Facultatea de Automatică și Calculatoare

Universitatea Politehnica Timișoara

Universitatea Politehnica Timișoara

Facultatea de Automatica si Calculatoare

Proiectarea cu Microprocesoare

Implementarea unui Microsistem cu Microprocesorul 8086

Realizat de:

Dobra Mihai

Anul universitar 2024-2025

Tema Proiectului

Proiectarea unui microsistem bazat pe microprocesorul 8086, care include unitate centrală, memorii EPROM și SRAM, interfețe seriale și paralele, minitastatură, leduri, afișaj cu 7 segmente și modul LCD, precum și programe pentru configurarea și funcționarea acestuia.

Structura microsistemului:

- unitate centrală cu microprocesorul 8086;

- 128 KB memorie EPROM, utilizând circuite 27C512;

- 64 KB memorie SRAM, utilizând circuite 62256;

- interfaţă serială, cu circuitul 8251, plasată în zona 04D0H – 04D2H sau 05D0H – 05D2H, în funcţie de poziţia microcomutatorului S1;- interfaţă paralelă, cu circuitul 8255, plasată în zona 0250H– 0256H sau 0A50H – 0A56H, în funcţie de poziţia microcomutatorului S2

;- o minitastatură cu 9 contacte;

- 10 led-uri;

- un modul de afişare cu 7 segmente, cu 8 ranguri (se pot afişa maxim 8 caractere hexa simultan);

- un modul LCD, cu 2 linii a câte 16 caractere fiecare, cu o interfaţă la alegerea studentului

Programele:

- 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

Unitatea de Control

Unitatea centrală a microsistemului proiectat are la bază microprocesorul 8086, un dispozitiv pe 16 biți, utilizat pentru controlul și procesarea datelor. Acest microprocesor oferă funcționalități avansate, incluzând magistrale separate pentru date și adrese (printr-o tehnologie de multiplexare), moduri de adresare variate și un set complex de instrucțiuni.

Configurarea microprocesorului și modul de lucru

Microprocesorul 8086 funcționează în două moduri principale care se pot seta prin pinul MN/MX#

- **Modul Minim:** In acest mod microprocesorul gestionează el însuși semnalele de control, activarea memoriilor si dispozitivele de intrare/ieșire, plus ca un are nevoie de un bus extern. Este modul pe care este folosit in proiect deoarece este adecvat pentru sisteme simple si de acea este conectat la VCC, pentru a fi activat MN pe “1” logic.
- **Modul Maxim:** Permite utilizarea unui coprocesor sau a unor periferice complexe, microprocesorul având un rol central de coordonare.

Semnalele principale ale microprocesorului

- **AD0-AD15 (Magistrala multiplexată):** Utilizează aceleași pini pentru date și adrese, reducând numărul de conexiuni necesare.
- **A15-A19:** Pini din rangurile de magistrala de adrese adiționale.
- **BHE# (Bus High Enable):** Indica daca are sau nu loc un transfer pe jumătatea superioara a magistralei de date.
- **ALE (Addres Latch Enable):** Activează memorarea adreselor din magistrala multiplexata.
- **RD# si WR#:** Controlează operațiile de citire/intrare respectiv scriere/ieșire, sincronizând accesul la magistrale.
- **M/IO# (Memory/Input-Output):** Determina daca operația curenta este destinata memoriei (când are valoarea 1) sau unui dispozitiv prin porturile de intrare/ieșire (valoarea 0)
- **DT/R# (Data Transmit/Receive):** Ne arata sensul transferului de pe magistrala de date (“1” indica transmisie si ”0” indica recepție)
- **DEN# (Data Enable):** Validează transferul de date pe magistrala.
- **RESET:** Pentru resetarea/inițializarea procesorului.
- **CLK:** Este intrarea semnalului clock, in acest caz cu o frecventa data de dispozitivul 8284A descris mai jos.
- **READY (Wait State Control):** Intrare pentru sincronizarea microprocesorului si dispozitivele lente, când este activ (nivel înalt) microprocesorul continua operația curenta. Când este inactiv (nivel jos) el intra într-o stare de așteptare (wait state) pana când perifericul sau memoria finalizează operațiunea.

Registrul de date 74x373

Acest dispozitiv cu 8 ranguri este utilizat pentru stocarea temporara a datelor, având trei stări de funcționare (activ, neactiv si a 3-a stare). In lucrare sunt amplasate 3 registre care sunt utilizate pentru a memora adresele multiplexate de la microprocesor plus semnalul BHE#. Pinul G (Enable) de la fiecare latch sunt legate la semnalul ALE pentru a sincroniza capturarea adreselor din magistrala multiplexată, astfel încât adresele stocate sa fie corect stabilite înainte de începerea oricărei operații de citire/scriere. Pinul OC# controlează activarea ieșirilor latch-ului si pentru a funcționa pe cele doua stări activ/neactiv trebuie conectat la masa (0 logic).

In primul registru (cu identificatorul U6) stochează biți de la A16 la A19 si semnalul BHE. Al doilea (U12) stochează biți de la AD0 la AD7. Si al treilea (U13) stochează biți de la AD8 la AD15.

Circuitul Amplificator/Separator bidirecțional 74x245

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 (U15) cuprinde rangurile de intrare AD8-AD15 si ieșirile la magistrala D8-D15.

Geratorul de Clock 8248A

Acesta generează semnalul de ceas necesar sincronizări operațiunilor interne ale microprocesorului si asigura stabilitatea întregului sistem. Generatorul primește semnalul brut de la oscilatorul de cristal, care este conectat la el, si îl prelucrează oferind un semnal stabil pe care il va folosi microprocesorul. In plus, furnizează si semnale auxiliare precum READY si RESET care le folosește microprocesorul.

In proiectul nostru (U4) la pini X1 si X2 se conectează cristalul de cuarț de 15MHz plus doua condensatoare de 20pF conectate la masa pentru a stabiliza oscilația. Pinul EFI (External Frequency Input) este conectat la masa pentru ca nu vom folosi un semnal de cea extern. Si la fel F/C# (Frequency/Crystal Select) selectează intre cristal si o sursa externa si in acest caz va fi legat la masa (selectând modul cristal). Si CSYNC (Clock Synchronization) va fi legat la masa deoarece el sincronizează generatorul cu alte surse de ceas externe, cea ce nu avem in proiect. La ieșire avem semnalele CLK, RESET si READY care ajung la intrările din 8086. Semnalul PCLK este o versiune mai lenta a semnalului CLK care se folosește pentru a sincroniza perifericele care funcționează la o frecventa mai mica, si care ne va ajuta la interfețe. Pinul RES# este intrarea pentru semnalul de resetare extern care îl legam la un buton de resetare si un circuit RC si o dioda pentru a asigura o resetare corecta la pornirea sistemului.

Conectarea Memoriilor

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)

Planificarea spațiului de adresare

Microprocesorul 8086 dispune de o magistrala de adresa pe 20 de biți, permițând accesarea unui spațiu maxim de 2^20 = 1MB. Deci toate adresele posibile pentru acest procesor cuprind rangul: 00000H-FFFFFH. Când se construiește harta memoriei trebuie stabilite doua date pentru fiecare circuit (bloc) de memorie, adresa de început si limitele de adresare.

- Pentru SRAM spațiul începe de la adresa 00000H, iar limita superioara este determinata de adăugarea capacității memoriei (64 KB), rezultând) 0FFFFH.
- Pentru EPROM am ales adresa de început 20000H, alocând următorii 128KB. (20000H + 1FFFFH = 3FFFFH).

Aceste ranguri pe care le am ales ne va ajuta la decodificarea memoriilor.

Memoria SRAM 62256

SRAM este utilizat pentru stocarea datelor temporare necesare procesării, având operații rapide de citire si scriere. Este o memorie volatila care își pierde informația o data ce alimentarea este întrerupta si stocarea datelor se realizează prin bistabile de tip flip-flop care nu necesita de operați periodice pentru refresh. Capacitatea chipului este de 32 KB necesitând 2 chipuri pentru a ajunge la capacitatea de 64 KB. In plus, se face împărțirea in 2 chipuri pentru ca procesorul 8086 lucrează cu adrese pare respectiv impare, de acea pot sa leg un bloc pe jumătatea superioară a magistralei de date respectiv jumătatea inferioara. Utilizează 15 biți de adresa (A0-A14). Conexiunile principale:

- **WE# (Write Enable):** este pinul care activează scrierea de date si fiecare chip SRAM are un semnale WE# diferit pentru a asigura accesul separat la adresele pare si impare. Sunt generate de un circuit logic combinațional (U16) format din doua porți sau cu semnalele WE, BHE si A0. Prima poarta sau combina semnalul BHE# care activează un semnal pe magistrala superioara de date D15-D8 cu WR# de la procesor, generând semnalul WE# pentru blocul de memorie care gestionează adresele impare (U7). Si a doua poarta ia semnalul WR# de la procesor si A0 care este bitul cel mai puțin semnificativ de la o adresa si daca este 0 e para, daca e 1 este impara, generând semnalul WE# pentru blocul care gestionează adresele pare (U11). Deci procesorul poate sa scrie independent un octet superior in blocul (U7), un octet inferior in blocul (U11) sau ambii octeți simultan.
- **OE# (Output Enable):** Activează ieșirile pentru citire si este conectat la pinul RD# de la 8086.
- **CE# (Chip Enable):** Activează circuitul pentru acces la date si este conectat la o ieșire al decodificatorului de memorii.

Memoria EPROM 27C512

EPROM este utilizat pentru stocarea codului programului accesul fiind exclusiv de citire. Fata de SRAM acest tip de memorie este non-volatila iar ștergerea datelor se face doar prin expunerea la radiații ultraviolete. Capacitatea chipului este de 64 KB necesitând 2 chipuri pentru a ajunge la capacitatea de 128 KB. Utilizează 16 biți de adresa (A0-A15). Conexiunile principale:

- **OE# (Output Enable):** La fel ca la SRAM.
- **CE# (Chip Enable):** La fel ca la SRAM.
- **VPP (Voltage Programming Pin):** După ce memoria a fost programata pinul VPP trebuie conectat la VCC deoarece in aceasta configurație memoria va funcționa doar in modul citire, iar datele programate rămân intacte.

Decodificarea memoriilor

Pentru a genera semnalele de selecție corect pentru fiecare memorie in funcție de adresă, am implementat un decodificator bazat pe un circuit 74LS138 si logica combinațională. Se construiește un tabel care are ca si coloane rangurile magistralei de adrese iar ca rânduri prima si ultima configurație de adresa a fiecărei tip de memorie.

Adresa (Hex)	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Memorie
0000fh	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SRAM
0ffffh	**0**	**0**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
20000h	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EPROM
3ffffh	**0**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Verde: Spațiul de adrese pentru SRAM

Albastru: spațiul de adrese pentru EPROM

Gri: Ranguri neschimbate, constante pe parcursul adreselor

Decodificarea Completa a semnalelor de selecție pentru memorii

Decodificarea Completa presupune utilizarea tuturor biților de adresa relevanți pentru generarea semnalelor de selecție. Pornind de la rangul cel mai semnificativ spre cel mai puțin se rețin acele ranguri care rămân nemodificate pentru orice configurație de adresa din zona respectiva. In cazul proiectului, rangurile relevante pentru SRAM sunt A19, A18, A17,A16, iar pentru EPROM sunt A19, A18 si A17. Deci funcțiile combinaționale ar fi:

$!Sel_{SRAM} = !A19 * !A18 * !A17 * !A16$

$!Sel_{EPROM} = !A19 * !A18 * A17$

Si ne ar trebui un decodificator 4 la 16 in care SRAM sa fie activ doar pe combinația 0000 la intrarea decodificatorului iar EPROM va fi activ in combinațiile 001X. Sunt multe ieșiri nefolosite din decodificator plus multe care activează EPROM-ul de in proiect se folosește o decodificare incompleta.

Decodificarea Incompleta a semnalelor de selecție pentru memorii

Observam ca sunt multe ranguri care sunt egale pentru SRAM si EPROM (A19 si A18), putem sa ignoram aceste ranguri si sa simplificam decodificatorul. Acum ne ar trebui doar un decodificator 2 la 4.

$!Sel_{SRAM} = !A17 * !A16$

$!Sel_{EPROM} = A17$

Problema este ca un circuit „va fi văzut de procesor” in mai multe zone din spațiul de adresare al procesorului, pentru ca cele doua ranguri ignorate vor fi „dont-care” si in acest caz va ocupa 4 zone egale de memorie cu dimensiunea respectiva.

Se putea folosi in acest caz si un circuit simplu logic combinațional, in loc de decodificator, cu o poarta si si un invertor pentru a selecta intre cele doua memorii doar cu rangurile A17 si A16

Circuitul decodificator 74LS139

In cazul asta, decodificatorul 74LS139 (U) este configurat pentru a genera semnale de selecție pentru memoriile EPROM si SRAM. Când microprocesorul transmite o adresa pe magistrala, decodificatorul si logica combinațională determina memoria care trebuie activata. Are 2 intrări si 4 ieșiri duplicate, adică sunt doua decodificatoare 2 la 4 in același bloc. Am omis unu din ele pentru ca este necesar doar de unul singur. La intrările 1A si 1B se leagă A16 respectiv A17. Pini Enable funcționează ca un „switch” in care se spune daca decodificatorului este on/off. Pini E1A# si E2B# vor fi totdeauna conectați la masă pentru ca decodificatorul va funcționa tot timpul timp ce E3 este conectat la M/IO# de la 8086 si va activa cu „1” logic funcționarea decodificatorului. Tabelul de adevăr pentru ieșirea decodificatorului este următorul:

A17	A16	Output
0	0	1Y0# (Ajunge la CE# de la SRAM)
0	1	1Y1# (Nu este necesar)
1	0	1Y2# (Ajunge la CE# de la EPROM)
1	1	1Y3# (Ajunge la CE# de la EPROM)

Se observa ca 1Y2# si 1Y3# sunt folosite pentru același scop. Pentru a rezolva acest conflict se plasează o poarta Si astfel încât când una din ele este activa va fi pe “0” logic activând memoriile EPROM

Nu este rezolvarea cea mai perfecta, se putea face cum am indicat mai sus fără un decodificator, cu o poarta sau si un inversor pentru a nu încarcă mai mult complexitatea proiectului.

Decodificatorul de porturi (interfete)

Interfețele seriala si paralela

Circuitul 8251 (Interfața Seriala)

Convertește datele paralele (din procesor) in date seriale pentru transmisie, sau primește date seriale si le transforma in paralele (U17). Permite transferul de date intre sisteme care se afla la distante mari unul de altul folosind un număr redus de fire. Exista doua feluri de comunicare, cea sincrona in care se folosește o linie de tact comun, si cea asincrona in care trebuie adăugată informație suplimentara la octetul care urmează a fi transferat definind un standard pentru transmisie (Bit de start, biți de stop, factor de multiplicare)

Pentru configurarea dispozitivului se folosește Cuvântul de mod care configurează modul de operație(biți de date, paritate, stop bit, sincronizare) si cuvânt de comanda care controlează starea de transmisie si recepție (începere, oprire).

- **Cuvânt de Mod**: In proiect se configurează astfel încât sa funcționeze in mod asincron si de acea are 2 biți de stop (11) in cazul asta, fără control de paritate (00) adică nu se adaugă bit suplimentar pentru verificarea parități, 8 biți de date (11) unde fiecare caracter transmis sau recepționat are 8 biti, si un factor de multiplicare x16 (10) care înseamnă ca rata de transfer a semnalului de clock este împărțita pentru a genera semnalul intern astfel încât sa fie de 16 ori mai rapid decât baund rate-ul dorit.
  In binar -> 11001110 In Hexa -> 0CEH.

- **Cuvânt de Comanda:** bitul de sincronizare (0) care e folosit doar in mod sincron pentru a găsi caracterul de sincronizare, bitul de reinițializare (0) care resetează starea interna a dispozitivului, bitul de comanda RTS (0) este un semnal folosit pentru controlul fluxului de date. Bitul de error reset (1) care resetează biți de eroare, bitul de break (0) transmite un semnal BREAK care reprezintă o stare continua de 0 logic pe linia de transmisie (TXD), utilizata pentru sincronizare sau semnalizarea unei condiții speciale, bitul de comanda recepție (1) care permite 8251 sa primească date prin linia RXD, bitul de comanda DTR (0) care este folosit pentru a indica faptul ca terminalul de date este gata pentru comunicare, si bitul de comanda transmisie (1) care permite transmisia de date de la 8251 prin lina TXD.
  In binar -> 00010101 In Hexa -> 15H.

- **Pini importanți: C/D#** (Comand/Data): selectează intre portul de date (0 logic) si portul de comanda/stări (1 logic), in cazul asta este legat la A1 de pe magistrala de adrese si se explica pentru ca avem asignata zona de adrese 04D0H -04D2H sau 05D9H -05D2H, in funcție de poziția microcomutatorului S1, in pereche adresele diferă prin bitul de pe rangul A1 cum se observa in tabelul de mai jos. Pinii **WR#** si **RD#** controlează operațiile de scriere si citire si sunt legați direct la 8086. Pinul **CS#** este conectat la ieșirea comutatorului S1 (U20) si el activează sau dezactivează circuitul. Pinul **RST** este legat la RESET de la 8284A si **CLK** este legat la PCLK, am explicat rolul lui înainte.

- **Pini de recepție**: in cazul acesta in proiect doar avem conectat doar pinul RXD la intrarea R1OUT de la MAX232 deoarece prin aceste recepționează datele bit cu bit. RXC# este folosit in modul sincron pentru a sincroniza recepția datelor cu un semnal de clock. RXRDY este folosit in handshake (sincronizare cu alte dispozitive) pentru ca el indica faptul daca 8251 are date disponibile pentru citire. Si SYNDET in mod sincron indica detectarea unui caracter de sincronizare iar in mod asincron poate fi utilizat ca intrare/ieșire programabila pentru control.
- **Pini de transmisie**: la fel ca la recepție, in proiect se folosește doar pinul TXD pentru a transmite datele bit cu bit la pinul T1IN al MAX232. Pini TXC# si TXRDY funcționează la fel cu cei de la recepție doar ca pentru transmitere de date. Pinul TXE indica starea buffer-ului, adică daca toate datele din el au fost transmise si buffer-ul este gol.
- **Pini de control pentru handshake** sunt folosite pentru asigurarea ca datele sunt transmise si recepționate doar când ambele parți sunt pregătite(DSR#, DTR#, CTS# si RTS#)dar nu se folosesc in proiect.

Prin pini TXD si RXD este conectat un transceiver RS-232 (MAX 232 - U47). Este un circuit de conversie de nivel logic folosit pentru a face compatibile nivelurile TTL/CMIS de la 8251 cu standardul RS-232 folosit in multe dispozitive seriale. Este conectat la el condensatoare pentru a asigura conversia corecta a nivelurilor de tensiune de la TTL/CMOS spre RS-232 si viceversa. De la TTL la RS-232 va transforma semnalul cu o Amplitudine mai mare si inversat. De exemplu: -> TTL (0V – 5V) la RS-232 (-12V - +12V).

Circuitul 8255 (Interfața Paralela)

Este utilizat pentru conectarea dispozitivelor periferice paralele, cum ar fi tastaturi, afișaje (cum este cazul cu un ecran LCD), sau alte module I/O. (U22) EL permite comunicarea bidirecționala intre procesor si perifericele transmițând sau receptând date in mod paralel prin intermediul celor 3 porturi de cate 8 biți fiecare (Port A, Port B si Port C), unde fiecare poate funcționa ca intrare sau ieșire. In total ar fi 24 de linii I/O care pot fi programate individual sau in grupuri. Are trei moduri de operare, in **Modul 0** (intrare/ieșire de baza), care este cel folosit in proiect, fiecare port poate fi configurat ca intrare sau ieșire si nu exista semnale de strobe sau handshake. **Modul 1** (intrare/iesire sincronizata) se folosesc porturile A si B pentru I/O cu posibilitatea de handshaking iar portul C e folosit pentru semnale de control. **Modul 2** (Transfer bidirecțional) Permite transmiterea bidirecționala de date, disponibil doar pentru portul A ,utilizând liniile din Port C pentru controlul direcție.

Pentru configurarea dispozitivului se folosește **cuvântul de control** care este configurat pentru a lucra cu ecranul LCD (U45) din proiect. Cel mai semnificativ bit este bitul de care specifica tipul de cuvânt de control 1 este moduri funcționale sau 0 pentru set/reset port C, așa ca se lasă pe 1 logic. Următori 2 biți sunt pentru a selecta modul de operare al grupului A, in cazul asta va fi modul 0 (00). Portul C superior care cuprinde pini (PC4-PC7) va fi configurat ca ieșire (0). Modul de funcționare al grupului B va fi mod 0 (0). Portul B va fi setat ca ieșire (0). Portul C inferior care cuprinde pini (PC0-PC3) va fi configurat ca ieșire (0). In binar -> 10000000 In Hexa -> 15H.

- **Pini importanți:** Pini **WR#**, **RD#, RESET** si **CS#** au același rol ca la 8251 doar ca CS# este legat la microcomutatorul S2 (U21). Pini **A0** si **A1** sunt intrarile care decid ce registru intern al circuitului sa fie selectat si funcționează ca un decodificator 2-la-4 intern in care primește cele doua intrări si generează patru ieșiri distincte fiecare corespunzând unei combinații a intrărilor. 00-> portul A, 01->portul B, 10->portul C, 11->registrul de control. In cazul asta, cum este asignata zona de adrese 0250H-0256H sau 0A50H-0A56H in funcție de poziția microcomutatorului S2, in pereche adresele diferă prin biți de pe rangul A1 si A2 si care sunt folosiți pentru pini de intrare A0 respectiv A1.

Decodificarea porturilor interfeței seriale si paralele (Circuitul 74LS138)

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 74LS138 (U19) si circuite logice pentru activarea sau dezactivarea acestuia (pini enable). Decodificatorul funcționează la fel ca 74LS139 doar ca are 3 intrări si 8 ieșiri. In acest caz nu se poate folosi un decodificator 2-la-4 pentru ca cele 4 ranguri de adrese pe care vrem sa le diferențiem pentru a le selecta unic, nu diferă in doar 2 biți din magistrala de adrese, in cazul asta ne mai trebuie încă un bit pentru a putea face distincția. Acei biții sunt rangurile A11, A10 si A8 cum se poate observa in tabel:

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

Verde: Adresele de porturi pentru Interfața Seriala cu comutatorul pe 0

Albastru: Adresele de porturi pentru Interfața Seriala cu comutatorul pe 1

Roșu: Adresele de porturi pentru Interfața Paralela cu comutatorul pe 0

Galben: Adresele de porturi pentru Interfața Paralela cu comutatorul pe 1

Gri: Ranguri de selecție pentru decodificator

Funcțiile combinaționale:

$!Sel_{SERIALA0} = !A11 * A10 * !A8$

$!Sel_{SERIALA1} = !A11 * A10 * A8$

$!Sel_{PARALELA0} = !A11 * !A10 * !A8$

$!Sel_{PARALELA1} = A11 * !A10 * !A8$

Este o decodificare incompleta pentru ca am omis primele ranguri A15-A12.

Tabelul de adevăr pentru ieșirea decodificatorului este următorul:

A11	A10	A8	Output
0	0	0	Y0# (Ajunge la pinul 3 al comutatorului S2 – interfața paralela)
0	0	1	Nefolosit
0	1	0	Y2# (Ajunge la pinul 3 al comutatorului S1 – interfața seriala)
0	1	1	Y3# (Ajunge la pinul 2 al comutatorului S1 – interfața seriala)
1	0	0	Y4# (Ajunge la pinul 2 al comutatorului S2 – interfața paralela)
1	0	1	Nefolosit
1	1	0	Nefolosit
1	1	1	Nefolosit

Se observa ca sunt doua microcomutatoare deoarece sunt doua intervale de adrese pentru fiecare interfața, si pentru a selecta intre una sau alta se apasă sau nu butonul. Când e „0”, adică nu e apăsat butonul, este legat la pinul 3 al comutatorului, iar când este apăsat butonul „1” atunci se leagă la pinul 2.

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 activând E3 al decodificatorului de memorie.

E2# = A15+A14+A13+A12+A5

Aceste ranguri sunt 0 peste toate intervalele de adrese folosite pentru interfețe. Sunt selectați așa astfel încât daca unul din ei este pe 1 înseamnă ca este alta adresa care nu e in rangul de adrese pentru interfețe si nu se va activa.

E3 = (A6 * A4) * (A9 ^ A7)

Rangurile A6 si A4 sunt pe 1 logic peste tot, folosind un AND iar A9 si A7 ori sunt pe 1 ori sunt pe 0 dar nici pe 1 sau pe 0 deodată, de acea se folosește un XOR. [1]

Conectarea Afișajelor si a minitastaturi la decodificatorul de porturi

Conectarea Ledurilor

Se plasează 10 leduri in proiect care se vor conecta la anod comun. Se conectează anodul cu un potențial mai mare decât cel de la catod. Diferența potențialelor dintre Anod si catod trebuie sa fie mai mare de cat tensiunea de prag (Vt) pentru ca dioda sa fie polarizata direct si ledul sa lumineze. Aceasta tensiune de prag poate diferi in funcție de lumina pe care o emite ledul. In plus, folosim o rezistenta de 330 de ohm in cazul asta conectat la Vcc si anod pentru limitarea curentului si prin varierea voltajului se reduce sau accentuează intensitatea luminoasa a ledului. Catodul ledurilor se conectează la ieșirile circuitelor de registru 74LS373 (explicat la unitatea de control) in care va trebui sa setam bistabilul pe (0 logic) pentru ca sa fie acea diferență de potențial ca ledul sa fie aprins, daca nu va fi stins (1 logic). Se folosesc doua registre pentru ca in primul (U32) conectam 8 leduri prin pini 1Q-8Q iar cele doua leduri ramase le conectam la 1Q respectiv 2Q de la (U33). Pinul OC# la grund si pini G din fiecare la o ieșire diferita la decodificatorul de porturi explicat mai jos.

Conectarea afișajelor cu 7 segmente

Se plasează 8 module de afișare de 7 segmente. Fiecare rang este compus de 8 leduri conectate împreuna (7 pentru segmente si unul pentru punct) si fiecare led este accesibil prin pini de la circuit. La fel ca si la leduri, exista doua tipuri de conexiuni, cu anod comun (este cea folosita in proiect) sau catod comun. Cele 8 ieșiri ale modulului sunt conectate la un registru 74LS373, si cum avem 8 ranguri vor fi in total 8 registre (U34,U35,U36,U37,U38,U39,U40 si U41). Cum se folosește catod comun pentru a aprinde ledul se va comanda cu (0 logic). Pentru a afișa un număr hexa pe un rang al modulului exista 2 soluții, cea hardware si cea software. Dar in cazul asta se folosește soluția software care poate afișa oricare configurație cu forma celor 7 segmente cerând un registru. Software-ul este mai complex ca in soluția hardware. Pentru a putea comanda un modul cu segmente multe ranguri exista doua soluții, cea multiplexata in timp si cea **nemultiplexata in timp**. In cazul asta se folosește cea nemultiplexata in care fiecare registru va fi comandat ca port de ieșire (prin pinul G) si vor memora configurațiile care se vor afișa. Avantajul este implementarea software mai simpla dar dezavantajul este o folosire mare de registre si circuite plus un consum mai mare, daca ar fi cea multiplexată am avea nevoie doar de un registru pentru toate modulele.

Tabelul pentru intrarea la registrul de date 74LS373 pentru afișarea cifrelor hexa pe un rang:

Hex	DP	G	F	E	D	C	B	A	In hexa
0	1	1	0	0	0	0	0	0	C0H
1	1	1	1	1	1	0	0	1	F9H
2	1	0	1	0	0	1	0	0	A4H
3	1	0	1	1	0	0	0	0	B0H
4	1	0	0	1	1	0	0	1	99H
5	1	0	0	1	0	0	1	0	92H
6	1	0	0	0	0	0	1	0	82H
7	1	1	1	1	1	0	0	0	F8H
8	1	0	0	0	0	0	0	0	80H
9	1	0	0	1	1	0	0	0	98H
A	1	0	0	0	1	0	0	0	88H
B	1	0	0	0	0	0	1	1	83H
C	1	1	0	0	0	1	1	0	C6H
D	1	0	1	0	0	0	0	1	A1H
E	1	0	0	0	0	1	1	0	86H
F	1	0	0	0	1	1	1	0	8EH

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, in cazul asta, tastele din prima coloana 1,4 si 7, tastele din a doua coloana 2,5 si 8, si tastele din a treia coloana 3, 6 si 9. Iar pinul 2 este conectat cu toate tastele din rândul respectiv, primul rând fiind tastele 1,2 si 3, al doilea este 4,5 si 6, si al treilea este 7,8 si 9. Cele 3 coloane sunt legate la o anodul unei diode si catodul diodei este legat la 3 ieșiri al circuitului registru 74LS373 (U43).Diodele sunt folosite ca protecție a ieșirilor portului de ieșire. Iar cele trei rânduri sunt conectate la VCC trecând prin 3 rezistente de 10kohm si la 3 intrări ale circuitului 74LS244 (U42). La început tastele sunt pe „iddle” adică nu trece curent prin ele (nu sunt conectate cu circuitul). Pentru a detecta o tastare trebuie sa aplicam principiul de scanare al tastaturi care începe prin a conecta coloanele, trecând prin dioda, la „0 logic” in registrul de ieșire (U43). Atunci când o tasta este apăsata, va curge curentul prin ea si cum exista acea diferența de potențial intre Vcc si legarea coloanei la 0 in registru, se va putea citi pe rând rândurile legate la intrarea circuitului U42 pentru a vedea daca tasta a fost apăsata sau nu. Circuitul U42 este folosit pentru amplificarea / separarea magistralelor unidirecționale, si in acest caz, când este apăsat o tasta pe rândul acelei taste se va genera un 0 logic care va fi transmis pe magistrala de date indicând ca pe rândul respectiv sa acționat o tasta. Știind ce coloana si ce rând sau acționat,

prin coordonate se știe ce tasta sa apăsat. Aceasta lucrare alternata intre cele doua porturi pentru a scana tastele prin coloane si rânduri se poate vedea in rutina de scanare.

Decodificarea porturilor pentru Afișaje si minitastatura (Circuitul 74LS154)

Se folosește decodificarea a porturilor de afișaje si a minitastaturi pentru a putea selecta unul dintre iei la un moment dat. In cazul asta cum avem 2 porturi pentru leduri, 8 pentru segmente si 2 pentru minitastatura, se folosește un decodificator 4-la-16 (U44). Acest decodificator are 4 intrări si 16 ieșiri dintre care vom folosi doar 12. Adresele de porturi le am ales in intervalul 0300H la 032Ch incrementat cu cate 4 deoarece biți care diferă intre ele si care sunt unice sunt rangurile A2,A3,A4 si A5. Aceste ranguri sunt conectați la intrarea decodificatorului astfel încât fiecare ieșire din decodificator sa corespunda a unui periferic specific. Toate porturile, mai puțin circuitul U42 de la minitastatura, sunt active pe „1” logic la intrarea pinilor G, de acea se leagă un inversor la fiecare ieșire a decodificatorului. Decodificare folosita este incompleta.

Tabelul de adrese pentru porturile de afișaje si minitastaura:

Adresa (Hex)	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Porturi Afisaje, tastatura
0300H	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	SL1
0304H	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	SL2
0308H	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	SA1
030CH	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	SA2
0310H	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	SA3
0314H	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	SA4
0318H	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	SA5
031CH	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0	SA6
0320H	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	SA7
0324H	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0	SA8
0328H	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	ST1
032CH	0	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	ST2#

Verde: Adresele de porturi pentru Leduri.

Albastru: Adresele de porturi pentru Modulu de segmente cu 8 ranguri

Galben: Adresele de porturi pentru minitastatura

Gri: Ranguri de selecție pentru decodificator

Tabelul de adevăr pentru ieșirea decodificatorului este următorul:

A5	A4	A3	A2	Output
0	0	0	0	Y0# (Conectat la portul 1 al ledurilor U32)
0	0	0	1	Y1# (Conectat la portul 2 al ledurilor U33)
0	0	1	0	Y2# (Conectat la portul 1 rang seven segment U34)
0	0	1	1	Y3# (Conectat la portul 2 rang seven segment U35)
0	1	0	0	Y4# (Conectat la portul 3 rang seven segment U36)
0	1	0	1	Y5# (Conectat la portul 4 rang seven segment U37)
0	1	1	0	Y6# (Conectat la portul 5 rang seven segment U38)
0	1	1	1	Y7# (Conectat la portul 6 rang seven segment U39)
1	0	0	0	Y8# (Conectat la portul 7 rang seven segment U40)
1	0	0	1	Y9# (Conectat la portul 8 rang seven segment U41)
1	0	1	0	Y10# (Conectat la portul de ieșire al minitastaturi U43)
1	0	1	1	Y11# (Conectat la portul de intrare al minitatataturi U42)

La fel ca la decodificatorul de porturi pentru interfețe, avem nevoie de semnale enable pentru activarea sau dezactivarea decodificatorului. Trebuie alese rangurile altfel încât adresele de la porturile de la afișaje si minitastatura sa nu interfereze cu adresele pentru porturile de la interfețe. Pentru E1# se folosește semnalul M/IO# de la 8086 (ca la decodificatorul pentru interfețe).

E2# = A15+A14+A13+A12+A11+A10

Pentru E2 am mai adăugat ranguri care sunt pe 0 logic pana la A10.

E3 = A9*A8

Sunt folosit A9 si A8 pentru ca sunt constanți pe 1 logic pe adresele selectate

Conectarea Modulul LCD (Interfața paralela)

Este plasat un modul LCD 1602 (U45) cu 2 linii a cate 16 caractere fiecare, cuplata la interfața paralela. Pentru simplificare se folosește interfața paralela pentru ca este ca mai comuna, ușor de implementat si transferul in paralel este mai rapid fiind ideal la actualizări rapide ale afișajului. Modul in acest caz funcționează cu modul pe 8 biți care folosește toți cei 8 pini de date (DB0-DB7), dar mai poate funcționa in modul de 5 biți care utilizează doar pini (D4-D7) pentru a economisii pinii de pe 8255 fiind necesara trimiterea datelor in doua pachete (high nibble si low nibble). Pinul Vo se folosește pentru reglarea contrastului afișajului ecranului si este conectat la un potențiometru pentru al putea ajusta. Pini BLA (Backlight Anode) si BLK(Backlight Cathode) controlează iluminarea de fundal a ecranului care face ca textul afișat sa fie mai lizibil sau nu si pentru ajustarea luminozități se conectează o rezistenta intre Vcc si BLA. Pinul RS(Register select) selectează daca semnalul trimis reprezintă o comanda (0 logic) sau date (1 logic). R/W selectează direcția datelor, adică cu 0 logic este scriere (de la 8255 către LCD) iar pentru 1 logic este citire. E este pinul de Enable care activează modulul cu o tranziție de la 0 la 1 logic. Cum este configurat 8255 in proiect, portul A va fi pentru transmise de date (DB0-DB7) iar portul C inferior este configurat pentru semnalele de control (R/S, R/W si E). Deci adresele de porturi pentru controlarea modulului sunt cunoscute.

Descrierea Software

Rutinele de programare ale circuitelor 8251 si 8255

```
; Ne definim adresele porturilor pentru 8251 si 8253
PORT_8251_1 EQU 04D0H
PORT_8251_2 EQU 05D0H
PORT_8255_1 EQU 0250H
PORT_8255_2 EQU 0A50H
;-------------------------------------------
; Subrutina pentru configurarea circuitului 8251
; Input: Al = 0 pentru S1 = 0, Al = 1 pentru S1 = 1
; Configuram Cuvantul de Mod (mod asincron): 2 biti de stop,
; fara paritate, 8 biti de date si factor de multiplicare 16
; Configuram Cuvantul de Comanda: Resetarea biti de eroare,
; activarea receptia datelor, activa transmiterea datelor
;-------------------------------------------
CONFIG_8251 PROC
   CMP AL,0            ; verificam daca S1 = 0 sau S1 = 1
   JE SERIAL_S1_0      ; daca S1 = 1, trecem la configurarea pentru S1 = 1
   MOV DX, PORT_8251_2  ; selectam portul 2
   JMP CONFIG_SERIAL    ; trecem la configurarea circuitului 8251

SERIAL_S1_0:
   MOV DX, PORT_8251_1  ; selectam portul 1

CONFIG_SERIAL:
   MOV AL, 0CEH   ; in binar = 11001110
   OUT DX, AL     ; scriem cuvantul de mod
   MOV AL, 15H    ; in binar = 00010101
   OUT DX, AL     ; scriem cuvantul de comanda
   RET

CONFIG_8251 ENDP
;-------------------------------------------
; Exemplu de utilizare a subrutinei CONFIG_8251
;-------------------------------------------
MOV AL, 0          ; setam AL = 0 pentru S1 = 0
CALL CONFIG_8251    ; apelam subrutina
;-------------------------------------------
; Subrutina pentru configurarea circuitului 8255
```

```
; Input: Al = 0 pentru S2 = 0, Al = 1 pentru S2 = 1
; (Cuvantul de control il vom configura astfel incat sa lucram cu ecrarnul LCD)
; Deci portul A fa vi de iesire, portul B ramane default, portul C inferior este de iesire (Semnalele:S,RW,E),
; si portul C superior ramane default, si modul de lucru 0 la grupul A.
;-----------------------------------------
CONFIG_8255 PROC
    CMP AL,0              ; verificam daca S2 = 0 sau S2 = 1
    JE PARALLEL_S2_0      ; daca S2 = 1, trecem la configurarea pentru S2 = 1
    MOV DX, PORT_8255_2   ; selectam portul 2
    JMP CONFIG_PARALLEL   ; trecem la configurarea circuitului 8255

PARALLEL_S2_0:
    MOV DX, PORT_8255_1   ; selectam portul 1

CONFIG_PARALLEL:
    MOV AL, 80H    ; in binar = 10000000
    OUT DX, AL;    ; scriem cuvantul de control
    RET

CONFIG_8255 ENDP
;-----------------------------------------
; Exemplu de utilizare a subrutinei CONFIG_8255
;-----------------------------------------
MOV AL, 0           ; setam AL = 0 pentru S2 = 0
CALL CONFIG_8255     ; apelam subrutina
```

Rutinele de emisie/recepție caracter pe interfața seriala

```
; Ne definim adresele porturilor ai 8251
PORT_8251_1 EQU 04D0H
PORT_8251_2 EQU 05D0H
;-----------------------------------------
; Subrutina pentru emisie caracter seriala
; Input: CL = caracterul de transmis
;        BL = 0 pentru S1 = 0, BL = 1 pentru S1 = 1
;-----------------------------------------
EMISIE_CARACTER_SERIALA PROC
    ; Calculam baza
    CMP BL,0              ; verificam intervalul
    JE  BASE_1_EMISIE     ; daca BL = 0, folosim baza PORT_8251_1
```

```
    MOV SI, PORT_8251_2   ; altfel, folosim baza PORT_8251_2
    JMP EMISIE_START      ; trecem la emisie

BASE_1_EMISIE:
    MOV SI, PORT_8251_1   ; folosim baza PORT_8251_1

EMISIE_START:
    ; Verificam TxRDY (bitul 1 din cuvantul de stare)
    MOV DX, SI            ; Adresa pentru date
    ADD DX, 2H            ; Adresa pentru comenzi/stari (04D2H sau 05D2H)

TX_WAIT:
    IN AL, DX             ; Citim cuvantul de stare
    RCR AL, 1             ; Testam bitul TxRDY
    JNC TX_WAIT           ; Daca TxRDY = 0, asteptam

    ; Trimitem caracterul
    MOV AL, CL            ; Incarcam caracterul
    MOV DX, SI            ; Adresa pentru date
    OUT DX, AL            ; Transmitem caracterul
    RET
TRANSMIT_CHAR ENDP
;--------------------------------------------
; Exemplu de utilizare a subrutinei EMISIE_CARACTER_SERIALA
;--------------------------------------------
MOV CL, 'A'                   ; caracterul de transmis
MOV BL, 0                     ; baza care o sa fie folosita
CALL EMISIE_CARACTER_SERIALA      ; apelam subrutina
;--------------------------------------------
; Subrutina pentru receptie caracter seriala
; Output: CL = caracterul receptionat
;         BL = 0 pentru S1 = 0, BL = 1 pentru S1 = 1
;--------------------------------------------
RECEPTIE_CARACTER_SERIALA PROC
    ; Calculam baza
    CMP BL,0              ; verificam intervalul
    JE  BASE_1_RECEPTIE   ; daca BL = 0, folosim baza PORT_8251_1
    MOV SI, PORT_8251_2   ; altfel, folosim baza PORT_8251_2
    JMP RECEPTIE_START    ; trecem la receptie
```

```
BASE_1_RECEPTIE:
    MOV SI, PORT_8251_1   ; folosim baza PORT_8251_1

RECEPTIE_START:
    ; Verificam RxRDY (bitul 2 din cuvantul de stare)
    MOV DX, SI            ; Adresa pentru date
    ADD DX, 2H            ; Adresa pentru comenzi/stari (04D2H sau 05D2H)

RX_WAIT:
    IN AL, DX             ; Citim cuvantul de stare
    RCR AL, 2             ; Testam bitul RxRDY
    JNC RX_WAIT           ; Daca RxRDY = 0, asteptam

    ; Citim caracterul receptionat
    MOV DX, SI            ; Adresa pentru date
    IN AL, DX             ; Citim caracterul
    MOV CL, AL            ; Stocam caraccterul in CL
    RET

RECEIVE_CHAR ENDP
;-----------------------------------------
; Exemplu de utilizare a subrutinei RECEPTIE_CARACTER_SERIALA
;-----------------------------------------
MOV BL, 0                     ; baza care o sa fie folosita
CALL RECEPTIE_CARACTER_SERIALA     ; apelam subrutina
; CL contine caracterul receptionat
```

Rutinele de emisie caracter pe interfața paralela

```
; Ne definim adresele porturilor ai 8255
PORT_8255_1 EQU 0250H
PORT_8255_2 EQU 0A50H
;-----------------------------------------
; Subrutina pentru emisie caracter paralela
; (ne focalizam pe afisajul LCD din proiect)
; Input: CL = caracterul de transmis
;        BL = 0 pentru S2 = 0, BL = 1 pentru S2 = 1
; (S2 este conmutatorului care selecteaza baza)
;-----------------------------------------
EMISIE_CARACTER_PARALELA PROC
```

```
    ; Pasul 1: calculam baza
    CMP BL,0            ; verificam intervalul
    JE  BASE_1          ; daca BL = 0, folosim baza PORT_8255_1
    MOV SI, PORT_8255_2   ; altfel, folosim baza PORT_8255_2
    JMP EMISIE_START      ; trecem la emisie

BASE_1:
    MOV SI, PORT_8255_1   ; folosim baza PORT_8255_1

EMISIE_START:
    ; Pasul 2: setam RS si RW pe portul C
    MOV DX, SI          ; Baza setata
    ADD DX, 04H          ; adunam cu 4 ca sa ajungem la adresa portului C
    MOV AL, 01H          ; setam RS=1 (caracter) si RW=0(scriere)
    OUT DX, AL          ; scriem pe portul C (semnalele de control pentru LCD)

    ; Pasul 3: trimitem caracterul pe portul A
    MOV DX, SI          ; Baza setata
    MOV AL, CL          ; caracterul de transmis
    OUT DX, AL          ; scriem pe portul A (caracterul de transmis)

    ; Pasul 4: generam semnalul de enable
    MOV DX, SI          ; Baza setata
    ADD DX, 04H          ; adunam cu 4 ca sa ajungem la adresa portului C
    OR AL, 04H          ; setam  bitul de eneable, E=1
    OUT DX, AL          ; scriem pe portul C semnalul de enable

    CALL DELAY_SHORT      ; introducem un delay scurt pentru a permite citirea informatiilor de catre LCD

    ; Pasul 5: resetam semnalul de enable
    AND AL, 01H          ; resetam bitul de enable, E=0
    OUT DX, AL          ; actualizam portul C

    RET
EMISIE_CARACTER_PARALELA ENDP
;-----------------------------------------
;Subrutina delay scurt
;-----------------------------------------
DELAY_SHORT PROC
    MOV CX, 0FFFH     ; setam un contor mare
```

```
DELAY_LOOP:
   LOOP DELAY_LOOP   ; decrementam contorul pana cand ajunge la 0
   RET
DELAY_SHORT ENDP
;-------------------------------------------
; Exemplu de utilizare a subrutinei EMISIE_CARACTER_PARALELA
;-------------------------------------------
MOV CL, 'A'                    ; caracterul de transmis
MOV BL, 0                      ; baza care o sa fie folosita
CALL EMISIE_CARACTER_PARALELA      ; apelam subrutina

; Explicatie semnal Enable:
; LCD-ul citeste informatiile (date sau comenzi)) doar cand detecteaza o
; tranzitie de la 0 la 1 pe semnalul de enable. Dupa aceasta tranzitie
; lcd-ul citeste informatiile si le proceseaza. Lasam un scurt delay
; intre generarea enable si setarea enable la 0, pentru a permite lcd-ului
; sa citeasca informatiile. Dupa care setam enable pe 0
; In alte dizpozitive se poate trasnmite dupaia inca un semnal de enable de reconfirmare a semnalului
; de enable, pentru a evita erorile de transmisie.
```

Rutina de scanare a minitastaturi

```
; Ne definim adresele porturilor pentru minitastaura
TAST_PORT1 EQU 0328H ; Adresa portului de iesire al tastaturii
TAST_PORT2 EQU 032CH ; Adresa portului de intrare al tastaturii
;-------------------------------------------
; Subrutina pentru scanarea minitastaturii
;-------------------------------------------
TAST_SCAN proc

   ; punem pe 0 prima coloana si verificam daca sau actionat tastele 1,4 sau 7
   MOV AL, 0FEh          ; activam prima coloana, adica punem pe 0 bitul 0 pentru a curge curentul (1111 1110)
   OUT TAST_PORT1, AL     ; scriem in portul de iesire al tastaturii
   IN AL, TAST_PORT2      ; citim din portul de intrare al tastaturii
   AND AL, 01H            ; verificam linea 1, daca sa apasat tasta 1 (bitul 0)
   JZ TASTA1              ; daca bitul 0 este 0, tasta 1 a fost apasata
   IN AL, TAST_PORT2      ; daca nu, citim din nou portul de intrare al tastaturii
   AND AL, 02H            ; verificam linea 2, daca sa apasat tasta 4 (bitul 1)
   JZ TASTA4              ; daca bitul 1 este 0, tasta 4 a fost apasata
   IN AL, TAST_PORT2
```

```
    AND AL, 04H             ; verificam linea 3, daca sa apasat tasta 7 (bitul 2)
    JZ TASTA7               ; daca bitul 2 este 0, tasta 7 a fost apasata

    ; punem pe 0 a doua coloana si verificam daca sau actionat tastele 2,5 sau 8
    MOV AL, 0FDh            ; activam a doua coloana, adica punem pe 0 bitul 1 pentru a curge curentul (1111 1101)
    OUT TAST_PORT1, AL      ; scriem in portul de iesire al tastaturii
    IN AL, TAST_PORT2       ; citim din portul de intrare al tastaturii
    AND AL, 01H             ; verificam linea 1, daca sa apasat tasta 2 (bitul 0)
    JZ TASTA2               ; daca bitul 0 este 0, tasta 2 a fost apasata
    IN AL, TAST_PORT2
    AND AL, 02H             ; verificam linea 2, daca sa apasat tasta 5 (bitul 1)
    JZ TASTA5               ; daca bitul 1 este 0, tasta 5 a fost apasata
    IN AL, TAST_PORT2
    AND AL, 04H             ; verificam linea 3, daca sa apasat tasta 8 (bitul 2)
    JZ TASTA8               ; daca bitul 2 este 0, tasta 8 a fost apasata

    ; punem pe 0 a treia coloana si verificam daca sau actionat tastele 3,6 sau 9
    MOV AL, 0FBh            ; activam a treia coloana, adica punem pe 0 bitul 2 pentru a curge curentul (1111 1011)
    OUT TAST_PORT1, AL      ; scriem in portul de iesire al tastaturii
    IN AL, TAST_PORT2       ; citim din portul de intrare al tastaturii
    AND AL, 01H             ; verificam linea 1, daca sa apasat tasta 3 (bitul 0)
    JZ TASTA3               ; daca bitul 0 este 0, tasta 3 a fost apasata
    IN AL, TAST_PORT2
    AND AL, 02H             ; verificam linea 2, daca sa apasat tasta 6 (bitul 1)
    JZ TASTA6               ; daca bitul 1 este 0, tasta 6 a fost apasata
    IN AL, TAST_PORT2
    AND AL, 04H             ; verificam linea 3, daca sa apasat tasta 9 (bitul 2)
    JZ TASTA9               ; daca bitul 2 este 0, tasta 9 a fost apasata

    JMP TAST_SCAN           ; daca nu s-a apasat nicio tasta, repetam procesul

;  Subrutine pentru tratarea fiecare taste apasata
TASTA1:
    CALL DELAY              ; asteapta stabilizarea contactelor
AST1:
    IN AL, TAST_PORT2       ; citim din nou starea liniilor
    AND AL, 01H             ; verificam daca tasta 1 este inca apasata
    JZ AST1                 ; daca da, continuam asteptarea
    CALL DELAY              ; asteptam eliberarea tastei
```

```
    ; aici se va executa codul pentru tasta 1
    JMP TAST_SCAN          ; dupa ce am terminat, revenim la scanarea tastaturii

TASTA2:
    CALL DELAY
AST2:
    IN AL, TAST_PORT2
    AND AL, 01H
    JZ AST2
    CALL DELAY

    ; aici se va executa codul pentru tasta 2
    JMP TAST_SCAN

TASTA3:
    CALL DELAY
AST3:
    IN AL, TAST_PORT2
    AND AL, 01H
    JZ AST3
    CALL DELAY

    ; aici se va executa codul pentru tasta 3
    JMP TAST_SCAN

TASTA4:
    CALL DELAY
AST4:
    IN AL, TAST_PORT2
    AND AL, 02H
    JZ AST4
    CALL DELAY

    ; aici se va executa codul pentru tasta 4
    JMP TAST_SCAN

TASTA5:
    CALL DELAY
AST5:
    IN AL, TAST_PORT2
```

```
    AND AL, 02H
    JZ AST5
    CALL DELAY

    ; aici se va executa codul pentru tasta 5
    JMP TAST_SCAN

TASTA6:
    CALL DELAY
AST6:
    IN AL, TAST_PORT2
    AND AL, 02H
    JZ AST6
    CALL DELAY

    ; aici se va executa codul pentru tasta 6
    JMP TAST_SCAN

TASTA7:
    CALL DELAY
AST7:
    IN AL, TAST_PORT2
    AND AL, 03H
    JZ AST7
    CALL DELAY

    ; aici se va executa codul pentru tasta 7
    JMP TAST_SCAN

TASTA8:
    CALL DELAY
AST8:
    IN AL, TAST_PORT2
    AND AL, 03H
    JZ AST8
    CALL DELAY

    ; aici se va executa codul pentru tasta 8
    JMP TAST_SCAN
```

```
TASTA9:
    CALL DELAY
AST9:
    IN AL, TAST_PORT2
    AND AL, 03H
    JZ AST9
    CALL DELAY

    ; aici se va executa codul pentru tasta 9
    JMP TAST_SCAN

TAST_SCAN endp
;-------------------------------------------
;Subrutina delay (Stabilizare contacte)
;-------------------------------------------
DELAY PROC
    MOV CX, 0FFFFH    ; setam un contor mare
DELAY_LOOP:
    LOOP DELAY_LOOP   ; decrementam contorul pana cand ajunge la 0
    RET
DELAY ENDP
```

Rutina de aprindere/stingere a unui led

```
; Ne definim adresele portului pentru leduri
LED_PORT1 EQU 0300h   ; Adresa portului de control al ledurilor 0-7
LED_PORT2 EQU 0304h   ; Adresa portului de control al ledurilor 8-9

; Definim si ledurile in hexa
LED1 EQU 0FEh
LED2 EQU 0FDh
LED3 EQU 0FBh
LED4 EQU 0F7h
LED5 EQU 0EFh
LED6 EQU 0DFh
LED7 EQU 0BFh
LED8 EQU 07Fh
LED9 EQU 002h
LED10 EQU 001h
;-------------------------------------------
```

```
; Subrutina pentru aprinderea uni led
; Input: AL = numarul ledului (0-9)
; Adica inainte se alege ce led vrem sa aprindem, numarul 8 in cazul nostru.(dam valoarea noi)
; MOV AL, 8
;-----------------------------------------
LED_ON proc
    CMP AL, 8          ; Verificam daca ledul este in portul 2
    JAE PORT_2_ON      ; Daca da vom folosi portul 2

    ; daca nu, folosim portul 1
    MOV DX, LED_PORT1  ; selectam portul 1
    MOV AL, LED8       ; aprindem ledul (in cazul nostru ledul 8)
    OUT DX, AL         ; trimitem valoarea in port
    RET                ; iesim din subrutina

    ; pentru al doilea port
PORT_2_ON:
    MOV DX, LED_PORT2
    MOV AL, LED9       ; aici nu prea conteaza ce led aprindem (pentru ca nu se va aprinde in acest caz)
    OUT DX, AL
    RET
LED_ON endp
;-----------------------------------------
; Exemplu de utilizare a subrutinei LED_ON
;-----------------------------------------
MOV AL, 8            ; setam AL = 8 pentru a aprinde ledul 8
CALL LED_ON          ; apelam subrutina
;-----------------------------------------
; Subrutina pentru stingerea uni led
; In cazul asta vom aprinde ledul 8 (dam noi valoarea)
; Input: AL = numarul ledului (0-9)
;-----------------------------------------
LED_OFF proc
    CMP AL, 8          ; Verificam daca ledul este in portul 2
    JAE PORT_2_OFF     ; Daca da vom folosi portul 2

    ; daca nu, folosim portul 1
    MOV DX, LED_PORT1  ; selectam portul 1
    MOV AL, 0          ; stingem ledurile
    OUT DX, AL         ; trimitem valoarea in port
```

```
    RET             ; iesim din subrutina

    ; pentru al doilea port
PORT_2_OFF:
    MOV DX, LED_PORT2  ; selectam portul 2
    MOV AL, 0
    OUT DX, AL
    RET
LED_OFF endp
;-------------------------------------------
; Exemplu de utilizare a subrutinei LED_OFF
;-------------------------------------------
MOV AL, 8           ; setam AL = 8 pentru a stinge ledul 8
CALL LED_OFF        ; apelam subrutina
```

Rutina de afișare a unui caracter hexa pe un rang cu segmente

```
; Ne definim adresele porturilor pentru segmente
SEG_PORT1 EQU 0308h   ; adresa primului port pentru segmente
SEG_PORT2 EQU 030Ch   ; adresa celui de-al doilea port pentru segmente
SEG_PORT3 EQU 0310h   ; adresa celui de-al treilea port pentru segmente
SEG_PORT4 EQU 0314h   ; adresa celui de-al patrulea port pentru segmente
SEG_PORT5 EQU 0318h   ; adresa celui de-al cincilea port pentru segmente
SEG_PORT6 EQU 031Ch   ; adresa celui de-al saselea port pentru segmente
SEG_PORT7 EQU 0320h   ; adresa celui de-al saptelea port pentru segmente
SEG_PORT8 EQU 0324h   ; adresa celui de-al optulea port pentru segmente

; Definim si caracterele in hexa
HEX_0 EQU C0h
HEX_1 EQU F9h
HEX_2 EQU A4h
HEX_3 EQU B0h
HEX_4 EQU 99h
HEX_5 EQU 92h
HEX_6 EQU 82h
HEX_7 EQU F8h
HEX_8 EQU 80h
HEX_9 EQU 98h
HEX_A EQU 88h
HEX_B EQU 83h
```

```
HEX_C EQU C6h
HEX_D EQU A1h
HEX_E EQU 86h
HEX_F EQU 8Eh
;-------------------------------------------
; Subrutina pentru afisarea unui segment
; Input: BL = numarul segmentului (1-8)
; AL = caracterul hexazecimal (0-9, A-F) - in cazul nostru vom alege numarul 4
;-------------------------------------------
SEG_ON proc
    CMP BL, 1         ; Verificam daca segmentul este in portul 1
    JAE PORT_1_ON     ; Daca da vom folosi portul 1

    CMP BL, 2         ; Verificam daca segmentul este in portul 2
    JAE PORT_2_ON     ; Daca da vom folosi portul 2

    CMP BL, 3         ; Verificam daca segmentul este in portul 3
    JAE PORT_3_ON     ; Daca da vom folosi portul 3

    CMP BL, 4         ; Verificam daca segmentul este in portul 4
    JAE PORT_4_ON     ; Daca da vom folosi portul 4

    CMP BL, 5         ; Verificam daca segmentul este in portul 5
    JAE PORT_5_ON     ; Daca da vom folosi portul 5

    CMP BL, 6         ; Verificam daca segmentul este in portul 6
    JAE PORT_6_ON     ; Daca da vom folosi portul 6

    CMP BL, 7         ; Verificam daca segmentul este in portul 7
    JAE PORT_7_ON     ; Daca da vom folosi portul 7

    ; daca nu, folosim portul 8
    MOV DX, SEG_PORT8  ; selectam portul 8
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET                ; iesim din subrutina

    ; pentru primul port
PORT_1_ON:
    MOV DX, SEG_PORT1  ; selectam portul 1
```

```
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

    ; pentru al doilea port
PORT_2_ON:
    MOV DX, SEG_PORT2  ; selectam portul 2
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

    ; pentru al treilea port
PORT_3_ON:
    MOV DX, SEG_PORT3  ; selectam portul 3
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

    ; pentru al patrulea port
PORT_4_ON:
    MOV DX, SEG_PORT4  ; selectam portul 4
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

    ; pentru al cincilea port
PORT_5_ON:
    MOV DX, SEG_PORT5  ; selectam portul 5
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

    ; pentru al saselea port
PORT_6_ON:
    MOV DX, SEG_PORT6  ; selectam portul 6
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

    ; pentru al saptelea port
```

```
PORT_7_ON:
    MOV DX, SEG_PORT7  ; selectam portul 7
    MOV AL, HEX_4      ; afisam caracterul 4
    OUT DX, AL         ; trimitem valoarea in port
    RET

SEG_ON endp
;-----------------------------------------
; Exemplu de utilizare a subrutinei SEG_ON
;-----------------------------------------
MOV BL, 4            ; setam BL = 4 pentru a aprinde segmentul 4
CALL SEG_ON          ; apelam subrutina
```

Bibliografia

1. Mircea Popa, ***"Sisteme cu microprocesoare"***, Editura Orizonturi Universitare, 2003. (Referință principală pentru detalii despre microprocesorul 8086 și structura unui micro-sistem.
2. D.V. Hall, **"Microprocessors and Interfacing: Programming and Hardware"**, McGraw-Hill, 1992
3. **27C512 Datasheet**, STMicroelectronics, https://www.alldatasheet.com/datasheet-pdf/pdf/23533/STMICROELECTRONICS/27C512.html (accesat pe 10 noiembrie 2024)
4. **Datasheet-ul 8255A (Programmable Peripheral Interface)**, Intel, https://www.alldatasheet.com/datasheet-pdf/pdf/66100/INTEL/8255A.html (accesat pe 16 noiembrie 2024)
5. **Datasheet-ul 8251A (Programmable Communication Interface)**, Intel, https://www.alldatasheet.com/datasheet-pdf/pdf/66096/INTEL/8251A.html (accesat pe 16 noiembrie 2024)
6. **74LS244 Datasheet**, Texas Instruments, https://www.alldatasheet.com/datasheet-pdf/pdf/28030/TI/74LS244.html (accesat pe 30 decembrie 2024)
7. **74LS373 Datasheet**, Texas Instruments, https://www.alldatasheet.com/datasheet-pdf/pdf/28029/TI/74LS373.html (accesat pe 30 decembrie 2024)
8. **"LCD interfacing"**, Technobyte, https://technobyte.org/lcd-interfacing-8051-8-bit-4-bit-8255/ (accesat pe 31 decembrie 2024).
9. **MAX232 Datasheet,** Maxim Integrated Products, https://www.alldatasheet.com/datasheet-pdf/view/73074/MAXIM/MAX232.html (accesat pe 1 ianuarie 2025).
10. Proiectul didactic și materialele suplimentare primite de la profesor. (Documentația de referință pentru cerințele tehnice și descrierea proiectului.)