Arhitectura Calculatoarelor 2.
Laborator 1,2.

Sistem de Dezvoltare cu Microprocesor Z80

Sistemul de dezvoltare cu microprocessor Z80 (notație SDM-Z80) este realizat în jurul microprocesorului Z80 şi cuprinde un modul hard şi un soft dedicat care rulează concomitent atît pe SDM-Z80 (programul monitor) cît şi pe un calculator PC (mediul de dezvoltare), conectate prin legatura serială RS-232.

SDM-Z80 permite dezvoltarea unor programe în limbaj de asamblare Z80, avînd acces la resursele microprocesorului şi ale sistemului. De asemenea, se pot vizualiza pe monitorul PC pînă la 8 semnale logice de pe placa SDM-Z80, pentru o mai bună înțelegere a modului de funcționare a microprocesorului.

1. Configurația hardware

Modulul cuprinde următoarele elemente:

- Microprocesor Z80 la frecvența de 1.25 MHz
- Memorie EPROM 8Ko care conține programul monitor
- Memorie RAM 8Ko pentru rularea programelor utilizator
- Circuit numărător/temporizator cu 4 canale pe 8 biți materializat prin circuitul integrat Z80-CTC
- Interfață paralelă cu 2 porturi pe 8 biți (A şi B) materializată prin circuitul integrat Z80-PIO
- Interfață serială cu 2 canale seriale independente (A şi B) materializată prin circuitul integrat Z80-SIO
- Interfață RS232 cu 2 canale: unul pentru PC (portul B) şi celălalt disponibil utilizatorului (portul A), are rolul de a converti nivelele de tensiune compatibile TTL de la intrările/ieşirile SIO în nivele de -12V/+12V.
- 8 leduri conectate pe ieşirile interfeței paralele (portul A) şi 2 butoane conectate pe intrările PB0 şi PB1 ale portului B
- Sursa de alimentare 5V (+/- 5%)
- Conector pentru legătura cu portul paralel al PC (pentru vizualizarea a 8 semnale de pe placă)
- Pini de configurare

2. Configurația software

Modulul hard SDM-Z80 conține în EPROM *programul monitor,* care permite conectarea plăcii la un calculator PC. Monitorul comunică pe linia serială cu *programul principal (PCZ80.EXE)* care rulează pe PC, executînd comenzile cerute de acesta.

Programul principal al sistemului de dezvoltare este un pachet integrat care optimizează procesul de dezvoltare a programelor de aplicație, oferind utilizatorului următoarele facilități:

2.1 Încărcarea, editarea şi salvarea programelor sursă, folosind editorul de text încorporat

Editorul permite: folosirea tastelor şi combinațiilor de taste uzuale pentru parcurgerea şi editarea fişierelor text, folosirea operațiilor pe blocuri de text prin intermediul ferestrei intermediare Clipboard (*Cut, Copy, Paste şi Clear*). Este un editor multi-fereastră, care permite deschiderea pentru editare a mai multe fişiere simultan, în ferestre separate, suprapuse (*Cascade*) sau alăturate

(*Tile*). De asemenea, editorul permite folosirea operațiilor de căutare/înlocuire a şirurilor de caractere.

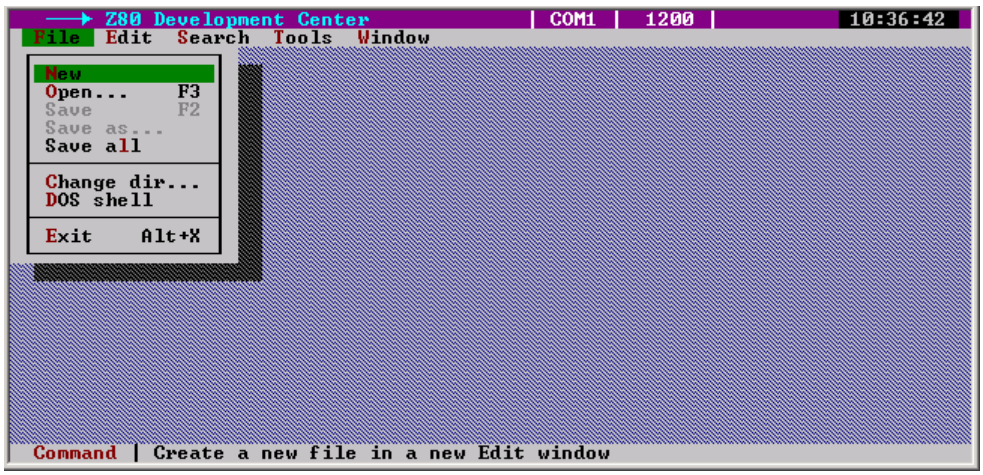
2.2 Apelarea unui program utilitar (asamblor) pentru generarea programului executabil

- încărcarea programului executabil pe placă
- depanarea programelor folosind comenzile programului monitor
- configurarea parametrilor programului

Parametrii programului sînt setările făcute de utilizator. Acestea se salvează la fiecare ieşire din program şi sînt refăcute automat la următoarea pornire a programului. Pe lînga parametrii programului se salveaza şi desktop-ul curent (toate ferestrele deschise, atît ca dispunere pe ecran cît şi ca dimensiune).

2.3 Descrierea meniurilor şi a casetelor de dialog

Sînt disponibile 5 meniuri principale: *File*, *Edit, Search*, *Tools şi Window*. Opțiunile din aceste meniuri determină executarea unor acțiuni sau afişarea unor casete de dialog. Sînt disponibile o serie de taste rapide (*hot keys*), pentru selecția opțiunilor mai des folosite. Pe ultima linie a ecranului se afişează un mesaj de ajutor pentru utilizator (*help contextual*).



Meniul ***File***

Este meniul pentru operațiile cu fişiere: crearea unui fişier nou, deschiderea pentru editare a unui fişier existent, salvarea unui fişier editat şi salvarea fişierului cu un alt nume. Opțiunea *Save all* permite salvarea tuturor fişierelor care sînt deschise în acel moment. Opțiunea *Change dir* se poate folosi pentru schimbarea directorului curent, iar cu *DOS Shell* se poate iesi temporar în sistemul de operare DOS. Revenirea în program se face cu comanda DOS exit. Opțiunea *Exit* determină terminarea programului. Selecția fişierului pentru deschidere se face printr-o fereastră de dialog ce afişează lista fişierelor. Se pot deschide mai multe fişiere simultan în editor. Fiecare fişier este încărcat într-o fereastră separată. Trecerea de la o fereastră la alta se poate face cu tasta *F6* (Next) sau cu *Alt+n*, unde n este numărul ferestrei dorite. La deschiderea fişierelor se face automat cascadarea ferestrelor (ferestrele se suprapun, dar bara de titlu pentru fiecare fereastră rămîne vizibilă).

Meniul ***Edit***

Opțiunea *Undo* permite anularea ultimei comenzi de editare. Meniul conține în continuare opțiunile pentru operațiile cu blocuri folosind fereastra intermediară Clipboard (*Cut, Copy, Paste şi Clear*). Aceste operații lucrează numai pe blocurile de text selectat (selectarea textului se poate face prin *drag* cu mouse-ul sau cu tastatura folosind *Shift+sg.jos/sg.sus*).

Fereastra Clipboard se poate deschide, ca şi orice altă fereastră de editare, folosind opțiunea *Show clipboard*.

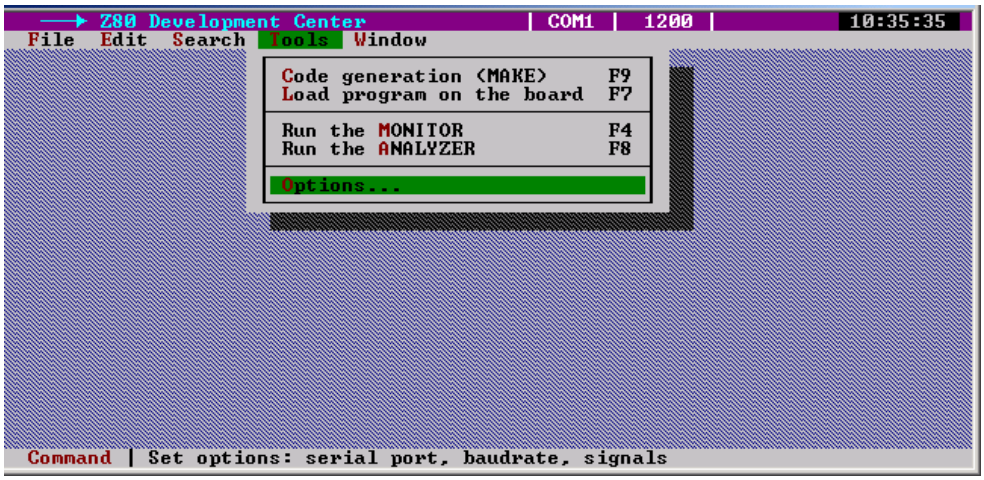
Meniul ***Search***

Meniul Search conține operațiile de căutare/înlocuire a şirurilor de caractere. Opțiunea *Find* permite căutarea aparițiilor unui şir de caractere în cadrul fişierului. La căutare se poate face diferența între literele mari şi mici (*Case sensitive*) şi, de asemenea, se pot căuta grupuri de caractere din cuvinte sau doar cuvinte întregi (*Whole words only*).

Opțiunea *Replace* permite căutarea şi înlocuirea aparițiilor unui şir de caractere în text cu un alt text specificat. Înlocuirea se poate face automat sau cu confirmarea utilizatorului. De asemenea se poate înlocui prima sau toate aparițiile textului (*Replace all*).

Ultima operație de căutare/înlocuire poate fi repetată folosind opțiunea *Search again*.

Meniul ***Tools***



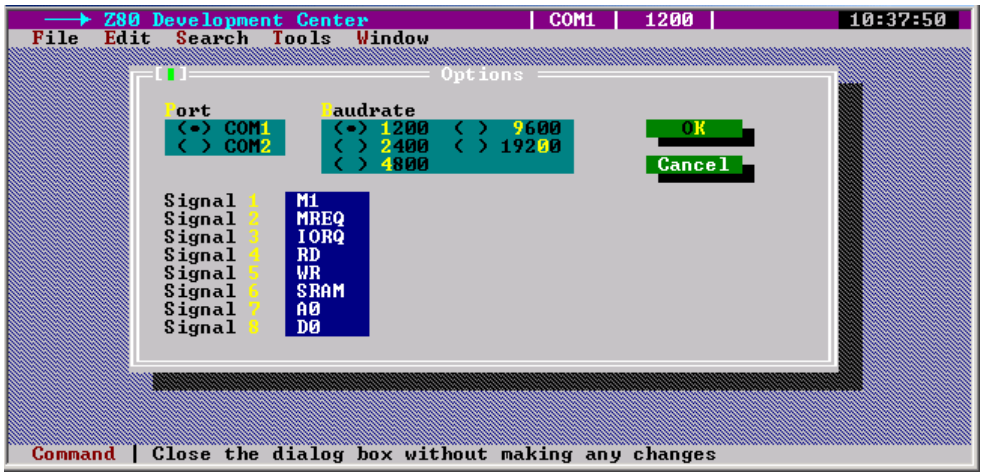
Opțiunea *Code generation (Make)* permite apelarea utilitarului (asamblorului) pentru generarea programului executabil (în format .HEX).

Opțiunea *Load program on the board* permite încărcarea pe placă a programului executabil (în format .HEX).

Opțiunea *Run the MONITOR* permite (într-o fereastră separată) depanarea programului încărcat pe placă utilizînd comenzile monitorului.

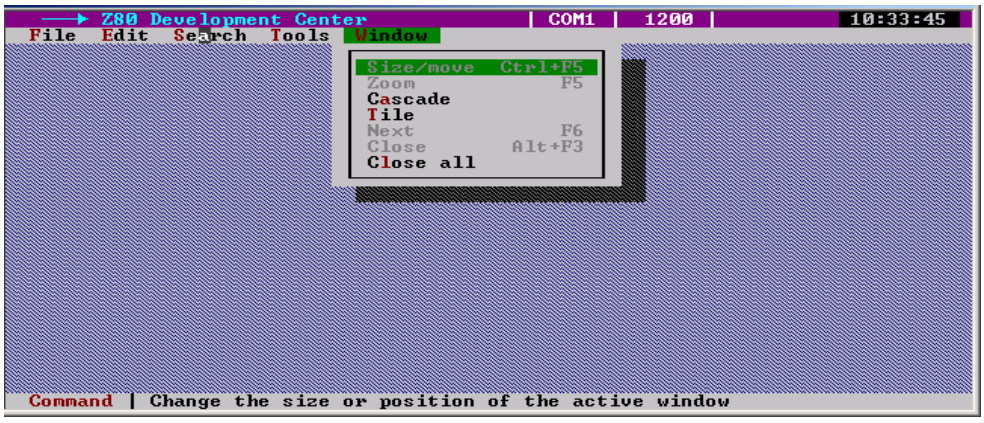
Opțiunea *Run the ANALYZER* permite (într-o fereastră separată) vizualizarea a 8 semnale logice de pe placă selectate în meniul *Options*.

Opțiunea *Options* permite configurarea unor parametri ai programului, care se salveaza în fişierul PCZ80.INI (din directorul curent) la ieşirea din program. La următoarea execuție a programului parametrii se încarcă din acest fişier, sau vor primi valorile implicite în cazul cînd fişierul nu există în directorul curent.



Caseta *Port* permite stabilirea portului serial folosit pentru comunicația între PC şi placa de dezvoltare.
Caseta *Baudrate* permite modificarea ratei de transfer cu care este programată interfața serială a PC-ului. Se poate opta pentru una din următoarele rate de transfer seriale: 1200 bauds (implicită), 2400, 4800, 9600, 19200 bauds. Comunicația între PC şi monitorul de pe placă se realizează la viteza de 1200 bauds (fixă), deci această rată de transfer este refăcuta automat de către programul principal la pornirea programului.
Semnalele logice de pe placă ce pot fi vizualizate sînt : M1, MREQ, IORQ, RD, WR, SELECȚIE RAM, A0, D0.

Meniul ***Window***



Acest meniu conține comenzile pentru gestiunea ferestrelor deschise. Unele ferestre pot fi redimensionate (mărite sau micşorate) şi pot fi mutate pe ecran în diverse pozitii. Folosind opțiunea *Next*, ferestrele pot fi parcurse circular. Opțiunile *Cascade* şi *Tile* permit aranjarea ferestrelor pe ecran în două moduri: suprapuse, respectiv alăturate. Cu *Close all* se pot închide toate ferestrele deschise la momentul respectiv.

3. Depanatorul

Depanatorul poate fi lansat în execuție din meniul principal, selectînd opțiunea *Run the MONITOR* din meniul *Tools*, sau prin folosirea tastei rapide *F4*.

După lansare, se afişează fereastra de lucru, mesajul şi prompter-ul dat de programul monitor. În continuare sînt disponibile toate comenzile monitorului. Tasta *Enter* execută comanda introdusă; dacă această comandă nu este recunoscută de monitor se afişează caracterul “?”.

Tasta *Escape* permite închiderea ferestrei de depanare şi revenirea în meniul principal.

3.1 Descrierea comenzilor monitorului

Vizualizarea conținutului memoriei
Sintaxa:

- **D adresa**
- **D adresa1,adresa2**

Comanda permite vizualizarea conținutului unei zone de memorie (RAM sau EPROM). Afișarea se face în modul obișnuit de afișare a zonelor de memorie. Pe fiecare rînd se afișează adresa și apoi 16 octeți de date în format hexazecimal.

Prima formă a comenzii afișează doar conținutul de la adresa selectată (un octet).

A doua formă a comenzii afișează zona de memorie delimitată de cele două adrese.

Modificarea conținutului memoriei
Sintaxa:

- **M adresa**

Comanda permite modificarea interactivă a locațiilor din memoria RAM începînd cu adresa selectată.

Modificarea se face interactiv în sensul că se va afișa adresa curentă și valoarea curentă a locației respective. În continuare, utilizatorul poate introduce o nouă valoare pentru acea locație, de la tastatură, sau poate trece la locația următoare (tasta Space) lăsînd locația curentă nemodificată. Monitorul va incrementa adresa curentă și va afișa conținutul ei. Operatorul va continua modificarea/examinarea conținutului memoriei, octet cu octet. Terminarea operației se face prin apăsarea tastei Enter.

Rularea programului utilizator
Sintaxa:

- **G adresa**

Comanda permite lansarea în execuție a programului utilizator începînd de la adresa selectată. Trebuie ca adresa specificată în comandă să fie efectiv adresa de început a unei instrucțiuni.

Vizualizarea conținutului registrelor
Sintaxa:

- **X**

Comanda permite vizualizarea conținutului curent al registrelor microprocesorului.

Vizualizare și modificare conținutului registrelor
Sintaxa:

- **XA (sau B, C…..)**

Comanda permite vizualizarea și modificarea conținutului unui registru al microprocesorului.

4. Sintaxa directivelor asamblorului

.code - definește segmentul de cod care conține programul (codurile instrucțiilor)
.segment xxx - definește un segment de program (cod sau date) cu numele " xxx "
.xxx - stabilește ca activ segmentul cu numele " xxx "

Noțiunea de segment nu are o implementare hardware la procesorul Z80. În consecință, folosirea directivelor de segment este opțională.

.equ <identif>,<const> - definește identificatorul *identif* și îi atribuie valoarea numerică *const.* Implicit, constanta se consideră a fi un număr exprimat în baza 10. Pentru a specifica baza 16 (constantă hexazecimală) se folosește sintaxa **h'const** (Exemplu: .equ spatiu,h'20)

.org <const> - asamblarea se face începînd de la adresa fizică *const*

<nume_var>: .db <const> - rezervă un octet de memorie pentru variabila <nume_var> la adresa curentă şi iniţializează variabila cu valoarea <const>. În locul constantei se poate preciza şi o listă de constante, despărţite prin virgulă, un caracter ASCII, încadrat între apostroafe sau un şir de caractere încadrat între ghilimele

Exemple:
```
var1:   .db     100
coduri: .db     0,107,h'3f,'A',spatiu
mesaj:  .db     "Ai grija la sintaxa!"
```

<nume_var> .rs n - rezervă o zonă de memorie de n octeţi, la adresa curentă, fără iniţializare.

Exemplu:
```
buffer: .rs  h'100
```

; - linie de program care nu se asamblează (comentariu)

.end - indică sfîrşitul fişierului de asamblat

```
;   exemplu de program în limbaj de asamblare
        .code                   ; segmentul de cod
        .org h'2100             ; adresa 2100h de la care se asambleaza programul

loop:
        ld a, h'22              ; instructiile programului
        ld b, (data1)
        ……..
        jp loop

        .db  1,2,3,4,5,'p'              ; date constante definite in segmentul
        .db  6,7,"test"                 ; de cod
        .dw  1,h'1234

        .segment .segment_date          ; segmentul de date
        .segment_date
        .org h'2300
data1:  .rs 1                   ; zona rezervata pentru variabila “data1”(un octet)
data2:  .rs 10*2                ; zona rezervata pentru un tablou de 10 cuvinte (20 octeti)

        .end loop               ; sfirsit asamblare
```

5. Instrucţiile microprocesorului Z80 (subset)

Se prezintă în continuare un subset minimal al setului de instrucţii al microprocesorului Z80. Pentru lista completă a instrucţiilor Z80 se poate consulta fişierul "opcode.txt".

Instrucţiile sînt grupate pe categorii, sub forma:

Mnemonică	**Cod binar (hex)**	**Descriere**

Instrucţii de transfer al datelor:

LD A,nn	3E nn	Încarcă în acumulator constanta **nn**
LD A,(hhll)	3A ll hh	Încarcă în acumulator octetul de la adresa **hhll.** Prin <hhll> s-a notat o constantă de 16 biţi, în care hh este octetul mai semnificativ iar ll octetul mai puţin semnificativ. În codul instrucţiei cei doi octeţi apar

		inversaţi, aceasta fiind o regulă hardware de reprezentare în memorie valabilă pentru Z80 ca şi, dealtfel, pentru alte procesoare (printre care toate procesoarelele din familia Intel).
LD (hhll),A	32 ll hh	Memorează la adresa **hhll** octetul din acumulator
LD A,B	78	Transfer din registrul B în registrul A
LD B,A	47	Transfer din registrul A în registrul B. Se pot transfera date între oricare două registre.
LD HL,hhll	21 ll hh	Încarcă în perechea de registre HL constanta hhll.
LD A,(HL)	7E	Încarcă în acumulator valoarea de la adresa de memorie conţinută de registrele HL.

Instrucţii de lucru cu porturile:

IN A,(nn)	DB nn	Citeşte în acumulator octetul de la portul **nn**
OUT (nn),A	D3 nn	Înscrie în portul **nn** octetul din acumulator

Instrucţii legate de lucrul cu întreruperile:

IM0	ED 46	Configurează întreruperile în modul 0
IM1	ED 56	Configurează întreruperile în modul 1
IM2	ED 5E	Configurează întreruperile în modul 2
EI	FB	Validează acceptarea întreruperilor mascabile
DI	F3	Invalidează acceptarea întreruperilor mascabile
LD I,A	ED 47	Încarcă în registrul I octetul din acumulator
RETI	ED 4D	Retur din întrerupere

Instrucţii de salt:

JP nnnn	C3 nnnn	Salt la adresa **nnnn**
JP Z,nnnn	CA nnnn	Salt la adresa **nnnn** dacă fanionul Z(zero) este activ
CALL nnnn	CD nnnn	Apel al subrutinei de la adresa **nnnn**
RET	C9	Retur din subrutină

Instrucţii aritmetice şi logice

AND nn	E6 nn	ŞI logic între acumulator şi constanta **nn**
OR nn	F6 nn	SAU logic între acumulator şi constanta **nn**
CP nn	FE nn	Compară conţinutul acumulatorului cu valoarea nn şi poziţionează corespunzător fanioanele
INC A	3C	Incrementarea acumulatorului
INC HL	23	Incrementează perechea de registre HL

6. Resursele sistemului de dezvoltare cu Z80.

Alocarea adreselor.

Resursele adresabile ale sistemului sînt cele care au fost menţionate mai sus, anume două capsule de memorie şi trei capsule cu porturi programabile. Schema de decodificare a adreselor este proiectată aşa fel încît aceste resurse sînt vizibile la adresele următoare.

0000h..1FFFh - Memorie EPROM. O capsulă de 8 kocteţi, conţine programul monitor.

2000h..3FFFh - Memorie RAM. O capsulă de 8 Kocteți, spațiu care este folosit astfel:

2000h..20FFh rezervat pentru stivă şi zona de lucru a monitorului

2100h..3FFFh disponibil pentru utilizator.

Rezervarea zonei 2000..20FF nu are o implementare hardware, ci trebuie înțeleasă ca o recomandare. Utilizatorul are posibilitatea să scrie în acea zonă, fie folosind comanda **M** a monitorului, fie prin program, dar consecințele vor fi "de natură imprevizibilă şi neplăcută", probabil blocarea sistemului care va trebui resetat.

Cele trei capsule cu porturi programabile sînt văzute, fiecare, sub forma a patru registre adresabile, utilizabile pentru scriere şi citire, prin care procesorul comunică cu portul, trimițînd comenzi şi citind stări, respectiv scriind/citind date. Funcționarea concretă a fiecărui circuit va face obiectul cîte unei lucrări de laborator. Spațiile de adrese alocate acestor dispozitive este următorul:

C0h..C3h	Z80-CTC
C3h..C7h	Z80-SIO
C8h..CBh	Z80-PIO

Rutine ale monitorului.

Unele rutine care fac parte din programul monitor pot fi folosite de către utilizatorul sistemului, adică pot fi apelate din programe scrise de către utilizator în memoria RAM.

- Afişarea pe display a valorii unui octet, în format hex, se poate face apelînd rutina de la adresa 026Dh. Octetul ce se doreşte a fi afişat trebuie încărcat în registrul acumulator.

- Afişarea pe display a unui caracter ASCII, al cărui cod de octet se găseşte în acumulator se poate face apelînd rutina de la adresa 06CAh.

- Incheierea unui program scris de utilizator se face predînd controlul programului înapoi către monitor. Concret, execuția comenzii *G xxxx* se traduce printr-o instrucție de salt la adresa *xxxx*, iar încheierea programului se va face printr-un salt la adresa 0000h. Efectul este acelaşi cu cel al apăsării tastei RESET, adică lansarea în execuție a programului monitor de la începutul său.

7. Desfăşurarea lucrării.

7.1 Pe baza informațiilor furnizate se va face o schemă bloc a modulului care să conțină următoarele elemente:

- microprocesorul, cele două capsule de memorie şi cele trei porturi, figurate ca blocuri
- două blocuri de decodificare a adreselor, unul pentru memorie şi celălalt pentru porturi, figurate ca black-box
- se vor pune în evidență urmatoarele semnale: magistrala de date, magistrala de adrese, semnalele de control MREQ/, IREQ/, RD/, WR/. Trebuie să reiasă clar ce linii de adresă intră în fiecare bloc.

7.2 Faceți o comparație între rutinele de monitor de la adresele 026Dh şi 06CAh. Care este diferența dintre ele. Scrieți cîte un scurt program care să testeze execuția acestor rutine.

7.3 Folosind instrucții Z80 dintre cele descrise mai sus, scrieți un program care:

- declară şi inițializează un "mesaj", sub forma unui şir de caractere încheiat cu valoarea 00h (ca în limbajul C).
- afişează acest şir pe display.