MONITORUL NOICE. ASAMBLAREA DIRECTĂ A PROGRAMELOR ÎN MEMORIE. ASAMBLAREA UNUI PROGRAM CU ASM85.

1. Obiective

Prin parcurgerea acestei ședințe de laborator studenții vor fi capabili:

- Să utilizeze facilităţile de operare locală ale microsistemului;
- Să folosească facilităţile de operare de la distanţă.

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

2.1. Operarea remote

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

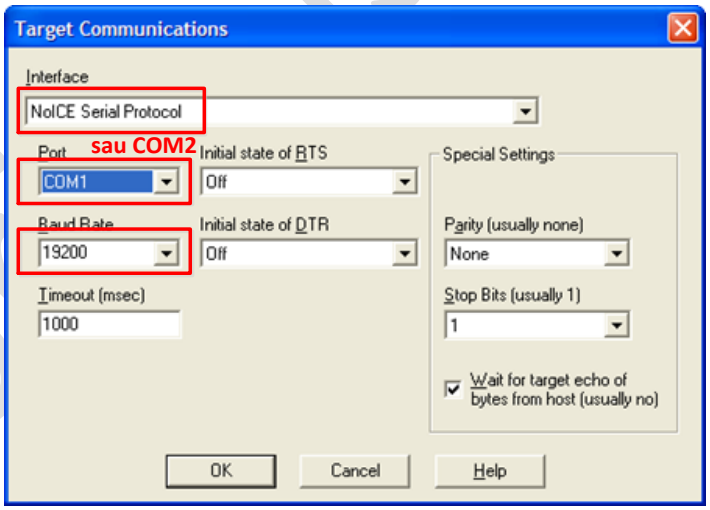

Figura 1. Configurarea comunicației cu programul NoICE85

În cazul în care a fost stabilită legătura cu monitorul rezident, pe ecran apare o fereastră similară cu cea din figura 2a. Dacă nu s-a reuşit stabilirea legăturii dintre PC şi EMAC Universal Trainer, pe ecran apare mesajul din figura 2b. În acest caz se verifică parametrii comunicaţiei şi cablul de legătură.

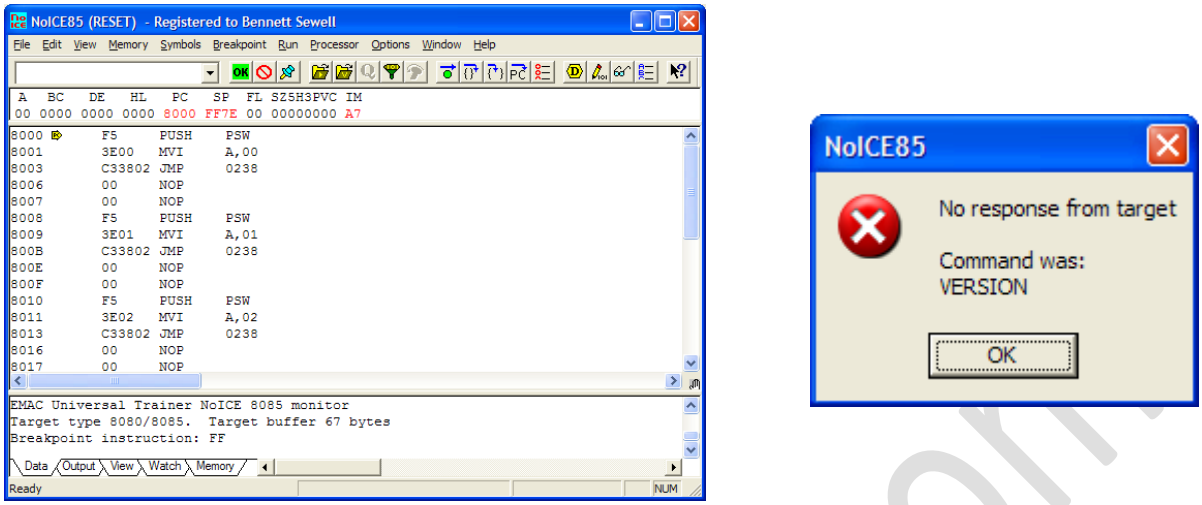

Figura 2. Legătura cu monitorul rezident
a) cu succes; **b)** eşuată.

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

- citire şi înscriere locaţii de memorie;
- citire şi înscriere registre ale utilizatorului;
- citire şi înscriere porturi de I/E;
- lansare în execuţie a programului de aplicaţie.

Cu ajutorul acestor comenzi, transparente pentru utilizator, sunt implementate facilităţile interfeţei grafice a depanatorului NoICE51:

- încarcă în memoria microsistemului codul executabil al programului de aplicaţie (**File-Play|Load**);
- editează, copiază şi caută informaţii în fişierul asociat ferestrei curente (**Edit**);
- vizualizează în fereastra principală a aplicaţiei codul programului în format dezasamblat, sursă sau mixt (**View**);
- vizualizează/editează conţinutul memoriei (**Memory-Dump|Edit**);
- asambleaza instrucţiuni direct în memorie (**Memory-Assemble**)
- urmăreşte evoluţia variabilelor programului (**Memory-Add watch**);
- defineşte, şterge şi afişează simbolurile specifice programului de aplicaţie (**Symbols**);
- inserează, afişează şi şterge punctele de oprire a execuţiei programului (**Breakpoint**);
- execută programul în regim pas cu pas sau automat, cu breakpoint (**Run**);
- vizualizează şi modifică conţinutul registrelor şi al porturilor de I/E (**Processor**);

2.2. Interfața NoICE85

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

2.3. Vizualizarea şi modificarea registrelor utilizatorului

Valorile iniţiale ale registrelor utilizatorului de la distanţă sunt afişate 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 condiţii sunt afişaţi atât în format hexa cât şi în binar (aşa cum intră în componenţa PSW) şi pot fi modificati de asemenea în ambele formate.

Conţinutul memoriei începând de la adresa iniţială din PC pentru programul utilizatorului (8000h) este afişat în fereastra principală a depanatorului, în format hexa şi dezasamblat (operaţia inversă asamblării, prin care se determină instrucţiunile corespunzătoare octeţilor 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 distanţă 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**.

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

Conţinutul memoriei poate fi modificat prin intermediul comenzii de meniu **Memory – Edit ...**, în care se poate stabili adresa locaţiei, valoarea şi se poate selecta tipul datei pentru afişare şi modificare.

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

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

Pornind de la adresa de început a secvenţei 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 acelaşi efect: execuţia următoarei instrucţiuni din programul utilizatorului.

În acest regim, microprocesorul execută doar o singură instrucţiune din programul utilizatorului la distanţă, după care controlul revine monitorului, care transmite către NoICE noile valori ale registrelor şi aşteaptă 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 instrucţiuni din programul utilizatorului. Acest program conţine o buclă infinită, în care pe

ledurile conectate la un port de ieşire este evidenţiată starea comutatoarelor conectate la un port de intrare. Să se modifice starea comutatoarelor şi să se urmărească schimbarea stării ledurilor din portul de ieşire.

Execuţia 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 distanţă şi aşteaptă o nouă comandă.

Se va observa cum execuţia programului utilizatorului se opreşte într-un anumit punct din buclă, în funcţie de momentul în care se apasă tasta TRP. În schimb, în cazul în care se resetează microsistemul, monitorul reface toate iniţializările, 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 iniţializată 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 fişier cu comanda **File – Save ...** în format binar (imagine memorie) sau Intel HEX (un format ASCII-hex introdus de Intel). După resetare, acest fişier poate fi reîncărcat în memoria microsistemului cu comanda **File – Load ...**, iar prezenţa programului în memorie poate fi observată cu o comandă **View – Source at PC**.

2.4. Etapele realizării şi rulării unui program

1. Crearea cu ajutorul unui editor de text (Ex: Notepad) a fişierului sursă. Fişierul sursă va avea extesia **.ASM** şi va conţine programul scris în limbaj de asamblare.

2. Asamblarea. Asamblarea este realizată cu ajutorul unui asamblor. Asamblorul folosit în cadrul şedinţelor 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 operaţiei de tip “drag and drop” a fişierului sursă peste iconiţa corespunzătoare asamblorului.

   În urma operaţiei de asamblare vor fi generate două fişiere:
   - *nume_fisier***.LST** - conţine erorile apărute pe parcursul asamblării şi liniile unde au apărut aceste erori;
   - *nume_fisier***.HEX** - reprezintă fişierul care va fi încărcat în memorie cu ajutorul programului NoIce.

3. Dacă nu au apărut erori pe parcursul operaţiei de asamblare, se lansează NoICE85. 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 fişierul sursă, se pot adăuga sau nu instrucţiunile dezasamblate din memoria microsistemului. Se observă că acestea coincid, ceea ce înseamnă că programul a fost corect asamblat şi încărcat.

4. Se încarcă registrul PC cu adresa la care a fost încărcat programul.

5. Se lansează programul în execuţie prin comanda **Run – Go (F5)**.

De aici încolo depanarea programului are loc beneficiind nu numai de codul dezasamblat, ci şi de toate facilităţile unei depanări simbolice: afişarea liniilor de cod sursă, urmărirea evoluţiei variabilelor programului, modificarea acestora, inserarea de puncte de oprire a execuţiei la adrese simbolice (etichetele instrucţiunilor) etc.

3. Aplicații propuse

3.1. Să se încarce regiștrii de date şi flag-urile cu următoarele valori:m

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

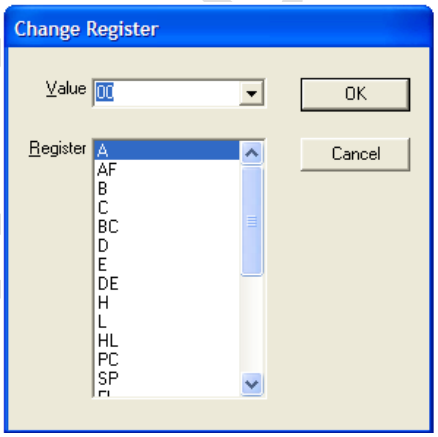

Figura 3. Modificarea regiștrilor

3.2. Să se încarce în memorie folosind comanda **Memory -> Assemble** începând de la adresa A000 următoarea secvență de instrucțiuni. ads 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 de pe portul de intrare și de pe portul de ieșire?

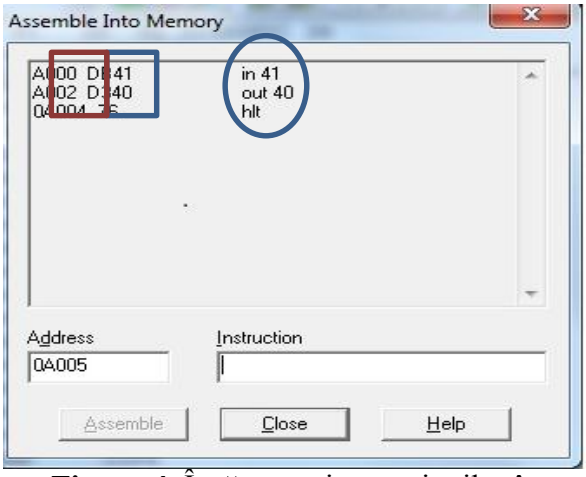

Figura 4. Încărcarea instrucțiunilor în memorie cu Memory -> Assemble

IN 41
OUT 40
HLT

3.3. 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 6.

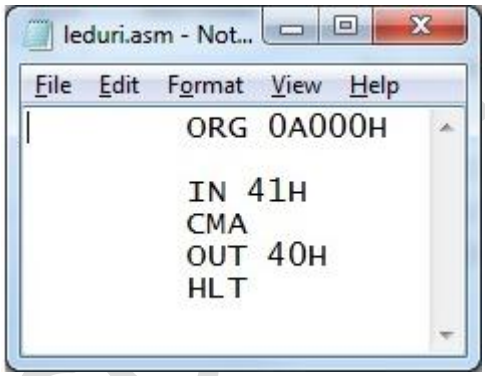

Figura 5. Fișierul leduri.asm

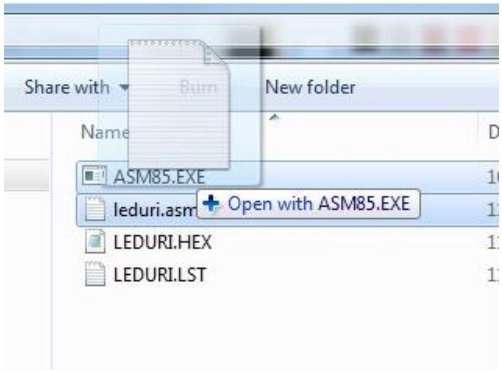

Figura 6. Execuția

Se vor genera două noi fișiere ce pot fi observate în figura 6. Fișierul **leduri.lst** va conține lista cu erorile apărute la execuție. Dacă fișierul respectiv este gol ca în figura 7, atunci nu există nici o eroare. Fișierul **leduri.hex** va putea fi încărcat în memorie.

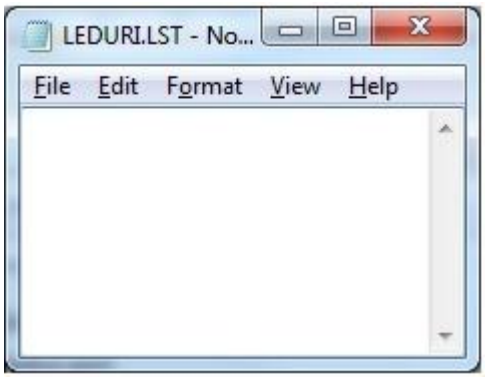

Figura 7. Fișierul ce conține erorile

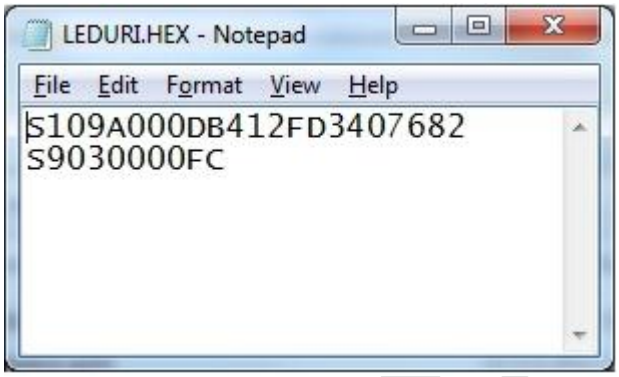

Figura 8. Fișierul leduri.hex generat

Să se încarce în memorie folosind comanda File -> Load fișierul leduri.hex. Să se vizualizeze conținutul de la adresa A000 similar cu figura 9.

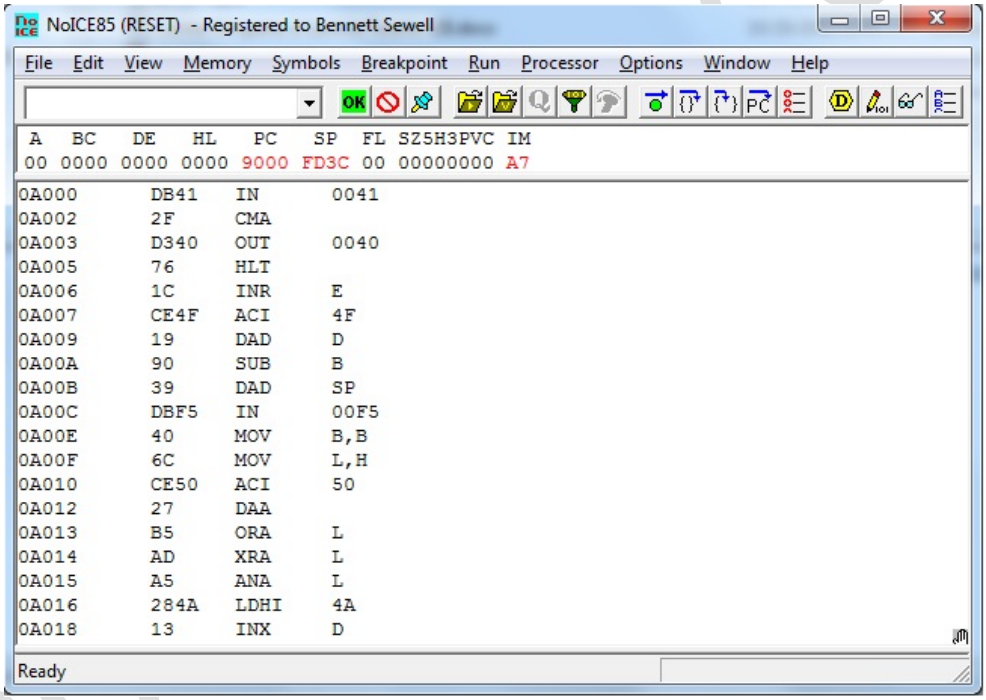

Figura 9. Conținutul memoriei începând cu locația A000

Să se lanseze în execuție programul.

3.4. Să se încarce în memorie folosind tastatura, 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:

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

3.5. Să se actualizeze ferestra de vizualizare a programului de la adresa din PC (A000h) cu comanda **View – Source at PC** şi să se verifice dacă instrucţiunile introduse coincid cu cele dorite. Acest mod de încărcare a programului este asemănător cu cel disponibil de la interfaţa locală a microsistemului.

3.6. Să se scrie în limbaj de asamblare secvența de instrucțiuni pentru a copia portul de intrare pe portul de ieşire. Se vor considera că pe portul de intrare sunt pe rând, urmatoarele valori: a) 00110011; b) 00001111.

3.7. Să se şteargă programul introdus anterior prin umplerea zonei de memorie de la 8000h cu 0 pe o distanţă de 16 locaţii cu comanda **Memory – Fill ...**, să se verifice acest lucru cu comanda **View – Source at PC**, apoi să se introducă din nou secvenţa de program de mai sus de data aceasta direct în limbaj de asamblare şi să se verifice acest lucru cu o nouă comandă **View – Source at PC**.

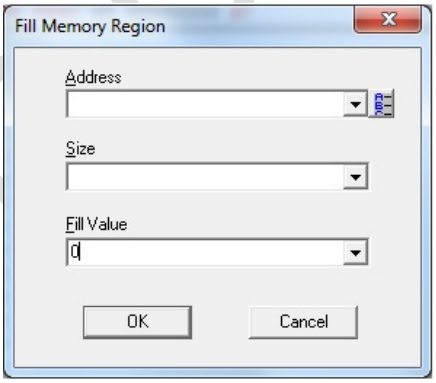

Figura 10. Comanda Fill Memory Region

3.8. Să se afişeze o zonă de 16 octeți începând de la 8000h, atât în fereastra principală cât şi în cea de date şi să se observe identitatea conţinutului memoriei.

3.9. Să se vizualizeze conținutul memoriei folosind tab-ul Memory

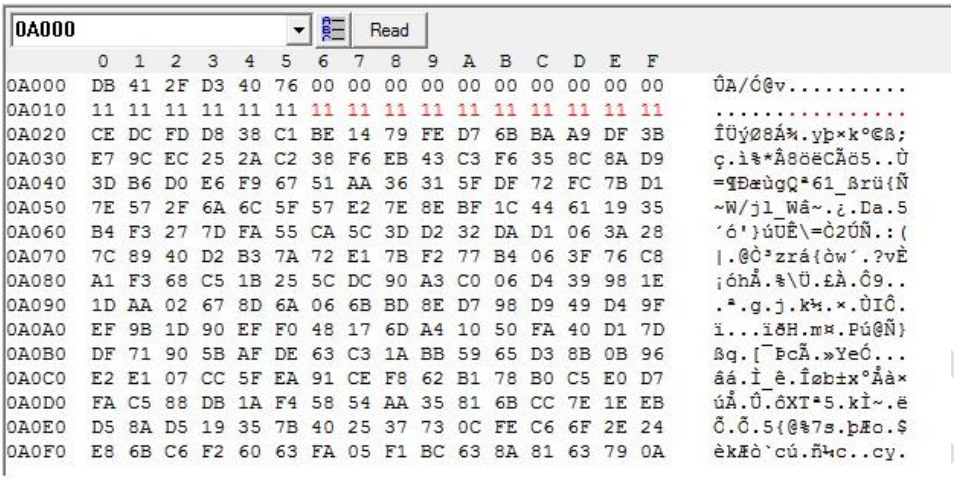

Figura 11. NoICE85 -> Memory

4. Test de verificare a cunoștințelor

1. Cum se mai numeşte registrul A al microprocesorului 8085? De ce?
2. Câţi biţi are cuvântul de stare al procesorului?
3. Ce sunt indicatorii de condiţii?
4. Ce este unitatea aritmetică şi logică?
5. Ce fel de operanzi prelucrează ALU?
6. Care este registrul care conţine octetul cel mai puţin semnificativ din registrul pereche H?
7. Câţi biţi are numărătorul de program?
8. Care este valoarea iniţială a registrelor PC şi SP, după resetarea de la conectarea alimentării?
9. În ce sens evoluează registrul PC, îm mod implicit, odată cu execuţia instrucţiunilor?
10. În ce sens creşte stiva la microprocesorul 8085?
11. Registrul SP conţine adresa ultimei locaţii ocupate din stivă sau adresa primei locaţii libere din stivă?
12. Câţi biţi are un cuvânt la microprocesorul 8085?

5. 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.