese
- AD15-AD0 = magistrala multiplexată de adrese/date cu 3 stări
- ALE (Address Latch Enable) = ieșire care se activează atunci când pe magistrala multiplexată de adrese/date sunt active adresele; se poate folosi pentru demultiplexarea magistralei prin încărcarea adreselor în register
- !RD (Read Control) = ieșire cu trei stări, activă atunci când microprocesorul execută un ciclu de citire sau de intrare
- !WR (Write Control) = ieșire cu trei stări, activă atunci când microprocesorul execută un ciclu de scriere sau de ieșire
- M/ !IO (Memory / Input-Output Control) = dacă are valoarea 1 înseamnă că se execută un ciclu de acces la memorie, iar dacă are valoarea 0 înseamnă că se execută un ciclu de transfer cu porturile de intrare/ieșire.

2. Registrul de date 74x373

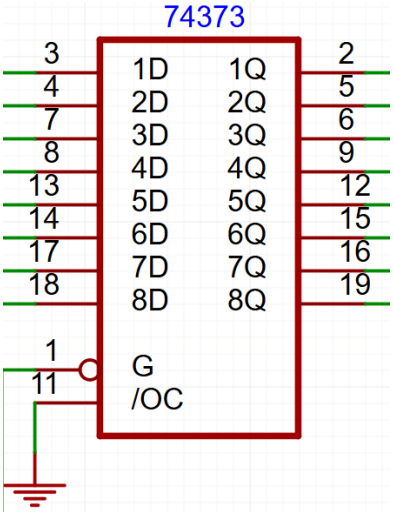
Microprocesorul 8086 utilizează o magistrală multiplexată (AD15-AD0), ceea ce înseamnă că aceiași pini sunt folosiți alternativ pentru a transmite atât adrese, cât și date.

Registrul 74x373 (numit și latch de adresă) este utilizat pentru a:

Separa adresele de date.

Menține adresele pe o magistrală de adrese separată (A19-A0) pe întreaga durată a unui ciclu de magistrală (timp în care magistrala multiplexată trece la transferul de date).

În schema Unității Centrale, sunt folosite trei circuite 74x373 pentru a capta cele 20 de linii de adresă (A19-A0) și semnalul ~BHE



Funcționarea circuitului 74x373 este controlată de două semnale principale: **G** (Gate/Enable) și **~OC** (Output Control).

Controlul prin **ALE** (Address Latch Enable)

Intrarea de control **G** a registrului 74x373 este conectată la ieșirea ALE (Address Latch Enable) a microprocesorului 8086.

- Când **ALE** = 1: Circuitul 8086 indică faptul că pe magistrala AD15-AD0 sunt active adresele. În această stare, registrul 74x373 devine transparent (funcționează ca o poartă), iar datele de la intrările 1D-8D (conectate la magistrala multiplexată) sunt transferate la ieșirile 1Q-8Q.
- Când **ALE** = 0: Adresa este blocată/capturată (latched) în registru. Ieșirile 1Q-8Q mențin conținutul anterior (adresa capturată), indiferent de schimbările de pe magistrala de date/adrese.

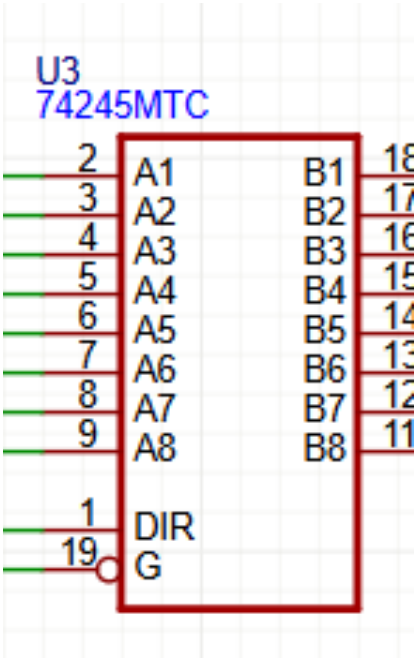
• Funcționarea:

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

Controlul Ieșirilor prin **~OC**

Intrarea **~OC** determină dacă ieșirile sunt active sau în starea de înaltă impedanță:

3. Circuitul amplificator/ separator bidirecţional 74x245:



● Funcţionarea:

/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

Este un circuit care permite transferul bidirecțional de date intre magistrale. In proiect sunt folosite doua, fiecare cu 8 ranguri, pentru a gestiona transferul de date a 8086 si restul componentelor. Pinul DIR (Direction) determina direcția fluxului de date intre cele doua magistrale si in proiect îl controlam cu semnalul DT/~R. Când citim fluxul de date este direcționat de la periferice/memorie spre microprocesor si la scriere este inversat. Pinul G# este la fel ca la 74x373 doar ca in cazul acesta este legat la pinul ~DEN descris mai sus.

Primul circuit 747x245 (U14) are intrările cu rangurile AD0 la AD7 din magistrala de date/adrese, si cu ieșirile D0-D7 din magistrala de adrese. Circuitul al doilea cuprinde rangurile de intrare AD8-AD15 si ieșirile la magistrala D8-D15.

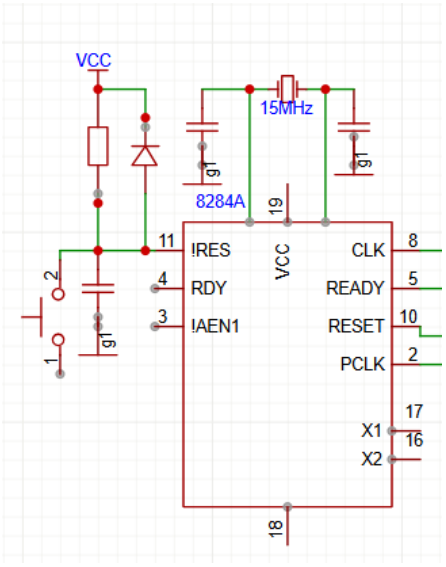
4. Generatorul de tact

Generatorul de tact 8284A este un circuit integrat esențial în arhitectura bazată pe microprocesorul 8086, deoarece 8086 nu are un circuit de ceas intern. Rolul său principal nu este doar de a genera semnalul de ceas, ci și de a sincroniza semnalele de control critice pentru buna funcționare a sistemului.

Cristalul de cuarț de **15 MHz** este stabilizat prin două condensatoare, iar frecvența internă a procesorului va fi o treime din aceasta (5 MHz).

Ieșiri:

- **CLK:** Semnalul de tact principal pentru 8086.
- **PCLK:** Un semnal de frecvență redusă (jumătate din CLK) pentru periferice.
- **RESET:** Circuitul integrează logica de resetare (trigger Schmitt), fiind conectat la un circuit RC extern și un buton de reset, asigurând inițializarea corectă a sistemului la pornire (Power-On Reset).
- **READY**

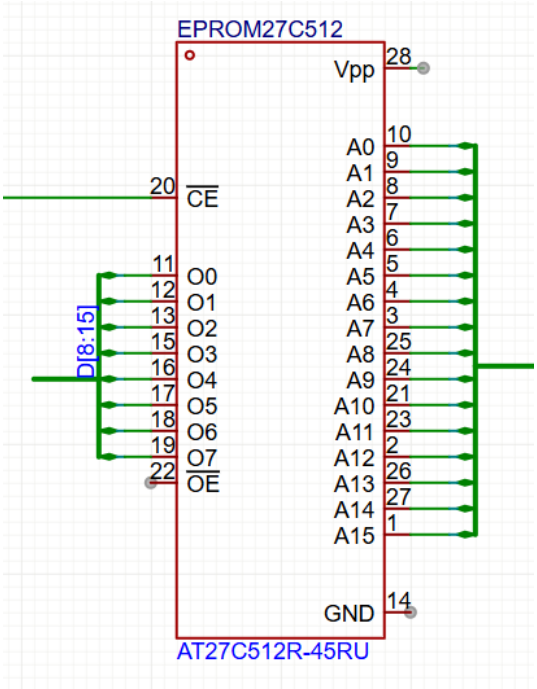


2.2. Subsistemul de Memorie

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

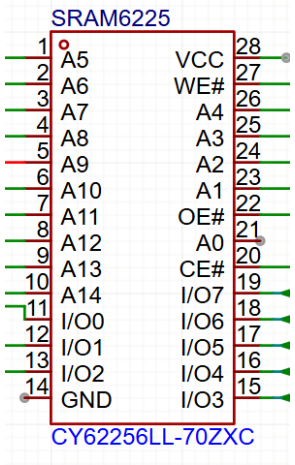
Memoria de tip EPROM este o memorie de tip ROM (Read Only Memory), care poate fi programată pe cale electrică. Este un tip de memorie nevolatilă, astfel, poate păstra informațiile stocate pentru perioade lungi de timp; în plus, permite citirea acestuia într-un mod nelimitat.

EPROM-urile sunt prin construcție compuse dintr-o serie de tablouri de tranzistoare (porți logice) ce sunt programate individual de pe un dispozitiv care furnizează tensiuni mai înalte. O memorie tip EPROM poate fi ștearsă după programare prin expunerea la o sursă puternică de lumină ultravioletă.



b. **Memoria SRAM**

Memoria SRAM (Static Random Access Memory) este destinată stocării temporare a datelor necesare procesării, oferind timpi de acces reduși pentru operațiile de citire și scriere. Aceasta este o memorie volatilă, datele fiind menținute doar pe durata prezenței tensiunii de alimentare. La nivel intern, stocarea se realizează prin circuite basculante bistabile (flip-flops), arhitectură care asigură stabilitatea informației fără a necesita cicluri periodice de reîmprospătare (refresh), spre deosebire de memoriile dinamice.



Proiectul utilizează doua tipuri de memorii:

- 128 KB de memorie EPROM, implementata cu doua circuite 27C512 (U9 si U10 , 64 KB flecare)
- 64 KB de memorie SRAM, implementata cu doua circuite 62256 (U7 si U11, 32 KB fiecare)

Decodificarea memoriilor

Adresa de intrare SRAM am ales 00000h

Avem 64Kb memorie SRAM -> Adresa de iesire SRAM=adr intrare+2^16-1 (64Kb = 2^6*2^10)

Adresa de intrare EPROM=adr iesire EPROM+1

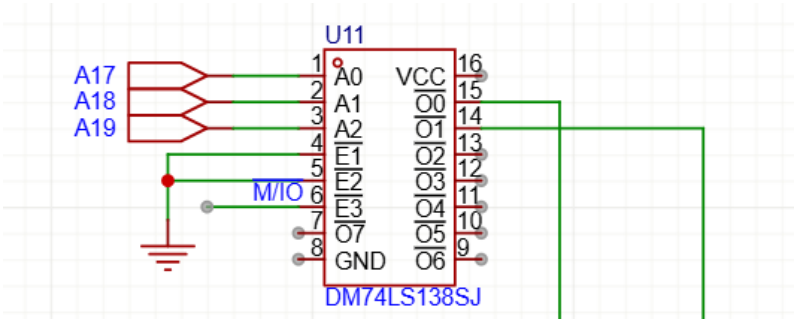
Avem 128Kb memorie EPROM -> Adresa de iesire EPROM=adresa de intrare+2^17-1 (128Kb = 2^7*2^10)

A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00	ZONA	Memoria
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000h	SRAM
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0FFFFh	SRAM
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20000h	EPROM
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3ffffh	EPROM

Rangurile relevante pentru SRAM sunt A19, A18, A17,A16, iar pentru EPROM sunt A19, A18 si A17. Deci funcțiile combinaționale ar fi:

Sel(SRAM)=~A19 & ~A18 & ~A17 & ~A16-> iesirea din decodificator pt SRAM este O0

Sel(EPROM)=~A19 & ~A18 & A17 -> iesirea din decodificator pt EPROM este O1



2.3. Interfețele serial și paralelă

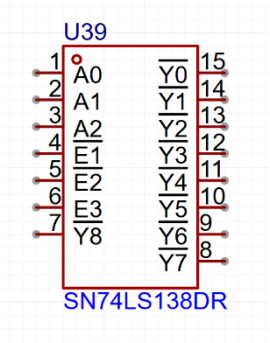
1.Decodificarea de porturi

Se folosește decodificarea a porturilor pe interfețe pentru a putea selecta una dintre ele si in intervalul de adrese dat. Aceasta selecție se face pe baza magistralei de adrese știind intervalul de adrese dedicat pentru fiecare interfața si creând funcții logice care activează interfața corecta când microprocesorul adresează unul dintre aceste intervale. Pentru acea se folosește un decodificator 3 la 8 si circuite logice.

Observăm că avem nevoie doar de cei 3 biti (A11, A9 și A8) pentru a vedea ce interfețe și în ce stări vom folosi. Astfel, cei 3 biți vor fi intrarile pentru decodificatoarele noaste(A0,A1 și A2).

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Acces	Adresa	Interfete
0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	Data	04D0H	Seriala0
0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0	Comanda	04D2H	S1=0
0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	Date	05D0H	Seriala1
0	0	0	0	0	1	0	1	1	1	0	1	0	0	1	0	Comanda	05D2H	S1=1
0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	Port A	0250H	
0	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0	Port B	0252H	Paralela0
0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	0	Port C	0254H	S2=0
0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	Control	0256H	
0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	Port A	0A50H	
0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	Port B	0A52H	Paralela1
0	0	0	0	1	0	1	0	0	1	0	1	0	1	0	0	Port C	0A54H	S2=1
0	0	0	0	1	0	1	0	0	1	0	1	0	1	1	0	Control	0A56H	

- Folosim decodificatorul 3 la 8 pentru a selecta zona de memorie corespunzătoare a interfețelor în cele 2 stări ale microcomutatoarelor S1 și S2.



- Sel(seriala, s1=0) = ~A11 & ~A9 ~A0 -> iesirea Y0
- Sel(seriala, s1=1) = ~A11 & ~A9 & A8 -> iesirea Y1
- Sel(paralela, s2=0) = ~A11 & A9 & ~A8 -> iesirea Y2
- Sel(paralela, s2=1) = A11 & A9 & ~A8 -> iesirea Y6
- Pentru a activa decodificatorul, avem nevoie de condițiile de Enable. Acestea depind de biți din magistrala de adrese care sunt fix pentru un anumit set de adrese. Pentru E1# îl legam directa la semnalul M/IO# de la 8086 care va fi activ pe 0 logic când lucram cu interfețele, iar când lucram cu memoria, va fi pe 1 logic.

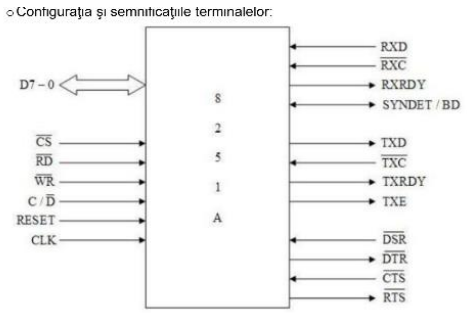
- Celelalte 2 condiții de enable sunt deduse din datbel și vor fi următoarele:

~E2 = A15 | A14 | A13 | A12 | ~A6 | A5 | ~A4 | A3 | A0

~E3 = (A10 & ~A9 &~A2) & (~A10 & A9)

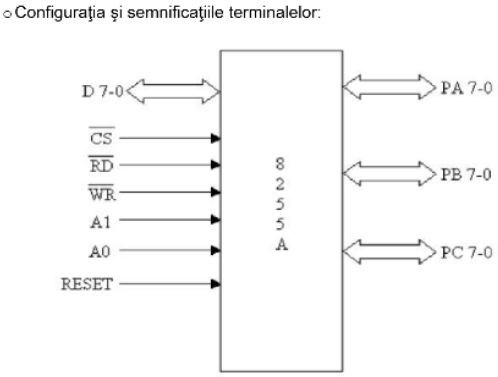
2.Circuitul 8251 (interfața serială)

Interfața serială este reprezentată de circuitele și programele de bază care asigură comunicare între unitatea centrală și echipamentele periferice, comunicare bit după bit. Datorită costului mai redus și a rezistenței la perturbații, transferul de tip serial este util atunci când informația trebuie transmisă pe distanțe mari

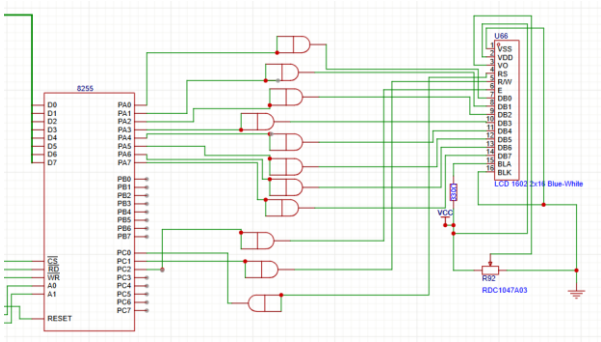


3.Circuitul 8255 (interfața paralelă)

Spre deosebire de transferul serial, unde transmisia se realizează secvențial (bit cu bit), transferul paralel permite transmiterea simultană a unui grup de biți (de exemplu, 8 biți/octet). Pe lângă liniile de date, acest tip de comunicație utilizează linii suplimentare pentru semnale de control și sincronizare (handshaking). Aceste semnale au rolul de a valida stabilitatea datelor pe magistrală sau de a confirma disponibilitatea receptorului de a prelua informația.



1. Interfața Modulului de Afișare Alfanumerică (LCD 1602)



În vederea vizualizării mesajelor textuale și a parametrilor de sistem, proiectul include un afișaj standard cu cristale lichide (LCD), modelul 1602. Acesta dispune de o capacitate totală de 32 de caractere, fiind structurat pe două linii a câte 16 caractere.

RS (Register Select): Realizează selecția între registrele interne ale afișajului:

- Nivel logic "0": Selectează Registrul de Instrucțiuni (destinat comenzilor de configurare, ștergere a ecranului sau setare a cursorului).
- Nivel logic "1": Selectează Registrul de Date (destinat transmiterii codurilor ASCII aferente caracterelor de afișat).

R/W (Read/Write): Determină sensul fluxului de date pe magistrală:

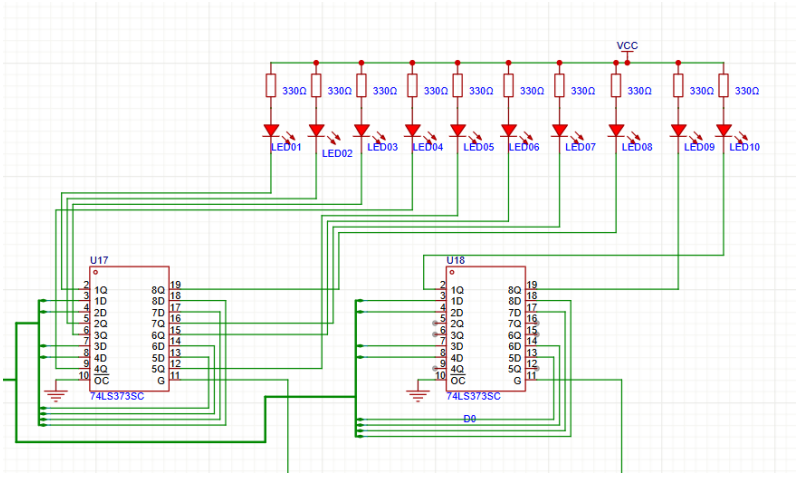
- Nivel logic "0" (Write): Activează scrierea datelor dinspre microprocesor în memoria LCD-ului.
- Nivel logic "1" (Read): Activează citirea datelor dinspre LCD către procesor (utilizată în special pentru verificarea bitului de stare Busy Flag).

E (Enable): Constituie semnalul de validare a operațiunii. Controllerul LCD preia starea liniilor de date și execută comanda la primirea unui impuls pe acest pin (validarea informației de pe magistrală are loc, de regulă, pe frontul descrescător al semnalului).

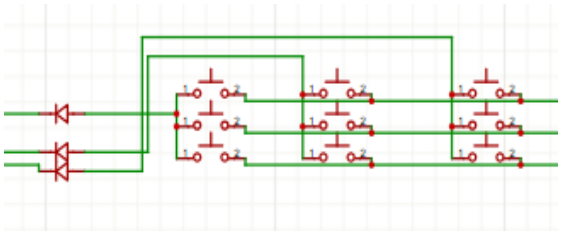
2.3. Perifericele

2. Conectarea LED-urilor

LED (light-emitting diode) este o diodă semiconductoare ce emite lumină la polarizarea directă a joncțiunii p-n. Avem nevoie, pentru acest microsistem de 10 LED-uri, deci vom avea nevoie de un registru 74x373 conectate la magistrală, pentru a le menține aprinse/stinse.



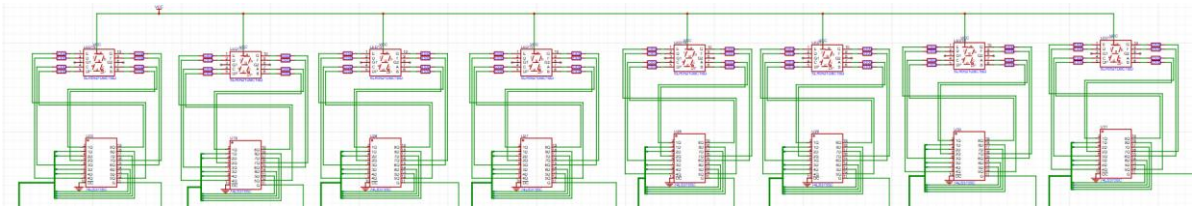
3. Conectarea minitastaturii:



Se plasează o minitastatura cu 9 contacte organizate ca o matrice prin trei linii si trei coloane. Fiecare tasta este un switch (comutator cu revenire) care are doi pini, primul pin este conectat cu toate tastele din coloana respectiva.

4. Conectarea afișajelor cu segmente

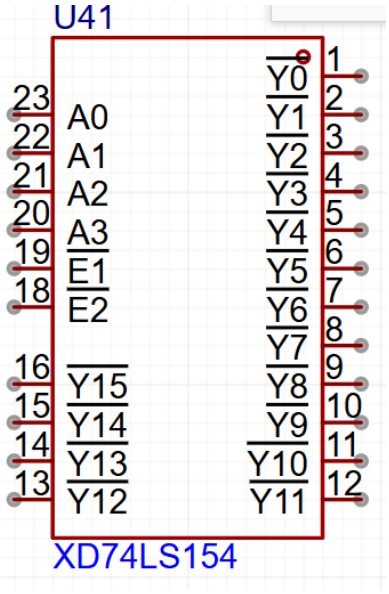
Sistemul include opt module de afișare de tip 7 segmente. Fiecare modul corespunde unui singur rang numeric și este compus din opt LED-uri: șapte pentru segmentele afișajului (a, b, c, d, e, f, g) și unul pentru punctul zecimal (DP). Toate cele opt segmente sunt accesibile individual, fiind conectate la un circuit registru. Acest aranjament asigură că fiecare afișaj poate fi controlat independent, permițând microprocesorului să scrie secvențial datele necesare afișării



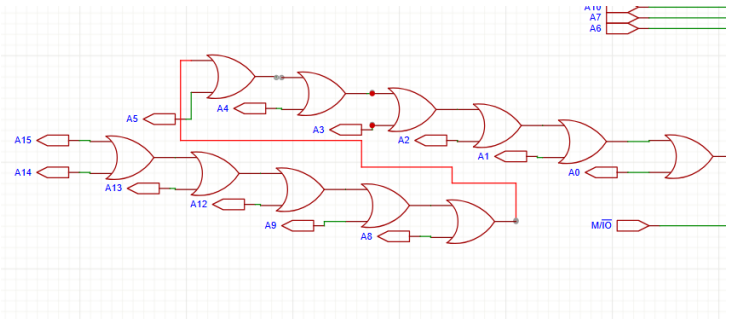
Decodificarea porturilor

Adresa	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Denumirea
0040h	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	SA1
0080h	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	SA5
00C0h	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	SA2
0440h	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	SA3
0480h	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	SA6
04C0h	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	SA4
0840h	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	ST1
0880h	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	SA7
08C0h	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	~ST2
0C40h	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	SL1
0C80h	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	SA8
0CC0h	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	SL2

- Folosim decodificatorul 4 la 16 pentru a gasi semnalele SA1, SA2, SA3, SA4, SA5, SA6, SA7, SA8, SL1, SL2, ST1, ST2
- Observăm că singurii biți care se schimbă sunt A11, A10, A7 și A6
- Astfel, aceștia vor fi selectoarele pentru decodificatorul nostru (intrările A0, A1, A2 și A3)
- La fel ca la decodificatorul de porturi pentru interfețe, avem nevoie de semnale enable pentru activarea sau dezactivarea decodificatorului. Pentru ~E1 se folosește M/~IO, la fel ca la decodificatorul pentru interfețe.
- ~E2 = A15 | A14 | A13 | A12 | A9 | A8 | A5 | A4 | A3 | A2 | A1 | A0 conform tabelului.



- Aici nu avem nevoie de ~E3



3. Rutinele de programare

Rutina de programare a circuitului 8251

Rutina de programare:

```
MOV DX, 04D2H   ; Port Comenzi/Stări (sau 05D2H)
MOV AL, 0CEH    ; Mod: 8 biți, fără paritate, factor x16
OUT DX, AL      ; Scriere cuvânt de mod
MOV AL, 15H     ; Comandă: Activare Tx/Rx, Reset erori
OUT DX, AL      ; Activare circuit
RET
```

Rutina de programare a circuitului 8255

Rutina de programare:

```
MOV DX, 0256H ; sau 0A56H. Adresa registrului de comandă
MOV AL, 81H ;
OUT DX, AL ; inițializare interfață
RET
```

Rutine de emisie/recepție caracter pe interfața serială

Transmisie la interfața serială:

```
TR:
MOV DX, 04D2H   ; Selectează adresa registrului de Stare/Comandă (05D2H dacă S1=1)
IN  AL, DX      ; Citește Cuvântul de Stare al circuitului 8251 în registrul AL
RCR AL, 1       ; Rotește bitul 0 (TxRDY) în flag-ul Carry (CF)
JNC TR          ; Sări la început (TR) dacă bufferul e Ocupat
                ; Dacă ajunge aici, bufferul este Liber (TxRDY=1)
MOV AL, CL      ; Se încarcă data (caracterul) care trebuie transmis din CL în AL
MOV DX, 04D0H   ; Selectează adresa registrului de Date (05D0H dacă S1=1)
OUT DX, AL      ; Scrie data în registrul de Date al 8251 pentru a iniția transmisia
RET             ; Revenire din subrutină
```

Recepție:

```
MOV DX, 04D2H   ; Adresa port Stare (Status)
IN  AL, DX      ; Citește status 8251
RCR AL, 2       ; Mută bitul RxRDY (bit 1) în Carry
JNC REC         ; Așteaptă până când RxRDY=1 (Polling)
MOV DX, 04D0H   ; Adresa port Date
IN  AL, DX      ; Citește caracterul primit
MOV CL, AL      ; Stochează rezultatul în CL
RET             ; Final rutină recepție
```

Transmisie la interfața paralelă:

```
PAR: MOV DX, 0254H ; Adresa Portului C (Registrul de Stare 8255)
IN AL, DX ; Citire starea curentă a perifericului
RCR AL, 1 ; Verifică bitul 0 (semnalul BUSY)
JNC PAR ; Așteaptă în buclă până când perifericul este gata (BUSY=1)
MOV AL, CL ; Încărcare caracter pentru transmisie
MOV DX, 0250H ; Adresa Portului A (Portul de Date 8255)
OUT DX, AL ; Scrie data pe Portul
AOR AL, 01H ; modificăm bitul 0 pe 1

MOV DX, 0252H ; sau 0A52H, adică adresele de port pentru portul B
OUT DX, AL ; !STB = 1, adică nu sunt date de citit
AND AL, 00H
OUT DX, AL ; !STB = 0, adică sunt date de citit
OR AL, 01H
```

OUT DX, AL ; !STB = 1, adică nu mai sunt date de citit
RET

Secventa ca led-urile sa lumineze

```
MOV    AL,00H
OUT    D0H,AL ;SL1
MOV    AL,00H
OUT    F0H,AL ;SL2
```

Secvența ca LED – ul să nu lumineze:

```
MOV    AL,FFH
OUT    D0H,AL ;SL1
MOV    AL,FFH
OUT    F0H,AL ;SL2
```

Secvenţe pentru afişarea caracterelor hexazecimale:

Afişarea cifrei 0 pe primul rang:

```
MOV AL,0C0H
OUT    10H,AL  ;SA1
```

Afişarea cifrei 2 pe primul rang:

```
MOV AL,0A2H
OUT    10H,AL  ;SA1
```

Afişarea cifrei 6 pe al 2 - lea rang:

```
MOV AL,82H
OUT    30H,AL ;SA2
```

Afişarea cifrei A pe al 3 – lea rang:

```
MOV AL,88H
OUT    50H,AL ;SA3
```

Afişarea cifrei 8 pe al 4 - lea rang:

```
MOV AL,80H
OUT    70H,AL ;SA4
```

Afişarea cifrei 9 pe al 5 - lea rang:

```
MOV AL,90H
OUT    40H,AL ;SA5
```

Afişarea cifrei A pe al 6 - lea rang:

```
MOV AL,88H
OUT    60H,AL ;SA6
```

Afişarea cifrei C pe al 7 – lea rang:

```
MOV AL,0C9H
OUT     A0H,AL ;SA7
```

Afişarea cifrei E pe al 8 - lea rang:

```
MOV AL,89H
OUT     E0H,AL ;SA8

REIA: MOV       AL,0FEH
OUT     90H,AL ;ST1
IN      AL,B0H  ;~ST2
AND     AL,01H
JZ      TASTA1
IN      AL,20H
AND     AL,02H
JZ      TASTA4
IN      AL,20H
AND     AL,04H
JZ      TASTA7
IN      AL,20H
AND     AL,08H
JZ      TASTA *

MOV     AL,0FDH
OUT     90H,AL ;ST1
IN      AL,B0H ;~ST2
AND     AL,01H
JZ      TASTA2
IN      AL,20H
AND     AL,02H
JZ      TASTA5
IN      AL,20H
AND     AL,04H
JZ      TASTA8
IN      AL,20H
AND     AL,08H
JZ      TASTA0

MOV     AL,0FBH
OUT     90H,AL ;ST1
```

```
IN      AL,B0H ;~ST2
AND     AL,01H
JZ      TASTA3
IN      AL,20H
AND     AL,02H
JZ      TASTA6
IN      AL,20H
AND     AL,04H
JZ      TASTA9
IN      AL,20H
AND     AL,08H
JZ      TASTAB

JMP REIA
TASTA1: CALL DELAY ; se așteaptă stabilizarea contactelor
AST1: IN        AL,B0H ; se citește din nou linia și se așteaptă dezactivarea
;tastei
AND     AL,01H
JZ      AST1
CALL DELAY
; operația corespunzătoare acționării tastei 1
JP
```

Bibliografie

- Mircea Popa, „Sisteme cu microprocesoare”, Editura Orizonturi Universitare, 2000.
- https://cv.upt.ro/mod/resource/view.php?id=50476 (Cursul 10)
- https://cv.upt.ro/mod/resource/view.php?id=48602 (Cursul 4)
- https://cv.upt.ro/mod/resource/view.php?id=49123 (Cursul 5)
- https://cv.upt.ro/mod/resource/view.php?id=48604 (Cursul 6)
- http://www.calc.fcim.utm.md/biblioteca/arhiva/Anul%20II/Semestru%20II/Arhitectura%20Calculatoarelor/Indrumar%20Proiect%20UCro(fr).pdf
- https://sites.google.com/site/labpmd/Laborator/conectarea-memoriilor