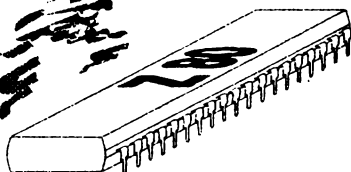




SERIA INFORMATICA

Mircea Mihail
POPOVICI



LIMBAJUL MASINA

al calculatoarelor
ZX SPECTRUM, HC,
TIM-S, COBRA, CIP,
JET, ...

EDITURA APH



Bucuresti 1993



SERIA INFORMATICA

Mircea - Mihail
Popovici

LIMBAJUL MASINA

al calculatoarelor
ZX SPECTRUM,
HC 90, TIM-S, COBRA,
CIP, JET ...

EDITURA APH



BUCURESTI 1993

Această lucrare inițializează cititorul în cunoașterea și aplicarea limbajului mașină al microprocesorului Z80 și este ilustrată prin circa 150 de rutine pe care le poate folosi în propriile sale programe.

În *capitolul 1* sînt prezentate sistemele de numerație (zecimal, binar și hexazecimal), structura de bază a unui microprocesor (magistralele de date și adrese, registre și fanioane) și asamblorul GENS3M21 (directive de asamblare, comenzile editorului, algoritmul de lucru).

Capitolul 2 este consacrat celor 13 seturi de instrucțiuni ale microprocesorului Z80; pentru fiecare set sînt prezentate mnemonicele și operațiile realizate, precum și fanioanele afectate.

Capitolul 3 ilustrează prin aplicații modul de folosire a instrucțiunilor în operații de bază (încărcare în memorie, operații aritmetice, influențarea unui bit și transferurile de blocuri din memorie).

Capitolul 4 este consacrat instrucțiunilor pentru cicluri, testări, rotații, și deplasări pe care le ilustrează prin 17 rutine utilizabile în programe proprii.

Capitolul 5 familiarizează cititorul cu modul de folosire a instrucțiunilor ce manipulează culori, sunete, și scrierea textelor; capitolul este ilustrat cu 35 de rutine.

Capitolul 6 tratează tastatura și afișajul oferind soluții pentru scanarea claviaturii, realizarea pauzelor și efecte coloristice sau de scriere prin intermediul a 10 rutine.

Capitolul 7 se referă la animație și întreruperi, prezentînd rutinele afișării, modul de elaborare a unui joc și tehnica de programare a întreruperilor.

Capitolul 8 oferă cititorului 50 de rutine care pot fi folosite în programe proprii; ele realizează sunete diverse, efecte vizuale și audiovizuale interesante precum și modalități diverse de scriere.

Capitolul 9 prezintă în detaliu dezasamblorul MONS3M21, ilustrat cu 5 rutine utile programatorilor.

© Copyright 1993 Editura APH---SRL
str. Cap. Preda nr. 12, sect. 5, 76437 București 69,
tel. 780.20.30, 780.93.97, 780.74.77

Tehnoredactare, coperta R. M. Hristev

Bun de tipar 1.3.1993 Apărut 1993

Format 59x84/16 Coli de tipar 21.5

Tiparul executat la Tipografia Garell Poligraphics SRL

1. NOȚIUNI INTRODUCTIVE

1.1. PRELIMINARII

• Microprocesorul Z80 a fost astfel proiectat încît să accepte semnale electrice notate cu:

1 dacă există semnal

0 dacă nu există semnal

și pe această bază să execute diferite operații comandate prin instrucțiuni. O instrucțiune arată, de pildă, în felul următor:

00111100

și reprezintă o instrucțiune în limbaj mașină (sau cod mașină). Deci

LIMBAJUL MAȘINĂ REPREZINTĂ O SERIE DE COMENZI PE CARE MICROPROCESORUL LE POATE ÎNTELEGE ȘI CARE SÎNT EXPRIMATE ÎN FORMĂ BINARĂ (0 ȘI 1).

Se observă că o astfel de instrucțiune este total diferită de cele cu care sîntem familiarizați la folosirea limbajului BASIC (de ex.: LET A = RND * 7). Evident că apare întrebarea "de ce este necesar limbajul mașină?".

Răspunsul este următorul: limbajul mașină are o serie de avantaje dintre care mai importante sînt următoarele:

execuție mai rapidă a programelor, ceea ce este primordial pentru multe programe cum ar fi programele de gestiune, de calcule științifice sau programele de divertisment (jocuri);

- **utilizarea mai eficientă a memoriei**;

- **programe mai scurte în memorie**, ceea ce este un criteriu de comparare a programelor cu performanțe egale.

Toate aceste avantaje sînt rezultatul direct al programării într-un

limbaj pe care microprocesorul poate să-l înțeleagă fără a-l traduce mai întâi. După cum se știe, când se lucrează în limbajul BASIC, programul monitor rezident în ROM operează după cum urmează:

- citește fiecare linie de instrucțiuni a programului;
- o traducere în limbaj mașină;
- execută fiecare instrucțiune a liniei respective;
- stochează rezultatele (dacă se cere).

Se apreciază că un calculator **consuma aproape 99% din timp pentru a traduce termenii din BASIC** și să descompună operațiile complexe în operații exprimate în sistemul de numerație binar și numai 1% pentru a le executa. Este evident că acest proces trebuie grăbit, ceea ce realizează limbajul mașină.

Principalele dezavantaje ale limbajului mașină sint:

- abstractizarea și rigiditate, deoarece lucrul implică o amplasare precisă a instrucțiunilor în programe;
 - blocarea (crahul) sistemului la erorile de sintaxă.
- Limbajul mașină este, indubitabil, greu de înțeles și de urmărit. De pildă instrucțiunea:

00100011

va fi înțeleasă numai de programatorii cu experiență. Prin urmare este necesar un limbaj intermediar care să permită înțelegerea lui de către om, dar nu și de calculator. Acesta este limbajul de asamblare care folosește abrevieri (mnemonice) pentru fiecare din aceste numere. De exemplu, instrucțiunea prezentată în limbajul mașină mai sus se scrie în limbaj de asamblare sub forma:

INC HL

unde **INC** este abrevierea de la **INCREASE** (**INC**rementează), iar **HL** este numele unui registru dublu. Pentru fiecare instrucțiune din limbajul de asamblare există o instrucțiune identică în limbajul mașină, cele două limbaje fiind echivalente. Rezultă că

LIMBAJUL DE ASAMBLARE ESTE FORMAT DIN REPREZENTAREA
ABREVIATĂ A INSTRUCȚIUNILOR LIMBAJULUI MAȘINĂ (CODUL
MAȘINĂ).

- Limbajul de asamblare poate fi convertit direct în limbaj mașină printr-un program numit asambler (ASSEMBLER) scris de programator sau de firme specializate în producere de software. Din ultima categorie

se recomandă asamblorul *GENS3M21* care va fi prezentat în această lucrare. Prin urmare:

**PROGRAMUL ASAMBLOR TRADUCE INSTRUCȚIUNILE DIN
LIMBAJUL DE ASAMBLARE (UȘOR DE FOLOSIT DE PROGRAMATOR)
ÎN LIMBAJ MAȘINĂ (ÎNȚELES DE MICROPROCESOR).**

Din cele prezentate anterior rezultă că este necesară studierea prealabilă a sistemelor de numerație, a modului cum este organizat microprocesorul și cum se folosește programul asamblor *GENS3M21*.

1.2. SISTEME DE NUMERAȚIE

Sistemele de numerație care vor fi examinate sînt sistemul zecimal, sistemul binar și sistemul hexazecimal. Toate aceste sisteme sînt **poziționale** adică poziția (rangul) unei cifre în număr conferă acestei cifre o anumită valoare.

Un sistem de numerație se scrie astfel:

$$C_p b^p + C_{p-1} b^{p-1} + C_{p-2} b^{p-2} + \dots + C_0 b^0 + C_{-1} b^{-1} + C_{-2} b^{-2} + \dots \quad (1)$$

unde: C sînt coeficienți reprezentați de cifrele sistemului;

p este poziția cifrei în număr;

$b \geq 1$ este baza sistemului de numerație și este determinată de numărul de cifre al sistemului ($b = 10$ la sistemul zecimal, $b = 2$ la sistemul binar și $b = 16$ la sistemul hexazecimal).

1.2.1. Sistemul zecimal

Cifrele folosite în sistemul zecimal ($b = 10$) sînt următoarele:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

1.2.1.1. Reprezentarea numerelor întregi fără semn

Relația (1) scrisă pentru baza $b = 10$ și în cazul numerelor fără semn capătă forma care urmează:

$$C_p 10^p + C_{p-1} 10^{p-1} + C_{p-2} 10^{p-2} + \dots + C_0 10^0 \quad (2)$$

Valoarea numărului este data de suma valorilor indicate de fiecare

cifră.

Exemplu:	numărul zecimal:	3532
	poziția (p)	3210
	mod de scriere	$3 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0 =$ $= 3000 + 500 + 300 + 2 = 3532$

1.2.1.2. Operații aritmetice

Se descriu numai operațiile de adunare și scădere. Astfel:

a) La adunarea a două numere zecimale apare noțiunea de **transport** în cazul în care adunînd cifrele aflate la o anumită poziție în cele două numere, rezultatul depășește cifra cea mai mare a sistemului (adică 9). În acest caz se va considera un transport de o unitate de pe poziția următoare spre stînga. În exemplele care urmează se indică transportul printr-o săgeată orientată spre stînga (\leftarrow):

$$\begin{array}{r} 1\ 3\ 7\ 2\ + \\ \hline 6\ 7\ 9\ 4 \\ \hline 8\ 1\ 6\ 6 \end{array} \qquad \begin{array}{r} \ 4\ 3\ 2\ + \\ \hline 8\ 4\ 7\ 9\ 9 \\ \hline 8\ 5\ 2\ 3\ 1 \end{array}$$

$\leftarrow \leftarrow$

$\leftarrow \leftarrow \leftarrow$

b) La scăderea a două numere zecimale apare împrumutul unei unități de la poziția anterioară, care în exemplele care urmează va fi indicat printr-o săgeată către dreapta (\rightarrow):

$$\begin{array}{r} \ 7\ 0\ 7\ 7\ - \\ \hline 6\ 7\ 9\ 4 \\ \hline 2\ 8\ 3 \end{array} \qquad \begin{array}{r} \ 7\ 0\ 0\ 0\ 1\ - \\ \hline \ 2\ 3\ 4 \\ \hline 1\ 6\ 9\ 7\ 6\ 7 \end{array}$$

Regulile de efectuare a operațiilor sînt aceleași în orice sistem de numerație.

1.2.2. Sistemul binar

1.2.2.1. Caracterizare

Calculatorul nu înțelege, nu stochează și nu execută operații decît asupra informațiilor care au două stări logice posibile: 0 sau 1, fals sau adevărat. Rezultă că sistemul binar este sistemul de numerație cel mai

potrivit pentru tratarea informațiilor deoarece el folosește cifrele

0 și 1

La rîndul său, sistemul binar stă la baza sistemului hexazecimal foarte util pentru reprezentarea octeților din memoria calculatorului

În cele ce urmează se vor folosi următorii termeni:

- bit pentru a desemna o cifră binară (0 sau 1);
- nibble (cuartet) pentru 4 biți;
- octet (byte) pentru 8 biți (sau cuvînt).

Relația generală (1) scrisă pentru baza $b=2$ și în cazul numerelor întregi și fără semn devine:

$$C_p 2^p + C_{p-1} 2^{p-1} + C_{p-2} 2^{p-2} + \dots + C_0 2^0$$

sau, pentru cazul unui octet:

$$C_7 2^7 + C_6 2^6 + C_5 2^5 + C_4 2^4 + C_3 2^3 + C_2 2^2 + C_1 2^1 + C_0 2^0$$

unde:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Exemplu: numărul binar 1 0 0 1 1 0 0 1
 poziția (p) 7 6 5 4 3 2 1 0
 mod de scriere $1 \cdot 2^7 + 0 + 0 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 + 0 + 1 \cdot 2^0 =$
 $= 128 + 16 + 8 + 1 = 153$

adică: $10011001_2 = 153_{10}$

Acest număr poate fi pus într-un **octet** care este dimensiunea celulei de memorie la calculatoarele compatibile ZX-SPECTRUM;

Bitul nr. 7 6 5 4 3 2 1 0 ← poziția

1	0	0	1	1	0	0	1
---	---	---	---	---	---	---	---

128 64 32 16 8 4 2 1 ← puterile 2^0 la 2^7

Numărul maxim într-un octet va fi:

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

128 64 32 16 8 4 2 1

adică $128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$

Conversia numerelor întregi pozitive din sistemul de numerație zecimal în cel binar se face împărțind numărul dat la baza 2. Rezultatul conversiei este reprezentat de numărul format din ultimul cît și resturile citite în ordine inversă.

Exemplu: $47_{10} = ?_2$

$$47 : 2 = 23 + \underline{1}$$

$$23 : 2 = 11 + \underline{1}$$

$$11 : 2 = 5 + \underline{1}$$

$$5 : 2 = 1 + \underline{0}$$

$$\text{Deci } 47_{10} = 101111_2$$

1.2.2.2. Operații aritmetice

Așa cum la adunarea zecimală se generează transport dacă suma a 2 cifre este mai mare ca 9 ducând cifra 1 spre stînga, tot așa cînd se adună 2 numere binare se generează transport dacă suma a două cifre este mai mare ca 1.

$$\begin{array}{r} \text{Exemplu: } 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ + \\ \quad \quad 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \\ \leftarrow \leftarrow \quad \leftarrow \leftarrow \end{array}$$

În cazul scăderilor se face împrumut de o unitate de la poziția anterioară.

$$\begin{array}{r} \text{Exemplu: } \quad \quad \rightarrow \rightarrow \rightarrow \rightarrow \\ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ - \\ \quad \quad \quad \quad \quad 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 1 \ 1 \ 0 \ 1 \ 0 \end{array}$$

1.2.3. Sistemul hexazecimal

Este ușor de observat că notația binară este incomodă iar cea zecimală neconvenabilă pentru afișarea adreselor locațiilor de memorie. Din aceste motive a fost adoptat sistemul de numerație hexazecimal care are baza $b = 16$ ce folosește 16 simboluri diferite pentru a reprezenta cele 16 cifre:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

unde literele semnifică valorile: **A=10, B=11, C=12, D=13, E=14 și F=15.**

Relația generală (1) scrisă pentru baza $b = 16$ și fără semn devine:

$$C_p 16^p + C_{p-1} 16^{p-1} + \dots + C_0 16^0 \quad (4)$$

Conversia numerelor întregi pozitive din sistemul de numerație zecimal în hexazecimal se obține împărțind numărul dat la baza 16.

Rezultatul conversiunii este reprezentat de numărul format din ultimul câț și resturile citite în ordinea inversă.

Exemplu: $1937_{10} = ?_{16}$

$$1937 : 16 = 121 + \underline{1}$$

$$121 : 16 = \underline{7} + \underline{9}$$

Rezultă: $1937_{10} = 791_{16}$

În tabelul 1.1 se prezintă o serie de conversii utile.

Tabelul 1.1

Zecimal	Binar	Hexa-zecimal	Zecimal	Binar	Hexa-zecimal
0	0	0	15	1111	F
1	1	1	16	10000	10
2	10	2	32	100000	20
3	11	3	64	1000000	40
4	100	4	128	10000000	80
5	101	5	256	100000000	100
6	110	6	512	1000000000	200
7	111	7	1024	10000000000	400
8	1000	8	2048	100000000000	800
9	1001	9	4096	1000000000000	1000
10	1010	A	8192	10000000000000	2000
11	1011	B	16384	2^{14}	4000
12	1100	C	32768	2^{15}	8000
13	1101	D	65535	2^{16-1}	FFFF
14	1110	E			

Din examinarea tabelului se constată următoarele:

- un *cuartet* (4 biți) necesită o cifră hexazecimală;
- un *octet* (8 biți) necesită două cifre hexazecimale;
- un *număr* pe 16 biți necesită 4 cifre hexazecimale

Prin urmare sistemul de numerație hexazecimal este mai concis.

$$\text{Ex.1: } \underbrace{1001}_9 \underbrace{1001}_9_2 = 99_{16} = 9 \cdot 16^1 + 9 \cdot 16^0 = 144 + 9 = 153_{10}$$

$$\text{Ex.2: } \underbrace{0100}_4 \underbrace{0000}_0 \underbrace{0000}_0 \underbrace{0000}_0_2 = 4000_{16} = 4 \cdot 16^3 = 16384_{10}$$

$$\text{Ex.3: } \underbrace{1011}_B \underbrace{0000}_0 \underbrace{0101}_5 \underbrace{1111}_F_2 = B05F_{16} = 11 \cdot 16^3 + 0 + 5 \cdot 16^1 + 5 + 5 \cdot 16^0 = 45151_{10}$$

1.2.4. Notăția zecimal codificat binar

În unele cazuri este preferabil să se efectueze calculele folosind numere zecimale, fără a fi transformate în binar sau hexazecimal. Astfel, cifrele hexazecimale necesită o ajustare (corectare) pentru a se elimina valorile de la A=10 la F=15. Dacă aceste simboluri apar, **se adaugă cifra 6** și rezultă o valoare ajustată. De pildă numărul hexazecimal C (care este egal cu 12 în sistemul de numerație zecimal) adunat cu 6 face 18 zecimal. Acest mod de reprezentare se numește **zecimal codificat binar** (BCD). Microprocesorul Z80 realizează ajustarea cu instrucțiunea

DAA

(ajustarea zecimală a registrului acumulator), care adună cifra 6 la grupurile de 4 biți care depășesc cifra 9.

1.2.5. Notăția pozitivă și negativă

Prin convenție, bitul numărului 7 (notat b7) reprezintă semnul numărului și anume:

- când are valoarea 0 numărul este pozitiv
- când are valoarea 1 numărul este negativ.

Poziția	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	0								1							

Pentru a transforma un număr pozitiv într-un negativ trebuie inversati toti bitii numărului binar (operație numită **complementare**) și apoi se adaugă 1 (operație numită **complement față de 2**).

Exemplu:	0	0	0	0	0	1	0	1	(este +5)
	1	1	1	1	1	0	1	0	(inversat)
	1	1	1	1	1	0	1	1	(+1 și este -5)
	0	0	0	0	0	1	0	0	(inversat)
	1	1	1	1	1	0	1	1	(+1 și este +5)

Cel mai mare număr care poate fi păstrat într-o locație de memorie folosind convenția de semne indicată (0 pe bitul b7) este

7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1

128 64 32 16 8 4 2 1 sau
 $64 + 32 + 16 + 8 + 4 + 2 + 1 = 127_{10}$

iar cel mai mic număr negativ (cu 1 pe bitul b7) are valoarea

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0

128 64 32 16 8 4 2 1 sau = -128_{10}

Deci o celulă de memorie (o locație) poate stoca numere între -128 și $+127$.

Microprocesorul Z80 are două instrucțiuni care pot fi folosite pentru aceste operații și anume:

CPL care complementează registrul acumulator;

NEG care face negativ conținutul registrului acumulator prin complementare și adunarea cifrei 1, într-o singură operație.

În mod analog se poate demonstra că două celule de memorie pot conține un număr între -32768 și $+32767$.

1.3. STRUCTURA DE BAZĂ A UNUI MICROPROCESOR

1.3.1. Generalități

• Microprocesorul reprezintă unitatea centrală de calculator (CPU - Central Processing Unit), încorporată într-o capsulă de circuit integrat. El citește instrucțiunile unui program aflat într-un bloc de memorie externă, le decodifică și apoi execută comenzile indicate în codul instrucțiunii lor (fig.1.1.a).

• Pentru a citi din blocul de memorie externă codul instrucțiunii de executat, microprocesorul trebuie să genereze o adresă pe care o va pune la dispoziția memoriei pînă cînd va apare data cerută din celula de memorie selectată pe baza acestei adrese. Pentru ca pe durata întregii

opeații de citire microprocesorul să mențină starea liniilor de adrese, el trebuie să aibă un registru tampon de adrese AB (Address Buffer).

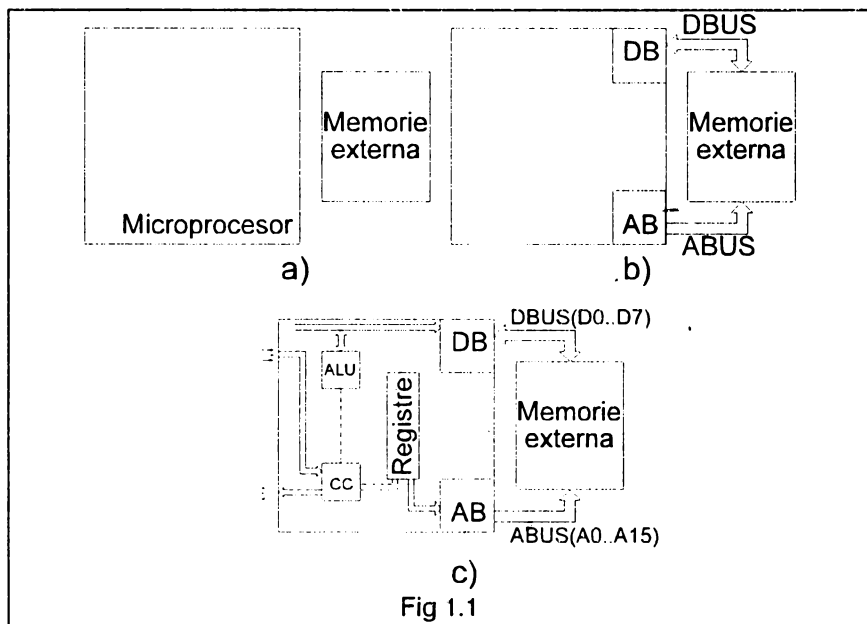


Fig 1.1

Informația codificată citită din memoria externă se va depune temporar într-un registru denumit registru tampon de date DB (Data Buffer). Acesta este un registru bidirecțional de 8 biți care delimitează interiorul microprocesorului de exterior.

Linile electrice pe care se va genera cuvântul binar de adresă poartă denumirea de magistrala de adrese ABUS (Address Bus), iar liniile electrice dedicate datelor citite/scrise în memorie formează magistrala de date DBUS (Data Bus) - fig.1.1.b.

- Se poate întâmpla instrucțiunea citită din memoria externă și depusă temporar în registrul de date DB să aibă următoarea semnificație (4): "citește conținutul locației de memorie a cărei adresă este cu 4 mai mare decât adresa curentă din registrul de date AB, adaugă la aceasta valoarea 5 și rescrie rezultatul în aceeași locație de memorie". Pentru a efectua această cerință, microprocesorul are nevoie de o unitate aritmetică care, în aritmetica binară, descrie operațiile folosind funcții logice. Din acest

motiv numele utilizat este unitate aritmetică și logică ALU (Arithmetic Logic Unit). Evident că pentru a executa cererile formulate în comanda anterior menționată microprocesorul trebuie să mai dispună și de o unitate de comandă CC (Comand Circuit), a cărei activitate este pilotată de un semnal de ceas cu frecvența de ordinul megahertzilor (la Z80A frecvența $\nu = 3,5$ MHz).

Comenzile de execuție pe care microprocesorul le dă sint transmise prin semnale electrice numite semnale de comandă, iar semnalele prin care culege informații cu privire la starea componentelor din sistemul calculatorului poartă denumirea de semnale de stare.

• În ipoteza că următoarea instrucțiune va utiliza rezultatul instrucțiunii precedente pentru a efectua o nouă operație aritmetică, atunci valoarea calculată trebuie citită din nou din memorie. Accesul suplimentar la memorie se economisește dacă microprocesorul va folosi câteva elemente de memorare temporară a datelor sau adreselor de memorie, numite **registre** și care se clasifică în registre speciale și registre de uz general (fig.1.1.c).

Ele pot fi asemuite cu niște căsuțe (sau sertare) în care se păstrează numere cuprinse între 0 și 255 (fig.1.2.)

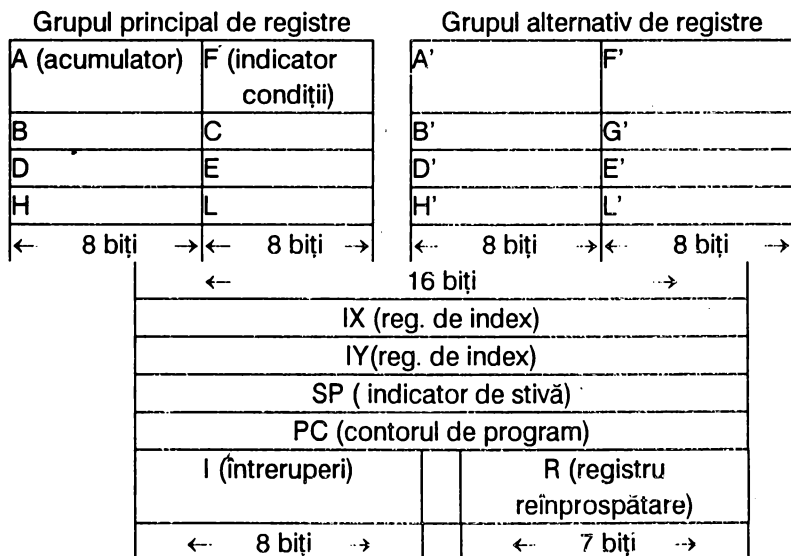


Fig 1.2

- Observatii: 1) Magistrala de date DBUS are 8 linii (notate D0 la D7) care pornesc din registrul tampon de date DB al microprocesorului și au rolul de a asigura transferul de date între microprocesor, memorie și dispozitivele de intrare/ieșire (I/E). Această magistrală este bidirecțională deoarece intră date când se efectuează o citire din memorie sau dispozitivele de I/E și ies date la efectuarea unei scrieri. Cel mai puțin semnificativ bit al octeților circulă pe linia D0.
- 2) Magistrala de adrese ABUS are 16 linii (notate A0 la A15) cu originea în registrul tampon de adrese AB al microprocesorului; ea este unidirecțională deoarece adresele ies din microprocesor pentru a fi transmise la circuitele de memorie și dispozitivele de I/E. Linia A0 conține bitul cel mai puțin semnificativ al adreselor.
- 3) Unitatea aritmetică și logică ALU poate să execute numai două operații aritmetice: adunarea și scăderea a două numere binare de câte 8 biți. Operațiile logice acționează de asemenea pe cuvinte de câte 8 biți cum sînt: și, sau, sau exclusiv, comparație și complementare (față de 1 și 2). Această unitate efectuează scăderile transformîndu-le în adunări prin transformarea numerelor de scăzut (negative) în complementul lor față de 2.

1.3.2. Registre speciale

1.3.2.1. Contorul program PC (Program Counter)

Acesta este un registru dublu de 16 biți destinat să memoreze adresa instrucțiunii ce urmează a fi executată. După ce se citește codul instrucțiunii curente din memorie (operația fetch), conținutul acestui registru este automat încrimentat cu 1. În acest fel se asigură secvențialitatea în execuția șirului de instrucțiuni în ordinea crescătoare a adreselor începînd de la adresa 0. Instrucțiunile de salt înscriu direct conținutul registrului PC cu valoarea dorită de programator, executîndu-

se programe a căror adresă de start diferă de 0 sau ramificații în programe în funcție de rezultatul unor calcule sau a unor evenimente detectabile prin program (salturi condiționate). Memoria adresabilă direct are $2^{16} = 65536$ locații de memorie distincte între 0 și 65535 (în hexazecimal între 0000 și FFFF).

1.3.2.2. Indicatorul de stivă SP (Stack Pointer)

Este un registru dublu cu lungimea de 16 biți care poate adresa întregul spațiu de memorie de la 0 la 65535. Pentru înțelegerea rolului său se menționează că în cursul execuției unui program apare frecvent necesitatea de a elibera unul sau mai multe registre cu scopul ca în ele să se înmagazineze temporar date noi. Informația precedentă din aceste registre este salvată într-o zonă de memorie din RAM numită **stivă**, în care legea care guvernează mișcarea datelor se numește *LIFO* (Last In First Out), adică "*Ultimul intrat este primul care pleacă*". Se observă ușor că dacă există un registru de adresă care automat incrementează/decrementează la operațiile de scriere și citire in/din memorie, atunci el poate controla toate operațiile de salvare/restaurare în stivă. Un asemenea registru de adresare a stivei este indicatorul de stivă SP care, la microprocesorul Z80, organizează o **stivă descrescătoare** și anume:

- la fiecare **salvare** se înscriu în memorie 2 octeți și conținutul indicatorului de stivă este decrementat cu 2:

$$SP = SP - 2$$

la fiecare **restaurare** se citesc din memorie 2 octeți și conținutul indicatorului de stivă este incrementat cu 2:

$$SP = SP + 2$$

Stiva este imperios necesară la apelarea subrutinelor și, mai ales, în cazul apelărilor multiple. O subrutină poate fi apelată dintr-un program principal folosind instrucțiunea

CALL adresă

unde "**adresă**" este adresa de început a subrutinei. Când se întâlnește această instrucțiune execuția programului principal este abandonată și se trece la execuția subrutinei pînă la întâlnirea instrucțiunii

RET

care marchează sfîrșitul subrutinei, cînd se revine în programul principal

la instrucțiunea imediat următoare instrucțiunii **CALL**. Odată cu executarea instrucțiunii **CALL**, adresa instrucțiunii următoare din programul principal este salvată în vârful stivei, iar la execuția instrucțiunii **RET** această valoare este citită din vârful stivei și încărcată în contorul program PC pentru a se relua firul întrerupt.

Se menționează că folosirea acestui registru trebuie făcută cu multă atenție întrucît orice eroare de programare poate conduce la crah-ul programului.

1.3.2.3. Registrul acumulator A (Acumulator)

Este un registru de 8 biți care, pe lângă memorarea unui octet, este **implicat în toate operațiile aritmetice și logice** deoarece unul din cei doi parametri asupra cărora se efectuează operațiile se află obligatoriu în acumulator, iar rezultatul operației se înscrie tot în acumulator.

1.3.2.4. Registrul indicatorilor de condiție F (Flag-fanion)

Conform numelui său, acest registru de 8 biți are rolul de a semnala, fiecare bit al său avînd semnificația lui iar starea unui bit (dată prin cifra 0 sau 1) indicînd satisfacerea/nesatisfacerea unor condiții date.

b7	b6	b5	b4	b3	b2	b1	b0
S	Z	-	H	-	P/V	N	C

Din acest motiv flagurile constituie indicatori de condiție și ei furnizează programatorului și unității de comandă CC a microprocesorului, informații privind natura rezultatului unei operații aritmetice sau logice efectuate. Din starea biților dedicați ai registrului F se poate afla dacă rezultatul operației efectuate este zero sau nu, dacă el este un număr pozitiv sau negativ etc. Semnificația biților este următoarea:

a) **Flagul S** (*Sign-semn*), care apare pe poziția cea mai semnificativă (bitul 7 notat b7), memorează semnul cu următoarea convenție:

$S = 0$ dacă numărul este pozitiv

$S = 1$ dacă numărul este negativ.

Starea flagului S se testează de programator de regulă prin două instrucțiuni de salt condiționat care permit ramificarea programului în

funcție de starea indicatorului de semn și anume:

JP P, nn (jump if plus-salt dacă pozitiv), care efectuează saltul la adresa nn dacă $S = 0$

JP M, nn (jump if minus-salt dacă negativ), care efectuează saltul la adresa nn dacă $S = 1$

b) **Flagul Z (Zero)** apare pe poziția bitului 6 (notat b6) și poate avea

- valoarea 1 dacă rezultatul operației aritmetice/logice este zero;

- valoarea 0 dacă rezultatul operației aritmetice/logice este nenul.

Acest flag se testează de programator prin instrucțiunile de salt condiționat:

JP NZ,nn (jump if not zero-salt dacă diferit de zero) - efectuează saltul la adresa nn dacă $Z = 0$;

JP Z, nn (jump if zero-salt dacă zero) efectuează saltul la adresa nn dacă $Z = 1$

Flagul Z este folosit și de circuitul de comandă CC al microprocesorului la execuția instrucțiunilor repetitive prin:

- instrucțiunile de comparație **CPDR** (Compare decrement repeat-compară decrementează repetă) și **CPIR** (Compare increment repeat-compară incrementează repetă), caz în care instrucțiunea care compară conținutul acumulatorului A cu celule de memorie aflate la adrese descrescătoare/crescătoare se termină dacă $Z = 1$, adică în cazul egalității numerelor comparate;

- instrucțiunile de transfer de blocuri de date la periferice **INDR**, **INIR** și **OTDR**, **OTIR** se termină de asemenea când $Z = 1$ (conținutul registrului B devine zero).

c) **Flagul H (Half carry transport la jumătate)**, care apare pe poziția bitului 4 (notat b4), reprezintă transportul produs în cursul unei operații aritmetice de la bitul 3 spre bitul 4 al acumulatorului, respectiv împrumutul de la bitul 4. Indicatorul este folosit și de instrucțiunea de ajustare zecimală **DAA** pentru a corecta rezultatul unei operații de adunare sau scădere în notația zecimal codificat binar **BCD**.

H	ADD	SUBTRACT
1	Există transport de la bitul 3 la bitul 4	Există împrumut de la bitul 4
0	Nu există transport	Nu există împrumut

d) **Flagul P/V (Parity/Overflow paritate/depășire)**, care apare pe

poziția bitului 2 (notat b_2), indică pe de o parte paritatea numărului din acumulator sau depășirea de domeniu, iar pe de altă parte este un detector al registrului dublu BC (dacă conținutul registrului dublu BC=0 atunci $P/V=0$), ceea ce este o condiție de terminare a repetițiilor, iar dacă BC=0 atunci $P/V=1$).

Cînd flagul P/V indică paritatea el are valorile $P/V=0$ dacă totalul cifrelor din acumulatorul A este un număr impar și respectiv $P/V=1$ cînd acesto total este un număr par.

Atunci cînd flagul indică depășirea domeniului $[-128, +127]$ are valorile $P/V=1$ dacă a existat depășire, respectiv $P/V=0$ dacă nu a existat depășire.

Testarea flagului P/V se face cu instrucțiunile de salt condiționat:

JP PO, nn (jump if parity odd) se execută salt la adresa nn dacă
 $P/V=0$

JP PE, nn (jump if parity even) se execută salt la adresa nn dacă
 $P/V=1$

Se precizează că indiferent de faptul că P/V indică paritatea sau depășirea, mnemonica lui face referire la paritate; aceasta reprezintă o dificultate în programare, deoarece programatorul trebuie să știe ce anume indică flagul P/V în momentul cînd folosește una din instrucțiunile de salt condiționat.

e) **Flagul N** (Nibble), care apare pe poziția bitului 1 (notat b_1), memorează tipul ultimei operații aritmetice efectuate și anume:

$N=1$ dacă operația a fost adunare;

$N=0$ dacă operația a fost scădere.

Nu se poate testa valoarea acestui indicator.

f) **Flagul C_i** (Carry - transport), care apare pe poziția cea mai puțin semnificativă (bitul 0 notat b_0), este afectat de operațiile aritmetice/logice și de clasa operațiilor de rotire/deplasare octet. Acest flag are înscrisă valoarea 1 ori de cîte ori apare un transport de la cifra cea mai semnificativă în sus (cazul unei adunări în aritmetica fără semn pentru numere între 0 și 255 cînd rezultă un număr mai mare ca 255). La scădere $C_i=1$ dacă scăzătorul este mai mare decît descăzutul. Flagul poate fi testat prin instrucțiunile de salt condiționat

JP NC, nn (jump if not carry) salt la adresa nn dacă $C_i=0$

JP C, nn (jump if carry) salt la adresa nn dacă $C_i=1$

Spre deosebire de ceilalți indicatori, flagul C_j poate fi modificat cu instrucțiunile:

SCF (*Set Carry Flag*) care cauzează $C_j = 1$

CCF (*Complement Carry Flag*) care cauzează complementarea conținutului C_j .

De asemenea, ca și celelalte flaguri testabile prin instrucțiuni de salt condiționat, flagul C_j poate dirija apeluri și reveniri condiționate din subrutine folosind instrucțiunile **CALL NC**, *nn*; **CALL C**, *nn*; **RET NC**; **RET C**.

Observatie: biții 3 și 5 (notați b3 și respectiv b5) ai registrului F sînt lipsiți de semnificație conținutul lor variind imprezibil în timpul funcționării microprocesorului.

1.3.2.5. Regitul vectorilor de întreruperi I (Interrupt register)

Este un registru cu lungimea de 8 biți care servește în modul de întreruperi 2 (notat *IM2*) la dirijarea (identificarea) sursei de cerere a întreruperilor. Prin întrerupere se înțelege fenomenul la apariția căruia microprocesorul abandonează programul în curs de rulare la cererea unui eveniment extern, deserveste evenimentul extern executînd un program dedicat special, după care se reîntoarce la programul abandonat reluînd execuția din locul unde fusese suspendată. Programatorul accede registrul I prin intermediul acumulatorului A folosind instrucțiunile:

LD I,A înscrie o valoare în registrul I;

LD A,I citește valoarea conținută în registrul I.

Observatii: 1) În blocul funcțional al circuitului de comanda CC apar:

- a) Bistabilii de validare/inhibare a sistemului de întreruperi IFF1 și IFF2. Dacă IFF1 și IFF2 au valoarea 1 atunci sistemul de întreruperi este validat și microprocesorul acceptă cererile de întrerupere, iar dacă au valoarea 0 sistemul de întreruperi este inhibat. În acest fel bistabilul IFF1 semnalează starea de validare/inhibare a întreruperilor, iar IFF2 stochează temporar valoarea lui IFF1 pe timpul tratării întreruperii nemascabile. Poziționarea celor doi bistabili pe 1 sau pe

0 se face cu instrucțiunile **EI**, respectiv **DI**.

b) Bistabilii în mod de întrerupere IMFa și IMFb care codifică unul din cele trei moduri de lucru în întreruperi care se face cu instrucțiunile **IM0**, **IM1** și **IM2**.

2) Microprocesorul Z80 acceptă două semnale de întrerupere:

- întrerupere nemascabilă (NMI), când Z80 răspunde într-un singur mod;
- întrerupere mascabilă (INT), când Z80 are trei moduri de tratare.

• **Întrerupere nemascabilă** este prioritară față de cea mascabilă. La punerea sub tensiune a calculatorului bistabilii *IFF1* și *IFF2* sînt forțați pe 0 ceea ce este echivalent cu invalidarea întreruperilor și în această stare microprocesorul nu acceptă întreruperi mascabile. Întreruperile sînt validate prin poziționarea bistabililor *IFF1* și *IFF2* pe 1 folosind instrucțiunea **EI**, iar, orice altă întrerupere va fi servită numai după execuția instrucțiunii care urmează după **EI**. Evident că atunci când microprocesorul acceptă o întrerupere *IFF1* și *IFF2* sînt aduși pe 0 pentru a inhiba acceptarea altor întreruperi pînă la o nouă instrucțiune **EI**. La execuția unor instrucțiuni **LD A,I** sau **LD A,R** starea lui *IFF2* este transmisă în flagul de paritate *P/V* permițîndu-se testarea sau memorarea ei și, implicit, refacerea prin program a valorii inițiale a bistabilului *IFF1* cu instrucțiunea **RETN**. Se menționează că întreruperea nemascabilă nu poate fi invalidată prin program.

• Microprocesorul Z80 poate fi programat pentru a răspunde la întreruperile nemascabile într-unul din modurile 0,1 sau 2 care sînt memorate cu ajutorul bistabililor IMFa și IMFb:

IMFa	IMFb	
0	0	Modul de întrerupere 0
0	1	Neutilizat
1	0	Modul de întrerupere 1
1	1	Modul de întrerupere 2

a) În modul 0 dispozitivul periferic care întrerupe plasează pe magistrala de date DBUS în ciclul de tratare a întreruperii orice instrucțiune, ideea fiind deci că instrucțiunea următoare este furnizată de dispozitivul care întrerupe. În general, această instrucțiune este o

instrucțiune restart (care realizează apeluri la subrutine plasate la 8 locații fixe din "pagina zero" din ROM). La inițializarea, microprocesorului intră automat în modul 0.

b) *Modul 1* este similar cu modul de răspuns la întreruperile nemascabile, diferența principală fiind execuția restartului la locația

0038₁₆

în loc de 0066₁₆

c) *Modul 2* este modul cel mai puternic de răspuns al microprocesorului care, cu un singur octet furnizat de dispozitivul care întrerupe, se execută un apel indirect la orice locație de memorie. În prealabil programatorul trebuie să aibă scrisă o tabelă cu adresele de început ale fiecărei rutine de serviciu care poate fi localizată în orice zonă de memorie RAM. La acceptarea întreruperii, microprocesorului formează un pointer de 16 biți cu ajutorul căruia ia din tabela de adrese valoarea adresei rutinei de serviciu corespunzătoare dispozitivului care întrerupe. Cei mai semnificativi biți ai pointerului sînt dați de conținutul registrului I încărcat anticipat. Cei mai puțin semnificativi 8 biți sînt generați de periferic, cu mențiunea că ultimul bit trebuind să fie 0, sînt necesari de fapt numai 7 biți, fapt care determină ca adresele de început ale rutinelor de serviciu să fie plasate în tabelă întotdeauna la adrese pare.

1.3.2.6. Registrul de reîmprospătare a memoriei dinamice R (Refresh)

Acesta este un registru cu lungimea de 7 biți care asigură reîmprospătarea memoriilor RAM dinamice printr-o numărătoare ciclică de la 0 la 127. Memoriile RAM dinamice păstrează informația în locații de memorie al căror element de memorare este un condensator cu capacitate foarte mică ($\leq 10^{-15}$ F), care poate pierde informația în timp datorită curenților de scurgere prin dielectric. Întrucît timpul limită în care nu este periclitată integritatea informației stocate este decirca 2 ms, aceste condensatoare trebuie repolarizate la valoarea lor inițială din 2 în 2 ms, operație denumită *reîmprospătarea memoriei dinamice*.

Conținutul registrului R poate fi înscris și citit prin intermediul acumulatorului A cu instrucțiunile

LD R, A - înscrie registrul R

LD A, R - citește conținutul resgirului R.

La inițializarea calculatorului registrul R este încărcat cu 0.

1.3.3. Registre de uz general

a) Registrele B și C sînt registre generale principale de 8 biți, al căror conținut este tratat de o multitudine de instrucțiuni. Figurarea lor pe aceeași linie în fig. 1.2. nu este întîmplătoare deoarece ele se pot atașa formînd un registru pereche (dublu) BC avînd lungimea de 16 biți. În acest caz B este octetul cel mai semnificativ (superior) iar C (octetul cel mai puțin semnificativ (inferior).

b) Registrele D și E sînt similare registrelor B și C, putînd deci forma un registru dublu DE de 16 biți cînd D este octetul cel mai semnificativ iar E octetul cel mai puțin semnificativ.

c) Registrele H și L diferă de cele anterioare B,C,D,E doar prin faptul că sînt implicate într-un număr mai mare de instrucțiuni. cînd se formează registrul dublu HL, registrul H ocupă octetul superior, iar L cel inferior (adică $H*256+L$). Cel mai mare număr care poate fi păstrat într-un registru pereche este $nn = 255*256 + 255 = 65535$.

Întrucît operațiile aritmetice și logice lucrează cu doi operanzi, unul dintre aceștia este obligatoriu stocat în acumulatorul A iar celălalt se află de obicei într-unul din registrele de uz general B,C,D,E,H,L. Există însă și posibilitatea ca cel de al doilea operand să fie locat în memorie la o adresă oarecare, caz în care adresa celulei (locației) unde este stocat operandul se înscrie în registrul dublu HL. Prin aceasta registrul dublu HL devine principalul instrument de adresare indirectă a unor operanzi locați în memorie la adrese cunoscute. Se precizează că nu există operații aritmetice și logice care să folosească pentru adresarea operandului stocat în memorie conținutul celorlalte registre duble BC și DE.

d) Registrele IX și IY se folosesc numai pentru stocarea unor adrese de memorie. Ele au fost concepute pentru a fi utilizate atunci cînd se efectuează operații aritmetice/logice asupra unor date așezate la adrese succesive de memorie, formînd astfel un tabel; prin urmare, registrele IX și IY conțin adresa de început a tabeli. Data dorită se va localiza

adăugind la adresa de început a tabelii (numită adresă de bază) un indice care specifică numărul de ordine al datei numit deplasament. Acest indice se specifică explicit în conținutul instrucțiunii, valoarea ei fiind adunată la adresa de bază în cursul execuției instrucțiunii formându-se adresa efectivă a celulei de memorie.

De pildă instrucțiunea

ADD A, (IX+5)

adună conținutul registrului A cu cel de-al cincelea element din tabelul de date care începe la adresa conținută în registrul IX. Evident, conținutul registrului IX nu se va modifica pe parcursul execuției instrucțiunii.

Se menționează că datele din tabelele adresate cu registrele index IX și IY au lungimea de 1 octet; deci numărul maxim de date este 256.

Tehnica de adresare a operanzilor locați în memorie folosind registrele IX și IY se numește adresare indexată.

e) Registrele A', B', C', D', E', F', H', L' sînt dublurile registrelor A, B, C, D, E, F, H, L al căror conținut poate fi interschimbat între ele prin instrucțiunea **EXX**. Se menționează că toate instrucțiunile care implică registrele de uz general B,C,D,E,H,L acționează asupra conținutului registrelor primare, iar pentru a opera cu conținutul registrelor secundare acest conținut trebuie transferat în registrele primare cu instrucțiunea **EXX**.

1.4. ASAMBLORUL GENS3M21

1.4.1. Prezentare

Programul asamblor *GENS3M21* a fost realizat de firma *HISOFT* în anul 1983; el se recomandă să fie încărcat la adrese joase (de ex: 24064) cu comenzile:

CLEAR adr-1; LOAD "GENS3M21" CODE adr

unde "adr" este adresa zecimală la care se va localiza programul.

După încărcarea sa în memoria calculatorului, asamblorul se lansează cu comanda

RANDOMIZA USR adr

Orice reintrare în asamblor din BASIC se va face cu comenzile:

RANDOMIZE UŞR adr+56 (cu ştergerea textului sursă - *intrare rece*)

RANDOMIZE USR adr+61 (cu păstrarea textului sursă - *intrare caldă*).

De exemplu, dacă programul GENS a fost încărcat la adresa *adr*=24064, atunci:

- după încărcare se va lansa cu comanda *RANDOMIZE USR 24064*;
- la reintrarea rece comanda va fi *RANDOMIZE USR 24120*;
- la reintrarea caldă comanda va fi *RANDOMIZE USR 24125*.

La prima comandă de lansare în execuție a asamblorului acesta emite mesajul

Buffer Size?

Se va răspunde **taștind** un număr întreg între 0 și 9 urmată de *ENTER*, sau numai *ENTER* care este echivalentă cu asumarea valorii 4. Aceste cifre reprezintă dimensiunea buffer-ului de incluziune în unități de 256 octeți. Dacă se tinde spre minimizarea spațiului ocupat de GENS, atunci se va răspunde cu 0 urmat de *ENTER*, caz în care se asumă un spațiu minim de 64 octeți.

După aceste operații apare prompterul ">", indicând activizarea editorului.

1.4.2. Modul de lucru

1.4.2.1. Generalități

- Programul GENS3M21 este un asamblor Z80 rapid în două etape care assemblează tot setul de mnemonice standard Z80. În urma invocării unei asamblări (folosind comanda A), programul cere specificarea dimensiunii tabelii de simboluri afișând mesajul

Table size.

Acesta va fi spațiul alocat tabelii în timpul asamblării. Dacă la acest mesaj se răspunde cu *ENTER*, atunci GENS va forma o tabelă de simboluri în funcție de dimensiunile textului. În cazul folosirii opțiunii

include

se recomandă specificarea unei dimensiuni mari pentru tabela de simboluri.

În continuare *GENS* întreabă utilizatorul dacă solicită opțiunile disponibile afișând mesajul

Options:

Acestea se introduc în cod zecimal, iar în cazul mai multor opțiuni simultane se va tasta suma lor. Opțiunile sînt următoarele:

- 1 (produce listarea tabelii de simboluri)
- 2 (nu generează cod obiect)
- 4 (nu produce listarea asamblării)
- 8 (produce listarea pe printer)
- 16 (plasează codul obiect după tabela de simboluri)
- 32 (decuplează rutina de testare a zonei de asamblare a codului obiect, fapt util la sporirea vitezei de asamblare)

Se precizează că în cazul opțiunii 16, directiva de asamblare *ENT* își pierde efectul; în acest caz pentru localizarea codului obiect se recurge la comanda editorului X care afișează locația afișitului textului (al doilea număr afișat), la care se adaugă dimensiunea tabelii de simboluri + 2).

• Asamblarea se face în două etape: în prima etapă *GENS* detectează eventualele erori și compilează tabela de simboluri, iar în etapa a doua generează codul obiect (textul sursă), evident dacă nu s-a utilizat opțiunea 2. În prima etapă nu se produce listarea decît în cazul detectării unei erori, situație în care se afișează linia respectivă împreună cu un cod în dreptul erorii sesizate și asamblarea este sistată. Apăsînd tasta E se intră în editor, orice altă comandă producînd asamblarea de la linia următoare. La sfîrșitul primei etape apare mesajul

Pass 1: errors: nn

Dacă s-au detectat erori asamblarea este oprită înainte de a se trece la etapa a doua. Dacă s-au depistat etichete ce nu au fost declarate în cîmpul de etichete va apare mesajul

** WARNING * 'label' absent*

care se va repeta pentru fiecare etichetă nedeclarată.

Codul obiect se generează în etapa a doua cînd se afișează și textul respectiv (dacă nu s-a folosit opțiunea 4).

Singurele erori care pot apare în etapa a doua sînt

** ERROR 10 * sau BAD ORG.*

Aceasta din urmă apare cînd codul obiect urmează să corupă *GENS*-ul, fila de text sau tabela de simboluri.

La sfîrșitul asamblării apare mesajul

Pass 2 : errors : nn

urmat de atenționări privind etichetele absente. În continuare apar mesajele

Table used : xxxx from yyyy

Executes : nnnnn

unde '*nnnnn*' reprezintă adresa de lansare a codului obiect.

Rularea codului obiect se face cu comanda editorului R.

În final, dacă s-a folosit opțiunea 1, apare o listă alfabetică a etichetelor împreună cu valorile asociate. Din acest motiv se recomandă ca opțiunea folosită să fie 5.

1.4.2.2. Formatul instrucțiunilor asamblorului

Fiecare linie generată de *GENS* are următorul format, unde comentariile sînt opționale:

<u>ETICHETA</u>	<u>MNEMONICA</u>	<u>OPERANZI</u>	<u>COMENTARIII</u>
START	LD	HL, număr	ia valoarea număr

- Etichetele sînt simboluri reprezentînd o informație de max.16 biți, constituite din max.6 caractere, primul fiind obligatoriu o literă (ex: loop, Loop, L(1),a,LDIR)
- Menemonicele si operanzii sînt cele care vor fi prezentate în capitolele consacrate instrucțiunilor Z80.

Observatii:

- 1) Pentru valorile numerice care nu sînt scrise în codul zecimal se vor folosi înaintea numărului respectiv următoarele simboluri:
 - # constantă hexazecimală
 - % constantă binară.
- 2) Expresiile sînt evaluate de la stînga la dreapta, iar dacă expresia este scrisă între paranteze aceasta este considerată ca reprezentarea unei adrese în memorie. De ex: **LD HL, (loc+5)** semnifică încărcarea registrului dublu HL cu valoarea de 16 biți conținută în locația loc + 5.

1.4.2.3. Directivele de asamblare

Asamblorul GENS recunoaște unele pseudonemionice specifice acestui program și care nu au efect asupra microprocesorului în timpul rulării; ele au rolul de a ghida asamblorul.

Pseudonemionicele sînt asamblate în mod identic cu instrucțiunile executabile; ele pot fi precedate de o etichetă (necesară la directiva EQU) și urmate de un comentariu. Directivele disponibile sînt următoarele:

- 1) *ORG expresie*
 - fixează numărătorul de locații la valoarea '*expresie*'. Dacă nu s-au folosit simultan opțiunile 2 și 16 iar *ORG* intenționează să corupă GENS3, fila de text sau tabela de simboluri, atunci apare mesajul 'Bad *ORG*' și asamblarea este întreruptă.
 - asociază unei etichete valoarea '*expresie*'
- 2) *EQU expresie*
- 3) *DEFB expresie, expresie,...*
 - fiecare expresie trebuie să fie de 8 biți (numere între 0 ... 255); octetul de la adresa curentă din numărătorul de locații este încărcat cu valoarea '*expresie*' iar numărătorul de locații este incrementat cu 1 (se repetă pentru fiecare '*expresie*').
- 4) *DEFW expresie, expresie, ...*
 - încarcă cuvîntul (2 octeți adică numere între 0 ... 65535) de la adresa curentă din numărătorul de locații cu valoarea '*expresie*' iar numărătorul de locații este incrementat cu 2. Întîi se încarcă octetul mai puțin semnificativ (se repetă pentru fiecare '*expresie*')
- 5) *DEFS expresie*
 - incrementează numărătorul de locații cu valoarea '*expresie*'

- echivalent cu rezervarea unui bloc de memorie cu dimensiunea 'expresie'.
- 6) *DEFM "s"*
- determină ca un număr de n octeți din memorie să conțină echivalentul în cod ASCII a șirului "s" de lungime n . Teoretic n poate fi cuprins între 1 și 255 inclusiv, dar în practică lungimea șirului este limitată de editor.
- 7) *ENT expresie*
- pune adresa de execuție a codului obiect asamblat la valoarea 'expresie'.

GENS3 mai dispune de 3 pseudomonemonice condiționale care permit includerea sau excluderea unor părți din textul sursă în procesul de asamblare;

- 8) *IF expresie*
- evaluează 'expresie' dacă rezultatul este 0 asamblarea liniilor subsecvente este oprită pînă la întâlnirea uneia din pseudomonemonicele ELSE sau END, iar dacă valoarea lui 'expresie' este nenulă asamblarea continuă în mod normal
- 9) *ELSE*
- are rolul de a cupla/decupla asamblorul; dacă asamblorul a fost cuplat înainte de primul ELSE, acesta va fi decuplat și invers.
- 10) *END*
- efectuează cuplarea asamblorului.

Se menționează că pseudomonemonicele condiționale nu pot fi incluse una în alta deoarece asamblorul nu verifică acest element: orice încercare în acest sens va da rezultate imprevizibile.

1.4.2.4. Comenzile editorului

- $N\ n, m$ (numerează filele de text începînd cu linia n și cu un increment m)
- $E\ n$ (editează linia cu numărul n)
- $D\ n, m$ (șterge toate liniile între n și m inclusiv; ștergerea unei singure linii se face egalînd n cu m , sau cu Dn urmată de ENTER)

1.4.2.5. Comenzile pentru asamblare și rulare a codului generat

- $I\ n, m$ (se inserează instrucțiunile începînd cu linia n și incrementul m ; ieșirea se face cu CS și 1)
- A (cauzează asamblarea textului începînd cu prima linie)
- R (determină executarea programului obiect)
- B (determină reîntoarcerea în BASIC, cînd codul mașină poate fi rulat cu comanda RANDOMIZE USR $nnnnn$, unde $nnnnn$ este adresa specificată în mesajul Executes:nnnnn).

1.4.2.6. Comenzile de bandă

Textul poate fi salvat/încărcat de pe bandă cu comenzile:

- O, \textit{nume} (salvează codul mașină utilizabil în programele utilizatorului)
- Pn, m, \textit{nume} (salvează codul sursă între liniile n și m cu numele dat)
- G, \textit{nume} (încarcă de pe bandă textul sursă cu numele dat; dacă se tastează G., se încarcă primul program de pe bandă).

1.4.3. Algoritm de lucru cu GENS3M21

- Încărcarea: LOAD "GENS3M21" CODE 24064, 10034:
RANDOMIZE USR 24064

Dacă se iese din GENS (prin apăsarea tastei B) se revine în program cu comenzile

- RANDOMIZE USR 24125** (cu păstrarea textului sursă) sau
RANDOMIZE USR 24064
- RANDOMIZE USR 24120** (cu distrugerea textului sursă)

• Elaborarea unui program în cod mașină:

1) *I10,10* (înserează textul sursă de la linia 10 cu pasul 10)

2) Se tastează programul sursă avînd primele două linii

10 ORG ADR; ADR-adresa de start în zecimal sau hexazecimal
a rutinei în cod mașină

20 ENT ADR

3) Pentru corectarea instrucțiunilor greșite: **E nr.linie** și **CR**

Pentru ștergerea liniilor: **Dn, m** (șterge de la linia *n* la *m* inclusiv)

Pentru renumerotarea liniilor: **Nn, m** (*n*-linia de început.; *m*-pasul)

Pentru ieșirea din editor: **CS** și **1**.

4) Asamblarea:

• Se tastează **A** după care apar mesajele:

Table size: se apasă **ENTER**

Options : se apasă **5** și apoi **ENTER**.

• Dacă s-a afișat mesajul "Executes:ADR", pentru a rula programul în cod mașină se tastează **R**, sau se trece în *BASIC* (tastînd **B**) și apoi se dă comanda: **RANDOMIZE USR ADR**.

La mesajul *NO TABLE SPACE* se reasamblează *cu A, 500 sau A, 1000*.

5) Salvarea pe casetă:

- rutina cod mașină: se tastează **O,** *nume program*

- textul sursă: se tastează **P1** *număr ultima linie, nume program*.

6) Determinarea lungimii codului mașină

a) La ultima instrucțiune a programului sursă se introduce eticheta **ZEND**.

b) Se apasă **CS** și **1** (pentru a se ieși din editor) și apoi se apasă **A** (pentru asamblare), programul afișînd mesajele:

Table size: se tastează **ENTER**

Options: se apasă **5** și apoi **ENTER**

Programul va afișa adresa de lansare a codului mașină și tabelul etichetelor programului sursă însoțite de adresele lor în hexazecimal:

Executes: ADR

ZEND XYZW

Pentru exemplul de cod mașină dat aceste afișări sînt

Executes: 60000

ZEND EA6B

c) Se convertește valoarea XYZW din hexazecimal în zecimal:

$$\text{XYZW}_{16} = x \cdot 4096 + y \cdot 256 + z \cdot 16 + w$$

iar lungimea codului mașină va fi:

$$L = (\text{XYZW}_{10} + 1) \cdot \text{ADR}$$

Pentru exemplul de cod mașină dat vor rezulta valorile:

$$\text{EA6B}_{16} = 14 \cdot 4096 + 10 \cdot 256 + 6 \cdot 16 + 11 = 60011, \text{ respectiv}$$

$$L = (60011 + 1) \cdot 60000 = 12 \text{ octeți. Deci rutina se va scrie: "nume" CODE 60000,12.}$$

7) Încărcarea unui text sursă de pe casetă

- Comanda G*, încarcă primul program de pe casetă (sau *G*, *nume*)
- Se tastează *L* (pentru listarea programului)
- Se tastează *E* (pentru o eventuală relocare schimbând adresa *ADR* din pseudoinstrucțiunile *ORG* și *ENT*, sau introducerea de noi instrucțiuni).
- Se tastează *A* (pentru asamblare), când apar mesajele:

Table size: se apasă ENTER

Options: se apasă 5 și apoi ENTER.

În continuare se aplică indicațiile date la salvarea programului pe casetă.

Observație: **Functia *USR*** permite accesul din BASIC la o rutină în cod mașină folosind instrucțiunea

*nr. linie comandă *USR ADR**

unde "*comandă*" semnifică **PRINT, RANDOMIZE, GO TO, RUN** sau **LET literă =**, iar *ADR* este adresa de start a codului mașină indicată în mesajul "*Executes: ADR*" al programului *GENS*.

2.

SETUL DE INSTRUCȚIUNI

Prin noțiunea de "set de instrucțiuni" se înțelege totalitatea instrucțiunilor pe care microprocesorul le recunoaște și le execută. Cu cât setul de instrucțiuni este mai mare și mai variat, avînd mai multe clase de instrucțiuni și implică tehnici diferite de adresare, cu atît microprocesorul va fi mai performant. Calculatoarele compatibile cu marca *SPECTRUM* au microprocesorul *Z80A* fabricat de firma *Zilog*, care recunoaște și execută 696 de instrucțiuni declarate și 458 instrucțiuni nedeclarate și ca atare nerecunoscute de asamblare și dezasamblare.

Microprocesorul *Z80A* funcționează la o frecvență $\nu = 3,5$ MHz, o perioadă avînd valoarea $T = 1/\nu = 1/3,5 \cdot 10^6 = 0,2857 \cdot 10^{-6} \text{s} = 0,2857 \mu\text{s}$. Instrucțiunea cea mai scurtă durează $4T = 1,1428 \mu\text{s}$ iar cea mai lungă $23T = 6,571 \mu\text{s}$.

În acest capitol sînt prezentate instrucțiunile microprocesorului *Z80* în următoarea succesiune:

- instrucțiuni de încărcare pe 8 biți;
- instrucțiuni de încărcare pe 16 biți;
- instrucțiuni de interschimbabilitate;
- instrucțiuni de transfer de blocuri de date;
- instrucțiuni de căutare în blocuri de memorie;
- instrucțiuni logice și aritmetice pe 8 biți;
- instrucțiuni aritmetice cu scop general și de control al CPU;
- instrucțiuni aritmetice pe 16 biți;
- instrucțiuni de rotație și de deplasare;
- instrucțiuni de testare și modificare la nivel de bit;
- instrucțiuni de salt;
- instrucțiuni de apel și întoarcere din rutine;
- instrucțiuni de intrare/ieșire.

Modul de folosire a instrucțiunilor în programe scrise în limbaj de

asamblare este studiat începînd cu capitolul 3.

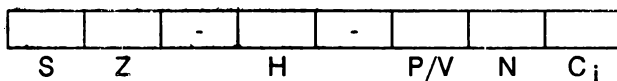
2.1. INSTRUCȚIUNI DE ÎNCĂRCARE PE 8 BIȚI

Instrucțiunile din acest grup permit încărcarea unui registru general cu o valoare imediată sau conținută într-un alt registru sau într-o locație de memorie, și stocarea într-un registru sau o locație de memorie a unei constante sau a conținutului unui alt registru general.

În formatele instrucțiunilor **registrele se scriu cu majuscule libere, iar locațiile de memorie cu majuscule cuprinse între paranteze:**

Notația utilizată pentru flaguri (indicatorii de condiție) va fi:

- ! (indicatorul este afectat conform rezultatului operației)
- (indicatorul nu este modificat de operație)
- 0 (indicatorul este forțat pe zero)
- 1 (indicatorul este pus pe 1)
- V (indicatorul P/V este poziționat în conformitate cu depășirea rezultatului)
- P (indicatorul P/V este poziționat în conformitate cu paritatea rezultatului)
- I (conține bistabilul de întreruperi IFF1)



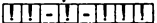
Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	LD r, r' r ← r'	r, r' = {A,B,C,D,E,H,L}	Conținutul registrului r' este transferat în registrul r	Nici unul
2	LD r, (mem) r ← (mem)	r = {A,B,C,D,E,H,L,} (mem) = {HL, IX+d, IY+d}	Conținutul registrului de memorie avînd adresa specificată de mem este transferat în registrul r	Nici unul

3	LD (mem), r (mem) ← r	(mem) = {HL, IX+d, IY+d} r = {A,B,C,D,E,H,L}	Conținutul registruului r este transferat în memorie la adresa specificată prin mem.	Nici unul
4	LD r, n r ← n	r = {A,B,C,D,E,H,L} n = 0..255 sau n ∈ [-128; +127]	Numărul n este înscris în registrul r.	Nici unul.
5	LD (mem),n (mem) ← n	mem = {HL, IX+d, IY+d}; n = 0...255 sau n [-128; +127]	Numărul n este înscris în locația de memorie a cărei adresă este specificată prin mem.	Nici unul
6	LD A, (rr) A ← (rr)	rr = {BC,DE}	Conținutul locației de memorie adresate prin registrul dublu rr este transferat în acumulator.	Nici unul
7	LD (rr),A (rr) ← A	rr = {BC,DE}	Conținutul acumulatorului A este înscris în locația de memorie avînd adresa specificată în registrul dublu r.	Nici unul
8	LD A, (nn) A ← (nn)	nn = 0..65535 sau nn ∈ [-32768; +32767]	Conținutul locației de memorie avînd adresa nn este transferat în acumulatorul A.	Nici unul
9	LD (nn),A (nn) ← A	nn = 0..65535 sau nn ∈ [-32768; +32767]	Conținutul acumulatorului A este depus în locația de memorie cu adresa nn	Nici unul
10	LD A,rs A ← rs	rs = {I,R}	Conținutul registruului special rs este transferat în acumulatorul A.	$\boxed{[I]} - \boxed{[O]} - \boxed{[I]} \boxed{[O]}$ SZ H R,NC

11	LD rs,A rs ← A	rs = {I,R}	Conținutul acumulatorului A este transferat în registrul special rs.	Nici unul
----	---------------------------------	------------	--	-----------

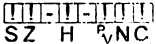
2.2. INSTRUCȚIUNI DE ÎNCĂRCARE PE 16 BIȚI

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	LD rr,nn rr ← nn	rr = {BC,DE,HL, SP,IX,IY} nn ∈ [0..65535]; nn ∈ [-32768; +32767]	Registrul dublu rr este încărcat cu valoarea nn.	Nici unul
2	LD rr,(nn) rr ← (nn)	rr = {BC,DE,HL, SP,IX,IY} nn ∈ [0..65535]; nn ∈ [-32768; +32767]	Registrul dublu rr este încărcat din memorie cu 2 octeți începînd de la adresa nn.	Nici unul
3	LD (nn),rr (nn) ← rr	nn ∈ [0..65535]; nn ∈ [-32768; +32767] rr = {BC,DE,HL, SP,IX,IY}	Registrul dublu rr este transferat în memorie în 2 locații succesive începînd cu adresa nn	Nici unul.
4	LD SP, rr SP ← rr	rr = {HL,IX,IY}	Indicatorul de stivă SP este încărcat din registrul dublu rr.	Nici unul

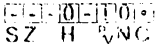
5	PUSH rr $(SP-2) \leftarrow rr_L$ $(SP-1) \leftarrow rr_H$	$rr = \{BC, DE, HL, AF, IX, IY\}$	Conținutul registrului dublu rr este salvat în memorie la adresa specificată de indicatorul de stivă SP. Salvarea se face la adrese descrescătoare, prima salvare implicînd octetul superior al registrului rr. Conținutul indicatorului de stivă se decrementează cu 2.	Nici unul.
6	POP rr $rr_H \leftarrow (SP+1)$ $rr_L \leftarrow (SP)$	$rr = \{BC, DE, HL, AF, IX, IY\}$	Registrul dublu rr este încărcat din memorie de la adresa specificată prin indicatorul de stivă SP. Conținutul locației cu adresă inferioară este transferat în octetul inferior al lui rr. Indicatorul de stivă SP este incrementat cu 2.	Nici unul pentru BC, DE, HL, IX și IY. Toate pentru AF  SZ H r _v NC.

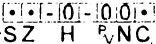
2.3. INSTRUCȚIUNI DE INTERSCHIMBABILITATE

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	EX DE, HL DE ↔ HL	-	Conținutul registrelor duble DE și HL este interschimbat.	Nici unul


2	EXX (BC) ↔ (BC') (DE) ↔ (DE') (HL) ↔ (HL')	-	Registrele secundare devin registre primare (de lucru) și invers	Nici unul
3	EX (SP),rr H ↔ SP+1 L ↔ (SP)	rr = {HL,IX,IY}	Conținutul registrului dublu rr este interschimbabil cu conținutul a 2 celule de memorie adresate prin indicatorul de stivă	Nici unul
4	EX AF,AF' AF ↔ AF'	-	Conținutul registrelor de stare (AF) primari și secundari este interschimbabil.	Toate  SZ H V NC

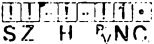
2.4. INSTRUCȚIUNI DE TRANSFER DE BLOCURI DE DATE

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	LDx (DE) ← (HL) DE = DE + 1 HL = HL + 1 BC = BC - 1	x poate fi; I=increment D=decrement	Conținutul celulei de memorie adresate prin registrul dublu HL este transferat în celula de memorie adresată prin registrul dublu DE. Conținutul lui DE și HL este incrementat cu 1, iar cel al lui BC este decrementat cu 1.	 SZ H V NC

2	LDxR (DE) ← (HL) DE = DE ± 1 HL = HL ± 1 BC = BC - 1	x poate fi: I = increment D = decrement	Se transferă un bloc de date de lungime egală cu BC. Blocul sursă începe la adresa dată de HL, iar blocul destinație începe la adresa dată de DE. Transferul are loc crescător (LDIR) sau descrescător (LDDR)	
---	---	---	---	---

2.5. INSTRUCȚIUNI PENTRU CĂUTAREA ÎN BLOCURI DE MEMORIE

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	CPx A → (HL) HL = HL ± 1 BC = BC - 1	x poate fi: I = increment D = decrement	Conținutul acumulatorului este copiat în cel al celei de memorie adresate prin HL. Indicatorul de adresă HL este incrementat cu 1 (CPD) sau decrementat cu 1 (CPI), conținutul acumulatorului A nu se schimbă, iar rezultatul comparației se regăsește în registrul de flag F.	 P = 1 pt. BC = 0 P = 0 pt. BC ≠ 0 Z = 1 dacă A = 0 Z = 0 dacă A ≠ 0

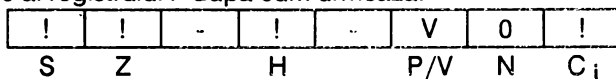
2	CPxR A → (HL) HL=HL±1 BC=BC-1	x poate fi: I=increment D=decrement	Conținutul acumulatorului A este copiat în cel al celei de memorie specificate în HL.Indicatorul de adresă al lui HL este incrementat (CPIR) sau decrementat (CPDR) cu 1. Numărătorul de octeți BC este decrementat cu 1. Dacă rezultatul comparării este egal, instrucțiunea se termină, iar dacă nu ea este reluată pînă cînd BC=0.	 P=0 pt.BC=0 P=1 pt.BC≠0 Z=1 dacă A=(HL) Z=0 dacă A≠(HL)
---	--	---	--	---

2.6. INSTRUCȚIUNI LOGICE ȘI ARITMETICE PE 8 BIȚI

În setul de instrucțiuni al microprocesorului Z80 sînt incluse și instrucțiuni ce efectuează operații de adunare, scădere, incrementare și decrementare, produs și sumă logică (ȘI, SAU), sumă modulo 2 (SAU EXCLUSIV) și comparări.

2.6.1. Adunare

Instrucțiunile de adunare pe 8 biți poziționează indicatorii de condiție ai registrului F după cum urmează:



Simbolul V arată că indicatorul P/V conține depășirea ce poate apare

în urma efectuării operației și anume: $V=1$ dacă există depășire, respectiv $V=0$ dacă nu există depășire. Flagul H se poziționează pe 1 dacă există transport din bitul b3, sau pe 0 în caz contrar, iar flagul C_i ia valoarea 1 dacă există transport de la bitul b7 sau 0 în caz contrar.

2.6.2. Scădere

Instrucțiunile de scădere pe 8 biți realizează scăderea între doi operanzi dintre care primul este obligatoriu plasat în acumulator. Rezultatul scăderii se depune în acumulator, iar indicatorii de poziție se poziționează după cum urmează:

!	!	-	!	-	V	1	!
S	Z		H		P/V	N	C_i

Flagul H este 1 dacă nu există împrumut din bitul b4 și 0 în caz contrar, iar indicatorul C_i este 1 dacă nu există împrumut și 0 în cazul contrar.

2.6.3. Instrucțiuni logice

Instrucțiunile din această categorie realizează operații logice (ȘI, SAU și SAU EXCLUSIV) între doi operanzi reprezentanți pe 8 biți, dintre care primul este obligatoriu plasat în acumulatorul A. Indicatorii de poziție se dispun astfel:

!	!	-	1	-	P	0	0
S	Z		H		P/V	N	C_i

Flagul P/V arată paritatea; $P=1$ pentru paritate pară și $P=0$ în caz contrar (paritate impară).

2.6.4. Comparări

Instrucțiunile din această categorie compară între ei doi operanzi reprezentați pe 8 biți, dintre care operandul unu (notat op1) este plasat obligatoriu în acumulator. Compararea se realizează prin scăderea internă $op1-op2$ în urma căreia se poziționează indicatorii de condiție astfel:

!	!	-	!	-	V	1	!
S	Z		H		P/V	N	C _i

Flagul H este 1 dacă nu există împrumut din bitul b4 și 0 în caz contrar, iar flagurile Z și C_i capătă valorile:

Z = 1 și C_i = 0 dacă op1 = op2

Z = 0 și C_i = 1 dacă op1 < op2

Z = 0 și C_i = 0 dacă op1 > op2

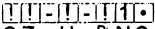
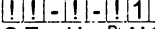
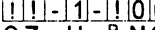
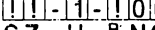
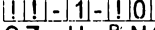
2.6.5. Incrementări și decrementări


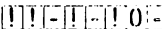
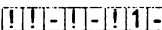
Setul de instrucțiuni de acest tip pe 8 biți poziționează indicatorii de condiție după cum urmează:

!	!	-	!	-	V	0	•
S	Z		H		P/V	N	C _i

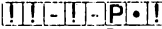

Flagul H este 1 dacă există transport din bitul b3 și 0 în caz contrar, iar flagul P/V arată depășirea (V) și anume: V = 1 dacă operandul a avut valoarea 127 înainte de incrementare, respectiv valoarea 128 înainte de decrementare; în caz contrar V = 0.



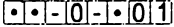
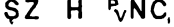
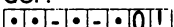
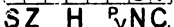
Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	ADD A,s A = A + s	s poate fi: r = {A,B,C,D,E,H, L} n ∈ 0..255 (HL), (IX+d), (IY+d)	Conținutul operandului s este adunat cu conținutul acumulatorului A și rezultatul este depus în A.	 SZ H P _V NC _i
2	ADC A,s A = A + S + C _i	Idem	Operandul s împreună cu indicatorul C _i se adună la conținutul acumulatorului A, iar rezultatul se depune în A.	 SZ H P _V NC _i

3	SUB s $A = A - s$	Idem	Operandul s este scăzut din conținutul acumulatorului A, iar rezultatul se depune în A	 SZ H P _v NC
4	SBC A,s $A = A - s - C_i$	Idem	Operandul s împreună cu indicatorul C _i se scad din conținutul acumulatorului A și rezultatul este depus în A	 SZ H P _v NC _i
5	AND s $A = A \wedge s$	Idem	Operație ȘI logic bit cu bit exercitată între octetul specificat de operandul s și octetul conținut în acumulatorul A, rezultatul fiind depus în A.	 SZ H P _v NC _i
6	OR s $A = A \vee s$	Idem	Operație SAU logic bit cu bit între octetul specificat de operandul s și octetul conținut de acumulatorul A, rezultatul fiind depus în A.	 SZ H P _v NC _i
7	XOR s $A = A \oplus s$	Idem	Operație SAU EXCLUSIV bit cu bit executată între octetul specificat de operandul s și octetul conținut în acumulatorul A, rezultatul fiind depus în A.	 SZ H P _v NC _i

8	CP s A-s	Idem	Conținutul operandului s este comparat cu conținutul acumulatorului A.	 SZ H P ₇ NC ₇
9	INC m m=m+1	m poate fi: r = {A,B,C,D,E,H,L (HL),(IX+d), (IY+d)}	Octetul specificat de operandul m este incrementat cu 1.	 SZ H P ₇ NC ₇
10	DEC m m=m-1	Idem	Octetul specificat de operandul m este decrementat cu 1.	 SZ H P ₇ NC ₇

2.7. INSTRUCȚIUNI ARITMETICE CU SCOP GENERAL ȘI DE CONTROL AL CPU

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	DAA		Ajustează condiționat acumulatorul A pentru operații de adunare (ADD,ADC,INC) sau de scădere (SUB,SBC,DEC, NEG) în cod BCD, adăugând +6, +60, +66,-6,-60, -66 la conținutul acumulatorului, în funcție de starea flagurilor C _i , N și H.	 SZ H P ₇ NC ₇
2	CPL A = \bar{A}		Conținutul acumulatorului este complementat față de 1.	 SZ H P ₇ NC ₇

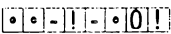
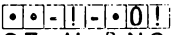
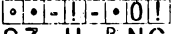
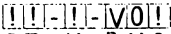
3	NEG A=0-A	-	Conținutul acumulatorului A este complementat față de 2 (operație echivalentă cu scăderea conținutului acumulatorului din 0.	 
4	xCF (SCF sau CCF) SCF: $C_i = 1$ CCF: $C_i = \overline{C_i}$	x=litera {S;C}	C _i : -ia valoarea 1 (SCF) -este negat (complementat) (CCF)	SCF:   CCF:  
5	NOP	-	CPU nu efectuează nici o operație în timpul acestui ciclu mașină.	nici unul
6	HALT	-	După execuția instrucțiunii microprocesorul se oprește, executînd fetch- uri (care nu se iau în considerare) pentru a se asigura reîmprospătarea memoriei dinamice (dacă este cazul). leșirea din această stare se face prin recepția unei întreruperi sau a unui semnal de RESET.	Nici unul.

7	xl (EI sau DI) EI:IFF1 = IFF2 = 1 DI:IFF1 = IFF2 = 0	x = litera {E;D}	<u>E</u> : sistemul de întrerupere se validează (dezactivează) acceptînd o întrerupere după execuția primei instrucțiuni care urmează după EI. Starea validată durează pînă la execuția primei instrucțiuni DI sau pînă la acceptarea primei cereri de întrerupere mascabilă. <u>D</u> : sistemul de întrerupere se inhibă. Nu se mai acceptă cererile de întrerupere mascabilă ci doar cele nemascabile. Starea de inhibare poate fi suspendată prin execuția unei instrucțiuni EI.	Nici unul
---	---	------------------	---	-----------

8	IMx (IM 0, IM 1 sau IM2)	x = cifra {0;1;2}	<p><u>IM 0</u>: determină CPU să lucreze în modul de întrerupere 0; dispozitivul care a generat întreruperea va depune pe magistrala de date codul unei instrucțiuni pe care CPU o execută (CALL, RST).</p> <p><u>IM 1</u>: determină CPU să lucreze în modul de întrerupere 1, executând automat o instrucțiune RST 56 la acceptarea unei întreruperi.</p> <p><u>IM 2</u>: determină CPU să lucreze în modul de întrerupere 2; prin aceasta se permite un apel indirect la orice locație de memorie pe care CPU îl formează astfel: cei mai semnificativi 8 biți reprezintă conținutul registrului vector al întreruperilor I, iar cei mai puțini semnificativi 8 biți sînt turnizați de dispozitivul care a generat întreruperea.</p>	Nici unul
---	--------------------------------	-------------------	---	-----------

2.8. INSTRUCȚIUNI ARITMETICE PE 16 BIȚI

Acest grup este alcătuit din 7 tipuri instrucțiuni care realizează operații de adunare, scădere, incrementare și decrementare.

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate.
1	ADD HL,rr HL=HL+rr	rr = {BC,DE,HL, SP}	Conținutul registrului dublu rr = {BC,DE,HL, SP} este adunat la conținutul registrului dublu HL, rezultatul fiind depus în HL.	 SZ H P V NC,
2	ADD IX,pp IX=IX+pp	pp = {BE,DE,IX, SP}	Conținutul registrului dublu pp = {BC,DE,IS, SP} este adunat la conținutul registrului dublu IX, rezultatul fiind depus în IX.	 SZ H P V NC,
3	ADD IY,ss IY=IY+ss	ss = {BC,DE,IY, SP}	Conținutul registrului dublu ss = {BC,DE,IY, SP} este adunat la conținutul registrului dublu IY, rezultatul fiind depus în IY.	 SZ H P V NC,
4	ADD HL,rr HL=HL+rr+C_i	rr = {BC,DE,HL, SP}	Conținutul registrului dublu rr = {BC,DE,HL, SP} se adună cu conținutul registrului dublu HL, și cu valoarea indicatorului C _i din registrul F _i ; rezultatul se depune în HL.	 SZ H P V NC,

5	SBC HL,rr HL=HL-rr-C_i	rr = {BC,DE,HL, SP}	Conținutul registrului dublu rr = {BC,DE,HL, SP} se scade din conținutul registrului dublu HL și din diferența obținută se scade valoarea flagului C _i al registrului F. Rezultatul se depune în HL.	! ! ! - ! ! - V ! ! SZ H R_v NC,
6	INC qq qq=qq+1	qq = {BC,DE,HL, IX,IY,SP}	Conținutul registrului dublu qq = {BC,DE,HL, IX,IY,SP} este incrementat cu 1	Nici unul
7	DEC qq qq=qq-1	qq = {BC,DE,HL, IX,IY,SP}	Conținutul registrului dublu qq = {BC,DE,HL, IX,IY,SP} este decrementat cu 1	Nici unul

2.9. INSTRUCȚIUNI DE ROTAȚIE ȘI DE DEPLASARE

Mnemonica acestor instrucțiuni folosesc ca primă literă pentru:

- rotiri: **R**;

- deplasări: **S**;

Cea de a doua literă a mnemonicii indică direcția

R=dreapta; **L**=stînga


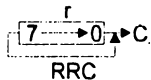
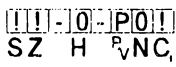
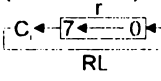

În cazul deplasărilor (S) se poate indica tipul deplasării

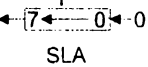
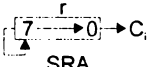
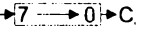
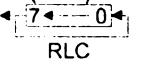
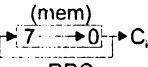
A=aritmetică; **L**=logică

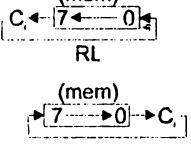
iar în cazul rotirilor (R) a treia literă a mnemonicii - dacă există-, indică o rotire circulară (C) însoțită de o deplasare a bitului b0 sau b7.

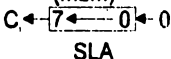
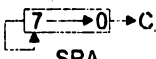
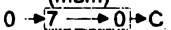
Grupul conține 16 instrucțiuni și poziționează biții de condiție în felul următor:

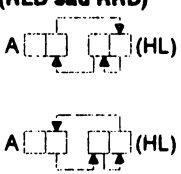
!	!	-	0	-	P	0	!
S	Z	H	P/V	N	C _i		

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	RxCr (RLC sau RRC)  	$x = \text{litera } \{L, R\}$ $r = \{B, C, D, E, H, L, A\}$	RLC: conținutul registrului r este <u>deplasat</u> cu o poziție la <u>stînga</u> . Bitul b7 se transferă în flagul C_i și în bitul b0. RRC: conținutul registrului r este <u>deplasat</u> cu o poziție la <u>dreapta</u> . Bitul b0 se transferă în flagul C_i și în bitul b7	
2	Rxr (RLr sau RRr)  	$x = \text{litera } \{L, R\}$ $r = \{B, C, D, E, H, L, A\}$	RLr: conținutul registrului r este <u>deplasat</u> cu o poziție la <u>stînga</u> . Bitul b7 se transferă în flagul C_i , iar C_i se transferă în bitul b0. RRr: conținutul registrului r este <u>deplasat</u> cu o poziție la <u>dreapta</u> . Bitul b0 se transferă în flagul C_i , iar C_i se transferă în bitul b7.	Idem

3	<p>SxAr (SLA r sau SRA r)</p>  <p>SLA</p>  <p>SRA</p>	<p>x = litera {L,R} r = {B,C,D,E,H,L,A}</p>	<p><u>SLA</u> r: conținutul registrului r este <u>deplasat</u> cu o poziție la <u>stînga</u>. Bitul b7 se transferă în flagul C_i, iar în bitul b0 se înserează 0.</p> <p><u>SRA</u> r: conținutul registrului r este <u>deplasat</u> cu o poziție la <u>dreapta</u>. Bitul b0 se transferă în flagul C_i, iar bitul b7 rămîne neschimbat.</p>	Idem
4	<p>SRL r</p> 	<p>r = {B,C,D,E,H,L,A}</p>	<p>Conținutul registrului r este <u>deplasat</u> cu o poziție la <u>dreapta</u>. Bitul b0 se transferă în flagul C_i, iar în bitul b7 se înserează 0.</p>	Idem
5	<p>RxC(mem) (RLC sau RRC)</p>  <p>RLC</p>  <p>RRC</p>	<p>x = litera {L,R} mem = {HL, IX + d, IY + d}</p>	<p><u>RLC (mem)</u>: conținutul celei de memorie adresate prin <u>mem</u> este <u>deplasat</u> cu o poziție la <u>stînga</u>. Bitul b7 se transferă în flagul C_i și în bitul b0.</p> <p><u>RRC (mem)</u>: conținutul celei de memorie adresate prin <u>mem</u> este <u>deplasat</u> cu o poziție la <u>dreapta</u>. Bitul b0 se transferă în flagul C_i și în bitul b7.</p>	Idem

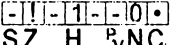
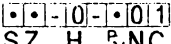
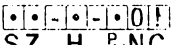
6	<p>Rx(mem) (RL sau RR) (mem)</p>  <p>RL</p> <p>(mem)</p>	<p>x = litera {L,R} mem = {HL,IX+d,I Y+d}</p>	<p>RL (mem): conținutul celulei de memorie adresate prin mem este deplasat la stînga cu o poziție. Bitul b7 se transferă în flagul C_i, iar C_i se transferă în bitul b0.</p> <p>RR (mem): conținutul celulei de memorie adresate prin mem este deplasat la dreapta cu o poziție. Bitul b0 se transferă în flagul C_i, iar C_i se depune în bitul b7.</p>	Idem
---	--	---	---	------

7	<p>SxA(mem) (SLA sau SRA) (mem)</p>  <p>SLA</p>  <p>SRA</p>	<p>x=litera {L,R} mem = {HL,IX+d, IY+d}</p>	<p>SLA (mem): conținutul locației de memorie adresate prin <u>mem</u> este <u>deplasat la stînga</u> cu o poziție. Bitul b7 se depune în flagul C_i, iar în bitul b0 se înserează 0. SRA (mem): conținutul locației de memorie adresate prin <u>mem</u> este <u>deplasat la</u> <u>dreapta</u> cu o poziție. Bitul b0 se depune în flagul C_i, iar bitul b7 rămîne neschimbant.</p>	Idem
8	<p>SRL (mem) (mem)</p> 	<p>mem = {HL,IX+d, IY+d}</p>	<p>Conținutul celulei de memorie adresate prin <u>mem</u> este <u>deplasat</u> cu o poziție <u>la dreapta</u>. Bitul b0 se depune în flagul C_i, iar în bitul b7 se depune valoarea 0.</p>	Idem

9	<p>RxD (RLD sau RRD)</p> 	x = litera (L,D)	<p>RLD: Conținutul celulei de memorie adresate prin conținutul registrului dublu HL este <u>rotit la stînga</u> folosind digitul inferior al acumulatorului A și anume:</p> <ul style="list-style-type: none"> - biții b3..b0 ai lui A trec în biții b3..b0 ai lui HL; - biții b3..b0 ai lui (HL) trec în biții b7..b4 ai lui (HL); - biții b7..b4 ai lui (HL) trec în biții b3..b0 ai lui A. <p>RRD: conținutul celulei de memorie adresate prin conținutul registrului dublu HL este <u>rotit la dreapta</u> folosind digitul inferior al acumulatorului A și anume:</p> <ul style="list-style-type: none"> - biții b3..b0 ai lui A trec în biții b7..b4 ai lui (HL); - biții b7..b4 ai lui (HL) trec în biții b3..b0 ai lui (HL); - biții b3..b0 ai lui (HL) trec în biții b3..b0 ai lui A. 	
---	---	------------------	--	--

2.10. INSTRUCȚIUNI DE TESTARE ȘI MODIFICARE LA NIVEL DE BIT

Cele nouă instrucțiuni din această grupă operează asupra unuia dintre cei opt biți ai unui registru general sau ai unei locații de memorie. Parametrul b din instrucțiune poate lua o valoare cuprinsă între poziția 0 (cel mai puțin semnificativ .s.b) și poziția 7 (cel mai semnificativ m.s.b.).

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	BIT b,m $Z = \overline{b_m}$	$b = \{0..7\}$ $m = \{B,C,D,E,H,L, A, (HL), (IX+d), (IY+d)\}$	Valoarea complementată a bitului b din operandul m este copiată în flagul Z	 SZ H P _V NC _i
2	RES b,m $b_m = 0$	$b = \{0..7\}$ $m = \{B,C,D,E,H,L, A(HL), (IX+d), (IY+d)\}$	În bitul b din operandul m se înscrie valoarea 0	Nici unul
3	SET b,m $b_m = 1$	$b = \{0..7\}$ $m = \{B,C,D,E,H,L, A, (HL), (IX+d), (IY+d)\}$	În bitul b din operandul m se înscrie valoarea 1.	Nici unul
4	xCF (SCF sau CCF) SCF: $C_i = 1$ CCF: $C_i = \overline{C_i}$	$x = \text{litera } \{S;C\}$	SCF: face $C_i = 0$ CCF: face $C_i = 1 - C_i$	 SZ H P _V NC _i  SZ H P _V NC _i

2.11. INSTRUCȚIUNI DE SALT

Instrucțiunile din această grupă realizează salturi condiționate și necondiționate și nu afectează biții de condiție (flagurile).

În cele ce urmează s-a notat condiția $c = \{NZ\text{-non zero, } NC\text{-non Carry (C}_i\text{), } C\text{-Carry, } PO\text{-paritate impară, } PE\text{-paritate pară, } P\text{-plus, } M\text{-}$

minus}

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează
1	JP nn PC=nn	nn = [0..65535] (o adresă)	În contorul program PC se înscrie adresa <u>nn</u> , iar programul execută un salt la adresa nn.
2	JP c, nn PC=nn	c = {NZ,Z,NC,C, PO,PE,P,M} nn = [0..65535]	Se testează câte un bit al registrului F. Dacă condiția căutată <u>c</u> este adevărată se execută salt la adresa <u>nn</u> , iar dacă această condiție este falsă programul continuă cu instrucțiunea următoare.
3	JR d PC=PC+d	d = [-128; +127] (deplasamentul) exprimat în complement față de 2).	Deplasamentul <u>d</u> este adunat la valoarea curentă a contorului program <u>PC</u> rezultând noua adresă la care va face un salt programul.
4	JR c,d a) NZ - dacă Z=1 continuă - dacă Z=0 salt la PC=PC+d b) Z - dacă Z=0 continuă - dacă Z=1 salt la PC=PC+d c) NC - dacă C _i =1 continuă - dacă C _i =0 salt la PC=PC+d d) C - dacă C _i =0 continuă - dacă C _i =1 salt la PC=PC+d	c = {NZ,Z,NC,C} este condiția d = [-128; +127] (deplasamentul)	Se testează câte un bit al registrului F. Dacă condiția este adevărată se efectuează saltul la adresa PC=PC+d, iar în caz contrar se execută instrucțiunea următoare a programului.
5	JP (rr) PC=rr	rr = {HL,IX,IY}	În contorul program PC se copiază conținutul registrului dublu rr = {HL,IX,IY}. Programul va executa un salt la adresa specificată prin rr. Conținutul registrului dublu rr rămâne neschimbat.

6	DJNZ d Dacă B=0 salt la PC=PC+d	$d = [-128; +127]$	Conținutul registrului B este decrementat cu 1. Dacă astfel s-a ajuns la valoarea $B = 0$ se execută un salt relativ la adresa $PC = PC + d$, iar dacă $B \neq 0$ se execută următoarea instrucțiune din program.
---	--	--------------------	--

2.12. INSTRUCȚIUNI DE APEL ȘI ÎNTOARCERE DIN RUTINE

Din acest grup de 6 instrucțiuni fac parte instrucțiunile care apelează subrutine și realizează întoarcerea din acestea. Ele nu afectează biții de condiție (flagurile).

Nr. crt.	Menmonica și operație	Semnificația	Ce se realizează
1	CALL nn $SP = SP - 1, (SP) \leftarrow PC_H,$ $SP = SP - 1, (SP) \leftarrow PC_L,$ $PC = nn$	$nn = [0..65535]$ (adresă)	Conținutul contorului program PC este salvat în vârful stivei, după care PC este încărcat cu noua adresă <u>nn</u> . Conținutul indicatorului de stivă SP este decrementat de două ori.
2	CALL c,nn $SP = SP - 1, (SP) \leftarrow PC_H,$ $SP = SP - 1, (SP) \leftarrow PC_L,$ $PC = nn$	$c = \{NZ,Z,NC,C, PO,PE,P,M\}$ (condiția) $nn = [0..65535]$	Dacă condiția este adevărată se execută salt la subrutină, adică se salvează adresa de revenire în vârful stivei, se încarcă <u>PC</u> cu <u>nn</u> și indicatorul de stivă SP este decrementat de două ori. Dacă condiția c este falsă, se execută instrucțiunea următoare.

3	RET $PC_L \leftarrow (SP), SP = SP + 1,$ $PC_H \leftarrow (SP), SP = SP + 1$		Adresa de revenire (din subrutină în programul apelant) din vârful stivei se încarcă în contorul program, după care se efectuează saltul. Conținutul indicatorului de stivă SP este incrementat de două ori.
4	RET c $PC_L \leftarrow (SP), SP = SP + 1,$ $PC_H \leftarrow (SP), SP = SP + 1$	$c = \{NZ, Z, NC, C, PO, PE, P, M\}$ (condiția)	Dacă condiția c este adevărată se execută revenirea la programul apelant, iar dacă această condiție este falsă se execută următoarea instrucțiune din program. Adresa de revenire în programul apelant se află în vârful stivei și se încarcă în contorul program PC, iar conținutul indicatorului stivei SP este incrementat de 2 ori.
5	RET_x (RETI sau RETN) $PC_L \leftarrow (SP), SP = SP + 1,$ $PC_H \leftarrow (SP), SP = SP + 1$ IFF2 = IFF1	$x = \text{litera} \{I; N\}$	RETI: revenire din întrerupere mascabilă. Adresa de revenire din vârful stivei (SP) se încarcă în contorul program PC, iar conținutul indicatorului de stivă SP este incrementat de două ori. Este necesar să se execute instrucțiunea EI înainte de RETI pentru a se revalida întreruperile mascabile. RETN: revenire din întrerupere nemascabilă. Adresa de revenire din vârful stivei de încarcă în contorul program PC, iar conținutul indicatorului de stivă este incrementat de două ori.

6	RST p $SP = SP - 1, (SP) \leftarrow PC_H$ $SP = SP - 1, (SP) \leftarrow PC_L$ $PC = p$	$p = [0; 8_H; 10_H;$ $18_H; 20_H; 28_H;$ $30_H; 38_H]$ (adresă din pagina 0)	Realizează transferul la o anumită subrutină dată de 8 adrese. Adresa de revenire se salvează în vârful stivei, contorul program PC se încarcă cu p și se execută saltul. Conținutul indicatorului de stivă SP este decrementat de două ori.
---	--	--	--

2.13. INSTRUCȚIUNI DE INTRARE/IEȘIRE

Instrucțiunile de intrare/ieșire permit un transfer de date între registrele generale sau o locație de memorie specificată de (HL) și oricare dintre cele 256 porturi de intrare/ieșire; adresa portului $n = [0...255]$ este specificată în instrucțiune.

Nr. crt.	Mnemonica și operația	Semnificația	Ce se realizează	Flaguri afectate
1	IN A, (n) $A = n$	$n = [0...255]$ (adresa portului)	Conținutul portului n este transferat în acumulatorul A al cărui conținut vechi se pierde	Nici unul
2	IN r, (C) $r = C$	$r = \{B, C, D, E, H, L, A\}$	Conținutul portului adresat prin conținutul registrului C este transferat în registrul r	$!! - 0 - !0 \cdot$ $SZ H P_V NC,$

3	INx (INI sau IND) <u>INI</u> (HL) = (C) HL = HL + 1 B = B - 1 <u>IND</u> (HL) = (c) HL = HL - 1 B = B - 1	x = litera {I;D}	Conținutul portului adresat prin conținutul registrului C este transferat la adresa specificată de HL. Indicatorul de adresă HL este incrementat (la INI) sau decrementat (la IND), iar numărătorul de octeți B este decrementat.	$\begin{array}{ c c c c c c c } \hline - & 1 & - & - & - & - & 1 & 0 \\ \hline \end{array}$ SZ H r_V NC _i
4	INxB (INIR sau INDR) <u>INIR</u> (HL) = (C) B = B - 1 HL = HL + 1 <u>INDR</u> (HL) = (C) B = B - 1 HL = HL - 1	x = litera {I,D}	Conținutul portului adresat prin conținutul registrului C este transferat la adresa specificată de HL. Indicatorul de adresă HL este incrementat (la INIR) sau decrementat (la INDR), iar numărătorul de octeți B este decrementat cu 1 până când B = 0	$\begin{array}{ c c c c c c c } \hline - & 1 & - & - & - & - & 1 & 0 \\ \hline \end{array}$ SZ H r_V NC _i
5	OUT (n), A n = A	n = [0..255] (adresa portului)	Conținutul acumulatorului A este transferat în portul n.	Nici unul
6	OUT(C),r (C) = r	r = {B,C,D,E,H,I, A}	Conținutul registrului r este transferat la portul avînd adresa specificată de registrul C.	Nici unul

3. FOLOSIREA INSTRUCȚIUNILOR ÎN OPERAȚII DE BAZĂ

Avînd cunoscute instrucțiunile limbajului de asamblare ale microprocesorului Z80, etapa următoare de studiu este utilizarea lor în programe simple și scurte pentru a le înțelege atît funcționalitatea proprie și interacțiunile cu celelalte instrucțiuni, cît și condițiile în care pot fi utilizate fără a se produce distrugerea (blocarea) programului.

Prin operații de bază se înțeleg operațiile uzuale și anume:

- încărcarea în memorie;
- operații aritmetice;
- influențarea unui bit;
- transferuri de blocuri de memorie.

3.1. NOȚIUNI INTRODUCTIVE

Pentru a putea folosi instrucțiunile prezentate în capitolul anterior, trebuie reamintite cîteva elemente importante și anume: rolul funcției **USR**, organizarea memoriei și a ecranului, structura variabilelor de sistem și codurile caracterelor.

3.1.1. Rolul funcției *USR*

După cum s-a menționat, funcția **USR** din **BASIC** permite accesul la rutinele (programele) în cod mașină, care sînt activate de o comandă **BASIC** scrisă înainte funcției **USR**, adică:

PRINT USR adr	LET literă = USR adr
RANDOMIZE USR adr	PRINT AT USR adr

```

RUN USR adr      IF USR adr
GO TO USR adr    PAUSE USR adr

```

unde "adr" este adresa de lansare a rutinei în cod mașină (ex: RANDOMIZE USR 60000). Pentru reîntoarcerea rutinei în BASIC, la sfârșitul programului scris cu asamblorul GENS3M21 se va scrie instrucțiunea RET, structura rutinei fiind:

```

10 ORG adr
20 ENT adr
:
      ZEND RET

```

Funcționarea funcției USR se obține astfel:

- calculatorul plasează adresa lui USR în registrul dublu **BC**;
- se execută rutina în cod mașină;
- când survine instrucțiunea **RET** calculatorul înapoiază conținutul registrului dublu BC.

Anticipînd paragraful 3.2-unde se vor aplica instrucțiunile de încărcare LD-, se realizează programul

```

10          ORG 60000
20          ENT 60000
30          LD B,0           ;registrul B se
                             incarca cu numarul
                             0
40          LD C,100        ;registrul C se
                             incarca cu numarul
                             100
50 ZEND      RET

```

Se assemblează programul (tastînd A1), se revine în BASIC(tastînd B) și, dînd comanda **PRINT USR 60000**, calculatorul va afișa cifra 100.

3.1.2. Organizarea memoriei calculatorului și organizarea ecranului

• Spațiul de memorie adresabil de către microprocesorul calculatoarelor compatibile cu *ZX-SPECTRUM* este împărțit în două zone după cum urmează:

- prima zonă conține memoria nevolatilă *ROM* (Read Only Memory) de 16Ko cuprinsă între adresele 0 și 16383;

- a doua zonă conține memoria volatilă RAM (Random Acces Memory) de 48 Ko subdivizată în două părți: memoria video și de program (de 16 ko între adresele 16384-32767) și memoria suplimentară (de 32 Ko între adresele 32768-65535).

În fig.3.1 este prezentată această repartizare a zonelor de memorie în care s-a reținut numai ceea ce este necesar programării în limbaj de asamblare.

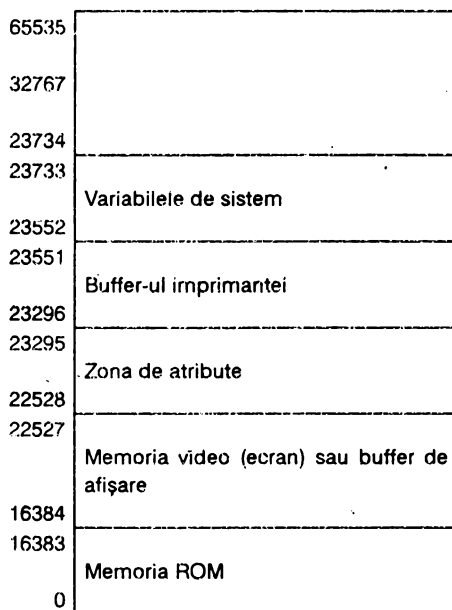


Fig.3.1.

- Ecranul este împărțit în trei părți egale, memorate una după alta, conform schemei din fig.3.2. Fiecare treime ocupă 32 caractere x8 pixeli un caracter x8 rînduri = 2048 octeți.

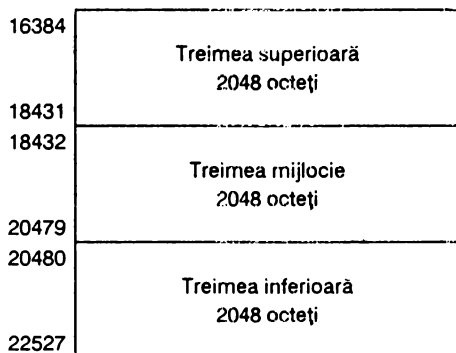


Fig. 3.2.

În fig.3.3.a este prezentată împărțirea ecranului în linii și coloane. O linie este o grupare de 8 rânduri, unde un rând este o succesiune de 256 pixeli (fig.3.3.b).

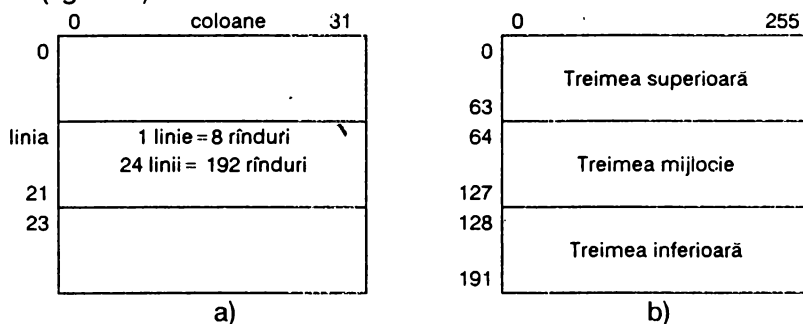


Fig.3.3.

Un asemenea rând este vizualizat de programul în BASIC

10 CLS : PLOT 0,80: DRAW 255,0

În total sînt pe ecran 24 linii cu 192 rânduri, numerotate respectiv 0-23 (la linii) și 0-191 (la rânduri) - fig.3.3.a,b.

În cadrul unui rând adresele octeților ce conțin imaginea lui sînt în ordine crescătoare, iar adresa primului octet al unui rând nu este întotdeauna egală cu adresa ultimului octet al liniei precedente plus 1.

Un program BASIC ce permite vizualizarea organizării memoriei video este următorul:

10 FOR r=16384 TO 22527 : POKE r, 255: NEXT r

În fig.3.4 este reprezentată harta memoriei video cu adresele date în zecimal pentru începutul și sfîrșitul liniei, precum și adresa

corespunzătoare a zonei de atribute.

Linia	Rîndul	Începutul liniei		Sfîrșitul liniei		
		Afișare	Atribut	Afișare	Atribut	
0	1	16384		16415		↑
	2	16640		16671		
	3	16896		16927		
	4	17152		17183		
	5	17408	22528	17439	22559	
	6	17664		17695		
	7	17920		17951		
	8	<u>18176</u>		<u>18207</u>		
1	1	16416		16447		
		...	22560	...	22591	
2	8	<u>18208</u>		<u>18239</u>		
	1	16448		16479		
3	8	<u>18240</u>		<u>18271</u>		Treimea superioară
	1	16460		16511		
4	8	<u>18272</u>		<u>18303</u>		
	1	16512		16543		
5	8	<u>18304</u>	22656	<u>18335</u>	22687	
	1	16544		16575		
6	8	<u>18336</u>		<u>18367</u>		
	1	16576		16607		
7	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
8	8	<u>18400</u>		<u>18431</u>		↓
		...	22720	...	22751	
9	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
10	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
11	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
12	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
13	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
14	8	<u>18368</u>		<u>18399</u>		
	1	16608		16639		
15	8	<u>18368</u>		<u>18399</u>		↓
	1	16608		16639		
8	1	<u>18432</u>	22784	<u>18643</u>	22815	↑
9	1	<u>18464</u>	22816	<u>18495</u>	22847	
10	1	<u>18496</u>	22848	<u>18527</u>	22879	
11	1	<u>18528</u>	22880	<u>18559</u>	22911	
12	1	<u>18560</u>	22912	<u>18591</u>	22943	
13	1	<u>18592</u>	22944	<u>18623</u>	22975	
14	1	<u>18624</u>	22976	<u>18655</u>	23007	
15	1	<u>18656</u>	23008	<u>18687</u>	23039	

16	1	<u>20480</u>	23040	<u>20511</u>	23071	↑ Freimea inferioară ↓
17	1	<u>20512</u>	23072	<u>20543</u>	23103	
18	1	<u>20544</u>	23104	<u>20575</u>	23135	
19	1	<u>20576</u>	23136	<u>20607</u>	23167	
20	1	<u>20608</u>	23168	<u>20639</u>	23199	
21	1	<u>20640</u>	23200	<u>20671</u>	23231	
22	1	<u>20672</u>	23232	<u>20703</u>	23263	
23	1	20704		20735		
		...	23264	...	23295	
	8	<u>22496</u>		<u>22527</u>		

Fig.3.4

În limbajul BASIC accesul la un punct oarecare al ecranului se realizează prin coordonatele sale x,y unde $x=0...255$ și $y=0..175$ (fig 3.5).

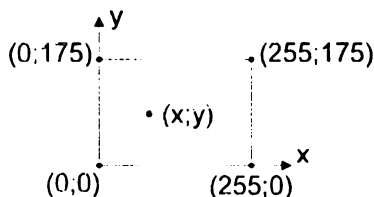
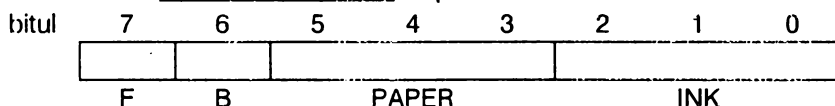


Fig.3.5

Prin urmare, ecranul folosește 24 linii (numerotate 0-23), dar spațiul utilizabil pentru instrucțiunile **PLOT**, **DRAW**, **CIRCLE**, **POINT** nu include și ultimile două linii de caractere (22-23).

- Zona de atribute video (22528-23295) conține informații referitoare la culorile pentru cerneală (**INK**) și hîrtie (**PAPER**) pe care le poate avea un punct de pe ecran. În scopul economisirii memoriei, pentru o matrice de 8×8 pixeli (cît este necesar pentru un caracter) s-a definit un singur octet de atribute. Astfel, primul octet al zonei corespunde caracterului de la intersecția liniei 0 cu coloana 0 (scris 0;0), al doilea octet corespunde caracterului (1;0), ș.a.m.d., ultimul octet corespunzînd caracterului (23,31).

Formatul octetului de atribute se prezintă astfel:



unde: F - atributul de FLASH (cu valorile 1 pentru cazul clipitor și 0 în caz contrar)

B - atributul de BRIGHT (cu valoarea 1 pentru cazul strălucitor și

0 în caz contrar);

PAPER - culoarea hîrtiei (cu valorile 0 =negru, 1 =albastru, 2 =roșu, 3 =purpuriu, 4 =verde, 5 =albastru deschis, 6 =galben, 7 =alb);

INK - culoarea cernelii (cu aceleași valori ca la PAPER).

3.1.3. Structura variabilelor de sistem

Octeții din memorie de la adresa 23552 la adresa 23733 sînt rezervați pentru operații specifice ale sistemului. Ei pot fi citiți pentru a se afla informații despre sistem, iar cîțiva dintre ei pot fi modificați. Acești octeți se numesc variabile de sistem și au cîte un nume. În cazul variabilelor formate din mai mulți octeți, primul va fi cel mai puțin semnificativ. Structura variabilelor de sistem este prezentată în tabelul 3.1, în care abrevierile din coloana 1 au următoarele semnificații:

- X variabila nu poate fi modificată;
 N modificarea variabilei nu are efect asupra sistemului;
 număr numărul de octeți ocupat de variabilă

Tabelul 3.1.

Tip	Adresă	Nume	Conținut
N8	23552	KSTATE	Folosită în citirea tastaturii
N1	23560	LAST K	Reține ultima tastă apăsată
1	23561	REPDEL	Durata (în 1/50 sec) cît trebuie ținută apăsată o tastă pentru a se repeta
1	23562	REPPER	Timpul (în 1/50 sec) după care se repetă o tastă apăsată
N2	23563	DEFADD	Adresa argumentelor funcțiilor definite de utilizator
N1	23565	K DATA	Al doilea octet pentru controlul culorii introduse de la tastatură
N2	23566	TVDATA	Controlul culorii, al lui AT și TAB pentru TV
X38	23568	STRMS	Adresa canalului atașat căi
22	23606	CHARS	Adresa generatorului de caractere minus 256
1	23608	RASP	Durata sunetului la eroare (bîzîitului)

1	23609	PIP	Durata sunetului la apăsarea unei taste (clic)
1	23610	ERR NR	Codul de mesaj minus 1
X1	23611	FLAGS	Diferiți indicatori de control ai sistemului BASIC
X1	23612	TVFLAG	Indicatori asociați cu TV
X2	23613	ERR SP	Adresa elementului din stiva mașinii, utilizat ca adresă de întoarcere în caz de eroare
N2	23615	LIST SP	Adresa de întoarcere la listările automate
N1	23617	MODE	Specifică cursorul (K, L, C, E, G)
2	23618	NEWPPC	Linia la care se sare
1	23620	NSPPC	Numărul instrucțiunii în linie la care se sare
2	23621	PPC	Numărul liniei pentru instrucțiunea în execuție
1	23623	SUBPPC	Numărul instrucțiunii din linie în execuție
1	23624	BORDCR	Culoarea borderului
2	23625	E PPC	Numărul liniei curente
X2	<u>23627</u>	VARS	Adresa variabilelor BASIC
N2	23629	DEST	Adresa variabilelor asignate
X2	23631	CHANS	Adresa datelor de canal
X2	23633	CURCHL	Adresa informației curente folosite pentru intrare sau ieșire
X2	<u>23635</u>	PROG	Adresa programului BASIC
X2	23637	NXTLIN	Adresa următoarei linii din program
X2	23639	DATADD	Adresa ultimului element din lista DATA
X2	23641	E LINE	Adresa comenzii introduse
2	23643	K CUR	Adresa cursorului
X2	23645	CH ADD	Adresa următorului caracter care urmează să fie interpretat
2	23647	XPTR	Adresa caracterului după semnul întrebării
X2	23649	WORKSP	Adresa spațiului de lucru temporar
X2	23651	STKBOT	Adresa inferioară a stivei calculator
X2	23653	STKEND	Adresa de început a spațiului liber

N1	23655	BREG	Registrul B al calculatorului
N2	23656	MEM	Adresa spațiului folosit pentru memoria calculatorului
1	23658	FLAGS2	Alți indicatori
X1	23659	DF SZ	Numărul liniilor din partea de jos a ecranului
2	23660	S TOP	Numărul liniei de sus a programului la listarea automată
2	23662	OLDPPC	Numărul liniei la care sare CONTINUE
1	23664	OSPPC	Numărul din linie la care sare CONTINUE
N1	23665	FLAGX	Diversi indicatori
N2	23666	STRLEN	Lungimea asignată șirului
N2	23668	T ADDR	Adresa următorului element din tabela de sintaxă
2	23670	SEED	Variabilă pentru RND
3	<u>23672</u>	FRAMES	Contorul timpului real
2	<u>23675</u>	UDG	Adresa primului grafic definit de utilizator
1	23677	COORDS	Coordonata x a ultimului punct PLOT-at
1	23678	-	Coordonata y a ultimului punct PLOT-at
1	<u>23679</u>	P POSN	Numărul poziției de scriere pe ecran
1	23680	PR CC	Octetul mai puțin semnificativ al adresei pentru noua poziție la care se imprimă prin LPRINT
1	23681	-	Nefolosit
2	23682	ECHO E	Numărul coloanei și al liniei
2	<u>23684</u>	DF CC	Adresa de afișare pe ecran prin PRINT
2	<u>23686</u>	DFCCL	Idem pentru partea de jos a ecranului
X1	<u>23688</u>	S POSN	Numărul coloanei pentru PRINT
X1	<u>23689</u>	-	Numărul liniei pentru PRINT
x2	<u>23690</u>	SPOSNL	Ca S POSN dar pentru partea de jos a ecranului
1	23692	SCR CT	Numără defilările de ecran
1	<u>23693</u>	ATTR P	Culoarea curentă
1	<u>23694</u>	MASK P	Folosit pentru culori transparente
N1	<u>23695</u>	ATTR T	Culori temporare
N1	<u>23696</u>	MASK T	Ca MASK P dar temporar

1	23697	PFLAG	Alți indicatori
N30	23696	MEMBOT	Arie memorie calculator
2	23728	-	Nefolosit
2	23730	RAMTOP	Adresa ultimului octet din aria sistemului BASIC
2	23732	P-RAMT	Adresa ultimului octet de RAM

Instrucțiunea/comanda **PEEK** permite citirea conținutului unui octet de memorie, iar instrucțiunea/comanda **POKE** permite modificarea acestui conținut. Conținutul octetului de modificare este un număr cuprins între 0 și 255 (numărul maxim ce se poate reprezenta cu un octet=8 biți). Pentru numere mai mari ca 255 se codifică valoarea adresei pe doi octeți, cel de al doilea avînd o pondere de 256 ori superioară primului octet:

PEEK (adr) +256*PEEK (adr+1)

unde "adr" este adresa octetului mai puțin semnificativ.

De exemplu, dacă se dorește să se știe care este *RAMTOP*-ul la calculatoarele compatibile cu *ZX-SPECTRUM*, se va folosi adresa variabilei de sistem *RAMTOP* din tabelul 3.1 (adică 23730) și cu comanda

PRINT PEEK 23730+256*23731

se va obține numărul 65367.

Comanda **PRINT PEEK (PEEK 23730+256*PEEK 23731)** afișează 62 (conținutul primului octet).

Pentru a se determina octetul semnificativ (h) și cel mai puțin semnificativ (l) pentru un număr mai mare ca 255, notat generic "adr", se procedează astfel în *BASIC* (luînd ca exemplu numărul 29000):

```
10 LET adr=29000:LET h=INT (adr/256):LET l=adr-
256*h: PRINT "Octetul mai puțin
semnificativ: "'TAB 14;"l=";l;"'"Octetul mai
semnificativ: "'TAB 14;"h=";h;"'"Proba: "'
adresa data: ";adr;"=";256*h+l
```

Calculatorul va afișa l=72 și h=113 (deci 29000=72+256*113)

Alte exemple:

```
PRINT PEEK 23675+256*PEEK 23676 afiseaza 65368 (adresa
UDG)
```

```
PRINT PEEK (PEEK 23675+256*PEEK 23676) afișează 0
```

```
PRINT PEEK 23635+256*PEEK23636 afișează 23755 (adresa
```

programului BASIC)
PRINT PEEK 9PEEK 23635+256*PEEK 23636) afișează 0

3.1.4. Codurile caracterelor

Aceste coduri sînt reunite în fig.3.6 și se obțin cu comanda

PRINT CODE "tasta"

Exemple: **PRINT CODE "P"** afișează 80, iar **PRINT CODE "INK"** afișează 217.

Invers, comanda

PRINT CHR\$ număr

afișează caracterul ce are acest cod (număr=0...255).

Exemple: **PRINT CHR\$ 80** afișează P, iar **PRINT CHR\$ 217** afișează INK

	0	1	2	3	4	5	6	7	8	9
0							Print virgulă	EDIT	Prompter stînga	Prompt dreapta
10	Cursor jos	Cursor sus	DELETE	ENTER			INK	PAPER	FLASH	BRIGHT
20	Comandă INVERSE	Comandă OVER	AT control	TAB control			Comandă	Comandă	Comandă	Comandă
30			Blanc	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[/]	↑	--	£	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	--	©	□	■
130	▣	▤	▥	▦	▧	▨	▩	▪	▫	▬
140	▭	▮	▯	▰	GRAFIC A	GRAFIC B	GRAFIC C	GRAFIC D	GRAFIC E	GRAFIC F
150	GRAFIC G	GRAFIC H	GRAFIC I	GRAFIC J	GRAFIC K	GRAFIC L	GRAFIC M	GRAFIC N	GRAFIC O	GRAFIC P
160	GRAFIC Q	GRAFIC R	GRAFIC S	GRAFIC T	GRAFIC U	RND	INKEY\$	PI	FN	POINT
170	SCREEN\$	ATTR	AT	TAB	VALS	CODE	VAL	LEN	SIN	COS
180	TAN	ASN	ACS	ATN	LN	EXP	INT	SQR	SGN	ABS
190	PEEK	IN	USR	STR\$	CHR\$	NOT	BIN	OR	AND	<-

200	> =	< >	LINE	THEN	TO	STEP	DEF FN	CAT	FORMAT	MOVE
210	ERASE	OPEN#	CLOSE#	MERGE	VERIFY	BEEP	CIRCLE	INK	PAPER	FLASH
220	BRIGHT	INVERSE	OVER	OUT	LPRINT	LUST	STOP	READ	DATA	RESTORE
230	NEW	BORDER	CONTINUE	DIM	REM	FOR	GO TO	GO SUB	INPUT	LOAD
240	LIST	LET	PAUSE	NEXT	POKE	PRINT	PLOT	RUN	SAVE	RANDOMIZE
250	IF	CLS	DRAW	CLEAR	RETURN	COPY				

Fig.3.6

OBSERVAȚIE: informațiile numerice oferite la paragraful 3.1 vor fi folosite în programele ce urmează

3.2. ÎNCĂRCAREA ÎN MEMORIE

a) Încărcarea registrelor simple

În cele ce urmează sînt oferite exemplificări de folosire a instrucțiunilor limbajului de asamblare și programe scrise în acest limbaj. Exemplificările vor fi analizate în tabele la care forma cea mai simplă este următoarea

Nr.liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie
-----------	------------------	--

Atunci cînd analiza implică referiri la starea flagurilor sau a stivei se vor adăuga rubrici noi la acest tabel. De asemenea, orice modificare de valori se va evidenția prin scriere cu caractere îngroșate subliniate.

3.2.1. Încărcarea registrelor (adresarea directă)

Instrucțiunea de încărcare este abreviată LD (de la LOAD). Cea mai simplă formă copiază date dintr-un registru $r' = \{B, C, D, E, H, L, A\}$ în alt registru $r = \{B, C, D, E, H, L, A\}$ și are mnemonica

LD r,r'

Exemplu: LD A,B (echivalent cu LET A=B), care semnifică "încarcă conținutul registrului B în acumulatorul A".

Aplicație: LD A, 20 ; A=20

LD E,A ; E=20

De asemenea, se poate încărca orice registru cu un număr

$n=0\dots255$:

LD r,n ; $r = \{R,C,D,E,H,L,A\}$; exemplu: **LD E,30**

Registrul **A** este singurul care poate fi încărcat cu conținutul unei locații de memorie

LD A, (nn) ; $nn=0\dots65535$; exemplu: **LD A, (30000)**

Analog se poate încărca o locație de memorie cu conținutul acumulatorului **A**

LD (nn),A ; exemplu : **LD (30000),A**

Celelalte registre simple (**B,C,D,E,H,L**) nu pot fi încărcate direct din memorie; asemenea operații necesită două instrucțiuni. De exemplu

LD A,(nn) sau **LD A,C**

LD C,A **LD (nn),A**

Acest mod de încărcare poartă denumirea de adresare directă.

Rezumând instrucțiunile de încărcare pe 8 biți sînt următoarele:

- **LD r,r'** ; $r, r' = \{B,C,D,E,H,L,A\}$; **LET r=r'**
- **LD r,n** ; $n=0\dots255$; **LET r=n**
- **LD A,(nn)** ; $nn=0\dots65535$; **LET A=PEEK nn**
- **LD (nn),A** ; $nn=0\dots65535$; **POKE nn,A**

În loc de a se scrie o adresă în program se poate folosi o etichetă scrisă înaintea instrucțiunii. Această etichetă poate primi o valoare inițială , utilizînd instrucțiunea asamblorului GENS3M21 abreviată

DEFB valoare numerică

Exemplificarea 3.1:

10	ORG 32000
20	ENT 32000
30	LD A, (ET1)
40	LD (22528) ,A
50	LD E,25
60	LD A,E
70	LD (22530) ,A
80	LD A, (22530)
90	LD (22529) ,A
100	LD (22528) ,A
110	LD A,E
120	LD (ET1) ,A
130	RET
140	ET1 DEFB 7

Analiza programului este următoarea:

Nr. liniei	Ce se realizează	Conținutul registrelor
30	Registrul A se încarcă cu valoarea din locația de memorie ET1	A=0; E=0
40	Locația de memorie 22528 se încarcă cu valoarea din registrul A	A=7
50	Registrul E se încarcă cu numărul 25	A=7
60	Registrul A se încarcă cu valoarea din registrul E	A=7; E=25
70	Locația de memorie 22529 se încarcă cu valoarea din registrul A	A=25 ; E=25
80	Registrul A se încarcă cu valoarea din locația de memorie 22530	A=48 ; E=25
90	Locația de memorie 22529 se încarcă cu valoarea din registrul A	A=48 ; E=25
100	Locația de memorie 22528 se încarcă cu valoarea din registrul A	A=48 ; E=25
110	Registrul A se încarcă cu valoarea din registrul E	A=48 ; E=25
120	Locația de memorie ET1 se încarcă cu valoarea din registrul A	A=25 ; E=25
130	Adresa de întoarcere este scoasă din stivă	A=25 ; E=25

Exemplul 3.1: introducerea numărului 45 în locația de memorie 60000

```

10          ORG 60000
20          ENT 60000
30          LD A,45
40          LD B,0           ;octetul high
50          LD C,A           ;C=45
60  ZEND      RET

```

Se assemblează programul și trecînd în *BASIC*, la comanda **PRINT** **USR 60000** calculatorul va afișa valoarea 45.

Iată două variante la acest program:

```

10          ORG 60000
20          ENT 60000
30          LD B,0
40          LD C,45

```


50	ZEND	RET
10		ORG 60000
20		ENT 60000
30		LD A, (ET)
40		LD B, 0
50		LET C, A
60	ZEND	RET
70	ET	DEFB 45

Ambele rutine se activează cu **PRINT USR 60000**.

b) Încărcarea registrelor duble

Registrele duble $dd = \{BC, DE, HL\}$ pot păstra numere $nn = 0 \dots 65535$. Prin convenție octetul superior (cel mai semnificativ h) este păstrat în primul registru (B,D,H), adică $256 \cdot h$, iar octetul inferior (cel mai puțin semnificativ) este păstrat în al doilea registru (C,E,L). Instrucțiunea este

LD dd, nn ; exemplu: **LD HL, 50000**

De asemenea, se poate încărca un registru dublu cu conținutul unei locații de memorie (nn):

LD dd, (nn); exemplu: **LD BC, (42000)**

respectiv conținutul unei locații de memorie cu un registru dublu

LD (nn), dd; exemplu: **LD (42000), DE**

Indicatorul de stivă SP se poate încărca cu conținutul registrelor $pp = \{HL, IX, IY\}$, folosind instrucțiunea

LD SP, pp ; exemplu: **LD SP, HL**

Nu există instrucțiuni care să încarce un registru dublu cu conținutul altui registru dublu: există însă instrucțiunea

EX DE, HL

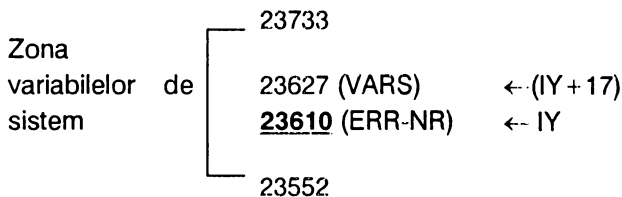
care schimbă conținutul registrului dublu DE cu conținutul registrului HL. Se menționează că mnemonica LD BC, HL nu este valabilă; ea se poate însă simula după cum urmează:

LD B, H

LD C, L

Registrele de index IX, IY pot lua locul registrului dublu HL în aproape toate instrucțiunile, avînd avantajul că adresarea se scrie ($IX + d$) sau ($IY + d$), calculîndu-se ca suma dintre conținutul registrului de index IX/IY și deplasamentul d specificat de instrucțiune. Este util de știut că atunci cînd calculatorul este pus sub tensiune, el afectează registrului IY

valoarea **23610** (adresa variabilei de sistem ERR-NR, conform tab.3.1). Aceasta permite ca prin diferite valori date deplasamentului *d*, să se acționeze asupra unor variabile de sistem, așa cum rezultă din schema următoare:



Este ușor de observat că registrele de index pot fi utilizate pentru parcurgerea tabelor, când deplasamentul *d* poate fi coloana iar registrul *IX* poate fi indicatorul de linie.

Rezumând, instrucțiunile pentru încărcarea registrelor duble sînt:

- LD *dd,nn* ; *dd* = {BC,DE,HL}; *nn* = 0...65535 ; LET *dd* = PEEK *nn* + 256 * PEEK (*nn* + 1)
- LD *dd,(nn)* ; *dd* = {BC,DE,HL}; *nn* = 0...65535: LET *dd* = PEEK *nn* + 256 * PEEK (*nn* + 1)
- LD (*nn*),*dd* ; *nn* = 0...65535; *dd* = {BE,DE,HL}; POKE *nn*,*d1*: POKE *nn* + 1,*d2* sau POKE *nn*, *dd* - INT(*dd*/256)*256: POKE (*nn* + 1), INT (*dd*/256)
- LD SP,*pp* ; *pp* = {HL,IX,IY}; LET SP = *pp*
- EX DE, HL

Exemplificarea 3.2:

```

10      ORG 32000
20      ENT32000
30      LD DE,256
40      LD E,4
50      LD (ET1),DE
60      LD (ET2),DE
70      LD A,2
80      LD (ET3),A
90      LD HL,(ET2)
100     EX DE,HL
110     LD D,0

```

```

120          RET
130 ET1      DEFW 32000
140 ET2      DEFB 0
150 ET3      DEFB 0

```

Programul folosește o nouă directivă de asamblare a asambleurului GENS3 abreviată DEFW prin care asambleurul rezervă două locații de memorie etichetei asociate. Numărul scris după DEFW reprezintă valoarea inițială. În acest program este ilustrată folosirea instrucțiunilor de încărcare a registrelor duble și se demonstrează că un registru dublu sau o locație de memorie dublă sînt formate respectiv din două registre simple sau două locații de memorie simple.

Nr. liniei	Ce se realizează	Conținutul registrelor
30	Registrul dublu DE se încarcă cu numărul 256	A=0; D=0; E=0; HL=0; DE=0
40	Registrul E se încarcă cu numărul 4	D=1; DE=256
50	Locația de memorie ET1 se încarcă cu valoarea din registrul dublu DE	D=1; E=4; DE=260
60	Locația de memorie ET2 se încarcă cu valoarea din registrul dublu DE	D=1; E=4; DE=260
70	Registrul A se încarcă cu numărul 2	D=1; E=4; DE=260
80	Locația de memorie ET3 se încarcă cu valoarea din registrul A	A=2; D=1; E=4; DE=260
90	Registrul dublu HL se încarcă cu valoarea din locația de memorie ET2	A=2; D=1; E=4; DE=260
100	Registrul dublu DE se schimbă cu valoarea din registrul dublu HL	A=2; E=4; HL=516; DE=260; D=1
110	Registrul D se încarcă cu numărul 0	A=2; D=2; E=4; HL=260; DE=516
120	Adresa de întoarcere este scoasă din stivă	A=2; D=0; E=4; HL=260; DE=4

Exemplul 3.2: expresia BASIC

PRINT PEEK 23684+256*PEEK 23685

afișează numărul 16384, valoarea care dă poziția de scriere PRINT din linia 0, coloana 0 (v.fig.3.4). Programul echivalent în limbaj de asamblare folosește variabila de sistem DF CC (cu adresa 23684-v.tab.3.1) care conține adresa de afișare pe TV prin PRINT:

```

10          ORG 60000
20          ENT 60000
30          LD BC, (23684) ; (DF CC)
40  ZEND    RET

```

Revenind în BASIC după asamblarea programului și tastănd PRINT USR 60000 se afișează 16384.

Exemplul 3.3. cu comanda

PRINT PEEK (PEEK 23684+256*PEEK 23685)

se obține conținutul primului octet de la poziția PRINT. Programul în limbaj de asamblare are forma următoare:

```

10          ORG 60000
20          ENT 60000
30          LD HL, (23684)
40          LD B,0
50          LD C, (HL)
60  ZEND    RET

```

Tastănd PRINT USR 60000 se va obține același rezultat.

Exemplul 3.4: variabile S-POSN conține coordonatele poziției PRINT (numărul coloanei la adresa 23688 și numărul liniei la adresa 23689 v. tab. 3.1). Programele care dau numărul de linie, respectiv de coloană sînt:

a) Numărul coloanei

```

10          ORG 60000
20          ENT 60000
30          LD BC, (23688)
40          LD B,0
50          RET

```

b) Numărul liniei

```

60          ORG 60050
70          LD BC, (23689)
80          LD B,0
90  ZEND    RET

```

Cu comanda **PRINT USR 60000: PRINT USR 60050** se obțin

numărul linie și respectiv numărul coloanei.

3.2.2. Adresarea indirectă

În paragraful anterior s-au utilizat instrucțiuni în care adresele locațiilor de memorie erau specificate direct în instrucțiune, motiv pentru care adresarea este numită *directă*. O altă modalitate de indicare a unei locații de memorie este adresarea indirectă, care constă în folosirea numărului păstrat într-un registru dublu. De exemplu:

LD B,(HL)

permite încărcarea în registrul B a conținutului locației de memorie a cărei adresă se află în registrul dublu HL.

Toate registrele simple $r = \{B, C, D, E, H, L, A\}$ pot fi încărcate utilizând registrul dublu HL.

Similar, locațiile de memorie pot fi încărcate din orice alt registru simplu $r = \{B, C, D, E, H, L, A\}$ folosind conținutul registrului dublu HL ca adresă. Exemplu: LD (HL),C.

Folosirea registrelor duble BC și DE pentru adresarea indirectă este limitată la registrul acumulator A. Exemple: LD (DE),A ; LD (BC),A. Rezumând, instrucțiunile pentru adresarea indirectă sînt:

- | | | |
|-------------|------------------------|-----------------|
| • LD r,(HL) | ; r = {B,C,D,E,H,L,A}; | LET r=PEEK(HL) |
| • LD (HL),r | ; r = {B,C,D,E,H,L,A}; | POKE HL,r |
| • LD A,(BC) | | ;LET A=PEEK BC |
| • LD A,(DE) | | ; LET A=PEEK DE |
| • LD (BC),A | | ; POKE BC,A |
| • LD (DE),A | | ; POKE DE,A |

Exemplificarea 3.3:

```

10          ORG 32000
20          ENT 32000
30          LD HL,ET1
40          LD C,HL
50          LD HL,ET2
60          LD B,(HL)
70          LD A,(BC)
80          LD DE,22528
90          LD (DE),A
100         LD (HL),0

```

110 LD BC, 32021
 120 LD (BC), A
 130 RET
 140 ET1 DEFB 20
 150 ET2 DEFB 125

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie
30	Registrul dublu HL se încarcă cu ET1	A=0; B=0; C=0; HL=0; BC=0
40	Registrul C se încarcă cu valoarea din locația de memorie (HL)	<u>HL = 32020; (HL) = 20;</u> <u>(BC) = 243; (DE) = 243</u>
50	Registrul dublu HL se încarcă cu ET2	C=20; HL=32020; <u>BC=20;</u> (HL) = 20; (BC) = 256; (DE) = 243
60	Registrul B se încarcă cu valoarea din locația de memorie (HL)	C=20; <u>HL = 32021;</u> <u>(HL) = 125; (BC) = 255;</u> (DE) = 243
70	Registrul A se încarcă cu valoarea din locația de memorie (BC)	<u>B = 125;</u> C=20; HL = 32021; <u>BC = 32020;</u> (HL) = 125; <u>(BC) = 20;</u> (DE) = 243
80	Registrul dublu DE se încarcă cu numărul 22528	<u>A = 20;</u> B = 125; C = 20; HL = 32021; BC = 32020; (HL) = 125; (BC) = 20; (DE) = 243
90	Locația de memorie (DE) se încarcă cu valoarea din registrul A	A = 20; B = 125; C = 20; HL = 32021; BC = 32020; <u>DE = 22528;</u> (HL) = 125; (BC) = 20; <u>(DE) = 48</u>
100	Locația de memorie (HL) se încarcă cu numărul 0	A = 20; B = 125; C = 20; HL = 32021; BC = 32020; DE = 22528; (HL) = 125; (BC) = 20; <u>(DE) = 20</u>

110	Registrul dublu BC se încarcă cu numărul 32021	A=20; B=125; C=20; HL=32021; BC=32020; DE=22528; (HL)=0 ; (BC)=20 ; (DE)=20
120	Locația de memorie (BC) se încarcă cu valoarea din registrul A	A=20; B=125; C=21 ; HL=32021; BC=32021 ; DE=22528; (HL)=0; (BC)=0; (DE)=20
130	Adresa de întoarcere este scoasă din stivă	A=20; B=125; C=21; HL=32021; BC=32021; DE=22528, (HL)=20 ; (BC)=20 ; (DE)=20

Exemplul 3.5: introducerea numărului 1000 în locația de memorie

60000

```

10          ORG 60000
20          ENT 60000
30          LD HL,1000
40          LD B,H
50          LD C,L
60  ZEND    RET

```

Tastînd **PRINT USR 60000**, calculatorul afișează 1000

3.3. OPERAȚII ARITMETICE DE BAZĂ

În această categorie de operații se includ:

- adunarea;
- scăderea;
- incrementarea și decrementarea.

3.3.1. Adunarea și flagul Carry (C_i)

În registrul fanioanelor F , flagurile sînt ordonate conform schemei următoare:

bitul	7	6	5	4	3	2	1	0
	S	Z	-	H	-	P/V	N	C_i

unde S-flagul de semn, Z-flagul zero, H-flagul de transport pe jumătate, P/V-flagul de paritate și depășire, N-flagul de adunare/scădere și C_i -flagul de transport.

Microprocesorul Z80 permite adunarea registrelor simple și a celor duble folosind instrucțiunile **ADD** și **ADC** după cum urmează:

a) Toate adunările registrelor simple se fac cu instrucțiunea ADD și implică obligatoriu acumulatorul A. Astfel, la acumulatorul A pot fi adunate:

- un număr $n=0\dots255$ când mnemonica este **ADD A,n** (ex: **ADD A,6**);
- conținutul unui registru $r \in \{B,C,D,E,H,L,A\}$, când mnemonica este **ADD A,r** (ex: **ADD A,B**);
- conținutul unei locații de memorie adresate indirect prin registrul dublu HL, mnemonica fiind **ADD A,(HL)**.

Rezultatul se păstrează în acumulatorul A, iar sursa adunării rămîne nemodificată. Se va reține deci că nu se poate aduna un octet (constantă, registru sau conținutul unei locații de memorie) decît la registrul acumulator A. Toate aceste instrucțiuni influențează flagurile S,Z,P/V și C_i în funcție de rezultatul adunării. De pildă **ADD A,0** (LET $A=A$) lasă A neschimbat dar face $C_i=0$ și $P/V=0$, fiind o modalitate de a zerofica flagul C_i .

b) Adunarea registrelor duble se face cu instrucțiunea ADD și implică obligatoriu registrul dublu HL. Astfel, la registrul HL se pot aduna numai registrele BC și DE, utilizînd instrucțiunile

ADD HL, BC ; ADD HL,DE

Rezultatul adunării se păstrează în HL iar celelalte registre rămîn nemodificate.

Se menționează că adunarea registrelor este corectă numai dacă rezultatul adunării este mai mic decît numărul maxim ce poate fi păstrat într-un registru (255, respectiv 65535). Dacă acest rezultat este mai mare decît numărul maxim, se generează transport și flagul $C_i=1$ (altfel $C_i=0$).

c) Microprocesorul Z80 mai dispune și de o altă formă de adunare atît pentru registrele simple, cît și pentru cele duble, cunoscută sub denumirea de "adunare cu Carry" și abreviată **ADC**. Această instrucțiune este similară cu **ADD** cu deosebirea că dacă flagul $C_i=1$ înaintea adunării, atunci rezultatul este incrementat cu 1 (adică se adaugă C_i).

Instrucțiunea ADC se folosește pentru adunarea a două numere de orice lungime.

Rezumând, instrucțiunile de adunare sînt următoarele: •

Instrucțiunea
ADD

ADD A,n ;n=0...255; LET A=A+n
 ADD A,r ;r={B,C,D,E,H,L,A};
 LET A=A+r
 IF (A+r) > 255 THEN LET C_i = 1
 IF (A+r) = 256 THEN LET Z = 1
 IF (A+r) > 127 AND (A+r) < 256
 THEN LET P/V = 0
 ADD A,(ss) ; (ss) = {(HL),(IX+d),(IY+d)};
 LET A=A+PEEK ss
 ADD A,0 face C_i = 0
 ADD HL,rr ; rr = {BC,DE}; LET HL=HL+rr

Instrucțiunea
ADC

ADC A,n ;n=0...255; LET A=A+n+C_i
 ADC A,r ;r={B,C,D,E,H,L,A};
 LET A=A+r+C_i
 ADC A,(HL) ;LET A=A+PEEK HL+C_i
 ADC HL,rr ; rr = {BC,DE};
 LET HL=HL+rr+C_i

Toate instrucțiunile ADC influențează flagurile S,Z,P/V,C_i. Nu există instrucțiune pentru a aduna o contantă la HL; aceasta se simulează astfel:

```
LD BC, n
ADD HL, BC
```

Dacă BC este folosit în alt scop, simularea este

```
LD A,L
ADD A,n
LD L,A
LD A,H
ADC A,0
LD H,A
```

Exemplificarea 3.4

```
10      ORG 32000
20      ENT 32000
36      LD DE,8740
40      LD BC,1260
50      LD A,E
60      ADD A,C
70      LD C,A
80      LD A,D
90      ADC A,B
100     LD B,A
110     LD HL,ET1
120     LD A,15
130     ADD A,(HL)
140     RET
150     ET1      DEFB 25
```

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Flagul C _i
30	Registrul DE se încarcă cu numărul 8740	A=0; B=0; C=0; D=0; E=0; HL=0; BC=0; DE=0; (HL)=0	C _i =0
40	Registrul BC se încarcă cu numărul 1260	D=34; E=36; DE=8740; (HL)=243	idem

50	Registrul A se adună cu valoarea din registrul E	B=4; C=236; D=34; E=36; BC=1260; DE=8740; (HL)=243	idem
60	Registrul A se adună cu valoarea din registrul C	A=36; B=4; C=236; D=34; E=36; BC=1260; DE=8740; (HL)=243	idem
70	Registrul C se încarcă cu valoarea din registrul A	A=16; B=4; C=236; D=34; E=36; BC=1260; DE=8740; (HL)=243	C_i=1
80	Registrul A se încarcă cu valoarea din registrul D	A=16; B=4; C=16; D=34; E=36; BC=1040; DE=8740; (HL)=243	idem
90	Registrul A se adună cu C _i și valoarea din registrul B	A=34; B=4; C=16; D=34; E=36; BC=1040; DE=8740; (HL)=243	idem
100	Registrul B se încarcă cu valoarea din registrul A	A=39; B=4; C=16; D=34; E=36; BC=1040; DE=8740; (HL)=243	C_i=0
110	Registrul HL se încarcă cu ET1	A=39; B=39; C=16; D=34; E=36; BC=10000; DE=8740; (HL)=243	idem

120	Registrul A se încarcă cu numărul 15	A=39; B=39; C=16; D=34; E=36; BC=10000; DE=8740; (HL)=25; HL=32019	idem
130	Registrul A se adună cu valoarea din locația (HL)	A=15 ; B=39; C=16; D=34; E=36; BC=10000; DE=8740; (HL)=25; HL=32019	idem
140	Adresa de întoarcere este scoasă din stivă	A=40 ; B=39; C=16; D=34; E=36; BC=10000; DE=8740; (HL)=25; HL=32019	idem

Exemplul 3.6: adunare module 256.

```

10          ORG 60000
20          ENT 60000
30          LD A,0           ; A=0
40          ADD A,0          ; A+0=0; Ci=0
50          LD C,A           ; C=A=0
60          LD B,0           ; B=0
70  ZEND          RET

```

Se revine în BASIC (după asamblarea rutinei) și se tastează:

```

10 CLS: INPUT "Introduceti nr.x",x: POKE 60001,x
20 INPUT "Introduceti nr.de adunat y",y: POKE
   60003,y
30 PRINT x;"+";y;"=";: PRINT USR 60000
40 PRINT AT 20,6; "ALTA ADUNARE (d/n) ?": PAUSE 0
50 GO TO 10*(INKEYS ="d" OR INKEY$="D")+
   60*(INKEY$="n" OR INKEY$="N")
60 CLS : STOP

```

Dacă se tastează, ca răspuns la cerința adresată prin instrucțiunea INPUT din liniile 10 și 20, cifrele 75 și 25 va rezulta rezulta 100. Dar dacă se tastează 150 și apoi 200 rezultatul va fi 94 (adică $350-256=94$),

respectiv rezultatul este modulo 256.

Exemplul 3.7: adunare modulo 65536

```

10          ORG 60000
20          ENT 60000
30          LD HL,0           ; HL=0
40          LD BC,0          ; BC=0
50          ADD HL,BC        ; HL+BC=0
60          LD B,H           ; B=H
70          LD C,L           ; C=L
80 ZEND     RET

```

Programul în Basic care folosește rutina în cod-mașină este

```

10 CLS; INPUT "Introduceti nr. x",x: POKE 60001,x-
  INT(x/256)*256: POKE 60002,INT(x/256)
20 INPUT "Introduceti nr. de adunat y",y: POKE
  60004,y-INT(y/256)*256: POKE 60005,INT(y/256)
30 PRINT x;"+";y;"=";: PRINT USR 60000
40 PRINT AT 20,6;"ALTA ADUNARE (d/n)?" : PAUSE 0
50 GO TO 10 *(INKEY$="d" OR INKEY$="D")+
  .60*(INKEY$="n" OR INKEY$="N")
60 CLS : STOP

```

Dacă se introduc valorile 14500 și apoi 560, rezultatul va fi 15060, dar dacă se introduc valorile 65535 și respectiv 3, rezultatul va fi 2 (adică $65538-65536=2$), respectiv modulo 65536.

Pentru a aduna direct două numere, ele se introduc în registrele HL și BC (numerele pot fi cuprinse în gama 0...65535):

Exemplul 3.8: adunarea numerelor 8000 și 2000

```

10          ORG 60000
20          ENT 60000
30          LD HL,8000       ; HL=8000
40          LD BC,2000       ; BC=2000
50          ADD HL,BC        ; HL=10000
60          LD B,H
70          LD C,L
80 ZEND     RET

```

Trecând în Basic și tastând PRINT USR 60000, calculatorul afișează 10000 (rezultatul adunării).

O variantă a acestui program este redată în continuare, unde s-au folosit etichete și directiva de asamblare DEFW:

```

10          ORG 60000
20          ENT 60000
30          LD HL, (NR1)      ; NR1-primul numar
40          LD BC, (NR2)     ; NR2-al doilea
                               numar
50          ADD HL, BC       ; HL=HL+BC=NR1+NR2
60          LD B, H
70          LD C, L
80  ZEND    RET
90  NR1     DEFW 8000        ; NR1=8000
100 NR2     DEFW 2000       ; NR2=2000

```

Trecînd în Basic și tastînd PRINT USR 60000 se obține rezultatul adunării numerelor NR1 cu NR2 (adică 10000).

3.3.2. Scăderea cu fanionul Carry (C_i)

a) Scăderea registrelor simple $r = \{B, C, D, E, H, L, A\}$ are loc numai cu acumulatorul A. Mnemonica **SUB** se scrie întotdeauna fără A, existînd formele

SUB n (scade n din A) unde $n = 0..255$

SUB r (scade r din A)

SUB (HL) (scade din A conținutul locației de memorie adresate indirect prin (HL))

Rezultatul scăderii se păstrează în A iar flagul $C_i = 1$ dacă rezultatul este în afara domeniului 0...255.

Se menționează că *nu există instrucțiune SUB pentru scăderea registrelor duble.*

Instrucțiunea **SUB A** face $A = 0$ și $C_i = 0$, reprezentînd o modalitate de a zerofica flagul C_i cînd nu este nevoie de acumulatorul A.

b) Scăderea numerelor de orice lungime se face cu instrucțiunea SBC. rezultatul fiind decrementat cu 1 dacă flagul C_i era 1 înainte de scădere. Este important ca flagul C_i să fie pus pe 0 la **SBC HL, BC** și respectiv **SBC HL, DE**. Prin urmare, mnemonicele sînt

SBC A, nn ; $nn = 0..65535$

SBC A, r ; $r = \{B, C, D, E, H, L, A\}$

SBC a, (ss) ; $(ss) = \{(HL), (IX+d), (IY+d)\}$

se scad din A
împreună cu C_i

Pentru a simula instrucțiunea SBC BC, DE care nu există se scrie

LD H,B

LD L,C

SBC HL,DE

SBC HL,rr; rr={BC,DE,HL,SP}

Observatii: 1) Flagul C_i este pus pe 1 (se spune "setează C_i ") cu instrucțiunea SCF.

2) Instrucțiunea care completează flagul C_i este CCF (realizează $C_i = 1 - C_i$).

Rezumînd cele prezentate, instrucțiunile de scădere sînt următoarele:

- SUB n ; n=0...255 ; LET A=A-n
- SUB r ; r={B,C,D,E,H,L,A} ; LET A=A-r; SUB A face A=0 și $C_i=0$
- SUB (HL) ; LET A=A-PEEK HL Nu există SUB pentru registrele duble.
- SBC A, nn ; nn=0...65535 ; LET A=A-nn- C_i
- SBC A,r ; r={B,C,D,E,H,L,A} ; LET A=A-r- C_i
- SBC A,(ss) ;(ss)={{(HL),(IX+d),(IY+d)} ; LET A=A-PEEK ss- C_i
- SBC HL,rr ;rr={BC,DE,HL,SP} ; LET HL=HL-rr- C_i

Observatii:

1) Pentru a face SUB HL,BC care nu există, se face $C_i=0$ și apoi se folosește SBC:

ADD A,0 ; $C_i=0$

SBC HL,BC

2) Pentru a face SBC BC,DE care nu există se simulează

LD H,B

LD L,C

SBC HL,DE

LD B,H

LD C,L

3)SCF face $C_i=1$ și CCF realizează $C_i=1-C_i$

Exemplul 3.9.: scăderea modulo 65536.

```

10          ORG 60000
20          ENT 60000
30          LD HL,0
40          LD BC,0
50          ADD A,0
60          SBC HL,BC

```

```

70          LD B,H
80          LD C,L
90  ZEND    RET

```

Revenind în Basic după asamblarea programului, se tastează

```

10  CLS: INPUT "Introduceti nr.x",x: POKE 60001,x-
    INT(x/256)*256: POKE 60002,INT(x/256)
20  INPUT "Introduceti nr.de scazut y",y: POKE
    60004,y-INT(y/256)*256: POKE 60005,INT(y/256)
30  PRINT x;"-";y;"=";: PRINT USR 60000
40  PRINT AT 20,7;"ALTA SCADERE (d/n)?: PAUSE 0
50  GO TO 10*(INKEY$="d" OR INKEY$="D")+
    60*(INKEY$="n" OR INKEY$="N"
60  CLS : STOP

```

Dacă se tastează succesiv

```

1000 și 1001 rezultă 65535
2000 și 2001 rezultă 65535
1000 și 1002 rezultă 65534
65534 și 65535 rezultă 65535
65535 și 65534 rezultă 1

```

Exemplul 3.10: cunoașterea memoriei libere

```

10          ORG 42000          ;MEMORIA LIBERA
20          ENT 42000
30          LD HL,0
40          ADD HL,SP
50          LD BC,(23653)     ;(STKEND) -v. tab. 3.1
60          SBC HL,BC
70          LD B,H
80          LD C,J
90  ZEND    RET

```

Cu PRINT USR 42000 se obține rezultatul.

Exemplul 3.11: lungimea programului BASIC

```

10          ORG 59988          ; LUNGIMEA
                                PROGRAMULUI BASIC
20          ENT 59988
30          LD DE,(23635)     ;(PROG) -v. tab. 3.1
40          LD HL,(23627)     ;(VARS) -v. tab. 3.1
50          SBC HL,DE
60          LD B,H
70          LD C,L
80  ZEND    RET

```


130 DEC HL
 140 LD (HL), C
 150 RET

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie
30	Registrul C se încarcă cu numărul 5	A = 0; C = 0; HL = 0; (HL) = 0
40	Registrul HL se încarcă cu numărul 22528	C = 5; (HL) = 240
50	Locația (HL) se încarcă cu valoarea din registrul C	C = 5; HL = 22528 ; (HL) = 48
60	Registrul C se incrementează cu 1	C = 5; HL = 22528; (HL) = 5
70	Registrul HL se incrementează cu 1	C = 6 ; HL = 22528; (HL) = 5
80	Locația (HL) se încarcă cu valoarea din registrul C	C = 6; HL = 22529 ; (HL) = 48
90	Registrul C se incrementează cu 1	C = 6; HL = 22529; (HL) = 6
100	Registrul HL se încarcă cu numărul 22550	C = 7 ; HL = 22529; (HL) = 6
110	Locația (HL) se încarcă cu valoarea din registrul C	C = 7; HL = 22550 ; (HL) = 48
120	Registrul C se decrementează cu 1	C = 7; HL = 22550; (HL) = 7
130	Registrul HL se decrementează cu 1	C = 6 ; HL = 22550; (HL) = 7
140	Locația (HL) se încarcă cu valoarea din registrul C	C = 6; HL = 22549 ; (HL) = 48
150	Adresa de întoarcere este scoasă din stivă	C = 6; HL = 22549; (HL) = 6

3.4. INSTRUCȚIUNI CARE INFLUENȚEAZĂ VALOAREA UNUI BIT

Microprocesorul Z80 permite operații asupra biților dintr-un octet.

Fiecare bit dintr-un registru sau locație de memorie adresată indirect de HL poate fi pus:

- pe 1, adică *SETat*, folosind instrucțiunea **SET**;
- pe 0, adică *RESetat*, folosind instrucțiunea **RES**.

Pentru a testa starea unui bit dintr-un registru sau locație de memorie se folosește instrucțiunea **BIT** care influențează flagul Z astfel:

- dacă bitul testat este 0, atunci $Z = 1$;
- dacă bitul testat este 1, atunci $Z = 0$.

S-au prezentat, de asemenea, instrucțiunile

SCF care realizează $C_i = 0$

CCF care determină $C_i = 1 - C_i$.

Rezumînd, instrucțiunile care influențează valoarea unui bit sînt:

- | | |
|-----------------------------|---|
| • SET b, (HL) | ; bit = 0...7 (numărul bitului); $b_r = 1$ |
| • RES b,r; b = 0...7 | ; $r = \{B,C,D,E,H,L,A\}$; $b_r = 0$ |
| • RES b, (mem) | ; $b = 0...7$; (mem) = {(HL), (IX+d), (IY+d)};
$b(\text{mem}) = 0$ |
| • BIT b,r | ; $Z = b_r$ (valoarea complementată a bitului
$b = 0...7$ din registrul $r = \{B,C,D,E,H,L,A\}$ este
copiată în flagul Z) |
| • BIT b, (mem) | ; valoarea complementată a bitului
$b = 0...7$ din celula de memorie
(mem) = {(HL), (IX+d), (IY+d)} este copiată în
flagul Z |
| • SCF | ; $C_i = 0$ |
| • CCF | ; $C_i = 1 - C_i$ |

Exemplificarea 3.6:

10	ORG 32000
20	ENT 32000
30	LD HL, 32020
40	SET 6, (HL)

50	INC (HL)
60	BIT 2, (HL)
70	RES 3, (HL)
80	LD B, 5
90	SET 7, B
100	RES 2, B
110	DEC B
120	BIT 5, B
130	RET

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Flaguri
30	Registrul HL se încarcă cu numărul 32020	A=0; B=0; HL=0; (HL)=0	S=0; Z=0; P/V=0
40	Pune pe 1 bitul b6 din locația (HL)	HL=32020; (HL)=104	
50	Locația (HL) se incrementează cu 1	HL=32020; (HL)=97	
60	Testează bitul b2 din locația (HL)	HL=32020; (HL)=98	
70	Pune pe 0 bitul b3 din locația (HL)	HL=32020; (HL)=98	P/V=1; Z=1
80	Registrul 8 se încarcă cu numărul 5	HL=32020; (HL)=98	Idem
90	Pune pe 1 bitul b7 din registrul B	B=5; HL=32020; (HL)=98	Idem
100	Pune pe 0 bitul b2 din registrul B	B=133; HL=32020; (HL)=98	Idem
110	Registrul B se decrementează cu 1	B=129; HL=32020, (HL)=98	Idem
120	Testează bitul b5 din registrul B	B=128; HL=32020; (HL)=98	S=1, P/V=0; Z=0
130	Adresa de întoarcere este scoasă din stivă	B=128; HL=32020; (HL)=98	Z=1, P/V=1

3.5. TRANSFERURI DE BLOCURI DE MEMORIE

Instrucțiunile prezentate anterior efectuau cel mult două operații. Există însă și instrucțiuni care realizează patru operații și din acest motiv sînt foarte puternice întrucît efectuează operații asupra blocurilor de memorie. În mnemonica lor apar literele

LD-încarcă; **I**- incrementează;

D- decrementează; **R** - repetă pînă numărătorul BC ajunge la zero

Rezumînd, instrucțiunile pentru transferul blocurilor de memorie sînt:

- | | | | | | | |
|---|-------------|---|------------|----------|----------|---------|
| • | LDI | ; | (DE)=(HL), | DE=DE+1, | HL=HL+1, | BC=BC-1 |
| • | LDD | ; | (DE)=(HL), | DE=DE-1, | HL=HL-1, | BC=BC-1 |
| • | LDIR | ; | (DE)=(HL), | DE=DE-1, | HL=HL+1, | BC=BC+1 |
| • | LDDR | ; | (DE)=(HL), | DE=DE-1, | B=BC-1 | |

Observatii:

1) La **LDI**, **LDD** conținutul celulei de memorie adresată prin registrul dublu HL este transferat în celula de memorie adresată prin registrul dublu DE. Aceste instrucțiuni afectează flagul P/V care devine 0 dacă după execuție numărătorul BC = 0 (altfel P/V = 1).

2) Instrucțiunile **LDIR**, **LDDR** transferă un bloc de date de lungime egală cu BC, dintr-o zonă de memorie într-alta. Blocul sursă începe la adresa specificată de HL, iar blocul destinație la adresa specificată de DE. Ele sînt similare instrucțiunilor **LDI**, **LDR**, dar se repetă pînă cînd numărătorul BC = 0. Evident, aceste instrucțiuni pun flagul P/V = pe 0.

Exemplificarea 3.7.: transferul treimii superioare a ecranului la mijlocul lui.

10	ORG 60000
20	ENT 60000
30	LD HL,16384
40	LD DE,18432
50	LD BC,2048
60	LDIR
70	RET

Datele 16384, 18432 și 2048 sînt cele indicate în fig.3.2, iar analiza programului este efectuată în tabelul care urmează.

Nr. liniei	Ce se realizează	Conținutul registrelor
30	Registrul HL se încarcă cu numărul 16384	HL=0; BC=0; DE =0
40	Registrul DE se încarcă cu numărul 18432	<u>HL = 16384</u>
50	Registrul BC se încarcă cu numărul 2048	HL = 16384; <u>DE = 18432</u>
60	Transferă un bloc (BC) octeți, de la (HL) la (DE) incrementînd	HL = 16384; DE = 18432; <u>BC = 2048</u>
70	Adresa de întoarcere este scoasă din stivă	<u>HL = 18432; DE = 20480;</u> <u>BC = 0</u>

Exemplul 3.12.: folosirea culorilor

```

10          ORG 60000          ;ATTRIBUTE
20          ENT $
30          LD HL,6400         ;sau LD HL,(23672)
40          LD DE,22528        ;primul octet al
                               zonei de atribute
50          LD BC,768          ;768 locatii care
                               memoreaza fiecare
                               atributele unei
                               matrice 8x8 pixeli

60          LDIR
70          RET

```

Programul BASIC corespunzător este următorul:

```

10 CLS : FOR i=1 TO 7: BEEP .02,i*7: RANDOMIZE USR
   60000: NEXT i
20 BEEP .02,56: CLS

```

Tastînd **POKE 60002,61** vor predomina caracterele negre, iar cu **POKE 60002,0** vor predomina restul culorilor.

Cu mici modificări se realizează efecte numai pe anumite zone ale ecranului; astfel:

- treimea superioară: LD DE,22528
LD BC, 256
- treimea mijlocie: LD DE,22784
LD BC,256
- treimea inferioară: LD DE,23040

```

LD BC,256
- primele două treimi: LD DE 22528
LD BC,2*256
- ultimile două treimi: LD DE,22784
LD BC,2*256
- jumătatea superioară: LD DE,22528
LD BC,384
- jumătatea inferioară: LD DE,22784
LD BC,384

```

Exemplul 3.13.: ecran mozaic

```

10          ORG 60000
20          ENT 60000
30          LD HL,0
40          LD DE,16384
50          LD BC,6144      ;6144-nr octetilor
                               ocupati de ecran

60          LDIR
70  ZEND          RET

```

Programul BASIC care exploatează această rutină are forma:

```

10 BORDER 1: CLS
20 RANDOMIZE USR 60000

```

Pentru efecte numai pe anumite porțiuni ale ecranului se procedează la modificările indicate mai jos:

```

- treimea superioară: LD DE,16384
LD BC,2048
- treimea mijlocie: LD DE,18432
LD BC,2048
- treimea inferioară: LD DE,20480
LD BC,2048
primele două treimi: LD DE 16384
LD BC,2*2048
- ultimile două treimi: LD DE,18432
LD BC,2*2048

```

4. FOLOSIREA INSTRUCȚIUNILOR PENTRU CICLURI, TESTĂRI, ROTAȚII ȘI DEPLASĂRI

Acest capitol pune în relație instrucțiunile care se referă la structurile programelor (cicluri, testări, comparații); ele modifică registrele și flagurile de o manieră diferită pe baza comparațiilor logice, deplasărilor și rotațiilor. Rutinele care vor fi prezentate sînt o introducere în programarea avansată.

4.1. SALTURI ȘI CICLURI (BUCLE)

4.1.1. Salturi necondiționate

Microprocesorul Z80 are un numărător de program PC care păstrează adresa instrucțiunii ce urmează să fie executată. Este evident că prin modificarea conținutului numărătorului de program **PC se poate executa un salt** la orice linie de program. Instrucțiunea pentru salt necondiționat are mnemonica JP și următoarele forme:

- **JP nn** ; nn = 0...65535; GO TO nn; PC = nn
 - **JP pq** ; pq = {HL,IX,IY}; GO TO pq; PC = pq
 - **JP (HL)** salt necondiționat la adresa din registrul HL.
- Instrucțiunea JP HL este interesantă deoarece corespunde la un GO TO X unde X este o variabilă; HL poate rezulta dintr-un calcul ca și X, ceea ce produce un GO TO calculat.

4.1.2. Salturi condiționate

Instrucțiunile pentru salturile condiționate sînt mult mai puternice decît cele de salt necondiționat. Microprocesorul va testa starea unui flag (C_i , Z, P/V,S) după care fie va executa saltul, fie va continua programul în secvență. Aceste instrucțiuni sînt:

- JP NC,nn salt la adresa nn=0...65535 dacă $C_i=0$
 - JP C,nn salt la adresa nn=0...65535 dacă $C_i=1$
 - JP NZ,nn salt la adresa nn=0...65535 dacă Z=0
 - JP Z,nn salt la adresa nn=0...65535 dacă Z=1
 - JP PO,nn salt la adresa nn=0...65535 dacă P/V=0
 - JP PE,nn salt la adresa nn=0...65535 dacă P/V=1
 - JP P,nn salt la adresa nn=0...65535 dacă S=0
 - JP M,nn salt la adresa nn=0...65535 dacă S=1
- Atunci cînd condiția este adevărată PC=nn, iar în caz contrar PC=C+3

Deși este posibilă introducerea adresei de salt sub forma de număr, dacă acesta nu este identic cu adresa corespunzătoare sistemul se poate bloca. Din acest motiv se recomandă folosirea etichetelor scrise în fața instrucțiunii unde se dorește saltul:

```

ET  → instructiune
    → instructiune
    → instructiune
    → JP NZ,ET
    → JP NZ,ET
    → instructiune
    → instructiune
ET1 → instructiune

```

Exemplificarea 4.1:

```

10          ORG 32000
20          ENT $           ;adresa de start
30          LD HL,22528
40          LD A,0
50  ET1     LD (HL),A
60          INC HL
70          INC A
80          JP Z, ET1
90  ET2     DEC (HL)

```

100 DEC HL
 110 JP Z, ET2
 120 RET

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Flaguri
30	Registrul HL se încarcă cu numărul 22528	$A = 0; HL = 0; (HL) = 0$	$C_i = 0; Z = 0$
40	Registrul A se încarcă cu numărul zero	<u>HL = 22528; (HL) = 48</u>	idem
50	Locația (HL) se încarcă cu valoarea din registrul A	$HL = 22528; (HL) = 48;$ <u>A = 0</u>	Idem
60	Registrul HL se incrementează cu 1	$HL = 22528; (HL) = 0;$ $A = 0$	Idem
70	Registrul A se incrementează cu 1	<u>HL = 22529; (HL) = 48;</u> $A = 0$	Idem
80	Dacă $Z = 1$ execută salt la ET1	<u>A = 1;</u> $HL = 22529;$ $(HL) = 48$	Idem
90	Locația (HL) se decrementează cu 1	$A = 1; HL = 22529;$ $(HL) = 48$	Idem
100	Registrul HL se decrementează cu 1	$A = 1; HL = 22529;$ <u>(HL) = 47</u>	Idem
110	Dacă $Z = 1$ execută salt la ET2	$A = 1, \mathbf{HL = 22528};$ $(HL) = 0$	Idem
120	Adresa de întoarcere este scoasă din stivă	$A = 1; HL = 22528;$ $(HL) = 0$	Idem

Exemplificarea este pur didactică deoarece nu se va executa salt niciodată.

4.1.3. Salturi relative

Dificultatea pe care o prezintă salturile necondiționate și condiționate

este aceea că dacă adresa de start a programului se schimbă, toate adresele de salt numerice trebuie modificate. Salturile relative nu prezintă însă această dificultate, deoarece adresa de salt rezultă din adunarea algebrică a valorii curente a contorului program PC cu un *deplasament* $d = \{+128; -127\}$:

- JR d salt la adresa $PC = PC + d$
- JR c,d $c = \{NZ, Z, NC, C\}$; salt la adresa $PC = PC + d$ dacă este îndeplinită condiția

Există un caz particular al saltului relativ condițional deosebit de folosit numit **DJNZ** (decrementează registrul B și salt dacă $B \neq 0$):

DJNZ d ; B=B-1 ; dacă $B \neq 0$ salt la adresa $PC = PC + d$

Instrucțiunea DJNZ permite repetarea unei secvențe de instrucțiuni de număr prestabilit de ori; numărul de repetări este egal cu conținutul registrului B la intrarea în ciclu, cu condiția ca B să nu fie implicat în instrucțiunile din ciclu:

```

ET1  → LD B,32          FOR B=32 TO 0 STEP -1
      instrucțiune     instrucțiune BASIC
      instrucțiune     instrucțiune BASIC
      instrucțiune     instrucțiune BASIC
      DJNZ ET1         NEXT b
  
```

Dacă se folosesc mai multe cicluri imbricate, atunci se utilizează alte registre conform aceluiași principiu.

```

ET1  → LD C,256        FOR C=0 TO 20
      instrucțiune     instrucțiune BASIC
      ...              ...
      INC C            ...
      JR ET1          NEXT C
  
```

În acest exemplu, ciclul este executat prin incrementarea registrului C; atunci când C a fost incrementat de 20 ori, rezultă $C_i = 0$, $Z = 1$ și se părăsește ciclul.

Modelul ciclurilor imbricate va fi deci:

```

ET1  → LD C,10        FOR C=10 TO STEP -1
      instrucțiune     instrucțiune BASIC
      instrucțiune     instrucțiune BASIC
ET2  → LD D,156      FOR D=0 TO 100
      instrucțiune     instrucțiune BASIC
      INC D            instrucțiune BASIC
      JR NZ,ET2       NEXT D
      instrucțiune     instrucțiune BASIC
      DEC C           instrucțiune BASIC
      JR NZ,ET1      NEXT C
  
```

Ciclul C va fi executat de 10 ori, iar pentru fiecare C ciclul D se va executa de 100 ori.

Se reamintește că instrucțiunea **NOP** determină ca microprocesorul să nu efectueze nici o operație, servind pentru introducerea de scurte răgazuri (4 cicluri de ceas, adică $1,14 \mu\text{s}$). Întrucât un program corect nu se realizează, de regulă, de la început trebuind să fie modificate unele instrucțiuni, este util să se introducă în program un număr de instrucțiuni **NOP** care vor fi ștergute prin introducerea instrucțiunilor corecte. Invers, când se șterge o instrucțiune, este rațional să se înlocuiască cu **NOP**.

Exemplificarea 4.2.

```

10          ORG 32000
20          ENT $
30          LD HL,ET1
40          LD B, (HL)
50          INC HL
60          LD E, (HL)
70          LD HL,0
80          LD D,0
90  ET 2    ADD HL,DE
100         DJNZ ET2
110        LD (ET3),HL
120        RET
130  ET1    DEFB 6
140        DEFB 58
150  ET3    DEFW 0

```

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie
30	Registrul HL se încarcă cu ET1	A = 0; B = 0; D = 0; E = 0; HL = 0 DE = 0; (HL) = 0
40	Registrul B se încarcă cu valoarea din locația (HL)	HL = 32018; (HL) = 6
50	Registrul HL se incrementează cu 1	B = 6; HL = 32018; (HL) = 6
60	Registrul E se încarcă cu valoarea din locația (HL)	B = 6; HL = 32019; (HL) = 58

70	Registrul HL se încarcă cu numărul 0	B = 6; E = 58 ; HL = 32019; DE = 58 ; (HL) = 58
80	Registrul D se încarcă cu numărul 0	B = 6; E = 58; HL = 0 ; DE = 58; (HL) = 243
90	Registrul HL se adună cu valoarea din registrul DE	B = 6; D = 0 ; E = 58; DE = 58; (HL) = 243
100	Decrementează registrul B; dacă B = 0 continuă, altfel salt relativ la adresa ET 2	B = 6; E = 58; HL = 58 ; DE = 58; (HL) = 42
110	Locația de memorie ET3 se încarcă cu valoarea din HL	B = 0 ; E = 58; HL = 348 ; DE = 58; (HL) = 76
120	Adresa de întoarcere este scoasă din stivă	E = 58; HL = 348; DE = 58; (HL) = 76

Exemplul 4.1.: albirea/înnegrirea unui rând de pixeli

```

10          ORG 60000
20          ENT $
30          LD HL,16384      ;adresa primului
                             rînd de pixeli
40          LD B,32         ;nr. caracterelor pe
                             o linie

50 ET1      LD (HL),0
60          IND HL
70          DJNZ ET1
80 ZEND     RET

```

Pentru a înnegri primul rînd de pixeli se înlocuiește în linia 50 cifra 0 cu 255, sau în BASIC: **POKE 60006, 255: CLS : RANDOMIZE USR 60000.**

Această rutină va mai fi folosită pe parcursul lucrării.

Exemplul 4.2.: deplasarea coloanelor de caractere spre stînga.

În acest program instrucțiunea **LDIR** se folosește pentru deplasarea unei linii de pixeli (192) spre stînga; în acest scop s-a construit un ciclu pentru a efectua această operație cu contorul BC (liniile 70-180). S-a folosit instrucțiunea **DEC C** în loc de **DEC BC** pentru că nu influențează flagul Z. Instrucțiunea **LD A,(DE)** stochează octetul primei coloane din stînga spre a-l repune după **LDIR** în ultima coloană din dreapta. Cele două instrucțiuni **NOP** sînt introduse pentru eventuale completări ale


```

30          DEFW 0
40          LD HL,22526
50          LD DE,22527
60          LD BC,192
70 ET2      LD (ET1),BC
80          LD BC,31
90          LD A,(DE)
100         LDDR
110         NOP
120         NOP
130         LD (DE),A
140         DEC DE
150         DEC HL
160         LD BC,(ET1)
170         DEC C
180         JR NZ,ET2
190 ZEND    RET
200 ET1     LD (HL),0

```

Pentru această rutină s-a folosit același principiu ca la rutina anterioară cu deosebirea că se pornește de la colțul inferior dreapta și că se utilizează instrucțiunea **LDDR**. Programul BASIC de folosire a rutinei este același cu cel prezentat la exemplul 4.2.

4.2. STIVA

Prin *stivă* se înțelege o zonă de memorie externă RAM folosită pentru stocarea provizorie a conținutului registrelor duble la adresa specificată de indicatorul de stivă SP. Acest indicator poate păstra un număr $n = 0..65535$.

Salvarea se face la adresele descrescătoare, prima salvare implicând octetul superior al registrului dublu (dd_H) și apoi cel interior (dd_L), iar conținutul indicatorului SP se decrementează cu 2. Instrucțiunea corespunzătoare este

PUSH dd ; dd = {AF,BC, DE, IX, IY}

și realizează operațiile

$(SP-1) = dd_H$; POKE SP-1, dd_1

$(SP-2) = dd_L$; POKE SP-2, dd_2

SP = SP - 2 ; LET SP = SP - 2

Datele pătrund în stivă pe la baza ei și le deplasează pe cele existente către vârful stivei.

- Scoaterea datelor din stivă se face cu instrucțiunea

POP dd ; dd = {AF, BC, DE, HL, IX, IY}

de la locația de memorie a cărei adresă se află în SP, și le încarcă în registrul dublu dd în vreme ce indicatorul SP este incrementat cu 2

$dd_H = (SP + 1)$

$dd_L = (SP)$

SP = SP + 2

Datele se scot din vârful stivei.

Stiva funcționează pe principiul LIFO "*ultima informație intrată va fi prima la ieșire*".

- Indicatorul de stivă SP poate fi implicat în instrucțiunile care utilizează registrele BC, DE, existînd următoarele forme:

SP nn	; nn = 0...65535 ; SP = nn
LD SP, (nn)	; SP = (nn)
LD (nn), SP	; (nn) = SP
LD SP, HL	; SP = HL
ADD HL, SP	; HL = HL + SP
ADC HL, SP	; HL = HL + C _i + SP
SBC HL, SP	; HL = HL - SP - C _i
INC SP	; SP = SP + 1
DEC SP	; SP = SP - 1
EX (SP), HL	; (SP) = HL

Exemplificarea 4.3.: în această exemplificare stiva are valoarea inițială 32254 și se incrementează în jos.

```

10          ORG 32000
20          ENT $
30          LD HL, 56789
40          LD DE, 34567
50          LD BC, 12345
60          PUSH HL
70          PUSH BC
80          PUSH DE
90          EX (SP), HL
100         LD BC, 0

```


110	POP DE
120	POP BC
130	POP HL
140	RET
150	INC HL

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Stiva
30	Registrul HL se încarcă cu numărul 56789	A = 0; HL = 0; DE = 0; SP = 32254	32014
40	Registrul DE se încarcă cu numărul 34567	HL = 56789; SP = 32254	idem
50	Registrul BC se încarcă cu numărul 12345	HL = 56789; DE = 34567; SP = 32254	Idem
60	Registrul HL se depune în stivă și SP se decrementează cu 2	HL = 56789; BC = 12345; DE = 34567; SP = 32254	idem
70	Registrul BC se depune în stivă și se decrementează cu 2	HL = 56789; BC = 12345; DE = 34567; SP = 32252	56789
80	Registrul DE se depune în stivă și SP se decrementează cu 2	HL = 56789; BC = 12345; DE = 34567; SP = 32250	56789 12345
90	Stiva se schimbă cu valoarea din registrul HL.	HL = 56789; BC = 12345; DE = 34567; SP = 32248	56789 12345 34567
100	Registrul BC se încarcă cu numărul 0	HL = 34567; BC = 12345; DE = 34567; SP = 32248	56789 12345 56789

110	Registrul DE se scoate din stivă și SP se incrementează cu 2	HL = 34567; DE = 34567; BC = 0 ; SP = 32248	idem
120	Registrul BC se scoate din stivă și SP se incrementează cu 2	HL = 34567; BC = 0; DE = 56789 ; SP = 32250	idem
130	Registrul HL se scoate din stivă și SP se incrementează cu 2	HL = 34567; BC = 12345 ; DE = 56789; SP = 32252	idem
140	Adresa de întoarcere este scoasă din stivă	HL = 56789 ; BC = 12345; DE = 56789; SP = 32254	idem

În continuare sînt prezentate o serie de rutine utile oricărui programator.

Exemplul 4.4.

```

10          ORG 60000          ;CHR$ SCROLL RAPID
                                STÎNGA
20          ENT 60000
30          LD B,192          ;numarul rîndurilor
                                ecranului
40          LD DE,16384
50          PUSH DE
60          POP HL
70          INC HL
80  ET1     PUSH BC
90          LD BC,31
100         LD A,(DE)
110         LDIR
120         DEC HL
130         LD (HL),0
140         INC HL
150         INC HL
160         INC DE
170         POP BC
180         DJNZ ET1
190  ZEND   RET

```

Programul BASIC corespunzător este următorul:

```
10 CLS : FOR j=0 TO 21: PRINT AT j,0;"32 caractere
   oarecare": NEXT j
20 PAUSE 100: FOR i=0 TO 31: RANDOMIZE USR 60000:
   NEXT i
```

Dacă se dorește o deplasare parțială și nu totală a ecranului, atunci în linia 20 variabila *i* se va dimensiona corespunzător: de exemplu pentru o deplasare cu 10 caractere, linia 20 se scrie

```
20 PAUSE 100: FOR i=0 TO 9: RANDOMIZE USR 60000:
   NEXT i
```

Cu unele modificări se pot obține deplasări pentru fiecare din zonele ecranului, după cum urmează:

- pentru treimea superioară: 30 LD B,64 ; numărul rîndurilor
 40 LD DE,16384 ; adresa primului
 octet
- pentru treimea mijlocie: 30 LD B,64
 40 LD DE,18432 ; adresa primului
 octet
- pentru treimea inferioară: 30 LD B,64
 40 LD DE,20480 ; adresa
 primului octet
- pentru primele 2/3 de ecran: 30 LD B,128 ; numărul rîndurilor
 40 LD DE,16384
- pentru ultimile 2/3: 30 LD B,128
 40 LD DE,18432

Pentru a se realiza scroll-uri spre dreapta se folosește rutina următoare:

Exemplul 4.5.:

```
10                   ORG 60000                   ;CHR$ SCROLL RAPID
                                          DREAPTA
20                   ENT $
30                   LD B,192                   ;numarul rîndurilor
                                          ecranului
40                   LD DE,22527               ;adresa ultimului
                                          octet din linia 23
50                   PUSH DE
60                   POP HL
70                   DEC HL
```

```

80  ETO          PUSH BC
90             LD BC,31          ;numarul liniilor
100            LD A, (DE)
110            LDDR
120            INC HL
130            LD (HL),0
140            DEC HL
150            DEC HL
160            DEC DE
170            POP BC
180            DJNZ ETO
190  ZEND       RET

```

Modificările pentru deplasările unor zone ale ecranului se fac astfel:

- pentru treimea superioară: 30 LD B,64
40 LD DE,18431
- pentru treimea mijlocie: 30 LD B,64
40 LD DE,20479
- pentru treimea inferioară : 30 LD B,64
40 LD DE,22527
- pentru primele 2/3 de ecran: 30LD B,128
40 LD DE,20479
- pentru ultimile 2/3: 30 LD B,128
40 LD DE,22527

Exemplul 4.6:

```

10             ORG 60000        ;FLASH
20             ENT $
30             LD HL,22528
40             LD B,3
50  L1         PUSH BC
60             LD B,0
70  L2         SET 7, (HL)
80             INC HL
90             DJNZ E2
100            POP BC
110            DJNZ L1
120  ZEND     RET

```

Pentru inhibarea comenzii **FLASH** în programul de mai sus se va înlocui linia 70 astfel:

```

70  L2         RES 7, (HL)

```

Exemplul 4.7:

```

10          ORG 60000          ;BRIGHT
20          ENT $
30          LD HL,22528
40          LD B,3
50 X1       PUSH BC
60          LD B,0
70 X2       SET 6, (HL)
80          INC HL
90          DJNZ X2
100         POP BC
110        DJNZ X1
120 ZEND    RET

```

Pentru inhibarea comenzii BRIGHT în rutina anterioară în linia 70 se va scrie:

```
70 X2       RES 6, (HL)
```

Rutinele de la exemplele 4.6 și 4.7 valorifică organizarea octetului de atribute:

bitul	7	6	5 4 3	2 1 0
	FLASH	BRIGHT	PAPER	INK

4.3. OPERAȚII LOGICE

Microprocesorul Z80 dispune de trei instrucțiuni logice

AND, OR și XOR

care sînt efectuate bit cu bit între acumulatorul A și un număr:

- dat $n = 0 \dots 255$;
- aflat într-un registru $r = \{B, C, D, E, H, L, A\}$;
- aflat într-o locatie de memorie adresată indirect de registrul dublu *HL* sau de registrele de index: $(mem) = \{(HL), (IX+d), (IY+d)\}$.

Rezultatul rămîne în acumulatorul A.

- La instrucțiunea **AND** (adică $\$I$), dacă acelasi bit din registrul A și respectiv numărul dat conține valoarea 1, atunci bitul rezultatului va fi 1 (în caz contrar va fi 0).

Exemplu: 0 1 1 0 1 1 0 0 AND

0 1 0 1 0 1 1 0

egal: 0 1 0 0 0 1 0 0

Deci formele posibile ale instrucțiunii AND sînt:

- **AND n** ; $n = 0..255$; $A = A \wedge n$
- **AND r** ; $r = \{B, C, D, E, H, L, A\}$; $A = A \wedge r$
- **AND (mem)** ; $(mem) = \{(HL), (IX+d), (IY+d)\}$
; $A = A \wedge (mem)$

Observatii:

1) Operația logică **AND** influențează flagurile S,Z, iar P/V și C_i sînt puși pe 0.

2) Dacă trebuie realizată operația logică $B = B \wedge C$ pentru care nu există instrucțiunea, ea se simulează astfel:

```
LD A,B
AND C
LD B,A
```

3) Instrucțiunea **AND A** lasă registrul A neschimbat și face flagul $C_i = 0$ (o modalitate de zeroificare a flagului C_i).

4) Instrucțiunea **AND 0** realizează $A = 0$ ca și **LD A,0** (care nu influențează flagurile).

5) Instrucțiunea **AND 255** lasă registrul A neschimbat.

6) Instrucțiunea **AND** poate masca anumiti biți dintr-un octet sau reseta (pune pe zero) grupuri de biți din acumulatorul A. Acest element este important pentru că se pot face calcule necesare fișierului de atribute sau pentru a se realiza modulo 8 util în manipularea atributelor.

De pildă:

AND 7 face ca A să aibă o valoare cuprinsă între 0 și 7, adică A este modulo 8

AND 6 dă lui A valorile 0,2,4 sau 6

AND 192 lasă biții 7 și 6 neschimbați, ceilalți fiind puși pe 0

AND 15 lasă cei 4 biți mai puțin semnificativi neschimbați și pune pe 0 cei 4 biți semnificativi.

- La instrucțiunea **OR** (adică **SAU**), dacă acelasi bit din registrul A sau respectiv numărul dat conține valoarea 1, atunci bitul rezultatului va fi 1 (afel va fi 0).

```
Exemplu: 0 1 1 0 1 1 0 0 OR
          0 1 0 1 0 1 1 0
```

egal: 0 1 1 1 1 1 0

Formele posibile ale instrucțiunii logice OR sînt următoarele:

- **OR n** ; $n=0...255$; $A=A \vee n$
- **OR r** ; $r=\{B,C,D,E,H,L,A\}$; $A=A \vee r$
- **OR (mem)** ; $(mem)=\{(HL),(IX+d),(IY+d)\}$; $A=A \vee (mem)$

Observatii:

1) Instrucțiunea logică **OR** influențează flagurile S,Z și pune pe 0 flagurile P/V și C_i .

2) Dacă trebuie realizată operația logică $B = B \vee C$ pentru care nu există instrucțiune, ea se simulează astfel:

LD A,B

OR C

LD B,A

3) Dacă $A=0$, atunci **OR n** este echivalent cu **LD A,n**, iar dacă $A=255$, atunci **OR n** lasă acumulatorul A neschimbat.

4) **OR A** face flagul $C_i=0$.

5) Se folosește adesea instrucțiunea **OR** pentru a seta anumiți biți (a le da valoarea 1). Astfel **OR 192** pune pe 1 biții b7 și b6 ai acumulatorului A restul rămînînd neschimbați. Dacă A conține atributele unui caracter, atunci instrucțiunea **OR 192** introduce **FLASH** (bitul b7) și strălucire (bitul b6), fără a se schimba culoarea **PAPER** (biții b5,b4,b3) sau a **INK**-ului (biții b2,b1,b0).

- La instrucțiunea **XOR** (adică **SAU EXCLUSIV**), dacă același bit din registrul A și respectiv numărul dat au aceeași valoare (0 sau 1), atunci bitul corespunzător rezultatului va fi 0 (în caz contrar va fi 1)

Exemplu: 0 1 1 0 1 1 0 0 **XOR**

0 1 0 1 0 1 1 0

egal: 0 0 1 1 1 0 1 0

Formele instrucțiunii logice XOR sînt următoarele:

- **XOR n** ; $n=0...255$; $A=A \vee n$
- **XOR r** ; $r=\{B,C,D,E,H,LA\}$; $A=A \vee r$
- **XOR (mem)** ; $(mem)=\{(HL),(IX+d),(IY+d)\}$; $A=A \vee (mem)$

Observatii:

1) Instrucțiunea logică **XOR**, influențează flagurile S,Z și pune pe 0 flagurile P/V și C_i.

2) Instrucțiunea **XOR** este utilă pentru inversarea individuală sau în grup a bitilor din registrul A. Astfel, **XOR A** face A=0 și C_i=0, iar XOR A scris de două ori la rând returnează A original.

Exemplul 4.8: se reia exemplul 3.10 într-o altă formă

```

10          ORG 65000          ;MEMORIA LIBERA
20          ENT $
30          LD HL,0
40          LD DE,(23653)      ;(STKEND)
50          AND A              ;A=A si Ci=0
60          SBC HL,DE         ; HL=HL-DE-Ci
70          PUSH HL
80          POP BC
90  ZEND      RET

```

Tastând **PRINT USR 65000** se obține rezultatul (numărul de octeți al memoriei libere).

4.4. COMPLEMENTAREA ȘI OPRIREA EXECUȚIEI PROGRAMULUI

- Instrucțiunea **CPL** execută $A = \bar{A}$ respectiv **LET A=1-A**, adică realizează complementul restrîns al acumulatorului A.
- Instrucțiunea **NEG** execută complementul adevărat al acumulatorului (adică **LET A=0-A**) și influențează flagurile astfel:
 - S și Z conform rezultatului
 - C_i=1 dacă A=0 înaintea operației **NEG**
 - P/V=1 dacă A=128 înaintea operației **NEG**
- Instrucțiunea **HALT** execută **NOP**-uri pînă la întreruperea sau resetarea microprocesorului. După ce întreruperea a fost tratată, se execută următoarea instrucțiune după **HALT**. Se menționează că circuitele calculatoarelor compatibile cu **ZX-SPECTRUM** forțează o întrerupere de tip $\overline{\text{INT}}$ la fiecare afișaj, toate la 1/50 secunde. Prin urmare, **HALT** va servi la sincronizarea programului cu operații din

afara microprocesorului.

Rezumînd, instrucțiunile pentru complementare și oprirea execuției programului sînt următoarele:

- **CPL realizează complementul restrîns $A = \bar{A}$; LET A = 1-A**

Simularea pentru un registru dublu dd:

LD A,d1

CPL

LD d1,A

LD A,d2

CPL

LD d2,A

(d1-primul registru simplu: d2-al doilea registru simplu)

- **NEG realizează complementul adevărat: $A = 0 - A$.
Dacă $A = 1$, atunci NEG face $A = -1$, iar dacă $A = -6$ instrucțiunea NEG face $A = 6$. Simularea**

CPL

INC A

realizează NEG fără a influența flagul C_i (celelalte flaguri sînt influențate de către INC A).

- **HALT oprește execuția programului pînă la apariția unei întreruperi; după tratarea întreruperii se execută instrucțiunea de după HALT.**

Exemplul 4.9.: realizarea instrucțiunii OVER

```

10          ORG 60000          ;OVER
20          ENT $
30  ET1     LD BC,0
40          LD HL,16384
50          LD DE,18432
60          LD BC,2048          ;nr.de octeti al
                                unei treimi de
                                ecran

70  ET2     LD (ET1),BC
80          LD B,(HL)
90          LD A,(DE)
100         XOR B
110        LD (HL),A
120        LD BC,(ET1)
130        INC HL

```

140	INC DE
150	DEC BC
160	LD A,B
170	OR C
180	OR NZ,ET2
190	ZEND RET

Rutina are un ciclu între liniile 70 și 180 care se efectuează de 2048 ori, așa cum indică contorul BC (linia 60). Instrucțiunea **DEC BC** nu influențează flagurile, iar **LD A,B** (linia 160) și **OR C** realizează B OR C când toți biții lui B și C sînt puși pe 0, flagul Z devenind 1. Instrucțiunea XOR din linia 100 este folosită pentru a realiza instrucțiunea **OVER** din BASIC.

În forma de mai sus, rutina supraimpresionează primele 8 linii peste liniile 9 la 16 ale ecranului dacă se tastează **RANDOMIZE USR 60000**. Tastînd **RANDOMIZE UȘR 60000** a doua oară se regăsește afișarea inițială.

Schimbînd valorile registrelor duble HL,DE,BC (liniile 40..60) se pot realiza alte supraimpresionări. Astfel:

- pentru primele 2 zone ale ecranului: 50 LD DE,20480
60 LD BC,2*2048
- pentru întreg ecranul: 50 LD DE,22496
60 LD BC,3*2048

4.5. TESTĂRI ȘI COMPARAȚII

4.5.1. Testarea fiecărui bit luat izolat

Instrucțiunea **BIT** oferă modalitatea de a testa un bit oarecare (b) dintr-un:

- registru oarecare r:

BIT b,r ; b=0...7 (numărul bitului)

$r \in \{B,C,D,E,H,L,A\}$; $Z = \overline{b_r}$

(valoarea complementată a bitului b este copiată în flagul Z)

octet de memorie (mem): **BIT b, (mem)** ;
 $(mem) = \{(HL), (IX + d), (IY + d)\}$
 $Z = \bar{b}(\text{mem})$ (valoarea complementată a
bitului b este copiată în flagul Z)

Este evident că salturile condiționate privind Z sau NZ se folosesc după instrucțiunile BIT, pentru a se decide instrucțiunea care urmează.

Pe baza fig.3.6 se reamintesc următoarele:

- codurile caracterelor afișabile sînt cuprinse între 32 (blanc) și 127 (©);
- literele majuscule merg de la 65 (A) la 90 (Z), iar literele minuscule de la 97 (a) la 122(z); minusculele au bitul b5 setat (pus pe 1) și sînt mai mari cu 32 ca cele majuscule.

4.5.2. Compararea constantelor, registrelor sau octeților de memorie

Compararea implică obligatoriu registrul acumulator A și se face cu instrucțiunile:

CP n (compară A cu numArul $n=0...255$); execută A-n cînd

$$\begin{cases} A > n & : C_i = 0 & ; Z = 0 \\ A = n & : C_i = 0 & ; Z = 1 \\ A < n & : C_i = 1 & ; Z = 0 \end{cases}$$

CP r (compară A cu registrul $r = \{B, C, D, E, H, L, A\}$; execută A-r)

CP (HL) (compară A cu conținutul locației de memorie HL) ; se execută A-(HL).

Scăderile efectuate nu modifică octeții, registrele sau locațiile de memorie ci numai flagurile C_i și Z așa cum s-a arătat pentru CP n.

Aceste instrucțiuni sînt foarte utilizate și urmate de un salt condiționat.

Correspondența cu limbajul BASIC este următoarea

IF comparație THEN salt sau decizie

De exemplu: CP 3 este echivalent cu

IF A > = 3 THEN LET $C_i = 0$

```

IF A < 3 THEN LET Ci = 1
IF A = 3 THEN LET Z = 1
IF A > 3 THEN LET Z = 0

```

Observatii:

1) Exemplul următor permite să se afle dacă un registru este pus pe zero:

```

LD A,0
CP C           IF C = 0
JR Z, eticheta THEN GO TO etichetă

```

2) Se poate transpune o comandă de felul următor

```

IF-THEN-ELSE (dacă-atunci-altfel)
IF C = 0 THEN LET C = 31 ELSE LET C = C + 1

```

cu corespondentul său în limbaj de asamblare:

```

LD A,0
CP C
JR Z,ET1
INC C
JR ET2
ET1 LD C,31
ET2 secvență de instrucțiuni

```

Exemplificarea 13.4:

```

10          ORG 32000
20          ENT $
30          LD A,5
40          CP 4
50          CP 5
60          CP 6
70          LD B,3
80          CP B
90          LD HL,ET1
100         CP (HL)
110        DEC (HL)
120        CP (HL)
130        ADD A,230
140        CP (HL)
150        RET
160 ET1     DEFB 6

```

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Flagurile
30	Registrul A se încarcă cu numărul 5	A=0; B=0; HL=0; (HL)=0	S=0; Z=0; C _i =0
40	Registrul A este comparat cu numărul 4	<u>A=5; (HL)=243</u>	idem
50	Registrul A este comparat cu numărul 5	A=5; (HL)=243	Idem
60	Registrul A este comparat cu numărul 6	A=5; (HL)=243	<u>Z=1</u>
70	Registrul B se încarcă cu numărul 3	A=5; (HL)=243	<u>C_i=1;</u> <u>Z=0;</u> <u>S=1</u>
80	Registrul A se compară cu registrul B	A=5; <u>B=3</u> ; (HL)=243	idem
90	Registrul HL se încarcă cu ET1	A=5; B=3; (HL)=243	<u>C_i=0;</u> <u>S=0</u>
100	Registrul A se compară cu locația de memorie (HL)	A=5; B=3; <u>HL=32021; (HL)=6</u>	idem
110	Locația (HL) se decrementează cu 1	A=5; B=3; HL=32021; (HL)=6	<u>C_i=1;</u> <u>S=1</u>
120	Registrul A se compară cu locația de memorie (HL)	A=5; B=3; HL=32021; <u>(HL)=5</u>	C _i =1; <u>S=0</u>
130	Registrul A se adună cu numărul 230	A=5; B=3; HL=32021; (HL)=5	<u>C_i=0;</u> <u>Z=1</u>

140	Registrul A este comparat cu locația de memorie (HL)	A=235; B=3; HL=32021; (HL)=5	Z=0
150	Adresa de întoarcere este scoasă din stivă	A=235; B=3; HL=32021; (HL)=5	S=1

Exemplul 4.10:

```

10          ORG adr          ;adr=adresa de start
                                pentru functia
                                INVERSE

20          ENT $

30          LD HL,16384

40  ET1     LD A, (HL)

50          CPL

60          LD (HL),A

70          INC HL

80          LD A,H

90          CP 88          ;88*256=22528

100         JR NZ, ET1

```

Rutina folosește instrucțiunea **CPL** (care realizează complementul restrîns al registrului A, adică $A=1-A$) pentru un fișier, ceea ce corespunde funcției **INVERSE** din *BASIC*. Tastînd **RANDOMIZE USR 60000** (adr=60000) este realizată această funcție, iar dacă se tastează încă o dată **RANDOMIZE USR 60000** se revine la normal.

Folosind însă un ciclu *BASIC* se obține un efect vizual interesant care poate fi utilizat în jocuri (exemplu: explozia unui vapor atins de o torpilă) sau ca o cortină de trecere între două ecrane:

```
10 FOR x=1 TO 10: RANDOMIZE USR 60000: NEXT x
```

Exemplul 4.11:

```

10          ORG adr          ;SCHIMBAREA
                                FISIERELOR ECRAN

20          ENT $

30          LD HL,16384      ;ADRESA ZONEI DE
                                MUTAT

40          LD DE,18432     ;ADRESA ZONEI UNDE
                                SE MUTA

50  ET1     LD A, (DE)

60          LD B, (HL)

70          LD (HL),A

80          LD A,B

```

```

90          LD (DE),A
100         INC DE
110         INC HL
120         LD A,H
130         CP 72          ;72*256=18432
140         JR NZ,ET1
150 ZEND    RET

```

Considerînd $adr=60000$ (adresa de start a rutinei) și tastînd **RANDOMIZE USR 60000**, rutina va muta liniile 1-8 în locul liniilor 9-16 (adică mută treimea superioară a ecranului în locul treimii mijlocii); dacă se tastează încă o dată **RANDOMIZE USR 60000** se revine la poziția inițială.

Pentru a se muta treimea superioară a ecranului în locul treimii inferioare în linia 40 se va scrie adresa 20480 în loc de 18432; evident că și acest caz dacă se tastează a doua oară **RANDOMIZE USR 60000** se revine la poziția inițială.

Se va reține această rutină pentru efecte speciale în diverse programe.

Pentru a muta un bloc oarecare de memorie în cod mașină, avînd lungimea lung, de la o adresă adr1 la o altă adresă adr2, rutina corespunzătoare va fi

```

10          ORG adresa start
20          ENT $
30          LD HL,adr1
40          LD DE,adr2
50          LD BC,lung
60          LDIR

```

De exemplu, un bloc de memorie la adresa 25000 și lungime 39240 trebuie mutat la adresa 23296; programul corespunzător va fi:

```

10          ORG 64240
20          ENT $
30          LD HL,25000
40          LD DE,23296
50          LD BC,39240
60          LD IR

```

4.5.3. Comparări cu repetiții, incrementeări și decrementeări

Există patru instrucțiuni care efectuează asemenea operații. În mnemonica lor se folosesc literele:

CP-compară; **I**-incrementează; **D**-decrementează; **R**-repetă.

Aceste instrucțiuni sînt următoarele:

- | | | | | |
|---|-------------|------------|------------|-----------|
| • | CPI | ; A ↔ (HL) | ; HL=HL+1 | ; BC=BC-1 |
| • | CPIR | ; A ↔ (HL) | ; (HL=HL+1 | ; BC=BC-1 |
| • | CPD | ; A ↔ (HL) | ; HL=HL-1 | ; BC=BC-1 |
| • | CPDR | ; A ↔ (HL) | ; HL=HL+1 | ; BC=BC-1 |

Conținutul registrului acumulator A este comparat cu cel al locației de memorie (HL). Indicatorul de adresă HL este incrementat cu 1 (la **CPI**, **CPIR**), respectiv este decrementat cu 1 (la **CPD**, **CPDR**). iar numărătorul de octeți este decrementat cu 1.

La instrucțiunile CPI, CPD flagurile devin:

$$P/V = \begin{cases} 0 & \text{dacă după efectuare } BC = 0 \\ 1 & \text{dacă după efectuare } BC \neq 0 \end{cases}$$

$$Z = \begin{cases} 1 & \text{dacă } A = 0 \\ 0 & \text{dacă } A \neq 0 \end{cases}$$

La instrucțiunile CPIR și CPDR dacă rezultatul comparării arată A=(HL) instrucțiunea se termină, iar în caz contrar ea este reluată pînă cînd BC=0. Flagurile devin

$$P/V = \begin{cases} 0 & \text{dacă după efectuare } BC = 0 \\ 1 & \text{dacă după efectuare } BC \neq 0 \end{cases}$$

$$Z = \begin{cases} 1 & \text{dacă } A = (HL) \\ 0 & \text{dacă } A \neq (HL) \end{cases}$$

Aceste instrucțiuni sînt utile pentru ca căuta un caracter într-un fișier, sau pentru a căuta un bloc de memorie.

Exemplul 4.12:

10		ORG adr	; RENUMEROTAREA
			LINIILOR BASIC
20		ENT \$	
30	ET1	DEFB 0	
40		LD HL, (23635)	; (PROG)
50		LD DE, 0	


```

60 ET2      LD B,10      ;PASUL DINTRE 2
                                LINII
70 ET3      INC DE
80          DJNZ ET3
90          LD (HL),D
100         INC HL
110        LD (HL),E
120        INC HL
130        INC HL
140        INC HL
150        LD A,13      ;13-CODUL LUI ENTER
160        LD C,0      ;PENTRU A EVITA BC=0
170        CPIR
180        LD BC,(23627) ;(VARS)
190        LD (ET1),HL ;PASTREAZA HL
200        AND A      ;Ci=0
210        SBC HL,BC
220        LD HL,(ET1) ;REFACE HL
230        JR C,ET2
240        NOP
250        LD HL,(23635) ;(PROG)
260 ET4      LD A,236   ;236-CODUL LUI GO TO
270        CP (HL)
280 ET5      JR Z,ET6
290        INC A      ;CODUL LUI GOSUB
300 ET6      CPI
310 ET7      JR Z,ET9
320        LD (ET1),HL ;PASTREAZA HL
330        LD BC,(23627) ;(VARS)
340        AND A      ;Ci=0
350        SBC HL,BC
360        LD HL,(ET1)
370        JR C,ET4
380        RET
390 ET8      LD A,14    ;CARACTER DE CONTROL
                                NUMAR
400        CP (HL)
410        JR Z,ET4
420        LD A,143    ;143-CODUL
                                CARACTERULUI GRAFIC
                                PATRAT NEGRU

```

```

430          LD (HL) , A
440          INC HL
450 ET9      JR ET8

```

Rutina realizează numerotarea liniilor BASIC cu pasul 10 și afișează după instrucțiunile **GO TO** și **GOSUB** un pătrat negru, în locul căruia programatorul trebuie să înscrie numărul liniei unde se va face saltul.

Numărul nou al liniei BASIC este în registrul dublu DE (linia 50) și pasul (incrementarea) în registrul B (linia 60). Înainte de **CPIR** se face flagul $C_i=0$. Deci $BC=0$ și la prima decrementare se face $BC=65535$. Caracterul de control cu codul 13 (**ENTER**), încărcat în registrul A (linia 150), termină fiecare linie BASIC. Rutina se termină la linia 240 unde s-ar putea introduce instrucțiunea **RET**. Dacă se lasă **NOP**, programul continuă cu o altă rutină care are rolul de a atrage atenția asupra instrucțiunilor **GO TO** și **GOSUB**; astfel **CP (HL)**-din linia 270 - și **CPI**- din linia 300- compară respectiv cu **GO TO** și **GOSUB**. În ipoteza că se întâlnesc asemenea instrucțiuni în programul BASIC ce se renumerotează, atunci se sare la linia 390 unde se compară cu numărul 14 (codul de control NUMĂR) ce semnifică faptul că următorii 5 octeți reprezintă un număr în virgulă mobilă și deci caracterele respective trebuie înlocuite, pe locul unde apare pătratul negru, cu numărul liniei unde se face saltul.

Exemplul 4.13:

```

10          ORG adr          ; SCROLL IN JOS
20          ENT $
30          LD HL,22527      ; ADRESA ULTIMULUI
                               OCTET ECRAN
40          LD DE,22495      ; ADRESA OCTETULUI DE
                               DEASUPRA

50 ET1      PUSH HL
60          PUSH DE
70          LD C,23          ; NUMARUL ULTIMEI
                               LINII
80 ET2      LD B,32          ; NUMARUL ULTIMEI
                               COLOANE

90 ET3      LA A, (DE)
100         LD (HL) , A
110         LD A, C
120         AND 7

```

```
130          CP 1
140          JR NZ,ET4
150          SUB A
160          LD (DE),A
170 ET4      DEC HL
180          DEC DE
190          DJNZ ET3
200          DEC C
210          JR Z,ET5
220          LD A,C
230          AND 7
240          CP 0
250          JR Z,ET6
260          CP 7
270          JR NZ,ET2
280          PUSH DE
290          LD DE,1792
300          AND A
310          SBC HL,DE
320          POP DE
330          JR ET2
340 ET5      POP DE
350          POP HL
360          DEC D
370          DEC H
380          LD A,H
390          CP 71
400          RET Z
410          JR ET1
420 ET6      PUSH HL
430          LD HL,1792
440          EX DE,HL
450          AND A
460          SBC HL,DE
470          EX DE,HL
480          POP HL
490 ZEND     JR ET2
```

Programul BASIC ce folosește această rutină este următorul:

```
10 FOR i=0 TO 21: RANDOMIZE USR adr: NEXT i
```

Se obține un efect vizual atractiv în care liniile "dispar" rînd pe rînd, într-o mișcare în jos.

4.6. ROTIRI ȘI DEPLASĂRI

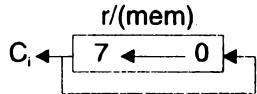
4.6.1. Rotiri de registre și locații de memorie

Din această categorie fac parte instrucțiunile de deplasare la care bitul care iese afară la un capăt al registrului sau locației de memorie este introdus la celălalt capăt. Flagul C_i face parte din numărul deplasat fiind al noulea bit sau primind starea bitului care iese din registru.

Există 4 instrucțiuni de rotire:

a) Rotire circulară la stînga

• **RLC r** ; $r = \{B, C, DD, E, H, K, A\}$

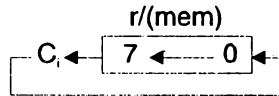


• **RLC (mem)** ; $(mem) = \{(HL), (IX+d), (IY+d)\}$

Conform schemei alăturate, se deplasează la stînga conținutul registrului r sau al locației de memorie (mem) valoarea bitului b7 fiind încărcată în bitul b0 și în flagul C_i .

b) Rotire la stînga

• **RL r** ; $r = \{B, C, D, E, H, L, A\}$



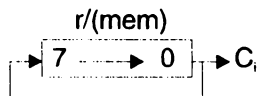
• **RL (mem)** ; $(mem) = \{(HL), (IX+d), (IY+d)\}$

Se deplasează la stînga conținutul registrului r sau al locației de memorie (mem), flagul C_i fiind al noulea bit (adică bitul b7 trece în C_i , iar C_i trece în b0).

Este util de menționat că instrucțiunilor **RL** pot fi înșiruite pentru a realiza înmulțirea cu 2 a unui număr de orice lungime.

c) Rotire circulară la dreapta

• **RRC r** ; $r = \{B, C, DE, H, L, A\}$

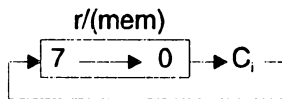


- **RRC (mem)** ; (mem) = {(HL),(IX+d),(IY+d)}

Instrucțiunile realizează o deplasare la dreapta a conținutului registrului r sau al locației de memorie (*mem*), valoarea bitului b0, fiind încărcată în bitul și flagul C_i .

d) Rotire la dreapta

- **RR r** ; r = {B,C,D,E,H,L,A}



- **RR (mem)** ; (mem) = {(HL),(IX+d),(IY+d)}

Se deplasează la dreapta conținutul registrului r sau al locației de memorie (*mem*), flagul C_i fiind al noulea bit (adică bitul b0 trece în C_i iar C_i trece în b7).

Instrucțiunile **RR** pot fi înșiruite pentru a realiza împărțirea cu 2 a unui număr de orice lungime.

Observatii:

1) Toate instrucțiunile de rotire (**RLD**, **RL**, **RRC**, **RR**) afectează flagurile S,Z,P/V și C_i .

2) Instrucțiunile **RLC** și **RRC** sînt utile pentru testarea secvențială a conținutului unui registru fără a-l altera.

3) Mai există 4 instrucțiuni de rotire care implică numai registrul acumulator A, motiv pentru care în mnemonica lor apare litera A

RLCA, RLA, RRCA, RRA

Ele afectează numai flagul C_i și sînt de două ori mai rapide.

Exemplificarea 4.5:

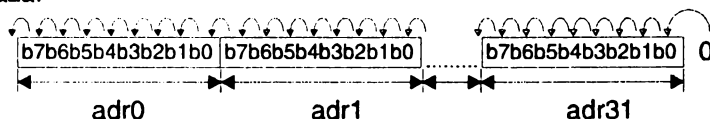
10		ORG 32000
20		ENT \$
30		LD HL,ET5
40		LD B,8
50	ET1	RLC (HL)
60		DJNZ ET1
70		LD B,8
80	ET2	RL (HL)
90		DJNZ ET2
100		LD B,8
110	ET3	RRC (HL)
120		DJNZ ET3
130		LD B,8

140 ET4 RR (HL)
 150 DJNZ ET4
 160 RET
 170 ET5 DEFB 105

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Flaguri
30	Registrul HL se încarcă cu ET5	$B=0; HL=0; (HL)=0$	$C_i=0;$ $P/V=0;$ $S=0$
40	Registrul B se încarcă cu numărul 8	<u>HL = 32028; (HL) = 105</u>	idem
50	Locația (HL) se rotește la stînga deplasînd b7 în C_i	$HL = 32028; (HL) = 105$ <u>B = 8</u>	idem
60	Decrementează B; dacă $B=0$ continuă, altfel salt la adresa ET1	$B=8; HL = 32028;$ $(HL) = 210$	<u>$C_i=0;$</u> <u>$P/V=1;$</u> <u>$S=1$</u>
70	Registrul B se încarcă cu numărul 8	<u>B=0;</u> $HL = 32028;$ <u>(HL) = 105</u>	<u>$C_i=1;$</u> <u>$S=0;$</u> $P/V = 1$
80	Locația (HL) și C_i se rotesc la stînga	<u>B = 8;</u> $HL = 32028;$ $(HL) = 105$	idem
90	Decrementează B; dacă $B=0$ continuă, altfel salt la adresa ET2	$B=8; HL = 32028;$ $(HL) = 211$	<u>$P/V=0;$</u> <u>$S=1$</u>
100	Registrul B se încarcă cu numărul 8	<u>B=0;</u> $HL = 32028;$ $(HL) = 180$	<u>$C_i=1;$</u> $S = 1$ <u>$P/V=1$</u>
110	Locația (HL) se rotește la dreapta deplasînd b0 în C_i	<u>B = 8;</u> $HL = 32028;$ $(HL) = 180$	idem
120	Decrementează B; dacă $B=0$ continuă, altfel salt la adresa ET3	$B=8; HL = 32028;$ $(HL) = 90$	<u>$C_i=0;$</u> <u>$P/V=1$</u> <u>$S=0$</u>

130	Registrul B se încarcă cu numărul 8	B=0 ; HL=32028; (HL)=180	C_i=1 P/V=1 S=1
140	Locația (HL) și C _i se rotesc la dreapta	B=8 ; HL=32028; (HL)=180	idem
150	Decrementează B; dacă B=0 continuă, altfel salt la adresa ET4	B=8; HL=32028; (HL)=218	C_i=0 ; S=1 P/V=0
160	Adresa de întoarcere este scoasă din stivă	B=0 ; HL=32028; (HL)=105	C _i =0; S=0 ; P/V=1

• Așa cum s-a putut constata în unele din exemplele din acest capitol, prin scroll al ecranului se înțelege deplasarea (defilarea) imaginii ecranului însoțită de pierderea ei, întrucât ceea ce iese în afara marginii ecranului se pierde. De pildă, pentru a realiza un scroll la stînga este necesar să se deplaseze fiecare din cele 192 rînduri ale ecranului, folosind instrucțiunile de rotire la nivel de octet, conform schemei care urmează:



Bitul b0 al adresei adr31 devine 0, iar bitul b7 al adresei adr0 se pierde. Programul trebuie să țină seama de adresele de început sau de sfîrșit al unei linii. De pildă, pentru un scroll spre stînga se pornește de la adresa 22527 (adr31 pentru ultima linie) și se decrementează de fiecare dată adresa curentă; în acest fel pot fi deplasate toate rîndurile deoarece de la ultima adresă (adr0) a rîndurilor 191 se sare la adresa adr31 a rîndului 183, apoi la rîndul 175 ș.a.m.d. pînă la rîndul 0.

Exemplul 4.14:

```

10          ORG 60000          ; SCROLL LENT STINGA
                                INTREGUL ECRAN
20          ENT $
30          LD HL,22527
40          LD C,192          ;nr. rîndurilor pe
                                ecran
50 ET1     LD B,32          ;nr.de adrese dintr-
                                un rînd

```


	70 ET2	RR (HL)
	80	INC HL
- deplasarea treimii inferioare:	30	LD HL,20480
	40	LD C,64
	70 ET2	RR (HL)
	80	INC HL
- deplasarea primelor 2 zone:	30	LD HL,16384
	40	LD C,128
	70 ET 2	RR (HL)
	80	INC HL
- deplasarea ultimelor 2 zone:	30	LD HL,18432
	40	LD C,128
	70 ET2	RR (HL)
	80	INC HL

- Prin *roll* se înțelege deplasarea ecranului într-o direcție dorită și readucerea a tot ce dispare în afara ecranului în partea opusă (deplasarea fără pierdere). Programul care urmează realizează un roll lent spre stînga.

Exemplul 4.15:

```

10          ORG 60000          ;ROLL LENT SPRE
                                STINGA

20          ENT $

30          LD HL,22527

40          LD C,32

50 ET1      LD B,192

60 ET2      RL (HL)

70          DEC HL

80          DJNZ ET2

90          DEC C

100         JR NZ,ET1

110 ZEND    RET

```

Programul *BASIC* aferent acestei rutine este următorul:

```

10 CLS : FOR n=0 TO 21: PRINT AT n,0; "32 caractere
oarecare": NEXT n
20 PAUSE 50 : FOR i=0 TO 255: RANDOMIZE USR 60000:
NEXT i

```

Similar exemplului anterior, modificările pentru roll-ul diverselor zone de ecran sînt aceleași și anume:

- deplasarea treimii superioare:	30	LD HL,18432
	50	LD B,64
- deplasarea treimii mijlocii:	30	LD HL,20479
	50 ET1	LD B,64
- deplasarea treimii inferioare:	30	LD HL,22527
	50 ET1	LD B,64
-deplasarea primelor 2 zone:	30	LD HL,20479
	50 ET1	LD B,128
- deplasarea ultimilor zone:	30	LD HL,22527
	50 ET1	LD B,128

Pentru un roll lent spre dreapta modificările de ad sint:

- deplasarea întregului ecran:	30	LD HL, 16384
	60 ET2	RR (HL)
	70	INC HL
- deplasarea treimii superioare:	30	LD HL,16384
	50 ET1	LD B,64
	60 ET2	RR (HL)
	70	INC HL
-deplasarea treimii mijlocii:	30	LD HL,18432
	50 ET1	LD B,64
	60 ET2	RR (HL)
	70	INC HL
- deplasarea treimii inferioare:	30	LD HL,20480
	50 ET1	LD B,64
	60 ET2	RR (HL)
	70	INC HL
- deplasarea primelor 2 zone:	30	LD HL,16384
	50 ET1	LD B,128
	60 ET2	RR (HL)
	70	INC HL
- deplasarea ultimilor 2 zone;	30	LD HL,18432
	50 ET1	LD B,128
	60 ET2	RR (HL)
	70	INC HL

O rutină care realizează un roll rapid spre stînga este următoarea:

Exemplul 4.16:

```

10          ORG 60000          ;ROLL RAPID STINGA
20          ENT $
30          LD HL,16385
40          LD DE,16384
50          LD BC,192
60 ET      PUSH BC
70          LD BC,31
80          LD A,(DE)
90          LDIR
100         LD (DE),A
110        INC HL
120        INC DE
130        POP BC
140        DEC C
150        JR NZ,ET
160        RET

```

Programul *BASIC* de folosire:

```

10 CLS : FOR n=0 TO 21: PRINT AT n.0;"32
   caractere": NEXT n
20 PAUSE 50: FOR i=0 TO 31: RANDOMIZE USR 60000:
   NEXT i

```

Pentru un roll rapid spre dreapta modificările de adus programului anterior sînt cele care urmează:

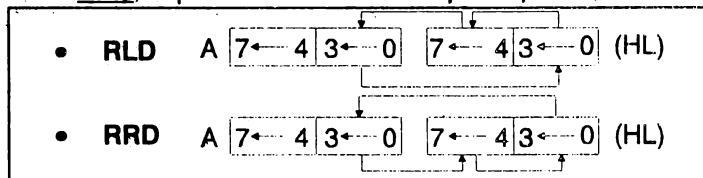
```

30          LD HL,22526
40          LD DE,22527
90          LDDR
110         DEC HL
120         DEC DE

```

4.6.2. Rotirea zecimală

Microprocesorul Z80 permite rotirea la stînga (L) și la dreapta (R) a grupurilor de 4 biți, implicînd acumulatorul A și locația de memorie (HL):



- La instrucțiunea **RLD**, conținutul locației de memorie (HL) este

rotit la stînga folosind digitul inferior al registrului A și anume: biții 3-0 ai lui A trec în biții 3-0 ai lui (HL), biții 3-0 ai lui (HL) trec în biții 7-4 ai lui (HL), iar biții 7-4 ai lui (HL) trec în biții 3-0 ai lui A.

Această instrucțiune poate fi folosită la înmulțirea cu 10 a unui număr zecimal și este utilă la izolarea celei mai semnificative cifre dintr-un număr zecimal.

- La instrucțiunea **RRD**, conținutul locației de memorie (HL) este rotit la dreapta folosind digitul inferior al registrului A după cum urmează: biții 3-0 ai lui A trec în biții 7-4 ai lui (HL), biții 7-4 ai lui (HL) trec în biții 3-0 ai lui (HL), iar biții 3-0 ai lui (HL) trec în biții 3-0 ai lui A.

Instrucțiunea **RRD** împarte la 10 un număr zecimal și permite izolarea cifrei mai puțin semnificative dintr-un număr zecimal.

Exemplificarea 4.6: înmulțirea și împărțirea unui număr BCD cu 10. Se reamintește că BCD (zecimal codificat binar) este notația prin care se pot efectua calcule cu numere zecimale fără a mai fi transformate în binar sau hexazecimal, prin operația de ajustare care consta în a aduna cifra 6 cînd în exprimarea numărului apar literele A-F. Se realizează această ajustare cu instrucțiunea **DAA**, folosită după **ADD**, **ADC**, **SUB**, **SBC**, pentru a ajusta conținutul acumulatorului A.

```

10          ORG 32000
20          ENT $
30          LD HL,ET1
40          LD A,0
50          RLD
60          INC HL
70          RLD
80          LD A,0
90          RRD
100         DEC HL
110        RRD
120        RET
130  ET1    DEFW 596

```

Nr. liniei	Ce se realizează	Conținutul registrelor	Flaguri
30	Registrul HL se încarcă cu ET1	A=0; HL=0	Z=0; N/V=0

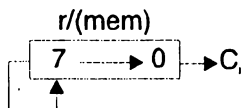
40	Registrul A se încarcă cu 0	<u>HL = 32018</u>	idem
50	Rotește la stînga conținutul zecimal al locației (HL) cu acumulatorul A	HL = 32018; <u>A = 0</u>	idem
60	Registrul HL se incrementează cu 1	<u>A = 5</u> ; HL = 32018	<u>P/V = 1</u>
70	Rotește la stînga conținutul zecimal al locației (HL) cu acumulatorul A	A = 5; <u>HL = 32019</u>	idem
80	Registrul A se încarcă cu 0	<u>A = 0</u> ; HL = 32019	P/V = 1; <u>Z = 1</u>
90	Rotește la dreapta conținutul zecimal al locației (HL) cu acumulatorul A	A = 0; HL = 32019	idem
100	Registrul HL se decrementează cu 1	<u>A = 5</u> ; HL = 32019	<u>Z = 0</u> ; P/V = 1
110	Rotește la dreapta conținutul zecimal al locației (HL) cu acumulatorul A	A = 5; <u>HL = 32018</u>	idem
120	Adresa de întoarcere este scoasă din stivă	A = 5; HL = 32018	idem

4.6.3. Deplasări

Deplasarea la dreapta (R) și la stînga (L) se realizează cu următoarele instrucțiuni:

a) Deplasarea la dreapta

- **SRA** r ; $r = \{B, C, D, E, H, L, A\}$
- **SRA** (mem); (mem) = $\{(HL), (IX+d), (IY+d)\}$

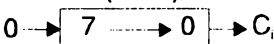


Conținutul registrului r sau al locației de memorie (mem) este deplasat la dreapta cu o poziție; bitul b_0 se transferă în flagul C_i iar bitul b_7 rămâne neschimbat.

Instrucțiunea **SRA** împarte la 2 numere pozitive și negative.

- **SRL** r ; $r = \{B, C, D, E, H, L, A\}$
- **SR**; (mem) ; (mem) = $\{(HL), (IX+d), (IY+d)\}$

$r/(mem)$



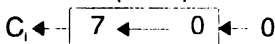
Conținutul registrului r sau al locației de memorie (mem) este deplasat la dreapta cu o poziție; bitul b_0 se transferă în flagul C_i , iar în bitul b_7 se inserează valoarea 0.

Instrucțiunea **SRL** împarte la 2 numere pozitive.

b) Deplasarea la stînga

- **SLA** r ; $r = \{B, C, D, E, H, L, A\}$
- **SLA** (mem) ; (mem) = $\{(HL), (IX+d), (IY+d)\}$

$r/(mem)$



Conținutul registrului r sau al locației de memorie (mem) este deplasat la stînga cu o poziție; bitul b_7 se transferă în flagul C_i , iar în bitul b_0 se inserează valoarea 0.

Instrucțiunea **SLA** înmulțește cu 2 un număr pozitiv.

Exemplificarea 4.7:

```

10          ORG 32000
20          ENT $
30          LD HL, ET4
40          LD B, 5
50  ET1     SRA (HL)
60          DJNZ ET1
70          LD (HL), 165
80          LD B, 5
90  ET2     SLA (HL)
100         DJNZ ET2
110        LD B, 8

```

120 ET3 SRL (HL)
 130 DJNZ ET3
 140 RET
 150 ET4 DEFB 105

Nr. liniei	Ce se realizează	Conținutul registrelor și al locațiilor de memorie	Flaguri
30	Registrul HL se încarcă cu ET4	A=0; B=0; HL=0; (HL)=0	$C_i = P/V = S = 0$
40	Registrul B se încarcă cu numărul 5	HL = 32024; (HL) = 105	idem
50	Locația (HL) se deplasează aritmetic la dreapta. Bitul b7 rămîne neschimbat	B = 5; HL = 32024; (HL) = 105	idem
60	Decrementează B; dacă B=0 continuă, altfel salt relativ la adresa ET1	B = 5; HL = 32024; (HL) = 52	$C_i = 1$
70	Locația (HL) se încarcă cu 165	B = 0; HL = 32024; (HL) = 3	$C_i = 0; P/V = 1$
80	Registrul B se încarcă cu 5	HL = 32024; (HL) = 165	idem
90	Locația (HL) se deplasează aritmetic la stînga; bitul b0 devine 0	B = 5; HL = 32024; (HL) = 165	idem
100	Decrementează B; dacă B=0 continuă altfel execută salt relativ la adresa ET2	B = 5; HL = 32024; (HL) = 74	$C_i = 1; P/V = 0$
110	Registrul B se încarcă cu 8	HL = 32014; (HL) = 160; B = 0	$P/V = 1; S = 1$

120	Locația (HL) se deplasează logic la dreapta; bitul b7 devine 0	B=8 ; HL=32024; (HL)=160	idem
130	Decrementează B; dacă B=0 continuă altfel salt relativ la adresa ET3	B=8; HL=32024 (HL)=80	P/V=1; S=0
140	Adresa de întoarcere este scoasă din stivă	B=0 ; HL=32024; (HL)=0	idem

Un efect vizual interesant de dispariție (*absorbție*) a caracterelor în **PAPER** folosind instrucțiunile de deplasare este realizat de rutina următoare:

Exemplul 4.17:

```

10                ORG 60000                ;ABSORBTIE SPRE
                                   DREAPTA
20                ENT $
30                LD HL,16384
40 ET1            LD C,32
50                AND A
60 ET 2           SRL (HL)
70                INC HL
80                DEC C
90                JR NZ,ET2
100               LD A,88
110               CP H
120               JR NZ,ET1
130 ZEND          RET

```

Programul BASIC care valorifică rutina are forma:

```

10 CLS : FOR n=0 21: PRINT AT n,0; "32 caractere":
   NEXT n
20 PAUSE 50: FOR i=0 TO 7: RANDOMIZE USR 60000:
   NEXT i

```

Același efect de "*absorbție*", dar de data aceasta spre stînga, este obținut cu rutina care urmează:

Exemplul 4.18:

```

10                ORG 60000                ;ABSORBTIE SPRE
                                   STINGA

```

```
20          ENT $
30          LD HL,22527
40 ET1      LD C,32
50          AND A
60 ET2      SLA (HL)
70          DEC HL
80          DEC C
90          JR NZ,ET2
100         LD A,63
110         CP H
120         JR NZ,ET1
130 ZEND    RET
```

5. FOLOSIREA INSTRUCȚIUNILOR PENTRU CULORI, SUNETE ȘI SCRIEREA TEXTELOR

Pentru perfecționarea programelor din acest capitol se prezintă mai întâi instrucțiunile de apelare a subrutinelor și respectiv instrucțiunile de intrare/ieșire (I/E).

5.1. APELAREA SUBRUTINELOR (SUBPROGRAMELOR)

Un subprogram (subrutină) este o parte componentă a unui program care este *apelat* de program principal. În acest scop se folosește instrucțiunea **CALL** care este similară cu instrucțiunea **JP** cu deosebirea că adresa următoarei instrucțiuni din secvență, păstrată în indicatorul de program PC, este depusă în stivă *înaintea saltului*. Instrucțiunea de revenire din subprogram este **RET**; ea scoate adresa de revenire din stivă și o încarcă în PC pentru ca programul principal să poată continua de unde a rămas. Se menționează că numărul de **PUSH**-uri și **CALL**-uri trebuie să fie egal cu numărul de **POP**-uri și **RET**-uri. Microprocesorul Z80 dispune de instrucțiuni **CALL** și **RET** necondiționate și respectiv condiționate:

CALL nn	CALL c,nn	unde c = {NC,C,NZ,Z} este condiția
RET	RETc	

Rezumînd, instrucțiunile pentru apelarea și revenirea din subrutine sînt:

•	[CALL nn RET	; nn=0..65535	[GOSUB nn RETURN
•	[CALL NC, nn RET NC	; nn=0..65535 C _i =0 pentru revenire	[GOSUB nn dacă C _i =0 RETURN dacă C _i =0
•	[CALL C, nn RET C	; nn=0..65535 ; C _i =1 pentru revenire	[GOSUB nn dacă C _i =1 RETURN dacă C _i =1
•	[CALL NZ, nn RET Z	; nn=0..65535 ; Z=0 pentru revenire	[GOSUB nn dacă Z=0 RETURN dacă Z=0
•	[CALL Z, nn RET Z	; nn=0..65535 Z=1 pentru revenire	[GOSUB nn dacă Z=1 RETURN dacă Z=1

Atunci cînd condiția $c = \{NC, C, NZ, Z\}$ nu este îndeplinită, subprogramul nu va fi apelat (CALL) sau nu se va reveni din el (RET).

Exemplificarea 5.1.: arată cum un subprogram poate fi apelat de un alt subprogram. Se reamintește că un subprogram poate să folosească un registru al cărui conținut să fie necesar în programul principal. Pentru aceasta, conținutul registrului respectiv trebuie salvat în stivă, adică memorat în altă parte unde poate fi regăsit la finele subprogramului.

```

10          ORG 32000          ;APELARI
                                NECONDITIONATE
20          ENT $
30          LD HL,22528
40          LD B,32
50          CALL SR1
60          LD HL,ET1
70 SR2     LD A,(HL)
80          ADD A,12B
90          LD (HL),A
100         RET
110 SR1     CALL SR2
120         INC HL
130         DJNZ SR1
140         RET

```

Nr. liniei	Ce se realizează	Conținutul registrelor și locațiilor de memorie	Stiva
30	Registrul HL se încarcă cu 22528	A=0; B=0; HL=0; (HL)=0	32014
40	Registrul B se încarcă cu 32	<u>HL = 22528; (HL) = 48</u>	idem
50	Pune adresa de întoarcere în stivă și apelează subrutina SR1	<u>B = 32;</u> HL = 22528; (HL) = 48	idem
60	Registrul HL se încarcă cu ET1	A = 176; HL = 22560; (HL) = 15	<u>32008</u>
70	Registrul A se încarcă cu valoarea din locația (HL)	B = 32; HL = 22528; (HL) = 48	32008 <u>32019</u>
80	Registrul A se adună cu 128	A = 48; B = 32; HL = 22528; (HL) = 48	idem
90	Locația (HL) se încarcă cu valoarea din registrul A	A = 176; B = 32; HL = 22528; (HL) = 48	idem
100	Adresa de întoarcere este scoasă din stivă	A = 176; B = 32; HL = 22528; (HL) = 176	idem
110	Pune adresa de întoarcere în stivă și apelează subrutina SR2	<u>B = 32;</u> HL = 22528; (HL) = 48	32008
120	Registrul HL se incrementează cu 1	A = 176; B = 32; HL = 22528; (HL) = 176	idem
130	Decrementează B; dacă B = 0 continuă altfel salt relativ la adresa SR1	A = 176; B = 32; HL = 22528; (HL) = 176	idem
140	Adresa de încercare este scoasă din stivă	A = 176; B = 0; HL = 22569; (HL) = 15	idem

Exemplul 5.1: se folosește rutina **CLS** din **ROM** de la adresa 3652

```

10                ORG adr                ;PARTIAL CLS
20                ENT $
30                LD B,12
40                CALL 3652
50 ZEND           RET

```

Cu **POKE (adr+1),x** unde **x**-numărul liniilor ce se șterg numărate de jos în sus ($x < 24$); se realizează un **CLS** parțial; de exemplu:

```

10 CLS : LET a$="32 caractere": FOR n=0 TO 21:
    PRINT AT n,0;a$: NEXT n
20 PAUSE 0: POKE 60001,10: RANDOMIZE USR 60000: REM
    adr=60000

```

Se prezintă în continuare două programe care realizează scroll (spre stînga, respectiv spre dreapta) dar într-o fereastră

Exemplul 5.2:

```

10                ORG adr                ;SCROLL LENT STINGA
                                                BOX
20                ENT $
30                LD HL,(ET8)
40                LD BC,(ET9)
50 ET1            AND A
60 ET2            RL (HL)
70                DEC HL
80                DEC B
90                JP NZ,ET2
100               LD HL,(ET8)
110               CALL ET 5
120               LD (ET8),HL
130               LD BC,(ET9)
140               DEC C
150               RET Z
160               LD (ET9),BC
170               JP ET1
180               LD HL,(ET8)
190               LD BC,(ET9)
200 ET3           AND A
210 ET4           RR (HL)
220               INC HL
230               DEC B
240               JP NZ,ET4

```

```
250          LD HL, (ET8)
260          CALL ET5
270          LD (ET8), HL
280          LD BC, (ET9)
290          DEC C
300          RET Z
310          LD (ET9), BC
320          JP ET3
330 ET5      LD A, H
340          AND 7
350          CP 7
360          JP Z, ET6
370          INC H
380          RET
390 ET6      LD A, L
400          AND 224
410          CP 224
420          JP Z, ET7
430          LD DE, 01760
440          AND A
450          SBC HL, DE
460          RET
470 ET1      LD A, H
480          CP 87
490          RET Z
500          LD DE, 32
510          ADD HL, DE
520          RET
530 ET8      ADD A, D
540          LD C, B
550 ET9      LD BC, 28
```

Ca toate rutinele din acest capitol, programul anterior este relocabil la orice adresă adr, iar fereastra se definește în felul următor:

POKE (adr + 108), TAB colțul dreapta sus;

POKE (adr + 109), 64(zona de sus)

72 (zona mijlocie)

80 (zona de jos);

POKE (adr + 110), înălțimea în pixeli;

POKE (adr + 111), lungimea în caractere.

Aceste elemente sînt ilustrate în fig.5.1

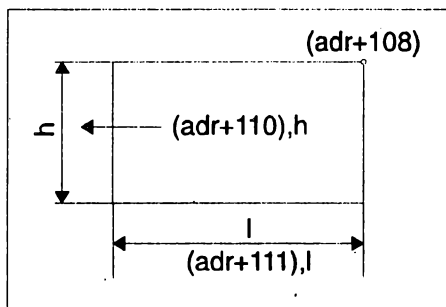


Fig 5.1

Programul BASIC de utilizare a rutinei este următorul:

```
10 BORDER 2: PAPER 1: INK 6: CLS : LET adr=60000:
  LET bs="32 caractere"
20 FOR n=0 TO 21: PRINT AT n,0;bs: NEXT n: PAUSE 50
30 FOR z=0 TO 255: POKE (adr+108),29: POKE
  (adr+109),72: POKE (adr+110),32: POKE
  (adr+111),28: RANDOMIZE USR adr: NEXT z
```

Exemplul 5.3:

```
10          ORG adr          ;SCROLL LENT DREAPTA
                BOX
20          ENT $
30          LD HL,(ET6)
40          LD BC,(ET7)
50 ET1       AND A
60 ET2       RR (HL)
70          INC HL
80          DEC B
90          JP NZ,ET2
100         LD HL,(ET6)
110        CALL ET3
120        LD (ET6),HL
130        LD BC,(ET7)
140        DEC C
150        RET Z
160        LD (ET7),BC
170        JP ET1
180 ET3      LD A,H
190         AND 7
200         CP 7
210         JP Z,ET4
220         INC H
```

```

230          RET
240 ET4      LD A,L
250          AND 224
260          CP 224
270          JP Z,ET3
280          LD DE,1760
290          AND A
300          SBC HL,DE
310          RET
320 ET5      LD A,H
330          CP 87
340          RET Z
350          LD DE,32
360          ADD HL,DE
370          RET
380 ET6      ADD A,D
390          LD C,B
400 ET7      LD BC,28

```

Definirea ferestrei se face astfel:

POKE (adr+71),TAB colțul stînga sus,

POKE (adr+72), 64(zona de sus)

72 (zona mijlocie)

80 (zona interioară),

POKE (adr+73), înălțime în pixeli,

POKE (adr+74),lungime în caractere (fig.5.2)

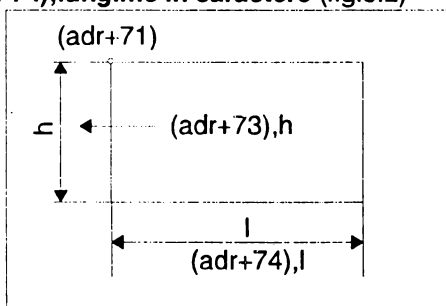


Fig.5.2

Programul *BASIC* aferent rutinei este următorul:

```

10 BORDER 2: PAPER 6: INK 1: CLS: LET adr=60000:
  LET bs="32 caractere"
20 FOR n=0 TO 21: PRINT AT n,0;bs: NEXT n: PAUSE 50
30 FOR z=0 TO 255: POKE (adr+71),2: POKE

```


(adr+72),72: POKE (adr+73),32: PÖKE (adr+74),
28: RANDOMIZE USR adr: NEXT z

5.2. INSTRUCȚIUNI DE INTRARE/IEȘIRE (I/E)

Microprocesorul Z80 permite folosirea a 256 porturi de intrare și 256 porturi de ieșire de 8 biți fiecare, externe memoriei sale.

Ele pot *introduce date* direct în acumulatorul A de la portul n cu instrucțiunea

IN A,(n) ; A = (n) ; n = 0...255-numărul portului

Similar se pot trimite date din acumulatorul A la portul n cu instrucțiunea

OUT (n),A ; (n) = A.

Aceste instrucțiuni nu afectează flagurile.

De asemenea se pot introduce date în orice registru de 8 biți folosind registrele r și C. Instrucțiunea

IN r,(C) ; r = C unde r = {B,C,D,E,H,L,A},

trimite conținutul registrului r la portul al cărui număr este în registrul C. Citirea tastaturii calculatorului se face cu această instrucțiune care afectează flagurile S,Z,P/V.

Instrucțiunea

IN F,(C)

este singura care afectează registrul F și anume numai fanioanele (flagurile).

Instrucțiunea

OUT (C),r

trimite conținutul registrului r la portul C

Instrucțiunile de I/E pot realiza și incrementări/decrementări folosind locația de memorie adresată de registrul HL și respectiv registrul C:

INI,IND,INIR, INDR OUTI,OUTD,OTIR,OTDR.

Rezumînd, instrucțiunile de I/E sînt următoarele:

- | | | |
|---|---|----------------------------------|
| • | IN A, (n) ; n = 0..255 (nr.portului) | ; A = (n); LET A = PEEK n |
| • | IN r,(C) ; r = {B,C,D,E,H,L,A} | ; r = C; LET r = PEEK rc |

•	INI	; (HL) = (C); HL = HL + 1; B = B - 1] Z = 1 dacă B = 0 după execuție, altfel Z = 0
•	IND	; (HL) = (C); HL = HL - 1; B = B - 1	
•	IN F, (C)] Se repetă pînă cînd B = 0
•	INIR	; (HL) = (c); HL = HL + 1; B = B - 1	
•	INDR	; (HL) = (C); HL = HL - 1; B = B - 1] Se repetă pînă cînd B = 0
•	OUT (n), A	; n = 0..255 (nr. portului)	
•	OUT (C), r	; r = B, C, D, E, H, L, A	; (C) = A; POKE n, A ; (C) = r; POKE rC, r
•	OUTI	; (C) = (HL) ; HL = HL + 1 ; B = B - 1] Se repetă pînă cînd B = 0
•	OUTD	; (C) = (HL) ; HL = HL - 1 ; B = B - 1	
•	OTIR	; (C) = (HL) ; HL = HL + 1 ; B = B - 1] Se repetă pînă cînd B = 0
•	OTDR	; (C) = (HL) ; HL = HL - 1 ; B = B - 1	

Cel mai important port este 254 folosit pentru introducerea datelor de la tastatură (biții b0..b6); este utilizat și pentru culoarea BORDER-ului (biții b0..b2).

Exemplificarea 5.2:

```

10          ORG 320000
20          ENT $
30          LD B, 191
40  ET1     LD C, 254
50          IN A, (C)
60          AND 7
70          LD D, A
80          OUT (254), A
90          DEC D
100         LD C, 254
110        OUT (C), D
120        JR ET1

```

Nr. liniei	Ce se realizează	Conținutul registrelor	Flagul S
30	Registrul B se încarcă cu 191	A=0; B=0; C=0; D=0	<u>S=0</u>
40	Registrul C se încarcă cu 254	<u>B=191</u>	idem
50	IN de portul al cărui număr 254 este în C, în registrul A	B=191; <u>C=254</u>	idem
60	ȘI logic între registrul A și numărul 7	<u>A=191</u> ; B=191; C=254	<u>S=1</u>
70	Registrul D se încarcă cu valoarea din registrul A	<u>A=7</u> ; B=191; C=254	<u>S=0</u>
80	OUT valoarea din registrul A și portul 254	A=7; B=191; C=254; <u>D=7</u>	idem
90	Registrul D se decrementează cu 1	A=7; B=191; C=254; D=7	idem
100	Registrul C se încarcă cu 254	A=7; B=191; C=254; <u>D=6</u>	idem
110	OUT valoarea din registrul D în portul al cărui număr 254 este în registrul C	<u>A=254</u> ; B=191; C=254; D=6	idem
120	Execută salt relativ la adresa ET1	A=254; B=191; C=254; D=6	idem

5.3. CULOAREA

Pentru fiecare poziție de afișare culoarea este definită printr-un octet de atribute care codifică diferitele caracteristici de culoare conform schemei.

bitul

7	6	5 4 3	2 1 0
FLASH	BRIGHT	PAPER	INK

Prin urmare este important să se cunoască pozițiile ocupate în octet de biții respectivi.

Programele următoare sînt de uz general pentru schimbarea culorilor PAPER-ului și INK-ului.

Exemplul 5.4:

```

10          ORG adr          ;SCHIMBAREA CULORII
                                PAPER
20          ENT $
30  ET1     DEFB 5           ;PAPER 5
40          DEFB 2           ;INK 2
50          LD HL,22528      ;ZONA ATRIBUTELOR
60          LD A, (ET1)      ;culoarea PAPER
70          AND 7            ;IZOLEAZA BITII
                                b0 ,b1 ,b2
80          ADD A,A          ;2A
90          ADD A,A          ;4A
100         ADD A,A          ;8A
110         LD B,A
120  ET2     LD A, (HL)
130         AND 199          ;MASCHEAZA CULOAREA
                                PRECEDENTA
140         OR B             ;PUNE CULOAREA
                                ALEASA
150         LD (HL),A
160         INC HL
170         LD A,91
180         SUB H
190         JR NZ,ET2
200  ZEND     RET

```

În linia 30 se introduce - prin *DEFB* - codul culorii pentru **PAPER**, iar instrucțiunea **AND 7** (linia 70) elimină cei 5 biți semnificativi pentru cazul cînd s-a introdus un cod de culoare mai mare decît 7. Cele trei instrucțiuni **ADD** (liniile 80-100) multiplică **A** cu 8 și au ca scop aducerea culorii **PAPER** în biții b5, b4, b3 ai registrului **A**. Instrucțiunea **AND 199** maschează culoarea precedentă și **OR B** introduce culoarea nouă.

Cînd fișierul de atribute, care are 768 octeți, este complet, atunci registrul dublu **HL** are valoarea 23296 (adică $H=91$, $HL=256*H+L=256*91=23296$), iar instrucțiunea **SUB H** va pune flagul **Z** pe 1 cînd $H=91$.

După asamblarea rutinei, se tastează următorul program *BASIC*:

```
10 CLS : FOR n=0 TO 21: PRINT "32 caractere": NEXT
  n
20 FOR i=0 TO 7: POKE adr,i: RANDOMIZE USR adr:
  PAUSE 50: NEXT i
```

O altă variantă de schimbare a culorii *PAPER* sub forma depunerii unor "pete" de culoare este realizată de rutina de mai jos:

Exemplul 5.5:

```
10          ORG adr          ;PETE DE CULOARE
                PAPER

20          ENT $
30          XOR A
40          LD B,A
50  ET      LD HL,22528
60          CALL PETE
70          LD HL,22528+256
80          CALL PETE
90          LD HL,22528+512
100         CALL PETE
110        ADD A,INCR
120        HALT
130        DJNZ ET
140  ZEND   RET
150  INCR   EQU 231
160  PETE   PUSH AF
170        ADD A,L
180        LD L,A
190        LD A,0
200        ADC A,H
210        LD H,A
220        LD (HL),8*5      ;CULOAREA PAPER*8
230        POP AF
240        RET
```

Rutina se activează cu comanda *BASIC* ; RANDOMIZE USR adr: scoțind instrucțiunea *HALT* schimbarea culorii *PAPER*-ului se face instantaneu. Efectul grafic poate fi obținut pe diverse zone ale ecranului modificând programul astfel: fără liniile 70-100 numai treimea superioară, fără liniile 70-80 și 90-100 numai treimea mijlocie, fără liniile 90-100

primele două zone etc.

Exemplul 5.6:

```

10          ORG adr1          ;SCHIMBAREA CULORII
                                INK
100        ENT $
110        LD HL,22528
120        LD A,(adr+1)      ;adr de la ex.5.4
130        AND 7
140        LD B,A
150 ET3    LD A,(HL)
160        AND 248
170        OR B
180        LD (HL),A
190        INC HL
200        LD A,91
210        SUB H
220        JR NZ,ET3
230        RET

```

Programul BASIC aferent este similar cu cel de la exemplul 5.4 cu modificarea liniei 20 după cum urmează:

```

20 FOR i=0 TO 7: POKE (adr+1),i: RANDOMIZE USR
   adr1: PAUSE 50: NEXT i

```

- Rutina de la exemplul 5.7 realizează efecte pe BORDER

Exemplul 5.7

```

10          ORG 60000          ;EFECTE PE BORDER
110        ENT S
120        LD HL,01343
130        PUSH HL
140        LD HL,65048
150        BIT 7,A
160        JR NZ,ET1
170        LD HL,03224
180 ET1    EX AF,AF
190        INC DE
200        DEC IX
210        LD A,2
220        LD B,A
230 ET2    DJNZ ET2
240        OUT (254),A
250        XOR 15

```

```

170          LD B,10
180          DEC L
190          JR NZ,ET2
200          DEC B
210          DEC H
220          JR NZ,ET2
230 ZEND     RET

```

Durata efectului și culoarea pot fi schimbate introducând noile valori prin **POKE 60000,durata (1;127)** și respectiv **POKE 60020, culoarea (0 la 7)**.

O aplicație interesantă este schimbarea atributelor într-o fereastră (box), așa cum se prezintă în următoarele patru rutine.

Exemplul 5.8:

```

10          ORG adr          ;SCHIMBAREA ATRIBUTE
                                BOX
20          ENT $
30          LD A,70
40          LD BC,773
50          LD DE,3609
60          LD HL,22495
70          PUSH DE
80          LD DE,32
90          INC B
100 ET1     ADD HL,DE
110        DJNZ ET1
120        LD B,C
130        INC B
140 ET2     INC HL
150        DJNZ ET 2
160        POP DE
170        LD B,D
180 ET3     PUSH HL
190        LD C,B
200        LD B,E
210 ET4     LD (HL),A
220        INC HL
230        DJNZ ET4
240        POP HL
250        PUSH DE
260        LD DE,32

```

```

270          ADD HL,DE
280          POP DE
290          LD B,C
300          DJNZ ET3
310 ZEND     RET

```

Se reamintește că atributele se determină folosind relația

INK (0-7)+8*PAPER (0-7)+64*BRIGHT (0-1)*FLASH (0-1)

De pildă pentru **INK 0, PAPER 5, BRIGHT 0 și FLASH 0** rezultă
 $0+8*5+0+0=40$.

Introducerea atributelor și a dimensiunilor ferestrei se face astfel:

**POKE (adr+1), ATTR : POKE (adr+4), linia: POKE (adr+3),
coloana: POKE (adr+6), lungime box: POKE (adr+7),
înălțime box**

unde *linia, coloana* sînt coordonatele colțului stînga sus ale box-ului.

De pildă, un program *BASIC* ce folosește această rutină ar fi următorul:

```

10 BORDER 1: CLS : LET a$="32 caractere": FOR n=0
   TO 21: PRINT AT n,0;a$: NEXT n
20 LET adr=60000: POKE (adr+1),40: POKE (adr+4),8:
   POKE (adr+3),4: POKE (adr+6),24: POKE (adr+7),4:
   RANDOMIZE USR adr

```

Rutina este utilă pentru a colora anumite zone din ecran unde sînt scrise concluzii sau formule. Evident că dacă boz-ul este identic cu întregul ecran, atunci instrucțiunile programului *BASIC* se modifică corespunzător:

**POKE (adr+1), ATTR: POKE (adr+4),0: POKE (adr+3),0:
POKE (adr+6),32: POKE (adr+7),22**

Exemplul 5.9:

```

10          IRG adr          ;SCROLL ATTRIBUTE BOX
                                IN SUS
20          ENT $
30          LD A,96
40          LD BC,772
50          LD DE,3608
60          LD HL,22495
70          PUSH AF
80          PUSH DE
90          LD DE,32
100         INC B

```



```

110 ET1      ADD HL,DE
120          DJNZ ET1
130          LD B,C
140          INC B
150 ET2      INC HL
160          DJNZ ET2
170          POP DE
180          LD B,D
190          DEC B
200 ET3      PUSH BC
210          LD B,0
220          LD C,E
230          PUSH DE
240          PUSH HL
250          LD DE,32
260          ADD HL,DE
270          POP DE
280          PUSH HL
290          LDIR
300          POP HL
310          POP DE
320          POP BC
330          LD A,B
340          DJNZ ET3
350          POP AF
360          LD B,E
370 ET4      LD (HL),A
380          INC HL
390          DJNZ ET4
400 ZEND    RET

```

Determinarea caracteristicilor box-ului se face la fel ca la exemplul 5.8 cu deosebirea că ele se dispun într-un ciclu la care variabila de ciclare (z) trebuie să fie în concordanță cu înălțimea box-ului. De pildă:

```

10 BORDER 1: CLS: FOR n=0 TO 21: PRINT AT n,0; "32
   caractere": NEXT n
20 LET adr=60000: FOR z=1 TO 10: POKE (adr+1),40:
   POKE (adr+4),4: POKE (adr+3),4: POKE (adr+6),24:
   POKE (adr+7),10 RANDOMIZE USR adr: NEXT z

```

Exemplul 5.10:

```

10          ORG adr          ;SCROLL ATTRIBUTE BOX

```

```
20      ENT $
30      LD A,24
40      LD BC,1028
50      LD DE,2584
60      LD HL,22495
70      DEC D
80      PUSH AF
90      LD A,B
100     ADD A,B
110     LD B,A
120     LD A,C
130     ADD A,E
140     LD C,A
150     PUSH DE
160     LD DE,32
170     INC B
180     ET1    ADD HL,DE
190     DJNZ ET1
200     LD B,C
210     ET2    INC HL
220     DJNZ ET2
230     POP DE
240     LD B,D
250     ET3    PUSH BC
260     LD B,0
270     LD C,E
280     PUSH DE
290     PUSH HL
300     LD DE,32
310     AND A
320     SBC HL,DE
330     POP DE
340     PUSH HL
350     LDDR
360     POP HL
370     POP DE
380     POP BC
390     LD A,B
400     DJNZ ET3
410     POP AF
```

```

420          LD B,E
430 ET4      LD (HL),A
440          DEC HL
450          DJNZ ET4
460 ZEND     RET

```

Observațiile privind caracteristicile box-ului sînt identice cu cele formulate la exemplul 5.9 și implicit programul *BASIC* de exploatare a rutinei este identic.

Exemplul 5.11:

```

10          ORG adr          ;SCROLL ATTRIBUTE BOX
                                STINGA
20          ENT $
30          LD A,112
40          LD BC,513
50          LD DE,1821
60          LD HL,22495
70          PUSH AF
80          PUSH DE
90          LD DE,32
100         INC B
110 ET1      ADD HL,DE
120         DJNZ ET1
130         LD B,C
140         INC B
150 ET2      INC HL
160         DJNZ ET2
170         POP DE
180         LD B,D
190         POP AF
200 ET3      PUSH BC
210         LD B,0
220         LD C,E
230         DEC C
240         PUSH DE
250         PUSH HL
260         POP DE
270         NOP
280         PUSH DE
290         INC HL
300         LDIR

```

```

310          LD (DE) , A
320          POP HL
330          LD DE , 32
340          ADD HL , DE
350          POP DE
360          POP BC
370          DJNZ ET3
380  ZEND    RET

```

Programul *BASIC* cu caracteristicile box-ului este identic cu cel de la exemplul 5.9.

O aplicație grafică interesantă reprezintă deplasarea atributelor (la dreapta, la stînga, în sus, în jos, într-o fereastră). În astfel de situații este rațional să se apeleze la instrucțiuni de tip repetitiv (*LDI*, *LDR*, *LDIR*, *LDDR*).

Exemplul 5.12:

```

10          ORG adr          ; SCROLL ATTRIBUTE
                                DREAPTA

120         ENT $

130         LD HL , 22528

140         PUSH HL

150         POP DE

160         LD B , 24

170  ET1     PUSH BC

180         LD A , (HL)      ; O ROTATIE COMPLETA

190         INC HL

200         LD BC , 8192     ; B=32 , C=0

210  ET2     LDI

220         DJNZ ET2        ; CICLUL DE 31 ORI

230         LD (DE) , A

240         INC DE

250         POP BC

260         DJNZ ET1

270  ZEND    RET

280         ORG adr+23      ; ATTRIBUTE PE TOT
                                ECRANUL

290         LD HL , 6400

300         LD DE , 22528

310         LD BC , 768

320         LDIR

330         RET

```

În liniile 180-230 s-a introdus exemplul 3.12 (folosirea atributelor) pentru a pune în valoare rutina de deplasare a acestora (locată la adresa *adr*):

```
10 LET aS="32 caractere": CLS : FOR n=0 TO 21:
   PRINT AT n,0;a$: NEXT n
20 PAUSE 50: RANDOMIZE USR (adr+23): REM aparitia
   atributelor
30 FOR i=1 TO 32: RANDOMIZE USR adr: NEXT i: REM
   deplasare attribute la dreapta
```

Exemplul 5.13:

```
240                ORG adr+35                ;SCROLL ATTRIBUTE
                                   STINGA

250                LD HL,23295
260                PUSH HL
270                POP DE
280                LD B,24
290 ET3            PUSH BC
300                LD A,(HL)
310                DEC HL \
320                LD BC,8192
330 ET4            LDD
340                DJNZ ET4
350                LD (DE),A
360                DEC DE
370                POP BC
380                DJNZ ET3
390                NOP
400                RET
```

După cum se poate constata această rutină s-a scris în continuarea celei precedente, din două motive: să folosească rutina de la adresa (*adr+23*) care umple ecranul cu atribute și respectiv să unească cele două efecte prin introducerea unei noi linii la programul *BASIC*:

```
40 FOR i=1 TO 32: RANDOMIZE USR (adr+35): NEXT i:
   REM deplasare attribute la stînga
```

Exemplul 5.14:

```
410                ORG adr+58                ;SCROLL ATTRIBUTE IN
                                   SUS
420 ET5            NOP
430                DEFS 31                ;REZERVA 31 OCTETI
440                LD HL,22528
```

```

450          LD DE,ET5
460          LD BC,32
470          LDIR
480          LD DE,22528
490          LD BC,736
500          LDIR
510          LD HL,ET5
520          LD BC,32
530          LDIR
540          RET

```

La programul *BASIC* anterior se adaugă linia

```

50 FOR i=1 TO 32: RANDOMIZE USR (adr+58): NEXT i:
  REM deplasare attribute în sus

```

Exemplul 5.15:

```

550          ORG adr+118          ;SCROLL ATTRIBUTE IN
                                   JOS
560 ET6      NOP
570          LD HL,23295
580          LD DE,ET6
590          LD BC,32
600          LDDR
610          LD DE,23295
620          LD BC,736
630          LDDR
640          LD HL,ET6
650          LD BC,32
660          LDDR          ;PUNE LINIA 24 IN
                                   PRIMA LINIE
670 ZEN      RET

```

Se adaugă linia *BASIC*:

```

60 FOR i=1 TO 32: RANDOMIZE USR (adr+118):
  NEXT i: REM deplasare attribute în jos

```

Eventual se poate face o repetare la nesfârșit a acestor rutine completând programul *BASIC* cu linia

```

70 GO TO 30

```

Exemplul 5.16:

```

10          ORG adr          ;SCROLL ATTRIBUTE BOX
                                   DREAPTA
10          ENT $
30          LD A,1

```

```
40          LD BC,2051
50          LD DE,3098
60          LD HL,22495
70          DEC D
80          PUSH AF
90          LD A,B
100         ADD A,D
110         LD B,A
120         LD A,C
130         ADD A,E
140         LD C,A
150         PUSH DE
160         LD DE,32
170         INC B
180 ET1     ADD HL,DE
190         DJNZ ET1
200         LD B,C
210 ET2     INC HL
220         DJNZ ET2
230         POP DE
240         LD B,D
250         INC B
260         POP AF
270 ET3     PUSH BC
280         LD B,0
290         LD C,E
300         DEC C
310         PUSH DE
320         PUSH HL
330         POP DE
340         LD A,(DE)
350         PUSH DE
360         DEC HL
370         LDDR
380         LD (DE),A
390         POP HL
400         LD DE,32
410         AND A
420         SBC HL,DE
430         POP DE
440         POP BC
```

```

450          DJNZ ET3
460 ZEND      RET
470          ORG adr+160      ;ATTRIBUTE PE TOT
                                ECRANUL
480          LD HL,6400
490          LD DE,22528
500          LD BC,768
510          LDIR
520          RET

```

Caracteristicile box-ului se stabilesc după modelul indicat la exemplul 5.9, iar programul *BASIC* este similar; de pildă pentru adresa *adr*=65143:

```

10 CLEAR 65000: LET adr=65143: RANDOMIZE USR
    adr+160: REM aparitia atributelor pe tot ecranul
20 FOR z=1 TO 255: POKE (adr+1),40: POKE (adr+4),4:
    POKE (adr+3),4: POKE (adr+6),24: POKE (adr+7),8:
    RANDOMIZE USR adr: NEXT z

```

• În încheierea paragrafului consacrat culorilor se prezintă 2 rutine care realizează un efect grafic coloristic ce poate fi folosit în programe:

Exemplul 5.17:

```

10          ORG adr          ;EFECT COLORISTIC
20          ENT $
30          LD B,127
40 ET1      LD HL,22528
50 ET2      LD A,R
60          LD (HL),A
70          INC HL
80          LD A,91
90          CP H
100         JR NZ,ET2
110        DJNZ ET1
120 ZEND      RET

```

Rutina are doar 17 octeți și se activează cu comanda **RANDOMIZE USR adr.**

Exemplul 5.18:

```

10          ORG adr          ;BENZI VERTICALE
                                COLORATE
20          ENT $
30          LD HL,22528
40          LD BC,3*256      ;CITE ZONE SE

```

COLOREAZA

```

50 ET1      LD A,L
60          AND 12      ;SE POT PUNE ALTE
                       VALORI: 20,44,60,111

70          RLCA
80          XOR 56
90          LD (HL),A
100         INC HL
110        DEC BC
120        LD A,B
130        OR C
140        JR NZ,ET1
150 ZEND    RET

```

Programul *BASIC* aferent rutinei:

```
10 BORDER 5: CLS : RANDOMIZE USR adr
```

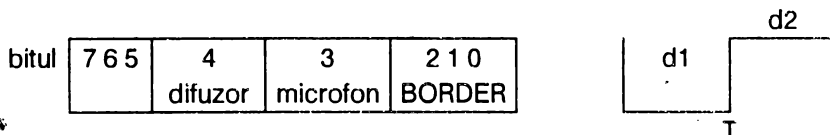
Pentru a colora parțial ecranul se efectuează modificările:

- pentru treimea superioară: **40 LD BC,256**
- pentru primele 2 zone: **40 LD BC,2*256**
- pentru ultima treime **30 LD HL,23040**
40 LD BC,256

5.4. SUNETE

5.4.1. Codificarea unei melodii

Portul de ieșire 254 comandă difuzorul (bitul b4), microfonul (bitul b3) și culoarea BORDER-ului (biții b2,b1,b0), conform schemei următoare:



Pentru a se acționa difuzorul trebuie pus bitul b4 pe 0 și apoi pe 1 pentru a se realiza o perioadă T . Numărul de perioade pe secundă este frecvența f și ea reprezintă înălțimea sunetului. Prin urmare, pentru a se

produce un sunet de frecvență f trebuie pus bitul b4 pe 0 un timp d_1 apoi pe 1 în timpul d_2 ; p pentru simetrie se face $d_1 = d_2$, dar se pot încerca durate asimetrice ceea ce modifică timbrul sunetului. În tabelul 5.1 sînt prezentate caracteristicile a trei octave: gama centrală (notată cu indicele 3), octava inferioară (notată cu indicele 2) și octava superioară (notată cu indicele 4). Tabelul conține pentru fiecare notă: frecvența f , perioada T și valorile corespunzătoare frecvenței cu care trebuie încărcat registrul dublu HL. Ca element de bază s-a luat frecvența hotei la3 (adica 430 Hz) a diapazonului; frecvențele notelor de sub la3 se determină prin împărțirea frecvenței anterioare la $r = \sqrt[12]{2} = 1,0594631 \approx 1,05$, iar cele de deasupra lui la3 prin înmulțire cu r . În mod analog, perioadele corespunzătoare notelor sub la3 s-au calculat cu relația $T = T_{\text{anterior}} * 1,05$ iar cele corespunzătoare notelor de deasupra lui la3 prin împărțire cu 1,05. Același procedeu s-a utilizat și pentru calcularea valorilor din coloana HL.

Tabelul 5.1

Nota	Frecvența f (Hz)	Perioada T (μ s)	HL
do2	127	7823	3363
do#	135	7384	3174
re	143	6970	2996
re#	152	6578	2828
mi	161	6209	2669
fa	170	5861	2519
fa#	180	5532	2378
sol	191	5221	2244
sol#	202	4928	2118
la	214	4652	2000
la#	227	4390	1887
si	241	4144	1781
do3	<u>256</u>	3911	1681
do#	270	3692	1587
re	286	3485	1498
re#	304	3289	1414
mi	322	3104	1334
fa	341	2930	1259

fa#	361	2766	1189
sol	383	2610	1122
sol#	405	2464	1059
la	<u>430</u>	<u>2326</u>	<u>1000</u>
la#	455	2195	943
si	482	2071	890
do4	511	1955	840
do#	541	1845	793
re	573	1742	749
re#	608	1644	707
mi	644	1552	667
fa	682	1465	629
fa#	723	1382	594
sol	766	1305	561
sol#	811	1231	529
la	860	1162	499
la#	911	1097	471
si	965	1035	445

Deoarece do3 are o frecvență de 256 cicli/sec, s-a stabilit valoarea 256 pentru a măsura durată unei secunde. Valorile duratelor se încarcă în registrul DE; prin urmare se vor reține următoarele valori

1 sec	1/2 sec	1/4 sec	1/8 sec
256	128	64	32

În fine, este de reținut că rutina de sunete din ROM este la adresa 949 și poate fi apelată cu instrucțiunea **CALL 949**.

- Primul program demonstrează producerea sunetelor.

Exemplul 5.19:

10	ORG adr	; SUNETELE CALCULATORULUI
20	ENT \$	
30	LD A,8	
40	LD BC,256	;durata sunetului
50 ET1	LD DE,127	;frecventa sunetului do2
60 ET2	LD H,A	;salveaza A
70	DEC DE	
80	LD A,D	

```

90          OR E
100         LD A,H
110         JR NZ,ET2
120         XOR 16          ;schimba bitul b4
                             din A
130         OUT (254),A     ;catre difuzor
140         LD H,A          ;salveaza A
150         DEC BC
160         LD A,B
170         OR C
180         LD A,H
190         JR NZ,ET1
200  ZEND   RET

```

Instrucțiunea **LD A,8** (linia 30) face bitul $b_3=1$, adică nu se exploatează microfonul. Atunci când bitul $b_4=1$, instrucțiunea **XOR 16** îl aduce pe 0 (și lasă ceilalți biți neschimbați), iar când $b_4=0$ aceeași instrucțiune **XOR 16** îl aduce pe 1.

Programul *BASIC* de folosire a rutinei este următorul:

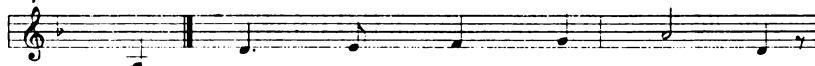
```

10 FOR n=128 TO 4 STEP -4: RANDOMIZE USR adr: POKE
   (adr+6),n: NEXT n

```

El schimbă de 31 ori frecvența sunetului diminuând perioada T (deci crește frecvența $f=1/T$).

- O modalitate de lucru este codificarea partiturii muzicale folosind datele din tabelul 5.1; un exemplu este ilustrat în fig.5.3 unde este prezentată această decodificare pentru primele două măsuri ale imnului "DEȘTEAPTĂ-TE ROMÂNE" de Anton Pann.



HL	2000	1498	1334	1259	1122	1000	1489
DE	128	160	64	128	128	256	128

Fig.5.3.

Programul corespunzător acestei codificări și care folosește rutina de sunete din ROM de la adresa 949 este următorul:

Exemplul 5.20:

```

10          ORG adr          ;DEȘTEAPTA-TE ROMANE
20          ENT $
30          LD HL,2000

```

40	LD DE,128	;durata unei patrimi
50	CALL 949	
60	LD HL,1498	
70	LD DE,160	;durata unei patrimi cu punct
80	CALL 949	
90	LD HL,1334	
100	LD DE,64	
110	CALL 949	
120	LD HL,1259	
130	LD DE,128	
140	CALL 949	
150	LD HL,1122	
160	LD DE,128	
170	CALL 949	
180	LD HL,1000	
190	LD DE,256	durata unei doimi
200	CALL 949	
210	LD HL,1498	
220	LD DE,128	
230	CALL 949	
240	ZEND RET	

Din analiza programului se vede că structura lui este foarte simplă și anume:

LD HL, codul corespunzător notei (din tabelul 5.1)

LD DE, durata notei

CALL 949

referindu-se la o singură notă din partitură, ceea ce este un dezavantaj de scriere în cazul melodiilor lungi. Din acest motiv, se prezintă în continuare o variantă care elimină acest dezavantaj, perechile de valori ce se introduc în cele două registre duble HL și respectiv DE fiind declarate în directiva de asamblare **DEFW**. Numărul perechilor de valori este încărcat în registrul B.

Exemplul 5.21:

10	ORG	;DESTEAPTA-TE ROMANE (var.2)
20	ENT \$	
30	LD B,7	
40	LD HL,ET1	

```

50          LD DE,ET2
60  ETO     PUSH BC
70          LD C,(HL)
80          INC HL
90          LD B,(HL)
100         INC HL
110        EX DE,HL
120        PUSH DE           ;non ET1 pe stiva
130        LD E,(HL)
140        INC HL
150        LD D,(HL)       ;DE=continut ET2
160        INC HL
170        PUSH BC         ;HL=continut ET1
180        EX (SP),HL     ;non ET2 pe stiva
190        CALL 949
200        POP DE
210        POP HL
220        POP BC
230        DJNZ ETO
240  ZEND   RET
250  ET1    DEFW 2000,1498,1334,1259,1122,1000,
           1498
260  ET2    DEFW 128,160,64,128,128,256,128

```

5.4.2. Sunete diverse

Programele care urmează oferă sunete variate, utile în programe proprii.

Exemplul 5.22

```

10          ORG adr           ;SUNET "IMPUSCATURI"
20          ENT $
30          LD B,10
40          PUSH BC
50          LD HL,768
60  L0      LD DE,1
70          PUSH HL
80          CALL 949
90          POP HL
100         LD DE,16
110        AND A

```

```

120          SBC HL,DE
130          JR NZ,L0
140          POP BC
150 ZEND     RET

```

Program *BASIC* de utilizare a rutinei:

```
10 FOR i=1 TO 10: RANDOMIZE USR adr: NEXT i
```

Exemplul 5.23

```

10          ORG adr          ;SUNET "OZN"
20          ENT $
30          LD DE,25698
40 L1       LD H,50
50          LD A,(23624)
60          RRA
70          RRA
80          RRA
90 L2       LD C,254
100         XOR 16
110         OUT (C),A
120         LD B,E
130 L3       DJNZ L3
140         DEC H
150         JR NZ,L2
160         INC E
170         DEC D
180         JR NZ.L1
190 ZEND     RET

```

Pot fi modificate frecvența, viteza, durata și derularea sunetului (în sus sau în jos) folosind instrucțiunile

POKE (adr+1), frecvența:

POKE (adr+2), viteza:

POKE (adr+4), durata:

POKE (adr+23),28 (în sus) sau 29 (în jos).

Exemplul 5.24:

```

10          ORG adr          ; SUNET "CLAXON"
20          ENT $
30          LD A,(23624)
40          RRA
50          RRA
60          RRA
70          LD B,240

```

```

80          LD C,254
90  ET1     DEC H
100         JR NZ,ET2
110         XOR 16
120         OUT (C),A
130         LD H,238
140  ET2     DEC L
150         JR NZ,ET1
160         XOR 16
170         OUT (C),A
180         LD L,254
190         DJNZ ET1
200         RET

```

Modificări se realizează cu:

POKE (adr+7), durata:

POKE (adr+18), frecvența 1:

POKE (adr+27), frecvența 2;

de exemplu:

```

10 LET adr=64675: POKE (adr+7),200: POKE
   (adr+18),238: POKE (adr+27),254: RANDOMIZE USR
   adr

```

Exemplul 5.25:

```

10          ORG adr          ;MULTIBEAP
20          ENT $
30          LD BC,64004
40          LD HL,512
50          LD DE,15
60  ET1     PUSH HL
70         PUSH DE
80         PUSH BC
90         CALL 949
100        POP BC
110       POP DE
120       POP HL
130       LD A,L
140       SUB C
150       LD L,A
160       DJNZ ET1
170  ZEND   RET

```

Cu **POKE (adr+1), decrement înălțime:POKE (adr+2),**

nr.note:POKE (adr+4) și POKE (adr+5), înălțime: P(OKE (adr+7) și POKE (adr+8), timp se fac modificări.

De pildă:

```
10 LET adr=63010: POKE (adr+1),4: POKE (adr+2),250:
   POKE (adr+4),33: POKE (adr+5),2: POKE
   (adr+7),15: POKE (adr+8),0: RANDOMIZE USR adr
```

5.4.3. Efecte pe BORDER cu sunete

- Cele trei programe care urmează realizează efecte pe BORDER însoțite de sunete care pot fi discrete sau stridente.

Exemplul 5.26:

```
10          ORG adr          ;EFECTE PE BORDER CU
                                SUNET DISCRET
20          ENT $
30          LD DE,1000
40 ET1      LD A,255
50          LD B,A
60 ET2      OUT (254),A
70          XOR A
80          DJNZ ET2
90          DEC DE
100         LD A,D
110         OR E
120         JR NZ,ET1
130 ZEND    RET
```

Tastînd **RANDOMIZE USR adr**, BORDER-ul își modifică culoarea, apar dungi subțiri însoțite de un sunet discret plăcut.

Exemplul 5.27:

```
10          ORG adr          ;EFECTE PE BORDER CU
                                SUNET INTRERUPT
20          ENT $
30          LD HL,1343
40          PUSH HL
50          LD HL,1664
60          BIT 7,A
70          JR Z,L1
80          LD HL,324
90 L1      EX AF,AF
```

```

100          INC DE
110          DEC IX
120          LD A,3
130 L2       LD B,A
140 L3       DJNZ L3
150          OUT (254),A
160          RRA
170          RRCA
180          LD B,131
190          DEC L
200          JR NZ,L3
210          DEC B
220          DEC H
230          JP P,L2
240 ZEND     RET

```

Cu **RANDOMIZE USR adr** se obțin dungi late pe **BORDER** și un sunet întrerupt cu semnificație de "atenționare".

Exemplul 5.28:

```

10          ORG adr          ;EFEȚTE PE BORDER CU
                               SUNET STRIDENT
20          ENT $
30          LD DE,1000
40 X1       LD B,255
50          LD A,(44880)
60 X2       OUT (254),A
70          DEC A
80          DJNZ X2
90          DEC DE
100         LD A,D
110         OR E
120         RET Z
130         JR Z1
140         NOP
150         NOP
160         LD DE,1000
170 X3      LD A,182
180         LD B,A
190 X4      OUT (254),A
200         XOR A
210         DJNZ X4

```

```

220          DEC DE
230          LD A,D
240          OR E
250          JR NZ,X3
260 ZEND     RET

```

Activarea rutinei se face cu comanda **RANDOMIZE USR adr**; se obțin dungi late suprapuse colorate diferit și un sunet pătrunzător de atenționare.

Toate cele trei rutine din exemplele 5.24...5.26 pot fi folosite în programe proprii ca efecte vizuale de trecere între două ecrane diferite (cortine).

- În fine, în încheierea paragrafului consacrat sunetelor se prezintă două rutine care realizează efecte pe **BORDER**, sunete și deplasarea caracterelor (absorbție în **PAPER**), respectiv defilarea atributelor.

Exemplul 5.29:

```

10          ORG adr          ;EFECTE BORDER,SUNET
                                SI CLS
20          ENT $
30          LD HL,22527
40          LD C,192
50 K1       LD B,32
60          XOR A
70 K2       RL (HL)
80          DEC HL
90          LD A,56
100         XOR B
110         OUT (254),A
120         DJNZ K2
130         DEC C
140         JR NZ,K1
150 ZEND     RET

```

Programul *BASIC* pentru această rutină:

```
10 FOR i=0 TO 7: RANDOMIZE USR adr: NEXT i
```

Se obține un sunet gen telefon, **CLS** prin absorbția caracterelor în **PAPER** (deplasare spre stânga) și dungi subțiri frunte pe **BORDER** colorat.

Exemplul 5.30:

```

10          ORG adr          ;EFECTE BORDER,
                                SUNET,DEPLASARE

```

ATTRIBUTE

20		ENT \$
30		LD A,0
40	M0	LD HL,22399
50		LD E,A
60		LD C,192
70	M1	LD B,32
80		XOR A
90	M2	RLC (HL)
100		DEC HL
110		XOR B
120		OUT (254),A
130		DJNZ M2
140		DEC C
150		JR NZ,M1
160		LD A,E
170		INC A
180		CP 8
190		JR NZ,M0
200	ZEND	RET

Cu **RANDOMIZE USR adr** rezultă dungi subțiri paralele pe **BORDER** colorat, un sunet gen telefon și o deplasare (roll) pentru fiecare coloană a ecranului cu revenire la imaginea inițială. Dacă se dorește ștergerea ecranului se va înlocui instrucțiunea **RLC (HL)** din linia 90 cu **RL (HL)**. Dacă se înlocuiește instrucțiunea **XOR A** din linia 80 cu **OR A** se obține un efect interesant.

5.5. SCRIEREA TEXTELOR

5.5.1. Scrierea unei linii cu 32 caractere

Alăturat se prezintă două rutine pentru scrierea unui text de maximum 32 caractere:

Prima rutină

LD A,2 ; canalul 2

A doua rutină

LD HL,TEXT

<pre> CALL 5633 ;selectioneaza ecranul LD HL,TEXT LD B,END-TEXT ET LD A, (HL) RST 16 INC HL DJNZ ET ZEND RET </pre>	<pre> CALL PRINT ZEND RET PRINT LD A, (HL) INC HL OR A RET Z RST 16 JR PRINT </pre>
---	--

După cum se observă, adresa textului ce trebuie afișat este încărcat în registrul dublu HL; textul propriu zis va fi introdus prin directiva de asamblare DEFM, iar poziția de scriere (echivalentă cu **PRINT AT linie, coloană**) se obține folosind codul pentru **AT** (adică 22-v.fig.3.6) urmată de numărul liniei, numărul coloanei introduse prin directiva **DEFB** (de exemplu: echivalentul lui **PRINT AT 11,3** se va scrie **DEFB 22,11,3**).

Se ilustrează aplicarea celor două rutine pentru afișarea textului
M.M.POPOVICI STOFWARE 1993

pe linia 11, începînd din coloana 3.

Exemplul 5.31:

```

10          ORG adr          ;AFISARE TEXT 32
                                CARACTERE (v.1)

20          ENT $

30          LD A,2

40          CALL 5633

50          LD HL,TEXT

60          LD B,END-TEXT

70 ET      LD A, (HL)

80          RST 16          ;RUTINA DE IMPRIMARE
                                A UNUI CHARACTER

90          INC HL

100         DJNZ ET

110        ZEND          RET

120        TEXT         DEFB 22,11,3    ;AT 11,3

130         DEFM "M.M.POPOVICI SOFTWARE 1993"

140        END

```

Pentru a se introduce culori, **FLASH**, **BRIGHT** etc. se vor folosi codurile următoare extrase din fig.3.6; ele se introduc în directiva DEFM:

Instrucțiunea BASIC	Cod
INK	16
PAPER	17
FLASH	18
BRIGHT	19
INVERSE	20
OVER	21
AT	22
TAB	23

De pildă, pentru a scrie textul anterior pe linia 11, coloana 3, cu **INK 1** și **PAPER 6**, directiva **DEFB** din exemplul 5.31 se va completa astfel

DEFB 22,11,3,16,1,17,6

Exemplul 5.32:

```

10          ORG adr          ;SCRIERE TEXT 32
                                CARACTERE (V.2)

20          ENT $

30          LD HL,TEXT

40          CALL PRINT

50  ZEND    RET

60  PRINT   LD A, (HL)

70          INC HL

80          OR A

90          RET Z

100         RST Z

110         JR PRINT

120  TEXT    DEFB 22,11,3,16,1,17,6 ;AT 11,3;
                                INK 1; PAPER 6

130         DEFM "M.M.POPOVICI SOFTWARE 1993"

140         DEFB 0

```

Programul BASIC de folosire a rutinei este următorul:

```
10 CLS : PRINT : RANDOMIZE USR adr
```

5.5.2. Scrierea textelor multiple

Pentru a scrie mai multe texte (linii cu maximum 32 caractere) cele doua rutine de scriere prezentate se modifică după cum urmează:

Prima rutină

```
PRNTXT      LD HL, TEXTE      ;adresa texte
```

```

LD B,numarul textelor (liniilor cu
max.32 caractere)
ET1      PUSH BC
LD B,(HL)      ;lungime text
INC HL        ;text propriu zis
CALL RUTPRN
POP BC
DJNZ ET1
ZEND     RET
RUTPRN  PUSH HL
        PUSH BC
        LD A,2
        CALL 5633
        POP BC
        POP HL
ET      LD A,(HL)
        RST 16
        INC HL
        DJNZ ET
        RET
TEXTE   DEFB LTXT1
TEXT1   DEFB parametri
        DEFM "text de afisat"
LTXT1   EQU $-TEXT1
TEXT2   DEFB parametri
        DEFM "text de afisat"
LTXT2   EQU S-TEXT2
...

```

Exemplul 5.33:

```

10      ORG adr      ;SCRIERE MULTI-TEXTE
                (v.1)
20      ENT $
30 PRINTXT LD HL,TEXTE
40      LD B,2      ;2-numarul textelor
50 ET1   PUSH BC
60      LD B,(HL)
70      INC HL
80      CALL RUTPRN
90      POP BC
100     DJNZ ET1
110 ZEND RET

```

```

120 RUTPRN    PUSH HL
130          PUSH BC
140          LD A,2
150          CALL 5633
160          POP BC
170          POP HL
180 ET       LD A, (HL)
190          RST 16
200          INC HL
210          DJNZ ET
220          RET
230 TEXTE    DEFB LTXT1
240 TEXT1    DEFB 22,11,3,16,1,17,6
250          DEFM "M.M.POPOVICI SOFTWARE 1993"
260 LTXT1    EQU $-TEXT1
270          DEFB LTXT2
280 TEXT2    DEFB 22,18,5,16,2,17,5
290          DEFM "PROGRAME IN COD MASINA"
300 LTXT2    EQU $-TEXT2

```

Se activează cu comanda **CLS : RANDOMIZE USR adr**

A doua rutină

```

PRNMT      LD HL,TEXTE
ET         CALL PRINT
          LD A, (HL)
          OR A
          RET Z
          JR RET
PRINT      LD A, (HL)
          INC HL
          OR A
          RET Z
          RST 16
          JR PRINT
TEXTE     DEFB parametri
          DEFM "text de afisat"
          DEFB 0           ;sfîrsit text 1
          DEFB parametri
          DEFB 0           ;sfîrsit text 2
          DEFB 0           ;sfîrsit texte

```


Exemplul 5.34:

```

10          ORG adr          ;SCRIERE MULTI-TEXTE
                                (v.2)
20          ENT $
30 PRINT   LD HL,TEXTE
40 ET      CALL PRINT
50         LD A,(HL)
60         OR A
70         RET Z
80         JR ET
90 PRINT   LD A,(HL)
100        INC HL
110        OR A
120        RET Z
130        RST 16
140        JR PRINT
150 TEXTE  DEFB 22,11,3,16,1,17,6
160        DEFM "M.M.POPOVICI SOFTWARE 1993"
170        DEFB 0
180        DEFB 22,18,5,16,2,17,5
190        DEFM "PROGRAME IN COD MASINA"
200        DEFB 0
210        DEFB 0

```

5.5.3. Scrierea cu aldine

Aldinele sînt caractere îngroșate, vizibile la o distanță mare de ecranul televizorului. Programul următor realizează asemenea caractere:

Exemplul 5.35:

```

10          ORG adr          ;CARACTERE ALDINE
20          ENT $
30          NOP
40          LD HL,15616      ;ADRESA CARACTERELOR
50          LD DE,64000
60          LD BC,768
70          LDIR
80          LD HL,64000
90          LD DE,768
100 L1      BIT 1,(HL)
110        JR Z,L2

```

```
120          SET 0, (HL)
130 L2       BIT 2, (HL)
140          JR Z, L3
150          SET 1, (HL)
160 L3       BIT 3, (HL)
170          JR Z, L4
180          SET 2, (HL)
190 L4       BIT 4, (HL)
200          JR Z, L5
210          SET 3, (HL)
220 L5       BIT 5, (HL)
230          JR Z, L6
240          SET 4, (HL)
250 L6       BIT 6, (HL)
260          JR Z, L7
270          SET 5, (HL)
280 L7       BIT 7, (HL)
290          JR Z, L8
300          SET 6, (HL)
310 L8       INC HL
320          DEC DE
330          LD A, D
340          OR E
350          CP 0
360          JR NZ, L1
370          LD A, 249
380          LD (23607), A
400 ZEND    RET
```

Rutina se lansează cu **RANDOMIZE USR adr**, după care scrierea se va face cu caractere albine. Se revine la caracterele normale cu **POKE 23607,60**.

6. TASTATURA ȘI AFIȘAJUL

Studiul structurii tastaturii și a ecranului conduce la sporuri de viteză și reprezintă baza unei tehnici de programe avansată.

6.1. TASTATURA

6.1.1. Analiza tastaturii

Tastele sînt racordate cu liniile magistralei de adrese A8...A15 (pe orizontală) și respectiv liniile magistralei de date D0...D4 (pe verticală).

a) Orizontal (5 taste)

CS,Z,X,C,V	(linia A8)
A,S,D,F,G	(linia A9)
Q,W,E,R,T	(linia A10)
1,2,3,4,5	(linia A11)
0,9,8,7,6	(linia A12)
P,O,I,U,Y	(linia A13)
ENTER,L,K,J,H	(linia A14)
SPACE,SS,M,N,B	(linia A15)

b) Vertical (8 taste)

CS,A,Q,1,0,P,	(linia D0)
ENTER,SPACE	
Z,S,W,2,9,0,L,SS	(linia D1)
X,D,E,3,8,I,K,M	(linia D2)
C,F,R,4,7,U,J,N	(linia D3)
V,G,T,5,6,Y,H,B	(linia D4)

Aceleași elemente sînt reunite în fig.6.1

	D0	D1	D2	D3	D4		
numărul portului de	254	CS	Z	X	C	V	A8
	253	A	S	D	F	G	A9
	251	Q	W	E	R	T	A10
	247	1	2	3	4	5	A11

intrare orizontal	239	0	9	8	7	6	A12
	223	P	O	I	U	Y	A13
	191	ENTER	L	K	J	H	A14
	127	SPACE	SS	M	N	B	A15

Fig.6.1

La apăsarea unei taste se realizează conectarea unei coloane cu o linie. iar octetul venit de la portul 254 conține 0 pe bitul corespunzător tastei apăsate. În mod normal coloanele magistralei de date sînt puse pe 1; dacă o linie este pe 0 și una din tastele liniei este apăsată coloana corespunzătoare va trece pe 0. Prin urmare, dacă toate liniile exceptînd una sînt puse pe 1, la magistrala de date se va citi starea tastelor de pe linia pusă pe 0, cu precizarea că o tastă este apăsată cînd bitul corespunzător pe coloană este 0.

Pentru înțelegere se presupune că se dorește să se tasteze dacă tasta 3 este apăsată. Pe liniile A8...A15 trebuie plasat următorul număr binar

A15	A14	A13	A12	A11	A10	A9	A8
1	1	1	1	0	1	1	1

↑ linia de
taste 1,2, 3, 4, 5

Se presupun următoarele două stări ale magistralei de date

D4	D3	D2	D1	D0	D4	D3	D2	D1	D0
0	0	1	1	0	1	0	0	1	1

Se constată că în a doua situație tasta 3 este apăsată.

6.1.2. Utilizarea rutinei de scanare a tastaturii

În ROM la adresa 654 se află o rutină de scanare (lectură) a tastelor care inițializează registrul BC cu 65278 și apoi apelează instrucțiunea **IN A,(C)**, punctul de plecare al unui ciclu cu mai multe instrucțiuni între care ultima este **RLC B**. Această instrucțiune de rotație la stînga aduce succesiv fiecare bit din registrul B pe 0 (la pornirea calculatorului B=254); prin urmare registrul B va avea ca valori succesive 254, 253, 251, 247, 239, 191 și 127 (v.fig.6.1). Instrucțiunea **IN A,(C)** plasează registrul BC pe magistrala de adrese, după care transferă în registrul

acumulator A conținutul a 8 porturi succesive. Cum registrul B conține octetul semnificativ, liniile de adrese A8, A9,...etc. vor fi pe 0. În mod normal liniile D0...D4 sînt puse pe 1, exceptînd cazul cînd se apasă o tastă.

Din cele precizate rezultă modul de a tasta dacă o tastă este apăsată:

- se selecționează unul din cele 8 porturi de intrare;
- se efectuează 1..5 rotații pentru a pune bitul respectiv în flagul C_i;
- se testează flagul C_i.

În cele ce urmează se prezintă trei exemple de testare:

a) Testul "*tasta SPACE este apăsată?*"; în caz afirmativ se comandă executarea unui program muzical MUZ

LD BC,32766 ;INT(32766/256) = 127 al 8-lea port orizontal

IN A,(C)

RRA ;pune bitul A0 în C_i

JP NC,MUZ ;salt la rutina MUZ dacă tasta SPACE este apăsată

continuare

b) Testul "*Tasta 8 este apăsată?*"; în caz afirmativ se execută rutina DEFDR

LD BC,61438 ;INT (61438/256) = 239 al 5-lea port orizontal

IN A,(C)

RRA ;pune bitul A2 în C_i

RRA ;pune bitul A1 în C_i

RRA ;pune bitul A0 în C_i

JP NZ,DEFDR ;salt la rutina DEFDR dacă tasta 8 este apăsată

continuare

c) Testul "*Tasta BREAK este apăsată?*"; în caz afirmativ rularea programului este oprită.

LD BC,32766 ;al 8-lea port orizontal

IN A,(C)

RRA

JR C,ET ;salt dacă BREAK nu este

	LD BC,65278	apăsată ;INT (65278/256) = 254 primul port orizontal
	IN A,(C) RRA JP NC,STOP	;salt la STOP dacă CS este apăsată
ET	continuare	;dacă nici BREAK și nici CS nu sînt apăsate

6.1.3. Utilizarea variabilei de sistem LAST-K

Variabila de sistem LAST-K (la adresa 23560, v.tab.3.1) conține codul ultimei taste apăsate. Bitul b5 al variabilei de sistem FLAGS (adresa 23611) se așează pe 1 dacă o nouă tastă este apăsată. Se menționează că microprocesorul Z80 execută de 50 de ori pe secundă o rutină care actualizează contorul de imagine, variabila de sistem FRAMES și cercetează tastatura actualizînd variabilele de sistem FLAGS și FLAST-K dacă este cazul (rutina nu se execută cînd se produc sunete). Prin urmare este suficient să se testeze variabilele de sistem pentru a se cunoaște tasta apăsată. Pentru exemplificare se reia într-o altă formă exemplul anterior de testare a tastei 8.

	SUB A	;pune A pe 0
	LD HL, 23611	;FLAGS
	BIT 5, (HL)	
	JR Z,ET	;salt dacă nici o tastă nu este apăsată
	LD A,(23560)	;(LAST-K)
	RES 5, (HL)	;repune pe 0 bitul b5 al lui FLAGS
	CP 56	;56 = codul cifrei 8
	JP Z,DEFDR	;salt la rutina DEFDR dacă tasta 8 este apăsată
ET	continuare	

Comparativ cu programul inițial acesta este mai lung dar prezintă avantajul că registrul A este pus pe 0, ceea ce permite să se facă *mai*

multe comparații și să se ia decizii adecvate. De pildă, dacă tasta 8 este apăsată se execută salt la rutina DEFDR care comandă o deplasare a ecranului la dreapta, iar dacă tasta 5 este apăsată se face salt la rutina DEFST care comandă o deplasare a ecranului la stînga. În acest scop, exemplul anterior se completează astfel:

CP 56	;56= codul tastei 8
JP Z, DEFDR	;salt la rutina DEFDR dacă tasta 8 este apăsată
CP 53	;53= codul tastei 5
JP Z,DEFST	;salt la rutina DEFST dacă tasta 5 este apăsată

6.1.4. Pauzele

După cum se știe, pauzele pot fi de două feluri: nelimitate și limitate.

a) Pauzele nelimitate sînt cele care așteaptă apăsarea unei taste pentru a se lua o decizie. Se examinează situațiile în care se obțin pauzele rezultate în BASIC cu instrucțiunile INPUT și INKEY\$.

- Echivalentul lui INPUT

Programul următor realizează o pauză prelungită pînă la apăsarea unei taste:

```
LD HL,23611          ;FLAGS
BIT 5, (HL)         ;Z= (HL) 5
ET1 JR Z, ET1
```

Acest ciclu se execută pînă se apasă o tastă; dacă se adaugă secvența

```
LD A,(23560)        ;(LAST-K)
RES 5,(HL)
```

se obține echivalentul instrucțiunii INPUT

- Echivalentul lui INKEY\$

Instrucțiunea INKEY\$ nu întrerupe derularea unui program BASIC dar se folosește de o tastă la un anumit moment. Prin urmare este suficient să se reia secvența de cod-mașină prezentată la folosirea variabilei de sistem LAST-K.

b) Pauzele limitate încetinesc sau variază derularea programului. Pentru a face pauze limitate se folosește timpul de execuție al

instrucțiunilor limbajului de asamblare. Știind că o perioadă de ceas T durează $0,2857 \mu\text{s}$ și cunoscând numărul de perioade pentru fiecare instrucțiune, pauzele se calculează cu precizie. Se reamintește că cea mai scurtă instrucțiune durează $4T$ adică $4 \cdot 0,2857 = 1,1428 \mu\text{s}$, iar cea mai lungă instrucțiune durează $23T$, adică $23 \cdot 0,2857 = 6,5711 \mu\text{s}$.

Pentru înțelegere se prezintă două exemplificări.

1) Durata unui ciclu simplu

```

LD B,32                ;7T
DJNZ ET                ;13 T când B≠0 sau 8T pentru
                        B = 0
:
ET    DEFB 0

```

Ciclul se execută de 32 ori; deci această pauză durează

$$0,2857 \cdot (7 + 31 \cdot 13 + 1 \cdot 8) = 0,2857 \cdot 418 = 119,4226 \mu\text{s}$$

Pentru a crește timpul se poate intercala o instrucțiune în ciclu

```

LD B,32                ;7T
ET1  LD IY,255         ;14T
      DJNZ ET1          ;13T pentru B≠0 sau 8T
                        pentru B = 0

```

În acest caz se execută de 32 ori 14T adică $32 \cdot 14 = 448T$ ceea ce face $448 \cdot 0,2857 = 127,9936 \mu\text{s}$ în plus.

2) Durata ciclurilor imbricate

```

LD HL,2047            ;10T
ET  DEC HL            ;6T
      LD A,H           ;4T
      OR L             ;4T
      JR NZ,ET         ;12T pentru Z=0 sau 7T
                        pentru Z = 1

```

Pauza durează: $10 + 26 \cdot 2047 + 21 = 53253T$ sau

$$53253 \cdot 0,2857 = 15214,382 \mu\text{s}$$

În acest program **DEC HL** nu influențează flagurile; instrucțiunea **OR L** face $Z = 1$ când $HL = 0$.

Exemplul 6.1: **BORDER** colorat (se folosesc pauze programate variabil, apeluri de subrutine și așteptarea tastării clapei **SPACE**).

```

10          ORG 60000          ;CULORI LATE PE
                        BORDER
20          ENT $

```



```

30 BORD      LD HL,212
40 ET1       OUT (254),A
50 ET2       DEC HL
60           LD A,H
70           OR L
80           JR NZ,ET2      ;PAUZA LINIILE
                                50..80

90           RET
100 ENT      HALT
110          LD A,0
120          LD HL,462
130          CALL ET1
140          LD B,6
150          LD C,0
160 ET3      INC C
170          LD A,C
180          CALL BORD
190          DJNZ ET3
200          INC C
210          LD A,C
220          OUT (254),A
230          LD BC,32766    ;al 8-lea port
                                orizontal
240          IN A,(C)
250          RRA            ;pune bitul A0 in Ci
260          JR C,ENT      ;SALT LA ENT DACA
                                TASTA SPACE ESTE
                                APASATA

270 ZEND     RET

```

Se trece în *BASIC* și se tastează **RANDOMIZE USR 60011**.

Exemplul 6.2.

```

10           ORG 60000      ;ATTRIBUTE CU SUNETE
20           ENT $
30           LD HL,768
40           LD DE,22528
50           LD BC,768
60           LDIR
70           LD B,248
80 ET0       LD HL,22528
90           PUSH BC
100          LD BC,768

```

```
110 ET1      LD D, (HL)
120          LD A, 0
130          CP D
140          JR Z, ET2
150          DEC D
160          LD (HL), D
170 ET2      INC HL
180          DEC BC
190          LD A, 0
200          CP B
210          JR NZ, ET1
220          CP C
230          JR NZ, ET1
240          LD HL, 128
250          LD DE, 1
260          CALL 949          ;RUTINA DE SUNETE
                                DIN ROM
270          LD HL, 176
280          LD DE, 1
290          CALL 949
300          LD HL, 240
310          LD DE, 2
320          CALL 949
330          POP BC
340          DJNZ ET0
350 ZEND     RET
```

Rutina realizează un efect grafic plăcut, motiv pentru care este folosită în diverse jocuri drept cortină între două ecrane. Coloritul este atractiv iar sunetul discret cu rol de atenționare.

6.2. AFIŞAJUL

Studiul memoriei ecran și al zonei de atribute a fost prezentat în capitolul 3. O sinteză a acestui studiu este redată în tabelul 6.1

Tabelul 6.1

Zona	Linia	Poziția PRINT	x = 16384	Poziția ATTR	y = 22528
Treimea superioară	0	16384	$x + 0 * 32$	22528	$y + 0 * 32$
	1	16416	$x + 1 * 32$	22560	$y + 1 * 32$
	⋮	⋮	⋮	⋮	⋮
	7	16608	$x + 7 * 32$	22752	$y + 7 * 32$
Treimea mijlocie	8	18432	$x + 8 * 32 + 1792$	22784	$y + 8 * 32$
	⋮	⋮	⋮	⋮	⋮
	15	18656	$x + 15 * 32 + 1792$	23008	$y + 15 * 32$
Treimea inferioară	16	204480	$x + 16 * 32 + 3584$	23040	$y + 16 * 32$
	⋮	⋮	⋮	⋮	⋮
	23	20704	$x + 23 * 32 + 3584$	23264	$y + 23 * 32$

6.2.1. Efecte cu atribute

Se prezintă rutina *PRATHL* care calculează adresa atributului unui caracter, cunoscând coordonatele *coloanei* (c) și *liniei* (l). Ea este echivalentul lui **PRINT AT I, c** unde coloana (c) se păstrează în registrul H și linia (l) în registrul L. Relația de calcul este

$$\text{adr} = \text{adb} + l + 32 * c \quad \text{unde adb} = 22528$$

```

PRATHL  PUSH HL
        PUSH DE
        LD A,H                ;în A este coloana c
        LD H,0                ;în HL este linia l
        ADD HL,HL              ;2c
        ADD HL,HL              ;4c
        ADD HL,HL              ;8c
        ADD HL,HL              ;16c
        ADD HL,HL              ;32c
        LD DE,22528           ;adb
        ADD HL,DE
        ADD A,L
        LD L,A
        LD A,0
        ADC A,H

```

```

LD H,A                ;HL=adb+ +32*c
LD (HL),40           ;8*PAPER ̵=40
POP DE
POP HL
RET

```

Cu ajutorul acestei rutine se pot elabora programe interesante; pentru exemplificare se prezintă următoarele două programe.

Exemplul 6.3: se realizează o cortină stînga-dreapta de culoare PAPER dorită (de exemplu PAPER 5) așa cum se indică în rutina PRATHL.

```

10                ORG 60000                ;CORTINA STINGA-
                                     DREAPTA PAPER
20                ENT $
30                LD DE,31
40 ET1            LD L,0
50 ET2            LD H,D
60                CALL PRATHL
70                LD H,E
80                CALL PRATHL
90                INC L
100               CP 22
110               LD A,L
120               JR NZ,ET2
130               HALT
140               INC D
150               DEC E
160               LD A,E
170               CP 15
180               JR NZ,ET1
190 ZEND          RET
200 PRATHL        PUSH HL
210               PUSH DE
220               LD A,H
230               LD H,0
240               ADD HL,HL
250               ADD HL,HL
260               ADD HL,HL
270               ADD HL,HL
280               ADD HL,HL
290               LD DE,22528

```

```

300          ADD HL,DE
310          ADD A,L
320          LD L,A
330          LD A,0
340          ADC A,H
350          LD H,A
360          LD (HL),40
370          POP DE
380          POP HL
390          RET

```

Rutina se activează cu comanda RANDOMIZE USR 60000. Cu mici modificări se pot realiza cortine stînga-dreapta pe unele zone ale ecranului. Astfel:

```

- numai pe treimea superioară:  110   CP 7
- pe primele două zone :        110   CP 15
- pe treimea inferioară :       110   CP 7
(v.tab.6.1)                      290   LD DE,23040

```

Exemplul 6.4: se realizează un efect vizual în care un dreptunghi de culoare albă, inițial de dimensiunea ecranului, se micșorează treptat pînă dispăre într-un *PAPER* de culoare bleu (*PAPER 5*).

```

 10          ORG 60000          ;SPIRALA
 20          ENT $
 30          LD HL,0
 40          LD BC,31
 50          LD DE,23
 60 ET1      CALL PRATHL
 70          INC H
 80          LD A,H
 90          CP C
100         JR NZ,ET1
110         INC D
120 ET2      CALL PRATHL
130         INC L
140         LD A,L
150         CP E
160         JR NZ,ET2
170         DEC C
180 ET3      CALL PRATHL
190         DEC H
200         LD A,H

```

```
210          CP B
220          JR NZ,ET3
230          DEC E
240 TEST     LD A,D
250          DEC A
260          SUB E
270          JR Z,END
280 ET 4     CALL PRATHL
290          DEC L
300          LD A,L
310          CP D
320          JR NZ,ET4
330          INC B
340          JR ET1
350 END      LD HL,2828
360 PRATHL   PUSH HL
370          PUSH BC
380 OPER     LD BC,17
390 ETO      DEC BC
400          LD A,C
410          AND 16
420          OR 5
430          OUT (254),A
440          LD A,B
450          OR C
460          JR NZ,ETO
470          POP BC
480          PUSH DE
490          LD A,H
500          LD H,0
510          ADD H1,HL
520          ADD HL,HL
530          ADD HL,HL
540          ADD HL,HL
550          ADD HL,HL
560          LD DE,22528
570          ADD HL,DE
580          ADD A,L
590          LD L,A
610          LD A,0
620          ADC A,H
```

```

630          LD A,H
640          LD (HL),5*8      ;PAPER 5
650          POP DE
660          POP HL
670          RET

```

- Următoarele două rutine realizează așa numitul efect "burete" prin care se obține ștergerea ecranului.

Exemplul 6.5: burete orizontal linie cu linie care schimbă culoarea PAPER-ului.

```

10          ORG 60000          ;BURETE ORIZONTAL
20          ENT $
30  ET1     PUSH AF
40          PUSH DE
50          PUSH HL
60          PUSH BC
70          LD A,96
80  ET2     DEC A
90          JR NZ,ET2
100         LD HL,15
110        LD DE,20
120        CALL 949
130        POP BC
140        POP HL
150        POP DE
160        POP AF
170        RET
180  ET3     PUSH BC
190        LD B,4
200  ET4     CALL ET1
210        DJNZ ET4
220        POP BC
230        RET
240        LD HL,22527
250        LD A,9
260        LD B,8
270  ET 5    PUSH BC
280        LD B,33
290  ET 6    INC HL
300        LD (HL),A
310        CALL ET3

```

```

320          DJNZ ET6
330          LD BC,32
340          ADD HL,BC
350          LD B,33
360 ET7      DEC HL
370          LD (HL),A
380          CALL ET3
390          DJNZ ET7
400          LD BC,32
410          ADD HL,BC
420          POP BC
430          DJNZ ET5
440 ZEND    RET

```

Exemplul 6.6: buretele care înconjoară marginea ecranului descriind dreptunghiuri descrescătoare ca dimensiuni (spirală)

```

10          ORG 60000          ;SPIRALA
20          ENT $
30          LD HL,22527
40          LD C,21
50          LD B,32
60 L6528    LD D,B
70 L6529    INC HL
80          LD (HL),A
90          CALL L6563
100         DEC D
110        JR NZ,L6529
120        DEC B
130        LD E,C
140        PUSH BC
150        LD BC,32
160 L6537   ADD HL,BC
170        LD (HL),A
180        CALL L6563
190        DEC E
200        JR NZ,L6537
210        POP BC
220        DEC C
230        LD D,B
240 L6542   DEC HL
250        LD (HL),A
260        CALL L6563

```



```

270          DEC D
280          JR NZ ,L6542
290          LD E ,A
300          LD A ,B
310          CP 11
320          RET Z
330          LD A ,E
340          DEC B
350          LD E ,C
360          PUSH BC
370          LD BC ,32
380 L6556     SBC HL ,BC
390          LD (HL) ,A
400          CALL L6563
410          DEC E
420          JR NZ ,L6556
430          POP BC
440          DEC C
450          JR L6528
460 L6563     PUSH AF
470          PUSH DE
480          PUSH HL
490          PUSH BC
500          LD A ,96
510 L6569     DEC A
520          JR NZ ,L6569
530          LD HL ,15
540          LD DE ,20
550          CALL 949          ;RUTINA DE SUNETE
                                DIN ROM

560          POP BC
570          POP HL
580          POP DE
590          POP AF
600 ZEND     RET

```

- În încheierea paragrafului consacrat efectelor vizuale se redă o rutină care simulează "o explozie", utilă în programele de divertisment la atingerea unei ținte, sau ca o cortină între ecrane. Rutina realizează un **BORDER** colorat, modificarea **INK**-ului, un sunet discret și o "iluminare" a ecranului care sugerează explozia.

Exemplul 6.7:

```

10          ORG 60000          ;SIMULARE "EXPLOZIE"
20          ENT $
30  ET1     LD HL,22528
40          LD D,0
50          LD BC,768
60  ET2     LD A,(HL)
70          CP 0
80          JR Z,ET3
90          LD D,255
100         DEC (HL)
110        LD A,16
120        OUT (254),A
130        XOR A
140        OUT (254),A
150  ET3     DEC BC
160        INC HL
170        LD A,B
180        OR C
190        JR NZ,ET2
200        CP D
210        JR NZ,ET1
220        LD HL,16384
230        LD DE,16385
240        LD (HL),0
250        LD BC,6144
260  ZEND   RET

```

6.2.2. Efecte de scriere

Pentru a da programelor un plus de atractivitate se apelează la scriere cu aldine în mod *normal* sau pe *verticală*, *scriere rotită cu 90°* sau *180°* și respectiv *scriere roll-uită*.

6.2.2.1. Scriere cu aldine (în mod normal sau pe verticală)

Subrutina care urmează calculează adresa unui caracter oarecare folosind variabila de sistem LAST-K (de la adresa 23560), în care se memorează codul ultimei taste apășate:

```

CHRADR LD A,(23560)                ;(LAST-K)
      SUB 32
      LD L, A
      ADD HL, HL
      ADD HL, HL
      ADD HL, HL
      LD BC, (23606)                ;(CHANS)
      INC B
      ADD HL, BC
      RET

```

Caracterele aldine (îngroșate) se obțin cu următorul program:

Exemplul 6.8:

```

10          ORG 60000                ;ALDINE
20          ENT $
30          CALL CHRADR
40          LD IX, (23675)           ; (UDG)
50          LD B, 8
60  ET1     LD A, (HL)
70          RRA
80          OR (HL)
90          LD (IX), A
100         INC IX
110        INC HL
120        DJNZ ET1
130  ZEND   RET
140  CHRADR LD A, (23560)            ; (LAST-K)
150        SUB 32
160        LD L, A
170        ADD HL, HL
180        ADD HL, HL
190        ADD HL, HL
200        LD BC, (23606)           ; (CHANS)
210        INC B
220        ADD HL, BC
230        RET

```

Pentru a se scrie cu aldine fie pe orizontală fie pe verticală se utilizează următoarele programe BASIC:

a) Scriere pe orizontală

```
10 CLS : LET a$="32 caractere"
```

```

20 FOR i=1 TO LEN a$
30 POKE 23560, CODE a$(i)
40 RANDOMIZE USR 60000
50 PRINT AT linie, i-1; "%"; CHR$ 8; "A": REM litera
  "A" în modul grafic
60 NEXT i

```

b) Scriere pe verticală

```

10 CLS : LET a$="22 caractere"
20 FOR i=1 TO LEN a$
30 POKE 23560, CODE a$(i)
40 RANDOMIZE USR 60000
50 PRINT AT i-1, coloana; "%"; CHR$ 8; "A": REM litera
  "A" în modul grafic
60 NEXT i

```

6.2.2.2. Scriere cu litere rotite (cu 90° sau 180°)

Există situații în care literele trebuie rotite după dorință (de exemplu diagramele la care pe axa ordonatelor trebuie înscris un text). Programul următor realizează un nou set de caractere rotite cu 90°.

Exemplul 6.9:

```

10          ORG adr          ; LITERE ROTITE CU 90
                                GRADE
20          ENT $
30          LD HL, 0
40          CALL CHRADR
50          LD C, 8
60 ET1      LD B, 8
70          LD IX, (23675)   ; (UDG)
80          LD A, (HL)
90 ET2      RRA
100         RL (IX)
110        INC IX
120        DJNZ ET2
130        JNC HL
140        DEC C
150        JR NZ, ET1
160 ZEND    RET
170 CHRADR  LD A, (23560)    ; (LAST-K)
180        SUB 32

```

```

190          LD L, A
200          ADD HL,HL
210          ADD HL,HL
220          ADD HL,HL
230          LD BC, (23606) ; (CHANS)
240          INC B
250          ADD HL,BC
260          RET

```

Programul BASIC aferent celor două modalități de scriere are forma:

a) Scriere rotită cu 90°

```

10 CLS : LET a$="32 caractere"
20 FOR i=1 TO LEN a$
30 POKE 23560,CODE a$(i)
40 RANDOMIZE USR adr .
50 PRINT AT linie, i-1;"A": REM litera "A" în modul
   grafic
60 NEXT i

```

b) Scriere rotită cu 180°

```

10 CLS : LET a$="22 caractere"
20 FOR i=1 TO LEN a$
30 POKE 23560,CODE a$(i)
40 RANDOMIZE USR adr
50 PRINT AT (i-linie),coloana;"A": REM litera "A"
   în modul grafic
60 NEXT i

```

6.2.2.3. Scriere roll

Rutina care urmează realizează efectul roll pentru un text de maximum 32 caractere, pe linia 32 a ecranului.

Exemplul 6.10:

```

10          ORG 60000          ;SRIERE ROLL PE
                               LINIA 23
20          ENT $
30          LD HL,32000
40          LD B,8
50 ET1      LD (HL),0
60          IND HL
70          DJNZ ET1
80          LD HL,32000

```

```
90          LD DE,20704      ;ADRESA LINIEI 23
100         LD B,8
110 ET2     LD A,(DE)
120         RES 0,A
130         RES 1,A
140         RES 2,A
150         RES 3,A
160         RES 4,A
170         RES 5,A
180         RES 6,A
190         LD (HL),A
200         LD A,1
210         ADD A,D
220         LD D,A
230         INC HL
240         DJNZ ET2
250         LD A,0
260         LD HL,20735     ;ADRESA SFIRSITULUI
                           LINIEI 23
270         LD C,8
280 ET3     PUSH HL
290         LD B,32
300         OR A
310 ET4     RL (HL)
320         DEC HL
330         DJNZ ET4
340         LD DE,256
350         POP HL
360         ADD HL,DE
370         DEC C
380         CP C
390         JR NZ,ET3
400         LD HL,32000
410         LD DE,20735
420         LD B,8
430 ET5     LD C,(HL)
440         LD A,128
450         CP C
460         JR NZ,ET6
470         LD A,(DE)
480         SET 0,A
```

```
490          LD (DE),A
500 ET 6     LD A,1
510          ADD A,D
520          LD D,A
530          INC HL
540          DJNZ ET5
550 ZEND     RET
```

Rutina se valorifică prin următorul program BASIC:

```
10 CLS : PRINT # 0;" PROGRAMAT DE M.M. POPOVICI
    1993"
20 RANDOMIZE USR 60000: IF INKEY$ ="" THEN GO TO 20
    Este posibil ca efectul roll să se realizeze și pe linia 24, caz în care
programul BASIC anterior se modifică după cum urmează:
10 CLS : PRINT # 0;"PROGRAMAT DE M.M. POPOVICI
    1993": REM text stationar
15 PRINT # 1; "    < PROGRAME IN COD MASINA >    "
20 RANDOMIZE USR 60000: IF INKEY$ ="" THEN GO TO 20
```

7. NOȚIUNI DESPRE ANIMAȚIE ȘI ÎNTRERUPERI

În acest capitol se prezintă noțiunile de bază ale animației și se dezvoltă un program de divertisment (joc). De asemenea se explică fundamentele întreruperilor.

7.1. ELEMENTELE ANIMAȚIEI : HAZARDUL ȘI DEPLASAREA

Se prezintă două modalități prin care se realizează mai rapid numere aleatoare în cod mașină decât o face instrucțiunea RND din BASIC.

a) Utilizarea variabilei de sistem nefolosite 23681

Principiul este următorul: se extrage din ROM un octet în care se poate depune un număr $n=0\dots255$.

Exemplul 7.1:

10	ORG 60000	; GENERAREA NUMERELOR ALEATOARE
20	ENT \$	
30	LD HL,23681	
40	INC (HL)	
50	LD L, (HL)	
60	LD H,25	; 6400=256*25 OCTETUL SEMNICATIV
70	LD C, (HL)	
80	LD B,0	
90	RET	

Programul *BASIC* are forma:

```
10 CLS
20 LET a=USR 60000: PRINT a
30 GO TO 20
```

Se vor afișa diverse numere aleatoare.

b) Utilizarea variabilei de sistem SEED (23670)

Principiul este următorul: se alege un număr care se depune în variabila de sistem SEED aflată la adresa 23670. Se înmulțește apoi acest număr cu un număr prim, iar rezultatul - de cele mai multe ori modulo 65536 - este stocat în variabila de sistem SEED pentru o folosire ulterioară.

Exemplul 7.2:

```
10          ORG 60000          ;NUMERE ALEATOARE
                (v.2)
20          ENT $
30          LD HL,(23670)    ;(SEED)
40          LD D,H
50          LD E,L
60          ADD HL,HL
70          ADD HL,DE
80          ADD HL,HL
90          ADD HL,DE
100         ADD HL,HĹ
110        ADD HL,HL
120        ADD HL,DE
130        INC L
140        LD (23670),HL
150        LD B,H
160        LD C,L
170        RET
```

Programul *BASIC* de folosință este identic.

7.2. RUTINELE AFIȘĂRII

Cu următorul program:

a) în BASIC

```
10 CLS : FOR n=16384 TO 22527: POKE n,255: NEXT n
```

b) în limbaj de asamblare

```

10          ORG 60000
20          ENT $
30          LD HL,16384
40          LD BC,3*2048
50  E1      LDV (HL),255
60          LD DE,10000      ;bucla temporizare
70  E2      DEC DE
80          LD A,D
90          OR E
100         JP NZ,E2
110        INC HL
120        DEC BC
130        LD A,B
140        OR C
150        JP NZ,E1
160        RET

```

(RANDOMIZE USR 60000)

se vizualizează organizarea ecranului. Afişajul pe ecran este divizat în 3 zone distincte a câte 8 linii fiecare conform schemei de mai jos

linia 0	16384
linia 7	18431
linia 8	18432
linia 15	20479
linia 16	20480
linia 23	22527

Observînd succesiunea de umplere a ecranului se constată că cele trei zone se umplu identic. De pildă, la prima zonă se umple rîndul 1 al primei linii, apoi rîndul 1 al liniei a 2-a, apoi rîndul 1 al liniei a 3-1 ş.a.m.d. pînă la 16389. Continuă apoi cu rîndul 2 al liniei 1, apoi rîndul 2 al liniei 2 etc, totul continuînd în aceeaşi manieră pînă la 18431. Evident, pentru zona a doua (de la 18432 la 20479) şi zona a treia (de la 20480 la 22527) se procedează identic.

Rezultă că sînt necesare unele rutine care să calculeze poziţiile de afişare. În acest scop se introduc următoarele notaţii:

OCTET pentru a indica octetul într-un rând de pixeli
PRINT pentru a indica poziția unui caracter (8 poziții OCTET)
PLOT pentru a indica un pixel într-o poziție OCTET

7.2.1. Poziția PRINT

Exemplul 7.3:

```

10          ORG 42200          ;POZITIA PRINT
20  POZPR   LD D,0
30          AND A              ;Ci =0
40          LD A,H
50          LD B,5
60  ET1     RLA
70          RL D
80          DJNZ ET1
90          ADD A,L
100         LD E,A
110        PUSH BC
120        POP BC
130        LD A,B
140        AND A,88           ;ADAUS 22528=256*88
150        LD B,A
160        BIT 0,D           ;Z=D0
170        JR NZ,ET2        ;SALT DACA Z=1
180        LD A,D
190        ADD A,7           ;SE ADAUGA
                               1792=256*7
200        LD D,A
210  ET2    BIT 1,D           ;Z=D1
220        JR Z,ET3         ;SALT DACA Z=1
230        LD A,D           ;DACA D1=1,Z=0
240        ADD A,14         ;SE ADAUGA
                               3584=256*14
250        LD D,A
260  ET3    LD A,D
270        ADD A,64         ;SE ADAUGA=256*64
280        LD D,A
290        RET

```

La sfârșitul acestei rutine registrul dublu BC este punctat pe poziția

ATTR și registrul dublu **DE** pe poziția **PRINT**. Examinînd programul se observă că nu conține vreun test (de pildă dacă **H** este cuprins între 0 și 23, iar **L** cuprins între 0 și 31).

7.2.2. Poziția octet

Întrucît toate aceste rutine sînt necesare pentru a se realiza elemente de animație, se recomandă ca ele să fie scrise în continuare celei precedente.

Exemplul 7.4:

```

300          ORG 42240          ;POZITIA OCTET
310  POZOC   LD A,H
320          AND 248           ;H modulo 8 si Ci=0
330          RRA
340          RRA
350          RRA
360          PUSH HL
370          LD H,A
380          CALL POZPR        ;SALT LA ADRESA
                               42400
390          POP HL
400          LD A,H
410          AND 7             ;IZOLEAZA NR.
                               RINDULUI (0..7)
420          ADD A,D
430          LD D,A

```

Registrul **DE** este punctat pe poziția **OCTET**

7.2.3. PLOT

Exemplul 7.5:

```

450          ORG 42258          ;PLOT
460  POSPL   LD A,L
470          AND 248           ;L modulo 8 si Ci=0
480          RRA
490          RRA
500          RRA
510          PUSH HL

```

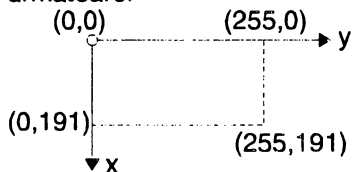
```

520          LD L,A
530          CALL POZOC      ;SALT LA ADRESA
                               42240

540          POP HL
550          LD A,L
560          AND 7
570          RET

```

La revenire registrul DE este punctat pe poziția OCTET, registrul A pe pixelul din această poziție, iar BC pe poziția ATTR. Se menționează că s-a folosit un nou sistem de axe de coordonate conform schemei următoare:



unde $x \in [0;255]$ și $y \in [0;191]$.

7.2.4. Adresa matricei de caractere

Exemplul 7.6.:

```

500          ORG 42274      ;MATRICEA
                               CARACTERELOR

590 MATCAR    BIT 7,A      ;Z=A7
600          JR Z,ET4      ;SALT DACA A<128
610          LD B,0        ;DACA A>127
620          LD C,A
630          LD HL,2011
640          ADD HL,BC     ;2011+COD UDG
650          JR ET5
660 ET4      LD H,0
670          LD L,A
680 ET5      ADD HL,HL     ;MULTIPLICAT CU 8
690          ADD HL,HL
700          ADD HL,HL
710          LD A,H
720          ADD A,188     ;SE ADAUGA
                               48128=256*128

730          LD H,A
740          RET

```

La reîntoarcerea din rutină registrul dublu HL este punctat pe primul octet al matricei de caractere, al cărui cod este în registrul A. În același timp, pentru a se putea afișa caracterele grafice definite de programator rutina începe cu un test. Astfel, dacă $A < 128$ se pune A în HL, se multiplică cu 8 și se adaugă 48128; dacă $A > 127$ (de ex.144 care este codul primului caracter grafic), atunci se pune acest număr (ex.144) în BC, se adaugă 2011, se multiplică cu 8 și se adaugă 48128. Rezultă astfel $2155 * 8 + 48128 = 65368$ (adresa primului caracter UDG)

Se atenționează că rutina folosește registrul BC și dacă acest registru servește programului propriu zis, conținutul lui trebuie salvat în stivă înainte de apelul rutinei și apoi restaurat.

7.2.5. Comanda PLOT

Exemplul 7.7:

```

750          ORG 42298          ;AFISAREA POZITIEI
                                PLOT

760 AFPL     CALL POZPL
770          CPL
780          AND 7
790          RLA
800          RLA
810          RLA
820          OR 199             ;PENTRU A FACE SET
830          LD HL,23300       ;ADRESA CARE CONTINE
                                UN OCTET DE MATRICE

840          BIT 7, (HL)       ;TESTUL BITULUI 7
850          JR NZ,ET6
860          SUB 64             ;PENTRU A FACE RES
870 ET6      LD (ET7+1),A
880          LD A, (DE)
890 ET7      SET 0,A
900          LD (DE),A
910          RET

```

La linia 890 se pot realiza 16 posibilități folosind SET 0 la 7 sau RES 0 la 7.

7.2.6. Mișcarea

Exemplul 7.8:

```

920          ORG 42328          ;MISCAREA
930          ENT $
940  STO     DEFW 42326
950  MISC    LD H,7             ;NR.LINIE 0
960         LD L,30            ;NR.COLOANA
970         LD C,170
980  L1     INC H
990         LD A,8
1000        ADD A,H
1010        LD H,A
1020        LD DE,42607        ;UDG+7
1030        LD (STO),DE
1040        LD B,8
1050  L0     PUSH BC
1060        CALL POZOC
1070        NOP
1080        LD BC,(STO)
1090        LD A,(BC)
1100        LD (DE),A
1110        DEC BC
1120        LD (STO),BC
1130        DEC H
1140        POP BC
1150        DJNZ L0
1160        NOP
1170        DEC C
1180        JR NZ,L1
1190        RET

```

Personajul care se mișcă este desenat în fig.7.1; datele numerice pentru definirea acestui nou caracter grafic sînt introduse cu directiva DEFB.

```

1200        ORG 42600          ;DEFINIREA UDG
1210        DEFB 60,126,219,255,189,165,165,36
1220        DEFB 0

```


știut că astfel de programe sînt elaborate de echipe de specialiști alcătuite din programatori, desenatori, muzicieni, plasticieni, scenariști și alte categorii de profesioniști ai genului care, printr-o muncă migăloasă, reușesc să realizeze cu inteligență și gust, o lume specifică și apropiată desenului animat.

Fără îndoială că o astfel de performanță de programare nu este la îndemîna oricui; din acest motiv există firme specializate în asemenea produse informatice, care își păstrează "secretele" printr-o protejare a programelor cu mijloace din ce în ce mai sofisticate.

În cele ce urmează se prezintă un program de divertisment foarte simplu, în care personajul principal (un om) trebuie să traverseze mai întîi o stradă foarte circulată în dublu sens de către vehicule rutiere diverse și pietoni, apoi o bandă deplasabilă cu păianjeni uriași și o zonă fluvială populată cu crocodili pe care navighează vapoare de diferite tonaje. În final se ajunge într-un spațiu marcat cu grilaje.

Pentru a elabora un asemenea joc trebuie rezolvate, în principal, trei probleme diferite și anume:

- modul în care trebuie deplasate anumite linii ale ecranului în sensuri inverse și cu viteze diferite, pe care urmează să se amplaseze vehiculele rutiere, pietonii, păianjenii, crocodilii și vehiculele navale;

- desenarea personajului central al jocului și a restului elementelor grafice;

- afișarea scorului și modul în care se punctează fiecare reușită a personajului central în tentativa sa de traversare.

a) Deplasarea diverselor linii ale ecranului se face conform modelelor de scroll prezentate în cap.4. O analiză a liniilor ecranului a condus la schema de mai jos.

<u>Sens deplasare</u> →	la stînga	la dreapta
Tip deplasare ↓		
lentă	Liniile 2,6,16,18	Linia 12
rapidă	Linia 10	Liniile 4,8,10,14

Subrutina care realizează aceste deplasări este următoarea:

Exemplul 7.10:

10

ORG 42244

;DEPLASARI LINII
DIFERITE

20

ENT \$

30	ET1	LD C,B
40	ET2	PUSH HL
50		LD DE,31
60		ADD HL,DE
70		LD A,(HL)
80		SBC HL,DE
90		RRA
100		LD B,32
110	ET3	LD A,(HL)
120		RRA
130		LD (HL),A
140		INC HL
150		DJNZ ET3
160		POP HL
170		INC H
180		DEC C
190		JR NZ,ET2
200		RET
210	ET4	LD C,8
220	ET5	XOR A
230		PUSH HL
240		LD DE,31
250		SBC HL,DE
260		LD A,(HL)
270		ADD HL,DE
280		RLA
290		LD B,32
300	ET6	LD A,(HL)
310		RLA
320		LD (HL),A
330		DEC HL
340		DJNZ ET6
350		POP HL
360		INC H
370		DEC C
380		JR NZ,ET5
390		RET
400		LD HL,16749
410		CALL ET4
420		LD HL,16512
430		CALL ET1

440		LD HL,16512
450		CALL ET1
460		LD HL,16607
470		CALL ET4
480		LD HL,18432
490		CALL ET1
500		LD HL,18432
510		CALL ET1
520		LD HL,18432
530		CALL ET1
540		LD A, (23673)
550		NOP
560		NOP
570		NOP
580		NOP
590		NOP
600		NOP
610		NOP
620		AND 2
630		JR Z,ET7
640		LD HL,18496
650		CALL ET1
660		LD HL,18496
670		CALL ET1
680		LD HL,18496
690		CALL ET1
700		JR ET8
710	ET7	LD HL,18527
720		CALL ET4
730		LD HL,18527
740		CALL ET4
750		LD HL,18527
760		CALL ET4
770	ET8	LD HL,18560
780		CALL ET1
790		LD HL,18624
800		CALL ET1
810		LD HL,18624
820		CALL ET1
830		LD HL,20511
840		CALL ET4

```

850          LD HL,20511
860          CALL ET4
870          LD HL,20575
880          CALL ET4
890          RET
900          LD HL,18560
910          CALL ET1
920          LD HL,18624
930          CALL ET1
940          LD HL,20511
950          CALL ET4
960          LD HL,20575
970          CALL ET4
980          RET
990          LD HL,16479
1000         CALL ET4
1010         LD HL,165512
1020         CALL ET1
1030         LD HL,18432
1040         CALL ET1
1050         RET
1060         LD HL,16479
1070         CALL ET4
1080         LD HL,16607
1090         CALL ET4
1100         LD HL,18560
1110         CALL ET1
1120         LD HL,18624
1130  ZEND   CALL ET1

```

Verificarea funcționării rutinei se poate face cu următorul program:

```

5 BORDER 5: PAPER 5: INK 1: CLS
10 FOR n=0 TO 21: PRINT AT n,0;"Linia";n: NEXT n
15 FOR i=0 TO 255: RANDOMIZE USR 42295: NEXT i

```

Rutina poate fi plasată la orice adresă adr, iar activarea ei se face cu comanda

RANDOMIZE USR (adr+51).

Această rutină se va salva pe bandă cu comanda

SAVE "trav" CODE adr, 250

b) Desenarea personajului central și a elementelor grafice (vehicule rutiere de diferite forme, pietoni, păianjen, valuri, crocodil, vapoare de

forme diferite, grilaj) se face după modelul cunoscut din BASIC de definire a noilor UDG. Astfel:

- pentru mersul spre dreapta: automobil (AB) și camion (CDE);
- pentru mersul spre stânga: automobil (HI) și camion (JKL);
- pieton (F), personajul central (G), val (M), păianjen (OP), grilaj (N), vapor (QRRS) și crocodil (TU).

Programul BASIC de definire a UDG-urilor este următorul:

```

10 CLS : LET a=PEEK 23675 +256*PEEK 23676: RESTORE
   30
20 FOR b=a TO a+167: READ c: POKE b,c: NEXT b
30 DATA 15,18,34,127,255,255,40,16,128,64,32,254,
   254,255,40,16
40 DATA 127,127,127,127,127,255,21,8,254,254,254,
   254,255,255,64,128
50 DATA 0,248,196,196,254,254,40,16,24,24,36,126,
   60,90,165,66
60 DATA 56,40,146,124,56,56,40,108,1,2,4,127,127,
   255,20,8
70 DATA 240,72,68,254,255,255,20,8,0,31,35,35,127,
   127,20,8
80 DATA 127,127,127,127,255,255,2,1,254,254,254,
   254,254,255,168,16
90 DATA 16,41,199,0,38,0,0,0,0,66,255,68,68,255,68,
   0
100 DATA 0,34,85,143,151,163,160,0,0,68,170,241,233,
   197,5,0
110 DATA 16,16,16,254,63,31,15,7,0,0,0,0,30,255,255,
   255
120 DATA 96,124,84,120,127,255,254,252,0,0,3,2,15,
   16,255,0,6,12,152,240,224,85,255,0

```

Programul se verifică tastând în modul grafic literele următoare:

AB CDE F G HI JKLM N OP QRRS TU

și se saleară cu comanda

SAVE "UDG" CODE 65368,168

c) Afisarea scorului și modul de joc sînt scrise în programul BASIC, care urmează (în care majusculele subliniate se tastează în modul grafic)

```

10 CLS : LOAD "trav" CODE adr,250: LOAD "UDG" CODE
   65368,168
20 GO TO 40

```

```

25 BEEP .01,b-a
30 PRINT OVER 1; PAPER 8; INK 8; AT a,y2;"G":
  RETURN
40 CLS : PRINT AT 11,3;"Doriti instructiuni (d/n)?"
50 PAUSE 0: IF INKEY$ ="d" OR INKEY$ ="D" THEN GO
  TO 300
60 GO TO 400
300 CLS : PRINT AT 0,11;"OBIECTIVUL" ' ' "Conduci un
  G evitînd AB CDE F QRRS TU. " ' ' "Un OP
  patruleaza în fata in-sulei."
310 PRINT ' ' "Trebuie sa intri în cele 4 ca- sute
  necompletate din partea de sus NNN NNN".
320 PRINT ' "Cînd cele 4 casute sînt comple- tate,
  viteza va creste, apare un OP si casutele se
  golesc."
330 PRINT ' ' ' AT 18,4;"Apasati o tasta oarecare":
  BEEP .02,40: PAUSE 4: BEEP .02,-40: PAUSE 0
340 CLS : PRINT AT 7,22;"TASTE:"
350 PRINT ' "(7sp)↑(14sp)< >": REM (7sp)=7
  blancuri; (14sp)=14 blancuri
360 PRINT FLASH 1; AT 11,6; "1"; FLASH 0;" 2 3 4 5 6
  7 8 "; FLASH 1;"9"; FLASH 0;" "; FLASH 1;"0"
370PRINT AT 18,4;"Apasati o tasta oarecare":
  BEEP.02,40: BEEP.02,-40: PAUSE 0
400 BRIGHT 1: PAPER 5: BORDER 5: CLS
410 LET hi=0
420 PRINT PAPER 4; AT 10,0;"(15 sp)OP(14 sp)"
430 LET lives=9: LET score=0: LET home=0
440 POKE (adr+181),201: POKE (adr+206),201: POKE
  (adr+225),201
450 PRINT AT 0,0; PAPER 4;"NNNN"; PAPER 7;" "; PAPER
  4; "NNNNNN"; PAPER 7;" "; PAPER 4; "NNNNNNNN";
  PAPER 7; " "; PAPER 4; "NNNNNNNN"; PAPER 7;" ";
  PAPER 4;"NNNN"
455 IF home <> 0 THEN GO TO 660
460 PRINT PAPER 4; INK 5;"32▣": REM tasta 3 în
  modul grafic (de 32 ori)
470 PRINT " QRRRS QRRRS QRRS QRRS "
480 PRINT INK 7;"(4sp)MMM(5sp)MMMM(5sp)MMM(4sp)M"
490 PRINT INK 2;"(2sp)TU(4sp)TU(5sp)TU(3sp)
TU(6sp)TU"

```

```

500 PRINT INK 7;"MM(2sp)M(6sp)MMMM(8sp)MMM(4sp) "
510 PRINT INK 1;"RS(5sp)QRRS(5sp)QRRRRS(6sp)QRRR"
520 PRINT INK 7;"(2sp)MM(2sp)MMMM(5sp)MM(6sp)MMM
    (6sp) "
530 PRINT "U(8sp)TU(6sp)TU(6sp)TU(4sp) "
540 PRINT PAPER 4;"32N": REM 32 litere N
550 PRINT PAPER 0; INK 7; AT 11,0;"32N": REM 32
    litere N
560 PRINT PAPER; INK 3:" AB F AB CDCDE F E
    CDE(6sp) "
570 PRINT PAPER 0; INK 7;"32-": REM 32 minusuri
580 PRINT PAPER 0; INK 5;" AB(4sp)AB(4sp)AB(2SP)
    AB(7sp)AB(4sp) "
590 PRINT PAPER 0; INK 7;"32=": REM 32 egaluri
600 PRINT PAPER 0; INK 4: "(15sp)HI(3sp)HI(7sp)HI
    (9sp)H"
610 PRINT PAPER 0: INK 7;"32-": REM 32 minusuri
620 PRINT PAPER 0; INK 6;"KL(3sp)HI(2sp)HI F(2sp)
    F(2sp)JKLKL(3sp)F(4sp)J"
630 PRINT PAPER 4;"32N": REM 32 litere N
640 PRINT PAPER 4;"32sp"
650 PRINT PAPER 1; INK 7;"SCOR:"; AT 21,11;" INC ";
    PAPER 5; INK 0;lives; PAPER 1; INK 7;"SCOR MAX"
660 LET x1=20: LET y1=16: LET x2=x1: LET y2=y1.
670 PRINT PAPER 8; INK 8; AT x1,y1;" "
680 RANDOMIZE USR (adr+51)
690 IF SCREENS (x2,y2)=" " THEN GO TO 880
700 LET a=x2: FOR b=25 TO 35: GOSUB 25: GOSUB 25:
    GOSUB 25: NEXT b
730 FOR a=x2 TO 20 STEP 2: GOSUB 25: NEXT a
740 LET lives=lives-1: PRINT AT 21,16;lives
750 LET x2=20
760 IF lives <> 0 TEHN GO TO 680
770 LET hi=score; PRINT AT 21,27;hi
790 PRINT FLASH 1; PAPER 7; AT 12,7;"SFIRSITUL
    JOCULUI"
800 PRINT AT 14,14;"ALT JOC (d/n)?"
840 IF INKEY$ ="n" OR INKEY$ ="N" THEN CLS: STOP
850 IF INKEY$ <> "d" THEN GO TO 840
860 PRINT PAPER 5; AT 21,7;"4sp": GO TO 415
880 IF x2 <> 0 THEN GO TO 1050

```

```
890 PRINT PAPER 8; INK 8; AT x1,y1;" "; AT x2,y2;
    "G"
900 RESTORE 920
910 FOR a=1 TO 8: READ b,c: BEEP b,c: NEXT a
920 DATA .1,11,.1,11,.8,16,.05,11,.05,16,.05,11,
    .05,16,1,20
930 LET home=home+1: LET score=score+50: PRINT AT
    21,7;score
940 IF home/4 <> INT (home/4) THEN GO TO 660
960 IF home=4 THEN POKE (adr+181),0
970 IF home=8 THEN POKE (adr+206),0
980 IF home=12 THEN POKE (adr+225),0
985 IF home>36 THEN GO TO 450
990 LET a=RND*31
1000 LET a=a+1
1005 IF a>31 THEN LET a=0
1010 IF SCREEN$(10,a)="" THEN GO TO 1000
1020 IF SCREEN$(10,a+1)="" THEN GO TO 1000
1030 PRINT PAPER 4; AT 10,a,"OP"
1035 RESTORE 920: FOR a=1 TO 8: READ b,c: BEEP b,c:
    NEXT a
1040 GO TO 450
1050 PRINT PAPER 8; INK 8; AT x2,y2;"G"
1060 LET x1=x2: LET y1=y2
1070 IF INKEYS <> "1" THEN GO TO 1100
1080 BEEP .001,33
1090 LET x2=x2-2: LET score=score+5: PRINT AT 21,7;
    score
1100 LET y2=y2+(INKEY$ ="0" AND y2 <> 31)-(INKEY$
    ="9" AND y2 <> 0)
1110 GO TO 670
```

7.4. ÎNTRERUPERILE

• Întreruperile sînt modalități subtile de programare care conduc la efecte vizuale spectaculoase ce sînt folosite în diverse categorii de programe cu precădere în cele de divertisment (jocuri). Mecanismul de funcționare al întreruperilor se explică în cele ce urmează.

Se consideră microprocesorul într-o stare inițială în care execută programul principal; la un moment dat sosește dela un periferic (de ex. tastatura) un semnal exterior care precizează că trebuie efectuată o întrerupere și furnizează, în același timp, originea semnalului de întrerupere. Prima operație pe care trebuie să o execute microprocesorul este să salveze în stivă datele care îi permit să reia programul principal după tratarea întreruperii, respectiv să salveze registrele și adresa de memorie de la care se va relua execuția. Din acest moment microprocesorul poate trece la tratarea întreruperii, adică la executarea unui subprogram ce are o adresă de început pe care microprocesorul o cunoaște datorită originii întreruperii.

După satisfacerea cererii de întrerupere se revine la programul principal, ceea ce implică reîncărcarea registrelor cu valorile salvate în stivă și continuarea execuției din locul specificat prin adresa de revenire.

Întreruperile sînt de două feluri: nemascabile (NMI) și mascabile (MI). La întreruperile nemascabile microprocesorul Z80 răspunde într-un singur mod, în timp ce pentru întreruperile mascabile există trei moduri de tratare (**IM 0**, **IM 1**, **IM 2**). Întreruperea nemascabilă este prioritară față de cea mascabilă.

- Întreruperile nemascabile nu pot fi ignorate; ele execută automat un **RST** sau **CALL** la adresa 102 din ROM; cu instrucțiunea **RETN** (revenire din întreruperea nemascabilă) microprocesorul revine la instrucțiunea următoare din programul principal. Acest tip de întrerupere este rezervat evenimentelor grave (cădere de tensiune, eroare de memorie) și este inaccesibil programatorului.

- Întreruperile mascabile sînt validate/invalidare de instrucțiunile **EI**, respectiv **DI**. Microprocesorul Z80 dispune de trei moduri de întreruperi mascabile și anume:

- a) În modul IM 0 se intră automat la inițializarea sistemului sau folosind instrucțiunea **IM 0**; în acest caz se execută una din cele 8 instrucțiuni **RST** de care dispune microprocesorul și a cărei adresă este înscrisă pe magistrala de date. Acest mod nu poate fi folosit de programator deoarece în configurația hard a calculatoarelor compatibile cu ZX-SPECTRUM nu există un circuit periferic care să genereze un cod de instrucțiune RST sau CALL.

- b) În modul IM 1, în care se intră cu instrucțiunea **IM 1**,

microprocesorul va executa programul din ROM de la adresa 56. Acest mod este folosit de interpretorul BASIC pentru scanarea (citirea) tastaturii și deci nu este util programatorului (la fiecare 1/50 secunde se lansează o cerere de întrerupere în urma căreia se execută un RST 56).

c) **Modul IM 2** este modul de întrerupere la dispoziția programatorului, adresa subprogramului de tratare a întreruperii găsindu-se într-o locație de memorie. Pentru calculul acestei locații registru l asigură octetul superior (semnificativ) de adresă, iar octetul inferior (mai puțin semnificativ) este asigurat de dispozitivul periferic care a cerut întreruperea. De la această locație de memorie se citesc în ordine doi octeți cu care se formează o nouă adresă la care se va executa apoi un **CALL**. Pentru înțelegerea modului în care se formează această nouă adresă se presupune că x este valoarea conținută în registrul I; pentru această valoare adresa care se formează va lua valori limită între

$$256*x \text{ și } 256*x+255,$$

respectiv 256 valori distincte, indiferent de octetul inferior care i se furnizează microprocesorului de către dispozitivul periferic ce a solicitat întreruperea. Dacă toți octeții de memorie între aceste două adrese limită vor conține aceeași valoare y, microprocesorul va citi din această tabelă de valori doi octeți identici aflați în succesiune și va crea cu ei noua adresă

$$256*y+y$$

indiferent de unde se citesc octeții.

La această adresă **adr1 = 256*y + y** se plasează subrutina care să deservească întreruperea dorită de programator.

- Subrutina de tratare a întreruperii trebuie să înceapă cu instrucțiunea **DI** (dezactivarea întreruperii) și să se termine cu **EI** (activarea întreruperii) și **RET**. De asemenea subrutina trebuie să conțină mai întâi o secvență de salvare în stivă a tuturor registrelor care vor putea fi alterate și în final o secvență de restaurare a registrelor pentru revenirea în programul principal.

Dacă se dorește ca întreruperea să funcționeze în paralel cu un program BASIC aflat sub interpretor, se va utiliza și instrucțiunea **CALL 703** (salt la adresa din ROM unde este plasată rutina de citire a tastaturii).

Rezumând, structura unei rutine pentru întreruperi va avea următoare formă:

INIT	ORG adr	;inițializarea întreruperii
	ENT \$	
	LD HL,adr2	;adr2 = 256*(y-1)-adresa tabelei de valori
	LD A,y	;y = octetul semnificativ
	LD B,0	;tabela începe la adr2 și conține numărul y
TABELA	LD (HL),A	
	INC HL	
	DJNZ TABELA	
	LD (HL),A	
	LD A,y-1	
	LD I,A	
	DI	
	IM 2	
	RET	
SUBRUT	ORG adr1	;adr1 = 256*y + y
	Eventual DI	
	PUSH AF	
	PUSH BC	
	PUSH DE	
	PUSH HL	
	Eventual o secvență de temporizare	
	CALL RUTINA	;apelarea rutinei de executat
	CALL 703	;apelarea rutinei de citire a tastelor
	POP HL	;restaurarea registrelor
	POP DE	
	POP AF	
	JP 56	;salt la adresa din ROM pentru tratarea întreruperilor mascabile
RUTINA	...	;rutina de executat în urma cererii de întrerupere

Activarea rutinei se face cu comanda **RANDOMIZE USR adr.**

Secvența de temporizare poate avea forma:

TEMP LD HL, 500
 WAIT DEC HL
 LD A,H
 OR L
 JR NZ,WAIT

Exemplul 7.11:

10	INIT	ORG 61697	;INITIALIZARE INTRERUPERI
20		ENT \$	
30		LD HL,61440	;adresa tabelii de valori
40		LD A,241	;Y=241
50		LD B,0	;tabela incepe la 62440 si contine numarul 241
60	TABELA	LD (HL),A	
70		INC HL	
80		DJNZ TABELA	
90		LD (HL),A	
100		LD A,240	;240=y-1
110		LD I,A	
120		DI	
130		IM 2	
140		EI	
150		RET	
160	SUBRUT	ORG 61937	;61937=256*241+241
170		DI	
180		PUSH AF	
190		PUSH BC	
200		PUSH DE	
210		PUSH HL	
220		CALL RUTINA	
230		CALL 703	
240		POP HL	
250		POP DE	
260		POP BC	
270		POP AF	
280		JP 56	
290	RUTINA	LD B,32	;rutina de executat
300		LD HL,23264	

```

310 ET      LD A,R
320        LD (HL),A
330        INC HL
340        DJNZ ET
350 ZEND    RET

```

Programul BASIC de folosire a rutinei:

```

10 BORDER 2: PAPER 0: INK 7: CLS
20 RANDOMIZE USR 61697
30 PRINT # 1; AT 1,4;"APASATI O TASTA OARECARE":
   PAUSE 0

```

Se obține linia 23 colorată și acest efect nu poate dispărea decât prin resetarea calculatorului.

Exemplul 7.12: deplasarea spre stînga ecranului a unui text cu max.768 caractere în timpul execuției unui program BASIC.

```

10          ORG 63993          ;DEPLASARE TEXT
20          ENT $
30 ET1      EQU $+3
40 ET2      EQU $+4
50 ET3      EQU $+7
60          JR ET3
70 LF9FB    LD BC,5377
80 LF9FE    LD L,(HL)
90          CALL M,50677
100         PUSH DE
110         PUSH HL
120         PUSH IX
130         LD A,(LF9FB)
140         AND A
150         JR NZ,LFA1F
160         LD BC,64530
170         LD (LF9FE),BC
180         LD (ET2),A
190 LFA16    POP IX
200         POP HL
210         POP DE
220         POP BC
230         POP AF
240         JP 56
250 LFA1F    LD A,(ET1)
260         AND A

```

```
270          JP NZ,LFA63
280          LA A,8
290          LD (ET1),A
300          LD DE,(LF9FE)
310          LD A,(DE)
320          INC DE
330          CP 13
340          JR NZ,LFA39
350          LD DE,64530
360          LD A,(DE)
370 LFA39    LD (LF9FE),DE
380          CP 32
390          JR C,LFA16
400          LD DE,(23606)
410          BIT 7,A
420          JR Z,LFA54
430          RES 7,A
440          SUB 16
450          JP C,LFA16
460          LD DE,(23675)
470 LFA54    LD L,A
480          LD H,0
490          ADD HL,HL
500          ADD HL,HL
510          ADD HL,HL
520          ADD HL,DE
530          LD DE,64520
540          LD BC,8
550          LDIR
560 LFA63    LD A,(ET2)
570          CP 24
580          JP NC,LFA16
590          AND 8
600          OR 64
610          LD H,A
620          LD A,(ET2)
630          OR 248
640          RRCA
650          RRCA
660          RRCA
670          LD L,A
```

680		LD IX,64520	
690		LD C,8	
700	LFA7F	PUSH HL	
710		LD B,32	
720		RL (IX+0)	
730	LFA86	RL (HL)	
740		DEC HL	
750		DJNZ LFA86	
760		POP HL	
770		INC H	
780		INC IX	
790		DEC C	
800		JP NZ,LFA7F	
810		LD A,(ET1)	
820		DEC A	
830		LD (ET1),A	
840		JP LFA16	
850		NOP	;între liniile 850 la 1070
1070		NOP	
1080		DI	
1090		LD HL,64256	
1100		LD DE,64257	
1110		LD (HL),249	
1120		LD BC,258	
1130		LDIR	
1140		LD A,251	
1150		LD I,A	
1160		IM2	
1170		XOR A	
1180		LD (ET1),A	
1190		LD DE,64530	
1200		LD (LF9FE),DE	
1210		LD HL,(23637)	
1220		LD BC,5	
1230		ADD HL,BC	
1240		LD BC,768	
1250	LFADD	LD A,(HL)	
1260		CP 13	
1270		JR Z,LFAEA	
1280		INC HL	

1290		INC DE
1300		DEC BC
1310		LD A,B
1320		OR C
1330		JR NZ,LFADD
1340	LFAEA	EI
1350	ZEND	RET

Programul *BASIC* de exploatare a rutinei va conține textul dorit a se traduce într-o instrucțiune **REM**; cu **POKE 63997,1** se activează traducerea iar cu **POKE 63995,0** se oprește. Numărul liniei pe care se dorește defilarea textului se obține cu **POKE 93997,linie** unde *linie*=[0;23].

```
10 CLS : RANDOMIZE USR 64180
20 REM Programul realizeaza o traducere spre
   stînga a unui sir de caractere în timp ce se
   ruleaza un program în BASIC
30 POKE 63995,1
40 POKE 63997,21
50 LIST : PAUSE 0
```


8. 50 RUTINE PENTRU PERFECTIONAREA PROGRAMELOR PROPRII

În acest capitol se prezintă 50 de rutine care perfecționează programele proprii (în cod-mașină sau în *BASIC*), aducând un spor de atractivitate. Din motive didactice ele au fost grupate în patru categorii distincte:

- sunete;
- efecte vizuale;
- efecte audio-vizuale;
- modalități de scriere.

Aceste rutine pot fi folosite în programele proprii după cum urmează:

- sunetele pentru sublinierea unei ide, a unui eveniment sau cu rol de atenționare;
- efectele vizuale și cele audio-vizuale cu rol principal de cortină între ecrane sau cu scop de a crea efecte surpriză;
- modalitățile de scriere pentru titluri, mărirea dimensiunilor caracterelor, scriere tip șenilă cu sunet etc.

8.1. SUNETE

Exemplul 8.1.:

10	ORG adr	; SUNET "CLAXON" (v.1)
20	ENT \$	
30	LD DE,15362	
40 ET1	LD H,9	
50	LD A,(23624)	

```

60          RRA
70          RRA
80          RRA
90  ET2     LD C,254
100         XOR 16
110        OUT (C),A
120        LD B,E
130  ET3     DJNZ ET3
140         DEC H
150         JR NZ,ET2
160         DEC E
170         DEC D
180         JR NZ,ET1
190  ZEND    RET

```

Activarea rutinei se face cu comanda **RANDOMIZE USR** adr.

Exemplul 8.2.:

```

10          ORG adr          ;SUNET "CLAXON" (v.2)
20          ENT $
30          LD B,30
40  ET1     PUSH BC
50          LD HL,256
60  ET2     LD DE,1
70          PUSH HL
80          CALL 949        ;949-RUTINA DE
                           SUNETE DIN ROM

90          POP HL
100         LD DE,16
110         AND A
120         SBC HL,DE
130         JR NZ,ET2
140         POP BC
150         DJNZ ET1
160  ZEND    RET

(RANDOMIZE USR adr)

```

Exemplul 8.3:

```

10          ORG adr          ;SUNET "TELEFON"
                           (CÎRÎT)
20          ENT $
30          LD BC,127
40          LD HL,170

```

```

50          LD DE,1
60          PUSH DE
70          PUSH BC
80          PUSH HL
90          CALL 949
100         POP HL
110        POP BC
120        POP DE
130        INC BC
140        INC HL
150        INC HL
160        INC HL
170        LD BC,65044
180        LD BC,60960
190 ZEND    RET

```

Programul BASIC de folosire a rutinei este următorul:

```
10 FOR i=1 TO 100: RANDOMIZE USR adr: NEXT i
```

Exemplul 8.4:

```

10          ORG adr          ;SUNET ZGOMOTOS
20          ENT $
30          LD DE,1
40          LD BC,0
50          LD H,0
60 ET       LD A,(BC)
70          LD L,A
80          PUSH HL
90          PUSH DE
100         PUSH BC
110        CALL 949
120        POP BC
130        POP DE
140        POP HL
150        INC BC
160        LD A,B
170        CP 7
180        JR NZ,ET
190 ZEND    RET

```

(RANDOMIZE USR adr)

Un astfel de efect sonor este folosit adesea în jocurile de succes.

Exemplul 8.5:

```

10          ORG adr          ;SUNET SUITOR-
                                COBORITOR
20          ENT $
30          LD BC,350
40          LD HL,400
50 ET 1     LD DE,3
60          PUSH HL
70          PUSH BC
80          CALL 949
90          POP BC
100         POP HL
110        INC HL
120        DEC BC
130        LD A,B
140        CP 0
150        JR NZ,ET1
160        LD BC,350
170 ET2     LD DE,3
180        PUSH BC
190        PUSH HL
200        CALL 949
210        POP HL
220        POP BC
230        DEC BC
240        DEC HL
250        LD A,B
260        CP 0
270        JR NZ,ET2
280 ZEND    RET

```

(RANDOMIZE USR adr)

Din această rutină poate fi exploatată numai o parte și anume liniile 10-150 cu următoarele introduceri:

```

55          PUSH DE
105         POP DE
155         RET

```

Programul *BASIC* are forma:

```
10 FOR t=1 TO 3: RANDOMIZE USR adr: NEXT t
```

- Urmează un număr de 4 rutine sonore cu o structură similară, diferențele fiind formate de valorile numerice introduce în registrele B,BC și DE; se obțin însă sunete diferite.

Exemplul 8.6:

```

10          ORG adr          SUNET "SIRENA"
20          ENT $
30          LD B,10
40 ET1     PUSH BC
50          LD HL,0
60 ET2     LD DE,1
70          PUSH HL
80          CALL 949
90          LD BC,1
100         LD DE,356
110         POP HL
120         ADD A,0
130         ADC HL,BC
140         PUSH HL
150         ADD A,0
160         SBC HL,DE
170         POP HL
180         JR C,ET2
190         POP BC
200         DJNZ ET1
210 ZEND    RET

```

Modificările menționate sînt indicate în tabelul următor:

	LINIILE MODIFICATE	DENUMIREA SUNETULUI
a)	30 LD B,5 90 LD BC,512 100 LD DE,4864	Sunete "misterioase"
b)	30 LD B,14 90 LD BC,25 100 LD DE,240	Sunet "Păsărele"
c)	30 LD B,1 90 LD BC,1 100 LD DE,356	Sunet "Glonte"

Rutina care urmează realizează 3 sunete diverse.

Exemplul 8.7:

```

10          ORG adr          ;TREI SUNETE
20          ENT $
30          PUSH BC
40          PUSH DE
50 ET1     LD B,E

```

```

60 ET2      DJNZ ET2
70          LD A(BC)
80          SET 0,A
90          SET 1,A
100         SET 2,A
110         OUT (254),A
120         INC C
130         DEC D
140         JR NZ,ET1
150         POP DE
160         POP BC
170 ZEND    RET

```

Programul BASIC de exploatare:

```

10 FOR x=1 TO 10: RANDOMIZE USR adr: NEXT x: PAUSE
   S
20 FOR y=1 TO 10: RANDOMIZE USR (adr+86): NEXT y:
   PAUSE 0
30 RANDOMIZE USR (adr+98)

```

Exemplul 8 8:

```

10          ORG adr                ;SUNET "SONERIE"
20          ENT $
30          PUSH BC
40          PUSH DE
50          XOR A
60          SET 0,A
70          SET 1,A
80          SET 2,A
90 ET1      LD B,E
100 ET2     DJNZ ET2
110         SET 4,A
120         OUT (254),A
130         LD B,E
140 ET3     DJNZ ET3
150         RES 4,A
160         OUT (254),A
170         DEC D
180         JR NZ,ET1
190         POP DE
200         POP BC
210 ZEND    RET

```

Exemplul 8.9:

```

10          ORG adr          ;SUNET COMPLEX
20          ENT $
30          LD HL,23728
40          LD (HL),254
50          INC HL
60          LD (HL),0
70          LD B,127
80  ET1     LD DE,25
90          LD H,1
100         LD A,(23728)
110         LD L,A
120         PUSH BC
130         CALL 949
140         POP BC
150         LD H,1
160         LD DE,5
170         LD A,(23729)
180         LD L,A
190         PUSH BC
200         CALL 949
210         POP BC
220         LD HL,23728
230         DEC (HL)
240         DEC (HL)
250         INC HL
260         INC (HL)
270         INC (HL)
280         DJNZ ET1
290  ZEND   RET

```

Exemplul 8.10:

```

10          ORG adr          ;9 SUNETE
20          ENT $
30  ET1     EQU $+23
40  ET2     EQU $+29
50          LD H,2
60  ET3     DEC H
70          NOP
80          NOP
90          NOP
100         JP NZ,ET4

```

```
110          RET
120 ET4      LD C,1
130 ET5      CALL ET8
140          LD A,C
150          LD (ET1),A
160          LD (ET2),A
170          LD B,99
180 ET6      LD A,7
190          OUT (254),A
200          LD A,99
210 ET7      DEC A
220          CP 0
230          JP NZ,ET7
240          LD A,23
250          OUT (254),A
260          DEC B
270          JP NZ,ET6
280          INC C
290          LD A,C
300          CP 100
310          JR NZ,ET5
320          JP ET3
330 ET8      LD D,20
340 ET9      DEC D
350          NOP
360          NOP
370          JP Z,ET11
380          LD E,1
390 ET10     DEC E
400          NOP
410          NOP
420          JP NZ,ET10
430          JP ET9
440 ET11     RET
```

Această rutină de 74 octeți are următorul program *BASIC* de folosire:

```
24 CLS : LET a=adr: POKE (a+37),23: REM adr=adresa
    de start a rutinei în cod masina
29 GO TO 100
31 POKE (a+1),VAL A$(1 TO 3)
32 POKE (a+55),VAL A$(4 TO 6)
33 POKE (a+11),VAL A$(7 TO 9)
```



```
34 POKE (a+47),VAL A$. (10 TO 12)
35 RETURN
36 GOSUB 31
38 LET M=USR a: RETURN
40 BORDER 0: PAPER 0: INK 7: CLS : PRINT AT 1,3;
  INVERSE 1;"SUNETE DIFERITE"; INVERSE 0; PRINT
  "1)Ipuscaturi"
100 LET A$="002020001100": GOSUB 36
108 LET A$="002010001100": GOSUB 36
110 PRINT "'2)Traiect în aer"
112 LET A$="020001075080": GOSUB 36
115 PRINT "'3)Mitraliera"
116 LET A$="040040001030": GOSUB 36
120 LET A$="040100001020": GOSUB 36
122 PRINT "'4)Hohote de rîs"
200 LET A$="002200001020": GOSUB 31
202 FOR I=2 TO 50: POKE (a+7),I: LET M=USR a: NEXT
  I: PAUSE 10
206 PRINT "'5)Armonii"
207 LET A$="050010001020": GOSUB 31
208 FOR I=20 TO 18 STEP -1: POKE (a+47),I: LET M=USR
  a: NEXT I: PAUSE 10
213 PRINT "'6)Decolare"
250 LET A$="006010001060": GOSUB 31
254 FOR I=50 TO 20 STEP -1: POKE (a+47),I: LET M=USR
  a: NEXT I: PAUSE 10
260 PRINT "'7)Revenire la sol"
300 LET A$="002010001040": GOSUB 31
302 FOR I=2 TO 60: POKE (a+55),I: LET M=USR a: NEXT
  I: PAUSE 10
310 PRINT "'8)Avion doborît"
500 LET A$="002010001020": GOSUB 31
502 FOR I=2 TO 255: POKE (a+55),I: LET M=USR a: NEXT
  I: PAUSE 10
510 PRINT "'9)Redresarea avionului"
512 LET A$="002010001020": GOSUB 31
515 FOR I=255 TO 2 STEP -1: POKE (a+55),I: LET M=USR
  a: NEXT I
```

8.2. EFECTE VIZUALE

Exemplul 8.11:

```

10          ORG adr          ;BENZI COLORATE
                                VERTICALE
100         ENT $
30          LD BC,320
40          RST 56
50          RLA
60          LD (HL),L
70          LDIR
80          LD HL,22528
90          LD BC,736
100 ET      LD A,L
110         AND 28
120         RLCA
130         XOR 56
140         LD (HL),A
150         INC HL
160         DEC BC
170         LD A,B
180         OR C
190         JR NZ,ET
200 ZEND    RET

```

(RANDOMIZE USR adr)

Exemplul 8.12:

```

10          ORG adr          ;CORTINA LATERALA
                                DIAGONALA
100         ENT $
30          LD DE,31
40          LD HL,22528
50          LD C,32
60          PUSH HL
70 ET1     LD (HL),0
80          ADD HL,DE
90          LD A,H
100         LD B,255
110 WAIT    DJNZ WAIT

```

```

120          CP 91
130          JR Z,ET2
140          JR NC,ET2
150          JR ET1
160 ET2      POP HL
170          INC HL
180          DEC C
190          LD A,C
200          CP 0
210          RET Z
220          PUSH HL
230          JR ET1

```

(RANDOMIZE USR adr)

• Se prezintă 14 rutine care realizează efecte cu ajutorul unui SCREEN încărcat la adresa 53000. Programul BASIC ce exploatează aceste rutine are forma următoare:

```

10 LOAD "mira" CODE 53000,6912: REM adresa
    obligatorie de incarcare a SCREEN-ului cu numele
    "mira", salvat la adresa 16384,6912
20 RESTORE 40: READ b$,x,y
30 RANDOMIZE USR x
40 DATA "nume",x,y: REM "nume" este numele
    rutinei,x=adresa de lansare a rutinei si
    y=lungimea rutinei în octeti

```

Exemplul 8.13:

```

10          ORG 60000          ;nume="PRINT" ;
                                x=60000; y=12
20          ENT $
30          LD DE,16384
40          LD HL,53000
50          LD BC,6912
60          LDIR
70 ZEND      RET

```

Deci programul BASIC va fi:

```

10 LOAD "mira" CODE 53000,6912
20 RESTORE 40: READ b$,x,y
30 TANDOMIZE USR x
40 DATA "PRINT",60000,12

```

Exemplul 8.14:

```

10          ORG 60025          ;nume="FADE" ;

```

X=60025 ; y=53

```

20          ENT $
30          LD HL,16384
40          LD DE,7
50          LD B,E
60  ET1     PUSH HL
70          PUSH BC
80  ET2     LD A,H
90          CP 88
100         JR NC,ET3
110         PUSH HL
120        RES 6,H
130        PUSH DE
140        LD DE,53000 ;ADRESA INCARCARE
                                SCREEN
150        NOP
160        ADD HL,DE
170        PUSH HL
180        POP IX
190        POP DE
200        POP HL
210        LD A,(IX+0)
220        LD (HL),A
230        ADD HL,DE
240        JR ET2
250  ET3     POP BC
260        POP HL
270        INC HL
280        HALT
290        DJNZ ET1
300        LD DE,16384
310        LD HL,53000
320        LD BC,6912
330        LDIR
340  ZEND    RET

```

În programul BASIC general linia 40 va fi: **40 DATA "FADE",60025,53**

Exemplul 8.15:

```

10          ORG 60078          ;nume="SHUTTER";
                                x=60078; y=51
20          ENT $
30          LD HL,53000

```

```

40          LD DE,6144
50          ADD HL,DE
60          LD DE,22528
70          LD BC,768
80          LDIR
90          LD D,128
100         LD B,8
110  ET1    PUSH BC
120         LD HL,53000
130         LD IX,16384
140         LD BC,6144
150  ET2    LD A,(HL)
160         AND D
170         LD (IXD+0),A
180         INC HL
190         INC IX
200         DEC BC
210         LD A,B
220         OR C
230         JR NZ,ET2
240         POP BC
250         RR D
260         SET 7,D
270         DJNZ ET1
280  ZEND   RET

```

(40 DATA "SHUTTER",60078,51)

Exemplul 8.16:

```

10          ORG 60129          ;nume="SLIDE-DOWN";
                                x=60129; y=75
10          ENI $
30          LD HL,53000
40          LD DE,6144
50          ADD HL,DE
60          LD DE,22528
70          LD BC,768
80          LDIR
90          LD HL,53000
100         LD B,192
110  ET1    PUSH BC
120         LD A,192
130         SUB A

```

```
140          LD H,A
150          LD L,0
160          LD A,H
170          AND 192
180          RRCA
190          RRCA
200          RRCA
210          ADD A,64
220          LD D,A
230          LD A,H
240          AND 7
250          ADD A,D
260          LD D,A
270          LD A,H
280          ADD A,A
290          ADD A,A
300          AND 224
310          LD E,A
320          LD A,L
330          AND 248
340          RRCA
350          RRCA
360          RRCA
370          OR E
380          LD E,A
390          PUSH DE
400          LD HL,53000
410          LD A,D
420          SUB 64
430          LD D,A
440          ADD HL,DE
450          POP DE
460          LD BC,32
470          LDIR
480          POP BC
490          HALT
500          DJNZ ET
510 ZEND     RET
```

(40 DATA "SLIDE-DOWN",60129,75)

Exemplul 8.17:

10

ORG 60204

;name="SLIDE-UP"

;x=60204; y=70

```
20          ENT $
30          LD HL,53000
40          LD DE,6144
50          ADD HL,DE
60          LD DE,22528
70          LD BC,768
80          LDIR
90          LD B,193
100 ET     PUSH BC
110        DEC B
120        LD H,B
130        LD L,0
140        LD A,H
150        AND 192
160        RRCA
170        RRCA
180        RRCA
190        ADD A,64
200        LD D,A
210        LD A,H
220        AND 7
230        ADD H,D
240        LD D,A
250        LD A,H
260        ADD A,A
270        ADD A,A
280        AND 224
290        LD E,A
300        LD A,L
310        AND 248
320        RRCA
330        RRCA
340        RRCA
350        OR E
360        LD E,A
370        PUSH DE
380        LD HL,53000
390        LD A,D
400        SUB 64
410        LD A,D
```

```

420          ADD HL,DE
430          POP DE
440          LD BC,32
450          LDIR
460          POP BC
470          HALT
480          DJNZ ET
490 ZEND     RET

```

(40 DATA "SLIDE-UP",60204,70)

Exemplul 8.18:

```

10          ORG 60274          ;nume="SLIDE-LEFT";
                                x=60274; y=67
20          ENT $
30          LD HL,53000
40          LD DE,6144
50          ADD HL,DE
60          LD DE,22528
70          LD BC,768
80          LDIR
90          LD IX,53000
100         LD DE,31
110        ADD IX,DE
120        LD HL,16415
130        INC DE
140        LD BC,8193
150 ET1    PUSH BC
160        LD B,8
170 ET2    PUSH BC
180        PUSH HL
190        PUSH IX
200        LD B,192
210 ET3    LD A,(IX+0)
220        AND C
230        LD (HL),A
240        ADD HL,DE
250        ADD IX,DE
260        DJNZ ET3
270        POP IX
280        POP HL
290        POP BC
300        RL C

```



```

310          INC C
320          HALT
330          DJNZ ET2
340          POP BC
350          DEC HL
360          DEC IX
370          DJNZ ET1
380 ZEND     RET

```

(40 DATA "SLIDE-LEFT",60274,67)

Exemplul 8.19:

```

10          ·   ORG 60341          ;name="SLIDE-RIGHT";
                                   x=60341;y=65
20          ENT $
30          LD HL,53000
40          LD DE,6144
50          ADD HL,DE
60          LD DE,22528
70          LD BC,768
80          LDIR
90          LD IX,53000
100         LD HL,16384
110        LD DE,32
120        LD BC,8320
130 ET1     PUSH BC
140        LD B,8
150 ET2     PUSH BC
160        PUSH HL
170        PUSH IX
180        LD B,192
190 ET3     LD A,(IX+0)
200        AND C
210        LD (HL),A
220        ADD HL,DE
230        ADD IX,DE
240        DJNZ ET3
250        POP IX
260        POP HL
270        POP BC
280        RR C
290        SET 7,C
300        HALT

```

```

310          DJNZ ET2
320          POP BC
330          INC HL
340          INC IX
350          DJNZ ET1
360 ZEND     RET

```

(40 DATA "SLIDE-RIGHT", 60341,65)

Exemplul 8.20:

```

10          ORG 60406          ;nume="ATTR-DOWN" ;
                                x=60406; y=24
20          ENT $
30          LD HL,53000
40          LD DE,16384
50          LD BC,6144
60          LDIR
70          LD B,24
80 ET      PUSH BC
90          LD BC,32
100         LDIR
110        POP BC
120        HALT
130        DJNZ ET
140 ZEND   RET

```

(40 DATA "ATTR-DOWN",60406,24)

Exemplul: 8.21:

```

10          ORG 60430          ;nume="ATTR-UP" ;
                                x=60430; y=35
20          ENT $
30          LD HL,53000
40          LD DE,16384
50          LD BC,6144
60          LDIR
70          LD BC,768
80          ADD HL,BC
90          PUSH HL
100         LD H,D
110        LD L,E
120        ADD HL,BC
130        LD D,H
140        LD E,L

```

```

150          POP HL
160          LD B,24
170 ET      PUSH BC
180          LD BC,32
190          LDDR
200          POP BC
210          HALT
220          DJNZ ET
230 ZEND    RET

```

(40 DATA "ATTR-UP",60430,35)

Exemplul 8.22:

```

10          ORG 60465          ;nume="ATTR-LEFT";
                                x=60465; y=52
20          ENT $
30          LD HL,53000
40          LD DE,16384
50          LD BC,6144
60          LDIR
70          LD BC,31
80          ADD HL,BC
90          PUSH DE
100         PUSH HL
110         POP IX
120         POP HL
130         ADD HL,BC
140         LD DE,32
150         LD B,32
160 ET1     PUSH BC
170         PUSH HL
180         PUSH IX
190         LD B,24
200 ET2     LD A,(IX+0)
210         LD (HL),A
220         ADD HL,DE
230         ADD IX,DE
240         DJNZ ET2
250         POP IX
260         POP HL
270         POP BC
280         DEC HL
290         DEC IX

```



```

30          LD HL,53000
40          LD DE,16384
50          LD BC,6144
60          LDIR
70          LD B,13
80  ET1     PUSH BC
90          PUSH HL
100         PUSH DE
110        LD HL,53000
120        LD DE,6528
130        ADD HL,DE
140        LD IX,22912
150        LD DE,32
160  ET2     ADD HL,DE
170        ADD IX,DE
180        DJNZ ET2
190        PUSH IX
200        POP DE
210        LD BC,32
220        LDIR
230        POP DE
240        POP HL
250        LD BC,32
260        LDIR
270        HALT
280        POP BC
290        DJNZ ET1
300  ZEND   RET

```

(40 DATA "ATTR-IN",60564,55)

Exemplul 8.25:

```

10          ORG 60619          ;nume="ATTR-OUT";
                                x=60619; y=75
20          ENT $
30          LD HL,53000
40          LD DE,16384
50          LD BC,6144
60          LDIR
70          LD B,13
80  ET1     PUSH BC
90          PUSH DE
100         LD HL,53000

```

```
110          LD DE,6112
120          ADD HL,DE
130          LD DE,32
140          LD IX,22496
150 ET2      ADD HL,DE
160          ADD IX,DE
170          DJNZ ET2
180          LD BC,32
190          PUSH IX
200          POP DE
210          LDIR
220          POP BC
230          LD HL,53000
240          LD DE,6912
250          ADD HL,DE
260          LD IX,23296
270          LD DE,65504
280 ET3      ADD HL,DE
290          ADD IX,DE
300          DJNZ ET3
310          PUSH IX
320          POP DE
330          LD BC,32
340          LDIR
350          HALT
360          POP BC
370          DJNZ ET1
380 ZEND     RET
```

(40 DATA "ATTR-OUT",60619,75)

După cum se poate constata, rutinele de la exemplele 8.13-8.25 au fost scrise cu adresele de start în succesiune astfel încît să se realizeze un program avînd lungimea de 696 octeți; acesta se exploatează cu următorul program BASIC:

```
10 CLS : PRINT "Incarcati un screen..."
20 LOAD "mira" CODE 53000,6912
30 FOR i=1 TO 13: READ b$,x,y
40 RANDOMIZE USR x
50 PRINT # 0; AT 1,16-(LEN b$/2);b$: PAUSE 100
60 NEXT i: RESTORE : GO TO 30
70 STOP
```

```

100 DATA "PRINT" , 60000 , 12 ; "FADE" , 60025 , 53 ;
    "SHUTTER" , 60078 , 51
110 DATA "SLIDE-DOWN" , 60129 , 75 , "SLIDE-UP" , 60204 , 70
120 DATA "SLIDE-LEFT" , 60274 , 67 , "SLIDE-RIGHT" , 60341 ,
    65
130 DATA "ATTR-DOWN" , 60406 , 24 , "ATTR-UP" , 60430 , 35
140 DATA "ATTR-LEFT" , 60465 , 52 , "ATTR-RIGHT" , 60517 , 47
    "ATTR-IN" , 60564 , 55 , "ATTR-OUT" , 60619 , 75

```

- Ultima dintre rutinele consacrate efectelor pe SCREEN realizează inversarea acestuia.

Exemplul 8.26:

10	ORG adr	;INVERSARE SCREEN (SAU TEXT)
20	ENT \$	
30	LD HL, 50000	
40	LD BC, 6144	
50 ET0	LD HL, 0	
60	INC HL	
70	DEC BC	
80	LD A, B	
90	OR C	
100	JR NZ, ET0	
110	LD A, 192	
120	LD HL, 16415	
130	LD DE, 50000	
140 ET1	LD B, 32	
150 ET2	BIT 0, (HL)	
160	JR Z, ET3	
170	EX DE, HL	
180	SET 7, (HL)	
190	EX DE, HL	
200 ET3	BIT 1, (HL)	
210	JR Z, ET4	
220	EX DE, HL	
230	SET 6, (HL)	
240	EX DE, HL	
250 ET4	BIT 2, (HL)	
260	JR Z, ET5	
270	EX DE, HL	
280	SET 5, (HL)	

```
290          EX DE,HL
300 ET5      BIT 3, (HL)
310          JR Z,ET6
320          EX DE,HL
330          SET 4, (HL)
340          EX DE,HL
350 ET6      BIT 4, (HL)
360          JR Z,ET7
370          EX DE,HL
380          SET 3, (HL)
390          EX DE,HL
400 ET7      BIT 5, (HL)
410          JR Z,ET8
420          EX DE,HL
430          SET 2, (HL)
440          EX DE,HL
450 ET8      BIT 6, (HL)
460          JR Z,ET9
470          EX DE,HL
480          SET 1, (HL)
490          EX DE,HL
500 ET9      BIT 7, (HL)
510          JR Z,ET10
520          EX DE,HL
530          SET 0, (HL)
540          EX DE,HL
550 ET10     DEC HL
560          INC DE
570          DJNZ ET2
580          DEC A
590          JR Z,ET12
600          LD B,64
610 ET11     INC HL
620          DJNZ ET11
630          JP ET1
640 ET12     LD HL,56175
650          LD DE,22528
660          LD C,24
670 ET13     LD B,32
680 ET14     LD A, (DE)
690          LD (HL),A
```



```

700          INC DE
710          DEC HL
720          DJNZ ET14
730          LD B,64
740 ET15     INC HL
750          DJNZ ET15
760          DEC
770          JR NZ,ET13
780          LD HL,50000
790          LD DE,16384
800          LD BC,6911
810          LDIR
820 ZEND     RET

```

Programul BASIC de folosire:

```

10 BORDER 2: PAPER 0: INK 7: CLS
20 LOAD "MIRA" SCREEN$ : PAUSE 0
30 FOR i=1 TO 5: RANDOMIZE USR adr: PAUSE 10: NEXT
  i

```

Exemplul 8.27:

```

10          ORG adr          ;EFECTUL "EXPLOZIE
                          BOMBE"
20          ENT $
30          LD HL,16384
40          LD B,192
50 ET1      PUSH BC
60          LD B,32
70 ET2      LD A,(HL)
80          CP 170
90          JR Z,ET3
100         LD (HL),254
110 ET3     INC HL
120         DJNZ ET2
130         POP BC
140         DJNZ ET1
150         LD A,7
160         OUT (254),A
170         LD HL,16384
180         LD B,192
190 ET4     PUSH BC
200         LD B,32
210 ET5     LD A,(HL)

```

```

220          CP 170
230          JR Z,ET6
240          LD HL,0
250 ET6      INC HL
260          DJNZ ET5
270          POP BC
280          DJNZ ET4
290          LD A,0
300          OUT (254),A
310 ZEND    RET

```

Această rutină de 55 octeți este elaborată pentru un joc în care se vizualizează efectul exploziei unei bombe; ea poate fi folosită și în alte scopuri cum ar fi rolul unei cortine între două ecrane succesive. În acest din urmă caz programul BASIC are forma:

```

10 BORDER 0: PAPER 0: INK 7: CLS
20 FOR i=1 TO 5: RANDOMIZE USR adr: NEXT i

```

Exemplul 8.28:

```

10          ORG adr          ;DREPTUNGHIURI
                COLORATE
                CRESCATOARE

20          ENT $
30          LD A,54
40 ET1      LD DE,32
50          LD HL,22896
60          LD BC,257
70 ET2      HALT
80          PUSH BC
90 ET3      LD (HL),A
100         DEC HL
110         DEC B
120        JR NZ,ET3
130 ET4     LD (HL),A
140        ADD HL,DE
150        DEC C
160        JR NZ,ET4
170        POP BC
180        PUSH AF
190        LD A,33
200        CP B
210        JR Z,ET9

```

```

220          INC B
230          LD A,24
240          CP C
250          JR Z,ET5
260          INC C
270 ET5      POP AF
280          PUSH BC
290 ET6      LD (HL),A
300          INC HL
310          DEC B
320          JR NZ,ET6
330 ET7      LD (HL),A
340          AND A
350          SBC HL,DE
360          DEC C
370          JR NZ,ET7
380          POP BC
390          PUSH AF
400          INC B
410          LD A,24
420          CP C
430          JR Z,ET8
440          INC C
450 ET8      POP AF
460          JR ET2
470 ET9      POP AF
480          PUSH AF
490          AND 7
500          OUT (254),A
510          POP AF
520          SUB 9
530          CP 247
540          JR NZ,ET1
550 ZEND     RET

```

(RANDOMIZE USR adr)

Exemplul 8.29:

```

10          ORG adr          ;PETE COLORATE PE
                                PAPER
20          ENT $
30          LD HL,(23563)
40          LD BC,4

```

50		ADD HL,BC
60		LD D, (HL)
70		LD BC, 8
80		ADD HL,BC
90		LD E, (HL)
100		LD (ET8),DE
110		ADD HL,BC
120		LD D, (HL)
130		ADD HL,BC
140		LD E, (HL)
150		LD (ET6),DE
160		ADD HL,BC
170		LD A, (HL)
180		AND 7
190		SLA A
200		SLA A
210		SLA A
220		LD (ET5),A
230		ADD HL,BC
240		LD A, (HL)
250		AND 1
260		JR Z,ET1
270		LD A, (ET5)
280		OR 64
290		LD (ET5),A
300	ET1	ADD HL,BC
310		LD A, (HL)
320		AND 1
330		JR Z,ET2
340		LD A, (ET5)
350		OR 128
360		LD (ET5),A
370	ET2	LD DE, (ET8)
380		LD A, (ET6)
390		CP 0
400		RET Z
410		LD A, (ET7)
420		CP 0
430		RET Z
440		LD (ET8),DE
450		LD A,E

460		AND 24
470		SRL A
480		SRL A
490		SRL A
500		OR 88
510		LD H,A
520		LD A,E
530		AND 7
540		OR A
550		RRA
560		RRA
570		RRA
580		RRA
590		ADD A,D
600		LD L,A
610		LD A,(ET6)
620		LD B,A
630	ET3	PUSH BC
640		PUSH HL
650		LD A,(ET7)
660		LD B,A
670	ET4	LD A,(HL)
680		AND 7
690		LD C,A
700		LD A,(ET5)
710		OR C
720		LD (HL),A
730		INC HL
740		DJNZ ET4
750		POP HL
760		LD BC,32
770		ADD HL,BC
780		POP BC
790		DJNZ ET3
800		RET
810	ET5	NOP
820	ET6	EX AF,AF'
830	ET7	NOP
840	ET8	INC B
850		LD C,0
860	ZEND	RET

Programul BASIC care folosește această rutină este următorul:

```

10 DEF FN c(x,y,h,v,c;b,f)=USR adr
20 BORDER 1: PAPER 4 : CLS
30 FOR i=1 TO 50: LET x1= INT(RND*17): LET y1=INT
   (RND*10): LET h1=INT(RND*16): LET
   v1=INT(RND*15): LET c1=INT(RND*7): RESTORE FN
   (x1,y1,h1,v1,c1,0,0): NEXT i
40 REM valorile sînt: x<32;y<24;x+h<32;y+v<24;
   c=culoarea (0..7);b=BRIGHT (0 sau 1);f=FLASH (1
   sau 0)

```

Exemplul 8.30:

10	ORG adr	;EFECTUL GRAFIC
		"MELC"
20	ENT \$	
30	JP ET10	
40 ET1	HALT	
50	LD HL,16384	
60	LD BC,6144	
70 ET2	LD (HL),0	
80	INC HL	
90	DEC BC	
100	LD A,B	
110	OR B	
120	JR NZ,ET2	
130	RET	
140 ET3	LD HL,22528	
150	LD DE,32	
160	LD C,24	
170 ET4	LD A,C	
180	AND 7	
190	LD B,A	
200	ADD A,A	
210	ADD A,A	
220	ADD A,A	
230	OR B	
240	RES 7,A	
250 ET5	LD B,A	
260	NOP	
270	LD A,C	
280	ADD A,7	
290 ET6	LD (HL),B	

```
300          INC HL
310          DEC A
320          JR NZ,ET6
330          LDA,C
340          DEC A
350 ET7      LD (HL),B
360          ADD HL,DE
370          DEC A
380          JR NZ,ET7
390          LD A,C
400          ADD A,7
410 ET8      LD (HL),B
420          DEC HL
430          DEC A
440          JR NZ,ET8
450          LD A,C
460          DEC A
470 ET9      LD (HL),B
480          SBC HL,DE
490          DEC A
500          JR NZ,ET9
510          INC HL
520          ADD HL,DE
530          HALT
540          HALT
550          DEC C
560          DEC C
570          JR NZ,ET4
580          RET
590 ET10     CALL ET3
600          HALT
610          HALT
620          HALT
630          LD HL,ET5
640          LD (HL),6
650          CALL ET3
660          LD HL,ET5
670          LD (HL),71
680          CALL ET1
690 ZEND     RET
```

(RANDOMIZE USR adr)

Exemplul 8.31:

```

10          ORG adr          ;EFECTUL "CHENAR
                                COLORAT"

120         ENT $

130         CALL LFA1E

140         LD HL,0

150         LD (LFB2B),HL

160         CALL LFAEB

170 LFA0C   CALL LFA7C

180         LD HL,15000

190         LD (LFB2B),HL

200         CALL LFAE8

210         CALL LFAF1

220         JR Z,LFA0C

230         RET

240 LFA1E   LD A,1

250         LD HL,22528

260         XOR A

270         LD (LFB2D),A

280         LD B,8

290 LFA29   LD (HL),56

300         INC HL

310         LD (HL),56

320         INC HL

330         LD (HL),16

340         INC HL

350         LD (HL),16

360         INC HL

370         DJNZ LFA29

380         LD HL,22560

390         LD DE,32

400         LD B,5

410 LFA3F   LD (HL),16

420         ADD HL,DE

430         LD (HL),16

440         ADD HL,DE

450         LD (HL),56

460         ADD HL,DE

470         LD (HL),56

480         ADD HL,DE

490         DJNZ LFA3F

```



```
400          LD (HL),16
410          ADD HL,DE
420          LD (HL),16
430          LD HL,22591
440          LD DE,32
450          LD B,5
460 LFA5A    LD (HL),56
470          ADD HL,DE
480          LD (HL),56
490          ADD HL,DE
500          LD (HL),16
510          ADD HL,DE
520          LD (HL),16
530          ADD HL,DE
540          DJNZ LFA5A
550          LD (HL),56
560          ADD HL,DE
570          LD (HL),56
580          ADD HL,DE
590          ADD HL,DE
600          LD DE,23264
610          LD HL,22528
620          LD BC,31
630          LDIR
640          RET
650 LFA7C    LD HL,LFB2D
660          INC (HL)
670          BIT 0,(HL)
680          JR Z,LFA91
690          CALL LFA9E
700          CALL LFAC4
710          CALL LFAAA
720          CALL LFAD0
730          RET
740 LFA91    CALL LFAAA
750          CALL LFAD0
760          CALL LFA9E
770          CALL LFAC4
780          RET
790 LFA9E    LD HL,22528
800          LD DE,22529
```

810		LD BC,31
820		LDDR
830		RET
840	LFAAA	LD HL,23263
850		LD DE,23295
860		LD BC,32
870		OR A
880		LD A,23
890	LFAB6	PUSH AF
900		LD A,(HL)
910		LD (DE),A
920		SBC HL,BC
930		EX DE,HL
940		SBC HL,BC
950		EX DE,HL
960		POP AF
970		DEC A
980		JR NZ,LFAB6
990		RET
1000	LFAC4	LD DE,23264
1010		LD HL,23265
1020		LD BC,31
1030		LDIR
1040		RET
1050	LFAD0	LD DE,22528
1060		LD HL,22560
1070		LD BC,32
1080		LD A,23
1090		OR A
1100	LFADC	PUSH AF
1110		LD A,(HL)
1120		LD (DE),A
1130		ADD HL,BC
1140		EX DE,HL
1150		ADD HL,BC
1160		EX DE,HL
1170		POP AF
1180		DEC A
1190		JR NZ,LFADC
1200		RET
1210	LFAE8	LD HL,(LFB2B)

```
1220 LFAEB      DEC HL
1230           LD A,L
1240           OR H
1250           JR NZ,LFAEB
1260           RET
1270 LFAF1      OR A
1280           LD A,254
1290           CALL NC,LFB20
1300           LD A,127
1310           CALL NC,LFB20
1320           LD A,191
1330           CALL NC,LFB20
1340           LD A,253
1350           CALL NC,LFB20
1360           CCF
1370           RET NC
1380           XOR A
1390           LD A,247
1400           CALL LFB20
1410           LD A,251
1420           CALL NC,LFB20
1430           LD A,239
1440           CALL NC,LFB20
1450           LD A,223
1460           CALL NC,LFB20
1470           RET C
1480           XOR A
1490           RET
1500 LFB20      IN A,(254)
1510           CPL
1520           AND 31
1530           JR NZ,LFB29
1540           XOR A
1150           RET
1560 LFB29      SCF
1570           RET
1580 LFB2B      SBC A,B           ;SAU DEFB 152
1590           LD A,(00018)
1600           NOP
1610 LFB2D      NOP           ;SAU DEFB 10
```

(BORDER 0:PAPER 0:INK 7:CLS:RANDOMIZE USR adr)

8.3. EFECTE AUDIO-VIZUALE

Exemplul 8.32:

```

10          ORG adr          ;CORTINA DREAPTA,
                                SUNET, STERGERE TEXT

20          ENT $

30          LD C,32

40          LD HL,22528

50  ET1     LD B,24

60          LD DE,32

70          PUSH HL

80  ET2     LD HL,18

90          ADD HL,DE

100         DJNZ ET2

110        PUSH BC

120        LD HL,208

130        LD DE,32

140        CALL 949

150        POP BC

160        POP HL

170        PUSH HL

180        LD B,24

190        LD DE,32

200  ET3     LD (HL),9

210        ADD HL,DE

220        DJNZ ET3

230        POP HL

240        INC HL

250        DEC HL

260        JR NZ,ET1

270  ZEND     RET

```

(RANDOMIZE USR adr)

Se poate ca textul să nu fie șters (acoperit de culoare) dacă se fac următoarele modificări:

```

200        LD (HL),15

250        DEC C

```

Exemplul 8.33:

```

10          ORG adr          ;CULORI ORIZONTALE
                                CU SUNET

100         ENT $

30 LF6E0    DEC B

40 LF6E1    ADD HL,SP

50          LD H,H

60          ADD A,B

70          ADD A,B

80          LD B,1

90 LF6E7    PUSH BC

100         LD A,(LF6E0)

110         DEC A

120         JR NZ,LF6F0

130         LD A,6

140 LF6F0    LD (LF6ED),A

150         SLA

160         SLA

170         SLA

180         LD HL,22528

190         LD B,18

200 LF6FE    PUSH BC

210         LD B,32

220 LF701    LD (HL),A

230         INC HL

240         DJNZ LF701

250         SUB 8

260         JR NZ,LF708

270         LD A,48

280 LF708    POP BC

290         DJNZ LF6FE

300         LD IX,LF6E1

310         CALL LF719

320         POP BC

330         DJNZ LF6E7

340         RET

350 LF719    PUSH HL

360         PUSH DE

370         PUSH BC

380         PUSH AF

390         LD H,(IX+0)

```

```
400          LD B, (IX+1)
410          LD D, (IX+2)
420          LD E, (IX+3)
430 LF72A    PUSH BC
440          LD A, (23624)
450          SRL A
460          SRL A
470          SRL A
480          SET 4,A
490          OUT (254),A
500          LD B,D
510 LF739    NOP
520          DJNZ LF739
530          LD B, (HL)
540          INC B
550          INC B
560 LF73F    NOP
570          NOP
580          NOP
590          NOP
600          DJNZ LF73F
610          RES 4,A
620          OUT (254),A
630          LD B,E
640 LF74A    NOP
650          DJNZ LF74A
660          INC D
670          INC E
680          INC HL
690          POP BC
700          DJNZ LF72A
710          POP AF
720          POP BC
730          POP DE
740          POP HL
750 ZEND    RET
```

Rutina, care realizează un PAPER colorat prin dungii orizontale fără să șteargă ecranul, este exploatată cu următorul program BASIC:

```
10 LET durata=10: LET linii=24: GOSUB 100: STOP
100 LET a=adr: POKE (a+6),durata: POKE (a+29),linii
110 RANDOMIZE USR (a+5): RETURN
```

120 REM durata=durata efectului; linii=nr.liniilor
pe care are loc efectul grafic

Exemplul 8.34:

10	ORG adr	;CORTINA STINGA- DREAPTA CU BENZI VERTICALE COLORATE SI SUNET
20	ENT \$	
30	CALL LE35C	
40	RET	
50	LE35C	LD IX,15
60		LD HL,16
70		LD B,16
80	LE365	PUSH BC
90		PUSH HL
100		CALL LE37D
110		PUSH IX
120		PUSH HL
130		CALL LE 370
140		POP HL
150		POP BC
160		INC HL
170		DEC IX
180		LD A,255
190		LD (23664),A
200		DJNZ LE365
210		RET
220	LE37D	LD B,24
230		LD A,(LE3E8)
240		ADD A,2
250		LD (LE3E8),A
260	LE387	PUSH BC
270		PUSH HL
280		CALL LE3A0
290		POP HL
300		LD A,(LE3E8)
310		AND 127
320		LD C,A
330		LD A,32
340		CALL LE3B7
350		LD DE,32

```
360          ADD HL,DE
370          POP BC
380          DJNZ LE387
390          RET
400  LEA3A0   LD A, (23664)
410          CP 0
420          RET Z
430          DEC A
440          LD (23664),A
450          LD C,A
460          LD B,A
470          LD A,16
480  LE3AE   OUT (254),A
490          DJNZ LE3AE
500          LD B,C
510          XOR A
520          OUT (254),A
530          RET
540  LE3B7   PUSH HL
550          PUSH AF
560          EX DE,HL
570          LD HL,22528
580          ADD HL,DE
590          LD (HL),C
600          LD HL,16384
610          LD A,D
620          OR A
630          JR Z,LE3C8
640          LD H,71
650  LE3C8   CP2
660          JR NZ,LE3CE
670          LD H,78
680  LE3CE   ADD HL,DE
690          POP AF
700          PUSH HL
710          LD L,A
720          LD H,0
730          ADD HL,HL
740          ADD HL,HL
750          ADD HL,HL
760          EX DE,HL
```



```

770          LD HL,LE3EA
780          ADD HL,DE
790          EX DE,HL
800          POP HL
810          LD B,8
820 LE3E0    LD A,(DE)
830          INC DE
840          LD (HL),A
850          INC H
860          DJNZ LE3E0
870          POP HL
880          RET
890 LE3E8    ADD A,B
900          NOP
910 LE3EA    NOP

```

(RANDOMIZE USR adr)

Exemplul 8.35:

```

10          ORG adr          ;EFFECTUL "MELC" CU
                                SUNET
20          ENT $
30          LD BC,7960
40          LD HL,22496
50 LF650    LD A,2
60 LF652    CALL LF68C
70          JR Z,LF674
80          CP 8
90          JR NZ,LF65D
100         LD A,2
110 LF65D   PUSH AF
120         LD A,17
130         OUT (254),A
140         POP AF
150         CALL LF6B9
160         JR Z,LF674
170         PUSH AF
180         LD A,1
190         OUT (254),A
200         POP AF
210         CP 8
220         JR Z,LF652
230         JR LF650

```

```
240 LF674      LD BC,7960
250           LD HL,22496
260 LF67A      LD A,33
270           CALL LF68C
280           RET Z
290           OUT (254),A
300           CALL LF6B9
310           RET Z
320           OR 16
330           OUT (254),A
340           JR LF67A
350 LF68C      PUSH BC
360           PUSH AF
370           RLCA
380           RLCA
390           RLCA
400           LD B,C
410 LF692      LD DE,32
420           ADD HL,DE
430           LD (HL),A
440           DJNZ LF692
450           POP AF
460           POP BC
470           CP 33
480           JR Z,LFCA0
490           INC A
500 LFCA0      CALL LF6DD
510           PUSH BC
520           PUSH AF
530           RLCA
540           RLCA
550           RLCA
560 LF6A8      INC HL
570           LD (HL),A
580           DJNZ LF6A8
590 LF6AC      POP AF
600           CP 33
610           JR Z,LF6B2
620           INC A
630 LF6B2      POP BC
640           CALL LF6DD
```

```
650          DEC B
660          DEC C
670          RET
680 LF6B9     PUSH BC
690          PUSH AF
700          RLCA
710          RLCA
720          RLCA
730          LD B,C
740 LF6BF     LD DE,32
750          OR A
760          SBC HL,DE
770          LD (HL),A
780          DJNZ LF6BF
790          POP AF
800          POP BC
810          CP 33
820          JR Z,LF6CF
830          INC A
840 LF6CF     CALL LF6DD
850          PUSH BC
860          PUSH AF
870          RLCA
880          RLCA
890          RLCA
900 LF6D7     DEC HL
910          LD (HL),A
920          DJNZ LF6D7
930          JR LF6AC
940 LF6DD     PUSH AF
950          PUSH BC
960          PUSH DE
970          PUSH HL
980          LD HL,416
990 LF6E4     DEC HL
1000         LD A,H
1010         OR L
1020         JR NZ,LF6E4
1030         POP HL
1040         POP DE
1050         POP BC
```

1060 POP AF

1070 ZEND RET

(RANDOMIZE USR adr)

Exemplul 8.36:

```
10          ORG adr          ;DUBLU EFECT "MELC"  
                                CU SUNET  
  
20          ENT $  
30 LFE08    JR BC,65342  
40          JR LFDF2  
50          LD A,1  
60          JR LFDF2  
70          XOR A  
80 LFD2     LD (LLFDE8),A  
90          LD HL,0  
100         LD DE,8216  
110        LD A,D  
120        CALL LFE72  
130        LD B,D  
140 LFE00   CALL LFEAA  
150        INC H  
160        CALL LFE56  
170        DJNZ LFE00  
180        DEC H  
190        JR LFE1E  
200 LFE0C   LD A,D  
210        OR A  
220        RET Z  
230        CALL LFE72  
240        LD B,D  
250 LFE13   CALL LFEAA  
260        INC H  
270        CALL LFE56  
280        DJNZ LFE13  
290        DEC D  
300        DEC H  
310 LFE1E   LD A,E  
320        OR A  
330        RET Z  
340        CALL LFE72  
350        LD B,E  
360 LFE25   CALL LFEAA
```

```
370          INC L
380          CALL LFE56
390          DJNZ LFE25
400          DEC E
410          DEC L
420          LD A,D
430          OR A
440          RET Z
450          CALL LFE72
460          LD B,D
470 LFE37    CALL LFEAA
480          DEC HL
490          CALL LFE56
500          DJNZ LFE37
510          DEC D
520          INC H
530          LD A,E
540          OR A
550          RET Z
560          CALL LFE72
570          LD B,E
580 LFE49    CALL LFEAA
590          DEC L
600          CALL LFE56
610          DJNZ LFE49
620          DEC E
630          INC L
640          JR LFE0C
650 LFE56    PUSH BC
660          PUSH AF
670          PUSH DE
680          INC D
690          LD BC,254
700 LFE5D    LD A,16
710          OUT(C),A
720          XOR A
730          OUT(C),A
740          LD A,16
750          OUT(C),A
760          XOR A
770          OUT(C),A
```

```
780          DEC D
790          JR NZ,LFE5D
800          POP DE
810          POP AF
820          POP BC
830          RET
840 LFE72    PUSH AF
850          LD A, (LFDE8)
860          OR A
870          JR Z,LFE8D
880          CP 1
890          JR Z,LFE91
900          POP AF
910          PUSH BC
920          AND 7
930          LD C,A
940          SLA A
950          SLA A
960          SLA A
970          OR C
980          POP BC
990          AND 63
1000         RET
1010 LFE8D   POP AF
1020         LD A,69
1030         RET
1040 LFE91   POP AF
1050         XOR A
1060         RET
1070 LFE94   PUSH AF
1080         PUSH DE
1090         LD D,0
1100        LD E,L
1110        EX DE,HL
1120        ADD HL,HL
1130        ADD HL,HL
1140        ADD HL,HL
1150        ADD HL,HL
1160        ADD HL,HL
1170        LD A,D
1180        ADD A,L
```

```

1190      LD L,A
1200      LD DE,20528
1210      ADC HL,DE
1220      POP DE
1230      POP AF
1240      RET
1250 LFEAA  PUSH HL
1260      CALL LFE49
1270      LD (HL),A
1280      POP HL
1290 ZEND   RET

```

[RANDOMIZE USR adr: RANDOMIZE USR (adr+5)]

Exemplu 8.37:

```

10      ORG adr           ;EFECTUL "STAR3D"
20      ENT $
30      LD HL,16384
40      LD DE,4096
50      LD D,24
60      LD L,8
70 LFCF3  LD H,80
80      LD B,8
90 LFCF7  LD (HL),0
100     INC H
110     DJNZ LFCF7
120     INC L
130     DEC D
140     JR NZ,LFCF3
150     LD D,24
160     LD L,232
170 LFD04  LD H,80
180     LD B,8
190 LFD08  LD (HL),0
200     INC H
210     DJNZ LFD08
220     INC L
230     DEC D
240     JR NZ,LFD04
250     LD A,0
260     LD (LFD0A),A
270     LD (LFD0A1),A
280     LD HL,2304

```

```
290          LD (LFDA2),HL
300 LFD1F    LD HL,22766
310          LD C,2
320          LD A,C
330          LD (LFDA4),A
340 LFD28    LD B,C
330 LFD29    LD A,(LFDA0)
340          LD (HL),A
350          INC HL
360          LD (HL),A
370          INC HL
380          LD A,16
390          OUT (254),A
400          LD A,0
410          OUT (254),A
420          DJNZ LFD29
430          LD DE,32
440          LD B,C
450 LFD3E    DEC DE
460          DEC DE
470          DJNZ LFD3E
480          ADD HL,DE
490          LD A,(LFDA4)
500          DEC A
510          LD (LFDA4),A
520          JR NZ,LFD28
530          LD DE,(LFDA2)
540          LD A,16
550          OUT (254),A
560 LFD54    DEC DE
570          LD A,D
580          OR E
590          JR NZ,LFD54
600          LD A,0
610          OUT (254),A
620          LD DE,(LFDA2)
630          DEC DE
640          DEC DE
650          DEC DE
660          DEC DE
670          DEC DE
```



```
680          DEC DE
690          DEC DE
700          DEC DE
710          DEC DE
720          DEC DE
730          DEC DE
740          DEC DE
750          DEC DE
760          DEC DE
770          DEC DE
780          DEC DE
790          LD (LFDA2),DE
800          LD B,C
810          INC B
820          LD DE,65504
830 LFD7A    ADD HL,DE
840          DJNZ LFD7A
850          DEC HL
860          DEC HL
870          INC C
880          INC C
890          LD A,C
900          LD (LFDA4),A
910          CP 18
920          NOP
930          JR NZ,LFD28
940          LD A,(LFDA0)
950          ADD A,8
960          AND 63
970          LD (LFDA0),A
980          LD A,(LFDA1)
990          INC A
1000         LD (LFDA1),A
1010        CP 17
1020        JR NZ,LFD1F
1030 ZEND    RET
1040 LFDA0   EX AF,AF'
1050 LFDA1   LD DE,128
1060 LFDA4   LD (DE),A
1070 LFDA2   NOP
```

Programul BASIC ce pune în valoare rutina prezentată este:

```

10 BORDER 0: PAPER 0: INK 7: CLS
20 PRINT AT 6,3; PAPER 5; INK 1;"M.M.POPOVICI
   SOFTWARE 1993"; AT 10,7; PAPER 6; INK 2; FLASH
   L; "EFFECT AUDIO-VIZUAL"
30 PLOT 0,172: DRAW 251,0: DRAW 0,-90: DRAW -251,0:
   DRAW 0,90
40 PAUSE 30: RANDOMIZE USR adr: CLS

```

Efectul obținut este foarte interesant și nu șterge textul scris pe ecran.

Exemplul 8.38:

```

10          ORG adr          ; EFECTUL "SPECTRES"
20          ENT S
30          LD L,1
40          LD C,23
50 LE488    LD B,32
60          LD H,0
70          LD A,0
80 LE48C    CALL LE4FA
90          INC H
100         DJNZ LE48C
110         CALL LE4CA
120         LD A,120
130         LD H,0
140         DEC L
150         LD A,L
160         CP 2
170         JR NZ,LE4A5
180 LEA41    LD A,122
190         JR LE4B3
200 LE4A5    CP 3
210         JR Z,LE4A1
220         CP 4
230         JR NZ,LE4B1
240         LD A,121
250         JR LE4B3
260 LE4B1    LD A,120
270 LE4B3    LD B,32
280 LE4B5    CALL LE4F4
290         INC H
300         DJNZ LE4B5

```

```
310      CALL LE4CA
320      INC L
330      INC L
340      DEC C
350      JR NZ, LE488
360      RET
370 LE4CA  PUSH AF
380      PUSH BC
390      PUSH DE
400      LD D, 80
410      LD BC, 254
420 LE4CC  LD A, D
430      NOP
440      NOP
450      NOP
460      LD E, D
470 LE4D1  LD A, 16
480      OUT (C), A
490      XOR A
500      OUT (C), A
510      LD A, 16
520      OUT (C), A
530      XOR A
540      OUT (C), A
550      DEC E
560      JR NZ, LE4D1
570      DEC D
580      JR Z, LE4F0
590      DEC D
600      JR Z, LE4F0
610      DEC D
620      JR Z, LEF40
630      DEC D
640      JR Z, LE4F0
650      JR LE4CC
660 LE4F0  POP DE
670      POP BC
680      POP AF
690      RET
700 LE4F4  PUSH HL
710      CALL LE4FB
```

```

720          LD (HL),A
730          POP HL
740          RET
750 LE4FB    PUSH AF
760          PUSH DE
770          LD D,0
780          LD E,L
790          EX DE,HL
800          ADD HL,HL
810          ADD HL,HL
820          ADD HL,HL
830          ADD HL,HL
840          ADD HL,HL
850          LD A,D
860          ADD A,L
870          LD L,A
880          LD DE,22528
890          ADC HL,DE
900          POP DE
910          POP AF
920          RET
930          AND 7
940          PUSH BC
950          LD B,0
960          LD C,254
970          OUT (C),A
980          POP BC
990 ZEND    RET

```

(RANDOMIZE USR adr)

Exemplul 8.39:

```

10          .          ORG adr          ;CHENAR COLORAT CU
                .          SUNET
20          ENT $
30          NOP
40          LD A,255
50          LD (LF03C),A
60          LD B,255
70 LF044    PUSH BC
80          CALL LF04C
90          POP BC
100         DJNZ LF044

```

```
110          RET
120 LF04C    LD HL,0
130          LD B,32
140 LF051    PUSH BC
150          LD A,R
160          LD C,A
170          LD A,0
180          CALL LFOB9
190          LD DE,736
200          ADD HL,DE
210          LD A,R
220          LD C,A
230          LD A,0
240          CALL LFOB9
250          LD DE,735
260          SBC HL,DE
270          POP BC
280          DJNZ LF051
290          LD HL,0
300          LD B,23
310 LF073    PUSH BC
320          LD A,R
330          LD C,A
340          LD A,0
350          CALL LFOA5
360          LD DE,31
370          ADD HL,DE
380          LD A,R
390          LD C,A
400          LD A,0
410          CALL LFOA5
420          INC HL
430          POP BC
440          DJNZ LF073
450          LD A,(LF03C)
460          CP 0
470          RET Z
480          DEC A
490          LD (LF03C),A
500          LD C,A
510          LD B,A
```

520		LD A,16
530	LF09A	OUT (254),A
540		DJNZ LF09A
550		LD B,C
560		XOR A
570		OUT (254),A
580		RET
590	LFOA3	POP HL
600		RET
610	LFOA5	PUSH HL
620		LD DE,769
630		SBC HL,DE
640		JR NC,LFOA3
650		NOP
660		POP HL
670		PUSH HL
680		LD DE,31
690		SBC HL,DE
700		JR C,LFOA3
710		NOP
720		POP HL
730	LFOB9	PUSH HL
740		PUSH AF
750		EX DE,HL
760		LD HL,22528
770		ADD HL,DE
780		LD (HL),C
790		LD HL,16384
800		LD A,D
810		OR A
820		JR Z,LFOCA
830		LD H,71
840	LFOA	CP 2
850		JR NZ,LFOD0
860		LD H,78
870	LFOD0	ADD HL,DE
880		POP AF
890		PUSH HL
900		LD L,A
910		LD H,0
920		ADD HL,HL

```

930      ADD HL,HL
940      ADD HL,HL
950      EX DE,HL
960      LD HL,LFOEA
970      ADD HL,DE
980      EX DE,HL
990      POP HL
1000 LFOE2  LD A,(DE)
1020      INC DE
1030      LD (HL),A
1040      INC H
1050      DJNZ LFOE2
1060      POP HL
1070      RET
1080 LFOEA  NOP

```

(RANDOMIZE USR adr)

Exemplul 8.40:

```

10      ORG adr           ;EFFECTUL "PLUS"
20      ENT $
30 L7A3A  CALL L7CDF
40      LD (L7A30),HL
50      LD (L7A34),HL
60      NOP
70      CALL L7AEE
80      LD HL,23264
90      LD (L7A36),HL
100     XOR A
110     LD (L7A38),A
120     LD B,12
130 L7A53  PUSH BC
140     LD HL,(L7A30)
150     PUSH HL
160     LD B,24
170     LD DE,32
180 L7A5D  LD (HL),27
190     ADD HL,DE
200     DJNZ L7A5D
210     POP HL
220     INC HL
230     LD (L7A30),HL
240     LD HL,(L7A32)

```

```
250          PUSH HL
260          LD B,22
270 L7A6D    LD (HL),27
280          ADD HL,DE
290          DJNZ L7A6D
300          POP HL
310          DEC HL
320          LD (L7A32),HL
330          LD HL,(L7A34)
340          PUSH HL
350          LD B,32
360 L7A7D    LD (HL),27
370          INC HL
380          DJNZ L7A7D
390          POP HL
400          ADD HL,DE
410          LD (L7A34),HL
420          LD HL,(L7A36)
430          PUSH HL
440          LD B,32
450 L7A8D    LD (HL),27
460          INC HL
470          DJNZ L7A8D
480          POP HL
490          CALL L7AF5
500          LD (L7A36),HL
510          CALL L7ADA
520          POP BC
530          DJNZ L7A53
540          LD A,(L7A38)
550          CP 1
560          JR Z,L7AC9
570          LD A,1
580          LD (L7A38),A
590          LD B,12
600          LD A,0
610          LD (L7A5D+1),A
620          LD (L7A6D+1),A
630          LD (L7A7D+1),A
640          LD (L7A8d+1),A
650          LD HL,22544
```



```
660          LD (L7A30) ,HL
670          LD HL,22543
680          LD (L7A32) ,HL
690          JR L7A53
700 L7AC9     LD A,27
710          LD (L7A5D+1) ,A
720          LD (L7A6D+1) ,A
730          LD (L7A7D+1) ,A
740          LD (L7A8D+1) ,A
750          RET
760 L7ADA     LD BC,1706
770 L7ADD     DEC BC
780          LD A,B
790          OR C
800          CP 0
810          JR NZ,L7ADD
820          LD HL,439
830          LD DE,13
840          CALL 949
850          RET
860 L7AEE     LD HL,22559
870          LD (L7A32) ,HL
880          RET
890 L7AF5     CCF
900          SBC HL,DE
910          INC HL
920          RET
930 L7CDF     LD A,255
940          LD HL,22560
950 ZEND      RET
960 L7A30     DEFW 0
970 L7A32     DEFW 0
980 L7A34     DEFW 0
990 L7A36     DEFW 0
1000 L7A38    DEFW 0
```

(RANDOMIZE USR adr)

8.4. MODALITĂȚI DE SCRIERE

Din rațiuni didactice, rutinele care urmează au fost împărțite în trei categorii:

- scrierea textelor curente (texte multiple);
- scrierea titlurilor;
- modalități combinate.

8.4.1. Scrierea textelor curente

Exemplul 8.41:

```

10          ORG adr          ;SCRIERE SENILA CU
                                SUNET SI EFECTE PE
                                BORDER
10          ENT $
30          LD DE,1
40          LD HL,0
50          LD B,72
60 ET1      LD C,L
70          LD A,HL
80          OUT (254),A
90          LD H,0
100         LD L,A
110        PUSH HL
120        PUSH DE
130        PUSH BC
140        CALL 949
150        POP BC
160        POP DE
170        POP HL
180        LD H,0
190        LD L,C
200        INC L
210        DJNZ ET1
220 ZEND    RET

```

Programul BASIC corespunzător este următorul:

```

10 BORDER 0: PAPER 0: INK 7: CLS
20 LET pas=1: LET LS="32□": REM tasta 4 în modul
   grafic de 32 ori
30 GOSUB 100
40 LET L$="CINESTOSTATICA STUDIAZA FORTELE CARE
   ACTIONEAZA ASUPRA ELEMENTE-LOR SI CUPLELOR
   MECANISMULUI IN TIMPUL UNUI CICLU ENERGETIC PRE-
   SUPUNIND CUNOSCUTA SI CONSTANTA VITEZA UNGHIULAR
   A A ELEMENT- LUI CONDUCATOR": GOSUB 100
50 LET L$=" O ASEMENEA IPOTEZA ESTE SIMPLI-FICATOAR
   E DEOARECE STAREA DE MISCARE A MASINII ESTE
   REZULTA- TUL ACTIUNII FORTELOR, IAR UNELE DIN
   FORTE (CUM SINT CELE DE I- NERTIE) SINT
   DEPENDENTE DE STA- REA DE MISCARE A MASINII.":
   GOSUB 100
60 LET L$="32□": REM tasta 4 în modul grafic de 32
   ori
70 GOSUB 100
80 STOP

```

Exemplul 8.42:

10	ORG adr	;SCRIERE RAPIDA, CU CURSOR SI SUNET
20	ENT \$	
30	LF5B4	RLCA
40	LF5B4	NOP
50		INC A
60		LD HL,15360
70		LD (LF5B5),HL
80		LD HL,(23627)
90	LF5C0	LD A,(HL)
100		CP 65
110		JR Z,LF5D4
120		CALL 6584
130		EX DE,HL
140		LD DE,HL
150		AND A
160		SBC HL,DE
170		ADD HL,DE
180		RET NC
190		JR LF5C0
200	LF5D4	INC HL

```
210          LD C, (HL)
220          INC HL
230          LD B, (HL)
240          INC HL
250          PUSH HL
260          POP IX
270          LD L, 0
280          LD H, 0
290 LF5E0    LD A, B
300          OR C
310          RET Z
320          PUSH BC
330          CALL LF5EF
340          POP BC
350          DEC BC
360          JR LF5E0
370 LF5EB    ADD HL, SP
380          EX AF, AF'
390          INC D
400          INC D
410 LF5EF    LD A, (IX+0)
420          PUSH AF
430          LD A, (LF637)
440          LD (LF5B4), A
450          LD A, 127
460          LD (LF637), A
470          LD DE, 0
480          CALL LF640
490          LD A, (LF5B4)
500          LD (LF637), A
510          POP AF
520          PUSH IX
530          PUSH AF
540          LD IX, LF5EB
550          LD (IX+1), 8
560          CALL LF689
570          POP AF
580          POP IX
590          LD D, 0
600          LD E, A
610          CALL LF640
```

620		JR NC,LF633
630		INC L
640		LD H,0
650		CALL LF640
660		JR NC,LF633
670		LD L,0
680		LD H,0
690		CALL LF640
700	LF633	INC H
710		INC IX
720		RET
730	LF637	RLCA
740	LF638	LD A,31
750		CP H
760		RET C
770		LD A,23
780		CP L
790		RET
800	LF640	CALL LF638
810		RET C
820		PUSH HL
830		PUSH BC
840		PUSH DE
850		LD C,H
860		LD H,0
870		LD B,H
880		SLA L
890		SLA L
900		SLA L
910		SLA L
920		RL H
930		SLA L
940		RL H
950		ADD HL,BC
960		PUSH HL
970		LD BC,22528
980		ADD HL,BC
990		LD A,(LF637)
1000		LD (HL),A
1010		SLA E
1020		RL D

1030		SLA E
1040		RL D
1050		SLA E
1060		RL D
1070		LD HL, (LF5B5)
1080		ADD HL, DE
1090		EX DE, HL
1100		POP HL
1110		SLA H
1120		SLA H
1130		SLA H
1140		SET 6, H
1150		LD B, 8
1160	LF67F	LD A, (DE)
1170		LD (HL), A
1180		INC DE
1190		INC H
1200		DJNZ LF67F
1210		POP DE
1220		POP BC
1230		POP HL
1240		RET
1250	LF689	PUSH HL
1260		PUSH DE
1270		PUSH BC
1280		PUSH AF
1290		LD H, (IX+0)
1300		LD B, (IX+1)
1310		LD B, (IX+2)
1320		LD B, (IX+3)
1330	LF69A	PUSH BC
1340		LD A, (23624)
1350		SRL A
1360		SRL A
1370		SRL A
1380		SET 4, A
1390		OUT (254), A
1400		LD B, D
1410	LF6A9	NOP
1420		DJNZ LF6A9
1430		LD B, (HL)

```

1440          INC B
1450          INC B
1460  LF6AF   NOP
1470          NOP
1480          NOP
1490          NOP
1500          DJNZ LF6AF
1510          RES 4,A
1520          OUT (254),A
1530          LD B,E
1540  LF6BA   NOP
1550          DJNZ LF6BA
1560          INC D
1570          INC E
1580          INC HL
1590          POP BC
1600          DJNZ LF69A
1610          POP AF
1620          POP BC
1630          POP DE
1640          POP HL
1650  ZEND    RET

```

Programul BASIC:

```

10 BORDER 0: PAPER 0: INK 7: CLS
20 LET PAPER=0: LET INK=7: LET BRIGHT=1: LET
   FLASH=0: LET SOUND=8: LET A=ADR
30 LET a$="TEXT (max.22 rînduri a cite max.32
   caractere)": GOSUB 100: STOP
100 POKE (a+131),INK+8*PAPER+64*BRIGHT+128*FLASH:
   POKE (a+97),SOUND
110 RANDOMIZE USR A: RETURN

```

Exemplul 8.43:

```

10          ORG adr          ;ROUTINA DE SCRIERE
                                CŪ LITERE MARITE
20          ENT $
30          LD HL,(23606)
40          INC H
50          LD A,(FIN+5)
60          LD C,A
70          LD B,0
80          RL C

```

90		RL B
100		RL C
110		RL B
120		RL C
130		RL B
140		ADD HL,BC
150		LD BC,8
160		LD DE,FIN+16
170		LDIR
180		LD HL,FIN+5
190		LD DE,(FIN+1)
200		LD BC,(FIN+3)
210		LD A,8
220	L5D19	PUSH AF
230		INC HL
240		LD A,B
250		SUB D
260		LD B,A
270		JR L5D25
280	L5D20	POP AF
290		DEC A
300		JR NZ,L5D19
310		RET
320	L5D25	LD A,8
330	L5D27	PUSH AF
340		PUSH HL
350		RLC (HL)
360		JR C,L5D3B
370	L5D2D	POP HL
380		LD A,E
390		ADD A,C
400		LD C,A
410		POP AF
420		DEC A
430		JR NZ,L5D27
440		LD A,(FIN+3)
450		LD C,A
460		JR L5D20
470	L5D3B	PUSH DE
480		PUSH BC
490	L5D3D	PUSH DE


```

500          PUSH BC
510 L5D3F    JR L5D4F
520 L5D41    DEC B
530          DEC D
540          JR NZ,L5D3F
550          POP BC
560          POP DE
570          INC C
580          DEC E
590          JR NZ,L5D3D
600          POP BC
610          POP DE
620          JR L5D2D
630 L5D4F    PUSH DE
640          PUSH BC
650          CALL 8933
660          POP BC
670          POP DE
680          JR L5D41
690 FIN      NOP
700          NOP
710          NOP
720          NOP
730          NOP
740          NOP
750          NOP
760          NOP
770          NOP
780          NOP
790          NOP
800          NOP
810          NOP
820          NOP
830 ZEND    NOP

```

Această rutină de numai 122 octeți permite mărirea caracterelor la orice dimensiune (lățime max. 32 caractere și înălțime max.22 caractere)

folosind variabila șir: z\$="1 2 3 4 5 6 7 8"

în care cifrele: 1 și 2 : numărul coloanei

3 și 4 : numărul liniei

5 și 6 : lățimea caracterului

7 și 8 : înălțimea caracterului

Programul BASIC aferent are forma următoare:

```

5 BORDER 2: PAPER 0: INK 7: CLS
10 LET z$="00000203RUTINA DE .SCRIS": GOSUB 120
20 LET z$="03000102caractere de orice marime":
  GOSUB 120
30 LET z$="05000102Cifrele 1-2=nr.coloanei": GOSUB
  120
40 LET z$="070801023-4=NR.LINIEI": GOSUB 120
50 LET z$="090801025-6=LATIME CHARACTER": GOSUB 120
60 LET z$="110801027-8=inaltime caracter": GOSUB
  120
70 STOP
120 LET adr: POKE (a+108),z$(5 TO 6): POKE (a+109),
  sZ(7 TO 8): POKE (a+111),175+PEEK (a+109)-
  8*zS(TO 2): POKE (a+110),8*Zs(3 TO 4): LET
  z$=z$(9TO)
130 FOR z=SGN PI TO LEN z$: POKE (a+112),CODE z$
  (z)-32: RANDOMIZE USR a: POKE (a+110),((PEEK
  (A+110))+8*(PEEK (a+108)))*((PEEK (a+110))+8*PEEK
  (a+108))<256: NEXT z: RETURN

```

8.4.2. Scrierea de titluri cu litere mărite

Exemplul 8.44:

10	ORG adr	;SCRIERE CU LITERE DUBLE SI SUNET
20	ENT \$	
30	LD DE,18435	
40	LD HL,TEXT1	
50	CALL L9CCC	
60	LD DE,20485	
70	LD HL,TEXT2	
80	CALL L9CCC	
90	RET	
100	L9C40	PUSH HL
110		LD L,A
120		LD H,0
130		ADD HL,HL
140		ADD HL,HL

```
150      ADD HL,HL
160      LD DE,15360
170      ADD HL,DE
180      LD DE,(23684)
190      LD B,8
200 L9C51 LD A,(HL)
210      SLA A
220      OR (HL)
230      LD DE,(A)
240      INC HL
250      PUSH HL
260      EX DE,HL
270      PUSH AF
280      CALL L9C6F
290      POP AF
300      EX DE,HL
310      OR (HL)
320      LD (DE),A
330      EX DE,HL
340      CALL L9C6F
350      EX DE,HL
360      POP HL
370      DJNZ L9C51
380      LD HL,23684
390      INC (HL)
400      POP HL
410      RET
420 L9C6F LD A,H
430      AND 7
440      CP 7
450      JR Z,L9C78
460      INC H
470      RET
480 L9C78 LD A,L
490      AND 14
500      CP 14
510      JR Z,L9C86
520      LD DE,1760
530      AND A
540      SBC HL,DE
550      RET
```

```
560 L9C86 LD DE,32
570 ADD HL,DE
580 RET
590 L9C8B LD A,(HL)
600 OR A
610 RET Z
620 CP 22
630 JR Z,L9C98
640 INC HL
650 CALL L9C40
660 JR L9C8B
670 L9C98 CALL L9C9D
680 JR L9C8B
690 L9C9D INC HL
700 LD C,(HL)
710 INC HL
720 LD B,(HL)
730 INC HL
740 PUSH HL
750 LD A,B
760 AND 248
770 ADD A,64
780 LD H,A
790 LD A,B
800 AND 7
810 RRCA
820 RRCA
830 RRCA
840 ADD A,C
850 LD L,A
860 LD (23684),HL
870 POP HL
880 RET
890 L9CB6 CALL L9C9D
900 L9CCC LD (23684),DE
910 L9CD0 LD A,(HL)
920 OR A
930 RET Z
940 CP 22
950 JR Z,L9CB6
960 INC HL
```

```

970          CALL L9C40
980          PUSH HL
990          LD HL,9000
1000 L9CDF   LD A, (HL)
1010         AND 16
1020         OUT (254),A
1030         DEC HL
1040         LD A,H
1050         OR L
1060         JR NZ,L9CDF
1070         POP HL
1080         JR L9CD0
1090 TEXT1   DEFM "M.M.POPOVICI software 1993"
1100         DEFB 0
1110 TEXT2   DEFM "PROGRAME IN COD-MASINA"
1120         DEFB 0

```

Exemplul 8.45:

```

10          ORG adr          ;SCRIERE CU LITERE
                                DUBLE COLORATE
20          ENT $
30          LD HL,20480
40          LD BC,2048
50 LFBFA   LD A, (HL)
60          AND A
70          RRA
80          OR (HL)
90          LD (HL),A
100         INC HL
110        DEC BC
120        LD A,B
130        OR C
140        JR NZ,LFBFA
150        LD HL,20480
160        LD DE,224
170        XOR A
180        LD C,8
190 LFCOE   LD B,32
200 LFC10   RRD
210        INC HL
220        DJNZ LDC10
230        ADD HL,DE

```

```
240          DEC C
250          JR NZ, LFC0E
260          LD HL, 22272
270          LD B, 4
280 LFC1E    PUSH BC
290          LD B, 32
300 LFC21    CALL LFC2E
310          DJNZ LFC21
320          LD BC, 32
330          ADD HL, BC
340          POP BC
350          DJNZ LFC1E
360          RET
370 LFC2E    PUSH BC
380          LD D, H
390          LD E, L
400          EX DE, HL
410          LD BC, 32
420          ADD HL, BC
430          EX DE, HL
440          LD B, 4
450 LFC39    LD A, (HL)
460          DEC H
470          LD (DE), A
480          DEC D
490          LD (DE), A
500          DEC D
510          DJNZ LFC39
520          EX DE, HL
530          LD BC, 2016
540          ADD HL, BC
550          EX DE, HL
560          LD B, 4
570 LFC49    LD A, (HL)
580          DEC H
590          LD (DE), A
600          DEC D
610          LD (DE), A
620          DEC D
630          DJNZ LFC49
640          LD BC, 2049
```

```

650          ADD HL,BC
660          POP BC
670 ZEND     RET
10 BORDER 0: PAPER 0: INK 7: CLS
20 PRINT # 0; TAB 6; INK 5;"RUTINA DE SCRIERE" ' '
   INK 4; "(3sp)OCUPA 99 OCTETI DE LA adr"'INK 6;
   '"Permite o colorare spectaculoasa"'; INK 3; TAB
   3;"M.M.POPOVICI SOFTWARE 1993"; AT 8,0;:
   RANDOMIZE USR adr

```

Exemplul 8.46: rutina care urmează, deși are o lungime mare comparativ cu restul programelor (610 octeți), realizează un efect de scriere extrem de atractiv și anume litere duble care se rotesc în jurul axului propriu vertical și sînt însoțite de un sunet de atenționare.

```

10          ORG adr          ;LITERE DUBLE ROTITE
                        CU SUNET
20 LD508     LD BC
30 LD 50B    DEC BC
40          LD A,0
50          OUT (254),A
60          LD A,16
70          OUT (254),A
80          LD A,B
90          OR C
100         JR NZ,LD50B
110         RET
120 LD519     LD A,(HL)
130         AND 127
140         CP 32
150         JR Z,LD525
160         PUSH HL
170         CALL LD536
180         POP HL
190 LD525     LD A,(HL)
200         AND 128
210         RET NZ
220         LD DE,(LDC88)
230         INC DE
240         INC DE
250         LD (LDC88),DE
260         INC HL

```

270		JR LD519
280	LD536	LD (LDC8C),A
290		LD B,36
300	LD53B	PUSH BC
310		LD A,37
320		SUB B
330		LD (LDC8E),A
340		CALL LD572
350		LD A,(LDC8D)
360		LD B,A
370		LD A,(LDC8F)
380		ADD A,B
390		CP 6
400		JR Z,LD55A
410		CP 255
420		JR Z,LD566
430		LD (LDC8D),A
440		JR LD56E
450		LD A,5
460		LD (LDC8D),A
470		LD A,255
480		LD (LDC8F),A
490		JR LD56E
500	LD566	XOR A
510		LD (LDC8D),A
520		INC A
530		LD (LDC8F),A
540	LD56E	POP BC
550		DJNZ LD53B
560		RET
570	LD572	LD A,(LDC8C)
580		LD L,A
590		LD H,0
600		ADD HL,HL
610		ADD HL,HL
620		ADD HL,HL
630		LD DE,15360
640		ADD HL,DE
650		LD A,(LDC8D)
660		LD DE,23296
670		AND A

680		JR NZ,LD590
690		LD BC,8
700		LDIR
710		JP LD655,
720	LD590	DEC A
730		JR NZ,LD5C9
740		LD B,8
750	LD595	PUSH BC
760		LD BC,0
770		LD A, (HL)
780		SLA A
790		RL C
800		SLA A
810		RL B
820		SLA A
830		RL C
840		SLA A
850		RL B
860		SLA A
870		RL C
880		SLA A
890		RL B
900		SLA A
910		RL C
920		SLA A
930		RL B
940		LD A,B
950		OR C
960		SLA A
970		SLA A
980		LD (DE),A
990		INC HL
1000		INC DE
1010		POP BC
1020		DJNZ LD595
1030		JP LD655
1040	LD5C9	DEC A
1050		JR NZ,LD5EA
1060		LD B,8
1070	LD5CE	PUSH BC
1080		LD BC,0

1090		LD A, (HL)
1100		AND 15
1110		JR Z, LD5D9
1120		LD, B, 8
1130	LD5D9	LD A, (HL)
1140		AND 240
1150		JR Z, LD5E0
1160		LD C, 16
1170	LD5E0	LD A, C
1180		OR B
1190		LD (DE), A
1200		INC HL
1210		INC DE
1220		POP BC
1230		DJNZ LD5CE
1240		JR LD655
1250	LD5EA	DEC A
1260		JR NZ, LD60C
1270		LD B, 8
1280	LD5EF	PUSH BC
1290		LD A, (HL)
1300		LD BC, 0
1310		AND 15
1320		JR Z, LD5FA
1330		LD B, 16
1340	LD5FA	LD A, (HL)
1350		AND 240
1360		JR Z, LD601
1370		LD C, 8
1380	LD601	LD A, C
1390		OR B
1400		LD (DE), A
1410		INC DE
1420		INC HL
1430		POP BC
1440		DJNZ LD5EF
1450		JP LD655
1460	LD60C	DEC A
1470		JR NZ, LD644
1480		LD B, 8
1490	LD611	PUSH BC

1500		LD BC,0
1510		LD A, (HL)
1520		SRL A
1530		RL C
1540		SRL A
1550		RL B
1560		SRL A
1570		RL C
1580		SRL A
1590		RL B
1600		SRL A
1610		RL C
1620		SRL A
1630		RL B
1640		SRL A
1650		RL C
1660		SRL A
1670		RL B
1680		LD A,B
1690		OR C
1700		SLA A
1710		SLA A
1720		LD (DE),A
1730		INC DE
1740		INC HL
1750		POP BC
1760		DJNZ LD611
1770		JR LD655
1780	LD644	LD B,8
1790	LD646	PUSH BC
1800		LD C, (HL)
1810		LD B,8
1820	LD64A	SLA C
1830		RRA
1840		DJNZ LD64A
1850		LD (DE),A
1860		INC HL
1870		INC DE
1880		POP BC
1890		DJNZ LD646
1900	LD655	LD HL,23296

1910		LD (LDC8A) ,HL
1920		LD B,25
1930	LD650	PUSH BC
1940		LD A, (LDC89)
1950		ADD A,25
1960		SUB B
1970		LD B,A
1980		SRL A
1990		SRL A
2000		SRL A
2010		CALL 3742
2020		LD A,B
2030		AND 7
2040		ADD A,H
2050		LD H,A
2060		LD A, (LDC89)
2070		LD B,0
2080		LD C,A
2090		ADD HL,BC
2100		LD DE, (LDC8A)
2110		POP BC
2120		PUSH BC
2130		LD A,B
2140		AND 3
2150		LD A, (DE)
2160		JR NZ,LD680
2170		INC DE
2180		LD (LDC8A) ,DE
2190	LD68B	LD BC,0
2200		RLCA
2210		RL B
2220		RRCA
2230		RL B
2240		SLA A
2250		RLCA
2260		RL B
2270		RRCA
2280		RL B
2290		SLA A
2300		RLCA
2310		RL B

2320	RRCA
2330	RL B
2340	SLA B
2350	RLCA
2360	RL B
2370	RRCA
2380	RL B
2390	SLA A
2400	RLCA
2410	RL C
2420	RRCA
2430	RL C
2440	SLA A
2450	RLCA
2460	RL C
2470	RRCA
2480	RL C
2490	SLA A
2500	RLCA
2510	RL C
2520	RRCA
2530	RL C
2540	SLA A
2550	RLCA
2560	RL C
2570	RRCA
2580	RL C
2590	LD (HL),B
2600	INC HL
2610	LD (HL),C
2620	POP BC
2630	DJNZ LD650
2640	LD A,(LDC89)
2650	AND 248
2660	LD L,A
2670	LD H,0
2680	SLA L
2690	RL H
2700	SLA L
2710	RL H
2720	LD A,(LDC88)

2730		LD C,A
2740		LD B,88
2750		ADD HL,BC
2760		LD A,(LDC8D)
2770		CP 3
2780		LD A,66
2790		JR NC,LD6F4
2800		LD A,70
2810	LD7F4	LD DE,32
2820		LD (HL),A
2830		INC HL
2840		LD (HL),A
2850		ADD HL,DE
2860		LD (HL),A
2870		DEC HL
2880		LD (HL),A
2890		ADD HL,DE
2900		LD (HL),A
2910		INC HL
2920		LD (HL),A
2930		ADD HL,DE
2940		LD (HL),A
2950		DEC HL
2960		LD (HL),A
2970		LD A,(LDC8E)
2980		SLA A
2990		SLA A
3000		SLA A
3010		LD H,A
3020		LD C,32
3030		LD A,(23560)
3040		CP 226
3045		NOF
3055		NOF
3060		LD C,1
3070	LD71B	LD B,H
3080	LD71C	DJNZ LD71C
3090		XOR A
3100		OUT (254),A
3110		LD B,H
3120	LD722	DJNZ LD722

```

3130          LD A,16
3140          OUT (254),A
3150          DEC C
3160          JR NZ,LD71B
3170          RET
3180 LDC88      DEFB 0
3190 LDC89      DEFB 0
3200 LDC8A      DEFB 6
3210 LDC8B      DEFB6
3220 LDC8C      DEFB 44
3230 LDC8D      DEFB 0
3240 LDC8E      DEFB 157
3250 LDC8F      DEFB 1
3260 TEXT      DEFS 32
3270 PROG      CALL LD508
3300          LD HL,TEXT
3310          CALL LD519
3320          CALL LD508
3330          RET
3340 ZEND      EQU $
3350          ENT PROG

```

```

10 LET START=adr: BORDER 0: PAPER 0: CLS
11 REM LINIA=linia în pixeli,COLOANA=coloana în
    caractere
12 LET LIN=40: LET COL=1: LET AS="MIRCEA POPOVICI":
    GOSUB 100
13 LET LIN=72: LET COL=8: LET AS="'prezinta'";
    GOSUB 100
14 LET LIN=14*8: LET COL=6: LET AS="'COD-MASINA'":
    GOSUB 100
15 STOP
100 FOR I=1 TO LEN AS-1
110 POKE (START+555+I),CODE A$(I)
120 NEXT I
130 POKE (START+555+LEN A$),CODE A$(LEN A$)+128
140 POKE (START+548),COL: POKE (START+549),LIN
150 RANDOMIZE USR (START+588): RETURN

```

Dacă se dorește o citire mai lentă a literelor atunci se vor efectua următoarele modificări în rutina prezentată:

- se scot liniile 3045 și 3055 și se introduce linia

3050

JR NZ,LD71B

-se modifică linia 3210 astfel

3210 LDC8B

DEFB 1291

8.4.3. Alte modalități

Exemplul 8.47: defilarea de jos în sus a unui text.

10	ORG adr	;SCROLL IN SUS TEXT
20	ENT \$	
30	AND A	
40	LD DE,20480	
50	LD B,1	
60	LFF92	PUSH BC
70		LD A,8
80	LFF95	EX AF,AF'
90		LD A,7
100	LFF98	LD H,D
110		LD L,E
120		INC H
130		PUSH HL
140		LD BC,32
150		LDIR
160		POP DE
170		DEC A
180		JR NZ,LFF98
190		LD BC,1792
200		SBC HL,BC
210		LD BC,32
220		LD BC,32
230		LDIR
240		POP DE
250		EX AF,AF'
260		DEC A
270		JR NZ,LFF95
280		LD BC,1760
290		ADD HL,BC
300		LD D,H
310		LD E,L
320		LD BC,32

330		SBC HL,BC
340		EX DE,HL
350		LDIR
360		POP BC
370		DJNZ LFF92
380		LD HL,22496
390		LD B,32
400	LFFCB	LD (HL),A
410		INC HL
420		DJNZ LFFCB
430		RET
440		NOP
450		NOP
460		LD B,D
470		LD B,D
480		LD A,H
490		LD B,B
500		LD B,B
510		NOP
520		NOP
530		INC A
540		LD B,D
550		LD B,D
560		LD D,D
570		LD C,D
580		INC A
590		NOP
600		NOP
610		LD A,H
620		LD B,D
630		LD B,D
640		LD A,H
650		LD B,H
660		LD B,D
670		NOP
680		NOP
690		INC A
700		LD B,B
710		INC A
720		LD (BC),A
730		LD B,D

```

740          INC A
750          NOP
760          NOP
770          CP 16
780          DJNZ X0
790          DJNZ X1
800          NOP
810          NOP
820          LD B,D
830 X0       LD B,D
840          LD B,D
850 X1       LD B,D
860          LD B,D
870 ZEND     RET

```

```

10 BORDER 0: PAPER 0: INK 7: CLS
15 FOR N=8 TO 15: PRINT AT N,4; INK 2; PAPER 6;"#";
   AT N,26"#": NEXT N
20 PRINT AT 10,7; INK 5;"RUTINA DE SCRIERE"; AT
   12,9; "DE JOS IN SUS"
25 FOR N=3 TO 13: PRINT AT 8,N*2-2; INK 2; PAPER
   6;"#"; AT 15,N*2-2;"#": NEXT N
30 RESTORE 240
50 FOR w=1 TO 11
60 PRINT AT 21,0; INK 0; PAPER 0; , , ,
70 READ A$:IF A$="STOP" THEN GO TO 230
80 FOR A=1 TO LEN A$:PRINT AT 21,A;A$(A); INK 5;
   "*" : IF A$(A) CHR$ 32 THEN BEEP .003,5: BEEP
   .005,10: BEEP .003,5
90 NEXT A: PRINT AT 21,A;" "
95 LET adr=60000
100 FOR N=1 TO 10: RANDOMIZE USR adr: NEXT N: NEXT w
110 STOP
240 DATA "VA SALUT!!",',',',',',',"ZX-SPECTRUM PREZINTA",
   "EFECTUL VIZUAL",',"SCRIERE SCROLL JOS IN SUS"',
   "REALIZAT DE",',"M.M.POPOVICI"
250 DATA "STOP"

```

Exemplul 8.48: scriere tip "șenilă" cu litere duble pe linia 21 (roll)

```

10          ORG adr          ; SCRIERE ROLL
          LITERE DUBLE PE
          LINIA 21

```

```
20          ENT $
30 L9C40    LD HL,3000
40          LD (L9CB4),HL
50          LD A,2
60          CALL 5633
70          XOR A
80          LD (23168),A
90          LD (23200),A
100         LD (23231),A
110        LD (23199),A
120 9C58    CALL L9C75
130        CALL L9CFA
140        LD HL,(L9C4B)
150        LD A,(HL)
160        CP 127
170        JP Z,L9C40
180        CP 142
190        JP Z,L9CEB
200        LD A,(23560)
210        CP 32
220        RET Z
230        JP L9C58
240 L9C75   LD A,16
250        RST 16
260        XOR A
270        RST 16
280        LD A,17
290        RST 16
300        XOR A
310        RST 16
320        LD HL,(L9CB4)
330        LD A,(HL)
340        CALL L9CB6
350        LD HL,(L9CB4)
360        INC HL
370        LD (L9CB4),HL
380        RET
390 L9CBE   LD DE,20640
400        LD B,8
410 L9C93   PUSH DE
420        PUSH BC
```

430		LD B,31
440		LD A, (HL)
450		RL A
460		RL A
470		LD (DE), A
480		INC DE
490	L9C9E	LD A, (DE)
500		LD L, A
510		LD H, 0
520		ADD HL, HL
530		ADD HL, HL
540		DEC DE
550		LD A, (DE)
560		XOR H
570		LD (DE), A
580		INC DE
590		LD A, L
600		LD (DE), A
610		INC DE
620		DJNZ L9C9E
630		POP BC
640		POP DE
650		INC D
660		DJNZ L9C93
670		RET
680	L9CB4	LD E, A
690		LD (HL), L
700	L9CB6	LD H, 0
710		LD L, A
720		ADD HL, HL
730		ADD HL, HL
740		ADD HL, HL
750		LD DE, (23606)
760		ADD HL, DE
770		LD DE, (23675)
780		LD B, 16
790	L9CC7	LD A, (HL)
800		LD (DE), A
810		INC DE
820		XOR5 A
830		LD (DE), A

```
840          INC HL
850          INC DE
860          DJNZ L9CC7
870          LD DE, (23675)
880          LD HL, 20639
890          LD B, 8
900 L9CD9    LD A, (DE)
910          LD (HL), A
920          INC DE
930          INC H
940          DJNZ L9CD9
950          LD HL, 20671
960          LD B, 8
970 L9CE4    LD A, (DE)
980          LD (HL), A
990          INC DE
1000         INC H
1010         DJNZ L9CE4
1020         RET
1030 L9CEB    LD B, 100
1040 L9CED    HALT
1050         DJNZ L9CED
1060         LD HL, (L9CB4)
1070         INC HL
1080         LD (L9CB4), HL
1090         JP L9C58
1100 L9CFA    LD B, 4
1110 L9CFC    HALT
1120         LD DE, 20608
1130         PUSH BC
1140         LD B, 8
1150 L9D03    PUSH DE
1160         PUSH BC
1170         LD B, 31
1180         LD A, (DE)
1190         RL A
1200         RL A
1210         LD (DE), A
1220         INC DE
1230 L9D0E    LD A, (DE)
1240         LD L, A
```

```

1250          LD H,0
1260          ADD HL,HL
1270          ADD HL,HL
1280          DEC DE
1290          LD A,(DE)
1300          XOR H
1310          LD (DE),A
1320          INC DE
1330          LD (DE),A
1340          INC DE
1350          DJNZ L9D0E
1360          POP BC
1370          INC D
1380          DJNZ L9D03
1390          CALL L9C8E
1400          POP BC
1410          DJNZ L9CFC
1420 ZEND      RET

```

```

10 BORDER 0: PAPER 0: INK 7: CLS
20 LET a$="Textul este introdus de la adresa 30000"
   +CHR$ 142"+7sp M.M.POPOVICI SOFTWARE"+CHR$
   142+"6spApasati SPACE pentru revenire în BASIC
   18spRUTINA ARE :234 OCTETI ! ! 4sp"+CHR$
   142+"8sp05.01.198615sp "+CHR$ 127: REM sp=
   blanc; 142=oprire temporara; 127=mesajul se
   repeta; SPACE=revenire la urmatoarea linie BASIC
30 GOSUB 100
40 PAUSE 0: STOP
100 FOR n=1 TO LEN a$: POKE 29999+n, CODE a$(n TO
   n): NEXT N: RETURN

```

Exemplul 8.49: dublarea caracterelor într-o fereastră (box)

```

10          ORG 56700          ;2*SCREEN IN BOX
20          ENT $
30          LD HL,(23563)
40          LD BC,4
50          ADD HL,BC
60          LD D,(HL)
70          LD C,8
80          ADD HL,BC
90          LD (LDE89),DE

```

```
110          ADD HL,BC
120          LD A,(HL)
130          LD (LDE8C),A
140          ADD HL,BC
150          LD A,(HL)
160          LD (LDE8B),A
170          LD A,(LDE8A)
180          LD B,A
190          LD A,(LDE8C)
200          ADD A,B
210          AND 224
220          JR Z,LDDA8
230          LD A,31
240          SUB B
250          LD (LDE8C),A
260 LDDA8    LD A,(LDE89)
270          LD B,A
280          LD A,(LDE8B)
290          ADD A,B
300          SUB 22
310          JR C,LDDBA
320          LD A,21
330          SUB B
340          LD (LDE8B),A
350 LDDBA    LD DE,(LDE89)
360          LD A,E
370          AND 24
380          OR 64
390          LD H,A
400          LD A,E
410          AND 7
420          OR A
430          RRA
440          RRA
450          RRA
460          RRA
470          ADD A,D
480          LD L,A
490          LD (LDE8D),HL
500          LD DE,16384
510          AND A
```

```
520          SBC HL,DE
530          LD DE,57344
540          ADD HL,DE
550          LD (LDE8F),HL
560          CALL LDE71
570          LD HL,(LDE8D)
580          LD DE,(LDE8F)
590          LD A,(LDE8B)
600          LD B,A
610  LDDEC   PUSH BC
620          LD BC,1026
630  LDDFO   POP BC
640          CALL LDE14
650          POP BC
660          DJNZ LDDFO
670          LD HL,(LDE8D)
680          CALL LDE4F
690          LD (LDE8D),HL
700          LD B,4
710          DEC C
720          JR NZ,LDDFO
730          LD DE,(LDE8F)
740          CALL LDE59
750          LD (LDE8F),DE
760          POP BC
770          DJNZ LDDEC
780          RET
790  LDE14   LD A,(LDE8C)
800          LD B,A
810          LD (LDE91),HL
820          LD (LDE93),DE
830  LDE1F   PUSH BC
840          CALL LDE36
850          POP BC
860          DJNZ LDE1F
870          LD HL,(LDE91)
880          LD DE,(LDE93)
890          PUSH HL
900          CALL LDE63
910          POP HL
920          INC H
```


930		INC H
940		INC D
950		RET
960	LDE36	LD A, (DE)
970		LD BC, 1026
980	LDE3A	PUSH BC
990		PUSH AF
1000		XOR A
1010		LD (HL), A
1020		POP AF
1030	LDE3F	RLA
1040		PUSH AF
1050		RL (HL)
1060		POP AF
1070		RL (HL)
1080		DJNZ LDE3F
1090		INC HL
1100		POP BC
1110		DEC C
1120		JR NZ, LDE3A
1130		INC DE
1140		RET
1150	LDE4F	LD A, 32
1160		ADD A, L
1170		LD L, A
1180		RET NC
1190		LD A, 8
1200		ADD A, H
1210		LD H, A
1220		RET
1230	LDE59	LD A, 32
1240		ADD A, E
1250		LD E, A
1260		RET NC
1270		LD A, 8
1280		ADD A, D
1290		LD D, A
1300		RET
1310	LDE63	LD A, (LDE8C)
1320		SLA A
1330		LD B, A

```
1340 LDE69      LD A, (HL)
1350            INC H
1360            LD (HL), A
1370            DEC H
1380            INC HL
1390            DJNZ LDE69
1400            RET
1410 LDE71      LD HL, 16384
1420            LD DE, 57344
1430            LD BC, 6656
1440            LDIR
1450            RET
1460            LD HL, 57334
1470            LD DE, 16384
1480            LD BC, 6656
1490            LDIR
1500            RET
1510 LDE89      NOP
1520 LDE8A      NOP
1530 LDE8B      LD B, 16
1540 LDE8D      ADD A, B
1550            LD C, B
1560 LDE8F      RET NZ
1570            RET PO
1580 LDE91      LD H, B
1590            LD C, (HL)
1600 LDE93      AND B
1610 ZEND      RET
1620 LDE8C      NOP
```

```
10 CLS : DEF FN q(y,x,u,v)=USR 56700: REM
   y,x=coordonatele coltului stînga sus al box-
   ului (coloana,linie); u=lungime box; v=înaltime
   box (de 2 ori mai mici)
20 FOR n=0 TO 21: PRINT AT n,0;"32*": REM de 32 de
   ori caracterul *
30 PAUSE 50: RANDOMIZE FN q(4,7,12,4): REM box la
   mijlocul ecranului (coloana 4, linia 7, lungime
   12*2, înaltime 4*2)
```

Programul de mai sus mărește de 2 ori o fereastră la mijlocul ecranului; evident o astfel de rutină este utilă pentru mărirea unor diagrame/scheme pe o porțiune a ecranului. Pentru a se mări întreg ecranul comanda va fi

RANDOMIZE FN q(0,0,16,11)

De pildă, pentru scrierea de titluri programul BASIC va fi:

```
10 BORDER 2: PAPER 1: INK 6: CLS
20 DEF FN q(y,x,u,v)=USR 56700
30 PRINT AT 8,10;"M.M.POPOVICI"; AT 10,9;"software
  1993"; AT 13,11;"COD-MASINA"
40 RANDOMIZE FN q(0,0,16,11)
```

Exemplul 8.50: roll caractere qvadruple în paralel cu caractere normale

10	ORG adr	;ROLL CARACTERE QVADRUPLE
20	ENT \$	
30 ET1	EQU \$+37	
40 ET2	EQU \$+140	
50 ET3	EQU \$+43	
60 ET4	EQU \$+42	
70	LD DE,65504	
80	LD A,(LF34B)	
90	LD L,A	
100	LD A,(LF34C)	
110	LD H,A	
120	LD B,8	
130 LF23D	LD A,(DE)	
140	LD (HL),A	
150	INC DE	
160	INC H	
170	DJNZ LF23D	
180	LD A,(LF34B)	
190	INC A	
200	LD L,A	
210	LD A,(LF34C)	
220	LD H,A	
230	LD B,8	
240 LF24E	LD A,(DE)	
250	LD (HL),A	
260	INC DE	

```
270          INC H
280          DJNZ LF24E
290          LD B,2
300          PUSH BC
310          LD D,64
320 LF259    LD HL,18176
330          LD A,D
340          LD (ET3),A
350          LD C,20
360          SLA (HL)
370          JR NC,LF268
380          LD C,0
390 LF268    INC HL
400          LD B,31
410 LF26B    SLA(HL)
420          JR NC,LF272
430          DEC HL
440          INC HL
450          INC HL
460 LF272    INC HL
470          DJNZ LF26B
480          DEC HL
490          LD A,C
500          CP 10
510          JR NC,LF27C
520          INC (HL)
530 LF27C    LD A,D
540          INC A
550          LD D,A
560          LD A,D
570          CP 72
580          JR NZ,LF259
590          LD A,32
600          LD (ET4),A
610          POP BC
620          DJNZ ET1
630          LD A,0
640          LD (ET4),A
650          LD HL,16433
660          LD DE,20511
670          LD B,8
```

680	LF299	LD A, (HL)
690		CP 128
700		JR C, LF2AD
710		PUSH BC
720		LD A, 170
730		LD B, 8
740	LF2A3	LD (DE), A
750		INC D
760		DJNZ LF2A3
770		LD B, 8
780	LF2A9	DEC D
790		DJNZ LF2A9
800		POP BC
810	LF2AD	INC H
820		LD A, E
830		LD E, 32
840		ADD A, E
850		LD E, A
860		DJNZ LF299
870		LD HL, 16401
880		LD DE, 18462
890		LD B, 8
900		LD A, (HL)
910		CP 127
920		JR C, LF2DF
930		PUSH BC
940		PUSH HL
950		LD HL, 65520
960		LD B, 8
970	LF2C9	LD A, (HL)
980		LD (DE), A
990		INC D
1000		INC HL
1010		DJNZ LF2C9
1020		LD B, 8
1030	LF2D1	DEC D
1040		DJNZ LF2D1
1050		INC E
1060		LD B, 8
1070	LF2D7	LD A, (HL)
1080		LD (DE), A

1090		INC D
1100		INC HL
1110		DJNZ LF2D7
1120		POP HL
1130		POP BC
1140	LF2DF	INC H
1150		LD A,E
1160		LD E,32
1170		ADD A,E
1180		LD E,A
1190		DJNZ ET2
1200		LD HL,500
1210		LD DE,1
1220		LD B,19
1230	LF2EF	PUSH HL
1240		PUSH DE
1250		PUSH BC
1260		CALL 949
1270		POP BC
1280		POP DE
1290		POP HL
1300		DEC HL
1310		DEC HL
1320		DEC HL
1330		DJNZ LF2EF
1340		LD DE,LF34D
1350		LD A,(LF34B)
1360		LD L,A
1370		LD A,(LF34C)
1380		LD H,A
1390		LD B,8
1400	LF309	LD A,(DE)
1410		LD (HL),A
1420		INC DE
1420		INC DE
1430		INC H
1440		DJNZ LF309
1450		LD A,(LF34B)
1460		INC A
1470		LD L,A
1480		LD A,(LF34C)

```
1490          LD H,A
1500          LD B,8
1510 LF31A    LD A,(DE)
1520          LD (HL),A
1530          INC DE
1540          INC H
1550          DJNZ LF31A
1560          LD HL,18433
1570          LD B,128
1580 LF325    PUSH BC
1590          LD B,31
1600 LF328    LD A,(HL)
1610          DEC HL
1620          LD (HL),A
1630          INC HL
1640          INC HL
1650          DJNZ LF328
1660          DEC HL
1670          LD (HL),0
1680          INC HL
1690          INC HL
1700          POP BC
1710          DJNZ LF325
1720          LD A,(LF34B)
1730          INC A
1740          LD L,A
1750          LD A,(LF34C)
1760          LD H,A
1770          LD A,(HL)
1780          CP 0
1790          JR Z,LF34A
1800          LD A,255
1810          LD (LF34E),A
1820 LF34A    RET
1830 LF34B    NOP
```

10 BORDER 0: PAPER 0: INK 7: CLS

20 PRINT AT 1,0;"PROGRAME IN LIMBAJ DE ASAMBLARE"

30 RANDOMIZE USR adr

40 IF INKEY\$="" THEN GO TO 30

Observatii: 1) Scotînd linia 1260 efectul are loc fără sunet.
 2) Rutina poate fi folosită în jocuri; de pildă deplasarea unui cartier de blocuri se realizează cu următorul program BASIC:

```

10 RESTORE 20: FOR f=USR "a" TO USR "p"+7: READ a:
   POKE f,A: NEXT f
20 DATA 24,24,255,255,189,189,255,255,0,248,216,
   255,223,253,223,255
30 DATA 0,0,0,0,0,0,85,255,0,0,3,15,63,255,255,255
40 DATA 60,60,255,255,85,255,85,255,0,0,3,15,13,63,
   53,255
50 DATA 0,255,129,255,129,255,129,255,255,255,165,
   255,165,255,165,255
60 DATA 16,16,16,16,16,19,255,255,0,0,0,7,31,149,
   255,255
70 DATA 231,255,165,231,165,231,165,255,24,60,24,
   60,24,255,255,255
80 DATA 0,24,24,60,255,219,255,255,0,15,15,13,253,
   255,181,255
90 DATA 0,224,248,168,248,168,255,170,0,0,0,118,84,
   126,76,76
100 BORDER 0: PAPER 0: INK 7: CLS : PLOT 0,159: DRAW
   255,0: PLOT 0,158: DRAW 255,0
110 FOR f=0 TO 31: PRINT AT 1,f; CHR$(INT(RND*16)
   +144): NEXT f: REM deşenarea blocurilor de
   locuinte
120 FOR f=0 TO 20: LET a=INT (RND*256): LET b=INT
   (RND*8)+167: PLOT a,b: NEXT f: REM desenarea
   stelelor pe cer
130 FOR f=0 TO 1: PRINT AT f,0; INK 5; PAPER 1;
   BRIGHT 1; OVER 1;"31sp"; AT f,13; PAPER 2; INK
   7;"3sp": NEXT f: REM cerul;sp=blanc
140 LET mc=USR adr: IF INKEY$="" TEHN GO TO 140

```


9. DEZASAMBLORUL MONS3M21

9.1. INTRODUCERE

MONS3M21 este un program în cod mașină avînd 6068 octeți și care este relocabil (adică se poate încărca la adresa dorită). Dacă se dorește reintrarea în **MONS**, după revenirea în *BASIC*, adresa de intrare trebuie să fie cu 29 mai mare decît adresa inițială. De exemplu, presupunînd că **MONS** va fi încărcat după împrejurări la o adresă joasă (ex.26000) sau înaltă (ex.55000), comenzile de încărcare și lansare în cele două cazuri vor fi:

LOAD "MONS3M21" CODE 26000: RANDOMIZE USR 26000

respectiv

LOAD "MONS3M21" CODE 55000: RANDOMIZE USR 55000

Pentru a reintra în **MONS** se vor folosi comenzile

RANDOMIZE USR 26029 respectiv **RANDOMIZE USR 55029**

Odată lansat, **MONS** afișează mesajul:

*** MONS3 copyright Hisoft 1983 ***

care va fi înlocuit, după cîteva secunde, de un panou frontal. Acest panou constă din registrele și fanioanele microprocesorului Z80, afișate împreună cu conținutul lor, precum și o zonă de 34 octeți avînd centrul marcat cu caracterele ">" și "<" în dreptul valorii curente a *indicatorului de memorie* (inițial valoarea acestuia este 0). Prima linie de sus de pe ecran reprezintă dezasamblarea instrucțiunii arătate de indicatorul de memorie.

La intrarea în **MONS** adresele afișate în cadrul panoului frontal sînt date în *hexazecimal* (care este sistemul de numerație folosit de **MONS**), dar se poate trece în sistemul zecimal prin comanda **SYMBOL SHIFT 3**.

Se menționează că adresele codului de dez asamblat trebuie introduse obligatoriu în hexazecimal.

Comenzile se tastează ca răspuns la propterul ">" afișat sub display-ul de memorie, atît cu litere minuscule cît și cu litere majuscule. Unele comenzi, al căror efect poate fi distructiv în cazul folosirii incorecte, necesită apăsarea tastei SYMBOL SHIFT (SS) odată cu litera corespunzătoare comenzii.

În cele ce urmează SYMBOL SHIFT și CAPS SHIFT se vor abrevia SS și respectiv CS. De exemplu "SYMBOL SHIFT Z" este echivalent cu "SS și Z".

Comenzile sînt executabile imediat nefiind necesară apăsarea tastei CR (ENTER), iar comenzile nevalabile sînt ignorate. Întregul panou frontal este reîmprospătat după fiecare comandă executată, permițînd observarea rezultatelor comenzii respective.

Întoarcerea în BASIC se face prin comanda CAPS SHIFT 1 (CS și 1).

9.2. COMENZILE MONS

Principalele comenzi ale dez asamblorului MONS3M21 sînt indicate în tabelul următor.

Nr.crt.	Comanda	Ce se realizează
1	SS și 3	Determină afișarea adreselor în zecimal (repetarea comenzii SS și 3 cauzează revenirea la sistemul hexazecimal).
2	SS și 4	Afișează o pagină de dez asamblare. Pentru continuarea vizualizării listing-ului se apasă o tastă oarecare, iar repetarea comenzii SS și 4 cauzează revenirea la panoul frontal.
3	CR (ENTER)	Incrementează indicatorul de memorie cu 1
4	CS și 7	Decrementează indicatorul de memorie cu 1
5	CS și 3	Decrementează indicatorul de memorie cu 8
6	CS și 8	Incrementează indicatorul de memorie cu 8
7	Virgula (,)	Reîmprospătează indicatorul de memorie să conțină adresa curentă din stivă dată de SP

8	G	<p>Caută în memorie un string specificat. Apare prompterul ":" la care se răspunde prin introducerea primului octet urmat de CR. De aici se continuă introducerea de octeți ca răspuns la delimitatorul ":" pînă la definirea string-ului căutat. Dacă string-ul este identificat, panoul frontal va fi reîmprospătat, indicatorul de memorie fiind poziționat pe primul caracter al string-ului.</p> <p>Pentru a determina noi apariții ale string-ului se utilizează comanda "n".</p>
9	h	<p>Convertește un număr zecimal în hexazecimal (ca răspuns la prompterul ":")</p>
10	l	<p>Produce o copie inteligentă (în sensul că permite copierea într-o zonă care se suprapune peste cea originală). Comanda solicită adresele de început (First:) și de sfîrșit (Last:) ale zonei de memorie ce trebuie copiată, precum și adresa la care trebuie copiată (To:). Toate adresele se specifică în hexazecimal.</p>
11	J	<p>Execută programul în cod mașină de la adresa tastată în hexazecimal ca răspuns la prompterul ":". Dacă se preconizează o întoarcere la panoul frontal după execuția programului, atunci se va fixa un "punct de oprire" (breakpoint)-v.comanda W.</p>
12	SS și k	<p>Continuă execuția programului de la adresa conținută în contorul program PC. De regulă această comandă este folosită în conjuncție cu comanda W.</p>
13	L	<p>Listează un bloc de memorie începînd cu adresa curentă conținută de indicatorul de memorie. Adresele vor fi date în hexazecimal sau zecimal în funcție de starea curentă a panoului frontal (v.SS și 3).</p>

14	M	Fixează indicatorul de memorie la valoarea tastată în hexazecimal ca răspuns la prompterul ":"
15	n	Caută următoarea apariție a string-ului specificat cu comanda G .
16	P	Umple memoria cu un octet dat între limite specificate. Ca răspuns la prompturile (First:), (Last:) și (Whit:) se vor tasta în hexazecimal adresele de început și de sfârșit ale blocului de memorie ce trebuie umplut, respectiv octetul cu care se va umple blocul.
17	Q	Interschimbă seturile de registre (cel standard AF,BC,DE,HL cu cel alternativ AF', BC', DE', HL'). Repetarea comenzii cauzează revenirea la afișarea setului standard.
18	SS și T	Fixează un breakpoint după instrucțiunea curentă și continuă execuția.

19 T

Dezasamblează un bloc de cod mașină (opțional pe imprimantă). La apariția prompterului (First:) și respectiv (Last:) se vor tasta în hexazecimal adresele de început și de sfârșit referitoare la codul mașină de dezasamblat. La mesajul (Printer?) se va răspunde cu Y (majusculă) în cazul când se dorește dezasamblarea pe imprimantă (orice altă tastă determină dezasamblarea pe ecran). În continuare se afișează (Text:) care cere adresa în hexazecimal a filei de text pe care o va produce dezasamblorul (acceptată de GENS3M21). Apoi apare mesajul (Workspace:) care cere adresa de început a memoriei rezervate pentru tabela de simboluri. Răspunzând cu CR adresa va fi 24576 (#6000). Programul va cere în mod repetat adresele de început (First:) și de sfârșit (Last:) ale tuturor cîmpurilor de date existente în blocul de memorie. De regulă se tastează CR la fiecare mesaj. La terminarea dezasamblării se afișează listing-ul dezasamblat (care poate fi oprit cu CR sau SPACE) și lungimea codului rezultat (End of text XXXXX unde XXXXX este adresa de sfârșit a filei de text). Etichetele sînt generate în codul dezasamblat sub forma LYYYY unde YYYYY este adresa absolută a etichetei dacă aceasta este cuprinsă între limitele dezasamblării; dacă adresa este în afara blocului dezasamblat ea va fi indicată în zecimal sau hexazecimal fără litera L. În cazul când se întîlnește un cod invalid, acesta se va dezasambla ca o instrucțiune NOP marcată cu un asterisc (*).

20	W	Fixează un breakpoint la adresa dată de indicatorul de memorie (în prealabil se va fixa acest indicator pe adresa dorită cu comanda M).
21	SS și 7	Execuția pas cu pas.
22	SS și P	Listare la imprimantă.

9.3. ALGORITMUL DE LUCRU

- 1) Se încarcă **MONS3M21** la adresa dorită (adr), eventual cu **CLEAR** (adr-1)
- 2) Se încarcă codul mașină de dezasamblat.
- 3) Se activează **MONS** cu comanda **RANDOMIZE USR adr** (iar dacă se dorește a se lucra în zecimal se tastează **SS și 3**).
- 4) Se află valorile în hexazecimal al adreselor de început și sfârșit ale codului mașină, folosind comanda **h**.
- 5) Se tastează **T** și apoi la mesajele (First:) și (Last:) se introduc adresele de început și de sfârșit ale codului de dezasamblat. La mesajul (**Printer?**) se va răspunde după dorință (o tastă oarecare dacă nu se dorește dezasamblarea la imprimantă, respectiv **Y** în caz contrar). La mesajul (**Text:**) se introduce adresa în hexazecimal a filei de text pentru **GENS3M21** (adr 1).
La restul mesajelor se va tasta de fiecare dată **CR**.
- 6) La terminarea dezasamblării se va nota lungimea codului rezultat, după care se tastează **SS și 4** (pentru revenire la panoul frontal) și apoi **CS și 1** (pentru a se reveni în **BASIC**). În continuare se salvează codul cu comanda **SAVE "nume" CODE adr1, lungime**.
- 7) Se resetează calculatorul, se încarcă **GENS2M21** și apoi se lucrează cu **GENS**-ul după modelul indicat la capitolul 2.

9.4. EXEMPLE DE DEZASAMBLARE

Exemplele care urmează au rezultat în urma dezasamblării unor blocuri în cod mașină; ele reprezintă rutine utile pentru programele

proprii.

Exemplul 9.1.

```

10          ORG 60899          ;PROTECTIE ANTI-
                                BREAK
120         ENT $
130         CALL 124
140         DEC SP
150         DEC SP
160         POP HL
170         LD BC,15
180         ADD HL,BC
190         EX DE,HL
200         LD HL,(23613)      ;(ERR SP)
210         LD (HL),E
220         INC HL
230         LD (HL),D
240         RET
250         HALT
260  LEDF6   CALL 654
270         LD A,E
280         CP 255
290         JR NZ,LEDF6
300         LD A,(23610)      ;(ERR NR)
310         CP 12
320         JR Z,LEE0F
330         CP 16
340         JR Z,LEE0F
350         CP 20
360         JR Z,LEE0F
370         JR LEE28
380  LEE0F   INC A
390         LD (23681)A      ;(23681) nefolosit
400         LD (IY+0),255
410         LD HL,7500
420         LD (23618),HL   ;(23618)=(NEWPC)
430         LH HL,0
440         LD (23620),HL   ;(NSPPC)
450         DEC SP
460         DEC SP
470         JP 7037
480  LEE28   JP 4867

```

Rutina se exploatează cu următorul program BASIC:

```

6 LET a=60899: LET b=7500 :REM b=nr.liniei unde se
  va transfera programul cînd se tasteaza BREAK
7 POKE (a+53),b-256*INT(b/256): POKE (a+54),INT
  (b/256): REM in locatiile de memorie (a+53) si
  (a+54) se scrie numarul liniei b=7500

```

```

8 RANDOMIZE USR a
9 CLS : FOR i=32 TO 255: PRINT CHR$ i: NEXT i: REM
  se va tasta BREAK în timp ce se afiseaza
  caracterele
10 STOP
7500 LET t$="13sp": PAPER 6: BRIGHT 1: INK 0: BORDER
  7: CLS : PLOT 0,0: DRAW 255,0: DRAW 0,175: DRAW
  -255,0: DRAW 0,-175
7505 PRINT AT 1,2; PAPER 5;t$;t$(2 TO): FOR i=2 TO
  20: PRINT AT i,1; PAPER 7;t$(2 TO); AT i-1,30:
  PAPER 5:"": NEXT i
7510 OVER 1: FOR f=72 TO 79: POKE 23681,f: LPRINT TAB
  3;"M.M.POPOVICI SOFTWARE 1993": NEXT f
7520 PAUSE 20: PRINT AT 20,1; PAPER 7;"Parola:"
7530 INPUT LINE p$
7540 IF p$="MIRCEA" THEN STOP
7550 PRINT AT 20,8; PAPER 0; INK 7; "NU ! ! !": BEEP
  .02,40: PAUSE 4: BEEP .02,-40: GO TO 7500

```

Exemplul 9.2

```

10          ORG 62200          ; LITERE DUBLE
10          ENT $
30 ET1      EQU $+248
40 ET2      EQU $+300
50 ET3      EQU $+252
60 ET4      EQU $+250
70 ET5      EQU $+249
80 ET6      EQU $+246
90          LD HL,(23563)
100         LD BC,4
110         ADD HL,BC
120         LD D,(HL)
130         LD BC,8
140         ADD HL,BC
150         LD E,(HL)
160         LD (ET1),DE
170         LD A,99
180         LD B,A
190         LD HL,ET2
200         LD (ET3),HL
210 LF312   PUSH BC
220         LD DE,(ET1)
230         LD A,30
240         CP D
250         JP P,LF325
260         LD D,0
270         INC E
280         INC E

```



```
290          LD (ET1),DE
300 LF325    LD A,20
310          CP E
320          JP M,LF36F
330          LD HL,(ET3)
340          LD A,(HL)
350          INC HL
360          LD (ET3),HL
370          CP 31
380          JP M,LF36F
390          CP 144
400          JP P,LF36F
410          SUB 32
420          LD BC,8
430          LD HL,(23606)
440          INC H
450 LF346    ADD HL,BC
460          DEC A
470          JR NZ,LF346
480          LD (ET4),HL
490          LD A,E
500          AND 24
510          OR 64
520          LD H,A
530          LD A,E
540          AND 7
550          OR A
560          RRA
570          RRA
580          RRA
590          RRA
600          ADD A,D
610          LD L,A
620          LD (ET6),HL
630          CALL LF371
640          LD A,(ET5)
650          INC A
660          INC A
670          LD (ET5),A
680          POP BC
690          DJNZ LF312
700          RET
710 LF36F    POP BC
720          RET
730 LF371    LD DE,LF3CE
740          LD B,32
750          LD A,0
760 LF378    LD (DE),A
```

770		INC DE
780		DJNZ LF378
790		LD DE, (ET4)
800		LD HL, LF3CE
810		LD B, 8
820	LF385	PUSH BC
830		LD A, (DE)
840		LD BC, 1026
850	LF38A	PUSH BC
860	LF38B	RLA
870		PUSH AF
880		RL (HL)
890		POP AF
900		RL (HL)
910		DJNZ LF38B
920		INC HL
930		POP BC
940		DEC C
950		JR NZ, LF38A
960		DEC HL
970		LD A, (HL)
980		PUSH AF
990		DEC HL
1000		LD A, (HL)
1010		INC HL
1020		INC HL
1030		LD (HL), A
1040		INC HL
1050		POP AF
1060		LD (HL), A
1070		INC HL
1080		INC DE
1090		POP BC
1100		DJNZ LF385
1110		LD HL, (ET6)
1120		LD DE, LF3CE
1130		LD C, 2
1140	LF3B1	PUSH HL
1150		LD B, 8
1160	LF3B4	LD A, (DE)
1170		LD (HL), A
1180		INC HL
1190		INC DE
1200		LD A, (DE)
1210		LD (HL), A
1230		INC DE
1240		DEC HL
1250		INC H

```

1260          DJNZ LF3B4
1270          POP HL
1280          LD A,32
1290          ADD A,L
1300          LD L,A
1310          JR NC,LF3CA
1340          LD A,8
1350          ADD A,H
1360          LD H,A
1370 LF3CA    DEC C
1380          JR NZ,LF3B1
1390          RET
1400 LF3CE    NOP

```

Următorul program BASIC pune în valoare rutina anterioară:

```

10 DEF FN d(x,y)=USR 62200
20 BORDER 2: PAPER 0: INK 6: CLS
30 LET n$="M.M.POPOVICI": GOSUB 100: RANDOMIZE FN
  d(4,8)
40 LET n$="COD-MASINA": GOSUB 100: RANDOMIZE FN
  D(6,14)
50 STOP
100 LET L=LEN n$: LET k=62499: FOR I=1 TO L: LET
  n=CODE n$(I): POKE k+i,n: NEXT i: POKE k+i,13:
  RETURN

```

Se definește funcția DEF FN d(x,y)=USR 62200 unde x,y=poziția PRINT (cu $x < 32$ și $y < 24$), în care x-coloana și y-linia. Textul este POKE - at ca un sir (n\$) în memorie la adresa 62500 (poate fi stocat un număr maxim de 16 caractere).

Exemplul 9.3

```

10          ORG 60000          ;COMPACTARE SCREEN$
20          ENT $
30          PUSH IX
40          PUSH IY
50          DI
60          LD IY,50002
70          LD IX,16384
80          LD DE,1
90          LD HL,1
100         LD A,(IX+0)
110         LD BC,6911
120 LFF45   CP (IX+1)
130         JR Z,LFF82
140 LFF4A   PUSH AF
150         LD A,D
160         CP 0
170         JR Z,LFF66

```

```
180          LD (IY+0),255
190          POP AF
200          LD (IY+1),A
210          INC IY
220          PUSH IY
230          PUSH BC
240          LD B,255
250 LFF5F    DEC DE
260          DJNZ LFF5F
270          POP BC
280          INC HL
290          JR LFF4A
300 LFF66    LD A,E
310          CP 0
320          JR Z,LFF79
330          POP AF
340          LD (IY+0)E
350          LD (IY+1),A
360          INC HL
370          INC IY
380          INC IY
390          JR LFF7A
400 LFF79    POP AF
410 LFF7A    LD A,(IX+1)
420          LD DE,1
430          JR LFF83
440 LFF82    INC DE
450 LFF83    INC IX
460          PUSH BC
470          PUSH AF
480          LD A,B
490          OR C
500          CP 0
510          JR Z,LFF90
520          POP AF
530          JR LFF45
540 LFF90    LD A,D
550          CP 0
560          JR Z,LFFAC
570          LD (IX+0),255
580          POP AF
590          LD (IY+1),A
600          INC IY
610          PUSH IY
620          PUSH BC
630          LD B,255
640 LFFA4    DEC DE
650          DJNZ LFFA4
```

```
660          POP BC
670          INC HL
680          PUSH AF
690          JR LFF90
700 LFFAC    LD A,E
710          CP 0
720          JR Z,LFFBB
730          POP AF
740          LD (IY+0),E
750          LD (IY+1),A
760          INC HL
770          JR LFFBFC
780 LFFBB    POP AF
790 LFFBFC   LD IX,50000
800          LD (IX+0),L
810          LD (IX+1),H
820          EI
830          POP IY
840          POP IX
850          RET
860          PUSH AF
870          PUSH BC
880          PUSH DE
890          PUSH HL
900          PUSH IX
910          PUSH IY
920          LD IX,50002
930          LD BC,(50000)
940          LD HL,16384
950          DEC BC
960 LFFE0    PUSH BC
970          LD B,(IX+0)
980          LD A,(IX+1)
990 LFFE7    LD (HL),A
1000         INC HL
1010         DJNZ LFFE7
1020         POP BC
1030         INC IX
1040         INC IX
1050         DEC BC
1060         LD A,B
1070         OR C
1080         JR NZ,LFFE0
1090         POP IY
1100         POP IX
1110         POP HL
1120         POP DE
1130         POP BC
```

```

1140          POP AF
1150          EI
1160  ZEND    RET

```

Cu această rutină se compactează un screen a cărui lungime inițială de 6912 octeți se reduce sensibil; noul screen este locat la adresa 50000.

Programul BASIC aferent este următorul:

```

10 BORDER 2: PAPER 0: INK 7: CLS
20 PRINT AT 4,0;"Pregateste caseta cu SCREEN$-ul"
   " "8sppentru compactat"
30 PAUSE 0
100 LOAD "" SCREEN$
110 LET a=60000: RANDOMIZE USR a
120 LET cit=PEEK 50000+256*PEEK 50001
130 LET cit=cit*2+2
170 CLS: PRINT "" " Au fost ocupati :"; INVERSE 1;
   cit; INVERSE 0;" octeti""8spadica:"; INVERSE
   1;cit/1024; INVERSE 0; " Ko": PAUSE 0
180 IF cit>6912 THEN CLS : PRINT "" ""
   "8spIMPOSIBIL !!!!!""11sp";cit;">6912": PAUSE
   0: GO TO 500
185 PRINT ""Vrei sa vezi SCREEN$ (d/n)?: PAUSE 0
186 IF INKEY$="d" THEN GO TO 200
187 IF INKEY$="n" THEN GO TO 300
200 RANDOMIZE USR (a+160)
210 PAUSE 0
300 CLS : PRINT # 1; AT 1,0;"Salvez SCREEN$-ul
   compactat (d/n)?: PAUSE 0
310 IF INKEY$="d" THEN GO TO 400
320 IF INKEY$="n" THEN GO TO 500
400 CLS : PRINT AT 8,0;"Pregateste caseta pentru
   salvat!": PAUSE 0: PRINT AT 11,2;"Adresa,
   lungimea: 50000,"; cit
410 SAVE "W" CODE 50000,cit
500 CLS
510 PRINT AT 6,6;"Alt SCREEN$ (d/n)?: PAUSE 0
520 IF INKEY$="d" THEN GO TO 20
530 IF INKEY$="n" THEN CLS : STOP

```

Rezultă un SCREEN\$ compactat cu numele "W" la adresa 50000, de lungime *cit* (calculată prin programul BASIC).

Folosirea SCREEN\$-ului compactat impune a se lucra cu rutina care urmează.

Exemplul 9.4

```

10          ORG 60000          ;DECOMPACTARE
          SCREEN$
20          ENT $

```

```

30          PUSH AF
40          PUSH BC
50          PUSH DE
60          PUSH HL
70          PUSH IX
80          PUSH IY
90          LD IX,50002
100         LD BC,(50000)
110        LD HL,16384
120        PUSH BC
130 LFFE0   PUSH BC
140        LD B,(IX+0)
150        LD A,(IX+1)
160 LFFE7   LD (HL),A
170        INC HL
180        DJNZ LFFE7
190        POP BC
200        INC IX
210        INC IX
220        DEC BC
230        LD A,B
240        OR C
250        JR NZ,LFFE0
260        POP IY
270        POP IX
280        POP HL
290        POP DE
300        POP BC
310        POP AF
320        EI
330 ZEND    RET

```

Programul BASIC are forma:

```

10 LOAD "W" CODE 50000: REM SCREEN$-UL COMPRIMAT
15 BORDER 5: PAPER 6: INK 2: CLS : LET a=60000: REM
   a-adresa rutinei
20 PRINT AT 4,10;"DECOMPACTARE"; AT 6,2;"SCREEN$-ul
   compactat la adresa"; AT 17,11;"ADR=50000"; AT
   20,9;"Apasa o tasta!"; PAUSE 0: LET adr=50000:
   GOSUB 1000
30 RANDOMIZE USR a: PAUSE 0
40 CLS : PRINT # 1; AT 0,9;"RELUAM (d/N)?: PAUSE 0
50 GO TO 15*(INKEY$="D")+60*(INKEY$="n")
60 STOP
1000 POKE (a+10),adr+2-256*INT ((adr+2)/256)
1010 POKE (adr+11),INT ((adr+2)/256)
1020 POKE (a+14),adr-256*INT (adr/256)

```

1030 POKE (a+15),INT (adr/256)
1040 RETURN

Exemplul 9.5

```

10          ORG 60000          ;LOAD SCREEN$ CU
                                EFECTE BORDER
20          ENT $
30          LD IX,0
40          LD DE,17
50          CALL LEA76
60          LD IX,16384
70          LD DE,6912
80          CALL LEA76
90          EI
100         RET
110 LEA76   CALL LEA8A
120         CALL LEAF6
130         INC DE
140 LEA7D   CALL LEAF6
150         LD (IX+0),L
160         INC IX
170         LD A,D
180         OR E
190         JR NZ,LEA7D
200         RET
210 LEA8A   DI
220         LD A,8
230         OUT (254),A
240         IN A,(254)
250         RRA
260         AND 32
270         OR 2
280         LD C,A
290         CP A
300 LEA98   RET NZ
310 LEA99   CALL LEAD2
320         JR NC,LEA98
330         LD HL,1045
340 LEAA1   DJNZ LEAA1
350         DEC HL
360         LD A,H
370         OR L
380         JR NZ,LEAA1
390         CALL LEACE
400         JR NC,LEA9E
410 LEAAD   LD B,156
420         CALL LEACE

```



```
430          JR NC, LEA98
440          LD A, 198
450          CP B
460          JR NC, LEA99
470          INC H
480          JR NZ, LEAAD
490 LEABC     LD B, 201
500          CALL LEAD2
510          JR NC, LEA98
520          LD A, B
530          CP 212
540          JR NC, LEABC
550          LD B, 201
560          CALL LEAD2,
570          RET
580 LEACE     CALL LEAD2
590          RET NC
600 LEAD2     LD A, 10
610 LEAD4     DEC A
620          JR NZ, LEAD4
630          AND A
640 LEAD8     INC B
650          RET Z
660          LD A, 127
670          IN A, (254)
680          RRA
690          RET NC
700          XOR C
710          AND 32
720          JR Z, LEAD8
730          LD A, C
740          CPL
750          LD C, A
760          LD A, D
770          AND 7
780          OR 8
790          OUT (254), A
800          XOR A
810          OR 8
820          SCF
830          OUT (254), A
840          RET
850 LEAF6     LD B, 178
860          LD L, 1
870 LEAFA     CALL LEACE
880          RET NC
890          LD A, 203
900          CP B
```

910	RL L
920	LD B,176
930	JR NC,LEAFA
940	SCF
950	DEC DE
960 ZEND	RET

Această rutină de 170 octeți realizează încărcarea unui SCREEN\$ cu o suită de efecte pe BORDER, acesta fiind succesiv static, cu dungi subțiri și rare, respectiv cu dungi dese. Activarea se face cu instrucțiunile:

```
10 BORDER 1: PAPER 1: INK 6: CLS
20 RANDOMIZE USR 60000
30 LOAD "" SCREEN$
```

BIBLIOGRAFIE SELECTIVĂ

1. *Lupu, Cr.Ștefănescu* - **St-Microprocesoare**, Editura Militară, București, 1986.
2. *Lupu Cr,ș.a.* - **Microprocesoare. Aplicații**. Editura Militară, București, 1982.
3. *Henrot, Marcel* - **La pratique du ZX Spectrum**, vol.2, Editions du PSI 1983.
4. *Patrubany,M* - **Totul despre microprocesorul Z80**, Editura Tehnică, București, 1989.
5. *Peterson J,L* - **Computer Organisation and Assembler Language Programmings**, Academic Press, New York, 1978.
6. *Petrescu A,ș.a.* - **A,B,C, de calculatoare și nu doar atît**, Editura Tehnică, București, 1990.
7. *Toacșe, Gh.* - **Introducere în microprocesoare**, Editura Științifică și enciclopedică, București, 1985.
8. *Zaks, R* - **Programming in the Z80**, Sybex, Inc.Berkeley, 1982.
9. xxx - **Documentațiile programelor GENS3M21 și MONS3M21.**
10. xxx - **Colecția de reviste Hobbit 1990-1992.**
11. xxx - **Colecția de reviste PC-Magazin 1990-1991.**
12. xxx - **Programul TUTOR**

SUMAR

1. NOȚIUNI INTRODUCIVE

1.1. PRELIMINARII.....	3
1.2. SISTEME DE NUMERAȚIE.....	5
1.2.1. Sistemul zecimal.....	5
1.2.1.1. Reprezentarea numerelor întregi fără semn	5
1.2.1.2. Operații aritmetice.....	6
1.2.2. Sistemul binar.....	6
1.2.2.1. Caracterizare.....	6
1.2.2.2. Operații aritmetice.....	8
1.2.3. Sistemul hexazecimal.....	8
1.2.4. Notația zecimal codificat binar	10
1.2.5. Notația pozitivă și negativă	10
1.3. STRUCTURA DE BAZĂ A UNUI MICROPROCESOR.....	11
1.3.1. Generalități.....	11
1.3.2. Registre speciale.....	14
1.3.2.1. Contorul program PC (Program Counter)	14
1.3.2.2. Indicatorul de stivă SP (Stack Pointer)	15
1.3.2.3. Registrul acumulator A (Acumulator)	16
1.3.2.4. Registrul indicatorilor de condiție F (Flag-fanion)	16
1.3.2.5. Registrul vectorilor de întreruperi I (Interrupt register).....	19
1.3.2.6. Registrul de reîmprospătare a memoriei dinamice R (Refresh)	21
1.3.3. Registre de uz general	22
1.4. ASAMBLORUL GENS3M21.....	23
1.4.1. Prezentare	23
1.4.2. Modul de lucru.....	24
1.4.2.1. Generalități.....	24
1.4.2.2. Formatul instrucțiunilor asamblorului.....	26
1.4.2.3. Directivele de asamblare.....	27
1.4.2.4. Comenzile editorului	29
1.4.2.5. Comenzile pentru asamblare și rulare a codului generat.....	29
1.4.2.6. Comenzile de bandă.....	29
1.4.3. Algoritm de lucru cu GENS3M21	29

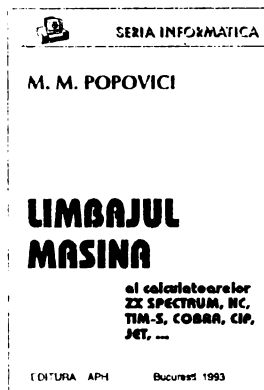
2. SETUL DE INSTRUCȚIUNI

2.1. INSTRUCȚIUNI DE ÎNCĂRCARE PE 8 BIȚI	33
2.2. INSTRUCȚIUNI DE ÎNCĂRCARE PE 16 BIȚI	35

2.3.	INSTRUCȚIUNI DE INTERSCHIMBABILITATE	36
2.4.	INSTRUCȚIUNI DE TRANSFER DE BLOCURI DE DATE	37
2.5.	INSTRUCȚIUNI PENTRU CĂUTAREA ÎN BLOCURI DE MEMORIE	38
2.6.	INSTRUCȚIUNI LOGICE ȘI ARITMETICE PE 8 BIȚI	39
	2.6.1. Adunare	39
	2.6.2. Scădere	40
	2.6.3. Instrucțiuni logice	40
	2.6.4. Comparări	40
	2.6.5. Incrementări și decrementări	41
2.7.	INSTRUCȚIUNI ARITMETICE CU SCOP GENERAL ȘI DE CONTROL AL CPU	43
2.8.	INSTRUCȚIUNI ARITMETICE PE 16 BIȚI	47
2.9.	INSTRUCȚIUNI DE ROTAȚIE ȘI DE DEPLASARE	48
2.10.	INSTRUCȚIUNI DE TESTARE ȘI MODIFICARE LA NIVEL DE BIT	54
2.11.	INSTRUCȚIUNI DE SALT	54
2.12.	INSTRUCȚIUNI DE APEL ȘI ÎNTOARCERE DIN RUTINE	56
2.13.	INSTRUCȚIUNI DE INTRARE/IEȘIRE	58
3.	FOLOSIREA INSTRUCȚIUNILOR ÎN OPERAȚII DE BAZĂ	
3.1.	NOȚIUNI INTRODUCATIVE	61
	3.1.1. Rolul funcției USR	61
	3.1.2. Organizarea memoriei calculatorului și organizarea ecranului	62
	3.1.3. Structura variabilelor de sistem	67
	3.1.4. Codurile caracterelor	71
3.2.	ÎNCĂRCAREA ÎN MEMORIE	72
	3.2.1. Încărcarea registrelor (adresarea directă)	72
	3.2.2. Adresarea indirectă	79
3.3.	OPERAȚII ARITMETICE DE BAZĂ	81
	3.3.1. Adunarea și flagul Carry (Ci)	81
	3.3.2. Scăderea cu fanionul Carry (Ci)	88
	3.3.3. Incrementarea și decrementarea	91
3.4.	INSTRUCȚIUNI CARE INFLUENȚEAZĂ VALOAREA UNUI BIT	93
3.5.	TRANSFERURI DE BLOCURI DE MEMORIE	95
4.	FOLOSIREA INSTRUCȚIUNILOR PENTRU CICLURI, TESTĂRI, ROTAȚII ȘI DEPLASĂRI	
4.1.	SALTURI ȘI CICLURI	98
	4.1.1. Salturi necondiționate	96
	4.1.2. Salturi condiționate	99
	4.1.3. Salturi relative	100
4.2.	STIVA	105
4.3.	OPERAȚII LOGICE	111
4.4.	COMPLEMENTAREA ȘI OPRIREA EXECUȚIEI PROGRAMULUI	114

4.5. TESTĂRI ȘI COMPARAȚII	116
4.5.1. Testarea fiecărui bit luat izolat.....	116
4.5.2. Compararea constantelor, registrelor sau octeților de memorie	117
4.5.3. Comparări cu repetiții, incrementeări și decrementări	122
4.6. ROTIRI ȘI DEPLASĂRI	126
4.6.1. Rotiri de registre și locații de memorie	126
4.6.2. Rotirea zecimală	133
4.6.3. Deplasări	135
5. FOLOSIREA INSTRUCȚIUNILOR PENTRU CULORI, SUNETE ȘI Scrierea Textelor	
5.1. APELAREA SUBROUTINELOR (SUBPROGRAMELOR).....	140
5.2. INSTRUCȚIUNI DE INTRARE/IEȘIRE (i/e)	147
5.3. CULOAREA.....	149
5.4. SUNETE	163
5.4.1. Codificarea unei melodii.....	163
5.4.2. Sunete diverse	168
5.4.3. Efecte pe BORDER cu sunete.....	171
5.5. SCRIEREA TEXTELOR	174
5.5.1. Scrierea unei linii cu 32 caractere.....	174
5.5.2. Scrierea textelor multiple.....	176
5.5.3. Scrierea cu aldine	179
6. TASTATURA ȘI AFIȘAJUL	
6.1. TASTATURA	181
6.1.1. Analiza tastaturii.....	181
6.1.2. Utilizarea rutinei de scanare a tastaturii.....	182
6.1.3. Utilizarea variabilei de sistem LAST-K.....	184
6.1.4. Pauzele.....	185
6.2. AFIȘAJUL	188
6.2.1. Efecte cu atribute.....	188
6.2.2. Efecte de scriere.....	196
6.2.2.1. Scriere cu aldine (în mod normal sau pe verticală).....	196
6.2.2.2. Scriere cu litere rotite (cu 90° sau 180°).....	198
6.2.2.3. Scriere roll.....	199
7. NOȚIUNI DESPRE ANIMAȚIE ȘI ÎNTRERUPERI	
7.1. ELEMENTELE ANIMAȚIEI : HAZARDUL ȘI DEPLASAREA	202
7.2. RUTINELE AFIȘĂRII	203
7.2.1. Poziția PRINT	205
7.2.2. Poziția octet	206
7.2.3. PLOT	206
7.2.4. Adresa matricei de caractere	207
7.2.5. Comanda PLOT	208
7.2.6. Mișcarea.....	209

7.3. ELABORAREA UNUI PROGRAM DE DIVERTISMENT (JOC).....	210
7.4. ÎNTRERUPERILE.....	218
8. 50 RUTINE PENTRU PERFECTIONAREA PROGRAMELOR PROPRII	
8.1. SUNETE	227
8.2. EFECTE VIZUALE	236
8.3. EFECTE AUDIO-VIZUALE	262
8.4. MODALITĂȚI DE SCRIERE.....	284
8.4.1. Scrierea taxtelor curente	284
8.4.2. Scrierea de titluri cu litere mărite.....	292
8.4.3. Alte modalități.....	306
9. DEZASAMBLORUL MONS3M21	
9.1. INTRODUCERE.....	323
9.2. COMENZILE MONS.....	324
9.3. ALGORITMUL DE LUCRU	328
9.4. EXEMPLE DE DEZASAMBLARE	328
BIBLIOGRAFIE SELECTIVĂ	



Lucrarea inițiază cititorul în cunoașterea și aplicarea instrucțiunilor microprocesorului Z80 și este ilustrată prin 150 de rutine care pot fi folosite în programe proprii.

Aceste rutine realizează sunete diverse, efecte vizuale și audio-vizuale interesante și atractive precum și modalități diverse de scriere cu litere de mărimi și forme diferite care defilează, se rotesc sau sînt "mitraliate" pe ecran.

De asemenea sînt arătate și ilustrate principiile organizării jocurilor precum și compilarea programelor BASIC sau dezasamblarea programelor scrise în

limbaj mașină.

Mircea Mihail Popovici

În Editura APH au apărut:

1. M.M. POPOVICI – **BASIC** pentru calculatoarele **Spectrum** e.t.c.
Vol. 1. **INSTRUCȚIUNI. EXERCITII. PROBLEME.**
2. M.M. POPOVICI – **BASIC** pentru calculatoarele **Spectrum** e.t.c.
Vol. 2. **COLECȚIE DE PROGRAME.**
3. R. M. HRISTEV – Introducere în **PROLOG** – Un limbaj al inteligenței artificiale.
4. R. M. HRISTEV – **GHIDUL** utilizatorului și programatorului **SPECTRUM.**
5. A. HRISTEV – **PROBLEME DE FIZICĂ** pentru licee, bacalaureat, admitere în facultăți, învățămînt politehnic.
Vol. 1. **Mecanica**, Vol 2. **Termodinamica**, Vol. 3. **Electricitate**, Vol. 4. **Optica.Fizica atomică.**

precum și alte cărți școlare de **fizică** și **matematică**.