SISTEMUL DE DEZVOLTARE. ARHITECTURA GENERALĂ A MICROPROCESORULUI 8085.

1. Obiective

Prin parcurgerea acestei ședințe de laborator studenții vor fi capabili:

- Să învețe structura internă şi modul de funcţionare al microprocesorului 8085.
- Să enumere resursele hardware ale microsistemului;
- Să explice modul de lucru al monitorului rezident în memoria ROM;
- Să utilizeze facilităţile de operare locală ale microsistemului;
- Să folosească facilităţile de operare de la distanţă.

2. Arhitectura microprocesorului 8085

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

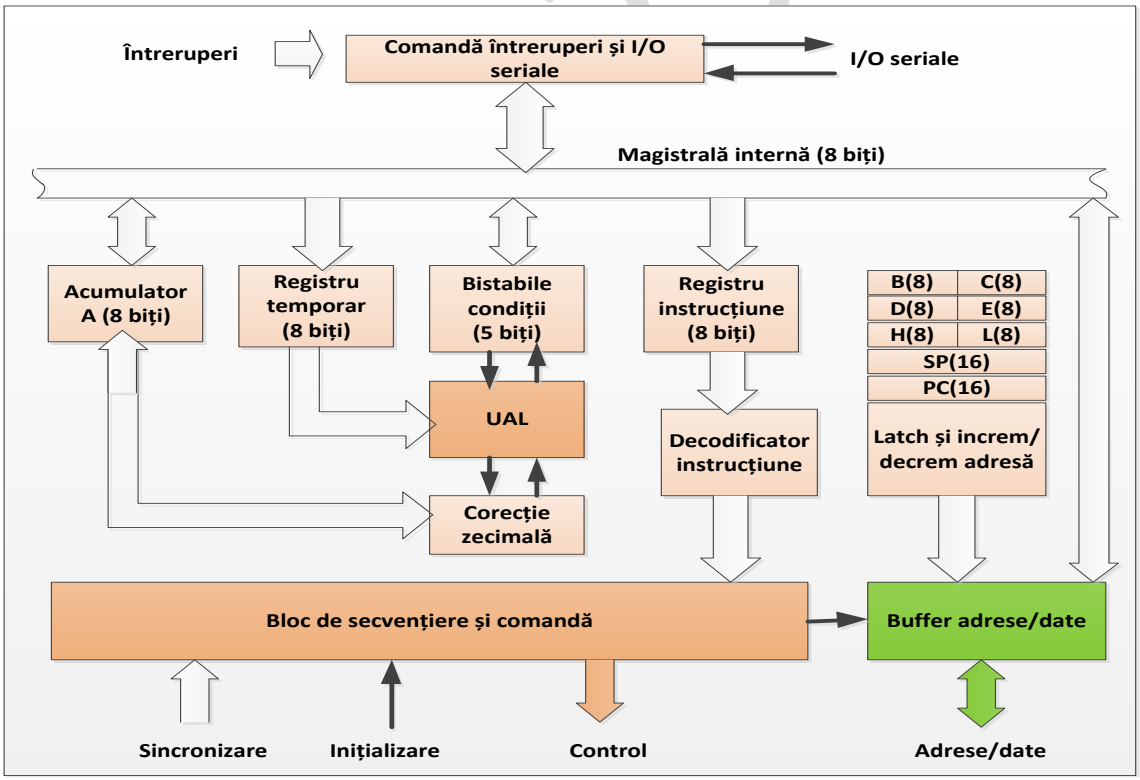


Figura 1. Schema bloc internă simplificată a microprocesorului 8085

3. Unitatea aritmetico-logică

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

Modul de desfăşurare a operaţiei şi caracteristicile rezultatului sunt memorate într-un set de bistabile denumiţi indicatori de condiţii (flags):

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

Starea acestor indicatori poate influenţa modul de desfăşurare şi rezultatul unor instrucţiuni aritmetice şi de ramificare condiţionată.
Acumulatorul, împreună cu flagurile formează cuvântul de stare al procesorului (PSW – processor status word).

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

4. Registrele interne și magistralele externe

4.1. Registrele interne

Registrele generale B, C, D, E, H şi **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 instrucţiuni adecvate, atât ca registre simple de 8 biţi, cât şi în perechi, ca registre extinse de 16 biţi:

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

Registrul instrucţiunii primeşte opcodul instrucţiunii curente şi îl menţine pe întreaga durată de execuţie a instrucţiunii.

Registrele speciale au **16** biţi şi îndeplinesc anumite funcţii ale microprocesorului. Astfel, **numărătorul de instrucţiuni (PC - Program Counter**) conţine în fiecare moment adresa instrucţiunii ce urmează a fi executată. Ele este **iniţializat cu 0 la resetarea microprocesorului** şi este implicit incrementat pentru secvenţe liniare de instrucţiuni, respectiv este modificat direct de către instrucţiunile 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 magistala 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 inplicit 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.

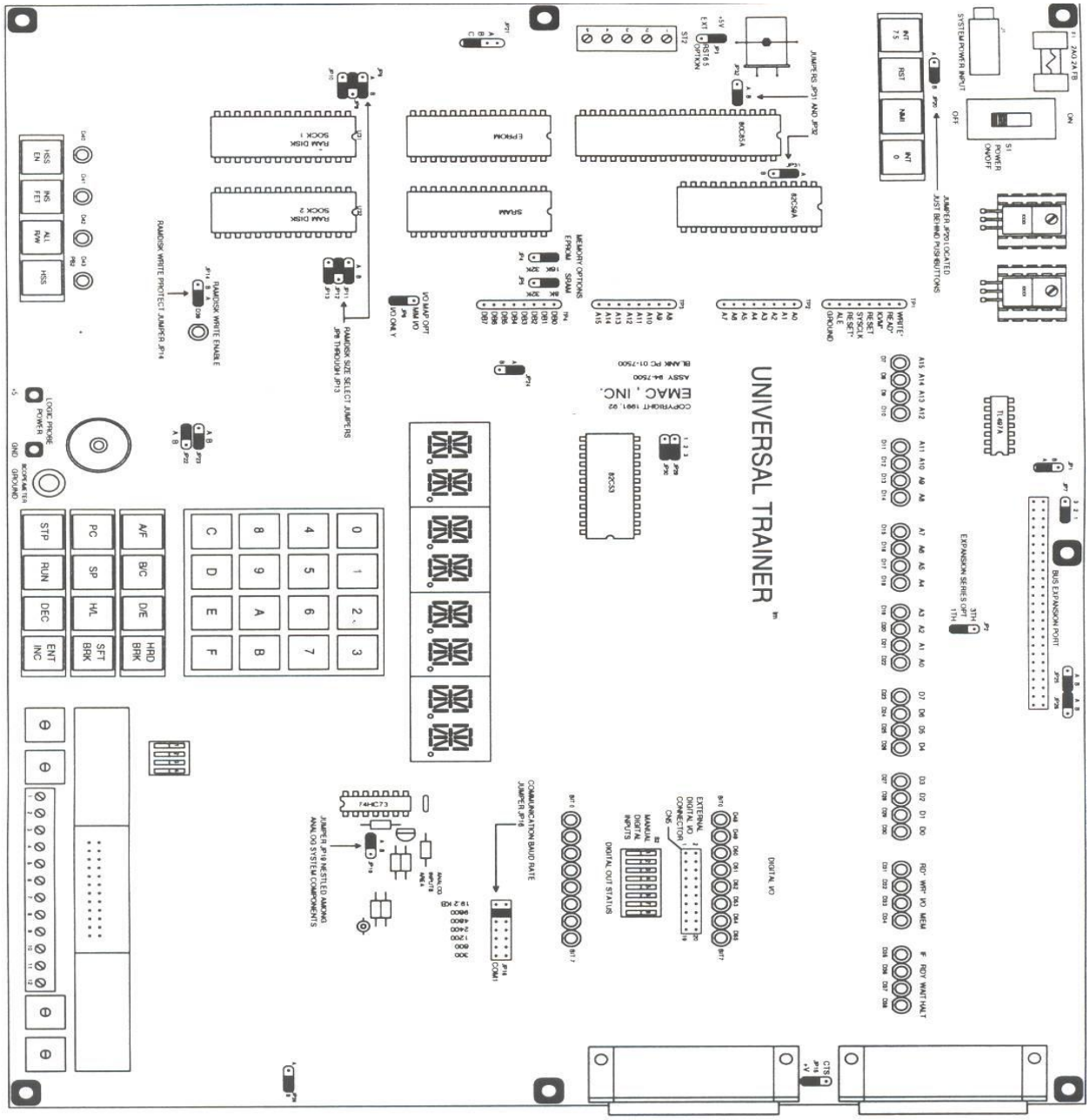
5. Modul de execuție al instrucțiunilor

Pentru a realiza **execuția** unei instrucțiuni microprocesorul:
- depune adresa instrucțiunii 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)



DRAWING 2.

Figura 2. Schema plăcii de dezvoltare

7. Resursele hardware ale sistemului cu microprocesor

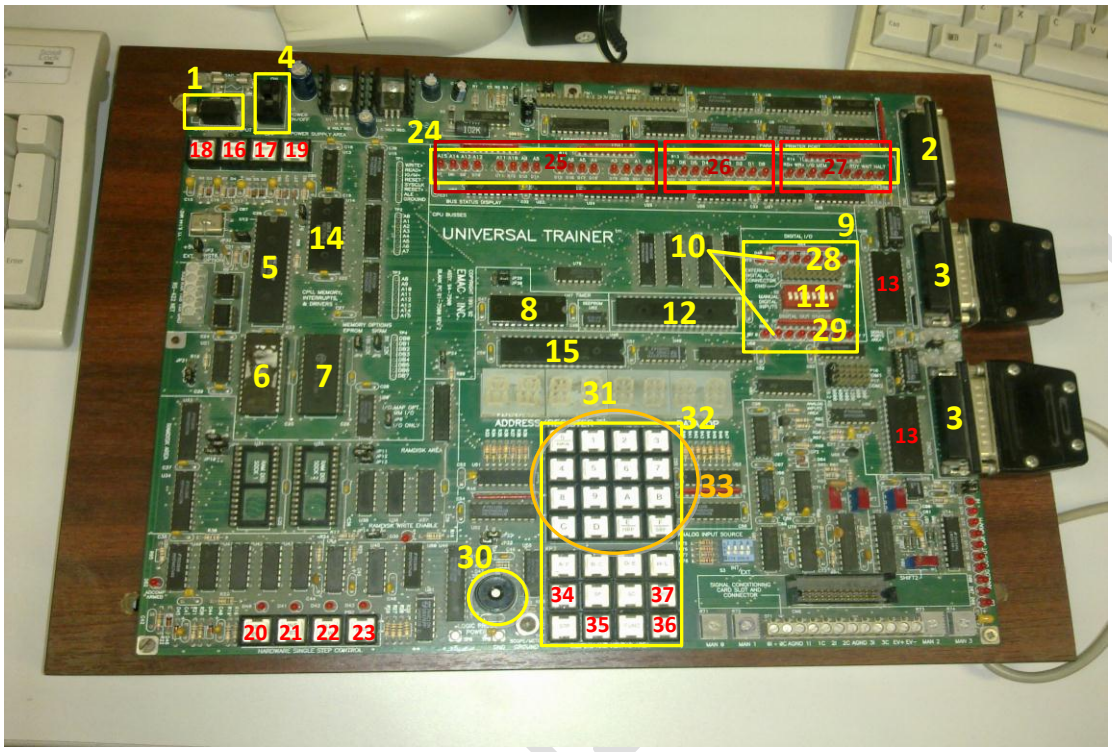


Figura 3. Principalele componente ale microsistemului

- **Mufă de alimentare** - premite alimentarea de la o sursă de tensiune externă de cel puţin 7V c.c.; **(1)**
- **Mufă port paralel** **(2)**
- **Mufe port serial** **(3)**
- **Comutator bipoziţional** - de conectare (ON) / deconectare (OFF) a alimentării **(4)**
- **Microprocesorul** - 80C85; **(5)**
- **Memoria ROM** - de 32 Kocteţi, la începutul căreia se află înscris monitorul rezident ce permite operarea locală şi de la distanţă;
  - locaţiile de memorie RAM 8000h÷FF00h sunt la dispoziţia utilizatorilor, pentru încărcarea şi execuţia programelor acestora; **(6)**

- zona de memorie RAM FF00h÷FFFFh nu trebuie modificată;

- **Memoria RAM** - de 32 Kocteţi, din care ultimele 256 de locaţii (FF00h÷FFFFh) sunt folosite de către monitorul rezident; **(7)**
- **Circuitul programabil de timp (Timer)** - cu 3 canale de timp, din care unul poate comanda un difuzor; **(8)**
- **Interfaţa paralelă** - cu 3 porturi de 8 biţi: **(9)**
  A - comandă un set de 8 led-uri de stare (D56÷D63) (40H); **(10)**
  B - conectat la 8 comutatoare DIP (S2), **(11)**
  Starea comutatoarelor este afişată şi pe leduri (D48÷D55) (41H); **(10)**
- **Controlerul pentru portul paralel** **(12)**
- **Circuitele de interfaţă serială** - pentru două canale de comunicaţie serială RS232: COM1 şi COM2 (COM1 este folosit de monitorul rezident); **(13)**
- **Controlerul de întreruperi** - cu 8 linii de întrerupere, conectat la linia INTR a microprocesorului; **(14)**
- **Controler pentru afișaj și tastatură** **(15)**
- **Butoane de inițializare**
  RST - resetează microsistemul, astfel încât, indiferent în de starea sa, monitorul rezident repornește, similar cazului de conectare a alimentării; **(16)**
  TRP - determină apariţia unei întreruperi nemascabile (care este acceptată necondiţionat de microprocesor), însmenând încetarea activităţii curente a microsistemului şi reintrarea în bucla de aşteptare de comenzi; **(17)**
  - această comandă este utilă pentru a întrerupe în orice moment execuţia unui program al utilizatorului, pentru a examina starea registrelelor sau pentru a vedea unde s-a blocat;
  INT7.5 **INT0** - permit generarea manuală a altor tipuri de întreruperi şi rolul lor va fi examinat în detaliu într-o lucrare ulterioară. **(18)** **(19)**
  HSS EN Hardware Single Step Enable – Activare execuție pas cu pas **(20)**

- **Butoane și logică HW pentru execuția pas cu pas**
  - **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: **(24)**
    - D7-D22 pentru magistrala de adrese (A15-A0); **(25)**
    - D23-D30 pentru magistrala de date (D7-D0); **(26)**
    - D31-D38 pentru comenzi (R, W, etc.); **(27)**
  - **D48-D63** - pentru vizualizarea porturilor I/O:
    - D48-D55 - pentru intrări; **(28)**
    - D56-D63 - pentru ieșiri; **(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 şi pornirea microsistemului, se verifică dacă toţi jumperii se află în poziţiile corecte, aşa cum se indică în figura 3. Nu se vor face niciodată modificări în configuraţia jumperilor în timp ce microsistemul se află sub tensiune.

Microsistemul are o construcţie robustă, dar asta nu înseamnă că în anumite condiţii nu poate fi deteriorat. El este special proiectat pentru a fi foarte deschis, astfel încât să se observe clar starea anumitor componente de semnalizare şi aceasta să fie uşor 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 şi acţionarea comutatorului S1 (POWER) pe poziţia ON, în câmpul ADRESA al afişajului apare adresa iniţială 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 şir de tonuri specifice, semn că microsistemul s-a iniţializat cu succes şi aşteaptă comenzi.

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

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. Programarea microprocesorului 8085

Un fișier sursă în limbaj de asamblare pentru ASM85 poate să conțină una sau mai multe linii de cod sursă. O linie de cod sursă are următorul format:

<etichetă> [:] <instrucțiune> <operanzi> [;] <comentariu>

În oricare din câmpurile menționate mai sus se pot utiliza atât litere mici cât și majuscule în orice combinație, asamblorul nefiind case sensitive.

Eticheta este opțională, ea reprezintă un nume simbolic dat adresei la care se va stoca instrucțiunea următoare. În cazul în care folosim eticheta, aceasta trebuie să fie localizată fix la începutul liniei din codul sursă. Opțional, după etichetă poate să urmeze caracterul ”:” (două puncte). Eticheta trebuie separată de câmpul care urmează prin cel puțin un ” ” (spațiu) sau ” ” (Tab). Pentru a păstra compatibilitatea cu alte asambloare eticheta va fi întotdeauna urmată de caracterul ”:”.

Câmpul aferent instrucțiunii poate fi amplasat pe aceeași linie cu eticheta sau pe linia următoare. În cazul în care nu există o etichetă, instrucțiunea nu trebuie plasată pe prima coloană, ci trebuie să fie precedată de cel puțin un spațiu sau un Tab. Dacă instrucțiunea are operanzi, aceștia trebuie separați de numele instrucțiunii printr-un spațiu sau un Tab. În cazul în care instrucțiunea are mai mulți operanzi, aceștia vor fi separați prin ”,” (virgulă).

Câmpul opțional dedicat comentariului trebuie să fie separat de operanzi prin cel puțin un spațiu sau un Tab și trebuie specificat prin ”;” (punct și virgulă) la începutul comentariului pentru a păstra compatibilitatea cu alte asambloare.

9.1. Instrucțiunea de inițializare

În categoria registrelor care pot fi inițializate regăsim registrele: A, B, C, D, E, H și L. Formatul general al instrucțiunii de inițializare este:

MVI r,data8 MoVe Immediate data data8 to register r $r \leftarrow data8$

0	0	**d**	**d**	**d**	1	1	0
data8							

În limbaj de asamblare, această instrucțiune are doi operanzi și semnificația că în registrul r se înscriu datele din data8. În limbaj mașină, instrucțiunea are doi octeți, poziționați în memorie la adrese consecutive. Primul octet reprezintă codul operației și cuprinde o serie de biți fixați care reprezintă codul operației și trei biți **(ddd)** care codifică registrul folosit de instrucțiune (registrul destinație) astfel:

r	**B**	**C**	**D**	**E**	**H**	**L**		**A**
ddd	000	001	010	011	100	101	110	111

Cel de-al doilea octet al instrucțiunii este reprezentat de o valoare numerică pe 8 biți care este încărcată în registrul specificat de opcod.

În limbajul de asamblare ASM85, un astfel de operand poate fi exprimat printr-o constantă numerică, o constantă simbolică a cărei valoare a fost definită în cadrul fișierului sursă sau printr-o expresie aritmetică sau logică cu constante numerice și/sau simbolice.

9.2. Instrucțiuni de intrare – ieșire

În cazul microprocesorului 8085, adresarea porturilor (I/O) se face pe 8 biți, deci adresa unui port poate să fie cuprinsă între 00h și FFh. Transferul I/O se realizează prin intermediul registrului A astfel: starea unui port de intrare (I) poate fi încărcată numai în A, iar comanda unui port de ieșire (O) se poate face numai cu octetul din A. Vorbim în acest caz de adresarea implicită.

Formatul general al instrucțiunilor de intrare-ieșire este prezentat în cele ce urmează:

IN port Input from port $A \leftarrow (port)$, $port = 00h \div FFh$

OUT port Output to port $(port) \leftarrow A$, $port = 00h \div FFh$

9.3. Instrucțiuni de salt

9.3.1. Instrucțiunea de salt necondiționat JMP

Orice modificare a registrului PC modifică ordinea de execuție a instrucțiunilor și este echivalentă cu un salt în program. Instrucțiunea care modfică registrul PC se numește JMP (jump) și are următoarea formă generală:

JMP addr JuMP to address $PC \leftarrow addr$

Primul operand al instrucțiunii nu este menționat, acesta fiind implicit registrul PC. Cel de-al doilea operand care este specificat, precizează adresa instrucțiunii care se va executa după JMP, adică adresa de memorie la care se face saltul. Saltul la adresa specificată se execută indiferent de starea curentă a programului (regiștre și indicatori de condiții).

9.3.2. Instrucuțiuni de salt condiționat

Formatul general al unei instrucțiuni de salt condiționat este următorul:

Jcondition addr Conditional JuMP to address $if(condition == true)$
$PC \leftarrow addr$
$else$
$PC \leftarrow PC + 3$

În cazul în care condiția precizată ”condition” este adevărată, se execută un salt la adresa precizată în instrucțiune , prin încărcarea ei în registrul PC. În cazul în care condiția este falsă, se continuă cu instrucțiunea următoare.

Corespondențele dintre condiție și codul acesteia sunt ilustrate în cele ce urmează:

Condiția	Descriere	Condiția este adevarată dacă	CCC
NZ	**Not Zero** – rezultatul este diferit de 0;	$Z = 0$	000
Z	**Zero** – rezultatul este egal cu 0;	$Z = 1$	001
NC	**Not Carry** – operația nu a produs transport/împrumut;	$CY = 0$	010
C	**Carry** – operația a produs transport/împrumut.	$CY = 1$	011

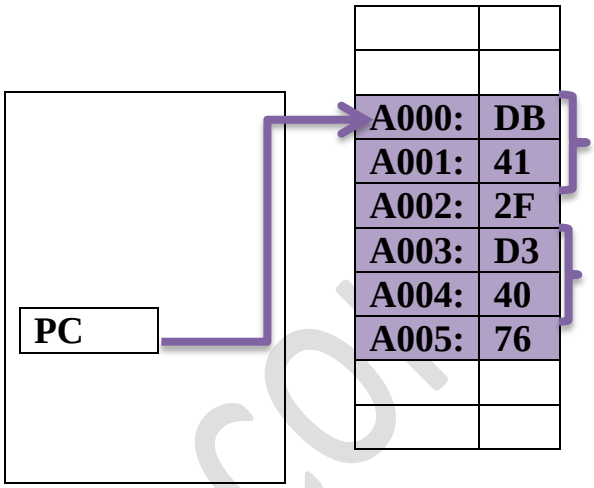
Cu ajutorul acestor instrucțiuni pot fi scrise structuri de program de tip ramificare simplă (if...else) sau multiplă (switch...case), dar și structuri de program repetitive/ bucle cu număr de pași cunoscut (for), cu test inițial (while) sau cu test final (do...while).

Simbolul predefinit $ poate fi folosit în orice instrucțiune pentru a ne referi la adresa acelei instrucțiuni. De aceea utilizarea ei într-o instrucțiune de salt face ca microprocesorul să execute continuu aceeași instrucțiune de salt la infinit.

10. Aplicații propuse

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

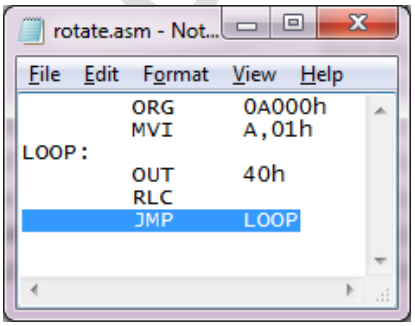
A001	**DB 41**	IN 41
A002	**2F**	CMA
A004	**D3 40**	OUT 40
A005	**76**	HLT



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

10.2. Să se scrie în memorie un program care rotește la dreapta un led și ilustrează procesul pe portul de ieșire.

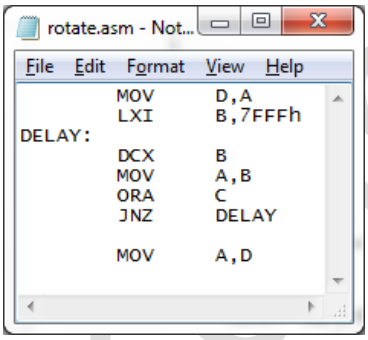
Indicație



a) De ce a fost inițializat registrul A cu valoarea 01h?
b) Ce se observă la lansarea în execuție?
c) Să se execute pas cu pas secvența de instrucțiuni astfel: cu oprire după fiecare instrucțiune și cu oprire după fiecare cod.
d) Câte apăsari ale butonului HSS sunt necesare în primul caz? Dar în cel de-al doilea?

10.3. Să se completeze codul de la problema 10.2. prin introducerea unui delay în program astfel încât rotirea ledului să poată fi observată fără a utiliza execuția pas cu pas.

Indicație



11. Referințe bibliografice

[1] C.Huțanu, M.Postolache, *Sisteme cu microprocesoare*, Editura Academică, Iași, 2001.
[2] Gh.Toacșe, *Introducere în microprocesoare*, Editura Științifică și Enciclopedică, București, 1985.
[3] ***, Universal Trainer, Lab Manual for Board Revisions R1 and R2, EMAC INC, 1993.
[4] ***, Universal Trainer, Reference Manual, EMAC INC, 1993.
[5] ***, Universal Trainer, Self Instruction Manual, EMAC INC, 1992.