PEDAGÓGIAI MŰHELY

19.

Gudenus László

FEJEZETEK A COMMODORE PLUS/4-ES GÉPI KÓDÚ PROGRAMOZÁSÁBÓL

Pest Megyei Pedagógiai Intézet

Budapest, 1990

Gudenus László

Fejezetek a Commodore +4 gépi kódú programozásából

Pest Megyei Pedagógiai Intézet

Budapest, 1990.

E szakköri füzet a PMPI Bevezetés a COMMODORE+4 gépi kódú programozásába című kiadványának folytatása, s mint ilyen a 65xx mikroprocesszor utasításkészletét már ismerő felhasználók számára ad segítséget a gépi kódú programozás technikájának elsajátításához.

Lektorálta: Sok Zoltán

ISSN 0238-3217
ISNB 963 04 0431 1

Kiadja a Pest Megyei Pedagógiai Intézet
Felelős kiadó: Dr. Szivi József igazgató
Felelős szerkesztő: Pap János
Készült a PMPI sokszorosítójában
Felelős vezető: Márkus István
Munkaszám: 55/1990

TARTALOMJEGYZÉK

	LEMEZ		SZÖVEG	LISTA
	"hypra 1+4 3056"	prg		
	"+4sys"	prg		
Bevezetés			4	
1.	"bas.titkos"	prg	6	F1
	"titkos"	seq		F2
	"titkos.mod"	prg		
2.	"bas.monitor"	prg	14	F5
	"monitor"	seq		F6
	"monitor.mod"	prg		
3.	"bas.tukrozes"	prg	24	F11
	"tukrozes"	seq		F12
	"tukrozes.mod"	prg		
4.	"bas.prim"	prg	31	F18
	"prim"	seq		F19
	"prim.mod"	prg		
5.	"bas.ketkepernyo"	prg	37	F23
	"demo"	prg		F23
	"ketkep"	seq		F26
	"ketkep.mod"	prg		
6.	"bas.skala"	prg	49	F31
	"skala"	seq		F32
	"skala.mod"	prg		
7.	"bas.hangseb"	prg	54	F34
	"hangseb"	seq		F36
	"hangseb.mod"	prg		
8.	"bas.pulzal"	prg	59	F39
	"pulzal"	seq		F40
	"pulzal.mod"	prg		
9.	"bas.sorok"	prg	64	F44
	"sorok"	seq		F45
	"sorok.mod"	prg		
	"karakterek"	prg		
10.	"bas.gvetites"	prg	67	F47
	"gvetites"	seq		F50
	"gvetites.mod"	prg		
11.	"bas.kvetites"	prg	70	F52
	"kvetites"	seq		F55
	"kvetites.mod"	prg		
12.	"bas.fenyujsag"	prg	74	F57
	"feny1"	seq		F59
	"feny1.mod"	prg		
	"feny2"	seq		F61
	"feny2.mod"	prg		
	"feny3"	seq		F64
	"feny3.mod"	prg		
Befejezés			86	

BEVEZETÉS

A szakköri füzet a PMPI Bevezetés a COMMODORE+4 gépi kódú programozásába c. kiadványa folytatása, s mint ilyen, a 65xx mikroprocesszor utasításkészletét már ismerő felhasználók számára ad segitséget a gépi kódú programozás technikájának elsajátításához, a C+4 lehetőségeinek megismeréséhez. Mind egyéni tanuláshoz, mind csoportos feldolgozáshoz ajánlható.
A gépi kódú programozás elsajátításának leghatékonyabb, mondhatni kizárólagos útja a feladatmegoldás, a gyakorlás. A szakköri füzet ehhez biztosit olyan kidolgozott feladatokat, amelyek egyrészt alkalmasak a gépi kódú programozás sajátosságainak bemutatására, másrészt viszont látványosak, érdekesek, szórakoztatóak is. A példák többségükben a grafikához kapcsolódnak, de szerepelnek a hanggal, az aritmetikával összefüggő programok, sőt, van egy mérési feladat is. A perifériák kezelésével azért nem foglalkoztunk, mert Babán Gábor és Masa István : Gépi kódú programozás kezdőknek és haladóknak C16 és PLUS/4 számitógépre c. könyve [I] ezt igen alaposan tárgyalja. Az egyes témák:

- titkosirás (táblázattal ill. a kizáró vagy művelettel)
- a beépitett monitor funkcióinak bővitése
- a grafikus képernyő tengelyes ill. középpontos tükrözése, különböző logikai műveletekkel
- négy byte-os egészek oszthatósági vizsgálata
- két grafikus képernyő megvalósitása
- zene megszakitásból
- hangsebesség mérése
- pulzáló, villogó, forgó stb. karakterek előállitása
- soronként más-más karakterkészlet megjelenitése
- a tár egy-egy területének a grafikus ill. a karakteres képernyőre vetitése
- fényújság megvalósitása (különböző szinteken).

Minden fejezetben gondot forditunk a gépi kódú program környezetére: mindenképpen mellékelünk betöltő, de többször bemutató programot is. Alkalmazzuk a BASIC-kel való kapcsolat mindhárom formáját, az USR függvényt, a SYS utasitást, valamint új utasitások bevezetését (BASIC-bővités, beszúrás).

Példát mutatunk szöveg-, egy és két byte-os egész, valamint négy byte-os egésszé konvertálható valós paraméterek átvételére, változó számban is.

A szakköri füzetben mintegy száz kitűzött feladat is szerepel; ezek részben a megoldott programok módosítását, részben pedig önálló programok készítését célozzák.

Az algoritmusok leírásánál általában Szlávi Péter és Zsakó László : Módszeres programozás című könyvét [II] követtük. Hivatkozunk még Erdős Iván : Commodore PLUS/4, C-16, C-116 ROM-lista című könyvére [III] is.

1. TITKOSÍRÁS

Elsőként a titkosírással foglalkozunk, amely viszonylag jelentéktelen feladat, vizsgálata mégis sok tanulságot szolgáltat a gépi kódú programozás módszereiről. Megértéséhez szükséges [I] 2.7 és 2.8 fejezetének az ismerete.

Minden titkosírás valamely, a titkosítandó karakterkészleten, mint értelmezési tartományon definiált, invertálható F függvényen alapul. A titkosírás kódoláskor F-nek a szövegen felvett értékét, dekódoláskor pedig F inverzének (G-nek) a titkosított szövegen felvett értékét adja vissza. Annyi titkosírás van, ahány F függvény definiálható a fenti feltételeknek megfelelően. Ebben a fejezetben két, lényegében különböző eljárást mutatunk be.

TÁBLÁZATON ALAPULÓ TITKOSÍRÁS

Az első módszer egy rögzített táblázaton alapul:

kar: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
F(kar): Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

(Mint látható, az egyszerűség kedvéért a betűknek a billentyűzeten elfoglalt helye alapján épitettük fel a táblázatot. Természetesen bármely olyan másik táblázat is megfelelne, amelynek alsó sorában minden betű pontosan egyszer szerepel.) A táblázat segitségével kódoláskor minden betűt az alatta lévővel, dekódoláskor pedig a felette lévővel helyettesitünk (a betűktől különböző karakterek - pld. a space - mindkét transzformációnál változatlanok maradnak).

Már itt megjegyezzük, hogy a gépben nem pontosan ezt a táblázatot fogjuk tárolni: mivel a felső sorban a 65...90 kódú karakterek szerepelnek, mégpedig növekvő sorrendben, tárolásuk feleslegesen foglalná a helyet. A tárba tehát csak az alsó sor kerül be; kezdetét TÁBLA jelöli.

A kódolandó szöveg tárba vitelének módjával egyelőre ne foglalkozzunk (ezt egyébként a SZÖVBE rutin végzi majd), hanem tételezzük fel, hogy a szöveg a tárban van, kezdetét az SZMUT (nullalapos) címen lévő mutató, hosszát pedig az SZH érték tárolja. A kódolás és a dekódolás programját a következő algoritmusok alapján irjuk meg (a lényegen nem változtat, de programozni egyszerűbb, ha a szöveg karaktereit

fordított sorrendben, a végétől az elejéig vesszük sorra):

```
TODA [táblázattal oda]
   Y:=SZH-1 [a SZÖVEG hossza]                          (*)
   Ciklus, amíg Y>=0
      Ha 'A'<=(SZMUT),Y és (SZMUT),Y<='Z',
         akkor X:=ASC((SZMUT),Y)-65                    (**)
               (SZMUT),Y:=TÁBLA,X
      Elágazás vége
      Y:=Y-1
   Ciklus vége
Eljárás vége
```

Az algoritmus a szöveget a tárba, eredeti helyére úgy tölti vissza, hogy a betűktől különböző karaktereket változatlanul hagyja, de minden betűt kicserél a táblázatban alatta lévőre. A (*) sorra azért van szükség, mert bár SZH a szöveg hosszát jelöli, a szöveg a tárban az (SZMUT)+0, (SZMUT)+1, ..., (SZMUT)+SZH-1 byte-okon helyezkedik el. Ha csak "túl sok" szövegváltozónk nincs, ezek a byte-ok $8000 fölött vannak, így elérésükhöz ROM-RAM lapozásra van szükség. A (**) sor kódolásánál ne feledkezzünk meg erről! Kódolva:

```
ram=$ff3f           ;ram-ra kapcsol
rom=$ff3e           ;rom-ra kapcsol
toda                ;táblázattal oda
       jsr szovbe   ;szöveg beolvasása
       ldy szh      ;szövegmutató
       dey
       sei
       sta ram      ;ram-ra kapcsol
ocikl  lda (szmut),y
       cmp #$41     ;<a?
       bmi okesz    ;igen
       cmp #$5b     ;>z?
       bpl okesz    ;igen
       sec
       sbc #$41     ;hányadik betű?
       tax          ;indexbe
       lda tabla,x
okesz  sta (szmut),y
       dey          ;szöveg vége?
       bpl ocikl    ;nem
       sta rom      ;rom-ra kapcsol
       cli
       rts
```

Figyeljük meg a gépi kódú programozás jellegzetességeit: a csökkenő ciklusváltozót tartalmazó ciklusokat, az indirekt ill. indexelt címzéseket, a táblázat használatát, összehasonlításnál a jelzőbitek állásától függő elágazást!
Itt jegyezzük meg azt is, hogy a ROM mögötti RAM-ban lévő táblázathoz (a szöveg is egy ilyen táblázat!) az itt alkalmazott (nullalapos) indirekt indexelt címzéssel a legegysze-

rübb hozzáférni.

A dekódolás algoritmusa:

```
TVISZ  [táblázattal vissza]
   Y:=SZH-1
   Ciklus, amíg Y>=0
      Ha 'A'<=(SZMUT),Y és (SZMUT),Y<='Z',
         akkor
            X:=25 [betűk száma-1]
            A:=(SZMUT),Y
            Ciklus, amíg A<>TÁBLA,X                    (*)
               X:=X-1
            Ciklus vége
               [X: hányadik betű az abc-ben]
            (SZMUT),Y:=X+65 [a betű kódja]
      Elágazás vége
      Y:=Y-1
   Ciklus vége
Eljárás vége
```

A kódolt szubrutin:

```
tvisz                ;táblázattal vissza
       jsr szovbe    ;szöveg beolvasása
       ldy szh       ;szövegmutató
       dey
       sei
       sta ram       ;ram-ra kapcs.
vcikl  lda (szmut),y
       cmp #$41      ;<a?
       bmi vkesz     ;igen
       cmp #$5b      ;>z?
       bpl vkesz     ;igen
       ldx #$19      ;betűk száma
xcik   cmp tabla,x
       beq megvan    ;igen
       dex           ;nincs
       bpl xcik      ;tovább
megvan txa           ;index át
       clc
       adc #$41      ;ennyiedik betű
vkesz  sta (szmut),y
       dey           ;szöveg vége?
       bpl vcikl     ;nem
       sta rom       ;rom-ra kapcs.
       cli
       rts
```

Újabb gépi kódú programozási jellegzetességgel találkoztunk: a táblázatban való kereséssel. Nem túl gyakori, ilyenkor mégis jellemző megoldás a ciklus közepéről való kiugrás (általában végfeltételes ciklusokat alkalmazunk).

Eddig olyan módszerrel foglalkoztunk, amelynél egy adott karakternek mindig ugyanaz a karakter felel meg, viszonylag egyszerű tehát a titkosírás dekódolása, csupán néhány alapvető törvényszerüséget (pld. a betűgyakoriságot) kell figyelembe venni. Jóval bonyolultabb azoknak a titkosírásoknak a

megfejtése, amelyekben nem állandó kulcs szerint történik a kódolás. További előny az, ha a kódolást és a dekódolást meghatározó függvény azonos (azaz: saját inverze), mert így ugyanazon algoritmus III. program segítségével végezhetjük mindkét műveletet. Erre a célra optimális az adott számmal bitenként végzett "kizáró vagy" (EOR, XOR) logikai művelet, amelyre köztudottan érvényes a következő azonosság:

(A EOR B) EOR B = A ,

tehát: ha ismertek az A és a B számok, akkor az A-nak B-vel való kódolása után kapott értéket B-vel ismételten kódolva visszakapjuk az eredeti A értéket. A kódolás (s ezen mostantól akár a dekódolást is érthetjük) módja a következő: rögzítünk egy 1 byte-os B számot, majd a szöveg minden A karakterét (,amelynek kódja asc(A)) rendre helyettesítjük az (asc(A) EOR B) kódú karakterrel. Még jobb a módszer, ha B értékét - természetesen megadott rendszer szerint - karakterről karakterre változtatjuk. Amennyiben a B értékek éppen egy szöveg karaktereinek ASCII kódjai, megjegyzésük is könnyebb. Mondhatjuk ezt úgy is, hogy az egyik szöveget a másik - mint kód - szerint titkosítottuk. Ezzel el is érkeztünk a másik módszer lényegéhez.

TITKOSÍRÁS VÁLTOZTATHATÓ KULCSSZÓVAL

Egyszerű a dolgunk abban az esetben, ha a kódolandó szöveg és a kulcsszó (ami természetesen akár több szó is lehet; nevezzük is inkább a továbbiakban kódnak) azonos hosszúságú, vagy a kód hosszabb, mint a szöveg. Baj van azonban akkor, ha a kód a rövidebb, mert karaktereit egyenként igénybe véve kevesebb marad, így a szöveg nem minden karakterét tudjuk titkosítani. Segít azonban a problémán, ha megállapodunk abban, hogy a kód karaktereit - amíg csak szükséges - ciklikusan ismételve többször is felhasználhatjuk. Megtehetjük azt is, hogy a szöveg karaktereit (mint az előbb) fordított sorrendben, a kód karaktereit viszont eredeti sorrendjükben vesszük igénybe. Ezt valósítja meg a következő algoritmus (az előző két algoritmushoz hasonlóan feltételezzük, hogy SZÖVEG és KÓD már a tárban vannak, SZMUT és KMUT mutatnak a kezdetükre, hosszuk pedig SZH ill. KH):

```
KÓDDAL [SZÖVEG titkosítása KÓD-dal, ciklikus ismétléssel]
   X:=0 [mutató: hol tartunk a KÓD-ban]
   Y:=SZH-1 [mutató: hol tartunk a SZÖVEG-ben]
   Ciklus, amíg Y>=0
      (SZMUT),Y:=(SZMUT),Y EOR (KMUT),X          (*)
      X:=X+1
      Ha X>KH, akkor X:=0
      Y:=Y-1
   Ciklus vége
Eljárás vége
```

A (*) sor kódolásakor több bonyodalom is adódik: nem csupán a ROM-RAM átkapcsolásra kell ügyelnünk, hanem arra is, hogy az indirekt indexelt címzés csak az Y-regiszterrel alkalmazható, ezt viszont a szövegpozíció mutatójaként már használjuk. Megoldást jelentene pld. egy külön mutató-byte beiktatása (erre a későbbiekben látunk majd példát; ez hasonló az itt alkalmazott SZBYTE-megoldáshoz, ahová a szöveg aktuális byte-ját mentjük ki), most azonban a vermet használó módszert választottuk, mivel ez is igen gyakori fogás a gépi kódú programozásnál.

A táblázatos megoldással szemben itt nem csak a betűket, hanem az összes karaktert titkosítottuk. Ennek elég furcsa következményei lehetnek akkor, ha az EOR-művelet egy "látható" karaktert ún. "vezérlő" karakterré alakít, mert kiíráskor ez soremelést, a kép letörlését, a kurzor "eltűnését" ill. elmozdulását, valamely ESC funkció életbe lépését stb. okozhatja. Ennek elkerülése érdekében egy újabb, "kompenzáló" EOR műveletet iktattunk be, amelynek hatására az átalakítás után minden betű képe - ha nem is betű, de legalább - "látható" karakter lesz. Ennek beiktatása - éppen a fentebb idézett azonosság miatt - nem jelent problémát, mert a dekódolásnál saját hatását semlegesíti. Nézzük a kódolt programot:

```
koddal
       ldx #$00       ;kódmutató
       ldy szh        ;szövegmutató
       dey
       sei
       sta ram        ;ram-ra kapcsol
cikl   lda (szmut),y  ;szöveg köv.
       sta szbyte     ;byte-ja
       tya            ;szövegmutató
       pha            ;a verembe
       txa            ;kódmutató
       tay            ;y-ba
       lda (kmut),y   ;kód köv. byte-ja
       inx            ;kódmut. növelése
       cpx kh         ;vége?
       bne xugr       ;nem, mehet
```

```
        ldx #$00        ;igen, elölről
xugr    eor szbyte      ;kódolás
        eor #$40        ;kompenzálás
        sta szbyte      ;tárolás
        pla             ;szövegmutató
        tay             ;a veremből
        lda szbyte      ;vissza a
        sta (szmut),y   ;helyére
        dey             ;szöveg vége?
        bpl cikl        ;nem, vissza
        sta rom         ;rom-ra kapcsol
        cli
        rts
```

Tulajdonképpen már készen vagyunk, csupán az "összeszerelés", a vezérlés és a paraméterátadás megoldása vár ránk.
Most a SYS utasitással történő hivást választottuk, bár csaknem ilyen egyszerüen megoldható a feladat az USR függvénnyel is; erre a következőkben láthatunk példát.
A vezérlésnek három funkció közül kell a megfelelőt kiválasztania. A "kóddal"-hoz két, a "toda" ill. a "tvisz" rutinokhoz egy-egy szövegparaméter beolvasásáról kell gondoskodnia, azaz még a szöveg beolvasása előtt szükség van annak jelzésére, hogy melyik funkciót akarjuk használni. Megoldás az is, hogy más-más belépési pontot irunk a SYS után, egyszerübb azonban az, hogy az egyes funkciókat "indexeljük", például a következőképpen:

koddal - 0
toda - 1
tvisz - 2 ,

s rögtön a hivás után ezt az indexet kérjük be, majd ennek értékétől függően beolvasunk egy vagy két szövegparamétert, végül a megfelelő rutinra ugrunk.
Első pillantásra ijesztő nagy munka; az is volna, ha nem használhatnánk a BASIC-interpreter e célokra készült rutinjait. Szerencsére ezeket minden további nélkül igénybe vehetjük, csupán a velük való kommunikációra kell ügyelnünk. Ehhez a következőket kell tudnunk:
A CHKOMA jelü rutin (cime: $9491) a BASIC-szövegben a következő vesszőre pozicionál.
A GETBYT rutin (cime: $9d84) a BASIC-szövegben következő, egy byte-os egész szám értékét olvassa az X regiszterbe.
A STRFLG (cime: $0d) rendszerváltozó jelzi, hogy szöveg vagy szám tipusú-e az aktuális változó (szövegnél $FF, számnál 0 az értéke).

A FRESTR ($9c48) rutin (amennyiben STRFLG értéke $FF ,) beolvassa az aktuális szövegváltozó értékét a memóriába; hosszát (1 byte) és címét (2 byte) leteszi a $47-től (VARPTR) kezdődő 3 byte-ra.

Ezek alapján már szinte gyerekjáték a program keretének összeállítása:

```
varptr=$47              ;változó mutatója
chkoma=$9491            ;vesszőig előre
strflg=$0d              ;stringjelző
frestr=$9c48            ;szöveges vált. beolv.
getbyt=$9d84            ;1 byte x-be
                        ;
                        ;
        jsr chkoma      ;vesszőig előre
        jsr getbyt      ;1 byte x-be olvasása
        cpx #$01        ;táblázattal oda?
        beq toda        ;igen
        cpx #$02        ;táblázattal vissza?
        beq tvisz       ;igen
                        ;egyébként kóddal
        jsr szovbe      ;szöveg beolvasása
        jsr chkoma      ;vesszőig
        lda #$ff        ;string jön
        sta strflg
        jsr frestr      ;kód-
        ldy #$02        ;paraméter
kolv    lda (varptr),y;beolv.
        sta kh,y        ;mutatók
        dey             ;másolása
        bpl kolv

  .
  .   ld. koddal
  .
toda                    ;táblázattal oda
  .
  .   ld. toda
  .
tvisz                   ;táblázattal vissza
  .
  .   ld. tvisz
  .
      ;
szovbe                  ;szöveg beolvasása
        jsr chkoma      ;vesszőig
        lda #$ff        ;string jön
        sta strflg
        jsr frestr      ;szöveg-
        ldy #$02        ;paraméter
szolv   lda (varptr),y;beolv.
        sta szh,y       ;mutatók
        dey             ;másolása
        bpl szolv
        rts
```

teljes lista a Függelék F1-F4 oldalán található; az assemler sajnálatos módon úgy nyomtatott, hogy a TABLA-ban saját agát felülírta. A TABLA helyesen e fejezet elején szerepel.

érdések és feladatok az 1. fejezethez

.1 Bővitsük a TODA és a TVISZ eljárásokat oly módon, hogy . betükön kivüli jeleket is egy táblázat szerint titkosit-ák!

.2 Hol lehetne elhelyezni az OKESZ ill. a VKESZ címkéket igy, hogy a program hatékonyabb legyen? Miben nyilvánul meg ı hatékonyság?

..3 A TVISZ algoritmus (*) sorában hiányzik az "és X>=0" eltétel. Mi garantálja, hogy a ciklus befejeződik, mielőtt (értéke negativ lenne?

1.4 Készitsünk olyan táblázatot, amellyel a TODA elvégzi a TVISZ funkcióját (azaz: mint KODDAL esetében, egyazon eljá-ást hivjuk kódoláskor és dekódoláskor is)!

1.5 Keressünk a KODDAL szubrutinhoz más, a #$40-nél "jobb" compenzáló értéket (azaz: még több betünek betü legyen a képe is)!

1.6 Hogyan egyszerüsödik a KODDAL rész, ha egy rögzitett értékkel képezett EOR segitségével kódolunk ill. dekódolunk?

1.7 Irjuk át a KODDAL szubrutint úgy, hogy ne csak a SZÖVEG, hanem a KOD karaktereit is fogyó változójú ciklussal vegyük sorra!

1.8 A KODDAL szubrutin CIKL ciklusában nem csupán a SZÖVEG ill. a KÓD szövegváltozók egy-egy byte-ja elhozásának idejére kapcsoltuk ki a ROM-ot, hanem az egész ciklus futásának idejére. Miért nem okozott ez problémát?

1.9 A gép ROM-jában, a $CD89...$CDAA részen fellelhető titkosirásban az interpreter alkotóinak neve. Fejtsük meg!

1.10 A TEDMON disassembler-táblázata a $F83D...$F91E területen helyezkedik el, de ebben a formájában nem olvasható. Próbáljuk megfejteni!

2. MONITOR

Ebben a fejezetben olyan programot fogunk készíteni, amely megvalósít néhány, a TEDMON-ban nem szereplő monitor-funkciót. Példát mutatunk az előző fejezetben említett szubrutinhívásra (USR+változó számú paraméter átvétele) s mutatunk egy olyan, a gépi kódú programozásban gyakran előforduló módszert, amelyet vezérlési feladatok megoldásánál ill. kis híján azonos programrészek esetén a hellyel való takarékosság érdekében szoktunk alkalmazni: a programmal módosíttatjuk önmagát. A fejezet megértéséhez újabb előismeret nem szükséges.

A TEDMON hiányosságai között a legbántóbb az, hogy a "T" funkció (=transfer, T A B C: az A és B által meghatározott terület másolása C-től kezdve) csak igen komoly korlátozással működik helyesen: nem eshet a célintervallum kezdete a forrásintervallumba, mert különben az áthelyezéskor a két intervallum közös része elvész, ugyanis a program a forrásintervallumot "alulról felfelé" viszi át a célintervallumba. $8000 fölött csak kényelmetlen a RAM-ból ill. a ROM-ból való másolás (előtte mindig a $07f8-on lévő TEDMON olvasáskapcsolót kell ellenőriznünk vagy átállítanunk), az viszont már kifejezetten hátrányos, hogy a TEDMON minden $8000 fölötti címet, ahonnan olvasott, hexadecimálisan kiír a képernyőre, nagyon lassúvá téve a másolást.

Teljesen hiányzik a TEDMON-ból bizonyos funkciók megvalósítása. Így például jó lenne, ha egy intervallum valamely részintervallumát úgy tudnánk törölni, hogy ne maradjon "üres" a helye, hanem a mögötte lévő rész zárkózzon föl. Ugyancsak hasznos lenne a beszúrási funkció, amelynek segítségével egy intervallum belsejébe úgy tudnánk egy másik intervallumot átmásolni, hogy ez az eredeti intervallum "jobb oldali" részét - maga előtt tolva - pontosan annyival másolja följebb, hogy a helyére ő maga éppen beférjen. Ez így eléggé körülményesen hangzik, de gondoljunk ugyanezeknek a funkcióknak pld. egy szövegszerkesztő-programban való alkalmazására: egy betű, szó vagy akár egy egész mondat (vagy bármely összefüggő szövegrész) törlésekor ill. beszúrásakor

A szubrutin működéséhez három paraméter kell: az egyik forrás elejére (FE), a másik a forrás végére (FV), a harmadik pedig - attól függően, hogy lf vagy fl másolásra v szükség - a cél elejére vagy végére (CÉL) mutat; ezeken k vül - bár nem feltétlenül volna rá szükség - használjuk darab (DB) számlálót, amely a másolás kezdetén mindig FV-FE különbséget mutatja, minden byte átmásolásakor érték 1-gyel csökkentjük.

Feltételezzük tehát azt, hogy FE és FV értéke adott, CÉL célintervallum elejére mutat, s a RAM-ROM választást már b állitottuk. Az algoritmus:

```
MÁSOL
   DB:=FV-FE
   Ha CÉL<FE vagy CÉL>FV,akkor JÓ
                         különben ROSSZ
Eljárás vége
JÓ  [a CÉL az FE...FV intervallumon kivül van, lf]
   Y:=0
   Ciklus, amig DB>=0
      (CÉL),Y:=(FE),Y    [másol]
      FE:=FE+1           [tovább]
      CÉL:=CÉL+1
      DB:=DB-1                    (**)
   Ciklus vége
Eljárás vége
ROSSZ [a CÉL az FE...FV intervallumon belül van, fl]
   CÉL:=CÉL+DB                    (*)
   Y:=0
   Ciklus, amig DB>=0
      (CÉL),Y:=(FV),Y    [másol]
      FV:=FV-1           [tovább]
      CÉL:=CÉL-1
      DB:=DB-1                    (**)
   Ciklus vége
Eljárás vége
```

Ez az algoritmus különösebb magyarázatot nem igényel, talá csak a (*) sorról érdemes megjegyezni: ez gondoskodik arról hogy (mivel fl átvitel szükséges,) CÉL értékét a célinter vallum végére állitsa.

A kódolás előtt még vegyük észre, hogy a (**) sor mind a JÓ mind a ROSSZ esetben szerepel, igy érdemes külön szubrutin ként kódolnunk. Egyelőre irjuk meg úgy a kódot (egyébként i ez a "terjedelmesebb" eset), hogy a ROM mögül kell a forrás elhoznunk; rövidesen megmutatjuk, hogyan tudjuk ezt - szükségnek megfelelően - egyszerüen módositani. Az algorit mus kódolva:

A szubrutin működéséhez három paraméter kell: az egyik a forrás elejére (FE), a másik a forrás végére (FV), a harmadik pedig - attól függően, hogy lf vagy fl másolásra van szükség - a cél elejére vagy végére (CÉL) mutat; ezeken kívül - bár nem feltétlenül volna rá szükség - használjuk a darab (DB) számlálót, amely a másolás kezdetén mindig az FV-FE különbséget mutatja, minden byte átmásolásakor értékét 1-gyel csökkentjük.

Feltételezzük tehát azt, hogy FE és FV értéke adott, CÉL a célintervallum elejére mutat, s a RAM-ROM választást már beállitottuk. Az algoritmus:

```
MÁSOL
   DB:=FV-FE
   Ha CÉL<FE vagy CÉL>FV,akkor JÓ
                         különben ROSSZ
Eljárás vége
JÓ  [a CÉL az FE...FV intervallumon kivül van, lf]
   Y:=0
   Ciklus, amig DB>=0
      (CÉL),Y:=(FE),Y    [másol]
      FE:=FE+1           [tovább]
      CÉL:=CÉL+1
      DB:=DB-1                    (**)
   Ciklus vége
Eljárás vége
ROSSZ [a CÉL az FE...FV intervallumon belül van, fl]
   CÉL:=CÉL+DB                     (*)
   Y:=0
   Ciklus, amig DB>=0
      (CÉL),Y:=(FV),Y    [másol]
      FV:=FV-1           [tovább]
      CÉL:=CÉL-1
      DB:=DB-1                    (**)
   Ciklus vége
Eljárás vége
```

Ez az algoritmus különösebb magyarázatot nem igényel, talán csak a (*) sorról érdemes megjegyezni: ez gondoskodik arról, hogy (mivel fl átvitel szükséges,) CÉL értékét a célintervallum végére állitsa.

A kódolás előtt még vegyük észre, hogy a (**) sor mind a JÓ, mind a ROSSZ esetben szerepel, igy érdemes külön szubrutinként kódolnunk. Egyelőre irjuk meg úgy a kódot (egyébként is ez a "terjedelmesebb" eset), hogy a ROM mögül kell a forrást elhoznunk; rövidesen megmutatjuk, hogyan tudjuk ezt - a szükségnek megfelelően - egyszerűen módositani. Az algoritmus kódolva:

```
masol                   ;másolórutin
        sec
        lda fv          ;db:=fv-fe
        sbc fe
        sta db
        lda fv+1
        sbc fe+1
        sta db+1
        lda cel+1
        cmp fe+1        ;cél<fe?
        bmi jo          ;célhi<fehi
        bne felso       ;célhi>fehi
        lda cel
        cmp fe          ;célhi=fehi és
        bmi jo          ;céllo<felo
felso   lda fv+1
        cmp cel+1       ;fv<cél?
        bmi jo          ;fvhi<célhi
        bne rossz       ;fvhi>célhi
        lda fv
        cmp cel         ;fvhi=célhi és
        bmi jo          ;fvlo<céllo
rossz                   ;rossz, visszafelé kell
        clc
        lda cel         ;cél:=cél+db
        adc db
        sta cel
        lda cel+1
        adc db+1
        sta cel+1
        ldy #$00
lecik   sei
        sta $ff3f       ;ram-rom vál.
        lda (fv),y
        sta $ff3e       ;rom eng.
        cli
        sta (cel),y
        lda fv
        bne fv1         ;fv:=fv-1
        dec fv+1
fv1     dec fv
        lda cel
        bne cel1        ;cél:=cél-1
        dec cel+1
cel1    dec cel
        jsr dbcsok
        bne lecik       ;nem, tovább
        rts
jo                      ;jó sorrend,növ.
        ldy #$00
felcik sei
        sta $ff3f       ;ram-rom vál.
        lda (fe),y
        sta $ff3e       ;rom eng.
        cli
        sta (cel),y
        inc fe
        bne fe1         ;fe:=fe+1
        inc fe+1
fe1     inc cel
```

```
        bne cel2        ;cél:=cél+1
        inc cel+1
cel2    jsr dbcsok
        bne felcik
        rts
        ;
        ;
dbcsok                  ;db-száml. csökk.
        lda db
        bne db2         ;db:=db-1
        dec db+1
db2     dec db
        lda db+1
        and db
        cmp #$ff        ;elfogyott?
        rts
```

(A megjegyzéseknél alkalmazott hi ill. lo jelölések a két-byte-os számok felső ill. alsó byte-ját jelentik.)

Első pillantásra szembetünő, hogy - a titkosirás kódolásánál tapasztaltakkal ellentétben - mennyivel terjedelmesebb az assembly lista az algoritmus szövegénél. Ennek oka megint csak a gépi kódú programozás néhány sajátossága ill. ezek folyománya: nincs neve, csak helye a változóknak, ezáltal a több byte-os adatokkal végzett müveleteknél a többszörösére növekszik a helyigény. Figyeljük meg ezt az összeadásnál (CÉL:=CÉL+DB), a kivonásnái (DB:=FV-FE) ill. az összehasonlitásnál (FE>CÉL és FV<CÉL)! Ügyes fogásokkal ezt bizonyos esetekben lerövidithetjük, erre látunk példát a dekrementálás (FV:=FV-1), az inkrementálás (FE:=FE+1) ill. a 0-val való összehasonlitás (AND DB) esetében.

Térjünk át a RAM-ROM probléma megoldására! Ehhez az alapvető segitséget az az észrevétel adja, hogy a két terület közötti választás mindössze egyetlen byte-on múlik (ti. hogy az $FF3F vagy az $FF3E byte-ba irunk-e). Ez a byte - az lf és az fl másolás miatt - természetesen programunkban két helyen is szerepet játszik. Ha kiszámitjuk, hogy hol van ez a két byte, s a MÁSOL eljárás meghivása előtt ezeket a szükségesre állitjuk, a program már az általunk kivánt állapotban találja őket. A kérdéses byte-ok éppen a FELCIK és a LECIK cimkék utáni 2. cimen vannak, igy már meg is irhatjuk két kis "kapcsolórutinunkat":

```
rom                     ;rom eng.
        lda #$3e
        sta lecik+2
        sta felcik+2
        rts
        ;
```

```
        ;
ram                     ;rom tiltás
        lda #$3f
        sta lecik+2
        sta felcik+2
        rts
```

Nyilvánvaló, hogy ROM-ból másoláskor fölöslegesen írjuk két alkalommal is $FF3E-t, de ez a hátrány elenyésző ahhoz az előnyhöz képest, amit ennek a lehetőségnek a kihasználásával a helyfoglalás terén nyerünk.

A számunkra szükséges területről való másolás eléréséhez nincs tehát más dolgunk, mint a megfelelő (ROM vagy RAM) kapcsolórutint meghívni a MÁSOL-ra való ugrás előtt.

Ezek után fogjunk hozzá a 0...4 funkciók algoritmusának megírásához (jelölje ezeket rendre AMÁS, OMÁS, TÖRÖL, ABESZ,OBESZ)! Tételezzük fel, hogy a C1, C2, C3, C4 paraméterek közül a szükségesek rendre a CÍM1, CÍM2, CÍM3, CÍM4 címeken a tárban vannak. Alig van már dolgunk AMÁS és OMÁS megírásával, s mivel ezek csak igen kevéssé térnek el egymástól, tevékenységük nagyobbik részét egy közös, MÁS nevű eljárásban meg tudjuk fogalmazni:

```
MÁS
   FE:=CÍM1
   FV:=CÍM2
   CÉL:=CÍM3
   MÁSOL
Eljárás vége

AMÁS
   RAM
   MÁS
Eljárás vége

OMÁS
   ROM
   MÁS
Eljárás vége
```

A TÖRÖL eljárás algoritmusa ugyancsak egyszerű:

```
TÖRÖL
   RAM
   FE:=CÍM3+1
   FV:=CÍM2
   CÉL:=CÍM1
   MÁSOL
Eljárás vége
```

Mivel az ABESZ és az OBESZ eljárások ugyancsak sok mindenben megegyeznek, gondoljuk meg, hogy miből áll közös tevékenységük! Elsőként C1...C2-nek C3...C4 hosszával való feljebb mozgatására van szükség (jelölje ezt BESZ), majd a felszaba-

dult területre C3...C4-et kell beraknunk (ez legyen RAK). A megfelelő algoritmusok:

```
BESZ
   RAM
   FE:=CÍM1
   FV:=CÍM2
   CÉL:=CÍM1+CÍM4-CÍM3+1
   MÁSOL
Eljárás vége

RAK
   FE:=CÍM3
   FV:=CÍM4
   CÉL:=CÍM1
   MÁSOL
Eljárás vége

ABESZ
   BESZ
   RAK
Eljárás vége

OBESZ
   BESZ
   ROM
   RAK
Eljárás vége
```

Ezeknek a részeknek a kódolt változata a Függelék F5-F10. lapjain megtalálható. Mivel újabb lényeges elemeket nem tartalmaz, itt nem ismételjük meg. Az egyetlen dolog, amit érdemes megfigyelnünk, az egyes részek "egymásra futásának" (pld. BESZ->MÁSOL, ABESZ->RAK stb.) megszervezése. Térjünk inkább át a paraméterek átvételének, az öt funkció szétválasztásának a kérdésére!

Ígéretünkhöz híven az USR függvényt fogjuk a program hívásához használni, mégpedig úgy, hogy argumentuma egy 0 és 4 közötti egész szám, az általunk választott funkció sorszáma legyen. Több helyen olvasható, hogy a SYS-sel való hívás egyáltalán nem engedi meg paraméterek átadását, az USR használata pedig mindössze 1 paraméter átadására ad módot. Az első cáfolatára már a titkosirásnál mutattunk példát, a másodikéra most fogunk: mind a 3 ill. 4 címparamétert ily módon vesszük át a BASIC-től; a hivó utasitás alakja

p=USR(mód),C1,C2,C3[,C4]

lesz. Nyilvánvalóan ismert, hogy az USR függvény használata előtt az ugrási címet a $0501-$0502 címekre be kell irni. Híváskor az USR függvény argumentuma lebegőpontos formában a FAC területére kerül. Az értékétől függő elágazást csak egész alakjában tudjuk megvalósitani, ezért használjuk a

$A327-en kezdődő FACINT rutint, amely a FAC:=INT(FAC) átalakitást valósitja meg; mivel a mi paraméterünk 256-nál kisebb szám, értékét a $65 címről tudjuk kiolvasni. Amennyiben ez nem 0 és 4 közé esik, a programot a $991C címre küldjük, amely ennek hatására "ILLEGAL QUANTITY ERROR" hibajelzéssel megáll. Az átvett értéktől függően lesz 3 vagy 4 címparaméterre szükségünk, s később ettől függ majd az is, hogy hová kell a programot irányitanunk. Ennek érdekében ezt a paramétert (pontosabban ennek kétszeresét, de erről később lesz szó) tárolnunk kell, erre használjuk a MÓD byte-ot.
A címek beolvasását a CÍMBE ($C38F) rutin segitségével végezzük; működéséről annyit kell tudnunk, hogy elrontja az X regiszter értékét, s a beolvasott két byte-os címet a LINNUM ($14-$15) helyre teszi le, LO-HI sorrendben. Már utaltunk rá, hogy a beolvasandó címek száma az átvett paramétertől függ, ennek kétszeresét - tehát az átveendő byte-ok számát - a HATÁR címen tároljuk.
Lássuk ezek után a programnak ezt a részét!

```
        jsr facint      ;par.->egész
        ldx param       ;mód
        bmi hiba        ;<0, hiba
        cpx #$05        ;>=5?
        bmi mehet       ;igen,hiba
hiba    jmp illq        ;ki:'ill. q.'
mehet   stx mod         ;elrakjuk
        asl mod         ;címhez 2*
        cpx #$03        ;hány cím kell?
        bpl negy        ;beszúrás,->4
        lda #$06        ;másolás-törlés,->3
        .byte $2c       ;átlátszó bit
negy    lda #$08        ;hány byte-ot
        sta hatar       ;kell beolvasni?
        ldx #$00
beolvc  txa             ;index
        pha             ;verembe
        jsr cimbe       ;cím beolv.
        pla             ;index
        tax             ;veremből
        lda linnum
        sta ciml,x      ;#<cím
        inx
        lda linnum+1
        sta ciml,x      ;#>cím
        inx
        cpx hatar       ;vége?
        bne beolvc      ;nem
```

Nagyon jellegzetes fogás a gépi kódú programozásnál az ún. "átlátszó bit" utasitás alkalmazása. Két fajtája van, az egyik, a $24 kódú, a nullalapos, a másik, a $2c kódú pedig az

abszolut címzésű. Az első az őt követő 1, a második pedig az őt követő 2 byte-ot "takarja el", igy az azon (ill. azokon) lévő utasitás(ok) hatástalan(ok) marad(nak). Jelen példánkat tekintve, ha nem beszúrásról van szó, a program az alábbi utasitásokat hajtja végre:

```
        lda #$06
        bit $08a9
        sta hatar
```

,azaz HATÁR tartalma 6, mert 3 cimet kell beolvasni, mig ha beszúrásról van szó, akkor

```
        bpl negy
        lda #$06
        .byte $2c      ;átlátszó bit
negy    lda #$08
        sta hatar
```

, igy HATÁR-ban 8 lesz, mert 4 címre van szükségünk. Az egész "trükköt" az teszi lehetővé, hogy a BIT utasitás csak a jelzőbiteket állitja, más hatása nincs, igy a tár és a regiszterek tartalma változatlan marad.

Most adunk magyarázatot arra, hogy miért a funkció sorszámának kétszeresét tároltuk MÓD-ban (azaz: miért kellett az asl mod utasitás). A vezérlést ugyanis egy jmp cccc utasitással adjuk majd át, de hogy valóban a megfelelő helyre kerüljön, előbb gondoskodnunk kell az ugrási cim cccc helyére irásáról.

Ehhez az egyes funkciók "belépési pontjait" egy táblázatba foglaltuk:

```
ugrtab .word amas,omas,torol,abesz,obesz
```

, majd innen a MÓD tartalmának megfelelően a cim két byteját a cccc helyére töltjük:

```
        ldx mod
        lda ugrtab,x   ;ugrási
        sta ugrik+1    ;cim
        lda ugrtab+1,x ;töltése
        sta ugrik+2
ugrik   jmp $0000
```

A $0000 cim nyilvánvalóan közömbös, a lényeg csupán az, hogy két byte helyet a címnek valamilyen módon lefoglaljunk.

Ez a módszer korántsem az egyedül lehetséges megoldás (a gépi kódú programozásra egyébként is különösen jellemző, hogy ugyanazt a funkciót sokféleképpen meg lehet valósitani), alkalmazása kiváltképpen akkor előnyös, ha nagyszámú elágazási lehetőség közül kell választanunk.

Kérdések és feladatok a 2. fejezethez

2.1 Próbálgassuk a most készített program funkcióit! Először vigyük a képernyőre a tár egyes területeit, mert itt látványosan követhető az egyes funkciók hatása! Ugyanezt grafikus képernyőn is megfigyelhetjük (a karakteres képernyő decimális kezdőcíme 3072, a grafikusé 8192). Vigyázzunk a 26214 ... 26628 részre, mert itt van a programunk!

2.2 Szó esett arról, hogy ennek a programnak is vannak hiányosságai. Példaként: két, egymáshoz csatlakozó intervallum közül próbáljuk beszúrni a másodikat az első elejére! Mit tapasztalunk? Hogyan lehet mégis megoldani a feladatot?

2.3 Maximálisan milyen méretű blokkot tudunk a TEDMON segítségével átmásolni, ha a célintervallum kezdete a forrásintervallumba esik?

2.4 Az átmásolandó byte-ok számát a DB:=FV-FE összefüggés alapján számoltuk ki, pedig ez (mivel a határokat mindenütt másoljuk) 1-gyel kevesebb, mint a másolandó byte-ok száma. Elvesztettünk egy byte-ot?

2.5 Becsüljük meg, hogy mennyi tárat igényelt volna a program akkor, ha a menet közben említett, helyet megtakarító fogásokat nem alkalmazzuk!

2.6 Vonjuk össze a RAM és a ROM rutinokat úgy, hogy - mivel csupán egyetlen utasításban térnek el egymástól - a kérdéses utasítást a program ide jutása előtt mindig a megfelelőre módosítjuk! Mennyi helyet takarítunk meg ezáltal?

2.7 Írjuk át a program vezérlő részét úgy, hogy az elágazáskor az ugrótáblából az X indexszel kiolvasott ugrási címet tegye a verembe, majd (miként ez a gépi kódú programoknál elég gyakori) RTS utasítással adja át a kívánt címre a vezérlést! Hogyan kell ehhez az ugrótáblát módosítani?

2.8 Milyen változtatást kell a programban végeznünk, ha valamilyen okból az ugrótábla nem módosítható? (ld. 2.7 feladat!)

2.9 Készítsünk olyan programot, amely úgy írja át magát, hogy futás közben más területre kerül, majd ott ismételten elindul! Milyen következményei lesznek egy ilyen program futásának?

2.10 Keressünk a ROM-ban átlátszó BIT utasításokat!

3. TÜKRÖZÉS

Ebben a fejezetben ismét egy elég terjedelmes, de viszonylag könnyen követhető feladattal foglalkozunk: a grafikus képernyőn lévő képet vízszintes vagy függőleges középvonalára vagy középpontjára tükröző programot készítünk. Elvárjuk a programtól, hogy a tükörképet - kívánságunk szerint - az eredeti és a tükörkép közötti AND, OR vagy EOR logikai művelet felhasználásával hozza létre; ez összesen 12 funkció megvalósítását jelenti. A fejezet megértéséhez szükséges előismeretek [I] 2.2 fejezetében magtalálhatók.

A BASIC-kel való kapcsolat megszervezését, a megfelelő funkció kiválasztását hagyjuk későbbre, kezdjünk hozzá az algoritmusok megírásához!

Programunkban semmi különösebb nehézség nincs, mert a kívánt műveletek mindegyike egyszerű tömbmásolásra vezethető vissza. Problémát legfeljebb az jelenthet, hogy a képernyőn nem a tárbeli sorrendjüknek megfelelően helyezkednek el az egyes tárcímek, hanem egy-egy 8*8 raszterpontos négyzeten belül (nevezzük ezt a továbbiakban karakterpozíciónak!) soronként, majd ezután jön ugyanígy a következő karakterpozíció. Valamilyen módon meg kell tehát szerveznünk a képernyő bejárását úgy, hogy minden pontján pontosan egyszer menjünk végig.

A függőleges tengelyre vonatkozó tükrözésnél a következő bejárást választottuk: a bal felső és a jobb felső sarokból indulunk, a "megfelelő módon" kicseréljük a kérdéses karakterpozícióhoz tartozó byte-okat, majd mindkettővel lépünk egyet a függőleges középvonal felé, s ezt ismételjük, amíg középre nem érünk. Ezután az alatta lévő sorban végrehajtjuk ugyanezt, s így tovább, amíg az utolsó sor közepén be nem fejezzük a bejárást. A fentebb említett "megfelelő módon" való csere nyilvánvalóan úgy érhető el, hogy a kérdéses byte-okat "átforgatjuk", azaz: az eddigi 0. bit ezután a 7., az eddigi 1. ezután a 6., ... s az eddigi 7. bit ezután a 0. helyre kerül. Előre számítva arra, hogy a különböző logikai műveletekkel végzett tükrözéskor az eredeti értékekre is szükség lesz, már eleve ne a saját helyükön végezzük el a byte-ok forgatását, hanem használjuk a BALR, BALU, JOBBR,

JOBBU címeket a bal- illetve jobboldali byte régi és új értékének tárolására. A CÍMB ill. CÍMJ értékek mindig az aktuális karakterpozíció első byte-ját tartalmazzák, belőlük indexeléssel jutunk el a mindenkori aktuális byte-okhoz. Lássuk tehát az algoritmust!

```
FÜGG. TENGELY
   CÍMB:=BALK  [bal felső sarok]
   CÍMJ:=JOBBK [jobb felső sarok]
   SSZÁML:=25  [sorok száma]
   Ciklus, amíg SSZÁML>0
      PSZÁML:=20 [pozíciók száma egy félsorban]
      Ciklus, amíg PSZÁML>0
         Ciklus Y:=7-től 0-ig -1-esével
            BALR:=(CÍMB),Y
            JOBBR:=(CÍMJ),Y
            BALU:=FORGAT(JOBBR)
            JOBBU:=FORGAT(BALR)
            (CÍMB),Y:=BALU      (*)
            (CÍMJ),Y:=JOBBU     (*)
         Ciklus vége
         CÍMB:=CÍMB+8
         CÍMJ:=CÍMJ-8
         PSZÁML:=PSZÁML-1
      Ciklus vége
      CÍMB:=CÍMB+160
      CÍMJ:=CÍMJ+480
   Ciklus vége
Eljárás vége

FÜGGVÉNY FORGAT(BYTE) [BYTE = 1 byte 8 bitjének vektora]
   Ciklus X:=0-tól 7-ig
      ÚJBYTE(X):=BYTE(7-X)
   Ciklus vége
   FORGAT(BYTE):=ÚJBYTE
Függvényeljárás vége
```

A FORGAT függvényeljárás kódolása nagyon egyszerű: pld. az egymást követő ASL B1 és ROR B2 utasítások nyolcszori ismétlése pontosan a kívánalmaknak megfelelően forgatja át B1-et B2-be. Nyilvánvaló, hogy a (függőleges középvonalra történő) négy különböző tükrözéshez ("simán", AND-del, OR-ral ill. EOR-ral) nagyon ügyetlen dolog lenne négyszer csaknem ugyanazt a programrészt megírnunk. Sokkal frappánsabb megoldás az, ha az előző fejezetben megismert módszert alkalmazzuk: a programmal átíratjuk önmagát úgy, ahogy azt az aktuális funkció végrehajtása megköveteli. Látható, hogy pld. az AND-del való tükrözéskor a fenti algoritmus (*) sorait kellene csak módosítanunk a következőkre:

```
(CÍMB),Y:=(CÍMB),Y AND BALU
(CÍMJ),Y:=(CÍMJ),Y AND JOBBU
```

Egy-egy ilyen sor kódolása nyilván csak több assembly sorban lehetséges, pld. a következő módon:

```
        lda balu
modb    and (cimb),y
        sta (cimb),y
        lda jobbu
modj    and (cimj),y
        sta (cimj),y
```

Ha tehát egy táblázatba gyüjtjük a nullalapos indirekt indexelt címzéshez tartozó AND, ORA és EOR kódokat, ezek közül mindenkor a megfelelőt töltve a modb+1 és a modj+1 címekre, programunk éppen a kívánt utasítást fogja végrehajtani. Látszólag baj van azonban a "sima" tükrözésnél: ekkor ugyanis a modb és a modj soroknak nem csak egy byte-ját kellene átirnunk, hanem mindkettőt (pld. NOP-ra), s ez mindenképpen bonyolítja a dolgot. Pofonegyszerű a megoldás viszont akkor, ha észrevesszük: semmi baj nem származik abból, ha ugyanazt az értéket egymás után kétszer töltjük ugyanarra a helyre!

```
        lda balu
modb    sta (cimb),y
        sta (cimb),y
        lda jobbu
modj    sta (cimj),y
        sta (cimj),y
```

Ez egyébként máskor természetesen butaság, de alkalmazása most komoly haszonnal jár: ugyanazt a módszert alkalmazhatjuk mind a négy esetre! Nem kell tehát mást tennünk, mint a táblázatba felvennünk az STA (cc),y kódot is, s a táblázat négy eleme közül a megfelelőt a modb és a modj címekre tölteni, máris helyesen dolgozik a program. Lássuk tehát a függőleges tükrözés végső változatát kódolva:

```
fugg                    ;függőleges tükr.
        lda #<balk      ;kezdeti
        sta cimb        ;értékek
        lda #>balk
        sta cimb+1      ;beállitása
        lda #<jobbk     ;mindkét
        sta cimj        ;mutatónál
        lda #>jobbk
        sta cimj+1
        lda #sorsz
        sta sszaml
sork    lda #pozszf     ;sorkezd.
        sta pszaml
pozk    ldy #$07        ;pozkezd.
bytek   ldx #$08        ;bytekezd.
        lda (cimb),y    ;régi bal
        sta balr
```

```
        lda (cimj),y  ;régi jobb
        sta jobbr
forgv   asl balr      ;balról
        ror jobbu     ;jobbra
        asl jobbr     ;jobbról
        ror balu      ;balra
        dex           ;kész?
        bne forgv     ;nem, tovább
        lda balu
modb    and (cimb),y  ;csere
        sta (cimb),y
        lda jobbu
modj    and (cimj),y
        sta (cimj),y
        dey           ;tart még a pozíció?
        bpl bytek     ;igen
        clc           ;nem
        lda cimb      ;balpozíció
        adc #$08      ;növelése
        sta cimb      ;8-cal
        lda cimb+1    ;(1 poz.)
        adc #$00
        sta cimb+1
        sec
        lda cimj      ;jobbpozíció
        sbc #$08      ;csökkentése
        sta cimj      ;8-cal
        lda cimj+1    ;(1 poz.)
        sbc #$00
        sta cimj+1
        dec pszaml    ;félsor vége?
        bne pozk      ;nem, vissza
        clc
        lda cimb      ;balsormutató
        adc #$a0      ;növelése
        sta cimb      ;160-nal
        lda cimb+1    ;(1 félsor)
        adc #$00
        sta cimb+1
        clc
        lda cimj      ;jobbsormutató
        adc #$e0      ;növelése
        sta cimj      ;480-nal
        lda cimj+1    ;(3 félsor)
        adc #$01
        sta cimj+1
        dec sszaml    ;sorok vége?
        beq vege      ;igen, kész
        jmp sork      ;nem, vissza
vege    rts
```

A vizszintes tengelyre és a középpontra vonatkozó tükrözések algoritmusát nem részletezzük, csupán a képernyő bejárásának módját ismertetjük vázlatosan. Ezek alapján , valamint a megjegyzésekkel bőségesen ellátott fordítási lista (ld. Függelék F11-F17. oldal) segítségével bárki tökéletesen megértheti a program müködését.

A vízszintes tengelyre vonatkozó tükrözésnél a kép bal felső és bal alsó sarkából indulunk (karakterpozíciónként). Ügyelnünk kell arra, hogy itt forgatásra ugyan nincs szükség, de a megfelelő karakterpozícióknál az egyes byte-ok helyét is meg kell cserélni: az egyik pozíció legalsó sora a másik legfelső sorába megy át és viszont stb. Amint egy karakterrel végeztünk, jobbra lépünk, sor végén pedig - a képernyő közepe felé haladva - új sort kezdünk. Páratlan számú sor van (25), így az utolsó lépésben a "forrássor" és a "célsor" megegyezik, aminek az a furcsa eredménye lesz, hogy a kétszeri tükrözés következtében ez a sor eredeti alakjában marad. Ennek megelőzése végett az utolsó sorhoz tartozó karakterpozícióknak csak az alsó ill. felső 4 byte-ját szabad figyelembe vennünk.

A középpontra vonatkozó tükrözésnél a kép bal felső és jobb alsó sarkából indulunk (ugyancsak karakterpozíciónként). Ebben az esetben mindkét tényező bonyolítja a helyzetet: forgatásra is szükség van (mint a függőleges tengelyre vonatkozó tükrözésnél), és a megfelelő karakterpozícióknál az egyes byte-ok helyét is meg kell cserélni (mint a vízszintes tengelyre való tükrözésnél). Amint egy karakterrel végeztünk, a "fölsővel" mindig balra, az "alsóval" mindig jobbra lépünk, míg a képernyő közepére nem érünk (ez egyébként pontosan 500-500 lépést jelent).

Kezdjünk hozzá a BASIC-kel való kapcsolat és a 12 funkciónak megfelelő elágazás megszervezéséhez! A programot most a

SYS26214,X

utasítással hívjuk, ahol 0<=X<=11 egész szám, amelynek értékét az alábbi táblázatból olvashatjuk ki:

	az előírt logikai kapcsolat			
	csere	and	or	eor
függ. tengely	0	1	2	3
vízsz. tengely	4	5	6	7
középpont	8	9	10	11

A paraméter beolvasásához az 1. fejezetben megismert CHKOMA és GETBYT rutinokat használjuk. A vízszintes tengelyre és a középpontra vonatkozó tükrözés esetén is szükségünk lesz az STA, AND, ORA, EOR kódjának betöltésére, így egyszerűbb ha - a másik két rutin konkrét ismerete nélkül - a megfelelő cí-

meket a MODA, MODF, MODE, MODU címkékkel jelöljük, s egyszerre hajtjuk végre a szükséges módosítást. A vezérlő rész kódja:

```
        jsr chkoma      ;vesszőig
        jsr getbyt      ;mód x-be
        txa
        pha             ;verembe
        and #$03        ;/4 utáni
        tax             ;maradék
        lda kodok,x     ;a megfelelő
        sta modb        ;kód betöltése
        sta modj        ;minden
        sta moda
        sta modf        ;módosító
        sta mode
        sta modu        ;helyre
        pla             ;veremből
        lsr a
        lsr a           ;osztás 4-gyel
        beq fugg        ;függ. tükr.
        lsr a           ;további 2-vel
        bne kp
        jmp viz         ;vizsz. tükr.
kp      jmp kozepp      ;középpontos
fugg                    ;függőleges tükr.
    .
    .
    .
```

A programot betöltő 111. bemutató BASIC program a paramétereket az abc első 12 betűje valamelyikének megnyomására veszi át, egy apró fogással lehetőséget biztosítva a parancs módban történő rajzolásra, majd a program folytatására.

Kérdések és feladatok a 3. fejezethez

3.1 Készitsük el a vizszintes tengelyre 111. a középpontra vonatkozó tükrözés algoritmusát!

3.2 Készitsünk programot a függőleges tengelyre vonatkozó tükrözéshez, amely a sorok kezdetének és végének címeit egy táblázatban tárolja, s minden sor befejeztével innen nézi ki a következő címet (tehát nem számol, mint a közölt program)!

3.3 A "MONITOR" prg. intézkedett illegális paraméter esetén a hiba kiirásáról. Hogyan viselkedik a "TÜKRÖZÉS" 0-nál kisebb vagy 11-nél nagyobb paraméter átadásakor? Vizsgáljuk meg ugyanezt a kérdést a "TITKOSÍRÁS" esetében is!

3.4 Hogyan módosulnak a tükrözési algoritmusok, ha egy sorfolytonos ábrázolású képernyőn kell a transzformációkat végrehajtani? Készitsünk programot ennek alapján - az egyértelműség miatt 24 sorosra csökkentett méretű - karakteres képernyő tükrözéseire! (Miért van szükség erre a korlátozásra?)

3.5 Milyen más szervezéssel lehetne megoldani a képernyő bejárását az egyes tükrözések végrehajtásakor?

3.6 A könnyebb érthetőség végett az egyes funkciók megvalósitásakor az adatok átmeneti tárolására más-más címeket vettünk igénybe (BALR, JOBBR, BALU, JOBBU, ABYTE, FBYTE, ELSR, UTOR, ELSU, UTOU, ld. F12-F17. oldal), bár ezt kevesebb hellyel is megoldhattuk volna, mint pld. a CÍMB, CÍMJ, CÍMA, CÍMF, CÍME, CÍMU esetében tettük. Minimálisan hány byte kell a fenti 10 érték elhelyezéséhez?

3.7 Készitsünk programot, amely 1 karakterpozicióny részt saját középpontjára tükröz! (Erre olyan esetben van szükség, amikor a képernyőnek páratlan számú sora és oszlopa is van.)

3.8 Az elemi geometriai tankönyvekből is ismeretes, hogy egy pontra vonatkozó tükörképet két, egymásra merőleges, s egymást az adott pontban metsző tengelyre vonatkozó tükrözés egymásutánjaként is megkaphatunk. Fölöslegesen dolgoztunk volna a középpontra vonatkozó tükrözést megvalósitó programmal?!

3.9 Készitsünk programot, amely a grafikus képernyő bal és jobb oldalán lévő képet megcseréli!

3.10 Készitsünk programot, amely a grafikus képernyő alsó és felső részén lévő képet megcseréli!

3.11 Készitsünk programot, amely a grafikus képernyő bal felső negyedét kinagyitja az egész képernyőre!

3.12 Készitsünk programot, amely a grafikus képernyőt "összehúzza" a közepén lévő ("vele koncentrikus"), negyedakkora területű téglalapra!

4. PRÍMEK

Ebben a fejezetben a több (pld. négy) byte-on ábrázolt egész számok aritmetikájával foglalkozunk. Feladatunk egy olyan szubrutin készitése, amely alkalmas annak eldöntésére, hogy egy, a gépben egészként ábrázolhatóaknál nagyobb (azaz 32767-et meghaladó) pozitiv egész szám prímszám-e vagy sem. Nyilvánvaló, hogy a kerekitési hibák miatt nem használhatunk valós tipusú változókat, igy csak a több byte-os egészek használatával érhetünk célt. Ha csak a ROM-ban rendelkezésünkre álló egyszerü beviteli lehetőséget használjuk ki, akkor is 31 bites számok vizsgálatára nyilik módunk, ami annyit jelent, hogy a M = 2 147 483 647-nél nem nagyobb számok körében vizsgálódhatunk, ami már nem lebecsülendő. A fejezet megértéséhez szükséges tudnivalók [I] 2.3 fejezetében megtalálhatók.

Az, hogy egy szám prim-e vagy sem, a legegyszerübben talán egy, a szám négyzetgyökéig terjedő prímszámtáblázat segitségével dönthető el. Vizsgáljuk meg kicsit részletesebben, megéri-e nekünk ehhez a feladathoz táblázatot konstruálnunk! Egy ilyen táblázat a legegyszerübben az eratoszthenészi szita segitségével épithető fel. A M-ig terjedő számtartományhoz SQR(M)-ig kellene a prímeket táblázatba gyüjtenünk, azaz az 1...46 341 intervallumból kellene a prímeket "kiszitálnunk". Ha az eredeti táblázatba a páros számokat már eleve be sem irjuk, akkor is 23 170 számmal kell indulnunk, s mivel ezek nagyrésze 255-nél nagyobb, tárolásukhoz kb. 46 kbyte szükséges. Igaz ugyan, hogy ez az összetett számok kiszitálása után maximum 10 kbyte-ra csökken, de még ez is nagyon sok. (Megjegyezzük, hogy ha nagyon gyors programra lenne szükségünk, s elegendő hely állna a tárban rendelkezésünkre, akkor valószinüleg mégis ezt a megoldást választanánk.)

Meg lehetne próbálkozni persze más adatszerkezettel, pld. egy összefüggő tárrészben minden számnak egy bitet megfeleltetni, amelynek értéke 1, ha a szám prim, és 0, ha nem prim; ekkor viszont elég bonyolult volna a prímek elérése, hiszen mindegyiket sorszámán keresztül tudnánk csak azonositani. Ez

a megoldás egyébként jelen esetben kb. 3 kbyte-nyi tárat igényelne (természetesen itt is csak a páratlan számokat tárolnánk).

Tudunk azonban táblázat nélkül is boldogulni. Igaz, hogy némi fölösleges munkát végeztetünk a programmal, de csak néhány byte-nyi munkaterületre lesz szükségünk, s jóval egyszerűbben meg tudjuk a programot szervezni. Egy apró trükkel pedig még arra is tudunk ügyelni, nehogy túl sokáig tartson a vizsgálat.

Gondolatmenetünk a következő: a 2-vel való oszthatóságot egy pillanat alatt el tudjuk dönteni, ez után viszont - 3-tól kezdve - csak a páratlan számokkal való oszthatóságot vizsgáljuk. Nyilván fölösleges a 9-cel való oszthatóságot külön néznünk, ha egy szám már 3-mal sem volt osztható, mégis megéri ezt elviselnünk, mert az elképzelhető osztók során nagyon egyszerű kettesével végigléptetni a programot.

Az osztásra nincs gépi kódú utasitás, igy nekünk kell olyan programot irnunk, amely eldönti, hogy egy A szám osztója-e egy B számnak vagy sem. Erre a legkézenfekvőbb módszer az, ha B-ből addig vonjuk ki A-t, mig csak negativ számot nem kapunk (ez előbb-utóbb bekövetkezik), s ha az előző eredmény - más szóval a legkisebb nem negativ maradék, amit úgy kapunk meg, hogy az első negativ eredményhez egyszer hozzáadjuk A-t - 0 volt, akkor A osztója B-nek, különben nem.

Igen ám, de ez igy nagyon sokáig tartana! Gondoljuk meg, hogy egy 2 milliárd körüli számból a 3-at 650 milliónál is többször kellene kivonnunk, mig 0-nál kisebb értéket kapunk, ami (mivel a négybyte-os számok kivonása legalább 80 óraciklust igényel) minimálisan 50 milliárd óraciklus, ami a 2 megahertznél lassúbb processzor miatt (még letiltott kép esetén is!) legalább 25 ezer sec-ot jelent! Ez kb. 7 óra, és még mindig csak a 3-mal végeztünk!

Lényegesen gyorsitana a tempón, ha 3-nak ismernénk egy olyan nagy többszörösét, ami az eredeti számnál nem sokkal kisebb, mert egyetlen kivonással nagymértékben közelebb kerülnénk a 0-hoz, igy a kivonás befejezéséhez is. Általában, ha B-nek A-val való oszthatóságára vagyunk kiváncsiak, akkor gyorsan haladhatunk, ha a kivonogatást A-nak egy B-hez közeli többszörösének kivonásával kezdjük. Az sem túl nagy baj, ha A-nak B-nél nagyobb többszörösével kezdünk, mert az egyszeri

kivonás után máris negatív eredményhez jutunk, azonnal "vissza kell adni", így - bár előre nem jutottunk, - kárt nem okoztunk a kivonással.

Kézenfekvő, hogy - mivel a számítógépen a 2-vel való szorzást ill. osztást egy egyszerű rotálással meg tudjuk valósítani, - A-nak valamely 2 hatvánnyal képezett szorzatával (jelöljük ezt A'-vel) kezdjük a kivonást, s ha ez tovább már nem megy, folytassuk A'/2-vel, majd A'/4-gyel s így tovább, utoljára A-val. De melyik többszörös legyen az A'? Az előző bekezdésben írtak szerint az sem okoz problémát, ha B-nél nagyobb többszörössel kezdünk, így egyértelmű, hogy A' A-nak azzal a 2-hatvánnyal való szorzata, amellyel még "éppen belefér" a 4 byte-ba. Pontosabban: A-t addig rotáljuk balra, amíg a legfelső, 31-es biten (MSB) 1-est nem kapunk; ez lesz a keresett A'.

Innen indulunk tehát, de meddig csökkentsük felezéssel A' értékét? Valamilyen módon mindenképpen figyelnünk kell, mert pld. A1=12 (binárisan 1100) és A2=3 (binárisan 11) ugyanazt az A'-t (binárisan 11000000000000000000000000000000) eredményezi, ezzel kezdünk, de meddig csináljuk a felezést? Az egyik megoldás az, hogy megőrizzük A értékét (erre egyébként is szükségünk lesz, mert ha A nem osztó, akkor A+2-vel kell folytatnunk a próbálkozást), s minden osztás (jobbra rotálás) után végzünk egy vizsgálatot, ami négy byte-os számoknál nem a legkellemesebb feladat; ennek eredményétől függően folytatjuk tovább az eljárást. Mi egy jobb megoldást választunk: egyszerűen megszámoljuk, hány bittel forgattuk balra A-t, s nyilvánvalóan ugyanennyivel kell vissza is forgatnunk.

Ezek után hozzá is foghatunk az algoritmus megírásához. Jelölje a két szám közül az osztandót ODO, az osztót OSZTO; a következő algoritmus az ODO/OSZTO maradékos osztás maradékát ODO-ban szolgáltatja:

```
MARAD
   SZÁML:=0
   Ciklus, amíg MSB=0
      OSZTO <--          [rotáció balra 1 bittel]
      SZÁML:=SZÁML+1
   Ciklus vége
```

```
      Ciklus, amíg SZÁML>0
         Ciklus, amíg ODO>=0
            ODO:=ODO-OSZTO
         Ciklus vége
         ODO:=ODO+OSZTO    [egyszer vissza kell adni!]
         OSZTO -->         [rotáció jobbra 1 bittel]
         SZÁML:=SZÁML-1
      Ciklus vége
   Eljárás vége
```

Most már milyen egyszerű! Kódolása sem bonyolult, de ügyelnünk kell néhány apróságra. Az OSZTO balra rotálásánál a jobb oldali byte-ba 0-t kell jobbról beléptetnünk, a többi byte-ba a tőle jobbra lévőből "kicsordult" carry értékét; ugyanez az ellenkező oldalról érvényes a jobbra rotálásnál. A "kivonogató" ciklusban nyilvánvalóan mind a négy byte-on el kell végeznünk a kivonást. Erre ciklust is szervezhetünk pld. az Y regiszter segitségével, s mivel a DEY utasitás nem állitja a carryt, az még mindig azt az értéket mutatja, amire az SBC beállitotta, azaz jelzi, hogy van-e szükség további kivonásra vagy nincs. Lássuk a kódot!

```
marad                ;maradékot számol
       lda #$00      ;egy adott osztónál
       sta szaml
fel    bit oszto     ;legfelső bit=0?
       bmi ki        ;igen, kiszáll
       asl oszto+3   ;ütközésig
       rol oszto+2   ;balra
       rol oszto+1   ;léptetés
       rol oszto
       inc szaml     ;+számlálás
       jmp fel
ki     ldy #3        ;odo:=
       sec           ;odo-
kivcik lda odo,y     ;osztó,
       sbc oszto,y   ;ahányszor
       sta odo,y     ;csak lehet
       dey           ;(+még egyszer)
       bpl kivcik
       bcs ki        ;eredmény>=0->tovább
       ldy #3        ;eredmény<0
       clc           ;egyszer vissza
hadcik lda odo,y     :kell adni
       adc oszto,y   ;odo:=
       sta odo,y     ;odo+osztó
       dey
       bpl hadcik
       lsr oszto     ;1-gyel
       ror oszto+1   ;jobbra
       ror oszto+2   ;léptetés
       ror oszto+3
       dec szaml     ;lehet még?
       bpl ki        ;igen, ismét
       rts
```

Szinte már csak néhány technikai kérdés van hátra: tároljuk a SZÁM címen az eredeti szám értékét, a HELY tartalmazza mindig az aktuális osztójelöltet, GYÖK pedig SZÁM négyzetgyökét. A beolvasást a

P = USR(SZÁM)

utasitással fogjuk végezni, eredményként pedig

P = 1 -et kapunk, ha SZÁM=1,

P = 0 -t, ha SZÁM prim, és

P SZÁM-nak a legkisebb primosztója egyébként.

A BASIC-be való visszatérés a következő eljárásokon keresztül történhet:

KIÍR : FAC-ot kerekiti, majd --> BASIC

EGY : FAC felső 3 byte-ját nullázza, majd --> KIÍR

PRÍM : FAC alsó byte-ját nullázza, majd --> EGY

Ezek után lássuk az algoritmust:

```
PRÍM-E
   FAC:=INT(FAC)
   Ha FAC=0, akkor FAC:=2          [0, egy osztója 2]
                    KIÍR
   Elágazás vége
   Ha FAC=1, akkor EGY
   Ha FAC<4, akkor PRÍM            [2 vagy 3]
   SZÁM:=FAC
   GYÖK:=INT(SQR(FAC))
   HELY:=3
   Ciklus
      ODO:=SZÁM
      OSZTÓ:=HELY
      MARAD
      HELY:=HELY+2
    amig ODO>0 és HELY<=GYÖK
   Ciklus vége
   Ha ODO=0, akkor FAC:=HELY-2     [nem prim]
             különben FAC:=0       [prim]
```

Néhány tudnivaló a kódoláshoz:

Az INTFL ($9FOB) rutin a FAC-ban lévő egész értéket lebegőpontossá alakitja.

Az SQR (($A5E4) rutin a FAC:=SQR(FAC) átalakitást hajtja végre.

A TULCS ($9fb2) címtől a program "OVERFLOW ERROR" hibajelzéssel megáll.

A TFA ($A291) és a TAF ($A281) rutinok az ARG:=FAC, ill. a FAC:=ARG másolásokat hajtják végre.

Külön felhivjuk a figyelmet az összehasonlitások kódolására, mivel négy byte-os adatokról van szó. Viszonylag egyszerü a 0-val való hasonlitás, mert mind a négy byte akkor és csak

akkor zérus, ha a négy byte-ot az ORA utasítással "egymásra másolva" is zérust kapunk. Érdemes a HELY>GYÖK összehasonlítást is alaposan tanulmányozni. Itt csak a főciklus kódját közöljük, a teljes program az F18-F22 oldalakon található.

```
főcikl                  ;főciklus
       jsr todo         ;osztandó töltése
       jsr toszto       ;osztó töltése
       jsr marad        ;maradék van-e?
       lda #$00         ;a maradéknak
       ldy #$03         ;mind a
nulc   ora odo,y        ;négy
       dey              ;byte-ja
       bpl nulc         ;nulla?
       cmp #$00
       beq helyki       ;igen, nem prím
       jsr helyno       ;akt. osztó növelése 2-vel
vizsg                   ;lehet-e még
       ldy #$00         ;növelni?
vcikl  lda gyok,y
       cmp hely,y
       beq vugr         ;hely,y=gyök,y , tovább
       bcc prim         ;hely>gyök, elég
       bcs focikl       ;hely<=gyök, tovább
vugr   iny              ;köv. byte
       cpy #$04         ;utolsó?
       bne vcikl        ;nem, vizsgálat folyt.
       beq focikl       ;tovább
```

Kérdések és feladatok a 4. fejezethez

4.1 Készítsünk programot, amely 8 byte-os egész számokat ad össze!

4.2 Készítsünk programot, amely 4 byte-os egészeket szoroz össze! Alkalmazzunk az osztásnál használt fogáshoz hasonló módszert! Gondoljunk arra, hogyan szorzunk a gyakorlatban többjegyű számokat!

4.3 Készítsünk programot, amely a 4 byte-on ábrázolható egészekre DEC->HEX ill. HEX->DEC átalakítást végez!

4.4 Készítsünk programot, amely a M számnál nagyobb számokat is beolvas! Próbálkozzunk a szövegként történő beolvasással, majd a tárban végezzük el az átalakítást!

4.5 Bántó a BASIC-program 3. funkciója (két szám közötti primek kiírása) használatakor, hogy nagy számok esetén lebegőpontos alakban írja ki a prímeket. Küszöböljük ki ezt a hibát!

4.6 Írjunk gépi kódú keretprogramot, amely elvégzi a BASIC-program 2. funkcióját!

5. KÉT KÉPERNYŐ

Az egyik legizgalmasabb fejezet következik, nem csupán azért, mert a gyakorlati munkában alkalmazható további jó tulajdonságokkal ruházza fel a gépet, hanem azért is, mert a gépi kódú programok hívásának új, talán legelegánsabb módjával, új utasítások bevezetésével fogunk megismerkedni. Ezeket az új utasításokat - a program lefutása után - a standard BASIC utasításokkal azonos módon, velük párhuzamosan alkalmazhatjuk (ezért hívják ezt az eljárást BASIC-bővítésnek is). A fejezetben foglaltak megértéséhez szükséges információk az [1] 2.2 fejezetében megtalálhatók. Gyakran - főként demonstrációs programok készítésénél - hiányzik az a lehetőség, hogy párhuzamosan két grafikus képernyőt használhassunk. (Pld. egy nagyszámú atom mozgását szemléltető, diffúziós szimulációs programot készítünk. A folyamatban részt vevő pillanatnyi konfiguráció folyamatos kijelzése csak a nagyfelbontású képernyőn valósítható meg, erre viszont a folyamatot leíró paraméterek időbeni változását ábrázoló grafikonok elhelyezéséhez is szükségünk lenne. Vagy egy másik példa: egy-egy grafikus ábra elkészítése általában hosszadalmas, s ha a program futása során többször is le kell rajzolnunk ugyanazt az ábrát - bár ez munkában nem sok többletet jelent, hiszen csak egy szubrutin ismételt meghívásáról van szó -, idővel unalmassá válhat a többszöri, hosszú várakozás.) Több előnye lenne tehát a két nagyfelbontású képernyő használatának.

Viszonylag egyszerűen megoldható az, hogy az eredetin kívül legyen egy másik, "passzív" képernyőnk is, amelyet bármikor láthatunk, de írni, rajzolni rájuk vagy törölni őket nem tudjuk; ehhez elegendő a TED $FF12-es regiszterét a megfelelő értékűre átírni (ezt egyik későbbi programunknál hasznosítani is fogjuk). Ez azonban korántsem jó megoldás, hiszen a "passzív" képet is meg kell egyszer rajzolni, s ennek a lehetősége nem adott, mert a ROM törlő és rajzoló rutinjai mindig az eredeti (szín- és fényerő $1800-tól, bittérkép $2000-től) képmemóriát használják. Elvileg nincs aka-

dálya annak, hogy a ROM-ban lévő rutinokat RAM-ba másoljuk, s - a 3. és 4. fejezetben látott példák megoldásához hasonlóan - egy önmagát módosító program segitségével mindig az aktuális képernyőre "irányitsuk" ezeket. Ez a megoldás - az átmásolt terület nagysága miatt - nagyon sok tárat igényel, megszervezése is elég bonyolult feladat.

Sokkal egyszerűbb és járhatóbb út az, amikor továbbra is az eredeti képernyő-memóriát (nevezzük ezt az egyszerüség kedvéért látható képnek) használjuk, de lehetővé tesszük, hogy ennek a területnek a tartalmát bármikor kicserélhessük egy előre lefoglalt, vele azonos nagyságú terület (nevezzük ezt raktárnak) tartalmával.Ez egy tömbcserélő ciklus segitségével viszonylag egyszerüen megvalósitható.

Két változó értékének cseréjét technikailag a következő módon szoktuk lebonyolitani: a két változó egyikének értékét elmentjük, a másik értékét az elsőbe töltjük, majd az elmentett értéket a második változóba tesszük. Nem sokkal bonyolultabb azonban az a módszer sem, amelynél mindkét változó értékét elmentjük, majd fordított sorrendben töltjük őket vissza. Kicsivel hosszabb, és kicsivel több tárat igényel, de van egy nagy előnye: két feltétel beiktatásával sokkal általánosabb mozgatásokat is meg tudunk oldani vele, nevezetesen a cserén kivül alkalmas lesz - természetesen bizonyos feltételek beállitásától függően - csak az első változó tartalmának a másodikba, illetőleg csak a második tartalmának az elsőbe másolására is.

Ez tehát azt jelenti, hogy ha algoritmusunkat igy szervezzük meg, akkor ezzel - némi időbeni hosszabbodás és egy kis többletmunka ellenében - nem csupán a két képernyő cseréjét, hanem egyúttal a látható képnek a raktárba, ill. a raktárnak a látható képre vitelét is meg tudjuk valósitani.

S ilyenkor kap vérszemet az ember: a 3. fejezetben, a TÜKRÖZÉS programban alkalmazott kis trükkel, egy-egy sor betoldásával alkalmassá tudjuk tenni programunkat arra, hogy a másolást - választásunk szerint - egyszerű cserével, vagy az AND, OR vagy EOR müveletek valamelyikével hajtsa végre!

A látható kép - mint már emlitettük - $2000-től kezdődik, a raktár helyét pld. $5000-től kezdődően jelölhetjük ki; a bittérkép mindkét esetben $1FFF byte területet foglal el. Gondoskodnunk kell azonban arról, hogy a BASIC ezt a terüle-

tet ne használja, a BASIC kezdetét tehát $7000 fölé (mondjuk $7001-re) kell helyeznünk. Ezáltal a $4000-től kezdődő $07FF hosszúságú (tehát éppen egy karakteres képernyőnyi) üres, kihasználatlan terület kiválóan alkalmas arra, hogy a karakteres képernyőhöz is készítsünk egy raktárat, ebből is kettőt tudjunk párhuzamosan használni. Cserélő algoritmusunkat már annak szellemében kezdjük tervezni, hogy mindkét képernyőfajta esetén alkalmas legyen a már kifejtett funkciók végrehajtására.

A LÁTHATÓ, RAKTÁR és HOSSZ értékeket - a grafikus ill. karakteres választástól függően - a tárban a megfelelő helyre kell tennünk. Ezt a KAR ill. GRA szubrutinok a következő egyszerű módon végzik el:

```
klathe=$0800          ;kar. látható eleje
krakte=$4000          ;kar. raktár eleje
khossz=$07ff          ;kar. kép hossza
glathe=$2000          ;graf. látható eleje
grakte=$5000          ;graf. raktár eleje
ghossz=$1fff          ;graf. kép hossza
lath=$00e0            ;címek
rakt=$00e2            ;helye
hossz=$00e6           ;hossz helye
kar                   ;karakt. kép
        lda #<klathe
        sta lath
        lda #>klathe
        sta lath+1
        lda #<krakte
        sta rakt
        lda #>krakte
        sta rakt+1
        lda #<khossz
        sta hossz
        lda #>khossz
        sta hossz+1
        jmp csere
gra                   ;graf. kép
        lda #<glathe
        sta lath
        lda #>glathe
        sta lath+1
        lda #<grakte
        sta rakt
        lda #>grakte
        sta rakt+1
        lda #<ghossz
        sta hossz
        lda #>ghossz
        sta hossz+1
        jmp csere
```

A csere algoritmusa:

```
CSERE [LÁTH, RAKT, HOSSZ már a tárban van]
   Y:=0
   Ciklus, amig HOSSZ>=0
      LBYTE:=(LÁTH),Y
      RBYTE:=(RAKT),Y
      Ha l->r kell, akkor (RAKT),Y:=LBYTE        (*)
      Ha r->l kell, akkor (LÁTH),Y:=RBYTE        (*)
      LÁTH:=LÁTH+1
      RAKT:=RAKT+1
      HOSSZ:=HOSSZ-1
   Ciklus vége
Eljárás vége
```

Az "l->r kell" ill. "r->l kell" jelölések annak rövidítései, hogy kértük-e a LÁTHATÓ->RAKTÁR ill. a RAKTÁR->LÁTHATÓ átvitelt; ha cserét kértünk, akkor természetesen mindkettő igaz. Hogy ezen lehetőségek közül melyiket kértük, azt a MIT byteon tároljuk: ha l->r kell, akkor a 7-es, ha r->l kell, a 6-os bitjét állitjuk 1-re (ezek ugyanis a BIT utasitással egyszerüen vizsgálhatók).

A (*)-gal jelölt sorok aszerint módosulnak, hogy az átvitelt simán, AND-del, OR-ral vagy EOR-ral kértük (a kódban a MOD1 és a MOD2 sorok). A kódolt algoritmus a következő:

```
csere                ;láth. és rakt. között
                     ;hossz db byte
        ldy #$00
        lda $ff06    ;kép
        and #$ef
        sta $ff06    ;letiltása
cscik   lda (lath),y ;ciklus indul
        sta lbyte
        lda (rakt),y
        sta rbyte
        bit mit      ;kell l->r?
        bpl lrnem    ;nem, tovább
        lda lbyte
mod1    sta (rakt),y ;művelet
        sta (rakt),y ;kirakja
lrnem   bit mit      ;kell r->l?
        bvc rlnem    ;nem, tovább
        lda rbyte
mod2    sta (lath),y ;művelet
        sta (lath),y ;kirakja
rlnem   inc lath
        bne at1      ;léptetés
        inc lath+1
at1     inc rakt     ;mindkét
        bne at2
        inc rakt+1
at2     dec hossz    ;hossz
        bne vizsg
        dec hossz+1  ;csökkentése
```

```
vizsg   lda hossz+1     ;elfogyott?
        bpl cscik       ;nem, tovább
        lda $ff06       ;kép
        ora #$10
        sta $ff06       ;engedélyezése
        rts
```

A gyorsabb végrehajtás érdekében a müveletek idejére letiltottuk a képet. Hátravan még a MOD1 ill. a MOD2 byte módosítása, de mivel ez szorosan összefügg az új utasitások végrehajtásával, később lesz szó róla.

Vegyük számba, hány funkciót várunk el programunktól, s adjunk egyúttal nevet is nekik! Hogy könnyen meg tudjuk jegyezni a kulcsszavakat, első karakterük utaljon arra, hogy grafikus vagy karakteres képről van-e szó, azaz legyen G vagy K! Második karakterük C, L vagy S legyen aszerint, hogy cseréről, a LÁTHATÓ kép RAKTÁRból töltéséről (LOAD) vagy a LÁTHATÓ kép RAKTÁRba mentéséről (SAVE) van szó! A harmadik karakterre csak akkor van szükség, ha valamilyen logikai müveletet is igénybe akarunk venni; ekkor az AND, az OR ill. az EOR kezdőbetüjével utaljunk erre. Mivel ez utóbbiaknak csak a grafikus kép töltése és mentése esetén van értelme, végeredményben az alábbi utasitásokat fogjuk használni:

```
KL
KS
KC
GL, GLA, GLO, GLE
GS, GSA, GSO, GSE
GC
```

Ez összesen tehát 12 új utasitást jelent, a BASIC-nek ezekkel való bővitését kell megoldanunk.

A COMMODORE+4 BASIC-je viszonylag egyszerü módon bővithető. Ennek módjáról elsősorban azt kell tudnunk, hogy az inicializáláskor (bekapcsolás vagy RESET után) az interpreter ROM-ból feltölti a BASIC és a KERNAL vektorok (a RAM-ban, a 3. lapon lévő) táblázatát. A későbbiekben, futása során az interpreter bizonyos utasitások és parancsok végrehajtása során a ROM-ból "kiugrik" megnézni egy-egy ilyen vektort, s az ott talált (indirekt) cimen folytatja futását. Ez a megoldás alapállapotban semmiféle különös dolgot nem jelent, mert ezek a cimek (általában) közvetlenül az őket "vizsgáló" utasitás mögé küldik vissza a programot. Kiválóan alkalmas viszont arra, hogy a vektorok átirásával olyan helyre irányitsuk a programot, ahol - még eredeti feladata folytatása e-

lött - az általunk igényelt funkciókat is elvégzi, vagy akár az eredeti tevékenység folytatása helyett egészen új ágra fusson. Hogy csak néhány példát említsünk: ennek felhasználásával működnek a különféle turbóprogramok, önindító programok készíthetők, megváltoztatható a perifériákkal való kapcsolat, egyszerűen letiltható vagy módosítható a tárban lévő program listázásának folyamata, a BASIC-sorok tokenizálása, utasítások végrehajtása stb. Sokat fogunk foglalkozni a megszakítás programozásával, erre ugyancsak az egyik vektor átírásával nyílik lehetőségünk. A BASIC-utasításokkal kapcsolatos három, egymástól jól elválasztható funkció, a tokenizálás, a listázás és a végrehajtás is egy-egy ilyen vektoron fut keresztül.

Tekintettel arra, hogy a BASIC-parancsok és utasítások végrehajtásának módjával [I] sajnálatos módon egyáltalán nem foglalkozik, sőt, csak a COMMODORE-64-es esetében lehet magyar nyelvű könyvekben az erre vonatkozó információkat felelni, ha vázlatosan is, de kénytelenek vagyunk ezzel a kérdéssel foglalkozni. A kérdéskör végleges, megnyugtató megértéséhez elengedhetetlennek tartjuk a ROM-lista tanulmányozását legalább a TEDMON, de még inkább [III] segítségével.

Mint az bizonyára ismert, a BASIC program- vagy parancssor begépelése után az input pufferbe kerül. Amennyiben sorszámmal kezdődik, az interpreter - a BASIC-utasításokat tokenjeikkel helyettesítve - elhelyezi a tárban a többi sor között, a sorszámának megfelelő helyen. Hangsúlyozzuk, hogy ekkor semmiféle szintaktikus ellenőrzést, a tokenizáláson (és néhány apróságon, mint pld. a sorszám hexadecimális címmé alakítása, a kezdő és befejező szóközök eliminálása stb.) kívül más módosítást nem végez. Ha az utasításkészletben nem szereplő utasítást írtunk be, akkor ezt - mivel a ROM-ban lévő kulcsszótáblában nem találja - a sor többi részével együtt (karakterenként) beteszi a tárba. Ha a programot listázzuk, az aktuális programsor sorszámát decimális formában kiírja, majd a nem token karaktereket (így pld. az előbb feltételezett hibás utasítás karaktereit) kódjuknak megfelelően, a tokeneket pedig a kulcsszótáblából kikeresett szöveggel írja ki. Végrehajtáskor a sorban a sorszámot mindenképpen valamilyen token követi - ha mégsem, akkor az interpreter a LET utasítást feltételezi -, a tokennek megfelelő utasítás címe

az ugrótáblából a verembe kerül, majd egy RTS hatására megkezdődik az utasítás végrehajtása. Ennek az utasításnak a szintaxisa szerint értelmezi az interpreter a sor további részét. Ilyenkor derül ki tehát az, ha hibás sort írtunk be, s a program futása hibajelzéssel megszakad.
Ha nem sorszámmal kezdődő sort, tehát parancssort adunk át, akkor az interpreter a tokenizálás és a végrehajtás funkcióját "összevonja" ugyan, de a kulcsszótáblában nem talált utasítás értelmezésénél hibajelzéssel megáll.
Hiába találtuk ki tehát olyan "ügyesen" a 12 új utasítás nevét, gépünk egyelőre nem hajlandó megérteni, mit is akarunk tőle. Meg kell "tanítanunk" az új, ún. felhasználói utasítások ismeretére. Erre lehetőség van, hiszen az elvileg felhasználható 128 token közül csak 127 foglalt, a $FE értéket az interpreter alkotói nem kötötték le, hanem segítségével lehetővé tették kétbyte-os tokenek (ezeket nevezzük felhasználói tokeneknek) s ezzel összefüggésben felhasználói utasítások definiálását. A definiáláshoz viszont elengedhetetlen az, hogy az interpreter továbbra is csak a ROM-ban lévő kulcsszótáblát pásztázza végig, így mind a tokenizálás, mind a listázás, mind pedig a végrehajtás eredeti menetét módosítanunk kell. E célból át kell írnunk az alábbi három vektor értékét:

FTOKV ($030C-030D) :a felhasználói utasítás tokenizálása
FLISTV ($030E-030F) :a felhasználói utasítás listázása a token ismeretében
FVEGRV ($0310-0311) :a felhasználói utasítás végrehajtása a token ismeretében

, mégpedig úgy, hogy az így megváltoztatott rutinok az általunk létrehozott utasítások kulcsszavaiból alkotott táblázatot használják, ugrási címeiket az általunk készített ugrótáblából vegyék, s ezáltal az ugyancsak általunk készített rutinokra jutva a kiválasztott felhasználói utasításnak megfelelő tevékenységet hajtsák végre.
Tokenizáló rutinunk helyes működéséhez fel kell állítanunk a felhasználói utasítások táblázatát. Ennek felépítésére alapvetően két különböző lehetőség van. Azonos hosszúságú kulcsszavak esetén egyszerűen egymás után írva őket egy táblázatban, index segítségével történő keresésre redukálódik a kezelése (így dolgozik pld. a JCL assembler része). Ha viszont

a kulcsszavak hossza változik (mint pld. a C+4 BASIC-jében), a kulcsszó utolsó karaktere kódja legfelső bitjének megemelésével (más szóval shifteléssel, tkp. a kód értéke $80-nal történő megnövelésével) jelezzük, hogy vége van egy kulcsszónak. (Egy, az input pufferbe került "potenciális" kulcsszónak a kulcsszótáblázat valamelyik elemével való egyezése csak az utolsó karakter kivételével jelent azonosságot, mert - éppen a kulcsszó vége shiftelése miatt - az utolsó karakternél $80 eltérést kell tapasztalnunk. Ez ad lehetőséget a COMMODORE-gépeken alkalmazott rövidítések használatára.)
Kulcsszótáblánkban utasításainkat tehát tetszőleges sorrendben felsorolhatjuk, mégis - az új utasítások funkcióját figyelembe véve, az egyszerű elágazási lehetőség biztosítása végett - a következő felépítést tartjuk a legcélszerűbbnek:

```
ktabl .byte 'glAglOglEgLgsAgsOgsEgSgCkLkSkC'
```

Ha tokenizáláskor az interpreter egy új utasítást a táblázatban felismer, egy kétbyte-os tokenre cseréli ki. Egy-egy ilyen kétbyte-os token első byte-ja mindig $FE, második byte-ja pedig az utasításnak a kulcsszavak közötti sorszáma shiftelt (azaz $80-nal növelt) értéke.
Ha listázáskor az interpreter felhasználói tokent talál, az utána következő byte értéke alapján táblázatunkból kikeresi és kilistázza a megfelelő kulcsszót.
Há már megírtuk az egyes utasítások esetén végrehajtandó gépi kódú rutinokat, akkor ezek címeit - a kulcsszótáblában meghatározott sorrendben - felsorolva készen is van az ugrótábla. Ha végrehajtáskor az interpreter $FE kódot talál, nem a ROM-béli, eredeti ugrótáblából, hanem a most készítettből teszi a verembe az ugrási címet. Megjegyezzük, hogy ez az assembly program írásakor külön munkát nem igényel, mert ha a rutinokat a nekik megfelelő új utasításokkal címkéztük, akkor a

```
ctabl .word gla,glo,gle,gl,gsa,gso,gse,gs,gc,kl,ks,kc
```

pszeudoutasítással az assemblerrel fel tudjuk építtetni. Véleményünk szerint a felhasználói utasítások használatával összefüggő ismeretek megértéséhez az alábbi programrészek és a ROM-lista tanulmányozása sok segítséget ad:

```
ftokv=$030c          ;felh. ut. tokenizálás vektor
flistv=$030e         ;felh. ut. listázás vektor
fvegrv=$0310         ;felh. ut. végrehajtás vektor
chrget=$0473         ;1 karakter olvasása
token=$8a07          ;token azonosítása
fhtok=$89d6          ;felhasználói token
nfhtok=$896c         ;nem felhasználói token
ltok=$8b9c           ;token keresése listázáshoz
       sei
       lda #<ftok    ;felh. ut.
       sta ftokv     ;tokenizálás
       lda #>ftok    ;helye
       sta ftokv+1
       lda #<flist   ;felh. ut.
       sta flistv    ;listázás
       lda #>flist   ;helye
       sta flistv+1
       lda #<fvegr   ;felh. ut.
       sta fvegrv    ;végrehajtás
       lda #>fvegr   ;helye
       sta fvegrv+1
       cli
       .
```

A felhasználói utasításokat tokenizáló és listázó rutinok:

```
ftok                 ;felh. ut. tokenizálása
       pha
       ldy #<ktabl   ;kulcsszótábla
       lda #>ktabl   ;címe
       jsr token     ;token azonosítás
       pla
       bcc vftok     ;nem felh. token
       lda $0b       ;talált
       pha           ;sorszám verembe
       jmp fhtok     ;felhaszn. token
vftok  jmp nfhtok    ;nem felh. token
flist                ;felh. ut. listázása
       tax
       sty $49
       ldy #>ktabl   ;kulcsszótábla
       sty $23
       ldy #<ktabl   ;címe
       jmp ltok      ;token keresése
```

A felhasználói utasításokat végrehajtó rutinok megírása előtt vissza kell zökkennünk eredeti feladatunkhoz, a két grafikus és a két karakteres képernyő alkalmazásához. Lássuk az egyes utasítások végrehajtását megvalósító részeket!

```
gl                   ;r->l kell
gla
glo
gle    nop           ;belép. pont
       lda #$40      ;v-bit=1
       sta mit
       jmp gra
```

```
gs                      ;l->r kell
gsa
gso
gse     nop             ;belép. pont
        lda #$80        ;n-bit=1
        sta mit
        jmp gra
gc      nop             ;r->l kell
        lda #$c0        ;l->r kell
        sta mit         ;v=n=1
        jmp gra
kl      nop             ;r->l kell
        lda #$40        ;v-bit=1
        sta mit
        jmp kar
ks      nop             ;l->r kell
        lda #$80        ;n-bit=1
        sta mit
        jmp kar
kc      nop             ;r->l kell
        lda #$c0        ;l->r kell
        sta mit         ;v=n=1
        jmp kar
```

Már nincs sok hátra, csak az, hogy a felhasználói utasitások helyes működése érdekében - mint azt megigértük -, visszatérjünk a MOD1 és a MOD2 byte-ok módositásának kérdéséhez. Ezzel kezdődik a felhasználói utasitások végrehajtását biztositó programrész:

```
fvegr                   ;felh. ut. végrehajtása
        sec
        sbc #$80        ;sorszám
        tay             ;tárolása
        and #$08
        bne sima        ;sima sta
        tya
        and #$03
        tax
        .byte $2c       ;átl.bit
sima    ldx #$03        ;a szüks.
        lda kodok,x     ;módositás
        sta mod1        ;beirása
        sta mod2
        tya
        asl a           ;2*
        tay
        lda ctabl+1,y   ;a címtáblázatból
        pha             ;a rutin
        lda ctabl,y     ;címe
        pha             ;a verembe
        jmp chrget      ;folytatás
        ;
kodok   .byte $31       ;and (cc),y
        .byte $11       ;ora (cc),y
        .byte $51       ;eor (cc),y
        .byte $91       ;sta (cc),y
```

Figyeljük meg a módosítás beírásának technikáját, a megfelelő rutin címének elhozását a táblázatból! Ismét alkalmaztuk az átlátszó BIT utasítást. Most érthetjük meg igazán, miért pont ezt a sorrendet választottuk a kulcsszavaknak: lényegesen egyszerűsödik a módosító kódok kiválasztása.

Most már tényleg csak egy kicsi, de nagyon lényeges feladatunk van: biztosítani a másik képernyő számára a területet. Az ezt megvalósító programrész a következő:

```
txttab=$2b              ;a basic terület kezdete
mvdflg=$75              ;grafikus ter. van-e?
new=$8a7b               ;basic new
ready=$867e             ;basic indítás
helyf                   ;helyfoglalás a képeknek
        lda #$70        ;a basic ter.
        sta txttab+1    ;új kezdete
        lda #$00        ;nullázás
        sta $7000       ;run-hoz
        lda #$ff        ;grafikus mód
        sta mvdflg      ;beállítása
        jsr new         ;basic new
        jmp ready       ;vissza az interpreterhez
```

Egyetlen sor igényel egy kis magyarázatot: a $7000 című byte-ot azért kell nullázunk, mert az interpreter a BASIC-területet közvetlenül megelőző byte-on (alapállapotban $1000-en, egy grafikus képernyő használata esetén $4000-en) csak 0 értéket fogad el, ellenkező esetben hibajelzéssel megáll.

A grafikus mód használata miatt felszabadul a $1001 ... $17FF terület, amely így alkalmassá válik programunk tárolására, ezért azt $1100-tól kezdődően fordíttattuk az assemblerrel a tárba. Hívása: sys 4352-vel vagy sysdec("1100")-val lehetséges.

A lemezen található DEMO program néhány példát mutat az új utasításokkal ismerkedni kívánóknak.

Kérdések és feladatok az 5. fejezethez

5.1 Legfeljebb hány grafikus képernyőt tudunk definiálni, ha csak egy karakteres képernyőre van szükségünk? Mire kell vigyáznunk, ha 3-nál többet definiálunk?

5.2 Legfeljebb hány karakteres képernyőt tudunk definiálni, ha grafikus képernyőre nincs szükségünk?

5.3 Miért van szükség a CSERE algoritmusban LBYTE és RBYTE alkalmazására, miért nem közvetlenül végezzük el a töltést?

5.4 Iktassuk ki a képkioltó részt a programból, így a töltés lefolyását - bár így is elég gyors - figyelemmel kísérhetjük!

5.5 Szervezzük meg a CSERE algoritmust úgy, hogy laponként léptetünk, s egy-egy lapon belül az indexelés adta lehetőséget kihasználjuk!

5.6 Tanulmányozzuk a BASIC interpreter kulcsszavait és ugrótáblájukat! ($818E ... $ 8470)!

5.7 Tanulmányozzuk a TEDMON parancsait és ugrótáblájukat! ($F570 ... $ F599)!

5.8 Miért nincs értelme a karakteres kép III. a gc esetében a logikai műveletek használatának?

5.9 A program melyik részét és hogyan kellene módosítanunk, ha a KÓDOK sorrendje STA, AND, ORA, EOR lenne?

5.10 A program melyik részét és hogyan kellene módosítanunk, ha a KTÁBL kódtábla értéke
'kLgLglAglOglEkSgSgsAgsOgsEkCgC' lenne?

5.11 Miért csak 128 token lehet maximálisan?

6. SKÁLA

A kissé fárasztó és terjedelmes program után most egy sokkal könnyebb feladattal foglalkozunk: a gép hanggenerátorát szólaltatjuk meg gépi kódból. Ennek és a következő fejezeteknek közös jellemzője, hogy valamilyen módon mindegyik kapcsolódik a megszakításhoz. A feladat megoldásához szükséges ismereteket [1] 2.4 és 2.6 fejezete tartalmazza.

A gép megszakításával, annak programozásával való ismerkedésünket egy igen egyszerű feladattal kezdjük: csaljunk ki hangot a gépből úgy, hogy közben a gép egyéb lehetőségei változatlanul rendelkezésünkre álljanak, tehát tudjunk pld. parancs módban számításokat végezni, BASIC-programot írni, futtatni stb.!

Az egyszerűség kedvéért mindössze egyetlen oktávra kiterjedő folyamatos skálázást várunk, azzal a megkötéssel, hogy - mivel ez hosszabb időn keresztül kissé idegesítő is lehet, - bizonyos billentyűk megnyomására hallgasson el, majd ugyanígy újra is tudjuk indítani. Ezeknek a billentyűknek a kiválasztása némi gondot okoz, hiszen - ha közben pld. éppen BASIC-sorokat írunk be - elronthatjuk velük a képernyő tartalmát. Olyan billentyűket kell tehát választanunk, amelyek együttes lenyomása egyébként soha nem fordul elő. Nem túl nagy a választék, mert a billentyűnek a képre író vagy a kurzort elmozgató hatása nem lehet, tehát csak a SHIFT, a CONTROL, a C= és az ESC gombok jöhetnek szóba. (Megtehetnénk azt is, hogy valamelyik funkcióbillentyűt az üres szöveggel definiáljuk, de ez már csorbítja a gép használatának lehetőségeit.)

Ezek közül egymagában egyik sem elegendő, mert a szerkesztés közben mindegyiket használjuk, így nem akaratunktól, hanem a beírandó szövegtől függ a ki-be kapcsolgatás; ebből kifolyólag legalább két gomb egyszerre lenyomása szükséges. Ebben a kettőben viszont semmiképpen nem szerepelhet az ESC, hiszen - újfent a beírandó szövegtől függően - ez tragikus következményekkel járhat (pld. az egész kép letörlése). A C= és a SHIFT együttesen már a két karakterkészlet közötti váltásra foglalt, így csak a CONTROL-SHIFT vagy a CONTROL-C= párok

maradnak meg; mi az utóbbit választjuk. Figyelnünk kell tehát - természetesen megszakításból -, hogy a CONTROL és a C= gombok le vannak-e nyomva. Ez viszonylag egyszerüen megoldható, mert az SHFLG ($0543) rendszerváltozónak a SHIFT lenyomásakor a 0., a C= esetén az 1., a CONTROL-nál pedig a 3. bitje magas állásba billen. Ezek a bitek egymástól függetlenül állitódnak, s a kérdéses billentyü elengedése után a megfelelő bit zérusra áll vissza. Azt kell tehát vizsgálnunk, hogy SHFLG értéke 6-e vagy sem; ha igen, akkor engedélyezett hang esetén ki, letiltott hang esetén pedig be kell kapcsolnunk a hanggenerátort.

A muzsikát most a 2. hangcsatorna segitségével fogjuk szolgáltatni - minden ugyanigy történik az 1. csatorna használata esetén. Mint az ismert, a hangot alapvetően a TEDSD ($FF11) regiszter bitjeinek állitásával tudjuk szabályozni: az 5. bit magas állása engedélyezi, alacsony állása letiltja a 2. csatornát, a hang intenzitása pedig az alsó 4 bit változtatásával állitható be (maximálisan 8-ra). A 2. hanggenerátor engedélyezését ill. tiltását tehát TEDSD 5. bitjének állitgatásával oldhatjuk meg. Ennek kódolását a legegyszerübben a következőképpen lehet elvégezni:

```
ki      lda tedsd
        and #$df
        sta tedsd

        .
        .
        .

be      lda tedsd
        ora #$20
        sta tedsd

        .
        .
        .
```

, egy elágazással kikapcsoláskor a KI, bekapcsoláskor a BE részre vezérlik a programot. Mi most egy bonyolultabb módszert választunk, annak szemléltetésére, hogy egy bit magasra állitásához ill. törléséhez miként lehet ugyanazt a programrészt felhasználni. Szükségünk van azonban annak jelzésére, hogy a változtatást végre kell-e hajtanunk vagy sem; erre a célra - mint az a gépi kódú programozásban igen gyakori - egy ENG byte-ot tartunk fenn, amelynek legfelső bitje állásától függően hajtjuk végre a módositást. Hogy ez helyesen működjék, minden változtatáskor MSB értékét meg kell

változtatnunk, de ez nagyon egyszerüen megvalósitható az EOR #$FF utasitás segitségével. A vizsgálatot és a ki-be kapcsolást megvalósitó programrész:

```
shflg=$0543          ;c= és ctrl jelzője
tedsd=$ff11          ;hang vezérlő regiszter
bonyod bit eng       ;engedély?
       bne vizsg     ;nincs, vizsg.
       lda tedsd     ;hang
       ora #$20      ;bekapcsolása
       sta tedsd     ;2. hanggen.
vizsg  lda shflg
       cmp #$06      ;ctrl+c=?
       bne vege      ;nem, marad
       lda eng       ;igen
       eor #$ff      ;az engedélyt
       sta eng       ;átirni
       lda tedsd     ;2. hangg.
       eor #$ff      ;engedélyt
       eor #$df      ;átirni
       sta tedsd
vege   ...
```

Még a technikai részletekhez tartozik az is, hogy a megszólaltatni kivánt hangokra vonatkozó kétbyte-os adatokat - amelyeknek $FF0F-be ill. $FF10-ba irása határozza meg a hangok frekvenciáját - táblázatba foglaljuk (rendre az 523, 494, 440, 392, 349, 330, 294, 262 Hz frekvenciájú hangokról van szó):

```
hanga .byte $2a,$1e,$02,$e3,$c0,$ad,$83,$55 ;alsó byte-ok
hangf .byte $03,$03,$03,$02,$02,$02,$02,$02 ;felső byte-ok
```

A hangokat emelkedő sorrendben akarjuk hallani, de mivel egyszerübb csökkenő ciklusváltozót alkalmaznunk, csökkenő sorrendben irtuk öket a táblázatba. A lényeget nem érinti, hogy egy egyszerü dallamról vagy egy egész szimfóniáról vane szó, ezért ugyanilyen táblázatot kellene felépitenünk a hangokból bonyolultabb dallamok esetén is.

Az egyes hangok időtartamát - amelyet, pontosan amelynek kettes komplemensét az SDT ($04fd) és az SDT+2 byte-okba kell beolvasnunk - szintén táblázatba kell foglalnunk; erre most azért nincs szükségünk, mert minden hangot azonos ideig szeretnénk hallani.

Az SDT és az SDT+2 byte-ok 0-ig való növeléséről a megszakitás gondoskodik, ezzel tehát nem kell törödnünk. A megszakitással azonban többlettevékenységet is szeretnénk végeztetni. Amint megszakitás érkezett, elsőként a hanggal kapcsolatban kell intézkednie; ha egy hang lefutott, be kell ol-

vasnia a következő hang adatait, s állandóan figyelemmel kísérni, hogy akarunk-e az engedélyezésen változtatni; ha igen, akkor ezt is el kell intéznie. Mindezek befejezése után a MSZF ($CE0E) címen folytathatja az eredetileg is elvégzendő tevékenységét. Most tehát nem "átírjuk", csupán kibővítjük a megszakító rutint. A bővítés algoritmusa:

```
MEGSZ
   Ha NEM(SDT=0 és SDT+2=0),
      akkor MSZF
      különben MUT:=MUT-1
               Ha MUT<0, akkor MUT:=7
               Y:=MUT
               $FF0F:=HANGA,Y
               $FF10:=HANGF,Y
               SDT:= idő alsó byte-ja
               SDT+2:= idő felső byte-ja
               BONYOD
MSZF [megszakitas folytatása]
```

Ezek alapján a kódolt programrész:

```
sdt=$04fd             ;hang időtartama
mszf=$ce0e            ;megsz. folytatása
      ;
megsz                 ;új megszakitás
      lda sdt         ;idő lejárt?
      ora sdt+2
      cmp #$00
      bne vege        ;még nem, tovább
      dec mut         ;igen, új hang
      bpl ugr         ;van még?
      lda #$07        ;nincs,->eleje
      sta mut
ugr   ldy mut         ;köv. hang
      lda hanga,y     ;betöltése
      sta $ff0f       ;alsó byte
      lda hangf,y
      sta $ff10       ;felső byte
      lda #$f8        ;időzítés
      sta sdt         ;alsó byte
      lda #$ff
      sta sdt+2       ;felső byte
      .
      .    [itt van a bonyod rész]
      .
vege  jmp mszf        ;tovább
```

A megszakitásbővítés érvényre jutásához át kell irnunk az MSZV ($0314) vektor értékét úgy, hogy a most elkészített rutinra mutasson. Ezt a következő programrész végzi el:

```
mszv=$0314          ;megsz. vektor
      ;
      ;
      sei
      lda #<megsz
      sta mszv      ;új
      lda #>megsz
      sta mszv+1    ;megszakítás
      lda tedsd
      ora #$08      ;maximális
      sta tedsd     ;hangerő
      lda #$08      ;mutató kezd.
      sta mut
      cli
      rts
```

Kérdések és feladatok a 6. fejezethez

6.1 Írjuk be a táblázatba egy egyszerű dallam hangjait! Készítsük el a változó hosszúságú hangok táblázatát is, s módosítsuk ennek megfelelően a programot!

6.2 Írjuk át a programot az 1. hanggenerátorra!

6.3 Írjunk kétszólamú zenét megszólaltató programot!

6.4 Írjuk át az engedélyező részt úgy, hogy nem használjuk az ENG byte-ot!

6.5 Írjunk olyan, megszakításból működő programot, amelynek hatására minden billentyű lenyomásakor más-más hang szólal meg! Hallgassuk meg így módon pld. ennek a mondatnak a "muzsikáját"!

6.6 Mi történik, ha a módosítást a SHIFT, a C= és a CONTROL gombok együttes lenyomásával próbáljuk végrehajtani? Hogyan kell a programot megváltoztatnunk, hogy a módosítás erre következzék be?

6.7 Írjunk olyan BASIC-programot, amely használja a 2. hanggenerátort! Hogyan befolyásolja a megszakításból működő programunk a BASIC-program futását? Mi változik meg akkor, ha a gépi kódú programban a hangerő beállítását végző részt áttesszük a megszakítási rutinba?

7. HANGSEBESSÉG

Ebben a fejezetben egy mérési feladattal foglalkozunk, egyszerü példát mutatva arra, hogyan lehet a számitógépet ilyesfajta feladatok megoldásához alkalmazni. A fejezet megértése különösebb előismereteket nem igényel.

Feladatunk a hangsebesség megmérése. Ehhez elvileg csak egy hangszóró és egy mikrofon, valamint egy óra és egy mérőrúd szükséges: a hangszórón kibocsátunk egy hangot, megmérjük azt az időt, mig ez a hang a mikrofonhoz ér, lemérjük a hangszóró és a mikrofon távolságát, és e két adatból máris kiszámithatjuk a hang sebességét.

Baj van azonban a pontossággal: hol keletkezik a hangszórón belül és hová érkezik a mikrofonon belül a hang, hogy mérhető meg pontosan ezeknek a helyeknek a távolsága? Pontos választ erre nem tudunk adni. Megkerülhető viszont a probléma, ha egy olyan cső áll rendelkezésünkre, amelyben egy egyenes mentén helyezkedik el egy hangszóró és két, azonos felépitésü mikrofon (pontosan ilyen sorrendben, mindkét mikrofon a hangszóró felé fordítva). Ez az eszköz ugyanis lehetőséget ad arra, hogy ún. differencia módszerrel mérjünk, azaz a fentebb vázolt elképzeléstől eltérően a hangimpulzusnak ne a hangszórótól a mikrofonig való eljutásának idejét, hanem a hangnak az egyik mikrofonhoz és a másik mikrofonhoz érkezése között eltelt időt mérjük. Nagyon egyszerüen és viszonylag pontosan tudjuk ugyanis az ehhez tartozó távolságot mérni, mert a két mikrofon azonos geometriai felépitéséből adódóan ez a távolság pontosan megegyezik pld. a két mikrofon peremének távolságával.

Most már csak az idő pontos mérését kell megoldanunk. Ehhez szükségünk van a TECHNOMIR család digitális input/output moduljára, amely a PORT ($FE18) cimen érhető el; ennek 8 bitje közül pld. a hangszórót a 0. bitre , az első mikrofont a 7., a másodikat a 6. bitre kötjük. A 0. bit irásakor a hangszóró egy impulzust bocsát ki, amely a mikrofonokhoz érve a 7., majd 6. bitet magasra állitja. Ezt a PORT cim állandó olvasásával figyelemmel tudjuk kisérni, s ha a 7. bit bebillenésekor elindítunk egy számlálót, majd a 6. bit bebillenésekor

megállítjuk, máris rendelkezésünkre áll egy viszonylag pontos óra (sokkal pontosabb, mint amilyen pontossággal a távolságot mérni tudtuk). A hangszóró és a mikrofonok tökéletlensége ill. a zavaró környezeti hatások miatt azonban számitanunk kell néhány eshetőségre: pld. hogy a hangszóró által kibocsátott impulzus valamilyen okból nem érkezik meg az első mikrofonhoz, vagy oda igen, de a másodikhoz nem. Ezek következménye az volna, hogy a gép a végtelenségig várna a számláló elindítására ill. megállítására, ezért ezt valahogy meg kell előznünk.

Ha ismerjük - márpedig ismerjük - a hang sebességének hozzávetőleges értékét, meg tudjuk becsülni, hogy kb. mennyi idő alatt kell elérnie a hangszórótól az első mikrofonig, ill. onnan a másodikig. Ezt az időtartamot - akár többszörösével -felülről becsülve tudunk egy olyan időkorlátot mondani, amelynél tovább már nem érdemes sem az egyik, sem pedig a másik esemény bekövetkezésére várni; nevezzük ezt a tétlen várakozás időtartamának. Ha pld. a cső hosszúsága néhány méteren belül van, akkor a fél másodperces várakozás mindenképpen eléggé bőséges ahhoz, hogy a hang - ha akar, - megérkezzék, mert ez alatt az idő alatt normál körülmények között 170 m-t kell megtennie, s ez a távolság a hőmérséklet ill. a nyomás változásával sem csökkenhet pld. 10 m alá (most természetesen csak az általunk megvalósitható körülményekre gondolunk). Egy 0.5 másodperc körüli tétlen várakozás tehát mindenképpen elegendő.

Előrebocsátjuk, hogy - mivel a processzor órajelét fogjuk használni - a pontosság érdekében a mérés tartama alatt le kell tiltanunk a megszakitást és a kép kijelzését. A tétlen várakozás maximális idejét TVI, a számláló értékét T jelöli. Lássuk ezek után a mérést megvalósitó algoritmust!

```
HSEB  Kép és megszakitás tiltása
      PORT(0):=1  {hangimpulzus indítása}
      Ciklus, amig T<=TVI és PORT(7)=0
         T:=T+1
      Ciklus vége
      Ha T>TVI, akkor T:=0 {rossz mérés}
                különben T:=0 {mérés indul}
                          Ciklus, amig T<=TVI és PORT(6)=0
                             T:=T+1
                          Ciklus vége
                          Ha T>TVI, akkor T:=0 {rossz mérés}
      Elágazás vége
      Kép és megszakitás engedélyezése
```

Eljárás vége

Az eljárás végeztével a számláló 0-t mutat akkor, ha a mérés sikertelen volt, ellenkező esetben pedig a mért időt tartalmazza. Ahhoz azonban, hogy ezt az időt másodpercekben (ill. annak törtrészeiben) ismerjük, két dologra van szükség: egyrészt az órajel frekvenciájának értékére, másrészt pedig annak ismeretére, hogy hány óraciklus kellett a számláló értékének 1-gyel való növeléséhez. Ezekből kiszámíthatjuk, hogy hány óraciklus futott le a mérés tartama alatt.

Az órajel frekvenciája - a szakirodalom szerint is - gépenként más és más, kb. 0.8 és 1.8 MHz közötti érték. Nem tudunk mit tenni, meg kell tehát mérnünk, mégpedig ugyanolyan körülmények között, mint ahogyan a hangsebesség mérését is végezzük (azaz letiltott kép és megszakítás mellett).

Módszerünk a következő: készítünk egy három byte-os számlálót, 0-ról 0-ra futtatjuk, megszámoljuk, hogy futása hány óraciklust igényel, majd stopperral megmérjük a futás idejét. Mivel ez - a fenti frekvenciákkal számolva - 1 percnél hosszabb, viszonylag pontosan mérhető, tehát elég jó eredményre számíthatunk. A szubrutin a következő:

```
t=$e0                   ;számláló helye
ted1=$ff06              ;képtiltáshoz
freki
        sei             ;2
        lda ted1        ;4
        and #$ef        ;2
        sta ted1        ;4
        lda #0          ;2
        sta t+2         ;3
        sta t+1         ;3
        sta t           ;3
ciklus  inc t+2         ;5
        bne ciklus      ;3-2
        inc t+1         ;5
        bne ciklus      ;3-2
        inc t           ;5
        bne ciklus      ;3-2
        lda ted1        ;4
        ora #$10        ;2
        sta ted1        ;4
        cli             ;2
        rts             ;
```

A rutint olyan helyre fordítjuk, ahol biztosan nem kell a relatív ugrásoknak átlépniük a laphatárt. A rutin lefutása a mi gépünkön 79.3 másodpercig tartott, s mivel az utasítások összesen 134 678 271 gépi ciklust igényelnek, ez kb. 1.698 MHz-es órajelet jelent.

Másik feladatunk a HSEB eljárás kódolása, majd ennek kapcsán a számláló-órajel átváltás arányának a meghatározása. A mérést megvalósító szubrutin:

```
port=$fe18          ;kapu
      ;
hseb                ;hangsebesség mérése
      sei           ;megszakitás
      lda ted1      ;és kép
      and #$ef      ;tiltása
      sta ted1
      lda #$01      ;indul
      sta port
      lda #0        ;számláló
      sta t+1       ;nullázása
      sta t         ;várakozáshoz
varak lda port      ;1. mikr.-ra vár
      and #$80      ;a 7. biten
      beq jon       ;jön, tovább
      inc t+1
      bne varak     ;különben
      inc t         ;számlál
      bne varak
      beq kesz      ;nem jött, rossz mérés
      lda #$00      ;számláló
      sta t         ;nullázása
      sta t+1       ;a méréshez
meres lda port      ;2. mikr.-ra vár
      and #$40      ;a 6. biten
      beq kesz      ;jön, kész
      inc t+1
      bne meres     ;különben
      inc t         ;számlál
      bne meres
kesz  lda ted1
      ora #$10      ;kép
      sta ted1      ;és
      cli           ;megszakitás
      rts           ;mehet
```

A kód ismeretében már kiszámithatjuk, hogy a mért időtartam 4103*(t)+16*(t+1) óraciklus, a tétlen várakozás időtartama pedig 1050368 óraciklus. Ez utóbbi (az órajel frekvenciája ismeretében) kb. 0.6 másodpercnek felel meg, ami a fentebb mondottak értelmében tökéletesen megfelel a célnak.

Mivel ehhez nagyon hasonló feladat olyan program készitése, amelynek a segitségével megmérhető, hogy mennyi idő telt el két billentyű megnyomása között, programunkat kissé átalakitva ezt a funkciót is beépitettük (a SZIMU szubrutin tartalmazza). Nem pontosan egyezik a hangsebességet mérő programrésszel, tőle a megszakitás engedélyezésében tér el, igy vonható ugyanis a legegyszerűbb formai párhuzam a kettő között, s igy is mindössze néhány ezrelék a hibája. (Aktiv

megszakitásnál a lenyomott billentyü kódja mindig a $C6 byte-ba kerül, a megszakitás letiltásával viszont megszünik az ezt kihasználó lehetőség.)

A programot BASIC-ből az USR-függvénnyel hívjuk, a következő választási lehetőségekkel:

P=USR(2) - az órajel frekvenciájának mérése

P=USR(1) - a mérés szimulálása a "j" és a "k" gombbal

P=USR(0) - a hangsebesség mérése

Ne mulasszuk el a rutin kezdőcimét ($6666 = 26214) az USR-vektorba irni! A paraméter átvételét, a megfelelő funkcióra való vezérlést már ismert módon oldjuk meg:

```
facint=$a327          ;fac->egész
illq=$991c            ;ki:"ill. q."
param=$65             ;fac paraméter része
        ;
        jsr facint    ;par.->egész
        ldx param     ;választás
        bmi hiba      ;<0, hiba
        cpx #$03      ;>=3?
        bmi mehet     ;nem, mehet
hiba    jmp illq      ;igen, ki: "ill."
mehet   cpx #$01      ;szimuláció?
        beq szimu     ;igen
        cpx #$02      ;órajel?
        beq freki     ;igen
hseb    ...
```

A feladathoz készitett BASIC-program kényelmes lehetőséget biztosit a mérés lebonyolitására.

Kérdések és feladatok a 7. fejezethez

7.1 Mekkora hibát okoz, ha a képet a mérés idejére nem tiltjuk le?

7.2 Irjuk át a SZIMU részt pontos mérésre alkalmassá (a billentyüzet megszakitástól független lekérdezésének módja [I] 2.5 fejezetében megtalálható)! Hasonlitsuk össze a két mérési eredményt!

7.3 Mennyi lesz az előző feladatban a tétlen várakozás ideje?

8. PULZÁL

A most következő feladat mintájára igen látványos hatásokat elérő programokat készithetünk, s néhány, a gépi kódú programozás lehetőségeit kihasználó, szellemes megoldást is látunk. Öt, erre a célra kiválasztott karaktert fogunk megszakitásból változtatni, ezáltal a shift-space (SHSP) egy pulzáló csillagként, a hatványozás jele (FNY) felfelé, a balra mutató nyil (BNY, nem azonos a kurzort balra mozgató billentyüvel, hanem a shiftelt =-jel!) lefelé, @-karakter balra, a]-jel pedig jobbra forgó hullámvonalnak látszik a képen. A fejezetben foglaltak megértéséhez ismernünk kell [I] 2.2 pontját, kiváltképpen a karaktergenerátor müködését, RAM-ba helyezését.

Karakterkészletünket a $7000 cimtől kezdődően fogjuk elhelyezni (természetesen a ROM-ból másoljuk át, s csak ezt a néhány karaktert módositjuk). Az egyes karaktereket definiáló 8-byte-os részek kezdete:

```
SHSP : $7300
FNY  : $70F0
BNY  : $70F8
@    : $7000
]    : $70E8 .
```

A pulzálás megvalósitásával kezdjük algoritmusunk elkészitését. Ehhez - a pulzáló csillag négy állapotának megfelelő négy karakter byte-jaiból - négy táblázatot készitünk:

```
m1      .byte $00,$00,$00,$00,$00,$00,$00,$00
m2      .byte $00,$00,$00,$18,$18,$00,$00,$00
m3      .byte $00,$00,$24,$18,$18,$24,$00,$00
m4      .byte $00,$42,$24,$18,$18,$24,$42,$00
```

, majd az egyes táblázatok kezdőcimét egy újabb táblázatba füzzük:

```
mut     .word m1, m2, m3, m4 .
```

Ennek a táblázatnak a cimeit ciklikusan olvasva, majd az igy kapott kezdőcimektől számitott 8 byte-ot $7300-tól a tárba másolva, pontosan a kivánt hatást fogjuk elérni. Hogy a MUT táblázatból melyik byte-ot olvastuk ki utoljára, a KÖR cimen tároljuk. A pulzálás algoritmusa:

```
PULZÁL
   Y:=KÖR
   Y:=Y+1
   PCÍM:=MUT,Y
   Y:=Y+1
   PCÍM+1:=MUT,Y
   Y:=Y+1
   KÖR:=Y
   X:=7
   Ciklus, amig X>=0
      SHSP,X:=PCÍM,X
   Ciklus vége
Eljárás vége
```

Térjünk át a forgatásokra! Itt is lehetne ezt a módszert alkalmazni, de túl sok tárra lenne hozzá szükség: minden karakter 64 byte-ot igényelne, s a program sem volna érdekes. Sokkal értelmesebbnek tünik a rendelkezésünkre álló léptető és forgató utasitásokat alkalmazni, már csak azért is, mert ha igy választunk, akkor bármiféle karakternek a forgatására alkalmas eljárásokhoz jutunk.

Meg kell terveznünk a majdan felfelé ill. lefelé küszö hullámvonalat:

```
álló .byte $40,$30,$0c,$02,$02,$0c,$30,$40
```

Különbséget kell tennünk a vizszintes és a függőleges irányú forgatások között, mert ezek egymástól lényegükben térnek el: mig a függőleges irányú forgatásnál a karakter byte-jait cserélgetjük, de egy-egy byte-on belül semmiféle változás nincs, addig a vizszintes forgatás esetében a karakter minden byte-ja a helyén marad, csak a bitjei változnak meg. A függőleges forgatás algoritmusa annyira egyszerü, hogy a le- és fölfelé irányuló forgatást közös ciklusban (bár két ciklusváltozóval) is megoldhatjuk (ST a vermet jelöli):

```
FFORG
   X:=6
   Y:=0
   FNY,0->ST [FNY legfelső byte-ja]
   BNY,7->ST [BNY legalsó byte-ja]
   Ciklus, amig X>=0
      BNY+1,X:=BNY,X [1-gyel lejjebb]
      FNY,Y:=FNY+1,Y [1-gyel följebb]
   Ciklus vége
   ST->BNY,0 [BNY legfelső byte-ja]
   ST->FNY,7 [FNY legalsó byte-ja]
Eljárás vége
```

Ez egy tipikusan gépi kódú szemléletet tükröző algoritmus, igy kódolása roppant egyszerü:

```
fforg                   ;függ. forgatások
        ldx #$06
        ldy #$00        ;mentés
        lda $70f0       ;felnyíl
        pha             ;eleje és
        lda $70ff       ;balnyíl
        pha             ;vége
fcikl   lda $70f8,x     ;balnyílban
        sta $70f9,x     ;lefelé
        lda $70f1,y     ;felnyílban
        sta $70f0,y     ;felfelé
        iny
        dex
        bpl fcikl
        pla             ;vissza
        sta $70f8       ;balnyíl eleje
        pla             ;és
        sta $70f7       ;felnyíl vége
        .
        .
        .
```

A vízszintes forgatáshoz is el kell készitenünk a hullámvonalat:

```
fekvő .byte $00,$81,$42,$42,$24,$24,$18,$00
```

A vízszintes (pld. balra) forgatáskor szemeljük ki a karakter egy byte-ját! A balra forgatáshoz az egyik lehetséges módszer az, hogy megvizsgáljuk a byte legfelső bitjének értékét, ha ez magas, akkor a carry-t is magasra, ha alacsony, a carry-t is alacsonyra állitjuk, majd végrehajtunk egy balra rotáló müveletet. A jobbra forgatásnál hasonlóképpen járunk el. Nem túl szép megoldás, még akkor sem, ha egy átlátszó BIT-utasitás segitségével egyszerüsitjük az elágazást (főként a jobbra forgatásnál nehézkes, mert itt még a bitvizsgálat is körülményesebb).

Sokkal elegánsabb, s a jobbra forgatás esetére is egyszerübben alkalmazható, ha a következőt csináljuk: kiforgatjuk a szélső bitet a carry-be, elvermeljük a jelzőbiteket, visszaléptetjük a byte-ot, kivesszük a veremből a jelzőbiteket, majd az innen kapott carry-vel újra forgatjuk a byte-ot. Ez igy igen bonyolultnak tünik, de az algoritmus jól mutatja a lényeget (SR a státuszregiszter):

```
VFORG
   X:=7
   Ciklus, amig X>=0 [8 byte van]
      @,X balra forgatása
      SR->ST [carry a verembe]
      @,X jobbra léptetése [de lehetne forgatása is!]
      ST->SR [előző carry vissza a veremből]
      @,X balra forgatása [carry-vel együtt]
      ],X jobbra forgatása
      SR->ST [carry a verembe]
      ],X balra léptetése [de lehetne forgatása is!]
      ST->SR [előző carry vissza a veremből]
      ],X jobbra forgatása [carry-vel együtt]
   Ciklus vége
Eljárás vége
```

Talán furcsának tűnik, de - mivel a pulzálásnak és a függőleges irányú forgatásnak egyaránt 8-8 byte-ot kell másolnia, kézenfekvőnek tűnt, hogy kódoláskor egyetlen ciklus felhasználásával szervezzük meg mindkét tevékenységet:

```
        ;pulzálás
        ldy kor
        cpy #$08        ;körbement?
        bne tolt        ;nem, tovább
        ldy #$00        ;igen, újra
tolt    lda mut,y
        sta vcikl+1     ;az akt.
        iny             ;rész
        lda mut,y       ;betöltése
        sta vcikl+2     ;pulz.-hoz
        iny
        sty kor
vforg                   ;vizsz. forgatások
        ldx #$07        ;áttöltés
vcikl   lda $7777,x     ;sh+space-be
        sta $7300,x
        rol $7000,x
        php             ;ill.
        lsr $7000,x     ;rotálása
        plp             ;kukac-ban
        rol $7000,x     ;balra
        ror $70e8,x
        php             ;és
        asl $70e8,x
        plp             ;zárójel-ben
        ror $70e8       ;jobbra
        dex
        bpl vcikl
```

Mivel a megszakitás igen gyakori (másodpercenként 100-szor következik be), ha minden megszakitás minden funkciót működtet, nagyon zavaró lesz a változások gyorsasága. Ennek kiküszöbölése végett egy számlálót alkalmazunk, amelynek állásától függően engedélyezzük a pulzálást ill. a forgatást, a következőképpen:

```
MEGSZ .
   $FF12:=#$C0 [a karaktergenerátor RAM-ban van]
   $FF13:=#$70 [$7000-től kezdve]
   Ha SZÁML/2 egész,
       akkor Ha SZÁML/32 egész, akkor PULZÁL
             Elágazás vége
             FFORG
             VFORG
   Elágazás vége
Eljárás vége
```

A teljes program egy olyan rutint is tartalmaz, amely a 25*40-es ill. a 23*38-as képernyő közötti "keretet" a most definiált, forgó hullámvonalakkal, a sarkokat pedig a pulzáló csillaggal tölti ki, ennek megvalósitását is érdemes tanulmányoznunk.

Kérdések és feladatok a 8. fejezethez

8.1 Készitsünk olyan balra forgató szubrutint, amely egy, az átlátszó bit utasitást használó elágazás segitségével működik!

8.2 Változtassuk meg a lassitások mértékét úgy, hogy lassan forgó hullámvonalat és ennél gyorsabban pulzáló csillagot kapjunk!

8.3 Változtassuk meg a programot úgy, hogy a csillagnak ne csak a növekedése, hanem a fogyása is fokozatos legyen!

8.4 Készitsünk olyan programot, amely egy karaktert úgy pulzál 9 ütemben, hogy a teli karakterből (inverz space), azaz 8*8-as négyzetből kiindulva, a bal felső sarkában lévő, 7*7, 6*6, ... 2*2, 1*1 méretű négyzetre zsugorodik, végül eltünik!

8.5 Készitsünk programot, amely egy karaktert "végigúsztat" a képernyő egy vizszintes során! (Ehhez természetesen mindig 2 karaktert kell balra forgatni!)

8.6 Készitsünk olyan programot, amely (az előző feladat szerint) egy karaktert balra, egy másikat pedig jobbra "úsztat" ugyanabban a sorban, egymással szemben! Ügyeljünk az "áttünésre" is, azaz ütközés közben ill. utána mindkettő olyan maradjon, mint előtte volt!

8.7 Készitsünk függőlegesen felfelé görgő fényujságot, azaz a legalsó sorba irt szöveg gördüljön felfelé végig a képernyőn!

9. SOROK

Ebben a fejezetben egy néhány soros programmal foglalkozunk. Amilyen rövid, annyira látványos: a képernyő mind a 25 sorában egyidejüleg más és más karakterkészlet hatását engedi érvényre jutni. Megértéséhez szükséges [I]-ből a megszakitások ismertetéséről szóló 2.4 fejezetet tanulmányozni.

A lemez KARAKTEREK nevü programfile-ja 25 különböző karakterkészletet tartalmaz, betöltéskor ezek folyamatosan kerülnek a RAM-ba, a $1400 cimtől kezdődően. A szokásos módon (a TED $FF12 és $FF13 regisztereinek beállitásával) bármelyiket használni tudjuk. Ha ezt a használatot csak a kép egy vizszintes sávjában akarjuk elérni, a rasztermegszakitás segitségével a sáv elejének megfelelő rasztersornál be-, a sáv végének megfelelőnél pedig visszaállitjuk a fenti regisztereket, igy előtte és utána az eredeti, ROM-ban lévő, a sávban pedig az általunk meghatározott RAM-béli karakterkészlet érvényesül. Amennyiben a sáv végén nem az eredetire kapcsolunk vissza, hanem egy másik, ugyancsak RAM-ban lévő karakterkészletre, úgy ez jut érvényre.

Soronkénti rasztermegszakitásra van tehát szükségünk. Mint tudjuk, egy-egy karaktersor 8 rasztersorból áll, feladatunk tehát azt elérni, hogy minden 8. rasztersorban megszakitás következzék be, s minden alkalommal új karakterkészletre mutasson $FF13 értéke. Azt viszont nekünk kell megmondanunk, hogy mely soroknál kérjük a megszakitást, nehogy az éppen egy karaktersor közepén jöjjön, s az aktuális karakterkészlet cimét is ki kell jelölnünk. Ezeket az értékeket két táblázatba foglaljuk: A rasztersorok táblázata:

```
TFF0B .BYTE $0A,$12,$1A,$22,$2A,$32,$3A,$42
      .BYTE $4A,$52,$5A,$62,$6A,$72,$7A,$82
      .BYTE $8A,$92,$9A,$A2,$AA,$B2,$BA,$C2
      .BYTE $CC,$02
```

Az egyes karakterkészletek kezdőcimének táblázata (felső byte):

```
TFF13 .BYTE $14,$18,$1C,$20,$24,$28,$2c,$30
      .BYTE $34,$38,$3C,$40,$44,$48,$4C,$50
      .BYTE $54,$58,$5C,$60,$64,$68,$6C,$70
      .BYTE $74,$78
```

Most először találkozunk olyan esettel, amikor a megszakítást nem csupán kibővítjük, hanem magát a megszakítási rutint átírjuk: a most említett feladatokon kívül csak a minimálisan szükséges tevékenységet (a magnó és a hang kezelését, valamint a billentyűzet figyelését) engedjük elvégezni, majd a $FCC3 ponton térünk vissza az eredeti rutinhoz.
Látszólag nincs más tennivalónk, mint a megszakításrutinnal a megfelelő táblázatokból ciklikusan betöltetni egy-egy értéket a $FF13 ill. a $FF06 regiszterekbe. Ekkor viszont azt a furcsa - bár korántsem meglepő - dolgot tapasztaljuk, hogy minden olyan alkalommal, amikor a billentyűzethez nyúlunk, a kép teljesen zavaros lesz. Ennek magyarázata egyszerű: egy rasztermegszakítás után a megszakítórutin - ha a billentyűzetvizsgáló rutin semmit nem észlelt - hamar befejeződik, a következő, tehát 8 sorral lejjebb lévő megszakítás minden nehézség nélkül érvényre tud jutni, más szóval: 8 rasztersor képernyőre írása elegendő időt hagy a megszakítórutin lefutásához. Más a helyzet viszont akkor, ha a billentyűzetet használjuk, mert ebben az esetben a megszakításra a lenyomott billentyű regisztrálásával kapcsolatosan még jó néhány feladat hárul, s csak ezek elvégzése után áll készen a következő megszakításra. Bár ez nagyon gyorsan lezajlik, a képkiírás még gyorsabb, s ez okozza a kép zavarossá válását.
Mi akkor a megoldás? Egyszerűen az, hogy a 8 rasztersoros megszakítási szakaszokban nem szabad engedélyeznünk a magnóval, a hanggal és a billentyűzettel kapcsolatos tevékenységet. Valamikor azonban mégis csinálnunk kell, de erre bőven van időnk, mert az összesen 311 rasztersornyi képkiírásból a mi 25 sávunk összesen 201 (25*8+1) rasztersort érint, egy 110 soros tartomány ("110 sornyi idő") még mindig rendelkezésünkre áll! Úgy kell tehát a megszakítást elkészítenünk, hogy a magnóval, a hanggal és a billentyűzettel kapcsolatos tevékenység csak a $CC. (=204.) rasztersorban történt megszakítás után jusson szóhoz. Ennek érdekében azonban jegyeznünk kell azt, hogy hányadik körben járunk; erre egy byte (ZPMUT) elegendő. Olyan sokat beszéltünk már a problémákról, hogy ezek után megdöbbentően rövidnek tűnik a program:

```
zpmut=$e7
hang=$cecd          ;hang kezelése
bill=$db11          ;billentyüzet kezelése
magno=$cfbf         ;magnó kezelése
```

```
megsz   lda $ff09
        sta $ff09
        lda #$c0        ;karakterkészlet
        sta $ff12       ;ram-ban
        ldx zpmut
        lda tff0b
        sta $ff0b,x     ;köv. rasztermegszakitás sora
        lda tff13
        sta $ff13,x     ;karakterkészlet helye
        inx
        cpx #$1a        ;a kép alján tartunk?
        bne ugr01       ;nem, ugorj!
        jsr hang        ;hang
        jsr magno       ;magnó
        jsr bill        ;bill.
        ldx #$00        ;nullázás
ugr01   stx zpmut
        jmp $fcc3       ;megsz. folyt.
```

A programnak helyet kellett keresnünk, mert a karakterek kiszorítottak minket megszokott helyünkről. Mi a $1380 (=4992) helyet találtuk a legalkalmasabbnak, igy programunk sys4992-re indul, s átirja a $0314-en lévő megszakitásvektor cimét MEGSZ-re, ettől kezdve az új megszakitás jut érvényre.
Kivételesen a betöltő BASIC-programra is érdemes odafigyelni. Valamilyen módon meg kellett oldanunk az igen terjedelmes, 105 blokknyi karakterkészlet gyors betöltését. Ha a BASIC-program inditásakor a gyorstöltő HYPRA LOAD már a tárban van, akkor minden megy tovább, mert elsőként a karaktereket tölti be, majd beolvassa és elinditja a gépi kódú programot, azaz aktivizálja az új megszakitást, s megjelenik a képen az angol ABC 26 betüje, soronként más és más karakterekkel. Ha a HYPRA LOAD még nincs a tárban, akkor a BASIC-program először ezt tölti be, majd ismételten saját magát (!), s innen minden a fentiek szerint folytatódik.

Kérdések és feladatok a 9. fejezethez

9.1 Vizsgáljuk meg, hogyan gondoskodik a BASIC-program a HYPRA LOAD betöltéséről!

9.2 Készitsünk programot, amely a 25*40 karakteres zöld szinü képernyő melletti keretrészt pirosra, a fölötte lévő keretrészt kékre, az alatta lévőt pedig sárgára festi!

9.3 A $0312-$0313 és a $0316-$0317 vektorok is a megszakitásokkal kapcsolatosak. Mi történik, ha a mostani program futása alatt átirjuk őket?

10. GVETÍTÉS

A gépi kódú programozás tanulásához elengedhetetlenül szükséges a számitógép belső felépitésének alapos ismerete. Nem elsősorban a chipek, a vezetékek stb. összességére, hanem az ún. "szoftver-felépitésre" gondolunk. Arra, hogy hol és milyen formában vannak az adatok és a programok a tárban, milyen rendszerváltozók vannak, hol és milyen beosztással helyezkednek el a képernyők, hogyan érhető el a billentyüzet stb. Az ezzel kapcsolatos legfontosabb ismereteket [1]-ben megtalálhatjuk. Megértésük lényegesen könnyebbé válhat azáltal, ha vizuálisan is szemléltetjük ezeket. A jelen és a következő fejezet - pontosabban az itt készülő programok - hathatós segédeszközt biztositanak a fentiek megismeréséhez azáltal, hogy láthatóvá teszik a gépben végbemenő változásokat. Megszakitásból müködnek, igy akár parancs módban, akár egy másik program futása közben folyamatosan "kivetitik" a memória egy-egy részét a képernyőre.

Elsőként a technikailag egyszerübb megoldású grafikus vetitéssel foglalkozunk.

Mint ismeretes, a BASIC 3.5 verzió kényelmes lehetőséget biztosit az ún. osztott képernyő használatára, azaz (karakteres sorokban gondolkodva) a felső 20 sor kijelzése közben a grafikus, az alsó 5 sorban pedig a karakteres képernyőre kapcsol, mindezt természetesen a rasztermegszakitás segitségével teszi. Mi most ugyanezt fogjuk csinálni, de - a nagyobb felhasználható képterület érdekében - nem a 20., hanem a 15. sorban váltunk, a felső részben grafikus, az alsóban pedig karakteres kijelzést irunk elő, s a grafikus részen a tár $0000-tól kezdődő területét mutatjuk. Az előző fejezetben, a kép összezavarodásával kapcsolatos tapasztalatunkból okulva, a magnó, a hang és a billentyüzet kezelésére csak az alsó részre eső megszakitás-intervallumban adjunk lehetőséget!

Mint az ismert, a képernyő felső részében a grafikus területet a TED $FF06 regisztere segitségével kapcsolhatjuk ki-be, az itt megjelenő bittérkép kezdőcimét a $FF12 regiszterben, a hozzá tartozó szin- és fényerőmemória cimét pedig a $FF14

regiszterben jelölhetjük ki. Ez utóbbi regiszter írásával - karakteres módban - az összefüggő szín- és képmemória közös kezdetét szabhatjuk meg. Ezeknek az értékeknek a többsége értelemszerűen adódik, csak a grafikus színmemóriához szükséges üres terület helye kérdéses; tegyük ezt pld. $6000-től kezdődően! (Ez ugyan némi problémát okoz a $6666-os címen kezdődő programunk miatt, de a 15 karaktersornak megfelelő terület nem ér el addig.) A könnyebb áttekinthetőség kedvéért itt is (szerény, mindössze kételemű) táblázatokba foglaljuk az adatokat:

```
tff06  .byte $1b,$3b
tff0b  .byte $00,$79
tff12  .byte $c4,$00
tff14  .byte $08,$60
```

Az új megszakítás a következő:

```
zpmut=$e7              ;ciklusmutató
megsz   lda $ff09
        sta $ff09
        ldx zpmut
        lda tff06,x    ;bittérkép
        sta $ff06      ;ki-be
        lda tff0b,x    ;következő raszter-
        sta $ff0b      ;megsz. sora
        lda tff12,x    :bittérkép
        sta $ff12      ;helye
        lda tff14,x    :színmemória
        sta $ff14      ;helye
        inx
        cpx #$02
        bne ugrik
        jsr $cecd      ;hang
        jsr $cfbf      ;magnó
        jsr $db11      ;bill.
        ldx #$00       ;nullázás
ugrik   stx zpmut
        jmp $fcc3
```

Az új megszakitást aktivizáló résznek a megszakitásvektor átirásán kivül feladata még a szín- és fényerömemória fentebb emlitett törlése (ez a bekapcsolási teszt után $FF-$00 byte-okból áll), valamint egy ablak létrehozása a képernyő alsó felén (hogy a grafikus rész ne "takarja el", amit a képernyőre irunk).

A gépi kódú részt betöltő ill. bemutató BASIC-program több érdekességre fölhivja a figyelmet. Ezek megismétlésére nincs szükség, csupán arra emlékeztetünk, hogy a gép felépitésének megismerésénél kiváltképpen igaz: a mások által előirt tevékenység gyakorlásánál lényegesen hasznosabb az önálló felfedező munka.

Egyszerüsége folytán a kódolt rész megértése külön magyarázatot nem igényel:

```
mszv=$314             ;megsz. vektor
esct=$de5e            ;"esc t" funkció
plot=$fff0            ;kurzorbeállitás
       sei
       lda #<megsz
       sta mszv
       lda #>megsz
       sta mszv+1
       cli
       clc
       ldx #$0f       ;az ablak
       ldy #$00       ;bal felső
       jsr plot       ;sarkának
       jsr esct       ;beállitása
       ldx #$00
       lda #$f1
cik    sta $6000,x    szin- és
       sta $6100,x    fényesség-.
       sta $6158,x    memória
       sta $6400,x    inicializálása
       sta $6500,x
       sta $6558,x
       inx
       bne cik
       rts
```

Kérdések és feladatok a 10. fejezethez

10.1 Végezzük el a BASIC-programban felvetett feladatokat!

10.2 Nem jelent problémát, hogy a fenti CIK ciklusban egyes területek átfedik egymást?

10.3 Módositsuk a programot, hogy olyan osztása legyen a képnek, mint a GRAPHIC 2 esetén! VIGYÁZZUNK A MOST KÉSZÍTETT MEGSZAKÍTÓPROGRAMRA!

10.4 Most a $0000-tól kezdődő területet láttuk. Honnan kezdve tudjuk még a tárat kivetiteni?

10.5 A $8000 fölötti részek vetitésekor RAM-ot vagy ROM-ot látunk? Mitől függ ez? Alakitsuk át úgy a programot, hogy ezt is választani lehessen!

11. KVETÍTÉS

Az előző fejezetben ismertetett program használata közben fény derül a hiányosságokra is: túl nehezen lehet egy-egy byte változását megfigyelni, a bittérképnek megfelelő kijelzési mód segítségével nem követhetők pontosan az ábrázolt értékek, nem könnyű tájékozódni, a memóriának csak bizonyos részeit láthatjuk, mert a képernyő alsó felére eső területek megjelenítéséhez módosítanunk kell a megszakítórutint stb. Ezek részbeni kiküszöbölése ill. a tár egy más nézőpontból való vizsgálata érdekében, ugyancsak segédeszközként javasoljuk a következő programot.

Az előző programban alkalmazott "fegyver", a rasztermegszakítás - a kezdőcím megadása után - automatikusan, a változást vele "egy időben" jelezte a képernyőn, viszont - mivel a kezdőcímet csak 8 kbyte-onként lehet változtatni - igen merevnek bizonyult. Ennek feloldását oly módon oldhatjuk meg, hogy nem a rasztermegszakítás alkalmazásával, hanem egyszerűbben, a megszakítás kibővítésével vizsgáljuk a tárat, azaz: mielőtt a "rendes" megszakítási tevékenységet megkezdené a gép, kiíratjuk vele a képernyőre pld. két lap tartalmát. Ez ugyan viszonylag sok időt vesz igénybe, a kijelzés nem lesz olyan friss, mint az előző esetben, de a kijelzendő terület rugalmasan változtatható. Ha például a KEZD címtől kezdődő két lapot akarjuk kivetíteni, az a következő eljárással lehetséges (KÉP a karakteres képernyő kezdete, $0C00):

```
VETÍTÉS
   X:=0
   Ciklus, amíg X<>0 [azaz: X<256]
      KÉP,X:=KEZD,X              [egyik lap]
      KÉP+$100,X:=KEZD+$100,X    [másik lap]
      X:=X+1
   Ciklus vége
Eljárás vége
```

Ennek a résznek a megszakításba építésével tehát pld. a 2. és a 3. lap vetítésének kódja:

```
megsz                   ;új megszakitás
       ldx #$00
ciklus lda $0200,x
       sta $0c00,x
       lda $0300,x
       sta $0d00,x
       inx
       bne ciklus
       jmp $ce0e        ;megszakitás folytatása
```

Ha a $0200 és a $0300 címeket átirjuk, ugyanez a megszakitás az új címeknek megfelelő területet vetiti ki. Ennek érdekében jól bevált fogásunkhoz fordulunk: magával a programmal iratjuk át saját részét. Az egyszerüség kedvéért most csak lapkezdettől számitott kétlaponként végezzük a kijelölést, tehát kétszer egy byte-ot, a CIKLUS+2 és a CIKLUS+8 címen lévőt kell módositanunk, mégpedig a kétlapos blokk kezdőcíme felső byte-ja és az ennél 1-gyel nagyobb szám lesznek az új értékek. A BASIC-től is csak ennek a blokknak a kezdőcimét kérjük be, pld. decimális alakban (ez komoly megkötést nem jelent, mert a DEC függvényt használhatjuk, mint azt a bemutató programban is tesszük). Mivel most változtatható a kivetitett terület, szükség van kezdő- és végcímének folyamatos kijelzésére is. Hogy erre is lássunk példát, ezt a feladatot is assembly nyelven oldjuk meg.

A felhasznált rutinok nagyobbik része már ismert, a CİMKID ($A45F) rutin decimálisan, a CİMKIH ($FAFF) rutin pedig hexadecimális formában irja ki az X regiszterben és az akkumulátorban található kétbyte-os számot (vigyázzunk, mert egymáshoz képest forditott sorrendben dolgoznak, az első az X regiszter értékét alsó, a második pedig felső byte-nak tekinti!). A PRIMM ($FF4F) rutin igen hasznos funkciója az, hogy a hivási hely mögött lévő szöveget irja a képre, az első 0 byte-ig, majd az ez után következő utasitásra ugrik.

A program müködésének megértése a részletes megjegyzések alapján már szinte gyerekjáték:

```
mszv=$314               ;megsz. vektor
linnum=$14              ;címmutató
cimbe=$c38f             ;cím beolvasása
cimkid=$a45f            ;cím kiirása dec. (x/a)
cimkih=$faff            ;cím kiirása hex. (a/x)
primm=$ff4f             ;szöveg kiirása
cr=$fb3a                ;új sor
esct=$de5e              ;"esc t" funkció
plot=$fff0              ;kurzorbeállitás
       ;
       ;
```

```
        jsr cimbe       ;a kezdőcim
        lda linnum      ;beolvasása
        beq beir        ;cim jó
hiba    jsr primm       ;kiirja
        .byte 'nem lapkezdet!',0
        rts             ;vissza
beir    ldx linnum+1    ;cim hi byte
        stx ciklus+2    ;vetitéshez
        stx tol+1       ;kiiráshoz
        inx             ;(cim+256) hi byte
        stx ciklus+8    ;vetitéshez
        stx ig+1        ;kiiráshoz
        clc
        ldx #$0d        ;az ablak
        ldy #$00        ;bal felső
        jsr plot        ;sarkának
        jsr esct        ;beállitása
        jsr primm       ;kiirja
        .byte ' a vetités',0
        ldx tol         ;kezdőcim
        lda tol+1       ;kiirása
        jsr cimkid      ;decimálisan
        jsr primm       ;kiirja
        .byte '-',0
        ldx ig          ;végcim
        lda ig+1        ;kiirása
        jsr cimkid      ;decimálisan
        jsr primm       ;kiirja
        .byte ' = $',0
        lda tol         ;kezdőcim
        ldx tol+1       ;kiirása
        jsr cimkih      ;hexában
        jsr primm       ;kiirja
        .byte '- $',0
        lda ig          ;végcim
        ldx ig+1        ;kiirása
        jsr cimkih      ;hexában
        jsr cr          ;új sor
        jsr esct        ;ablak bf sarka
        sei
        lda #<megsz     ;új
        sta mszv        ;megszakitás
        lda #>megsz     ;beirása
        sta mszv+1
        cli
        rts
```

Itt is érvényesek az előző fejezet végén mondottak: minél többet kisérletezzünk, próbálkozzunk (persze ne értékes programmal a tárban, az ilyet előbb mentsük ki!), hiszen ez az igazi megismerés útja.

Kérdések és feladatok a 11. fejezethez

11.1 Végezzük el a BASIC-programban felvetett feladatokat!

11.2 Módositsuk úgy a programot, hogy ne csak lapkezdettől tudjon vetiteni!

11.3 A $8000 fölötti részek vetitésekor RAM-ot vagy ROM-ot látunk? Mitől függ ez? Alakitsuk át úgy a programot, hogy ezt is választani lehessen!

11.4 Módositsuk az assembly részt, ha a címvizsgálatról, a hibajelzésről, a címek kiirásáról és az ablak beállitásáról a BASIC-program gondoskodik!

11.5 Készitsük el a GVETÍTÉS programját úgy, hogy (a KVETÍTÉS mintájára) nem a rasztermegszakitást használjuk, hanem a megszakitórutin kibövitését!

12. FÉNYÚJSÁG

Utolsó feladatunk: fényújság készítése. Ez azt jelenti, hogy egy, általunk a memóriába vitt szöveget a képernyő egy előre kijelölt sorában úgy kell megjelenítenünk, mint ha a sorral, mint egy 40 karakter hosszúságú "ablakkal" balról jobbra végighaladnánk a szövegen, majd kezdenénk elölről az egészet s így tovább. Másként közelítve: először a szöveg első 40 karakterét, majd a 2. karaktertől kezdődő 40 karakterét, ezután a 3. karaktertől kezdődő 40 karakterét stb. írnánk a képernyő kijelölt sorába. A szöveget "végtelenítjük", azaz utolsó karaktere után mindig elölről ismételjük. Szemléletesen úgy is elképzelhetjük, hogy egy, a szöveg hosszával megegyező henger palástjára írjuk a szöveget, s ezt a hengert egy 40 karakter szélességü ablak előtt forgatjuk.
Az algoritmus egyszerü: a képen, a kérdéses sorban lévő 40 karakter közül a hátsó 39 mindegyikét egy hellyel balra léptetjük (vigyázzunk, a sorrend nem közömbös!), az ezáltal felszabaduló 40. helyre berakjuk a szöveg következő karakterét, s ezt addig folytatjuk, amig csak szükséges. Gondoljuk végig: ez a módszer arra az esetre is helyes eljárást ad, amikor a sorban még nem a szöveg karakterei vannak.
Algoritmusunk megírásához szükséges egy SZMUT mutató, amely a szöveg aktuális karakterére mutat. A képernyő kijelölt sora kezdetének címe KÉP; feltételezzük, hogy a szöveg a tárban van, elejére a SZÖVEG mutató mutat, hosszát pedig az SZH címen találjuk. Az algoritmus:

```
FÉNYÚJSÁG
   SZMUT:=-1 [szövegmutató az eleje elé]
   Ciklus, amíg szükséges
      Ciklus X:=0-tól 38-ig
         KÉP,X:=KÉP+1,X     [egy lépés balra]
      Ciklus vége
      Ha SZMUT=SZH, akkor SZMUT:=-1
      SZMUT:=SZMUT+1
      KÉP,39:=(SZÖVEG),SZMUT
   Ciklus vége
Eljárás vége
```

Ugyanannak a feladatnak a megvalósítása - az igényektől függően - többféle szinten történhet Erre a megoldás során többször fogunk utalni, sőt, három, egymástól különböző, fo-

kozatosan "fejlettebb" megoldást kódolva is bemutatunk. Láthatjuk majd, hogy lényegét tekintve mindegyik a fenti egyszerü algoritmuson alapul.

Az első, a legegyszerűbb változat: a szöveg a képernyő alsó sorában mozog, s - a SKÁLA programban alkalmazott megoldás alapján - a CONTROL és a C= gombok együttes megnyomására áll meg. A program gyors futása miatt szükségessé vált a lassitás a SZÁML két byte-os számláló segitségével. Ne feledkezzünk meg arról sem, hogy az ASCII és a képernyőkód eltérése miatt a képre irás előtt a karakter kódjának 6. bitjét alacsonyra kell állitanunk. A RAM-ROM kapcsolás szükségességét már - a TITKOSÍRÁSnál - kifejtettük. Ezek után a program az algoritmus szinte mechanikus kódolásával adódik:

```
kep=$0fc0              ;alsó sor kezdete
shflg=$543             ;c= és ctrl gombok
szh=$df                ;szöveg hossza
szoveg=$e0             ;szöveg helye
ram    = $ff3f         ;ram-ra kapcs.
rom    = $ff3e         ;rom-ra kapcs.
       ;
       ;
       lda #$ff        ;mutató
       sta szmut
ciklus                 ;mozgatás
       ldx #$00
eltol  lda kep+1,x     ;eltolás
       sta kep,x       ;balra
       inx
       cpx #$27        ;sor vége?
       bne eltol       ;nem, tovább
       ldy szmut
       cpy szh         ;szöveg vége?
       bne ugrke       ;nem, tovább
       ldy #$ff        ;igen, elölről
ugrke  iny             ;szövegmut.
       sty szmut       ;növelése
       sei
       sta ram
       lda (szoveg),y ;1 új karakter
       sta rom
       cli
       and #$bf        ;(képernyőkód)
       sta kep+39      ;beléptetése
lass   inc szaml       ;lassitás
       bne lass
       inc szaml+1
       bne lass
       lda shflg
       cmp #$06        ;c=+ctrl?
       bne ciklus      ;nem, tovább
       rts             ;igen, vége
```

A szövegparaméter beolvasása sys 26214,a$-ral, tehát pontosan ugyanúgy történik, mint a TITKOSÍRÁS című programban. Ami többlet: gondoskodni arról, hogy ne kerüljünk a kurzorral a fényújság területére, vagyis az utolsó sorba. Ezt - mint a PULZÁL, a GVETÍTÉS és a KVETÍTÉS programokban - egy ablak definiálásával oldjuk meg. A FÉNYÚJSÁG1 program kódja tehát:

```
chkoma=$9491          ;vesszőig előre
strflg=$0d            ;stringjelző
frestr=$9c48          ;szöveges változó beolvasása
escb=$de67            ;esc 'b' funkció
varptr=$47            ;szövegváltozó leírói
       ;
       ;
       jsr chkoma     ;vesszőig
       lda #$ff       ;szövegváltozó
       sta strflg     ;jön
       jsr frestr     ;szöveg be
       ldy #$02
szolv  lda (varptr),y ;leírók
       sta szh,y      ;másolása
       dey
       bpl szolv
       dec szh        ;1 többlet le
ablak  lda #$17       ;24. sor
       ldx #$27       ;40. oszlop
       jsr escb       ;jobb alsó sarok
       .
       .  [itt következik a fenti kód]
       .
```

Ez tehát az első szint, az eredeti feltételeknek eleget tévő, a beadott szöveget az alsó sorban mozgató fényújság. Elégedettek is lehetnénk vele, de nem biztos, hogy szerencsés a fényújság tempójának megválasztása: kinek lassabb, kinek gyorsabb felelne meg inkább. Elvárható tehát, hogy a sebességet változtatni lehessen, pld. gyorsítani a CONTROL, lassítani pedig a C= gombok lenyomásával. Erre egy viszonylag egyszerű megoldási lehetőség kínálkozik: a LASS ciklusba lépés előtt annak lefutási számát kell kisebbre ill. nagyobbra választanunk aszerint, hogy gyorsabb vagy lassúbb tempót igénylünk-e.

Észre sem vettük, már meg is fogalmaztuk a 2. szintet, sőt, megoldási utat is találtunk rögtön hozzá. Ahogy ezzel elkészülünk, eszünkbe jut egy alkalmazási lehetőség: milyen jól lehetne ezt a fényújságot menüként használni. A képernyő nagy részét nyugodtan tanulmányozhatjuk, mialatt a program az alsó - mindössze egy sornyi - részen, tehát kis helyen ciklikusan nagy mennyiségű információt közöl, folyamatosan

tájékoztatva ezzel minket választási lehetöségeinkről. Ehhez azonban az kell, hogy a fényújság mozgatása ne kösse le gépünk minden "erejét", jusson még a főprogram futtatására is - de hiszen ez már ismerős a SKÁLA, a PULZÁL stb. programokból: megszakitásból kell működtetni (3. szint)!

Közben más probléma is felvetődik: a szöveg nem folyamatosan, hanem karakterenként mozog, pedig mennyivel szebb lenne, ha simán, gördülékenyen kúszna odébb. Halványan emlékezünk arra, hogy a két TED-vezérlő regiszter segitségével vizszintes és függőleges finom scroll is megvalósitható. Igaz, ez csak a kép egészére vonatkozóan oldja meg, de igy is szépen csinálja; ez egy másik 3. szint. Ne feledkezzünk meg arról, hogy vizszintes finom scroll helyes működése feltételezi a $FF07 regiszter 3. bitjének magasra állitását, s ezzel a sorszélesség 38-ra csökkentését!

A rasztermegszakitás segitségével készitett programjaink - elsősorban a SOROK - példát mutattak arra, hogyan lehet elérni, hogy a kép egészére vonatkozó hatások csak egy sávban tudjanak érvényre jutni: rasztermegszakitást kell alkalmaznunk! Ez mindkét (előbb emlitett) harmadik szint továbbfejlesztése, máris megvan a 4. szint! Elvárásaink összefoglalva: az alsó sorban, változtatható sebességgel, folyamatosan gördül a szöveg, miközben másra is használhatjuk a gépet. Ennek megvalósitását mutatja be a FÉNYÚJSÁG2 program.

Már a SOROK, a GVETÍTÉS és a KVETÍTÉS cimü programoknál tapasztaltuk: ahhoz, hogy az egyes megszakitási sávokra előirandó regiszterértékeket helyes ütemben juttassuk érvényre, szükségünk van egy mutatóra: ez most az RMUT lesz. Gondolatmenetünk a következő: ciklikusan két, RMUT aktuális értékétől függő rasztermegszakitást kell kérnünk, az egyiket a fényújság sorának felső, a másikat az alsó rasztersorához érve. A fényújságon kivül 40 karakter széles képre és normál kijelzésre van szükség, mig a fényújságon belül 38 karakteres szélességre, és egy - a SCROLL eljárásban - változtatható mértékü balratolásra. Ezek mindketten függnek a $FF07 regiszterbe irt értéktől.

Eddigi példáinkhoz hasonlóan most is táblázatokba (TFF0B és TFF07) foglaljuk a TED-be irandó értékeket, s az egyes rasztermegszakitások bekövetkeztekor - RMUT mint index segitségével - innen olvassuk be a megfelelő regiszterbe őket. Ami

új, az az, hogy - míg eddig konstans táblázatokkal dolgoztunk, - most a $FF0B regiszterbe irandó első érték állandóan 8, a második viszont 0-tól 7-ig változhat, annak függvényében, hogy éppen mekkora sebességet választottunk. Sebességváltáskor tehát a táblázat megfelelő elemét át kell írnunk. A későbbiek megértése érdekében megjegyezzük, hogy a táblázatba írásnak még semmi, a képen látható jele nincs, mert az új sebesség csak akkor jut érvényre, amikor a fényújság sávjába lépéskor bekövetkező megszakitás irja a $FF07 regisztert (természetesen már az új értékkel).

Az algoritmus által megoldandó feladatok tehát: RMUT értékét 0 és 1 között váltogatni, RMUT mint index felhasználásával a táblázatokból irni a TED regisztereket, továbbá minden második körben (hogy miért nem minden körben, azt csak később értjük meg) adjunk módot a SCROLL eljárásban a táblázat egy elemének megváltoztatására (meg még valamire, de ezt is csak később fogjuk látni):

```
MEGSZAK
   X:=RMUT
   $FF0B:=TFF0B,X [köv. megszakitás rasztersora]
   $FF07:=TFF07,X [képszél. + finom mozgatás]
   X:=X+1
   Ha X=2, akkor SCROLL [mozgás param. beállitása + ?]
                X:=0 [kezdje elölről]
   Elágazás vége
   RMUT:=X
Megsz. folytatása
```

Kezdjünk hozzá a SCROLL eljárás elkészitéséhez! Először vegyük számba, hogy tulajdonképpen milyen adatok határozzák meg a TFF07 táblázat változó elemének az értékét! (Sokat fogunk erről a byte-ról beszélni, az egyszerüség kedvéért nevezzük el SCROLLMUT-nak!)

A sebességgel kapcsolatban: fényújságunknak van egy "természetes" tempója. Ezen azt értjük, hogy a már müködő, de a szabályozásra még fel nem készitett program milyen szaporán mozgatja a szöveget. Bár nem túl bonyolult, de mégis nehézkes volna olyan szabályozást megvalósitani, amely egyaránt alkalmassá tenné a programot ennél gyorsabb ill. lassúbb sebességgel való mozgatásra, hiszen egyik esetben a lépések megnyújtásáról, a másiknál pedig késleltető ciklusok beiktatásáról van szó. Próbáljuk ki, hogy a vizszintes finom scroll alkalmazásával (pld. az előző szinten emlitett programban) milyen sebességgel mozog balra a kép! Láthatjuk,

hogy elég komótosan mozog, igy - szemben a FÉNYÚJSÁG1-ben tapasztaltakkal - itt éppen nem lassitásra, hanem inkább gyorsitásra van szükség. Ezt úgy gondoljuk megvalósitani, hogy a sebességet tartalmazó SEB byte értékének megfelelő számú raszterponttal toljuk balra a szöveget minden olyan alkalommal, amikor a program a SCROLL eljárást végrehajtja. SEB értékét a TEMPÓ eljárásban a CONTROL gombbal növelni, a C= gombbal pedig csökkenteni tudjuk. Értékének ésszerű határok között tartásáról természetesen nekünk kell gondoskodnunk; a tapasztalat azt mutatja, hogy az 1...15 (zárt) intervallum már nagyon széles sebességtartományt biztosit. Kezdeti értékét 3-ra állitjuk, s mivel ez a kijelölt tartományban van, csak arra kell ügyelnünk, hogy növelésekor felfelé, csökkentésekor pedig lefelé ne engedjük innen kijutni.

El kell érnünk, hogy a SCROLL eljárás úgy módositsa a SCROLLMUT értékét, hogy a szöveg SEB számú raszterpontnyival balra kerüljön. Emlékeztetünk arra, hogy $FF07 alsó 3 bitjének értéke tkp. a jobbratolódás mértéket mutatja, az 1 raszterpontnyival balra mozgatás - általában - úgy valósitható meg, hogy a byte értékét 1-gyel csökkentjük. Kisebb probléma van akkor, ha a byte értéke éppen 0, mert ebben az esetben a karaktert át kell léptetnünk a tőle balra lévő pozicióba, majd a $FF07 byte-ot 7-re kell állitanunk (formálisan: mivel egy karakterpozició 8 raszterpont széles, a 8-cal balra és egyúttal 7-tel jobbra léptetés végeredményben 1 raszterpont balra). Nagyon egyszerű tehát a megoldás: egy SEB-szer lefutó ciklus magjába teszünk egy olyan lépést, amely SCROLLMUT értékét minden alkalommal 1-gyel csökkenti. Mást nem is kell tennünk azokban az esetekben, amikor SCROLLMUT értéke e változtatás következtében sem lesz negativ. Az előzőek alapján viszont már arra az esetre is tudjuk a "receptet", ha mégis negativ lesz: SCROLLMUT feltöltése 7-tel + a szöveg balra tolása 1 karakterrel + 1 új karakter beléptetése. Nem szükségszerű ekkor csinálnunk, de adjunk módot a sebesség megváltoztatására a TEMPÓ eljárás meghivásával. Most már világos, miért mondtuk fentebb, hogy a SCROLL még valami mást is csinál a finom mozgás paraméterének beállitásán kivül: elvégzi a durva mozgást. Nézzük az algoritmust:

```
SCROLL [durva m.+finom m. param. beállitás]
   SMUT:=SEB [akt. sebesség betöltése]
   Ciklus, amig SMUT>0
      SCROLLMUT:=SCROLLMUT-1
      Ha SCROLLMUT=-1,
         akkor SCROLLMUT:=7
               ELTOLÁS [a kép 1 karakterrel balra]
               BELÉP [1 új karakter a szövegből]
               TEMPÓ [a sebesség szabályozása]
      Elágazás vége
      SMUT:=SMUT-1
   Ciklus vége
ELTOLÁS [a szöveg 1 karakterrel balra]
   X:=0
   Ciklus, amig X<38 [sorhossz]
      KÉP,X:=KÉP+1,X
   Ciklus vége
Eljárás vége
BELÉP [1 új karakter a szövegből]
   Y:=SZMUT
   Ha Y=SZH, akkor Y:=-1
   Y:=Y+1
   SZMUT:=Y
   KÉP,38:=(SZÖVEG),Y AND #$BF [a képernyőkód miatt]
Eljárás vége
TEMPÓ [a sebesség szabályozása]
   Ha CONTROL és SEB<$0F,
      akkor SEB:=SEB+1
      különben Ha C= és SEB>1,
                  akkor SEB:=SEB-1
   Elágazások vége
Eljárás vége
```

Mivel az egyes eljárásokat már értjük, a kódolásnál az egyes részeket nem minden esetben választottuk szét, de így is könnyen felismerhetők:

```
megsz                ;új megszakitás
       lda $ff09
       sta $ff09
       ldx rmut
       lda tff0b,x   ; következő raszter-
       sta $ff0b     ;megsz. helye
       lda tff07,x   ;vizszintes scroll-
       sta $ff07     ;mutató
       inx
       cpx #$02
       bne ugr01     ;nem, tovább
       jsr hang      ;igen
       jsr scroll    ;vizsz. scroll
       jsr magno     ;magnó kezelése
       jsr bill      ;bill. kezelése
       ldx #$00      ;nullázás
ugr01  stx rmut
       jmp $fcc3     ;megsz. tovább
       ;
       ;
```

```
scroll                  ;durva mozg. + finom mozg. beáll.
        lda seb         ;sebesség
        sta smut        ;beáll.
megy    ldx tff07+1     ;finomscroll mértéke
        dex             ;balra
        stx tff07+1
        cpx #$ff        ;poz. vége?
        bne vissza      ;nem, vissza
        lda #$07        ;igen
        sta tff07+1     ;újratöltés
        ldx #$00
eltol   lda kep+1,x     ;eltolás
        sta kep,x       ;balra
        inx
        cpx #$26        ;sor vége?
        bne eltol       ;nem, tovább
        ldy szmut
        cpy szh         ;szöveg vége?
        bne ugrke       ;nem, tovább
        ldy #$ff        ;igen, elölről
ugrke   iny             ;szövegmut.
        sty szmut       ;növelése
        sta ram
        lda (szoveg),y  ;1 új karakter
        and #$bf        ;(képernyőkód)
        sta rom
        sta kep+38      ;beléptetése
        jsr tempo       ;sebesség szabályozása
vissza  dec smut        ;sebességmutató
        bne megy        nem 0, tovább
        rts
        ;
        ;
tempo                   ;a sebesség szabályozása
        lda shflg
        cmp #$04        ;ctrl?
        beq nov         ;igen, ->növ.
        cmp #$02        ;comm.?
        bne kesz        ;nem, vissza
        lda seb         ;sebesség
        cmp #$01        ;=1?
        beq kesz        ;igen, vissza
        dec seb         ;csökkentés
        bpl kesz        ;vissza
nov     lda seb         ;sebesség
        cmp #$0f        ;=maximum?
        beq kesz        ;igen,vissza
        inc seb         ;növelés
kesz    rts
```

Még annyit jegyzünk meg, hogy a FÉNYÚJSÁG1 programnak a szöveget beolvasó, az ablakot beállító és a megszakításvektort átíró része változtatás nélkül alkalmas most is ezeknek a feladatoknak az ellátására.

Mostani programunk már kényünk-kedvünk szerint működik, minden eddigi kívánságunknak eleget tesz. Még jobb lenne azonban, ha a fényújság úgy futna, hogy a képet még egyetlen

sorral sem kurtitaná meg, azaz képernyönknek mind a 25 sorát saját céljainkra használhatnánk! Hogyan lehetne ezt (az 5. szintet jelentő) feladatot megoldani?

Az biztos, hogy a 25-nél több különböző sor használata egy újabb memóriaterületet igényel, mert a képernyötár 1024 byte-ja ugyan tartalmaz fölösleges címeket ($0FE8-0FFF), ezek azonban együttesen egy fél sort sem tesznek ki. Ebből következik, hogy fényújságunkat egy másik képernyön kell futtatnunk. Mint a KÉTKÉPERNYÖböl is tudjuk, az aktuális karakteres képernyö címét a TED $FF14 regisztere tartalmazza. Feladatunk tehát világos: miután az eredeti képernyö frissitése során elhagytuk a 25 karakteres sort és kiértünk a keretre, rasztermegszakitást kérünk, áttérünk a másik kép kijelzésére, majd vissza. No, ezt igy nagyon könnyü elmondani, de még messze vagyunk a megvalósitástól!

Ha az előző szinthez hasonlóan, egyszerüen elvégeznénk a táblázat módositását, hogy a kereten kérjünk megszakitást, az égvilágon semmi változást nem tapasztalnánk, mert csak annyi történne, hogy az eredeti képernyö keretének egy sávja helyett a másik képernyö keretének megfelelő sávját látnánk. El kell érnünk azt, hogy amikor a másik képernyöre érünk, úgy "érezze" a TED (mert sok minden más mellett a kép megjelenitését is ő végzi), mint ha a képernyö "papir" részének kiirása következne, magyarul be kell csapnunk a TED-et. Ebben a $FF1C-$FF1D címeken lévő videosor-számláló lesz segitségünkre: ez mindig annak a rasztersornak a sorszámát tartalmazza (a legfelső bitet $FF1C-n), amelynek a kijelzése éppen most van folyamatban. Ha ezt az értéket átirjuk, a TED "úgy érzi", hogy a képnek azon a részén jár, ahová a számláló mutat. Ezt próbáljuk kihasználni a feladat megoldásához.

Helyezzük el új képernyönk szinmemóriáját a $6000 címtől kezdve, ennek következtében a képmemória $6400 és $67e7 között helyezkedik el. Képernyönknek csak a felső sorát veszszük igénybe a fényújsághoz, nem baj tehát, hogy a most készitendő programunkkal ($6666->) átfedik egymást. A megszakitásokat a következőképpen képzeljük el (tudván, hogy a "papir" aljának rasztersorszáma $CC, s mi ez alatt szeretnénk a fényújságot megjeleniteni; az eredeti képernyöt "régi"-nek, a fényújságos képernyöt "új"-nak nevezzük):

Amikor RMUT

értéke	0	1	, akkor
a megszakitás	a régi	az új	képernyő
	$D0	$18	sorában generálódik.
Kijelezzük	az új	a régi	képernyőt, a raszter-
számlálót	$00	$E8	értékre állitjuk,
majd	$18-nál	$D8-nál	kérjük a következő

megszakitást. Ez úgy is fogalmazható, hogy a régi képernyő $D0...$E8 közötti három sorát az új képernyő $00...$18 közötti három sorával cseréljük fel.

A program inicializáló része annyiban tér el az előzöktől, hogy nem kell ablakot csinálni, ki kell viszont takarítani a tárnak az új képernyő fölső 3 sorába eső részét, valamint gondoskodni kell a TED eddig nem szerepelt regisztereinek az irásáról is:

```
kep=$6400              ;szöveg helye
szm=$6000              ;szinmemória

        .  [ld. az előző programot!]

        ldx #$78
tcik    lda #$20       ;kép- és
        sta kep,x      ;szinmemória
        lda #$00       ;törlése
        sta szm,x
        dex
        bpl tcik
        rts
megsz                  ;új megszakitás
        lda $ff09
        sta $ff09
        ldx rmut
        lda tff1c,x    ;rasztersor
        sta $ff1c      ;számláló
        lda tff1d,x    ;felső-alsó
        sta $ff1d      ;része
        lda tff0b,x    ;köv. raszter-
        sta $ff0b      ;megsz. helye
        lda tff07,x    ;vizsz. scroll-
        sta $ff07      ;mutató
        lda tff14      ;képernyő
        sta $ff14      ;kezdete
        inx
        cpx #$02
        bne ugr01      ;nem, tovább
        jsr hang       ;igen, hangkezelés
        jsr scroll     ;vizsz. scroll
        jsr magno      ;magnó kezelése
        jsr bill       ;bill. kezelése
        ldx #$00       ;nullázás
ugr01   stx rmut
        jmp $fcc3      ;megsz. tovább
```

```
        ;
        ;
tff07   .byte $08,$01
tff0b   .byte $D0,$18
tff14   .byte $08,$60
tff1c   .byte $00,$00
tff1d   .byte $E8,$00
```

A program futtatása után rájövünk, hogy amit olyan szépen elképzeltünk, semmit sem ér a gyakorlatban: látjuk ugyan a két részt, a képernyő széléről azt is le tudjuk olvasni, hogy a megszakítás úgy történik, ahogyan programoztuk, mert a kereten lévő 3 sor ténylegesen 38 karakteres, ellentétben a többivel, de mind ezen a részen, mind pedig a három sor után teljesen "mákos" a képernyő. Ennek oka valószínüleg valami időzitési probléma, pontosabban mi sem tudjuk. Ha viszont "kisérletezgetni" kezdünk az egyes értékekkel, változást tapasztalunk, pld. a következő táblázatnál:

```
tff07   .byte $08,$01
tff0b   .byte $C0,$08
tff14   .byte $08,$60
tff1c   .byte $00,$01
tff1d   .byte $E8,$36 ,
```

s ennek számos módositásánál egy zavaros képernyőt látunk, de az ehhez "közeli" táblázatok mindegyikénél többé-kevésbé fellelhetők a mozgó szöveg részei!

Próbáljunk meg a gépnek kedvében járni: iktassunk be a ciklusba egy harmadik megszakitást is, amelynek segitségével esetleg "helyrerázódik" a rasztermegszakitás! Ehhez csak egy utasitást (cpx #$02) és a táblázatot kell módositanunk. A kép máris megnyugszik, mindenütt a sor elején kezdődik, csupán egy, a szemet bántó "remegés" zavarja a látványt, de apró módositások után, a következő értékeknél gyönyörüen kúszik a kép alatt a szöveg (, amely a $6428 cimen, tehát az új képernyő második sorában van):

```
tff07   .byte $08,$01,$08
tff0b   .byte $03,$EE,$cd
tff14   .byte $60,$60,$08
tff1c   .byte $01,$00,$00
tff1d   .byte $36,$bd,$fa
```

Bárhogyan is kisérleteztünk, az első sort nem sikerült teljes egészében a képre vinni. Úgy tünik, ezekhez a dolgokhoz mélyebb hardver-ismeretekre lenne szükség. A program következő sorait módositanunk kell (ezek már a módositott értékek):

```
kep=$6428               ;kép
kepm=$6400              ;képmemória
szm=$6000               ;szinmemória
          .
          .
          .
        ldx #$50
tcik    lda #$20        ;kép- és
        sta kepm        ;szinmemória
        lda #$00        ;törlése
        sta szm
        dex
        bpl tcik
          .
          .
          .
        cpx #$03
          ...
```

Nagy keservesen a végére értünk tehát, példát mutatva arra, hogy a tárgyi tudás, a jó ötletek, a sok tapasztalat és a türelmes, kitartó munka mellett szükség van egy jó adag intuicióra és szerencsére is.

Kérdések és feladatok a 12. fejezethez

12.1 Készitsünk jobbra mozgó fényújságokat (pld. héber feliratokhoz)!

12.2 Valósitsuk meg azokat a fényújság-szinteket, amikről szó volt!

12.3 Írjuk át a sebességet tágabb határok között változtathatóvá!

12.4 Oldjuk meg FÉNYÚJSÁG2-t a képernyő fölső sorában!

12.5 Próbálkozzunk FÉNYÚJSÁG3-nak a képernyő fölötti megoldásával!

12.6 Készitsünk olyan fényújságot, amely a képernyő bal szélső oszlopában felfelé mozgat egy szöveget, FÉNYÚJSÁG2 technikájával!

12.7 Mi változik meg, ha a MEGSZAK algoritmust úgy írjuk át, hogy minden megszakitás alkalmával lehetőség nyíljon a SCROLL-lal való módositásra?

BEFEJEZÉS

Remélhetőleg sikerült a C+4 gépi kódú programozásába bepillantást nyújtani, bár a gép lehetőségeit korántsem meritettük ki: még számtalan érdekes, látványos-hangzatos hatást lehet elérni a megszakitás segitségével, a TED funkcióinak kihasználásával. Még csak nem is érintettük (az [I]-ben szereplő témákon kivül) a felhasználói kapu (user port) alkalmazási lehetőségeit, a beépitett szoftvernek a lapozási technika segitségével való tanulmányozását, a programvédelmi módszereket, a tömbkezelő algoritmusok (pld. rendezések, keresések) témakörét stb. Nagyon érdekesnek igérkezik az ún. nem szabványos gépi kódú utasitások tanulmányozása, ehhez kiváló segédeszköz az általunk is használt JCL assembler. Ennek kulcsszótáblájában ugyan nem szerepelnek ezek az utasitások, de a helyüket kihagyták, ezzel biztositva a bővités lehetőségét. Nem foglalkoztunk a BASIC-programozást segitő programok körével sem: ilyen pld. egy olyan gépi kódú program készitése, amely bármely gépi kódú program byte-jait DATA-sorokba fűzve ir egy BASIC-programot, vagy alkalmas a program szövegében szereplő karaktersorozatok keresésére, cseréjére; a lista oldalankénti, "el nem futó" kiirására, változók értékeinek megőrzésére a program javitásakor; az input puffer áthelyezésével 88 karakteresnél hosszabb sorok bevitelére; a MERGE funkció megvalósitására; a program sűritésére; rossz hivatkozások megkeresésére; névvel azonositott eljárások definiálására; a program soronkénti végrehajtására stb. Egyet ne felejtsünk el: csak úgy válhat valaki gépe és a gépi kódú programozás jó ismerőjévé, ha sokat foglalkozik vele, s önálló felfedező utakat tesz. Igaz ugyan Horner kétbalkéz-szabálya, amely szerint: "A tapasztalat egyenes arányban nő a tönkretett berendezések számával" (Murphy), mégse kiméljük gépünket: meg fogja hálálni.

F Ü G G E L É K

BASIC-programok
és assembly
fordítási listák

- - - - - - - - - - -

```
10 ifww=1thenprint"q            szevasz! jo munkat!"::end
20 print"ss3qq a program ket titkosirast mutat be."
30 print"qq az elso az a$ szoveges valtozot tit-    kositja
egy tablazat ertekei alapjan."
40 print" hivasa:"
50 print"       kodolaskor:   sys26214,1,a$
dekodolaskor: sys26214,2,a$."
60 print"qq a program az a$ szoveges valtozot tit-  kositja a
b$ szoveges valtozo erteke"
70 print" alapjan, a megfelelo karakterek kozot-  ti kizaro
vagy muvelet segitsegevel."
80 print" hivasa:"
90 print"       kodolaskor:   sys26214,0,a$,b$"
100 print" a visszafejtes (a kizaro vagy muvelet
tulajdonsaga miatt) ugyanilyen modon"
110 print" tortenik."
120 ww=1:load"titkos.mod",8,1
```

```
.....page # 1

Line#  Addr Code        Source

00001  0000            ;put"@0:titkos"
00002  0000            ;*********************************************
00003  0000            ;*                                           *
00004  0000            ;* az a$ valtozo tartalmanak titkositasa     *
00005  0000            ;*    tablazat alapjan.                      *
00006  0000            ;*    hivasa: sys26214,1,a$                  *
00007  0000            ;*    visszafejtes: sys26214,2,a$            *
00008  0000            ;*                                           *
00009  0000            ;* az a$ valtozo tartalmanak titkositasa     *
00010  0000            ;*    a b$ kod tartalma szerint.             *
00011  0000            ;*    hivasa: sys26214,0,a$,b$               *
00012  0000            ;*    visszafejtes: ugyanigy                 *
00013  0000            ;*                                           *
00014  0000            ;*********************************************
00015  0000                    * = $6666
00016  6666            ram     = $ff3f          ;ram-ra kapcsol
00017  6666            rom     = $ff3e          ;rom-ra kapcsol
00018  6666            varptr  = $47            ;valtozo mutatoja
00019  6666            chkoma  = $9491          ;vesszoig elore
00020  6666            strflg  = $0d            ;stringjelzo
00021  6666            frestr  = $9c48          ;szoveges valt. beol
00022  6666            getbyt  = $9d84          ;1 byte x-be
00023  6666            szh     = $df            ;szoveghossz
00024  6666            szmut   = $e0            ;szovegmutato
00025  6666            kh      = $e2            ;kodhossz
00026  6666            kmut    = $e3            ;kodmutato
00027  6666            ;
00028  6666            ;
00029  6666 20 91 94           jsr chkoma       ;vesszoig elore
00030  6669 20 84 9d           jsr getbyt       ;1 byte x-be olvasas
00031  666c e0 01              cpx #$01         ;tablazattal oda?
00032  666e f0 4d              beq toda         ;igen
00033  6670 e0 02              cpx #$02         ;tablazattal vissza?
00034  6672 f0 6e              beq tvisz        ;igen
00035  6674            koddal                   ;egyebkent koddal
00036  6674 20 0e 67           jsr szovbe       ;szoveg beolvasasa
00037  6677 20 91 94           jsr chkoma       ;vesszoig
00038  667a a9 ff              lda #$ff         ;string jon
00039  667c 85 0d              sta strflg
00040  667e 20 48 9c           jsr frestr       ;kod-
00041  6681 a0 02              ldy #$02         ;parameter
00042  6683 b1 47      kolv    lda (varptr),y   ;beolv.
00043  6685 99 e2 00           sta kh,y         ;mutatok
00044  6688 88                 dey              ;masolasa
00045  6689 10 f8              bpl kolv
00046  668b a2 00              ldx #$00         ;kodmutato
00047  668d a4 df              ldy szh          ;szovegmutato
00048  668f 38                 dey
00049  6690 78                 sei
00050  6691 8d 3f ff           sta ram          ;ram-ra kapcsol
00051  6694 b1 e0      cikl    lda (szmut),y    ;szoveg kov.
00052  6696 8d 23 67           sta szbyte       ;byte-ja
00053  6699 98                 tya              ;szovegmutato
00054  669a 48                 pha              ;a verembe
00055  669b 8a                 txa              ;kodmutato
00056  669c a8                 tay              ;y-ba
```

```
line#  Addr Code        Source

00057  669d b1 e3             lda (kmut),y      ;kod kov. byte-ja
00058  669f e8                inx               ;kodmut. noveles
00059  66a0 e4 e2             cpx kh            ;vege?
00060  66a2 d0 02             bne xugr          ;nem, mehet
00061  66a4 a2 00             ldx #$00          ;igen, elolrol
00062  66a6 4d 23 67  xugr     eor szbyte        ;kodolas
00063  66a9 49 40             eor #$40          ;kompenzalas
00064  66ab 8d 23 67          sta szbyte        ;tarolas
00065  66ae 68                pla               ;szovegmutato
00066  66af a8                tay               ;a verembol
00067  66b0 ad 23 67          lda szbyte        ;vissza a
00068  66b3 91 e0             sta (szmut),y     ;helyere
00069  66b5 88                dey               ;szoveg vege?
00070  66b6 10 dc             bpl cikl          ;nem, vissza
00071  66b8 8d 3e ff          sta rom           ;rom-ra kapcsol
00072  66bb 58                cli
00073  66bc 60                rts
00074  66bd             ;
00075  66bd             ;
00076  66bd             toda                    ;tablazattal oda
00077  66bd 20 0e 67           jsr szovbe        ;szoveg beolvasasa
00078  66c0 a4 df              ldy szh           ;szovegmutato
00079  66c2 88                 dey
00080  66c3 78                 sei
00081  66c4 8d 3f ff           sta ram           ;ram-ra kapcs
00082  66c7 b1 e0       ocikl  lda (szmut),y     ;szoveg kov. kar.
00083  66c9 c9 41              cmp #$41          ;<a?
00084  66cb 30 0b              bmi okesz         ;igen, marad
00085  66cd c9 5b              cmp #$5b          ;>z?
00086  66cf 10 07              bpl okesz         ;igen, marad
00087  66d1 38                 sec
00088  66d2 e9 41              sbc #$41          ;hanyadik betu?
00089  66d4 aa                 tax               ;indexbe
00090  66d5 bd 24 67           lda tabla,x
00091  66d8 91 e0       okesz  sta (szmut),y     ;helyere
00092  66da 88                 dey               ;szoveg vege?
00093  66db 10 ea              bpl ocikl         ;nem, vissza
00094  66dd 8d 3e ff           sta rom           ;rom-ra kapcsol
00095  66e0 58                 cli
00096  66e1 60                 rts
00097  66e2             ;
00098  66e2             ;
00099  66e2             tvisz                   ;tablazattal vissza
00100  66e2 20 0e 67           jsr szovbe        ;szoveg beolvasasa
00101  66e5 a4 df              ldy szh           ;szovegmutato
00102  66e7 88                 dey
00103  66e8 78                 sei
00104  66e9 8d 3f ff           sta ram           ;ram-ra kapcs
00105  66ec b1 e0       vcikl  lda (szmut),y     ;szoveg kov.
00106  66ee c9 41              cmp #$41          ;<a?
00107  66f0 30 12              bmi vkesz         ;igen, marad
00108  66f2 c9 5b              cmp #$5b          ;>z?
00109  66f4 10 0e              bpl vkesz         ;igen, marad
00110  66f6 a2 19              ldx #$19          ;betuk szama
00111  66f8 dd 24 67  xcik     cmp tabla,x       ;az x-edik?
00112  66fb f0 03              beq megvan        ;igen, ->megvan
```

```
Line#  Addr Code        Source

00113  66fd ca                dex                     ;nincs,
00114  66fe 10 f8              bpl xcik                ;tovabb
00115  6700 8a          megvan txa                     ;index at
00116  6701 18                clc
00117  6702 69 41              adc #$41                ;ennyiedik betu
00118  6704 91 e0        vkesz  sta (szmut),y          ;helyere
00119  6706 88                dey                     ;szoveg vege?
00120  6707 10 e3              bpl vcikl               ;nem, vissza
00121  6709 8d 3e ff           sta rom                 ;rom-ra kapcs
00122  670c 58                cli
00123  670d 60                rts
00124  670e             ;
00125  670e             ;
00126  670e             ;
00127  670e             szovbe                         ;szoveg beolvasasa
00128  670e 20 91 94            jsr chkoma              ;vesszoig
00129  6711 a9 ff               lda #$ff                ;string jon
00130  6713 85 0d               sta strflg
00131  6715 20 48 9c            jsr frestr              ;szoveg-
00132  6718 a0 02               ldy #$02                ;parameter
00133  671a b1 47        szolv  lda (varptr),y          ;beolv.
00134  671c 99 df 00            sta szh,y               ;mutatok
00135  671f 88                  dey                     ;masolasa
00136  6720 10 f8               bpl szolv
00137  6722 60                  rts
00138  6723             szbyte * = *+1
            51 57 45 52 54 59 55 49 4f 50 41 53 44 46 47 48 4a 4b 4c 5a 58
            43 56 42 4e 4d
00139  6724             tabla .byte  qwertyuiopasdf;tablazatbnm'

end of assembly, error count = 00000

chkoma  9491    cikl    6694    frestr  9c48    getbyt  9d84
kh      00e2    kmut    00e3    koddal  6674    kolv    6683
megvan  6700    ocikl   66c7    okesz   66d8    ram     ff3f
rom     ff3e    strflg  000d    szbyte  6723    szh     00df
szmut   00e0    szolv   671a    szovbe  670e    tabla   6724
toda    66bd    tvisz   66e2    varptr  0047    vcikl   66ec
vkesz   6704    xcik    66f8    xugr    66a6
```

```
10 ifww=1thengetkeyx$:new
20 poke1281,102:poke1282,102
30 print"Sq           funkciok es hivasuk:"
40 print"qJp=usr(0),c1,c2,c3    ram-bol masolja a"
50 print"                       c1-c2 reszt
60 print"                       c3-tol kezdve
70 print"qJp=usr(1),c1,c2,c3    rom-bol masolja a"
80 print"                       c1-c2 reszt
90 print"                       c3-tol kezdve
100 print"qJp=usr(2),c1,c2,c3     a c1-c2 elejerol
110 print"                       a c3-ig terjedo
120 print"                       reszt torli
130 print"qJp=usr(3),c1,c2,c3,c4 ram-bol beszurja a"
140 print"                       c1-c2 elejere
150 print"                       a c3-c4 reszt
160 print"qJp=usr(4),c1,c2,c3,c4 rOm-bol beszurja a"
170 print"                       c1-c2 elejere
180 print"                       a c3-c4 reszt
190 print"q      c1,c2,c3,c4 decimalis cimek,          a
hatarok mindenutt beleertendok!";
200 ww=1:load"monitor.mod",8,1
```

```
Line#  Addr Code        Source

00001  0000             ;put"@0:monitor"
00002  0000             ;*************************************************
00003  0000             ;*                                               *
00004  0000             ;* a program nehany, a tedmon-ban nem szereplo   *
00005  0000             ;* monitor-funkciot valosit meg.                 *
00006  0000             ;*                                               *
00007  0000             ;* hasznalata elott a $0501-$502 cimen levo      *
00008  0000             ;* usr-vektort at kell irni $6666-ra!            *
00009  0000             ;*                                               *
00010  0000             ;* hasznalata:                                   *
00011  0000             ;*                                               *
00012  0000             ;* p=usr(0),c1,c2,c3      ram-bol masolja a      *
00013  0000             ;*                        c1-c2 reszt            *
00014  0000             ;*                        c3-tol kezdve          *
00015  0000             ;* p=usr(1),c1,c2,c3      rom-bol masolja a      *
00016  0000             ;*                        c1-c2 reszt            *
00017  0000             ;*                        c3-tol kezdve          *
00018  0000             ;* p=usr(2),c1,c2,c3      a c1-c2 resz elejet    *
00019  0000             ;*                        c3-ig torli            *
00020  0000             ;* p=usr(3),c1,c2,c3,c4   ram-bol beszurja       *
00021  0000             ;*                        a c1-c2 elejere        *
00022  0000             ;*                        a c3-c4 reszt          *
00023  0000             ;* p=usr(4),c1,c2,c3,c4   rom-bol beszurja       *
00024  0000             ;*                        a c1-c2 elejere        *
00025  0000             ;*                        a c3-c4 reszt          *
00026  0000             ;*                                               *
00027  0000             ;* c1,c2,c3,c4 decimalis cimek,                  *
00028  0000             ;* a hatarok mindenutt beleertendok!             *
00029  0000             ;*                                               *
00030  0000             ;*************************************************
00031  0000                     * = $6666
00032  6666             linnum   = $14              ;cim tarolo
00033  6666             cimbe    = $c38f            ;ketbyte-os ertek beolv
00034  6666             facint   = $a327            ;fac egessze konv.
00035  6666             illq     = $991c            ;hibajelzes
00036  6666             param    = $65              ;usr parametere
00037  6666             fe       = $e0              ;forras eleje
00038  6666             fv       = $e2              ;forras vege
00039  6666             cel      = $e4              ;cel
00040  6666             db       = $e6              ;hany byte?
00041  6666             ;
00042  6666             ;
00043  6666 20 27 a3           jsr facint          ;par.->egesz
00044  6669 a6 65              ldx param           ;mod
00045  666b 30 04              bmi hiba            ;<0, hiba
00046  666d e0 05              cpx #$05            ;>=5?
00047  666f 30 03              bmi mehet           ;igen, hiba
00048  6671 4c 1c 99     hiba  jmp illq            ;ki:'ill. q.'
00049  6674 8e 03 68     mehet stx mod             ;elrakjuk
00050  6677 0e 03 68           asl mod             ;cimhez 2*
00051  667a e0 03              cpx #$03            ;hany cim kell?
00052  667c 10 03              bpl negy            ;beszuras,->4
00053  667e a9 06              lda #$06            ;masolas-torles,->3
       667f 2c
00054  6680                   .byte $2c           ;atlatszo bit
00055  6681 a9 08        negy  lda #$08
```

```
Line#  Addr Code         Source

00056  6683 8d 04 68           sta hatar          ;kell beolvasni?
00057  6686 a2 00              ldx #$00
00058  6688 8a           beolvc txa               ;index
00059  6689 48                 pha                ;verembe
00060  668a 20 8f c3           jsr cimbe          ;cim beolv.
00061  668d 68                 pla                ;index
00062  668e aa                 tax                ;verembol
00063  668f a5 14              lda linnum
00064  6691 9d fb 67           sta ciml,x         ;#<cim
00065  6694 e8                 inx
00066  6695 a5 15              lda linnum+1
00067  6697 9d fb 67           sta ciml,x         ;#>cim
00068  669a e8                 inx
00069  669b ec 04 68           cpx hatar          ;vege?
00070  669e d0 e8              bne beolvc         ;nem, vissza
00071  66a0 ae 03 68           ldx mod
00072  66a3 bd f1 67           lda ugrtab,x       ;ugrasi
00073  66a6 8d b0 66           sta ugrik+1        ;cim
00074  66a9 bd f2 67           lda ugrtab+1,x     ;toltese
00075  66ac 8d b1 66           sta ugrik+2
00076  66af 4c 00 00    ugrik  jmp $0000
00077  66b2                    ;
00078  66b2                    ;
00079  66b2                    amas                ;ram-bol masol
00080  66b2 20 e8 67                jsr ram       ;rom tiltas
00081  66b5 a2 05              mas   ldx #$05
00082  66b7 bd fb 67           amasc  lda ciml,x   ;a 3 cimet
00083  66ba 95 e0                     sta fe,x     ;eredeti
00084  66bc ca                        dex          ;sorrendben
00085  66bd 10 f8                     bpl amasc    ;atmasolja
00086  66bf 4c 56 67                  jmp masol    ;usgyi!
00087  66c2                    ;
00088  66c2                    ;
00089  66c2                    omas                ;rom-bol masol
00090  66c2 20 df 67                  jsr rom      ;rom enged.
00091  66c5 4c b5 66                  jmp mas      ;lo. amas
00092  66c8                    ;
00093  66c8                    ;
00094  66c8                    torol               ;torles
00095  66c8 20 e8 67                  jsr ram      ;rom tiltas
00096  66cb ad ff 67                  lda cim3
00097  66ce 85 e0                     sta fe       ;cimek
00098  66d0 ad 00 68                  lda cim3+1   ;atvetele
00099  66d3 85 e1                     sta fe+1
00100  66d5 e6 e0                     inc fe       ;fe:=cim3+1
00101  66d7 d0 02                     bne fe2
00102  66d9 e6 e1                     inc fe+1
00103  66db ad fd 67           fe2    lda cim2
00104  66de 85 e2                     sta fv
00105  66e0 ad fe 67                  lda cim2+1
00106  66e3 85 e3                     sta fv+1     ;fv=cim2
00107  66e5 ad fb 67                  lda ciml
00108  66e8 85 e4                     sta cel
00109  66ea ad fc 67                  lda ciml+1   ;cel:=ciml
00110  66ed 85 e5                     sta cel+1
00111  66ef 4c 56 67                  jmp masol    ;usgyi!
```

```
Line#  Addr Code        Source

00112  66f2             ;
00113  66f2             ;
00114  66f2             abesz                          ;beszuras ram-bol
00115  66f2 20 1f 67           jsr besz                 ;feljebb tolja
00116  66f5             rak                            ;helyere teszi
00117  66f5 ad ff 67           lda cim3
00118  66f8 85 e0              sta fe                   ;fe:=cim3
00119  66fa ad 00 68           lda cim3+1
00120  66fd 85 e1              sta fe+1
00121  66ff ad 01 68           lda cim4
00122  6702 85 e2              sta fv
00123  6704 ad 02 68           lda cim4+1
00124  6707 85 e3              sta fv+1                 ;fv=cim4
00125  6709 ad fb 67           lda cim1
00126  670c 85 e4              sta cel
00127  670e ad fc 67           lda cim1+1               ;cel:=cim1
00128  6711 85 e5              sta cel+1
00129  6713 4c 56 67           jmp masol                ;usgyi!
00130  6716             ;
00131  6716             ;
00132  6716             obesz                          ;beszuras rom-bol
00133  6716 20 1f 67           jsr besz
00134  6719 20 df 67           jsr rom                  ;rom eng.
00135  671c 4c f0 66           jmp rak                  ;ld. abesz
00136  671f             ;
00137  671f             ;
00138  671f             besz                           ;beszuro rutin
00139  671f 20 e8 67           jsr ram                  ;rom tiltas
00140  6722 ad fb 67           lda cim1
00141  6725 85 e0              sta fe                   ;fe:=cim1
00142  6727 ad fc 67           lda cim1+1
00143  672a 85 e1              sta fe+1
00144  672c ad fd 67           lda cim2
00145  672f 85 e2              sta fv                   ;fv:=cim2
00146  6731 ad fe 67           lda cim2+1
00147  6734 85 e3              sta fv+1
00148  6736 38                 sec                      ;cel:=1
00149  6737 ad fb 67           lda cim1                 ;+cim1
00150  673a 6d 01 68           adc cim4                 ;+cim4
00151  673d 85 e4              sta cel
00152  673f ad fc 67           lda cim1+1
00153  6742 6d 02 68           adc cim4+1
00154  6745 85 e5              sta cel+1
00155  6747 38                 sec
00156  6748 a5 e4              lda cel
00157  674a ed ff 67           sbc cim3                 ;-cim3
00158  674d 85 e4              sta cel
00159  674f a5 e5              lda cel+1
00160  6751 ed 00 68           sbc cim3+1
00161  6754 85 e5              sta cel+1
00162  6756             masol                          ;masolorutin
00163  6756 38                 sec
00164  6757 a5 e2              lda fv                   ;db:=fv-fe
00165  6759 e5 e0              sbc fe
00166  675b 85 e6              sta db
00167  675d a5 e3              lda fv+1
```

```
Line#  Addr Code        Source

00168  675f e5 e1             sbc fe+1
00169  6761 85 e7             sta db+1
00170  6763 a5 e5             lda cel+1
00171  6765 c5 e1             cmp fe+1          ;cel<fe?
00172  6767 30 47             bmi jo            ;celhi<fehi
00173  6769 d0 06             bne felso         ;celhi>fehi
00174  676b a5 e4             lda cel
00175  676d c5 e0             cmp fe            ;celhi=fehi es
00176  676f 30 3f             bmi jo            ;cello<felo
00177  6771 a5 e3      felso  lda fv+1
00178  6773 c5 e5             cmp cel+1         ;fv<cel?
00179  6775 30 39             bmi jo            ;fvhi<celhi
00180  6777 d0 06             bne rossz         ;fvhi>celhi
00181  6779 a5 e2             lda fv
00182  677b c5 e4             cmp cel           ;fvhi=celhi es
00183  677d 30 31             bmi jo            ;fvlo<cello
00184  677f             rossz                   ;rossz, vissza kell
00185  677f 18                clc
00186  6780 a5 e4             lda cel           ;cel:=cel+db
00187  6782 65 e6             adc db
00188  6784 85 e4             sta cel
00189  6786 a5 e5             lda cel+1
00190  6788 65 e7             adc db+1
00191  678a 85 e5             sta cel+1
00192  678c a0 00             ldy #$00
00193  678e 78          lecik  sei
00194  678f 8d 3f ff           sta $ff3f         ;ram-rom val.
00195  6792 b1 e2             lda (fv),y
00196  6794 8d 3e ff           sta $ff3e         ;rom eng.
00197  6797 58                cli
00198  6798 91 e4             sta (cel),y
00199  679a a5 e2             lda fv
00200  679c d0 02             bne fv1           ;fv:=fv-1
00201  679e c6 e3             dec fv+1
00202  67a0 c6 e2      fv1    dec fv
00203  67a2 a5 e4             lda cel
00204  67a4 d0 02             bne cel1          ;cel:=cel-1
00205  67a6 c6 e5             dec cel+1
00206  67a8 c6 e4      cel1   dec cel
00207  67aa 20 d0 67           jsr dbcsok
00208  67ad d0 df             bne lecik         ;nem, tovabb
00209  67af 60                rts
00210  67b0             ;
00211  67b0             ;
00212  67b0             jo                      ;jo sorrend, nov.
00213  67b0 a0 00              ldy #$00
00214  67b2 78          felcik sei
00215  67b3 8d 3f ff           sta $ff3f         ;ram-rom val.
00216  67b6 b1 e0             lda (fe),y
00217  67b8 8d 3e ff           sta $ff3e         ;rom eng.
00218  67bb 58                cli
00219  67bc 91 e4             sta (cel),y
00220  67be e6 e0             inc fe
00221  67c0 d0 02             bne fe1           ;fe:=fe+1
00222  67c2 e6 e1             inc fe+1
00223  67c4 e6 e4      fe1    inc cel
```

```
Line#  Addr Code         Source

00224  67c6 d0 02              bne cel2            ;cel:=cel+1
00225  67c8 e6 e5              inc cel+1
00226  67ca 20 d0 67    cel2    jsr dbcsok
00227  67cd d0 e3              bne felcik
00228  67cf 60                 rts
00229  67d0             ;
00230  67d0             ;
00231  67d0             dbcsok                      ;db-szaml. csokk.
00232  67d0 a5 e6              lda db
00233  67d2 d0 02              bne db2             ;db:-1
00234  67d4 c6 e7              dec db+1
00235  67d6 c6 e6      db2     dec db
00236  67d8 a5 e7              lda db+1
00237  67da 25 e6              and db
00238  67dc c9 ff              cmp #$ff            ;elfogyott?
00239  67de 60                 rts
00240  67df             ;
00241  67df             ;
00242  67df             rom                         ;rom eng.
00243  67df a9 3e              lda #$3e
00244  67e1 8d 90 67           sta lecik+2
00245  67e4 8d b4 67           sta felcik+2
00246  67e7 60                 rts
00247  67e8             ;
00248  67e8             ;
00249  67e8             ram                         ;rom tiltas
00250  67e8 a9 3f              lda #$3f
00251  67ea 8d 90 67           sta lecik+2
00252  67ed 8d b4 67           sta felcik+2
00253  67f0 60                 rts
00254  67f1             ;
00255  67f1             ;
          b2 66 c2 66 c8 66 f2 66 16 67
00256  67f1             ugrtab.word amas,omas,torol,abesz,obesz
00257  67fb             cim1    * = *+2
00258  67fd             cim2    * = *+2
00259  67ff             cim3    * = *+2
00260  6801             cim4    * = *+2
00261  6803             mod     * = *+1
00262  6804             hatar   * = *+1

end of assembly, error count = 00000

abesz   66f2    amas    66b2    amasc   66b7    beolvc  6688
besz    671f    cel     00e4    cel1    67a8    cel2    67ca
cim1    67fb    cim2    67fd    cim3    67ff    cim4    6801
cimbe   c38f    db      00e6    db2     67d6    dbcsok  67d0
facint  a327    fe      00e0    fe1     67c4    fe2     66db
felcik  67b2    felso   6771    fv      00e2    fv1     67a0
hatar   6804    hiba    6671    illg    991c    jo      67b0
lecik   679e    linnum  0014    mas     66b5    masol   6756
mehet   6674    mod     6803    negy    6681    obesz   6716
omas    66c2    param   0065    rak     66f5    ram     67e8
rom     67df    rossz   677f    torol   66c8    ugrik   66af
ugrtab  67f1
```

```
10 ifww=1then20:elsegraphic1,1:graphic0,1:ww=1:
load"tukrozes.mod",8,1
20 rem
30 print"Sq]]]a program a kepernyo fuggoleges"
40 print"q]]]vagy vizszintes kozepvonalara "
50 print"q]]]vagy a kozeppontjara vonatkozo"
60 print"q]]]tukrozest hajt vegre."
70 print"q]]]az eredeti es a tukorkep kozotti"
80 print"q]]]and, or ill. eor logikai kap-"
90 print"q]]]csolatot is elorhatjuk."
100 print"qq]]]a menube a 'space' lenyomasaval,"
110 print"q]]]parancsmodban tortent rajzolas "
120 print"q]]]utan pedig a 'help' gomb segitse-"
130 print"q]]]gevel juthatunk.":
140 key8,"Sq0220"+chr$(13)
150 getkeyx$
160 graphic1,1
170 fori=0to360step20:locate270,40:drawto5*log(1+i):i:next
180 forj=30to160step10:locate80,20
190 fori=1to10:drawtoi;j+13*i:nexti,j
200 box,33,77,77,88,23,1:circle,190,70,20,40,,,222:
circle,250,140,40,,,,,60
210 char,5,15,"bemutato abra":char,6,17,"menu:
space":getkeyx$
220 graphic0,1
230 print"q]valassz!"
240 print"qq]  parancs modban rajzolas      r"
250 print"q]  vege                          v"
260 print"qq]  tukrozes    sima  and  or   eor"
270 print"q]fugg. teng.     a    b    c    d"
280 print"q]vissz. teng.    e    f    g    h"
290 print"q]kozeppont       i    j    k    l"
300 print"qja tukrozesi gombokat a menube valo"
310 print"]visszateres nelkul is hasznalhatod!"
320 getkeyx$
330 if x$=" "then220
340 if x$="v"thengraphic0,1:print"qqq     legyen maskor is
szerencsenk!": end
350 if x$="r"thengraphic0,1:print"qqlne felejtsd el: a
visszaut: 'help'!":stop
360 if x$<"a"or x$>"l"then320
370 graphic1:sys26214,asc(x$)-65
380 getkeyx$:goto330
```

```
.....page # 1

Line#  Addr Code        Source

00001  0000             ;put"@0:tukrozes"
00002  0000             ;*********************************************
00003  0000             ;*                                           *
00004  0000             ;*  a program tukrozeseket hajt vegre a nagy- *
00005  0000             ;*  felbontasu kepernyon, figyelembe veve az  *
00006  0000             ;*  altalunk eloirt logikai kapcsolatot az    *
00007  0000             ;*  eredeti es a tukorkep kozott.             *
00008  0000             ;*                                           *
00009  0000             ;*  hivasa: sys26214,x , ahol x az alabbi     *
00010  0000             ;*  tablazatbol kiolvashato egesz parameter:  *
00011  0000             ;*                                           *
00012  0000             ;*                  az eloir logikai kapcsolat *
00013  0000             ;*                     csere  and   or    eor  *
00014  0000             ;* fugg. tengely         0     1     2     3   *
00015  0000             ;* vizsz. tengely        4     5     6     7   *
00016  0000             ;* kozeppont             8     9     10    11  *
00017  0000             ;*                                           *
00018  0000             ;*********************************************
00019  0000                     * = $6666
00020  6666             fel       = $e3           ;felf. szaml.
00021  6666             le        = $e4           ;lef. szaml.
00022  6666             cimb      = $e5           ;balcim helye
00023  6666             cimj      = $e7           ;jobbcim helye
00024  6666             cimf      = $e5           ;felso cim helye
00025  6666             cima      = $e7           ;also cim helye
00026  6666             cime      = $e5           ;elso cim helye
00027  6666             cimu      = $e7           ;utolso cim helye
00028  6666             balk      = $2000         ;balkezdet
00029  6666             jobok     = $2138         ;jobbkezdet
00030  6666             alsk      = $2000+7680    ;also kezdet
00031  6666             felk      = $2000         ;felso kezdet
00032  6666             elsk      = $2000         ;elso kezdet
00033  6666             utok      = $2000+7992    ;utolso kezd.
00034  6666             sorsz     = 25            ;sorok szama
00035  6666             pozszf    = 20            ;pozicio a fel kepernyon
00036  6666             pozszt    = 40            ;pozicio a teljes kepernyon
00037  6666             magas     = 12            ;oszloomag. a felkepernyon
00038  6666             felkep    = 500           ;a fel kepenyo pozic. szama
00039  6666             chkoma    = $9491         ;kov. vesszoig elore
00040  6666             getbyt    = $9d84         ;1 byte x-be
00041  6666             ;
00042  6666             ;
00043  6666 20 91 94           jsr chkoma        ;vesszoig
00044  6669 20 84 9d           jsr getbyt        ;mod x-be
00045  666c 8a                 txa
00046  666d 48                 pha               ;verembe
00047  666e 29 03              and #$03          ;/4 utani
00048  6670 aa                 tax               ;maradek
00049  6671 bd 25 68           lda kodok,x       ;a megfelelo
00050  6674 8d ce 66           sta modb          ;kod betoltese
00051  6677 8d d5 66           sta modj          ;minden
00052  667a 8d 5c 67           sta moda
00053  667d 8d 55 67           sta modf          ;modosito
00054  6680 8d f3 67           sta mode
00055  6683 8d ea 67           sta modu          ;helyre
00056  6686 68                 pla               ;verembol
```

```
Line#   Addr Code        Source

00057   6687 4a                  lsr a
00058   6688 4a                  lsr a                ;osztas 4-gyel
00059   6689 f0 09               beq fugg             ;fugg. tukr.
00060   668b 4a                  lsr a                ;tovabbi 2-vel
00061   668c d0 03               bne kp
00062   668e 4c 1e 67            jmp vizs             ;vizsz. tukr.
00063   6691 4c a4 67    kp      jmp kozepp           ;kozepponfos
00064   6694             ;
00065   6694             ;
00066   6694             fugg                         ;fuggoleges tukr.
00067   6694 a9 00               lda #<balk           ;kezdeti
00068   6696 85 e5               sta cimb             ;ertekek
00069   6698 a9 20               lda #>balk
00070   669a 85 e6               sta cimb+1           ;beallitasa
00071   669c a9 38               lda #<jobbk          ;mindket
00072   669e 85 e7               sta cimj             ;mutatonal
00073   66a0 a9 21               lda #>jobbk
00074   66a2 85 e8               sta cimj+1
00075   66a4 a9 19               lda #sorsz
00076   66a6 8d 33 68            sta sszaml
00077   66a9 a9 14       sork    lda #pozszf          ;sorkezd.
00078   66ab 8d 34 68            sta pszaml
00079   66ae a0 07       pozk    ldy #$07             ;pozkezd.
00080   66b0 a2 08       bytek   ldx #$08             ;bytekezd.
00081   66b2 b1 e5               lda (cimb),y         ;regi bal
00082   66b4 8d 29 68            sta balr
00083   66b7 b1 e7               lda (cimj),y         ;regi jobb
00084   66b9 8d 2b 68            sta jobbr
00085   66bc 0e 29 68    forgv   asl balr             ;balrol
00086   66bf 6e 2c 68            ror jobbu            ;jobbra
00087   66c2 0e 2b 68            asl jobbr            ;jobbrol
00088   66c5 6e 2a 68            ror balu             ;balra
00089   66c8 ca                  dex                  ;kesz?
00090   66c9 d0 f1               bne forgv            ;nem, tovabb
00091   66cb ad 2a 68            lda balu
00092   66ce 31 e5       modb    and (cimb),y
00093   66d0 91 e5               sta (cimb),y         ;csere
00094   66d2 ad 2c 68            lda jobbu
00095   66d5 31 e7       modj    and (cimj),y
00096   66d7 91 e7               sta (cimj),y
00097   66d9 88                  dey                  ;tart meg a pozicio?
00098   66da 10 d4               bpl bytek            ;igen, ujabb byte
00099   66dc 18                  clc                  ;nem
00100   66dd a5 e5               lda cimb             ;balpozicio
00101   66df 69 08               adc #$08             ;novelese
00102   66e1 85 e5               sta cimb             ;8-cal
00103   66e3 a5 e6               lda cimb+1           ;(1 poz.)
00104   66e5 69 00               adc #$00
00105   66e7 85 e6               sta cimb+1
00106   66e9 38                  sec
00107   66ea a5 e7               lda cimj             ;jobbpozicio
00108   66ec e9 08               sbc #$08             ;csokkentese
00109   66ee 85 e7               sta cimj             ;8-cal
00110   66f0 a5 e8               lda cimj+1           ;(1 poz.)
00111   66f2 e9 00               sbc #$00
00112   66f4 85 e8               sta cimj+1
```

```
Line#  Addr Code         Source

00113  66f6 ce 34 68           dec pszaml          ;felsor vege?
00114  66f9 d0 b3              bne pozk            ;nem,vissza
00115  66fb 18                 clc
00116  66fc a5 e5              lda cimb            ;balsormutato
00117  66fe 69 a0              adc #$a0            ;novelese
00118  6700 85 e5              sta cimb            ;160-nal
00119  6702 a5 e6              lda cimb+1          ;(1 felsor)
00120  6704 69 00              adc #$00
00121  6706 85 e6              sta cimb+1
00122  6708 18                 clc
00123  6709 a5 e7              lda cimj            ;jobbsormutato
00124  670b 69 e0              adc #$e0            ;novelese
00125  670d 85 e7              sta cimj            ;480-nal
00126  670f a5 e8              lda cimj+1          ;(3 felsor)
00127  6711 69 01              adc #$01
00128  6713 85 e8              sta cimj+1
00129  6715 ce 33 68           dec sszaml          ;sorok vege?
00130  6718 f0 03              beq vege            ;igen, kesz
00131  671a 4c a9 66           jmp sork            ;nem,vissza
00132  671d 60           vege  rts
00133  671e                    ;
00134  671e                    ;
00135  671e                    viz                 ;vizszintes tukrozes
00136  671e a9 00                 lda #<felk       ;kezdeti
00137  6720 85 e5                 sta cimf         ;ertekek
00138  6722 a9 20                 lda #>felk
00139  6724 85 e6                 sta cimf+1       ;beallitasa
00140  6726 a9 00                 lda #<alsk       ;mindket
00141  6728 85 e7                 sta cima         ;mutatonal
00142  672a a9 3e                 lda #>alsk
00143  672c 85 e8                 sta cima+1
00144  672e a9 0c                 lda #magas       ;felkep magassaga
00145  6730 8d 33 68              sta sszaml
00146  6733 a9 28           sorkv lda #pozsz1      ;sorkezdet
00147  6735 8d 34 68              sta pszaml
00148  6738 a0 07           pozkv ldy #$07         ;teljes
00149  673a a2 00                 ldx #$00         ;karakternyi
00150  673c ad 33 68              lda sszaml       ;utolso sor?
00151  673f d0 04                 bne tolt         ;nem, ->tolt
00152  6741 a0 03                 ldy #$03         ;fel
00153  6743 a2 04                 ldx #$04         ;karakternyi
00154  6745 84 e4           tolt  sty le           ;szamlalok
00155  6747 86 e3                 stx fel          ;toltese
00156  6749 a4 e4           masol ldy le
00157  674b b1 e5                 lda (cimf),y     ;a ket byte
00158  674d 8d 2e 68              sta fbyte        ;masolasa
00159  6750 a4 e3                 ldy fel
00160  6752 b1 e7                 lda (cima),y
00161  6754 8d 2d 68              sta abyte
00162  6757 ad 2e 68              lda fbyte
00163  675a a4 e3                 ldy fel
00164  675c 91 e7           moda  sta (cima),y     ;muvelet
00165  675e 91 e7                 sta (cima),y
00166  6760 ad 2d 68              lda abyte
00167  6763 a4 e4                 ldy le
00168  6765 91 e5           modf  sta (cimf),y     ;muvelet
```

```
Line#   Addr Code          Source

00169   6767 91 e5                 sta (cimf),y
00170   6769 e6 e3                 inc fel
00171   676b c6 e4                 dec le                  ;vege?
00172   676d 10 da                 bpl masol               ;nem, vissza
00173   676f 18                    clc
00174   6770 a5 e5                 lda cimf                ;felso
00175   6772 69 08                 adc #$08                ;mutato
00176   6774 85 e5                 sta cimf                ;novelese
00177   6776 a5 e6                 lda cimf+1              ;8-cal
00178   6778 69 00                 adc #$00                ;(1 poz.)
00179   677a 85 e6                 sta cimf+1
00180   677c 18                    clc
00181   677d a5 e7                 lda cima                ;also
00182   677f 69 08                 adc #$08                ;mutato
00183   6781 85 e7                 sta cima                ;novelese
00184   6783 a5 e8                 lda cima+1              ;8-cal
00185   6785 69 00                 adc #$00                ;(1 poz.)
00186   6787 85 e8                 sta cima+1
00187   6789 ce 34 68              dec pszaml              ;sor vege?
00188   678c d0 aa                 bne pozkv               ;nem, vissza
00189   678e 38                    sec
00190   678f a5 e7                 lda cima                ;also
00191   6791 e9 80                 sbc #$80                ;mutato
00192   6793 85 e7                 sta cima                ;csokkentese
00193   6795 a5 e8                 lda cima+1              ;640-nel
00194   6797 e9 02                 sbc #$02                ;(2 sor)
00195   6799 85 e8                 sta cima+1
00196   679b ce 33 68              dec sszaml              ;van meg sor?
00197   679e 30 03                 bmi vegev               ;nincs, vege
00198   67a0 4c 33 67              jmp sorkv
00199   67a3 60          vegev     rts
00200   67a4             ;
00201   67a4             ;
00202   67a4             kozepp    ;                       ;kozeppontos
00203   67a4 a9 00                 lda #<elsk              ;kezdeti
00204   67a6 85 e5                 sta cime                ;ertekek
00205   67a8 a9 20                 lda #>elsk
00206   67aa 85 e6                 sta cime+1              ;beallitasa
00207   67ac a9 38                 lda #<utok              ;mindket
00208   67ae 85 e7                 sta cimu                ;mutatonal
00209   67b0 a9 74                 lda #>utok
00210   67b2 85 e8                 sta cimu+1
00211   67b4 a9 f4                 lda #<felkep            ;poz. szama
00212   67b6 8d 33 68              sta sszaml
00213   67b9 a9 01                 lda #>felkep
00214   67bb 8d 34 68              sta sszaml+1            ;pszaml
00215   67be a0 07       pozkk     ldy #$07                ;teljes
00216   67c0 a2 00                 ldx #$00                ;karakterny:
00217   67c2 84 e4                 sty le                  ;szamlalok
00218   67c4 86 e3                 stx fel                 ;toltese
00219   67c6 a4 e4       masolk    ldy le
00220   67c8 b1 e5                 lda (cime),y            ;a ket
00221   67ca 8d 2f 68              sta elsr                ;byte
00222   67cd a4 e3                 ldy fel                 ;masolasa
00223   67cf b1 e7                 lda (cimu),y
00224   67d1 8d 30 68              sta utor
```

```
Line#  Addr Code        Source

00225  67d4 a2 08           ldx #$08          ;forg. szama
00226  67d6 0e 2f 68  forgk  asl elsr
00227  67d9 6e 32 68        ror utou          ;a ket byte
00228  67dc 0e 30 68        asl utor          ;tukrozese
00229  67df 6e 31 68        ror elsu
00230  67e2 ca              dex
00231  67e3 d0 f1           bne forgk
00232  67e5 ad 32 68        lda utou
00233  67e8 a4 e3           ldy fel
00234  67ea 91 e7     modu  sta (cimu),y      ;muvelet
00235  67ec 91 e7           sta (cimu),y      ;visszairas
00236  67ee ad 31 68        lda elsu
00237  67f1 a4 e4           ldy le
00238  67f3 91 e5     mode  sta (cime),y      ;muvelet
00239  67f5 91 e5           sta (cime),y      ;visszairas
00240  67f7 e6 e3           inc fel
00241  67f9 c6 e4           dec le            ;vege?
00242  67fb 10 c9           bpl masolk        ;nem, vissza
00243  67fd 18              clc
00244  67fe a5 e5           lda cime          ;elso cim
00245  6800 69 08           adc #$08          ;novelese
00246  6802 85 e5           sta cime          ;8-cal
00247  6804 a5 e6           lda cime+1        ;(1 poz.)
00248  6806 69 00           adc #$00
00249  6808 85 e6           sta cime+1
00250  680a 38              sec
00251  680b a5 e7           lda cimu
00252  680d e9 08           sbc #$08          ;utolso cim
00253  680f 85 e7           sta cimu          ;csokkentese
00254  6811 a5 e8           lda cimu+1        ;8-cal
00255  6813 e9 00           sbc #$00          ;(1 poz.)
00256  6815 85 e8           sta cimu+1
00257  6817 ce 33 68        dec sszaml        ;uj poz.
00258  681a d0 03           bne ugrk
00259  681c ce 34 68        dec sszaml+1
00260  681f ad 34 68  ugrk   lda sszaml+1      ;elfogyott?
00261  6822 10 9a           bpl pozkk         ;nem, ->pozkk
00262  6824 60              rts
       91
00263  6825           kodok .byte $91         ;sta (cc),x
       31
00264  6826                 .byte $31         ;and (cc),x
       11
00265  6827                 .byte $11         ;ora (cc),x
       51
00266  6828                 .byte $51         ;eor (cc),x
       00
00267  6829           balr  .byte 0           ;regi balold.
       00
00268  682a           balu  .byte 0           ;uj baloldal
       00
00269  682b           jobbr .byte 0           ;regi jobbold.
       00
00270  682c           jobbu .byte 0           ;uj jobboldal
       00
00271  682d           abyte .byte       0     ;also byte tar.
```

```
.....page # 6

Line#  Addr Code        Source


            00
00272  682e             fbyte .byte          0      ;felso byte tar.
            00
00273  682f             elsr  .byte          0      ;regi elso
            00
00274  6830             utor  .byte          0      ;regi utolso
            00
00275  6831             elsu  .byte          0      ;uj elso
            00
00276  6832             utou  .byte          0      ;uj utolso
            00
00277  6833             sszaml.byte 0               ;sorszamlalo
            00
00278  6834             pszaml.byte 0               ;szamlalo a soron belu
00279  6835 ea                  nop

end of assembly, error count = 00000

abyte   682d     alsk    3e00     balk    2000     balr    6829
balu    682a     bytek   66b0     chkoma  9491     cima    00e7
cimb    00e5     cime    00e5     cimf    00e5     cimj    00e7
cimu    00e7     elsk    2000     elsr    682f     elsu    6831
fbyte   682e     fel     00e3     felk    2000     felkep  0144
forgk   67d6     forgv   66bc     fugg    6694     getbyt  9d84
jobbk   2138     jobbr   682b     jobbu   682c     kodok   6825
kozepp  67a4     kp      6691     le      00e4     magas   000c
masol   6749     masolk  67c6     moda    675c     modb    66ce
mode    67f3     modf    6765     modj    66d5     modu    67ea
pozk    66ae     pozkk   67be     pozkv   6738     pozszf  0014
pozszt  0028     pszaml  6834     sork    66a9     sorkv   6733
sorsz   0019     sszaml  6833     tolt    6745     ugrk    621f
utok    3f38     utor    6830     utou    6832     vege    671d
vegev   67a3     viz     671e
```

```
10 ifww=0thenww=1:load"prim.mod",8,1
20 poke1281,102:poke1282,102
30 print"ssSqqq  a program harom lehetoseget nyujt:"
40 print"qqqq  r1R egy adott szam prim voltanak el-
dontese"
50 print"qqq  r2R egy adott szam primtenyezos fel-
bontasa"
60 print"qqq  r3R primszamok ket adott szam kozott"
70 getkeyx$
80 ifx$<"1"orx$>"3"then70
90 onval(x$)goto100,150,290
100 print"ssS"
110 gosub430:sz=s
120 ifusr(sz)=0thenprint"qq  primszam!":elseprint"qq
osszetett szam!"
130 gosub450:ifx$="v"thenend
140 ifx$="i"then100:else30
150 print"ssS"
160 gosub430:sz=s:k=1:l=0:c=0
170 do
180 : p=usr(sz)
190 : ifp=0thenexit
200 : ifl<pthenk=k+1
210 : printp;
220 : l=p
230 : sz=sz/p
240 : ifsz=pthenc=1
250 loop
260 ifk=1thenprint"q primszam":elseprintsz;:print"q
osszesen";k-c;"kulonbozo primosztoja van."
270 gosub450:ifx$="v"thenend
280 ifx$="i"then150:else30
290 print"ssSqq  ird be a kisebbik szammot!":k=0
300 gosub430:sk=s
310 print"qq  ird be a nagyobbik szamot!"
320 gosub430:sn=s
330 ifsk>snthenprint"qq  nagyobb szamot kerek!":goto320
340 s=sk:print"qqq  a primszamok";s;"es";sn;"kozott:":
print:ifs=2thenprint2;:k=1
350 ifs=2*int(s/2)thens=s+1
360 downiles<=sn
370 : ifusr(s)=0thenprints;:k=k+1
380 : s=s+2
390 loop
400 print:print"q osszesen";k;"primszam vanq":printsk;
"es";sn;"kozott."
410 gosub450:ifx$="v"thenend
420 ifx$="i"then290:else30
430 s=0:input"qq  a szam:";s
440 ifs<=1thenprint"qq  1-nel nagyobb szamot kerek!":
goto430:elseprint:return
450 print"qq  r1Rsmet/rmRenu/rvRege?"
460 getkeyx$:ifx$<>"v"andx$<>"i"andx$<>"m"then460:elsereturn
```

```
line#  Addr Code          Source

00001  0000              ;put"@0:prim
00002  0000              ;*********************************************
00003  0000              ;*                                           *
00004  0000              ;* a program egy 2 147 483 648-nal kisebb    *
00005  0000              ;* sz egesz szamrol eldonti, hogy prim-e,    *
00006  0000              ;* ha nem, megadja a legkisebb primosztojat. *
00007  0000              ;*                                           *
00008  0000              ;* az usr-vektort at kell irni $6666-ra!     *
00009  0000              ;*                                           *
00010  0000              ;* hivasa: p=usr(sz) , ahol sz a szam,       *
00011  0000              ;* p=0, ha sz prim,                          *
00012  0000              ;* p=1, ha sz=1 es                           *
00013  0000              ;* p sz-nek egy primosztoja egyebkent        *
00014  0000              ;*                                           *
00015  0000              ;*********************************************
00016  0000                     * = $6666
00017  6666              facint   = $a327          ;fac:=int(fac)
00018  6666              sqr      = $a5e4          ;fac:=sqr(fac)
00019  6666              tfa      = $a291          ;arg:=fac
00020  6666              taf      = $a281          ;fac:=arg
00021  6666              primm    = $ff4f          ;szoveg kiirasa
00022  6666              tulcs    = $9fb2          ;'overflow error' ki
00023  6666              intfl    = $9f0b          ;egesz->lebegop.(fac
00024  6666              fac      = $61
00025  6666              arg      = $69
00026  6666              round    = $70            ;kerekites
00027  6666              ;
00028  6666              ;
00029  6666              tszam                     ;a szam attoltese
00030  6666 20 91 a2           jsr tfa             ;mentes arg-ba
00031  6669 20 27 a3           jsr facint          ;egessze konv.
00032  666c a5 61              lda fac             ;kitevo
00033  666e f0 25              beq paros           ;szam=0, ->paros
00034  6670 c9 a0        kitevo cmp #$a0           ;belefer?
00035  6672 30 03              bmi tovabb          ;>0, jo
00036  6674 4c b2 9f           jmp tulcs           ;'overflow'
00037  6677 a9 00        tovabb lda #$00
00038  6679 05 62              ora fac+1
00039  667b 05 63              ora fac+2
00040  667d 05 64              ora fac+3
00041  667f d0 0e              bne mehet           ;>256, mehet
00042  6681 a5 65              lda fac+4           ;nem, egybyte-os
00043  6683 29 fc              and #$fc            ;felso 6 bit =0?
00044  6685 d0 08              bne mehet           ;nem, mehet
00045  6687 a5 65              lda fac+4
00046  6689 c9 01              cmp #$01
00047  668b f0 7d              beq egy             ;=1
00048  668d d0 77              bne prim            ;=2 v. 3
00049  668f a5 65        mehet lda fac+4           ;utolso byte
00050  6691 29 01              and #$01            ;paros?
00051  6693 d0 13              bne ptl             ;nem,->ptl
00052  6695 a9 02        paros lda #$02            ;paros, tehat
00053  6697 8d c3 67           sta hely+3          ;a 2 osztja
00054  669a a9 00              lda #$00            ;2->hely
00055  669c 8d c0 67           sta hely
00056  669f 8d c1 67           sta hely+1
```

```
Line#  Addr Code        Source

00057  66a2 8d c2 67          sta hely+2
00058  66a5 4c 1f 67          jmp helyki          ;ki: hely
00059  66a8 a0 03       ptl   ldy #$03            ;szam
00060  66aa b9 62 00    szcikl lda fac+1,y        ;helyere
00061  66ad 99 bc 67          sta szam,y          ;toltese
00062  66b0 88                dey
00063  66b1 10 f7             bpl szcikl
00064  66b3             tgyok                     ;a gyok szamitasa
00065  66b3 20 81 a2          jsr taf             ;fac:=arg
00066  66b6 20 e4 a5          jsr sqr             ;fac:=sqr(fac)
00067  66b9 20 27 a3          jsr facint          ;egessze konv.
00068  66bc a0 03             ldy #$03            ;gyok
00069  66be b9 62 00  .gycikl lda fac+1,y        ;helyere
00070  66c1 99 c4 67          sta gyok,y          ;toltese
00071  66c4 88                dey
00072  66c5 10 f7             bpl gycikl
00073  66c7 a9 03             lda #$03            ;hely
00074  66c9 8d c3 67          sta hely+3          ;kezdoertek
00075  66cc a9 00             lda #$00            ;=3
00076  66ce 8d c2 67          sta hely+2
00077  66d1 8d c1 67          sta hely+1
00078  66d4 8d c0 67          sta hely
00079  66d7             focikl                    ;fociklus, adott szam-hoz
00080  66d7 20 7a 67          jsr todo            ;osztando toltese
00081  66da 20 86 67          jsr teszto          ;oszto toltese
00082  66dd 20 2c 67          jsr marad           ;maradek van-e?
00083  66e0 a9 00             lda #$00            ;a maradeknak
00084  66e2 a0 03             ldy #$03            ;mind a
00085  66e4 19 b4 67    nulc  ora odo,y           ;negy
00086  66e7 88                dey                 ;byte-ja
00087  66e8 10 fa             bpl nulc            ;nulla?
00088  66ea c9 00             cmp #$00
00089  66ec f0 31             beq helyki          ;igen, nem prim
00090  66ee 20 92 67          jsr helyno          ;akt. oszto nov.2-vel
00091  66f1             vizsg                     ;lehet-e meg?
00092  66f1 a0 00             ldy #$00            ;novelni?
00093  66f3 b9 c4 67    vcikl lda gyok,y
00094  66f6 d9 c0 67          cmp hely,y
00095  66f9 f0 04             beq vugr            ;hely,y=gyok,y,->tovabb
00096  66fb 30 09             bcc prim            ;hely>gyok, eleg
00097  66fd 10 d8             bcs focikl          ;hely<=gyok, tovabb
00098  66ff c8          vugr  iny                 ;kov. byte
00099  6700 c0 04             cpy #$04            ;utolso?
00100  6702 d0 ef             bne vcikl           ;nem, vizsg. folyt.
00101  6704 f0 d1             beq focikl          ;tovabb
00102  6706             ;
00103  6706             ;
00104  6706 a9 00       prim  lda #$00            ;prim eseten
00105  6708 85 65             sta fac+4           ;ki: 0
00106  670a a9 00       egy   lda #$00            ;a felso
00107  670c 85 62             sta fac+1           ;3 byte
00108  670e 85 63             sta fac+2           ;nullazasa
00109  6710 85 64             sta fac+3
00110  6712 a9 00       kiir  lda #$00            ;elojel=0 .
00111  6714 85 66             sta fac+5
00112  6716 85 70             sta round           ;kerek.=0
```

```
ine#  Addr Code        Source

0113  6718 a9 a0              lda #$a0
0114  671a 85 61              sta fac              ;kitevo
0115  671c 4c 0b 9f           jmp intfl            ;ertek ki
0116  671f                   ;
0117  671f                   ;
0118  671f                   helyki                ;hely
0119  671f a2 04              ldx #$04             ;attoltese
0120  6721 bd bf 67          hkcik lda hely-1,x   ;fac-ba
0121  6724 95 61              sta fac,x
0122  6726 ca                 dex
0123  6727 d0 f8              bne hkcik
0124  6729 4c 12 67           jmp kiir             ;ertek ki
0125  672c                   ;
0126  672c                   ;
0127  672c                   marad                 ;maradekot szamo
0128  672c a9 00              lda #$00             ;egy adott oszto
0129  672e 8d c8 67           sta szaml
0130  6731 2c b8 67          fel   bit oszto       ;legfelso bit=0?
0131  6734 30 12              bmi ki               ;igen, kiszall
0132  6736 0e bb 67           asl oszto+3          ;utkozesig
0133  6739 2e ba 67           rol oszto+2          ;balra
0134  673c 2e b9 67           rol oszto+1          ;leptetes
0135  673f 2e b8 67           rol oszto
0136  6742 ee c8 67           inc szaml            ;+szamlalas
0137  6745 4c 31 67           jmp fel
0138  6748 a0 03              ki    ldy #3         ;odo:=
0139  674a 38                 sec                  ;odo-
0140  674b b9 b4 67          kivcik lda odo,y      ;oszto,
0141  674e f9 b8 67           sbc oszto,y          ;ahanyszor
0142  6751 99 b4 67           sta odo,y            ;csak lehet
0143  6754 88                 dey                  ;(+meg egyszer)
0144  6755 10 f4              bpl kivcik
0145  6757 b0 ef              bcs ki               ;eredmeny>=0->to
0146  6759 a0 03              ldy #3               ;eredmeny<0
0147  675b 18                 clc                  ;egyszer vissza
0148  675c b9 b4 67          hadcik lda odo,y      ;kell adni
0149  675f 79 b8 67           adc oszto,y          ;odo:=
0150  6762 99 b4 67           sta odo,y            ;odo+oszto
0151  6765 88                 dey
0152  6766 10 f4              bpl hadcik
0153  6768 4e b8 67           lsr oszto            ;1-gyel
0154  676b 6e b9 67           ror oszto+1          ;jobbra
0155  676e 6e ba 67           ror oszto+2          ;leptetes
0156  6771 6e bb 67           ror oszto+3
0157  6774 ce c8 67           dec szaml            ;lehet meg?
0158  6777 10 cf              bpl ki               ;igen, ismet
0159  6779 60                 rts
0160  677a                   ;
0161  677a                   ;
0162  677a                   todo                  ;osztando felto
ol
0163  677a a0 03                    ldy #$03
0164  677c b9 bc 67          todoc   lda szam,y
0165  677f 99 b4 67                  sta odo,y     ;negy byte
0166  6782 88                        dey
0167  6783 10 f7                    bpl todoc
0168  6785 60                        rts
```

```
  Addr Code         Source


  6786              ;
  6786              toszto                        ;oszto feltoltese a hely-bol
  6786 a0 03                ldy #$03
  6788 b9 c0 67     toszc   lda hely,y
  678b 99 b8 67             sta oszto,y           ;negy byte
  678e 88                   dey
  678f 10 f7                bpl toszc
  6791 60                   rts
  6792              ;
  6792              ;
  6792              helyno                        ;az akt.  oszto novelese
  6792 18                   clc                   ;2-vel
  6793 a9 02                lda #$02
  6795 6d c3 67             adc hely+3
  6798 8d c3 67             sta hely+3            ;hely:=
  679b a9 00                lda #$00              ;hely+2
  679d 6d c2 67             adc hely+2
  67a0 8d c2 67             sta hely+2
  67a3 a9 00                lda #$00
  67a5 6d c1 67             adc hely+1
  67a8 8d c1 67             sta hely+1
  67ab a9 00                lda #$00
  67ad 6d c0 67             adc hely
  67b0 8d c0 67             sta hely
  67b3 60                   rts
  67b4              ;
  67b4              ;
       00 00 00 00
  67b4              odo     .byte $00,$00,$00,$00;osztando
       00 00 00 00
  67b8              oszto   .byte $00,$00,$00,$00;oszto
       00 00 00 00
  67bc              szam    .byte $00,$00,$00,$00;eredeti szam
       00 00 00 00
  67c0              hely    .byte $00,$00,$00,$00;az akt. oszto
       00 00 00 00
  67c4              gyok    .byte $00,$00,$00,$00;sqr(szam)
       00
  67c8              szaml   .byte $00

: assembly, error count = 00000

   0069     egy      670a     fac      0061     facint   a327
   6731     focikl   66d7     gycikl   66be     gyok     67c4
   675c     hely     67c0     helyki   671f     helyno   6792
   6721     intfl    9f0b     ki       6748     kiir     6712
   6670     kivcik   674b     marad    672c     mehet    668f
   66e4     odo      67b4     oszto    67b8     paros    6695
   6706     primm    ff4f     ptl      66a8     round    0070
   a5e4     szam     67bc     szaml    67c8     szcikl   66aa
   a281     tfa      a291     tgyok    66b3     todo     677a
   677c     toszc    6788     toszto   6786     tovabb   6677
   6666     tulcs    9fb2     vcikl    66f3     vizsg    66f1
   66ff
```

```
10 ifww=1then30
20 ww=1:load"ketkep.mod",8,1
30 print"Sqqq]]]ha nem ismered az uj utasitasokat,"
40 print"qqq]]]hivd be a 'demo' cimu programot!qqqqqq"
50 sysdec("1100")
```

```
10 print"ssSqqq  a program bemutatja a ket grafikus"
20 print"q  es a ket karakteres kepernyo haszna-"
30 print"q  lataval osszefuggo uj utasitasokat"
40 print"q  es alkalmazasukat. kiir egy-egy"
50 print"q  utasitast, es a kepernyon kovetheto"
60 print"q  ennek hatasa."
70 print"qqq":gosub890
80 print"Sqqq  a 12 uj utasitas mindegyike parancs"
90 print"q  modban is hasznalhato:"
100 print"q        kl"
110 print"q        ks"
120 print"q        kc"
130 print"q        gl,gla,glo,gle"
140 print"q        gs,gsa,gso,gse"
150 print"q        gc"
160 print"qqq":gosub890:gosub700:print:gosub890
170 gosub700
180 print"ssSqqq   karakteres 1."
190 ks
200 print"Sqqqqqqqq   karakteres 2."
210 kc
220 graphic1,1:char,20,8,"grafikus 2.":box,33,77,55,111,23
230 gc
240 graphic1,1:char,20,3,"grafikus 1.":fori=0to330step30
:locate160,100:drawto60;i:next
250 graphic0
260 print"sqqqqqqqqqqqqqqqqqqqq"+chr$(27)+"t"
270 print"Sq      karakteres kepernyo muveletek
    bemutatasa."
280 gosub890
290 print"kc":print"  a lathato kep es a masik kep csereje
          (1:=2 es 2:=1)"
300 gosub890:kc:print"kc"
310 print"     ugyanez meg egyszer, visszafele":gosub890:kc
320 print"qks":print"    a lathato kep elmentese  (2:=1)":
gosub890:ks
330 print"S                elmentette":print:gosub890:gosub910
340 a$="most irunk erre a kepre egy szoveget":gosub840
350 print"kc":print"  a lathato kep es a masik kep csereje
          (1:=2 es 2:=1)"
360 gosub890:kc:gosub910
```

```
370 a$="erre megkulonboztetesul egy masikat":gosub840
380 print"kc":print"  a lathato kep es a masik kep csereje
          (1:=2 es 2:=1)"
390 gosub890:kc:print"S              visszacserelte"
:print:gosub890
400 print"ssSqqqqqqqqqqqqqqqqqqqq"+chr$(27)+"t"
410 print"           letoroltuk a kepet"
420 print"kl":print"    a lathato kep feltoltese  (1:=2)"
:gosub890:kl
430 print"S":gosub890
440 graphic2
450 print"Sq        grafikus kepernyo muveletek
      bemutatasa."
460 gosub890
470 print"gc":print"  a lathato kep es a masik kep csereje
          (1:=2 es 2:=1)"
480 gosub890:gc:print"qqqgc"
490 print"     ugyanez meg egyszer, visszafele":gosub890:gc
500 print"qgs":print" a lathato kep elmentese siman (2:=1)"
:gosub890:gs
510 print"qqq               elmentette":print:gosub890:scnclr
520 print"sqqqqqqqqqqqqqqqqqqq"+chr$(27)+"t           ide
most rajzolunk!"
530 fori=0to150step30:circle,160,100,10,45,,,i:next
540 print"qqqgc":print"  a lathato kep es a masik kep csereje
          (1:=2 es 2:=1)"
550 gosub890:gc
560 print"qqqgc":print"  a lathato kep es a masik kep csereje
          (1:=2 es 2:=1)"
570 gosub890:gc:print"qqq             visszacserelte":
print:gosub890
580 print"gla":print"   a lathato kep feltoltese and-del"
590 print"             (1:=1and2)":gosu
600 scnclr:print"sqqqqqqqqqqqqqqqqqqqq"+chr$(27)+"t
ide most rajzolunk!"
610 fori=0to150step30:circle,160,100,10,45,,,i:next
620 print"glo":print"   a lathato kep feltoltese or-ral"
630 print"              (1:=1or2)":gosub890:glo:print:
print:print:gosub890
640 scnclr:print"sqqqqqqqqqqqqqqqqqqqq"+chr$(27)+"t
ide most rajzolunk!"
650 fori=0to150step30:circle,160,100,10,45,,,i:next
660 print"gle":print"  a lathato kep feltoltese eor-ral"
670 print"             (1:=1eor2)":gosub890:gle
680 print"qqqgle":print"  meg egyszer az eor-ral (1:=1eor2) "
:gosub890:gle
690 print"q        visszakaptuk az eredetit":print:gosub890
:gosub700:end
```

```
700 graphic0:print"ss5q  az uj utasitasok szintaxisa:"
710 print"q  1. karakter k - karakteres kepernyo"
720 print"q                g - grafikus kepernyo"
730 print"q  2. karakter l - toltes
740 print"q                s - mentes
750 print"q                c - csere
760 print"q  3. karakter nincs - sima
770 print"q                a - and M
780 print"                        M csak
790 print"               o - or  CC gl vagy gs
800 print"                        N utan
810 print"               e - eor N
820 print"q  jo munkat!"
830 return
840 fori=1tolen(a$)
850 k=asc(mid$(a$,i,1))
860 if k=32thenpoke3073+i,32:goto880
870 poke3073+i,asc(mid$(a$,i,1))-64
880 next:return
890 print"          nyomj meg egy gombot!"
900 getkeyx$:return
910 forw9=1to1000:next:return
```

```
Line#  Addr Code        Source

00001  0000            ;put"@0:ketkep"
00002  0000            ;*****************************************
00003  0000            ;*                                       *
00004  0000            ;*  a program segitsegevel ket grafikus es   *
00005  0000            ;*  ket karakteres kepernyot hasznalhatunk,  *
00006  0000            ;*  melyek kozott 12 (parancs modban is      *
00007  0000            ;*  hasznalhato) utasitas segitsegevel muve- *
00008  0000            ;*  leteket vegezhetunk.                     *
00009  0000            ;*                                           *
00010  0000            ;*  a program hivasa: sys 26214, az aktivi-  *
00011  0000            ;*  zalas utan hasznalhato utasitasok:       *
00012  0000            ;*  kl,ks,kc : toltes, mentes, csere         *
00013  0000            ;*             a karakteres kepernyok kozott,*
00014  0000            ;*  gl,gs,gc : toltes, mentes, csere         *
00015  0000            ;*             a grafikus kepernyok kozott,  *
00016  0000            ;*  gla,glo,gle,gsa,gso,gse : gl ill. gs     *
00017  0000            ;*  and-del, or-ral, eor-ral.                *
00018  0000            ;*                                           *
00019  0000            ;*****************************************
00020  0000                    * = $1100
00021  1100            ftokv   = $030c          ;felh. ut. tokenizalas v.
00022  1100            flistv  = $030e          ;felh. ut. listazas v.
00023  1100            fvegrv  = $0310          ;felh. ut. vegrehajtas v.
00024  1100            chrget  = $0473          ;1 kar. olvasasa
00025  1100            token   = $8a07          ;token azonositasa
00026  1100            fhtok   = $89d6          ;felhaszn. token
00027  1100            nfhtok  = $896c          ;nem felh. token
00028  1100            ltok    = $8b9c          ;token ker. listazashoz
00029  1100            txttab  = $2b            ;a basic terulet kezdete
00030  1100            mvdflg  = $75            ;grafikus ter. van-e?
00031  1100            new     = $8a7b          ;basic new
00032  1100            ready   = $867e          ;basic indit.s
00033  1100            klathe  = $0800          ;kar. lathato eleje
00034  1100            krakte  = $4000          ;kar. raktar eleje
00035  1100            khossz  = $07ff          ;kar. kep hossza
00036  1100            glathe  = $2000          ;graf. lathato eleje
00037  1100            grakte  = $5000          ;graf. raktar eleje
00038  1100            gnossz  = $1fff          ;graf. kep hossza
00039  1100            lath    = $00e0          ;cimek
00040  1100            rakt    = $00e2          ;helye
00041  1100            hossz   = $00e6          ;hossz helye
00042  1100            ;
00043  1100            ;
00044  1100 78                  sei
00045  1101 a9 33               lda #<ftok       ;felh. ut.
00046  1103 8d 0c 03            sta ftokv        ;tokenizalas
00047  1106 a9 11               lda #>ftok       ;helye
00048  1108 8d 0d 03            sta ftokv+1
00049  110b a9 47               lda #<flist      ;felh. ut.
00050  110d 8d 0e 03            sta flistv       ;listazas
00051  1110 a9 11               lda #>flist      ;helye
00052  1112 8d 0f 03            sta flistv+1
00053  1115 a9 53               lda #<fvegr      ;felh. ut.
00054  1117 8d 10 03            sta fvegrv       ;vegrehajtas
00055  111a a9 11               lda #>fvegr      ;helye
00056  111c 8d 11 03            sta fvegrv+1
```

```
Line#  Addr Code        Source

00057  111f 58                cli
00058  1120           helyf                    ;helyfoglalas a kepekn
00059  1120 a9 70             lda #$70         ;a basic ter.
00060  1122 85 2c             sta txttab+1     ;uj kezdete
00061  1124 a9 00             lda #$00         ;nullazas
00062  1126 8d 00 70          sta $7000        ;run-hoz
00063  1129 a9 ff             lda #$ff         ;grafikus mod
00064  112b 85 75             sta mvdflg       ;beallitasa
00065  112d 20 7b 8a          jsr new          ;basic new
00066  1130 4c 7e 86          jmp ready        ;vissza az interpreter
00067  1133           ;
00068  1133           ;
00069  1133           ftok                     ;felh. ut. tokenizalas
00070  1133 48                pha
00071  1134 a0 34             ldy #<ktabl      ;kulcsszotabla
00072  1136 a9 12             lda #>ktabl      ;cime
00073  1138 20 07 8a          jsr token        ;token azonositasa
00074  113b 68                pla
00075  113c 90 06             bcc vftok        ;nem felh. token
00076  113e a5 0b             lda $0b          ;talalt, sorszam
00077  1140 48                pha              ;verembe
00078  1141 4c d6 89          jmp fhtok        ;felhaszn. token
00079  1144 4c 6c 89    vftok jmp nfhtok       ;nem felh. token
00080  1147           ;
00081  1147           ;
00082  1147           flist                    ;felh. ut. listazasa
00083  1147 aa                tax
00084  1148 84 49             sty $49
00085  114a a0 12             ldy #>ktabl      ;kulcsszotabla
00086  114c 84 23             sty $23
00087  114e a0 34             ldy #<ktabl      ;cime
00088  1150 4c 9c 8b          jmp ltok         ;token keresese
00089  1153           ;
00090  1153           ;
00091  1153           fvegr                    ;felh. ut. vegrehajtas
00092  1153 38                sec
00093  1154 e9 80             sbc #$80         ;sorszam
00094  1156 a8                tay              ;tarolasa
00095  1157 29 08             and #$08
00096  1159 d0 05             bne sima         ;sima sta
00097  115b 98                tya
00098  115c 29 03             and #$03
00099  115e aa                tax
       115f 2c
00100  115f                .byte $2c          ;atl.bit
00101  1160 a2 03       sima  ldx #$03         ;a szuks.
00102  1162 bd 2f 12          lda kodok,x      ;modositas
00103  1165 8d c8 11          sta mod1         ; beirasa
00104  1168 8d d4 11          sta mod2
00105  116b 98                tya
00106  116c 0a                asl a            ;2*
00107  116d a8                tay
00108  116e b9 54 12          lda ctabl+1,y    ;cimtablazatbol
00109  1171 48                pha              ;a rutin
00110  1172 b9 53 12          lda ctabl,y      ;cime
00111  1175 48                pha              ;a verembe
```

```
Line#  Addr Code        Source

00112  1176 4c 73 04          jmp chrget          ;folytatas
00113  1179               ;
00114  1179               ;
00115  1179               kar                     ;karakt. kep
00116  1179 a9 00               lda #<klathe
00117  117b 85 e0               sta lath
00118  117d a9 08               lda #>klathe
00119  117f 85 e1               sta lath+1
00120  1181 a9 00               lda #<krakte
00121  1183 85 e2               sta rakt
00122  1185 a9 40               lda #>krakte
00123  1187 85 e3               sta rakt+1
00124  1189 a9 ff               lda #<khossz
00125  118b 85 e6               sta hossz
00126  118d a9 07               lda #>khossz
00127  118f 85 e7               sta hossz+1
00128  1191 4c ac 11            jmp csere
00129  1194               ;
00130  1194               ;
00131  1194               gra                     ;graf. kep
00132  1194 a9 00               lda #<glathe
00133  1196 85 e0               sta lath
00134  1198 a9 20               lda #>glathe
00135  119a 85 e1               sta lath+1
00136  119c a9 00               lda #<grakte
00137  119e 85 e2               sta rakt
00138  11a0 a9 50               lda #>grakte
00139  11a2 85 e3               sta rakt+1
00140  11a4 a9 ff               lda #<ghossz
00141  11a6 85 e6               sta hossz
00142  11a8 a9 1f               lda #>ghossz
00143  11aa 85 e7               sta hossz+1
00144  11ac               ;
00145  11ac               ;
00146  11ac               csere                   ;lath. es rakt.
00147  11ac               ;kozott
00148  11ac               ;hossz db byte
00149  11ac a0 00               ldy #$00
00150  11ae ad 06 ff            lda $ff06          ;kep
00151  11b1 29 ef               and #$ef
00152  11b3 8d 06 ff            sta $ff06          ;letiltasa
00153  11b6 b1 e0        cscik  lda (lath),y       ;ciklus indul
00154  11b8 8d 2d 12            sta lbyte
00155  11bb b1 e2               lda (rakt),y
00156  11bd 8d 2e 12            sta rbyte
00157  11c0 2c 33 12            bit mit            ;kell l->r?
00158  11c3 10 07               bpl lrnem          ;nem, tovabb
00159  11c5 ad 2d 12            lda lbyte
00160  11c8 91 e2        mod1   sta (rakt),y       ;muvelet
00161  11ca 91 e2               sta (rakt),y       ;kirakja
00162  11cc 2c 33 12      lrnem bit mit            ;kell r->l?
00163  11cf 50 07               bvc rlnem          ;nem, tovabb
00164  11d1 ad 2e 12            lda rbyte
00165  11d4 91 e0        mod2   sta (lath),y       ;muvelet
00166  11d6 91 e0               sta (lath),y       ;kirakja
00167  11d8 e6 e0        rlnem  inc lath
```

```
Line#  Addr Code        Source

00168  11da d0 02             bne at1             ;leptetes
00169  11dc e6 e1             inc lath+1
00170  11de e6 e2       at1   inc rakt            ;mindket
00171  11e0 d0 02             bne at2
00172  11e2 e6 e3             inc rakt+1
00173  11e4 c6 e6       at2   dec hossz           ;hossz
00174  11e6 d0 02             bne vizsg
00175  11e8 c6 e7             dec hossz+1         ;csokkentese
00176  11ea a5 e7       vizsg lda hossz+1         :elfogyott?
00177  11ec 10 c8             bpl cscik           ;nem, tovabb
00178  11ee ad 06 ff          lda $ff06           ;kep
00179  11f1 09 10             ora #$10
00180  11f3 8d 06 ff          sta $ff06           ;engedelyezese
00181  11f6 60                rts
00182  11f7             ;
00183  11f7             ;
00184  11f7             ;vegrehajtas
00185  11f7             gl                        ;r->l kell
00186  11f7             gla
00187  11f7             glo
00188  11f7 ea          gle   nop                 ;belep. pont
00189  11f8 a9 40             lda #$40            ;v-bit=1
00190  11fa 8d 33 12          sta mit
00191  11fd 4c 94 11          jmp gra
00192  1200             gs                        ;l->r kell
00193  1200             gsa
00194  1200             gso
00195  1200 ea          gse   nop                 ;belep. pont
00196  1201 a9 80             lda #$80            ;n-bit=1
00197  1203 8d 33 12          sta mit
00198  1206 4c 94 11          jmp gra
00199  1209 ea          gc    nop                 ;r->l kell
00200  120a a9 c0             lda #$c0            ;l->r kell
00201  120c 8d 33 12          sta mit             ;v=n=1
00202  120f 4c 94 11          jmp gra
00203  1212 ea          kl    nop                 ;r->l kell
00204  1213 a9 40             lda #$40            ;v-bit=1
00205  1215 8d 33 12          sta mit
00206  1218 4c 79 11          jmp kar
00207  121b ea          ks    nop                 ;l->r kell
00208  121c a9 80             lda #$80            ;n-bit=1
00209  121e 8d 33 12          sta mit
00210  1221 4c 79 11          jmp kar
00211  1224 ea          kc    nop                 ;r->l kell
00212  1225 a9 c0             lda #$c0            :l->r kell
00213  1227 8d 33 12          sta mit             ;v=n=1
00214  122a 4c 79 11          jmp kar
              00
00215  122d             lbyte .byte 0             ;lathato keprol
              ff
00216  122e             rbyte .byte $ff           ;raktarbol
              31
00217  122f             kodok .byte $31           ;and (cc),y
              11
00218  1230                   .byte $11           ;ora (cc),y
              51
```

```
ne#  Addr Code        Source

0219 1231                    .byte $51          ;eor (cc),y
     91
0220 1232                    .byte $91          ;sta (cc),y
     ff
0221 1233             mit    .byte $ff          ;mit csinaljon?
0222 1234             ;n-bit=1:l->r kell
0223 1234             ;v-bit=1:r->l kell
     47 4c c1 47 4c cf 47 4c c5 47 cc 47 53 c1 47 53 cf 47 53 c5 47
     d3 47 c3 4b cc 4b d3 4b c3 00
0224 1234             ktabl .byte 'glAglOglEgLgsAgsOgsEgSgCkLkSkC',0
     f7 11
0225 1253             ctabl .word gla
     f7 11
0226 1255                    .word glo
     f7 11
0227 1257                    .word gle
     f7 11
0228 1259                    .word gl
     00 12
0229 125b                    .word gsa
     00 12
0230 125d                    .word  gso
     00 12
0231 125f                    .word  gse
     00 12
0232 1261                    .word  gs
     09 12
0233 1263                    .word  gc
     12 12
0234 1265                    .word  kl
     1b 12
0235 1267                    .word  ks
     24 12
0236 1269                    .word  kc

nd of assembly, error count = 00000

:1     11de    at2     11e4    chrget  0473    cscik   11b6
sere   11ac    ctabl   1253    fhtok   89d6    flist   1147
istv   030e    ftok    1133    ftokv   030c    fvegr   1153
vegrv  0310    gc      1209    ghossz  1fff    gl      11f7
a      11f7    glathe  2000    gle     11f7    glo     11f7
ra     1194    grakte  5000    gs      1200    gsa     1200
se     1200    gso     1200    helyf   1120    hossz   00e6
ar     1179    kc      1224    khossz  07ff    kl      1212
athe   0800    kodok   122f    krakte  4000    ks      121b
abl    1234    lath    00e0    lbyte   122d    lrnem   11cc
ok     9b9c    mit     1233    mod1    11c8    mod2    11d4
dflg   0075    new     8a7b    nfhtok  896c    rakt    00e2
yte    122e    ready   867e    rlnem   11d8    sima    1160
ken    8a07    txttab  002b    vftok   1144    vizsg   11ea
```

```
10 ifww=1thensys26214:new
20 print"ssSqq a program megszakitasbol folyamatosan"
30 print"q skalazik egy oktavot, majd ujra kezdi."
40 print"qq ez alkalmasint kisse idegesito is le-"
50 print"q het, ezert a 'c=' es a 'control' gom-"
60 print"q gombok lenyomasaval elhallgattathato,"
70 print"q de ugyanigy ujra is indithato."
80 print"qq reset utan hivasa: sys 26214"
90 print"qqq   ha mehet, nyomj meg egy
gombot!":getkeyx$:ww=1:load"skala.mod",8,1
```

```
Line#  Addr Code        Source

00001  0000             ;put"@0:skala
00002  0000             ;*********************************************
00003  0000             ;*                                           *
00004  0000             ;* a program megszakitasbol folyamatosan ska- *
00005  0000             ;* lazik egy oktavot, majd ujra kezdi.       *
00006  0000             ;* ez alkalmasint kisse idegesito is lehet,  *
00007  0000             ;* ezert a 'c=' es a "control' gombok lenyoma- *
00008  0000             ;* saval elhallgattathato, de ugyanigy ujra is *
00009  0000             ;* indithato.                                *
00010  0000             ;*                                           *
00011  0000             ;* hivasa: sys 26214                         *
00012  0000             ;*                                           *
00013  0000             ;*********************************************
00014  0000                    * = $6666
00015  6666             shflg    = $0543          ;c= es ctrl jelzoje
00016  6666             sdt      = $04fd          ;hang idotartama
00017  6666             mszv     = $0314          ;megsz. vektor
00018  6666             mszf     = $ce0e          ;megsz. folytatasa
00019  6666             tedsd    = $ff11          ;hang vezerlo regiszter
00020  6666             ;
00021  6666             ;
00022  6666 78                 sei
00023  6667 a9 80              lda #<megsz
00024  6669 8d 14 03           sta mszv            ;uj
00025  666c a9 66              lda #>megsz
00026  666e 8d 15 03           sta mszv+1          ;megszakitas
00027  6671 ad 11 ff           lda tedsd
00028  6674 09 08              ora #$08            ;maximalis
00029  6676 8d 11 ff           sta tedsd           ;hangero
00030  6679 a9 08              lda #$08            ;mutato kezd
00031  667b 8d e6 66           sta mut
00032  667e 58                 cli
00033  667f 60                 rts
00034  6680             ;
00035  6680             ;
00036  6680             megsz                       ;uj megszakitas
00037  6680 ad fd 04           lda sdt             ;ido lejart?
00038  6683 0d ff 04           ora sdt+2
00039  6686 c9 00              cmp #$00
00040  6688 d0 49              bne vege            ;meg nem, tovabb
00041  668a ce e6 66           dec mut             ;igen, uj hang
00042  668d 10 05              bpl ugr             ;van meg?
00043  668f a9 07              lda #$07            ;nincs,->eleje
00044  6691 8d e6 66           sta mut
00045  6694 ac e6 66     ugr   ldy mut             ;kov. hang
00046  6697 b9 d6 66           lda hanga,y         ;betoltese
00047  669a 8d 0f ff           sta $ff0f           ;also byte
00048  669d b9 de 66           lda hangf,y
00049  66a0 8d 10 ff           sta $ff10           ;felso byte
00050  66a3 a9 f8              lda #$f8            ;idozites
00051  66a5 8d fd 04           sta sdt             ;also byte
00052  66a8 a9 ff              lda #$ff
00053  66aa 8d ff 04           sta sdt+2           ;felso byte
00054  66ad 2c e7 66           bit eng             ;engedely?
00055  66b0 d0 08              bne vizsg           ;nincs, vizsg
00056  66b2 ad 11 ff           lda tedsd           ;hang
```

```
.....page # 2

Line#  Addr Code         Source


00057  66b5 09 20              ora #$20            ;bekapcsolasa
00058  66b7 8d 11 ff           sta tedsd           ;2. hanggen.
00059  66ba ad 43 05    vizsg  lda shflg
00060  66bd c9 06              cmp #$06            ;ctrl+c=?
00061  66bf d0 12              bne vege            ;nem, marad
00062  66c1 ad e7 66           lda eng             ;igen,
00063  66c4 49 ff              eor #$ff            ;az engedelyt
00064  66c6 8d e7 66           sta eng             ;atirni
00065  66c9 ad 11 ff           lda tedsd           ;2. hangg.
00066  66cc 49 ff              eor #$ff            ;engedelyt
00067  66ce 49 df              eor #$df            ;atirni
00068  66d0 8d 11 ff           sta tedsd
00069  66d3 4c 0e ce    vege    jmp mszf           ;tovabb
00070  66d6                    ;
00071  66d6                    ;
           2a 1e 02 e3 c0 ad 83 55
00072  66d6             hanga .byte $2a,$1e,$02,$e3;hangok also byte-ja
           03 03 03 02 02 02 02 02
00073  66de             hangf .byte $03,$03,$03,$02;hangok felso byte-ja
           00
00074  66e6             mut   .byte $00            ;hanyadik hang?
           00
00075  66e7             eng   .byte $00            ;00-hang,ff-csend

end of assembly, error count = 00000

eng     66e7    hanga   66d6    hangf   66de    megsz   6680
mszf    ce0e    mszv    0314    mut     66e6    sdt     04fd
shflg   0543    tedsd   ff11    ugr     6694    vege    66d3
vizsg   66ba
```

```
10 ifww=0thenww=1:load"hangseb.mod",8,1
20 poke1281,102:poke1282,102
30 print"ssSqq  a program modot ad a processzor ora-"
40 print"q  jele frekvenciajanak megmeresere,"
50 print"q  a hangsebesseg ket mikrofonnal "
60 print"q  torteno meresere, tovabba e meres"
70 print"q  (kozelito pontossagu) szimulalasara."
80 print"qq  a mereshez szukseges a 'technomir'"
90 print"q  csalad digitalis input ill. output"
100 print"q  modulja (a $fe18 cimen erheto el).qq"
110 gosub710
120 print"ssSqq az orajel frekvenciajat ismered (i/n)?"
130 getkeyx$:ifx$="i"then220
140 ifx$<>"n"then130
150 print"q akkor most megmerjuk!":print"q keszits elo egy
stoppert!"
160 print"qq azt az idot kell megmerned, amig":print"q a kep
sotet lesz!"
170 print"qq ha kesz, egyszerre nyomd meg az"
180 print"q inditogombot es a 'space' billentyut!"
190 getkeyx$:p=usr(2)
200 cf=134678271:print"q hany masodpercet mertel";:inputft
210 fr=cf/ft:print"q a frekvencia:";fr;"hertz":goto260
220 fr=1698000:print"q  1.698 megahertz (i/n)?"
230 getkeyx$:ifx$="i"then260
240 ifx$<>"n"then230
250 print"q  akkor ird be ide (hertzben)!           q
";:inputfr
260 fr=int(fr):print"ssSqq  tehat a frekvencia:";fr;"hz"
270 print"qqq  valassz:":print"qqq     r1R  hangsebesseg
merese"
280 print"qqq     r2R  a meres szimulalasa"
290 print"qqq     r3R  vege"
300 getkeyx$:ifx$<"1"or x$>"3"then300
310 onval(x$)goto330,550
320 print"q  akkor szevasz!":end
330 print"ssSqq   ird be a ket mikrofon
tavolsagat":print:input"  cm-ben:";ta
340 print"ssSqq a varakozasi ido"
350 print"q a hangszoro es az elso mikrofon ill."
360 print"q az elso es a masodik mikrofon kozti"
370 print"q tavolsag megtetelere":print
380 print"     ";1050368/fr;"sec."
390 print"q ebbol kifolyolag csak aq"
400 printta/1050368*fr/10;"m/s-ot meg-"
410 print"q halado sebessegek meresere alkalmas."
420 print"q megfelel (i/n)?"
430 getkeyx$:ifx$="i"then460
440 ifx$<>"n"then430
450 print"q akkor allitsd kisebbre a
tavolsagot!qq":gosub710:goto330
460 print"ssSqq most mar hozzakezdhetunk a mereshez!"
470 print"q mindent beallitottal (i/n)?"
480 getkeyx$:ifx$="i"then500
490 ifx$<>"n"then480
```

```
500 p=usr(0)
510 ifpeek(224)+peek(225)=0thenprint"q  rossz meres!qqq"
:goto540
520 print"q  a mert sebesseg:q"
530 printta/10/((4103*peek(224)+16*peek(225))/fr);"m/sqqq"
540 fori=1to2000:next:gosub710:goto260
550 print"ssSqq a varakozasi ido"
560 print"q az inditas es a 'j' gomb ill. a"
570 print"q 'j' es a 'k' gomb megnyomasa kozottq"
580 print"     ";984832/fr;"sec."
590 print"q ugy igyekezz, hogy ennel gyorsabb legy!"
600 print"q felkeszultel (i/n)?"
610 getkeyx$:ifx$="i"then630
620 ifx$<>"n"then610
630 print"ssSqq most mar hozzakezdhetunk!"
640 print"q eloszor a 'space'-szel indits, majd"
650 print"q gyorsan nyomd meg a 'j'-t es a 'k'-t!":getkeyx$
660 p=usr(1)
670 ifpeek(224)+peek(225)=0thenprint"q  rosszul
csinaltad!qqq":goto700
680 print"q  a mert ido:q"
690 print(3847*peek(224)+15*peek(225))/fr;"secqqq"
700 fori=1to2000:next:gosub710:goto260
710 print"    ha mehet, nyomj meg egy gombot!":poke239,0
:getkeyx$:return
```

```
Line#  Addr Code          Source

00001  0000              ;put"@0:hangseb"
00002  0000              ;***********************************************
00003  0000              ;*                                             *
00004  0000              ;* a program modot ad a processzor orajele     *
00005  0000              ;* frekvenciajanak megmeresere, a hangsebesseg *
00006  0000              ;* ket mikrofonnal torteno meresere, tovabba   *
00007  0000              ;* e meres (kozelito pontossagu) szimulalasara *
00008  0000              ;*                                             *
00009  0000              ;* a mereshez szukseges a technomir csalad     *
00010  0000              ;* digitalis input ill. output modulja         *
00011  0000              ;*                                             *
00012  0000              ;* a funkciok hivasa:                          *
00013  0000              ;*                                             *
00014  0000              ;* p=usr(2) orajel merese, ellenorzese         *
00015  0000              ;* p=usr(1) a meres szimulalasa                *
00016  0000              ;* p=usr(0) hangsebesseg merese                *
00017  0000              ;*                                             *
00018  0000              ;***********************************************
00019  0000                    *= $6666
00020  6666              t       = $e0          ;szamlalo helye
00021  6666              ted1    = $ff06        ;keptiltashoz
00022  6666              port    = $fe18        ;kapu
00023  6666              primm   = $ff4f        ;szovegkiiras
00024  6666              facint  = $a327        ;fac->egesz
00025  6666              illq    = $991c        ;ki:'ill. q. error'
00026  6666              param   = $65          ;fac parameter resze
00027  6666              keyflg  = $c6          ;billentyu figyelese
00028  6666              j       = $22          ;'j' le van nyomva
00029  6666              k       = $25          ;'k' le van nyomva
00030  6666              ;
00031  6666              ;
00032  6666 20 27 a3           jsr facint       ;par.->egesz
00033  6669 a6 65              ldx param        ;valasztas
00034  666b 30 04              bmi hiba         ;<0, hiba
00035  666d e0 03              cpx #$03         ;>=3?
00036  666f 30 03              bmi mehet        ;nem, mehet
00037  6671 4c 1c 99     hiba  jmp illq         ;igen, ki:'ill.'
00038  6674 e0 01        mehet cpx #$01         ;szimulacio?
00039  6676 f0 48              beq szimu        ;igen
00040  6678 e0 02              cpx #$02         ;orajel?
00041  667a f0 7f              beq freki        ;igen
00042  667c              ;
00043  667c              ;
00044  667c              hseb                   ;hangsebesseg merese
00045  667c 78                 sei              ;megszakitas
00046  667d ad 06 ff           lda ted1         ;es kep
00047  6680 29 ef              and #$ef         ;tiltasa
00048  6682 8d 06 ff           sta ted1
00049  6685 a9 01              lda #$01         ;indul
00050  6687 8d 18 fe           sta port
00051  668a a9 00              lda #0           ;szamlalo
00052  668c 85 e1              sta t+1          ;nullazasa
00053  668e 85 e0              sta t            ;varakozashoz
00054  6690 ad 18 fe     varak lda port         ;1. mikr.-ra var
00055  6693 29 80              and #$80         ;a 7. biten
00056  6695 f0 0a              beq jon          ;jon, tovabb
```

```
Line#  Addr Code        Source

00057  6697 e6 e1             inc t+1
00058  6699 d0 f5             bne varak           ;kulonben
00059  669b e6 e0             inc t               ;szamlal
00060  669d d0 f1             bne varak
00061  669f f0 15             beq kesz            ;nem jott, rossz merc
00062  66a1 a9 00       jon   lda #$00            :szamlalo
00063  66a3 85 e0             sta t               ;nullazasa
00064  66a5 85 e1             sta t+1             ;a mereshez
00065  66a7 ad 18 fe    meres lda port            ;2. mikr-ra var
00066  66aa 29 40             and #$40            ;a 6. biten
00067  66ac f0 08             beq kesz            ;jon, kesz
00068  66ae e6 e1             inc t+1
00069  66b0 d0 f5             bne meres           ;kulonben
00070  66b2 e6 e0             inc t               ;szamlal
00071  66b4 d0 f1             bne meres
00072  66b6 ad 06 ff    kesz  lda ted1
00073  66b9 09 10             ora #$10            ;kep
00074  66bb 8d 06 ff          sta ted1            ;es
00075  66be 58                cli                 ;megszakitas
00076  66bf 60                rts                 ;mehet
00077  66c0             ;a mert idotartam:
00078  66c0             ;4103*(t)+16*(t+1) oraciklus
00079  66c0             ;a tetlen varakozas idotartama:
00080  66c0             ;1050368 oraciklus
00081  66c0             ;
00082  66c0             ;
00083  66c0             szimu                       ;szimulacio
00084  66c0 ad 06 ff          lda ted1              ;kep
00085  66c3 29 ef             and #$ef              ;tiltasa
00086  66c5 8d 06 ff          sta ted1
00087  66c8 a9 00             lda #0                ;szamlalok
00088  66ca 85 e1             sta t+1               ;nullazasa
00089  66cc 85 e0             sta t                 ;varakozashoz
00090  66ce a5 c6       szvar lda keyflg            ;j-re
00091  66d0 c9 22             cmp #j                ;varakozik
00092  66d2 f0 0a             beq szjon             ;jon, tovabb
00093  66d4 e6 e1             inc t+1
00094  66d6 d0 f6             bne szvar             ;kulonben
00095  66d8 e6 e0             inc t                 ;szamlal
00096  66da d0 f2             bne szvar
00097  66dc f0 14             beq szkesz            ;nem jott, rossz
00098  66de a9 00       szjon lda #$00              :szamlalo
00099  66e0 85 e0             sta t                 :nullazasa
00100  66e2 85 e1             sta t+1               ;a mereshez
00101  66e4 a5 c6       szmer lda keyflg            :k-ra
00102  66e6 c9 25             cmp #k                :varakozik
00103  66e8 f0 08             beq szkesz            ;jon, tovabb
00104  66ea e6 e1             inc t+1
00105  66ec d0 f6             bne szmer             :kulonben
00106  66ee e6 e0             inc t                 ;szamlal
00107  66f0 d0 f2             bne szmer
00108  66f2 ad 06 ff    szkesz lda ted1
00109  66f5 09 10             ora #$10              ;kep
00110  66f7 8d 06 ff          sta ted1              ;engedelyezese
00111  66fa 60                rts
00112  66fb             ;a mert idotartam:
```

```
.....page # 3

Line#  Addr Code         Source


00113  66fb              ;3847*(t)+16*(t+1) oraciklus
00114  66fb              ;!! az aktiv megszakitas miatt !!
00115  66fb              ;!! csak kozelito erteku       !!
00116  66fb              ;a tetlen varakozas idotartama:
00117  66fb              ;984832 oraciklus
00118  66fb              ;
00119  66fb              ;
00120  66fb              freki                          ;a beallitashoz
00121  66fb 78                   sei                    ;2
00122  66fc ad 06 ff             lda ted1               ;4
00123  66ff 29 ef                and #$ef               ;2
00124  6701 8d 06 ff             sta ted1               ;4
00125  6704 a9 00                lda #0                 ;2
00126  6706 85 e2                sta t+2                ;3
00127  6708 85 e1                sta t+1                ;3
00128  670a 85 e0                sta t                  ;3
00129  670c e6 e2        ciklus  inc t+2                ;5
00130  670e d0 fc                bne ciklus             ;3-2
00131  6710 e6 e1                inc t+1                ;5
00132  6712 d0 f8                bne ciklus             ;3-2
00133  6714 e6 e0                inc t                  ;5
00134  6716 d0 f4                bne ciklus             ;3-2
00135  6718 ad 06 ff             lda ted1               ;4
00136  671b 09 10                ora #$10               ;2
00137  671d 8d 06 ff             sta ted1               ;4
00138  6720 58                   cli                    ;2
00139  6721 60                   rts                    ;6
00140  6722              ;134 678 271
00141  6722 .            ;gepi ciklus
00142  6722              ;79.3 sec alatt:
00143  6722              ;1.698 MHz orajel
00144  6722              ;
00145  6722              ;

end of assembly, error count = 00000

ciklus  670c    facint  a327    freki   66fb    hiba    6671
hseb    667c    illq    991c    j       0022    jon     66a1
k       0025    kesz    66b6    keyflg  00c6    mehet   6674
meres   66a7    param   0065    port    fe18    primm   ff4f
szimu   66c0    szjon   66de    szkesz  66f2    szmer   66e4
szvar   66ce    t       00e0    ted1    ff06    varak   6690
```

```
10 ifww=1then80
20 print"ssSqq   a program megszakitasbol mozgat":print"q
nehany karaktert.
30 print"q   ezeket a program lefutasa utan is":print"q
korlatozas nelkul hasznalhatod."
40 print"q   reset utan a":print"qq                sys 26214"
50 print"qq   utasitassal ujra 'aktivizalhatod'":print"q
oket.
60 print"qqq  ha mehet, nyomj meg egy billentyut!"
70 ww=1:load"pulzal.mod",8,1
80 getkeyx$:sys26214
90 print"Sq                  felnyil:":print
100 print" ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑"
110 print"q                  balnyil:":print
120 print" ←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←"
130 print"q                   kukac:":print
140 print" @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"
150 print"q                  zarojel:":print
160 print" ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]"
170 print"q              shift-space:":print
180 print"                                              "
190 print"q  ha eleg, nyomj meg egy billentyut!":getkeyx$
```

```
Line#  Addr Code          Source

00001  0000              ;put"@0:pulzal"
00002  0000              ;*********************************************
00003  0000              ;*                                           *
00004  0000              ;* a program megszakitasbol mozgat nehany    *
00005  0000              ;* karaktert: a felnyil felfele,             *
00006  0000              ;* a balnyil lefele, a kukac balra,          *
00007  0000              ;* a szogletes zaro zarojel jobbra forgo     *
00008  0000              ;* hullamvonalnak, a shift-space pedig       *
00009  0000              ;* pulzalo csillagnak latszik a kepen.       *
00010  0000              ;*                                           *
00011  0000              ;* hivasa: sys 26214                         *
00012  0000              ;*                                           *
00013  0000              ;*********************************************
00014  0000                      * = $6666
00015  6666              mszf    = $ce0e           ;megsz. folyt.
00016  6666              mszv    = $0314           ;megsz. vektor
00017  6666              escr    = $de48           ;'esc r' funkcio
00018  6666              bsout   = $ffd2           ;akku kiirasa
00019  6666              jfsar   = $0c27           ;jobb felso sarok
00020  6666              zp      = $e0             ;mutato a nullalapon
00021  6666              ;
00022  6666              ;
00023  6666 78                    sei
00024  6667 a9 f3                 lda #<megsz       ;megszak.
00025  6669 8d 14 03              sta mszv          ;vektor
00026  666c a9 66                 lda #>megsz       ;atirasa
00027  666e 8d 15 03              sta mszv+1
00028  6671 58                    cli
00029  6672 a2 00                 ldx #0            ;kar. gen.
00030  6674 bd 00 d0     kcik     lda $d000,x       ;masolasa
00031  6677 9d 00 70              sta $7000,x       ;rom-bol
00032  667a bd 00 d1              lda $d100,x       ;ram-ba
00033  667d 9d 00 71              sta $7100,x
00034  6680 bd 00 d2              lda $d200,x       ;$7000-tol
00035  6683 9d 00 72              sta $7200,x
00036  6686 bd 00 d3              lda $d300,x
00037  6689 9d 00 73              sta $7300,x
00038  668c e8                    inx
00039  668d d0 e5                 bne kcik
00040  668f              ;karakterek atirasa
00041  668f a2 07                 ldx #$07
00042  6691 bd 70 67     karcik   lda fekvo,x       ;kukac es
00043  6694 9d e8 70              sta $70e8,x       ;zarojel
00044  6697 9d 00 70              sta $7000,x       ;helyett
00045  669a bd 78 67              lda allo,x        ;nyilak
00046  669d 9d f0 70              sta $70f0,x       ;helyett
00047  66a0 9d f8 70              sta $70f8,x
00048  66a3 ca                    dex               ;hullamvonal
00049  66a4 10 eb                 bpl karcik
00050  66a6 20 48 de              jsr escr          ;kiskepernyo
00051  66a9              ;keret bal es jobb szele
00052  66a9 a2 18                 ldx #$18          ;sorok szama
00053  66ab a9 27                 lda #<jfsar
00054  66ad 85 e0                 sta zp            ;jobb felso
00055  66af a9 0c                 lda #>jfsar       ;saroktol
00056  66b1 85 e1                 sta zp+1
```

```
Line#  Addr Code        Source

00057  66b3 a0 00      szcikl ldy #$00
00058  66b5 a9 1e             lda #$1e            ;felnyil
00059  66b7 91 e0             sta (zp),y          ;jobb szelre
00060  66b9 c8               iny
00061  66ba a9 1f             lda #$1f            ;balnyil
00062  66bc 91 e0             sta (zp),y          ;bal szelre
00063  66be 18               clc
00064  66bf a5 e0             lda zp
00065  66c1 69 28             adc #$28            ;zp:=
00066  66c3 85 e0             sta zp              ;zp+40
00067  66c5 a5 e1             lda zp+1            ;egy sorral
00068  66c7 69 00             adc #$00            ;lejjebb
00069  66c9 85 e1             sta zp+1
00070  66cb ca               dex
00071  66cc d0 e5             bne szcikl
00072  66ce            ;keret also es felso szele
00073  66ce a9 00             lda #$00
00074  66d0 a2 27             ldx #$27            ;sorhossz
00075  66d2 a9 00      kepcik lda #$00            ;feltoltese
00076  66d4 9d 00 0c          sta $0c00,x         ;kukac
00077  66d7 a9 1d             lda #$1d            ;feiulre
00078  66d9 9d c0 0f          sta $0fc0,x         ;zarojel
00079  66dc ca               dex                 ;alulra
00080  66dd 10 f3             bpl kepcik
00081  66df a9 60             lda #$60            ;sh+space
00082  66e1 8d 00 0c          sta $0c00           ;bf
00083  66e4 8d 27 0c          sta $0c27           ;jf
00084  66e7 8d c0 0f          sta $0fc0           ;ba
00085  66ea 8d e7 0f          sta $0fe7           ;ja sarok
00086  66ed a9 93             lda #$93            ;keptorles
00087  66ef 20 d2 ff          jsr bsout
00088  66f2 60               rts
00089  66f3               ;
00090  66f3               ;
00091  66f3               megsz                   ;uj megsz.
00092  66f3 a9 c0             lda #$c0            ;kar. gen.
00093  66f5 8d 12 ff          sta $ff12           ;ram-ban
00094  66f8 a9 70             lda #$70            ;$7000-tol
00095  66fa 8d 13 ff          sta $ff13
00096  66fd ee a9 67          inc szaml
00097  6700 ad a9 67          lda szaml           ;lassitashoz
00098  6703 29 01             and #$01
00099  6705 d0 66             bne vege            ;forgataskor
00100  6707 ad a9 67          lda szaml
00101  670a 29 1e             and #$1e            ;pulzalaskor
00102  670c d0 1a             bne fforg
00103  670e            ;pulzalas
00104  670e ac a8 67          ldy kor             ;pulzalas
00105  6711 c0 08             cpy #$08            ;korbement?
00106  6713 d0 02             bne tolt            ;nem, tovabb
00107  6715 a0 00             ldy #$00            ;igen, ujra
00108  6717 b9 80 67   tolt   lda mut,y
00109  671a 8d 4f 67          sta vcikl+1         ;az akt.
00110  671d c8               iny                 ;resz
00111  671e b9 80 67          lda mut,y           ;betoltese
00112  6721 8d 50 67          sta vcikl+2         ;pulz.-hoz
```

```
Line#  Addr Code        Source

00113  6724 c8                 iny
00114  6725 8c a8 67           sty kor
00115  6728              fforg                       ;fugg. forgatasok
00116  6728 a2 06               ldx #$06
00117  672a a0 00               ldy #$00            ;mentes:
00118  672c ad f0 70            lda $70f0           ;felnyil
00119  672f 48                  pha                 ;eleje es
00120  6730 ad ff 70            lda $70ff           ;balnyil
00121  6733 48                  pha                 ;vege
00122  6734 bd f8 70     fcikl  lda $70f8,x         ;balnyilban
00123  6737 9d f9 70            sta $70f9,x         ;lefele
00124  673a b9 f1 70            lda $70f1,y         ;felnyilban
00125  673d 99 f0 70            sta $70f0,y         ;felfele
00126  6740 c8                  iny
00127  6741 ca                  dex
00128  6742 10 f0               bpl fcikl
00129  6744 68                  pla                 ;vissza:
00130  6745 8d f8 70            sta $70f8           ;felnyil vege
00131  6748 68                  pla                 ;es
00132  6749 8d f7 70            sta $70f7           :balnvil eleje
00133  674c              vforg                       ;vizsz. forgataso
00134  674c a2 07               ldx #$07
00135  674e bd 77 77     vcikl  lda $7777,x         ;attoltese
00136  6751 9d 00 73            sta $7300,x         ;sh+space-be
00137  6754 3e 00 70            rol $7000,x
00138  6757 08                  php                 :ill.
00139  6758 5e 00 70            lsr $7000,x         ;rotalasa
00140  675b 28                  plp                 ;kukac-ban
00141  675c 3e 00 70            rol $7000,x         ;balra
00142  675f 7e e8 70            ror $70e8,x
00143  6762 08                  php                 ;es
00144  6763 1e e8 70            asl $70e8,x
00145  6766 28                  plp                 ;zarojel-ben
00146  6767 7e e8 70            ror $70e8,x         ;jobbra
00147  676a ca                  dex
00148  676b 10 e1               bpl vcikl
00149  676d 4c 0e ce     vege   jmp mszf            ;megsz. folyt.
       00 81 42 42 24 24 18 00
00150  6770              fekvo  .byte $00,$81,$42,$42,$24,$24,$18,$00
       40 30 0c 02 02 0c 30 40
00151  6778              allo   .byte $40,$30,$0c,$02,$02,$0c,$30,$40
       88 67 90 67 98 67 a0 67
00152  6780              mut    .word m1,m2,m3,m4
       00 00 00 00 00 00 00 00
00153  6788              m1     .byte $00,$00,$00,$00,$00,$00,$00,$00
       00 00 00 18 18 00 00 00
00154  6790              m2     .byte $00,$00,$00,$18,$18,$00,$00,$00
       00 00 24 18 18 24 00 00
00155  6798              m3     .byte $00,$00,$24,$18,$18,$24,$00,$00
       00 42 24 18 18 24 42 00
00156  67a0              m4     .byte $00,$42,$24,$18,$18,$24,$42,$00
       00
00157  67a8              kor    .byte $00
       00
00158  67a9              szaml  .byte $00
```

```
.....page # 4

Line#  Addr Code         Source


end of assembly, error count = 00000

allo    6778    bsout   ffd2    escr    de48    fcikl   6734
fekvo   6770    fforg   6728    jfsar   0c27    karcik  6691
kcik    6674    kepcik  66d2    kor     67a8    mi      6788
m2      6790    m3      6798    m4      67a0    megsz   66f3
mszf    ce0e    mszv    0314    mut     6780    szaml   67a9
szcikl  66b3    tolt    6717    vcikl   674e    vege    676d
vforg   674c    zp      00e0
```

```
10 onwwgoto30,40:ifpeek(1383)<>68then60
20 print"ssS":char,16,8,"turelem!":ww=1:load"karakterek",8,1
30 char,11,12,"mar nem tart sokaig!":ww=2: load"sorok.mod",
8,1: getkeyx$
40 scnclr:fori=1to24:print"        abcdefghijklmnopqrstuvwxyz
     ";:next
50 print"        abcdefghijklmnopqrstuvwxyz";:sys4992:
getkeyx$:end
60 key1,"dL"+chr$(34)+"hy*"+chr$(13)+"rU"+chr$(13)+"dL"+
chr$(34)+"bas.sorok"+chr$(13)+"rU"+chr$(13)
70 print"ssSqq]]a program a rasztermegszakitas segit-"
80 print"q]]segevel ugyanazt a szoveget minden"
90 print"q]]sorban mas es mas karaktergenerator":print"q
felhasznalasaval jeleniti meg."
100 print"qq]]a karakterkeszletek terjedelme 105":print"q
blokk, toltesuk hosszadalmas.qq"
110 ifpeek(815)=6thenprint"q]]ha mehet, nyomd meg a
'space'-t! ":goto130
120 print"q]]ha mehet, nyomd meg az 'f1'-et!"
130 ifpeek(198)=4thenstop
140 ifpeek(198)=60then20:else130
```

```
....page # 1

ine#  Addr Code        Source

0001   0000            ;put"@0:sorok
0002   0000            ;*********************************************
0003   0000            ;*                                           *
0004   0000            ;*  soronkent mas-mas karakterkeszlet meg-   *
0005   0000            ;*  jelentetese a $1400-$7c00-ig terjedo     *
0006   0000            ;*  tartomanybol, $0400-asaval               *
0007   0000            ;*                                           *
0008   0000            ;*  hivasa: sys4992                          *
0009   0000            ;*                                           *
0010   0000            ;*********************************************
0011   0000                    * = $1380
0012   1380            mszv    = $314
0013   1380            zpmut   = $e7
0014   1380            hang    = $cecd         ;hang kezelese
0015   1380            bill    = $db11         ;billentyuzet kezelese
0016   1380            magno   = $cfbf         ;magno kezelese
0017   1380            ;
0018   1380            ;
0019   1380 78                 sei
0020   1381 a9 8d              lda #<megsz     ;uj
0021   1383 8d 14 03           sta mszv        ;megszak.
0022   1386 a9 13              lda #>megsz     ;beall.
0023   1388 8d 15 03           sta mszv+1
0024   138b 58                 cli
0025   138c 60                 rts
0026   138d            ;
0027   138d            ;
0028   138d ad 09 ff   megsz   lda $ff09
0029   1390 8d 09 ff           sta $ff09
0030   1393 a9 c0              lda #$c0        ;karakterkeszlet
0031   1395 8d 12 ff           sta $ff12       ;ram-ban
0032   1398 a6 e7              ldx zpmut
0033   139a bd bb 13           lda tff0b,x     ;kov. raszterm. sora
0034   139d 8d 0b ff           sta $ff0b
0035   13a0 bd d5 13           lda tff13,x     ;kar. keszlet helye
0036   13a3 8d 13 ff           sta $ff13
0037   13a6 e8                 inx
0038   13a7 e0 1a              cpx #$1a
0039   13a9 d0 0b              bne ugr01
0040   13ab 20 cd ce           jsr hang        ;hang
0041   13ae 20 bf cf           jsr magno       ;magno
0042   13b1 20 11 db           jsr bill        ;bill.
0043   13b4 a2 00              ldx #$00        ;nullazas
0044   13b6 86 e7      ugr01   stx zpmut
0045   13b8 4c c3 fc           jmp $fcc3       ;megsz. folyt.
0046   13bb            ;
0047   13bb            ;
               0a 12 1a 22 2a 32 3a 42 4a 52 5a
0048   13bb                    tff0b .byte $0a,$12,$1a,$22,$2a,$32,$3a,$42,$4a,$
               62 6a 72 7a 82 8a 92 9a a2 aa b2 ba c2 cc 02
0049   13c6                    .byte $62,$6a,$72,$7a,$82,$8a,$92,$9a,$a2,$
$ba,$c2,$cc,$02
               14 18 1c 20 24 28 2c 30 34 38 3c 40
0050   13d5                    tff13 .byte $14,$18,$1c,$20,$24,$28,$2c,$30,$34,$
$40
               44 48 4c 50 54 58 5c 60 64 68 6c 70 74 78
0051   13e1                    .byte $44,$48,$4c,$50,$54,$58,$5c,$60,$64,$
$70,$74,$78
```

```
.....page # 2

Line#  Addr Code          Source


end of assembly, error count = 00000

bill     db11      hang     cecd      magno    cfbf      megsz    138d
mszv     0314      tff0b    13bb      tff13    13d5      ugr01    13b6
zpmut    00e7
```

```
10 ifww=1then150
20 print"ssSq  a program a gep memoriajaba nyujt":print"q
bepillantast a megszakitas segitse-"
30 print"q  gevel. a kepernyo felso reszere a":print"q
0-4799 cimu memoriarekeszek 'kepet'"
40 print"q  vetiti ki. a vetites a grafikus":print"q
kepernyo felepitesenek megfeleloen,"
50 print"q  karakterhelyenkent s azon belul":print"q
soronkent tortenik, ami nemikeppen"
60 print"q  neheziti a tajekozodast, de pontos":print"q
ertekekre ugysem lesz szuksegunk,"
70 print"q  a lenyeg enelkul is latszik."
80 gosub760
90 print"ssSq  a programbol is szemleltetheto":print"q
jelensegeket gombnyomasra lathatod,"
100 print"q  ami programbol nem megy, azokra":print"q  nehany
tippet adok, de leginkabb az"
110 print"q  onallo kiserletezessel tudod a":print"q  gepet
megismerni."
120 print"q  ha menet kozben a program leall,":print"q  a
'help' billentyuvel folytathatod!":print:print
130 key8,"cont"+chr$(13)
140 gosub760:print"ssS":ww=1:load"gvetites.mod",8,1
150 sys26214
160 print"Sq     borzalmas egy latvany, nem?!qq":gosub760
170 print"Sqsoron mindig karaktersort fogunk
erteni!qq":gosub760
180 print"Sqa legfeltunobb talan az a ket resz, ame-lyik
akkor is vibral, ha nem is nyulsz"
190 print"semmihez: az egyik az 1. sor kozepen, a masik a 7.
sor bal oldalan. mindketto"
200 print"az oraval kapcsolatos: az elso a szam-  lalo, a
masodik egy segedszamlalo."
210 print"          tovabb: 'help'";
220 stop
230 print"Sqmost piszkosul megindult valami a maso- dik sor
kozepen, es ez a program futa-"
240 print"sa kozben mindig ilyesmit csinal, mert  ez a gepi
kodu program altal hasznalt"
250 print"un. 'rendszer-verem', amely a $123-$1ff teruleten
helyezkedik el."
260 gosub760
270 print"Sqa program futasa utan irj egy kis gepi  kodu
rutint, amely - nem tul sokszor,"
280 print"mondjuk max. 128-szor - elteszi az akku-mulator
tartalmat a verembe (a pha uta-"
290 print"sitas segitsegevel), kozben figyeld a  kepet es a
monitor veremmutato-erteket!"
300 gosub760
310 print"Sqmost en nem piszkalok bele, de azert egykicsit
megvillogtatom ezt a reszt, hogy"
320 print"biztosan tudd, hol van a verem, de nem  megyek a
vegeig, mert abbol gubanc lesz!q"
330 gosub760:print"qq      ha eleg, akkor meg
egyszer!":k=dec("1c8"):l=15
```

```
340 i=dec("123"):dowhilei<k:pokei,l:getx$:ifx$<>""thenexit
:elsei=i+1:loop
350 ifx$=""thenl=255-l:goto340
360 print"Sqa rendszer-verem nem tevesztendo ossze  a
basic-veremmel, amely a basic progra-"
370 print"mok szubrutinjainak]visszateresi cimeit tarolja. ez
a $6ec-$7af teruleten van,"
380 print"a 6. sor kozepetol latszik, bo fel sor.":gosub760
390 print"Seloszor kitakaritom, de csak a nagyjat, mert itt
sem akarok bajt csinalni."
400 fori=dec("6ec")todec("790"):pokei,0:next:gosub760
410 print"Smost elinditok 35 egymasba agyazott
szubrutint, figyelj jol!":gosub760:i=0
420 i=i+1:ifi>35then430:elsegosub420
430 i=i-1:ifi=0then440:elsereturn
440 print"Smegnezned meg egyszer (i/n)?              ha
allandoan nyomod az 'i'-t, szuper"
450 print"latvanyban lesz reszed!":getkeyx$:ifx$="i"then390
460 print"Sqa kovetkezo nagyobb terulet a kazetta-  es az
rs232 puffer, $333-$436-ig, a 3."
470 print"sor kozepetol egy sor hosszan.           ezt is
villogtatom egy kicsit."
480 print"(kivalo terulet gepi kodu rutinoknak!)"
490 gosub760:print"qq      ha eleg, akkor meg
egyszer!":k=dec("472"):l=15
500 i=dec("333"):dowhilei<k:pokei,l:getx$:ifx$<>""thenexit
:elsei=i+1:loop
510 ifx$=""thenl=255-l:goto500
520 print"Sqerdemes meg a funkciobillentyuk terule- tet
megnezni ($55f-5e6, az 5. sorban)."
530 gosub760
540 print"Sa 'help' kivetelevel most mindnek 16 db $ff kodu
karaktert ( )."":gosub760
550 fori=1to7:keyi,"                    ":next
560 print"Sa 'help' kivetelevel most mindnek 16 db 'space'
karaktert adunk.":gosub760
570 fori=1to7:keyi,"                    ":next
580 print"Smegnezed meg (i/n)?":getkeyx$:ifx$="i"then540
590 print"Sprobalgasd egy kicsit, adj az egyes
billentyuknek hosszabb-rovidebb erteke-"
600 print"ket, de vigyazz a 'help'-re, mert ezzel tudsz
visszajonni!":stop:gosub770
610 print"Sqprogrambol mar csak a szin- ($800-$be7) es a
kepmemoriat ($c00-$fe7) tudom meg-"
620 print"mutatni. a szinmemoria a 7. sor kozepe- tol, a
kepmemoria a 10. sor kozepetol"
630 print"lathato.":gosub760
640 print"Sa szinmemoriaban valtozast pld.valame-  lyik
szinbillentyuvel idezhetunk elo,"
650 print"de csak az adott karakterhelynek   meg- feleloen.
az egeszet egy torlessel at"
660 print"tudjuk allitani, bar elsore csak az ak- tualis
ablak szinet valtoztatja meg."
670 gosub760
680 print"Sa kepmemoria, a definialt ablakkal onma-gaert
beszel. probalgasd egy kicsit!":stop
```

```
690 gosub770
700 print"Saz ez utan lathato terulet mar a basic  program
ill. a valtozok terulete,"
710 print"ide mar nem tudlak elkiserni, de ajan-  lom, hogy
figyeld meg a kovetkezoket:"
720 print" - tombok definialasa - ures hely;        - valtozok
ertekeinek modositasa;"
730 print" - basic sorok beirasa, javitasa stb.
jo szorakozast!"
740 print"q    ha mehet, nyomj meg egy gombot!";:getkeyx$
750 print"qqqqqq              szevasz! jo munkat!":end
760 print"q    ha mehet, nyomj meg egy
gombot!":getkeyx$:return
770 print"ssSqqqqqqqqqqqqqqq"+chr$(27)+"t":return
```

```
.....page # 1

Line#  Addr Code        Source

00001  0000             ;put"@0:gvetites"
00002  0000             ;************************************************
00003  0000             ;*                                              *
00004  0000             ;*  a program a kepernyo felso 15 soraba vetiti *
00005  0000             ;*  a memoria $0000-$12bf-ig terjedo reszet.    *
00006  0000             ;*                                              *
00007  0000             ;*  hivasa: sys 26214                           *
00008  0000             ;*                                              *
00009  0000             ;************************************************
00010  0000                     * = $6666
00011  6666             mszv    = $0314          ;megsz. vektor
00012  6666             zpmut   = $e7            ;ciklusmutato
00013  6666             esct    = $de5e          ;'esc t' funkcio
00014  6666             plot    = $fff0          ;kurzorbeallitas
00015  6666             ;
00016  6666             ;
00017  6666 78                  sei
00018  6667 a9 97               lda #<megsz
00019  6669 8d 14 03            sta mszv
00020  666c a9 66               lda #>megsz
00021  666e 8d 15 03            sta mszv+1
00022  6671 58                  cli
00023  6672 18                  clc
00024  6673 a2 0f               ldx #$0f         ;az ablak
00025  6675 a0 00               ldy #$00         ;bal felso
00026  6677 20 f0 ff            jsr plot         ;sarkanak
00027  667a 20 5e de            jsr esct         ;beallitasa
00028  667d a2 00               ldx #$00
00029  667f a9 f1               lda #$f1
00030  6681 9d 00 60     cik    sta $6000,x      ;szin- es
00031  6684 9d 00 61            sta $6100,x      ;fenyesseg-
00032  6687 9d 58 61            sta $6158,x      ;memoria
00033  668a 9d 00 64            sta $6400,x      ;inicializalasa
00034  668d 9d 00 65            sta $6500,x
00035  6690 9d 58 65            sta $6558,x
00036  6693 e8                  inx
00037  6694 d0 eb               bne cik
00038  6696 60                  rts
00039  6697             ;
00040  6697             ;
00041  6697 ad 09 ff     megsz   lda $ff09
00042  669a 8d 09 ff            sta $ff09
00043  669d a6 e7               ldx zpmut
00044  669f bd cc 66            lda tff06,x      ;bitterkep
00045  66a2 8d 06 ff            sta $ff06        ;ki-be
00046  66a5 bd ce 66            lda tff0b,x      ;kov. raszter-
00047  66a8 8d 0b ff            sta $ff0b        ;megsz. sora
00048  66ab bd d0 66            lda tff12,x      ;bitterkep
00049  66ae 8d 12 ff            sta $ff12        ;helye
00050  66b1 bd d2 66            lda tff14,x      ;szinmem.
00051  66b4 8d 14 ff            sta $ff14        ;helye
00052  66b7 e8                  inx
00053  66b8 e0 02               cpx #$02
00054  66ba d0 0b               bne ugrik
00055  66bc 20 cd ce            jsr $cecd        ;hang
00056  66bf 20 bf cf            jsr $cfbf        ;magno
```

```
.....page # 2

Line#  Addr Code         Source


00057  66c2 20 11 db            jsr $db11          ;bill.
00058  66c5 a2 00               ldx #$00           ;nullazas
00059  66c7 86 e7        ugrik  stx zpmut
00060  66c9 4c c3 fc            jmp $fcc3
00061  66cc               ;
00062  66cc               ;
00063  66cc               ;
00064  66cc               ;
            1b 3b
00065  66cc               tff06 .byte $1b,$3b
            00 79
00066  66ce               tff0b .byte $00,$79
            c4 00
00067  66d0               tff12 .byte $c4,$00
            08 60
00068  66d2               tff14 .byte $08,$60

end of assembly, error count = 00000

cik     6681     esct    de5e     megsz   6697     mszv    0314
plot    fff0     tff06   66cc     tff0b   66ce     tff12   66d0
tff14   66d2     ugrik   66c7     zpmut   00e7
```

```
10 ifww=1then140
20 print"ssSqq  a program a gep memoriajaba nyujt":print"q
bepillantast a megszakitas segitse-"
30 print"q  gevel. a kepernyo felso reszere a":print"q
memoria ket egymas utan kovetkezo"
40 print"q  lapjat (osszesen 512 byte-nyi teru-"
50 print"q  letet) vetiti ki, majd decimalisan"
60 print"q  es hexadecimalisan kiirja e resz":print"q
hatarait is.qq"
70 gosub1020
80 print"ssSqq  a programbol is szemleltetheto":print"q
jelensegeket gombnyomasra lathatod,"
90 print"q  ami programbol nem megy, azokra":print"q  nehany
tippet adok, de leginkabb az"
100 print"q  onallo kiserletezessel tudod a":print"q  gepet
megismerni."
110 print"q  ha menet kozben a program leall,":print"q  a
'help' billentyuvel folytathatod!":print:print
120 key8,"cont"+chr$(13)
130 gosub1020:print"ssS":ww=1:load"kvetites.mod",8,1
140 sys26214,0
150 print"Sq     borzalmas egy latvany, nem?!qq":gosub1020
160 print"Sqne felejtsd el, hogy a kukac kepernyo-  kodja 0,
ezert minden olyan helyen, ahol"
170 print"a tarban 0 van, kukacot latsz!":gosub1020
180 print"Sqa felso reszen, az 5. sor bal oldalan"
190 print"van egy resz, amelyik vibral, ez a harom"
200 print"byte-os ora ($a3-$a5), 1/60 sec-onkent"
210 print"szamlal a $a5-on levo szamlalo, a $a4-es"
220 print"256-szor lassabban, a $a3-as ujabb 256-"
230 print"szor lassabban, mig el nem eri a 24-et,"
240 print"majd mindharom byte-ot nullazza.":gosub1020
250 print"Sqmost atirjuk az orat, figyelj jol!"
260 print"q           ti$="+chr$(34)+"000000"+chr$(34)
:gosub1020:ti$="000000"
270 print"Sq most megint atirjuk!"
280 print" rovidesen lathatod a valtast!"
290 print"q           ti$="+chr$(34)+"235958"+chr$(34):
gosub1020:ti$="235958"
300 ifti$>"111111"then300
310 print"q ket masodperc mulva kellett volna, de   joval
kesobb tortent! tudod-e, miert?"
320 print"q ismeteljuk (i/n)?":getkeyx$:ifx$="i"then250
:elseifx$<>"n"then320
330 print"Sqa legalso sor allandoan vibral, itt van a gepi
kodu programok altal hasznalt"
340 print"un. 'rendszer-verem', amely a $123-$1ff teruleten
helyezkedik el."
350 print"qa kepen ez a 7. sor kozepetol a vetitettkep vegeig
terjedo resz.":gosub1020
360 print"Sqa program futasa utan irj egy kis gepi  kodu
rutint, amely - nem tul sokszor,"
370 print"mondjuk max. 128-szor - elteszi az akku-mulator
tartalmat a verembe (a pha uta-"
```

```
380 print"sitas segitsegevel), kozben figyeld a   kepet es a
monitor veremmutato-erteket!"
390 gosub1020
400 print"Sqmost en nem piszkalok bele, de azert egykicsit
megvilloqtatom ezt a reszt, hogy"
410 print"biztosan tudd, hol van a verem, de nem  megyek a
vegeig, mert abbol gubanc lesz!q"
420 gosub1020:print"qq      ha eleg, akkor meg egyszer!"
:k=dec("1cf"):l=64
430 i=dec("123"):dowhilei<k:pokei,l:getx$:ifx$<>""
thenexit:elsei=i+1:loop
440 ifx$=""thenl=256-l:goto430
450 print"Sqmeg itt. a 0. lapon talalunk nehany     rendszer
valtozot, amit erdemes megnezni."
460 print"egy kicsit pihenek, te addig probald ki a
kovetkezoket! (vissza: 'help')"
470 gosub1020
480 print"S-kurzor cime a 6. sor elejen ($c8->)"
490 print"-stop gomb a 4. sor kozepen ($91)"
500 print"-bill. jelzo az 5. sor vegen ($c6)"
510 print"-fac = software akkumulator ($61-66)     a 3. sor
kozepen"
520 print"-a fac->ascii konverzio puffere ($100->) a 7. sor
kozepen";
530 stop:gosub1030
540 print"Sqterjunk at a kovetkezo erdekes reszre,  a
funkciobillentyuk teruletere!"
550 gosub1020
560 sys26214,dec("500")
570 print"Sqez a $55f-$5e6-ig terjedo resz, a kepen a 3. sor
kozepetol kezdodo bo 3 sor."
580 gosub1020
590 print"Sqa 'help' kivetelevel most mindnek 16 db $ff kodu
karaktert ( ) adunk ertekul."
600 gosub1020
610 fori=1to7:keyi,"                    ":next
620 print"Sqa 'help' kivetelevel most mindnek 16 db 'space'
karaktert adunk.":gosub1020
630 fori=1to7:keyi,"                    ":next
640 print"Sq  megnezed meg (i/n)?":getkeyx$:ifx$="i"then590
650 print"Sqprobalgasd egy kicsit, adj az egyes
billentyuknek hosszabb-rovidebb erteke-"
660 print"ket, de vigyazz a 'help'-re, mert ezzel tudsz
visszajonni!"
670 stop:gosub1030
680 print"Sq emeljuk feljebb a kepet, hogy az also   resz
folytatasat is lassuk!"
690 gosub1020:sys26214,dec("600")
700 print"Sq itt, a 6. sor vegetol lathato majdnem   5 sor
hosszusagban a basic-verem."
710 print" nem tevesztendo ossze az 1. lapon levo
rendszer-veremmel!":gosub1020
720 print"Sq a basic-verem a basic programok szub-
rutinjainak visszateresi cimeit es"
730 print" adatait tarolja, a $6ec-$7af teruleten
helyezkedik el."
```

```
740 gosub1020
750 print"Sq eloszor kitakaritom, de csak a nagyjat, mert itt
sem akarok bajt csinalni."
760 fori=dec("6ec")todec("790"):pokei,0:next:gosub1020
770 print"Sq most elinditok 35 egymasba agyazott
szubrutint, figyelj jol!":gosub1020:i=0
780 i=i+1:ifi>35then790:elsegosub780
790 i=i-1:ifi=0then800:elsereturn
800 print"Sq megnezned meg egyszer (i/n)?            ha
allandoan nyomod az 'i'-t, szuper"
810 print" latvanyban lesz reszed!":getkeyx$:ifx$="i"then760
820 sys26214,dec("8100")
830 print"Sq meg megmutatom a tarban a kulcsszavak
tablazatat, ez nem mozog, de azert"
840 print" erdemes megnezni, fokent kisbetu-nagy-  betu
modban.":gosub1020
850 sys26214,dec("ff00")
860 print"Sq befejezesul nezd meg a felso sorban a   ted chip
regisztereit!"
870 print"q (alul ismet a nullalap; miert?)qq":gosub1020
880 sys62158
890 print"ssSqq vege a kirandulasnak, de te meg tovabb"
900 print" szorakozhatsz! nehany javaslat:"
910 print"q - a program kiirtasa, uj programsorok
beirasa, javitas;"
920 print" - valtozok ertekei;"
930 print" - tombok definialasa, tombelemek erte-    kenek
modositasa;"
940 print" - az input puffer viselkedese;"
950 print" - a szin- es kepmemoria viselkedese;"
960 print" - a kazettapuffer vizsgalata (de ehhez    olyan
program kell, ami tolteskor nem"
970 print"   oltja ki a kepet, pld. her-turboval
kimentett programok);"
980 print" - a lemezpuffer viselkedese;"
990 print" - az i/o terulet figyelese stb."
1000 print"q    ha mehet, nyomj meg egy gombot!";:getkeyx$
1010 print"q                szevasz! jo munkat!":end
1020 print"q    ha mehet, nyomj meg egy
gombot!":getkeyx$:return
1030 print"ssSqqqqqqqqqqqqqq"+chr$(27)+"t":return
```

```
.....page # 1

Line#  Addr Code.        Source

00001  0000              ;put"@0:kvetites
00002  0000                    * = $6666
00003  6666              ;*************************************************
00004  6666              ;*                                               *
00005  6666              ;* a program a kepernyo felso 13 soraba vetiti   *
00006  6666              ;* a memoria cim-nel kezdodo ket teljes lapjat,  *
00007  6666              ;* majd decimalisan es hexadecimalisan is kiir-  *
00008  6666              ;* ja a tartomany hatarait.                      *
00009  6666              ;*                                               *
00010  6666              ;*     hivasa: sys 26214,cim                     *
00011  6666              ;*                                               *
00012  6666              ;* vigyazz! cim-be decimalis lapkezdetet irj!    *
00013  6666              ;*                                               *
00014  6666              ;*************************************************
00015  6666              mszv     = $314              ;megsz. vektor
00016  6666              linnum   = $14               ;cimmutato
00017  6666              cimbe    = $c38f             ;cim beolvasasa
00018  6666              cimkid   = $a45f             ;cim kiirasa dec. (x/a)
00019  6666              cimkih   = $faff             ;cim kiirasa hex. (a/x)
00020  6666              primm    = $ff4f             ;szoveg kiirasa
00021  6666              cr       = $fb3a             ;uj sor
00022  6666              esct     = $de5e             ;'esc t' funkcio
00023  6666              plot     = $fff0             ;kurzorbeallitas
00024  6666              ;
00025  6666              ;
00026  6666 20 8f c3           jsr cimbe             ;a kezdocim
00027  6669 a5 14              lda linnum            ;beolvasasa
00028  666b f0 13              beq beir              ;cim jo, mehet!
00029  666d 20 4f ff     hiba  jsr primm             ;kiirja
       4e 45 4d 20 4c 41 50 4b 45 5a 44 45 54 21 00
00030  6670                    .byte 'nem lapkezdet!',$00
00031  667f 60                 rts                   ;vissza
00032  6680 a6 15        beir  ldx linnum+1          ;cim hi-b.
00033  6682 8e f9 66           stx ciklus+2          ;vetiteshez
00034  6685 8e 0a 67           stx tol+1             ;kiirashoz
00035  6688 e8                 inx                   ;(cim+256) hi-b.
00036  6689 8e ff 66           stx ciklus+8          ;vetiteshez
00037  668c 8e 0c 67           stx ig+1              ;kiirashoz
00038  668f 18                 clc
00039  6690 a2 0d              ldx #$0d              ;az ablak
00040  6692 a0 00              ldy #$00              ;bal felso
00041  6694 20 f0 ff           jsr plot              ;sarkanak
00042  6697 20 5e de           jsr esct              ;beallitasa
00043  669a 20 4f ff           jsr primm             ;kiirja
       20 41 20 56 45 54 49 54 45 53 3a 20 00
00044  669d                    .byte ' a vetites: ',$00
00045  66aa ae 09 67           ldx tol               ;kezdocim
00046  66ad ad 0a 67           lda tol+1             ;kiirasa
00047  66b0 20 5f a4           jsr cimkid            ;decimalisan
00048  66b3 20 4f ff           jsr primm             ;kiirja
       2d 00
00049  66b6                    .byte '-',$00
00050  66b8 ae 0b 67           ldx ig                ;vegcim
00051  66bb ad 0c 67           lda ig+1              ;kiirasa
00052  66be 20 5f a4           jsr cimkid            ;decimalisan
00053  66c1 20 4f ff           jsr primm             ;kiirja
```

```
Line#  Addr Code        Source

           20 3d 20 24 00
00054  66c4                    .byte ' = $',$00
00055  66c9 ad 09 67             lda tol                ;kezdocim
00056  66cc ae 0a 67             ldx tol+1              ;kiirasa
00057  66cf 20 ff fa             jsr cimkih             ;hexaban
00058  66d2 20 4f ff             jsr primm              ;kiirja
           2d 20 24 00
00059  66d5                    .byte '- $',$00
00060  66d9 ad 0b 67             lda ig                 ;vegcim
00061  66dc ae 0c 67             ldx ig+1               ;kiirasa
00062  66df 20 ff fa             jsr cimkih             ;hexaban
00063  66e2 20 3a fb             jsr cr                 ;uj sor
00064  66e5 20 5e de             jsr esct               ;ablak bf sarka
00065  66e8 78                   sei
00066  66e9 a9 f5                lda #<megsz            ;uj
00067  66eb 8d 14 03             sta mszv               ;megszakitas
00068  66ee a9 66                lda #>megsz            ;beirasa
00069  66f0 8d 15 03             sta mszv+1
00070  66f3 58                   cli
00071  66f4 60                   rts
00072  66f5               ;
00073  66f5               ;
00074  66f5               megsz                        ;uj megszakitas
00075  66f5 a2 00                 ldx ##$00
00076  66f7 bd 00 02      ciklus lda $0200,x
00077  66fa 9d 00 0c             sta $0c00,x            ;ket
00078  66fd bd 00 03             lda $0300,x            ;lap
00079  6700 9d 00 0d             sta $0d00,x            ;vetitese
00080  6703 e8                   inx
00081  6704 d0 f1                bne ciklus
00082  6706 4c 0e ce             jmp $ce0e
           00 00
00083  6709               tol    .word $0000            ;a vet. resz kezd.
           ff ff
00084  670b               ig     .word $ffff            ;es vege

end of assembly, error count = 00000

beir    6680    ciklus  66f7    cimbe   c38f    cimkid  a45f
cimkih  faff    cr      fb3a    esct    de5e    hiba    666d
ig      670b    linnum  0014    megsz   66f5    mszv    0314
plot    fff0    primm   ff4f    tol     6709
```

```
10 onwwgoto50,220,380
20 print"ssSqqqq]]a program a fenyujsag megvalositasara"
30 print"q]]mutat harom lehetoseget.":gosub710
40 ww=1:load"feny1.mod",8,1
50 print"ssSqqqq]]a legegyszerubb formajaban egy gepi"
60 print"q]]kodu program a kepernyo also soraban"
70 print"q]]mozgat egy szoveget."
80 gosub470
90 print"qqq]]]leallitani ugy tudod, ha a "
100 print"q]]'control' es a 'c=' gombokat":print"q]]egyszerre
nyomod meg."
110 sys26214,s$
120 gosub710
130 gosub580
140 print"Sqq]]ennek a megoldasnak a kovetkezo"
150 print"q]]hatranyai vannak:"
160 print"qq]]- darabosan mozog, nem folyamatosan;"
170 print"q]]- sebessege nem szabalyozhato;"
180 print"q]]- a fenyujsag mukodese kozben"
190 print"q]]  a gep masra nem hasznalhato."
200 gosub710
210 ww=2:load"feny2.mod",8,1
220 print"ssSqqqq]]a masodik valtozatban ugyancsak"
230 print"q]]az also sorban mozog a szoveg,"
240 print"q]]de ezt a megszakitas mukodteti, igy"
250 print"q]]az elobbi hatranyok nem jelentkeznek."
260 gosub470
270 print"qq]]a 'control' es a 'c=' gombokkal"
280 print"q]]allithatod a sebesseget!"
290 sys26214,s$
300 gosub710
310 gosub580
320 print"Sqq]]ennek a megoldasnak mar csak"
330 print"q]]az a hatranya, hogy elvesz egy sort"
340 print"q]]a kepernyobol, amire esetleg mas"
350 print"q]]celbol szuksegunk lenne."
360 gosub710
370 sys62158:ww=3:load"feny3.mod",8,1
380 print"ssSqqqq]]a harmadik valtozatban mar a"
390 print"q]]kepernyo alatt mozog a szoveg."
400 gosub470
410 print"qq]]a 'control' es a 'c=' gombokkal"
420 print"q]]allithatod a sebesseget!"
430 sys26214,s$
440 gosub710
450 gosub580
460 end
470 s$="   *   "
480 s$=s$+"ez a program alkalmas egy max. 248 karakterbol
allo szoveg "
490 ifww<3thens$=s$+"utolso sorban valo ":goto510
500 s$=s$+"kepernyo alatti "
510 s$=s$+"megjelenitesere."
520 s$=s$+" az input puffer 88 karakternyi merete miatt a
hosszabb szoveg csak "
```

```
530 s$=s$+"reszletekben irhato be. a vesszok bajt "
540 s$=s$+"okozhatnak, legjobb, ha idezojeleket
hasznalsz!":return
550 sys26214,s$
560 print"Sqqq]]a control es a commodore gombokkal
allithatod a sebesseget!q"
570 gosub710
580 t$="   *  "
590 print"Sqq]]]ha mindent ertesz, te is irhatsz be"
600 print"q]]]uj szoveget!"
610 print:input"q]]]a szoveg:";a$
620 if len(a$)+len(t$)>254thenprint"tul hosszu!":goto610
630 t$=t$+" "+a$:print"Sq]az eddigi
szoveg:":print"q";t$:print
640 print"q]van meg (i/n)":getkeyx$:ifx$="i"then610
650 sys26214,t$
660 print"qq]rtRoldas":print"q]ruRj szoveg":print"q]rvRege"
670 getkeyx$:a$=""
680 ifx$="v"thenreturn
690 ifx$="u"thent$="   *  ":goto610
700 ifx$<>"t"then670:else610
710 print"qqqqq]]]]ha mehet, nyomj meg egy
gombot!";:getkeyxx$:return
```

```
Line#  Addr Code          Sour e

00001  0000              ;put"@0:feny1
00002  0000              ;*********************************************
00003  0000              ;*                                           *
00004  0000              ;*  az also sorban szakaszosan kiirja        *
00005  0000              ;*  az a$ szoveget (max. 255 karakter)       *
00006  0000              ;*                                           *
00007  0000              ;*  hivasa: sys 26214,a$                     *
00008  0000              ;*                                           *
00009  0000              ;*  kilepes: control es c= segitsegevel      *
00010  0000              ;*                                           *
00011  0000              ;*********************************************
00012  0000                     * = $6666
00013  6666              ram      = $ff3f          ;ram-ra kapcs.
00014  6666              rom      = $ff3e          ;rom-ra kapcs.
00015  6666              chkoma   = $9491          :vesszoig elore
00016  6666              strflg   = $0d            ;stringjelzo
00017  6666              frestr   = $9c48          ;szoveges valtozo beol
00018  6666              escb     = $de67          ;esc 'b' funkcio
00019  6666              kep      = $0fc0          ;also sor kezdete
00020  6666              shflg    = $543           ;c= es ctrl gombok
00021  6666              szh      = $df            ;szoveg hossza
00022  6666              szoveg   = $e0            ;szoveg helye
00023  6666              varptr   = $47            ;szovegvalt. leiroi
00024  6666              hang     = $cecd          ;hang kezelese
00025  6666              magno    = $cfbf          :magno kezelese
00026  6666              bill     = $db11          ;billentyuzet
00027  6666              ;
00028  6666              ;
00029  6666 20 91 94           jsr chkoma          ;vesszoig
00030  6669 a9 ff              lda #$ff            ;szovegvaltozo
00031  666b 85 0d              sta strflg          ;jon
00032  666d 20 48 9c           jsr frestr          ;szoveg be
00033  6670 a0 02              ldy #$02
00034  6672 b1 47       szolv  lda (varptr),y      ;leirok
00035  6674 99 df 00           sta szh,y           ;masolasa
00036  6677 88                 dey
00037  6678 10 f8              bpl szolv
00038  667a c6 df              dec szh             ;1 tobblet le
00039  667c a9 17       ablak  lda #$17            ;24. sor
00040  667e a2 27              ldx #$27            ;40. oszlop
00041  6680 20 67 de           jsr escb            ;jobb also sarok
00042  6683 a9 ff              lda #$ff            ;mutatok
00043  6685 8d c3 66           sta szmut
00044  6688              ;
00045  6688              ;
00046  6688              ciklus                    :mozgatas
00047  6688 a2 00              ldx #$00
00048  668a bd c1 0f    eltol  lda kep+1,x         ;eltolas
00049  668d 9d c0 0f           sta kep,x           ;balra
00050  6690 e8                 inx
00051  6691 e0 27              cpx #$27            ;sor vege?
00052  6693 d0 f5              bne eltol           ;nem, tovabb
00053  6695 ac c3 66           ldy szmut
00054  6698 c4 df              cpy szh             ;szoveg vege?
00055  669a d0 02              bne ugrke           ;nem, tovabb
00056  669c a0 ff              ldy #$ff            ;igen, elolrol
```

```
Line#  Addr Code         Source

00057  669e c8           ugrke  iny                    ;szovegmut.
00058  669f 8c c3 66             sty szmut              ;novelese
00059  66a2 78                   sei
00060  66a3 8d 3f ff             sta ram
00061  66a6 b1 e0                lda (szoveg),y         ;1 uj karakter
00062  66a8 8d 3e ff             sta rom
00063  66ab 58                   cli
00064  66ac 29 bf                and #$bf               ;kepernyokod
00065  66ae 8d e7 0f             sta kep+#$27           ;beleptetese
00066  66b1 ee c4 66      lass   inc szaml
00067  66b4 d0 fb                bne lass
00068  66b6 ee c5 66             inc szaml+1
00069  66b9 d0 f6                bne lass
00070  66bb ad 43 05             lda shflg
00071  66be c9 06                cmp #$06               ;c=+ctrl?
00072  66c0 d0 c6                bne ciklus             ;nem, tovabb
00073  66c2 60                   rts                    ;igen, vege
            00
00074  66c3              szmut .byte 0                  ;szovegmutato
            00 00
00075  66c4              szaml .byte $00,$00

end of assembly, error count = 00000

ablak   667c    bill    db11    chkoma  9491    ciklus  6688
eltol   668a    escb    de67    frestr  9c48    hang    cecd
kep     0fc0    lass    66b1    magno   cfbf    ram     ff3f
rom     ff3e    shflg   0543    strflg  000d    szaml   66c4
szh     00df    szmut   66c3    szolv   6672    szoveg  00e0
ugrke   669e    varptr  0047
```

```
.....page # 1

Line#  Addr Code         Source

00001  0000              ;put"@0:feny2
00002  0000              ;*********************************************
00003  0000              ;*                                           *
00004  0000              ;*  az also sorban folyamatosan kiirja       *
00005  0000              ;*  az a$ szoveget (max. 255 karakter)       *
00006  0000              ;*                                           *
00007  0000              ;*  hivasa: sys 26214,a$                     *
00008  0000              ;*                                           *
00009  0000              ;*  sebessege a control es a commodore gombok *
00010  0000              ;*  segitsegevel szabalyozhato               *
00011  0000              ;*                                           *
00012  0000              ;*********************************************
00013  0000                     * = $6666
00014  6666              ram      = $ff3f          ;ram-ra kapcs.
00015  6666              rom      = $ff3e          ;rom-ra kapcs.
00016  6666              chkoma   = $9491          ;vesszoig elore
00017  6666              strflg   = $0d            ;stringjelzo
00018  6666              frestr   = $9c48          ;szoveges valtozo beol
00019  6666              mszv     = $314           ;megszak. vektor
00020  6666              kep      = $0fc0          ;also sor kezdete
00021  6666              shflg    = $0543          ;ctrl es c= gombok
00022  6666              szh      = $df            ;szoveg hossza
00023  6666              szoveg   = $e0            ;szoveg helye
00024  6666              varptr   = $47            ;szovegvalt. leiroi
00025  6666              hang     = $cecd          ;hang kezelese
00026  6666              magno    = $cfbf          ;magno kezelese
00027  6666              bill     = $db11          ;billentyuzet
00028  6666              escb     = $de67          ;esc 'b' funkcio
00029  6666              ;
00030  6666              ;
00031  6666 a9 17              lda #$17            ;ablak: 24. sor
00032  6668 a2 27              ldx #$27            ;40. oszlop
00033  666a 20 67 de           jsr escb            ;jobb also sarok
00034  666d 78                 sei
00035  666e 20 91 94           jsr chkoma          ;vesszoig
00036  6671 a9 ff              lda #$ff            ;szovegvaltozo
00037  6673 85 0d              sta strflg          ;jon
00038  6675 20 48 9c           jsr frestr          ;szoveg be
00039  6678 a0 02              ldy #$02
00040  667a b1 47        szolv lda (varptr),y      ;leirok
00041  667c 99 df 00           sta szh,y           ;masolasa
00042  667f 88                 dey
00043  6680 10 f8              bpl szolv
00044  6682 c6 df              dec szh             ;1 tobblet le
00045  6684 a9 9a              lda #<megsz         ;beirjuk
00046  6686 8d 14 03           sta mszv            ;az uj
00047  6689 a9 66              lda #>megsz         ;megszakitas
00048  668b 8d 15 03           sta mszv+1          ;erteket
00049  668e a9 ff              lda #$ff            ;mutatok
00050  6690 8d 36 67           sta szmut
00051  6693 a9 00              lda #$00
00052  6695 8d 37 67           sta rmut            ;inic.
00053  6698 58                 cli
00054  6699 60                 rts
00055  669a              ;
00056  669a              ;
```

```
Line#  Addr Code        Source

00057  669a             megsz                    ;uj megszakitas
00058  669a ad 09 ff            lda $ff09
00059  669d 8d 09 ff            sta $ff09
00060  66a0 ae 37 67            ldx rmut
00061  66a3 bd 30 67            lda tff0b,x      ;kov. raszter-
00062  66a6 8d 0b ff            sta $ff0b        ;megsz. helye
00063  66a9 bd 32 67            lda tff07,x      ;vizsz. scroll-
00064  66ac 8d 07 ff            sta $ff07        ;mutato
00065  66af e8                  inx
00066  66b0 e0 02               cpx #$02
00067  66b2 d0 0e               bne ugr01        ;nem, tovabb
00068  66b4 20 cd ce            jsr hang         ;igen, hangkezeles
00069  66b7 20 c8 66            jsr scroll       ;vizsz. scroll
00070  66ba 20 bf cf            jsr magno        ;magno kezelese
00071  66bd 20 11 db            jsr bill         ;bill. kezelese
00072  66c0 a2 00               ldx #$00         ;nullazas
00073  66c2 8e 37 67    ugr01   stx rmut
00074  66c5 4c c3 fc            jmp $fcc3        ;megsz. tovabb
00075  66c8             ;
00076  66c8             ;
00077  66c8             scroll                   ;mozgatas
00078  66c8 ad 35 67            lda seb          ;sebesseg
00079  66cb 8d 34 67            sta smut         ;beall.
00080  66ce ae 33 67    megy    ldx tff07+1      ;finomscroll
00081  66d1 ca                  dex              ;balra
00082  66d2 8e 33 67            stx tff07+1
00083  66d5 e0 ff               cpx #$ff         ;poz. vege?
00084  66d7 d0 2f               bne vissza       ;nem, vissza
00085  66d9 a9 07               lda #$07         ;igen,
00086  66db 8d 33 67            sta tff07+1      ;ujratoltes
00087  66de a2 00               ldx #$00
00088  66e0 bd c1 0f    eltol   lda kep+1,x      ;eltolas
00089  66e3 9d c0 0f            sta kep,x        ;balra
00090  66e6 e8                  inx
00091  66e7 e0 26               cpx #$26         ;sor vege?
00092  66e9 d0 f5               bne eltol        ;nem, tovabb
00093  66eb ac 36 67            ldy szmut
00094  66ee c4 df               cpy szh          ;szoveg vege?
00095  66f0 d0 02               bne ugrke        ;nem, tovabb
00096  66f2 a0 ff               ldy #$ff         ;igen, elolrol
00097  66f4 c8          ugrke   iny              ;szovegmut.
00098  66f5 8c 36 67            sty szmut        ;novelese
00099  66f8 8d 3f ff            sta ram
00100  66fb b1 e0               lda (szoveg),y   ;1 uj karakter
00101  66fd 29 bf               and #$bf         ;kepernyokod
00102  66ff 8d 3e ff            sta rom
00103  6702 8d e6 0f            sta kep+38       ;beleptetese
00104  6705 20 0e 67            jsr tempo        ;sebesseg szabalyoza.a
00105  6708 ce 34 67    vissza  dec smut         ;sebessegmutato
00106  670b d0 c1               bne megy ,nem 0, tovabb
00107  670d 60                  rts
00108  670e             ;
00109  670e             ;
00110  670e             tempo                    ;a sebessegszabalyozasa
00111  670e ad 43 05            lda shflg
00112  6711 c9 04               cmp #$04         ;ctrl?
```

```
Line#  Addr Code        Source

00113  6713 f0 10             beq nov             ;igen, ->nov
00114  6715 c9 02             cmp #$02            ;comm.?
00115  6717 d0 16             bne kesz            ;nem,vissza
00116  6719 ad 35 67          lda seb             ;sebesseg
00117  671c c9 01             cmp #$01            ;=1?
00118  671e f0 0f             beq kesz            ;igen,vissza
00119  6720 ce 35 67          dec seb             ;csokkentes
00120  6723 10 0a             bpl kesz            ;vissza
00121  6725 ad 35 67    nov   lda seb             ;sebesseg
00122  6728 c9 0f             cmp #$0f            ;=maximum?
00123  672a f0 03             beq kesz            ;igen,vissza
00124  672c ee 35 67          inc seb             ;noveles
00125  672f 60          kesz  rts
00126  6730               ;
00127  6730               ;
            c2 cc
00128  6730               tff0b .byte $c2,$cc
            08 01
00129  6732               tff07 .byte $08,$01
            00
00130  6734               smut  .byte 0           ;sebessegmutato
            03
00131  6735               seb   .byte 3           ;sebesseg
            00
00132  6736               szmut .byte 0           ;szovegmutato
            00
00133  6737               rmut  .byte 0           ;rasztermutato
00134  6738                     * = $6500

end of assembly, error count = 00000

bill    db11    chkoma  9491    eltol   66e0    escb    de57
frestr  9c48    hang    cecd    kep     0fc0    kesz    672f
magno   cfbf    megsz   669a    megy    66ce    mszv    0314
nov     6725    ram     ff3f    rmut    6737    rom     ff3e
scroll  66c8    seb     6735    shflg   0543    smut    6734
strflg  000d    szh     00df    szmut   6736    szolv   667a
szoveg  00e0    tempo   670e    tff07   6732    tff0b   6730
ugr01   66c2    ugrke   66f4    varptr  0047    vissza  6708
```

```
Line#  Addr Code        Source

00001  0000             ;put"@0:feny3
00002  0000             ;*********************************************
00003  0000             ;*                                           *
00004  0000             ;*  kulon kepernyon folyamatosan kiirja      *
00005  0000             ;*  az a$ szoveget (max. 255 karakter)       *
00006  0000             ;*                                           *
00007  0000             ;*  hivasa: sys 26214,a$                     *
00008  0000             ;*                                           *
00009  0000             ;*  sebessege a control es a commodore gombok *
00010  0000             ;*  segitsegevel szabalyozhato               *
00011  0000             ;*                                           *
00012  0000             ;*********************************************
00013  0000                    * = $6666
00014  6666             ram      = $ff3f          ;ram-ra kapcs.
00015  6666             rom      = $ff3e          ;rom-ra kapcs.
00016  6666             chkoma   = $9491          ;vesszoig elore
00017  6666             strflg   = $0d            ;stringjelzo
00018  6666             frestr   = $9c48          ;szoveges valtozo beolvasas
00019  6666             mszv     = $314           ;megszak. vektor
00020  6666             kep      = $6428          ;kep
00021  6666             kepm     = $6400          ;kepmemoria
00022  6666             szm      = $6000          ;szinmemoria
00023  6666             shflg    = $0543          ;ctrl es c= gombok
00024  6666             szh      = $df            ;szoveg hossza
00025  6666             szoveg   = $e0            ;szoveg helye
00026  6666             varptr   = $47            ;szovegvalt. leiroi
00027  6666             hang     = $cecd          ;hang kezelese
00028  6666             magno    = $cfbf          ;magno kezelese
00029  6666             bill     = $db11          ;billentyuzet
00030  6666             ;
00031  6666             ;
00032  6666 78                 sei
00033  6667 20 91 94           jsr chkoma        ;vesszoig
00034  666a a9 ff              lda #$ff          ;szovegvaltozo
00035  666c 85 0d              sta strflg        ;jon
00036  666e 20 48 9c           jsr frestr        ;szoveg be
00037  6671 a0 02              ldy #$02
00038  6673 b1 47        szolv lda (varptr),y    ;leirok
00039  6675 99 df 00           sta szh,y         ;masolasa
00040  6678 88                 dey
00041  6679 10 f8              bpl szolv
00042  667b c6 df              dec szh           ;1 tobblet le
00043  667d a9 a2              lda #<megsz       ;beirjuk
00044  667f 8d 14 03           sta mszv          ;az uj
00045  6682 a9 66              lda #>megsz       ;megszakitas
00046  6684 8d 15 03           sta mszv+1        ;erteket
00047  6687 a2 50              ldx #$50
00048  6689 a9 20        tcik  lda #$20          ;kep- es
00049  668b 9d 00 64           sta kepm,x        ;szinmem.
00050  668e a9 00              lda #$00
00051  6690 9d 00 60           sta szm,x
00052  6693 ca                 dex
00053  6694 10 f3              bpl tcik
00054  6696 a9 ff              lda #$ff          ;mutatok
00055  6698 8d 62 67           sta szmut
00056  669b a9 00              lda #$00
```

```
Line#  Addr Code         Source

00057  669d 8d 5c 67           sta rmut              ;inic.
00058  66a0 58                 cli
00059  66a1 60                 rts
00060  66a2             ;
00061  66a2             ;
00062  66a2             megsz                        ;uj megszakitas
00063  66a2 ad 09 ff           lda $ff09
00064  66a5 8d 09 ff           sta $ff09
00065  66a8 ae 5c 67           ldx rmut
00066  66ab bd 53 67           lda tff1c,x           ;rasztersor
00067  66ae 8d 1c ff           sta $ff1c             ;szamlalo
00068  66b1 bd 56 67           lda tff1d,x           ;felso-also
00069  66b4 8d 1d ff           sta $ff1d             ;resze
00070  66b7 bd 4d 67           lda tff0b,x           ;kov. raszter-
00071  66ba 8d 0b ff           sta $ff0b             ;megsz. helye
00072  66bd bd 4a 67           lda tff07,x           ;vizsz. scroll-
00073  66c0 8d 07 ff           sta $ff07             ;mutato
00074  66c3 bd 50 67           lda tff14,x           ;kepernyo
00075  66c6 8d 14 ff           sta $ff14             ;kezdete
00076  66c9 e8                 inx
00077  66ca e0 03              cpx #$03
00078  66cc d0 0e              bne ugr01             ;nem, tovabb
00079  66ce 20 cd ce           jsr hang              ;igen, hangkezeles
00080  66d1 20 e2 66           jsr scroll            ;vizsz. scroll
00081  66d4 20 bf cf           jsr magno             ;magno kezelese
00082  66d7 20 11 db           jsr bill              ;bill. kezelese
00083  66da a2 00              ldx #$00              ;nullazas
00084  66dc 8e 5c 67     ugr01 stx rmut
00085  66df 4c c3 fc           jmp $fcc3             ;megsz. tovabb
00086  66e2             ;
00087  66e2             ;
00088  66e2             scroll                       ;mozgatas
00089  66e2 ad 5a 67           lda seb               ;sebesseg
00090  66e5 8d 59 67           sta smut              ;beall.
00091  66e8 ae 4b 67     megy  ldx tff07+1           ;finomscroll
00092  66eb ca                 dex                   ;balra
00093  66ec 8e 4b 67           stx tff07+1
00094  66ef e0 ff              cpx #$ff              ;poz. vege?
00095  66f1 d0 2f              bne vissza            ;nem, vissza
00096  66f3 a9 07              lda #$07              ;igen,
00097  66f5 8d 4b 67           sta tff07+1           ;ujratoltes
00098  66f8 a2 00              ldx #$00
00099  66fa bd 29 64     eltol lda kep+1,x           ;eltolas
00100  66fd 9d 28 64           sta kep,x             ;balra
00101  6700 e8                 inx
00102  6701 e0 26              cpx #$26              ;sor vege?
00103  6703 d0 f5              bne eltol             ;nem, tovabb
00104  6705 ac 5b 67           ldy szmut
00105  6708 c4 df              cpy szh               ;szoveg vege?
00106  670a d0 02              bne ugrke             ;nem, tovabb
00107  670c a0 ff              ldy #$ff              ;igen, elolrol
00108  670e c8           ugrke iny                   ;szovegmut.
00109  670f 8c 5b 67           sty szmut             ;novelese
00110  6712 8d 3f ff           sta ram
00111  6715 b1 e0              lda (szoveg),y        ;1 uj karakter
00112  6717 8d 3e ff           sta rom
```

```
.....page # 1

Line#  Addr Code        Source


00113  671a 29 bf             and #$bf            ;kepernyokod
00114  671c 8d 4e 64          sta kep+38          ;beleptetese
00115  671f 20 28 67          jsr tempo           ;sebesseg szabalyozasa
00116  6722 ce 59 67    vissza dec smut           ;sebessegmutato
00117  6725 d0 c1             bne megy ,nem 0, tovabb
00118  6727 60                rts
00119  6728             ;
00120  6728             ;
00121  6728             tempo                     ;a sebessegszabalyozasa
00122  6728 ad 43 05          lda shflg
00123  672b c9 04             cmp #$04            ;ctrl?
00124  672d f0 10             beq nov             ;igen, ->nov
00125  672f c9 02             cmp #$02            ;comm.?
00126  6731 d0 16             bne kesz            ;nem,vissza
00127  6733 ad 5a 67          lda seb             ;sebesseg
00128  6736 c9 01             cmp #$01            ;=1?
00129  6738 f0 0f             beq kesz            ;igen,vissza
00130  673a ce 5a 67          dec seb             ;csokkentes
00131  673d 10 0a             bpl kesz            ;vissza
00132  673f ad 5a 67    nov   lda seb             ;sebesseg
00133  6742 c9 0f             cmp #$0f            ;=maximum?
00134  6744 f0 03             beq kesz            ;igen,vissza
00135  6746 ee 5a 67          inc seb             ;noveles
00136  6749 60          kesz  rts
00137  674a             ;
00138  674a             ;
       08 01 08
00139  674a             tff07 .byte $08,$01,$08
       03 ee cd
00140  674d             tff0b .byte $03,$ee,$cd
       60 60 08
00141  6750             tff14 .byte $60,$60,$08
       01 00 00
00142  6753             tff1c .byte $01,$00,$00
       36 bd fa
00143  6756             tff1d .byte $36,$bd,$fa
       00
00144  6759             smut  .byte 0             ;sebessegmutato
       03
00145  675a             seb   .byte 3             ;sebesseg
       00
00146  675b             szmut .byte 0             ;szovegmutato
       00
00147  675c             rmut  .byte 0             ;rasztermutato

end of assembly, error count = 00000

bill    db11    chkoma  9491    eltol   66fa    frestr  9c48
hang    cecd    kep     6428    kepm    6400    kesz    6749
magno   cfbf    megsz   66a2    megy    66e8    mszv    0314
nov     673f    ram     ff3f    rmut    675c    rom     ff3e
scroll  66e2    seb     675a    shflg   0543    smut    6759
strflg  000d    szh     00df    szm     6000    szmut   675b
szolv   6673    szoveg  00e0    tcik    6689    tempo   6728
tff07   674a    tff0b   674d    tff14   6750    tff1c   6753
tff1d   6756    ugr01   66dc    ugrke   670e    varptr  0047
```

A PEDAGÓGIAI MŰHELY SOROZATBAN MEGJELENT KIADVÁNYOK JEGYZÉKE

1/ Mikus Katalin: Ki ? Miről ? Kinek ? 1987

2/ Szöllősi Zsuzsanna: A filozófiaoktatás helyezete Pest megyében 1988

3/ Az osztályfőnöki nevelőmunka tapasztalatainak összegzése

4/ Az általános iskolai fakultáció szerepe a differenciált képességfejlesztésben

5/ Fürstné dr. Kólyi Erzsébet: Munkaközösségek az iskolában

6/ Mikus Katalin: Bázisintézmények Pest megyében

7/ Muskovits Lászlóné: Középtávú pedagógiai program

8/ Humanisztikus iskola 1989

9/ Farkas Károly-Kőrösné Mikis Márta: Játszd el a teknőcöt!

10/ Németh Ferenc: Számítástechnika fakultáció

11/ Gudenus László: Bevezetés a Commodore Plus/4 gépi kódú programozásába

12/ Dr. Abrudbányay János: A napközis nevelőmunka alapkérdései II.

13/ Kunczené Fellegi Katalin: Eredményvizsgálat matematikából

14/ Dr. Gulyás Sándor: Olvasásvizsgálatok Pest megyében

15/ Pedagógiai innováció 1990

16/ Diósi Gábor: A gyógypedagógiai nevelés története Pest megyében

17/ Farkas-Kőrösné: Játszd el a teknőcöt! 3-4. osztály

18/ Farkas Károly: LOGO példatár