

L11. PREZENTAREA SISTEMULUI DE DEZVOLTARE ȘI ARHITECTURII GENERALE A MICROPROCESORULUI 8085. INTRODUCEREA UNUI PROGRAM DE COPIERE DIRECTĂ / INVERSATĂ ȘI SINGULARĂ / ÎN BUCLĂ A UNUI PORT DE INTRARE PE UN PORT DE IEȘIRE DE LA TASTATURA LOCALĂ, CU INTERPRETAREA CODURILOR.

1. Obiective

Prin parcurgerea acestei ședințe de laborator studenții vor fi capabili:

- Să învețe structura internă și modul de funcționare al microprocesorului 8085.
- Să enumere resursele hardware ale microsistemului;
- Să explice modul de lucru al monitorului rezident în memoria ROM;
- Să utilizeze facilitățile de operare locală ale microsistemului;
- Să folosească facilitățile de operare de la distanță.

2. Arhitectura microprocesorului 8085

Principalele blocuri interne ale microprocesorului 8085 sunt prezentate în Figura 1:

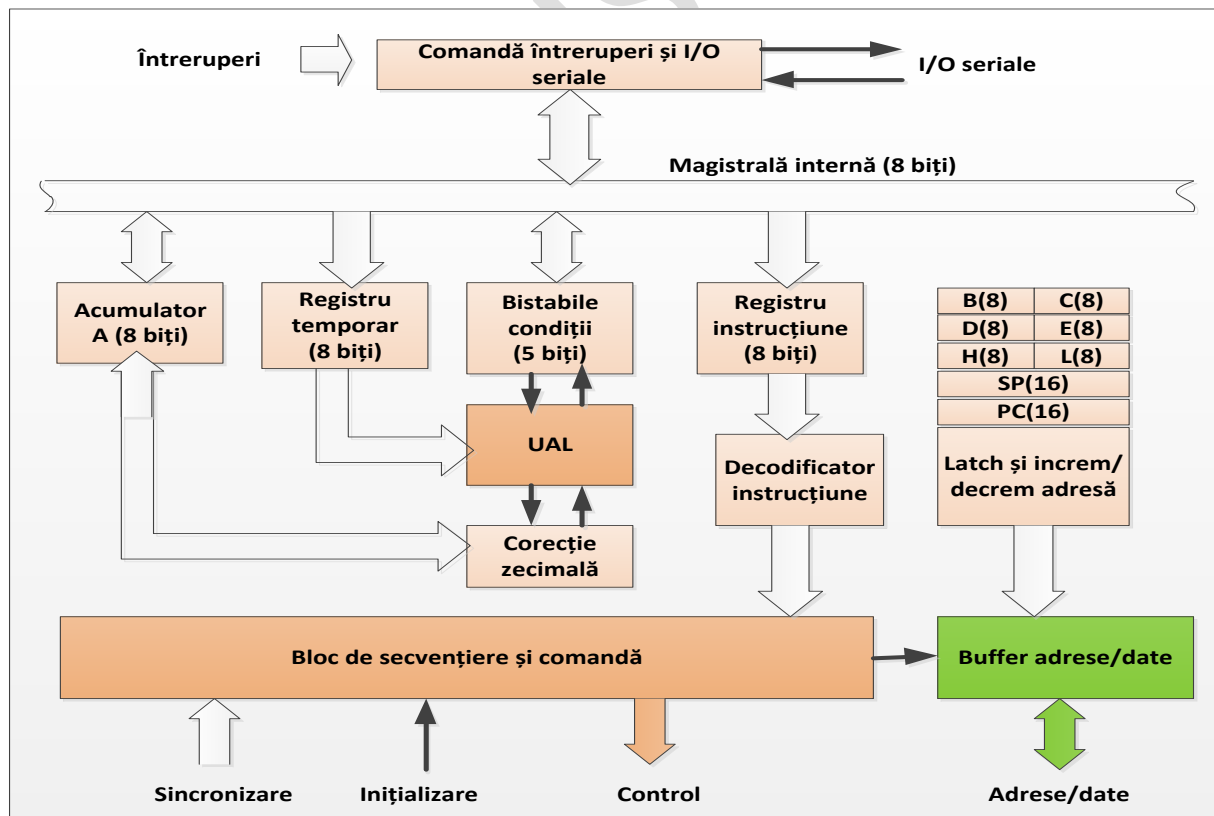


Figura 1. Schema bloc internă simplificată a microprocesorului 8085

3. Unitatea aritmetico-logică

Unitatea aritmetico-logică execută operațiile aritmetice și logice între maxim doi operanzi întregi fără semn reprezentați pe 8 biți. În mod obligatoriu unul dintre operanzi trebuie să fie în registrul A, iar celălalt, dacă există, într-un registru temporar. Registrul A va stoca la final rezultatul operației și de aceea este denumit și registru **acumulator**.

Modul de desfășurare a operației și caracteristicile rezultatului sunt memorate într-un set de bistabile denumiți indicatori de condiții (flags):

- **Z** – zero (rezultat zero al operației aritmetice sau logice);
- **S** – sign (semnul rezultatului, cel mai semnificativ bit al acumulatorului);
- **P** – parity (rezultatul are un număr par/impar de biți 1);
- **CY** – carry;
- **AC** - auxiliary carry (transport/împrumut de la / spre bitul 7, respectiv bitul 3 al rezultatului ultimei operații aritmetice).

Starea acestor indicatori poate influența modul de desfășurare și rezultatul unor instrucțiuni aritmetice și de ramificare condiționată.

Acumulatorul, împreună cu flagurile formează cuvântul de stare al procesorului (PSW – processor status word).

$$PSW = A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 S Z AC P CY$$

4. Registrele interne și magistralele externe

4.1. Registrele interne

Registrele generale B, C, D, E, H și L sunt 6 registre de lucru de 8 biți. Acestea pot fi folosite pentru memorarea datelor și rezultatelor intermediare ale programului. Sunt accesibile programului prin instrucțiuni adecvate, atât ca registre simple de 8 biți, cât și în perechi, ca registre extinse de 16 biți:

- B și C (sau registrul extins B);
- D și E (sau registrul extins D);
- H și L (sau registrul extins H).

Registrul instrucțiunii primește opcodul instrucțiunii curente și îl menține pe întreaga durată de execuție a instrucțiunii.

Registrele speciale au 16 biți și îndeplinesc anumite funcții ale microprocesorului. Astfel, **numărătorul de instrucțiuni (PC - Program Counter)** conține în fiecare moment adresa instrucțiunii ce urmează a fi executată. Ele este **inițializat cu 0 la resetarea microprocesorului** și este implicit incrementat pentru secvențe liniare de instrucțiuni, respectiv este modificat direct de către instrucțiunile de ramificare.

4.2. Magistralele externe

Magistrala sistemului (MS) este magistrala la care se conectează toate celelalte componente pentru a schimba informații între ele. Aceasta este formată din:

- **Magistrala de adrese (MA)** – grupul de linii unidirecționale, prin care coordonatorul sistemului indică locația de memorie sau portul de intrare/ieșire la care dorește să aibă acces;
- **Magistrala de date (MD)** – grupul de linii bidirecționale, pe care circulă informația (instrucțiuni și date) între componentele sistemului;
- **Magistrala de control (MC)** – liniile uni sau bidirecționale cu rolul de a:
 - Stabili tipul de acces pe magistrala sistemului:
 - Selecția spațiului de adresare (memorie sau porturi I/E);
 - Controlul sensului de transfer pe magistrala de date (citire sau scriere);
 - Sincroniza microprocesorul cu dispozitivele externe mai lente (wait);
 - Realiza dialogul dintre microprocesor și dispozitivul I/E care solicită întreruperi;
 - Realiza dialogul dintre microprocesor și un alt controller care dorește accesul la magistrala sistemului (DMA Direct Memory Access).

Magistrala sistemului este controlată în mod implicit de către microprocesor, care depune adresa pe magistrala de adrese și activează liniile de control ale magistralei de control, în funcție de operația pe care o execută la un moment dat.

5. Modul de execuție a instrucțiunilor

Pentru a realiza **execuția** unei instrucțiuni microprocesorul:

- extrage adresa instrucțiunii de pe magistrala de adrese;
- activează semnalele de citire a memoriei;
- citește codul instrucțiunii din memorie pe magistrala de date;
- decodifică instrucțiunea
- execuția propriu-zisă a instrucțiunii constituită din:
 - operații interne: aritmetice, logice, de control a stării microprocesorului;
 - operații pe magistrală: transfer de operanzi pe magistrala de date.

Pentru **citirea** unui operand microprocesorul:

- depune adresa operandului pe magistrala de adrese;
- activează semnalele de citire memorie sau dispozitivul I/E;
- citește operandul din memorie pe magistrala de date.

Pentru **scrierea** unui operand microprocesorul:

- depune adresa operandului pe magistrala de adrese;
- depune operatorul pe magistrala de date.

6. Placa de dezvoltare (EMAC Universal Trainer)

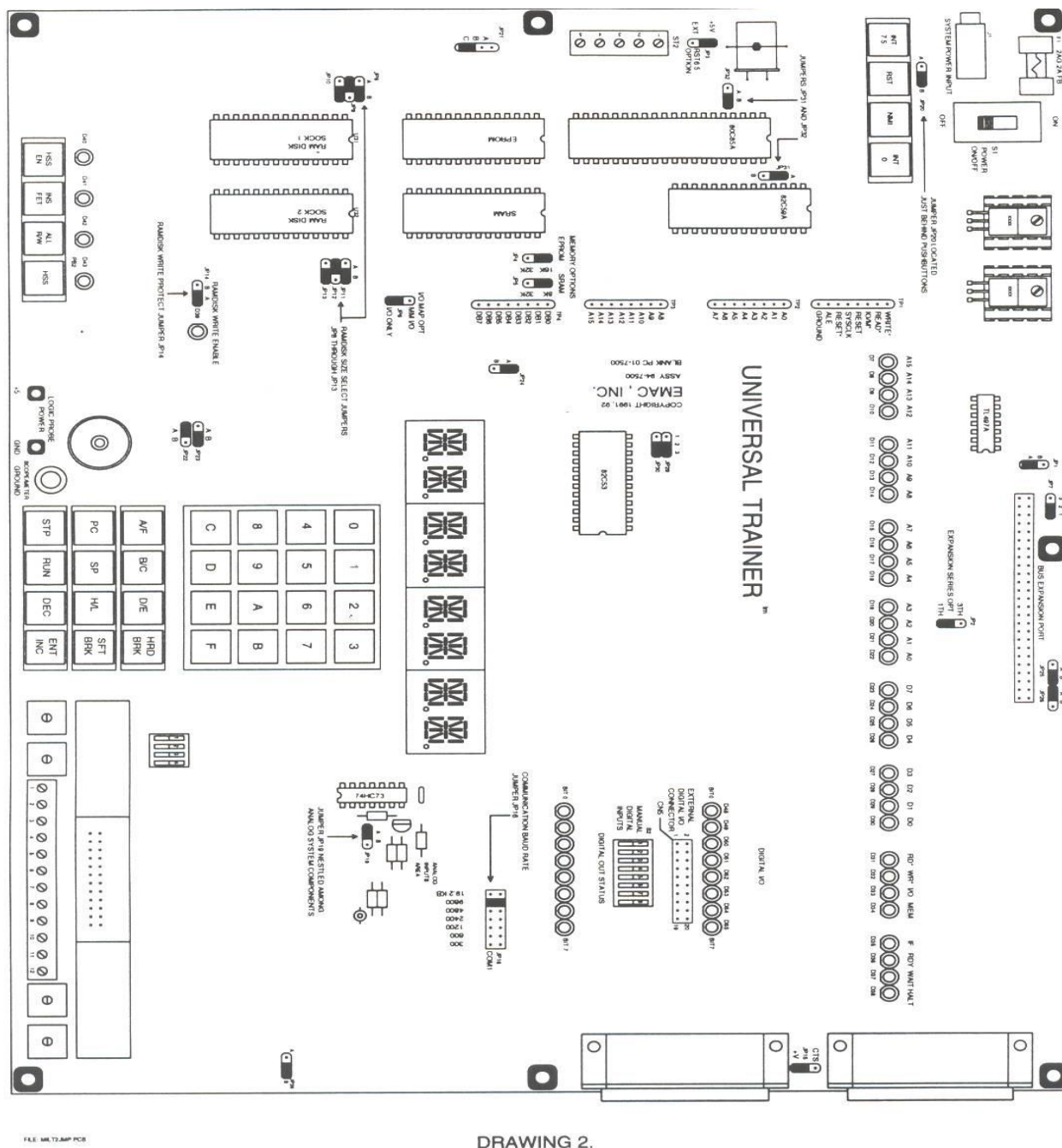


Figura 2. Schema plăcii de dezvoltare

Resursele hardware ale microsistemului

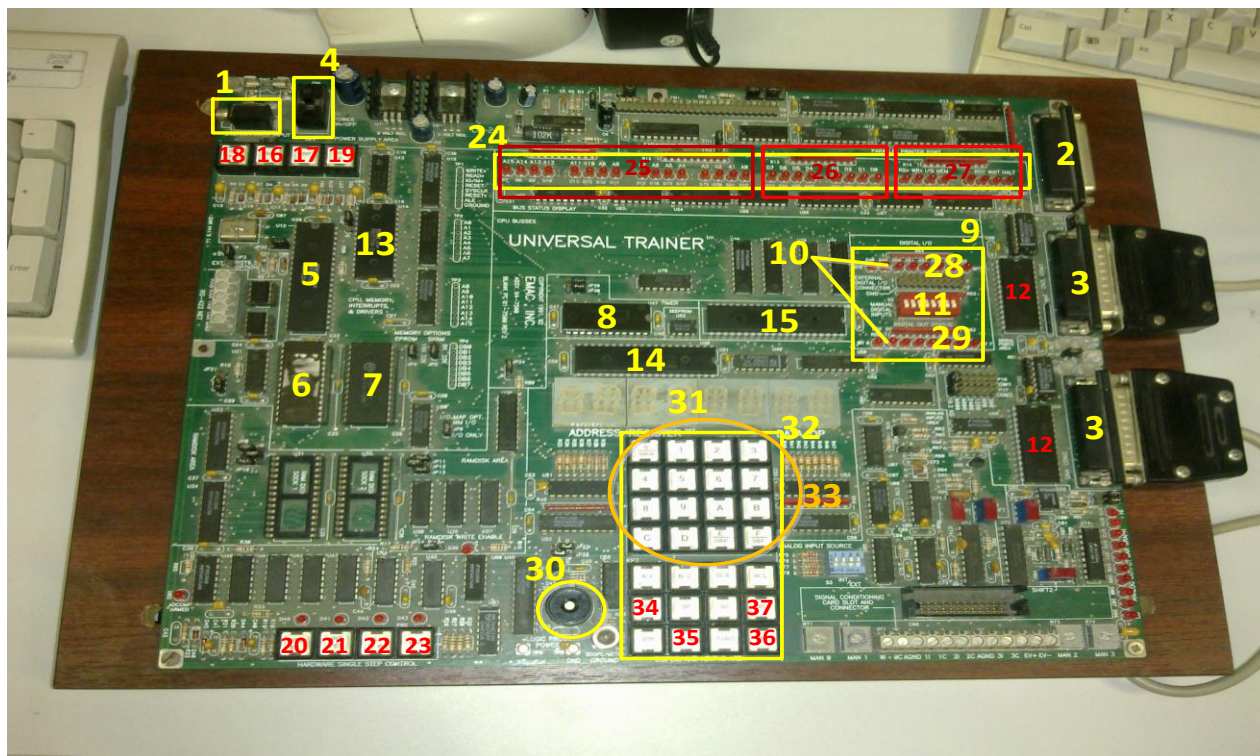


Figura 3. Principalele componente ale microsistemului

Legendă:

- **Mufă de alimentare** - permite alimentarea de la o sursă de tensiune externă de cel puțin 7V c.c.; (1)
- **Mufă port paralel** (2)
- **Mufe port serial** (3)
- **Comutator bipozițional** - de conectare (ON) / deconectare (OFF) a alimentării (4)
- **Microprocesorul** - 80C85; (5)
- **Memoria ROM** - de 32 Kocteți, la începutul căreia se află înscris monitorul rezident ce permite operarea locală și de la distanță;
- locațiile de memorie RAM 8000h÷FF00h sunt la dispoziția utilizatorilor, pentru încărcarea și execuția programelor acestora; (6)
- **zona de memorie RAM FF00h÷FFFFh nu trebuie modificată;**
- **Memoria RAM** - de 32 Kocteți, din care ultimele 256 de locații (FF00h÷FFFFh) sunt folosite de către monitorul rezident; (7)
- **Circuitul programabil de** - cu 3 canale de timp, din care unul poate comanda un difuzor; (8)

 timp (Timer)			
• Interfața paralelă	- cu 3 porturi de 8 biți:		(9)
	A - comandă un set de 8 led-uri de stare (D56÷D63) (40H);		(10)
	B - conectat la 8 comutatoare DIP (S2), a căror stare este afișată și pe leduri (D48÷D55) (41H);		(11)
• Circuitele de interfață serială	- pentru două canale de comunicație serială RS232: COM1 și COM2 (COM1 este folosit de monitorul rezident);		(12)
• Controlerul de întreruperi	- cu 8 linii de întrerupere, conectat la linia INTR a microprocesorului;		(13)
• Controler pentru afișaj și tastatură			(14)
• Controlerul pentru portul paralel			(15)
• Butoane de inițializare	RST	- re setează micro sistemul, astfel încât, indiferent în de starea sa, monitorul rezident repornește, similar cazului de conectare a alimentării;	(16)
	TRP	- determină apariția unei întreruperi nemascabile (care este acceptată necondiționat de microprocesor), însmenând încetarea activității curente a micro sistemului și reintrarea în bucla de așteptare de comenzi; - această comandă este utilă pentru a întrerupe în orice moment execuția unui program al utilizatorului, pentru a examina starea registrelor sau pentru a vedea unde s-a blocat;	(17)
	INT7.5	- permit generarea manuală a altor tipuri de întreruperi și rolul lor va fi examinat în detaliu într-o lucrare ulterioară.	(18)
	INT0		(19)
• Butoane și logică HW pentru execuția pas cu pas	HSS EN	Hardware Single Step Enable – Activare execuție pas cu pas	(20)
	INS FET	Se oprește după fiecare instrucțiune	(21)
	ALL R/W	Se oprește după fiecare cod	(22)
	HSS	Execuție pas cu pas	(23)
• Led-uri	D7-D38	- pentru vizualizarea stării magistralelor: D7-D22 pentru magistrala de adrese (A15-A0); - D23-D30 pentru magistrala de date (D7-D0); - D31-D38 pentru comenzi (R, W, etc.);	(24)
			(25)
			(26)
			(27)
	D48-D63	- pentru vizualizarea porturilor I/O: D48-D55 - pentru intrări; - D56-D63 - pentru ieșiri;	(28)
			(29)
• Difuzor			(30)
• Display alfa-numeric			(31)

• Tastatura		(32)
hexazec	0, 1,..., 9, A,...,F	(33)
PC	Program Counter	(34)
RUN	- lansează în execuție	(35)
ENT	- incrementează adresa	(36)
DEC	- decrementează adresa	(37)

Înainte de alimentarea cu tensiune și pornirea microsistemului, se verifică dacă toți jumperii se află în pozițiile corecte, așa cum se indică în figura xx. Nu se vor face niciodată modificări în configurația jumperilor în timp ce microsistemul se află sub tensiune.

Microsistemul are o construcție robustă, dar asta nu înseamnă că în anumite condiții nu poate fi deteriorat. El este special proiectat pentru a fi foarte deschis, astfel încât să se observe clar starea anumitor componente de semnalizare și aceasta să fie ușor de interpretat de către utilizator. De aceea, **se va avea grija ca pe placă să nu ajungă obiecte metalice sau fire conductoare neizolate.**

După introducerea mufei jack și acționarea comutatorului S1 (POWER) pe poziția ON, în câmpul ADRESA al afișajului apare adresa inițială din PC-ul utilizatorului local (**8001**), iar în câmpul DATA - valoarea din memorie de la această adresă. De asemenea, în difuzor se va auzi un șir de tonuri specifice, semn că microsistemul s-a inițializat cu succes și așteaptă comenzi.

7. Aplicații de vizualizare/editare a conținutului unor registre și locații de memorie pe placa de dezvoltare

7.1. Operarea remote

Modul de operare remote se realizează prin intermediul programului NoICE85, care permite încărcarea și depanarea aplicațiilor la distanță, la nivel de cod sursă, direct pe sistemul cu microprocesor. NoICE85 furnizează utilizatorului o interfață grafică, facilitând depanarea programelor. Pentru a putea stabili legătura cu monitorul rezident al microsistemului, NoICE85 trebuie configurat (din meniul principal **Options|Target Communications**) astfel:



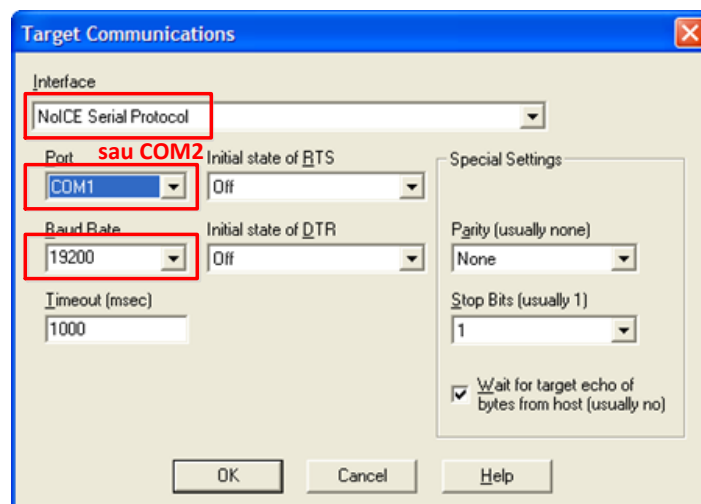


Figura 4. Configurarea comunicației cu programul NoICE85

În cazul în care a fost stabilită legătura cu monitorul rezident, pe ecran apare o fereastră similară cu cea din figura 5a. Dacă nu s-a reușit stabilirea legăturii dintre PC și EMAC Universal Trainer, pe ecran apare mesajul din figura 5b. În acest caz se verifică parametrii comunicației și cablul de legătură.

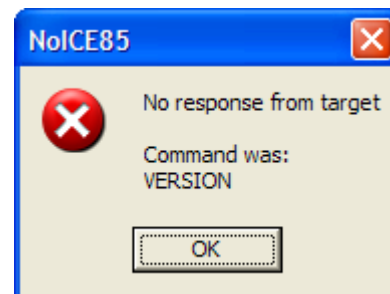
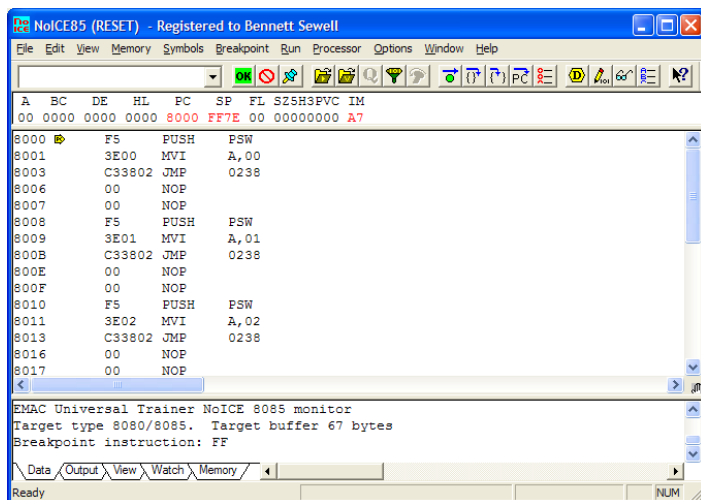


Figura 5. Legătura cu monitorul rezident

a) cu succes;

b) eșuată.

Comenzile transmise de NoICE85 pe legătura serială, care sunt primite și executate de monitor sunt comenzi simple, de forma:

- citire și înscriere locații de memorie;
- citire și înscriere registre ale utilizatorului;
- citire și înscriere porturi de I/E;
- lansare în execuție a programului de aplicație.

Cu ajutorul acestor comenzi, transparente pentru utilizator, sunt implementate facilitățile interfeței grafice a depanatorului NoICE51:

- încarcă în memoria microsistemului codul executabil al programului de aplicație (**File-Play|Load**);
- editează, copiază și caută informații în fișierul asociat ferestrei curente (**Edit**);
- vizualizează în fereastra principală a aplicației codul programului în format dezasamblat, sursă sau mixt (**View**);
- vizualizează/editează conținutul memoriei (**Memory-Dump|Edit**);
- assemblează instrucțiuni direct în memorie (**Memory-Assemble**);
- urmărește evoluția variabilelor programului (**Memory-Add watch**);
- definește, șterge și afișează simbolurile specifice programului de aplicație (**Symbols**);
- inserează, afișează și șterge punctele de oprire a execuției programului (**Breakpoint**);
- execută programul în regim pas cu pas sau automat, cu breakpoint (**Run**);
- vizualizează și modifică conținutul registrelor și al porturilor de I/E (**Processor**);

Interfața NoICE85

După alimentarea și inițializarea microsistemului, acesta așteaptă comenzi. Aceste comenzi pot veni nu numai de la operatorul local, prin tastatura și butoanele amplasate pe placă, ci și de la portul serial COM1 al microsistemului, care se conectează la un PC pe care se execută depanatorul NoICE85.

Vizualizarea și modificarea registrelor utilizatorului

Valorile inițiale ale registrelor utilizatorului de la distanță sunt afișate permanent pe o serie de butoane amplasate sub bara de butoane de comandă a NoICE85. Aceste valori pot fi modificate direct printr-un simplu click pe registrul respectiv.

Se observă faptul că indicatorii de condiții sunt afișați atât în format hexa cât și în binar (așa cum intră în componența PSW) și pot fi modificați de asemenea în ambele formate.

Conținutul memoriei începând de la adresa inițială din PC pentru programul utilizatorului (8000h) este afișat în fereastra principală a depanatorului, în format hexa și dezasamblat (operația inversă asamblării, prin care se determină instrucțiunile corespunzătoare octeților din memorie). Pentru a se baleia o zonă de memorie se pot folosi tastele Page Up-Page Dn sau bara de control pentru defilarea ferestrei pe verticală. Să se baleieze zona de memorie 8000h÷8100h.

Pentru a se vizualiza o zonă de memorie aflată la o distanță mai mare de cea curentă, se poate utiliza una din comenzile din meniu **View – Source at ...| Disassemble at ...**.

Se poate reveni în orice moment la adresa curentă din PC, cu comanda **View – Source at PC**.

Conținutul memoriei poate fi vizualizat și ca o zonă de date, în format hexa și ASCII în fereastra Data, cu comanda **Memory – Dump...** și în continuare cu tasta **F2**.

Conținutul memoriei poate fi modificat prin intermediul comenzii de meniu **Memory – Edit ...**, în care se poate stabili adresa locației, valoarea și se poate selecta tipul datei pentru afișare și modificare.

Dar depanatorul NoICE dispune de un mod mai avantajos de încărcare a programului. Astfel, folosind comanda **Memory – Assemble...** utilizatorul de la distanță poate introduce o secvență de program direct în limbaj de asamblare, instrucțiune cu instrucțiune. Operația de asamblare este efectuată automat de către NoICE.

Să se șteargă programul introdus anterior prin umplerea zonei de memorie de la A000h cu 0 pe o distanță de 16 locații cu comanda **Memory – Fill ...**, să se verifice acest lucru cu comanda **View – Source at PC**, apoi să se introducă din nou secvența de program de mai sus de data aceasta direct în limbaj de asamblare și să se verifice acest lucru cu o nouă comandă **View – Source at PC**.

Pornind de la adresa de început a secvenței de program (A000h), se va executa programul în regim pas cu pas, cu una din comenzile **Run – Step Over | Step Into | Step Instruction** sau cu un click pe butoanele de comandă asociate acestor comenzi sau cu tastele F7 | F8 | F9 care, în acest caz au același efect: execuția următoarei instrucțiuni din programul utilizatorului.

În acest regim, microprocesorul execută doar o singură instrucțiune din programul utilizatorului la distanță, după care controlul revine monitorului, care transmite către NoICE noile valori ale registrelor și așteaptă o nouă comandă.

Se execută programul în regim liber (free run) cu comanda **Run – Go** sau cu un click pe butonul corespunzător sau apăsând tasta F5. În acest regim, microprocesorul execută continuu numai instrucțiuni din programul utilizatorului. Acest program conține o buclă infinită, în care pe ledurile conectate la un port de ieșire este evidențiată starea comutatoarelor conectate la un port de intrare. Să se modifice starea comutatoarelor și să se urmărească schimbarea stării ledurilor din portul de ieșire.

Execuția liberă a programului utilizatorului local poate fi oprită numai prin activarea unei linii de întrerupere (tasta TRP) sau prin resetarea sistemului (tasta RST), când monitorul preia controlul, transmite către NoICE85 starea registrelor utilizatorului la distanță și așteaptă o nouă comandă.

Se va observa cum execuția programului utilizatorului se oprește într-un anumit punct din buclă, în funcție de momentul în care se apasă tasta TRP. În schimb, în cazul în care se resetează microsistemul, monitorul reface toate inițializările, iar PC-ul utilizatorului devine 8000h. Programul introdus se mai păstrează în memorie numai dacă oprirea s-a produs cu TRP. La resetare, zona de memorie care începe la A000h este inițializată de monitor, ceea ce determină distrugerea programului utilizatorului.

Pentru a evita reintroducerea programului prin una din cele două metode prezentate anterior, odată introdus, acesta poate fi salvat într-un fișier cu comanda **File – Save ...** în format binar (image memorie) sau Intel HEX (un format ASCII-hex introdus de Intel). După resetare, acest fișier poate fi reîncărcat în memoria microsistemului cu comanda **File – Load ...**, iar prezența programului în memorie poate fi observată cu o comandă **View – Source at PC**.

7.2. Etapele realizării și rulării unui program:

1. Crearea cu ajutorul unui editor de text (Ex: Notepad) a fișierului sursă. Fișierul sursă va avea extensia **.ASM** și va conține programul scris în limbaj de asamblare.
2. Asamblarea. Asamblarea este realizată cu ajutorul unui asamblor. Asamblorul folosit în cadrul ședințelor de laborator este ASM85.EXE. Asamblarea poate fi realizată în două moduri:
 - a. prin apelarea asamblorului în linia de comandă
`ASM.EXE nume_fisier.ASM`
 - b. prin apelarea automată a asamblorului prin efectuarea operației de tip “drag and drop” a fișierului sursă peste iconița corespunzătoare asamblorului.
 În urma operației de asamblare vor fi generate două fișiere:
 - **nume_fisier.LST** - conține erorile apărute pe parcursul asamblării și liniile unde au apărut aceste erori;
 - **nume_fisier.HEX** - reprezintă fișierul care va fi încărcat în memorie cu ajutorul programului Nolce.
3. Dacă nu au apărut erori pe parcursul operației de asamblare, se lansează NolCE85. După stabilirea legăturii cu microsistemul se încarcă în memoria micro-sistemului programul executabil în format Intel HEX cu comanda **File – Load ... – TEST.HEX**. Se verifică încărcarea lui în memorie cu comanda **View – Disassemble at ... – A000h**. Cu comanda **View – Mixed source /disassembly** se comută între două moduri de vizualizare a programului în fereastra principală: la liniile de cod sursă din fișierul sursă, se pot adăuga sau nu instrucțiunile dezasamblate din memoria microsistemului. Se observă că acestea coincid, ceea ce înseamnă că programul a fost corect asamblat și încărcat.
4. Se încarcă registrul PC cu adresa la care a fost încărcat programul.
5. Se lansează programul în execuție prin comanda **Run – Go (F5)**.

De aici încolo depanarea programului are loc beneficiind nu numai de codul dezasamblat, ci și de toate facilitățile unei depanări simbolice: afișarea liniilor de cod sursă, urmărirea evoluției variabilelor programului, modificarea acestora, inserarea de puncte de oprire a execuției la adrese simbolice (etichetele instrucțiunilor) etc.

8. Aplicații de conectare a plăcii la un calculator și de examinare a conținutului memoriei cu monitorul NolCE

Se consideră secvența de instrucțiuni introduse în memorie începând cu adresa A000.

Adresa	Cod mașină	Cod sursă
A001	DB 41	IN 41
A002	2F	CMA

A004	D3 40	OUT 40
A005	76	HLT

Se observă marcat cu roșu opcodul instrucțiunii în cod mașină. Acesta este urmat de o constantă care este fie o adresă fie o valoare numerică. La execuția secvenței de instrucțiuni registrul PC ia următoarele valori: A000, A001, A002, A003, A004 și A005.

9. Aplicații propuse

9.1. Să se încarce registrele de date ale utilizatorului cu următoarele valori:

- | | |
|----------------|---------------|
| a) $A = 12h$; | g) $L = 0$; |
| b) $B = 4Dh$; | h) $CY = 1$; |
| c) $C = 01h$; | i) $Z = 1$. |
| d) $D = 9Fh$; | |
| e) $E = F6h$; | |
| f) $H = 4$; | |

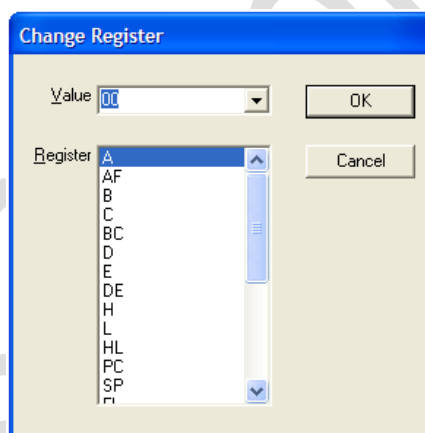
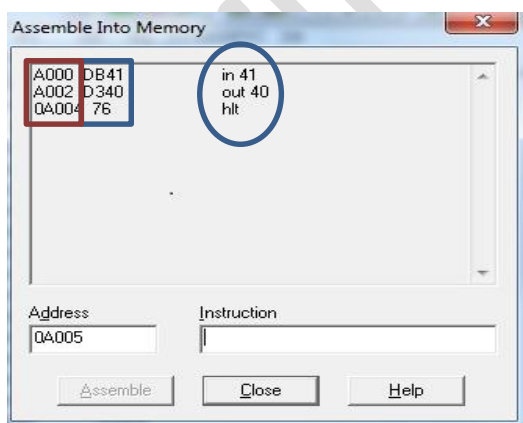


Figura 6. Modificarea regiștrilor

9.2. Să se încarce în memorie folosind comanda **Memory -> Assemble** începând de la adresa A000 următoarea secvență de instrucțiuni:



IN 41
OUT 40
HLT

Figura 7. Încărcarea instrucțiunilor în memorie cu Memory -> Assemble

Vizualizați de pe placă modul în care s-a modificat conținutul adreselor A000 – A004. Ce observați? Să se noteze efectul secvenței de cod la execuție (Atenție! Se va starta execuția începând cu adresa A000). Ce se întâmplă cu ledurile (28) și (29)?

- 9.3. Să se creeze pe desktop un folder cu numele grupei **110xA/B**. În folderul respectiv să se copie executabilul **ASM85.exe**. Să se creeze un document text nou în care să se scrie următoarele instrucțiuni precizate în figura 8. Să se salveze documentul respectiv cu Save as, All files, cu numele **leduri.asm**. Apoi, folosind “drag and drop” să se execute fișierul **leduri.asm** în **ASM85.exe** similar ca în figura 9.

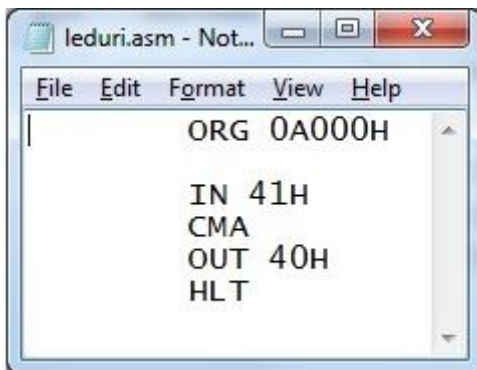


Figura 8. Fișierul leduri.asm

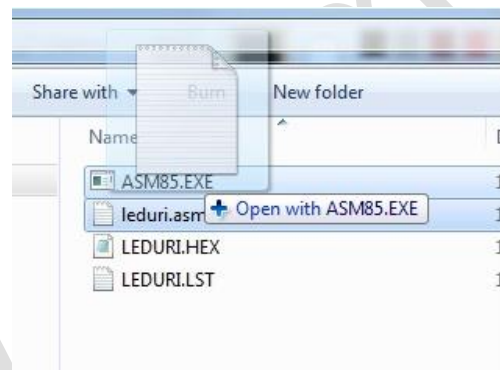


Figura 9. Execuția

Se vor genera două noi fișiere ce pot fi observate în figura 9. Fișierul **leduri.lst** va conține lista cu erorile apărute la execuție. Dacă fișierul respectiv este gol ca în figura 10, atunci nu există nici o eroare. Fișierul **leduri.hex** va putea fi încărcat în memorie.

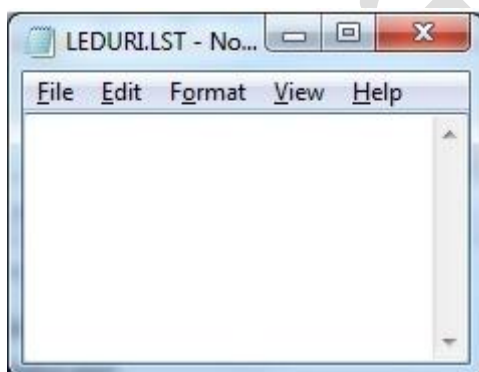


Figura 10. Fișierul ce conține erorile

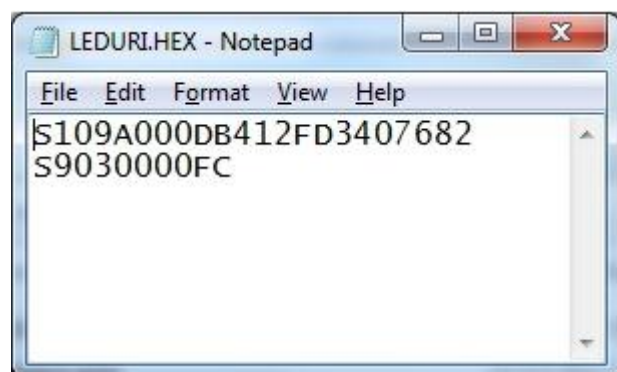


Figura 11. Fișierul leduri.hex generat

Să se încarce în memorie folosind comanda File -> Load fișierul leduri.hex. Să se vizualizeze conținutul de la adresa A000 similar cu figura 12.

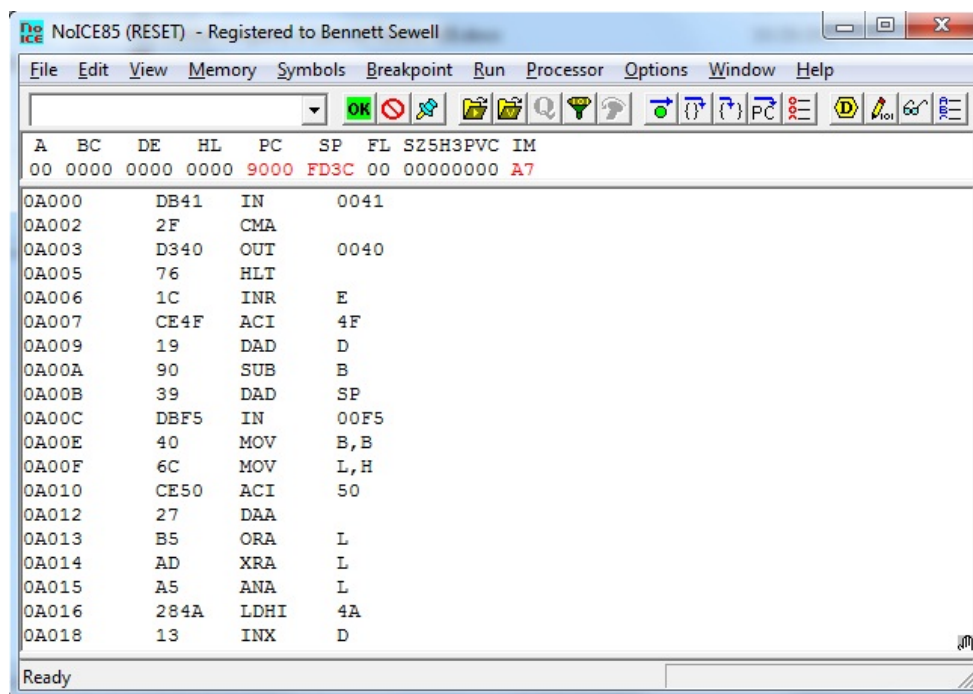


Figura 12. Conținutul memoriei începând cu locația A000

Să se lanseze în execuție programul.

- 9.4. Să se încarce în memorie folosind tastatura (33), începând de la adresa A000h, următoarea secvență de octeți:

Adresa	Cod mașină
A000	DB
A001	41
A002	D3
A003	40
A004	76

Să se starteze execuția de la adresa A000:

- De pe placa folosind butoanele de pe tastatura (33);
- Din NoICE85 folosind comanda **Run -> Go From : A000**.

- 9.5. Să se actualizeze fereastra de vizualizare a programului de la adresa din PC (A000h) cu comanda **View – Source at PC** și să se verifice dacă instrucțiunile introduse coincid cu cele dorite. Acest mod de încărcare a programului este asemănător cu cel disponibil de la interfața locală a microsistemului.

- 9.6. Să se scrie în limbaj de asamblare secvența de instrucțiuni pentru a copia portul de intrare (28) pe portul de ieșire (29). Se vor considera că pe portul de intrare sunt pe rând, următoarele valori: a) 00110011; b) 00001111.

- 9.7. Să se ștergă programul introdus anterior prin umplerea zonei de memorie de la 8000h cu 0 pe o distanță de 16 locații cu comanda **Memory – Fill ...**, să se verifice acest lucru cu comanda **View – Source at PC**, apoi să se introducă din nou secvența de program de mai sus de data aceasta direct în limbaj de asamblare și să se verifice acest lucru cu o nouă comandă **View – Source at PC**.

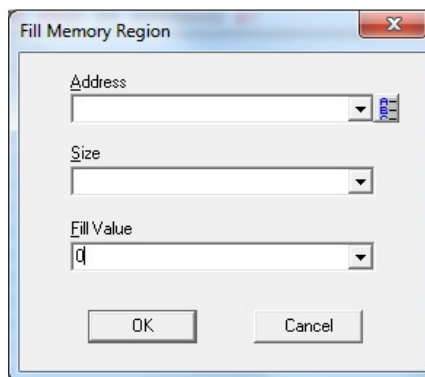


Figura 13. Comanda Fill Memory Region

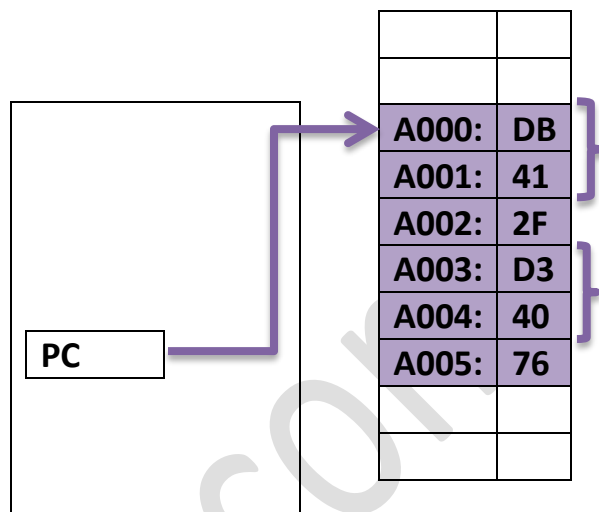
- 9.8. Să se afișeze o zonă de 16 octeți începând de la 8000h, atât în fereastra principală cât și în cea de date și să se observe identitatea conținutului memoriei.
- 9.9. Să se vizualizeze conținutul memoriei folosind tab-ul Memory

0A000	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0A000	DB	41	2F	D3	40	76	00	00	00	00	00	00	00	00	00	00	ŮA/ó@v.....
0A010	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	
0A020	CE	DC	FD	D8	38	C1	BE	14	79	FE	D7	6B	BA	A9	DF	3B	İÜý08Á%.yb*k°@B;
0A030	E7	9C	EC	25	2A	C2	38	F6	EB	43	C3	F6	35	8C	8A	D9	ç.i*Â88èCÀ85..Ů
0A040	3D	B6	D0	E6	F9	67	51	AA	36	31	5F	DF	72	FC	7B	D1	=ŸDæùgQ*61_sxü{N
0A050	7E	57	2F	6A	6C	5F	57	E2	7E	8E	BF	1C	44	61	19	35	~W/jl Wã~.ž.Da.5
0A060	B4	F3	27	7D	FA	55	CA	5C	3D	D2	32	DA	D1	06	3A	28	'ó'}úŮE\=02ŮN.:(
0A070	7C	89	40	D2	B3	7A	72	E1	7B	F2	77	B4	06	3F	76	C8	.00°zrã{òw'.?vÈ
0A080	A1	F3	68	C5	1B	25	5C	DC	90	A3	C0	06	D4	39	98	1E	;ónĀ.š\Ů.ĒĀ.09..
0A090	1D	AA	02	67	8D	6A	06	6B	BD	8E	D7	98	D9	49	D4	9F	.*.g.j.k*.x.ŮI0.
0A0A0	EF	9B	1D	90	EF	F0	48	17	6D	A4	10	50	FA	40	D1	7D	i...i0H.m*.Pú@N}
0A0B0	DF	71	90	5B	AF	DE	63	C3	1A	BB	59	65	D3	8B	0B	96	Bq.[_BcĀ.»Ye0...
0A0C0	E2	E1	07	CC	5F	EA	91	CE	F8	62	B1	78	B0	C5	E0	D7	áš.i_è.i0b±x°Āā*
0A0D0	FA	C5	88	DB	1A	F4	58	54	AA	35	81	6B	CC	7E	1E	EB	úĀ.Ů.0XT*5.kĩ~.è
0A0E0	D5	8A	D5	19	35	7B	40	25	37	73	0C	FE	C6	6F	2E	24	Ů.Ů.5{0@7s.pÆo.\$
0A0F0	E8	6B	C6	F2	60	63	FA	05	F1	BC	63	8A	81	63	79	0A	èkÆò`cú.ñ4c..cy.

Figura 14. NoiCE85 -> Memory

9.10. Să se scrie în memorie următoarea secvență de cod:

A001	DB 41	IN 41
A002	2F	CMA
A004	D3 40	OUT 40
A005	76	HLT



Să se precizeze opcodul fiecărei instrucțiuni și operanzii dacă este cazul.

10. Test de verificare a cunoștințelor

- 1) Cum se mai numește registrul A al microprocesorului 8085? De ce?
- 2) Câți biți are cuvântul de stare al procesorului?
- 3) Ce sunt indicatorii de condiții?
- 4) Ce este unitatea aritmetică și logică?
- 5) Ce fel de operanzi prelucrează ALU?
- 6) Care este registrul care conține octetul cel mai puțin semnificativ din registrul pereche H?
- 7) Câți biți are numărătorul de program?
- 8) Care este valoarea inițială a registrelor PC și SP, după resetarea de la conectarea alimentării?
- 9) În ce sens evoluează registrul PC, în mod implicit, odată cu execuția instrucțiunilor?
- 10) În ce sens crește stiva la microprocesorul 8085?
- 11) Registrul SP conține adresa ultimei locații ocupate din stivă sau adresa primei locații libere din stivă?
- 12) Câți biți are un cuvânt la microprocesorul 8085?

11. Referințe bibliografice

- [1] *Sisteme cu microprocesoare*, Îndrumar de laborator.
- [2] *Sisteme cu microprocesoare*, Note de curs.
- [3] Gh.Toacșe, *Introducere în microprocesoare*, Editura Științifică și Enciclopedică, București, 1985.