RUCZ
LAJOS

RUTINRÓL RA

BEPILLANTÁS A SINCLAIR
SPECTRUM GÉPI KÓDÚ
PROGRAMOZÁSÁBA

LSI Alkalmazástechnika

Tanácsadó Szolgálat

R U T I N R Ó L - R U T I N R A

Bepillantás a SINCLAIR Spectrum gépi kódú világába

Írta: Rucz Lajos

Lektorálta: Bankó Miklós

Szerkesztette: Kökényesi Judit

LSI Alkalmazástechnikai Tanácsadó Szolgálat, Budapest, 1986

T A R T A L O M J E G Y Z É K

T A R T A L O M J E G Y Z É K

ELŐSZÓ

A számítástechnika tért hódít a világban. A technika fejlődése lehetővé tette a mikrogépek széleskörű elterjedését régebben kisebb vállalatoknál, manapság pedig már a háztartásokban is. Napjainkban egy új most piacra került gép is megvásárolható, nem is beszélve a régebbi konstrukciók árainak rohamos zuhanásáról. A ZX Spectrum személyi számítógép 1982-ben jelent meg Angliában, s az eltelt időszak alatt millió feletti darabszámot értékesítettek szerte a világon. Nincs pontos információnk a hazai Spectrum tulajdonosok számát illetően, mindenesetre a hivatalos - kb. 20 ezres - szám azért mégis jelent valamit.

A COMMODORE 64, és az IBM PC számítógépek mellett nem is tudunk példát mondani olyan mikrogépekre, melyekre már annyi játék és felhasználói program készült, mint a Spectrumra. Ez is jelzi a gép népszerűségét és ehhez fűződően a programozástechnika kimeríthetetlenségét.

Kezdetben a gép kezelésével ismerkedünk, megtanuljuk rajta a BASIC nyelvet, s pihenésképpen játszunk a rendelkezésre álló játékprogramokkal. Egy idő után viszont kevésnek bizonyul mindaz, amit BASIC nyelven ki tudunk csiholni a gépből. Bizonyos, hogy minden személyi számítógép tulajdonos szeretne továbblépni a géppel kapcsolatos ismeretek elsajátításában. Áhítattal nézzük a játékprogramok programozástechnikai színvonalát, ötletességét, grafikáját. Nem titok, a játékprogramok mintegy 95 %-át gépi kódban írják.

Mondjuk ki nyíltan, idegenkedünk ettől a területtől. A BASIC nyelv alapos elsajátítását követően mindenkiben felmerül a gépi kódú programozás gondolata. Már a programlista látványától is sokan megijednek, neki sem fognak hogy megtanulják legalább az alapismereteket. Mások próbálkoznak, többször nekifognak, de az ide vonatkozó szakirodalmak szinte "kínai" nyelvezete nem igazán segíti őket a kezdeti lépések megtételében. Csak néhányan akadnak, akik sikerrel lépik át a BASIC és a gépi kódú programozás közötti éles határt.

Könyvünk első két fejezetét azoknak szánjuk, akik eddig reményt sem láttak arra, hogy túljussanak a kezdeti nehézségeken. Az első fejezet részletesen áttekinti az alapismereteket és a Z-80 processzor működési mechanizmusát, majd eligazodási lehetőséget nyújt az utasításcsoportok között. Ez utóbbit nem volt célunk a legmélyebb részletekig tárgyalni, hiszen erre több szakkönyv is megjelent az elmúlt évek folyamán:

Z-80 mikroprocesszor sorozat I-VIII.	- Ipari Informatikai Központ (1983)
ZX Spectrum BASIC és gépi kódú programozás	- Ipari Informatikai Központ (1984)
Z-80 Assembler	- LSI ATSz. (1985)
Z-80 Software tábla	- LSI ATSz. (1985)

A kezdőknek természetesen ajánlott, az említett irodalmak könyvünkkel párhuzamosan történő áttanulmányozása is.

A második fejezet egyfajta megközelítési módszert szemléltet arra vonatkozólag, hogy mikor előnyösebb a BASIC helyett a gépi kódot használnunk. Szemléletes példákat találunk arra, hogy a legtöbb esetben a gépi kódú rutinok javára drasztikus időnyereséget írhatunk. Ebből azt a következtetést vonhatjuk le, hogy BASIC nyelvű programjainkat, de különösen azoknak igen lassan futó részeit érdemes gépi kódú rutinokkal kiváltani.

A könyv második részében található rutinokkal gyakran találkozunk felhasználói programjainkban, így a Beta Basic-ben és a Master Toolkit-ban is (Ezeknek a programoknak részletes ismertetése megtalálható a "SINCLAIR Spectrum játék és program" c. könyvben - LSI ATSz 1986.), ahol gyakran gondot okoz hogy a toolkit program 4-5 Kbyte-ot, vagy még ennél is többet lefoglal a memóriából, s így a felhasználható memóriaterületünk nagysága jelentősen csökken, esetenként akár 50 %-kal is.

A könyvben közölt rutinokat külön-külön is használhatjuk. Egy-egy rutin használatakor a memóriaterületünk szinte nem is csökken, hiszen a bonyolultabb rutinok sem haladják meg a kb. fél Kbyte-os méretet. Természetesen egyszerre több rutin is elhelyezhető a memória tetszőleges területén, a memóriában levő program helyigényétől függően. Végül, de nem utolsó sorban rutinjainkat beépíthetjük saját BASIC vagy gépi kódú programjainkba is.

Célunk a rutinok ismertetésén túl azok megértetése is. Ezt segítik elő az un. comment-ek (megjegyzések) minden assembly lista 4. oszlopában.

A rutinok beolvasásával kapcsolatban tudjuk, hogy a legtöbb monitorprogram csak hexadecimális formában engedi meg az adatbyte-ok bevitelét, mégis szándékosan közöljük a kódokat decimális formában, mert akik nem vágnak neki a kód értelmezésének, s nem ismerik az assembler ill. monitorprogramok használatát, azoknak a legcélszerűbb a DATA sorba szedett kódokat READ-POKE segítségével FOR-NEXT ciklusban beolvastatni.

A legtöbb rutin listájának első oszlopában találkozunk az ADDR+nn címhivatkozással. Ez azt jelenti, hogy a rutint nem fix memóriahelyre tesszük le, hanem bárhová elhelyezhető a szabad memóriaterületen.

FIGYELEM ! ADDR+00-nak bármilyen címet is válasszunk, a rutin beolvasása előtt mindig állítsuk be a RAMTOP-ot ennél eggyel alacsonyabb címre.

Tehát: CLEAR ADDR-01 (ENTER), hogy biztosítsuk rutinunk számára a szabad memóriaterületet, hiszen a RAMTOP alatti terület általában foglalt.

A képzettebbek a mnemonikok alapján bármelyik assembler program segítségével megoldhatják a rutinok bevitelét (Assembler és monitorprogramok használata részletesen megtalálható a "SINCLAIR Spectrum játék és program" c. könyvben - LSI ATSz. 1986.)

Hisszük, hogy könyvünket haszonnal forgatja majd minden Spectrum tulajdonos, gépi kódban jártas és járatlan egyaránt.

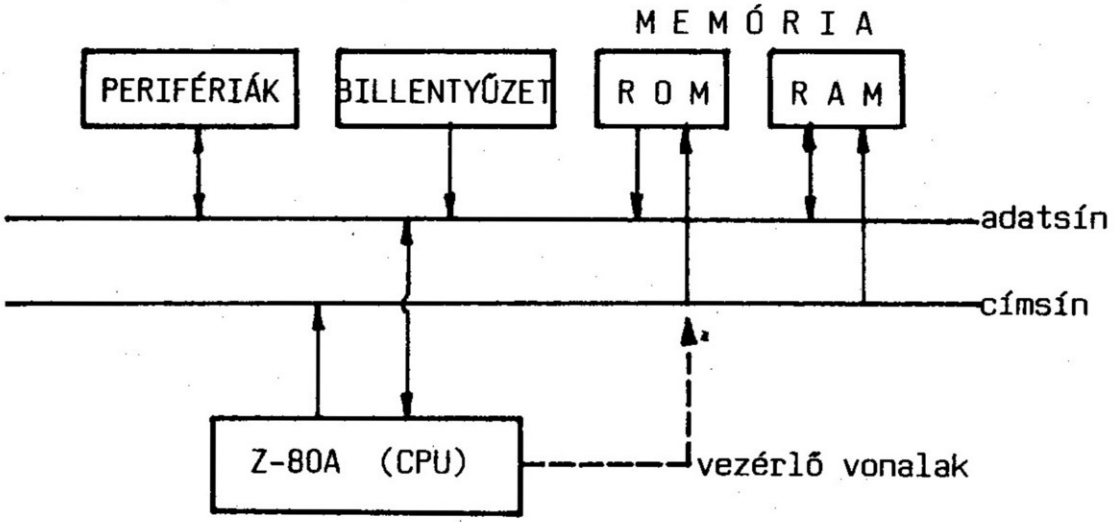
Budapest, 1986 A Kiadó

1.fejezet

A Z-80 MIKROPROCESSZOR MŰKÖDÉSE ÉS UTASÍTÁSCSOPORTJAI

1.1 Alapfogalmak

Legelőször a számítógép működésének alapjait tekintjük át:



Számítógépünk, első ránézésre egy apróka fekete doboz tucatnyi billentyűvel és néhány szines csíkkal. A dobozon belül akár hisszük-akár nem, többmillió tranzisztor található, néhány tucat integrált áramkörben, melyek speciális funkciókat látnak el.
Térjünk a tárgyra!

Mikrogépünk memóriája 8 bites tárolókból épül fel, s minden memóriarekeszhez egy cím van hozzárendelve.
A memóriának két csoportját különböztetjük meg, melyek közül a ROM (Read Only Memory) csak olvasható memória, a RAM (Random Access Memory) pedig írható és olvasható.
A ROM memória a Spectrumban 16 Kbyte, mely tartalmazza az operációs rendszert, itt találhatjuk meg a BASIC-et gépi kóddá átalakító "tolmácsot", az Interpreter-t, valamint a gép beépített karakterkészletét, stb.
(A ROM memória részletes leírása megtalálható a "ZX Spectrum ROM programja" c. könyvben - Ipari Informatikai Központ 1985.)
A RAM memória egyidőben összesen 48 Kbyte lehet. Akiknek 16 Kbyte-os gépük volt, a legtöbben már kibővítették gépüket 48 Kbyte-osra. Ez egyébként még tovább bővíthető, un. lapozós memóriával. Mint tudjuk, a RAM terület is különböző részekre tagozódik, melyek részletes tárgyalásába most nem bocsájtkozunk, mert megtalálható a Spectrum kézikönyvben.

A memória, a billentyűzet, a perifériák és a mikroprocesszor (továbbiakban: processzor) közötti kommunikációra az ULA segítségével van lehetőség.
A kommunikáció megvalósításához elengedhetetlenül szükségesek:

- -adatsín (adatbusz)
- -címsín (címbusz)

-vezérlővonal(-ak)

A Z-80-nál a címsín 16, az adatsín 8 vonalból áll. A processzor a vezérlővonalakon át befolyásolja környezetének állapotát és azokon át kap jeleket a környezet állapotáról.
A processzor bonyolult belső felépítésű. Nem kívánjuk a HARDWARE-t részletesen tárgyalni, azt viszont mindenképpen ismertetjük, ami a gépi kódú programozás megértéséhez elengedhetetlen.

A processzort általában négy fő részre szokták osztani:

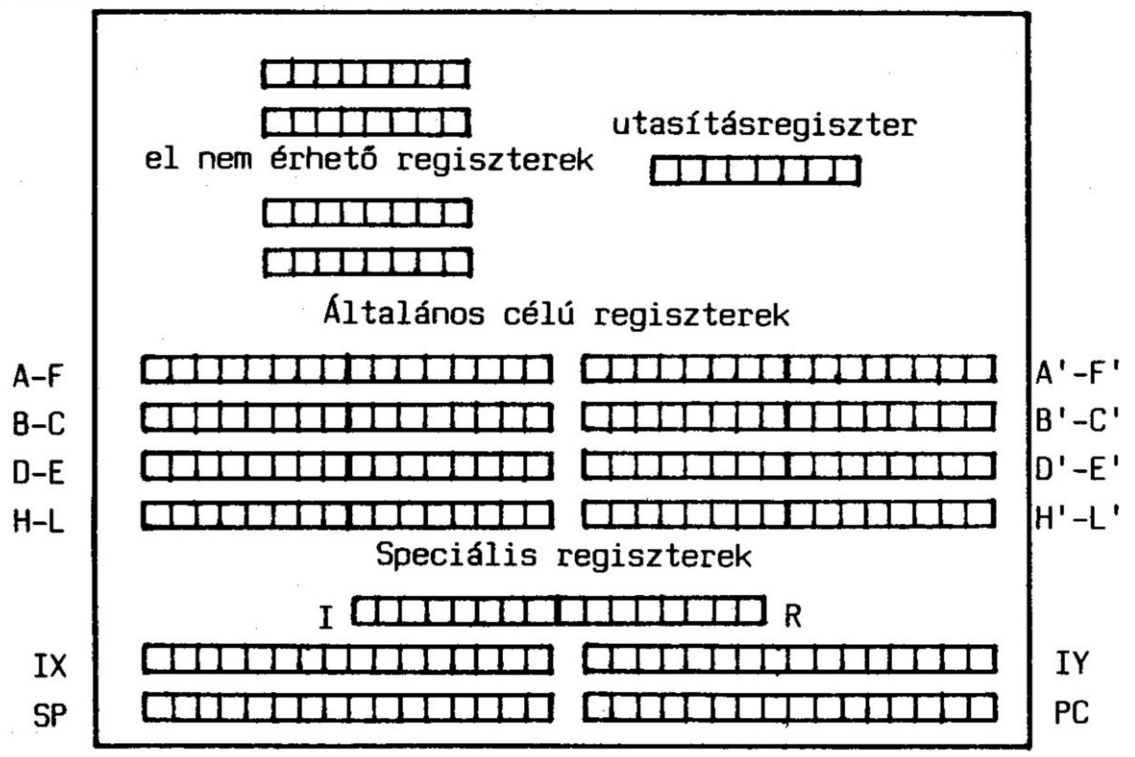
a. vezérlőegység
b. utasítástáblázat
c. aritmetikai és logikai egység
d. regiszterek

Könyvünkben az első három rész ismertetésére nem térünk ki.

A regisztereket először három csoportra bontjuk:

a. elérhetetlen regiszterek
b. utasításregiszter
c. elérhető regiszterek

A programozó számára elérhetetlen regisztereket a processzor saját céljaira használja, erről a későbbiekben még szólunk.
Az utasítás regiszterbe íródik be az aktuális gépi kódú utasítás a memóriából.
A legfontosabb csoport azok a regiszterek, melyek a programozó számára közvetlenül elérhetők.



Szemléltető ábránkon jól láthatók a regiszterek csoportjai.

Vannak aktív, és un. vesszős regiszterek, ezek tartalma (megfelelő utasítással) felcserélhető.
Ez a csere bármikor elvégezhető, vagyis a vesszős regisztereket lényegében átmeneti tárolásra alkalmazzuk, arra az időre, amíg az aktív regisztereket más célra használjuk.

Az aktív regiszterek is több szempont szerint feloszthatók.
A bitek száma szerint megkülönböztetünk 8 és 16 bites regisztereket, a másik szempont pedig a felhasználás szerint: általános célú és speciális regiszterek.

Általános célú regiszterek:

A-F ; B-C ; D-E ; H-L (8 bites regiszterek)

Az A regiszter az "akkumlátor": a legtöbb aritmetikai, logikai, ill. összehasonlító művelet eredménye ebben a regiszterben születik.
A B-C ; D-E és H-L regiszterek külön-külön, vagy párban is kezelhetők.
Az F regiszter kilóg a sorból és megérdemli, hogy külön is beszéljünk róla. Az F a FLAGS angol szó rövidítése.Ez szótári formában zászlókat jelent, de a számítástechnikai nyelv "jelzőbitek"-ként könyveli el. Ennek a regiszternek a tartalmát a processzor bitenként külön-külön kezeli. A programozás szempontjából két legfontosabb bitjét emeljük ki:

0.bit - átviteli jelzőbit
6.bit - zérus jelzőbit

Az átviteli jelzőbitet általában C-vel jelölik (nem szabad összekeverni a C regiszterrel).

A zérus jelzőbit jelölése Z.

Nemsokára igen sűrűn fogunk találkozni ezekkel a bitekkel, rutinjaink írása közben.

Speciális regiszterek:

I-R (8 bites regiszterek)
IX ; IY ; SP és PC (16 bites regiszterek)

Az I regiszter a megszakítás-vektor regiszter, később lesz még róla szó.

Az R regiszter a memóriafrissítő regiszter. Ez utóbbi csak HARDWARE működési oldalról fontos, programozástechnikailag a Spectrumban semmi jelentősége nincs.

Az IX és IY regiszterpárok 16 bitesek, segítségükkel a memória közvetetten címezhető. Erről is lesz még szó.

Az SP (Stack-pointer) regiszter a verem-mutató. Ez egy memóriacím, amely a RAM processzor-stack (verem) területének tetejére mutat. A verembe menthetünk ki ideiglenesen regiszterpár tartalmakat (PUSH), melyeket később visszatölthetünk (POP). Megjegyezzük, hogy a visszatérési címek is itt tárolódnak. A verembe pakolt adatok közül mindig a legutoljára bevitt lesz a verem tetején, és a verem-mutató is erre a címre mutat. Ha kiveszünk egy adatot a veremből, nem a következő adat fog a verem-mutatóhoz lépni, hanem a verem-mutató tartalma mutat a ténylegesen soron következő adatra.

Alaphelyzetben a verem-mutató a 65344. címre mutat, vagyis a 65344/45 rekeszeket címezi meg.

A PC (Program-Counter) regiszter a program-számláló. Gépi kódú program futtatásakor értéke mindig a végrehajtás alatt álló utasítás memóriabeli címe, az utasítás végrehajtása során automatikusan tovább lép.

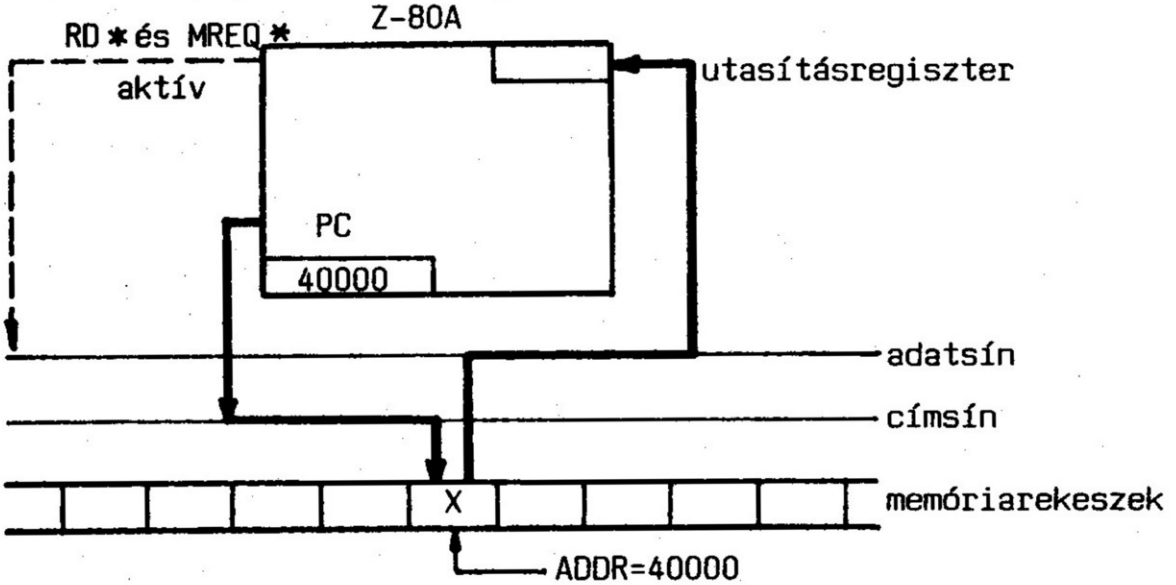
Mielőtt az utasításcsoportok áttekintését elkezdenénk, foglalkozzunk egy kicsit a gépi kódú programozás alapfogalmaival.

A gépi kód számok meghatározott sorrendű sorozata, nem általános, hanem speciális programozási nyelv, melyet kizárólag a hozzá tartozó típusú processzor képes értelmezni.
A gépi kód is utasításokat használ, úgy mint bármely magas szintű programnyelv (pl. BASIC,C,FORTH stb.), megértése viszont sokkal nagyobb figyelmet követel. A számok sorozata nem elegendő a gépi kódú program értelmezéséhez, ezért használjuk az un. assembly formátumot. Nincs igazán egységes megállapodás az assembly lista pontos felépítését illetően.
A legtöbb assembler és disassembler program az assembly formátum egyes oszlopait hexadecimális számjegyekkel jelzi, és azt is tudjuk, hogy a rendelkezésre álló irodalmak szinte kivétel nélkül ezt a formátumot követik. Mi elszakadunk ettől, mert úgy érezzük a tizenhatos számrendszertől idegenkedik a hétköznapi ember, számára jóval barátságosabb a mindennapi jól bevált tizes számrendszer használata.

Könyvünkben az assembly lista első oszlopában találjuk az adott gépi kódú utasítás első byte-jának memóriabeli címét decimálisan (vagy indexelt hivatkozással: pl. ADDR+nn), a második oszlopban az utasítás byte-ját vagy byte-jait (magát a gépi kódot) szintén decimálisan, a harmadik oszlopban a un. mnemonikokat (a gépi kódú utasítások emberhez közelebb álló-angol szavak rövidítéseiből származó-megnevezéseit), és végül az utolsó oszlopban a hagyományos - pontosvesszővel elhatárolt - comment(megjegyzés) mezőt, mely elősegíti az adott utasítás, vagy rutin megértését.

Nézzük meg mi is történik valójában egy gépi kódú program futtatásakor.

A RANDOMIZE USR ADDR (ahol ADDR tetszőleges cím, azaz a program kezdőcíme) utasítás kiadásakor a PC regiszterbe automatikusan az ADDR cím kerül, majd a processzor a címsínen keresztül megcímzi az adott rekeszt. A rekesz tartalma úgy kerül az adatsínre, hogy a processzor az RD* és MREQ* vezérlővonalakat aktív állapotba helyezi. Ezután pedig az adatsínen megjelent adatot automatikusan az utasításregiszterbe írja.



Az utasítást a processzor a vezérlőegységben elhelyezett utasítástáblájából azonosítja, és végrehajtja az annak megfelelő funkciót. A végrehajtás után a PC automatikusan tovább lép és a következő rekesz tartalma kerül kiolvasásra.

Egy Z-80 utasítás hossza 1-4 byte-ig terjedhet.

pl. 1 byte-os:	ADDR+nn	201	RET	
pl. 2 byte-os:	ADDR+nn	62,50	LD	A,50
pl. 3.byte-os:	ADDR+nn	33,223,87	LD	HL,22495
pl. 4.byte-os:	ADDR+nn	237,75,178,92	LD	BC,(23730)

A következőkben tekintsük át egy utasítás végrehajtásának fázisait, ez utóbbi 4 byte-os utasítás alapján.

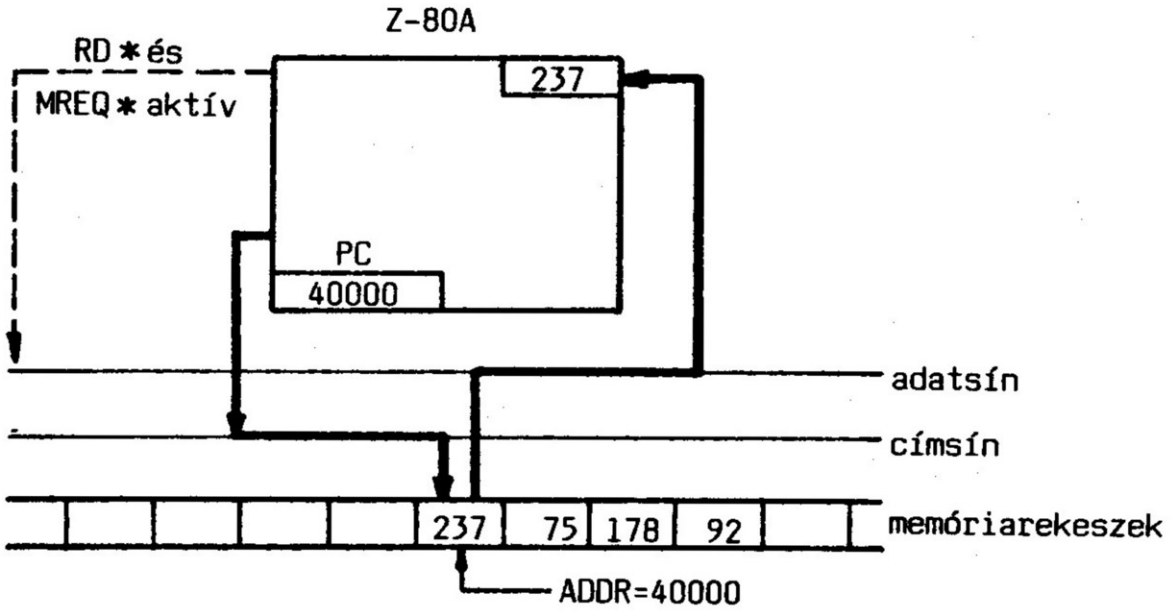
Képzeljük el a processzor beépített utasításkészletét, mint egy táblázatot. Ha az előzőleg ismertetett módon a megcímzett rekeszből az utasításregiszterbe 75 kerül, akkor a processzor a táblázatából az ahhoz tartozó LD C,E utasítást fogja végrehajtani, vagyis áttölti C regiszterbe az E regiszter tartalmát.
Mivel a Z-80 több mint 255 utasítást ismer, szükség van un. táblázati eltolási kódokra. Ilyen a 203 (hexadecimálisan: CB),vagy az itt látható 237 (HEX ED) is. Hatásukra a táblázat egy másik

területét fogja megvizsgálni a processzor vezérlőegysége, s ebből adódik, hogy a 75 kód értelme 203 után BIT 1,E ill. 237 után LD BC,(NN) lesz. Az utasítás végrehajtásának folyamatát tekintsük át egy mintapéldán keresztül.

Ha mintapéldánk a 40000. címen található (ADDR=40000), akkor a a rekeszek tartalma:

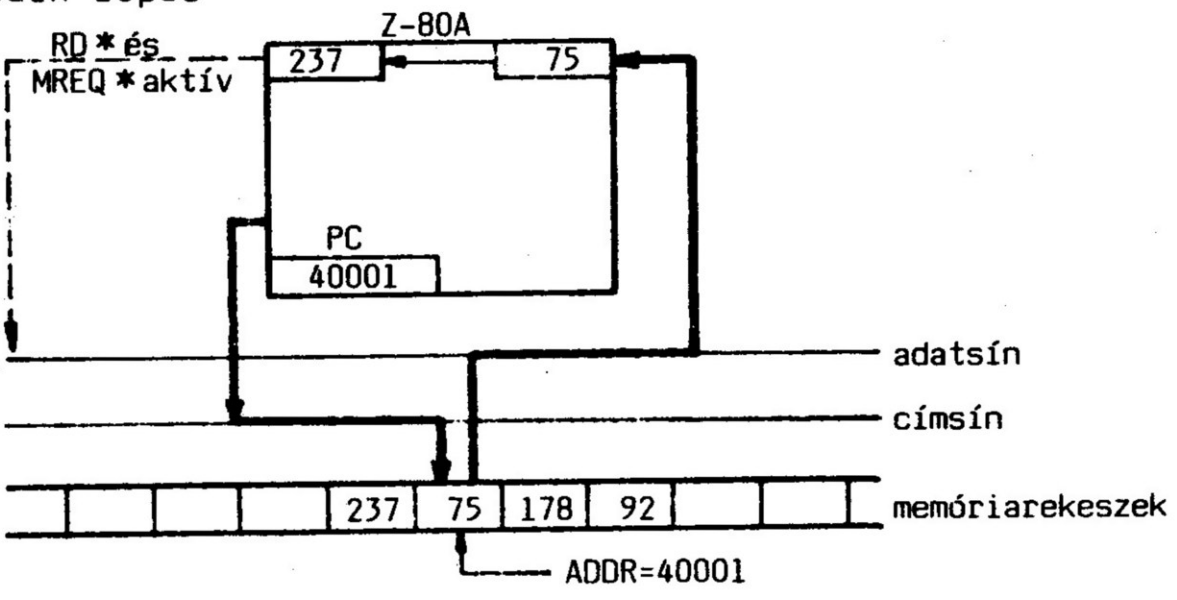
```
40000 - 237
40001 -  75
40002 - 178
40003 -  92
```

- első lépés



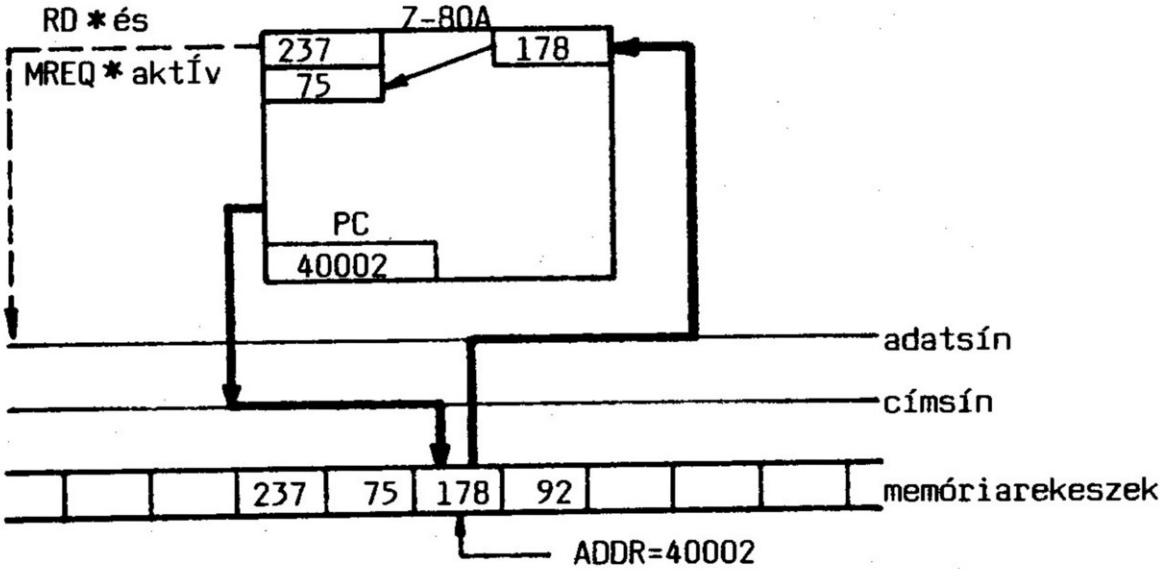
A processzor a RANDOMIZE USR 40000 utasítás kiadását követően beírja a PC-be a 40000. címet, majd ugyanezt a 40000-ret megcímezi a címsínen keresztül. Az RD* és MREQ* vezérlővonalakat aktív állapotba helyezi (READ és MEMORY REQUEST), aminek hatására a 40000. cím tartalma, vagyis 237 az adatsínre kerül, innen pedig a processzor beolvassa az utasításregiszterbe.

- második lépés



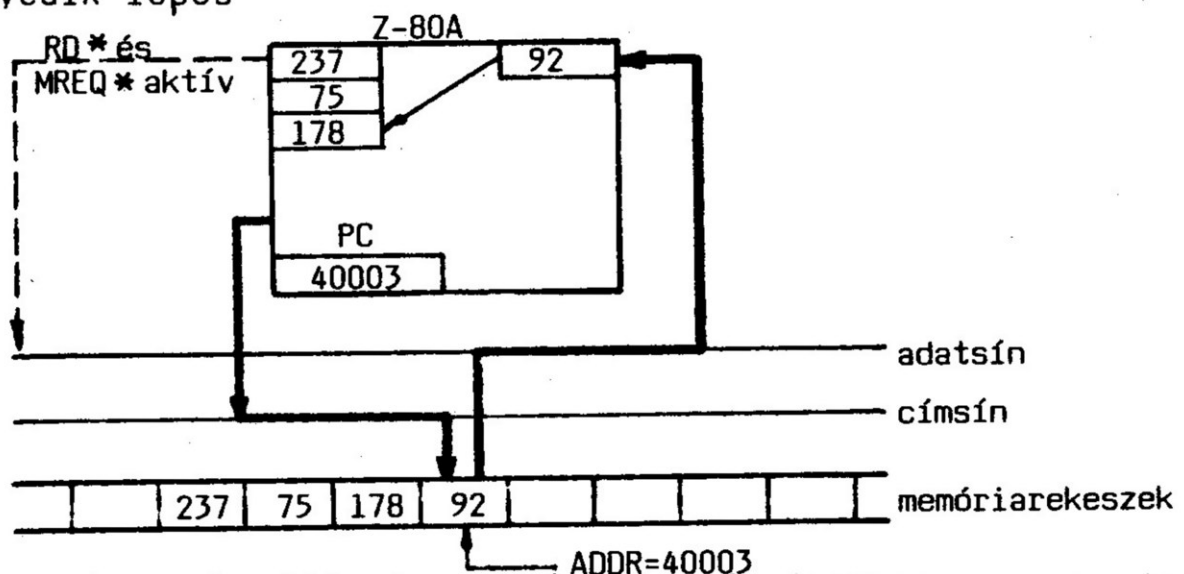
Az utasításregiszterben tárolt értéket (kódot) megvizsgálja a vezérlőegység és megállapítja, hogy nem végrehajtható utasítást olvastunk be, hanem egy eltolási értéket, így az utasításregiszter tartalmát áttölti a vezérlőegység belső (általunk elérhetetlen) regisztereinek egyikébe. PC-t lépteti eggyel, s a már előbb megismert memóriakiolvasási folyamatot megismétli.

- harmadik lépés



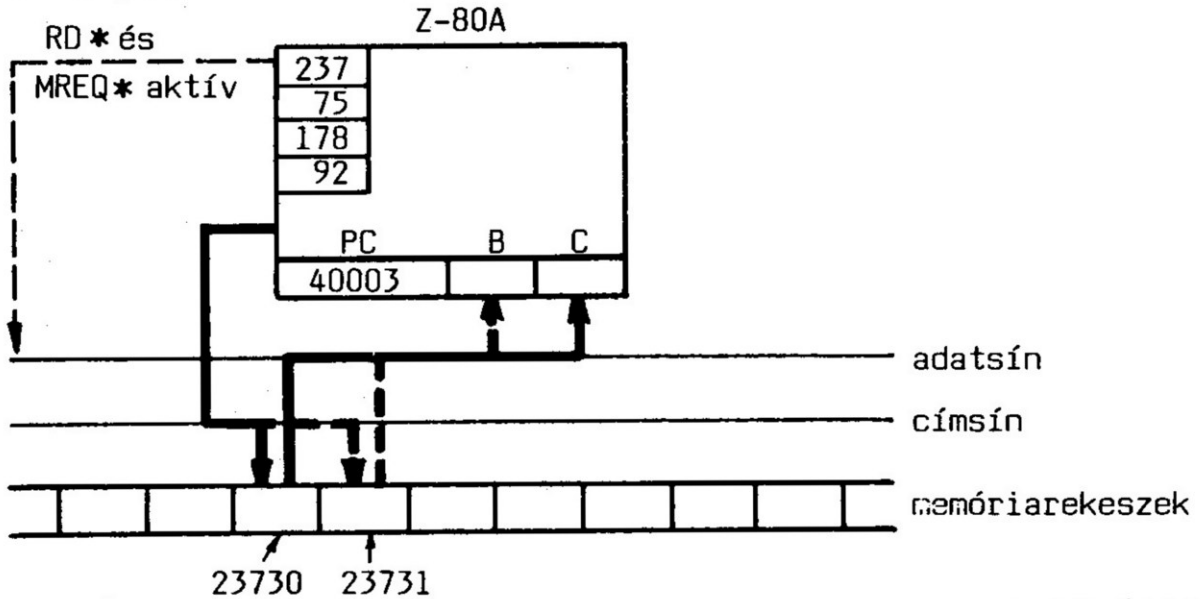
A processzor azonosítja a 75-ös kódot a megfelelő táblázatból, melynek eredménye: LD BC,(NN), és megállapítja hogy az utasítást csak akkor tudja végrehajtani, ha rendelkezésre áll egy két byte-os cím, vagyis két lépésben azt kell beolvasnia. PC-t tovább lépteti az utasításregiszter tartalmát ismét áttölti belső regisztereinek egyikébe és ismét végrehajtja a kiolvasási folyamatot (PC = 40002).

- negyedik lépés



A beolvasásra került cím alsó byte-ját áttölti az utasításregiszterből belső regiszterei egyikébe, PC-t tovább lépteti (PC = 40003), és beolvassa a cím felső byte-ját is.

- ötödik lépés



Együtt a négy byte, lehetőség van az utasítás: 'LD BC,(23730)' végrehajtására. A processzornak a BC regiszterpárba kell töltenie a 23730/23731 memóriarekeszek tartalmát. A kiolvasási folyamat a megismert módon zajlik le, azzal a különbséggel, hogy a végrehajtás alatt PC értéke változatlan marad. Előbb kiolvasásra kerül a 23730. memóriarekesz tartalma, és az adatsínről a C regiszterbe kerül, majd a 23731. memóriarekesz tartalma kerül az adatsínre, és a B regiszterbe töltődik. Amikor az utasítást a processzor végrehajtotta, PC-t automatikusan tovább lépteti (PC= 40004) és beolvasásra kerül a következő adatbyte (függetlenül attól hogy ott mi található, így ha ki akarunk lépni jelenlegi mintaprogramunkból, a 40004-es memóriahelyen el kell helyeznünk egy 201-es kódot (RET) a visszatéréshez).

Eddig csak olvastunk a memóriából, ha viszont adatbyte-ot akarunk beírni egy memóriarekeszbe, akkor a processzor előbb elhelyezi az adatot az adatsínen, majd aktív állapotba hozza a WR* és MREQ* (WRITE és MEMORY REQUEST) vezérlő vonalakat.

1.2 A Z-80 utasításcsoportjai

A Z-80 utasításcsoportjait az ismert Dr. Ian Logan féle rendszer szerint tekintjük át. Az egyes utasítások részletes ismertetésére nem törekszünk, akik ezt igénylik, lapozzák fel a bevezetőben megemlített könyveket.
Ahol lehetőség nyílt, megpróbáltunk analógiát keresni az adott gépi kódú utasítás BASIC megfelelőjére. Nehogy félreértsük, a gépi kódú utasítások alatt látható BASIC mintaprogramok futási eredményét nem lehet és nem is érdemes összevetni a gépi kódú utasításokéval. Az analógia csak szemléltető jellegű, ezzel is elősegítve a könnyebb megértést. Gépi kódban a regiszterek töltését a BASIC változók értékadásához hasonlítottuk. A BASIC analógiákban a változónevek mindig az adott gépi kódú utasítás regisztereire értendők.

1.csoport - NOP 'üres' utasítás

PC tovább lép, de az utasítás hatására más nem történik. BASIC-ben leginkább a REM utasításra hasonlít. Programjaink belsejében - ha nem szükséges - memóriatakarékossági okokból nem célszerű használni.

2.csoport - Regiszter töltése konstanssal

```
pld. ADDR+nn   62,50          LD   A,50        ;gépi kódban
     10 LET A=50                               ;BASIC-ben
```

3.csoport - Regiszterek értékeinek átmásolása ill. cseréje

-regiszterből-regiszterbe

```
pld. ADDR+nn   65             LD   B,C         ;gépi kódban
     10 LET B=C                                ;BASIC-ben
```

-regiszterpárból-verem mutatóba

```
pld. ADDR+nn   249            LD   SP,HL       ;gépi kódban
     10 LET SP=HL                              ;BASIC-ben
```

-két regiszterpár cseréje

```
pld. ADDR+nn   235            EX   DE,HL       ;gépi kódban
     10 LET CS=DE                              ;BASIC-ben
     20 LET DE=HL
     30 LET HL=CS
     40 REM CS=átmeneti változó
```

-segédregiszterek cseréje

```
pld. ADDR+nn   8              EX   AF,A'F'
```

4.csoport - Regiszterek töltése a memória adott című rekeszéből

-abszolút címzés

```
pld. ADDR+nn   42,64,156      LD   HL,(40000) ;gépi kódban
     10 LET L=PEEK 40000: LET H=PEEK 40001    ;BASIC-ben
```

-indirekt címzés

```
pld. ADDR+nn   126            LD   A,(HL)      ;gépi kódban
     10 LET A=PEEK HL                          ;BASIC-ben
```

-indexelt címzés

```
 pld. ADDR+nn   221,126,30   LD   A,(IX+30)   ;gépi kódban

      10 LET A=PEEK (IX+30)                   ;BASIC-ben
```

5.csoport - Memóriarekesz töltése regisztertartalommal vagy konstanssal

-abszolút címzés

```
 pld. ADDR+nn   50,0,128     LD   (32768),A   ;gépi kódban

      10 POKE 32768,A                         ;BASIC-ben
```

-indirekt címzés

```
 pld. ADDR+nn   54,27        LD   (HL),27     ;gépi kódban

      10 POKE HL,27                           ;BASIC-ben
```

-indexelt címzés

```
 pld. ADDR+nn   221,119,228  LD   (IX-28),A   ;gépi kódban

      10 POKE IX-28,A                         ;BASIC-ben
```

Megjegyzés: Indexelt címzésnél a konstans értéke 0-255-ig terjedhet.
Ha a konstans 0-127, akkor az eltolás is +0...+127.
Ha a konstans 128-255, akkor az eltolás értékét 256-ból le kell vonnunk, és negatív előjelt kap:-128..-1

6.csoport - Összeadó utasítások

-regiszterek vagy regiszterpárok összegzése (túlcsordulás az átviteli jelzőbitbe)

```
 pld. ADDR+nn   128          ADD  A,B         ;gépi kódban

      10 LET A=A+B                            ;BASIC-ben
      20 IF A>255 THEN LET CF=1: LET A=A-256
      30 REM CF=átviteli jelzőbit
```

-regiszterek vagy regiszterpárok tartalmának növelése eggyel

```
 pld. ADDR+nn   3            INC  BC          ;gépi kódban

      10 LET BC=BC+1                          ;BASIC-ben
```

-regiszterek vagy regiszterpárok összegzése (hozzáadódik az átviteli jelzőbit is)

```
pld. ADDR+nn   141          ADC  A,L        ;gépi kódban

     10 LET A=A+L+CF                       ;BASIC-ben
     20 REM CF=átviteli jelzőbit
```

Megjegyzés: Az átviteli jelzőbitet általában C-vel jelöljük, de mivel a szemléltető analóg BASIC listákban a C regiszterrel könnyen összekeverhető lenne, így CF-et használunk.

7.csoport - Kivonó utasítások

-regiszterek kivonása
(alulcsordulás az átviteli jelzőbitbe kerül)

```
pld. ADDR+nn   146          SUB  D          ;gépi kódban

     10 LET A=A-D                          ;BASIC-ben
     20 IF A<0 THEN LET CF=1: LET A=256-A
```

-regiszterek vagy regiszterpárok tartalmának csökkentése eggyel

```
pld. ADDR+nn   43           DEC  HL         ;gépi kódban

     10 LET HL=HL-1                        ;BASIC-ben
```

-két regiszter vagy regiszterpár "valódi" kivonása

```
pld. ADDR+00   167          AND  A          ;gépi kódban
     ADDR+01   237,82       SBC  HL,DE

     10 LET CF=0                           ;BASIC-ben
     20 LET HL=HL-DE-CF
```

Megjegyzés: Az AND A utasítás az átviteli jelzőbitet törli. Azért szükséges, mert az SBC HL,DE utasítás azt is levonná, ha értéke 1 lenne.

8.csoport - Összehasonlító utasítások

```
pld. ADDR+nn   254,50       CP   50
```

BASIC analógiát az ugró utasítások végén találunk. Az összehasonlítási műveletek az adott regiszter vagy szám tartalmát mindig az A regiszter tartalmával hasonlítják össze. Az eredmények a jelzőbiteket befolyásolják.

```
Legyen:  A = 'A' regiszter tartalma
         N = Összehasonlítandó regiszter tartalma,
             vagy egy szám
        CF = Átviteli jelzőbit
         Z = Zérus jelzőbit
```

A jelzőbitek alakulása a következő:

A < N	A = N	A > N
CF= 1	CF= 0	CF= 0
Z = 0	Z = 1	Z = 0

9.csoport - Logikai utasítások

```
pld. ADDR+nn   175          XOR  A
```

A logikai műveletek eredménye bitenként:

1 AND 1 = 1	1 OR 1 = 1	1 XOR 1 = 0
0 AND 1 = 0	0 OR 1 = 1	0 XOR 1 = 1
1 AND 0 = 0	1 OR 0 = 1	1 XOR 0 = 1
0 AND 0 = 0	0 OR 0 = 0	0 XOR 0 = 0

Megjegyzés: A logikai utasítások mindig az A regiszter és az utasításban meghatározott regiszter között hajtódnak végre. Az eredményt az A regiszter tartalmazza.

10.csoport - Ugrást végrehajtó utasítások

-abszolút címzés

```
pld. ADDR+nn   195,0,128    JP   32768       ;gépi kódban

     10 RANDOMIZE USR 32768                  ;BASIC-ben
```

-indirekt címzés

```
pld. ADDR+nn   233          JP   (HL)        ;gépi kódban

     10 RANDOMIZE USR (256*PEEK H+PEEK L)    ;BASIC-ben
```

-relatív címzés

```
pld. ADDR+00   24,4         JR   ADDR+06
     ADDR+02   0            NOP
     ADDR+03   0            NOP
     ADDR+04   0            NOP
     ADDR+05   0            NOP
     ADDR+06   24,248       JR   ADDR+00
```

A BASIC-ben a GO TO-hoz hasonlít a legjobban. Relatív címzés előre és hátra is mutathat. Előre ugráskor az utasítás címétől (most ADDR+00) ki kell számolnunk a célhely távolságát (most 6 byte) és ennél kettővel kevesebbet kell írni az utasítás második byte-jának helyére (most ADDR+01-be). Visszaugráskor a startcímtől (jelenleg ADDR+06) ki kell számolnunk a célhely távolságát (ismét 6 byte), és ezt az értéket ki kell vonnunk 254-ből. A kapott eredményt kell a visszaugrási utasítás második

byte-jának helyére írnunk (most ADDR+07-re). Így kaptuk meg jelenleg a 4 ill. a 248 kódokat. Természetesen ez a mintapélda csak demonstráció, nem érdemes futtatni, mert az oda-vissza ugrásból nem tudunk kiszállni, következésképpen "lefagy" a gép. A maximális ugrási távolság előre 127, visszafelé pedig 128 byte. BASIC-ben ezt valahogy így írhattuk volna fel:

```
10 GO TO 60
20 REM
30 REM
40 REM
50 REM
60 GO TO 10
```

-jelzőbitek állapotától függő ugrás

Abszolút és relatív címzéssel használatosak. Ugrás előtt be kell állítanunk a jelzőbitek értékét, az ugrás feltételét, akár összehasonlító művelet, vagy egy logikai feltétel segítségével.

Nézzünk erre két példát:

```
1.pld.  ADDR+00   62,50       LD   A,50
        ADDR+02   254,60      CP   60
        ADDR+04   218,0,128   JP   C,32768
        ADDR+07   201         RET
```

Az összehasonlítás eredménye azt hozza, hogy az átviteli jelzőbit értéke 1-re állítódik be, tehát ugrás történik a 32768-as címre. Ha az ADDR+03. címen 40-et helyeztünk volna el, 60 helyett, akkor az átviteli jelzőbit zérus maradt volna, és az ugrás nem történik meg. Ebben az esetben a következő gépi kódú utasítást (RET) hajtja végre a gép, ami jelen esetben visszatérést jelent a BASIC-hez.

BASIC megfelelője:

```
10 LET A=50
20 IF A<60 THEN GO TO XX
30 STOP
40 REM XX=ugrás sorszáma
```

```
2.pld.  ADDR+00   14,20       LD   C,20
        ADDR+02   13          DEC  C
        ADDR+03   175         XOR  A
        ADDR+04   177         OR   C
        ADDR+05   32,251      JR   NZ,ADDR+02
        ADDR+07   201         RET
```

Ebben a rutinban amíg a C regiszter tartalma nem lesz zérus, addig visszaugrás történik az ADDR+02 címre, ha eléri a zérust, a program futása a következő utasításnál folytatódik (jelen esetben RET).

BASIC megfelelője:

```
10 LET C=20
20 LET C=C-1
30 LET A=0
40 IF C<>0 THEN GO TO 20
50 STOP
```

11.csoport - Ciklusutasítás (DJNZ e)

A hozzá tartozó konstans kiszámítása a JR utasításnál leírtak szerint történik. A ciklusok számát a B regiszter tartalmazza, ha értéke eléri a zérust, kilép a ciklusból és a soron következő utasítást hajtja végre. Ha alaphelyzetben B értékét zérusra állítjuk be, akkor 256 ciklus történik.

```
pld. ADDR+00   6,10          LD   B,10          ;10 ciklus
     ADDR+02   62,0          LD   A,0
     ADDR+03   60            INC  A             ;A értéke a végén
     ADDR+04   16,253        DJNZ ADDR+03       ;10 lesz.
     ADDR+06   201           RET
```

BASIC megfelelője:

```
10 LET B=10: LET A=0
20 FOR i=1 TO B
30 LET A=A+1
40 NEXT i
50 STOP
```

12.csoport - Stack műveletek

-regisztertartalom ideiglenes kimentése a verembe

```
pld. ADDR+00   197           PUSH BC            ;BC a verembe
       .         .             .                ;BC-vel egyéb
       .         .             .                ;műveletek
       .         .             .                ;végezhetők.
     ADDR+nn   193           POP  BC            ;BC elmentett
                                                ;tartalmának
                                                ;visszatöltése.
```

Megjegyzés: A PUSH és POP utasítások egymásba ágyazhatók. Ezt a lehetőséget használhatjuk ki a DJNZ utasítás alkalmazásakor a ciklusszám értékének növelésére.

-stack mutató cseréje egy regiszterpár tartalmával

```
pld. ADDR+nn   227           EX   (SP),HL
```

-szubrutin leágazások

```
pld. ADDR+nn   205,0,128     CALL 32768
```

A visszatérési címet a veremben tárolja. A CALL utasítással megcímzett szubrutinnak RET utasítással kell záródnia. Ha a szubrutin lefutott, a processzor megnézi a SP által jelzett cím tartalmát a veremben, és annak megfelelően visszatér a leágazás helyére. BASIC-ben a GO SUB utasításnak felel meg.

-gépi kódú rutin vagy szubrutin vége

```
pld. ADDR+nn   201          RET
```

Gépi kódú (CALL utasítással meghívott) szubrutin befejezésekor a BASIC RETURN-nek, míg gépi kódú program végén a BASIC STOP-nak felel meg.

-belépés a 16 kbyte ROM egyes területeire
(RST utasítások)

Nyolc ilyen utasítást ismer a Spectrum Z-80. Ezek az utasítások nem általános Z-80 parancsok, csupán a belépést segítik elő a ROM megfelelő területére. Legfőbb előnyük, hogy egy byte-osak, és az adott rekeszt nem kell direkt ugró utasítással meghívnunk (ez esetben 3 byte-ot igényelne az utasítás). Ezek a belépési pontok a ROM legelején a 0. címtől 8 byte-os emelkedéssel helyezkednek el.

```
a. ADDR+nn   199          RST  00   ;Meghívja a ROM-beli RESTART
                                     ;rutint, ez ekvivalens a
                                     ;RANDOMIZE USR 0-val.

b. ADDR+00   207          RST  08   ;Megnyitja a hibacsatornát,
   ADDR+01   10           DEFB 10   ;és a 10-es hibakódnak megfe-
                                     ;lelő "Integer out of range"
             .                       ;hibaüzenet megjelenik.

c. ADDR+00   62,65        LD   A,65 ;Az A regiszterben megadjuk
                                     ;a kiíratandó karakter ASCII
                                     ;kódját.
   ADDR+01   215          RST  16   ;Majd meghívjuk a PRINT ru-
                                     ;tint a ROM-ban.
```

Megjegyzés: A PRINT rutinról az 5. fejezetben még részletesen lesz szó.

```
d. ADDR+nn   223          RST  24   ;Meghívjuk a COLLECT karak-
                                     ;terlehívó rutint. Ha a
                                     ;CH ADD rendszerváltozó ál-
                                     ;tal kijelölt címen nyomtat-
                                     ;ható karakter van, úgy an-
                                     ;nak ASCII kódja az A re-
                                     ;giszterbe kerül.
```

Megjegyzés: Ezt az utasítást könyvünkben a REMKILL rutin alkalmazza. (ld.később)

```
e. ADDR+nn    231          RST  32    ;Ez a COLLECT NEXT CHAR ru-
                                       ;tin meghívása, az előzőhöz
                                       ;hasonló, csak az A regisz-
                                       ;ter az előbbi után követke-
                                       ;ző karakter ASCII kódját
                                       ;fogja tartalmazni.

f. ADDR+00    239          RST  40    ;Belépünk a lebegőpontos
                                       ;kalkulátorba.
   ADDR+01    3            DEFB 3     ;A következő adatbyte a mű-
                                       ;velet (most kivonás) kódja.
                                       ;A kalkulátor az adott műve-
                                       ;letet mindig a lebegőpontos
                                       ;verem felső vagy két leg-
                                       ;felső adatával végzi el.
   ADDR+02    56           DEFB 56    ;Kilépünk a lebegőpontos
                                       ;kalkulátorból.
```

Megjegyzés: A lebegőpontos kalkulátor alkalmazására láthatunk példát a 2. fejezetben.

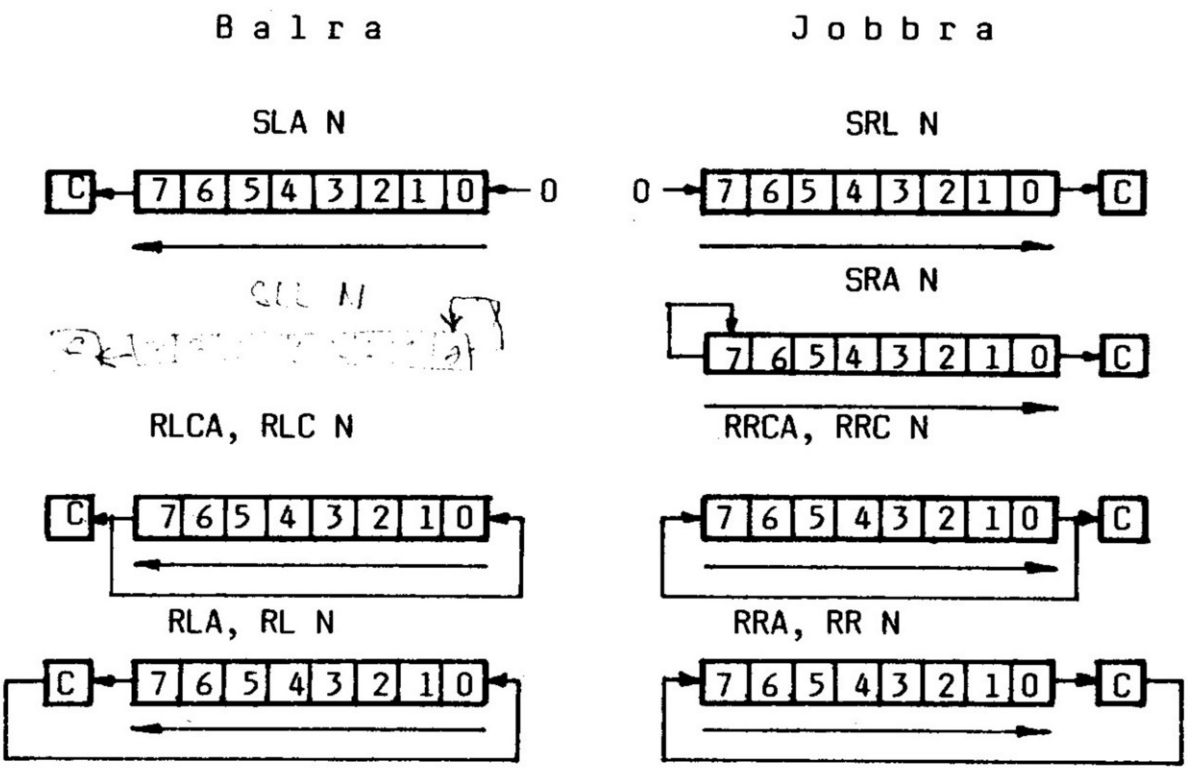
```
g. ADDR+nn    247          RST  48    ;Meghívjuk a MAKE BC SPACES
                                       ;rutint, amely különböző
                                       ;kalkulációkhoz helyet biz-
                                       ;tosít a munkaterületen.
                                       ;A foglalandó helyek számát
                                       ;előzetesen a BC-ben kell
                                       ;megadni.

h. ADDR+nn    255          RST  56    ;Lehetővé teszi maszkolt
                                       ;megszakítás alatt, hogy
                                       ;léptessük a belső órát és
                                       ;használni tudjuk a billen-
                                       ;tyűzetet.
```

Megjegyzés: A megszakítást kezelő rutinoknál alkalmazzuk, az előbb említett feltételek biztosítására.

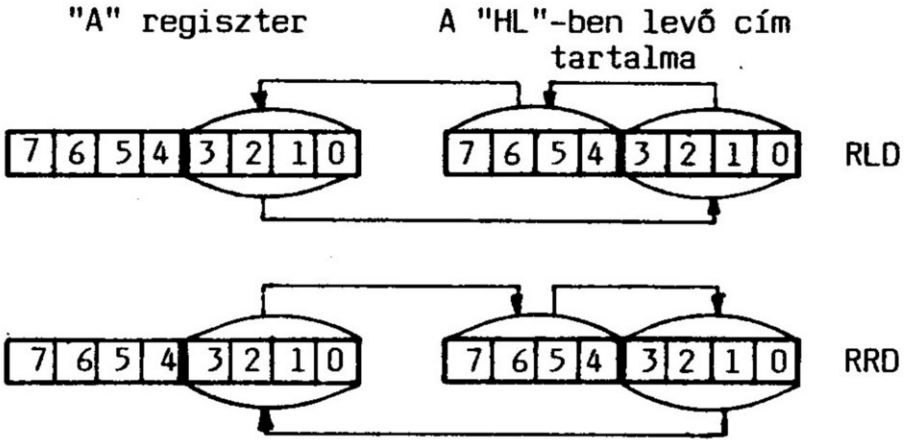
13.csoport - Bit és digitléptető műveletek

Ezek az utasítások általában az adott regiszter tartalmát bitenként léptetik bal vagy jobb irányba, és befolyásolják az átviteli jelzőbit állapotát is. Működésüket kis ábrákkal szemléltetjük. Az ábrákon C az átviteli jelzőbit jelzése, N pedig az utasításhoz tartozó jelzés, általában egy regiszter jele szokott lenni, hogy az adott utasítást melyik regiszter tartalmára hajtsa végre. Amelyik utasítás mellett nem találunk N jelzést, az mindig az A regiszterre értendő. Az RLD és RRD utasítások mindig az A regiszter és a HL regiszterpár által megcímzett rekesz között hajtódnak végre. Ez utóbbiak egyikére hasznos példát találunk a MEMORY ANALYSE rutinban (3.fejezet), a többi bitléptető utasítás inkább a scroll megvalósításánál segít s erre is találunk példákat a scroll rutinok között (4.fejezet).

B I T L É P T E T É S



D I G I T L É P T E T É S



14.csoport - Bitkezelő utasítások

-BIT utasítás

Megvizsgálja az adott regiszter megfelelő bitjét, és a zérus jelzőbitbe annak negáltját helyezi el.

```
pld. ADDR+nn   203,72        BIT  1,B        ;B regiszter első
                                             ;bitjét vizsgálja.
```

-SET utasítás

Az adott regiszter megfelelő bitjét 1-re állítja, függetlenül annak tartalmától.

```
pld. ADDR+nn   203,247      SET  6,A        ;Az A regiszter 6.
                                            ;bitjét 1-re állít-
                                            ;ja.
```

-RES utasítás

A SET ellentéte, az adott bitet törli.

```
pld. ADDR+nn   203,183      RES  6,A        ;Az A regiszter 6.
                                            ;bitjét törli.
```

A SET és RES alkalmazására nézzünk egy egyszerű rutint:

```
     ADDR+00   33,0,88      LD   HL,22528   ;Az ATTR terület
                                            ;kezdőcímét HL-be
                                            ;töltjük.
     ADDR+03   6,3          LD   B,3        ;3-szor 256 ciklus
     ADDR+05   197          PUSH BC         ;BC a verembe.
     ADDR+06   6,0          LD   B,0        ;256 ciklus.
     ADDR+08   203,254      SET  7,(HL)     ;Az adott byte 7.
                                            ;bitjét 1-re állít-
                                            ;juk.
     ADDR+10   35           INC  HL         ;Növeljük a címet.
     ADDR+11   16,251       DJNZ ADDR+08    ;Tovább lépünk, ha
                                            ;már 256-szor meg-
                                            ;történt.
     ADDR+13   193          POP  BC         ;BC-t visszatöltjük
     ADDR+14   16,245       DJNZ ADDR+05    ;Tovább lépünk, ha
                                            ;a külső ciklus már
                                            ;3-szor megtörtént.
     ADDR+16   201          RET             ;Vége
```

Az itt látható rutin hatására a teljes képernyőn FLASH 1 üzemmódot váltunk ki, hisz az ATTR memória byte-ok 7. bitjeit 1-re állítottuk.
Ezt a lehetőséget felhasználhatjuk a FLASH 0, BRIGHT 1 valamint a BRIGHT 0 kiváltására is:

```
     ADDR+08   203,190      RES  7,(HL)     ;FLASH 0
     ---------------------------------------------
     ADDR+08   203,246      SET  6,(HL)     ;BRIGHT 1
     ---------------------------------------------
     ADDR+08   203,182      RES  6,(HL)     ;BRIGHT 0
```

15.csoport - Blokkezelő utasítások

-nem automatikus léptetéssel

pld. ADDR+nn 237,160 LDI

-automatikus léptetéssel

pld. ADDR+nn 237,176 LDIR

Ezeknél az utasításoknál a HL-ben tárolt tartalom átadódik a DE-ben tárolt címre. Nem automatikus esetben BC eggyel csökken, HL és DE pedig egy címmel tovább lép, s mi döntjük el kiadunk-e újabb nem automatikus utasítást. Automatikus esetben ez az átadás mindaddig folytatódik, amíg BC tartalma el nem éri a zérust. Nézzünk alkalmazására egy mintapéldát:

Képernyő-grafikánkat szeretnénk eltárolni a memória hátsó területén, hogy később onnan előhívhassuk. Az átolvasó rutin a következő:

```
ADDR+00   33,0,64     LD HL,16384      ;HL-ben a képernyő kezdő-
                                       ;címe.
ADDR+03   17,64,156   LD DE,40000      ;DE-ben a tárolás kezdő-
                                       ;címe.
ADDR+06   1,0,27      LD BC,6912       ;BC-ben az átmozdítandó
                                       ;byte-ok száma.
ADDR+09   237,176     LDIR             ;Automatikus blokkmozga-
                                       ;tás.
ADDR+11   201         RET              ;Vége
```

Gépeljük be a következő BASIC programot:

```
10 FOR i=10 TO 70 STEP 10
20 CIRCLE 125,75,i
30 NEXT i
40 FOR i=22528 TO 23295
50 POKE i,204
60 NEXT i
```

Ez a program köröket rajzol a képernyőre, majd ATTR értékekkel tölti fel a képernyőt. Feltételezzük, hogy a képet el akarjuk tárolni a memóriában a 40000. címtől, tehát 46911-ig. A rutinnak így ezen a területen kívül kell esnie. Így legyen pl. ADDR=50000.
Gépi kódú rutinunk beolvasása rövidsége miatt mindenképpen DATA sorból javasolt (ami egyébként is egy igen egyszerű módszer a kódok beolvasására), és a következőképpen néz ki:

```
100 FOR i=50000 TO 50011
110 READ a: POKE i,a: NEXT i
120 DATA 33,0,64,17,64,156,1,0,27,
    237,176,201
```

Adjuk ki: RUN 100 (ENTER) és a kódok a memóriába kerülnek az 50000. címtől.
Ezután egészítsük ki a BASIC programot:

```
70 RANDOMIZE USR 50000
80 STOP
```

Futtassuk a programot az elejétől: RUN (ENTER) és a kép eltárolódik a memóriában. Most gépeljük be:

```
CLEAR 39999: NEW (ENTER)
```

Ne ijedjünk meg, a copyright felirat megjelenik a képernyőn, de a kép még a memóriában van.
Visszahívásához meg kell cserélnünk a HL és DE regiszterpárok tartalmát. Ehhez egyszerűen gépeljük be:

```
POKE 50001,64: POKE 50002,156:
POKE 50004,0: POKE 50005,64  (ENTER)
```

Gépeljük be:

```
RANDOMIZE USR 50000 (ENTER)
```

és az eltárolt kép megjelenik a képernyőn.

16.csoport - Adat be/kiviteli (INPUT/OUTPUT) utasítások

Az IN utasítások segítségével adatokat olvashatunk be a billentyűzetről, vagy meghatározott perifériáról, az OUT utasítás segítségével pedig adatokat vihetünk ki különböző perifériákra. Felvetődhet a kérdés, hogy dönti el a processzor, hogy pl. beolvasáskor az adat-byte ne a memóriából kerüljön az adatsínre, hanem pl. a billentyűzetről. Ha a processzor az adatot külső eszköztől várja (a billentyűzetet is ide soroljuk), akkor nem a MREQ* vonalat teszi aktív állapotba az RD* vezérlővonallal egyidejűleg, hanem az IORQ* (INPUT/OUTPUT REQUEST) vonalat. Természetesen íráskor az IORQ* vonal mellett a WR* vezérlő vonal lesz aktív.

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

SY.S SPACE

Az IN utasítást leggyakrabban a billentyűzet leolvasásához használjuk, s mivel itt több probléma felvetődhet, ezt nézzük meg egy kicsit komolyabban.

A billentyűzetet a legcélszerűbb félsorokra osztani. Ezeket a félsorokat megszámozzuk 0-7-ig az ábrának megfelelően.

A beolvasást elvégző IN utasítás előtt mindig el kell helyeznünk az A regiszterben egy értéket, amelyik meghatározza, hogy melyik félsort vizsgáljuk. A továbbiakban a kérdést egy igen tanulságos mintapélda alapján szemléljük végig:

Az a célunk, hogy adott helyen a SYMBOL SHIFT és a SPACE együttes megnyomására lépjen tovább programunk, egyébként várjon. Az említett billentyűk a 7.sz. félsorban találhatók, ebből következik, hogy az A regiszterbe egy olyan érteket kell írnunk, amelynek minden bitje 1-es, csak a 7. bit 0 értékű, tehát az érték 127. Ezzel a módszerrel bármelyik félsorra meghatározhatjuk az A regiszter szükséges értékét, csak mindig a félsor sorszámának megfelelő bitet kell kinulláznunk. Most már megvan az aktuális félsor, a félsoron belül elvégezzük a billentyűzet vizsgálatát. Programunk jelenleg a következő formát mutatja:

```
ADDR+00   62,127     LD   A,127     ;"A"regiszter szerint a
                                    ;jobb alsó félsort vizs-
                                    ;gáljuk.
ADDR+02   219,254    IN   A,(254)   ;A vizsgálat megtörténik,
                                    ;a beolvasott adat-byte
                                    ;az "A" regiszterben van.
```

Természetesen most mindenképpen egy vizsgálatot kell beépíteni az IN utasítás után, amely megnézi az A regiszter tartalmát, és ha megállapítja, hogy a SYMBOL SHIFT és a SPACE egyszerre lett megnyomva, akkor ugorjon tovább, egyébként pedig ugorjon vissza a billentyűzet vizsgáló rutinra.
A következő feladat tisztázni, hogy az A regiszterben milyen érték tárolódott el, a vizsgálatot követően. Egy félsor 5 billentyűt tartalmaz, ábránkon ezeket is megszámoztuk kívülről befelé haladva. Beolvasáskor - ha nem nyomtunk meg billentyűt - akkor a beolvasott byte 0.-4. bitjéig, vagyis az alsó 5 bit biztos hogy 1 értékű lesz. Szándékosan hagytuk ki a felső három bitet, hisz első látásra nincs jelentősége, hogy azoknak a biteknek a tartalma beolvasáskor 1 vagy 0 értékű. Ha megnyomunk egy billentyűt az adott félsorban, akkor a félsorban kijelölt számú bit törlődik. Több billentyű megnyomása esetén az adott billentyűknek megfelelő bitek mind törlődnek, s az adat-byte ennek megfelelően módosul. A SYMBOL SHIFT kívülről számítva a második, a SPACE pedig az első billentyű, vagyis a 0. és az 1. bit törlődik abban az esetben, ha ezt a két billentyűt egyszerre megnyomjuk.
Most térjünk vissza előző problémánkra, milyen érték is íródik be az A regiszterbe.
Ha azt állítjuk, hogy alaphelyzetben (tehát akkor ha nincs billentyű megnyomva) a beolvasott adat-byte minden bitje 1 értékű, akkor 255 lesz beolvasva, viszont ha megnyomjuk az említett két billentyűt, akkor az alsó két bit törlődik, azaz ebből hármat le kell vonnunk, az eredmény 252. Ez a megközelítés viszont hibás és sajnos több szakirodalom feltételezi alapállapotban a 8 bit

aktív állapotát. Ebből adódóan van akinek fut az adott rutin a gépén, van akinek nem. Ez annak a következménye, hogy a számunkra érdektelen felső három bit értéke nem minden esetben 1 értékű. A Spectrum kézikönyvből megtudhatjuk, hogy a 6. bit az EAR csatlakozón érkezett jelet olvassa le. Ennek a bitnek az alapállapota nem minden esetben 1 értékű. Bárki meggyőződhet róla, hogy saját gépén milyen ez az alapállapot:

```
10 PRINT AT 0,0; IN 63486
20 GO TO 10
```

Ha futtatáskor a képernyő bal felső sarkában 255 jelenik meg, akkor az említett bit alapértéke 1, 192 esetén pedig zérus. Ha jelet kapcsolunk az EAR csatlakozóra, az érték 192 és 255 között vibrálva oda-vissza vált, azaz a 6. bit hol 1, hol zérus értékű, a beolvasott jeleknek megfelelően.

Az itt említett probléma elkerülésére legcélszerűbb, ha a beolvasott byte felső három bitjét töröljük egy megfelelő AND utasítás kiadásával, és a maradék 5 bitre, már biztosan rendben elvégezhetjük a vizsgálatot. Mintapéldánk tehát a következőképpen folytatódik:

```
ADDR+04   230,31     AND  31          ;A beolvasott byte felső
                                      ;három bitjét töröljük.
ADDR+06   254,28     CP   28          ;Ha a SYMBOL SHIFT és a
                                      ;SPACE billentyűt egy-
                                      ;szerre lenyomjuk, a zé-
                                      ;rus jelzőbit (Z) 1-re
                                      ;vált.
ADDR+08   32,246     JR   NZ,ADDR+00  ;Ha nincs 1-es érték a Z
                                      ;jelzőbitben, akkor ug-
                                      ;rás vissza,
ADDR+10   201        RET              ;egyébként vége.
```

Ha az elkészült mintapéldát beolvassuk a memóriába (tetszőleges szabad helyre) és futtatjuk (RANDOMIZE USR ADDR+00), látszólag "lefagy" a gép, pedig semmi mást nem csinál, csak várja a SYMBOL SHIFT és a SPACE együttes megnyomását. Ha ezeket a billentyűket megnyomjuk, megjelenik az OK üzenet.

Az OUT utasítások adat-byte kiírását teszik lehetővé perifériákra. A hangszórót is perifériaként kezeli a rendszer a 254-es port-on így az "OUT (254),A" utasítás segítségével érdekes hanghatásokat érhetünk el. A hangok gépi kódú elérésére több példát találhatunk a "SINCLAIR Spectrum játék és program" c. könyvben LSI ATSz. 1986.

17.csoport - A megszakítást kezelő utasítások

A megszakítást kezelő rutinokra később találunk még példát (pl. CLOCK), itt csak a leglényegesebb elemeket emeljük ki.

```
pld. ADDR+00   62,254        LD   A,254
     ADDR+02   237,71        LD   I,A
     ADDR+04   237,94        IM   2          ;Áttérünk 2-es meg-
                                             ;szakítási módba.
     ADDR+06   201           RET
     ------------------------------------------------------------
     ADDR+07   62,63         LD   A,63
     ADDR+09   237,86        IM   1          ;Visszatérünk 1-es
                                             ;megszakítási módba,
     ADDR+11   237,71        LD   I,A        ;és I regiszterbe
                                             ;visszatöltjük a 63-
                                             ;as megszakítási
                                             ;vektor értéket.
     ADDR+13   201           RET
```

A 2-es megszakítási módra való áttérés azt jelenti, hogy minden 1/50-ed másodpercben végrehajtunk egy felhasználói gépi kódú rutint. A felhasználói rutin kezdőcíme nem tetszőleges, de könnyen származtatható, annak alsó és felső byte-ját egy külön "NN" címen kell elhelyezni.Az NN cím a:

(256*I regiszter tartalma)+H

képletből számolható ki, ahol H az un. HARDWARE szám. Ez utóbbi általában 255, abban az esetben, ha a megszakítást kérő jel nem külső eszköztől származik. Ennek alapján felhasználói rutinunk kezdőcímének alsó/felső byte-ját a 65279/65280 címeken kell elhelyezni. Félre ne értsük, nem az itt közölt néhány sor a rutin, amit futtatni akarunk, amit itt látunk az csak a kapcsoló, amivel a RANDOMIZE USR ADDR+00 utasítás kiadásával térhetünk át a 2-es megszakítási módba, és a RANDOMIZE USR ADDR+07 utasítással kapcsolhatunk vissza.
A kapcsoló rutin bárhol elhelyezhető a futtatni kívánt rutin mellett a memória szabad területén. Természetesen általában összetoljuk ezeket, hogy minél kevesebb helyet foglaljanak el.

18.csoport - Egyéb utasítások

-akkumlátor tartalmának bitenkénti negálása

```
   ADDR+nn   47          CPL
```

-akkumlátor kettes komplemensének előállítása

```
   ADDR+nn   237,68      NEG
```

-átviteli jelzőbit 1-be állítása

```
   ADDR+nn   55          SCF       ;használjuk a scroll ru-
                                   ;tinokban.
```

-átviteli jelzőbit komplementálása (1-ből 0 ill. 0-ból 1 lesz)

```
ADDR+nn   63          CCF         ;használjuk a scroll ru-
                                  ;tinokban.
```

-akkumlátor tartalmának BCD korrekciója

```
ADDR+nn   39          DAA         ;használjuk a CLOCK ru-
                                  ;tinban.
```

-processzor utasításvégrehajtásának felfüggesztése a következő megszakítás kérésig

```
ADDR+nn   118         HALT
```

Az utasításcsoportok áttekintése után maradt még tisztázandó kérdés. Itt-ott találkozunk az assembly listában olyan mnemonikokkal, melyek egyik csoportban sem találhatók meg. Ezek többsége a DEF... formát mutatja (DEFINITION).
Ezeket az utasításokat assembler utasításoknak nevezzük, használatuk megkönnyíti rutinjaink elkészítését, ha assemblert használunk.
Nem célunk mélyrehatóan foglakozni az assembler programok használatával, más irodalmakban ez részletesen megtalálható. Itt most kizárólag azokat a jelölésformákat ismertetjük, melyek a könyvben található rutinok assembly listáiban előfordulnak.

- DEFB x - ahol x egy 0-255 közötti szám.
  Ez a jelölés egyetlen adat-byte-ot határoz meg.
  Használjuk a ROM rutinokban, pl. az RST 08 utasítás után adat-byte formájában közöljük a hibakódot, vagy pl. az RST 40 utasítás után a műveleti kódokat is.

- DEFM "karakterlánc"
  - A harmadik fejezetben, az EXAMINE rutinban találkozunk vele, és egy meghatározott string memóriabeli elhelyezkedését szemlélteti.

- DEFS x - ahol x egy 0-255 közötti szám.
  Ez a jelölés arra utal, hogy a memória meghatározott helyén x számú byte-ot üresen hagyunk, későbbi célokra. Szintén az EXAMINE rutinban láthatunk erre példát, a 17 byte-os fejléc helybiztosítására.

- DEFW NN - ahol NN egy 0-65535 közötti szám.
  A számot két byte-on tároljuk.
  A 3.fejezetben a FREE MEMORY rutinban találkozunk ezzel.

Reméljük az itt közölt DEFiníciók elősegítik azoknak a rutinoknak a megértését, ahol ezek a jelölésformák előfordulnak.

A fejezet végén néhány szót szeretnénk szólni a hibakeresésről, elsősorban azoknak, akik nem vállalkoznak az assembler programok használatára, és egyszerű BASIC DATA sorból próbálják beolvasni a kódokat a memóriába.

-Begépelés után a programot mentsük ki magnetofonra vagy microdrive-ra, mert ha kipróbáláskor "elszáll", kezdhetjük újra begépelni.

-Adjuk ki RUN (ENTER), azaz olvastassuk be az adatokat a memóriába. Ha a beolvasás "E out of DATA" hibajelzéssel megáll, akkor egy vagy több adatot nem adtunk meg, vagy a címet írtuk be esetleg hibásan. Listázzuk ki a programot, és hasonlítsuk össze a DATA sort az eredeti gépi kódú számsorral.

-Ha a beolvasás "OK" üzenettel fejeződik be, még mindig előfordulhat, hogy egy vagy több adatot hibásan gépeltünk be, főleg hosszabb program esetén. A másolatunk rendelkezésre áll, úgyhogy megpróbálhatjuk futtatni a gépi kódot (RANDOMIZE USR ...).

-Ha elszáll a rutin, ismét töltsük be a kimentett programot a tároló eszközről, és végezzük el az adatok ellenőrzését.

-A hiba kijavítása után programunkat mentsük ki a javított változatban is, és futtassuk újra.

-Problémamentes futás esetén kimenthetjük gépi kódú rutinunkat végleges formájában is: SAVE"név"CODE kezdőcím,hossz (ENTER).

Assembler programok használatánál kisebb a hiba lehetősége, de itt is célszerű a forrásszöveget kimenteni mielőtt az assemblált kódot futtatnánk, mert ha hiba lenne, a forrásszöveget visszatölthetjük és a hiba kijavítható.

És végül, de nem utolsó sorban hívjuk fel a figyelmet arra, hogy gépi kódú programunk bárhol is helyezkedjen el a memóriában, a RAMTOP-ot mindig állítsuk át ennek megfelelően.
(CLEAR ADDR-01, vagyis ha kezdőcímünk 60000, akkor CLEAR 59999.)

2.fejezet

BASIC VAGY GÉPI KÓD ?

2.1 Az első megközelítés

Lehetetlen feladat néhány oldalban aprólékos útmutatót adni arra, hogy mikor és miért érdemesebb a gépi kódot használni.
Ebben a fejezetben néhány tanácsot szeretnénk adni az átvezető út behatárolásához.

-Nem érdemes így kezdeni: "Melyik BASIC utasításnak mi a gépi kódú megfelelője ? Melyiket érdemesebb inkább gépi kódban használni ?" Egy utasítás önmagában nagyon kevés annak megállapításához, hogy programunkban megtérül-e a befektetett energia. Sokkal több szempontot kell figyelembe vennünk ennek kiderítéséhez.
-Programunkat mindig egészben kell vizsgálni. A gépi kódú programozás szempontjából meghatározó szerepe van a program jellegének, hosszának, memóriaigényének, és nem utolsó szempont a futási sebessége sem.
-Játék jellegű programok esetén fontos a folyamatosabb mozgás, gyorsabb reakció lehetőség stb., így nem járunk rosszul, ha hosszától függetlenül gépi kódban írjuk meg.
-Egyéb kategóriáju BASIC programunk igen sokféle jellegű lehet, s így a "mit érdemes ?" kérdés megválaszolása igen nehéz.
-A matematikai jellegű programok:függvénydefiníciók, egyenletek, stb. futási ideje gépi kódban is igen nagy, s ha még azt is figyelembe vesszük, hogy memóriaigényük általában nem nagy mértékű, akkor bátran kimondhatjuk, ezeket BASIC-ben érdemesebb futtatnunk.
-A memóriamanipulációk: tömbmozgatások, tömbrendezések stb. gépi kódban összehasonlíthatatlanul gyorsabban lefutnak, mint pl. BASIC-ben.
-Az összetettebb grafikai utasítások (pl. DRAW, CIRCLE) rutinjai a ROM-ban találhatók. BASIC-ből az Interpreter is ezeket a rutinokat aktivizálja, így ezeknek a rutinoknak a futási ideje még akkor sem lesz számottevően gyorsabb, ha gépi kódból történik meghívásuk.
-A döntési utasítások BASIC-ben jelentősen lelassítják a program futási sebességét, ezt kiküszöbölhetjük a gépi kód segítségével.
-Néhány lehetőség BASIC-ből nem érhető el (pl. megszakítási rutinok kezelése, bitkezelés, stb.), ezeket tehát mindenképpen csak a gépi kód segítségével oldhatjuk meg.

Az itt felsorolt néhány szempont, valamint a fejezetben található mintapéldák nem adják meg a pontos választ a fejezet címére, de lehetőséget biztosítanak ahhoz, hogy a felhasználó elinduljon a válaszkeresés útján. Sok gyakorlásra és munkára van

szükség, hogy megállapítsuk, programunk milyen formában lásson napvilágot, elérve a leggazdaságosabb programozási időbefektetést és a megkívánt futási időt.

2.2 Aritmetikai műveletek

Induljunk el a következő mintapéldán. Vegyünk két számot, legyen az egyik 20, a másik pedig 100. Ezekkel a számokkal végezzünk el tetszőleges aritmetikai műveleteket, és ne csak egyszer, hanem többször, pl. 1000-szer egymás után. Az erre alkalmas programot készítsük el BASIC-ben és gépi kódban is, a programokat pedig két változatban futtassuk. Először úgy, hogy minden számítás eredménye jelenjen meg a képernyőn, a második esetben pedig ezt elhagyjuk. Így arról is tájékoztatást fogunk kapni, hogy a PRINT mennyire lassítja le a program futását.

Elsőként válasszuk aritmetikai műveletnek az összeadást. Gépeljük be a következő BASIC programot:

```
10 LET a=20
20 LET b=100
30 FOR i=1 TO 1000
40 LET c=a+b
50 PRINT AT 0,0;c
60 NEXT i
```

Futtatáskor azt látjuk, hogy a képernyő bal felső sarkában megjelenik a 120, az összeadás eredménye. Az OK üzenetre kb. fél percet kell várnunk, ennyi idő alatt fut le a program.
Vegyük ki az 50-es sort, és anélkül futtassuk újra programunkat. A futási idő jelentősen lerövidül, mintegy 10 másodpercre.
Ha ezt is kipróbáltuk bátran írjuk meg az itt látható BASIC program gépi kódú megfelelőjét:

Tudnunk kell, hogy ha használni akarjuk gépi kódban is a képernyő felső 22 sorát, akkor legelőször is meg kell nyitni a képernyő megfelelő csatornáját.

```
ADDR+00   62,2        LD   A,2       ;A 2 a képernyő felső ré-
                                     ;szének csatornája.
ADDR+02   205,1,22    CALL 5633      ;Ez egy ROM rutin, amely
                                     ;az 5633. (HEX-1601) cí-
                                     ;men található és elvégzi
                                     ;a csatorna megnyitását.
```

Az aritmetikai műveletet ezerszer hajtjuk végre, tehát legcélszerűbb ha egy regiszterpár tartalmát 1000-rel töltjük fel. Ezt majd mindig egyenként csökkentjük, s ha elérte a zérust, akkor a futásnak vége. Aritmetikai műveletek, s így az összeadás is a ROM lebegőpontos kalkulátorában hajtódnak végre. A lebegőpontos kalkulátor felhasználja saját célra a processzor belső regisztereit, így annak a regiszternek a tartalmát, amelybe az ezret töltjük, még a lebegőpontos kalkulátor meghívása előtt ki kell menteni az átmeneti tárolóba, a verembe. Legyen a válasz-

tott regiszterpár a DE (azért nem egy regisztert választottunk, mert egy regiszterbe max. csak 255-öt tudunk tölteni).

```
ADDR+05   17,232,3      LD   DE,1000    ;A DE regiszterpárba ez-
                                        ;ret töltünk, és ezt ki-
ADDR+08   213           PUSH DE         ;mentjük a verembe.
```

Láthatjuk, hogy az ezret hogyan kell beültetni a memóriába. Mindig elől van az alsó byte, amelynek értéke 0-255-ig terjed, és utána következik a felső byte, melynek értékét 256-tal szorozni kell.

232+(3*256)=1000

Számok alsó/felső byte bontására előnyösen felhasználható a SEED rendszerváltozó. Ha kiadjuk: RANDOMIZE nn (ENTER), ahol nn az a 0 és 65535 közé eső szám amelynek az alsó/felső byte-ját keressük, akkor a PRINT PEEK 23670 (ENTER) megadja a szám alsó, míg a PRINT PEEK 23671 (ENTER) pedig a felső byte-ját. Próbáljuk ki az ezer esetében, meglátjuk az eredmény stimmel.

A lebegőpontos kalkulátor az aritmetikai műveletekhez a lebegőpontos vermet használja. Itt jegyezzük meg, hogy a lebegőpontos vermet nem szabad összekeverni a processzor által használt processzor veremmel (amit a PUSH-POP művelettel is használunk). Ez utóbbiról az első fejezetben már volt szó.
A processzor a lebegőpontos kalkulátor működtetésekor mindig a lebegőpontos verem tetején levő számmal (pl. szinusz számítás, vagy négyzetgyökvonás esetén), vagy számokkal (pl. összeadás, vagy szorzás esetén) hajtja végre az aritmetikai műveletet. Összeadáskor a verem két legfelső elemét adja össze, és az eredményt visszatölti a verem tetejére.
Adatok bevitele a lebegőpontos verembe legcélszerűbben a BC regiszterpár segítségével lehetséges.

```
ADDR+09   1,20,0        LD   BC,20      ;A BC regiszterpárba 20-
                                        ;at töltünk(az egyik ösz-
                                        ;szeadandó).
ADDR+12   205,43,45     CALL 11563      ;Meghívjuk a ROM 11563.
                                        ;(HEX-2D2B)címén levő ru-
                                        ;tint, amely a BC tartal-
                                        ;mát tölti át a lebegő-
                                        ;pontos verembe.
ADDR+15   1,100,0       LD   BC,100     ;A BC regiszterpárba 100-
                                        ;at töltünk (a másik ösz-
                                        ;szeadandó).
ADDR+18   205,43,45     CALL 11563      ;BC tartalmát ismét át-
                                        ;töltjük a lebegőpontos
                                        ;verembe.
```

A második művelet végrehajtásakor a 100 került a lebegőpontos verem tetejére, a 20 pedig a verem második eleme lett. Ezután már meghívhatjuk a lebegőpontos kalkulátort.

```
ADDR+21   239          RST  40      ;Belépünk a lebegőpontos
                                     ;kalkulátorba.
ADDR+22   15           DEFB 15      ;A 15-ös kód azt jelenti,
                                     ;hogy a kalkulátor adja
                                     ;össze a lebegőpontos ve-
                                     ;rem tetején levő két
                                     ;elemet. Végrehajtás után
                                     ;az eredmény a verem te-
                                     ;tején lesz.
ADDR+23   56           DEFB 56      ;Az 56. kód jelzi a kal-
                                     ;kulátornak, hogy több
                                     ;számítás nincs, kilépés
                                     ;történik.
```

A lebegőpontos verem tetején levő elemet (az eredményt)ismét egy erre alkalmas ROM rutin segítségével írathatjuk ki a képernyőre. A képernyő felső részét már megnyitottuk, a kiíratás előtt pozícionálnunk kell a kiírás helyét. Ez legyen az AT 0,0 pozíció, azaz a képernyő bal felső sarka.

```
ADDR+24   62,22        LD   A,22    ;Az AT kódja.
ADDR+26   215          RST  16
ADDR+27   62,0         LD   A,0     ;A sor pozíciója.
ADDR+29   215          RST  16
ADDR+30   62,0         LD   A,0     ;Az oszlop pozíciója.
ADDR+32   215          RST  16
ADDR+33   205,227,45   CALL 11747   ;Meghívjuk a ROM 11747.
                                     ;(HEX-2DE3) címen talál-
                                     ;ható rutinját, amely a
                                     ;lebegőpontos verem te-
                                     ;tején levő elemet kiírja
                                     ;a képernyő előre megha-
                                     ;tározott helyére.
```

A pozícionálásról ill. átfogóan a PRINT-ről az 5. fejezetben bővebben van szó.
A műveletet elvégeztük, a processzor veremből elővehetjük az előzőleg eltárolt értéket, ami most 1000.

```
ADDR+36   209          POP  DE      ;DE tartalmát visszatölt-
                                     ;jük a veremből.
```

DE tartalmát csökkenteni kell eggyel, és mindig meg kell vizsgálni, hogy elérte-e a zérust. Ha igen, akkor egy RET utasítás segítségével kilépünk, vagyis a futásnak vége, ha nem akkor pedig visszavezéreljük rutinunkat egészen az ADDR+08. címre, azaz DE új értékét ismét eltesszük a verembe, és a számítást megismételjük.

```
ADDR+37   27           DEC  DE      ;DE-t csökkentjük.
ADDR+38   122          LD   A,D     ;A D regiszter tartalmát
                                     ;áttöltjük az A regisz-
                                     ;terbe.
```

```
ADDR+39   179          OR   E          ;Az A tartalmát, ami meg-
                                       ;egyezik D tartalmával OR
                                       ;kapcsolatba hozzuk E-re-
                                       ;giszter tartalmával. A
                                       ;zérus jelzőbit csak ak-
                                       ;kor lesz egyes, ha a DE
                                       ;regiszterpár zérus.
ADDR+40   32,222       JR   NZ,ADDR+08 ;Ellenkező esetben még az
                                       ;ezerrel nem végeztünk,
                                       ;és ugrás történik vissza
                                       ;az ADDR+08. címre.
ADDR+42   201          RET             ;Ha a zérus jelzőbit 1
                                       ;volt, akkor vége.
```

A program tehát önmagában semmitmondó, hisz egymás után ezerszer ugyanazt végezzük el, most a mintapélda az összehasonlítás célját szolgálja.
A gépi kódú rutin ebben a formájában 11 másodperc alatt lefut, ha viszont itt is elhagyjuk a PRINT funkciót, vagyis kinullázzuk az ADDR+24-ADDR+35 közötti memóriaértékeket, akkor nem egész 1 másodperc a futás ideje.
A programot próbáljuk ki különböző aritmetikai műveletek végrehajtásával is. A BASIC programban a 40. sort kell módosítani,pl. szorzáshoz:

```
40 LET c=a*b, vagy
```

négyzetgyökvonáshoz:

```
40 LET c=SQR b
```

A gépi kódú programban az ADDR+22. címen levő értéket kell megváltoztatni a műveletnek megfelelően (összeadás= 15; kivonás= 3; szorzás = 4 ; osztás = 5 ; szinusz = 31 ; természetes alapú logaritmus = 37 ill. négyzetgyökvonásnál = 40).
Itt most csak a négy alapműveletet és három fontosabb aritmetikai műveletet emeltünk ki, mivel ezekkel végeztünk pontos időbemérést az összehasonlításhoz.
A futási idő leméréséhez bárki felhasználhatja a 3. fejezetben ismertetett CLOCK rutint a stopper változatban. Ez igaz, hogy csak két század másodperces pontosságut ad, de ez bőven elegendő az arányok érzékeléséhez.
A BASIC program elején a RANDOMIZE USR 65159 utasítással kapcsolható be a stopper, a kikapcsolás pedig a RANDOMIZE USR 65152 utasítással történik. A stopper indítása előtt az időértéket le kell nullázni !
A gépi kódú program első utasítása lehet a CALL 65159 és utolsó előtti a CALL 65152 a stopper vezérléséhez. A gépi kódú program utolsó utasítása a RET, a visszatéréshez.
Az előbb megemlített 7 aritmetikai művelet futási időeredményeit a következő táblázat tartalmazza:

Művelet	BASIC		gépi kód	
	PRINT-tel	PRINT nélkül	PRINT-tel	PRINT nélkül
+	00:25:88	00:09:68	00:11:30	00:00:98
-	00:26:20	00:09:86	00:11:60	00:01:14
*	00:27:40	00:10:10	00:12:80	00:01:38
/	00:34:58	00:12:18	00:12:96	00:03:44
SIN	01:24:74	00:56:84	01:10:40	00:48:42
LN	01:47:80	01:27:24	01:33:76	01:18:62
SQR	02:38:48	02:20:04	02:24:24	02:11:44

Az időérték perc:másodperc:századmásodperc felosztásban értendő!

Jól látható, hogy a bonyolultabb aritmetikai műveletek végrehajtási ideje gépi kódban is elég magas, így megalapozott az a vélemény, hogy csak indokolt esetben írjuk át aritmetikai műveletekkel teletűzdelt BASIC programunkat gépi kóddá.

2.3 Keresés és rendezés

Ez két olyan fontos terület, amelynél a döntési funkciók alkalmazása miatt BASIC-ben igen lassú a végrehajtási idő.

A keresésre a legegyszerűbb bemutató példa, hogy egy meghatározott adat-byte előfordulását keressük a memória meghatározott területén.

Mintapéldánkban nézzük meg, hány darab 201-es adatbyte van a ROM területen, tehát 0-16383-ig.

Gépeljük be a következő BASIC programot:

```
10 CLS
20 INPUT a: REM kezdőcím
30 INPUT b: REM hossz
40 INPUT c: REM keresett adat-byte
50 LET d=0
60 POKE 23692,255
70 FOR i=a TO a+(b-1)
80 IF PEEK i=c THEN PRINT i: LET d=d+1
90 NEXT i
100 PRINT: PRINT d;" db."
```

A 23692-es címen azért adtunk meg 255-öt, hogy a "scroll?" üzenet megjelenését a folyamatos PRINT-elés mellett átmenetileg megszüntessük.
Ha a BASIC programot futtatjuk, a futási eredmény: 206 db.

Lemértük a futási időket is:

```
PRINT-tel:      03:32:34
PRINT nélkül:   03:16:86
```

Írjuk meg a program gépi kódú megfelelőjét:

Ha a képernyő előzetes tartalmát törölni akarjuk, előbb meg kell hívnunk egy alkalmas ROM rutint, amely ezt minden probléma nélkül elvégzi.

```
ADDR+00   205,107,13   CALL 3435      ;Meghívjuk a 3435. (HEX-
                                      ;0D6B) címen levő ROM ru-
                                      ;tint, amely törli a kép-
                                      ;ernyőt.
```

A már ismert módon megnyitjuk a képernyő felső részét:

```
ADDR+03   62,2         LD   A,2       ;A képernyő felső részé-
                                      ;nek csatornája.
ADDR+05   205,1,22     CALL 5633      ;Megnyitjuk a csatornát.
```

A "scroll?"-t ideiglenesen megszüntetjük:

```
ADDR+08   62,255       LD   A,255     ;Az A regiszterbe 255-öt
                                      ;töltünk és ezt az érté-
ADDR+10   50,140,92    LD   (23692),A ;ket áttöltjük az SCR CT
                                      ;rendszerváltozóba.
```

A keresés kezdőcímét és a vizsgált memóriarész hosszát célszerű a rutin területén kívül elhelyezni, ahonnan ezeket a rutin ki tudja olvasni.

```
ADDR+13   42,0,91      LD   HL,(23296) ;HL regiszterpárba tölt-
                                       ;jük a 23296/97 címek
                                       ;tartalmát.
ADDR+16   237,91,2,91  LD   DE,(23298) ;DE-be töltjük a 23298/99
                                       ;címek tartalmát.
```

Ez azt jelenti, hogy a 23296/97 címekre alsó/felső byte formában el kell helyezni a rutin meghívása előtt a vizsgált terület kezdőcímét, jelen példánkban zérust, ill. a 23298/99. címeken a hosszt, ami most példánkban 16384. Ennek az értéknek az alsó/felső byte-ja: 0/64, így a megfelelő címekre ezt kell beírni.
HL veszi fel a kezdőcím és DE a hossz értékét.
Ha meg akarjuk számoltatni, hogy hány darab 201-es byte van a ROM területen, célszerű egy regiszterpárt kinevezni, amelyik a végrehajtás alatt mindig növeli eggyel az értékét, ha a gép 201-es byte-ot talált, s így megszámolja hány darab van belőle. Ennek természetesen zérus alapértéket kell megadnunk.

```
ADDR+20   1,0,0        LD   BC,0      ;BC fogja számolni a 201-
                                      ;es byte-ok számát.
```

Az adatok beállítása megtörtént, most meg kell írni a végrehajtó rutint, amely egy belső magból fog állni. Megvizsgáljuk, hogy az adott címen 201-es byte van-e ? Ha igen, akkor a futás a belső magon folytatódik, ahol a képernyőre kiíratjuk az aktuális címet és természetesen léptetjük a számlálót. Ellenkező esetben ezt a magot egyszerűen átugorjuk.

```
ADDR+23   126          LD   A,(HL)      ;Az A regiszterbe beol-
                                        ;vassuk az aktuális cím
                                        ;tartalmát.
ADDR+24   254,201      CP   201         ;Itt nézzük meg, hogy a
                                        ;201-es byte-e ? Ha igen,
                                        ;akkor a zérus jelzőbit 1
                                        ;értéket vesz fel és nem
                                        ;történik elugrás, másk-
ADDR+26   32,18        JR   NZ,ADDR+46  ;ülönben ugrás az ADDR+46.
                                        ;címre.
ADDR+28   229          PUSH HL          ;Kimentjük a regiszterpá-
ADDR+29   213          PUSH DE          ;rok aktuális értékeit,
ADDR+30   197          PUSH BC          ;mivel a lebegőpontos
                                        ;adatátvitel használja
                                        ;ezeket a regisztereket.
ADDR+31   229          PUSH HL          ;Itt nem ideiglenes táro-
ADDR+32   193          POP  BC          ;lásról van szó, hanem a
                                        ;legegyszerűbb módszert
                                        ;látjuk regiszterpárok
                                        ;tartalmának áttöltésére
                                        ;egy másik regiszterpár-
                                        ;ba.
ADDR+33   205,43,45    CALL 11563       ;BC tartalmát áttöltjük a
                                        ;lebegőpontos verembe.
ADDR+36   205,227,45   CALL 11747       ;A lebegőpontos verem
                                        ;tartalmát kiíratjuk a
                                        ;képernyőre.
ADDR+39   193          POP  BC          ;A kimentett regiszter-
ADDR+40   209          POP  DE          ;tartalmakat a megfelelő
ADDR+41   225          POP  HL          ;sorrendben visszatölt-
                                        ;jük a veremből.
ADDR+42   62,13        LD   A,13        ;Az ENTER kódja, vagyis
ADDR+44   215          RST  16          ;"kocsi vissza"és soreme-
                                        ;lés történik.
ADDR+45   3            INC  BC          ;A számlálót növeljük.
ADDR+46   35           INC  HL          ;A vizsgálat címét növel-
                                        ;jük.
ADDR+47   27           DEC  DE          ;A hossz értékét csök-
ADDR+48   122          LD   A,D         ;kentjük, D-t A-ba tölt-
ADDR+49   179          OR   E           ;jük és megnézzük, hogy
                                        ;DE elérte-e a zérust ?
ADDR+50   32,227       JR   NZ,ADDR+23  ;Ha nem érte el, ugrás az
                                        ;ADDR+23. címre.
```

Ez a ciklus addig dolgozik, amíg DE értéke zérus nem lesz, vagyis nem kell több címet vizsgálni. Megtörténhet a végleges kiértékelés, kiíratjuk BC tartalmát, ami a 201-es kódot tartalmazó címek számát számolja össze.

```
ADDR+52   62,13        LD   A,13        ;Az ENTER kódja.
ADDR+54   215          RST  16          ;Sort emelünk.
ADDR+55   205,43,45    CALL 11563       ;BC-t áttöltjük a lebe-
                                        ;gőpontos verembe és ki-
```

```
ADDR+58    205,227,45    CALL 11747      ;íratjuk a képernyőre.
ADDR+61    62,32         LD   A,32       ;A SPACE kódja.
ADDR+63    215           RST  16         ;Hagyunk egy SPACE-t.
ADDR+64    62,100        LD   A,100      ;A kis "d"betű ASCII kód-
                                         ;ja.
ADDR+66    215           RST  16         ;A kis"d"betű megjelenik.
ADDR+67    62,98         LD   A,98       ;A kis "b"betű ASCII kód-
                                         ;ja.
ADDR+69    215           RST  16         ;A kis"b"betű megjelenik.
ADDR+70    62,46         LD   A,46       ;A "." ASCII kódja.
ADDR+72    215           RST  16         ;A pont megjelenik.
ADDR+73    201           RET             ;Vége.
```

Ha PRINT nélkül akarjuk futtatni a rutint, akkor az ADDR+28.-tól az ADDR+44. címig írjunk be zérus byte-ot (NOP).
A rutinnál mért végrehajtási idők a következők:

PRINT-tel: 00:15:22
PRINT nélkül: 00:00:40

Természetesen a rutin felhasználásával rövidebb, vagy hosszabb területet és más adat-byte előfordulását is vizsgálhatjuk, de ahogy ez a példából is kiderült inkább gépi kódban, mint BASIC-ben.

A tömbrendező program a végrehajtási idő tekintetében még élesebb határt húz a BASIC és a gépi kód közé.
Helyezzünk el 1000 db.0-254 közé eső véletlen számot tetszőleges helyen a memóriában. A véletlen elemek kezdőcíme legyen most a 31000. memóriarekesz, tehát az ezredik elem a 31999. címen fog elhelyezkedni.
A véletlen számok generálása BASIC-ben könnyebb, és még ezer elemes minta esetén is viszonylag gyors, ezért ezt BASIC-ben oldjuk meg. Az elemek nagyságrendbe szedése már nem mindegy, hogy BASIC-ben történik meg, vagy gépi kódban, ezt meg is tekintjük.
A véletlen elemek beolvasására gépeljük be a következő BASIC programot:

```
10 FOR i=31000 TO 31999
20 POKE i,INT(255*RND)
30 NEXT i
40 STOP
```

Futtassuk: RUN (ENTER) és mintegy fél perc alatt az 1000 véletlenszám a memóriában van. Erről könnyen meggyőződhetünk, ha begépeljük és futtatjuk a következő sort:

```
2000 FOR i=31000 TO 31999: PRINT i;"=";PEEK i:
     NEXT i
```

Nem érdemes természetesen végignézni az ezer elemet, hiszen az elejétől a végéig csak a 0-254-ig terjedő véletlen számok sorozatát látjuk.
A közismert nevén "buborék" módszernek is ismert sorbarendező mechanizmus BASIC változata jelen példára a következő:

```
 50 FOR i=31999 TO 31001 STEP-1
 60 FOR j=31001 TO i
 70 IF PEEK j>PEEK (j-1) THEN GO TO 110
 80 LET a=PEEK j
 90 POKE j,PEEK (j-1)
100 POKE (j-1),a
110 NEXT j
120 NEXT i
130 STOP
```

A programot futtathatjuk: RUN (ENTER), de nem érdemes megvárni, amíg lefut, mert kb. 4 és fél óra alatt fejezné be működését.

A gépi kódú program megírását kezdjük a kiinduló adatok beolvasásával. Ha az induló címet a 23296/97 rekeszekben tároljuk el és a hosszt a 23298/99 memóriahelyeken, úgy az előző rutinhoz hasonlóan:

```
ADDR+00   42,0,91       LD   HL,(23296) ;HL-be beolvassuk a kez-
                                        ;dőcímet.
ADDR+03   237,91,2,91   LD   DE,(23298) ;DE-be betöltjük a hossz
                                        ;értékét.
```

Megjegyezzük, hogy a hossz megadásakor a rendezendő hossznál eggyel kevesebbet kell megadni, ugyanis 1000 elemnél az első ciklusban 999 adatpár vizsgálata történik.
Most kiszámítjuk a végcím értékét, ez még szükséges lesz a későbbiekben.

```
ADDR+07   25            ADD  HL,DE      ;HL-ben alakul ki a vég-
                                        ;cím.
ADDR+08   213           PUSH DE         ;A hossz-1 (DE) és végcím
ADDR+09   229           PUSH HL         ;(HL) értékét kimentjük a
                                        ;verembe.
```

A csere mechanizmushoz az IX regiszterpárt fogjuk felhasználni, mivel így egyszerűbb a művelet végrehajtása.

```
ADDR+10   221,42,0,91   LD   IX,(23296) ;IX-be betöltjük az indu-
                                        ;ló cím értékét.
ADDR+14   221,35        INC  IX         ;Ezt az értéket növeljük.
ADDR+16   221,126,0     LD   A,(IX+0)   ;Két egymás melletti cím
ADDR+19   221,190,255   CP   (IX-1)     ;tartalmát összehasonlít-
                                        ;juk. Ha az alacsonyabb
                                        ;cím (IX-1) tartalma na-
                                        ;gyobb, mint az aktuális
                                        ;(IX+0) cím tartalma, ak-
                                        ;kor van átviteli jelző-
                                        ;bit és nem történik el-
```

```
ADDR+22    48,12          JR   NC,ADDR+36 ;ugrás, máskülönben ugrás
                                          ;az ADDR+36. címre.
ADDR+24    221,126,0      LD   A,(IX+0)   ;A-ba betöltjük a maga-
ADDR+27    221,70,255     LD   B,(IX-1)   ;sabb cím tartalmát, B-be
ADDR+30    221,112,0      LD   (IX+0),B   ;pedig az alacsonyabbét,
ADDR+33    221,119,255    LD   (IX-1),A   ;és kicseréljük őket.
ADDR+36    221,229        PUSH IX         ;IX tartalmát áttöltjük a
ADDR+38    209            POP  DE         ;DE regiszterpárba.
ADDR+39    225            POP  HL         ;HL-t a végcímre állít-
ADDR+40    229            PUSH HL         ;juk és visszatöltjük.
ADDR+41    175            XOR  A          ;Az átviteli jelzőbitet
ADDR+42    237,82         SBC  HL,DE      ;töröljük és megnézzük,
                                          ;hogy az IX regiszterpár,
                                          ;amelynek értékét most a
                                          ;DE tartalmazza, elérte-e
                                          ;a végcímet ?
                                          ;Ha igen, akkor van zérus
                                          ;jelzőbit és nem történik
                                          ;elugrás, máskülönben ug-
ADDR+44    32,223         JR   NZ,ADDR+14 ;rás vissza az ADDR+14.
                                          ;címre a következő adat-
                                          ;pár vizsgálatához.
```

Ha megtörtént a sor vizsgálata, a ciklusok értékét csökkentjük, és megvizsgáljuk, hogy DE elérte-e a zérust ? Ha nem, úgy a folyamat kezdődik újra, de eggyel kevesebb pár vizsgálatával, máskülönben a rutinnak vége.

```
ADDR+46    225            POP  HL         ;Visszatöltjük az adato-
ADDR+47    209            POP  DE         ;kat.
ADDR+48    27             DEC  DE         ;A ciklusértékeket csök-
ADDR+49    43             DEC  HL         ;kentjük.
ADDR+50    122            LD   A,D        ;Ha DE zérus, akkor
ADDR+51    179            OR   E          ;nincs ugrás és a rutin-
ADDR+52    32,210         JR   NZ,ADDR+08 ;nak vége, máskülönben
                                          ;ugrás történik az ADDR+
                                          ;08. címre.
ADDR+54    201            RET             ;Vége.
```

Ha a memóriában van még a 31000-31999-ig generált véletlenszám sorozat, akkor írjuk be a kezdőcímet (31000) a 23296/97 címekre.

```
POKE 23296,24: POKE 23297,121 (ENTER)
```

A hossz értéket (1000) pedig adjuk meg a 23298/99 címeken:

```
POKE 23298,232: POKE 23299,3 (ENTER)
```

A rutin végrehajtási ideje nem éri el az 1 percet, a vizsgált véletlenszám minták alapján általában 30 és 32 másodperc közötti időeredmények születtek, ha az ezer elem értékét kinullázzuk,

vagyis csak vizsgálat van, de csere nincs, akkor ez az érték 24 másodpercre szorítható le.

Ez az eredmény számottevően eltérő a BASIC 4 és fél órás végrehajtási idejétől. Ehhez úgy érezzük nincs mit hozzáfűzni !

2.4 Billentyűs hangszer

Bizonyára sokakban felmerült már annak gondolata, hogy a ZX Spectrum számítógépet egyszerű billentyűs hangszerként alkalmazza.

A gép képességeit nincs értelme összehasonlítani egy szintetizátor által nyújtott szolgáltatásokkal.

Kiindulásképpen azt a célt tűzzük ki, hogy egy bizonyos hang csak egy adott billentyű megnyomásakor szólaljon meg.
Mintapéldánkban az alap hangskála 8 egész hangját (C-D-E-F-G-A-H-C) ültetjük rá 8 billentyűre, pl. az A-tól a K-ig.

BASIC-ben a BEEP utasítás segítségével van lehetőségünk a hangképzésre. Célszerű igen rövid tartamot megadni, pl. 0.01 másodpercet hangonként, hogy ne legyen annyira szaggatott a hang megszólalása, de még így is elég "darabos" a hangunk, s ez mind a BASIC sebességének köszönhető. Igaz ha ránézünk a BASIC programra, azt is mondhatnánk, miért nem küldjük vissza soronként a vezérlést a 10. sorra, akkor nem kellene még akkor is megvizsgálni a többi billentyű állapotát, ha egy billentyűt éppen megnyomtunk. Ez igaz, de ebben az esetben gondoljuk csak el milyen különbség lesz "darabosságban" az "A" és pl. a "K" billentyű által keltett hang között. Mindezek után az következik, hogy ezt a mintapéldát nincs értelme BASIC-ben alkalmazni, ennek ellenére most közöljük a BASIC listát:

```
 10 LET a$=INKEY$
 20 IF a$="a" THEN BEEP 0.01,0
 30 IF a$="s" THEN BEEP 0.01,2
 40 IF a$="d" THEN BEEP 0.01,4
 50 IF a$="f" THEN BEEP 0.01,5
 60 IF a$="g" THEN BEEP 0.01,7
 70 IF a$="h" THEN BEEP 0.01,9
 80 IF a$="j" THEN BEEP 0.01,11
 90 IF a$="k" THEN BEEP 0.01,12
100 GO TO 10
```

Ahhoz, hogy aránylag folyamatos hang megszólalását tegyük lehetővé egy billentyű megnyomásakor, programunkat gépi kódban kell megírnunk.
Először tervezzük meg a program vázlatát, hogy elsősorban milyen részegységekből kell összeállítani.
Hangonként le kell olvastatnunk a billentyűzetet, adott billentyű megnyomása esetén meg kell oldanunk a hangkeltést, s gépi kódú program lévén valahol a végén szintén billentyűzet leolvasással (pl. SPACE billentyűre) biztosítani kell a rutinból való kilépést is, mert máskülönben a rutinból való kilépés csak a hálózati csatlakozó kihúzásával, vagy egy RESET segítségével lehetséges.

Nézzük meg a program egy elméleti építőelemét:

```
LD   A,n          - n az adott félsornak megfelelő érték.
IN   A,(254)      - beolvassuk az adatbyte-ot a billen-
                    tyűzetről.
AND  31           - töröljük a felső három bitet.
CP   m            - m a megnyomott billentyűnek megfele-
                    lően módosul. Ha m és a beolvasott
                    byte megegyezik, akkor lesz 1 a zérus
                    jelzőbitben.
JR   NZ,tovább    - Ha nincs 1 a zérus jelzőbitben, akkor
                    átugorjuk a hangkeltő utasításokat.
LD   DE,tartam    - DE-be töltjük a hang tartamát.
LD   HL,magasság- HL-be töltjük a hangmagasságnak meg-
                    felelő értéket.
CALL 949          - Meghívjuk a ROM 949. (HEX-03B5) címén
                    levő BEEPER rutint, amely DE és HL
                    előzetes beállítását követően kiadja
                    a megfelelő hangot.
```

Ha összeszámoljuk, hogy az egyes utasítások hány byte-osak, azt az eredményt kapjuk, hogy a program egy itt bemutatott építőeleme 19 byte-ot foglal le a memóriából. Ez nyolc hangra 152 byte és még hozzá kell adni a programból való kilépés lehetőségét is.

Itt kell néhány szóban beszélnünk az abszolút és relatív címzés gyakorlati alkalmazásáról. Az eddigi példákban is láttuk, hogy ha a zérus vagy az átviteli jelzőbit állapotától függő kisebb ugrást kellett végrehajtani, célszerű volt a relatív ugrások (JR = JUMP RELATIVE) alkalmazása. Ennek több előnye van, egyrészt 1 byte megtakarítást érhetünk el az abszolút címzésekhez képest, másrészt programunk minden gond nélkül bárhova betölthető és futtatható.

Azt is szem előtt kell tartani, hogy a programozó általában hosszabb gépi kódú programját adott memóriahelyre szerkeszti össze, vagyis pl. ez utóbbi esetben nem célja a relatív ugrások alkalmazása a teljes rutinban.

Ebben a rutinban a rutin végéről vissza kell térnünk a rutin elejére, hogy a billentyűzetet levizsgáló rutinok újra működhessenek. Az első fejezetben már láttuk, hogy relatív ugrás előre csak 127, míg visszafele csak 128 byte távolságra történhet, ezért ha most e miatt az egy eset miatt (program végéről visszatérés a program elejére) nem akarunk abszolút címzést használni, célszerű a rutin közepén egy un. "átmeneti ugró állomás"-t elhelyezni, amelynek semmi más célja nem lesz, mint hogy a program végéről ide ugrunk, majd innen a program elejére. Így már megoldható a relatív ugrás. Igaz néhány byte-tal hosszabb lett a rutinunk, de bárhova betölthetjük a memóriába.

Ezek után nézzük a konkrét megvalósítást:

```
ADDR+00   62,253      LD   A,253       ;Az A-G-ig terjedő fél-
ADDR+02   219,254     IN   A,(254)     ;sort olvassuk le.
ADDR+04   230,31      AND  31          ;A felső három bitet tö-
                                       ;röljük.
```

```
ADDR+06   254,30        CP   30           ;Az "A" lett megnyomva?
ADDR+08   32,9          JR   NZ,ADDR+19   ;Ha nem, ugrás az ADDR+19
                                          ;címre.
ADDR+10   17,3,0        LD   DE,3         ;A tartam megközelítőleg
                                          ;egy század másodperc.
ADDR+13   33,106,6      LD   HL,1642      ;A közép C hangnak megfe-
                                          ;lelő hangmagasság.
ADDR+16   205,181,3     CALL 949          ;Meghívjuk a BEEPER ROM
                                          ;rutint.
ADDR+19   62,253        LD   A,253        ;Az A-G-ig terjedő fél-
ADDR+21   219,254       IN   A,(254)      ;sort olvassuk le.
ADDR+23   230,31        AND  31           ;A felső három bitet tö-
                                          ;röljük.
ADDR+25   254,29        CP   29           ;Az "S" lett megnyomva?
ADDR+27   32,9          JR   NZ,ADDR+38   ;Ha nem, ugrás az ADDR+38
                                          ;címre.
ADDR+29   17,3,0        LD   DE,3         ;A tartam megközelítőleg
                                          ;egy század másodperc.
ADDR+32   33,179,5      LD   HL,1459      ;A közép D hangnak megfe-
                                          ;lelő hangmagasság.
ADDR+35   205,181,3     CALL 949          ;Meghívjuk a BEEPER ROM
                                          ;rutint.
ADDR+38   62,253        LD   A,253        ;Az A-G-ig terjedő fél-
ADDR+40   219,254       IN   A,(254)      ;sort olvassuk le.
ADDR+42   230,31        AND  31           ;A felső három bitet tö-
                                          ;röljük.
ADDR+44   254,27        CP   27           ;A "D" lett megnyomva?
ADDR+46   32,9          JR   NZ,ADDR+57   ;Ha nem, ugrás az ADDR+57
                                          ;címre.
ADDR+48   17,3,0        LD   DE,3         ;A tartam megközelítőleg
                                          ;egy század másodperc.
ADDR+51   33,17,5       LD   HL,1297      ;A közép E hangnak megfe-
                                          ;lelő hangmagasság.
ADDR+54   205,181,3     CALL 949          ;Meghívjuk a BEEPER ROM
                                          ;rutint.
ADDR+57   62,253        LD   A,253        ;Az A-G-ig terjedő fél-
ADDR+59   219,254       IN   A,(254)      ;sort olvassuk le.
ADDR+61   230,31        AND  31           ;A felső három bitet tö-
                                          ;röljük.
ADDR+63   254,23        CP   23           ;Az "F" lett megnyomva?
ADDR+65   32,9          JR   NZ,ADDR+76   ;Ha nem, ugrás az ADDR+76
                                          ;címre.
ADDR+67   17,3,0        LD   DE,3         ;A tartam megközelítőleg
                                          ;egy század másodperc.
ADDR+70   33,198,4      LD   HL,1222      ;A közép F hangnak megfe-
                                          ;lelő hangmagasság.
ADDR+73   205,181,3     CALL 949          ;Meghívjuk a BEEPER ROM
                                          ;rutint.
ADDR+76   62,253        LD   A,253        ;Az A-G-ig terjedő fél-
ADDR+78   219,254       IN   A,(254)      ;sort olvassuk le.
ADDR+80   230,31        AND  31           ;A felső három bitet tö-
                                          ;röljük.
ADDR+82   254,15        CP   15           ;A "G" lett megnyomva?
```

```
ADDR+84   32,13        JR   NZ,ADDR+99 ;Ha nem, ugrás az ADDR+99
                                       ;címre.
ADDR+86   17,3,0       LD   DE,3       ;A tartam megközelítőleg
                                       ;egy század másodperc.
ADDR+89   33,61,4      LD   HL,1085    ;A közép G hangnak megfe-
                                       ;lelő hangmagasság.
ADDR+92   205,181,3    CALL 949        ;Meghívjuk a BEEPER ROM
                                       ;rutint.
ADDR+95   24,2         JR   ADDR+99    ;Ugrás tovább.
-------------------------------------------------------------
ADDR+97   24,157       JR   ADDR+00    ;Ugrás vissza az elejére.
-------------------------------------------------------------
ADDR+99   62,191       LD   A,191      ;A H-ENTER-ig terjedő
ADDR+101  219,254      IN   A,(254)    ;félsort olvassuk le.
ADDR+103  230,31       AND  31         ;A felső három bitet tö-
                                       ;röljük.
ADDR+105  254,15       CP   15         ;A "H" lett megnyomva?
ADDR+107  32,9         JR   NZ,ADDR+118;Ha nem,ugrás az ADDR+118
                                       ;címre.
ADDR+109  17,3,0       LD   DE,3       ;A tartam megközelítőleg
                                       ;egy század másodperc.
ADDR+112  33,196,3     LD   HL,964     ;A közép A hangnak megfe-
                                       ;lelő hangmagasság.
ADDR+115  205,181,3    CALL 949        ;Meghívjuk a BEEPER ROM
                                       ;rutint.
ADDR+118  62,191       LD   A,191      ;A H-ENTER-ig terjedő
ADDR+120  219,254      IN   A,(254)    ;félsort olvassuk le.
ADDR+122  230,31       AND  31         ;A felső három bitet tö-
                                       ;röljük.
ADDR+124  254,23       CP   23         ;A "J" lett megnyomva?
ADDR+126  32,9         JR   NZ,ADDR+137;Ha nem,ugrás az ADDR+137
                                       ;címre.
ADDR+128  17,3,0       LD   DE,3       ;A tartam megközelítőleg
                                       ;egy század másodperc.
ADDR+131  33,87,3      LD   HL,855     ;A közép H hangnak megfe-
                                       ;lelő hangmagasság.
ADDR+134  205,181,3    CALL 949        ;Meghívjuk a BEEPER ROM
                                       ;rutint.
ADDR+137  62,191       LD   A,191      ;A H-ENTER-ig terjedő
ADDR+139  219,254      IN   A,(254)    ;félsort olvassuk le.
ADDR+141  230,31       AND  31         ;A felső három bitet tö-
                                       ;röljük.
ADDR+143  254,27       CP   27         ;A "K" lett megnyomva?
ADDR+145  32,9         JR   NZ,ADDR+156;Ha nem,ugrás az ADDR+156
                                       ;címre.
ADDR+147  17,3,0       LD   DE,3       ;A tartam megközelítőleg
                                       ;egy század másodperc.
ADDR+150  33,37,3      LD   HL,805     ;A közép C-nél egy oktáv-
                                       ;val magasabb hangmagas-
                                       ;ság.
ADDR+153  205,181,3    CALL 949        ;Meghívjuk a BEEPER ROM
                                       ;rutint.
```

A végén egy lehetőséget kell biztosítanunk a programozó számára a rutinból való kilépésre:

```
ADDR+156  62,127      LD   A,127       ;A B-SPACE-ig terjedő
ADDR+158  219,254     IN   A,(254)     ;félsort olvassuk le.
ADDR+160  230,31      AND  31          ;A felső három bitet tö-
                                       ;röljük.
ADDR+162  254,30      CP   30          ;A SPACE lett megnyomva?
ADDR+164  32,187      JR   NZ,ADDR+97  ;Ha nem, ugrás az ADDR+97
                                       ;címre, onnan pedig visz-
                                       ;sza a rutin elejére.
ADDR+166  201         RET              ;SPACE esetén kilépünk a
                                       ;rutinból.
```

Ha a DE regiszterpár értékének blokkonként nem hármat, hanem 26-ot adunk meg, úgy a tartam értékét hangonként kb. 1 tized másodpercre növeltük, míg 261 esetén megközelítőleg 1 másodpercet kapunk. Próbáljuk ki a rutin futtatását különböző tartam értékekkel.
A hangmagasság értéke a: (437500/f)-30.125 képlet eredményének az egész része, ahol f az adott hang frekvenciája.
A frekvenciaértékek egész és félhangokra egy teljes oktávra a következő táblázatban találhatók:

C	-	261.63	F#	-	369.99
C#	-	277.18	G	-	392.00
D	-	293.66	G#	-	415.30
D#	-	311.13	A	-	440.00
E	-	329.63	A#	-	466.16
F	-	349.23	H	-	493.88
			>C	-	523.25

Próbáljuk meg mi történik, ha pl. az A billentyű nyomva tartása mellett megnyomjuk H-t, J-t, vagy a K billentyűt. Ne lepődjünk meg, az utóbbi három billentyű másik félsorhoz tartozik, így funkciói az A billentyű megnyomásával párhuzamosan végrehajtódnak. Gépi kódban több billentyű együttes megnyomását is külön funkciónak értelmezhetjük, így az említett lehetőségekkel több szólamú, un. "ál-polifónikus" hangszert készíthetünk.
A félhangokkal, oktávváltással való kibővítést, a többszólamú átalakítást, valamint a hangok finomítását gyakorlásképpen az olvasóra bízzuk.

Megjegyzés: Az egyes hangok frekvenciái oktávonként feleződnek, ill. duplázódnak.

2.5 Készítsünk rajzoló-programot

A legegyszerűbb rajzoló-program, amely megvizsgálja a billentyűzetet és a billentyű megnyomás függvényében négy fő ill. négy átlós irányban vonalat húz. Ezt kibővíthetjük az OVER 0/1 kapcsolóval, azaz pl. egy meglevő vonalat törölhetünk is a rajzunkról. Az egyszerűség kedvéért most nem érdemes mélyebb szintű mintapéldát szerkeszteni, másrészt egy igen jó szolgáltatásokkal rendelkező rajzolóprogramot pl.THE ARTIST (leírása megtalálható a "SINCLAIR Spectrum játék és program" c. könyvben - LSI ATSz. 1986.), nem érdemes saját magunknak elkészíteni, ha az rendelkezésre áll.
Gépeljük be előbb a BASIC változatot:

```
 10 LET x=125: LET y=85
 20 PLOT x,y
 30 LET a$=INKEY$
 40 IF a$="w" THEN LET y=y+(y<175)
 50 IF a$="a" THEN LET x=x-(x>0)
 60 IF a$="x" THEN LET y=y-(y>0)
 70 IF a$="d" THEN LET x=x+(x<255)
 80 IF a$="q" THEN LET x=x-(x>0)*(y<175):
    LET y=y+(y<175)*(x>0)
 90 IF a$="z" THEN LET x=x-(x>0)*(y>0):
    LET y=y-(y>0)*(x>0)
100 IF a$="c" THEN LET x=x+(x<255)*(y>0):
    LET y=y-(y>0)*(x<255)
110 IF a$="e" THEN LET x=x+(x<255)*(y<175):
    LET y=y+(y<175)*(x<255)
120 IF a$="1" THEN OVER 1
130 IF a$="2" THEN OVER 0
140 PLOT x,y
150 GO TO 30
```

Mint ahogy a BASIC programban látjuk, annak eldöntésére, hogy a vonal elérte-e a képernyő szélét, egy egyszerű logikai formulát használunk. A döntésekből származó időveszteség némileg csökkenthető, ha a billentyűmegnyomást is beolvasztjuk a belső logikai formulába, pl. a 40. sor így is felírható:

```
40 LET y=y+(y<175)*(a$="w")
```

Ennek megfelelően természetesen a többi sor is átírható. A BASIC program által rajzoló-programunk sebessége nem igazán állítható be pontosan arra az értékre, ami rajzoláshoz a legmegfelelőbb.
A BASIC-hez hasonlóan gépi kódban is az S billentyű körüli 8 billentyű segítségével oldjuk meg a vezérlést, vagyis:

W = fel	Q = balra/fel
A = balra	Z = balra/le
D = jobbra	E = jobbra/fel
X = le	C = jobbra/le

Ezen kívül az 1 illetve 2 billentyűk az OVER be/kikapcsolását oldják meg. Gépi kódban szükség van a rutinból való kilépési lehetőségre is, ezt oldjuk meg változatlanul a SPACE billentyű segítségével.

Első feladatunk a kiindulási pozíció beállítása.

```
ADDR+00   30,125       LD   E,125      ;E-be töltjük az x,   D-
ADDR+02   22,85        LD   D,85       ;be az y koordinátát és
ADDR+04   75           LD   C,E        ;ugyanezt BC-be is át-
ADDR+05   66           LD   B,D        ;töltjük.
ADDR+06   213          PUSH DE         ;DE-t eltesszük a verem-
                                       ;be, mert a pont megjele-
                                       ;nítő szubrutin felülírná
                                       ;értékét.
ADDR+07   205,229,34   CALL 8933       ;Meghívjuk a ROM 8933.
                                       ;(HEX-22E5) címén talál-
                                       ;ható szubrutint, amely a
                                       ;BC-ben megadott koordi-
                                       ;náták helyén (C-ben az x
                                       ;és B-ben az y) pontot
                                       ;helyez a képernyőre.
ADDR+10   209          POP DE          ;DE értékét, amely a min-
                                       ;denkori koordináta pozí-
                                       ;ciót fogja tartalmazni
                                       ;rajzoláskor, visszatölt-
                                       ;jük a veremből.
```

Megkezdhetjük a billentyűzet vizsgáló utasítások beépítését, itt azonban annyit megjegyeznénk, hogy aki egy ilyen programot papíron elkezd megtervezni, ebben a fázisban még nem tudja, hogy az adott billentyűnek megfelelő pont-léptető szubrutin pontosan hova fog esni a memóriában, így célszerű a relatív vagy abszolút címzések paramétereit átlépni, s ezeket akkor pótoljuk, amikor a rutin teljes mértékben felépült.
Assembler programok megengedik a cimkék használatát, ekkor az előbb említett problémák nem jelentkeznek.
(A cimke egy karakterlánc, amely a rutin bármely címéhez hozzárendelhető, így elegendő a cimkével történő hivatkozás pl. egy ugró utasítás esetén. Az assembler a fordítás során ezeket a cimkéket figyelembe veszi, és a gépi kód annak megfelelően alakul ki.)

Az elvi vázlat megvizsgálása után ki tudtuk számolni, hogy az egyes szubrutinok hol fognak elhelyezkedni, úgyhogy a következőkben az ugrási címhivatkozások már ezekre a címekre történnek.

```
ADDR+11   62,247       LD   A,247      ;Az 1-5-ig terjedő fél-
ADDR+13   219,254      IN   A,(254)    ;sort olvassuk le.
ADDR+15   230,31       AND  31         ;Töröljük a felső három
                                       ;bitet.
ADDR+17   254,30       CP   30         ;Az "1" lett megnyomva ?
ADDR+19   40,120       JR   Z,ADDR+141 ;Ha igen, ugrás történik
                                       ;az ADDR+141.címre.
```

```
ADDR+21   62,247      LD   A,247        ;Az 1-5-ig terjedő fél-
ADDR+23   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+25   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+27   254,29      CP   29           ;A "2" lett megnyomva ?
ADDR+29   40,118      JR   Z,ADDR+149   ;Ha igen, ugrás történik
                                        ;az ADDR+149.címre.
ADDR+31   62,127      LD   A,127        ;A B-SPACE-ig terjedő
ADDR+33   219,254     IN   A,(254)      ;félsort olvassuk le.
ADDR+35   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+37   254,30      CP   30           ;A SPACE lett megnyomva ?
ADDR+39   200         RET  Z            ;Ha igen, kilépünk a
                                        ;programból.
ADDR+40   62,251      LD   A,251        ;A Q-T-ig terjedő fél-
ADDR+42   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+44   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+46   254,29      CP   29           ;A "W" lett megnyomva ?
ADDR+48   40,107      JR   Z,ADDR+157   ;Ha igen, ugrás történik
                                        ;az ADDR+157.címre.
ADDR+50   62,251      LD   A,251        ;A Q-T-ig terjedő fél-
ADDR+52   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+54   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+56   254,30      CP   30           ;A "Q" lett megnyomva ?
ADDR+58   40,106      JR   Z,ADDR+166   ;Ha igen, ugrás történik
                                        ;az ADDR+166.címre.
ADDR+60   62,251      LD   A,251        ;A Q-T-ig terjedő fél-
ADDR+62   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+64   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+66   254,27      CP   27           ;Az "E" lett megnyomva ?
ADDR+68   40,115      JR   Z,ADDR+185   ;Ha igen, ugrás történik
                                        ;az ADDR+185.címre.
ADDR+70   62,253      LD   A,253        ;Az A-G-ig terjedő fél-
ADDR+72   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+74   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+76   254,30      CP   30           ;Az "A" lett megnyomva ?
ADDR+78   40,124      JR   Z,ADDR+204   ;Ha igen, ugrás történik
                                        ;az ADDR+204.címre.
ADDR+80   62,253      LD   A,253        ;Az A-G-ig terjedő fél-
ADDR+82   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+84   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
ADDR+86   254,27      CP   27           ;A "D" lett megnyomva ?
ADDR+88   40,123      JR   Z,ADDR+213   ;Ha igen, ugrás történik
                                        ;az ADDR+213.címre.
ADDR+90   62,254      LD   A,254        ;A CS-V-ig terjedő fél-
ADDR+92   219,254     IN   A,(254)      ;sort olvassuk le.
ADDR+94   230,31      AND  31           ;Töröljük a felső három
                                        ;bitet.
```

```
ADDR+96   254,27        CP   27           ;Az "X" lett megnyomva ?
ADDR+98   40,122        JR   Z,ADDR+222   ;Ha igen, ugrás történik
                                          ;az ADDR+222.címre.
ADDR+100  62,254        LD   A,254        ;A CS-V-ig terjedő fél-
ADDR+102  219,254       IN   A,(254)      ;sort olvassuk le.
ADDR+104  230,31        AND  31           ;Töröljük a felső három
                                          ;bitet.
ADDR+106  254,29        CP   29           ;A "Z" lett megnyomva ?
ADDR+108  40,121        JR   Z,ADDR+231   ;Ha igen, ugrás történik
                                          ;az ADDR+231.címre.
ADDR+110  62,254        LD   A,254        ;A CS-V-ig terjedő fél-
ADDR+112  219,254       IN   A,(254)      ;sort olvassuk le.
ADDR+114  230,31        AND  31           ;Töröljük a felső három
                                          ;bitet.
ADDR+116  254,23        CP   23           ;A "C" lett megnyomva ?
ADDR+118  40,19         JR   Z,ADDR+139   ;Ha igen, ugrás történik
                                          ;az ADDR+139.címre, és
                                          ;innen tovább az ADDR+250
                                          ;címre. Az ADDR+139. cí-
                                          ;men is egy átmeneti ál-
                                          ;lomás található, mivel
                                          ;relatív ugrással közvet-
                                          ;lenül nem tudjuk megcé-
                                          ;lozni az ADDR+250. re-
                                          ;keszt.
```

Most járunk kb. a rutin felénél, tekintve hogy a szubrutinok végrehajtása után a vezérlést vissza akarjuk adni a billentyűzet leolvasó blokkoknak, másrészt ugyanezt kell tennünk akkor is, ha nem nyomtunk meg billentyűt, így most feltétlenül szükséges egy ugró utasítás, nem a legelső címre, hiszen a középre pozícionálásra nincs szükségünk, hanem az ADDR+04. címre.

```
ADDR+120  24,138        JR   ADDR+04      ;Ugrás vissza az ADDR+04.
                                          ;címre.
```

Az egyes szubrutinok végrehajtása után is ehhez a címhez kell eljutnunk, de gondoljunk csak arra, van-e értelme időbefektetésünknek, ha olyan gyors lesz a vonal húzásának sebessége, hogy nem tudjuk képpont-pontosan pozícionálni. Ebből adódóan be kell kalkulálni egy időtagot, de lehetőleg olyat, amellyel szabályozni is tudjuk a végrehajtás sebességét.
A legcélszerűbb egymásba ágyazott két üres ciklus elhelyezése, de tekintve hogy a verem-műveletek utasítás végrehajtási ideje elég magas, elhelyeztünk a belső cikluson belül 2 ilyen, önmagában értelmetlen utasításpárt. A külső ciklusban a B regiszter értékét 13-ra, a belsőben pedig zérusra választottuk, vagyis a processzor üresen 13*256=3328 ciklust hajt végre, hogy kb. elérjük a BASIC változat sebességét. Ha ezt az értéket változtatjuk, pl. a 13 helyett 255-öt írunk, sokkal lassabb lesz a vonal sebessége, míg ha B regiszter mindkét értékének 1-et adunk, igen gyors lesz a végrehajtási sebesség. Az egyénileg megfelelő ill. a rajz jellegéhez igazodó sebesség beállítását mindenki maga elvégezheti.

```
ADDR+122  6,13          LD   B,13          ;Beállítjuk a külső cik-
                                           ;lus értékét,
ADDR+124  197           PUSH BC            ;és eltesszük a verembe.
ADDR+125  6,0           LD   B,0           ;Beállítjuk a belső cik-
ADDR+127  0             NOP                ;lus értékét.
ADDR+128  229           PUSH HL            ;Kétszer-két lényegtelen
ADDR+129  225           POP  HL            ;verem-műveletet hajtunk
ADDR+130  229           PUSH HL            ;végre az időtag értéké-
ADDR+131  225           POP  HL            ;nek befolyásolására.
ADDR+132  16,250        DJNZ ADDR+128      ;Ciklus 256-szor.
ADDR+134  193           POP  BC            ;Ha lefutott a 256 ciklus
                                           ;BC-t kivesszük a verem-
                                           ;ből.
ADDR+135  16,243        DJNZ ADDR+124      ;Ciklus 13-szor.
ADDR+137  24,237        JR   ADDR+120      ;Ugrás vissza az ADDR+120
                                           ;címre, és onnan vissza
                                           ;újra a billentyűzet leol-
                                           ;vasásához.
```

Most már látjuk, hogy az egyes szubrutinoktól az ADDR+122. címre kell eljutnunk, és az időtagon keresztül adódik vissza a vezérlés a billentyűzetet leolvasó utasításokhoz.
Most helyezzük el azt a közbenső relatív ugrási állomást, amely a C billentyű megnyomásánál vált szükségszerűvé.

```
ADDR+139  24,109        JR   ADDR+250      ;Ugrás tovább az ADDR+250
                                           ;címre.
```

Ezután elkezdhetjük a 10 szubrutin (8 irány+2 OVER állapot) beépítését. Mivel minden szubrutintól az ADDR+122.címre kell visszatérni, és az utolsó szubrutinok megint kicsúsznak a relatív ugrás maximális távolságából, ezért ha nem akarunk ismét abszolut címhivatkozásokat használni, a legcélszerűbb a lépcsőzetes visszaugrálás. Ez azt jelenti, hogy minden szubrutin végrehajtása után az előtte levő szubrutin utolsó visszaugró utasítására történik a vezérlés átadása, innen ismét az előtte levőhöz, és a sorban átadással végül is bárhonnan eljutunk a keresett címhez. Itt megjegyezzük, hogy függetlenül az ugró utasítások gyors végrehajtási idejétől pl. egy programban nem biztos, hogy előnyös az ilyen forma alkalmazása, ha egyetlen abszolút ugrással közvetlenül visszaugorhatunk.

```
ADDR+141  62,21         LD   A,21          ;Az OVER kódja.
ADDR+143  215           RST  16
ADDR+144  62,1          LD   A,1           ;Bekapcsoljuk az OVER 1
ADDR+146  215           RST  16            ;állapotot.
ADDR+147  24,229        JR   ADDR+122      ;Ugrás vissza az ADDR+122
                                           ;címre.
-------------------------------------------------------------------
ADDR+149  62,21         LD   A,21          ;Az OVER kódja.
ADDR+151  215           RST  16
ADDR+152  62,0          LD   A,0           ;Kikapcsoljuk az OVER 1
ADDR+154  215           RST  16            ;állapotot.
```

```
ADDR+155  24,246        JR   ADDR+147    ;Ugrás vissza az ADDR+147
                                         ;címre.
-------------------------------------------------------------------
ADDR+157  20            INC  D           ;Az y koordinátát növel-
ADDR+158  122           LD   A,D         ;jük eggyel, majd A-ba
ADDR+159  254,176       CP   176         ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén.
ADDR+161  32,1          JR   NZ,ADDR+164 ;Ha nem, ugrás tovább.
ADDR+163  21            DEC  D           ;Máskülönben visszaállít-
                                         ;juk az eredeti állapotot
ADDR+164  24,245        JR   ADDR+155    ;és ugrás vissza az ADDR+
                                         ;155. címre.
-------------------------------------------------------------------
ADDR+166  20            INC  D           ;Az y koordinátát növel-
ADDR+167  122           LD   A,D         ;jük eggyel, majd A-ba
ADDR+168  254,176       CP   176         ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén.
ADDR+170  32,3          JR   NZ,ADDR+175 ;Ha nem, ugrás tovább.
ADDR+172  21            DEC  D           ;Máskülönben visszaállít-
                                         ;juk az eredeti állapotot
ADDR+173  24,245        JR   ADDR+164    ;és ugrás vissza az ADDR+
                                         ;164. címre.
ADDR+175  29            DEC  E           ;Az x koordinátát csök-
ADDR+176  123           LD   A,E         ;kentjük 1-gyel majd A-ba
ADDR+177  254,255       CP   255         ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén(0-1=255)
ADDR+179  32,2          JR   NZ,ADDR+183 ;Ha nem, ugrás tovább.
ADDR+181  28            INC  E           ;Máskülönben visszaállít-
ADDR+182  21            DEC  D           ;juk az eredeti állapotot
ADDR+183  24,235        JR   ADDR+164    ;és ugrás vissza az ADDR+
                                         ;164. címre.
-------------------------------------------------------------------
ADDR+185  20            INC  D           ;Az y koordinátát növel-
ADDR+186  122           LD   A,D         ;jük eggyel, majd A-ba
ADDR+187  254,176       CP   176         ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén.
ADDR+189  32,3          JR   NZ,ADDR+194 ;Ha nem, ugrás tovább.
ADDR+191  21            DEC  D           ;Máskülönben visszaállít-
                                         ;juk az eredeti állapotot
ADDR+192  24,245        JR   ADDR+183    ;és ugrás vissza az ADDR+
                                         ;183. címre.
ADDR+194  28            INC  E           ;Az x koordinátát növel-
ADDR+195  123           LD   A,E         ;jük eggyel majd A-ba
ADDR+196  254,0         CP   0           ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén(255+1=0)
ADDR+198  32,2          JR   NZ,ADDR+202 ;Ha nem, ugrás tovább.
ADDR+200  29            DEC  E           ;Máskülönben visszaállít-
ADDR+201  21            DEC  D           ;juk az eredeti állapotot
```

```
ADDR+202  24,235      JR   ADDR+183    ;és ugrás vissza az ADDR+
                                       ;183. címre.
-------------------------------------------------------------
ADDR+204  29          DEC  E           ;Az x koordinátát csök-
ADDR+205  123         LD   A,E         ;kentjük 1-gyel és  A-ba
ADDR+206  254,255     CP   255         ;áttöltve megnézzük, túl-
                                       ;menne-e a képernyő szé-
                                       ;lén.
ADDR+208  32,1        JR   NZ,ADDR+211;Ha nem, ugrás tovább.
ADDR+210  28          INC  E           ;Máskülönben visszaállít-
                                       ;juk az eredeti állapotot
ADDR+211  24,245      JR   ADDR+202    ;és ugrás vissza az ADDR+
                                       ;202. címre.
-------------------------------------------------------------
ADDR+213  28          INC  E           ;Az x koordinátát növel-
ADDR+214  123         LD   A,E         ;jük eggyel, majd A-ba
ADDR+215  254,0       CP   0           ;áttöltve megnézzük, túl-
                                       ;menne-e a képernyő szé-
                                       ;lén.
ADDR+217  32,1        JR   NZ,ADDR+220;Ha nem, ugrás tovább.
ADDR+219  29          DEC  E           ;Máskülönben visszaállít-
                                       ;juk az eredeti állapotot
ADDR+220  24,245      JR   ADDR+211    ;és ugrás vissza az ADDR+
                                       ;211. címre.
-------------------------------------------------------------
ADDR+222  21          DEC  D           ;Az y koordinátát csök-
ADDR+223  122         LD   A,D         ;kentjük 1-gyel és  A-ba
ADDR+224  254,255     CP   255         ;áttöltve megnézzük, túl-
                                       ;menne-e a képernyő szé-
                                       ;lén.
ADDR+226  32,1        JR   NZ,ADDR+229;Ha nem, ugrás tovább.
ADDR+228  20          INC  D           ;Máskülönben visszaállít-
                                       ;juk az eredeti állapotot
ADDR+229  24,245      JR   ADDR+220    ;és ugrás vissza az ADDR+
                                       ;220. címre.
-------------------------------------------------------------
ADDR+231  21          DEC  D           ;Az y koordinátát csök-
ADDR+232  122         LD   A,D         ;kentjük 1-gyel és  A-ba
ADDR+233  254,255     CP   255         ;áttöltve megnézzük, túl-
                                       ;menne-e a képernyő szé-
                                       ;lén.
ADDR+235  32,3        JR   NZ,ADDR+240;Ha nem, ugrás tovább.
ADDR+237  20          INC  D           ;Máskülönben visszaállít-
                                       ;juk az eredeti állapotot
ADDR+238  24,245      JR   ADDR+229    ;és ugrás vissza az ADDR+
                                       ;229. címre.
ADDR+240  29          DEC  E           ;Az x koordinátát csök-
ADDR+241  123         LD   A,E         ;kentjük 1-gyel majd A-ba
ADDR+242  254,255     CP   255         ;áttöltve megnézzük, túl-
                                       ;menne-e a képernyő szé-
                                       ;lén(0-1=255)
ADDR+244  32,2        JR   NZ,ADDR+248;Ha nem, ugrás tovább.
ADDR+246  28          INC  E           ;Máskülönben visszaállít-
```

```
ADDR+247  20           INC  D            ;juk az eredeti állapotot
ADDR+248  24,235       JR   ADDR+229     ;és ugrás vissza az ADDR+
                                         ;229. címre.
-------------------------------------------------------------------
ADDR+250  21           DEC  D            ;Az y koordinátát csök-
ADDR+251  122          LD   A,D          ;kentjük 1-gyel és A-ba
ADDR+252  254,255      CP   255          ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén.
ADDR+254  32,3         JR   NZ,ADDR+259  ;Ha nem, ugrás tovább.
ADDR+256  20           INC  D            ;Máskülönben visszaállít-
                                         ;juk az eredeti állapotot
ADDR+257  24,245       JR   ADDR+248     ;és ugrás vissza az ADDR+
                                         ;248. címre.
ADDR+259  28           INC  E            ;Az x koordinátát növel-
ADDR+260  123          LD   A,E          ;jük eggyel, majd A-ba
ADDR+261  254,0        CP   0            ;áttöltve megnézzük, túl-
                                         ;menne-e a képernyő szé-
                                         ;lén.
ADDR+263  32,2         JR   NZ,ADDR+267  ;Ha nem, ugrás tovább.
ADDR+265  29           DEC  E            ;Máskülönben visszaállít-
ADDR+266  20           INC  D            ;juk az eredeti állapotot
ADDR+267  24,235       JR   ADDR+248     ;és ugrás vissza az ADDR+
                                         ;248. címre.
-------------------------------------------------------------------
```

A rutin jelenlegi állapotában meghíváskor nem törli a képernyőt, ez szükség szerint megoldható a "kereső" rutinnál leírtak szerint
Az itt ismertetett rutinnal csak a képernyő felső 22 sorát tudjuk kihasználni. Azok a rajzoló-programok, amelyek lehetővé teszik a folyamatos manipulációt a teljes képernyőre (pl. még kör rajzolásához is), más, sokkal bonyolultabb algoritmus szerint működnek.

A fejezetben ismertetett gépi kódú programokhoz megpróbáltunk az általánoshoz képest nagyobb mélységű magyarázatot fűzni, hogy ne maradjon nyitott kerdés. Reméljük a bemutatott mintapéldák mindenki számára meggyőzőek és lehetőséget biztosítanak a továbblépéshez.

És egy kis reklám:

3.fejezet

PILLANTSUNK BE A TOOLKIT RUTINOK BELSEJÉBE

Fejezetünk nem kívánja az összes létező TOOLKIT rutin működését és felépítését bemutatni. Csak a lényegesebb rutinok feltárására törekedtünk.

Bizonyára találkozott már minden Spectrum tulajdonos olyan problémával, hogy az adott TOOLKIT program sok helyet lefoglalt a memóriából, s tekintve a saját BASIC vagy gépi kódú program hosszúságát, párhuzamosan nem volt lehetőség a TOOLKIT rutinjainak aktivizálására. Az itt ismertetett gépi kódú rutinok nem váltják meg a világot, de mindenképpen segítenek kiküszöbölni hasonló problémákat.

Külön-külön beolvashatók (betölthetők) a memória tetszőleges szabad területére. Egyes rutinokat saját programjainkba is beolvaszthatunk, a legtöbbjét viszont BASIC programunk szerkesztésekor használhatjuk fel előnyösen.

A rutinok ismertetésén túl az is célunk volt, hogy azok működési elvét is megértessük. Ez persze gyakorlattól függően hol több, hol még több energiát fog követelni az olvasótól.

Itt láthatjuk majd, hogy a rutinok nem csupán byte-okból, hanem egymástól elhatárolható modulokból épülnek fel, és ezek a modulok újabb rutinok építőköveként is használhatók.

3.1 ALTER rutin

Célja: Egy adott ASCII kódot kicserél egy másikra a BASIC listában. Ez igen hasznos lehet ha pl. egy egyszerű aritmetikai változó nevét az egész programban meg akarjuk változtatni.

Az ADDR+01 címen kell megadni a régi, és az ADDR+03 címen az új ASCII kódot.

Nézzünk egy mintapéldát:

```
10 LET a=20
```

Ebben a programban az "a" karaktert szeretnénk kicserélni "z"karakterre. Az "a" karakter kódja 97, a "z" karakteré pedig 122. A megfelelő címeken ezeket az értékeket kell megadnunk.

```
ADDR+00   22,97        LD   D,97        ;D-be betöltjük a régi,
ADDR+02   30,122       LD   E,122       ;E-be pedig az új ASCII
                                        ;kódot.
ADDR+04   42,83,92     LD   HL,(23635)  ;HL-be töltjük a BASIC
ADDR+07   35           INC  HL          ;terület kezdőcímét, majd
ADDR+08   35           INC  HL          ;négy lépésben előállít-
ADDR+09   35           INC  HL          ;juk az első értelmezhető
```

```
ADDR+10   35            INC  HL          ;karakter memóriabeli cí-
ADDR+11   126           LD   A,(HL)      ;mét, s ezt áttöltjük az
                                         ;A regiszterbe.
ADDR+12   14,14         LD   C,14        ;Megnézzük, hogy az ASCII
ADDR+14   185           CP   C           ;kód 14-e ? (Ez a számáb-
                                         ;rázolás kódja)
ADDR+15   32,3          JR   NZ,ADDR+20  ;Ugrás, ha nem.
ADDR+17   35            INC  HL          ;Máskülönben lépés to-
ADDR+18   24,243        JR   ADDR+07     ;vább 5 byte-tal.
ADDR+20   14,13         LD   C,13        ;Megnézzük, hogy az ASCII
ADDR+22   185           CP   C           ;kód ENTER-e ?
ADDR+23   32,12         JR   NZ,ADDR+37  ;Ha nem, ugrás.
ADDR+25   35            INC  HL          ;Máskülönben lépés tovább
                                         ;a következő címre.
ADDR+26   237,75,75,92  LD   BC,(23627)  ;BC-be töltjük a változók
ADDR+30   167           AND  A           ;területének kezdőcímét,
                                         ;töröljük az átviteli
ADDR+31   237,66        SBC  HL,BC       ;jelzőbitet és megnézzük,
ADDR+33   9             ADD  HL,BC       ;hogy van-e még BASIC ?
ADDR+34   208           RET  NC          ;Ha nincs, befejezzük,
ADDR+35   24,226        JR   ADDR+07     ;máskülönben ugrás visz-
                                         ;sza.
ADDR+37   186           CP   D           ;Ha az ASCII kód nem
ADDR+38   32,226        JR   NZ,ADDR+10  ;egyezik a keresettel,
                                         ;ugrás vissza.
ADDR+40   115           LD   (HL),E      ;Az adott címen elhelyez-
                                         ;zük az új ASCII kódot.
ADDR+41   24,223        JR   ADDR+10     ;Ugrás vissza.
```

Ha futtatjuk a rutint, ismételt listázás esetén már a következőt látjuk a képernyőn:

```
10 LET z=20
```

Természetesen az itt közölt mintapélda nem igazán mutatja a rutin valódi hasznát, hiszen egy karakter megváltoztatására BASIC programunkban nem sok értelme lenne egy ilyen rutint alkalmazni, viszont pl. egy 30-40 kbyte hosszú BASIC program esetén már megtérül a rutin bevitelébe fektetett energia.

3.2 ATTR FILL rutin

Célja: A képernyő meghatározott területét (box) kitölti a megadott ATTR kóddal.

A rutinban az ADDR+01 címen kell megadni az ATTR kód értékét, az ADDR+03 címen a box bal felső karakterének oszlop pozícióját (0-31), ill. az ADDR+04 címen ugyanennek a karakternek a sor pozícióját (0-23), végül a box szélességét az ADDR+06 címen, magasságát pedig az ADDR+07 címen kell elhelyezni a memóriában.

A rutin listájában beolvasáskor átmenetileg írjunk ezeknek az értékeknek zérust, mivel a rutin működésének bemutatásához mellékelt BASIC program ezeket megfelelően beállítja.

```
ADDR+00   62,0          LD   A,0        ;A-ban van az ATTR kód.
ADDR+02   1,0,0         LD   BC,0       ;BC-be töltjük a box bal
                                        ;felső karakterének AT
                                        ;pozícióját.
ADDR+05   17,0,0        LD   DE,0       ;DE-be töltjük a box mé-
                                        ;retét.
ADDR+08   33,223,87     LD   HL,22495   ;HL-be töltjük az induló
                                        ;címet.
ADDR+11   213           PUSH DE         ;A box méretét kimentjük
                                        ;a verembe.
ADDR+12   17,32,0       LD   DE,32      ;DE-be töltjük a két egy-
                                        ;más alatti karakter ATTR
                                        ;címe közötti byte kü-
                                        ;lönbséget.
ADDR+15   4             INC  B          ;Növeljük a sor pozíciót,
ADDR+16   25            ADD  HL,DE      ;és HL-ben képezzük a box
                                        ;feletti sor szélső cí-
                                        ;mét.
ADDR+17   16,253        DJNZ ADDR+16    ;Ciklus a sorok számának
                                        ;megfelelően.
ADDR+19   65            LD   B,C        ;Betöltjük B-be az oszlop
ADDR+20   4             INC  B          ;pozíciót, majd HL-ben
ADDR+21   35            INC  HL         ;számitjuk ki a box bal
ADDR+22   16,253        DJNZ ADDR+21    ;felső karakter pozició-
                                        ;jának címét.
ADDR+24   209           POP  DE         ;Visszatöltjük a box mé-
ADDR+25   66            LD   B,D        ;retét, és D-be áttöltjük
                                        ;a magasságot.
ADDR+26   229           PUSH HL         ;Kimentjük a verembe a
ADDR+27   72            LD   C,B        ;box bal felső pozíciójá-
ADDR+28   67            LD   B,E        ;nak címét, majd a magas-
                                        ;ságot C-be, a szélessé-
                                        ;get pedig B-be töltjük.
ADDR+29   119           LD   (HL),A     ;Az adott képernyőcímre
ADDR+30   35            INC  HL         ;betöltjük az ATTR kódot,
                                        ;majd növeljük a címet.
ADDR+31   16,252        DJNZ ADDR+29    ;Ugrás vissza, ha a sor-
                                        ;nak nincs vége.
ADDR+33   225           POP  HL         ;Visszatöltjük a box bal
                                        ;felső karakter pozició
                                        ;címét.
ADDR+34   213           PUSH DE         ;Kimentjük a box méretét.
ADDR+35   17,32,0       LD   DE,32      ;DE-be töltjük két egymás
                                        ;alatti karakter ATTR cí-
                                        ;me közötti különbségét.
ADDR+38   25            ADD  HL,DE      ;Egy sorral lejjebb lé-
ADDR+39   209           POP  DE         ;pünk és visszatöltjük a
                                        ;box méretét.
ADDR+40   65            LD   B,C        ;B-be töltjük a magassá-
                                        ;got.
ADDR+41   16,239        DJNZ ADDR+26    ;Ugrás vissza, ha van még
                                        ;hátra sor,
ADDR+43   201           RET             ;máskülönben vége.
```

A rutint tetszőleges helyre beolvashatjuk a memóriába, most a mintapéldában ADDR+00 értéke legyen 60000.
Ha a rutin már a memóriában van, gépeljük be és futtassuk a következő BASIC programot:

```
 10 FOR i=0 TO 56 STEP 8
 20 FOR j=11 TO 0 STEP -1
 30 LET a=i: LET b=j: LET c=j:
    LET d=((11-j)*2)+10: LET e=
    ((11-j)*2)+2
 40 GO SUB 100: RANDOMIZE USR 60000
 50 NEXT j: BORDER i/8: NEXT i
 60 GO TO 10
100 POKE 60001,a: POKE 60003,b: POKE 60004,c:
    POKE 60006,d: POKE 60007,e
110 RETURN
```

Több játékprogramban találkozunk hasonló megoldással. A bemutató mintapéldát az egyszerűség kedvéért írtuk meg BASIC-ben, természetesen gyakorlásképpen átírhatjuk gépi kódba és egybeolvaszthatjuk az ismertetett gépi kódú rutinnal.

3.3 CLOCK rutin

Célja: Alaphelyzetben a pontos idő kijelzésére használhatjuk, ez hasznos tájékoztatás lehet, ha nem akarunk elkésni valahonnan. Ez utóbbi célt segíti elő a riasztási idő beállításának lehetősége is. A stopper funkcióval lehetőség nyílik BASIC vagy gépi kódú programjaink futási idejének lemérésére illetve összehasonlítására.

Az itt ismertetett rutinnal több TOOLKIT programban is találkozunk. Lényeg, hogy egy óra:perc:másodperc felosztású órát járatunk a képernyő jobb felső sarkában.
Mivel a rutinban igen sok az abszolút címre hivatkozás, másrészt mindenképpen a memóriaterület végén érdemes elhelyezni, ezért a rutint kötött memóriahelyre töltjük be. Ennek következtében így ebben a formában nem helyezhető át máshova, csak ha átírjuk a megfelelő címeket is.

Az óra bekapcsolása a RANDOMIZE USR 65159 utasítással történik, megállítása pedig a RANDOMIZE USR 65152 utasítással lehetséges. Természetesen az is megoldható, hogy kikapcsoljuk a képernyőről, miközben tovább jár, de erről majd később lesz szó.

Az óra:perc:másodperc értékeit három memóriahelyen tudjuk beállítani:

```
POKE 65364,óra
POKE 65365,perc
POKE 65366,másodperc
```

A memóriahelyre beírandó értéket számolnunk kell az

N = (16 * első számjegy) + második számjegy

képlet segítségével.
Ha az órát 11-re akarjuk beállítani, úgy a 65364-es címre 17-et kell bevinnünk.
A rutin saját, időzítő ciklus szerint működik, azaz nem a FRAMES rendszerváltozó értéke szerint lépteti az időt.

A 216 byte-ból álló CLOCK rutin 5 kisebb modulra bontható:

-az óra be/kikapcsolása
-az óra léptetése
-a kiíratandó karakterek szétosztása
-kétszámjegyű időérték bontása egy memóriahelyről
-karakter kiíratás

Tekintsük át most az egyes modulokat:

-az óra be/kikapcsolása

```
65152   62,63           LD   A,63
65154   237,86          IM   1          ;Kikapcsoljuk az órát és
65156   237,71          LD   I,A        ;I-be visszatöltjük az 1-
                                        ;es megszakítási módban
                                        ;szükséges értéket (63).
65158   201             RET             ;Vége
-------------------------------------------------------------------
65159   1,149,254       LD   BC,65173   ;A 65023. és 65024. memó-
65162   237,67,255,253  LD   (65023),BC ;riahelyekre betöltjük a
                                        ;megszakításvezérelt ru-
                                        ;tin kezdőcímét.
65166   62,253          LD   A,253      ;Az I regiszterbe töltött
65168   237,71          LD   I,A        ;érték alapján a 2-es
                                        ;megszakítási módban meg-
                                        ;címezzük a 65023.rekeszt
65170   237,94          IM   2          ;Áttérünk 2-es megszakí-
                                        ;tási módba, vagyis be-
                                        ;kapcsoljuk az órát.
65172   201             RET             ;Vége
```

-az óra léptetése

```
65173   221,229         PUSH IX         ;Kimentjük a regisztere-
65175   245             PUSH AF         ;ket, hogy visszatéréskor
65176   197             PUSH BC         ;biztosítsuk az azonos
65177   213             PUSH DE         ;állapotot.
65178   229             PUSH HL
65179   58,83,255       LD   A,(65363)  ;Beolvassuk, csökkentjük
65182   61              DEC  A          ;eggyel és visszatöltjük
65183   50,83,255       LD   (65363),A  ;az időciklus változójá-
                                        ;nak értékét.
```

```
65186   194,231,254     JP    NZ,65255    ;Ha ez a változó nem zé-
                                          ;rus, ugrás a 65255.címre
65189   62,50           LD    A,50        ;Ha zérus, akkor ismét
65191   50,83,255       LD    (65363),A   ;50-re állítjuk be és
                                          ;visszatöltjük.
65194   0               NOP               ;Helyet hagyunk kilépési
65195   0               NOP               ;lehetőségre, a szerepét
65196   0               NOP               ;később ismertetjük.
65197   58,86,255       LD    A,(65366)   ;Beolvassuk a másodperc
65200   167             AND   A           ;értékét, majd növeljük
65201   206,1           ADC   A,1         ;eggyel és BCD korrekciót
65203   39              DAA               ;végzünk.
65204   50,86,255       LD    (65366),A   ;Az értéket visszatölt-
                                          ;jük.
65207   254,96          CP    96          ;Eléri a másodperc a 60-
65209   194,231,254     JP    NZ,65255    ;at? Ugrás, ha nem.
65212   175             XOR   A           ;Máskülönben a másodperc
65213   50,86,255       LD    (65366),A   ;értékét lenullázzuk.
65216   58,85,255       LD    A,(65365)   ;Beolvassuk a perc érté-
65219   167             AND   A           ;két, majd növeljük egy-
65220   206,1           ADC   A,1         ;gyel és BCD korrekciót
65222   39              DAA               ;végzünk.
65223   50,85,255       LD    (65365),A   ;Az értéket visszatölt-
                                          ;jük.
65226   254,96          CP    96          ;Eléri a perc a 60-at ?
65228   194,231,254     JP    NZ,65255    ;Ugrás, ha nem.
65231   175             XOR   A           ;Máskülönben a perc érté-
65232   50,85,255       LD    (65365),A   ;két lenullázzuk.
65235   58,84,255       LD    A,(65364)   ;Beolvassuk az óra érté-
65238   167             AND   A           ;két, majd növeljük egy-
65239   206,1           ADC   A,1         ;gyel és BCD korrekciót
65241   39              DAA               ;végzünk.
65242   50,84,255       LD    (65364),A   ;Az értéket visszatölt-
                                          ;jük.
65245   254,19          CP    19          ;Eléri az óra a 13-at?
65247   194,231,254     JP    NZ,65255    ;Ugrás, ha nem.
65250   62,1            LD    A,1         ;Máskülönben az óra érté-
65252   50,84,255       LD    (65364),A   ;két 1-re állítjuk és ezt
                                          ;visszatöltjük.
```

-a kiíratandó karakterek szétosztása

```
65255   221,33,24,64    LD    IX,16408    ;Betöltjük a kiírás kez-
                                          ;dőpozícióját.
65259   58,84,255       LD    A,(65364)   ;Betöltjük az óra érté-
65262   205,26,255      CALL  65306       ;két és kiíratjuk.
65265   62,10           LD    A,10        ;Kiíratunk egy kettős-
65267   205,45,255      CALL  65325       ;pontot.
65270   58,85,255       LD    A,(65365)   ;Betöltjük a perc értékét
65273   205,26,255      CALL  65306       ;és kiíratjuk.
65276   62,10           LD    A,10        ;Kiíratunk egy kettős-
65278   205,45,255      CALL  65325       ;pontot.
65281   58,86,255       LD    A,(65366)   ;Betöltjük a másodperc
```

```
65284   205,26,255      CALL 65306      ;értékét és kiíratjuk.
65287   33,24,88        LD   HL,22552   ;HL-be betöltjük a kiírás
                                        ;első pozíciójának ATTR
                                        ;címét.
65290   6,8             LD   B,8        ;8 karaktert jelenítünk
                                        ;meg.
65292   54,199          LD   (HL),199   ;A karakter fekete-fehé-
                                        ;ren villog.
65294   35              INC  HL         ;HL tovább lép.
65295   16,251          DJNZ 65292      ;Ciklus 8-szor.
65297   225             POP  HL         ;Visszatöltjük a kimen-
65298   209             POP  DE         ;tett regisztereket.
65299   193             POP  BC
65300   241             POP  AF
65301   221,225         POP  IX
65303   195,56,0        JP   56         ;A ROM 56. (HEX-38) címét
                                        ;meghívva gondoskodunk a
                                        ;billentyűzettel kapcso-
                                        ;latos funkciókról.
```

-kétszámjegyű időérték bontása egy memóriahelyről

Az időértékeket úgy tároltuk el három memóriahelyen, hogy első számjegyüknek a 16 szorosát vettük, majd hozzáadtuk a második számjegyet. Ebből adódóan az első számjegy a felső négy biten (4-7), míg a második számjegy az alsó négy biten (0-3) foglal helyet.

```
65306   245             PUSH AF         ;Kimentjük az A regiszter
65307   203,63          SRL  A          ;tartalmát, majd a felső
65309   203,63          SRL  A          ;négy bitet áttoljuk az
65311   203,63          SRL  A          ;alsó négy bit helyére.
65313   203,63          SRL  A
65315   205,45,255      CALL 65325      ;Kiíratjuk az idő első
                                        ;számjegyét.
65318   241             POP  AF         ;Visszahívjuk az időérté-
65319   230,15          AND  15         ;ket, és töröljük a felső
                                        ;négy bitet.
65321   205,45,255      CALL 65325      ;Kiíratjuk az idő második
                                        ;számjegyét is.
65324   201             RET             ;Visszatérés a szubrutin-
                                        ;ból.
```

-karakter kiíratás

```
65325   221,229         PUSH IX         ;Kimentjük a kiírási po-
                                        ;zíciót.
65327   42,54,92        LD   HL,(23606) ;Betöltjük HL-be a karak-
                                        ;termutató értékét.
65330   17,128,1        LD   DE,384     ;Az itt látható számítási
65333   25              ADD  HL,DE      ;mechanizmus segítségével
65334   235             EX   DE,HL      ;kikeressük az aktuális
65335   111             LD   L,A        ;karakter megfelelő byte-
```

```
65336   38,0          LD   H,0        ;ját a karaktermutató ál-
65338   41            ADD  HL,HL      ;tal meghatározott karak-
65339   41            ADD  HL,HL      ;tertáblából.
65340   41            ADD  HL,HL
65341   25            ADD  HL,DE
65342   17,0,1        LD   DE,256     ;A karakter byte-jai egy-
                                      ;más alatt 256 byte-on-
                                      ;ként helyezkednek el.
65345   6,8           LD   B,8        ;1 karakter = 8 byte.
65347   126           LD   A,(HL)     ;A-ban találjuk a megfe-
                                      ;lelő byte-ot.
65348   238,255       XOR  255        ;A byte inverzét képez-
                                      ;zük.
65350   221,119,0     LD   (IX+0),A   ;A nyolc byte egymás a-
65353   35            INC  HL         ;latt megjelenik a képer-
65354   221,25        ADD  IX,DE      ;nyőn.
65356   16,245        DJNZ 65348      ;Ciklus 8-szor.
65358   221,225       POP  IX         ;Visszatöltjük és eggyel
65360   221,35        INC  IX         ;növeljük a kiírási pozí-
                                      ;ciót.
65362   201           RET             ;Visszatérés a szubrutin-
                                      ;ból.
```

Ha beolvastuk a rutint, a pontos idő beállítása előtt írjuk be a 65363. memóriahelyre az időciklus változójának alapértékét, az 50-et: POKE 65363,50 (ENTER).

Állítsuk be a pontos időt és indítsuk el az órát a RANDOMIZE USR 65159 utasítással. Az óra járni kezd a képernyő jobb felső sarkában. (Természetesen a képernyőn bárhol megjeleníthető, csak a megfelelő kiírási pozíciókat kell átírni.)

Most nézzük meg, hogyan tudjuk az órát kikapcsolni a képernyőről, vagyis nem látjuk az órát, de az változatlanul jár. Ehhez a kiíratási rutin belépő pontjára kell egy RET utasítást elhelyezni: POKE 65325,201. Ha vissza akarjuk kapcsolni az órát a képernyőre, ugyanerre a címre vissza kell írni a 221-es kódot.

Most térjünk vissza arra a pontra, hogy miért is hagytunk üresen három memóriarekeszt (65194, 65195 és 65196) ? Itt elhelyezhetünk egy kiugró utasítást több célból is. A TOOLKIT programokban ezek az óra rutinok általában "riasztási" (ALARM) üzemmóddal lettek kombinálva. Alakítsuk át most mi is rutinunkat úgy hogy ha az idő elér egy megadott értéket, akkor hangjelzéssel jelezzen.

Ehhez készíteni kell egy szubrutint, amelyre a programból az említett helyen kiugrunk. Az ALARM megvalósításához választani kell ismét három címet, ahol megadjuk a "riasztási" időt, vagyis ha az óra eléri ezt az értéket, akkor jelez.

Ezek a címek legyenek a következők: 65061, 65062 és 65063.

A riasztási idő beállítását pedig a pontos idő beállításához hasonlóan végezzük el:

```
POKE 65061,óra
POKE 65062,perc
POKE 65063,másodperc
```

Az ALARM szubrutint helyezzük el a 65064-65113 memóriacímek között.

```
65064   58,37,254        LD   A,(65061)   ;Beolvassuk a riasztási
                                          ;óra értéket.
65067   237,75,83,255    LD   BC,(65363)  ;B-be töltjük az óra pil-
                                          ;lanatnyi értékét.
65071   184              CP   B           ;Elérte az óra pillanat-
                                          ;nyi értéke a riasztási
                                          ;óra értéket ?
65072   194,173,254      JP   NZ,65197    ;Ha nem, ugrás vissza.
65075   58,38,254        LD   A,(65062)   ;Beolvassuk a riasztási
                                          ;perc értéket.
65078   237,75,84,255    LD   BC,(65364)  ;B-be töltjük a perc pil-
                                          ;lanatnyi értékét.
65082   184              CP   B           ;Elérte a perc pillanat-
                                          ;nyi értéke a riasztási
                                          ;perc értéket ?
65083   194,173,254      JP   NZ,65197    ;Ha nem, ugrás vissza.
65086   58,39,254        LD   A,(65063)   ;Beolvassuk a riasztási
                                          ;másodperc értéket.
65089   237,75,85,255    LD   BC,(65365)  ;B-be töltjük a másodperc
                                          ;pillanatnyi értékét.
65093   184              CP   B           ;Elérte a másodperc pil-
                                          ;lanatnyi értéke a riasz-
                                          ;tási másodperc értéket ?
65094   194,173,254      JP   NZ,65197    ;Ha nem, ugrás vissza.
65097   62,78            LD   A,78        ;A programból való kiug-
65099   50,171,254       LD   (65195),A   ;rás címét a riasztásra
                                          ;vezéreljük.
65102   17,10,0          LD   DE,10       ;A hang tartama.
65105   33,0,6           LD   HL,1536     ;A hang magassága.
65108   205,181,3        CALL 949         ;Meghívjuk a ROM BEEPER
                                          ;rutinját.
65111   195,173,254      JP   65197       ;Ugrás vissza.
```

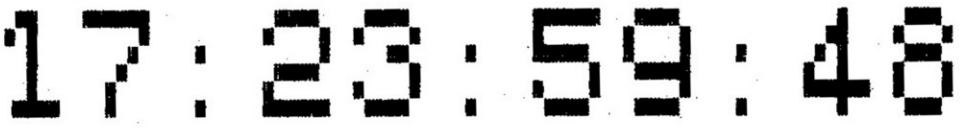
Ha az ALARM szubrutint beolvastuk a memóriába, állítsuk meg az óránkat, ha az jár, majd ezt követően írjuk be a fő-programba a kiugró utasítást:

```
65194   195,40,254       JP   65064       ;Meghívjuk az ALARM szub-
                                          ;rutint.
```

Ha a pillanatnyi idő eléri a riasztási időt, úgy a programból való kiugrás címe megváltozik (65064 helyett 6510? lesz) és minden egyes másodperc léptetésénél rövid hangjelzéssel jelez. A riasztási hangjelzés kikapcsolása úgy lehetséges, hogy a 65195. címre visszaírjuk a 40-es kódot, tehát: POKE 65195,40 (ENTER).

Meg kell jegyeznünk, hogy a BEEPER ROM rutin tiltja a maszkolható megszakításokat, így az óra minden hangjelzés alatt, tehát másodpercenként kb. 5 század másodpercet késni fog.

Ha kipróbáltuk és működik, tegyünk egy újabb lépést. Alakítsuk át az óránkat stopperré. Az 1/50-ed másodperces ciklusidőt figyelembe véve jelenlegi rutinunkat problémamentesen két század másodperc pontosságú stopperré alakíthatjuk át.



A stopper szubrutint helyezzük el a memóriában a 65025-65060 memóriacímek közé.

```
65025   58,87,255       LD   A,(65367)   ;Betöltjük a tized/század
                                         ;másodperc értékét.
65028   167             AND  A           ;Az értéket 2-vel növel-
65029   206,2           ADC  A,2         ;jük és BCD korrekciót
65031   39              DAA              ;végzünk.
65032   50,87,255       LD   (65367),A   ;Az időértéket vissza-
                                         ;töltjük.
65035   254,0           CP   0           ;Az érték eléri a 100-at?
65037   194,231,254     JP   NZ,65255    ;Ugrás, ha nem.
65040   195,173,254     JP   65197       ;Máskülönben ugrás vissza
65043   0               NOP              ;a 65197. címre.
-------------------------------------------------------------------
65044   205,26,255      CALL 65306       ;Előző számjegy kiíratá-
                                         ;sa.
65047   62,10           LD   A,10        ;Kíratunk egy kettős-
65049   205,45,255      CALL 65325       ;pontot.
65052   58,87,255       LD   A,(65367)   ;Betöltjük a tized/század
65055   205,26,255      CALL 65306       ;másodperc értéket és ki-
                                         ;íratjuk.
65058   195,7,255       JP   65287       ;Ugrás vissza.
```

Mint látható a stopper szubrutin két elválasztott részből áll, hiszen külön meg kell oldani az értékek léptetését és a számjegyek kiíratását is. A stopper aktivizálásához eredeti programunkban néhány cím tartalmát módosítani kell.
Először állítsuk meg az órát, ha jár (RANDOMIZE USR 65152), majd gépeljük be a következő változtatásokat:

```
POKE 65190,1   - Az időt minden 1/50-ed másodperc elteltekor
                 léptetjük.
POKE 65195,1   - A 65194. címről a 65025. címre ugrunk.
POKE 65257,21  - A kiírás kezdőpozíciója 16405. lesz.
POKE 65284,195: POKE 65285,20: POKE 65286,254
               - Leágazunk a kiíratási rutinból.
POKE 65288,21  - Az ATTR kezdőpozíció 22549. lesz.
POKE 65291,11  - 11 karaktert jelenítünk meg.
```

Ha mindent módosítottunk ismét indítsuk el az órát: RANDOMIZE USR 65159 és lám a stopper megjelenik.

Amennyiben a memóriában van az ébresztő rutin is, próbáljuk meg összehangolni ezeket.
Ha jár az óra, állítsuk meg, és végezzük el a következő módosításokat:

POKE 65073,1: POKE 65084,1: POKE 65095,1: POKE 65112,1
- Az ébresztő rutinról a vezérlés a stopper rutinra adódik át.

POKE 65103,1 - A hang tartamát le kell vennünk két század másodperc alá, mert problémába ütközhetünk.

POKE 65195,40 - A 65194. címről a 65064. címre ugrunk.

Újra futtatva stopperként is üzemelő óránkat már riasztásra is felhasználhatjuk.

Az óra jelenlegi állapotában 12 órás, vagyis 12 óra után 01 következik. Ha át akarjuk állítani 24 órás órára, akkor a következő két cím tartalmát kell módosítani:

```
POKE 65246,36
POKE 65251,0
```

A léptető rutinban a DAA gépi kódú utasítás BCD (Binárisan Kódolt Decimális) korrekciót hajt végre. Ennek megértésére nézzünk egy példát. A perc értéke pl. éppen 39-ről lép 40-re. A 39 a memóriában: (3*16)+9 = 57 kódként tárolódik. Az ADC A,1 utasítás ezt az értéket növeli meg eggyel, tehát 57-ből 58 lesz, viszont a 40-nek megfelelő kód ettől eltérő: (4*16)+0 = 64. Ebből adódóan a megfelelő korrekciót el kell végezni, és erre alkalmas a DAA utasítás.

A CLOCK rutin még tovább bővíthető, finomítható. Azt is megoldhatjuk, hogy a digitális számjegyek kódjait elhelyezzük a memóriában, és a 65328/29 címeken a karaktermutató értékét megfelelően átállítjuk. Ennek következtében az óra saját készítésű digitális számjegyekkel fog a képernyőn megjelenni.
Természetesen a rutint egy egész képernyőt betöltő óra vezérlésére is felhasználhatjuk. A további kimunkálást az olvasóra bízzuk.

Megjegyzés: Reméljük a RAMTOP megfelelő átállítása (CLEAR 65022) nem lett elmulasztva !

3.4 DELETE rutin

Célja: Több BASIC sor gyors törlése.

Szinte valamennyi ismert BASIC bővítő és TOOLKIT program utasításkészletében megtalálható. Nagy terjedelmű BASIC program kezelése közben gyakran szükségünk lehet nagyobb blokkok eltávolítására. A NEW sajnos paraméter nélküli. Mennyivel könnyebb lenne pl. csak ennyit kiadni: NEW 100-3000, a 100.-3000.-ig terjedő sorok törléséhez. Ez sajnos nem így van. Segéd software nélkül

sorok törlésére a legegyszerűbb módszer az, hogy begépeljük a törlendő sor sorszámát és megnyomjuk az ENTER-t. Ez a módszer nagyobb területek törlése esetén idegölő tevékenység.

Az itt közölt 96 byte hosszú rutin segítségével gyorsan és hatékonyan törölhetünk bármekkora blokkot BASIC programjainkból. A rutin bárhol elhelyezhető a memória szabad területén, de egymagában a legcélszerűbb az UDG terület alatt elhelyezni. Ettől függetlenül ne felejtsük el a RAMTOP megfelelő beállítását!

```
ADDR+00   237,75,176,92 LD   BC,(23728) ;BC-be töltjük az első
                                        ;törlendő sor sorszámát.
ADDR+04   205,X,Y       CALL ADDR+62    ;Meghívjuk a címkereső
                                        ;szubrutint. X és Y az
                                        ;ADDR+62 .cím alsó és
                                        ;felső byte-ja.
ADDR+07   229           PUSH HL         ;Az első törlendő sor
                                        ;címét kimentjük.
ADDR+08   237,75,118,92 LD   BC,(23670) ;BC-be töltjük az utolsó
                                        ;törlendő sor sorszámát.
ADDR+12   205,X,Y       CALL ADDR+62    ;Meghívjuk a címkereső
                                        ;szubrutint.
ADDR+15   35            INC  HL         ;HL tovább lép a sor
ADDR+16   35            INC  HL         ;hosszát meghatározó me-
                                        ;móriahelyre.
ADDR+17   94            LD   E,(HL)     ;DE-be betöltjük az u-
ADDR+18   35            INC  HL         ;tolsó törlendő program
ADDR+19   86            LD   D,(HL)     ;sor hosszát.
ADDR+20   25            ADD  HL,DE      ;HL-ben számítjuk ki az
ADDR+21   35            INC  HL         ;első megmaradó sor cí-
ADDR+22   209           POP  DE         ;mét, majd visszatöltjük
                                        ;az első törlendő sor
                                        ;címét.
ADDR+23   167           AND  A          ;Kiszámoljuk a törlendő
ADDR+24   237,82        SBC  HL,DE      ;byte-ok számát és ki-
ADDR+26   229           PUSH HL         ;mentjük a verembe.
ADDR+27   25            ADD  HL,DE      ;DE-be kerül az első
ADDR+28   235           EX   DE,HL      ;megmaradó sor címe, az
ADDR+29   229           PUSH HL         ;első törlendő sor címét
                                        ;pedig kimentjük.
ADDR+30   42,101,92     LD   HL,(23653) ;HL-be töltjük a tarta-
                                        ;lék memóriaterület cí-
                                        ;mét.
ADDR+33   167           AND  A          ;Levonjuk az első meg-
ADDR+34   237,82        SBC  HL,DE      ;maradó sor címét, majd
ADDR+36   68            LD   B,H        ;áttöltjük a BC regisz-
ADDR+37   77            LD   C,L        ;terpárba.
ADDR+38   225           POP  HL         ;Visszatöltjük az első
ADDR+39   235           EX   DE,HL      ;törlendő sor címét és
                                        ;felcseréljük az első
                                        ;megmaradó sor címével.
ADDR+40   237,176       LDIR            ;A program felesleges
                                        ;területét felülírjuk a
                                        ;megmaradó sorokkal.
```

```
ADDR+42   42,75,92        LD   HL,(23627) ;HL-be betöltjük a vál-
                                          ;tozóterület címét.
ADDR+45   209             POP  DE         ;A veremből elővesszük a
                                          ;törölt byte-ok számát.
ADDR+46   167             AND  A          ;A törölt byte-ok szám-
ADDR+47   237,82          SBC  HL,DE      ;nak megfelelően előbbre
ADDR+49   34,75,92        LD   (23627),HL ;hozzuk a változóterület
                                          ;címét.
ADDR+52   42,89,92        LD   HL,(23641) ;Ezt követően a beírás
ADDR+55   167             AND  A          ;alatt álló parancs cí-
ADDR+56   237,82          SBC  HL,DE      ;mét is a törölt byte-ok
ADDR+58   34,89,92        LD   (23641),HL ;számának megfelelően
                                          ;kell átállítani.
ADDR+61   201             RET             ;Vége

-címkereső szubrutin

ADDR+62   42,83,92        LD   HL,(23635) ;HL-be töltjük a BASIC
ADDR+65   86              LD   D,(HL)     ;terület kezdőcímét, DE-
ADDR+66   35              INC  HL         ;be pedig a HL címe ál-
ADDR+67   94              LD   E,(HL)     ;tal meghatározott sor-
                                          ;számot.
ADDR+68   235             EX   DE,HL      ;Felcseréljük a cím és a
                                          ;hozzá tartozó sorszám
                                          ;értékét.
ADDR+69   167             AND  A          ;Megvizsgáljuk, hogy a
ADDR+70   237,66          SBC  HL,BC      ;törlendő sor megegye-
ADDR+72   9               ADD  HL,BC      ;zik-e a program aktuá-
ADDR+73   235             EX   DE,HL      ;lis sorával.
ADDR+74   40,18           JR   Z,ADDR+94  ;Ugrás, ha igen.
ADDR+76   35              INC  HL         ;DE-be betöltjük az ak-
ADDR+77   94              LD   E,(HL)     ;tuális BASIC sor hosz-
ADDR+78   35              INC  HL         ;szát.
ADDR+79   86              LD   D,(HL)
ADDR+80   35              INC  HL
ADDR+81   25              ADD  HL,DE      ;HL-ben számítjuk ki a
                                          ;következő BASIC sor cí-
                                          ;mét.
ADDR+82   237,91,75,92    LD   DE,(23627) ;DE-be beolvassuk a vál-
                                          ;tozók címét.
ADDR+86   167             AND  A          ;Megnézzük, hogy van-e
ADDR+87   237,82          SBC  HL,DE      ;még BASIC ?
ADDR+89   25              ADD  HL,DE
ADDR+90   56,229          JR   C,ADDR+65  ;Ha van, ugrás vissza.
ADDR+92   207             RST  08         ;Ellenkező esetben a ru-
ADDR+93   10              DEFB 10         ;tin leáll "Integer out
                                          ;of range" hibaüzenet-
                                          ;tel.
ADDR+94   43              DEC  HL         ;Visszalépünk a BASIC
                                          ;sor kezdőcímére.
ADDR+95   201             RET             ;Visszatérés a szubru-
                                          ;tinból.
```

A rutin futtatása előtt az első és utolsó törlendő sor sorszámát meg kell adni. A 23728/29 rendszerváltozókat a gép nem használja, így szabadon felhasználhatók. Kapóra jön a SEED rendszerváltozó (23670/71) is.

Legyen "a" az első törlendő sor sorszáma, ill."b" az utolsóé. Gépeljük be sorban:

```
POKE 23728,a-256*INT(a/256)
POKE 23729,INT(a/256)
RANDOMIZE b
RANDOMIZE USR ADDR+00
```

és ismételt listázásnál az a-b közé eső sorok már nem fognak megjelenni.

Megjegyzés: Az első és utolsó törlendő sor számának léteznie kell.

3.5 EXAMINE rutin

Célja: Program-fejlécek tartalmának kiolvasása és kijelzése a képernyőn.

A rutinban elég sok az abszolút címhivatkozás, így kötött memóriaterületre ismertetjük a gépi kódú rutint. A rutin kezdőcíme a 64716. memóriarekesz és a rutin 652 byte hosszon egészen az UDG karakterkészlet kezdőcíméig tart. Ne ijedjünk meg, a mintegy fél kbyte hosszúság nem a program bonyolultságát tükrözi, elég sok string kiíratását oldjuk meg a képernyőn, s ez jelentősen megnöveli a rutin hosszát.

A rutin a RANDOMIZE USR 64716 utasítással hívható meg, ekkor várja a kazettás magnetofonról a fejléc betöltését. A fejléc betöltődése után pedig kiírja az aktuális adatokat. A végén választhatunk, hogy akarunk-e új fejlécet beolvasni, vagy kilépünk a programból.

```
64716   205,107,13      CALL 3435       ;Meghívjuk a ROM-beli CLS
                                        ;törlőrutint (HEX-0D6B)
64719   33,220,252      LD   HL,64732   ;HL-ben megadjuk az éke-
                                        ;zetes karakterek kezdő-
                                        ;címét,
64722   17,88,255       LD   DE,65368   ;DE-ben pedig az UDG te-
                                        ;rület kezdőcímét.
64725   1,56,0          LD   BC,56      ;8 karaktert generálunk
                                        ;az UDG területen.
64728   237,176         LDIR            ;Áthelyezzük a byte-okat
                                        ;az UDG területre.
64730   24,56           JR   64788      ;Ugrás a 64788.címre.
64732   0,40,0,56,68,   DEFB 0,40,..
        68,56,0         ...56,0         ;kis "ö" betű.
64740   16,16,56,68,    DEFB 16,16,..
        120,64,60,0     ...60,0         ;kis "é" betű.
64748   16,16,0,56,     DEFB 16,16,..
        68,68,56,0      ...56,0         ;kis "ó" betű.
```

```
64756   16,16,0,48,     DEFB 16,16,..
        16,16,56,0      ...56,0         ;kis "í" betű.
64764   8,8,56,4,60,    DEFB 8,8,..
        68,60,0         ...60,0         ;kis "á" betű.
64772   8,8,60,66,66,   DEFB 8,8,..
        66,60,0         ...60,0         ;nagy "Ó" betű.
64780   8,74,66,66,66,  DEFB 8,74,..
        66,60,0         ...60,0         ;nagy "Ú" betű.
64788   62,2            LD   A,2        ;Megnyitjuk a képernyő
64790   205,1,22        CALL 5633       ;felső részét (ROM rutin,
                                        ;HEX-1601).
64793   62,22           LD   A,22       ;Az AT kódja.
64795   215             RST  16
64796   62,0            LD   A,0        ;A sor-pozíció.
64798   215             RST  16
64799   62,4            LD   A,4        ;Az oszlop-pozíció.
64801   215             RST  16
64802   17,45,253       LD   DE,64813   ;Megadjuk az adott helyen
                                        ;kiíratandó string első
                                        ;karakterének címét.
64805   1,54,0          LD   BC,54      ;54 karaktert írunk ki.
64808   205,60,32       CALL 8252       ;Meghívjuk a string-ki-
                                        ;irató ROM rutint (HEX-
                                        ;203C).
64811   24,55           JR   64868      ;Ugrás a 64868.címre.
64813   70,69,74,79,    DEFM "FEJOLVASÓ(EXAMINE) RUTIN
        76,86,65,83,            Indítsd el a magnót!"
        149,40,69,88,
        65,77,73,78,                    ;szöveg elhelyezése a me-
        69,41,32,82,                    ;móriában.
        85,84,73,78,
        32,32,32,32,
        32,32,32,32,
        32,32,73,110,
        100,147,116,115,
        100,32,101,108,
        32,97,32,109,
        97,103,110,146,
        116,33
64867   0               DEFB 0          ;A szöveg lezárása.
64868   55              SCF             ;Az átviteli jelzőbitet
                                        ;1-re állítjuk.
64869   62,0            LD   A,0        ;A-ba töltjük a szinkron-
                                        ;byte értékét.
64871   221,33,117,253  LD   IX,64885   ;A fejlécet a 64885. cím-
                                        ;től töltjük be a memóri-
                                        ;ába.
64875   17,17,0         LD   DE,17      ;A fej 17 byte hosszú.
64878   205,86,5        CALL 1366       ;Belépünk a ROM LOAD ru-
                                        ;tinjába(HEX-0556).
64881   48,241          JR   NC,64868   ;17 byte-tól eltérő fej,
                                        ;vagy hibás betöltés ese-
                                        ;tén ugrás vissza.
```

```
64883   24,17           JR   64902     ;Ugrás a 64902. címre.
64885   0,0,0,0,0,0,    DEFS 17        ;17 byte-ot szabadon ha-
        0,0,0,0,0,0,0,                 ;gyunk, ide kerül a fej-
        0,0,0,0,0                      ;információ.
64902   58,117,253      LD   A,(64885) ;A-ba betöltjük a fejléc
                                       ;első byte-ját.
64905   167             AND  A         ;Ha az AND A művelet
                                       ;eredményeképpen 1 lesz
                                       ;a zérus jelzőbitben, ak-
64906   202,39,254      JP   Z,65063   ;kor BASIC programról van
                                       ;szó és ugrás történik a
                                       ;65063. címre.
64909   254,1           CP   1         ;Ha a vizsgált byte érté-
                                       ;ke 1, akkor numerikus
                                       ;tömbről van szó, és ug-
64911   202,213,253     JP   Z,64981   ;rás történik a 64981.
                                       ;címre.
64914   214,2           SUB  2         ;Ha a vizsgált byte-ból
                                       ;levonunk kettőt és az
                                       ;eredmény zérus, akkor
                                       ;string-tömb, és ugrás
64916   202,254,253     JP   Z,65022   ;történik a 65022. címre.
64919   17,217,254      LD   DE,65241  ;DE-be töltjük a "Gépi
                                       ;kód" szöveg kezdőcímét.
64922   1,8,0           LD   BC,8      ;8 karakter hosszú a szö-
                                       ;veg.
64925   46,5            LD   L,5       ;A kiírás sor pozíciója.
64927   205,194,254     CALL 65218     ;Meghívjuk a szövegkiíró
                                       ;szubrutint.
64930   17,206,254      LD   DE,65230  ;DE-be töltjük a "File
                                       ;neve:"szöveg kezdőcímét.
64933   1,10,0          LD   BC,10     ;A szöveg 10 karakter
                                       ;hosszú.
64936   46,7            LD   L,7       ;A kiírás sor pozíciója.
64938   205,194,254     CALL 65218     ;Meghívjuk a szövegkiíró
                                       ;szubrutint.
64941   205,172,254     CALL 65291     ;Meghívjuk a file nevét
                                       ;kiíró szubrutint.
64944   17,11,255       LD   DE,65291  ;DE-be töltjük az "Adatok
                                       ;hossza:" szöveg kezdőcí-
                                       ;mét.
64947   1,14,0          LD   BC,14     ;A szöveg hossza 14 ka-
                                       ;rakter.
64950   46,9            LD   L,9       ;A kiírás sor pozíciója.
64952   205,194,254     CALL 65218     ;Meghívjuk a szövegkiíró
                                       ;szubrutint.
64955   205,183,254     CALL 65207     ;Meghívjuk az adatok
                                       ;hosszát kiíró szubrutint
64958   17,63,255       LD   DE,65343  ;DE-be töltjük a "Start-
                                       ;cím:" szöveg kezdőcímét.
64961   1,9,0           LD   BC,9      ;9 karakter hosszú a szö-
                                       ;veg.
64964   46,11           LD   L,11      ;A kiírás sor pozíciója.
```

```
64966   205,194,254      CALL 65218      ;Meghívjuk a szövegkiíró
                                         ;szubrutint.
64969   237,75,130,253   LD   BC,(64898) ;BC-be töltjük a startcím
                                         ;értékét.
64973   205,43,45        CALL 11563      ;Meghívjuk a BC-t a lebe-
                                         ;gőpontos verembe töltő
                                         ;ROM rutint(HEX-2D2B).
64976   205,227,45       CALL 11747      ;Meghívjuk a lebegőpontos
                                         ;verem tartalmát kiíratő
                                         ;ROM rutint(HEX-2DE3).
64979   24,39            JR   65020      ;Ugrás a 65020. címre.
-------------------------------------------------------------------
64981   17,240,254       LD   DE,65264   ;DE-be töltjük a "Nume-
                                         ;rikus tömb" szöveg kez-
                                         ;dőcímét.
64984   1,14,0           LD   BC,14      ;14 karakter hosszú a
                                         ;szöveg.
64987   46,5             LD   L,5        ;A kiírás sor pozíciója.
64989   205,194,254      CALL 65218      ;Meghívjuk a szövegkiíró
                                         ;szubrutint.
64992   17,206,254       LD   DE,65230   ;DE-be töltjük a "File
                                         ;neve:"szöveg kezdőcímét.
64995   1,10,0           LD   BC,10      ;10 karakter hosszú a
                                         ;szöveg.
64998   46,7             LD   L,7        ;A kiírás sor pozíciója.
65000   205,194,254      CALL 65218      ;Meghívjuk a szövegkiíró
                                         ;szubrutint.
65003   205,172,254      CALL 65196      ;Meghívjuk a file nevét
                                         ;kiíró szubrutint.
65006   17,11,255        LD   DE,65291   ;DE-be töltjük az "Adatok
                                         ;hossza:" szöveg kezdőcí-
                                         ;mét.
65009   1,14,0           LD   BC,14      ;14 karakter hosszú a
                                         ;szöveg.
65012   46,9             LD   L,9        ;A kiírás sor pozíciója.
65014   205,194,254      CALL 65218      ;Meghívjuk a szövegkiíró
                                         ;szubrutint.
65017   205,183,254      CALL 65207      ;Meghívjuk az adatok
                                         ;hosszát kiíró szubrutint
65020   24,39            JR   65061      ;Ugrás a 65061. címre.
-------------------------------------------------------------------
65022   17,255,254       LD   DE,65279   ;DE-be töltjük a "String
                                         ;tömb" szöveg kezdőcímét
65025   1,11,0           LD   BC,11      ;11 karakter hossszú a
                                         ;szöveg.
65028   46,5             LD   L,5        ;A kiírás sor pozíciója.
65030   205,194,254      CALL 65218      ;Meghívjuk a szövegkiíró
                                         ;szubrutint.
65033   17,206,254       LD   DE,65230   ;DE-be töltjük a "File
                                         ;neve:" szöveg kezdőcímét
65036   1,10,0           LD   BC,10      ;10 karakter hosszú a
                                         ;szöveg.
65039   46,7             LD   L,7        ;A kiírás sor pozíciója.
```

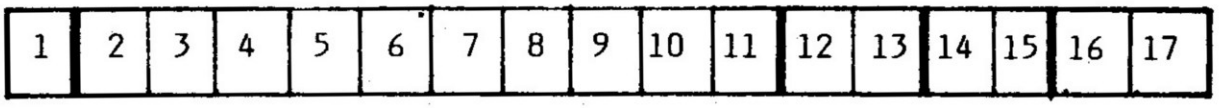
```
65041   205,194,254     CALL 65218      ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65044   205,172,254     CALL 65196      ;Meghívjuk a file nevét
                                        ;kiíró szubrutint.
65047   17,11,255       LD   DE,65291   ;DE-be töltjük az "Adatok
                                        ;hossza:" szöveg kezdőcí-
                                        ;mét.
65050   1,14,0          LD   BC,14      ;14 karakter hosszú a
                                        ;szöveg.
65053   46,9            LD   L,9        ;A kiírás sor pozíciója.
65055   205,194,254     CALL 65218      ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65058   205,183,254     CALL 65207      ;Meghívjuk az adatok
                                        ;hosszát kiíró szubrutint
65061   24,98           JR   65161      ;Ugrás a 65161.címre.
-------------------------------------------------------------------
65063   17,226,254      LD   DE,65320   ;DE-be töltjük a "BASIC
                                        ;program" szöveg kezdőcí-
                                        ;mét.
65066   1,13,0          LD   BC,13      ;13 karakter hosszú a
                                        ;szöveg.
65069   46,5            LD   L,5        ;A kiírás sor pozíciója.
65071   205,194,254     CALL 65218      ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65074   17,206,254      LD   DE,65230   ;DE-be töltjük a "File
                                        ;neve:"szöveg kezdőcímét.
65077   1,10,0          LD   BC,10      ;10 karakter hosszú a
                                        ;szöveg.
65080   46,7            LD   L,7        ;A kiírás sor pozíciója.
65082   205,194,254     CALL 65218      ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65085   205,172,254     CALL 65196      ;Meghívjuk a file nevét
                                        ;kiíró szubrutint.
65088   17,11,255       LD   DE,65291   ;DE-be töltjük az "Adatok
                                        ;hossza:" szöveg kezdőcí-
                                        ;mét.
65091   1,14,0          LD   BC,14      ;14 karakter hosszú a
                                        ;szöveg.
65094   46,9            LD   L,9        ;A kiírás sor pozíciója.
65096   205,194,254     CALL 65218      ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65099   205,183,254     CALL 65207      ;Meghívjuk az adatok
                                        ;hosszát kiíró szubrutint
65102   17,26,255       LD   DE,65306   ;DE-be töltjük az "Auto-
                                        ;start:" szöveg kezdőcí-
                                        ;mét.
65105   1,13,0          LD   BC,13      ;13 karakter hosszú a
                                        ;szöveg.
65108   46,11           LD   L,11       ;A kiírás sor pozíciója.
65110   205,194,254     CALL 65218      ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
```

```
65113   42,130,253      LD   HL,(64898) ;HL-be töltjük azt a tar-
                                        ;talmat, amely ha 0-9999
                                        ;közé esik akkor az adott
                                        ;sorszámon autostartos a
                                        ;program.
65116   1,16,39         LD   BC,10000  ;BC-be 10 ezret töltünk.
65119   175             XOR  A         ;Az átviteli jelzőbitet
65120   237,66          SBC  HL,BC     ;töröljük és különbséget
                                        ;képzünk.
65122   250,106,254     JP   M,65130   ;Ha az eredmény negatív,
                                        ;akkor programunk auto-
                                        ;startos és ugrás törté-
                                        ;nik a 65130. címre.
65125   62,45           LD   A,45      ;A-ba töltjük a "-" jel
                                        ;ASCII kódját, és ezt ki-
65127   215             RST  16        ;íratjuk a képernyőre.
                                        ;(nincs autostart)
65128   24,10           JR   65140     ;Ugrás a 65140. címre.
65130   237,75,130,253  LD   BC,(64898) ;BC-be töltjük az auto-
                                        ;start sorszámát.
65134   205,43,45       CALL 11563     ;BC-t áttöltjük a lebegő-
                                        ;pontos verembe.
65137   205,227,45      CALL 11747     ;Kiíratjuk a képernyőre a
                                        ;lebegőpontos verem tar-
                                        ;talmát.
65140   17,40,255       LD   DE,65320  ;DE-be töltjük a "Prog./
                                        ;változók hossza:" szöveg
                                        ;kezdőcímét.
65143   1,22,0          LD   BC,22     ;22 karakter hosszú a
                                        ;szöveg.
65146   46,13           LD   L,13      ;A kiírás sor pozíciója.
65148   205,194,254     CALL 65218     ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65151   237,75,132,253  LD   BC,(64900) ;BC-be töltjük a program/
                                        ;változók hosszát.
65155   205,43,45       CALL 11563     ;Ezt áttöltjük a lebegő-
                                        ;pontos verembe és kií-
65158   205,227,45      CALL 11747     ;ratjuk a képernyőre.
-------------------------------------------------------------------
65161   17,73,255       LD   DE,65353  ;DE-be betöltjük az "Új
                                        ;fejléc I/N?" szöveg kez-
                                        ;dőcímét.
65164   1,14,0          LD   BC,14     ;14 karakter hosszú a
                                        ;szöveg.
65167   46,20           LD   L,20      ;A kiírás sor pozíciója.
65169   205,194,254     CALL 65218     ;Meghívjuk a szövegkiíró
                                        ;szubrutint.
65172   62,223          LD   A,223     ;A jobb oldali, alulról
                                        ;harmadik félsort olvas-
                                        ;suk le.
65174   219,254         IN   A,(254)   ;A beolvasott adatbyte az
                                        ;A regiszterbe kerül.
```

```
65176   230,31          AND   31          ;A felső három bitet tö-
                                          ;röljük.
65178   254,27          CP    27          ;Ha az "I" billentyű lett
65180   40,11           JR    Z,65192     ;megnyomva,ugrás a 65192.
                                          ;címre.
65182   62,127          LD    A,127       ;A jobb alsó félsort ol-
                                          ;vassuk le.
65184   219,254         IN    A,(254)     ;A beolvasott adatbyte az
                                          ;A regiszterbe kerül.
65186   230,31          AND   31          ;A felső három bitet tö-
                                          ;röljük.
65188   254,23          CP    23          ;Ha az "N" billentyű nem
65190   32,236          JR    NZ,65172    ;lett megnyomva, ugrás a
                                          ;65172. címre,
65192   201             RET               ;máskülönben vége.
65193   195,204,252     JP    64716       ;Ugrás az indító címre.
-------------------------------------------------------------------
65196   33,118,253      LD    HL,64886    ;HL-be töltjük a fejléc
                                          ;file nevét tároló kezdő-
                                          ;címet.
65199   6,10            LD    B,10        ;A file neve 10 karakter.
65201   126             LD    A,(HL)      ;Az A regiszterbe tölt-
                                          ;jük az adott karakter
                                          ;ASCII kódját.
65202   215             RST   16          ;Megjelenítjük a karak-
                                          ;tert a képernyőn.
65203   35              INC   HL          ;A címmutatót növeljük a
                                          ;következő karakterre.
65204   16,251          DJNZ  65201       ;Ciklus 10-szer.
65206   201             RET               ;Visszatérés a szubrutin-
                                          ;ból.
-------------------------------------------------------------------
65207   237,75,128,253  LD    BC,(64896)  ;BC-be töltjük az adat-
                                          ;hossz értékét.
65211   205,43,45       CALL  11563       ;BC-t áttöltjük a lebegő-
                                          ;pontos verembe.
65214   205,227,45      CALL  11747       ;Kiíratjuk a képernyőre a
                                          ;lebegőpontos verem tar-
                                          ;talmát.
65217   201             RET               ;Visszatérés a szubrutin-
                                          ;ból.
-------------------------------------------------------------------
65218   62,22           LD    A,22        ;Az "AT" kódja.
65220   215             RST   16
65221   125             LD    A,L         ;Betöltjük A-ba L tartal-
                                          ;mát, ami a sor pozíciót
                                          ;tartalmazza.
65222   215             RST   16
65223   62,0            LD    A,0         ;Az oszloppozíció.
65225   215             RST   16
65226   205,60,32       CALL  8252        ;Meghívjuk a stringkiíró
                                          ;ROM rutint (HEX-203C).
```

```
65229   201             RET             ;Visszatérés a szubrutin-
                                        ;ból.
-------------------------------------------------------------------
65230   70,105,108,101, DEFM "File neve:"
        32,110,101,118,
        101,58
65240   0               DEFB 0
65241   71,145,112,105, DEFM "Gépi kód"
        32,107,146,100
65249   0               DEFB 0
65250   66,65,83,73,67, DEFM "BASIC Program"
        32,80,114,111,
        103,114,97,109
65263   0               DEFB 0
65264   78,117,109,101, DEFM "Numerikus tömb"
        114,105,107,117,
        115,32,116,144,
        109,98
65278   0               DEFB 0
65279   83,116,114,105, DEFM "String tömb"
        110,103,32,116,
        144,109,98
65290   0               DEFB 0
65291   65,100,97,116,  DEFM "Adatok hossza:"
        111,107,32,104,
        111,115,115,122,
        97,58
65305   0               DEFB 0
65306   65,117,116,111, DEFM "Auto indítás:"
        32,105,110,100,
        147,116,148,115,
        58
65319   0               DEFB 0
65320   80,114,111,103, DEFM "Prog./változók hossza:"
        46,47,118,148,
        108,116,111,122,
        146,107,32,104,
        111,115,115,122,
        97,58
65342   0               DEFB 0
65343   83,116,97,114,  DEFM "Startcím:"
        116,99,147,109,
        58
65352   0               DEFB 0
65353   150,106,32,102, DEFM "Új fejléc I/N?"
        101,106,108,145,
        99,32,73,47,78,
        63
65367   0               DEFB 0
-------------------------------------------------------------------
```

A rutinból könnyen kifejthető, de ismétlés jelleggel tekintsük át a fejléc felépítését:



Az első byte határozza meg a program file típusát.

0 = BASIC program
1 = Numerikus tömb
2 = String tömb
3 = Gépi kód

A 2.-11. byte-ok tartalmazzák sorban a file nevének ASCII kódjait. A 12. és a 13. byte együttesen határoz meg egy 16 bites értéket, ez az adatok hossza.
A 14. és 15. byte is együtt számít, szerepe más BASIC és gépi kódú program esetén is:

-BASIC program esetén ez a két byte az automatikus indítás sorszámát tartalmazza, ha értéke 0 és 9999 közé esik. Ettől eltérő esetben (leggyakrabban 32768) a program nem autostartos.
-Gépi kódú program esetén itt tárolódik el az a cím ahonnan az adatokat betöltjük a memóriába. Az itt megadott címet felülbírálhatjuk a LOAD""CODE utasítás mellett megadott paraméterrel, ugyanis akkor betöltéskor a megadott címet veszi figyelembe a gép.

A 16. és 17. byte-oknak BASIC program esetén van jelentőségük, ugyanis a BASIC program + változók együttes hosszát adják meg.

3.6 FILL rutin

Célja: Zárt, vonalakkal határolt terület kitöltése tinta (INK) színnel.

A kitöltés ATTR értékét előzetesen be kell állítani a 23728. címen (Pl. POKE 23728,56 = fekete tinta és fehér papír).

```
ADDR+00   58,125,92    LD   A,(23677)   ;DE-be betöltjük a COORDS
ADDR+03   95           LD   E,A         ;rendszerváltozóból a
ADDR+04   58,126,92    LD   A,(23678)   ;legutoljára megadott
ADDR+07   87           LD   D,A         ;PLOT pozíció koordináta-
                                        ;it, vagyis E regiszterbe
                                        ;az x-et és D regiszterbe
                                        ;az y-t.
ADDR+08   58,176,92    LD   A,23728     ;Az ATTR-T rendszerválto-
ADDR+11   50,143,92    LD   (23695),A   ;zóba (23695) betöltjük a
                                        ;kitöltés ATTRIBUTUM ér-
                                        ;tékét, ezt még a rutin
                                        ;futtatása előtt be kell
                                        ;állítani a 23728-as cí-
                                        ;men.
```

```
ADDR+14   75            LD   C,E        ;BC-be áttöltjük a ki
ADDR+15   66            LD   B,D        ;duló koordináta pozí
                                        ;ót. A rutin futása al
                                        ;DE mindig a kiinduló,
                                        ;pedig a pillanatnyi
                                        ;zíciót tartalmazza.
ADDR+16   205,X,Y       CALL ADDR+71    ;Meghívjuk az egy k
                                        ;pontsort kitöltő szu
                                        ;rutint. "X" és "Y"
                                        ;ADDR+71 cím alsó és fe
                                        ;sõ byte-ja.
ADDR+19   75            LD   C,E        ;C-ben a kiinduló x koo
                                        ;dináta.
ADDR+20   4             INC  B          ;Az y koordinátát növe
                                        ;jük egy pozícióval,
ADDR+21   120           LD   A,B        ;áttöltjük A-ba.
ADDR+22   254,175       CP   175        ;Ha az y koordináta elé
                                        ;a 175. pozíciót (képe
                                        ;nyõ felsõ széle), akk
ADDR+24   48,18         JR   NC,ADDR+44 ;ugrás történik.
ADDR+26   197           PUSH BC         ;Az ideiglenes és kiind
ADDR+27   213           PUSH DE         ;ló koordináta pozíci
                                        ;kimentjük a verembe.
ADDR+28   205,206,34    CALL 8910       ;Meghívjuk a POINT rutin
                                        ;(ROM HEX-22CE), ezutá
ADDR+31   205,213,45    CALL 11733      ;pedig az FP-TO-A rutin
                                        ;(ROM HEX 2DD5).A két RO
                                        ;rutin eredményeképpen
                                        ;megállapítjuk, hogy
                                        ;következõ PLOT pozíci
                                        ;tinta (INK), vagy papí
                                        ;(PAPER) színû? Tinta e
                                        ;setén az A regiszter ér
                                        ;téke 1 egyébként zéru
                                        ;lesz.
ADDR+34   209           POP  DE         ;Visszatöltjük a verem
ADDR+35   193           POP  BC         ;bõl a kiinduló és ideig
                                        ;lenes koordináta pozíci
                                        ;ókat.
ADDR+36   60            INC  A
ADDR+37   61            DEC  A
ADDR+38   254,1         CP   1          ;Ha a következõ pozíci
ADDR+40   32,230        JR   NZ,ADDR+16 ;nem tinta színû, ugrá
                                        ;vissza.
ADDR+42   66            LD   B,D        ;BC-ben ismét a kiindul
ADDR+43   75            LD   C,E        ;koordináta pozíció.
ADDR+44   205,X,Y       CALL ADDR+71    ;Meghívjuk az egy kép
                                        ;pontsort kitöltõ szubru
                                        ;tint. "X" és "Y" az ADDR
                                        ;+71. cím alsó és felsõ
                                        ;byte-ja.
ADDR+47   75            LD   C,E        ;C-ben ismét a kiinduló x
                                        ;koordináta pozíció.
```

```
ADDR+48   5             DEC  B          ;Az y koordinátát csök-
                                         ;kentjük egy pozícióval,
ADDR+49   120           LD   A,B        ;és áttöltjük A-ba.
ADDR+50   254,1         CP   1          ;Ha az y koordináta elé-
ADDR+52   56,16         JR   C,ADDR+70  ;ri a 0. pozíciót (képer-
                                         ;nyő alsó széle), akkor
                                         ;ugrás történik és vége.
ADDR+54   197           PUSH BC         ;Az ideiglenes és kiin-
ADDR+55   213           PUSH DE         ;duló koordináta pozíciót
                                         ;kimentjük a verembe.
ADDR+56   205,206,34    CALL 8910       ;Meghívjuk a POINT ROM
                                         ;rutint, ezután pedig az
ADDR+59   205,213,45    CALL 11733      ;FP-TO-A ROM rutint. Ha a
                                         ;következő pont pozíció
                                         ;tinta színű, A értéke 1
                                         ;lesz, egyébként zérus.
ADDR+62   209           POP  DE         ;Visszatöltjük a veremből
ADDR+63   193           POP  BC         ;a kiinduló és ideiglenes
                                         ;koordináta pozíciókat.
ADDR+64   60            INC  A
ADDR+65   61            DEC  A
ADDR+66   254,1         CP   1          ;Ha a következő pozíció
ADDR+68   32,229        JR   NZ,ADDR+43 ;nem tinta színű, ugrás
                                         ;vissza.
ADDR+70   201           RET             ;Vége
```

Egy képpontsort kitöltő szubrutin

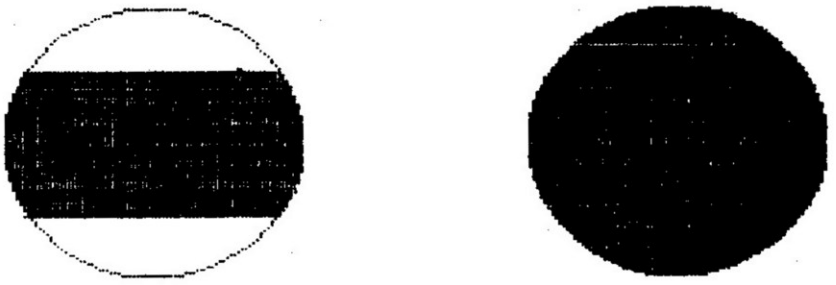
```
ADDR+71   75            LD   C,E        ;C-ben a kiinduló x koor-
ADDR+72   197           PUSH BC         ;dináta. Kimentjük a ki-
ADDR+73   213           PUSH DE         ;induló és ideiglenes ko-
                                         ;ordinátákat.
ADDR+74   205,229,34    CALL 8933       ;Meghívjuk a PLOT ROM ru-
                                         ;tint (HEX-22E5).
ADDR+77   209           POP  DE         ;Visszatöltjük a kiinduló
ADDR+78   193           POP  BC         ;és ideiglenes koordináta
                                         ;pozíciókat.
ADDR+79   12            INC  C          ;Az x koordináta pozíciót
ADDR+80   121           LD   A,C        ;növeljük eggyel és visz-
                                         ;szatöltjük A-ba.
ADDR+81   254,255       CP   255        ;Ha az x koordináta pozí-
ADDR+83   48,16         JR   NC,ADDR+101;ció eléri a 255. értéket
                                         ;(képernyő jobb széle),
                                         ;akkor ugrás történik.
ADDR+85   197           PUSH BC         ;Kimentjük a kiinduló és
ADDR+86   213           PUSH DE         ;ideiglenes koordináta
                                         ;pozíciókat.
ADDR+87   205,206,34    CALL 8910       ;Meghívjuk a POINT és az
ADDR+90   205,213,45    CALL 11733      ;FP-TO-A ROM rutinokat.
ADDR+93   209           POP  DE         ;Visszatöltjük a kiindu-
ADDR+94   193           POP  BC         ;ló és ideiglenes koordi-
                                         ;náta pozíciókat.
ADDR+95   60            INC  A
```

```
ADDR+96   61           DEC  A
ADDR+97   254,1        CP   1           ;Ha a következő pozíció
ADDR+99   32,227       JR   NZ,ADDR+72  ;nem tinta színű, ugrás
                                        ;vissza.
ADDR+101  75           LD   C,E         ;C-ben a kiinduló koordi-
                                        ;náta.
ADDR+102  197          PUSH BC          ;Kimentjük az ideiglenes és
ADDR+103  213          PUSH DE          ;kiinduló koordináta po-
                                        ;zíciókat.
ADDR+104  205,229,34   CALL 8933        ;Meghívjuk a PLOT ROM ru-
                                        ;tint (HEX-22E5).
ADDR+107  209          POP  DE          ;Visszatöltjük a koordi-
ADDR+108  193          POP  BC          ;náta pozíciókat.
ADDR+109  13           DEC  C           ;Az x koordináta pozíciót
ADDR+110  121          LD   A,C         ;csökkentjük eggyel és
                                        ;visszatöltjük A-ba.
ADDR+111  254,1        CP   1           ;Ha az x koordináta pozí-
ADDR+113  56,16        JR   C,ADDR+131  ;ció eléri a 0. értéket
                                        ;(képernyő bal széle),
                                        ;akkor ugrás történik.
ADDR+115  197          PUSH BC          ;Kimentjük a koordináta
ADDR+116  213          PUSH DE          ;pozíciókat.
ADDR+117  205,206,34   CALL 8910        ;Meghívjuk a POINT, majd
ADDR+120  205,213,45   CALL 11733       ;az FP-TO-A ROM rutinokat
                                        ;(HEX-22CE és 2DD5).
ADDR+123  209          POP  DE          ;Visszatöltjük a koordi-
ADDR+124  193          POP  BC          ;náta pozíciókat.
ADDR+125  60           INC  A
ADDR+126  61           DEC  A
ADDR+127  254,1        CP   1           ;Ha a következő pozíció
ADDR+129  32,227       JR   NZ,ADDR+102 ;nem tinta színű, akkor
                                        ;ugrás vissza.
ADDR+131  201          RET              ;Visszatérés a szubrutin-
                                        ;ból.
```

Az itt ismertetett FILL rutin hibája, hogy egyszerűbb alakzatoknál is megfelelő helyen kell a PLOT helyét pozícionálnunk, bonyolultabb alakzatok esetén előfordulhat, hogy a rutint nem is elegendő egyszer futtatni, új pozíciót kell megadnunk és ismét futtatnunk kell.
Nézzünk erre két egyszerűbb példát:

Feltételezzük, hogy a FILL rutin a 40000. címtől lett beolvasva a memóriába. Gépeljük be a következő BASIC programot:

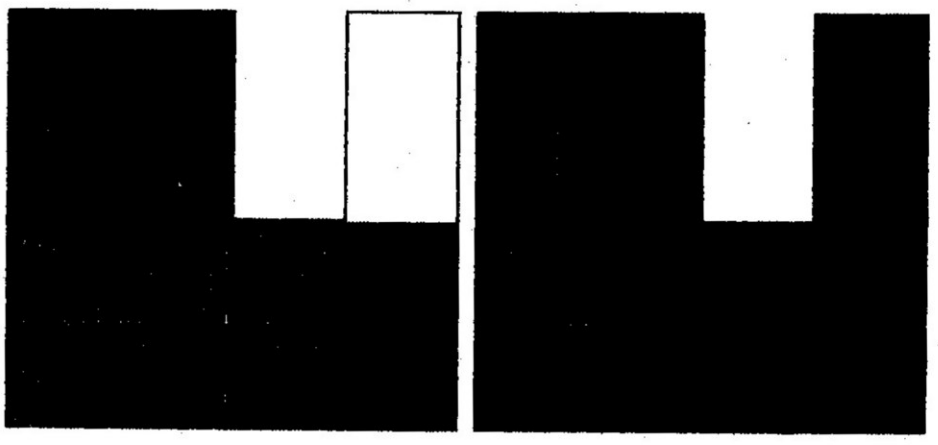
```
10 CIRCLE 64,85,61
20 PLOT 24,85
30 RANDOMIZE USR 40000
40 CIRCLE 191,85,61
50 PLOT 191,85
60 RANDOMIZE USR 40000
70 STOP
```



Ha futtatjuk, azt tapasztaljuk, hogy a bal oldali kört hibásan tölti ki a FILL. Az orvosság: a PLOT x pozíciójának meg kell egyezni a kör (CIRCLE) x pozíciójával, így a második kör esetében hibátlan.

A következő példa esetében már bonyolultabb a helyzet:

```
100 LET a=5: GO SUB 200
110 LET a=130: GO SUB 200
120 PLOT 20,40: RANDOMIZE USR 40000
130 PLOT 140,40: RANDOMIZE USR 40000:
    PLOT 230,150: RANDOMIZE USR 40000
140 STOP
200 PLOT a,20: DRAW 0,140: DRAW 60,0:
    DRAW 0,-70: DRAW 30,0: DRAW 0,70:
    DRAW 30,0: DRAW 0,-140: DRAW -120,0:
    RETURN
```



Futtatáskor a bal oldali ábra hiányosan töltődik ki. Ebben az esetben csak az segít, hogy a megmaradt területet újra pozícionáljuk és kitöltjük, mint ahogy a jobb oldali ábránál tettük. Azok a FILL rutinok, amelyek probléma nélkül kitöltenek bármilyen zárt alakzatot (pl.Beta-Basic FILL), más algoritmus szerint működnek és néhány száz byte-tal hosszabbak is.

3.7 FREE MEMORY rutin

Célja: Kijelzi a még rendelkezésre álló szabad memóriaterület nagyságát.

A legtöbb felhasználói segédprogram speciális utasítással (pl. Beta Basic-ben a MEMORY$) lehetővé teszi a programozó számára, hogy BASIC program írása közben visszajelzést kapjon a még rendelkezésre álló szabad memóriaterület nagyságáról.
Az ismert

PRINT 65535-USR 7962 (ENTER)

formula alkalmazható, de nem igazán kényelmes. Az sem az igazi, ha ezt a formulát az utolsó BASIC sorba írjuk és mindig RUN 9999 utasítással győződünk meg a memória nagyságáról.
Mint tudjuk a Spectrumnál előnyösen alkalmazható a kettes megszakítási rendszer. Használjuk fel ezt a lehetőséget a memória kijelzésére is.
Induljunk ki abból, hogy a szükséges érték a SYMBOL SHIFT és a SPACE billentyű együttes megnyomására jelenjen meg a képernyőn.
Az itt ismertetett rutin a RANDOMIZE USR 65288 utasítással aktivizálható. Ekkor látszólag nem történik semmi, megjelenik az OK üzenet. Ha megnyomjuk egyszerre a SYMBOL SHIFT és a SPACE billentyűket, akkor a képernyő jobb alsó részében megjelenik a keresett érték.
A memória kijelzése után bármelyik billentyű megnyomásával tovább léphetünk.

Mielőtt beolvassuk a rutint, a RAMTOP-ot ne felejtsük el a beállítani a 65278. címre egy megfelelő CLEAR utasítással!

```
65279   15              DEFW 65295      ;Ez a megszakításvezérelt
65280   255                             ;rutin indítócíme.
-------------------------------------------------------------------
65281   62,63           LD   A,63       ;Átkapcsolunk 1-es meg-
65283   237,86          IM   1          ;szakítási módba, kilé-
65285   237,71          LD   I,A        ;pünk a memóriakijelző
65287   201             RET             ;rutinból.
-------------------------------------------------------------------
65288   62,254          LD   A,254      ;Áttérünk 2-es megszakí-
65290   237,71          LD   I,A        ;tási módba, belépünk a
65292   237,94          IM   2          ;memóriakijelző rutinba.
65294   201             RET
-------------------------------------------------------------------
65295   255             RST  56         ;Gondoskodunk a billen-
                                        ;tyűzet leolvasásáról.
65296   243             DI              ;Letiltjuk a megszakítást
65297   197             PUSH BC         ;és kimentjük a regisz-
65298   213             PUSH DE         ;tereket az azonos visz-
65299   229             PUSH HL         ;szatérési állapot biz-
65300   245             PUSH AF         ;tosításához.
65301   62,127          LD   A,127      ;Leolvassuk a jobb alsó
                                        ;billentyű félsort.
65303   219,254         IN   A,(254)    ;A SYMBOL SHIFT és SPACE
```

```
65305   230,31          AND  31          ;együttes megnyomásakor
65307   254,28          CP   28          ;ugrás történik a sza-
65309   204,38,255      CALL Z,65318     ;bad memória tartalmat
                                         ;számító és kijelző szub-
                                         ;rutinra.
65312   241             POP  AF          ;Visszatöltjük a regisz-
65313   225             POP  HL          ;tereket.
65314   209             POP  DE
65315   193             POP  BC
65316   251             EI               ;Engedélyezzük a megsza-
                                         ;kítást.
65317   201             RET              ;Kilépés
-------------------------------------------------------------------
65318   237,75,178,92   LD   BC,(23730)  ;BC-be kerül a RAMTOP cí-
65322   205,43,45       CALL 11563       ;me, és ezt áttöltjük a
                                         ;lebegőpontos verembe egy
                                         ;megfelelő ROM rutin (HEX
                                         ;2D2B)segítségével.
65325   237,75,89,92    LD   BC,(23641)  ;BC-be kerül az aktuális
                                         ;BASIC utasítás címe, és
65329   205,43,45       CALL 11563       ;ezt áttöltjük a lebegő-
                                         ;pontos verembe.
65332   239             RST  40          ;Megnyitjuk a lebegőpon-
                                         ;tos kalkulátort és kü-
65333   3               DEFB 3           ;lönbséget képzünk a le-
                                         ;begőpontos verem felső
                                         ;elemei között.
65334   56              DEFB 56          ;Kilépünk a lebegőpontos
                                         ;kalkulátorból.
65335   1,139,0         LD   BC,139      ;Betöltjük BC-be a RAMTOP
                                         ;alatti stackterület alap
                                         ;memóriaigényét.
65338   205,43,45       CALL 11563       ;Ezt az értéket áttöltjük
                                         ;a lebegőpontos verembe,
65341   239             RST  40          ;és megnyitjuk a lebegő-
                                         ;pontos kalkulátort.
65342   3               DEFB 3           ;Ismét különbséget kép-
65343   56              DEFB 56          ;zünk, majd kilépünk a
                                         ;lebegőpontos kalkulátor-
                                         ;ból.
65344   205,107,13      CALL 3435        ;Meghívjuk a CLS ROM ru-
                                         ;tint (HEX-0D6B) a képer-
                                         ;nyő törléséhez.
65347   62,22           LD   A,22        ;Az "AT" kódja.
65349   215             RST  16
65350   62,1            LD   A,1         ;Mivel a képernyő fel-
65352   215             RST  16          ;ső részét nem nyitottuk
65353   62,27           LD   A,27        ;meg, ez a sor pozíció a
65355   215             RST  16          ;képernyő alsó sorát jel-
                                         ;zi.
65356   62,20           LD   A,20        ;Az INVERZ kiírás kódja.
65358   215             RST  16
```

```
65359   62,1          LD   A,1        ;Bekapcsoljuk az INVERZ
65361   215           RST  16         ;módot.
65362   205,227,45    CALL 11747      ;Kiíratjuk a képernyőre
                                      ;a memóriatartalmat a le-
                                      ;begőpontos veremből, egy
                                      ;megfelelő ROM rutin (HEX
                                      ;2DE3) segítségével.
65365   201           RET             ;Visszatérés a szubrutin-
                                      ;ból.
```

3.8 MEMORY ANALYSE rutin

Célja: Kijelzi a memóriatartalmat adott kezdőcímtől a következő formában:
decimális cím/decimális tartalom/hexa tartalom/ASCII kód

A rutin futtatása előtt az ADDR+09/10 rekeszekben kell megadni a kezdőcím alsó/felső byte-ját.

```
ADDR+00   205,107,13   CALL 3435       ;Meghívjuk a CLS ROM ru-
                                       ;tint (HEX-0D6B) a képer-
                                       ;nyő törléséhez.
ADDR+03   62,2         LD   A,2        ;Megnyitjuk a képer-
ADDR+05   205,1,22     CALL 5633       ;nyőt (ROM HEX-15F7).
ADDR+08   17,X,Y       LD   DE,cím     ;DE-ben adjuk meg a kez-
                                       ;dőcím értékét."X" és "Y"
                                       ;a kezdőcím alsó/felső
                                       ;byte-jai.
-------------------------------------------------------------------
ADDR+11   26           LD   A,(DE)     ;Kimentjük a cím tartal-
ADDR+12   50,176,92    LD   (23728),A  ;mát a 23728.memóriahely-
                                       ;re.
ADDR+15   33,176,92    LD   HL,23728   ;HL-ben megcélozzuk ezt
                                       ;a rekeszt, ez szükséges
                                       ;lesz a hexa számok áta-
                                       ;lakításakor az RLD uta-
                                       ;sításhoz.
ADDR+18   66           LD   B,D        ;A vizsgálandó memória-
ADDR+19   75           LD   C,E        ;címet BC-be is áttöltjük
ADDR+20   229          PUSH HL         ;és a regisztereket ki-
ADDR+21   213          PUSH DE         ;mentjük a verembe.
ADDR+22   205,43,45    CALL 11563      ;BC tartalmát, amely most
                                       ;a vizsgált memóriacím,
                                       ;áttöltjük a lebegőpontos
ADDR+25   205,227,45   CALL 11747      ;verembe, és kiíratjuk a
                                       ;képernyőre (ROM HEX 2D2B
                                       ;ill. HEX-2DE3).
ADDR+28   209          POP  DE         ;A vizsgált memóriacímet
                                       ;visszatöltjük.
ADDR+29   62,32        LD   A,32       ;A cím kiíratása után egy
ADDR+31   215          RST  16         ;SPACE-t hagyunk.
```

```
ADDR+32   26           LD   A,(DE)      ;A vizsgált cím tartal-
                                        ;mát A-ba töltjük, majd
ADDR+33   6,0          LD   B,0         ;átírjuk a BC regiszter-
ADDR+35   79           LD   C,A         ;párba.
ADDR+36   213          PUSH DE          ;A vizsgált címet ismét
                                        ;kimentjük a verembe.
ADDR+37   205,43,45    CALL 11563       ;BC tartalmát, amely most
ADDR+40   205,227,45   CALL 11747       ;a vizsgált memóriacím
                                        ;tartalma áttöltjük a le-
                                        ;begőpontos verembe, majd
                                        ;kiíratjuk a képernyőre.
ADDR+43   209          POP  DE          ;A regiszterpárokat, me-
ADDR+44   225          POP  HL          ;lyek a viżsgált valamint
                                        ;az ideiglenes tárolócí-
                                        ;meket tartalmazzák, a
                                        ;veremből visszatöltjük.
ADDR+45   62,32        LD   A,32        ;A decimális tartalom ki-
ADDR+47   215          RST  16          ;íratása után hagyunk egy
                                        ;SPACE-t.
ADDR+48   6,2          LD   B,2         ;A tartalom hexadecimális
                                        ;formája kétszámjegyű, az
                                        ;átalakítás két lépésben
                                        ;történik.
ADDR+50   175          XOR  A           ;Az A regisztert töröl-
                                        ;jük.
ADDR+51   237,111      RLD              ;Előbb HL tartalmának el-
                                        ;ső, majd második négy
                                        ;bitjét alakítjuk alkal-
ADDR+53   198,48       ADD  A,48        ;mas HEXADECIMÁLIS szám-
                                        ;má.
ADDR+55   254,58       CP   58          ;Ha az A regiszter tar-
                                        ;talma nagyobb, mint a 9-
ADDR+57   48,21        JR   NC,ADDR+80  ;es számjegy ASCII kódja,
                                        ;akkor ugrás történik.
ADDR+59   215          RST  16          ;Kiíratjuk a hexa számje-
                                        ;gyet.
ADDR+60   16,244       DJNZ ADDR+50     ;Ha a második hexa szám-
                                        ;jegy még nem lett kiír-
                                        ;va, ugrás vissza.
ADDR+62   62,32        LD   A,32        ;A hexa tartalom kiíra-
ADDR+64   215          RST  16          ;tása után hagyunk két
ADDR+65   62,32        LD   A,32        ;SPACE-t.
ADDR+67   215          RST  16
ADDR+68   26           LD   A,(DE)      ;Az A-ba töltjük a vizs-
                                        ;gált cím tartalmát.
ADDR+69   254,32       CP   32          ;Ha az ASCII kód értéke
                                        ;kisebb, mint 32, akkor
                                        ;nem nyomtatható karakter
ADDR+71   56,11        JR   C,ADDR+84   ;és ugrás történik.
ADDR+73   215          RST  16          ;Kiíratjuk a karaktert,
                                        ;kulcsszót, és nem nyom-
                                        ;tatható karakter esetén
                                        ;a kérdőjelet.
```

```
ADDR+74   62,13        LD   A,13        ;Sort emelünk és a kiíra-
ADDR+76   215          RST  16          ;tási pozíciót a követke-
                                        ;ző sor elejére állítjuk.
ADDR+77   19           INC  DE          ;A következő címre lé-
                                        ;pünk.
ADDR+78   24,187       JR   ADDR+11     ;A kijelző ciklust újra
                                        ;kezdjük az ADDR+11. cím-
                                        ;től.
-------------------------------------------------------------------
ADDR+80   198,7        ADD  A,7         ;Az A regiszter tartalmát
                                        ;növeljük azért, hogy a
                                        ;10,11,...,15 számok meg-
                                        ;felelő hexa  A,B,...,F
                                        ;számjegyekké alakuljanak.
ADDR+82   24,231       JR   ADDR+59     ;Ugrás vissza az ADDR+59.
                                        ;címre.
-------------------------------------------------------------------
ADDR+84   62,63        LD   A,63        ;Az A regiszterbe töltjük
                                        ;a kérdőjel ASCII kódját.
ADDR+86   24,241       JR   ADDR+71     ;Ugrás vissza az ADDR+71.
                                        ;címre.
```

A rutin működése a ROM bizonyos területének vizsgálatakor:

```
12491 182 B6  ACS
12492 32 20
12493 34 22   "
12494 213 D5  MERGE
12495 229 E5  RESTORE
12496 213 D5  MERGE
12497 205 CD  STEP
12498 127 7F  ©
12499 45 2D   -
12500 235 EB  FOR
12501 227 E3  READ
12502 65 41   A
12503 205 CD  STEP
12504 127 7F  ©
12505 45 2D   -
12506 120 78  x
12507 169 A9  POINT
12508 79 4F   O
12509 225 E1  LLIST
12510 205 CD  STEP
12511 169 A9  POINT
12512 48 30   0

scroll?
```

3.9 ON ERROR GO TO rutin

Célja: Megakadályozza, hogy a program futás közben az "OK...", az "End of file". valamint a "STOP statement..." üzenetek kivételével - hibaüzenettel leálljon.

Hiba esetén a vezérlés átadódik a program egy meghatározott sorszámú sorára, és a futás onnan folytatódik tovább.(Mindez magába foglalja a BREAK tiltását is.)
Annak a sornak a sorszámát ahova hiba esetén a vezérlést átadjuk, alsó/felső byte formában az ADDR+40/41 címeken kell megadni.
Először tekintsük át a rutint:

```
ADDR+00   205,9,12     CALL 3081       ;Ne keressük meg az ezen
                                       ;a címen található ROM
                                       ;rutint, ugyanis a memó-
                                       ;riában ezen a címen 201
                                       ;(RET) kódot találunk.
                                       ;Nagyon érdekes trükk a
                                       ;gépi kódú programozás-
                                       ;ban, hogy a rutin bárhol
                                       ;van, megkeresi saját ma-
                                       ;gát. A szubrutinból való
                                       ;közvetlen visszatéréskor
                                       ;az SP által kijelölt ve-
                                       ;rem tartalma ADDR+03, de
                                       ;amikor ennek alapján a
                                       ;vezérlés erre a címre
                                       ;adódik át, a processzor
                                       ;automatikusan növeli az
                                       ;SP mutató értékét 2 re-
                                       ;kesszel. Fontos! Ez nem
                                       ;jelenti az előző értékek
ADDR+03   59           DEC  SP         ;törlését, így ha SP-t
ADDR+04   59           DEC  SP         ;ezzel a 2 rekesszel
                                       ;visszaléptetjük, SP is-
                                       ;mét az ADDR+03-at célo-
ADDR+05   225          POP  HL         ;za meg, és ez egyszerűen
                                       ;kiolvasható egy POP uta-
                                       ;sítás segítségével, pl.
                                       ;a HL regiszterpárba.
ADDR+06   1,15,0       LD   BC,15      ;Ha hozzáadunk 15-öt, ak-
ADDR+09   9            ADD  HL,BC      ;kor HL-ben kialakul az
                                       ;ADDR+18. cím ( bárhol is
                                       ;van a rutin a memóriá-
                                       ;ban), amit visszatérési
                                       ;címként szeretnénk fel-
                                       ;használni.
ADDR+10   235          EX   DE,HL      ;Ezt a címet áttöltjük a
                                       ;DE regiszterpárba, majd
ADDR+11   42,61,92     LD   HL,(23613) ;DE-t pedig az ERR SP
ADDR+14   115          LD   (HL),E     ;rendszerváltozó által
```

```
ADDR+15   35              INC  HL          ;meghatározott címre, ami
ADDR+16   114             LD   (HL),D      ;hiba esetén a mindenkori
                                           ;visszatérési címet tar-
                                           ;talmazza, vagyis ha hiba
                                           ;van, a vezérlés az ADDR+
                                           ;18.címre adódik át.
ADDR+17   201             RET              ;Vége
-------------------------------------------------------------------
ADDR+18   59              DEC  SP          ;SP-t ismét visszaállít-
ADDR+19   59              DEC  SP          ;juk.
ADDR+20   58,58,92        LD   A,(23610)   ;A-ba töltjük az ERR NR
                                           ;rendszerváltozó tartal-
                                           ;mát, amely a mindenkori
                                           ;hibakódnál eggyel keve-
                                           ;sebb érték.
ADDR+23   254,255         CP   255         ;Ha ez az érték 255, va-
                                           ;gyis a hibakód 0 (OK ü-
                                           ;zenet), akkor ugrás tör-
ADDR+25   40,29           JR   Z,ADDR+56   ;ténik.
ADDR+27   254,7           CP   7           ;Ha ez az érték 7, vagyis
                                           ;a hibakód 8(End of file)
ADDR+29   40,25           JR   Z,ADDR+56   ;akkor ugrás történik.
ADDR+31   254,8           CP   8           ;Ha az érték 8, vagyis a
                                           ;hibakód 9 (STOP state-
ADDR+33   40,21           JR   Z,ADDR+56   ;ment), akkor ugrás tör-
                                           ;ténik.
ADDR+35   253,54,0,255    LD   (IY+0),255  ;Alaphelyzetben az IY re-
                                           ;giszterpár értéke:23610.
                                           ;Most így erre a címre,
                                           ;vagyis az ERR NR rend-
                                           ;szerváltozóba töltünk
                                           ;255-öt, ami a 0-ás (OK)
                                           ;üzenetkódnak felel meg.
ADDR+39   33,X,Y          LD   HL,sorszám  ;HL-be töltjük annak a
                                           ;sornak a sorszámát,
                                           ;ahova hiba esetén az ug-
                                           ;rás történik.
ADDR+42   34,66,92        LD   (23618),HL  ;Ezt a sorszámot áttölt-
                                           ;jük a NEWPPC rendszer-
                                           ;változóba, amely az ug-
                                           ;rás sorszámának tároló-
                                           ;sára szolgál.
ADDR+45   175             XOR  A           ;Az A regisztert töröljük
ADDR+46   50,68,92        LD   (23620),A   ;és így zérust írunk az
                                           ;NSPPC rendszerváltozóba,
                                           ;amely ugrás esetén a so-
                                           ;ron belüli utasítás szá-
                                           ;mát tárolja.
ADDR+49   253,203,1,254   SET  7,(IY+1)    ;A 23611.(FLAGS)rendszer-
                                           ;változó 7.bitjét 1-re
                                           ;állítjuk, ez a "Parancs-
                                           ;sor végrehajtása" jelzés.
```

```
ADDR+53   195,125,27   JP   7037     ;Átugrunk a ROM STM-T-RET
                                     ;rutinjába(HEX-1B7D),vég-
                                     ;rehajtás alatt is mindig
                                     ;ide adódik át a vezérlés
                                     ;ha nem nyomtunk BREAK-et
ADDR+56   51           INC  SP       ;SP-t a belépési helyzet-
ADDR+57   51           INC  SP       ;re állítjuk vissza.
ADDR+58   195,3,19     JP   4867     ;Átugrunk a ROM MAIN-4
                                     ;rutinjába (HEX-1303). Ez
                                     ;a szerkesztési műveletek
                                     ;egyik belépési pontja,
                                     ;vagyis kilépünk az
                                     ;"ON ERROR GO TO" rutin-
                                     ;ból.
```

A rutin alkalmazására nézzünk egy példát:
Helyezzük el a rutint pl. a 65000. címtől, tehát a RAMTOP-ot állítsuk be ennek megfelelően (CLEAR 64999).
Gépeljük be a következő szemléltető BASIC programot:

```
   10 RANDOMIZE USR 65000
   20 CIRCLE 120,70,70
   30 INPUT a
   40 FOR i=1 TO 600: PRINT "a";: NEXT i
   50 STOP
 9999 CLS: PRINT AT 0,0: "Ezt az ON ERROR
      GO TO rutin okozza!": PAUSE 0: CLS:
      GO TO 20
```

Mielőtt kipróbálnánk, a 9999-es sorszám alsó/felső byte-ját el kell helyezni a rutinban, a megfelelő helyen. A 9999 alsó byte-ja 15, a felső pedig 39. Ebből adódóan gépeljük be:

```
POKE 65040,15: POKE 65041,39 (ENTER)
```

Most már futtathatjuk a programot: RUN (ENTER)

Az INPUT-ra válaszoljunk a STOP kulcsszóval (SYMBOL SHIFT+A). ENTER megnyomása után nem a várt "H STOP in INPUT..." hibaüzenet fog megjelenni, hanem az amit a 9999. sorban megadtunk, és a program futása onnan folytatódik tovább. Nyomjunk meg bármit, és az újabb INPUT-ra adjunk meg egy helyes értéket, pl. egy számot. A képernyő kezd megtelni "a" betűkkel. Ne várjuk meg a végét, próbáljuk BREAK-kel megszakítani a futást. Ismét működik a rutin. Ha újabb próbálkozás után megvárjuk, hogy az "a" betűkkel megteljen a képernyő, a futás rendben megáll "STOP statement..." üzenettel.

Természetesen, ha valakinek csak az a célja, hogy mások számára ne engedjen bepillantást saját BASIC programjába, úgy a program futását nem szabad hagyni leállni, a végére be kell iktatni egy döntést, amely megkérdezi, hogy újra futtatjuk-e a programot, vagy nem? Ez utóbbi esetben közvetlenül ugorhatunk pl. egy NEW utasításra.

Programunkat mindenképpen autostartos formában kell magnetofonra rögzíteni, hogy az első utasítássor végrehajtásával azonnal aktivizáljuk a rutint.

Ekkor még mindig fennáll annak a veszélye, hogy fáradozásunk hiábavaló volt, hisz a MERGE"" utasítás segítségével az autostart elkerülhető, így a program ismét listázhatóvá válik.

Oldjuk meg ezt a problémát is, az ON ERROR GO TO rutin ugrássorszámát változtassuk meg 9999-ről 9998-ra (a rutinban is módosítsuk ennek megfelelően 14/39-re), és a 9999. sorszámon helyezzünk el egy egyedülálló és önmagában jelentéktelen REM utasítást:

9999 REM (ENTER)

Most meg kell keresnünk ennek a sorszámnak a memóriabeli címét. Ez hosszabb programok esetén PEEK segítségével hosszú ideig tarthat, legegyszerűbb egy igen rövid gépi kódú rutint alkalmazni:

```
ADDR+00   42,73,92      LD   HL,(23625) ;HL-be töltjük a kurzor-
                                        ;ral jelzett aktuális sor
                                        ;sorszámát.
ADDR+03   205,110,25    CALL 6510       ;Meghívjuk a sorcímkereső
                                        ;LINE ADDR ROM rutint(HEX
                                        ;196E).A cím a HL-be ke-
                                        ;rül.
ADDR+06   68            LD   B,H        ;HL-t áttöltjük BC-be,
ADDR+07   77            LD   C,L        ;mert a PRINT USR utasí-
                                        ;tás a BC tartalmát írja
                                        ;ki a képernyőre.
ADDR+08   201           RET             ;Vége
```

Ha a kurzort a 9999. sorra állítjuk, és meghívjuk ezt a rövid rutint (PRINT USR ADDR+00), akkor megkapjuk a keresett címet. A MERGE"" 'megvadításához' a sor hosszát jelző byte-okat zérusra kell átírnunk, vagyis, ha a 9999. sor megkapott címe: nn, akkor POKE nn+2,0: POKE nn+3,0 (ENTER).

Most mentsük ki programunkat autostartos formában, és próbáljuk meg MERGE"" segítségével visszatölteni. Láthatjuk a várt eredmény nem marad el.

Ha arra gondolunk, hogy valaki még így is feltöri a lakatot és megnézheti BASIC listánkat, nincs más hátra írjuk meg programunkat gépi kódban.

3.10 PATTERN rutin

Célja: Un. "Sakktábla-hálót" rajzol a képernyőre, mely segítségével könnyebb a tájékozódás pl. PRINT vagy rajzoló műveletek során.

```
ADDR+00   33,0,88       LD   HL,22528   ;HL-be töltjük az ATTR
                                        ;terület kezdőcímét.
ADDR+03   6,12          LD   B,12       ;12*2 sort töltünk fel.
ADDR+05   197           PUSH BC         ;BC-t kimentjük.
ADDR+06   6,16          LD   B,16       ;Páratlan sor feltöltése
                                        ;16*2 lépésben.
```

```
ADDR+08   126          LD   A,(HL)      ;A képernyő adott pozíci-
ADDR+09   203,247      SET  6,A         ;óját BRIGHT 1 fényesség-
ADDR+11   119          LD   (HL),A      ;re állítjuk.
ADDR+12   35           INC  HL          ;Tovább lépünk a követke-
                                        ;ző címre.
ADDR+13   126          LD   A,(HL)      ;A képernyő adott pozíci-
ADDR+14   203,183      RES  6,A         ;óját BRIGHT 0 fényesség-
ADDR+16   119          LD   (HL),A      ;re állítjuk.
ADDR+17   35           INC  HL          ;Tovább lépünk a követke-
                                        ;ző címre.
ADDR+18   16,244       DJNZ ADDR+08     ;Ugrás vissza, ha a sor
                                        ;még nincs kész.
ADDR+20   6,16         LD   B,16        ;Páros sor feltöltése 16*
                                        ;2 lépésben.
ADDR+22   126          LD   A,(HL)      ;A képernyő adott pozíci-
ADDR+23   203,183      RES  6,A         ;óját BRIGHT 0 fényesség-
ADDR+25   119          LD   (HL),A      ;re állítjuk.
ADDR+26   35           INC  HL          ;Tovább lépünk a követke-
                                        ;ző címre.
ADDR+27   126          LD   A,(HL)      ;A képernyő adott pozíci-
ADDR+28   203,247      SET  6,A         ;óját BRIGHT 1 fényesség-
ADDR+30   119          LD   (HL),A      ;re állítjuk.
ADDR+31   35           INC  HL          ;Tovább lépünk a követke-
                                        ;ző címre.
ADDR+32   16,244       DJNZ ADDR+22     ;Ugrás vissza, ha a sor
                                        ;még nincs kész.
ADDR+34   193          POP  BC          ;BC-t visszatöltjük a ve-
                                        ;remből.
ADDR+35   16,224       DJNZ ADDR+05     ;Ugrás vissza, ha a 24
                                        ;sor még nincs kész,
ADDR+37   201          RET              ;máskülönben vége.
```

3.11 REMKILL rutin

Célja:A BASIC programban elhelyezett REM megjegyzéseket törli, ezáltal memóriamegtakarítást érhetünk el.

```
ADDR+00   42,83,92     LD   HL,(23635)  ;HL-be töltjük a BASIC
                                        ;terület kezdőcímét,
ADDR+03   126          LD   A,(HL)      ;A-ba pedig annak tartal-
ADDR+04   230,192      AND  192         ;mát. Ha ennek az érték-
                                        ;nek a 6. vagy 7. bitje 1
                                        ;értékű, akkor nincs BA-
                                        ;SIC és meghívjuk a ROM
                                        ;"OK" üzenet rutinját a
ADDR+06   194,176,27   JP   NZ,7088    ;7088.(HEX-1BB0)címen.
ADDR+09   50,129,92    LD   (23681),A   ;A 23681.címet A-val, va-
                                        ;gyis most zérussal tölt-
                                        ;jük fel.
ADDR+12   35           INC  HL          ;HL-t tovább léptetjük a
ADDR+13   35           INC  HL          ;sor hosszának meghatáro-
                                        ;zásához.
```

```
ADDR+14   78           LD   C,(HL)      ;A sor hosszát a BC re-
ADDR+15   35           INC  HL          ;giszterpárba töltjük.
ADDR+16   70           LD   B,(HL)
ADDR+17   35           INC  HL          ;HL most a sor első ér-
                                        ;telmezhető karakterének
                                        ;címét tartalmazza.
ADDR+18   34,93,92     LD   (23645),HL  ;Az adott karakter kiér-
ADDR+21   34,11,92     LD   (23563),HL  ;tékeléséhez a CH ADD és
                                        ;DEF ADD rendszerválto-
                                        ;zókba töltjük HL-t.
ADDR+24   9            ADD  HL,BC       ;HL most a következő sor
                                        ;első címére mutat.
ADDR+25   34,176,92    LD   (23728),HL  ;Ezt a címet kitöltjük a
                                        ;23728/29. memóriahelyek-
                                        ;re.
ADDR+28   223          RST  24          ;Meghívjuk a ROM-beli ka-
                                        ;raktervizsgáló rutint. A
                                        ;rutin a CH ADD rendszer-
                                        ;változóban található cí-
                                        ;men levő karaktert nézi
                                        ;meg. A vizsgálat végén A
                                        ;regiszter fogja tartal-
                                        ;mazni a karakter ASCII
                                        ;kódját.
ADDR+29   254,58       CP   58          ;Ha a karakter egy ket-
ADDR+31   40,51        JR   Z,ADDR+84   ;tőspont, akkor ugrás
                                        ;történik az ADDR+84.cím-
                                        ;re.
ADDR+33   254,14       CP   14          ;Ha a karakter ASCII kód-
ADDR+35   56,47        JR   C,ADDR+84   ;ja kisebb, mint 14 akkor
                                        ;ugrás történik az ADDR+
                                        ;84. címre.
ADDR+37   254,234      CP   234         ;Ha a karakter nem a REM
ADDR+39   32,34        JR   NZ,ADDR+75  ;kulcsszó, akkor ugrás
                                        ;az ADDR+75. címre.
ADDR+41   58,129,92    LD   A,(23681)   ;Megvizsgáljuk, hogy a
ADDR+44   254,1        CP   1           ;sor elején vagyunk-e? (A
                                        ;23681.cím tartalma ebben
                                        ;a rutinban mindig zérus
                                        ;a sor elején.)
ADDR+46   56,47        JR   C,ADDR+95   ;Ha igen, akkor ugrás az
                                        ;ADDR+95. címre.
ADDR+48   43           DEC  HL          ;A REM a soron belül lett
ADDR+49   54,13        LD   (HL),13     ;elhelyezve, így vissza-
                                        ;lépünk a kettőspont ka-
                                        ;rakterre és a helyére
                                        ;ENTER-t (13.-as kód) he-
                                        ;lyezünk el.
ADDR+51   35           INC  HL          ;Visszalépünk a REM uta-
ADDR+52   229          PUSH HL          ;sításra és kimentjük az
                                        ;ehhez tartozó címet.
```

```
ADDR+53   237,91,11,92 LD   DE,(23563) ;DE-be visszatöltjük az
                                       ;adott (REM-et tartalma-
                                       ;zó) sor első értelmezhe-
                                       ;tő karakterének címét.
ADDR+57   167          AND  A          ;HL-ben előállítjuk a REM
ADDR+58   237,82       SBC  HL,DE      ;nélküli új sor hosszát.
ADDR+60   235          EX   DE,HL      ;A csere után HL tartal-
                                       ;mazza a sor első karak-
                                       ;terének címét, DE pe-
                                       ;dig az új sor hosszt.
ADDR+61   43           DEC  HL         ;Kétszeri visszalépéssel
ADDR+62   114          LD   (HL),D     ;felülírjuk a régi hossz
ADDR+63   43           DEC  HL         ;byte értékeit az új ér-
ADDR+64   115          LD   (HL),E     ;tékekkel.
ADDR+65   225          POP  HL         ;Visszatöltjük a soron
                                       ;belüli REM kulcsszó cí-
                                       ;mét, és ezt
ADDR+66   235          EX   DE,HL      ;áttöltjük DE-be.
ADDR+67   42,176,92    LD   HL,(23728) ;HL-be betöltjük a követ-
                                       ;kező BASIC sor első cí-
                                       ;mét.
ADDR+70   205,229,25   CALL 6629       ;Meghívjuk a helymegszün-
                                       ;tető "RECLAIM" ROM ru-
                                       ;tint(HEX-19E5). A rutin
                                       ;meghívásakor DE-nek tar-
                                       ;talmaznia kell az első
                                       ;megszüntetendő hely cí-
                                       ;mét, HL-nek pedig az el-
                                       ;ső megmaradó hely címét.
ADDR+73   24,184       JR   ADDR+03    ;Ugrás vissza.
-------------------------------------------------------------------
ADDR+75   14,0         LD   C,0        ;Belépünk a ROM-beli EACH
ADDR+77   22,1         LD   D,1        ;rutinba(HEX-1998), amely
ADDR+79   30,0         LD   E,0        ;a mindenkori BASIC uta-
ADDR+81   205,152,25   CALL 6552       ;sítás végét (utolsó ka-
                                       ;rakterét) keresi meg.
                                       ;Visszatéréskor HL-ben az
                                       ;utasítás utolsó karakte-
                                       ;rének címe lesz.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ADDR+84   253,52,71    INC  (IY+71)    ;Az IY regiszterpár alap-
                                       ;értéke 23610. Most így a
                                       ;23681. cím tartalmát nö-
                                       ;veljük eggyel.
ADDR+87   35           INC  HL         ;Tovább lépünk egy cím-
ADDR+88   34,93,92     LD   (23645),HL ;mel, és a CH ADD rend-
                                       ;szerváltozóba az új cí-
                                       ;met írjuk.
ADDR+91   48,191       JR   NC,ADDR+28 ;Ugrás vissza, ha a BASIC
                                       ;sornak még nincs vége.
ADDR+93   24,164       JR   ADDR+03    ;A BASIC sor végén ugrás
                                       ;vissza a vizsgálat kez-
                                       ;detéhez.
```

```
ADDR+95   43           DEC  HL          ;A rutin akkor ugrik ide,
ADDR+96   43           DEC  HL          ;ha a REM a sor elején
ADDR+97   43           DEC  HL          ;van. Ilyenkor visszalép-
ADDR+98   43           DEC  HL          ;tetjük a cím értékét a
                                        ;sor első címére, és el-
ADDR+99   24,221       JR   ADDR+66     ;ugrunk a helymegszüntető
                                        ;rutinra.
```

Ha a rutint beolvastuk a memóriába, gépeljük be próbaként a következő kis BASIC programot:

```
10 FOR i=1 TO 10: REM soron belul
20 CLS
30 REM sor elejen
```

Ezután futtassuk a gépi kódú rutint: RANDOMIZE USR ADDR+00 és ha kilistázzuk a programunkat, már a következő formát mutatja:

```
10 FOR i=1 TO 10
20 CLS
```

Természetesen hosszabb programok esetén érezhető igazán a rutin előnyös szolgáltatása.

3.12 RENUMBER rutin

Célja:BASIC programsorok sorszámainak átsorszámozása.

Az itt ismertetett rutin igen egyszerű felépítésű, nem képes a GO TO ill. GO SUB utasítások paramétereinek átsorszámozására.

```
ADDR+00   237,107,83,92 LD   HL,(23635) ;HL-be töltjük a BASIC
                                        ;terület kezdőcímét.
ADDR+04   1,10,0        LD   BC,10      ;BC-be töltjük a lépésköz
                                        ;értékét.
ADDR+07   17,10,0       LD   DE,10      ;DE-be töltjük az átsor-
                                        ;számozás induló sorszá-
                                        ;mát.
ADDR+10   213           PUSH DE         ;DE-t és HL-t kimentjük a
ADDR+11   229           PUSH HL         ;verembe.
ADDR+12   237,91,75,92  LD   DE,(23627) ;DE-be töltjük a változók
ADDR+16   175           XOR  A          ;címét, majd töröljük az
ADDR+17   237,82        SBC  HL,DE      ;átviteli jelzőbitet és
                                        ;megnézzük, hogy van-e
ADDR+19   225           POP  HL         ;BASIC program ? Vissza-
ADDR+20   209           POP  DE         ;töltjük a regisztereket,
ADDR+21   200           RET  Z          ;befejezzük, ha nincs BA-
                                        ;SIC.
ADDR+22   197           PUSH BC         ;BC-t kimentjük a verembe
ADDR+23   114           LD   (HL),D     ;Beírjuk a BASIC terület
ADDR+24   35            INC  HL         ;első címére a kezdeti
ADDR+25   115           LD   (HL),E     ;sorszámot.
ADDR+26   35            INC  HL         ;Az adott BASIC sor hosz-
ADDR+27   78            LD   C,(HL)     ;szát visszaírjuk BC-be.
ADDR+28   35            INC  HL
```

```
ADDR+29   70          LD   B,(HL)
ADDR+30   35          INC  HL          ;Tovább lépünk és az új
ADDR+31   9           ADD  HL,BC       ;címhez a hosszt hozzáad-
ADDR+32   235         EX   DE,HL       ;juk, majd felcseréljük
                                       ;az aktuális sorszámmal.
ADDR+33   193         POP  BC          ;A veremből kivesszük a
                                       ;lépésközt és hozzáadjuk
ADDR+34   9           ADD  HL,BC       ;az aktuális sorszámhoz.
ADDR+35   235         EX   DE,HL       ;Visszacseréljük a sor-
                                       ;szám és a mindenkori
ADDR+36   24,228      JR   ADDR+10     ;kezdőcím értékét, majd
                                       ;visszaugrunk.
```

A rutin meghívása előtt a lépésközt az ADDR+05/06 címeken, míg az átsorszámozás induló sorszámát az ADDR+08/09 címeken kell elhelyezni. Jelenleg a 10-es sorszámtól és 10-es emelkedésű átsorszámozást hajt végre futtatás esetén. A rutin a teljes BASIC programot átsorszámozza. Az eredeti program első sora kapja az induló sorszám értékét, a többi sorszám pedig a lépésköz szerint alakul.

4.fejezet

SCROLL-LEXIKON

A 'scroll' szótári értelemben 'kézirattekercs', de a számítástechnikában teljesen más jelentéstartalma van.

A 'scroll' angol szó a középkorból származik. Amikor a szónokok a tér közepén, vagy a kastélyban elkezdték hosszú mondókájukat, kezükben nem papírlapokat tartottak, hanem papírtekercseket, és ezt görgették tovább annak megfelelően, ahogy a szöveggel haladtak. Még ma sem furcsa dolog, ha egy államfő, vagy nevesebb politikus több órán át tartó beszámolóját hasonló módon tartja meg, igaz azóta a technika sokat változott, de a papírtekercs csévélése formailag a régi maradt. Ezt a folyamatot görgetésnek is hívjuk, s a számítástechnika innen vette a képernyő tartalom mozgatására ezt az elnevezést.

Ebben a fejezetben megpróbálunk egy alapos áttekintést nyújtani a ZX-Spectrum számítógéppel megoldható 'scroll' lehetőségekre. Az eddigi irodalmak csak érintőlegesen tárgyalták ezt a területet. Hol egy bal irányú 'finom-scroll'-t ismertünk meg, hol esetleg egy másik tipust pl. egy magazin hasábján.

Az itt közölt lehetőségek egymással ötvözhetők, továbbfejleszthetők, a programozó beépítheti saját BASIC vagy gépi kódú programjaiba is ezeket a rutinokat.

4.1 Általános csoportosítás

A 'scroll' különböző szempontok szerint csoportosítható, most először tekintsük át ezeket:

1.Finomsága szerint
- a/Kisfelbontású, karakterenként (LORES)
- b/Nagyfelbontású, képpontonként (HIRES)

2.Iránya szerint
- a/Főirányú
  - -felfelé (UP)
  - -lefelé (DOWN)
  - -balra (LEFT)
  - -jobbra (RIGHT)
- b/Átlós irányú
  - -balra/fel (NW)
  - -jobbra/fel (NE)
  - -balra/le (SW)
  - -jobbra/le (SE)

3.Területe szerint
- a/Teljes képernyő
- b/Felső 1/3
- c/Középső 1/3
- d/Alsó 1/3
- e/Felső 2/3
- f/Alsó 2/3
- g/Kijelölt képernyőterület (BOX)

4.Görgetett információ tartalma szerint
a/Képernyő file görgetése
b/Színmemória (ATTRIBUTUM) görgetése

5.Görgetés jellege szerint
a/A képernyő szélén túllépő információ elvész (OFF-LINE)
b/A képernyő szélén túllépő információ belép a képernyő másik szélén(WRAP-AROUND)
c/Az előzőhöz hasonló, de a visszatérő bitek negálódnak (INVERZ)

6.Görgetés technikája szerint
a/Karakterről-karakterre folyamatosan az egész képernyőre (LINE)
b/Fodrozódó (RIPPLE) hatás. A'WRAP-AROUND' jelleg minden karakterre külön-külön működik.
c/Szétfoszladozó (SHUTTER) hatás. Az 'OFF-LINE' jelleg minden karakterre külön működik, s így a képernyő nyolc mozzanat után kitisztul.

Az itt ismertetett csoportokból egymással igen sok variáció építhető össze. Mi most csak a legfontosabb alap-scroll rutinok gépi kódú programjait ismertetjük, ebből adódóan nem követjük pontról pontra az itt látható általános csoportosítást sem.

4.2 Képernyő-file scroll rutinok

-HIRES scroll felfelé

```
ADDR+00   33,0,64     LD   HL,16384   ;HL-ben a képernyő első
ADDR+03   6,3         LD   B,3        ;címe, és ciklusonként
ADDR+05   197         PUSH BC         ;3 db. képernyőharmad.
ADDR+06   6,8         LD   B,8        ;Harmadonként 8 db.karak-
ADDR+08   197         PUSH BC         ;tersor.
ADDR+09   6,7         LD   B,7        ;Minden sorban 7 db. kép-
ADDR+11   197         PUSH BC         ;pont mozdulat.
ADDR+12   6,32        LD   B,32       ;Soronként 32 karakter.
ADDR+14   229         PUSH HL         ;A kezdőcímet eltesszük a
                                      ;verembe.
ADDR+15   17,0,1      LD   DE,256     ;Különbség két egymás a-
                                      ;latt levő byte között.
ADDR+18   0           NOP
ADDR+19   0           NOP
ADDR+20   25          ADD  HL,DE      ;Beállítjuk az aktuális
ADDR+21   209         POP  DE         ;képernyőcímet.
ADDR+22   126         LD   A,(HL)     ;Átmásoljuk az egymás a-
ADDR+23   18          LD   (DE),A     ;latti byte-okat.
ADDR+24   235         EX   DE,HL      ;Eredeti állapot.
ADDR+25   35          INC  HL         ;Növeljük a címet.
ADDR+26   16,242      DJNZ ADDR+14    ;Ciklus 32-szer.
```

```
ADDR+28   17,224,0     LD   DE,224      ;Különbség két egymás a-
ADDR+31   0            NOP              ;latti képpontsor szélső
ADDR+32   0            NOP              ;címei között.
ADDR+33   25           ADD  HL,DE       ;Tovább lépünk a követke-
ADDR+34   193          POP  BC          ;ző képpontsorra.
ADDR+35   16,230       DJNZ ADDR+11     ;Ciklus 7-szer.
ADDR+37   193          POP  BC
ADDR+38   120          LD   A,B
ADDR+39   254,1        CP   1           ;Ha a harmadban az utolsó
ADDR+41   40,26        JR   Z,ADDR+69   ;sor volt ugrás, ha nem
ADDR+43   197          PUSH BC          ;akkor a két sor közötti
ADDR+44   6,32         LD   B,32        ;átvitel történik.
ADDR+46   17,224,6     LD   DE,1760     ;Különbség két karakter-
ADDR+49   229          PUSH HL          ;sor között.
ADDR+50   167          AND  A           ;Az átviteli jelzőbitet
ADDR+51   237,82       SBC  HL,DE       ;töröljük és különbséget
ADDR+53   209          POP  DE          ;képzünk.
ADDR+54   126          LD   A,(HL)      ;Byte-ok átvitele két ka-
ADDR+55   18           LD   (DE),A      ;raktersor között.
ADDR+56   235          EX   DE,HL       ;Eredeti állapot.
ADDR+57   35           INC  HL          ;Növeljük a címet.
ADDR+58   16,242       DJNZ ADDR+46     ;Ciklus 32-szer.
ADDR+60   17,0,7       LD   DE,1792
ADDR+63   167          AND  A           ;Az átviteli jelzőbitet
ADDR+64   237,82       SBC  HL,DE       ;töröljük és HL-t beál-
ADDR+66   193          POP  BC          ;lítjuk a következő ka-
                                        ;raktersorra.
ADDR+67   16,195       DJNZ ADDR+08     ;Ciklus 8-szor.
ADDR+69   193          POP  BC
ADDR+70   120          LD   A,B
ADDR+71   254,1        CP   1           ;Az utolsó harmad volt?
ADDR+73   40,20        JR   Z,ADDR+95   ;Ha igen, ugrás az ADDR+
                                        ;95. címre.
ADDR+75   197          PUSH BC
ADDR+76   6,32         LD   B,32
ADDR+78   17,32,0      LD   DE,32       ;A következő harmad kez-
ADDR+81   229          PUSH HL          ;dőcíme a HL pillanatnyi
ADDR+82   0            NOP              ;értékétől 32 byte távol-
ADDR+83   0            NOP              ;ságra van.
ADDR+84   25           ADD  HL,DE       ;Beállítjuk a címet.
ADDR+85   209          POP  DE
ADDR+86   126          LD   A,(HL)      ;Két harmad közötti kép-
ADDR+87   18           LD   (DE),A      ;pontsor átvitele.
ADDR+88   235          EX   DE,HL
ADDR+89   35           INC  HL          ;Növeljük a címet.
ADDR+90   16,242       DJNZ ADDR+78     ;Ciklus 32-szer.
ADDR+92   193          POP  BC
ADDR+93   16,166       DJNZ ADDR+05     ;Ciklus 3-szor.
ADDR+95   6,32         LD   B,32        ;32 karakter hosszon itt
ADDR+97   54,0         LD   (HL),0      ;történik a felesleges
ADDR+99   35           INC  HL          ;képpontsor letörlése.
ADDR+100  16,251       DJNZ ADDR+97     ;Ciklus 32-szer.
ADDR+102  201          RET              ;Vége
```

Bizonyára feltűnik, hogy a listában itt-ott üres NOP utasítással találkozunk. A program működése szempontjából ennek nincs jelentősége, de így könnyebb lesz az ettől némileg különböző 'HIRES scroll lefelé' rutin ismertetése. A felhasználó természetesen az üres utasításokat kiveheti a programból, ezáltal byte-okat takaríthatunk meg.

-HIRES scroll lefelé

Tekintettel arra, hogy a két rutin között igen sok az azonos elem, így csak azokat a sorokat közöljük itt, melyeket meg kell változtatni az irány megfordításához.

```
ADDR+00   33,255,87    LD   HL,22527    ;HL-ben a képernyő utolsó
                                        ;címe.

ADDR+18   167          AND  A           ;Az átviteli jelzőbitet
ADDR+19   237,82       SBC  HL,DE       ;töröljük és beállítjuk a
                                        ;képernyőcímet.

ADDR+25   43           DEC  HL          ;Csökkentjük a címet.

ADDR+31   167          AND  A           ;Az átviteli jelzőbitet
ADDR+32   237,82       SBC  HL,DE       ;töröljük és beállítjuk a
                                        ;képernyőcímet.

ADDR+50   0            NOP
ADDR+51   0            NOP
ADDR+52   25           ADD  HL,DE       ;Beállítjuk a képernyőcí-
                                        ;met.

ADDR+57   43           DEC  HL          ;Csökkentjük a címet.

ADDR+63   0            NOP
ADDR+64   0            NOP
ADDR+65   25           ADD  HL,DE       ;Beállítjuk a képernyőcí-
                                        ;met.

ADDR+82   167          AND  A           ;Az átviteli jelzőbitet
ADDR+83   237,82       SBC  HL,DE       ;töröljük és beállítjuk a
                                        ;képernyőcímet.

ADDR+89   43           DEC  HL          ;Csökkentjük a címet.

ADDR+99   43           DEC  HL          ;Csökkentjük a címet.
```

-HIRES scroll balra

```
ADDR+00   33,255,87    LD   HL,22527    ;HL-be töltjük a képernyő
                                        ;utolsó címét.
ADDR+03   6,192        LD   B,192       ;A mozgatandó képpontso-
ADDR+05   197          PUSH BC          ;rok száma.
ADDR+06   229          PUSH HL          ;HL-t áttöltjük IX-be, és
```

```
ADDR+07   221,225       POP  IX          ;a képernyő másik szélén
ADDR+09   221,126,225   LD   A,(IX-31)   ;levő bitet az átviteli
ADDR+12   7             RLCA             ;jelzőbitbe töltjük.
ADDR+13   55            SCF              ;Az átviteli jelzőbitet
ADDR+14   63            CCF              ;1-re állítjuk és ennek
                                         ;inverzét képezzük.
ADDR+15   6,32          LD   B,32        ;1 sor = 32 karakter.
ADDR+17   203,22        RL   (HL)        ;Balra léptetjük a byte-
                                         ;okat.
ADDR+19   43            DEC  HL          ;Csökkentjük a címet,
ADDR+20   16,251        DJNZ ADDR+17     ;32-szer.
ADDR+22   193           POP  BC          ;Ha egy sor kész, ugrás
ADDR+23   16,236        DJNZ ADDR+05     ;ismét vissza, összesen
                                         ;192-szer.
ADDR+25   201           RET              ;Vége
```

-HIRES scroll jobbra

A megváltoztatandó sorok az előbbihez képest a következők:

```
ADDR+00   33,0,64       LD   HL,16384    ;HL-be töltjük a képernyő
                                         ;első címét.

ADDR+09   221,126,31    LD   A,(IX+31)   ;A-ba töltjük a képernyő
                                         ;másik szélén levő byte-
                                         ;ját, és a
ADDR+12   15            RRCA             ;szélső bitet betöltjük
                                         ;az átviteli jelzőbitbe.

ADDR+17   203,30        RR   (HL)        ;Jobbra léptetjük a byte-
                                         ;okat.
ADDR+19   35            INC  HL          ;Növeljük a címet.
```

Ebben a formában mindkét irányban OFF-LINE jellegű a görgetés, de ez könnyen átalakítható:

```
ADDR+13   0             NOP              ;A görgetés INVERZ jelle-
                                         ;gű lett.

ADDR+14   0             NOP              ;WRAP-AROUND jellegű gör-
                                         ;getést kaptunk.
```

-HIRES BOX scroll balra

A 'BOX' scroll-ok előnye, hogy beállíthatjuk a görgetendő terület nagyságát a képernyőn.

A bal-irányú rutinban a görgetendő terület (box) jobb felső sarkának képernyő címét az ADDR+01/02 memóriahelyeken kell megadni, a box magasságát az ADDR+04 címen képpontokban és végül a box szélességét karakterekben számolva az ADDR+05 címen.

```
ADDR+00   33,80.64      LD   HL,16464    ;HL-ben adjuk meg a ru-
                                         ;tinhoz a számítási kez-
ADDR+03   1,51,8        LD   BC,2099     ;dőcímet, BC-ben pedig a
                                         ;box méretét.
ADDR+06   34,0,91       LD   (23296),HL  ;A kezdőcímet és a box
ADDR+09   237,67,2,91   LD   (23298),BC  ;méretét kimentjük állan-
                                         ;dó memóriahelyekre.
ADDR+13   229           PUSH HL          ;A kezdőcímet a verembe
                                         ;is kimentjük.
ADDR+14   33,1,1        LD   HL,257      ;A 'WRAP-AROUND' üzemmód
ADDR+17   88            LD   E,B         ;biztosításához a box
ADDR+18   22,0          LD   D,0         ;szélessége függvényében
ADDR+20   167           AND  A           ;ki kell számítanunk az
ADDR+21   237,82        SBC  HL,DE       ;ADDR+31 utasítás elto-
ADDR+23   125           LD   A,L         ;lási tényezőjét. Ezt be-
ADDR+24   50,X,Y        LD   (ADDR+33),A;töltjük az ADDR+33. cím-
                                         ;re.(X és Y az ADDR+33.
                                         ;cím alsó és felső byte-
                                         ;ja).
ADDR+27   225           POP  HL
ADDR+28   229           PUSH HL          ;HL tartalmát áttöltjük
ADDR+29   221,225       POP  IX          ;az IX regiszterpárba.
ADDR+31   221,126,248   LD   A,(IX-8)    ;Az ADDR+08. címen a box
                                         ;szélességét 8 karakter-
                                         ;nyire állítottuk be, te-
                                         ;hát ennek megfelelően az
                                         ;ADDR+33. címre 248 író-
                                         ;dik.
ADDR+34   7             RLCA             ;Az átviteli jelzőbitbe
                                         ;íródik át a másik szélső
                                         ;bit.
ADDR+35   0             NOP              ;WRAP AROUND jellegű gör-
ADDR+36   0             NOP              ;getés.
ADDR+37   203,22        RL   (HL)        ;Az adott byte bitjeit
                                         ;balra léptetjük.
ADDR+39   43            DEC  HL          ;Csökkentjük a címet és
ADDR+40   5             DEC  B           ;az oszlop értéket is.
ADDR+41   32,250        JR   NZ,ADDR+37  ;Ha az adott képpontsor
                                         ;még nincs kész, ugrás
                                         ;vissza.
ADDR+43   42,0,91       LD   HL,(23296)  ;Visszaírjuk HL-be a sor
                                         ;induló byte-jának címét.
ADDR+46   24,15         JR   ADDR+63     ;Ugrás a cím számláló ru-
                                         ;tinra.
ADDR+48   34,0,91       LD   (23296),HL  ;A következő sor induló
                                         ;címét visszatöltjük a
                                         ;tárolóhelyre.
ADDR+51   237,75,2,91   LD   BC,(23298)  ;Betöltjük a box méretét.
ADDR+55   13            DEC  C           ;A sorok számát csökkent-
                                         ;jük.
ADDR+56   200           RET  Z           ;Vége, ha nincs több sor.
ADDR+57   237,67,2,91   LD   (23298),BC  ;BC tartalmát visszaírjuk.
```

```
ADDR+61   24,221        JR   ADDR+28     ;Visszaugrunk a következő
                                         ;képpontsor léptetéséhez.
ADDR+63   124           LD   A,H         ;Megvizsgáljuk, hogy ka-
ADDR+64   230,7         AND  7           ;raktersor váltás van-e ?
ADDR+66   254,7         CP   7
ADDR+68   40,3          JR   Z,ADDR+73   ;Ha igen, ugrás történik
                                         ;az ADDR+73. címre.
ADDR+70   36            INC  H           ;A H regiszter értékének
                                         ;eggyel történő növelése
                                         ;a címet 256-tal növeli,
                                         ;vagyis egy képpontsorral
                                         ;lejjebb lépünk.
ADDR+71   24,231        JR   ADDR+48     ;Ugrás vissza.
ADDR+73   125           LD   A,L         ;Megvizsgáljuk, hogy van-
ADDR+74   230,224       AND  224         ;e képernyő harmad vál-
ADDR+76   254,224       CP   224         ;tás ?
ADDR+78   40,8          JR   Z,ADDR+88   ;Ha nem, ugrás az ADDR+88
                                         ;címre.
ADDR+80   17,224,6      LD   DE,1760     ;Elvégezzük a képernyő-
ADDR+83   167           AND  A           ;harmadok közötti szüksé-
ADDR+84   237,82        SBC  HL,DE       ;ges cím korrekciót.
ADDR+86   24,216        JR   ADDR+48     ;Ugrás vissza.
ADDR+88   124           LD   A,H         ;A képernyő aljához ér-
                                         ;tünk ?
ADDR+89   254,87        CP   87
ADDR+91   40,211        JR   Z,ADDR+48   ;Ha igen, ugrás vissza.
ADDR+93   17,32,0       LD   DE,32       ;Elvégezzük a karakterso-
ADDR+96   25            ADD  HL,DE       ;rok közötti szükséges
                                         ;cím korrekciót.
ADDR+97   24,205        JR   ADDR+48     ;Ugrás vissza.
```

A rutin jelenlegi felépítésében WRAP AROUND jellegű görgetést végez.
Végezzük el a következő módosításokat:

```
ADDR+36   63            CCF              ;INVERZ scroll
```

Ha még ezen kívül a következőt is elvégezzük:

```
ADDR+35   55            SCF              ;OFF LINE jellegű görge-
                                         ;tést kapunk.
```

A jelenlegi rutin egy 8 karakter széles és 61 képpont magas box görgetését végzi el a képernyő bal felső szélén.

-HIRES BOX scroll jobbra

Az irány megváltoztatásához a következő módosításokat kell elvégezni:
A rutin kezdetén HL regiszterpárba a box bal felső sarkának képernyőcímét kell magadni.

NOP utasítással helyettesítjük a következő memóriahelyeket:
ADDR+13, ADDR+14, ADDR+15, ADDR+16, ADDR+18, ADDR+19
ADDR+20, ADDR+21, ADDR+27

```
ADDR+22   29          DEC  E
ADDR+23   123         LD   A,E

ADDR+34   15          RRCA

ADDR+37   203,30      RR   (HL)
ADDR+39   35          INC  HL
```

A WRAP AROUND, INVERZ és OFF LINE görgetés megadása a már ismert módon történik.

-RIPPLE scroll balra

```
ADDR+00   33,255,87   LD   HL,22527    ;HL-be töltjük a képernyő
                                       ;utolsó címét.
ADDR+03   14,32       LD   C,32        ;1 sor = 32 karakter.
ADDR+05   167         AND  A           ;Az átviteli jelzőbitet
                                       ;töröljük.
ADDR+06   203,6       RLC  (HL)        ;Az adott byte-ot biten-
                                       ;ként balra léptetjük, a
                                       ;7. bit mindig a 0. he-
                                       ;lyére lép.
ADDR+08   43          DEC  HL          ;HL és C regiszterek tar-
ADDR+09   13          DEC  C           ;talmát csökkentjük.
ADDR+10   32,250      JR   NZ,ADDR+06  ;Ha a C regiszter tartal-
                                       ;ma nem zérus, ugrás
                                       ;vissza,
ADDR+12   62,63       LD   A,63        ;máskülönben megvizsgál-
ADDR+14   188         CP   H           ;juk, hogy elértük-e a
                                       ;16384. címet ?
ADDR+15   32,242      JR   NZ,ADDR+03  ;Ha nem, ugrás vissza.
ADDR+17   201         RET              ;Vége
```

-RIPPLE scroll jobbra

Az előzőhöz képest a következő sorokat kell megváltoztatni:

```
ADDR+00   33,0,64     LD   HL,16384    ;HL-be töltjük a képernyő
                                       ;első címét.

ADDR+06   203,14      RRC  (HL)        ;Az adott byte-ot jobbra
                                       ;léptetjük, a 0. bit min-
ADDR+08   35          INC  HL          ;dig a 7.bit helyére lép.

ADDR+12   62,88       LD   A,88        ;Megvizsgáljuk, hogy el-
ADDR+14   188         CP   H           ;értük-e a 22527. címet.
```

-SHUTTER scroll balra

A RIPPLE scroll balra programjában csak az ADDR+06 cím tartalmát kell megváltoztatni:

```
ADDR+06   203,38      SLA  (HL)      ;Balra léptetjük az adott
                                     ;byte-ot bitenként, a 0.
                                     ;bit mindig 0-val töltő-
                                     ;dik fel.
```

-SHUTTER scroll jobbra

A RIPPLE scroll jobbra programjában csak az ADDR+06 cím tartalmát kell megváltoztatni:

```
ADDR+06   203,62      SRL  (HL)      ;Jobbra léptetjük biten-
                                     ;ként az adott byte-ot, a
                                     ;7.bit mindig 0-val töl-
                                     ;tődik fel.
```

-LORES scroll felfelé

A rutin a ROM-ban foglal helyet, így közvetlenül elérhető. A teljes képernyőt egy sorral feljebb mozgató rutin belépési pontjai a 3190. és a 3582. decimális címek lehetnek.
Gépeljük be a következő rövid BASIC programot:

```
10 CIRCLE 125,75,75
20 PAUSE 0
30 RANDOMIZE USR 3190
```

Futtassuk a programot, és ha a kör felrajzolódott, nyomjunk meg egy billentyűt. Láthatjuk, hogy a képernyő egy sorral feljebb lép.
Beléphetünk a rutinba a 3584. címen is, de ebben az esetben előzőleg nekünk kell a B regiszterben megadni, hogy alulról számítva hány sort szeretnénk (0 - 23) felfelé görgetni ?
Alsó 17 sor esetén a következő gépi kódú rutint alkalmazhatjuk:

```
ADDR+00   6,17        LD   B,17
ADDR+02   205,0,14    CALL 3584
ADDR+05   201         RET
```

-LORES scroll lefelé

```
ADDR+00   33,255,87   LD   HL,22527   ;HL-be töltjük a végcímet
ADDR+03   17,223,87   LD   DE,22495   ;DE-be pedig a 8 byte-tal
ADDR+06   229         PUSH HL         ;feljebb eső címet, majd
ADDR+07   213         PUSH DE         ;kimentjük őket a verem-
                                      ;be.
ADDR+08   14,23       LD   C,23       ;23 sort léptetünk eggyel
ADDR+10   6,32        LD   B,10       ;lejjebb, 32 oszlopban.
ADDR+12   26          LD   A,(DE)     ;Az aktuális byte-ot egy
ADDR+13   119         LD   (HL),A     ;sorral lejjebb helyez-
ADDR+14   121         LD   A,C        ;zük.
```

```
ADDR+15   230,7        AND  7            ;Ha a forrás sor még nem
ADDR+17   254,1        CP   1            ;ért képernyő harmad vál-
ADDR+19   32,2         JR   NZ,ADDR+23   ;táshoz, ugrás történik,
ADDR+21   151          SUB  A            ;máskülönben az adott
ADDR+22   18           LD   (DE),A       ;harmad felső sorának
                                         ;byte-jait zérussal tölti
                                         ;fel.
ADDR+23   43           DEC  HL           ;A képernyő címeket lép-
ADDR+24   27           DEC  DE           ;tetjük visszafelé.
ADDR+25   16,241       DJNZ ADDR+12      ;Ciklus 32-szer.
ADDR+27   13           DEC  C            ;Csökken a sorok száma.
ADDR+28   40,21        JR   Z,ADDR+51    ;Ugrás az ADDR+51. címre,
                                         ;ha nincs több sor.
ADDR+30   121          LD   A,C          ;A forrás karaktersor át-
ADDR+31   230,7        AND  7            ;lépi a képernyő harma-
ADDR+33   254,0        CP   0            ;dot ?
ADDR+35   40,24        JR   Z,ADDR+61    ;Ha igen, ugrás történik.
ADDR+37   254,7        CP   7            ;A cél karaktersor is át-
                                         ;lépi a képernyő harma-
ADDR+39   32,225       JR   NZ,ADDR+10   ;dot? Ha nem, ugrás visz-
                                         ;sza.
ADDR+41   213          PUSH DE           ;Elvégezzük a két képer-
ADDR+42   17,0,7       LD   DE,1792      ;nyőharmad közötti cím
ADDR+45   167          AND  A            ;korrekciót a cél karak-
ADDR+46   237,82       SBC  HL,DE        ;tersorra.
ADDR+48   209          POP  DE
ADDR+49   24,215       JR   ADDR+10      ;Ugrás vissza.
ADDR+51   209          POP  DE           ;Visszatöltjük az eredeti
ADDR+52   225          POP  HL           ;címeket és mindkettőt
ADDR+53   21           DEC  D            ;tovább léptetjük egy
ADDR+54   37           DEC  H            ;képpont sorral.
ADDR+55   124          LD   A,H          ;Ha már végeztünk soron-
ADDR+56   254,79       CP   79           ;ként mind a 8 képpont-
ADDR+58   200          RET  Z            ;sorral, akkor vége,
ADDR+59   24,201       JR   ADDR+06      ;máskülönben ugrás vissza
ADDR+61   229          PUSH HL           ;Elvégezzük a két képer-
ADDR+62   33,0,7       LD   HL,1792      ;nyőharmad közötti cím-
ADDR+65   235          EX   DE,HL        ;korrekciót a forrás ka-
ADDR+66   167          AND  A            ;raktersorra.
ADDR+67   237,82       SBC  HL,DE
ADDR+69   235          EX   DE,HL
ADDR+70   225          POP  HL
ADDR+71   24,193       JR   ADDR+10      ;Ugrás vissza.

-LORES scroll balra

ADDR+00   6,192        LD   B,192        ;192 sort kell léptetni.
ADDR+02   17,0,64      LD   DE,16384     ;DE-be töltjük a képernyő
ADDR+05   213          PUSH DE           ;kezdőcímét és a vermen
ADDR+06   225          POP  HL           ;keresztül áttöltjük HL-
                                         ; be.
ADDR+07   35           INC  HL           ;A címet növeljük.
ADDR+08   197          PUSH BC           ;A sorok számát kimentjük
```

```
ADDR+09   1,31,0        LD   BC,31       ;majd megadjuk, hogy 31
                                         ;oszlopot mozgatunk egy
                                         ;lépéssel balra.
ADDR+12   26            LD   A,(DE)      ;A-ban eltároljuk a szél-
                                         ;ső byte értékét (WRAP
                                         ;AROUND jelleghez).
ADDR+13   237,176       LDIR             ;Egy karaktersornyi kép-
                                         ;pont tartalmat byte-on-
                                         ;ként balra viszünk.
ADDR+15   43            DEC  HL          ;Visszalépünk a sor vé-
                                         ;gére.
ADDR+16   54,0          LD   (HL),0      ;A sor szélén levő byte-
                                         ;ot töröljük (OFF LINE
                                         ;jellegű görgetés).
ADDR+18   35            INC  HL          ;HL-t és DE-t a következő
ADDR+19   35            INC  HL          ;sor elejére pozícionál-
ADDR+20   19            INC  DE          ;juk.
ADDR+21   193           POP  BC          ;Ha még nincs kész a 192
ADDR+22   16,240        DJNZ ADDR+08     ;sor, akkor ugrás vissza.
ADDR+24   201           RET              ;Vége
```

A WRAP AROUND jellegű görgetéshez a következő változtatást kell tennünk:

```
ADDR+16   119           LD   (HL),A      ;Az adott byte-ot áttölt-
ADDR+17   0             NOP              ;jük a sor egyik széléről
                                         ;a másikra.
```

-LORES scroll jobbra

Az előző rutinhoz képest a következő változtatásokat kell tenni:

```
ADDR+02   17,255,87     LD   DE,22527    ;HL-be töltjük a képernyő
                                         ;végcímét.

ADDR+07   43            DEC  HL          ;A címet csökkentjük.

ADDR+13   237,184       LDDR             ;Egy karaktersornyi kép-
                                         ;pont tartalmat byte-on-
                                         ;ként jobbra teszünk.
ADDR+15   35            INC  HL          ;Visszalépünk a sor végé-
                                         ;re.

ADDR+18   43            DEC  HL          ;HL-t és DE-t a követke-
ADDR+19   43            DEC  HL          ;ző sor végére pozício-
ADDR+20   27            DEC  DE          ;náljuk.
```

A változtatás a WRAP AROUND jelleghez természetesen itt is működik.

Ha nem a teljes képernyőt akarjuk görgetni, hanem annak pl.valamely harmadát vagy harmadait, akkor az előbbi két rutinban néhány módosítást kell elvégezni.

A LORES scroll balra rutinban:

```
-Felső 1/3 képernyő     ADDR+00  6,64        LD B,64
                        ADDR+02  17,0,64     LD DE,16384
-Középső 1/3 képernyő   ADDR+00  6,64        LD B,64
                        ADDR+02  17,0,72     LD DE,18432
-Alsó 1/3 képernyő      ADDR+00  6,64        LD B,64
                        ADDR+02  17,0,80     LD DE,20480
-Felső 2/3 képernyő     ADDR+00  6,128       LD B,128
                        ADDR+02  17,0,64     LD DE,16384
-Alsó 2/3 képernyő      ADDR+00  6,128       LD B,128
                        ADDR+02  17,0,72     LD DE,18432
```

A LORES scroll jobbra rutinban:

```
-Felső 1/3 képernyő     ADDR+00  6,64        LD B,64
                        ADDR+02  17,255,71   LD DE,18431
-Középső 1/3 képernyő   ADDR+00  6,64        LD B,64
                        ADDR+02  17,255,79   LD DE,20469
-Alsó 1/3 képernyő      ADDR+00  6,64        LD B,64
                        ADDR+02  17,255,87   LD DE,22527
-Felső 2/3 képernyő     ADDR+00  6,128       LD B,128
                        ADDR+02  17,255,79   LD DE,20469
-Alsó 2/3 képernyő      ADDR+00  6,128       LD B,128
                        ADDR+02  17,255,87   LD DE,22527
```

4.3 ATTRIBUTUM scroll rutinok

-ATTRIBUTUM scroll lefelé

A rutin hívása előtt néhány értéket be kell állítani. Az ADDR +01 címen adjuk meg azt az ATTR kódot, amivel az elgörgetett területet fogja feltölteni (0-255).
A scroll behatárolható egy meghatározott nagyságú képernyőterületre (box), így meg kell adni a box bal felső karakterének oszloppozícióját (0-31) az ADDR+03 címen, valamint a box bal felső karakterének sor pozícióját is (0-23) az ADDR+04 címen. Ezen túl a box méretére is szükség van, a szélességet (1-32) az ADDR+06 címen, míg magasságát (1-24) pedig az ADDR+07 címen kell megadni a rutin futtatása előtt.
Mintaprogramunkban legyen az ATTR kód 255, a box bal felső pozíciója a 14/5 (sor/oszlop) érték és válasszuk a box méretét a következőre: szélessége = 10; magassága = 5.

```
ADDR+00   62,255      LD   A,255      ;A-ba töltjük az ATTR kó-
                                      ;dot.
ADDR+02   1,5,14      LD   BC,3589    ;BC-be töltjük a box bal-
                                      ;felső karakterének pozí-
                                      ;cióját.
ADDR+05   17,10,5     LD   DE,1290    ;DE-be töltjük a box mé-
                                      ;retét.
ADDR+08   33,223,87   LD   HL,22495   ;HL-be töltjük az induló
                                      ;címet.
```

```
ADDR+11   21          DEC  D          ;Csökkentjük a magasság-
                                      ;értéket.
ADDR+12   245         PUSH AF         ;Kimentjük az ATTR kódot.
ADDR+13   120         LD   A,B        ;Összegezzük a kiinduló
ADDR+14   130         ADD  A,D        ;sor ill. magasság érté-
                                      ;két,
ADDR+15   71          LD   B,A        ;és ezt áttöltjük B-be.
ADDR+16   121         LD   A,C        ;Összegezzük a kiinduló
ADDR+17   131         ADD  A,E        ;oszlop ill. szélesség
                                      ;értékét,
ADDR+18   79          LD   C,A        ;és ezt áttöltjük C-be.
ADDR+19   213         PUSH DE         ;A box méretét kimentjük.
ADDR+20   17,32,0     LD   DE,32      ;1 sor = 32 karakter.
ADDR+23   4           INC  B          ;Box alsó sor pozíció + 1
ADDR+24   25          ADD  HL,DE      ;HL-ben képződik a box
ADDR+25   16,253      DJNZ ADDR+24    ;jobb alsó karakter fe-
                                      ;letti pozíció címe.
ADDR+27   65          LD   B,C        ;B-be töltjük a box jobb
ADDR+28   0           NOP             ;oldali oszlop pozíció-
                                      ;ját.
ADDR+29   35          INC  HL         ;HL-ben képződik a box
ADDR+30   16,253      DJNZ ADDR+29    ;jobb alsó karakterpozí-
                                      ;ciójának címe.
ADDR+32   209         POP  DE         ;A box méretét vissza-
                                      ;töltjük.
ADDR+33   66          LD   B,D        ;B-be töltjük a box ma-
ADDR+34   0           NOP             ;gasságát és kimentjük a
ADDR+35   197         PUSH BC         ;verembe.
ADDR+36   6,0         LD   B,0        ;B-t töröljük.
ADDR+38   75          LD   C,E        ;C-be töltjük a box szé-
                                      ;lességét, majd a box mé-
ADDR+39   213         PUSH DE         ;retét és a jobb alsó po-
ADDR+40   229         PUSH HL         ;zíció címét kimentjük.
ADDR+41   17,32,0     LD   DE,32      ;Két karaktersor közötti
                                      ;byte különbség.
ADDR+44   167         AND  A          ;Az átviteli jelzőbitet
ADDR+45   237,82      SBC  HL,DE      ;töröljük és egy sorral
                                      ;feljebb lépünk a blokk-
                                      ;mozgatáshoz.
ADDR+47   209         POP  DE         ;HL és DE fogja tartal-
ADDR+48   229         PUSH HL         ;mazni az egymás alatti
                                      ;karakterek címeit.
ADDR+49   237,184     LDDR            ;Egy box szélességű ATTR
                                      ;sort eggyel lejjebb he-
                                      ;lyezünk.
ADDR+51   225         POP  HL         ;Az adatokat visszatölt-
ADDR+52   209         POP  DE         ;jük.
ADDR+53   193         POP  BC
ADDR+54   120         LD   A,B        ;Ugrás vissza, ha van még
ADDR+55   16,234      DJNZ ADDR+35    ;hátra sor.
ADDR+57   241         POP  AF         ;Visszatöltjük az új ATTR
                                      ;kódot.
```

```
ADDR+58   67          LD   B,E        ;B-be töltjük a box szé-
ADDR+59   119         LD   (HL),A     ;lességét és az adott el-
ADDR+60   43          DEC  HL         ;görgetett sor helyét
                                      ;feltöltjük a megadott
                                      ;ATTR értékkel.
ADDR+61   16,252      DJNZ ADDR+59    ;Ciklus a box szélességé-
                                      ;nek megfelelően.
ADDR+63   201         RET             ;Vége
```

-ATTRIBUTUM scroll felfelé

Az előbbi rutin könnyen átalakítható az irány megváltoztatására. Először is 9 cím tartalmát üres 'NOP' utasítással kell helyettesíteni. Ezek a címek a következők:
ADDR+11, ADDR+13, ADDR+14, ADDR+15, ADDR+16, ADDR+17, ADDR+18, ADDR+44 és végül ADDR+45.

Ezen túl a következő módosításokat kell elvégezni:

```
ADDR+28   4           INC  B

ADDR+34   5           DEC  B

ADDR+46   25          ADD  HL,DE      ;Egy sorral lejjebb lé-
                                      ;pünk a blokkmozgatáshoz.

ADDR+49   237,176     LDIR            ;Egy box szélességű ATTR
                                      ;sort eggyel feljebb he-
                                      ;lyez.

ADDR+60   35          INC  HL
```

Természetessen a NOP utasítások itt is az egyszerűbb átírás célját szolgálják, ha ezeket elhagyjuk és ennek megfelelően a belső relatív címzéseket átírjuk, a rutin ugyanúgy futni fog. Ezzel így egy-két byte-os memóriamegtakarítást érhetünk el.

-ATTRIBUTUM scroll jobbra

Alakítsuk át az'ATTRIBUTUM scroll lefelé' rutint az ADDR+34. címtől:

```
ADDR+34   4           INC  B          ;A box magasságát növel-
                                      ;jük.
ADDR+35   241         POP  AF         ;Az ATTR kódot vissza-
                                      ;töltjük.
ADDR+36   197         PUSH BC         ;BC-t kimentjük.
ADDR+37   6,0         LD   B,0        ;B-t töröljük és C-be
ADDR+39   75          LD   C,E        ;töltjük a box szélessé-
ADDR+40   13          DEC  C          ;gét, majd ez utóbbit
                                      ;csökkentjük.
ADDR+41   213         PUSH DE         ;A box méretét és a jobb
```

```
ADDR+42   229          PUSH HL          ;alsó karakterpozíció cí-
                                        ;mét kimentjük a verembe.
ADDR+43   209          POP  DE          ;Ez utóbbit DE-be vissza-
ADDR+44   26           LD   A,(DE)      ;töltjük, és tartalmát az
                                        ;A-ba visszük WRAP AROUND
                                        ;görgetéshez.
ADDR+45   213          PUSH DE          ;DE-t ismét kimentjük.
ADDR+46   43           DEC  HL          ;HL-t DE-hez képest egy-
                                        ;gyel visszaléptetjük, és
ADDR+47   237,184      LDDR             ;a jobbra léptetés egyen-
                                        ;ként megtörténik.
ADDR+49   18           LD   (DE),A      ;Az eltárolt értéket át-
                                        ;írjuk a képernyő - box
                                        ;adott sorának másik vé-
                                        ;gére.
ADDR+50   225          POP  HL          ;A címet kimentjük.
ADDR+51   17,32,0      LD   DE,32       ;Két karaktersor közötti
                                        ;byte különbség.
ADDR+54   167          AND  A           ;Az átviteli jelzőbitet
ADDR+55   237,82       SBC  HL,DE       ;töröljük és egy sorral
                                        ;feljebb lépünk.
ADDR+57   209          POP  DE          ;Az adatokat visszatölt-
ADDR+58   193          POP  BC          ;jük és visszaugrunk, ha
ADDR+59   16,231       DJNZ ADDR+36     ;van még hátra sor,
ADDR+61   201          RET              ;máskülönben vége.
```

-ATTRIBUTUM scroll balra

Az átalakításhoz az előbbihez képest előbb itt is néhány cím tartalmát üres 'NOP' utasítással kell helyettesíteni. Ezek a következők:
ADDR+11, ADDR+13, ADDR+14, ADDR+15, ADDR+16, ADDR+17, ADDR+18, ADDR+34, ADDR+54 és végül ADDR+55.
Ezek után végezzük el a következő módosításokat:

```
ADDR+28   4            INC  B

ADDR+46   35           INC  HL
ADDR+47   237,176      LDIR             ;A balra léptetés egyen-
                                        ;ként megtörténik.

ADDR+56   25           ADD  HL,DE       ;Egy sorral lejjebb lé-
                                        ;pünk.
```

A bal/jobb irányú ATTRIBUTUM scroll rutinok jelenlegi felépítésükben WRAP AROUND jelleggel görgetik a képernyőt. Az OFF LINE tipusú görgetéshez mindkét rutinban a következő módosítást végezzük el:

```
ADDR+44   0            NOP              ;'A' regiszter értéke zérus
                                        ;marad, így az ADDR+49
                                        ;címen történő visszaol-
                                        ;vasáskor a sor másik vé-
                                        ;gére semmi sem kerül.
```

5.fejezet

HASZNÁLJUK KI A PRINT KÜLÖNBÖZŐ LEHETŐSÉGEIT

5.1 Hagyományos PRINT

Nem kívánjuk ismertetni a PRINT BASIC-ből elérhető lehetőségeit, hiszen minden Spectrum tulajdonos attól a pillanattól kezdve, hogy a gép a birtokában van, használja ezt az utasítást különböző variációkban. A hagyományos PRINT alatt most itt az utasítás gépi kódú alaplehetőségeit tekintjük át.

Egyes karakterek kinyomtatására előnyösen alkalmazható az 1. fejezetben már megismert RST 16-os utasítás. Ez a PRINT ROM rutin egyik belépési pontja. A PRINT-tel kapcsolatos összes paramétert (pozíció, színállapot, OVER stb.) a ROM rutin hívása előtt egyenként meg kell adni az A regiszterben. Minden paraméter bevitele után külön meg kell hívnunk az RST 16-os utasítást. Minden próbálkozásunk hiábavaló, ha a képernyő megfelelő csatornáját előzetesen nem nyitjuk meg. A képernyő felső 22 sorának megnyitása is ROM rutinnal lehetséges, a belépési pont az 5633.-as (HEX-1601) cím. A csatorna megnyitása előtt a csatorna megfelelő kódját az A regiszterben kell megadni. A képernyő felső része a 2-es csatorna. Az eddig elmondottak alapján egy karaktert a következőképpen jeleníthetünk meg a képernyőn:

```
ADDR+00   62,2          LD   A,2        ;A képernyő felső részé-
                                        ;nek a csatornája.
ADDR+02   205,1,22      CALL 5633       ;Megnyitjuk a csatornát.
ADDR+05   62,65         LD   A,65       ;A nagy "A" betű ASCII
                                        ;kódja.
ADDR+07   215           RST  16         ;Az "A" betű megjelenik a
                                        ;legutolsó PRINT pozíció-
                                        ;ban.
ADDR+08   201           RET             ;Vége
```

Ekkor azonban karakterünk még csak a legutoljára érvényes kiírási pozícióban fog megjelenni, alaphelyzetben tehát a bal felső sarokban. A következő mintapélda az előbbi kibővítése, és arra mutat példát, hogyan lehet a PRINT állapot paramétereit beállítani:

```
ADDR+00   62,2          LD   A,2        ;A képernyő felső részé-
                                        ;nek csatornája.
ADDR+02   205,1,22      CALL 5633       ;Megnyitjuk a csatornát.
ADDR+05   62,16         LD   A,16       ;Az INK kódja.
ADDR+07   215           RST  16
ADDR+08   62,2          LD   A,2        ;A tinta szín piros lesz.
ADDR+10   215           RST  16
ADDR+11   62,17         LD   A,17       ;A PAPER kódja.
ADDR+13   215           RST  16
ADDR+14   62,6          LD   A,6        ;A papír szín sárga lesz.
ADDR+16   215           RST  16
ADDR+17   62,19         LD   A,19       ;A BRIGHT kódja.
```

```
ADDR+19   215          RST  16
ADDR+20   62,1         LD   A,1        ;A fényességet 1-re ál-
                                       ;lítjuk.
ADDR+22   215          RST  16
ADDR+23   62,18        LD   A,18       ;A FLASH kódja.
ADDR+25   215          RST  16
ADDR+26   62,1         LD   A,1        ;A villogás bekapcsoló-
                                       ;dik.
ADDR+28   215          RST  16
ADDR+29   62,20        LD   A,20       ;Az INVERSE kódja.
ADDR+31   215          RST  16
ADDR+32   62,1         LD   A,1        ;A kiírás inverz lesz.
ADDR+34   215          RST  16
ADDR+35   62,21        LD   A,21       ;Az OVER kódja.
ADDR+37   215          RST  16
ADDR+38   62,1         LD   A,1        ;Az OVER aktív állapotú
                                       ;lesz.
ADDR+40   215          RST  16
ADDR+41   62,22        LD   A,22       ;Az AT kódja.
ADDR+43   215          RST  16
ADDR+44   62,5         LD   A,5        ;A sor-pozíció.
ADDR+46   215          RST  16
ADDR+47   62,10        LD   A,10       ;Az oszlop-pozíció.
ADDR+49   215          RST  16
ADDR+50   62,65        LD   A,65       ;A kiíratandó karakter
                                       ;ASCII kódja.
ADDR+52   215          RST  16
ADDR+53   201          RET             ;Vége
```

Ha ezt a mintapéldát futtatjuk, megjelenik az "A" betű az AT 5,10 pozícióban sárga alapon, piros színnel, inverzben, fényesen és villogva. Az itt látható mintapéldából kitűnik, hogy egy karakterre a paraméterek beállítása időigényes munka, BASIC-ben sokkal könnyebben megoldhatjuk ugyanezt. Mérlegelni kell, hol alkalmazzuk ezeket a rutinokat, egyedül csak a PRINT miatt nem sok értelme van a gépi kódú átírásnak, hosszabb gépi kódú program esetén is meg kell vizsgálni, hogy csak néhány karakter kiíratásáról ill. PRINT paraméterek beállításáról van-e szó, vagy hosszabb szövegeket kell megjelenítenünk ? Ez utóbbi esetben célszerű a ROM-ba beépített string-kiíró rutint meghívnunk. A karakter-stringet bárhol elhelyezhetjük a memóriában, csak az első string karakter memóriabeli címét és a string hosszát kell ismernünk. Mielőtt meghívjuk a stringkiíratő ROM rutint, a DE regiszterpárban kell megadni a string első karakterének memóriabeli címét ill. a BC regiszterpárban a string hosszát. A string megjelenítéséhez természetesen előzetesen meg kell nyitnunk a képernyő megfelelő csatornáját, és be kell állítanunk a kiírás egyéb paramétereit az előbb megismertek alapján.

Nézzünk egy egyszerű mintapéldát, írjuk ki a képernyőre azt, hogy "Csinos hosszú string!".

```
ADDR+00   62,2          LD   A,2          ;A képernyő felső részé-
                                          ;nek a csatornája.
ADDR+02   205,1,22      CALL 5633         ;Megnyitjuk a csatornát.
ADDR+05   62,22         LD   A,22         ;Az AT kódja.
ADDR+07   215           RST  16
ADDR+08   62,12         LD   A,12         ;A sor-pozíció.
ADDR+10   215           RST  16
ADDR+11   62,5          LD   A,5          ;Az oszlop-pozíció.
ADDR+13   215           RST  16
ADDR+14   17,X,Y        LD   DE,ADDR+24   ;DE-be töltjük a string
                                          ;első karakterének memó-
                                          ;riabeli címét."X" és "Y"
                                          ;az ADDR+24.cím alsó/fel-
                                          ;ső byte-ja.
ADDR+17   1,21,0        LD   BC,21        ;BC-be töltjük a string
                                          ;hosszát karakterekben
                                          ;kifejezve.
ADDR+20   205,60,32     CALL 8252         ;Meghívjuk a string-kiíró
                                          ;ROM rutint (HEX-203C).
ADDR+23   201           RET               ;Ha a kiírás megtörtént,
                                          ;akkor vége.
ADDR+24   67,115,105,   DEFM "Csinos hosszú string!"
          110,111,115,                    ;szöveg eltárolása a me-
          32,104,111,                     ;móriában.
          115,115,122,
          117,32,115,
          116,114,105,
          110,103,33
ADDR+45   0             DEFB 0            ;Az adott stringet mindig
                                          ;zérus byte-nak kell le-
                                          ;zárnia, ez nem számít
                                          ;bele a string hosszába !
                                          ;(BC-ben sem kell beszá-
                                          ;mítani.)
```

Még egy fontos dolog! Az itt ismertetett rutinok nem oldják meg kiírás előtt a képernyő törlését. Ez gépi kódból a ROM-beli CLS rutinnal lehetséges:

```
ADDR+NN   205,107,13    CALL 3435         ;Meghívjuk a képernyőtör-
                                          ;lő ROM rutint (HEX-0D6B)
```

5.2 PRINT nagyított karakterekkel

A karakterek x és y irányú nagyítását egy kb. 300 byte hosszú rutin segítségével megoldhatjuk. Erről a rutinról feltételezzük, hogy a legtöbb Spectrum tulajdonos birtokában van, hiszen az eredeti - géphez mellékelt - DEMO kazettán többször is megtalálható "mCODE" néven. Az utóbbi időben egyre nagyobb azoknak az aránya, akik ZX Spectrum+ számítógépet vesznek. Ehhez a géphez már nem az eredeti - PSION Software által forgalmazott - DEMO

kazettát mellékelték, hanem a DORLING Software "GOLD STAR" nevű kazettáját. Nem kívánjuk most itt az assembly listát közölni, de a Spectrum+ tulajdonosok kedvéért közöljük a rutin BASIC beolvasó programját.

Ez a rutin az eredeti kazettán a 32256. címtől töltődik be 300 byte hosszon, igaz a rutin teljes hossza csak 277 byte. Most itt is az említett memóriaterületre olvassuk be a rutint:

```
10 FOR i=32256 TO 32532
20 READ a
30 POKE i,a
40 NEXT i
50 DATA 33,15,91,126,35,34,0,91,111,60,200,38,0,41,41,41,
        237,75,54,92,9,62,8,50,4,91,58,11,91,50,9,91,58,
        10,91,50,8,91
60 DATA 62,9,50,5,91,126,35,34,2,91,7,50,6,91,58,5,91,61,
        32,50,58,4,91,61,32,24,58,14,91,71,58,12,91,79,58,
        10,91,129,5
70 DATA 32,252,50,10,91,42,0,91,195,3,126,50,4,91,58,13,
        91,71,58,9,91,128,50,9,91,42,2,91,195,32,126,50,
        5,91,58,12,91
80 DATA 71,58,9,91,50,7,91,58,13,91,79,197,205,164,126,193,
        58,7,91,60,50,7,91,13,32,241,58,8,91,60,50,8,91,
        5,32,221,58,6,91
90 DATA 195,48,126,128,64,32,16,8,4,2,1,58,142,92,238,255,
        71,58,141,92,160,71,58,8,91,230,248,111,58,7,91,
        254,192,208,31,31,31,230,31
100 DATA 103,203,28,203,29,203,28,203,29,203,28,203,29,62,88,
        180,103,58,142,92,166,176,119,58,7,91,71,230,7,246,
        64,103,120,31,31,31
110 DATA 230,24,180,103,120,23,23,230,224,111,58,8,91,71,31,
        31,31,230,31,181,111,235,33,156,126,120,230,7,79,
        6,0,9,70,26
120 DATA 33,6,91,203,70,40,3,176,18,201,47,176,47,18,201
```

Ez a rutin az eredeti DEMO kazettán azért erre a cimre töltődik, mert egyaránt szólt mind a 16, mind a 48 kbyte RAM memóriájú gépek tulajdonosainak.

Ha a rutint beolvastuk, mentsük ki. Adjunk CLEAR 32255 utasítást, majd NEW-t hogy a beolvasó BASIC program törlődjön. A rutin önmagában még nem elegendő, néhány dolgot meghatározott címeken be kell állítanunk.
Gépeljük be a következő BASIC sorokat:

```
9000 LET xx=(256-8*xs*LEN p$)/2
9010 LET i=23306: POKE i,xx: POKE i+1,yy:
     POKE i+2,xs: POKE i+3,ys: POKE i+4,8
9020 LET i=i+4: LET w=LEN p$
9030 FOR n=1 TO w: POKE i+n,CODE p$(n): NEXT n
9040 POKE i+w+1,255: LET w=USR 32256:RETURN
```

Ez lesz a felhasználói szubrutinunk, ami a gépi kódot is meghívja.

A szubrutin hívása előtt a következő változókat kell megadnunk:

yy=0...186 - a megjelenítendő szöveg felső szélének pozíciója képpontokban értve, és a képernyő felső szélétől számolva.
xs=1,2,..n - a karakterlánc karakterenként vett x irányú nagyítási mértéke (alapállapot: 8 képpont, tehát a kétszeres nagyítás karakterenként 16 képpontot határoz meg, és így tovább).
ys=1,2,..n - a karakterlánc karakterenként vett y irányú nagyítási mértéke (ld.xs).
p$="szöveg"- idézőjelek között kell megadnunk a szöveget, ami 1 karakter is lehet, hossza függ a nagyítás mértékétől.

A rutin működésekor a gép beépített karakterkészletét vizsgálja, így grafikus karakterek megjelenítésére csak úgy van lehetőség, ha a karaktermutató értékét átállítjuk. (Ez megoldható a 23606 és 23607 rendszerváltozókkal - ld. később.)
A 9000 sorszámon induló szubrutin először elvégzi a szöveg x irányú középre pozícionálását, majd a megadott szöveg karaktereit egyenként eltárolja megfelelő memóriahelyeken, ahonnan később azzal a gépi kódú rutin manipulálni tud.

Nézzünk a rutin alkalmazására egy szemléletes példát:

```
 10 LET a=0: LET b=1: LET c=1: GO SUB 100
 20 LET a=8: LET b=2: LET c=2: GO SUB 100
 30 LET a=24: LET b=2: LET c=3: GO SUB 100
 40 LET a=48: LET b=3: LET c=3: GO SUB 100
 50 LET a=72: LET b=3: LET c=4: GO SUB 100
 60 LET a=102: LET b=2: LET c=6: GO SUB 100
 70 LET a=150: LET b=3: LET c=2: GO SUB 100
 80 LET a=166: LET b=1: LET c=2: GO SUB 100
 90 LET a=184: LET b=2: LET c=1: GO SUB 100
 95 PAUSE 0: STOP
100 LET yy=a: LET xs=b: LET ys=c: LET p$=
    "ZX-Spectrum": GO SUB 9000
110 RETURN
```

Természetesen az előzőleg ismertetett-9000. sortól kezdődő-szubrutinnak is a memóriában kell lennie.
Adjuk ki RUN (ENTER) és az ábrán látható szövegek megjelennek a képernyőn. A PAUSE 0 utasítás billentyű megnyomásra vár, azért helyeztük el, hogy a képernyő alsó sorai is láthatóak legyenek.

Gyakorlati problémát jelenthet, hogy a rutin a 32256 címtől található, s aki 48 kbyte-os géppel rendelkezik joggal szeretné, ha a memória végén lenne, hisz így pl. a BASIC program hossza sem annyira kötött. A rutinban 5 db. abszolút címhivatkozás van, ezeket kell átírni az új területnek megfelelően és a rutin már át is helyezhető. Erre is bemutatunk egy példát.
Helyezzük át a rutint a 32256 címről a 65024 címre. Ne felejtsük el a RAMTOP-ot lejjebb helyezni! Az eredeti állapothoz képest végezzük el a következő módosításokat:

```
POKE 32343,254: POKE 32363,254: POKE 32384,254:
POKE 32411,254: POKE 32508,254  (ENTER)
```

A rutin tehát ebben az állapotában már áthelyezhető az új címre. Az áthelyezés BASIC-ben kicsit lassú, gépi kódban az LDIR utasítással ez sokkal gyorsabban megoldható. Mivel erre a könyvben már többször volt utalás, most csak a rövid beolvasó rutint ismertetjük:

```
1000 FOR i=50000 TO 50011
1010 READ a: POKE i,a: NEXT i
1020 DATA 33,0,126,17,0,254,1,21,1,237,176,201
```

Most RUN 1000 (ENTER) és átolvasó rutinunk eltárolódik az 50000 címtől. Nincs más hátra: RANDOMIZE USR 50000 (ENTER) és előbbi rutinunknak a 65024 címen kell lennie. Mielőtt az új címen futtatnánk a rutint érdemes kimenteni:

```
SAVE "név"CODE 65024,277 (ENTER)
```

A 9040-es sorban írjuk át a 32256-os címhivatkozást 65024-re. Mentsük ki a BASIC programot is, majd futtassuk a rutint kedvünk szerint.

5.3 Képpont-pozícionált PRINT

Igazán jó szolgáltatás, amely a Beta Basic-ben is megtalálható. Elszakadunk a merev és kötött 32 oszlop ill. 22+2 sor képernyő felosztástól és képpontonkénti pozícionálásra térünk át. A kiíratandó string első karakterének bal felső képpont-pozícióját kell megadnunk, méghozzá a következő formában:
A 23728-as memóriahelyen kell megadni az x pozíciót, ami balról jobbra számítva nő 0-255-ig, ill. a 23729-es memóriahelyen pedig az y pozíciót, pontosabban annál 8-cal kevesebbet.
Először a rutint ismertetjük, majd utána áttekintjük a rutinnal megoldható kiíratási lehetőségeket.
A rutin futtatása előtt az a$ változóban tároljuk el a kiíratásra szánt szöveget, vagyis azt a rutin a változóterületen keresi meg. A megértéshez tudni kell, hogy ha a változó területen nincs definiált változó, akkor annak első byte-ja a 128-as értéket veszi fel. A különböző típusú változóknak más és más "fej"-byte-juk van, így pl. az egyszerű egykarakteres változó első byte-ja a 96+n értéket kapja, ahol n a karakter sorszáma ("a"esetén n=1,"b" esetén n=2 stb.). String-változó esetén ez az érték 64+n, tehát ezt kell kikeresnie és azonosítania. Bővebb információkhoz tanácsoljuk a Spectrum kézikönyv 24. fejezetének ide vonatkozó részét a változók felépítéséről.

```
ADDR+00   151          SUB  A           ;Az A regisztert töröljük
ADDR+01   71           LD   B,A         ;BC és DE minden bitjét
ADDR+02   79           LD   C,A         ;töröljük.
ADDR+03   87           LD   D,A
ADDR+04   95           LD   E,A
ADDR+05   42,75,92     LD   HL,(23627)  ;HL-be kerül a változóte-
                                        ;rület kezdőcíme.
ADDR+08   126          LD   A,(HL)      ;Ha nincs definiált vál-
ADDR+09   254,128      CP   128         ;tozó a memóriában, akkor
ADDR+11   200          RET  Z           ;vége.
ADDR+12   203,127      BIT  7,A         ;A zérus jelzőbitbe írjuk
                                        ;az A regiszter 7.bitjé-
                                        ;nek negáltját.
ADDR+14   32,31        JR   NZ,ADDR+47  ;Ha ez nem zérus, ugrás
                                        ;az ADDR+47. címre.
ADDR+16   254,96       CP   96          ;Ha egyszerű egykarakte-
ADDR+18   48,19        JR   NC,ADDR+39  ;res aritmetikai változót
                                        ;talál, ugrás az ADDR+39.
                                        ;címre.
ADDR+20   254,65       CP   65          ;Az a$ változó ?
ADDR+22   32,2         JR   NZ,ADDR+26  ;Ha nem, ugrás az ADDR+26
                                        ;címre.
ADDR+24   84           LD   D,H         ;DE-be áttöltjük az a$
ADDR+25   93           LD   E,L         ;változó meghatározó
ADDR+26   122          LD   A,D         ;byte-ját.
ADDR+27   179          OR   E           ;Ha DE nem zérus (a$ ese-
                                        ;tén), akkor ugrás törté-
ADDR+28   32,38        JR   NZ,ADDR+68  ;nik az ADDR+68. címre.
ADDR+30   213          PUSH DE          ;DE-t kimentjük, majd
```

```
ADDR+31   35          INC  HL          ;HL-t a változó hosszát
ADDR+32   94          LD   E,(HL)      ;meghatározó byte-okra
ADDR+33   35          INC  HL          ;léptetve DE-be betöltjük
ADDR+34   86          LD   D,(HL)      ;a változó hosszát.
ADDR+35   25          ADD  HL,DE       ;HL-ben képezzük az adott
                                       ;string utolsó byte-jának
                                       ;címét.
ADDR+36   209         POP  DE          ;DE-t visszatöltjük.
ADDR+37   24,5        JR   ADDR+44     ;Ugrás az ADDR+44.címre.
ADDR+39   35          INC  HL          ;Léptetjük a címmutató
ADDR+40   35          INC  HL          ;értékét.
ADDR+41   35          INC  HL
ADDR+42   35          INC  HL
ADDR+43   35          INC  HL
ADDR+44   35          INC  HL
ADDR+45   24,217      JR   ADDR+08     ;Ugrás az ADDR+08.címre.
ADDR+47   254,224     CP   224         ;FOR-NEXT ciklusváltozó ?
ADDR+49   56,6        JR   C,ADDR+57   ;Ha nem, ugrás az ADDR+57
                                       ;címre.
ADDR+51   213         PUSH DE          ;DE-t eltesszük, és a he-
ADDR+52   17,18,0     LD   DE,18       ;lyére 18-at töltünk.
ADDR+55   24,234      JR   ADDR+35     ;Ugrás az ADDR+35.címre.
ADDR+57   203,111     BIT  5,A         ;Numerikus vagy string-
ADDR+59   40,225      JR   Z,ADDR+30   ;tömb esetén ugrás az
                                       ;ADDR+30. címre.
ADDR+61   35          INC  HL          ;Több betűs aritmetikai
ADDR+62   203,126     BIT  7,(HL)      ;változó esetén megkeres-
                                       ;sük az utolsó betűt.
ADDR+64   40,251      JR   Z,ADDR+61   ;Ha még nem az utolsó,
                                       ;ugrás vissza.
ADDR+66   24,227      JR   ADDR+39     ;Ugrás vissza az ADDR+39.
                                       ;címre.
-------------------------------------------------------------
ADDR+68   35          INC  HL          ;DE-be betöltjük a string
ADDR+69   94          LD   E,(HL)      ;változó hosszát.
ADDR+70   35          INC  HL
ADDR+71   86          LD   D,(HL)
ADDR+72   35          INC  HL          ;HL most a kiíratandó ka-
                                       ;rakterre mutat.
ADDR-73   126         LD   A,(HL)      ;Az A regiszterbe töltjük
                                       ;a karakter ASCII kódját.
ADDR+74   213         PUSH DE          ;A regisztereket kiment-
ADDR+75   229         PUSH HL          ;jük.
ADDR+76   24,39       JR   ADDR+117    ;Ugrás az ADDR+117.címre.
-------------------------------------------------------------
ADDR+78   225         POP  HL          ;A regisztereket vissza-
ADDR+79   209         POP  DE          ;töltjük.
ADDR+80   229         PUSH HL          ;HL-t kimentjük.
ADDR+81   33,176,92   LD   HL,23728    ;HL-be töltjük a kiíra-
                                       ;tandó karakter x pozíci-
                                       ;ójának tároló címét.
ADDR+84   6,8         LD   B,8         ;Jobbra lépünk 8 képpont-
ADDR+86   52          INC  (HL)        ;nyit.
```

```
ADDR+87   16,253        DJNZ ADDR+86     ;Ciklus 8-szor.
ADDR+89   0             NOP
ADDR+90   58,176,92     LD   A,(23728)   ;Ezt a megnövelt értéket
                                         ;A-ba töltjük.
ADDR+93   254,0         CP   0           ;Ha még nem értük el a
ADDR+95   32,13         JR   NZ,ADDR+110;képernyő végét, akkor
                                         ;ugrás az ADDR+110.címre.
ADDR+97   62,0          LD   A,0         ;Az x pozíciót a képernyő
ADDR+99   50,176,92     LD   (23728),A   ;elejére állítjuk,
ADDR+102  58,177,92     LD   A,(23729)   ;és betöltjük A-ba az y
                                         ;pozíció értékét.
ADDR+105  214,8         SUB  8           ;Az y pozíciót csökkent-
                                         ;jük 8 képpontnyival,
ADDR+107  50,177,92     LD   (23729),A   ;majd visszatöltjük.
ADDR+110  225           POP  HL          ;HL-t (a karakter válto-
                                         ;zóterületi címét)vissza-
                                         ;töltjük.
ADDR+111  27            DEC  DE          ;A karakterek számának
                                         ;mutatóját csökkentjük.
ADDR+112  122           LD   A,D         ;Ha van még kiíratandó
ADDR+113  179           OR   E           ;karakter, akkor ugrás
ADDR+114  32,212        JR   NZ,ADDR+72  ;az ADDR+72. címre,
ADDR+116  201           RET              ;máskülönben vége.
-------------------------------------------------------------------
ADDR+117  38,0          LD   H,0         ;A karakter ASCII kódját
ADDR+119  111           LD   L,A         ;áttöltjük L-be.
ADDR+120  41            ADD  HL,HL       ;Az itt látható számítá-
ADDR+121  41            ADD  HL,HL       ;si mechanizmussal megha-
ADDR+122  41            ADD  HL,HL       ;tározzuk az adott karak-
ADDR+123  237,91,54,92  LD   DE,(23606)  ;ter memóriabeli címét, a
ADDR+127  25            ADD  HL,DE       ;karaktermutató szerinti
                                         ;karaktertáblában.
ADDR+128  6,8           LD   B,8         ;A karaktert 8 sorból
                                         ;építjük össze.
ADDR+130  14,7          LD   C,7         ;(7-0)8 lépésben az adott
ADDR+132  86            LD   D,(HL)      ;byte-ot bitenként balra
ADDR+133  203,34        SLA  D           ;léptetjük és levizsgál-
                                         ;juk, hogy az átviteli
                                         ;jelzőbit milyen értékű ?
ADDR+135  56,8          JR   C,ADDR+145  ;1 esetén ugrás az ADDR+
                                         ;145. címre.
ADDR+137  13            DEC  C           ;C-t csökkentjük, és ha
                                         ;még nem érte el a zérust
ADDR+138  32,249        JR   NZ,ADDR+133;ugrás vissza az ADDR+133
                                         ;címre.
ADDR+140  35            INC  HL          ;Máskülönben a karakter
                                         ;következő byte-jára lé-
                                         ;pünk.
ADDR+141  16,243        DJNZ ADDR+130    ;Ciklus 8-szor.
ADDR+143  24,189        JR   ADDR+78     ;Ugrás vissza az ADDR+78.
                                         ;címre.
```

```
ADDR+145  245          PUSH AF          ;Kimentjük a regisztere-
ADDR+146  197          PUSH BC          ;ket.
ADDR+147  213          PUSH DE
ADDR+148  229          PUSH HL
ADDR+149  42,176,92    LD   HL,(23728)  ;HL-be töltjük a kiírási
                                        ;pozíciót (H-ba y-t és L-
                                        ;be x-et).
ADDR+152  62,6         LD   A,6         ;BC-ben képezzük a PLOT
ADDR+154  145          SUB  C           ;rutinba való belépéshez
ADDR+155  133          ADD  A,L         ;a megjelenítendő pont x
ADDR+156  79           LD   C,A         ;és y koordinátáját.
ADDR+157  120          LD   A,B
ADDR+158  132          ADD  A,H
ADDR+159  71           LD   B,A
ADDR+160  205,229,34   CALL 8933        ;Meghívjuk a ROM-beli
                                        ;PLOT rutint (HEX-22E5).
ADDR+163  225          POP  HL          ;Visszatöltjük a regisz-
ADDR+164  209          POP  DE          ;tereket.
ADDR+165  193          POP  BC
ADDR+166  241          POP  AF
ADDR+167  24,224       JR   ADDR+137    ;Ugrás vissza az ADDR+
                                        ;137. címre.
```

Ha a rutint beírtuk-tetszőleges helyre-a memóriába, nézzünk egy egyszerű mintapéldát:

```
10 POKE 23728,0: POKE 23729,168
20 LET a$="Olyan mint a normal kiiras"
30 RANDOMIZE USR ADDR+00
```

Futtatáskor tényleg nem hoz váratlan meglepetést.
Egészítsük ki programunkat:

```
40 POKE 23728,3: POKE 23729,165
50 LET a$="Ez kicsit jobbra/lejjebb van"
60 RANDOMIZE USR ADDR+00
```

Láthatjuk, hogy a kiírás kezdőpontját képpont-pontossággal beállíthatjuk.

Azt szeretnénk, hogy ne 32 hanem 42 oszlopban jelenjenek meg a karakterek a képernyőn. Ennek semmi akadálya, csak egy-két változtatást kíván.

```
POKE ADDR+85,6: POKE ADDR+94,252
```

Írjunk 35-ös sorszámmal STOP-ot:

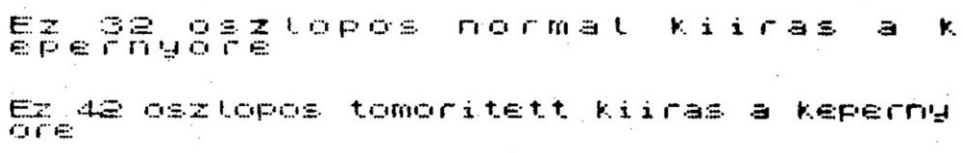
```
35 STOP
```

Ismét futtassuk: RUN (ENTER).
Lám első szövegünk a 42 oszlopos formában jelenik meg.

A 36 oszlopos kiírásnál az ADDR+85 címen 7-et kell megadnunk.
Van még egy lehetőség: kövérített karaktereket rajzolhatunk a rutin segítségével.

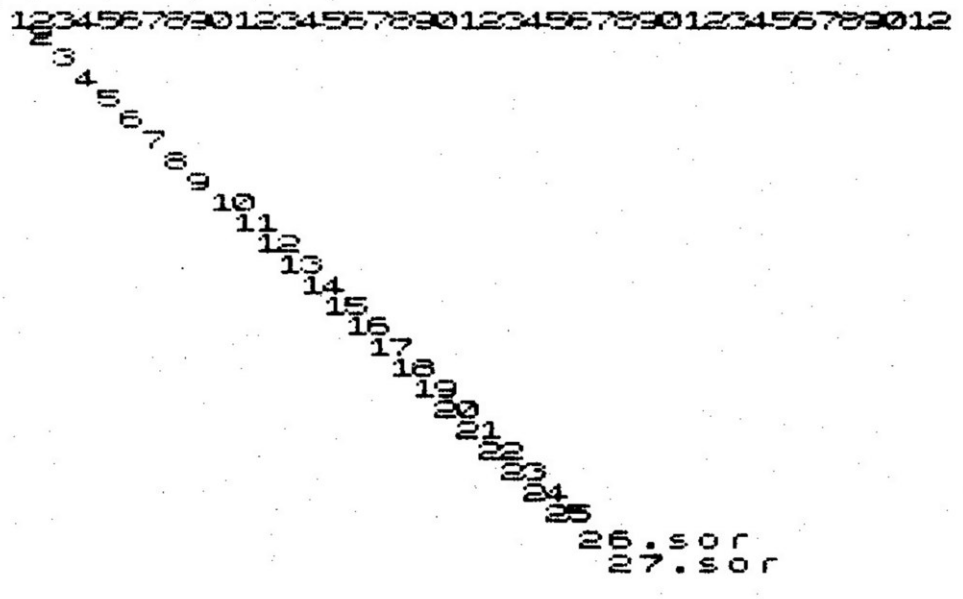
```
100 FOR i=32 TO 34
110 GO SUB 140
120 NEXT i
130 STOP
140 POKE 23728,i: POKE 23729,100
150 LET a$="Z X   S P E C T R U M"
160 REM minden karakter után hagyjunk egy SPACE-t!
170 RANDOMIZE USR ADDR+00
180 RETURN
```

és RUN 100 (ENTER).
A következő ábrán látható szöveg-kiírási formák a rutin eddigi eredményeit szemléltetik.



Z X S P E C T R U M

Eddig sikerült elérni az x irányú oszlopméret változtatását, de a sorokat is írhatjuk sűrűbben. Legcélszerűbb 8 helyett 7 képpont magas karaktersorokat előállítani, így a képernyő felső részén 25 sorba írhatunk. A 42 oszlop és 27 sor előállításához már csak egy értéket kell megváltoztatnunk a rutinban.
Írjuk be: POKE ADDR+106,7 (ENTER).
A következő ábra a tömörített sorok és oszlopok elhelyezkedését szemlélteti:



Egy utolsó tanács a rutinnal kapcsolatban!
A jelenlegi átalakított állapot mellett gépeljük be a következő BASIC mintapéldát:

```
200 POKE 23728,0: POKE 23729,168
210 LET a$="Ez a szoveg most hosszabb mint 42
    karakter, igy soremelesnek kell kovetkeznie"
220 RANDOMIZE USR ADDR+00
```

Futtassuk a programot: RUN 200 (ENTER), és a szöveg annak rendje és módja szerint tömörített formában soremeléssel jelenik meg a képernyőn.
A 200-as sorban a 23728-as címre most írjunk 1-et, vagyis a kiírási pozíciót x irányban 1 képpontnyival jobbra helyezzük. Újbóli futtatásnál sajnos azt tapasztalhatjuk, hogy nincs soremelés, a szöveg egymásra íródik. A rutin az adott tömörítésnek megfelelően kiszámítja, hogy mikor kell sort emelnie. 42 oszlopos kiíratás esetén a kezdeti x pozíciónak 6-tal, 36 oszlopos változatban 7-tel és végül normális esetben 8-cal oszthatónak kell lennie.
Soremelés után mindig az x=0 koordináta pozíciótól kezdi az új sor megjelenítését. Ebből kiindulva, képpont-pontosan pozícionált szövegünk ne legyen olyan hosszú, hogy kilépjen a képernyő szélén, mert meglepetések érhetnek. Hosszabb szöveg esetén pedig tartsuk be az előbb említett feltételeket.

5.4 PRINT több karakterkészlettel

Mint ismeretes egy időben összesen 21 db. UDG (felhasználó által definiált grafikus karakter) jeleníthető meg minden különösebb beavatkozás nélkül a képernyőn. Ez pl. a magyar ABC ékezetes karaktereinek elegendő, viszont kevésnek bizonyul ha egy teljes karakterkészletet szeretnénk lecserélni egy másikra.
Az angol ABC 26 nagy, és ugyanennyi kisbetűt használ, s akkor még az írásjeleket nem is említettük. Azt, hogy a számítógép milyen formájú karaktert rendel az adott ASCII kódok mellé, a CHARS rendszerváltozóban eltárolt címmutatótól függ. Ez a rendszerváltozó két byte-on, a 23606 és 23607 memóriahelyeken található. Bekapcsoláskor értéke automatikusan a következő lesz:

23606-0
23607-60

Ez azt jelenti, hogy a 60*256=15360-as ROM címre mutat. Jól jegyezzük meg, a karaktertábla sohasem ezen a címen kezdődik, hanem mindig 256 byte-tal feljebb. A ROM beépített karaktertáblájának kezdőcíme így a 15616-os memóriarekesz. A Spectrum a ROM-ban 96 különféle karaktert (nagy-kisbetűt, írásjelet, szimbólumot stb.) tárol el, ez összesen 96*8=768 byte-ot igényel. Ezek a karakterek a CHR$ 32 - CHR$ 127 kódok mellé vannak rendelve, vagyis az első karakter a SPACE és a 15616-os cím a SPACE 8 byte-ja közül a legfelső, értéke egyértelműen zérus. Értelemszerűen 8 byte-tal feljebb, vagyis a 15624-es címen kezdődik a kö-

vetkező karakter (felkiáltójel) 8 byte-ja, és így tovább. Ha a CHARS rendszerváltozó tartalmát megváltoztatjuk, a címmutató más lesz, de a működés zavartalan, hisz csak az ASCII kódokhoz rendelt megjelenítési tartalom változik meg. Így ezt a lehetőséget tetszés szerint felhasználhatjuk az új karakterkészletekkel való manipulációra. Egy mondaton belül-melyet a képernyőre szeretnénk írni- átválthatunk a karakterkészletek között, így pl. a lényegesebb szavakat vastagított formában jeleníthetjük meg stb.
Új karakterkészletek generálása lassú és időigényes munka. Az itt látható karakterkészletek kódjait megtalálhatjuk a függelékben.

Karakterkészletek:

1. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
2. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
3. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
4. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
5. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
6. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
7. ABCDEFGHIJKLMNOPQRSTUVWXYZ
   abcdefghijklmnopqrstuvwxyz
8. АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШ
   ЩЪЫЬЭЮЯ
   абвгдеёжзийклмнопрстуфхцчш
   щъыьэюя
9. ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
   αβγδεζηθικλμνξοπρστυφχψω
10. ĄĄĄÇČĆĘĚĚĚĮŁŃØÓÓÖÖŘŠÞŪŪŪŪŽŽ
    ąąąçčćęěěěįłńøóóööřšþūūūūžž

Elsősorban a nagy és kisbetűket emeltük ki, mert a felhasználás oldaláról ezeknek van nagyobb jelentőségük. Az írásjelek, szimbólumok, ill. számok átalakítása nem annyira szükségszerű, de a felhasználó ezeket már könnyen a betűformákhoz igazíthatja. A Spectrum beépített betűkészletét azok kedvéért közöljük, akik ezt más tipusú számítógépen kívánják felhasználni.

5.5 Forgatott PRINT

Bizonyára minden Spectrum tulajdonos aki használta már a "Melbourne Draw" rajzolóprogramot (részletes leírása megtalálható a "SINCLAIR Spectrum játék és program" c. könyvben - LSI ATSz 1986.) észrevette, hogy ez a program rendelkezik egy egyszerű, de érdekes lehetőséggel, a forgatott PRINT-tel. A forgatás annyit jelent, hogy karaktereinket 90, 180, vagy 270 fokkal elforgatva is kiírathatjuk a képernyőre. Ezt felhasználhatjuk pl.képernyő grafikáink díszítésére, "meghökkentő" feliratozására. Az itt ismertetett rutin mindössze 52 byte hosszú, és egy teljes karakterkészletet forgat el 90 fokkal jobbra. Természetesen az alap karakterkészletünkkel csak úgy manipulálhatunk, hogy előbb kimentjük RAM területre, mivel az a ROM területen nem írható felül.
Először is a rutint ismertetjük:

```
ADDR+00   33,X,Y      LD   HL,cím      ;HL-be töltjük a RAM te-
                                       ;rületen elhelyezkedő ka-
                                       ;rakterkészletünk első
                                       ;címét ( "X" és "Y" az
                                       ;adott cím alsó/felső
                                       ;byte-ja).
ADDR+03   6,96        LD   B,96        ;96 karaktert forgatunk
                                       ;meg.
ADDR+05   197         PUSH BC          ;BC-t kimentjük a verem-
                                       ;be.
ADDR+06   30,128      LD   E,128       ;E regiszter 7. bitjét
                                       ;1-re állítjuk.
ADDR+08   6,1         LD   B,1         ;B regiszter 0. bitjét 1-
                                       ;re állítjuk.
ADDR+10   14,0        LD   C,0         ;C regisztert töröljük.
ADDR+12   229         PUSH HL          ;Kimentjük a karakter
                                       ;adott byte-jának címét.
ADDR+13   123         LD   A,E         ;A-ba töltjük az E re-
                                       ;giszter tartalmát,
                                       ;amelyben mindig csak egy
                                       ;bit egyes.
ADDR+14   166         AND  (HL)        ;Összehasonlítjuk a ka-
                                       ;rakter adott byte-jával.
ADDR+15   254,0       CP   0           ;Ha nincs az E regiszter
                                       ;és a karakter adott
                                       ;byte-ja között közös 1-
                                       ;es bit, akkor ugrás tör-
ADDR+17   40,3        JR   Z,ADDR+22   ;ténik az ADDR+22. címre.
ADDR+19   121         LD   A,C         ;Itt összegezzük a karak-
```

```
ADDR+20   128          ADD  A,B          ;ter byte-onként vett
                                         ;azonos bitjeit.
ADDR+21   79           LD   C,A          ;C-ben képződik a 90 fok-
                                         ;kal elfordított új ka-
                                         ;rakter meghatározott
                                         ;byte-ja.
ADDR+22   203,32       SLA  B            ;B regiszter tartalmát
                                         ;balra léptetjük, a 7.bit
                                         ;az átviteli jelzőbitbe
                                         ;kerül.
ADDR+24   35           INC  HL           ;Növeljük a karakterbyte
                                         ;címét.
ADDR+25   48,242       JR   NC,ADDR+13   ;Ha még nem végeztünk a 8
                                         ;byte-tal, ugrás az ADDR+
                                         ;13.címre.
ADDR+27   225          POP  HL           ;Visszatöltjük a karakter
                                         ;első byte-jának címét.
ADDR+28   197          PUSH BC           ;Az elfordított karakter
                                         ;adott byteját kimentjük.
ADDR+29   203,59       SRL  E            ;E regiszter tartalmát
                                         ;jobbra léptetjük, a 0.
                                         ;bit az átviteli jelző-
                                         ;bitbe kerül.
ADDR+31   48,231       JR   NC,ADDR+08   ;Ha még nem vizsgáltunk
                                         ;meg oszloponként minden
                                         ;bitet, ugrás vissza.
ADDR+33   17,7,0       LD   DE,7         ;DE-be 7-et töltünk, ami
                                         ;a különbözet a karakter
                                         ;legfelső és legalsó
                                         ;byte-jának címe között.
ADDR+36   6,8          LD   B,8          ;8 byte átírása történik.
ADDR+38   25           ADD  HL,DE        ;HL-ben most a karakter
                                         ;legalsó byte-jának címe
                                         ;van.
ADDR+39   209          POP  DE           ;Egyenként visszaírjuk a
                                         ;veremből a 90 fokkal el-
                                         ;forgatott karakter byte-
                                         ;jait, melyeket sorban az
ADDR+40   115          LD   (HL),E       ;E regiszter tartalmaz.
ADDR+41   43           DEC  HL           ;1 byte-tal feljebb lé-
                                         ;pünk.
ADDR+42   16,251       DJNZ ADDR+39      ;Ciklus 8-szor.
ADDR+44   17,9,0       LD   DE,9         ;A következő karakter el-
ADDR+47   25           ADD  HL,DE        ;ső byte-jának címét ál-
                                         ;lítjuk elő.
ADDR+48   193          POP  BC           ;A ciklusok számát visz-
                                         ;szatöltjük.
ADDR+49   16,210       DJNZ ADDR+05      ;Ciklus 96-szor.
ADDR+51   201          RET               ;Vége
```

Nézzünk a rutin alkalmazására egy mintapéldát.Feltételezzük,hogy a rutint beolvastuk a 60000 címtől kezdve. Generálnunk kell egy karakterkészletet a RAM területen, erre a legcélszerűbb, ha a

beépített karakterkészletet mint egy memóriablokkot átmentjük a memória felső területére. Mint már említettük a beépített karakterkészlet a 15616-os címtől 768 byte hosszon található. A 15616 os cím alsó/felső byte-ja 0/61, az egyszerűség kedvéért helyezzük át a 0/201 címre, azaz az új kezdőcím az 51456 lesz. Ennek megfelelően ADDR+01-re (most 60001-re) zérust, és ADDR+02-re (most 60002-re) 201-et írunk. A memória átmozgatását már ismerjük, a következő formában egyszerüen elvégezhető:

```
10 FOR i=50000 TO 50011
20 READ a: POKE i,a: NEXT i
30 DATA 33,0,61,17,0,201,1,0,3,237,176,201
40 RANDOMIZE USR 50000
50 STOP
```

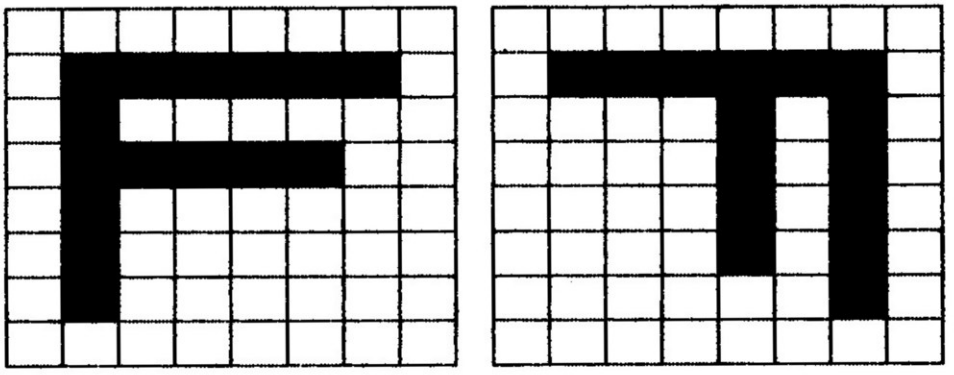
Adjuk ki: RUN (ENTER), és a karakterkészlet már a RAM-területen is megtalálható. Mint láttuk a karakterkészlet-mutató 256 bytetal mutat lejjebb az első karakter első byte-jának címénél, vagyis ha most kiadjuk:

POKE 23607,200 (ENTER), látszólag semmi változás nincs, de a gép már az új, áthelyezett karakterkészletet használja. Ha a forgató rutin is rendben a memóriában van, bátran adjuk ki:
RANDOMIZE USR 60000 (ENTER) és visszatéréskor programunk listája is elforgatva jelenik meg. A gépi kód hívását még háromszor megismételve visszakapjuk az eredeti állapotot.

Gépeljük be a következő BASIC programot:

```
 60 RESTORE 150
 70 PRINT AT 5,8;"NORMAL"
 80 RANDOMIZE USR 60000
 90 FOR i=10 TO 11: READ a$: PRINT AT i,15;
    a$: NEXT i
100 RANDOMIZE USR 60000
110 PRINT AT 15,6;"TTOTIDROF"
120 RANDOMIZE USR 60000
130 FOR i=11 TO 9 STEP-1: READ a$: PRINT
    AT i,5;a$: NEXT i
140 RANDOMIZE USR 60000
150 DATA "L","E","F","E","L"   és RUN 60  (ENTER).
```

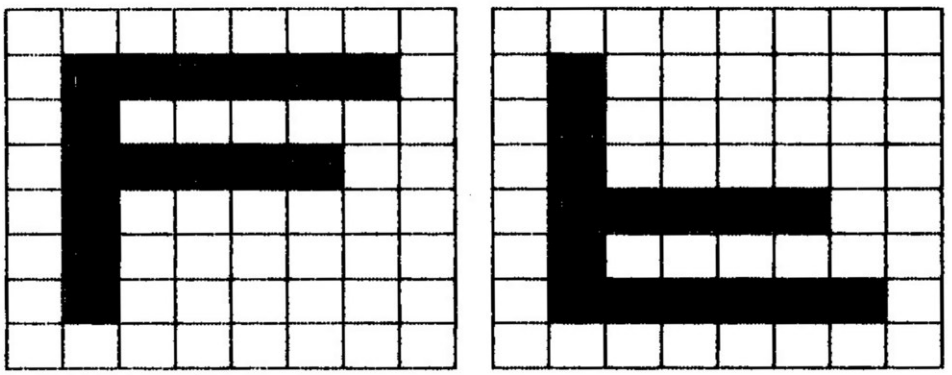
Az elfordítási mechanizmus két fázisa az ábrán látható:



5.6 Tükrözött PRINT

A tükrözött PRINT azt jelenti, hogy karaktereinket külön x és külön y tengelyükre is áttükrözhetjük, ezzel is bővítve a kiíratási lehetőségeket.

5.6.1 -x tengelyre



Mint ahogy az ábrából is kitűnik, az x tengelyre tükrözés gyakorlatilag azt jelenti, hogy a legfelső byte-ot kicseréljük a legalsóval, majd mindkét oldalról egyet beljebb lépünk, és ismét végrehajtjuk a cserét, és így tovább. A rutin a tükrözést egyszerűbb módon hajtja végre, mégpedig úgy hogy a byte-okat egyenként kimenti a verembe, majd a címet visszaállítja a kiinduló pozícióra, és a kimentett értékeket egyenként visszatölti.
Mivel a veremben a legutoljára kimentett érték lesz elsőnek elérhető, visszatöltéskor kialakul a fordított sorrend.

```
ADDR+00   33,0,201      LD   HL,51456   ;Feltételezzük, hogy az
                                        ;áthelyezett karakter-
                                        ;készlet még az 51456.cí-
                                        ;men található.
```

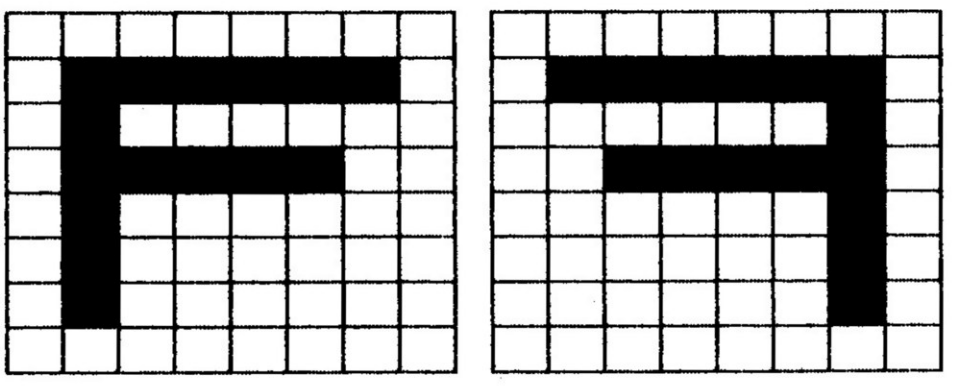
```
ADDR+03   6,96          LD   B,96        ;96 karaktert tükrözünk.
ADDR+05   197           PUSH BC          ;BC-t kimentjük a verembe
ADDR+06   6,8           LD   B,8         ;1 karakter 8 byte.
ADDR+08   93            LD   E,L         ;DE-be áttöltjük a karak-
ADDR+09   84            LD   D,H         ;ter adott byte-jának cí-
                                         ;mét.
ADDR+10   126           LD   A,(HL)      ;A-ba töltjük a cím tar-
                                         ;talmát.
ADDR+11   35            INC  HL          ;A címet növeljük, és a
ADDR+12   245           PUSH AF          ;tartalmat kimentjük a
                                         ;verembe.
ADDR+13   16,251        DJNZ ADDR+10     ;Ciklus 8-szor.
ADDR+15   6,8           LD   B,8         ;8-szor visszaírjuk a
ADDR+17   241           POP  AF          ;kezdeti címtől a kimen-
ADDR+18   18            LD   (DE),A      ;tett byte-okat,fordított
ADDR+19   19            INC  DE          ;sorrendben.
ADDR+20   16,251        DJNZ ADDR+17     ;Ciklus 8-szor.
ADDR+22   193           POP  BC          ;Visszatöltjük a ciklusok
                                         ;számát.
ADDR+23   16,236        DJNZ ADDR+05     ;Ciklus 96-szor.
ADDR+25   201           RET              ;Vége
```

Ha az előző program állapota még megvan, akkor töltsük be ezt a rutint pl. a 61000. címtől és nézzük meg a BASIC mintapéldát:

```
100 RESTORE 200
110 FOR i=11 TO 20: READ a$: PRINT AT 10,i;
    a$: NEXT i
120 RANDOMIZE USR 61000
130 RESTORE 200: FOR i=11 TO 20: READ a$:
    PRINT AT 11,i;a$: NEXT i
140 RANDOMIZE USR 61000
200 DATA "T","U","K","O","R","K","E","P","E","M"
```

és RUN 100 (ENTER).
A képernyőn valóban a szöveg tükörképe jelenik meg.

5.6.2 -y tengelyre



Azt hinnénk, az y tengelyre tükrözés ennél jóval bonyolultabb, hisz az előző példában egyszerűen felcserélgettük a byte-okat. Érdemes megnézni a rutin elvét, még egy byte-tal rövidebb is az előzőnél.

```
ADDR+00   33,0,201      LD   HL,51456    ;Feltételezzük, hogy az
                                         ;elhelyezett karakter-
                                         ;készlet még az 51456.cí-
                                         ;men van.
ADDR+03   6,96          LD   B,96        ;96 karaktert tükrözünk.
ADDR+05   197           PUSH BC          ;BC-t kimentjük a verembe
ADDR+06   62,8          LD   A,8         ;Egy karakter 8 byte.
ADDR+08   6,8           LD   B,8         ;Byte-onként 8 bit.
ADDR+10   203,30        RR   (HL)        ;A C regiszterben alakul
ADDR+12   203,17        RL   C           ;ki az adott byte y-ra
ADDR+14   16,250        DJNZ ADDR+10     ;vett tükörképe.
ADDR+16   113           LD   (HL),C      ;A tükrözött byte-ot
                                         ;visszaírjuk az eredeti
                                         ;byte helyére.
ADDR+17   35            INC  HL          ;A karakter következő
                                         ;byte-jára lépünk.
ADDR+18   61            DEC  A           ;Ha van még byte, akkor
ADDR+19   32,243        JR   NZ,ADDR+08  ;ugrás az ADDR+08. címre.
ADDR+21   193           POP  BC          ;Visszatöltjük a ciklusok
                                         ;számát.
ADDR+22   16,237        DJNZ ADDR+05     ;Ciklus 96-szor.
ADDR+24   201           RET              ;Vége.
```

Ismét feltételezzük az előbbi állapot változatlanul hagyását, s töltsük be ezt a rutint pl. a 62000. címtől. Egészítsük ki az előbbi BASIC programunkat:

```
150 RESTORE 200
160 FOR i=5 TO 14: READ a$: PRINT AT i,15;
    a$: NEXT i
170 RANDOMIZE USR 62000
180 RESTORE 200: FOR i=5 TO 14: READ a$:
    PRINT AT i,16;a$: NEXT i
190 RANDOMIZE USR 62000
```

Futtassuk az előzővel együtt: RUN 100 (ENTER).
A képernyőn megjelenik az x és y irányban tükrözött szöveg.

A fejezetben ismertetett rutinokból egy egységes egész építhető össze, hisz mindegyik a karaktermutató szerint meghatározott karakterkészlettel manipulál, s így pl. a tükrözött vagy elforgatott karaktereket nagyíthatjuk is, tömöríthetjük is a képernyőn, s nem utolsó sorban teljes mértékben elhagyhatjuk a BASIC-ből való hivatkozást. Ez utóbbiak alaposabb kimunkálását az olvasóra bízzuk.

P l u s z e g y r á a d á s

(CLEAR 32767)

```
65290    243              DI
65291    33,0,128         LD   HL,32768
65294    6,8              LD   B,8
65296    219,254          IN   A,(254)
65298    203,119          BIT  6,A
65300    32,2             JR   NZ,65304
65302    203,254          SET  7,(HL)
65304    203,62           SRL  (HL)
65306    16,244           DJNZ 65296
65308    203,14           RRC  (HL)
65310    35               INC  HL
65311    124              LD   A,H
65312    254,254          CP   254
65314    32,234           JR   NZ,65294
65316    251              EI
65317    201              RET
-------------------------------------
65318    243              DI
65319    33,0,128         LD   HL,32768
65322    6,8              LD   B,8
65324    203,70           BIT  0,(HL)
65326    40,4             JR   Z,65332
65328    62,0             LD   A,0
65330    211,254          OUT  (254),A
65332    62,255           LD   A,255
65334    211.254          OUT  (254),A
65336    203,6            RLC  (HL)
65338    16,240           DJNZ 65324
65340    203,6            RLC  (HL)
65342    35               INC  HL
65343    124              LD   A,H
65344    254,254          CP   254
65346    32,230           JR   NZ,65322
65348    251              EI
65349    201              RET
```

Megjegyzés: DIGITÁLIS HANGRÖGZÍTÉS.
Gépeljük be: RANDOMIZE USR 65290, majd indítsuk el a magnetofont. A rögzíteni kívánt zenei részlet, vagy szöveg elérésekor nyomjuk meg az ENTER-t. A felvétel OK üzenettel fejeződik be.
A digitális effektek visszajátszása a RANDOMIZE USR 65318 (ENTER) utasítással lehetséges.

A.FÜGGELÉK

MIRE JÓ A DISASSEMBLER (DISTRON)

A disassembler programok segítségével a memóriában helyet foglaló gépi kódú programjainkról (a kódok számokból álló sorozatáról) assembly listát készíthetünk. A Z-80 mnemonikok segítségével lehetőség nyílik a gépi kódú programban való eligazodásra, másrészt megkönnyíti hibakeresési munkánkat is. Gondoljunk csak arra, mennyivel könnyebb DATA sorból hibásan beolvasott programunk hibafeltárása, ha nem az egyes kódokat ellenőrizzük végig, hanem a mnemonikok alapján szinte ránézésre "kibökjük" a hibát.

A legtöbb disassembler program nem önálló programként került forgalomba, hanem monitor-programmal együtt. Sokszor nem célunk a monitor használata, ha csak disassemblálni akarunk, másrészt nem mindegy, hogy memóriánkban mekkora a szabadon felhasználható terület, ez is alátámasztja sok esetben a disassembler program önálló használatát. Egy disassembler program akkor rugalmas, ha a memória teljes területét át tudja pásztázni, vagyis bárhova áthelyezhető. Végül de nem utolsó sorban előnyös, ha nem csak a hexadecimális, hanem a decimális számok kezelését is lehetővé teszi.

Azért választottuk a DISTRON Disassembler (DK'TRONICS 1983) program ismertetését, mert rendelkezik az összes itt említett lehetőséggel, másrészt ismertetése kimaradt a "SINCLAIR Spectrum játék és program" c. könyvből (LSI ATSz. 1986). Úgy gondoljuk a Spectrum tulajdonosok zöme rendelkezik ezzel a rendkívül rugalmas programmal, s ha nem ismerné a program összes lehetőségeit, és szolgáltatásait, most pótolhatja hiányzó ismereteit.

A DISTRON 48 kbyte-os változatának (csak ezzel foglalkozunk) betöltése előtt a biztonság kedvéért adjuk ki:

CLEAR 59999 (ENTER) és ezután

LOAD ""CODE (ENTER)

A DISTRON a betöltés után a

RANDOMIZE USR 60000 (ENTER) utasítással aktivizálható.

Az első kérdésre (Hex or decimal output ?) a "h" vagy "d" megfelelő billentyűk megnyomásával kell válaszolni, és ennek megfelelően jelenik meg az assembly lista.
A következő kérdésre (Start address?) a disassemblálandó terület kezdőcímét , míg az utolsó kérdésre (End address?) pedig annak végcímét kell megadni.
Mind a három input bevitele után meg kell nyomni az ENTER billentyűt.
Akár decimális, akár hexadecimális output-ot kértünk, input-ban a kezdő és végcímet mindkét számrendszerben megadhatjuk, csak a hexadecimális számnak egy '&' jellel kell kezdődnie. A hexade-

cimális számok betűi kis és nagybetűk is lehetnek. Bevitelkor tehát pld. 9720, &25f8 és &25F8 mind ugyanazt jelenti.
A végcímet nem kötelező megadni, elég megnyomni az ENTER-t, ez esetben az assembly listát a megadott kezdőcímtől a memória végéig jelzi ki.

A képernyőn egyszerre maximum 22 sor output jelenik meg, és ha van még hátra disassemblálás, akkor megjelenik az ismert scroll? üzenet a képernyő alján. Ilyenkor több mindent tehetünk:

P - képernyőtartalom másolása nyomtatóra (Hard-copy)
R - új start-címet állíthatunk be
N vagy SPACE - megállítja a disassemblálást
Bármi más - a disassemblálás folytatódik

A DISTRON nagy előnye, hogy az RST 08 és RST 40 utasítások után következő adatbyte-okat képes felismerni és, nem kapunk "hamis" assembly listát, mint sok más disassembler program használatakor. Itt megjegyeznénk, hogy az RST utáni számjegy programozási hibából adódóan mindig hexadecimális formában jelenik meg, függetlenül attól, hogy decimális vagy hexadecimális output-ot kértünk.
Az RST 08 utasítások után a hiba kódját, mint DEFB adatbyte-ot azonosítja, és megjeleníti a hibának megfelelő hibakódot is.
Egy tetszőleges decimális output ennek megfelelően így néz ki:

```
63450   207.............RST 08
63451   10..............defb 10=B
  .     .                 .
  .     .                 .
```

Az RST 40 a Spectrum lebegőpontos kalkulátorát hívja meg. Erre is nézzünk meg egy tetszőleges listát:

```
45000   239.............RST 28
45001   15..............defb 15=ADD
45002   56..............defb 56=EXIT
```

Láthatjuk, decimális output esetén is hexadecimális érték van az RST utasítás mögött.

Egy tetszőleges ROM terület hexadecimális output-ja.:

```
25F8    CD 30 25........call 2530
25FB    28 28...........jr z,2625
25FD    ED 4B 76 5C.....ld bc,5c76
2601    CD 2B 2D........call 2D2B
2604    EF..............RST 28
2605    A1..............defb A1=1
2606    0F..............defb 0F=ADD
  .     .                 .
  .     .                 .
```

A Spectrum lebegőpontos kalkulátorának részletesebb áttanulmányozására lehetőség van a "ZX Spectrum BASIC és gépi kódú programozás" c. könyvben - Ipari Informatikai Központ 1985.

Egy fontos kérdés maradt még nyitott, a DISTRON áthelyezésének lehetősége. A DISTRON bárhova áthelyezhető a szabad memóriahelyre !
Alaphelyzetben a DISTRON a 60000. címtől töltődik be. Ha vizsgálandó kódunk is ezen a területen van, két dolgot tehetünk.

- a disassemblálandó kódot helyezzük át (ez nem mindig szerencsés dolog)
- a DISTRON-t helyezzük át (ez rugalmasabb)

A DISTRON alaphelyzetben 3474 byte hosszú, de a disassembler ebből csak 2912 byte-ot használ. A maradék kb. 500 byte az áthelyező mechanizmus. Ha kiadjuk:

RANDOMIZE USR 62912 (ENTER), lehetőség van megadni azt az új címet, ahova a DISTRON-t át akarjuk helyezni. Pld. adjunk meg 40000-ret, majd nyomjuk meg az ENTER-t és egy szempillantás alatt áthelyeződik a DISTRON a 40000. címre, és ennek megfelelően természetesen a belső címzések is átíródnak.

A DISTRON az új területről a következők szerint menthető ki:

SAVE "DISTRON" CODE új cím,2912 (ENTER)

Természetesen az új, áthelyezett DISTRON már nem tartalmazza az áthelyező mechanizmust. Egy újabb helyre csak úgy helyezhetjük át ismét, ha előbb betöltjük az eredeti programot a 60000. címre és az áthelyezési műveletet újra elvégezzük.

Néhány tanács:
- Áthelyezéskor 60000.-nél nagyobb címet ne adjunk meg !
- Ha meghívjuk a DISTRON-t, törlődik a változóterület !

Végül összegezzük a DISTRON előnyeit:

- Az output lehet decimális vagy hexadecimális.
- Az RST 08 ill. RST 40 (HEX RST 28) utáni disassemblált lista korrekt.
- Az assembly listát a nyomtatóra küldhetjük.
- A bemeneti paramétereket megadhatjuk decimális vagy hexadecimális formában is.
- A végcím megadása nem kötelező (nyomtatónál tanácsos).
- Áthelyezhető a memóriában.

	Nagybetűk	Kisbetűk
A -	0,60,66,66,126,66,66,0	0,0,56,4,60,68,60,0
B -	0,124,66,124,66,66,124,0	0,32,32,60,34,34,60,0
C -	0,60,66,64,64,66,60,0	0,0,28,32,32,32,28,0
D -	0,120,68,66,66,68,120,0	0,4,4,60,68,68,60,0
E -	0,126,64,124,64,64,126,0	0,0,56,68,120,64,60,0
F -	0,126,64,124,64,64,64,0	0,12,16,24,16,16,16,0
G -	0,60,66,64,78,66,60,0	0,0,60,68,68,60,4,56
H -	0,66,66,126,66,66,66,0	0,64,64,120,68,68,68,0
I -	0,62,8,8,8,8,62,0	0,16,0,48,16,16,56,0
J -	0,2,2,2,66,66,60,0	0,4,0,4,4,4,36,24
K -	0,68,72,112,72,68,66,0	0,32,40,48,48,40,36,0
L -	0,64,64,64,64,64,126,0	0,16,16,16,16,16,12,0
M -	0,66,102,90,66,66,66,0	0,0,104,84,84,84,84,0
N -	0,66,98,82,74,70,66,0	0,0,120,68,68,68,68,0
O -	0,60,66,66,66,66,60,0	0,0,56,68,68,68,56,0
P -	0,124,66,66,124,64,64,0	0,0,120,68,68,120,64,64
Q -	0,60,66,66,82,74,60,0	0,0,60,68,68,60,4,6
R -	0,124,66,66,124,68,66,0	0,0,28,32,32,32,32,0
S -	0,60,64,60,2,66,60,0	0,0,56,64,56,4,120,0
T -	0,254,16,16,16,16,16,0	0,16,56,16,16,16,12,0
U -	0,66,66,66,66,66,60,0	0,0,68,68,68,68,56,0
V -	0,66,66,66,66,36,24,0	0,0,68,68,40,40,16,0
W -	0,66,66,66,66,90,36,0	0,0,68,84,84,84,40,0
X -	0,66,36,24,24,36,66,0	0,0,68,40,16,40,68,0
Y -	0,130,68,40,16,16,16,0	0,0,68,68,68,60,4,56
Z -	0,126,4,8,16,32,126,0	0,0,124,8,16,32,124,0
---	---	---
A -	0,0,126,70,126,70,70,0	0,0,60,6,62,70,62,0
B -	0,0,124,98,124,98,124,0	0,48,48,62,34,34,62,0
C -	0,0,126,70,64,70,126,0	0,0,30,48,48,48,30,0
D -	0,0,126,70,70,70,126,0	0,6,6,62,70,70,62,0
E -	0,0,126,96,126,96,126,0	0,0,60,70,124,96,62,0
F -	0,0,126,96,126,96,96,0	0,14,24,28,24,24,24,0
G -	0,0,126,64,78,70,126,0	0,0,62,70,70,62,6,60
H -	0,0,98,98,126,98,98,0	0,96,96,124,70,70,70,0
I -	0,0,24,24,24,24,24,0	0,24,0,56,24,24,60,0
J -	0,0,12,12,12,12,60,0	0,6,0,6,6,6,38,28
K -	0,0,100,100,126,70,70,0	0,48,44,56,56,44,38,0
L -	0,0,96,96,96,96,126,0	0,24,24,24,24,24,14,0
M -	0,0,126,86,86,86,86,0	0,0,108,86,86,86,86,0
N -	0,0,126,70,70,70,70,0	0,0,124,70,70,70,70,0
O -	0,0,126,98,98,98,126,0	0,0,60,70,70,70,60,0
P -	0,0,126,98,126,96,96,0	0,0,124,70,70,124,96,96
Q -	0,0,124,100,100,100,126,0	0,0,662,70,70,62,6,6
R -	0,0,126,98,124,70,70,0	0,0,30,48,48,48,48,0
S -	0,0,126,96,126,6,126,0	0,0,60,96,60,6,124,0
T -	0,0,126,24,24,24,24,0	0,24,60,24,24,24,14,0
U -	0,0,98,98,98,98,126,0	0,0,70,70,70,70,60,0
V -	0,0,98,98,98,52,24,0	0,0,70,70,44,44,24,0
W -	0,0,106,106,106,106,126,0	0,0,70,86,86,86,44,0
X -	0,0,98,98,60,70,70,0	0,0,70,44,24,44,70,0
Y -	0,0,98,98,126,24,24,0	0,0,70,70,70,62,6,60
Z -	0,0,126,6,24,96,126,0	0,0,126,12,24,48,126,0

	Nagybetűk	Kisbetűk
A -	24,60,102,126,102,102,102,0	0,0,60,6,62,102,62,0
B -	124,102,102,124,102,102,124,0	0,96,124,102,102,124,0
C -	60,102,96,96,96,102,60,0	0,0,60,96,96,96,60,0
D -	120,108,102,102,102,108,120,0	0,6,6,62,102,102,62,0
E -	126,96,96,120,96,96,126,0	0,0,60,102,126,96,60,0
F -	126,96,96,126,96,96,96,0	0,14,24,62,24,24,24,0
G -	60,102,96,110,102,102,60,0	0,0,62,102,102,62,6,124
H -	102,102,102,126,102,102,102,0	0,96,96,124,102,102,102,0
I -	60,24,24,24,24,24,60,0	0,24,0,56,24,24,60,0
J -	30,12,12,12,12,108,56,0	0,6,0,6,6,6,6,60
K -	102,108,120,112,120,108,102,0	0,96,96,108,120,108,102,0
L -	96,96,96,96,96,96,126,0	0,56,24,24,24,24,60,0
M -	99,119,127,107,99,99,99,0	0,0,102,127,127,107,99,0
N -	102,118,126,126,110,102,102,0	0,0,124,102,102,102,102,0
O -	60,102,102,102,102,102,60,0	0,0,66,102,102,102,60,0
P -	124,102,102,124,96,96,96,0	0,0,124,102,102,124,96,96
Q -	60,102,102,102,102,60,14,0	0,0,62,102,102,62,6,6
R -	124,102,102,124,120,108,102,0	0,0,124,102,96,96,96,0
S -	60,102,96,60,6,102,60,0	0,0,62,96,60,6,124,0
T -	126,24,24,24,24,24,24,0	0,24,126,24,24,24,14,0
U -	102,102,102,102,102,102,60,0	0,0,102,102,102,102,62,0
V -	102,102,102,102,102,60,24,0	0,0,102,102,102,60,24,0
W -	99,99,99,107,127,119,99,0	0,0,99,107,127,62,54,0
X -	102,102,60,24,60,102,102,0	0,0,102,60,24,60,102,0
Y -	102,102,102,60,24,24,24,0	0,0,102,102,102,62,12,120
Z -	126,6,12,24,48,96,126,0	0,0,126,12,24,48,126,0
	---	---
A -	60,102,102,126,102,102,102,0	0,0,60,6,62,102,62,0
B -	124,102,102,124,102,102,124,0	96,96,124,102,102,102,124,0
C -	60,102,96,96,96,102,60,0	0,0,60,102,96,102,60,0
D -	120,108,102,102,102,108,120,0	6,6,62,102,102,102,62,0
E -	126,96,96,124,96,96,126,0	0,0,60,102,126,96,60,0
F -	126,96,96,124,96,96,96,0	28,48,48,124,48,48,48,0
G -	60,102,96,110,102,102,60,0	0,0,62,102,102,62,6,60
H -	102,102,102,126,102,102,102,0	96,96,124,102,102,102,102,0
I -	126,24,24,24,24,24,126,0	24,0,56,24,24,24,60,0
J -	662,12,12,12,12,108,56,0	24,0,56,24,24,24,24,112
K -	102,108,120,112,120,108,102,0	96,96,102,108,120,108,102,0
L -	96,96,96,96,96,96,126,0	56,24,24,24,24,24,60,0
M -	99,119,127,107,107,99,99,0	0,0,62,96,60,6,124,0
N -	102,102,118,126,110,102,102,0	0,0,124,102,102,102,102,0
O -	60,102,102,102,102,102,60,0	0,0,60,102,102,102,60,0
P -	124,102,102,124,96,96,96,0	0,0,124,102,102,124,96,96
Q -	60,102,102,102,108,54,0	0,0,62,102,102,62,6,7
R -	124,102,102,124,108,102,102,0	0,0,108,118,96,96,96,0
S -	60,102,96,60,6,102,60,0	0,0,62,96,60,6,124,0
T -	126,24,24,24,24,24,24,0	48,48,124,48,48,48,48,0
U -	102.102,102,102,102,102,60,0	0,0,102,102,102,102,62,0
V -	102,102,102,102,102,60,24,0	0,0,102,102,102,60,24,0
W -	99,99,107,107,127,119,99,0	0,0,99,107,107,127,54,0
X -	102,102,60,24,60,102,102,0	0,0,102,60,24,60,102,0
Y -	102,102,102,60,24,24,24,0	0,0,102,102,102,62,6,60
Z -	126,6,12,24,48,96,126,0	0,0,126,12,24,48,126,0

	Nagybetűk	Kisbetűk
A	24,60,102,102,126,102,102,0	0,0,120,12,124,204,118,0
B	252,102,102,124,102,102,252,0	224,96,124,102,102,102,220,0
C	60,102,192,192,192,102,60,0	0,0,60,102,96,102,60,0
D	248,108,102,102,102,108,248,0	28,12,124,204,204,204,118,0
E	254,98,104,120,104,98,254,0	0,0,60,102,126,96,60,0
F	254,98,104,120,104,96,240,0	28,54,48,120,48,48,120,0
G	60,102,192,192,206,102,62,0	0,0,62,102,102,62,6,124
H	102,102,102,126,102,102,102,0	224,96,108,118,102,102,230,0
I	126,24,24,24,24,24,126,0	24,0,56,24,24,24,60,0
J	30,12,12,12,204,204,120,0	6,0,14,6,6,102,102,60
K	230,102,108,120,108,102,230,0	224,96,102,108,120,108,230,0
L	240,96,96,96,98,102,254,0	56,24,24,24,24,24,60,0
M	198,238,254,254,214,198,198,0	0,0,108,254,214,214,198,0
N	198,230,246,222,206,198,198,0	0,0,220,102,102,102,102,0
O	56,108,198,198,198,108,56,0	0,0,60,102,102,102,60,0
P	252,102,102,124,96,96,240,0	0,0,220,102,102,124,96,240
Q	56,108,198,198,218,204,118,0	0,0,118,204,204,124,12,30
R	252,102,102,124,108,102,230,0	0,0,220,118,96,96,240,0
S	60,102,96,60,6,102,60,0	0,0,60,96,60,6,124,0
T	126,90,24,24,24,24,60,0	48,48,124,48,48,54,28,0
U	102,102,102,102,102,102,60,0	0,0,102,102,102,102,62,0
V	102,102,102,102,102,60,24,0	0,0,102,102,102,60,24,0
W	198,198,198,214,254,238,198,0	0,0,198,214,214,254,108,0
X	198,108,56,56,108,198,198,0	0,0,198,108,56,108,198,0
Y	102,102,102,60,24,24,60,0	0,0,102,102,102,62,6,124
Z	254,198,140,24,50,102,254,0	0,0,126,12,24,48,126,0
	---	---
A	12,22,38,38,62,166,67,0	0,0,29,34,100,191,0,0
B	60,82,62,51,51,179,110,0	14,18,36,58,98,189,0,0
C	92,180,120,96,96,99,60,0	0,0,28,36,96,191,0,0
D	124,150,115,51,51,166,124,0	1,2,60,72,200,119,0,0
E	92,52,24,60,96,99,60,0	0,0,28,36,120,159,0,0
F	113,158,112,60,48,176,96,0	15,25,18,60,248,87,144,224
G	156,114,124,96,102,62,70,60	0,0,31,50,212,63,72,112
H	164,103,102,254,102,102,195,0	7,9,18,60,228,71,0,0
I	44,28,12,12,12,45,30,0	4,0,8,16,48,223,0,0
J	22,30,22,6,38,71,60,0	2,0,12,56,216,127,144,240
K	178,114,60,50,50,179,67,0	14,18,28,52,124,167,0,0
L	76,188,88,24,24,113,62,0	6,9,18,20,56,223,0,0
M	66,70,110,118,86,215,66,0	0,0,84,170,170,87,0,0
N	35,50,58,46,38,162,67,0	0,0,44,90,182,39,0,0
O	158,115,99,99,99,98,60,0	0,0,28,38,101,184,0,0
P	126,179,115,62,48,176,96,0	0,0,20,58,246,103,64,192
Q	156,114,99,99,123,102,62,3	0,0,30,38,253,8,24,48
R	124,162,54,60,54,162,99,0	0,32,56,72,152,31,0,0
S	78,89,62,12,6,70,60,0	0,4,12,22,58,221,0,0
T	118,158,88,24,24,25,14,0	6,12,31,56,88,143,0,0
U	97,178,114,50,50,50,28,0	0,0,38,108,236,55,0,0
V	98,181,116,52,52,24,8,0	0,0,34,87,180,56,0,0
W	69,237,109,125,119,102,36,0	0,0,107,218,90,108,0,0
X	98,181,60,24,56,173,198,0	0,0,18,45,76,179,0,0
Y	98,182,54,54,30,70,70,60	0,0,9,50,220,63,72,112
Z	126,140,88,24,48,97,254,0	0,60,40,92,159,36,40,48

	Nagybetűk	Kisbetűk
A	120,172,36,68,124,68,116,154	0,0,56,68,52,76,100,58
B	124,162,82,212,82,210,98,156	0,32,32,36,52,36,36,88
C	40,84,144,144,144,162,68,56	0,0,16,40,64,64,72,48
D	124,166,106,106,74,82,98,156	0,48,24,4,52,68,68,56
E	40,84,144,156,144,162,68,56	0,0,56,68,72,64,64,56
F	28,40,108,168,40,40,176,64	0,24,16,56,16,16,16,40
G	24,52,80,156,146,170,68,56	0,0,40,68,68,52,2,124
H	104,168,80,212,90,210,98,154	0,32,32,44,50,34,34,100
I	96,168,80,80,208,80,80,176	0,16,0,48,16,16,16,24
J	120,140,52,20,52,20,104,144	0,8,0,8,8,8,40,24
K	112,160,36,114,44,104,68,178	0,64,32,32,54,34,44,38
L	116,168,40,104,40,104,48,94	0,8,16,16,16,16,16,24
M	52,74,74,218,74,74,74,194	0,0,106,62,42,42,42,42
N	48,88,204,68,212,68,72,150	0,0,104,36,36,36,38,36
O	24,52,106,162,162,170,148,120	0,0,28,36,36,36,36,24
P	104,164,42,98,50,108,160,48	0,72,36,36,36,44,32,80
Q	56,116,170,162,162,170,84,62	0,40,72,72,72,104,8,20
R	120,164,42,170,100,42,34,218	0,0,40,116,32,32,32,32
S	28,44,72,252,150,122,20,248	0,0,60,68,56,20,36,120
T	126,148,16,80,144,186,68,56	0,8,24,8,8,8,8,12
U	100,164,36,100,164,164,84,58	0,0,36,100,36,36,36,22
V	100,146,18,18,82,34,36,24	0,64,44,36,36,36,36,24
W	74,170,42,42,42,106,74,52	0,64,38,42,42,42,42,28
X	68,166,24,24,60,24,164,66	0,0,196,42,16,40,196,66
Y	196,82,210,82,196,88,128,124	0,0,44,36,36,20,8,112
Z	102,154,4,8,224,32,66,188	0,0,48,16,56,8,8,16
Ą	0,60,66,126,66,70,68,6	0,56,4,60,68,60,16,8
Â	8,20,60,66,126,66,66,0	8,20,56,4,60,68,60,0
Á	8,8,60,66,126,66,66,0	8,8,56,4,60,68,60,0
Ç	0,60,66,64,66,60,16,24	0,28,32,32,32,28,8,12
Č	20,8,60,66,64,66,60,0	20,8,28,32,32,32,28,0
Č	16,8,60,66,64,66,60,0	16,8,28,32,32,32,28,0
Ę	0,126,64,124,64,126,16,24	0,56,68,120,64,60,16,24
Ê	20,8,126,64,124,64,126,0	40,16,56,68,120,64,60,0
Ê	8,20,126,64,124,64,126,0	16,40,56,68,120,64,60,0
É	16,16,126,64,124,64,126,0	16,16,56,68,120,64,60,0
Î	8,20,28,8,8,8,28,0	16,40,0,48,16,16,56,0
Ł	0,32,32,48,32,96,62,0	0,16,16,24,16,48,12,0
Ň	20,74,98,82,74,70,66,0	40,16,120,68,68,68,68,0
Ø	1,62,70,74,82,97,124,128	0,2,60,76,84,100,120,128
Ô	8,20,60,66,66,66,60,0	16,40,0,56,68,68,56,0
Ó	8,8,60,66,66,66,60,0	16,16,0,56,68,68,56,0
Ö	20,0,60,66,66,66,60,0	0,40,0,56,68,68,56,0
Ő	20,20,60,66,66,66,60,0	40,40,0,56,68,68,56,0
Ř	40,16,124,66,124,72,68,0	20,8,28,32,32,32,32,0
Š	20,8,62,64,60,2,124,0	40,16,56,64,56,4,56,0
Ş	48,72,80,72,72,80,64,64	0,24,36,40,36,40,32,32
Ü	8,86,74,66,66,66,60,0	16,40,16,68,68,68,56,0
Ú	8,74,66,66,66,66,60,0	16,16,68,68,68,68,56,0
Ü	36,0,66,66,66,66,60,0	0,40,0,68,68,68,56,0
Ű	36,36,66,66,66,66,60,0	0,40,40,68,68,68,56,0
Ż	24,24,126,12,24,48,126,0	16,40,16,124,24,48,124,0
Ž	20,8,126,12,24,48,126,0	0,40,16,124,24,48,124,0

B.FÜGGELÉK-Karakterkészletek CIRILL és GÖRÖG

	Nagybetűk	Kisbetűk
А	0,60,66,66,126,66,66,0	0,0,56,4,60,68,60,0
Б	0,124,64,124,66,66,124,0	1,6,4,28,36,36,24,0
В	0,124,66,124,66,66,124,0	0,56,36,60,34,34,60,0
Г	0,60,32,32,32,32,32,0	0,0,24,36,32,32,96,0
Д	0,30,18,34,34,66,126,66	0,0,60,68,68,60,4,76
Е	0,62,64,124,64,64,62,0	0,0,56,68,120,64,60,0
Ё	36,0,62,64,124,64,62,0	40,0,56,68,124,64,60,0
Ж	0,146,84,124,84,84,146,0	0,0,73,42,62,42,73,0
З	0,62,2,4,2,34,28,0	0,0,62,4,12,34,28,0
И	0,66,70,74,82,98,66,0	0,0,68,68,68,68,56,0
Й	36,90,70,74,82,98,66,0	40,16,68,68,68,68,56,0
К	0,68,72,112,72,68,66,0	0,32,40,48,48,40,36,0
Л	0,14,18,18,34,34,66,0	0,0,2,6,10,18,34,0
М	0,10,30,42,42,74,74,0	0,0,10,30,42,42,74,0
Н	0,66,66,126,66,66,66,0	0,0,68,68,124,68,68,0
О	0,60,66,66,66,66,60,0	0,0,56,68,68,68,56,0
П	0,126,68,68,68,68,68,0	0,0,120,68,68,68,68,0
Р	0,124,66,66,124,64,64,0	0,0,120,68,68,120,64,64
С	0,60,64,64,64,64,60,0	0,0,28,32,32,32,28,0
Т	0,254,146,146,146,146,146,0	0,0,104,84,84,84,84,0
У	0,66,66,66,62,2,124,0	0,0,34,34,34,30,2,60
Ф	18,124,146,146,146,146,124,16	0,0,28,42,42,42,28,8
Х	0,66,36,24,24,36,66,0	0,0,34,20,8,20,34,0
Ц	0,68,68,68,68,68,126,2	0,0,36,36,36,36,30,2
Ч	0,68,68,68,60,4,6,0	0,0,36,36,28,4,6,0
Ш	0,146,146,146,146,146,254,0	0,0,42,42,42,42,30,0
Щ	0,146,146,146,146,146,127,1	0,0,84,84,84,84,62,2
Ъ	0,98,16,28,18,18,28,0	0,0,48,16,28,18,28,0
Ы	0,66,66,90,102,66,63,1	0,0,36,36,60,36,30,2
Ь	0,32,32,56,36,36,56,0	0,0,32,32,56,36,56,0
Э	0,124,2,62,2,2,124,0	0,0,120,4,60,4,120,0
Ю	0,76,82,114,82,82,76,0	0,0,76,82,114,82,76,0
Я	0,62,66,66,62,34,66,0	0,0,30,34,30,18,34,0
---	---	---
A	0,60,66,66,126,66,66,0	0,0,2,116,136,136,116,2
B	0,124,66,124,66,66,124,0	0,56,68,88,68,88,64,64
Γ	0,126,66,64,64,64,64,0	0,74,180,48,72,72,72,48
Δ	16,40,40,68,68,130,254,0	5,6,4,60,68,68,60,0
E	0,126,64,124,64,64,126,0	0,0,56,68,48,68,56,0
Z	0,126,4,8,16,32,126,0	56,68,48,64,64,56,4,120
H	0,66,66,126,66,66,66,0	0,80,168,168,40,8,10,4
Θ	124,130,130,186,130,130,124,0	16,40,42,28,72,168,168,16
I	0,28,8,8,8,8,28,0	0,0,0,48,16,16,56,0
K	0,68,72,112,72,68,66,0	0,0,66,44,52,100,102,0
Λ	16,16,16,40,40,68,238,0	16,8,8,12,20,36,66,0
M	0,66,102,90,66,66,66,0	0,18,18,18,47,32,64,128
N	0,66,98,82,74,70,66,0	0,0,4,66,36,40,48,0
Ξ	254,130,0,124,0,130,254,0	12,48,12,48,12,2,124,0
O	0,60,66,66,66,66,60,0	0,0,56,68,68,68,56,0
Π	0,126,68,68,68,68,68,0	0,0,64,62,36,36,70,0
P	0,124,66,66,124,64,64,0	0,56,68,68,56,16,8,112
Σ	0,126,32,16,16,32,126,0	4,24,32,60,34,34,60,0
T	0,254,16,16,16,16,16,0	0,0,4,56,80,20,8,0
Y	0,130,68,40,16,16,16,0	0,8,4,68,168,48,32,64
Φ	124,16,56,84,56,16,124,0	0,76,82,82,60,16,16,0
X	0,66,36,24,24,36,66,0	0,0,100,24,16,48,76,0
Ψ	0,146,146,84,56,16,16,0	0,0,86,84,56,16,16,0
Ω	0,124,130,130,198,40,238,0	0,0,0,68,130,146,108,0

ISBN 963 592 5689

Felelős kiadó: dr. Kovács Magda
Engedélyszám: 48610
Készült: 3000 példányban
Szabadság MGTSZ Nyomdaüzeme, Gyál – 86-54/sz.
Felelős vezető: Tóth Antal

Ára: 149,– Ft