DR. HERENDI ISTVÁN,
PATAKY ZSUZSA

SZÁMÍTÓGÉPES FOGÁSOK, TRÜKKÖK C 16-RA ÉS C PLUS/4-RE

ORSZÁGOS PEDAGÓGIAI INTÉZET

VORKER®

Számítógépes fogások, trükkök C 16-ra és C Plus/4-re

Főszerkesztő:

dr. Szűcs Barna

Szerkesztő:

Borbola István

A kiadvány szerzői:

Dr. Herendi István, Pataky Zsuzsa

Bírálták:

Dr. Makay Árpád

Varjú Károly

ISBN 963 682 160 7 (Ö)
ISBN 963 682 172 O

Kiadja az Országos Pedagógiai Intézet

Felelős kiadó: Szabolcsi Miklós

TARTALOMJEGYZÉK

BEVEZETÉS 13

1. A C 16, C Plus/4 számítógép rendszertechnikai felépítése 15
1.1.
1.1.1. Információtárolás a számítógépben 15
1.1.2. A számítógépes rendszer hardver egységei 17
1.1.3. A központi egység és a tár együttműködése. A gépi ciklus 19
1.1.4. A perifériák kapcsolódása, kezelése 20
1.1.4.1.
1.2. A számítógép szoftver elemei 21
1.2.1. A BASIC interpreter működése 21
1.2.1.1. STOP és RESET 22
1.2.2. Az interpreter működése 22
1.3. A memória szervezése 23
1.3.1. A memória felosztása. A tár szervezése 23
1.3.1.1.

2. BASIC programozói fogások 25
2.1.
2.1.1. A BASIC program elhelyezkedése a memóriában 25
2.1.1.1.
2.2. A BASIC program optimalizálása, gyorsítása 26
2.2.1. Fogások a BASIC mutatókkal 28
2.2.2. BASIC programok láncolása 28
2.2.3. Törölt program aktivizálása a memóriában 28
2.3. A beépített MONITOR 29
2.3.1. A BASIC program mutatóinak átírása MONITOR segítségével 29
2.3.1.1.
2.3.2. A gépi kódú program és a BASIC program kapcsolata 30
2.3.3. Gépi kódú program beírása MONITOR segítségével 31
2.3.4. Gépi kódú program betöltése BASIC program futása közben .. 32
2.3.4.1.
2.3.5. Gépi kódú program DATA sorokká való átírása 32
2.3.6. A SYS utasítás használata 33
2.3.6.1.

2.3.7. Az USR utasítás használata 34
2.3.7.1.
2.3.8. Képernyőablakok előállítása, kezelése 35
2.3.9. Képernyőablakok kijelölése POKE utasítással 36
2.3.10. Kettő vagy több képernyőablak kezelése 37
2.3.11. Felirat állandó megjelenítése a képernyőn 38
2.3.12. A képernyőablakok színezése 38
2.3.12.1.
2.3.13. Vezérlő karakterek a képernyő kezeléséhez 40
2.3.14. Fogások a DEL/INS billentyűvel 40
2.3.15. Kvázigrafikus lehetőségek rajzokhoz 41
2.4. A géphez kapcsolt perifériák 41
2.4.1. A kazetta puffer használata 41
2.4.2. A kazettás magnetofon állapotának lekérdezése 42
2.4.2.1.
2.4.3. Automatikus programindítás 43
2.4.3.1.
2.4.4. Periféria ellenőrzése futtatás közben 43
2.5. A kurzor kezelése 44
2.5.1. A kurzor pozícionálása és kioltása 44
2.5.2. A lenyomott billentyű lekérdezése 44
2.5.2.1.
2.5.3. A lenyomott billentyű ismétlése 46
2.5.4. Input utasítás kérdőjel nélkül 46
2.5.5. Dinamikusbillentyűzet-technika 46
2.6. Növeljük meg a beépített interpreter tudását! 47
2.6.1. Önmagát futtatás közben módosító program 47
2.6.1.1.
2.6.2. Gépi kódú rutin elhelyezése a BASIC program sorai között ... 48
2.6.3. A LIST parancs módosítása 49
2.6.4. A BASIC bővítése: új szavak definiálása 49
2.6.4.1.
2.7. Programok védelme 54
2.7.1. Memóriacímek módosítása 54
2.7.2. Egyszerű programvédelem 55
2.7.3. Programvédelem a LIST befolyásolásával 55
2.7.3.1.

3. Gépi kódú programozás. Fogások gépi kódban programozóknak 57
3.1.
3.1.1. A KERNAL rutinok aktivizálása 57
3.1.2. Ciklus szervezése gépi kódban 58

3.1.3. IF...THEN...ELSE gépi kódban 58
3.1.3.1.
3.2. A képernyő gépi kódú kezelése 59
3.2.1. A képernyő kikapcsolása és görgetése 59
3.2.2. A megszakító rutin módosítása, a karakterkészlet átmásolása .. 60
3.2.3. A memóriablokkok átkapcsolása 61
3.2.4. A KERNAL ROM olvasása PEEK segítségével 62
3.2.4.1.

4. A C 16 és C Plus/4 gépekbe épített TED csip 64
4.1.
4.1.1. A TED csip áttekintése 64
4.1.1.1.
4.1.2. A TED csip néhány regiszterének módosítása 65

5. A képernyő tartalmának kinyomatása 66
5.1.
5.1.1. A virtuális és a valós nyomtatás 66
5.1.2. Hard copy készítése karakteres képernyőről 66
5.1.3. Hard copy készítése nagy felbontású képernyőről 67
5.1.3.1.

FELDOLGOZÁSI JAVASLAT 68

A SZOFTVER MELLÉKLET TARTALOMJEGYZÉKE 71

IRODALOMJEGYZÉK 75

TÁRGYMUTATÓ 76

A SOROZAT TAGJAI

Az Országos Pedagógiai Intézet Számítástechnikai Programirodájának gondozásában 1987/88-ban 12 számítástechnikai témakörhöz készült oktatási anyag C 16-os és C Plus/4-es Commodore házi számítógépekre:

1. Algoritmusok, játékok:

A kiadvány tömören, de közérthetően, szemléletes példákon keresztül világít rá a probléma—algoritmus—program kapcsolatra, elvi és gyakorlati tanácsokat ad a jól felépített programok készítéséhez. A hozzá tartozó kazetta tizenkét, önmagában is alkalmazható, számítástechnikai és matematikai tanulságokkal teli programot tartalmaz. A programok jórészt listázhatók, így példát mutatnak egy színvonalas programozási stílus kialakítására, a gyakorlottabb felhasználóknak lehetőséget adnak arra, hogy más programok készítésénél is hasznosítsák az itt bemutatott programozási fogásokat. *Ajánlott:* ált. isk. 7., 8. osztályosainak és középiskolásoknak.

2. A számítógép és környezete:

Mindazoknak készült, akik szeretnék összekötni saját számítógépüket a külvilággal. 8 db különböző interfész és periféria tervezése kapcsán ismerteti meg az olvasót az interfész tervezés és illesztés fogásaival. Útmutató és 8 kapcsolási rajz, kész nyomtatott áramköri tervrajz. *Ajánlott:* ált. isk. és középiskolai tanároknak, számítástechnikai amatőröknek, 8. osztályos és középiskolai tanulóknak, tanárjelölt főiskolai és egyetemi hallgatóknak.

3. Programmodulok:

A felhasználó 55 db szubrutint, programmodult, ill. önálló felhasználói programot kap. Ezek egy része gépi kódú rutin, melyet az átlagos felhasználó nem tud megírni, de beépítheti munkájába. A felhasználói programok ötletet adhatnak hasonló programok megírásához. Az egyes rutinok, programok alkalmazási lehetőségeinek, működésének leírását útmutató tartalmazza. *Ajánlott:* ált. iskolai tanároknak, tanulóknak, kezdő amatőr programozóknak.

4. Adatkezelés számítógéppel C 16-ra és C Plus/4-re:

A témakör feldolgozását segítő útmutató az adatkezelés, adatfeldolgozás alapfogalmaival és folyamatával foglalkozik. Megismertet az adatfeldolgozó rendszerek

fejlesztésével és a rendszerek alkalmazásbavételével. A feldolgozást 5 program és 2 minta-adatállomány segíti. *Ajánlott:* ált. és középiskolai tanároknak, fakultációs tananyagként, továbbképzési anyagként.

5. Szövegfeldolgozás számítógéppel:

A témakör feldolgozásához a felhasználó megkapja a Nemzetközi ABC szövegszerkesztő programot, amely magyar, cirill, német, lengyel, görög karakterkészletet és különleges matematikai jeleket (gyökjel, integráljel, alsó és felső index stb.) is tartalmaz. A szövegek és különleges karakterek nyomtathatók — akár fűzött kézírást imitálva is. A kézírásos nyomtatás főleg alsó tagozatban számíthat sikerre. A szövegszerkesztő minden olyan szolgáltatást tud, amely a tudományos munkához, vagy a tanári munkához kell. A feldolgozást útmutató és bemutató programok is segítik. *Ajánlott:* tanároknak, kutatóknak, íróknak, adminisztrátoroknak, ill. fakultációk anyagaként.

6. Számítógépes grafika C 16-on és C Plus/4-en:

Az útmutató és a mintaprogramok a felhasználói kézikönyvből meg nem tanulható különleges grafikus lehetőségekkel ismertetik meg az olvasót. A többszínű háttérszín üzemmód mellett a bittérképes színes grafikák elkészítésének fortélyait is megtanulhatja a felhasználó. Az útmutató és a mintaprogramok áttanulmányozása után az olvasó saját maga is képes lesz a játékprogramokból ismert színes és mozgó figurák „szellemecskék" programozására. *Ajánlott:* ált. isk. 7., 8. osztályosainak, középiskolásoknak és a grafika programozását most kezdő programozóknak.

7. Hangkeltés C 16-os (és C Plus/4-es) számítógéppel:

Az anyaghoz az útmutató mellett bemutató programok is tartoznak. Ezek áttanulmányozása alkalmassá teszi a felhasználót egy és többszólamú dallamok programozására, az interrupt alatti programozási lehetőségek kihasználására. A programok és dallamok az ált. iskolai ének-zene oktatáshoz igazodnak, így felhasználhatók számítógépes motivációra. *Ajánlott:* ált. isk. 7., 8. osztályosoknak, középiskolásoknak, valamint ált. iskolai énektanároknak, kezdő programozóknak.

8. Számítógépes fogások, trükkök C 16-ra és C Plus/4-re:

Az útmutató és a hozzá tartozó 16 program azok számára mutatja be a C 16-on és a C Plus/4-en alkalmazható fogásokat, akik a BASIC programozás alapjain túljutottak. A közölt rendszerváltozók, memóriacímek és gépi kódú programok megismertetik az olvasót az alkalmazások különleges lehetőségeivel. *Ajánlott:* kezdő programozóknak, általános és középiskolásoknak, tanároknak.

9. Számítógép és video:

Az anyaghoz útmutató, videofilm és bemutató programok tartoznak. A videofilm a hozzá tartozó számítógépes programokkal egy megvalósítható rendszert és egy példát mutat be a képmagnó és a házi számítógép összekapcsolására, az oktatásban történő alkalmazására. Az anyag az elveken túl egy interfész elkészítésével is megismerteti az olvasót. Elsősorban tanároknak, oktatástechnikai szakembereknek ajánlott!

10. Számítógéppel vezérelt mérések:

Az útmutató megismerteti az olvasót a legfontosabb méréstani fogalmakkal, a mérések tervezésének, elvégzésének, értékelésének elméleti és műszaki-, méréstani alapjaival. A mérőberendezések és mérési összeállítások jelátvivő funkcionális egységeként a Tudományszervezési és Informatikai Intézetnél kapható Techno-MIR moduláris interfészrendszer berendezéseit mutatja be és használja fel a anyag. A témakörhöz tartozó 21 db professzionális program egyrészt az oktatómunkában és a bemutató kísérletekben alkalmazható számítógép-vezérlésű intelligens műszert szimulál (pl. tároló oszcilloszkópot, 8 csatornás logikai analizátort, digitális multimétert stb.) másrészt az iskolai kísérletekhez szükséges mérési összeállításokat vezérli. *Ajánlott:* általános és középiskolai tanároknak, speciális szakképző intézeteknek, pedagógusképző intézeteknek bemutatási és oktatási célra, pedagógiai intézeteknek a továbbképzéshez.

11. Számítógépes irányítástechnika:

Az anyaghoz útmutató, az útmutatóban ismertetett berendezésekhez 11 db mintaprogram tartozik. Ezek áttanulmányozása megismerteti az olvasót a számítógépes vezérlések és szabályozások alapfogalmaival, a mechatronikus modelleket működtető berendezések (kapcsolómodulok, interfészek) elkészítésével. A mintaprogramok elemzése példát mutat az irányítási algoritmusok programozására. Az irányító rendszerek jelátvivő funkcionális egységeként a Tudományszervezési és Informatikai Intézetnél kapható Techno-MIR interfészrendszer elemeit, érzékelő-, erősítő berendezéseknek és beavatkozó szerveknek, berendezéseknek pedig saját készítésű, a felhasználó által is reprodukálható eszközöket használ az anyag. Ezek leírásait és elkészítési módját is tartalmazza. *Ajánlott:* az általános és középiskolai technika tárgy oktatásához, fakultációkhoz, szakköri feldolgozáshoz.

12. Számítógépek összekapcsolása, helyi oktatóhálózatok:

Az anyag egy hazánkban eddig nyomtatásban még meg nem jelent területtel, a Commodore házi számítógépek (és a VIDEOTON TV COMPUTER) összekapcsolásának gyakorlati megvalósításával és e számítógépekből összeállított helyi oktatóhálózatok programozásával foglalkozik. Megismerteti a felhasználóval a számítógépek közötti kapcsolat hardver és szoftver eszközeit, a kapcsolatteremtés és az adat-

átvitel alapfogalmait. Megépített és reprodukálható kapcsolásokkal és kész felhasználói programokkal segíti, hogy az anyagot feldolgozó olvasó saját maga is összekapcsolhasson számítógépeket. A megszerzett ismeretek segítségével a minimális elektronikai ismeretekkel és készüléképítési gyakorlattal rendelkező felhasználó is meg tud valósítani számítógépek közötti adatátvitelt. Az útmutató második része az oktatásban bevált és sikerrel alkalmazott, a szegedi VORKER Kisszövetkezet által kifejlesztett TC—NET oktatóhálózatokkal, azok programozási fogásaival foglalkozik. Az anyag 6 db kapcsolást, 1 db nyomtatott áramköri rajzot és 12 db professzionális felhasználói programot tartalmaz. *Ajánlott:* általános és középiskolai tanároknak, 3., 4. éves középiskolai tanulóknak, számítástechnikai amatőröknek, középfokú szakképző intézeteknek, tanárképző intézeteknek, pedagógiai intézeteknek, továbbképző intézeteknek stb.
Valamennyi itt felsorolt anyag (útmutató és program), valamint a TC—NET megrendelhető a szegedi VORKER Kisszövetkezettől.

Cím:
VORKER Kisszövetkezet
6701. SZEGED, Pf.: 711
Tel.: H—06-62-*26-144*
Telex: 82688

A VORKER Kisszövetkezet vállalja — megrendelés alapján — a felsorolt témakörökben bemutatók, alap- és továbbképzések tartását, szervezését is.

ELŐSZÓ

Az olvasó a szegedi Tarjánvárosi IV. számú Általános Iskolában fejlesztett 12 pedagógiai program (tanári kézikönyv, szoftver és/vagy beültetési rajzok) egyikét tartja a kezében. A munkában több mint hatvanan vettek részt: általános és középiskolai tanárok, főiskolai, egyetemi oktatók, tudományos intézetek munkatársai.

Az Országos Pedagógiai Intézet Számítástechnikai Programirodája és a szegedi iskola közös vállalkozása összefügg a számítástechnikai alapműveltség körülhatárolására irányuló erőfeszítésekkel. Kétségtelen, hogy a számítástechnikai-informatikai alapismeretek és az alapműveltség kapcsolatának feltárása elsősorban teoretikus jellegű munkát igényel. Az 1970-es években induló tartalmi korszerűsítés tapasztalatai azonban arra hívják fel a figyelmünket, hogy az iskolák nélkül, az iskolai hagyományok és a pedagógia öntörvényeinek mellőzésével nem hozhatók létre olyan dokumentációk, amelyeket a gyakorlat magáénak érez, s amelyekkel a pedagógusok többsége közösséget vállal.

Az alapműveltség témakörében már eddig is több tanulmány látott napvilágot. Viszonylag kevesebb szó esett azonban arról, hogy miként lehet pedagógiailag szervezett ismeretrendszerré formálni az itthon is és külföldön is már megfogalmazott elképzeléseket (amelyek között meglepően szoros összefüggés lelhető fel).

Az elképzelésekről szóló következő rövid ismertetés nem tekinthető állásfoglalásnak abban a kérdésben, hogy mi a jobb: ha a számítástechnikai-informatikai alapismeretekhez a diákok az *óratervben,* különféle tantárgyakban jutnak hozzá; *szakköri vagy fakultációs* programok keretében, esetleg *külön tantárgyként* biztosítják ezt számukra.

A fejlesztők munkáját a következő feltételezések irányították:

1. A számítástechnikai-informatikai ismeretek és az ez iránti érdeklődés nehezen tagolható az eddig megszokott módon, vagyis életkor és iskolatípus szerint. Az elkészült anyagoknak tehát egyaránt szolgálniok kell az általános iskolákat és a középfokú iskolákat.
2. Az iskolák fogadókészsége eltérő (pl. más-más a felszerelés, a tanárok felkészültsége, a szoftverellátottság stb.). Ez aligha teszi lehetővé az eddig megszokott, a mindenki számára kötelező tantervi előírásokat, sémákat. Az igényekhez tehát csak a többféle lehetőség egyidejű bemutatásával kerülhetünk közelebb.
3. A 12 pedagógiai program és ezek járulékai alapul szolgálhatnak egy majdan országos érvényű informatika tantárgy kidolgozásához. A témák ilyen rendszere a már ma megvalósítható iskolai tevékenységeket tükrözi. Az informatika több más témája-

ról az iskolák jórésze számára ma még csupán leírás adható (pl. országos, nemzetközi hálózatok). A későbbiekben elképzelhető, hogy az iskolákban is lehetőség lesz az informatika teljes skálájának bemutatására.

4. A 12 rész között biztosíthatók „átjárások": a részek egymáshoz illeszthetők és kombinálhatók egymással.

5. A pedagógiai programokban megfogalmazott elképzelések érvényesítése érdekében az elinduláshoz szükséges a megfelelő segédletek elkészítése.

6. A pedagógiai programokhoz készült járulékok (beültetési rajzok, szoftverkészlet) egyik részének tantárgyi alkalmazásokra kell épülnie (pl. matematika, mérések), másik részük viszont az informatikának a mindennapi életben megjelenő gyakorlatát tükrözze (pl. adatkezelés, szövegszerkesztés).

Reméljük, hogy munkánk segíteni fogja a pedagógiai gyakorlatot és ezáltal hozzájárul e téma elméleti kérdéseinek tisztázásához.

Örömmel és tisztelettel fogadjuk munkánk olvasójának, használójának észrevételeit.

Dr. Szűcs Barna

BEVEZETŐ

A BASIC nyelvű programozás elsajátítása után sok felhasználó tovább akar lépni a számítógép megismerésében. A szakkört vagy fakultációt vezető tanár, az érdeklődő diák tudni akarja, hogy mi okoz egy-egy megfejthetetlennek tűnő hibát. S ha már valamit megtudtak, akkor az újabb kérdések következnek.
Ebben a megismerésben segít ez a kiadvány. Technikai szempontból a C 16-ot és a C Plus/4-et egységesen kezeltük. Az egyes fejezetekben ötleteket, tippeket adunk a számítógép alaposabb megismeréséhez és a használatához. A rendszertechnikai felépítés leírása után a BASIC interpreterrel, a már futó BASIC programok optimalizálásával foglalkozunk. Példákkal segítjük a gépi kódú programozással próbálkozókat. Megmutatjuk, hogyan lehet a beépített gépi kódú rutinokat a saját programjainkban felhasználni. A gép megszakítási rendszerét és a belső óra működését, továbbá még sok ötletet, fogást ismertetünk. Ezzel a segédlettel, valamint a felsorolt szakirodalom segítségével az olvasó is képes lesz újabb hasznos ötletek kidolgozására.

1. A C 16, C Plus/4 SZÁMÍTÓGÉP RENDSZERTECHNIKAI FELÉPÍTÉSE

A számítógép használata során a gép hardver és szoftver elemeit egységes rendszerként kezeljük. Ebben a témakörben megismerkedünk a legfontosabb rendszertechnikai fogalmakkal, valamint az interpreter működésével.

1.1. A BASIC nyelvet értő, a programozásban már bizonyos fokú jártasságot szerzett felhasználók egyre többször találkoznak olyan kérdésekkel, amelyeket nem, vagy csak nehezen tudnak megválaszolni. Úgy érzik, hogy jó lenne „belelátni" a gépbe, milyen alkatrészek vannak benne. Jó lenne tudni, hogy ezek az alkatrészek hogyan működnek, hogyan kapcsolódnak egymáshoz, hogyan alakul ki a rendszer, amely már maga a számítógép.

Ha valaki felbontja a gép dobozát, akkor láthatja, hogy egyetlen műanyag lapon sok-sok alkatrészt helyeztek el. Ezek gondoskodnak a programok beolvasásáról, tárolásáról. Feladatuk az is, hogy a futtatás közben a gép és a perifériák pontos együttműködését megteremtsék.

Vizsgáljuk meg, hogyan tárolja a számítógép az információt, és **milyen részegységek alkotják a C 16-ot és a C Plus/4-et!**

1.1.1. *Információtárolás a számítógépben*

Az egységes rendszerbe foglalt részegységek közös nyelve a kód. A számítógépek működése ezeknek a kódoknak az előállításán, tárolásán, majd pedig visszaalakításán, értelmezésén alapszik.

Úgy is fogalmazhatunk, hogy a számítógép olyan kódfeldolgozó automata, amely a működéséhez szükséges információt önmaga tartalmazza. Ezt a kódsorozatot meg is lehet változtatni, azaz a számítógépet programozni lehet.

A számítógépben az információ alapegysége a bit. Állapota, azaz a benne tárolt információ a jelölésünk szerint 1 vagy 0 lehet. Nyolc ilyen bitet összefogva kapjuk a bájtot, amely a bájt-szervezésű gépek legkisebb címezhető egysége.

Ha egy számítógép működése nem a bájtszervezésen alapul, akkor szószervezésűnek nevezzük. Egy gépi szó 24, 36 stb. számú bit hosszúságú lehet.

A kettes számrendszerben a számábrázolás a felhasználó szempontjából nem túl kényelmes. A bináris szám legtöbbször hosszú és nehezen áttekinthető.

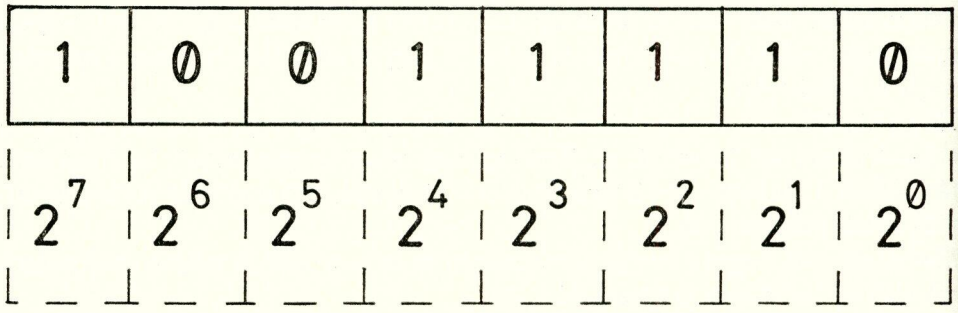
A hexadecimális számábrázolás sokkal tömörebb ábrázolást tesz lehetővé. A bináris ábrázolású szám számjegyeit jobbról balra haladva négyes csoportokba fogjuk össze.

Így azonban újabb számjegyekre van szükségünk, ugyanis $2^4=16$, ami már nem egy, hanem kétjegyű decimális szám. Így újabb jeleket kell bevezetnünk. Legyenek ezek 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F!

Gyakorlati szempontból az 1 bájton ábrázolt 8 bites bináris szám 2 négybites számra bontható, s ezek pedig már kifejezhetők hexadecimális formában. Például: a decimális 158-at írjuk fel bináris és hexadecimális formában is!

128	$128=2^7$
16	$16=2^4$
8	$8=2^3$
4	$4=2^2$
+2	$2=2^1$
158	

Ezt a számot 1 bájton így ábrázoljuk:



azaz $158_{10}=10011110_2$.

A nyolc bit közül a bal szélső 4 darab: 1001, amiről tudjuk, hogy

$$1001_2 = 9_{16}.$$

A jobb szélső 4 bit: 1110, és erre igaz, hogy

$$1110_2 = E_{16}.$$

Ezek után kijelenthetjük, hogy

$$158_{10} = 10011110_2 = 9E_{16}$$

A $9E_{16}$ szám szokásos jelölése: \$9E, azaz a szám elé \$ jelet írunk.

Használjuk még az oktális, azaz a nyolcas számrendszert is, amikor a kettes számrendszerben ábrázolt szám számjegyeit jobbról balra haladva nem négyes, hanem hármas csoportokba fogjuk össze. A nyolcas számrendszer legnagyobb számjegye a 7. A hexadecimális ábrázolásról az eljárás fordított irányú alkalmazásával térhetünk át a bináris számábrázolásra.

1.1.1.1. A különböző számrendszerekben történő számábrázolás különösen a gépi kódú programozás szempontjából igen lényeges ismeret. Törekedjünk arra, hogy a tanulók készségszintre jussanak a számrendszerek között történő átváltásokban, azaz megbízhatóan tudjanak konvertálni!

1.1.2. A számítógépes rendszer hardver egységei

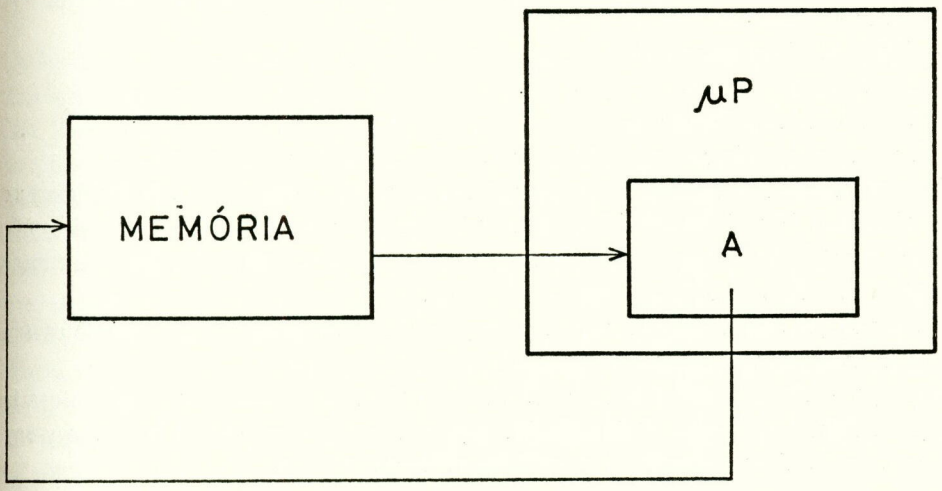
A C 16 és a C Plus/4 gépeket mikroszámítógépeknek is nevezzük. A mikroszámítógépek hardver rendszerének középpontjában a mikroprocesszor áll. E köré csoportosulnak a központi tár és az I/0 műveleteket végző egységek. Összekapcsolásukról egy speciális vezetékrendszer, a busz gondoskodik.

A mikroprocesszorok 3 fő részből állnak:
— regiszterekből
— műveletvégző egységből
— vezérlő egységből.
Ezek a buszon keresztül tartják egymással a kapcsolatot.

Egy mikroprocesszor általában több regisztert tartalmaz. Tárolják a mikroprocesszor által a memóriából kiolvasott adatot. Az adattal dolgozik a mikroprocesszor. Az eredmény innen kerülhet a memóriába. Egy regiszter tárolókapacitása általában 1 bájt vagy egy gépi szó.

Az utasításszámláló regiszter (PC) minden számítógépben megtalálható. Feladata a következő utasítás címének tárolása. A C 16-ban, a C Plus/4-ben az utasításszámláló regiszter 16 bites, hiszen a 64 kilobájt címzéséhez ennyi bitre van szükség. Ha a programban nincsenek elágazások, akkor az utasításszámláló tartalma minden utasítás végrehajtása után az őt követő utasítás címére mutat. Azonban, ha a program feltételes vagy feltétel nélküli ugró utasításokat hajt végre (IF...THEN...ELSE és GOTO elágazások), akkor az utasításszámláló új értéket kap az egyszerű továbbszámlálás helyett.

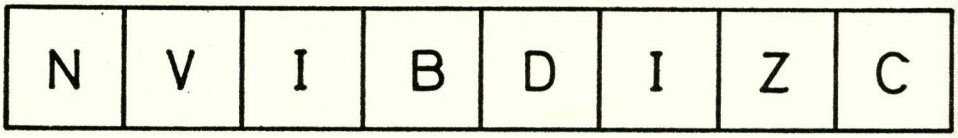
Az akkumulátor (A) nevű regiszter a mikroprocesszor központi regisztere. Az aritmetikai és a logikai műveletekben részt vevő adatokat, tehát az operandusokat, valamint a műveletek eredményeit a mikroprocesszor az akkumulátorban helyezi el, és a műveleteket is ott végzi el. Az adatforgalom egyes lépései:



A C 16 és a C Plus/4 két indexregisztert is tartalmaz: az X regisztert és az Y regisztert. Ezek szintén részt vesznek az adatforgalomban olyan számlálóként, amelyek például táblázatok kezelésekor az egyes táblaelemek sorszámait, indexeit tárolják.
A veremregisztert az alprogramok olyan átmeneti tárolóhely kezeléséhez használják, ahol a program futása során a főprogram legfontosabb jellemzőit ideiglenesen tárolni lehet.
Az állapotregiszter bitjei az utoljára végrehajtott utasítás eredményéről adnak felvilágosítást. Egyúttal kapcsolóként szolgálnak a feltételes utasításokhoz. Sorrendben ezek a bitek:

C átvitel, carry
Z nulla, zéró
I megszakítás, interrupt
D decimális művelet
B megszakítás, break
V túlcsordulás, overflow
N negatív.

A kapcsoló bitek ismerete a gépi kódú programot készítőknek igen lényeges. Az állapotregiszter bitjei:



7. bit 0. bit

A mikroprocesszor második fontos egysége a műveletvégző egység, más néven az aritmetikai-logikai egység (ALU). Alkalmas a következő feladatok elvégzésére:
— bináris összeadás
— Boole algebrai műveletek
— komplemens képzés
— adatok eltolása, léptetése.
Minden adatkezelési művelet felbontható ezekre az elemi műveletekre.
A mikroprocesszor harmadik fontos egysége a vezérlő egység (CU). Ez értelmezi az utasításokat. Végrehajtásuk céljából összehangolja a számítógép valamennyi egységének a működését úgy, hogy a részegységek működése a program futásának megfelelő sorrendben és időben következzenek be.
Gondoskodik arról, hogy az adatok a megfelelő helyre kerüljenek. A vezérlő egység indítja az egyes áramkörök működését.
A központi tár, más néven a memória tárolja a gép működéséhez szükséges információk kódjait. A C 16 és a C Plusz/4 ún. bájt szervezésű gép, tehát a tárban egy-egy címezhető bájtban, más szóval rekeszben találhatók a kódok.

A kódok csoportosítása:
— utasításkód
— számadat
— címadat
— definiálatlan érték.
Az utasításkódból „tudja" a gép, hogy milyen műveletet kell elvégeznie; a címből, hogy melyik adattal és hogy hová tegye az eredményt.
Az egyes utasítások és az adatok hossza különböző lehet. Az egymást követő bájtokon a négyféle kód váltakozva is előfordulhat, a gépi kódú programot készítő programozó egyéni elgondolása szerint. A kódok sorrendje azonban egy adott programon belül szigorúan kötött.
Egy 8 bites tárrekeszben 256 féle bitkombináció alakítható ki. Ezzel — ha címadatot képezünk — a 64 kilobájt nem címezhető. Ha azonban 2 bájtot használunk erre a célra, akkor $256 \times 256 = 65536$ címezhető memóriahelyünk van. Úgy is mondhatjuk, hogy a memóriában 256 darab 256 bájtos ún. lap van, és a gépet lapszervezésűnek nevezzük. A COMMODORE gépekben a legelső, az ún. nullás lap megkülönböztetett szerepet kapott. Címzésére külön utasításokat is kidolgoztak.
Az információ áramlására szolgáló vezetékek a buszok. A C 16 és a C Plus/4 gépekben a 16 bites utasításszámláló regiszter (PC) vagy címregiszter 16 vezetékkel, a címbusszal kapcsolódik a központi egységhez. Az adatbuszon 1 — 1 bájt 8 bitje halad együtt, ezért az adatbusz 8 bites.

1.1.2.1. A tanulók értsék, hogyan épül fel a számítógép a hardver részegységekből! Értsék a mikroprocesszor fő részeinek a feladatát! A hexadecimális számábrázolás ismeretében decimálisan adott memóriacímeket alakítsanak át hexadecimálissá! Végezzenek el konverziót a fordított irányban is!

1.1.3. A központi egység és a tár együttműködése. A gépi ciklus

A számítógép mint rendszer akkor működik helyesen, ha a központi egység mindig hibátlanul szervezi meg a részegységek működését, az utasítások és az adatok áramlását. Ennek alapja a vezérlő egység és a központi tár összehangolt munkavégzése.
Ha műveletvégzés közben a memóriából kell adatot kiolvasni, akkor ezt tárhoz fordulásnak nevezzük. Minden ilyen esemény egy alapidőtartamot jelent, amit gépi ciklusnak nevezünk. Értéke az adott géptípustól függ.
A gépi ciklus kisebb egységekre, az ún. ütemekre bontható. Ennek magyarázata az, hogy az egyes gépi utasítások a végrehajtási időben különböznek egymástól. Például az akkumulátor tartalmának 1 bittel balra tolása (2-vel való szorzása) kevesebb időt igényel, mintha például az akkumulátor tartalmát egy adott tárcím tartalmával hasonlítjuk össze.
Az ütemeket ún. órajel időtartamokra bontjuk. Az órajelet a rendszer oszcillátor jeléből állítják elő, amely a rendszer legkisebb időegysége.

Az alap időegység tehát hardver szempontból az órajel, programozási szempontból pedig a gépi ciklus.

Elemezzük a gép működését 2 szám összeadása közben! Az egyes programlépéseket a gépi kódú programozásban megszokott módon jelöljük.

$5000 LDA $7000 A gép utasításszámlálója a $5000-es címre mutatott az utasítás végrehajtása előtt. Innen kiolvasta az LDA utasítás kódját, az $AD értéket. Ez az utasítás 3 bájtos, tehát az utasításkód értelmezése után az utasításszámláló tartalma $5003 lesz.

Az utasítás végrehajtása a $7000 cím kiválasztásával és kiolvasásával folytatódik. Az adatbuszon a $7000-es cím tartalma az (A) akkumulátorba kerül. Az állapotregiszter bitjei új értéket vehetnek fel. Ha a $7000 cím tartalma nulla, akkor az állapotregiszterben a Z bit 1-re változik. Ha pedig 127-nél nagyobb lenne, akkor az N bit változna 1-re.

$5003 ADC #$01 A gép értelmezi az utasításkódot. Az utasítás 2 bájt hosszú (1 bájt az utasításkód, és mögötte 1 bájt az adat), ezért az utasításszámláló az utasítás végrehajtása után a $5005 címre fog mutatni.

Az utasítás a $5004-es memóriacím tartalmát hozzáadja az átvitelbittel együtt az akkumulátor tartalmához (amit az előző utasítás a $7000-es címről olvasott ki).

A művelet végrehajtása során túlcsordulás keletkezhet. Az átviteli bit az állapotregiszterben 0-ról 1-re változhat.

$5005... A program végrehajtása itt folytatódik.

Vannak olyan utasítások, amelyek nem fordulnak a központi tárhoz. Ha például regiszterek között végzünk műveletet, akkor a tárhoz fordulás időigénye, üteme kimarad. Ezekben az ütemekben más, hasznos tevékenység is elvégezhető, pl. a képernyő frissítése.

1.1.3.1. Ebben a fejezetben igen sok az új ismeret. A tanuló értse és jegyezze meg, hogy a gép részegységei összehangoltan működnek. A helyes működést az órajel szerint a gép maga szervezi meg.

1.1.4. A perifériák kapcsolódása, kezelése

A számítógéphez kapcsolt külső készülékeket perifériáknak nevezzük. Segítségükkel lehet adatot be- és kivinni, mozgatni, tárolni.

A C 16 és a C Plus/4-es gépek és a perifériák összekapcsolásának egyszerű eszköze az I/0 port. A portok (kikötő, kapu) általában véve lehetnek egyirányúak (bemeneti vagy kimeneti portok), de lehetnek kétirányúak is.

A porthoz érkező jelet egy áramkör megőrzi, s onnan lekérdezhető akkor is, ha a porton már nem bemeneti, hanem kimeneti lesz az adatforgalom iránya.

A belső adatbuszra kötött kimeneti port az adatkiviteli utasításkor megőrzi az adatbuszon érkező jelet mindaddig, amíg a periféria át tudja venni az adatot.

A perifériák kijelölésére, címzésére két megoldás terjedt el:

1. virtuális tárrekeszként kezelt perifériák esetében a be- és kimenetek címei erre a célra fenntartott tárcímek. A perifériák (pl. a képernyő) úgy kezelhetők, mint a tárrekeszek.
2. a tártól függetlenül kezelt perifériák címzésére speciális utasítások szolgálnak. Ezek írják le az adatátvitel irányát, a perifériát azonosító címrészt és a parancsot.

A C16 és a C Plus/4 gépekben speciális feladatokat lát el a TED csip. Ez egy nagy bonyolultságú integrált áramköri elem, ún. BOÁK (berendezés orientált áramkör). A központi egység processzorától sok funkciót átvállal. Vezérlése 32 valódi és 2 látszólagos regiszter (memóriacím) programozásával történik.

1.1.4.1. A számítógéphez a felhasználói porton át lehet perifériákat csatolni. Speciális illesztőkkel a tanulók is készíthetnek saját perifériákat (lásd a sorozat más segédleteit és útmutatóit).

1.2. A számítógép szoftver elemei teszik lehetővé a gép programozását

1.2.1. A BASIC interpreter működése

A számítógép kényelmes kezelését, programozhatóságát a memóriában levő gépi kódú programok biztosítják.

Ennek főbb részei: — képernyőszerkesztő
— BASIC interpreter
— perifériakezelő-rutinok.

A képernyőn a kurzorvezérlő billentyűkkel ún. full-screen üzemmódban dolgozhatunk, azaz a képernyő bármely pontját elérhetjük a nyíl billentyűkkel.
A BASIC szövegszerkesztő segítségével lehet a programsorokat beírni, listázni, törölni, módosítani.
Az elkészített programot a BASIC interpreter futtató rutinjaival működtetjük.
A gép bekapcsolása után egy áramkör addig várakoztatja a processzort, amíg valamennyi alkatrészen stabilizálódik a tápfeszültség. Ha ez kialakult, akkor a processzor kiolvassa a $FFFC—$FFFD címről a RESET vektort. Ez átirányítja a vezérlést a $FFF6 címre, ahol elindul a ROM ellenőrzése, bekapcsolása. A RESET rutin a $F2A4 címen indul. Inicializálja a memóriában a ROM-ot, az I/O csipet, majd a RAM-ot, a KERNAL vektorokat, a video területet, s végül a BASIC-be ugrik.
A gép működése a $8000-es címen folytatódik. Az interpreter beállítja a működéséhez szükséges memóriavektorokat. Ellenőrzi, hogy a külső csatlakozón keresztül külső ROM-ot dugaszoltak-e a gépbe.
A sikeres inicializálás végén a gép a képernyőn kiírja a bekapcsolás végét jelző COMMODORE... feliratot. Legvégül megjelnik a READY felirat.
Figyeli a gép azt is, hogy a bekapcsolás során a STOP gombot lenyomva tartjuk-e.

Ha igen, akkor a végrehajtást már a RESET rutin átirányítja a MONITOR bekapcsolására.

1.2.1.1. Érdemes megjegyezni, hogyha a gép használata közben a RESET gombot és a STOP billentyűt egy időben nyomjuk meg, akkor megmenthetjük a tárban levő BASIC programunkat is. Igaz ugyan, hogy a MONITOR fog bejelentkezni, de X és a RETURN billentyű lenyomása után a BASIC program mutatói változatlanok maradnak. A program tehát listázható és futtatható.

1.2.2. Az interpreter működése

Az interpreter, más néven tolmácsprogram működése ún. BASIC vektorok segítségével történik. Ezekben vannak azok a legfontosabb tárcímek, amelyek a BASIC interpreter működését irányítják.
Az interpreter a begépelt sort átveszi, és a benne levő BASIC alapszavakat egybájtos kódokká tömöríti. Ez az ún. tokenizálás. Fordítottja a detokenizálás, amelyre a listázás során van szükség.
Az interpreter a tokenizált sort befűzi a meglevő programsorok közé úgy, hogy a szigorúan növekvő sorszámozás megmaradjon. Ismert tény, hogy azonos sorszám esetén a legutoljára beírt sor az érvényes.
Szerkesztés közben a programleíró bájtok, a pointerek tartalma is változik. Ez okozza, hogyha pl. programpróba közben egy sort módosítunk, akkor az addig használt változók aktuális értékét elveszítjük.
A futtatás során az interpreter ciklikusan veszi sorról sorra a programot. Valamennyi BASIC alapszó feldolgozását egy-egy gépi kódú rutin végzi. Ezek gyakran össze is kapcsolódnak, és egymással kommunikálhatnak is. Kitüntetett kommunikációs terület a memória legelső 256 bájtja, a már korábban említett nullás lap. Itt vannak a legfontosabb rendszerváltozók, pointerek. Módosításukkal érdekes hatások érhetők el. Erről később lesz szó.

1.2.2.1. A tanuló ismerje, hogy a BASIC programot a belső szoftverével lehet a gépbe beírni, futtatni.
A tanár ismerje az interpreter rendeltetését és az interpreter ciklus fogalmát!

1.3. A memória szervezése

1.3.1. A memória felosztása. A tár szervezése

A központi processzornak 8 adatvezetéke és 16 címvezetéke van. Ezzel egyszerre 65536 memóriabájtot tud címezni. Ebből a 64 kilobájtból a C 16 és a C 116 gépben az alsó 16 kilobájton RAM, tehát olvasható/írható memória van. A felső 32 kilobájton ROM, tehát csak olvasható memória van. A C Plus/4 gépen a 32 kilobájt mellett a teljes címtartományban 64 kilobájt RAM is van. Ezt a gép szinte teljesen ki tudja használni, s ebből adódik a bekapcsolás vagy RESET után olvasható „60671 bytes free" szöveg a képernyő felső részén.
A teljes címtartomány egy kis részén nem RAM-ot vagy ROM-ot, hanem perifériabájtokat találunk. Ez az ún. I/0 címtartomány $FD00-tól $FDDF-ig tart.
Az $FDDØ—$FDDF címek átírásával lehet a gépbe hátul behelyezett programmodulokat bekapcsolni. Használatukhoz a rendszer alapos ismerete szükséges. Ha ezekhez a címekhez meggondolatlanul hozzányúlunk, ez a gép „kiadását" okozhatja.

Memóriatérkép

Cím	Tartalom
0 — 2047	Rendszerváltozók, pointerek helye
2048 — 3071	Színmemória: a képernyőre kerülő karakterek színét, fényerejét, villogását határozza meg.
3072 — 4095	Képernyő memória: a képernyőre kerülő karakterek kódját tartalmazza. A memóriában 40×25 bájt a képernyő területe.
4096 — 16383	BASIC munkaterület: a BASIC program és az adatok elhelyezésére szolgál.

A nagy felbontású grafika bekapcsolása után a BASIC munkaterület mérete lecsökken. Ekkor a megfelelő címek kiosztása így alakul:

6144—7167	A nagy felbontású képernyő pontjainak fényerőleírását tartalmazza.
7168—8191	A nagy felbontású képernyő pontjainak színleírását tartalmazza.
8192—16383	A nagy felbontású képernyő pontjait adja meg 320×200 vagy 160×200-as méretben.

A memória felső 32 kilobájtjában a következő tartalmat találjuk:

32768—53247	BASIC ROM, itt található többek között a BASIC szövegszerkesztő s az interpreter.

53248—55295	Karaktergenerátor ROM: ez tartalmazza 8 bájtonként az egyes karaktereknek a képernyőre kerülő formáját. Átírására, új karakterek definiálására külön programot lehet írni.
55296—65535	KERNAL ROM: a gép működéséhez nélkülözhetetlen, alapvető gépi kódú rutinok gyűjteménye.
64768—65529	I/O RAM: az input/output műveletek végzéséhez szükséges terület.

A memória legvégén, a $FF81—$FFF5 címeken egy lényeges terület található: a KERNAL rutinok ún. ugrótáblája. Segítségével érdekes trükköket alkothatunk.

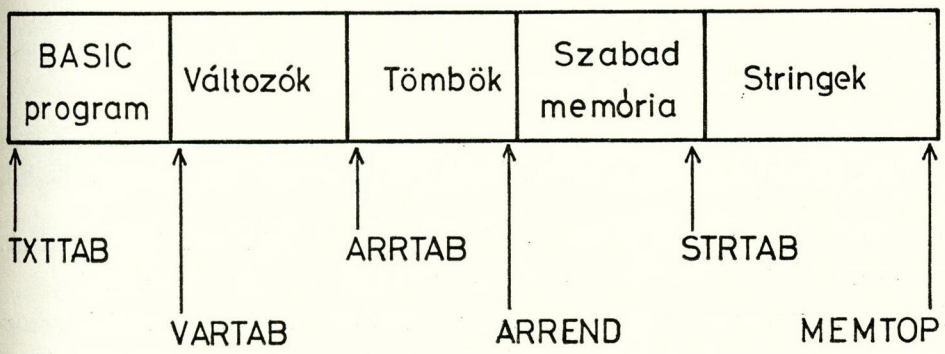
1.3.1.1. A tanuló értse, hogy a számítógép memóriája több egységre bontható. Legyen képes a különböző memóriacímeket különböző számrendszerekben kifejezni.
A tanár értse, hogy a gépben a programok végrehajtását magasan szervezett belső programrendszer segíti.

2. BASIC PROGRAMOZÓI FOGÁSOK

2.1. A BASIC nyelvet gyorsan el lehet sajátítani. Ahhoz azonban, hogy valóban jó és hatékony programot írjunk, még sok ismeretet kell elsajátítani.

2.1.1. A BASIC program elhelyezkedése a memóriában

A memória felosztásáról, a ROM és a RAM terület kiosztásáról más helyen már szó volt. A BASIC programoknak fenntartott terület a gép bekapcsolása után a $1000 címen kezdődik. Felső határa a gép kiépítettségétől függ, a C 16-on bővítő nélkül $3FFF, a C Plus/4-en $7FFF.
A COMMODORE gépeken BASIC munkaterület felosztása a következő:

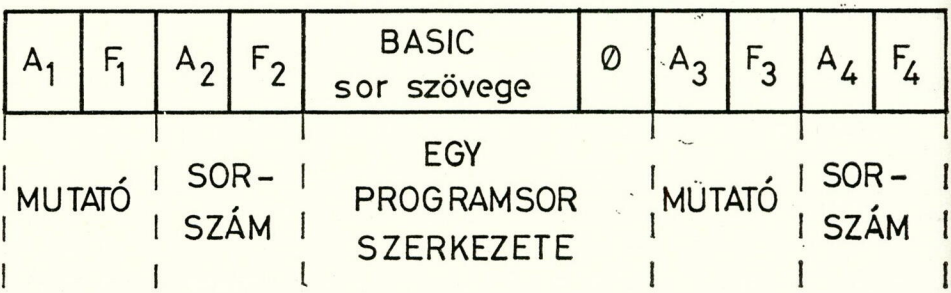


A rövidítések a memóriaterület egy elkülönített helyén levő 2 bájtos mutatók, ún. pointerek vagy vektorok. Tartalmuk azok a címek, amelyek a BASIC program szerkesztése és futttása közben az egyes területek kezdetét mutatják.

Mutató	Memóriacím	Tartalom
TXTTAB	43—44	A BASIC program szövegének kezdete.
VARTAB	45—46	A BASIC változók kezdete.
ARRTAB	47—48	A BASIC tömbök kezdete.
ARREND	49—50	A BASIC tömbök vége.
STRTAB	51—52	A stringterület alsó határa.
MEMTOP	55—56	A munkaterület felső vége.

Ha PRINT FRE(0) parancsot begépeljük, akkor az STRTAB—ARREND különbséget kapjuk meg.

A BASIC program begépelése után, a szerkesztés befejezésekor a dinamikus munkaterület üres. Ide helyezi majd el a változókat az interpreter.
Fontos tudnivaló, hogyan tároljuk a BASIC program egy sorát a memóriában:



A MUTATÓ 2 bájton a következő BASIC sor A_1 bájtjára, pontosabban annak a címére mutat. A cím kiszámítása:

$$\text{cím} = A_1 + 256 \times F_1$$

A SORSZÁM az adott BASIC sor címkéjét, sorszámát tartalmazza az előbb megismert

$$\text{sorszám} = A_2 + 256 \times F_2 \text{ módon.}$$

A BASIC sor szövegében a BASIC alapszavakat 1 bájtos ún. tokenek formájában találjuk. Például a PRINT alapszó tokenje $99. A szöveget egy bináris nulla bájt zárja.
A BASIC program utolsó sorának záró nullája után 2 újabb nulla bájtot találunk. Az interpreternek — ha nincs END alapszó — ez jelzi a program fizikai végét.

2.1.1.1. A tanulóértse, hogyan kapcsolódnak a BASIC utasítások sorokká és programmá! A tanár ismerje, hogy az interpreter a futtatás közben, valamint új sorok begépelésekor mutatók láncolatával dolgozik!

2.2.1. A BASIC program optimalizálása, gyorsítása

Az átlagos hosszúságú BASIC programok futási ideje többé-kevésbé elfogadható mértékű. Vannak azonban olyan feladatok, amikor a végrehajtás során a gép túlságosan lassúnak tűnik: sokat számol, vagy sokat kell várni egy-egy válaszra. Ilyenkor még nem feltétlenül a gépi kódú program írására kell gondolni. Elegendő lehet, ha jól átgondolt módon állítjuk össze a programunkat. A társzerkezet és a BASIC program memóriabeli elhelyezkedésének ismeretében apró fogásokkal lehet a program futását gyorsítani.

Ha a gyakran használt programrészeket a program elejére tesszük, időt nyerünk. Az ismétlődő részeket szubrutinba foglalva szintén a program elején érdemes elhelyezni, mert az interpreter a GOTO, a GOSUB tokenek feldolgozásánál a következő módon dolgozik: ha az aktuális sorszám a kisebb, akkor az interpreter tovább halad a sorokon. Ellenkező esetben a BASIC program elejéről kezdi a hivatkozott sor keresését. Nem létező sorszám esetén hibajelzést kapunk.
A gyakran használt változókat már a program legelső sorában érdemes aktivizálni. Különösen érvényes ez a ciklusváltozókra. Az interpreter elhelyezi a tárban a $2D-$2E mutatók által megjelölt címtől a változókat. A visszakeresés értelemszerűen kevesebb időt igényel, ha a kérdéses változót nem a végén, hanem az elején tartalmazza a változólista a memóriában.
Hasonló a helyzet a tömbökkel is. A dimenzionálás az előfordulás sorrendjében történik. Emiatt a tömbelemek elérési ideje is függ attól, hogy az adott tömb hanyadik a többi tömb között.
Sok idő mehet el a konverzióval is. Ha a stringből egész számot, egész számból valósat stb. igen gyakran kell képezni, akkor a felhasznált idő rohamosan nőni fog.
Gyakran használt mennyiségeket célszerű segédváltozókban tárolni. Bonyolult képletek számításánál sok időt nyerhetünk így.
Ciklusszervezésnél az előbb említett jelenségek halmozódhatnak, s a veszteség vagy a nyereség jelentős lesz. A ciklusok megszervezésénél — ha lehet — írjunk egyetlen sorba minél több utasítást!
Időt nyerünk akkor is, ha a POKE utasítás operandusai nem számkonstansok, hanem változók.
A PRINT utasítás által kiírt szövegkonstansokat célszerű a PRINT-ben elhelyezni.
A BASIC program forrásszövegét összenyomva, komprimálva szintén sok időt takaríthatunk meg. A C 64 HELP+ programjával ez a következő módon érhető el:
— kapcsoljuk be a C 64-et
— töltsük be és indítsuk el a HELP+ programot
— POKE 43,1: POKE 44,10
— LOAD"PR1", 8
— C
— SAVE "TÖMÖRPR1", 8
— kapcsoljuk be a C Plus/4-et
— DLOAD"TÖMÖRPR1
— LIST
Tömörítés előtt és tömörítés után a PRINT FRE(0) paranccsal írassuk ki a szabad memória nagyságát! Jelentős számú szabad bájthoz jutottunk. Ez különösen a bővítő nélküli C 16 használata során lehet előnyös. A tömörítés a program futási idejét is lerövidíti.
2'2.1.1. A foglalkozáson készítsünk mintaprogramot, amelyben nincsenek meg az előbb felsorolt tulajdonságok!
A programot átszerkesztve jelentős időnyereséghez jutunk.

2.2. Fogások a BASIC mutatókkal

2.2.1. BASIC programok láncolása

Gyakran előfordul, hogy egy programot több részletben készítünk el. Az elkészült szegmenseket össze tudjuk fűzni egységes programmá. Ez akkor sikerül, ha az egymás után betöltött részek sorszámai szigorúan növekvő sorrendben követik egymást. Az egyes lépések, ha az 1. program már a memóriában van:

— PRINT PEEK(43) PEEK (44) PEEK(45)

Ha a 3. szám 2, vagy 2-nél nagyobb, akkor ezt írjuk:

— POKE 43, PEEK(45) —2: POKE 44, PEEK(46).

Ha pedig 0 vagy 1, akkor ezt:

— POKE 43, PEEK(45)+254: POKE44, PEEK(46)—1

— LOAD"2. program", 8

Ezután az első pontban kiolvasott első 2 számot POKE-kal beírjuk a 43-as és a 44-es címre:

— POKE 43,...: POKE 44,...

LIST hatására a két összefűzött programrész együtt listázható.

Az eljárás az első ponttól ismételhető.

2.2.2. Törölt program aktivizálása a memóriában

A NEW parancs a BASIC terület pointereit alaphelyzetbe állítja. Mivel ilyenkor maga a programszöveg nem törlődik, ezért lehetőségünk van a pointerek visszaállítására. Egy gépi kódú rutinnal ez a következő módon lehetséges:

MONITOR

```
2
110:    00D8
120:    00D8

                                    .OPT P4
                                    *=   $00D8
                              ;
                              ; ************************
                              ;  POINTEREK VISSZAALLITASA
                              ;  NEW UTAN
                              ; ************************
200:    00D8 A9 01                  LDA  #$01
210:    00DA A8                     TAY
220:    00DB 91 2B                  STA  ($2B),Y
230:    00DD 20 18 88               JSR  $8818
240:    00E0 20 4B 88               JSR  $884B
250:    00E3 20 93 8A               JSR  $8A93
260:    00E6 4C 03 87               JMP  $8703
```

BASIC-ben elhelyezve ez így írható:

```
10 DATA A9,01,A8,91,2B,20,18,08
20 DATA 20,4B,88,20,93,8A,4C,03
30 DATA 87,96
40 RESTORE 10
50 FOR I=216 TO 232
60 READ X$
70 POKE I,DEC(X$)
80 NEXT I
90 REM AKTIVIZALAS:  SYS 216
```

A rutin aktivizálása: SYS 216

2.2.2.1. Az útmutatóhoz tartozó szoftveranyagban ehhez a témakörhöz tartozó programokat is talál. A foglalkozáson alkalmazzák és elemezzék ezeket!

2.3. A C 16 és a C Plus/4 gépekben a beépített MONITOR segítségével is módosíthatjuk a BASIC nyelven írt programjainkat.

2.3.1. A BASIC program mutatóinak átírása MONITOR segítségével

A MONITOR M funkcióját felhasználhatjuk a BASIC program mutatóinak az átírására.
Kövessük végig a bemutatott példát!

— kapcsoljuk be a gépet
— 10 A=1
  20 B=2
— LIST
— MONITOR
— M
  kiíródik a memória tartalma hexadecimális formában. Egy félbájt =4 bit, ezért az ott tárolható érték $0—$F közé esik. Tehát, ha például egy memóriabájtban 241-et találunk decimálisan, akkor a képernyőre F1-et jelez ki az M funkció.
  A $002B címen 01-et, a $002C címen 10-et látunk. Ezeket a számokat kiolvasva és felcserélve 1001-et kapunk. Ez azt is jelenti, hogy a $2B—$2C pointerek a $1001 memóriacímet jelölik a BASIC programszöveg első bájtjaként. A nyilakkal a kurzort tetszőleges helyre vezethetjük. Így pl. a $002B cím tartalmát 01-ről 09-re cserélve, a RETURN billentyűt lenyomva a BASIC programszöveg kezdetének mutatóját $1001-től $1009-re állítjuk át.
— vezessük a kurzort a képernyő aljára!
— X
— LIST hatására az eredeti BASIC programból csak a 2. sor jelenik meg.

Ha a mutatókat visszaállítjuk az eredeti értékekre, akkor a LIST után újra láthatjuk a program 10-es sorát is.
MONITOR-ban írjuk át a $100D cím tartalmát 1-gyel nagyobbra! Az X és LIST gépelése után a program listáján a 20-as sor B-je C-re változott.
2.3.1.1. A most megismert módon módosítsunk kis mintaprogramokat!

2.3.2. A gépi kódú program és a BASIC program kapcsolata

A BASIC nyelvű programozást megismerve a gépi kódú programozásra nincs szükség. Később jövünk rá, hogy a különböző bonyolultabb fogásokat igazán gépi kódban lehet megírni. Ugrásszerűen megnövelhető a program sebessége. Lecsökkenthető a memóriaigény. A játékprogramban gyorsabban fog mozogni egy pont, folyamatosnak tűnik majd a mozgás. A sebesség növelése természetesen másfajta alkalmazásban is előnyös lehet. Egy perifériát vezérlő programban nem lehet mellékes, hogy a perifériát milyen időközönként, milyen sebességgel tudjuk lekérdezni vagy oda milyen gyakran tudunk jeleket kiküldeni.
Gyakran előfordul, hogy a BASIC-ben és a gépi kódban írt programrészeket összekapcsoljuk, ilyenkor több probléma merülhet fel.
Egyik lényeges kérdés, hogy hol találunk alkalmas helyet a memóriában a gépi kódú program számára. A BASIC program elhelyezését ugyanis az interpreter elvégzi, de a gépi kódú rutinnak mi keresünk helyet.
A részletes memóriatérképet tanulmányozva a memóriában több helyen fedezünk fel üres bájtokat. Attól függően, hogy mekkora a betöltendő gépi kódú program, különböző megoldások közül választhatunk. Üres címek a memóriában, ha egy-egy perifériát éppen nem használunk:

$0332—$03F2	Szalagpuffer
$03F7—$0436	RS—232 puffer
$07B1—$07CC	Szalag munkaterület
$07CD—$07D8	RS—232 munkaterület
$1000—$7FFF	BASIC munkaterület BASIC program nélkül

Szokásos módszer az is, hogy a BASIC programban egy olyan sort helyezünk el, amelyben a REM után annyi „üres" bájtot hagyunk, amennyi a mi gépi kódú rutinunk számára szükséges.
Szintén elterjedt megoldásnak számít, hogy a BASIC munkaterület rovására a stringterület felső határát „lejjebb" hozzuk. A BASIC program és a tár kapcsolatát leíró fejezetben az ARREND és az STRTAB közötti területet csökkentjük azáltal, hogy a $56—$57 című bájtokon tárolt MEMTOP érték helyébe a szükséges bájtszámmal kevesebb értéket írunk. Egy adott programban a RUN-nal való indítás után a legelső sorban helyezhetjük el ezeket a POKE utasításokat. Például, ha a $57 cím tartalmát 1-gyel csökkentjük, ezáltal 256 bájtot teszünk szabaddá.

Előfordul az is, hogy gépi kódú programot a BASIC program szövege előtt helyezünk el. Ekkor azonban a BASIC program mutatóit a program betöltése előtt parancsokkal kell átállítani, ami a használat szempontjából nem kényelmes.
A memóriaszervezés, valamint egy-egy saját program tárkezelése ismeretében más ötletes megoldások is elképzelhetők.

2.3.3. Gépi kódú program beírása MONITOR segítségével

Gépi kódú programokat a MONITOR segítségével is beírhatunk. Erre a célra a MONITOR M, D, A, S, V, G parancsait használjuk.

— MONITOR
— M 2000
— $2000-től kezdődő tárterületen most definiálatlan memóriatartalom van
— D 2000
hatására a tártartalom disassembler formában jelenik meg

```
— A 2000 LDA   #$00
         TAX
         STA   $0C00, X
         INX
         TXA
         SBC   #$FF
         BEQ   $200F
         JMP   $2000
         BRK
```

ezzel beírtunk egy gépi kódú programot a $2000-es memóriacímtől
— D 2000
megjelenik az általunk módosított tártartalom a megfelelő címekkel és a hexadecimális kiírással
— G 2000
a gépi kódú rutin lefutását a képernyőn megjelenő teljes karakterkészlet jelzi
— S „karakter" 01 2000 2020
ezzel a kazettás egységre írhatjuk a gépi kódú rutint
— a magnetofon számlálóját léptessük vissza a felvétel elejére!
— V „karakter"
begépelésére a gép ellenőrző olvasást végez
— kapcsoljuk ki, majd be a gépet!
— MONITOR
— L „karakter" 01
sorok hatására az előbb felvitt program betöltődik a memóriába, s a
— G 2000
sor hatására újra megjeleníthető a karakaterkészlet a képernyőn.

2.3.4. Gépi kódú program betöltése BASIC program futása közben

Gépi kódú rutint a BASIC programban a DATA utasítással kezdődő sorokban is elhelyezhetünk. A DATA után álló, 0—255 közötti számokat a gépi kódú rutin bájtjaiban kell betölteni. Természetesen nemcsak decimális, hanem hexadecimális számrendszerben ábrázolt számokat is elhelyezhetünk a DATA utasításban.

2.3.4.1. Gyakoroljuk a gépi kódok betöltését az alábbi programok felhasználásával!

```
10 INPUT "KEZDOCIM=";KC
20 INPUT "VEGCIM=";VC
25 FOR I=KC TO VC
30 READ X
35 POKE I,X
40 NEXT I
50 DATA 169,3,162, ...
```

```
10 INPUT "KEZDOCIM=";KC
20 INPUT "VEGCIM=";VC
30 FOR I=KC TO VC
40 READ X$
50 POKE I,DEC(X$)
60 NEXT I
70 DATA A9,03,A2, ...
```

Egy-egy gépi kódú rutin betöltéséhez előre megadhatjuk azt a számot, amit a kódok decimális értékének összegzésével kapunk. A betöltés akkor hibátlan, ha a beolvasott kódok összege megegyezik a kontrollösszeggel.

2.3.5. Gépi kódú program DATA sorokká való átírása

Az előző fejezetben említett feladatot fordított irányban is el lehet végezni. Gyakran előfordul, hogy a gépi kódú programozásban gyakorlott programozó a MONITOR alatt beírt gépi kódú rutinját BASIC DATA sorokká akarja konvertálni. A BASIC programok szerkezetének és a memóriatérképnek az ismeretében ez nem is olyan nehéz. Írhatunk olyan BASIC programot, amely a következőképpen működik:

— lekérdezi a gépi kódú rutin kezdő és végcímét
— a programnak egy REM kezdetű, és általunk kellő hosszúságúra beállított sorában a REM tokenjét ($8F) a DATA tokenjére ($83) cseréli.
— ciklusban a DATA tokenje után teszi a gépi kódú rutin bájtjainak tartalmát.

Célszerű a programot úgy megírni, hogy a legelső sora legyen a REM... kezdetű sor.

Ekkor egy DELETE 11-parancs hatására a memóriában csak a nekünk szükséges sor marad.
A mellékelt szoftver anyagban szintén található DATÁZÓ program. Töltsék be, használják ezt a programot is!

2.3.6. A SYS utasítás használata

Egy gépi kódú programot elkészítve kérdés az, hogyan lehet ezt a BASIC programból aktivizálni.
A BASIC nyelv lehetőséget teremt gépi kódú utasítássorozat indítására. A SYS alapszó mögé a hívott gépi kódú rutin decimális belépési címét kell írni. Mellette — vesszővel elválasztva — további paraméterek helyezhetők el.
A SYS utasítás előtt a memória tárcímeit POKE utasításokkal is feltölthetjük. Így értékeket adhatunk át az A, X, Y regisztereknek, valamint az állapotregiszternek is:

MEMÓRIACÍM:		REGISZTER:
$07F2	2034	A
$07F3	2035	X
$07F4	2036	Y
$07F5	2037	PSW állapotregiszter.

Használatát parancs módban ki is próbálhatjuk:

```
10 POKE 2035,6        :REM VOL 6
20 SYS DEC("B8C0")
30 SOUND 1,600,20
40 POKE 2035,3        :REM VOL 3
50 SOUND 1,600,20
60 SYS DEC("B8C0")
```

Lehet olyan megoldás is, hogy a memória valamely szabad címére helyezzük el a bemenő paramétert. Például az F változó értékét, amely 0—65535 közé eshet, akarjuk átadni a gépi kódú, $0340-en kezdődő gépi kódú rutinnak. A memóriában szabad cím található az adott feladatban a $F0—$F1 címeken.
A programrészlet:

```
100 F=5000
110 FB=INT(F/256)
120 FA=F-FB*256
130 POKE 240,FA
140 POKE 241,FB
150 SYS DEC("0340")
160 REM  FOLYTATAS
```

A SYS cím, paraméter típusú utasítás feldolgozásához a saját gépi kódú programban kell felkészülni. Legyen a konkrét példa egy olyan szubrutin, amely a decimális 10000-ről indul, és a paramétere vagy 1, vagy 0 lehet. Hibás paraméter esetén hibajelzést kell adni.
Hívása: SYS 10000,0.
A gépi kódú rutin elejét közöljük:

```
2
30:     2710                    .OPT P4
40:     2710                    *=   $2710
                        ;
                        ; ********************
                        ;  PARAMETER ATADASA
70:     2710
                        ;  SYS CIM,PARAMETER
                        ;
                        ;  A PARAMETER NULLA
                        ;  VAGY 1 LEHET.
                        ; ********************
130:    2710 20 D8 9D           JSR  $9DD8     ;GETCOM+GETBYT
140:    2713 E0 00              CPX  #$00      ;A PARAMETER NULLA-E
150:    2715 F0 69              BEQ  $2780     ;ELAGAZAS CIM1-RE
160:    2717 E0 01              CPX  #$01      ;A PARAMETER EGYES-E
170:    2719 F0 75              BEQ  $2790     ;ELAGAZAS CIM2-RE
180:    271B 4C A9 94           JMP  $94A9     ;EGYEBKENT SYNTAX ERROR
```

2.3.6.1. Gyakoroltassuk a SYS utasítás használatát! Írjuk be a közölt programrészeket!

2.3.7. Gépi kódú rutin aktivizálása más módon: az USR utasítás használata

A BASIC interpreter az USR utasítást is ismeri. Az USR mellé zárójelben egy 0—65535 közé eső szám írható. Ez lesz az USR utasítással hívott függvény paramétere. A függvényt gépi kódban kell megírni, és ennek a gépi kódú rutinnak a belépési pontját, kezdőcímét kell elhelyezni az $0501—$0502 címen.
A $0500 címen a JMP gépi kódú utasítás kódja van ($4C).

2.3.7.1. A tanulók legyenek tisztában a belépési pont fogalmával!
Példaként hajtsuk végre a PRINT USR(4) parancsot!
Az USR függvény használatával és a MONITOR segítségével ez így valósítható meg:

MONITOR

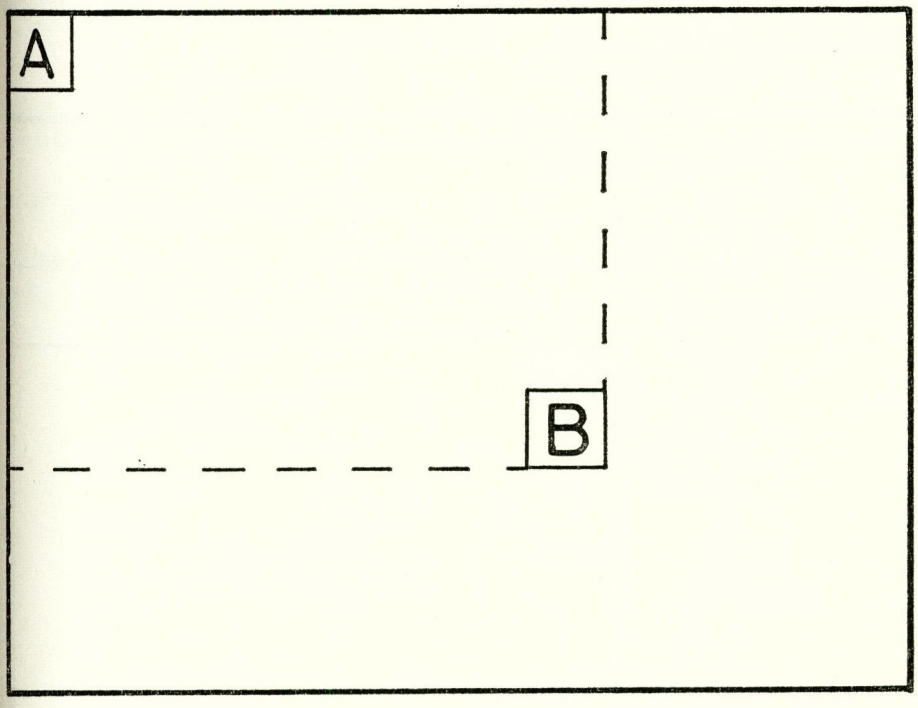
M 500
az $0501-re $E4-et, a $0502-re $A5-öt írjunk. Az SQR függvényt a BASIC ROM $A5E4 címen tartalmazza.

X

PRINT USR(4)
2 jelenik meg az SQR(4) =2 összefüggés alapján.
Az SQR függvényt a példa kedvéért használjuk.
Helyette bármilyen saját rutin használható.

2.3.8. Képernyőablakok előállítása, kezelése

Játékok és kvázigrafikus képernyők, de oktatóprogramok készítése közben is előfordulhat, hogy a képernyőt, a karakteres képernyőt több részre osztva akarjuk használni. Ez az ún. „ablakolás" problémája.
Az alábbiakban megmutatjuk, hogyan lehet a képernyő egy-egy téglalap alakú darabjának kijelölését, definiálását az ESC billentyű segítségével megoldani.
Jelöljük ki az ábrán mutatott ablakot!



Nyomjuk meg a HOME billentyűt! A kurzor a képernyő bal felső sarkába kerül. Most az ESC, majd a T billentyű következik! A kurzort vezessük jobbra és lefelé egy tetszőleges pontba! Nyomjuk meg az ESC és utána a B billentyűt! Ezek a képernyőn „fekete jelet", tehát olvasható karakterképet ugyan nem adnak, de ha most lenyomjuk a RETURN-t, akkor már használhatjuk is az így kijelölt részképernyőt.

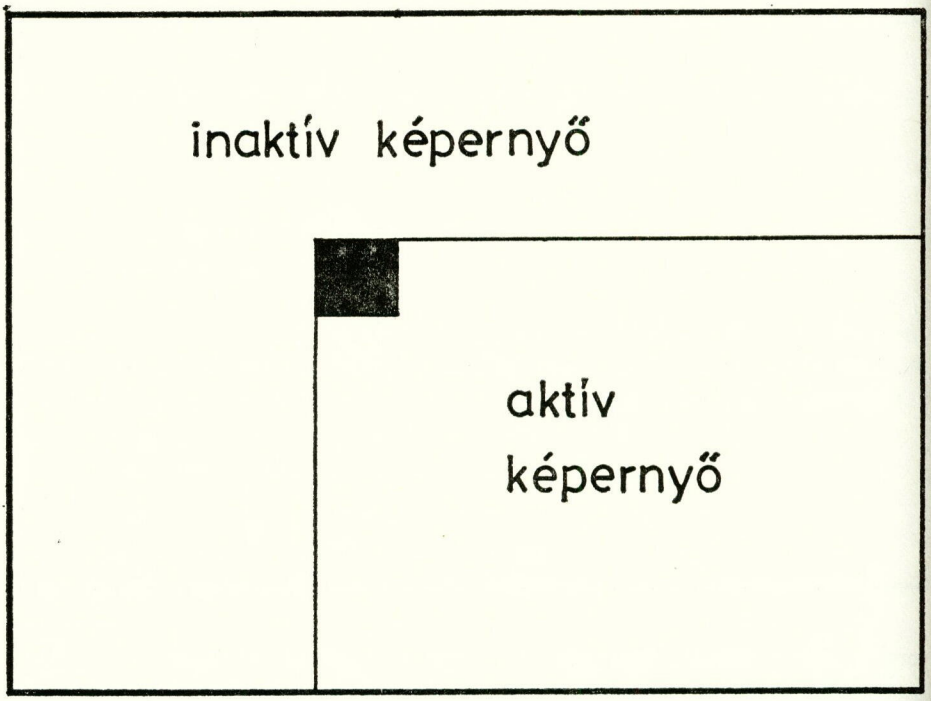
Ugyanez BASIC nyelven:

```
1 PRINTCHR$(147)CHR$(27)"T"
2 PRINTCHR$(17)CHR$(17)
3 PRINTCHR$(17)CHR$(17)
4 PRINTCHR$(17)CHR$(17)
5 FOR I=1 TO 39: PRINT"]";: NEXT I
6 PRINTCHR$(27)"B"
```

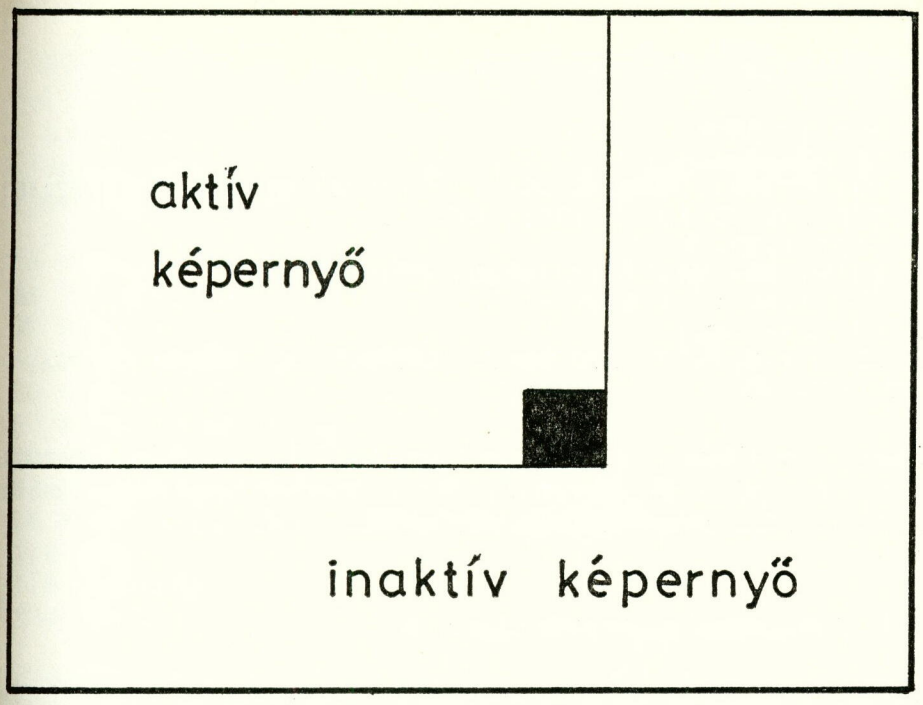
2.3.9. Képernyőablakok kijelölése POKE utasítással

Az előző pontban leírt ablakolás POKE utasításokkal is megvalósítható. A példát több változatban is kipróbálhatjuk.

1. Az aktív képernyő legyen a jobb alsó sarokban:
   POKE 2022, sor: POKE 2023, oszlop
   Az eredményt az ábra mutatja:

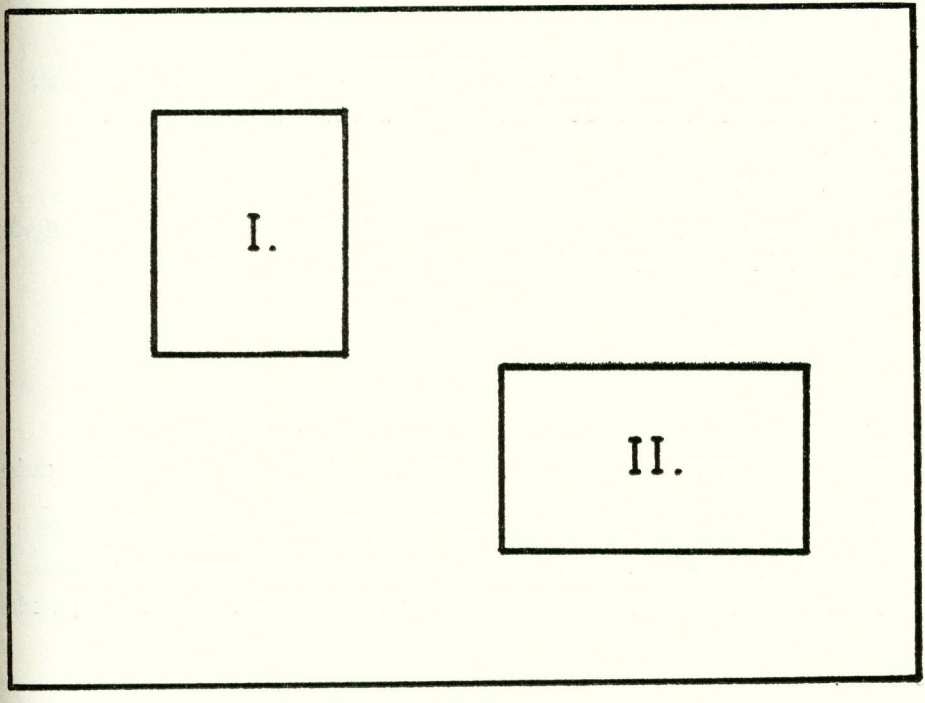


2. Az aktív képernyő a bal felső sarokban legyen:
   POKE 2021, sor: POKE 2024, oszlop



2.3.10. Kettő vagy több képernyőablak kezelése

Lehetőség van arra is, hogy nemcsak 1, hanem 2, vagy több ablakot jelöljünk ki a képernyőn. Végezzük el a következő feladatot: jelöljük ki az ábrán mutatott ablakokat!



A beállítást végző BASIC program sorai:

```
10 REM ABLAKOK
20 PRINTCHR$(147)
30 PRINT"S]]]]]QQQQ"CHR$(27)"T]]]]]]]QQQQQQ"CHR$(27)"B"
40 FOR I=1 TO 10: PRINTI: NEXT I
50 CHAR 1,20,20,""
60 PRINTCHR$(27)"T]]]]]QQQQQ"CHR$(27)"B"
70 FOR I=10 TO 1 STEP -1
80 PRINTI: NEXT I
90 CHAR 1,7,7,"": REM AZ 1. NEGYZET EGY BELSO PONTJA
100 GOTO 10
```

Kettőnél több ablakot is hasonló módon tudunk kijelölni. Fontos tudni, hogy ugyanabban a pillanatban csak egyetlen ablakot kezelhetünk. A többi tartalma ott van a képernyőn, de a hozzájuk tartozó terület nem aktív.

2.3.11. Felirat állandó megjelenítése a képernyőn

„Érdekes" képernyő készítésekor találkozunk azzal a problémával, hogyan lehet egy feliratot állandóan a képernyőn tartani, a képernyő görgetése nélkül.
Az ESC funkciók gyakoroltatása közben kipróbálhatjuk az alábbi megoldást:

```
100 SYS 55432      :REM ESC N
110 PRINT TAB(13)"***  COMMODORE  ***"
120 SYS 56926:     :REM ESC T
```

A képernyő legfelső sorában állandóan ott látható a *** C–16 *** felirat.
A feliratot a CLEAR/HOME billentyűvel letörölni nem lehet. Kikapcsolása a
SYS 55432
parancs kiadásával lehetséges.

2.3.11.1. Gyakoroljuk a 2.3.8.—2.3.11. pontokban leírt fogásokat, trükköket! Egyéni programkészítéssel, feladatok kitűzésével tegyük érdekessé az órai munkát!

2.3.12. A képernyőablakok színezése

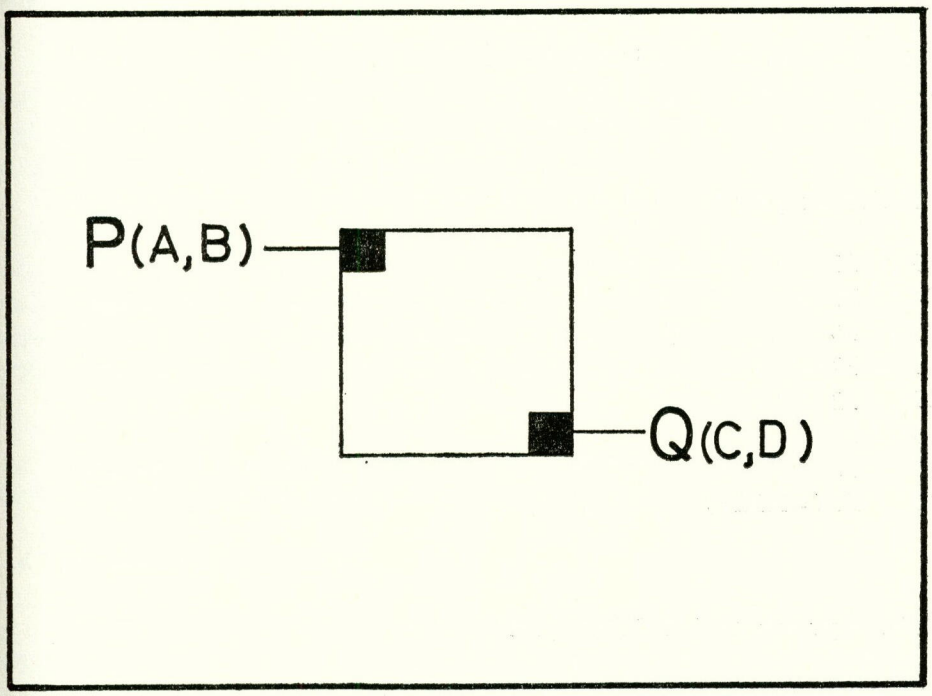
Ha már tudunk ablakokat definiálni, akkor próbáljuk meg a színezésüket is!
A képernyőablak színét meg tudjuk változtatni. Ezzel az ablak kezelése hatásosabb, mutatósabb lesz.
A színezés egyik lehetséges módja az, hogy átírjuk a színmemória megfelelő bájtjainak a tartalmát. A színmemória kezdete $0800, azaz decimálisan 2048. Minden egyes bájt a hozzá tartozó karakterkép színét, fényerejét, villogását tartalmazza, vezérli.

0—3. bit színkód
4—6. bit fényerő
7. bit villogás

Az alábbi programmal villogtatni tudjuk — ez fekete-fehér képernyőn is jól látszik — a képernyőablak tartalmát:

```
10 FOR I=1 TO 1000
20 PRINT"X";
30 NEXT I
40 A=10: B=8:          REM P(10,8)
50 C=20:D=18:          REM Q(20,18)
60 K=(A-1)*40+B+2047
70 FOR I=A TO C-1
80 FOR L=1 TO D-B
90 POKE K+L,128:       REM VILLOGAS KODJA
100 NEXT L
110 K=K+40
120 NEXT I
```

Az ablak helye:



2.3.12.1. A képernyőablakok kezelése látványos, jó programok írására ad lehetőséget A tanulók szívesen készítenek az ötleteik alapján ilyen programokat.

2.3.13. Vezérlőkarakterek a képernyő kezeléséhez

A képernyő tartalmának szerkesztéséhez hasznos összefoglalást adunk a fontosabb kurzormozgató és vezérlő kódokról:

CHR kód	Hatása:
17	kurzor le
145	kurzor fel
29	kurzor jobbra
157	kurzor balra
13	RETURN
141	SHIFT+RETURN
19	HOME
147	CLEAR
14	Kisbetűk, vagy SHIFT+nagybetűk
142	nagybetűk, vagy SHIFT+grafikus jelek
8	SHIFT+COMM tiltása
9	SHIFT+COMM engedélyezése
18	inverz kijelzés
146	nem inverz kijelzés
20	törlés
148	beszúrás

Funkcióbillentyűk:

133	F1
134	F3
135	F5
136	F7
137	F2
138	F4
139	F6
140	F8 (=HELP)

2.3.14. Fogások a DEL/INS billentyűvel

A képernyőn megjelenő input formájának vezérlése, alakítása többek között a DEL billentyűvel is befolyásolható. Ennek a billentyűnek a kódja CHR$(20).
Az érdekesebb alkalmazási lehetőségek közül bemutatunk egy rövid programot:

```
10 PRINTCHR$(147)
20 CHAR 1,9,23,"*"
30 CHAR 1,30,23,CHR$(95)
40 FOR I=29 TO 9 STEP -1
50 CHAR 1,1,23,CHR$(20)
60 NEXT I
70 CHAR 1,9,23,CHR$(95)
80 CHAR 1,9,23,""
90 CHAR 1,1,23,"B U M M !"
100 VOL 8: SOUND 3,600,9: VOL 0
110 CHAR 1,9,22,".": CHAR 1,8,24,"."
120 CHAR 1,9,23,".": CHAR 1,9,24,"."
130 CHAR 1,10,22,".": CHAR 1,10,24,"."
140 GOTO 10
```

A kis jel mozgatása kiinduló eleme lehet egy bonyolultabb játéknak. A balra mozgatás helyett jobbra is fordíthatjuk a jelet a 40-es sor FOR...NEXT ciklusának a megfordításával, valamint a CHR$(20)-nak CHR$(148)-ra cserélésével.

2.3.15. Kvázigrafikus lehetőségek rajzokhoz

A grafikus karaktereket karakteres üzemmódban használjuk. Nagy felbontású képernyőn is tudunk hasonló jeleket definiálni. Ez utóbbi rajzokat az SSHAPE utasítás segítségével stringváltozóban is eltárolhatjuk. Ezt a változót később a GSHAPE utasítással lehet a képernyő tetszőleges helyére visszatölteni.
A változó helyigénye 256 bájtnál több nem lehet. Ha az Ury: C–16... könyv 103. oldalán közölt példát begépeljük, akkor tapasztaljuk, hogy az ékezetes betűket tartalmazó A$, U$, O$ stringek hossza 28—28 bájt. A normál „A" betű kiírásához (1 bájtos!) 22 bájt hosszú stringre van szükség. Az elhatárolandó jelet a nagy felbontású képernyőn kell megtervezni, és megjeleníteni a megfelelő utasításokkal (CIRCLE, DRAW stb.). A string visszatöltése újabb grafikai lehetőséget ad játékok tervezéséhez.

2.4. A memóriában a megfelelő címek, vektorok átállításával a géphez kapcsolt perifériák kezelése is érdekessé tehető.

2.4.1. A kazettapuffer használata

A C 16 és a C Plus/4 gépeken a kazettás egység kezeléséhez szükséges pufferterület a $0332—$03F2 memóriacímeken található. A puffer pointere a $B6—$B7 vektorban van.
A gépi kódú szubrutinokat igen sokszor a kazettapufferben helyezzük el. Ennek természetesen az a feltétele, hogy ezzel egy időben ne legyen beolvasás a kazettáról!

Betöltés a kazettapufferbe:

```
10 DATA 162,0,189,0,208,157,0,116
20 DATA 189,0,209,157,0,117,202
30 DATA 208,241,96
40 RESTORE 10
50 FOR I=832 TO 849
60 READ A
70 POKE I,A
80 NEXT I
90 SYS 832
100 REM  FOLYTATAS
```

A kazettapuffer pointerét mi magunk is változtatni tudjuk. Ezzel elérhetjük, hogy a $B6—$B7 címek tartalma tetszés szerinti memóriaterületre mutasson.
Például:
POKE 182,3: POKE 183,247
hatására a szalagpuffert áthelyeztük a $03F7 címen kezdődő RS—232 pufferbe.

2.4.2. A kazettás magnetofon állapotának lekérdezése

A számítógép használata közben előfordul, hogy a DATASETTE magnetofont bekapcsolva felejtjük.
Üzemállapotáról a következő módon győződhetünk meg:

PRINT PEEK(1)	Jelentése:
200	Kikapcsolt állapot
192	PLAY/REW/FWD
208	REC+PLAY
216	Hibás jel

A motor bekapcsolását a $07FC bájt jelzi. A $0001 adatregiszterből a magnófunkcióról kapunk információt.
A magnetofon motorját POKE utasításokkal kapcsolgatni tudjuk. Nyomjuk le a PLAY gombot!

```
10 REM BEKAPCSOLAS:
20 POKE 2044,128: POKE 1,PEEK(1) OR 8
30 GETKEY X$
40 REM
50 REM KIKAPCSOLAS:
60 POKE 2044,4: POKE 1,PEEK(1) AND 247
70 GETKEY X$
80 GOTO 10
```

2.4.2.1. A 2.4.2. fejezetben közöltek alapján programok titkosítására is van lehetőségünk. Készíttessünk a tanulókkal olyan programot, amely csakis akkor indul el, ha a kazettás egység valamely gombját lenyomjuk! Használjuk hozzá ezt a sort:
1 WAIT 1,255,200
A program csak a REC+PLAY lenyomására indul el, ha első sora ez lesz:
1 WAIT 1,55.

2.4.3. Automatikus programindítás

A készen vásárolható programok között több olyan akad, amely a kazettáról való betöltés után azonnal el is indul. Van olyan is, amelyik a STOP billentyűvel sem állítható meg. Ilyenkor a kazettán egy kis betöltő programot helyezünk el a RUN-nal indítható BASIC program előtt. Ezt a gépi kódú programot kell betölteni, s ez gondoskodik arról, hogy a közvetlenül mögötte tárolt BASIC program a tárba kerüljön. Ha a BASIC program elején rendszerfunkciókat vezérlő POKE-okat helyezünk el, akkor hatástalanítani tudjuk a STOP billentyűt is.

2.4.3.1. A mellékelt szoftver anyagban szintén találunk AUTOSTART szubrutint. Töltsük be, és indítsuk el! A program lefutása után a kazettát hagyjuk a magnóban, vagy jegyezzük fel a számláló állását!
Az új programot a begépelés után közvetlenül a most felvett gépi kódú program után kell a szalagra venni!
Betöltéskor a szalagot a gépi kódú program kezdetéhez kell állítani.

2.4.4. Periféria ellenőrzése futtatás közben

Ha a program futtatása közben akarunk nyomtatni, de nincs hozzáfűzve a számítógéphez a nyomtató, akkor hibajelzést kapunk:
DEVICE NOT PRESENT ERROR
Ugyanez előfordulhat lemezegység használata közben is.
A hibát a BASIC program futása közben is lekezelhetjük:

```
100 OPEN 4,4
110 IF ST<>-128 THEN PRINT"NINCS BEKAPCSOLVA A NYOMTATO!": END
120 REM  FOLYTATAS
```

A lemezegység ellenőrzéséhez a
100 OPEN 8,8
utasítást kell megadni.

2.5. A kurzor kezelésére szintén tudunk módszereket bemutatni

2.5.1. *A kurzor pozícionálása és kioltása*

A képernyőn a kiírás helyét kényelmesen be lehet állítani CHAR utasítással. Más módon ugyanilyen hatást érhetünk el a kurzor pozícionálásával. A fizikai sor számát (1—25 között) a $CD, az oszlop sorszámát (1—40 között) a $CA memóriacím tartalmazza. Ezek a bájtok POKE utasítással átírhatók.
Például:

```
1 POKE 202,10
2 POKE 205,20
3 A=1: PRINT A
4 POKE 1319,13
5 POKE 239,1
6 REM   RUN-T A LEGALSO SORBAN ADJUNK!
```

(A hatás fokozható, ha a billentyűzetpufferbe beírjuk a RETURN kódját.)

RUN

A kurzor nem villog a RUN R betűjén, hanem a program folyamatosan újrafuttatja önmagát. Lásd a dinamikus billentyűzet-technika című fejezetet!
A kurzor kikapcsolásához a TED csip $ØC—$ØD regisztereit használjuk. A $FFØC cím jobb szélső 2 bitje és a $FFØD cím 8 bitje a képernyő 1000 bátjánál többet, 1023-at tud címezni. Így a kurzort a következő módon tudjuk kivinni a videomátrix területéről, azaz a képernyőről:

```
10 POKE 65292,PEEK(65292) OR 3
20 POKE 65293,255
30 GOTO 10
```

RUN

2.5.2. A lenyomott billentyű lekérdezése

A billentyűzet közvetlen lekérdezése gyorsabb programfutást tesz lehetővé. Ehhez a 198-as memóriacímet ($C6) kell PEEK-kel kiolvasni. A jelek kiolvasásához ad segítséget a következő táblázat:

T \ E	1	2	3	4	5	6	7	8	9	0
1	4	Z	S	E		5	R	D	6	A
2	F	T	X	7		G	8	B	H	C
3	V	9	I	J	0	M	K	O	N	U
4	P	L	↑		:	-	,	←	*	↓
5	→		=	+	/	1	HOME		2	;
6		Q								SPACE
7										
8										
9										
0	RETURN	£					7	3	W	DEL

Ha azt is tudni akarjuk, hogy a SHIFT, a COMMODORE vagy a CNTRL billentyű is lenyomott állapotban van-e, akkor ezt az 1347-es decimális cím tartalma jelzi:

PEEK (1347)	Jelentése:
1	SCHIFT
2	COMMODORE
3	SHIFT+COMMODORE
4	CNTRL
5	SHIFT+CNTRL
6	COMMODORE+CNTRL

2.5.2.1 Önálló programírással gyakorolhatjuk a billentyűk lekérdezését. Ehhez már az előző fejezetekben megismert trükköket is érdemes felhasználni.

2.5.3. A lenyomott billentyű ismétlése

Ha egy billentyűt lenyomva tartunk, akkor a képernyőn sorfolytonosan haladva sokszor egymás után megjelenik a megfelelő karakter képe.
Ha azonban a $Ø54Ø címre Ø-t írunk, akkor az ismétlő funkciót kikapcsolhatjuk.
Kikapcsolás: POKE 1344,Ø.
Bekapcsolás: POKE 1344,128.

2.5.4. INPUT utasítás kérdőjel nélkül

Az INPUT utasításhoz érve a program végrehajtása során a képernyőn kérdőjel jelenik meg. Kiírását elkerülhetjük a következő módon:

```
10 OPEN 3,Ø
20 PRINT "A= ";
30 INPUT#3,A
```

A megoldás azon alapul, hogy a képernyőt ezúttal nem outputra, hanem inputra nyitottuk meg.

2.5.5. Dinamikus billentyűzet-technika

Ha a GET vagy a GETKEY utasítást használva el akarjuk kerülni a billentyűk felesleges lenyomogatásából adódó kellemetlenségeket, akkor célszerű a
POKE 239,Ø
utasítást elhelyezni a GET előtt a programban. Ezen a címen van ugyanis az az érték, amely megmondja, hogy hány karakter van az 1319—1328 decimális címeken található billentyűzetpufferban. Nullát írva a 239-es címre a gép üresnek veszi a puffert.
Fordítva gondolkodva:
a pufferben elhelyezhetünk legfeljebb 10 darab karaktert, azaz 10 kódot. Ezzel érdekes hatást érhetünk el. Ezt mutatja az alábbi program:

```
10 PRINTCHR$(147)"NEW"
20 PRINTCHR$(17)CHR$(17)"LOAD"CHR$(34)"PROGRAMNEV"CHR$(34)",8"
30 POKE 1319,19       :REM HOME
40 POKE 1320,13       :REM RETURN
50 POKE 1321,13
60 POKE 1322,19
70 POKE 1323,82       :REM R
80 POKE 1324,117      :REM SHIFT U
90 POKE 1325,32
100 POKE 1326,13
110 POKE 239,8         :REM 8 DARAB KOD
```

Ha a programot RUN-nal elindítjuk, akkor mágneslemezről betölti és elindítja a megadott nevű programot. Ha nincs bekapcsolva a lemezegység, természetesen hibajelzést kapunk.

2.5.5.1. A dinamikus billentyűzet-technika lehetőséget ad programok folyamatos indítására. Célszerű ezt közös munkával ki is próbálni. Alakítsunk csoportokat, s a csoportok készítsenek egy-egy programot! Ezeket a programokat indítsuk ezzel a módszerrel!

2.6. A BASIC program szerkezetének ismeretében tegyük „ügyesebbé" a számítógépet! Növeljük meg a beépített interpreter tudását!

2.6.1. Önmagát futtatás közben módosító program

A dinamikus billentyűzet-technika segítségével olyan BASIC programot tudunk készíteni, amely a futása közben önmagát is tudja módosítani, bővíteni:

```
10 PRINT"IRD BE X TETSZOLEGES FUGGVENYET!"
20 INPUT "(PL. SIN(X)+1) :";X$
30 POKE 239,3
40 POKE 1319,19
50 POKE 1320,13
60 POKE 1321,13
70 PRINTCHR$(147)"150 DEFFNA(X)="X$":RETURN"
80 PRINT"GOTO 100"
90 SYS DEC("86E7")
100 PRINTCHR$(147)
110 GOSUB 150
120 INPUT"X ERTEKE= ";X: IF X=0 THEN END
130 PRINT"FN(X)=";FNA(X)
140 GETKEY X$: GOTO 100
```

Ha egyszeri futtatás után kilistázzuk ezt a programot, akkor a legutolsó sora

150 DEFFNA(X)=... : RETURN

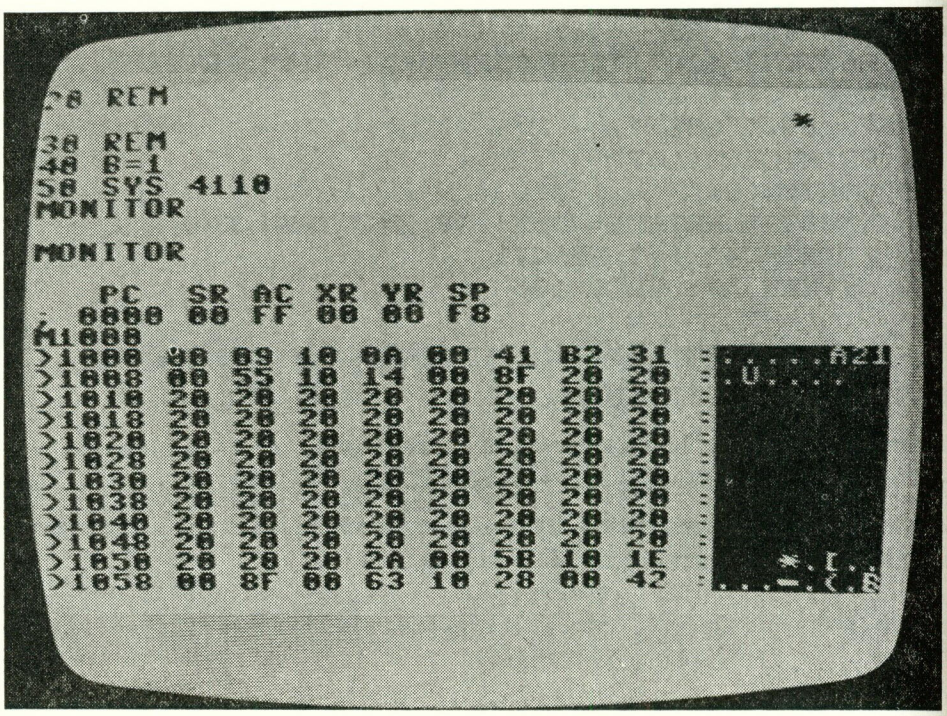
sor lesz. A ... helyén az X$ karaktervváltozó értékét látjuk.

2.6.1.1. Adjuk értékül X$-nak az X ↑ 3 jelsorozatot! Ennek megfelelően a futtatás közben beírt új függvényutasítás X harmadik hatványát fogja kiszámítani. Természetesen X-nek bármely más függvényét is megadhatjuk.

Az előző fejezetekben már közöltünk fogásokat arra, hogy hol helyezhetünk el a memóriában gépi kódú rutinokat. Most a gépi kódú rutint magában a BASIC programban fogjuk elhelyezni, illetve megmutatjuk ennek a módszerét.
Nézzük a következő példát!

```
10 A=1
20 REM
30 REM
40 B=1
50 SYS 4110:  REM A 20-AS SOR REM-JE UTAN VAN A GEPI KODU RUTINUNK
```

MONITOR
M 1000
Láthatjuk, hogy $100E-től a $1053 címen álló $2A értékig a 70 darab szóköz kódja, a $20 van. Ezen a területen elhelyezhetjük a példaként feltételezett, 70 bájtot elfoglaló gépi kódú rutint. Ha a rutin kezdőcíme éppen a $100E, akkor a SYS 4110 utasítással hívható a BASIC programsorok között „elbújtatott" rutin.
Ha a $1053 cím tartalmát $2A-ról $20-ra változtatjuk, akkor a következő LIST listájában nyomtalanul „eltűnik" a hosszú REM sor.



2.6.3. *A LIST parancs módosítása*

A program szövegének listázása közben igen nehéz el is olvasni a sorokat. Igaz, hogy a listázás lassítható, de ennél jobb megoldást is tudunk javasolni.
A programszöveg képernyőre történő listázását az alábbi rutinnal módosítani tudjuk. A SHIFT billentyű lenyomására a listázás megáll, s a billentyű felengedésére pedig folytatódik.
Írjuk be:

MONITOR

```
2
110:   2000                    .OPT P4
120:   2000                    *=   $2000
                         ;
                         ; ************************
                         ;  LIST RUTIN MODOSITASA
                         ; ************************
200:   2000 48                 PHA
210:   2001 AD 43 05           LDA  $0543      ;SHIFT KAPCSOLO
220:   2004 D0 FB              BNE  $2001      ;LENYOMVA
230:   2006 68                 PLA
240:   2007 4C 6E 8B           JMP  $8B6E      ;LIST RUTIN
```

Ha ezt a kis rutint $2000-re írjuk, akkor a listázás előtt a

POKE 774,0: POKE 775,32

parancsot kell adni ($0306—$0307 a listázás BASIC vektora).
Részletesebben:

$0306 = 774_{10}
$0307 = 775_{10}
$2000 = 0 + 32 * 256 = 8192_{10}

2.6.4. *A BASIC bővítése, új szavak definiálása*

A C 16-hoz és a C Plus/4-hez adott BASIC kézikönyvek részletesen tartalmazzák a BASIC alapszavak listáját és ezek leírását. Azonban nem csupán ezeket az alapszavakat használhatjuk, hanem arra is van lehetőségünk, hogy újakat készítsünk. Megtehetjük, hogy pl. „!" jellel kezdődően akár több ilyen fajta alapszóból álló saját szókészletünk legyen.
Gyakorlati szempontból ez teszi lehetővé, hogy a SYS és az USR alapszavak mellett ezzel a módszerrel is aktivizálhassunk gépi kódú rutint a BASIC program futtatása közben.

Az interpreter az eredetileg beépített BASIC alapszókészletet és az alapszavakat feldolgozó BASIC rutinok címét tartalmazó táblázatot a következő memóriacímeken találja:

$818E—$8382	alapszavak
$8383—$847Ø	az alapszavak sorrendjében itt találjuk az egyes alapszavakat feldolgozó beépített rutinok címét.

Minden alapszó utolsó karakterének kódja $8Ø-nal, azaz decimális 128-cal nagyobb, mintha csak belső karakter lenne. A feldolgozó rutin címe 1-gyel kisebb, mint ahol valójában a memóriában van.
A BASIC interpreter vektortáblázata egyebek között tartalmazza a következő címeket:

Cím:	Funkció:
$Ø3ØC—$Ø3ØD	felhasználói parancs értelmezése, tokenizálása
$Ø3ØE-$Ø3ØF	felhasználói token listázása
$Ø31Ø—$Ø311	felhasználói token feldolgozása, végrehajtása

Első lépésben ezt a három funkciót kell gépi kódú szubrutin formájában kidolgozni, és a memóriában elhelyezni. Ez egyúttal új felhasználói alapszavak definiálását, tehát az eredeti BASIC alapszókészlet bővítését jelenti.
MONITOR-ba lépve definiáljunk 2 új utasítást! Az egyik a !SORT, a másik az ABC lesz.

M

>61ØØ után írjuk be a megfelelő hexadecimális kódokat:

!=$21; S=$53; O=$4F; R=$52;

T=$54; vagy $D4. Itt a $D4 kódot kell használni, hiszen megállapodás szerint így jelezzük a kulcsszó végét.

A felsorolt hexadecimális számokból a !SORT alapszó rakható össze.

Az ABC kulcsszó betűinek a kódjai:

A=$41; B=$42; C=$43. Itt most a
$43+$8Ø=$C3 értékkel jelezzük az ABC kulcsszó végét.

A következő feladat, hogy az általunk most definiálandó szavakat az interpreterrel elfogadtassuk. Helyezzük el az ehhez szükséges 3 szubrutint $6ØØØ-től:

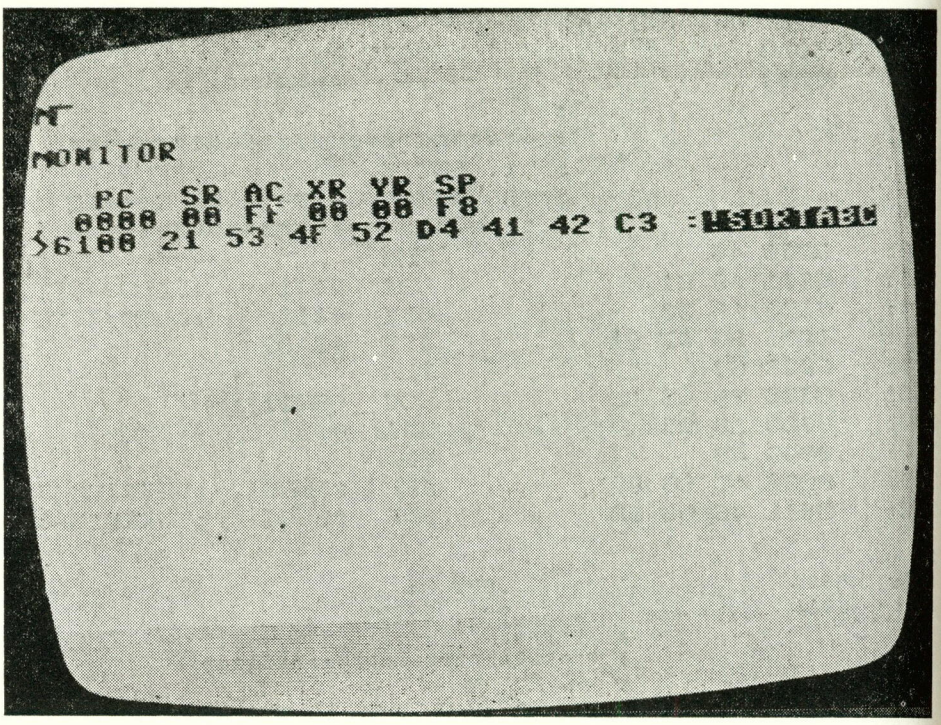
```
2
110:   6000                    .OPT P4
120:   6000                    *=   $6000
                        ;
                        ; ***************************
                        ;  1. RUTIN
                        ;  BOVITETT C-16-RA ES C+4-RE
                        ; ***************************
200:   6000 48                 PHA
210:   6001 A9 61              LDA  #$61     ;FELSO BYTE
220:   6003 A0 00              LDY  #$00     ;ALSO BYTE
230:   6005 20 07 8A           JSR  $8A07
240:   6008 68                 PLA
250:   6009 90 06              BCC  $6011
260:   600B A5 0B              LDA  $0B
270:   600D 48                 PHA
280:   600E 4C D6 89           JMP  $89D6    ;USER TOKEN
290:   6011 4C 6C 89           JMP  $896C    ;TOKENIZALAS

2
110:   6014                    .OPT P4
120:   6014                    *=   $6014
                        ;
                        ; ***************************
                        ;  2. RUTIN
                        ;  BOVITETT C-16-RA ES C+4-RE
                        ; ***************************
200:   6014 AA                 TAX
210:   6015 84 49              STY  $49
220:   6017 A0 61              LDY  #$61     ;SZOTABLAZAT
230:   6019 84 23              STY  $23
240:   601B A0 00              LDY  #$00
250:   601D 4C 9C 8B           JMP  $8B9C    ;USER TOKEN LISTAZASA
260:   6020 EA                 NOP

2
110:   6021                    .OPT P4
120:   6021                    *=   $6021
                        ;
                        ; ***************************
                        ;  3. RUTIN
                        ;  BOVITETT C-16-RA ES C+4-RE
                        ; ***************************
200:   6021 38                 SEC
210:   6022 E9 80              SBC  #$80       ;SHIFTEZETT-E
220:   6024 0A                 ASL
230:   6025 B9 11 61           LDA  $6111,Y    ;USER TOKENEK RUTINJAI
240:   6028 48                 PHA
250:   6029 B9 10 61           LDA  $6110,Y
260:   602C 48                 PHA
270:   602D 4C 73 04           JMP  $0473
```

A 3 szubrutin a felhasználói alapszavakat hozzákapcsolja az eredeti BASIC szókészlethez.

A kezdőcímüket helyezzük el az előző táblázatban közölt címeken! Az új állapotot a fénykép mutatja:



Az előző fényképen $6110-től 2—2 számot látunk. Egyelőre ezek azok a címek, amelyek a !SORT és az ABC felhasználói alapszavakat feldolgozó gépi kódú rutinok belépési pontjai.
Ezek azonban még nem valódi, „éles" memóriacímek, hanem csupán 2 másik BASIC alapszó kölcsönvett, ideiglenes címei.
A tokenizálás és a listázás már így is hibátlan.
Próbáljuk ki!
A BASIC program:

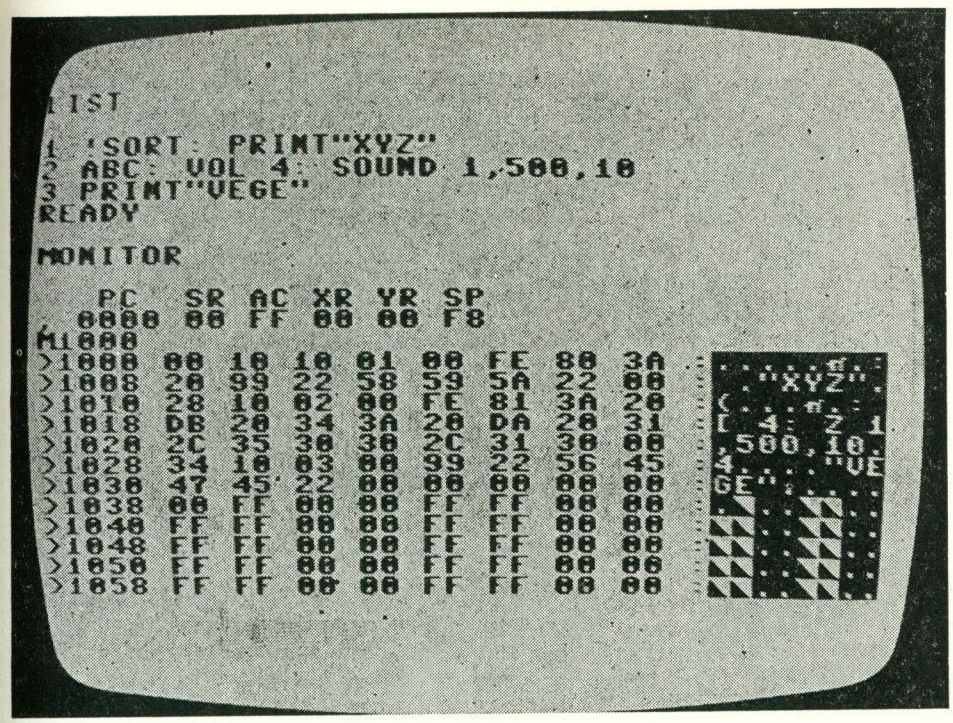
```
1 !SORT
2 ABC:VOL 4:SOUND1,500,10
3 PRINT"VEGE"
```

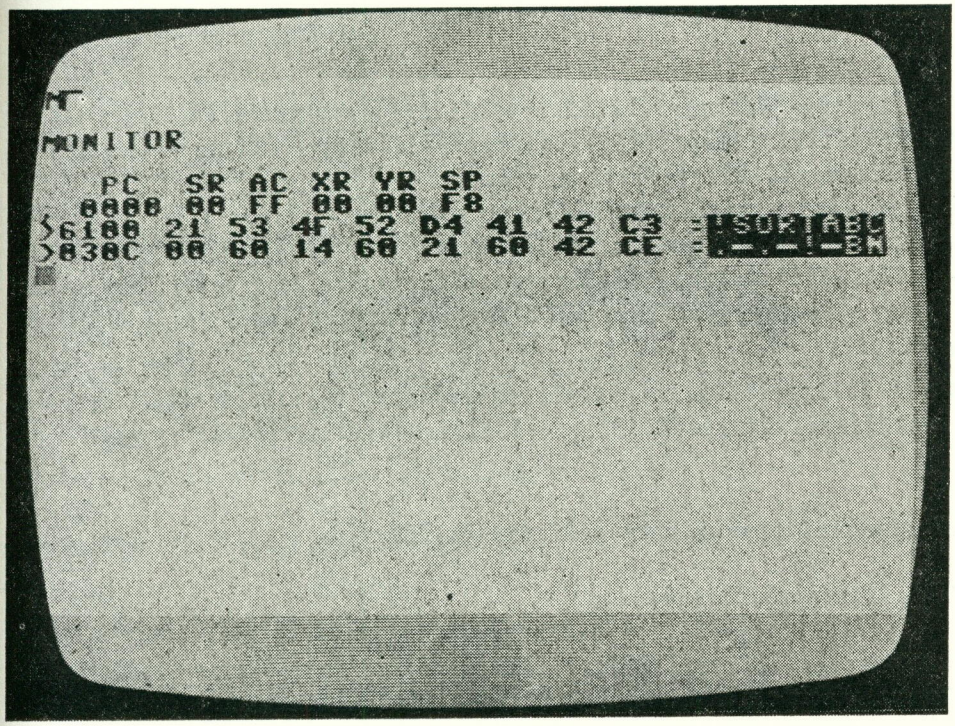
LIST

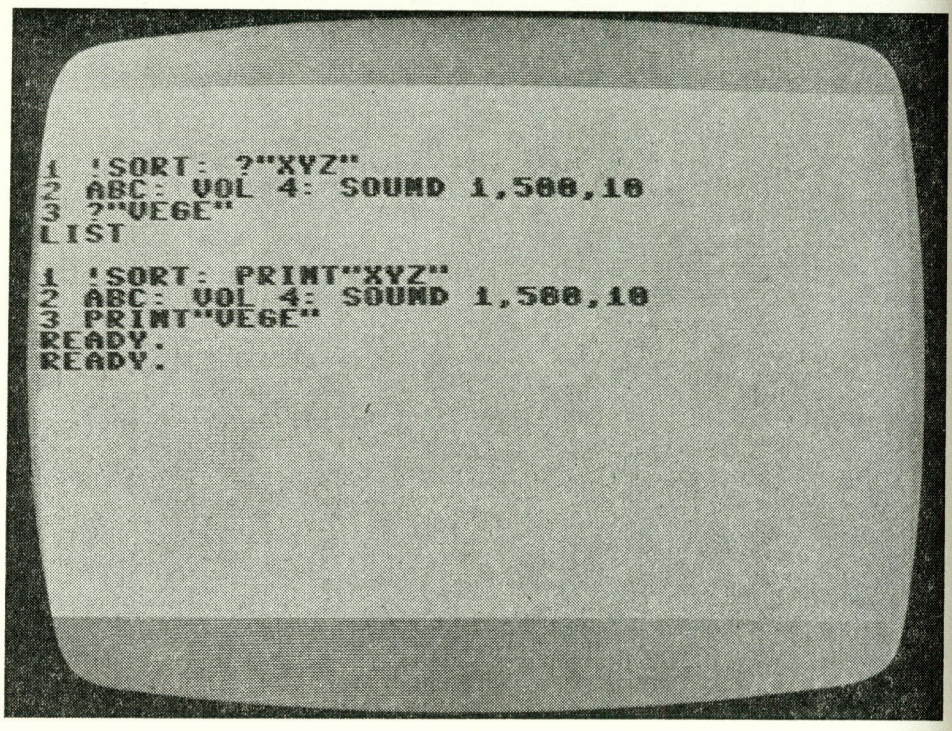
...

MONITOR

M 1000


LIST
1 'SORT: PRINT"XYZ"
2 ABC: VOL 4: SOUND 1,500,10
3 PRINT"VEGE"
READY
MONITOR
PC SR AC XR YR SP
0000 00 FF 00 00 F8
M1000
>1000 00 10 10 01 00 FE 80 3A
>1008 20 99 22 58 59 5A 22 00
>1010 28 10 02 00 FE 81 3A 20
>1018 DB 20 34 3A 20 DA 20 31
>1020 2C 35 30 30 2C 31 30 00
>1028 34 10 03 00 99 22 56 45
>1030 47 45 22 00 00 00 00 00
>1038 00 FF 00 00 FF FF 00 00
>1040 FF FF 00 00 FF FF 00 00
>1048 FF FF 00 00 FF FF 00 00
>1050 FF FF 00 00 FF FF 00 00
>1058 FF FF 00 00 FF FF 00 00


MONITOR
PC SR AC XR YR SP
0000 00 FF 00 00 F8
>6100 21 53 4F 52 D4 41 42 C3
>030C 00 60 14 60 21 60 42 CE



Az interpreter tehát már hiba nélkül tokenizálja a programot. (A !SORT tokenje FE8Ø, az ABC tokenje FE81.) A listázás is hibátlan. A futtatás során a hibátlan működéshez még szükség van az új rutinokra.
Követelmény a rutinokkal szemben, hogy
— utolsó végrehajtott utasításuk RTS legyen,
— a $3B—$3C mutató az aktuális soron belül és az új utasítás végrehajtása után vagy a sor végét jelző nulla bájtra, vagy az utasítást követő :-ra, a $3A címére mutasson!

2.6.4.1. Ez a fejezet sok ismeretet tartalmazó, viszonylag nehezen elsajátítható rész. Használjuk sokszor a MONITOR-t, a ROM listát tartalmazó szakkönyveket!

2.7. Az elkészített programokat védelemmel is elláthatjuk.

2.7.1. Memóriacímek módosítása

A memóriában bőségesen találunk olyan tárcímeket, amelyek megváltoztatásával a program védelme is elérhető. Egy részletesebb felsorolást közlünk:

POKE címe:	Értéke:	Hatása:
774	187	A LIST csak a sorszámokat listázza
774	110	A LIST a teljes szöveget listázza
22	35	A LIST a sorszámokat nem írja ki
22	25	A LIST kiírja a sorszámokat is
816	136	A SAVE hatástalan
816	164	A SAVE használható
814	239	A LOAD nem működik
814	74	A LOAD működik
1343	0	A billentyűzet inaktiv
1343	10	A billentyűzet aktív
806	103	A RUN/STOP hatástalan
806	101	A RUN/STOP hatásos
1344	64	A lenyomott billentyű nem ismétel
1344	128	A lenyomott billentyű ismétel
1339	0—16	A karakter színe
2029	0—16	Kurzor alatti szín
240	0—255	Addig vár, amíg egy tetszőleges billentyűt le nem nyomunk.

2.7.2. Egyszerű programvédelem

A POKE utasítások táblázatának felhasználásával egyszerű programvédelmet tudunk megvalósítani. A BASIC programot az AUTOSTART rutin felhasználásával vigyük ki a kazettára! Így elérjük, hogy a LOAD hatására a program betöltődik a memóriába, s a betöltést követően automatikusan el is indul.
A program legelső utasításai a következők legyenek:

```
1 POKE 806,103:  REM RUN/STOP TILTÄSA
2 POKE 816,136:  REM SAVE TILTASA
3 POKE 774,187:  REM LIST TILTASA
```

2.7.3. Programvédelmet a LIST befolyásolásával is elérhetünk.

A következő POKE-okkal a LIST kiadása után érdekes hatást érhetünk el:

POKE-ok:	Hatásuk:
POKE 774,164 POKE 775,242 POKE 774,0	A gép alapállapotba kerül Az első sorszám megjelenik, s ezt a BREAK üzenet követi

POKE 774,27	Hasonló az előzőhöz
POKE 774,50	Hasonló az előzőhöz
POKE 774,12Ø	A listában a változókat END szóval helyettesíti.
POKE 774,14Ø	A listázás nem hatásos. A RUN/STOP hatástalan
POKE 774,15Ø	A lista végtelen sok END-ből áll
POKE 774,2ØØ	Az első sorszámot kiírja, s BREAK jelenik meg
POKE 774,11Ø	
POKE 775,139	Alapértelmezés.

2.7.3.1. A tanulók szívesen próbálnak ki meglepő trükköket. A felsoroltak mellett bizonyára ők is felfedeznek újabb POKE változatokat. Ehhez segítségül a BASIC vektorok táblázatát lehet használni.

3. A GÉPI KÓDÚ PROGRAMOZÁS. FOGÁSOK GÉPI KÓDBAN PROGRAMOZÓKNAK

3.1. A gépi kódú programozás néhány fejezetben már szóba került. Most újabb fogásokat mutatunk meg.

3.1.1. KERNAL rutinok aktivizálása

A KERNAL a személyi számítógépekben azokat az alapvető fontosságú gépi kódú rutinokat tartalmazza, amelyekkel a gép és az ember közötti kapcsolat megvalósul. Kezelhető a billentyűzet, a képernyő, lebonyolítható a belső adatforgalom, elvégezhetők a számítások stb.
A KERNAL rutinokat a BASIC interpreter és a szövegszerkesztő mellett a programozó is felhasználhatja. Ehhez a gépekhez tartozó szakirodalom ad segítséget.
Válasszuk ki a SCREEN nevű KERNAL rutint!
A MONITOR segítségével próbáljuk meg aktivizálni is!
A rutin feladata a képernyőforma megadása, a méret lekérdezése. Memóriabeli kezdőcíme $FFED. Paraméterregiszterei X és Y, ezekben adja át a képernyő sorainak és oszlopainak a számát.

MONITOR

```
2
110:   2000                    .OPT P4
120:   2000                    *=   $2000
                          ;
                          ; ********************
                          ;  SREEN KERNAL RUTIN
                          ;  AKTIVIZALASA
                          ; ********************
200:   2000 20 ED FF           JSR  $FFED      ;SREEN RUTIN
210:   2003 86 2B              STX  $2B
220:   2005 84 2C              STY  $2C
230:   2007 00                 BRK

>002 B
```

A $2B cím tartalma $28, ami decimális 40-et jelent. Hasonlóan a $2C cím tartalma decimális 25.
A képernyő kikapcsolását egy másik KERNAL rutinnal oldhatjuk meg:

```
2
110:    2000                        .OPT P4
120:    2000                        *=   $2000
                            ;
                            ; ************************
                            ;  KEPERNYO KIKAPCSOLASA
                            ; ************************
200:    2000 20 64 E3               JSR  $E364      ;KEPKIOLTAS
210:    2003 4C 00 20               JMP  $2000
220:    2006 00                     BRK
```

G 2000

A JSR $E364 utasítás a képernyőt kikapcsolja, s az egész képernyő színe a keret színére változik. Ez a RESET gombbal szüntethető meg.

3.1.2. Ciklus szervezése gépi kódban

Gyakran előforduló feladat ciklus szervezése gép kódban. Legyen a cél a memória 255 bájtjának átmásolása. A megfelelő BASIC sor ez lenne:

```
10 FOR X=1 TO 255
20 POKE 4096+X-1,PEEK(8192+X-1)
30 NEXT X
```

Gépi kódban pedig:

```
2
110:    0340                        .OPT P4
120:    0340                        *=   $0340
                            ;
                            ; ********************
                            ;  CIKLUS SZERVEZESE
                            ; ********************
200:    0340 A2 00                  LDX  #$00       ;CIKLUSVALTOZO
210:    0342 BD 00 20               LDA  $2000,X    ;INNEN
220:    0345 9D 00 10               STA  $1000,X    ;IDE
230:    0348 CA                     DEX             ;CIKLUSVALTOZO CSOKKENTESE
240:    0349 D0 F7                  BNE  $0342
250:    034B 60                     RTS
```

3. 1. 3. IF... THEN... ELSE gépi kódban.

Gyakran előfordul, hogy a számítás részeredményétől függően a program elágazáshoz ér. Ilyenre mutat példát az alábbi részlet:

```
Z
110:   2000                    .OPT P4
120:   2000                    *=   $2000
                          ;
                          ; ***************
                          ;  IF-THEN-ELSE
                          ; ***************
200:   2000 A0 00              LDY  #$00       ;NULLAZNI FOGUNK
210:   2002 A2 0A              LDX  #$0A       ;CIKLUS FUTASSZAMA
220:   2004 94 50              STY  $50,X      ;URES BYTE
230:   2006 CA                 DEX             ;X=X-1
240:   2007 10 FB              BPL  $2004      ;VISSZA HA POZITIV
250:   2009 90 15              BCC  $2020      ;HA CARRY=0
260:   200B C9 20              CMP  #$20       ;20-AT KERES
270:   200D D0 21              BNE  $2030      ;HA NEM EGYENLO
280:   200F EA                 NOP             ;TOVABB
```

3.1.3.1. A MONITOR D funkciójával keressenek a $8000-től hasonló programrészeket a ROM-ban!

3.2. A képernyő gépi kódban történő kezelése sok programozó terve között előfordul. Ehhez adunk néhány ötletet.

3.2.1. A képernyő kikapcsolása és görgetése

A C 16 és a C Plus/4 gépekben a TED csip végzi a kép előállítását, a hanggenerálást, az I/O vezérlést.
A képernyő tartalmának előállítása, frissítése is a TED feladata. Ha az $FF06-os cím 4. bitjét nullára állítjuk, akkor a képernyő színe a keret színére változik, de szöveges információ nem látszik rajta.
A képernyő kikapcsolása:

```
1 POKE DEC("FF06"),PEEK(DEC("FF06")) AND 239
```

A képernyő visszakapcsolása:

```
1 POKE DEC("FF06"),PEEK(DEC("FF06")) OR 16
```

A képernyőn megjelenített kép mozgatását, fel/le, jobbra/balra történő eltolását is a TED végzi. A megszokott mozgatás 1 karakter nagyságú és felfelé irányul. A $FF06-os memóriacím átírásával, a jobb szélső 4 bit változtatásával a függőleges görgetést végezhetjük. Az $FF07-es cím ugyanezen bitjei pedig a vízszintes eltolás értékét adják meg.
Próbáljuk ki:

MONITOR

M FF06 itt a jobb fél bájt tartalmát csökkentve a képernyő elmozdul. Az értéket visszaállítva az eredeti állású képet kapjuk.

Ugyanezt POKE utasításokkal is kipróbálhatjuk: C=DEC („FFØ6"): POKE C, PEEK (C) AND 24Ø OR7.
A karakteres képernyő görgetésére az ESC billentyű segítségével is van lehetőség. A billentyű megnyomása után vezérlőkarakterekkel lehet mozgatni a képernyő tartalmát.
ESC+W: a képernyő lefelé mozog.
ESC+V: a képernyő felfelé mozog.
Hasonló lehetőségekről már az előző fejezetekben volt szó.
A képernyő kikapcsolása gépi kódban:

```
2
110:   3000                    .OPT P4
120:   3000                    *=   $3000
                         ;
                         ; ************************
                         ;  KEPERNYO KIKAPCSOLASA
                         ; ************************
200:   3000 78                 SEI
210:   3001 AD 06 FF           LDA  $FF06       ;TED REGISZTER
220:   3004 29 EF              AND  #$EF
230:   3006 8D 06 FF           STA  $FF06
240:   3009 58                 CLI
250:   300A 60                 RTS
```

A képernyő bekapcsolása gépi kódban:

```
2
110:   3000                    .OPT P4
120:   3000                    *=   $3000
                         ;
                         ; ************************
                         ;  KEPERNYO BEKAPCSOLASA
                         ; ************************
200:   3000 78                 SEI
210:   3001 AD 06 FF           LDA  $FF06       ;TED REGISZTER
220:   3004 09 10              ORA  #$10
230:   3006 8D 06 FF           STA  $FF06
240:   3009 58                 CLI
250:   300A 60                 RTS
```

3.2.2. A megszakító rutin módosítása, a karakterkészlet átmásolása.

A gépben egy olyan gépi rutin is működik, amely az ún. megszakításokra figyel. Egyszerű példával szemléltetve ez a rutin lép működésbe, amikor a program futása közben a STOP billentyűt megnyomjuk. Természetesen ez a gépi rutin más feladatot is ellát. A megszakítást figyelő és lekezelő rutin másik neve: IRQ rutin. Vektora az IRQ vektor, amely a $Ø314—$Ø315 memóriacímen található.
A vektor módosítása, új rutin írása pl. akkor fordul elő, amikor a karaktergenerátort a ROM-ból áthelyezzük a RAM-ba.

Ennek bemutatásával más útmutató foglalkozik. Az IRQ rutin áthelyezésének módját az alábbi példa szemlélteti:

```
2
110:    0340                    .OPT P4
120:    0340                    *=   $0340
                        ;
                        ; ************************
                        ;  IRQ RUTIN ATHELYEZESE
                        ;  KEZDOCIME $034B
                        ; ************************
200:    0340 A9 4B              LDA  #$4B       ;ALSO BYTE
210:    0342 A0 03              LDY  #$03       ;FELSO BYTE
220:    0344 8D 14 03           STA  $0314      ;IRQ VEKTOR
230:    0347 8C 15 03           STY  $0315      ;IRQ VEKTOR
240:    034A 60                 RTS
```

```
2
110:    034B                    .OPT P4
120:    034B                    *=   $034B
                        ;
                        ; ***************
                        ;  UJ IRQ RUTIN
                        ; ***************
200:    034B A9 C0              LDA  #$C0       ;KARAKTERKESZLET A RAMBAN
210:    034D A0 11              LDY  #$11       ;$1000-TOL
220:    034F 8D 12 FF           STA  $FF12      ;TED REGISZTER
230:    0352 8C 13 FF           STY  $FF13      ;TED REGISZTER
240:    0355 4C 0E CE           JMP  $CE0E      ;UGRAS AZ IRQ RUTINBA
```

A karakterkészlet átmásolása $1000-re:

MONITOR

T D000 D800 1000

G 340
Ha újból MONITOR-ba lépünk, és átírjuk a $1000-től az első 8 bájt tartalmát, akkor a @ jel új formában jelenik meg.
Ha 1000 helyett 5000-et,
LDY #$11 helyett LDY #$51-et írunk, akkor a karakterkészlet $5000-re kerül.

3.2.3. A memóriablokkok átkapcsolása

Ismert tény, hogy ha az F1 billentyűt lenyomjuk, akkor lehetőségünk van a beépített C Plus/4-es szövegszerkesztő stb. használatára. Ezek a programok a BASIC és a KERNAL ROM „mögött" vannak. Hogyan lehet ezt a ROM-ok közötti átkapcsolást megvalósítani?
A SYS 1525 utasítással a beépített gépi kódú rutint indítjuk:

```
2
110:   05F5                    .OPT P4
120:   05F5                    *=   $05F5
                        ;
                        ; **********************
                        ;  MEMORIA ATKAPCSOLAS
                        ; **********************
200:   05F5 A2 05              LDX  #$05        ;TAR INICIALIZALAS
210:   05F7 A9 80              LDA  #$80        ;$0508-TOL
220:   05F9 8D F1 05           STA  $05F1
230:   05FC A9 03              LDA  #$03
240:   05FE 8D F0 05           STA  $05F0
250:   0601 8D F4 05           STA  $05F4
260:   0604 A5 FB              LDA  $FB         ;AKTUALIS ROM TERKEP
270:   0606 4C FA FC           JMP  $FCFA       ;MASIK ROM HIVASA
280:   0609 10 10              BPL  $061B
```

Az $FCFA cím feltétel nélküli ugrással az $FC89 címre irányítja a végrehajtást. Az átkapcsolt ROM-ban szubrutin hívása történik. Ezzel elindul a KERNAL „mögött" lévő program.

3.2.4. A KERNAL ROM olvasása PEEK segítségével

Válasszunk a C Plus/4-ben egy ROM-beli címet! Ha erre POKE utasítást adunk, akkor azt a gép hibajelzés nélkül végrehajtja.
Például: POKE 64444, 255.
Olvassuk ki most már ennek a bájtnak a tartalmát:
PRINT PEEK (64444)
Az eredmény: 255.
Ugyanakkor, ha MONITOR-ban megnézzük a $FBBC cím tartalmát, akkor benne $Ø1-et találunk.
A jelenség magyarázata az, hogy a POKE utasítás nem a ROM-ban, hanem a „mögötte" álló RAM-ban helyezte el a megadott számot, és a PEEK is onnan olvasta vissza.
Ha a ROM tartalmát akarjuk RAM-ba másolni, és ott saját célból módosítani, akkor ezt a másolást az alábbi programrészlettel végezhetjük el:

```
10 REM $CF96 CIM KIOLVASASA
20 POKE 162,DEC("CF"): REM FELSO BYTE
30 POKE 161,DEC("96"): REM ALSO BYTE
40 POKE 2036,0       : REM $07F4
50 SYS 53142         : REM KERNAL ROM OLVASASA: $CF96
60 PRINT PEEK(2034)  : REM $07F2
```

A kiolvasott érték decimális 44, azaz pontosan $CF96 címen kezdődő
$CF98 2C F8 O7 BIT $Ø7F8
gépi kódú utasítás $2C kódja.

3.2.4.1. A számítógép ilyen mélységű ismerete csak a legérdeklődőbb tanulóktól várható. Elegendő az is, ha a legutolsó fejezetben közölt programrészeket közösen feldolgozzák, és együtt elemzik.
A témakör feldolgozásához jól használható a szoftvermellékletben található többféle program. Ezeket érdemes betölteni, kilistázni. Az érdekesnek tartott ötleteket, fogásokat saját, kisebb, nagyobb programokba be lehet építeni.

4. A C 16 ÉS C Plus/4 GÉPEKBE ÉPÍTETT TED CSIP

4.1. Az előző fejezetekben többször említettük már ezt a részegységet. Itt most bővebben ismertetjük.

4.1.1. A TED csip áttekintése

A TED csip a C 16 és a C Plus/4 gépeknek lényeges alkatrésze. Feladata a videokimenet kezelése, a rendszer időzítése, a dinamikus RAM vezérlése, a ROM kiválasztása és a billentyűzet lekérdezése.
A TED a karaktergenerátor kivételével mindig RAM-ot olvas. Egyedüli kivétel a PRINT által használt karakterformákat leíró ún. karaktergenerátor, amely RAM-ban és ROM-ban egyaránt lehet.
A TED gondoskodik a különböző képernyőkezelési módok megvalósításáról. Kezeli a karakteres és a nagy felbontású képernyőt. Előállítja és megjeleníti a színeket.
A TED segítségével lehet a képernyő méretét 25*40-ről 24*38-ra lecsökkenteni. Próbáljuk ki:

```
10 C=DEC("FF06")
20 A=PEEK(C): B=PEEK(C+1)
30 POKE C,A: POKE C+1,B: GETKEY X$
40 POKE C,PEEK(C) AND 247: GETKEY X$
50 POKE C+1,PEEK(C+1) AND 247: GETKEY X$
60 GOTO 30
```

A GETKEY X$ elhagyható. Ebben az esetben a képernyő méretváltozásait folyamatossá tehetjük.
A TED a működése során megszakítási jeleket küld a központi csip számára. A különböző típusú megszakításokat a $FFØA címregiszter bitjeit Ø—1-re állítva tilthatjuk, illetve engedélyezhetjük:

b7: nem használt
b6: a 3. időzítő engedélyezése
b5: nem használt
b4: a 2. időzítő engedélyezése
b3: az 1. időzítő engedélyezése
b2: fénytollmegszakítás (nincs beépítve)
b1: raszterregiszter-megszakítás
b0: a következő raszterregiszter 9. bitje.

A TED összesen 32 regiszterrel rendelkezik. Ezekről részletes leírás található.
A regiszterek közül kiemeljük a belső időzítésre szolgáló regisztereket. Memóriacímük:

$FF00	1. számláló alsó bájtja	TIMER 1.
01	számláló felső bájtja	
02	2. számláló alsó bájtja	TIMER 2.
03	számláló felső bájtja	
04	3. számláló alsó bájtja	TIMER 3.
05	számláló felső bájtja	

Az 1. számláló alsó és felső bájtja soros intervallum-időzítő. A 2. és a 3. számláló úgy működik, hogy $FF-ről 0-ra csökkennek, majd újra indulnak. Ezek átírhatók, és pillanatnyi értékük bármikor lekérdezhető.
Írjunk egy kb. 10 másodperces késleltetőt a BASIC és a gépi kódú rutin összekapcsolásával:
a gépi kódú rutin:

```
2
110:   3000                    .OPT P4
120:   3000                    *=   $3000
                          ;
                          ; ************************
                          ;  10 MASODPERCES IDOZITO
                          ; ************************
200:   3000 A2 FF              LDX  #$FF
210:   3002 20 EA E2           JSR  $E2EA       ;20 MSEC
220:   3005 20 EA E2           JSR  $E2EA
230:   3008 CA                 DEX
240:   3009 D0 F7              BNE  $3002
250:   300B 60                 RTS
```

aktivizálása:

```
1 PRINT TI$: SYS 28672: PRINT TI$
```

4.1.1.1. Az előbb leírt késleltető rutin különböző perifériák kezelésében is hasznos. Olvassa el a többi útmutatót is!

4.1.2. A TED csip néhány regiszterét módosíthatjuk. Segítségükkel a programok titkosításához is használható ötleteket kapunk.

Az érdekesebb POKE-okat a mellékelt szoftver anyag egyik programja részletesen tartalmazza.

5. A KÉPERNYŐ TARTALMÁNAK KINYOMTATÁSA

5.1. A BASIC nyelven programozók jól ismerik a PRINT utasítást. Segítségével a képernyőre írhatunk.

5.1.1. A virtuális és a valós nyomtatás

A képernyőre történő kiírást virtuálisnak, látszólagosnak nevezhetjük. Hasonlóan virtuális nyomtatásnak nevezzük azt is, amikor először mágneses adathordozóra írjuk az információt, s a valós nyomtatásra csak később kerül sor.
A BASIC nyelven programozó elsősorban a képernyőre nyomtat. Előfordul azonban, hogy egy képernyő tartalmát papíron kinyomtatva, „kopírozva" is meg akarjuk őrizni. Ha erre előre gondolunk, akkor a CMD parancs is elegendő lehet, de külön rutinokat is lehet készíteni az ún. hard copy (hardkopi) céljára.
A memóriatérképpel kapcsolatban megismertük a különböző rendeltetésű tárterületeket. A kis és a nagy felbontású képernyőterületek más-más információt hordoznak.
A különböző hard copy rutinoknak ezek felépítéséhez kell igazodniuk.
Először azt kell eldöntenünk, hogy milyen típusú képernyőről akarunk valós nyomtatást készíteni.
A helyzetet bonyolítja, hogy többféle nyomtató van forgalomban. Ezek általában mátrixnyomtatók, amelyek egy-egy karakter kinyomtatását egy mozgatható fejben elhelyezett tűrács, más szóval tűmátrix segítségével végzik. A tűmátrix a nyomtató típusától függően 8∗8-as, 7∗6-os stb. lehet.

5.1.2. Hard copi készítése karakteres képernyőről

A karakteres képernyő 25 sorban, 40 oszlopban tárolja a betűket, számokat és egyéb jeleket. Kinyomtatásuk viszonylag egyszerű, bár ha a képernyőn különleges jelek is vannak, akkor a hard copi rutin bonyolultabbá válik.

5.1.3. Hard copi készítése nagy felbontású képernyőről

A bittérkép használata esetén a kép egy 320*200-as pontrácsban van. A gép az eredeti karakterhelyek 8*8-as bitmátrixát úgy helyezi el, hogy soronként haladva minden 8*8-as mátrixból vízszintes szeleteléssel 8 egymást követő bájtot tesz egymás mellé. Így lesz a 25*40 bájtból, azaz 1000 bájtból 8*1000=8000 bájt. Ez a 8000 bájt tartalmazza a 320*200-as képet.
A nyomtatóra való „leképezést" nehezíti, hogy az ún. 7 tűs nyomtatók egyszerre csak 7 pontsort tudnak kiírni. A 8 tűs típusok esetében a nyomtatás egyszerűbb.

Nyomtatós típus	Karaktermátrix:
SEIKOSHA GP—100 VC	6*7
MPS—801	6*7
SEIKOSHA GP—290X	5*8
SEIKOSHA GP—700	7*8
C 1526	8*8
MPS—802	8*8
EPSON RX—80	9*9
EPSON FX—80	11*9
EPSON FX—100	11*9

5.1.3.1. Az útmutatóhoz csatolt szoftvermelléklet hard copi rutinokat is tartalmaz. Készítsünk „érdekes" képernyőtartalmakat, nyomtatásra érdemes ábrákat, rajzokat! Ezeket a tanulók akár versenyszerűen is megtervezhetik, kinyomtathatják.

Az útmutatóhoz tartozó szoftvermelléklet a következő programokat tartalmazza:

SCROLL. BAS	Képernyőt mozgató szubrutin
DATAZ. BAS	Gépi kódot DATA sorba épít
IDO RUTIN. BAS	Digitális óra a képernyőn
DUMP. BAS	BASIC változókat listáz
MERGE. BAS	MERGE szubrutin
HCOPY1. KAR. BAS	Karakteres hard copi rutin
CIMKELISTA. BAS	BASIC program cimkéit listázza
RENEW. BAS	RENEW szubrutin
ABLAKOK. BAS	Képernyőablakolás
G8. DATA. BAS	Grafikus hard copi rutin
G7. DATA. BAS	Grafikus hard copi 7.
KÉPKIBE. BAS	Képernyő kapcsolgatása
POKE—OK. BAS	Rendszerbájtok átírása
PARAM1. BAS	Paraméterek átadása
PARAM2. BAS	Paraméterek átadása
AUSTART. DATA. BAS	Autostart szubrutin

FELDOLGOZÁSI JAVASLAT

A szakmai anyag, továbbá az egyes tanórákon rendelkezésre álló idő ismeretében a „Számítógépes fogások, trükkök C 16-ra, C Plus/4-re" című útmutatóhoz feldolgozási javaslatot adunk. A tanórákon, a foglalkozásokon az elméleti anyag átadása mellett jut elegendő idő a gyakorlásra is. Tapasztalataink szerint a tanulóknak az órai munkában mutatott öntevékenysége újabb ötletek, programozási fogások kidolgozását eredményezheti. Ezt is figyelembe véve megváltoztatható az általunk javasolt időbeosztás, és módosítható az ütemezés.

1. foglalkozás: Információtárolás a számítógépben (1.1.1.)
   Gyakorlati feladat: a számrendszerek közötti átváltás, a konverzió gyakorlása.
2. foglalkozás: A számítógépes rendszer hardver egységei (1.1.2.)
   Gyakorlati feladat: Képernyőablakok előállítása, kezelése (2.3.8.) és (2.3.9.).
3. foglalkozás: A központi egység és a tár együttműködése. A gépi ciklus. (1.1.3.)
4. foglalkozás: A BASIC intepreter működése (1.2.1.), (1.2.2.)
   Gyakorlati feladat: Kettő vagy több képernyőablak kijelölése (2.3.10.).
   felirat állandó megjelenítése a képernyőn (2.3.11.)
5. foglalkozás: A memória felosztása. A tár szervezése (1.3.1.).
   Gyakorlati feladat: A számrendszerek közötti átváltásokat ezen a foglalkozáson is célszerű gyakorolni.
6. foglalkozás: A BASIC program elhelyezkedése a memóriában (2.1.1.).
   Gyakorlati feladat: A BASIC program mutatóinak az átírása a MONITOR alkalmazásával (2.3.1.).
7. foglalkozás: A BASIC programok gyorsítása, optimalizálása (2.2.1.).
   Gyakorlati feladat: A 2.2.1.1. szerint készítsünk gyakorló programokat! Mérjük a TI$ változóval a futási időt!
8. foglalkozás: BASIC programok láncolása (2.2.1.). Törölt program aktivizálása a memóriában (2.2.2.).
   Gyakorlati feladat: A két fejezet mintafeladatait a foglalkozáson is próbáljuk ki, futtassuk le! (2.2.2.1.)
9. foglalkozás: A gépi kódú program és a BASIC kapcsolata (2.3.2.).
   Gyakorlati feladat: Gépi kódú program beírása MONITOR segítségével (2.3.3.).
10. foglalkozás: Gépi kódú program betöltése BASIC program futása közben (2.3.4.). Gépi kódú program DATA sorokká való átírása (2.3.5.).
    Gyakorlati feladat: A két fejezet mintafeladatainak, továbbá a mellékelt szoftver anyag programjainak futtatása (2.3.4.1.).
11. foglalkozás: A SYS és az USR utasítások használata (2.3.6.) és (2.3.7.).
    Gyakorlati feladat: A képernyőablakok színezése (2.3.12.).
12. foglalkozás: Fogások a DEL/INS billentyűvel (2.3.14.). A kazettás egység állapotának a lekérdezése (2.4.2.).
    Gyakorlati feladat: A két fejezet feladatainak beírása, futtatása.

13. foglalkozás: Automatikus programindítás a mellékelt szoftver anyagban található AUSTART. BAS nevű program felhasználásával.
14. foglalkozás: A kurzor működésének módosítása (2.5.1.)—(2.5.4.).
    Gyakorlati feladat: A dinamikus billentyűzet-technika kipróbálása a (2.5.5.)-ben leírtak alapján.
15. foglalkozás: BASIC program módosítása a program futása közben (2.6.1.).
    Gyakorlati feladat: A (2.6.2.) és (2.6.3.) fejezet feldolgozása.
16. foglalkozás: A BASIC bővítése, új szavak definiálása (2.6.4.). I. rész.
    Gyakorlati feladat: A fejezetben leírt lépések begépelése, alkalmazása a (2.6.4.1.) figyelembevételével.
17. foglalkozás: A 16. foglalkozás anyagának folytatása. II. rész.
18. foglalkozás: Memóriacímek módosítása (2.3.1.). Egyszerű programvédelem (2.7.2.).
    Gyakorlati feladat: A két fejezet anyagának kipróbálása, gyakorlása a (2.7.3.1.)-gyel együtt.
19. foglalkozás: Gépi kódú rutin aktivizálása (3.1.1.). A képernyő kikapcsolása és görgetése (2.3.1.).
    Gyakorlati feladat: A fejezetekben közölt példák próbálgatása, fejlesztése.
20. foglalkozás: A megszakító rutin módosítása (3.2.2.). A ROM olvasása (3.2.4.).
    Gyakorlati feladat: A két fejezetben közölt lépések gyakorlása.
21. foglalkozás: A TED csip áttekintése (4.1.1.).
    Gyakorlati feladat: A késleltető rutin kipróbálása a 4.1.1.1. figyelembevételével.
22. foglalkozás: A képernyő tartalmának kinyomtatása (5.1.2.) és (5.1.3.)
    Gyakorlati feladat: Végezzünk nyomtatást a mellékelt szoftver anyagban található hard copi rutinok felhasználásával!

A 22 foglalkozásra bontott feldolgozási javaslat a tanulócsoportok és a tárgyi felszerelés ismeretében átdolgozható. Az elméleti részt tartalmazó fejezetek anyaga szűkíthető, a gyakorlati feladatok egyéni ötletekkel bővíthetők.

A „SZÁMÍTÓGÉPES FOGÁSOK, TRÜKKÖK C 16-ra, C Plus/4-re"

című oktatási útmutató szoftver mellékletének leírása

A szoftvermelléklet leírása:

A melléklet 16 darab szubrutint és önálló programot tartalmaz. Ezek egy részében a fogások, trükkök programozása gépi kódban történik, majd pedig DATA sorok formájában építettük be a BASIC nyelvű szubrutinokba. Tehát valamennyi program és szubrutin BASIC nyelven hozzáférhető!

1. SCROLL. BAS

Ez a szubrutin a képernyőt jobbra-balra képes folyamatosan mozgatni. Helye a tárban 16218-tól 148 bájt.

Bekapcsolása:	SYS 16218.
Mozgatás balra 1 karakterrel:	SYS 16275.
Mozgatás jobbra 1 karakterrel:	SYS 16323.
Mozgatás balra 1 ponttal:	SYS 16247,1.
Mozgatás jobbra 1 ponttal:	SYS 16247,Ø.

A szubrutint a 96Ø-as sortól egy bemutató szakasz követi. Ez a rész a felhasználás során kitörölhető.

2. DATAZO.BAS

A megadott decimális kezdőcímtől a memória tartalmát decimális számkódokká alakítja, és elhelyezi a megadott sorszámú BASIC sortól kezdve. A program működése közben a listázás folyamatos. A DATA sorok beírásához a RETURN és NYÍL billentyűket kell használni.

3. IDŐ RUTIN.BAS

A gépi kódban írt rutin a képernyő jobb felső sarkában jelzi az időt. Helye a tárban 32256-tól 186 bájt.

Paraméterezése:

1. lehetőség: a 430-as DATA sorban a megfelelő ASCII kódokat kell elhelyezni

   OO:PP:MM:T formában.

2. lehetőség:

POKE 32432,ASC (ÓRA1)
POKE 32433,ASC (ÓRA2)
POKE 32434,ASC („:")
POKE 32435,ASC (PERC1)
POKE 32436,ASC (PERC2)
POKE 32437,ASC („:")
POKE 32438,ASC (MP1)
POKE 32439,ASC (MP2)
POKE 32440,ASC („:")
POKE 32441,ASC (TMP1)
Hívása: SYS 32256.

4. DUMP.BAS

A szubrutin feladata a BASIC változók nevének és típusának a listázása. Hosszú BASIC programokban hibás változónevek keresésére jól alkalmazható.
Helye a tárban 16100-tól 274 bájt.
Hívása: SYS 16100.

5. MERGE.BAS

A gépi kódban írt rutin feladata kettő vagy több BASIC program vagy rutin összekapcsolása.
Helye a tárban 16331-től 35 bájt.

Használata:

1. RUN
2. SYS 16331
3. LOAD "1. program"
4. SYS 16355
5. LOAD "2. porgram"

Az 1. RUN előtt a 180-as sorból a REM-et törölni kell!

6. RENEW.BAS

A szubrutin segítségével NEW kiadása után is visszaállíthatók a BASIC program mutatói, azaz a program újra listázható, használható lesz. Helye a tárban 16270-től 16376-ig terjed.

Hívása:

1. RUN
2. SYS 16270.

7. CIMKELISTA.BAS

A BASIC-ben írt rutin feladata a programsorok elején álló sorszámok listázása.

8. ABLAKOK.BAS

Ez a program fogásokat mutat be a képernyőablakok színezésére, valamint több ablak egyidejű használatára.

9. KÉPEKIBE.BAS

A program 3 szubrutint tartalmaz:

1. A képernyő kikapcsolása gépi kódban.
   Helyigénye 26001-től 10 bájt.
   Hívása: SYS 26001.
2. A képernyő visszakapcsolása gépi kódban.
   Helyigénye 26012-től 10 bájt.
   Hívása: SYS 26012.
3. A képernyő 25-ször 40-es méretének 24-szer 38-as méretre csökkentése POKE-okkal.

   Hívása: GOSUB 380.

10. PARAM1.BAS

A program a PARAM2.BAS program számára a memória szabad címein elhelyez egy 0—65535 közé eső egész számot.
A memóriacím 28672—28673. A példában átadott szám: 13579.

11. PARAM2.BAS

A program a PARAM1.BAS program által átadott értéket a közösen használt memóriacímekről kiolvassa és kiírja.

12. POKE—OK.BAS

A program példákat mutat olyan memóriacímek tartalmának megváltoztatására, melyek a BASIC interpreter és a KERNAL működését befolyásolják.

13. AUSTART.DATA.BAS

A rutin az előkészítés után automatikusan elindítja a C Plus/4-en a rutin által betöltött BASIC programot.
Előkészítés:

1. Az AUSTART.DATA.BAS program betöltése
2. A kazetta beállítása (számláló)

3. RUN
4. SYS 16ØØ
5. NEW
6. Az újabb BASIC program beírása
7. SAVE

Az autostarttal indítandó program betöltése előtt a kazettát az 1. pontban feljegyzett számlálóálláshoz állítsuk!
A rutin helyigénye: 16ØØ-tól 46 bájt és 679-től 42 bájt.

14. HCOPY.KAR.BAS

Az önálló program formájában közölt rutin feladata a karakteres képernyő kinyomtatása printerre.
Paramétere: nincs.

15. GB.DATA.BAS

A gépi kódú rutin feladata nagy felbontású grafikus képernyőről hard copi készítése printerre.
A nyomtató 8 tűs típusú, pl. MPS—8Ø2, C—1526 stb.
Helyigénye 28671-től 384 bájt.
Hívása: SYS28672.

16. G7.DATA.BAS

A gépi kódú rutin feladata nagy felbontású grafikus képernyőről hard copi készítése printerre.
A nyomtató 7 tűs típusú, pl. MPS—8Ø1, MPS—8Ø3.
Helyigénye 5872-től 6144-ig.
Hívása: SYS 5888.
(A DATA sorok mögött a 430-as sortól bemutató programrész található.)

IRODALOMJEGYZÉK

[1] C—16, C PLUS/4 programozói útmutató NOVOTRADE RT., 1987.
[2] Erdős Iván: Commodore ROM lista,
PLUS/4, C—16, C—116
LSI Alkalmazástechnikai Tanácsadó Szolgálat Budapest, 1986.
[3] Erdős Zoltán: Rendszerváltozók és I/O címek NOVOTRADE—OCTASOFT
[4] Gnädig Péter: Felhasználói tokenek
Mikroszámítógép magazin, 1987/4.
[5] Dr. Kovács Magda: Egyszerűen a mikroszámítógépről
LSI Alkalmazástechnikai Tanácsadó Szolgálat Budapest, 1985.
[6] Tassonyi Kadocsa: Grafikus hardcopy C—16-ra
Mikroszámítógép magazin, 1986. augusztus.
[7] Tóth Viktor: A Commodore 16-os belső felépítése NOVOTRADE, 1986.
[8] Dr. Ury László: Commodore C—16 BASIC és felhasználói kézikönyv.
LSI Alkalmazástechnikai Tanácsadó Szolgálat Budapest, 1985.
[9] Vancsó Gyula: Mikroszámítógép-elemek a tervezésben
Műszaki Könyvkiadó, Budapest, 1981.

TÁRGYMUTATÓ

ablakolás 35
adatvezeték 19
adatforgalom 17
adatok eltolása 18
akkumulátor 17
aktuális sorszám 27
alprogram 18
aritmetikai művelet 17
aritmetikai-logikai egység (ALU) 18
átírható bájtok 65
átvitelbit 20
autostart 43

bájt 15
bájt szervezésű gépek 15
BASIC alapszavak 22
BASIC alapszó-feldolgozás 50
BASIC alapszókészlet bővítése 49
BASIC interpreter 21
BASIC interpreter vektortáblája 22
BASIC munkaterület 23
BASIC nyelv 25
BASIC parancs 33
BASIC pointerek visszaállítása 28
BASIC program autostarttal 43
BASIC program bővítése futás közben 47
BASIC program mutatói 22
BASIC ROM 23
BASIC rutinok címe 50
BASIC szókészlet (saját) 50
BASIC vektorok 22
Bemeneti irányú adatforgalom 20
Berendezésorientált-áramkör 21
betöltő program 43
billentyűzet közvetlen lekérdezése 44
billentyűzetpuffer 44
bináris nulla bájt 46
bináris összeadás 18
bináris szám 15
bit 15
bitkombináció 19
Boole algebrai műveletek 18
bővítő nélküli C 16 25
busz 17

CHAR utasítás 44
ciklus szervezése gépi kódban 58
ciklusváltozó 58
címadat 19
címezhető egység 15
CIRCLE utasítás 41
CMD parancs 66
címvezeték 23
csak olvasható memória 23

DATA utasítás 32
datázó program 71
decimális művelet 18
DEL/INS billentyű 40
detokenizálás 22
digitális óra a képernyőn 71
dimenzionálás 27
dinamikus billentyűzet technika 44
dinamikus munkaterület 25
dinamikus RAM vezérlése 64
DRAW utasítás 41

egész szám 27
egyirányú port 20
egyszerű programvédelem 55
elágazások a programban 17
ESC billentyű 35
ESC funkciók 38

F1 billentyű 40
fekete jel 35
felhasználó 50
felhasználói parancs 50
felhasználói token 50
feltétel nélküli ugrás 17
feltételes ugrás 17
főprogram 18
függvény paraméterei 34
full-screen képernyő 21
funkcióbillentyűk 40

futtatható program 22
gép programozása 15
gépi ciklus 19
gépi kódú ciklus szervezése 58
gépi kódú program 18
gépi kódú rutin belépési pontja 33
gépi kódú rutin hívása 34
gépi kódú rutin paraméterei 33
gépi kódú szubrutin 28
gépi szó 15
grafikus hard copi 67
grafikus karakterek 41
GSHAPE utasítás 41
hanggenerálás 59
hard copi 66
hard copi rutinok 66
hardver 17
hexadecimális számábrázolás 15
hexadecimális számrendszer számjegyei 16
HOME billentyű 35
I/O csip 21
I/O címtartomány 23
I/O műveleteket végző egység 17
I/O port 20
I/O RAM 24
I/O vezérlés 59
idegen kód 19
időzítő engedélyezése 65
index 18
indexregiszter 15
információ 46
INPUT kérdőjel letiltása 46
interpreter 21
interpreterciklus 22
írható/olvasható memória 23
IRQ rutin áthelyezése 60
IRQ vektor 60
jelek definiálása 15
JMP utasítás 34
karaktergenerátor 24
karakter vízszintes szeletelése 67
karakterek definiálása 24
karakterek fényereje 23
karakterek színe 23
karakterek villogása 23
karakteres hard copi 66
karakteres képernyő 41
karakterkészlet átmásolása 61
kazettapuffer 41
kazettás egység 41
kilobájt 19
kép előállítása 64
képernyőablakok 35
képernyőablakok kijelölése 35
képernyőablakok színezése 38
képernyő görgetése 38
képernyő időigénye 20
képernyő kikapcsolása 57
képernyő (karakteres) 64
képernyő (kvázigrafikus) 35
képernyőmemória 23
képernyőszerkesztő 21
képernyő visszakapcsolása 57
KERNAL ROM 24
KERNAL rutinok 24
KERNAL rutinok ugrótáblája 24
KERNAL vektor 21
kétirányú port 20
kettes számrendszeren 15
kimeneti irányú adatforgalom 20
kód 15
kódfeldolgozó automata 15
kódok elöállítása 15
kódok értelmezése 15
kódok tárolása 15
kódok visszaalakítása 15
kódsorozat 15
kommunikáció 22
komplemens képzés 18
kontrollösszeg 32
konverzió 16
központi tár 17
külső csatlakozó 21
külső ROM 21
kurzor ismétlő funkció 46
kurzor kikapcsolása 44
kurzor pozícionálása 44
kurzorvezérlő billentyűk 21
kvázigrafika 35
lap 19
lapszervezés 19
lemezegység ellenőrzése 43
LIST módosítása 49
listázható program 22
logikai művelet 17
mátrixnyomtató 66
megszakítás 60
megszakítás bit 64
megszakítási jel 64
megszakítások típusai 64

megszakítást figyelő rutin 60
megszakítást lekezelő rutin 60
memória 18
memória átmásolása 58
memória szervezése 23
memóriavektorok 21
mikroprocesszor 17
mikroszámítógép 17
MONITOR 22
MONITOR M funkciója 31
MONITOR parancsai 31
művelet eredménye 17
műveletvégző egység 17
nagy felbontású grafika 23
negatív bit 18
nulla bit 18
nullás lap 19
nyolcas számrendszer 16
oyomtató 66
oktális számrendszer 16
oktatóprogram 68
operandus 17
órajel-időtartam 19
oszcillátorjel 19
paraméterregiszter 57
parancsmód 33
perifériabájt 23
periféria címzése 21
perifériakezelő rutinok 21
periféria kijelölése 20
periféria lekérdezése 30
perifériavezérlő program 30
perifériára jelek küldése 30
pointerek 22
POKE utasítások táblázata 55
programbetöltő 31
programelágazás 17
program fizikai vége 26
program futási ideje 26
program komprimálása 27
program memóriaigénye 30
program sebessége 27
program szegmens 47
program védelme 54
programleíró bájtok 22
programok beolvasása 43
programok tárolása 43
programok titkosítása 43
programozás 21
puffer pointere 41
RAM 21
raszterregiszter 65
regiszter 17
rekesz, tárrekesz 18
rendszer időzítése 64
RENEW szubrutin 28
RESET rutin 21
RESET vektor 21
rendszerváltozó 22
ROM kiválasztása 62
RS—232 puffer 30
segédváltozó 27
sor befűzése 22
soros intervallum-időzítő 65
SQR függvény 35
SSHAPE utasítás 41
STOP billentyű 43
SYS alapszó 33
szabad memória nagysága 27
szalagpuffer 30
számábrázolás 15
számadat 19
számítógép szoftver elemei 21
számkonstans 27
számláló regiszter 18
szerkesztés 22
színmemória 23
szoftver 21
szószervezés 15
szövegkonstans 25
szövegszerkesztő CPlus/4-en 21
sztring 25
sztringváltozó 41
sztring visszatöltése 41
sztringterület 25
szubrutin 27
tápfeszültség 21
tárhoz fordulás időigénye 20
tárolókapacitás 17
tártól függetlenül kezelt perifériák 21
TED csip 21
teljes címtartomány 19
TIMER I.II.III. 65
titkosított program 54
token (felhasználói) 22
tokenek 22
tokenizálás 22
túlcsordulás bit 20
tűmátrix 66
üres bájt 30

USR utasítás 34
utasításcím 17
utasításkód 19
utasításszámláló regiszter 17
ütem 19
valós szám 27
változó értéke 22
változó értékének átadása 27
változó helyigénye 27
végrehajtási idő 19
veremregiszter 18
vezérlő egység 17
video kimenet 64
video mátrix 44
video terület 21
virtuális nyomtatás 66
virtuális tárrekesz 21
X regiszter 18
Y regiszter 18
záró nulla bájt 26

88-1938 — Szegedi Nyomda
Felelős vezető: Surányi Tibor igazgató

A VORKER Kisszövetkezet ajánlja a Tisztelt Megrendelők figyelmébe az alábbi szoftver termékeket és — szolgáltatásokat.

1. C 16, C Plus/4, C 64 és Spectrum gépekre kifejlesztett oktató és játékprogramokat, amelyek Magyarországon a legolcsóbbak. Ezek közül az idegen és magyar nyelv, a matematika, a kémia, a biológia és a fizika tantárgyi oktatóprogramok segítik a pedagógusok és a tanulók munkáját.

2. Egy- és kétszemélyes játékok, valamint totóprogramok segítik a szabadidő jobb eltöltését.

3. IBM kompatibilis gépekre vállaljuk a munkaügyi és bérszámfejtő rendszerünk adaptálását.

Különleges hardver ajánlatunk: a világon egyedülálló fejlesztésünk a TC-NET+4 számítógép-interfész, amellyel 16 db C 16 illetve C Plus/4 számítógépet kapcsolhatunk lokális hálózatba. Ez két fontos célt szolgál: egyrészt az összes gép ilyen módon használhat egyetlen mágneslemezt és nyomtató egységet, másrészt megvalósítja számítástechnikai vonalon azokat az előnyöket, amelyeket a nyelvoktatás területén egy nyelvi labor biztosít. Ily módon számítástechnikai kabinet hozható létre kis anyagi ráfordítással.

Várjuk érdeklődésüket!

VORKER® Vállalatközi
Organizációs Ipari Szolgáltató
és Kereskedelmi Kisszövetkezet
Szeged, Pf: 711.
Telex: 82-688
Telefon: (62) 26-144
(62) 25-479