# L8. STRUCTURA SISTEMULUI CU MICROPROCESOR. EXEMPLIFICAREA PE PLACA DE DEZVOLTARE. APLICAȚII DE CONECTARE A PLĂCII LA UN CALCULATOR ȘI DE EXAMINARE A CONȚINUTULUI MEMORIEI CU MONITORUL NoICE.

## 1. Obiective

Prin parcurgerea acestei ședințe de laborator studenții vor fi capabili:

- Să utilizeze instrucţiunile de iniţializare a registrelor;
- Să utilizeze instrucţiunile de I/O pentru citirea porturilor de intrare şi înscrierea porturilor de ieşire;
- Să înveţe să utilizeze instrucţiunile de ramificare necondiţionată şi condiţionată şi să construiască structuri de program caracteristice programării structurate de tip *for* în limbaj de asamblare;

## 2. Structura sistemului cu microprocesor și exemplificarea pe placa de dezvoltare

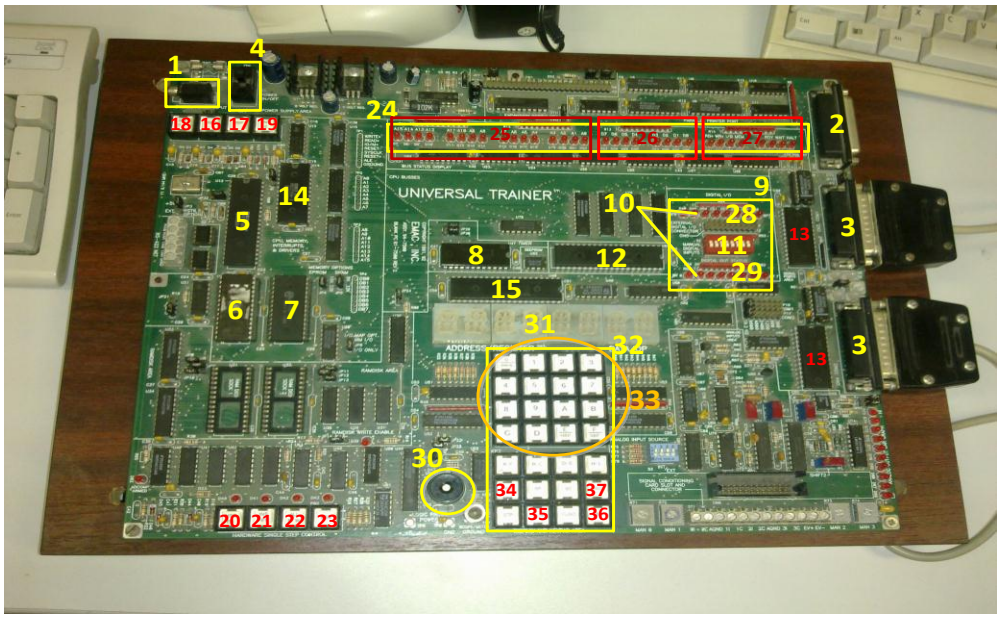



**Figura 1.** Principalele componente ale microsistemului

- **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;
  - această comandă este utilă pentru a întrerupe în orice moment execuţia unui program al utilizatorului, pentru a examina starea **(17)**

| | | | |
|---|---|---|---|
| | | registrelelor sau pentru a vedea unde s-a blocat; | |
| | **INT7.5** | - permit generarea manuală a altor tipuri de | **(18)** |
| | **INT0** | întreruperi şi rolul lor va fi examinat în detaliu | **(19)** |
| | | într-o lucrare ulterioară. | |
| • **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: | **(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)** |

## 3. 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.

## 4. 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.

## 4.1. 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.

## 4.2. 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 | $A \leftarrow (port)$<br>$port = 00h \div FFh$ |
| **OUT port** | Output to port | $(port) \leftarrow A$<br>$port = 00h \div FFh$ |

## 4.3. Instrucțiuni de salt

### 4.3.1. 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).

### 4.3.2. Instrucuțiuni de salt condiționat

Formatul general al unei instrucțiuni de salt condiționat este următorul:

**Jcondition addr** Conditional JuMP to address

$$if\,(condition == true)$$
$$\quad PC \leftarrow addr$$
$$else$$
$$\quad PC \leftarrow PC + 3$$

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 |

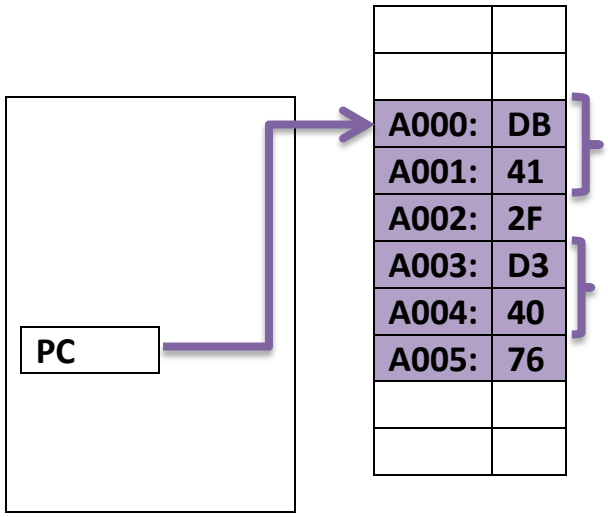
Cu ajutorul acestor instrucțiuni pot fi scrise structuri de program de tip ramificare simplă (if...else) sau multiplă (switch...case), dar și structuri de program repetitive/ bucle cu număr de pași cunoscut (for), cu test inițial (while) sau cu test final (do...while).

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.

## 5. Aplicații propuse

**5.1.** Să se scrie în memorie următoarea secvență de cod:

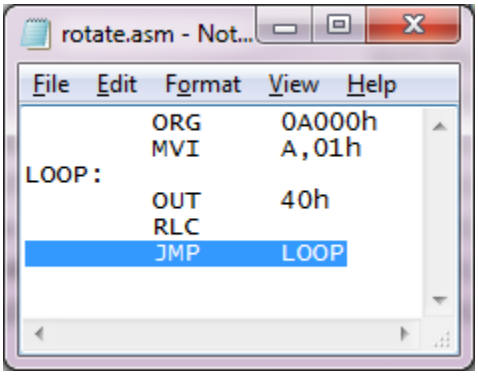
| | | |
|---|---|---|
| A001 | **DB 41** | IN 41 |
| A002 | **2F** | CMA |
| A004 | **D3 40** | OUT 40 |
| A005 | **76** | HLT |




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

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

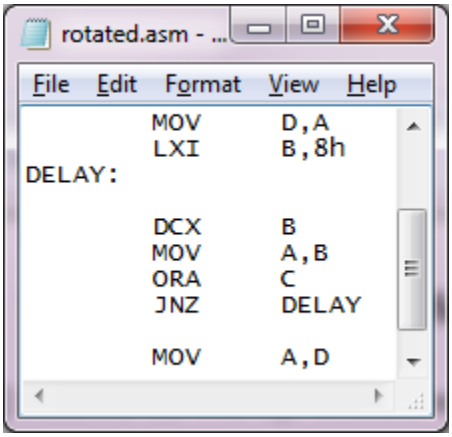
Indicație




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?

**5.3.** Să se completeze codul de la problema **5.2.** prin introducerea unui delay în program astfel încât rotirea ledului să poată fi observată fără a utiliza execuția pas cu pas.

Indicație




## 6. Referințe bibliografice

[1] *Sisteme cu microprocesoare*, Îndrumar de laborator.
[2] C.Huțanu, M.Postolache, *Sisteme cu microprocesoare*, Editura Academică, Iași, 2001.
[3] Gh.Toacșe, *Introducere în microprocesoare*, Editura Științifică și Enciclopedică, București, 1985.