Ára: 49,- Ft

ENTERPRESS

IDŐSZAKOS KIADVÁNY AZ ENTEPRISE SZÁMÍTÓGÉPEK FELHASZNÁLÓINAK

MÁTRIX Kft.

1990. NOVEMBER - DECEMBER

Tisztelt Olvasó!

Íme, a kezében tartja az ENTERPRESS második számát.

Egy kicsit ünnep ez a mai nap, legalábbis számunkra, akik az újságot szerkesztjük és kiadjuk. Titokban persze reméljük, hogy olvasóink, ha nem is ünnepelnek, de megosztják velünk örömüket. Végtére is a mi örömünk az olvasó öröme...

Egy újság életében nem az első szám az igazi szenzáció. Nagy boldogság, az kétségtelen, de a valódi csoda a második, harmadik, sokadik szám megjelenése. Első számot is nehéz csinálni, de a következőt összehozni, a lapot életben tartani sokkal nehezebb.

Amikor ezeket a sorokat a nyomdába adjuk, úgy tűnik, az első számból az összes példány elfogyott az újságárusoknál. Ez nagy öröm számunkra, mert azt jelenti, hogy az olvasóknak kell az újság, szükség van rá. Egy csipetnyi keserűség is, hiszen annak, aki most keresi, annak már nem jutott. Szeretnénk ha minden ENTERPRISE felhasználó kezében ott lehetne az újság, a legkisebb faluban is megvehetné, akinek ez az egyik (vagy egyetlen) lehetősége, hogy hírekhez információhoz jusson gépével kapcsolatban, esetleg (sors)társakra találjon. Ezért felhívjuk minenkinek a figyelmét, hogy nem muszáj a megrendelőlapot kivágniuk, az újság a Postánál ugyanúgy megrendelhető, mint bármelyik laptársunk.

Mintha csak szándékosan velünk incselkedne a sors, az első szám megjelenése után jelentették be a papírárak emelését. Ez a mondat szerencsére nem kelt pánikot, hiszen az olvasó már az újságosstandon észrevette, hogy ez nem tükröződik a lap árában. Már most megígérhetjük, hogy mindaddig ameddig lehet, nem fogjuk emelni az árat. Ellenkezőleg: törekedni fogunk arra, hogy – azonos ára mellett – a lap terjedelmét megnöveljük. Nem tévedünk, ha kijelentjük: ez a fogadalmunk, illetve törekvésünk az olvasók egyhangú egyetértésével találkozik. Ez azonban nem csak tőlünk függ, hanem az olvasóktól is, mégpedig kétszeresen.

Az olvasó azzal tud hozzájárulni a terjedelem, és így a példányszám növeléséhez, hogy megrendeli a lapot, hiszen a példányszám a biztonságot jelentő megrendelések számának függvénye. A másik olvasói hozzájárulás a színvonalas cikkek, nem túlságosan hosszú programok, közérdekű észrevételek, levelek, kritikák, vélemények eljuttatása hozzánk. Mi mindent, ami az olvasók széles táborára tartozik, közölni fogunk. A közölt cikkekért pedig természetesen honoráriumot is fizetünk. Mivel jelenleg semmiféle segéderővel nem rendelkezünk, arra kérjük a lelkes vállalkozokat, hogy műveiket az általunk feldolgozható formában juttassák el hozzánk. Az érdeklődők olvassák el a lapban megjelent felhívást, és kérjük, hogy ennek megfelelően járjanak el.

Lapzártakor még nem fejeztük be a beérkezett kérdőívek feldolgozását. Az olvasók nagy része nem érte be egyszerű igen-nem válasszal, most már elárulhatjuk, hogy a kérdések megfogalmazásával szándékosan ezt akartuk kiprovokálni. Egy valami bíztató számunkra: a válaszolók többsége nem mulasztotta el leírni, hogy aggódik a lapért, nehogy ez is úgy járjon mint sok más kezdeményezés, amely az első mene után feladta, megszűnt. Ez a sok-sok hangon megszólaló féltés ad erőt a folytatáshoz és tölt el felelősségérzettel olvasóink iránt. Ez a második szám azt bizonyítja – és teljes erőnkkel azon fogunk dolgozni, hogy a következő számok is azt bizonyítsák -, hogy igyekszünk megfelelni az olvasók elvárásának.

A szerkesztők

Tartalom:

ENTERPRESS

Időszakos kiadvány
Az ENTERPRISE számítógépek
Felhasználóinak
I. évfolyam 2. Szám
1990. november-december
Kiadja a MÁTRIX
Számítástechnikai, Kereskedelmi és
Szolgáltató Kft.
Székesfehérvár
Felelős kiadó:
Annus István ügyvezető
Felelős szerkesztő:
Ujlaki László (UL)
A lap szerkesztői:
Hajnal Csaba (HCs)
DevilSoft (Devil)
Bozai Gábor (BG)
Technikai szerkesztő:
Bartha István
A szerkesztőség és a kiadó címe:
8000 Székesfehérvár
Dózsa György tér 10.
Telefon: (22) 16-520/141
Telefax: (22) 11-585
Levélcím:
ENTERPRESS
1399 Budapest
Pf. 701/334
Lapunk az ENTERPRISE
Computers GmbH., München
Kelet-európai képviseletének, a VTGe
Electronics Ltd.-nek
szakmai támogatását élvezi.
Nyomja a
VIDEOTON Nyomda
Székesfehérvár
Felelős vezető:
Csizmadia Ferenc
ISSN 0866-1820
Terjeszti a
Magyar Posta

Előfizethető a Posta Hírlapelőfizetési és Lapellátási Irodánál
(Budapest 1900)
Ára: 49,. Ft

Megjelenik kéthavonta

Assembly

2. rész

Miután az előző részben tisztáztuk a legfontosabb alapfogalmakat, elkezdjük az ismerkedést a Z80 logikai architektúrájával.

Sok olvasó kérdezte, hogy tulajdonképpen hány részes lesz, és mit fog tartalmazni e sorozat. Mivel az assembly programozás egy óriási téma, így a sorozat addig fog tartani, amíg lesz ENTERPRESS. A kurzus célja pedig leginkább az, hogy az EXOS-t alaposan megismertesse az érdeklődővel, és mineneki képes legyen önállóan EXOS-on futó alkalmazásokat írni.

A Z80 regiszterei

A regiszterek a processzor belsejében elhelyezkedő tárrekeszek, amelyekben különféle aritmetikai, logikai műveleteket végezhetünk. Ezek az A, B, C, D, E, H, L regiszterek. Közülük legfontosabb az A regiszter, amelyet akkumulátornak is szokás nevezni. Sok műveletet csak az akkumulátor közreműködésével lehet elvégezni, mert ez adja a művelet egyik operandusát. A regiszterek 8 bitesek, de lehetőség van az összevonásukra és így kapjuk a 16 bites BC, DE, HL regisztereket. Az akkumulátor „párja" az F (Flag) regiszter, amely szintén 8 bites. A regiszter bitjei a legutoljára elvégzett művelet eredményére vonatkozó információkat szolgáltatnak. A teljes flag regiszterben csak 6 bit van kihasználva:

S	Z		H		P/V	N	C

az F regiszter felépítése

Programunkban a flag regiszter bitjeit vizsgálva végezzük a különféle aritmetikai műveleteket, feltételes elágazásokat. Egy „átlagos" program leggyakrabban a Z (Zero) és a C (Carry), átvitel) biteket szokta figyelni, kezelni. A jelzőbitek használatát példánkban fogjuk látni.

A processzorban két, azonos elemekből álló regiszterkészlet van, a másik csoportot az ún. vesszős regiszterek alkotják: A', F', B', C', D', E', H', L'. A két készletet csak külön-külön lehet használni, köztük az átkapcsolás egy utasítás kiadásával lehetséges.

A regiszterkészlethez tartoznak még a PC (Program Counter, programszámláló), SP (Stack Pointer, verem mutató), IX (Index X), IY (Index Y) 16 bites regiszterek.

PC: a következő végrehajtandó utasításra mutat a tárban.

SP: az adatok ideiglenes tárolására szolgáló veremterületet címzi. A verembe legutoljára betett adatot vehetjük ki onnan legelőször, ez az ún. LIFO (Last In - First out, utolsónak be - elsőnek ki) hozzáférés. Az SP adatbetételkor csökken, kivételkor nő. A processzor is a vermet használja szubrutira ugrás előtt: a stack-re teszi a PC tartalmát, amely már a szubrutinhívó utasítás utáni utasításra mutat. Amikor a CPU végrehajtotta a szubrutint, leveszi a verem tetejéről az ottani szót, és betölti azt a PC-be, így folytatódhat a program végrehajtása. A verem – megfelelő kezeléssel – paraméterátadásra is használható.

IX, IY: ezekhez a regiszterekhez egy byte-os eltolásértéket adva ún. indexelt címzés lehetséges. Akkor használjuk őket, amikor egy táblázatban akarunk adatokat kezelni. A PC, SP, IX, IY regiszteknek nincsen vesszős fajtája.

Van még két speciális regiszter: az R (Refresh, frissítés), amelyet a RAM memóriák frissítésére használ a CPU, és az I (Interrupt, megszakítás) a Z80 egy különleges (megszakításos) üzemmódjában szolgáltatja a végrehajtandó rutin címének egyik részét.

Eddigi ismereteink már elégségesek ahhoz, hogy nekifogjunk az utasítások megismerésének. A nyelv ismerete nem azonos az utasítások ismeretével, a nyelv programozási technikáját kell elsajátítani.

Adatmozgató utasítások

Ezekkel az utasításokkal adatokat tölthetünk a regiszterekbe, tárhelyekbe. A töltés az LD (LoaD, tölt) utasítással lehetséges. Lássuk azokat az utasításokat, amelyekkel értéket adhatunk egy adott regiszternek!

LD A,n LD B,n LD C,n LD D,n LD E,n LD H,n LD L,n

Az utasítást közvetlenük az érték követi, n egybyte-os adatot jelenti. Például az

LD L,27

hatására decimális 27 kerül az L regiszterbe. A 16 bites regiszterekbe töltő utasítások hasonlóak az előzőkhöz:

LD BC,nn LD DE,nn LD HL,nn

LD SP,nn LD IX,nn LD IY,nn

Az utasítást az nn 16 bites adat követi, amelynek regiszterbeli elhelyezkedését még nem ismérjük: a 16 bites adat 2 regiszterben tárolódik (ezt már tudjuk), az alsó (Low) és a felső (High) 8 biten. A felsőn tárolódik (egész számként) az adat 256-al osztott része, az alsón pedig az osztás maradéka lesz. Legyen 57269 a 16 bites adatunk, nézzük meg a felső és az alsó byteok értékét!

High=57269/256=223

Low=57269–223*256=181

57269=181+223*256

A BC, DE, HL regisztereknél mindig az első regiszter (B, D, H) a magas, a másik az alacsony. Ezek után világos, hogy az

LD HL,57269

utasítás hatására H-ban 223, L-ben 181 lesz.(Az első részben felírtuk hexadecimális formában az 57269-et, ami DFB5h-val volt egyenlő. Vegyük észre, hogy a hexadecimális formátumnál azonnal adódik a felső és az alsó byte értéke: DFh = 223, B5h = 181!) Regiszterből regiszterbe tölthetünk adatot a következő utasításokkal:

LD A,A	LD B,A	LD C,A	LD D,A	LD E,A	LD H,A	LD L,A
LD A,B	LD B,B	LD C,B	LD D,B	LD E,B	LD H,B	LD L,B
LD A,C	LD B,C	LD C,C	LD D,C	LD E,C	LD H,C	LD L,C
LD A,D	LD B,D	LD C,D	LD D,D	LD E,D	LD H,D	LD L,D
LD A,E	LD B,E	LD C,E	LD D,E	LD E,E	LD H,E	LD L,E
LD A,H	LD B,H	LD C,H	LD D,H	LD E,H	LD H,H	LD L,H
LD A,L	LD B,L	LD C,L	LD D,L	LD E,L	LD H,L	LD L,L

LD A,I

LD I,A

LD A,R

LD R,A

Az utasításban szereplő első regiszterbe töltődik a második regiszter tartalma. Az

LD A,H

utasításnál tehát az A-ba töltődik a H regiszter tartalma, miközben H változatlan marad. Csak a veremmutatóba tölthetünk 16 bites regiszterből adatot, az

LD SP,HL LD SP,IX LD SP,IY

utasításokkal. Triviális, hogy az

LD SP,IX

hatására SP egyenlő lesz IX értékével. A tárból is tölthetünk adatot regiszter(ek)be, 1 byte esetén csak az akkumulátorba tudunk ilyen közvetlen címzéssel adatot mozgatni:

LD A,(addr)

Az addr jelenti azt a tárcímet ($0 \leq$ addr $\leq$ 65535), ehonnan az adatot meg akarjuk szerezni. Az

LD A,(57269)

hatására az 57269-es tárrekesz tartalma az akkumulátorban lesz. A tárcímet mindig zárójelek közé kell tennünk, ez jelzi a tárcímre hivatkozást.

A tárból 16 bites regiszterbe töltő utasításból több is van:

LD BC,(addr) LD DE,(addr) LD HL,(addr)
LD IX,(addr) LD IY,(addr) LD SP,(addr)

Itt a 16 bites regiszter alsó byte-ba az addr, a felső byte-ba az addr + 1 címen levő adat kerül. Például ha az addr egyenlő 57269-el, és az 57269-es címen 145, az 57270-esen pedig 17, akkor az

LD HL,(57269)

után HL egyenlő lesz 4497-el (H=17, L=145; L+256*H=4497). A tárcímet 16 bites regiszterben is megadhatjuk, így már több regiszterbe mozgathatunk a tárból adatot.

LD A,(BC) LD A,(DE) LD A,(HL) LD A,(IX+d) LD A,(IY+d)
LD B,(HL) LD B,(IX+d) LD B, (IY+d)
LD C,(HL) LD C,(IX+d) LD C, (IY+d)
LD D,(HL) LD D,(IX+d) LD D, (IY+d)
LD E,(HL) LD E,(IX+d) LD E, (IY+d)
LD H,(HL) LD H,(IX+d) LD H, (IY+d)
LD L,(HL) LD L,(IX+d) LD L, (IY+d)

Az előzőek után nem kell sokat magyarázni: az adott regiszterbe kerül annak a tárrekesznek a tartalma, amelynek címét a zárójelek közé tett 16 bites regiszter tartalmazza. Annál inkább magyarázatra szorul például az

LD C,(IY+d)

utasításban szereplő „d”. Az ilyen utasításokban az indexregiszterek „mellé” is tudunk címezni. A „d” egy 8 bites eltolási (displacement, elmozdítás) értéket jelent, fixen beírva az utasításba. Az eltolással az indexregisztertől a tárban lefelé 128, felfelé 127 byte-al tudunk eltérni. A 8 bitnek efajta értelmezését kettes komplemens ábrázolásnak nevezik, amellyel a Z80 aritmetikánál részletesebben megismerkedhetünk. Ezek szerint az

LD C,(IY–89)

hatására a C regiszterben lesz az IY értékénél 89-el kisebb tárrekesz tartalma. Nézzük meg az előzőek fordítottját: írjuk ki a tárba a regiszterek tartalmát! Ismét csak az akkumulátjor tartalmát tudjuk közvetlenül a tárba tölteni az

LD (addr),A

utasítással, amelynek végrehajtása után az „addr” című rekeszben lesz az akkumulátor tartalma. Az ide tartozó 16 bites regisztereket „letöltő” utasítások:

LD (addr),BC LD (addr),DE LD (addr),HL
LD (addr),IX LD (addr),IY
LD (addr),SP

A végrehajtás után az „addr” címen a regiszter alsó, az „addr+1” címen a felső része lesz. Így ha BC-ben 4497 van, és a cím 57269-el egyenlő, akkor az

LD (57269),BC

után 57269-en 145, az 57270-en pedig 17 lesz. Amikor 8 bites regiszter tartalmát akarjuk kiírni, és a tárcím értéke 16 bites regiszterben van, akkor az

LD (BC),A LD (DE),A LD (HL),A
LD (HL),B LD (HL),C
LD (HL), D LD (HL),E
LD (HL),H LD (HL),L

utasítások között választhatunk. Ugyanezt az indexregiszterekkel is megtehetjük:

LD (IX+d),A
LD (IX+d),B LD (IX+d),C
LD (IX+d),D LD (IX+d),E
LD (IX+d),H LD (IX+d),L
LD (IY+d),A
LD (IY+d),B LD (IY+d),C
LD (IY+d),D LD (IY+d),C
LD (IY+d),H LD (IY+d),L

Biztos vagyok benne, hogy nem szükséges elmagyarázni az utasítások hatását. Inkább lássuk helyette az adatmozgató utasítások utolsó három darabját!

LD (HL),nn
LD (IX+d),nn
LD (IY+d),nn

Az utasításokat átfutva azt láthatjuk, hogy az akkumulátornak tényleg kitüntetett szerepe van, minden utasításformában előfordul. Vannak furcsára sikeredett utasítások is. Mire jó például az

LD B,B

utasítás? Tulajdonképpen semmire: a regiszterrel semmi sem történik, a jelzőbiteket sem állítja az F-ben. Egy valamire azonban tökéletes: bárminek az elrontása nélkül lehet vele az „időt húzni”, azaz időzíteni. Egy ilyen utasítás az ENTERPRISE esetében 1 mikroszekundumos végrehajtási időt jelent. Működőképes-e az

LD L,(HL)

utasítás? Igen, mert a címképzésben szereplő HL a regiszterbe olvasáskor már nem él, az L-be normálisan betöltődik a rekesz tartalma. Érdekes az

LD (HL),L

viselkedése is. A HL által címzett rekeszbe írhatjuk saját címének alsó (illetve H esetén felső) byte-ját.

Adatcserélő utasítások

A két regiszterkészlet közötti átváltást az EX (EXchange, csere) utasításokkal tehetjük meg. Az

EX AF,AF’

utasítással váltogathetunk az akkumulátorok között. Az

EXX

végrehajtása után a másik B, C, D, E, H, L regiszterekkel dolgozhatunk. Persze az, hogy melyik regiszterkészletet tekintjük vesszősnek, teljesen mindegy. Ha ilyen cserélő utasításokat használunk, nem szabad belekeverednünk abba, hogy éppen melyikre van szükségünk. Gyors csere valósítható meg a HL és a DE regiszterek között az

EX DE,HL

végrehajtásával. A BC regiszter elrontásától eltekintve (ettől elég nehéz eltekinteni...) ugyanezt valósítja meg az

LD B,D LD C,E LD D,H LD E,L LD H,B LD L,C

utasítássor is. A BC regiszterre DE átmeneti tárolására van szükség, hiszen DE-t közvetlenül felülírjuk a HL tartalmával, BC HL-be töltése előtt. Az eredmény lényegében ugyanaz, csak éppen a hat utasítás 6 mikroszekundum alatt fut le a Z80-ban, a kivesézett utasítás 1 mikroszekundumos végrehajtási idejével szemben. Ehhez a csoporthoz tartoznak a veremhez kapcsolódó utasítások is:

EX (SP),HL EX (SP),IX EX (SP),IY

A művelet végrehajtását követően a verem tetején található adat betöltődik az illető regiszterbe, annak eredeti tartalma pedig a verem tetején lesz.

Aritmetika

Regiszter, regiszterpár és tárrekesz tartalmát az INC (INCrement, növelés utasítással tudjuk eggyel növelni:

INC A INC B INC C INC D INC E INC H INC L

Például ha az akkumulátorban 38 van, és végrehajtjuk az inkrementáló utasítást, akkor az akkumulátorban 39 lesz, ilyenkor a Z bit értéke végig 0. Amikor a regiszterben a legnagyobb ábrázolható szám (255) van, az utasítás végrehajtását követően a regisztertartalom nullával lesz egyenlő. Ekkor a Z bit jelzi 1-be billenésével a zérus eredményt. A 16 bites regiszterek tartalmát az

INC BC INC DE INC HL INC IX INC IY INC SP

utasításokkal tudjuk eggyel növelni. Ezek az utasítások egyik jelzőbit állapotát sem változtatják meg. Tárrekesz tartalmát növelhetjük az

INC (HL) INC (IX+d) INC (IY+d)

utasításokkal. Ha a megcímzett tárrekesz tartalma nullázódik a Z bit 1-be billen.

A növelő utasítások kiegészítői az egyesével csökkenő DEC (DECrement, csökkenés) utasítások. 8 bites regiszterek tartalmát lehet csökkenteni a

DEC A DEC B DEC C DEC D DEC E DEC H DEC L

utasításokkal, amelyek ugyanúgy állítják a Z bitet, mint a növelők.

Például amikor az akkumulátorban 1 van (Z bit ekkor még 0), az utasítás kiadása után az akkumulátor tartlama 0 lesz (ekkor a Z bit 1-be billen, jelezve a zérus eredményt). Ha ezt követően ismét dekrementáljuk a regisztert, annak értéke 255 lesz (és a Z bit ismét nullázódik). A 16 bites regiszterek tartalmát csökkentő

DEC BC DEC DE DEC HL DEC IX DEC IY DEC SP

utasítások növelő párjaikhoz hasonlóan szintén nem állítják a jelzőbiteket. Állítják viszont a Z bitet a

DEC (HL) DEC (IX+d) DEC (IY+d)

utasítások, melyekkel az eggyel növelő, csökkentő utasítások sorát le is zárhatjuk.

Összeadás és kivonás 8 biten

Ezeknél az aritmetikai műveleteknél mutatkozik meg leginkább az akkumulátor kiemelt szerepe. Mindig az A regiszter tartalmazza a művelet egyik operandusát, és az eredmény is az akkumulátorba kerül, felülírva annak eredeti tartalmát. Az összeadást az ADD (ADD, összead) utasításokkal tehetjük meg. Közvetlenül az akkumulátor tartalmához adhatjuk az „n" 8 bites számot az

ADD A,a

utasítással. Ha a művelet elvégzése előtt az A regiszter tartalma 67, akkor az

ADD A,43

után az akkumulátorban 110 lesz. Ha az akkumulátor értéke nulla, akkor ezt a Z bit 1-es értéke jelzi. Az összeadó utasítások a C bitet is állítják, ennek mechanizmusa a következő: tegyük fel, hogy az A egyenlő 200-zal, a hozzáadandó érték pedig 110-zel. A két szám összege 310, de ezt nyilván nem lehet egyetlen byte-on ábrázolni, a művelet elvégzésekor az akkumulátor tartalma ennek megfelelően túlcsordul, ezt jelzi a C bit 1-es értéke. Ekkor a C bittel együtt az akkumulátor tartalmazza a pontos eredményt, ha szükséges, akkor a túlcsordult bit további összeadásokhoz felhasználható (ezek ún. többszörös pontosságú aritmetikai műveletek). Példánkban a művelet előtt az A tartalma:

1	1	0	0	1	0	0	0

Az összeadás után a C bit és az A tartalma:

1	0	0	1	1	0	1	1	0

A C bit mindig a legutoljára elvégzett művelet eredményét mutatja (ugyanúgy, mint a többi). Ha a C bitet a byte mellé „ragasztjuk", és a kilenc bitnyi értéket együttesen meghatározzuk, akkor magkapjuk a 310-et (a C bit helyiértéke 256, az akkumulátorban 54 van, 256+54=310). Az akkumulátor tartalmához másik regiszter értékét adhatjuk az

ADD A,A ADD A,B ADD A,C ADD A,D
ADD A,E ADD A,H ADD A,L

utasításokkal.

Ha az akkumulátorhoz egy tárrekesz tartalmát akarjuk hozzáadni, akkor az

ADD A,(HL) ADD A,(IX+d) ADD A,(IY+d)

utasításokat használhatjuk.

A művelet során képződő átvitel bitet az akkumulátorhoz adhatjuk az ADC (ADd with Carry, összeadás átvitellel) utasításokkal, melyek a C és a Z biteket ugyanúgy állítják, mint ahogyan azt fentebb láttuk. Az „n" 8 bites közvetlen operandus értékén kívül a C-t is az akkumulátorhoz adja az

ADC A,n

utasítás. Ha például a művelet elvégzése előtt a C bit értéke 1 (azaz az előző műveletnél átvitel keletkezett), és az akkumulátor tartalma 250-nel egyenlő, akkor az

ADC A,15

végrehajtását követően a C bit értéke egy lesz (hiszen újabb átvitel keletkezett), az A regiszrerben pedig 10 lesz (250 + 15 + 1 = 266, ebből C bit = 1, az akkumulátorban 266–256 = 10).

Az átvitel és egy regiszter értékét az

ADC A,A ADC A,B ADC A,C ADC A,D
ADC A,E ADC A,H ADC A,L

utasításokkal adhatjuk az akkumulátor tartalmához. Ha az operandus a tárban van, akkor az összeadást az

ADC A.(HL) ADC A,(IX+d) ADC A,(IY+d)

utasításokkal hajthatjuk végre.

A SUB (SUBtract, kivon) és az SBC (SuBtract with Carry, kivonás átvitellel) utasításokkal tudunk az akkumulátor tartalmából kivonni. Közvetlen értéket a

SUB n

utasítással vonhatunk ki az akkumulátorból. Ha az A regiszter tartalmából vele egyenlő adatot vonunk ki a Z bit egy lesz (hiszen az eredmény zérus lesz). Amikor a kivonandó nagyobb, mint az akkumulátor tartalma, akkor a C bit 1 lesz, jelezve ezzel az alulcsordulást. legyen az A regiszterben 5, ekkor a

SUB 10

után a C bit értéke 1, a regiszterben pedig 251 lesz. Eljött az idő arra, hogy tisztázzuk a számok kettes koplemens ábrázolását. Ugyanis a művelet után eredményül kapott átvitel és 251-es érték a –5-öt ábrázolja, a következők szerint: az indexelt címzésnél már láthattuk, hogy a „d" eltolási értékkel az indexregiszter által mutatott címtől lefelé (mínuszban 128 felfelé (pluszban) 127 byte-tal tudunk eltérni. Ebből talán már látható a 7. bit kitüntetett előjelet helyettesítő szerepe. Ha a 7. bit értéke 1, akkor negatív számról van szó, a többi 7 (6–0) biten a szám lesz ábrázolva. Az előjeles összeadások és kivonások korrekt elvégezhetőségéhez a negatív számokat kettes komplemensükkel kell ábrázolni: a szám abszolút értékében minden bitet ellenkezőjére változtatunk, és ehhez még egyet hozzáadunk. Írjuk fel a –5-öt! Az abszolút érték:

0	0	0	0	0	1	0	1

negálva a biteket:

1	1	1	1	1	0	1	0

egyet hozzáadva:

1	1	1	1	1	0	1	1

aminek értéke a példánkban kapott 251. Kettes komplemens ábrázolásban tehát a –128 és a +127 közötti számokat tudunk ábrázolni, és a Z80 ilyen formában megadott számokkal műveleteket tud végezni. A

SUB A SUB B SUB C SUB D SUB E SUB H SUB L

utasításokkal regiszter tartalmát vonhatjuk ki az akkumulátorból.(Érdekes, hogy ezeknél az utasítsáoknál nem kell külön (és feleslegesen) jelölni az akkumulátort, (úgy mint például az ADD A,H-nál.) Tárban lévő adatot a

SUB (HL) SUB (IX+d) SUB (IY+d)

utasításokkal tudunk az akkumulátorból kivonni. Az alulcsordult bitet is kivonhatjuk az

SBC A,n

utasítással. Tehát ha az A regiszterben 5 van, a C bit 1, akkor az

SBC A,10

után az akkumulátorban –6 (250) lesz. Sejthető, hgoy regisztertartalmat az SBC A,A SBC A,B SBC A,C SBC A,D SBC A,E SBC A,H SBC A,L tárbeli adatot pedig az

SBC A,(HL) SBC A,(IX+d) SBC A,(IY+d)

utasításokkal tudunk kivonni. A jelzőbitek itt ugyanúgy állítódnak, mint ahogyan azokat a példáknál láttuk.

(folytatjuk) *-HCS-*

A PASCAL

Blaise Pascal, francia matematikus, filozófus, élt 1623-tól 1662-ig. Már 1642-ben, 19 éves korában elkészítette a világ valószínűleg első működő (mechanikus) számítógépét, az arithmométert, amelynek több eredeti példánya máig is működőképesen fennmaradt. Pascal így jogosan tekinthető a számítástechnika legelső úttörőjének Később számos jelentős felfedezést tett a matematika és más tudományok területén. Méltán nevezték el Pascalról a korszerű számítástechnikának az egyik legtökéletesebb és talán legszebb nyelvét.

1. BEVEZETÉS
Miért a Pascal?

A számítástechnika és -tudomány a leggyorsabban fejlődő területe az emberi tevékenységnek. Már nem is évről évre, hanem szinte hónapról hónapra jelennek meg az újabb gépek, programok, és viharos gyorsasággal fejlődnek a programozási nyelvek is. Ennek a szédületes iramú fejlődésnek azonban megvan az árnyoldala is.

A terület harmonikus fejlődésének a menete az volna, ha a felhasználók megmondanák, hogy mire van szükségük; a programozók kifejtenék, hogy milyen szoftvereszközökkel tudnák megoldani ezeket a feladatokat; a rendszerprogramozók kitalálnák, hogy ehhez milyen eszközöket (operációs rendszereket, programnyelveket) kell kifejleszteni; végül a hardvergyártók megalkotnák az ehhez szükséges - és leginkább alkalmas - számítógépeket.

A valóságban ennek kezdettől fogva pont az ellenkezője zajlik: a hardvergyártók újabb és újabb gépeket hoznak létre, melyek mindegyike sokkal nagyobb teljesítményű, mint elődje, de szinte figyelembe sem veszik a felhasználóik tényleges igényeit. A rendszerprogramozók csonkig rágják a ceruzájukat, mire az egyre gyorsabb és nagyobb, de egyáltalán nem átgondolt szerkezetű gépekre a megfelelő rendszerprogramokat kifejlesztik. A kiinduláskor elkövetett hiba tovább gyűrűzik a felhasználóig.

Ennek eklatáns példája a FORTRAN programozási nyelv története. Senki sem vitatja el a nyelv érdemeit (egy generáció nőtt fel rajta, és feladatok tömkelegét oldották meg segítségével); azonban a nyelv illogikus, szedett-vedett szabályaiból is kiviláglik, hogy az elsőrendű szempont nem az volt, hogy mennyire képes a programozót segíteni, hanem, hogy könnyű volt hozzá fordítóprogramot fabrikálni. A szakemberek hiába dolgozták ki ugyanabban az időben a logikusabb és a feladatmegoldásra sokkal alkalmasabb Algol-60 programnyelvet, mindenki a Fortrant használta. Miért? Mert az futott a Nagy Kék, az IBM számítógépein.

A BASIC a Fortran kisöccse. Más előnye nincs, mint hogy könnyű megvalósítani, és hogy rendkívül elterjedt. Szokás sok-sok más előnyét is emlegetni. A szakember számára világos, hogy ezek az úgymond előnyök mind szakmai, mind pedagógiai szempontból valójában hátrányt jelentenek.

A számítástudomány pedig mindig időben létrehozta az adott kor ismeretei szerinti korszerű, jól használható programozási nyelveket. Ezek egyike a Pascal. A mi hibánk, ha nem használjuk ki előnyeit.

Lássuk hát, hogy mit tud a Pascal, mondjuk, a BASIC-kel szemben.

Sebesség

A Pascal nyelven írt program a gépi kódéhoz hasonló - a BASIC után szinte fantasztikusnak tűnő - sebességgel fut. Ez abból következik, hogy a Pascal programot futtatás előtt a fordítóprogram gépi kódú programmá alakítja át. Ez néhány nagyságrenddel gyorsabb futást eredményez, mint a végrehajtás pillanatában értelmezett BASIC programé.

Ennek az érvnek van egy kis szépséghibája, mégpedig, hogy végső soron a BASIC program is lefordítható gépi kódra. A gyakorlatban azonban a BASIC fordítóprogramok nem nagyon terjedtek el: a BASIC tipikusan értelmezett, interpretált nyelv maradt Az IS-BASIC ismert fordítóprogramja, a ZZZIP, túlságosan sok megkötést ír elő a lefordítani szánt programnál; tulajdonképpen nem azt a BASIC változatot valósítja meg, amelyiket a géppel gyárilag adott interpreter ismer. Sajnos, az IS-BASIC nyelvet pontosan egyik legrokonszenvesebb vonásától, a lokális változók használatának lehetőségétől fosztja meg. A ZZZIP fordítóprogramnak ugyanakkor a használata sem túlságosan praktikus, különösen lemezegység nélküli gépen.

A programszerkezetek

A programszerkezetek jelentősége tulajdonképpen nem igazán jelentkezik mindaddig, amíg az elkészítendő program mérete nem haladja meg a húsz-harminc sort. Egy ekkora program mindig áttekinthető marad, akármilyen össze-vissza módon is van szerkesztve. A probléma akkor jelentkezik, amikor nagyobb programot kell (ne adja Isten, határidőre, lehetőleg hibátlanul) készíteni. A program bonyolultsága – és ezzel együtt áttekinthetetlensége, a hibalehetőségek száma – a program méretével nem lineárisan, hanem ennél sokkal erősebben növekszik. A megoldás, ha kizárólag csak olyan jól definiált programszerkezeteket használunk, amelyek jól strukturált, könnyen áttekinthető, könnyen tesztelhető és javítható programot eredményeznek. A BASIC nagy hibája, hogy az egyetlen valódi programszerkezet,

A program hossza és bonyolultsága közötti összefüggés katasztrofális hatását világítsuk meg egy példával. Tegyük fel, hogy a hossztól a bonyolultság négyzetesen függ. Ez azt jelenti, hogy a kétszer, háromszor hosszabb program négyszer, kilencszer bonyolultabb. Most tételezzük fel, hogy egy százsoros program elkészítéséhez egy napra volt szükségünk. Egy komoly ügyviteli program vagy rendszerprogram néhányszor tízezer sorból áll. Elvállaltuk egy tízezer soros program megírását, mondjuk négy hónapos határidővel, abból kiindulva, hogy a százszor akkora program megírásához elég a száz nap. Természetesen minden határidőmódosítást folyamatosan túlléptünk. Ez nem is meglepő, hiszen a négyzetes hossz-bonyolultság összefüggés alapján számolva a program bonyolultsága nem százszoros, hanem tízezerszeres. Ez azt jelenti, hogy a várható elkészítési idő tízezer nap, azaz mintegy 27 év... egy egész dolgos élet!
A valóságban a bonyolultság ennél még nagyobb is lehet.

VERSENYFUTÁS

1.	2.	3.	4.	5.	6.
Üres ciklus	18	0.8	0.34	53	2.35
Egész értékadás	33	0.5	0.26	127	1.92
Egész összeadás	13	0.6	0.41	32	1.46
Egész kivonás	14	0.6	0.46	30	1.3
Egész szorzás	18	4.4	3.3	5.5	1.33
Egész osztás	22	6.6	6.5	3.4	1.02
String értékadás	26	2.5	5.2	5	0.48
Rész-string kiemelése	38	1.2 *)	0.5 *)	–	–
Konkatentáció	24.74	7.4	12.38	2	0.6
Lebegőpontos értékadás	34.5	–1.0	32	–	–
Lebegőpontos összeadás	53	–	2.3	23	–
Lebegőpontos kivonás	55.5	–	2.35	24	–
Lebegőpontos szorzás	78.5		30.9	2.5	–
Lebegőpontos osztás	251.5	–	40.18	6.3	–

1. A feladat (10 000-szer végrehajtva)
A végrehajtási idő (s): 2 BASIC 3. ZZZIP 4. Pascal esetén
A relatív gyorsaság: 5. PASCAL a BASIC-hez 6. Pascal a ZZZIP-hez hasonlítva

A teszt körülményei: normál 4 Mhz-es ENTERPRISE 128-as gép a gyári beállítással. Az összehasonlított programok: IS-BASIC 2.1 verzió, ugyanaz a program (pontosabban a program megfelelő részei) a ZZZIP 1.1 verziójával lefordítva, Turbó-Pascal 2.00A verzió.

A teszt során az. összes művelet eredménye a tisztán ráfordított idő, azaz az értékadás mért idejéből levontuk a ciklus szervezésére fordított időt, az összeadás idejéből emellett levontuk az értékadás idejét is stb. Ebből adódik a rész-string kiemelésénél a Pascal és a ZZZIP esetében kapott negatív időérték (a művelet tehát gyorsabban megy végbe, mint a stringértékadás).

A teszt nem igazán mérvadó, mert az IS-BASIC nem ismeri az egész típust. Így valójában nem képes egész műveleteket végezni. Ezért az „egész” műveleteket a BASIC-ben kis egész értékeket eredményező kis egész értékekkel végeztük. Az eredmények mégis jellemzők a két programnyelvre, mert a sebesség szempontjából legkritikusabb műveleteknél is (ciklusszervezés, karakter- vagy képpont-koordináták számítása) a felhasználó kénytelen a sokkal lassabb lebegőpontos műveleteket használni.

amit ismer, az utasítás vagy utasítássor. A Pascal által ismert programszerkezetek segítségével viszont szinte ideális tisztaságú programokat készíthetünk.

A programszerkezetekhez tartozik a Pascal igen jól definiált és következetes eljárás- és függvénydeklarációja, illetve ezek hívása. Nagyon fontos a paraméterátadás (a beavatottabaknak: az érték szerinti illetve a cím szerinti átadás), a lokális változók (általában a lokális objektumok), és itt említjük meg a rekurziót, mint bizonyos fajta feladatok egyetlen lehetséges, sok más feladatnak pedig egyszerűen csak célszerű megoldását.

Az adatszerkezetek

A BASIC példáján felnőtt programozó az adatszerkezetek fogalmát még kevésbé érti, mint a programszerkezetekét. Az iparszerű programozástechnikában pedig az adatszerkezeteknek a programszerkezetekkel azonos vagy talán még fontosabb szerepük van.

A BASIC csak néhány egészen elemi adatszerkezetet ismer, mégpedig az egész és a valós számokat (sok megvalósítás ezek közül is csak az egyiket), valamint a (karakter)stringet (magyarul füzért). Képes ezekből összetett adatszerkezeteket (tömböket) létrehozni. Ismeri még tulajdonképpen a fiié (a magyar neve állítólag állomány) fogalmát, de a BASIC felfogásában ez nem igazán adatszerkezet.

Ez a szűklátókörűség tisztán csak hagyomány: a számítógépeket legelőször csak numerikus alkalmazásokra szánták (mindenki tudja, hogy a számítógép a pontos célzás igényének köszönheti létét: az első számítógép a tüzérségnél használt lőelemképzőt akarta kiváltani). Ezeknél az első alkalmazásoknál ez a néhány szerkezet is elegendő volt.

Manapság az élet minden területén használják a számító- (helyesebben szólva: adatfeldolgozó) gépeket. Az alkalmazásoknak csak kis hányada dolgozik számokkal, a többségük mindenfajta egyéb objektumokon operál. A Pascal, bár eleve több adattípussal rendelkezik, azt is lehetővé teszi, hogy a programozó tetszés szerinti, új adattípusokat hozzon létre. így a program algoritmusa a feladat megoldásához legjobban alkalmazkodó adattípusokkal dolgozhat.

Gépfüggetlenség

Egy-egy program kifejlesztése igen sok eszközt és ráfordítást igényel. All ez a mi saját programjainkra is, de különösen a hatalmas méretű gyári programokra. Jogos a törekvés, hogy a sokszor több ember több évi munkáját követelő program ne csak egy géptípuson működjön. A számítógépek gyors generációváltása is indokolja, hogy az előző gépre fejlesztett program a következőn is használható legyen. A BASIC nyelvjárásai annyira szerteágazóak, hogy szinte bizonyosra vehető a kudarc, ha egy másik rendszeren fejlesztett BASIC programot próbálunk átvenni. Gondoljunk csak a valamennyire szabványosnak felfogható Microsoft BASIC-re, az IBM gépeken és utánzóikon futó GW-BASIC-re, vagy az ENTERPRISE rokonszenves, de egyáltalán nem szabványos alig-BASIC-jére.

A Pascal nyelvet olyan gondosan tervezték, hogy a legtöbb fordítóprogramja szinte változtatás nélkül valósítja meg. Ugyanakkor igen magas szinten szabványosítják az idők folyamán mégis szükségessé váló módosításokat, kiegészítéseket. Ez garanciát ad arra, hogy a leporolt régi Pascal program változatlanul, vagy igen kicsiny, jól definiálható változtatással lefordul majd a legújabb fordítóprogrammal, és futni fog a következő, még gyorsabb, még többet nyújtó gépen is.

Egyéb

Sokáig lehetne még sorolni a Pascal feltétlen előnyeit a BASIC- szerű programozási nyelvekkel szemben. A teljsségre törekvésre nem is gondolva it most csak megemlítünk néhányat.

A Pascal eljárásai (ezek felelnek meg a BASIC-ben mellesleg nem túl ügyesen megvalósított szubrutin fogalmának) és függvényei para- méterezve hívhatók meg. Ennek elemi szükségességét minden programozó ismeri. További óriási előny, hogy az eljárásokon és függvényeken belül lokális (csak az adott eljárás vagy függvény határain belül értelmezett) változók létezhetnek, amelyek még névazonosság esetén sem léphetnek konfliktusba más eljárások vagy a főprogram változóival.

A Pascal nagy előnye még, hogy rendkívüli nyelvi gazdagsága ellenére könnyen fordítható, és megdöbbentően gyors fordítóprogram készíthető hozzá. Ennek legjobb példája a közismert Turbó-Pascal.

Még egy momentum: a Pascal program- és adatszerkezetei közel állnak a normális emberi gondolkodás fogalmaihoz. Teljesen jogos az az igény, hogy ne az embernek kelljen a számítógép korlátáihoz idomulnia, hanem a gépet emljük fel az ember intellektuális szintjére. Ez ugyanakkor azt is jelenti, hogy a Pascalt megtanulni is könnyű. Ugyanakkor a tapasztalat azt mutatja, hogy akik a BASIC használatával kezdtek el programozni, később alig képesek megszabadulni a beléjük rögzült rossz programozási megoldásoktól. Pedig semmi akadálya nincs annak, hogy már a legelső pillanatban ne BASIC-ben, hanem Pascalban kezdjük tanulni a programozást.

Végül felmerül a kérdés: az ENTERPRISE felhasználói milyen módon próbálkozhatnak meg a Pascal nyelvű programozással? Nos, a csak magnóval rendelkező felhasználóknak elérhető a Spectrumosok által is ismert HiSoft-Pascal, amely egy elég teljes, többé-kevébé jól sikerült megvalósítás. Saját, beépített editorral dolgozik, gyors, és a memóriában fordít, a program azonnal futtatható. Ugyanakkor a kész programból önállóan betölthető, futtatható kód állítható elő. A Hi- Soft-Pascal szépséghibája, hogy - mivel nem ENTERPRISE-ra készült - nem tudja közvetlenül kezelni a gép grafikai és egyéb lehetőségeit. Ez azonban senkit ne keserítsen el: készült ugyanis egy olyan kiterjesztés, amellyel az ENTRPRISE minden előnyös tulajdonsága kihasználható. Ezt a kiegészítést - mivel elég nagy terjedelmű, kénszerűen folytatásokban - közreadjuk.

Akik lemezegységgel is rendelkeznek, a Turbó-Pascal különböző verzióit használhatják. Ez a program megjelenése óta folyamatosan sztár mind az azóta gyakorlatilag már csak a hobbi-kategóriában megtalálható nyolcbites, mind pedig az egyre inkább mindenkinek elérhetővé váló IBM-kompatibilis számítógépeken programozók körében.

(folytatjuk...)
- UL -

A NYELVEK KÖZÖTT

Hogy az adatokat értékelhessük, egy példa: egy 100 x 100 elemű mező nullázása (pl. egy játékmező előkészítése során) a BASIC esetén 51 másodpercig tart) 18 s a ciklusszervezés és 33 s az értékadás), míg ugyanez a ZZZIP-pel lefordított BASIC program esetén csak 1,3 másodpercig (0,8 s és 0,5 s). A Turbo-Pascal esetén a művelet mindössze 0,6 másodpercet igényel (0,34 s és 0,26 s).

Egy érdekesség: ahhoz, hogy az eredmények egyáltalán mérhetők legyenek, a Pascal program és a ZZZIP-pel lefordított BASIC program esetében kénytelenek voltunk nem tízezer, hanem százezer ciklusr lefuttatni, spt, az első négy tesztnél (üres ciklus, értékadás, összeadás és kivonás) egyenesen egymillió műveletet elvégeztetni. A táblázatban már természetesen a korrigált adatot adtuk meg. Majdnem bizonyos, hogy ezzel „előnyt adtunk” a BASIC-nek (a számábrázolás és a műveletvégzés sajátosságai miatt a nagyobb számoknál a sebesség csökken; ez egyébként látható abból, hogy a „lebegőpontos” műveleteknél a BASIC lassabb, mint önnön maga az „egész” műveleteknél, holott tudjuk, hogy mind a két esetben egyaránt lebegőpontos műveleteket végez a gép).

Nézzük például a sakkjáték programozását. BASIC-ben gondolkodva a sakktábla ... a 8-as numerikus (pl. egész típusú) mátrix, az egyes elemeinek adott érték mondja meg, hogy milyen figura áll az adott mezőn. A király értéke legyen 6, a gyalogé 1; az üres mező megfleelője a 0 érték. A figura színének meghatározásához két lehetőségünk van: vagy valahogy belekódoljuk az értékbe a színt, pl. a sötét figura a saját értékével szerepel, a világoséhoz pedig mindig hozzáadunk 10-et, vagy negáljuk, vagy pedig definiálunk egy másik 8x8-as mátrixot, ahol a figura színét adjuk meg. Egyik megoldás sem igazán szerencsés: az első esetben állandóan kódolni és dekódolni kell a figurák értékét, a másik esetben két adatmezőt kell állandóan indexelni és naprakészen tartani. Egyik esetben sem világos valahol a program közepén, hogy a TABLA(1,J)=15
értékadás melyik szín melyik figuráját állítja az adott mezőre.
A Pascal megengedi, hogy definiáljuk a FIGURA típusát. Az ilyen típusú változók és konstansok értéke egyrészt megadják a figura színét, másrészt az értékét is, mégpedig jól érthetően. A

```
TYPE
FIGURA = RECORD
SZINE: (SOTET,VILAGOS);
ERTEKE: (URES, GYALOG, FUTAR, CSIKO, BASTYA, VEZER, KIRALY);
END;
```

típusdeklaráció a sakkfigurák, illetve a sakktábla mezői számára egy olyan típust határoz meg, aminek a SZINE összetevője csak a SOTET és a VILAGOS értéket veheti fel, az ERTEKE összetevő pedig vagy üres, vagy pedig a még felsorolt hat érték egyike lehet. Ekkor - a megfelelő változódeklarációk után - jogosak (és teljesen érthetőek) az ilyen műveletek:

```
AKTUALIS_FIGURA := TABLA [.... 1];
IF AKTUALIS_FIGURASZINE = SOTET THEN
GYOZTEM
ELSE
VESZTETTEM
;
F TABLA [SOR, OSZLOP].ERTEKE = KIRALY THEN
SAKK
;
```

Lehetőségek Páratlan Tárháza (LPT)

Az eddig ENTERPRISE-ra készült programok közül szinte csak a demókból kaphattunk némi ízelítőt számítógépünk grafikai képességeiből. Pedig a számunkra felkínált lehetőségek szinte korlátlanok. Ezeknek a látványos megoldásoknak a megismeréséhez szeretnék némi segítséget nyújtani ebben a rovatban.
Először az LPT felépítését ismertetem, majd kisebb-nagyobb példaprogramokkal próbálok bemutatni néhány trükköt, ötletet Természetesen ezek megértéséhez illetve használatához ismerni kell az assembly programozás alapjait valamint az ENTERPRISE memóriakezelését
Az ENTERPRISE grafikájáért a NICK chip felelős. A működésébe való beavatkozáshoz négy port áll rendelkezésünkre.
Ezen portok és szerepük a következő:
80h FIXBIAS
b0..b4 A felső nyolc palettaszín felső 5 bitje.
b5..b6 A külső színbemenet prioritása.
b7 A beépített hangszóró kapcsolója. (0=bekapcsolva)
81h BORDER A keretszín beállítására használható.
82h LPTL Az LPT (Line Paraméter Table-sorparaméter tábla) kétbájtos kezdőcímének középső nyolc bitje (b4..b11).
83h LPTH
b0..b3 Az LPT kezdőcímének felső négy bitje. b4..b5 Nincs használva.
b0 A sorparaméter számláló órajelének engedélyezése (0=tiltott az órajel).
b7 Ha a bit értéke 0 akkor újratöltődik az LPT bázismutatója.
A NICK chip portjai után lássuk a memóriakezelését A NICK csak 64 kbájtot tud megcímezni, de azt közvetlenül teszi attól függetlent!, hogy az be van-e lapozva vagy nincs. Ezt a grafikus memória területet a felső 4 szegmens alkotja (0FCh..0FFh). Ezeket a Z80 aszerint címzi, hogy melyik lapra vannak belapozva, a Nick azonban folytonosan 0000h-tól 0FFFFh-ig. A videó memóriához való hozzáférésben Nick chipnek prioritása van, ezért itt a programok lassabban futnak és nem időzíthetők.
Minden LPT egyenként 16 bájto6 sorparaméter blokkokból áll. Egy-egy LPB (Line Paraméter Block-sorparaméter blokk) a képernyőnek egy részét definiálja. Ennek a képernyőrésznek MODSOR a neve.
Egy sorparaméter blokk felépítése a következő:
SC Az LPB által meghatározott MODSOR pixelsorainak száma kettes komplemens alakban. (256-sorok száma)
1 MB Az adott MODSOR videó üzemmódja.
b7 VINT Beállítva a MODSOR kezdetén megszakítás generálódik.
b6..b5 Szín mód.
00 - Kétszínű üzemmód
01 - Négyszínű üzemmód
10 - 16 színű üzemmód
11 - 256 színű üzemmód
b4 VRES Beállítva teljes a függőleges felbontás, törölve az adott MODSOR minden pixelsora azonos lesz.
b3..bl Videó mód.
000 - VSYNC mód (szinkronizálási mód)
001 - PIXEL mód
010 - ATTRIBUTE mód
011 - CH256 mód
100 - CH128 mód
101 - CH64 mód
110 - nem használt
111 - LPIXEL mód
b0 RELOAD Beállítva újratöltődik a sorparaméter számláló báziscíme.
2 LM b0..b5 Bal margó.
b6..b7 Színvezérlés kétszínű grafikus módban.
3 RM b0..b5 Jobb margó.
b6..b7 Színvezérlés kétszínű karakteres módban.
4...5 LD1 Elsődleges videó adatcím.
6...7 LD2 Másodlagos videó adatcím.
8...15 COL0-COL7 Első nyolc palettaszín. 256 színű üzemmódban nem használt.

Egy LPT definiálásához ismerni kell a TV-kép felépítését A teljes kép 625 sora két egyenként 3125 soros fél képből áll. Az egyik fél kép tartalmazza a páros sorszámú sorokat a másik a páratlanokat Ahhoz tehát, hogy a kép ne hullámozzon pontosan 3125 sorból kell állnia az LPT által definiált MODSOR-oknak. Ilyenkor a két fél kép teljesen megegyezik egymással. Természetesen lehetőség van a teljes 625 soros képernyő kihasználásához. Ezt interlace üzemmódnak hívják. Ilyenkor az egyik fél képdefiníció végén nem állítjuk be a RELOAD bitet, hanem folytatjuk a második fél képpel és csak ennek a végen utasítjuk a Nick-et, hogy kezdje élőről az LPT olvasását
A tökéletes állóképhez még egy dologra kell ügyelnünk, mégpedig a szink- ronizációra. A pixelsorok vízszintes szinkronizációját a gép elvégzi helyettünk, a függőleges képszinkront azonban nekünk kell biztosítani. Ezt legegyszerűbben a következő két LPB-vel tehetjük meg:
0FCh,10h,00h,3Fh,00,00,00,00,00,00,00,00,00,00,00,00
0FFh,10h,3Fh,20h,00,00,00,00,00,00,00,00,00,00,00,00
Szinkrortizációs módban a bal és a jobb margó a szinkron be- illetve kikapcsolásának a helyét vezérli a soron belül. Az első blokk négy pixelsort definiál (100h–FCh=04h). Ezen LPB legelején bekapcsolódik a szinkronizáció és bekapcsolva is marad egészen a második blokk feléig. Mivel a második LPB csak egy pixelsort jelent ezért a szinkron összesen 4.5 sor idejére van bekapcsolva. Tehát még 3125–4.5=308 sort kell definiálnunk.

A Nick minden sort 57 szakaszra oszt fel és minden szakaszban két bájtot tud beolvasni. Az első nyolc szakaszban beolvassa a soronkövetkező LPB-t Ezután a beolvasott sorparaméter blokk alapján előállítja az aktuális képsort A sor a képernyőn a bal és jobb margó által meghatározott helyre kerül. Az utolsó három szakaszban történik a videó memória frissítése. A bal margó legkisebb lehetséges értéke tehát nyolc, a jobb margó pedig maximum 36h lehet A margókon kívül eső terület keretszínű lesz.
A videó módokat 3 fő részre oszthatjuk. Ezek a grafikus mód, a karakteres és a szinkron üzemmód. A szinkron problémájával már foglalkoztunk, most lássuk a grafikus képmódokat
Ebből három féle áll a rendelkezésünkre: pixel, attribútum és lpixel. A pixel és lpixel módok között csak a vízszintes felbontásban van különbség, mégpedig az lpixel mód fele akkora felbontású mint a pixel. így memóriát takaríthatunk meg, ha megelégszünk a kisebb felbontással. Ezen videó módokban a képernyőbájtok jelentése a színmódtól függ.
Kétszínű módban minden bitnek egy képpont felel meg. Ha a megfelelő bit törölve van, akkor az adott képpont a Col0-nak megfelelő színt veszi fel, ha a bit egyes, akkor a Col1-et
Négyszínű képernyőn minden bájt négy képpontot határoz meg a következő bitkiosztással:
b3,b7 b2,b6 bl,b5 b0,b4
Az egyes bitpárok egy-egy színt határoznak meg a Col0-Col3 palettaszínek közül.
16 színű módnál kell legjobban odafigyelni a bitkiosztásra, amely az alábbi:
bl,b5,b3,b7 b0,b4,b2,b6
Mint látható, itt már csak két pontot jelent egy bájt Mivel az LPT-ben csak nyolc palettaszint tudunk megadni, a második nyolc (Col8-Col16) megadásához a fentebb ismertetett FIXBIAS portot kell felhasználni. Ebből következik, hogy a második nyolc szín csak olyan egymás után következő szín lehet, amelyeknek első tagja nyolccal osztható pozíción kezdődik. Ez a második nyolc szín az egész képernyőn ugyanaz.
Már most érdemes megfigyelni, hogy a fenti módok bármelyikében a kép egypixelnyi scrollozása egyetlen bitforgató utasítással lehetséges.
A legegyszerűbb videó mód a 256 színű. Ennél minden képpontnak egy bájt felel meg. A képpont színe a bájt értékének megfelelő szín lesz. Itt tehát az LPT paletta színeit nem használjuk.
A következő részben megnézzük a fennmaradó videó módokat, valamint egy példaprogramot az LPT felépítésére és bekapcsolására.

- DEVIL -

Videofüggvények

Sokféle megoldás létezik már a megnyitott videocsatornák kezdőcímének meghatározásához. Talán a legelegánsabb az, amikor a videolap szegmens és ofszetcímét meghatározó függvényeket a Basic alaputasításai közé láncoljuk. Ezt teszi meg az alábbi assembly forráslista. A fordítást a GEN-nel végezhetjük el, 2-es fejlécű relokálható állományt kell készíteni. Miután az 1. listát begépeltük, a biztonság kedvéért rögzítsük azt a

P1,99,VFUNC.ASM

utasítássorral. A fordítás az

A,,R 2

Paranccsal indítható. A tárgykódot az

O,,VFUNC.BAS

kiadásával rögzíthetjük.
A lefordított programot Basic-be természetesen a LOAD paranccsal tudjuk betölteni. Ekkor két új függvény, a VSAG() és a VOFS() könnyíti a lapkezdőcímek meghatározását. A függvények paraméterként egy nyitott videocsatorna számát várják. Adjuk ki például a

PRINT VSEG(102);VOFS(102)

utasítást. A képernyőn ekkor a nullás editorcsatornához rendelt videolap kezdőcímének szegmens és ofszetcíme jelenik meg.

Akiknek nincs birtokukban a GEN Assembler, a 2. listán látható program lefuttatása után állíthatják elő a bővítőprogramot.

2. lista

```
100 PROGRAM ,,VFUNC.INI"
110 ALLOCATE 230
120 CODE P=HEX$(,,0,02,9F,00,1F,00,00,00")
130 CODE =HEX$(,,00,00,00,00,00,00,00,00")
140 CODE =HEX$(,,2B,14,C8,A4,71,40,A4,40")
150 CODE =HEX$("61,37,19,04,05,62,79,18")
160 CODE =HEX$(,,A6,28,14,88,0C,27,23,28")
170 CODE =HEX$(,,80,E4,65,30,99,0F,22,E0")
180 CODE =HEX$(,,68,28,48,21,83,60,00,74")
190 CODE =HEX$(,,00,00,43,08,00,1A,E8,00")
200 CODE =HEX$(,,00,18,00,0E,02,3D,C2,07")
210 CODE =HEX$(,,A0,08,78,00,11,99,5F,EF")
220 CODE =HEX$(,,70,43,24,00,00,06,01,0A")
230 CODE =HEX$(,,C4,F2,31,4E,07,00,00,00")
240 CODE =HEX$(,,00,00,60,10,AC,53,22,91")
250 CODE =HEX$(,,E0,60,00,01,98,05,C0,1C")
260 CODE =HEX$(,,C3,F6,6C,2F,00,64,B3,60")
270 CODE =HEX$(,,70,03,99,80,07,03,BD,9F")
280 CODE =HEX$(,,86,F5,7B,36,10,00,32,5A")
290 CODE =HEX$(,,E9,90,00,E0,3A,07,05,00")
300 CODE =HEX$(,,04,02,76,B8,2C,00,7C,5B")
310 CODE =HEX$(,,88,02,C7,D0,30,0O,EE,0B")
320 CODE =HEX$(,,5B,88,02,66,03,49,F1,92")
330 CODE =HEX$("67,6B,80,80,0C,97,88,84")
340 CODE =HEX$(,,7C,4E,73,BC,44,2E,A0,1B")
350 CODE =HEX$(,,9D,E2,21,2E,06,DC,EF,11")
360 CODE =HEX$(,,0B,A0,06,E7,C3,00,0A,00")
370 CODE =HEX$(,,00,00,00,00,00,00,00,00")
380 CODE Q=HEX$(,,0,00,00,00,00")
390 OPEN #1:"VFUNC.BAS" ACCESS OUTPUT
400   FOR R=P TO Q+4
410 PRINT #1:CHR$(PEEK(R));
420 NEXT
430 CLOSE #1
```

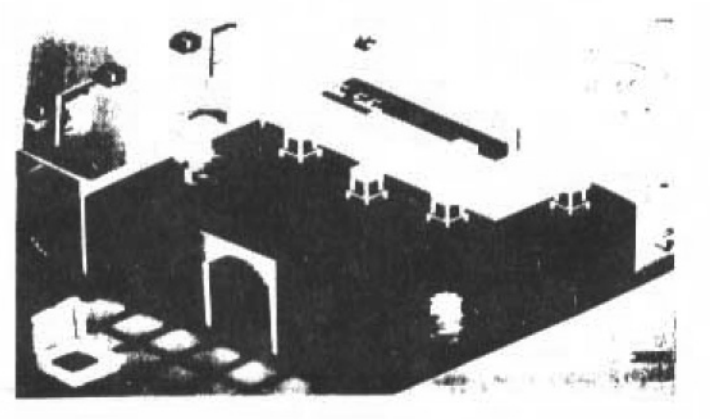
1. lista

```
                        ;VSEG() and VOFS()
                        ;HiSoft DEVPAC/GEN forrást ista
MS0     DEFM "VSEG() and VOFS() are ready."
        DEFB 13,10      ;BEJELENTKEZÉS
MS1     DEFB"$"         ;
START   ENT $           ;INDÍTÁSI PONT
        LD HL,DV0FS     ;
        RST #10         ;ROM BASIC CALL
        DEFB #80,0      ;VOFS BELANCOLASA
        LD HL,DVSEG     ;
        RST #10         ;ROM BASIC CALL
        DEFB #80,0      ;VSEG BELANCOLASA
        LD B,0          ;LEKERDEZES
        LD C,4          ;DEF CHAN
        RST #30         ;EXOS CALL
        DEFB 16         ;ALAPÉRT. CSAT.
        LD A,D          ;
        LD BC.MS1-MS0   ;UZENETHOSSZ
        LD DE,MS0       ;
        RST #30         ;EXOS CALL
        DEFB 8          ;ÜZENET KIIRASA
        RET             ;
DVOFS   DEFW 0          ;ÜRES
        DEFB 12         ;GÉPI FV
        DEFB 4          ;NEVHOSSZ
        DEFM "VOFS"     ;FVNEV
        DEFW GOVOFS     ;A FV CÍME
        DEFB 0          ;P0-N LESZ
DVSEG   DEFW 0          ;ÜRES
        DEFB 12         ;GÉPI FV
        DEFB 4          ;NEVHOSSZ
        DEFM ,,VSEG"    ;FVNEV
        DEFW GOVSEG     ;A FV CÍME
        DEFB 0          ;P0-N LESZ
GOVOFS  CALL RUTIN1     ;ELSO RUTIN HIVASA
        AND 00111111B   ;OFSZET MASZK
        CALL RUTIN2     ;MÁSODIK RUTIN HIVASA
        RET             ;
GOVSEG  CALL RUTIN1     ;ELSO RUTIN HIVASA
        AND 11000000B   ;SZEGMENS MASZK
        RLCA            ;BELEPTETES JOBBRÓL
        RLCA            ;BELEPTETES JOBBRÓL
        OR 11111100B    ;ELSO VIDEOSZEGMENS
        LD L,A          ;
        XOR A           ;A=0
        CALL RUTIN2     ;MÁSODIK RUTIN HIVASA
        RET             ;
RUTIN1  RST #10         ROM BASIC CALL
        DEFB #99,0      ;() ELLENORZÉS
        JR C,ERR1       ;SYNTAX ERROR
        RST #10         ;ROM BASIC
        DEFB #A,0       ;INTEGER ÉRTÉK?
        JR NZ,ERR4      ;NEM INTEGER
        RST #10         ;ROM BASIC CALL
        DEFB #6,0       ;EGESZ A HL-BE
        LD A,H          ;
        OR A            ;A=0?
        JR NZ,ERR2      ;NEM,HL NAGYOBB,MINT 255;
        LD A,L          ;CSATORNASZÁM
        LD B,3          ;VIDEOLAP KEZDOCÍM
        RST #30         ;EXOS CALL
        DEFB 11         ;SPEC. FUNKC.
        OR A            ;VOLT HIBA?
        JR NZ,ERR3      ;IGEN,A CSAT. NINCS NYITVA
        LD H,B          ;
        LD L,C          ;HL-BE A NICK CÍM
        LD A,H          ;SZEGMENSSZAMHOZ
        RET             ;
RUTIN2  LD H,A          ;
        RST #10         ;ROM BASIC CALL
        DEFB 2,0        ;HL A BASIC TAR TETEJERE
        RET             ;
ERR1    POP AF          ;A CALL MIATT
        LD HL,20030     ;"Identifier excepted"
        RST #20         ;HIBAÜZENET ES MEGSZAKÍTÁS
ERR2    POP AF          ;A CALL MIATT
        LD HL,1002      ;"Overflow in numeric ..."
        RST #20         ;HIBAÜZENET ES MEGSZAKÍTÁS
ERR3    POP AF          ;A CALL MIATT
        LD HL,7004      ;,,Channel not open"
        RST #20         ;HIBAÜZENET ES MEGSZAKÍTÁS
ERR4    POP AF          ;A CALL MIATT
        LD HL,1000      ;,,Unexcepted value given"
        RST #20         ;HIBAÜZENET ES MEGSZAKÍTÁS
        END             ;FORDITÁS VÉGE
```

Intelligens billentyűzetkezelő

Bizonyára mindenki tapasztalta már,hogy az ENTERPRISE megjegyzi a legutoljára lenyomott billentyűt. Ez a jelenség sokszor zavaró, általában azonban jó lenne, ha többet jegyezne meg. Ezen a problémán egy rendszerbővítővel lehet segíteni, amely tárolja a leütött billentyűk kódját.
Az alábbi Basic program egy ilyen rendszerbővítőt állít elő és ment ki háttértárolóra. Elindítása előtt tehát állítsuk felvételre a magnót vagy tegyünk lemezt a lemezegységbe.
A bővítő hiányossága, hogy nem nyitja meg újra a billentyűzetcsatornát, amelyet az EXOS lezár a bővítő inicializálásakor. Ezért betöltése után érdemes egy meleg RESET-et végrehajtani.

```
100 PROGRAM „KEYBPC.BAS”
110 OPEN #1 :„KEYBPC” ACCESS OUTPUT
120 LET ELLENOR=0
130 FOR 1=1 TO 726
140 READ A
150 PRINT #1:CHR$(A);
160 LET ELLENOR=ELLENOR+A 170 NEXT I 180 CLOSE #1
190 TEXT 40
200 IF ELLEN0R<>64449 THEN
210 PRINT „HIBA VAN AZ ADATOKBAN”
220 ELSE
230 PRINT „KÉSZ”
235 END IF
240 DATA 0,7,100,2,0,0,0,0,0,0,0,0,0,0,0
250 DATA 0,60,191,128,98,129,51,248, 16,192,29,69,160,37,184
260 DATA 128,26,219,89,140,165,144, 0,70,85,0,0,64,0,7
270 DATA 184,85,94,50,136,192,66,24, 5,4,50,89,47,132,75
280 DATA 5,240,56,16,201,106,177,89, 176,128,1,128,245,31,63
290 DATA 194,48,176,0,2,19,0,61,193, 15,22,11,68,112,12
300 DATA 31,63,194,48,192,0,2,71,0, 61,193,10,242,123,36
310 DATA 240,183,27,178,4,48,56,1, 252,4,96,4,195,75,230
320 DATA 0,140,32,249,87,178,73,100,82.201.8.64.32.59.25
330 DATA 78,71,51,73,188,220,32,24,139.134,0,
    160,104,128,64
340 DATA 32,16,8,4,4,147,113,208, 202,108,54,26,76,230,83
350DATA113,208,64,107,50,158,76,70,243,9,200,200,32,55,156
360 DATA 12,167,35,9,208,222,114,23,2,129,162,1,0,128,64
370 DATA 32,16,32,4,3,17,200,228,96, 32,34,25,78,198,147
380 DATA 97,204,222,102,58,8,9,7,83, 113,156,194,114,60,139
390 DATA 129,64,208,0,0,0,0,16,0,32,108,0,0,0,8
400 DATA 37,145,75,36,34,121,4,164, 68,132,192,16,96,6,11
410 DATA 0,193,96,8,40,1,5,128,96, 208,12,30,1,131,128
420 DATA 48,128,6,12,0,194,32,24,8, 1,2,192,7,207,241
430 DATA 148,118,1,87,140,149,171,230,72,233,90,
    190,91,178,25
440 DATA 176,72,4,116,203,95,185,129,100,119,105,
    17,64,50,101
450 DATA 175,222,193,6,104,1,222,205,115,59,132,
    42,245,243,245
460 DATA 150,1,39,10,1,167,229,187, 0,106,62,0,179,89,142
470 DATA 225,147,45,126,192,56,15,128,140,214,
    99,184,100,172,95
480 DATA 43,217,173,7,104,112,80,89,98,132,103,84,0,169,96
490 DATA 190,118,148,214,11,224,152,233,100,190,
    91,138,4,226,50
500 DATA 120,24,0,237,88,63,135,227,128,232,248,
    124,211,80,43
510 DATA 218,106,17,240,253,166,168,105.171.0.32.0.0.44.240
520 DATA 177,16,62,213,237,53,67,77,88,193,12,1,199,131,37
530 DATA 146,249,130,121,9,14,153,203,245,184,
    128,10,62,16,140
540 DATA 158,75,241,211,65,126,50,89,175,135,76,
    197,250,221,152
550 DATA 186,118,50,79,203,113,64,81,182,180,115,2,5,0,161
560 DATA 11,21,126,182,105,173,21, 236,209,227,45,
    94,60,12,34
570 DATA 68,43,53,243,53,146,237,17,181,25,46,
    214,128,233,110
580 DATA 190,39,187,75,107,85,240,132,80,189,118,
    148,157,98,0
590 DATA 72,88,92,162,20,1,21,74,97, 0,68,242,166,16,3
600 DATA 68,34,5,232,24,20,119,17, 132,31,131,37,162,249,146
610 DATA 47,83,29,199,76,245,248,201,104,190,97,
    196,151,246,108
620 DATA 35,0,97,157,221,236,216,58, 0,195,82,187,217,176,92
630 DATA 1,134,156,104,179,96,136,3,13,164,194,
    102,192,176,6
640 DATA 26,81,222,205,128,160,12,54,107,189,
    198,245,106,132,64
650 DATA 8,5,187,180,164,219,89,180,214,44,56,5,0,175,105
660 DATA 172,218,47,23,27,36,0,0,0,0,0,0,0,0,0
670 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
680 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
690 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
700 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
710 DATA 0,0,0,1,128,0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```



TIPPEK – TRÜKKÖK

Hol a kurzor?

A videolapokon a kurzort a PRINT AT utasítással állíthatjuk be. Hiányoznak azonban ennek fordítottjai, a kurzorpozíció olvasását támogató függvények. Szerencsére a videokezelő rendelkezik ezzel a funkcióval, csupán a megfelelő rutinnal kell felébresztenünk ezt a lehetőségét. A kurzor y koordinátáját a WHEREY, az x koordinátáját! a WHEREX függvény adja vissza. Az egyetlen szükséges paraméter egy nyitott videocsatorna számát tartalmazza.

```
100 PROGRAM "UHEREXY.BAS"
110 PRINT #102,AT 7,13:;
120 LET Y=WHEREY(102):LET X=UHEREX(102) 130 PRINT Y,X
140 DEF WHEREY(C)
150 LET WHEREY=WHERE(C,0)
160 END DEF 170 DEF WHEREX(C)
180 LET UHEREX=WHERE(C,1)
190 END DEF
200 DEF WHERE(C,M)
210 PRINT #C:CHR$(27);"?";
220 GET #C:Y$:GET #C:X$
230 IF M=0 THEN
240       LET WHERE=0RD(Y$)-32
250 ELSE
260       LET WHERE=ORD(X$)-32
270       END IF
280 END DEF
```

EXDOS képernyő

Néha rettentően idegesítő az EXDOS parancsértelmezőnek az a tulajdonsága, hogy a begépelt parancs a megjelenített directory-val, vagy a kiíratott file-al együtt elúszik. Basic-ben dolgozva segíthetünk ezen a helyzeten, ha a képernyő utolsó négy sorának valamelyikében kiadjuk az

:EXDOS #0

parancsot. Ezután parancsainkat a nullás editorcsaíoma dolgozza fel, de az összes megjelenítés az EXDOS saját videocsatornáján fog történni. A rendszerparancsok stabilan állnak a képernyő alsó négy sorában, és a kettőspontot sem kell eléjük kitennünk. A parancsmódból a [STOP] leütésével léphetünk ki, az [ESC] nem hajlandó erre.

A LoriGRAPH gyorsítása

A LoriGRAPH (szegényes) ikonos menürendszeren keresztül irányítható, így viszonylag könnyen tanulható. Amikor a felhasználó már igazán jól tudja kezelni, akkor nem nézi, hogy mi van a képernyőn, hanem mindent „csuklóból" csinál. Ebben az esetben nagyon bosszantó lehet a program lassúsága. Ez főleg a grafikus képernyőkön való mozgásoknál jelentkezik. A lustaságot nem az alkalmazott programnyelv, hanem a billentyűzet okozza.

Az editor betöltő része Basic-ben íródott, ezért ebbe a részbe könnyedén belejavíthatunk. A billentyűzet gyorsítására is van lehetőség, módosítani kell a RATE KEY és a DELAY_KEY rendszerváltozók értékét. Először a IX)AD paranccsal töltsük be a LoriGRAPH-ot, majd Írjuk a következőket:

```
271  SET KEY DELAY 5
272  SET KEY RATE 3
```

Az itt megadott értékek természetesen nem kötelezőek, mindenki a maga ízlése szerint állíthatja be őket. Ha mindkét rendszerváltozó egyenlő eggyel, akkor a LoriGRAPH kezeléséhez már nagyon gyors ujjak kellenek.

Németből angolba szoftverúton

A következő rövid Basic program a német- és a kétnyelvű gépek tulajdonosainak nyújthat segítséget, és a lemezegységgel rendelkezőknek is hasznos lehet. Sok olyan program van, amely csak angol gépen fut, vagy csak magnóról tölthető be. Ezen segít ez a program, amely a számítógépet alapkiépítésű angol géppé változtatja, mindenfele fizikai beavatkozás nélkül.

```
10 FOR A=0 TO 16
20 READ B
30 SPOKE 255,11193+A,B
40 NEXT A
50 SPOKE 255,16279,189
60 SPOKE 255,16277,189
70 SPOKE 255,16278,171
80 EXT "BASIC"
90 DATA 0,0,0,0,0,0,0,0,1,0,0,0,5,0,0,0,0
```

A program angol gépen is futtatható, ha a 90-es sorban az 5-ös szám helyett 4-est írunk, ugyanis az angol gépen a Basic cartridge a 4-es szegmensen van.

Tehát a program utolsó sora angol gépre a következő:

```
90 DATA 0,0,0,0,0,0,0,0,1,0,0,0,4,0,0,0,0
```

String a kéz alá

Egy jól megírt programtól elvárható, hogy lehetőleg minden bevivendő adatot a felhasználó keze alá adjon, megmentve őt a monoton bebillentyűzéstől. Erre ad egy Basic nyelvű megoldást a listán látható program. A függvénynek sorrendben a következő paramétereket kell átadnunk: promptszöveg, előre definiált sztring, y koordináta, x koordináta. Az ASCII 184 a kurzor balra mozgatását idézi elő – a sztringben is.

```
100 PROGRAM „INPSTR.BAS"
110 LET M$=„Kedvenced:"
120 LET A$="ENTERPRISE"
130 LET A$=INPSTR$(M$,A$,3,3)
140 PRINT AS
150 DEF INPSTR$(P$,Z$,Y,X)
160 FOR K=1 TO LEN(Z$)
170 LET Z$=Z$&CHR$(184)
180 NEXT
190 LET P$=P$&" „&Z$
200 INPUT AT Y,X,PROMPT P$:Z$
210 LET INPSTR$=Z$
220 END DEF
```

A vonalzósor beállítása

Aki dolgozott már igazi szövegszerkesztővel, az megszokta, hogy a vonalzósor (ruler line) tetszőlegesen beállítható a szövegben – mondjuk – megjegyzés-sorként megadott minta alapján. Az ENTERPRISE gép beépített szövegszerkesztőjével is csinálhatunk hasonlót. Lépjünk be a szövegszerkesztőbe, és írjuk be például a következő két sort:

```
L . 1 . 2 . 3 . 4 . 5 . 6 R
```

ALT-F2 [CTRL-cursor-right CTRL-F3]* ALT-F4

Mentsük ki a PRINT paranccsal (nem a SAVE-vel) például R65 (65 karakter széles vonalzósor) néven. Most álljunk rá a kurzorral az első sorra, és hajtsuk végre a második sorban súgott utasításokat: ALT-F2 (töröljük a TAB pozíciókat); CTRL-kurzor-jobbra és CTRL-F3 ismételve, míg ki nem érünk a jobb margóra (az új TAB pozíciók beállítása); ALT-F4 (a jobb margó beállítása). Ezután törölhetjük a két sort. Ha újra szükség lesz a beállításra, a kurzorral szövegmentes helyre állva újra betölthetjük; mivel ASCII file-ként van elmentve, a szöveget érintetlenül hagyva hozzáíródik. Hasonló módon tetszőleges vonalzósormintákat készíthetünk. Ha gyakran kell a vonalzósort váltogatni, a mintákat tartsuk RAM-diszken, gyorsabb lesz a betöltés.

GET DEXTER

Az ENTERPRISE felhasználók többsége a Spectrum átiratok mellett szívesen látna egy-egy valódi ENTERPRISE játékot olyan grafikával, amely megmutatja az ENTERPRISE grafikai lehetőségeit. A GET DEXTER sajnos nem az a játék, mely ilyen szuper grafikával rendelkezne, viszont ennek ellenére azt hiszem, élvezet játszani Reni Herbulot programjával.

2119-et írunk. A Traux 219 nevű bolygón lévő űrállomáson fél évvel ezelőtt felrobbant a laboratórium, és olyan mérgező gáz öntötte el a központot, amely felgyorsítja az öregedés folyamatát. A modern technika (és a szerencse) segítségével nyolc tudós kivételével mindenkit sikerült megmenteni. A bolygóról érkezett jelentések szerint éppen nyolc, ellenszerrel teletöltött injekciós tű van az űrállomáson szétszóródva (micsoda véletlen!), amelyet viszont be kell adni az ott rekedt embereknek, hiszen ők már mozogni is alig bírnak, valamint az ajtók kinyitásához szükséges kódkulcsokat sincsen erejük felemelni és használni. Ez a veszélyes feladat Dexterre, a Nemzetbiztonsági és Életfenntartási Szolgálat egyik tisztjére (azaz ránk) vár. Dexterre rábízták ezt a feladatot, de figyelmeztették, hogy a hazatérése csak úgy biztosított, ha megvan a főkapu kinyitásához szükséges nyolc számból álló kód. Ezeket a napról-napra öregedő ottrekedt emberektől kaphatjuk meg, viszont mindegyikük csak egy kóddal rendelkezik.

A játék betöltése után a pályákat nézhetjük végig demó formájában. Már itt megfigyelhetjük, hogy a zenéről sajnos megfeledkeztek a program készítői. Ha már meguntuk ezt a szuper, de nem létező zenét, akkor nyomjuk le a SPACE billentyűt, és hagyjuk addig lenyomva, amíg el nem indul a játék, azaz

a DEMO szó helyett a “CODE” felirat jelenik meg a képernyő alsó részén. Az irányítás bármelyik joystickkel (Int; Ext1; Ext2) vagy a “Q”, “A”, “I”, “O”, “SPACE” billentyűkkel történik, valamint a további billentyűk a következők:

ENTER - tárgy felvétele

D - tárgy letétele (DROP)

P - tárgy elmozdítása (pl. ha egy tű az ágy alatt van)

R - fütyülés Bird X-nek, a velünk lévő madárnak.

Első pillantásra a szobában rohangáló robotokon illetve Bird X-en akad meg a szemünk. A robotokkal való közvetlen érintkezés természetesen nem tanácsos, ennek jele az ENERGY felirat után lévő szám rohamos csökkenése. Ha az energiánk már vészesen a nulla felé közeledik, menjünk valamelyik benzinkúthoz hasonló energiafeltöltő fülkéhez, ahol Dexter újra visszakapja életkedvét. És a kis madarunk haszna mi lehet vajon? Hát például nem vagyunk egyedül. De evvel még semmire se megyünk, a robotok úgysem hagynak minket unatkozni. Hát akkor vajon mire való? “Megvan! Heuréka! Megtaláltam!” Segítségével ugorhatunk fel néhány helyre úgy, hogy fütyülünk neki, és őrá ugrunk először, majd róla ugrunk fel a kiszemelt helyre.

Tehát Dexter feladata a megöregedett emberek meggyógyítása a fecskendők segítségével, melyekért cserébe a papák kódokat adnak. Eddig oké, ezt a térkép segítségével könnyen meg lehetne csinálni, csak ne lennének ott azok a fránya robotok. Viszont ezeket az önműködő energiazabálókat is el lehet pusztítani a következő tárgyakkal:

Rajzszög - kacsa, porszívó

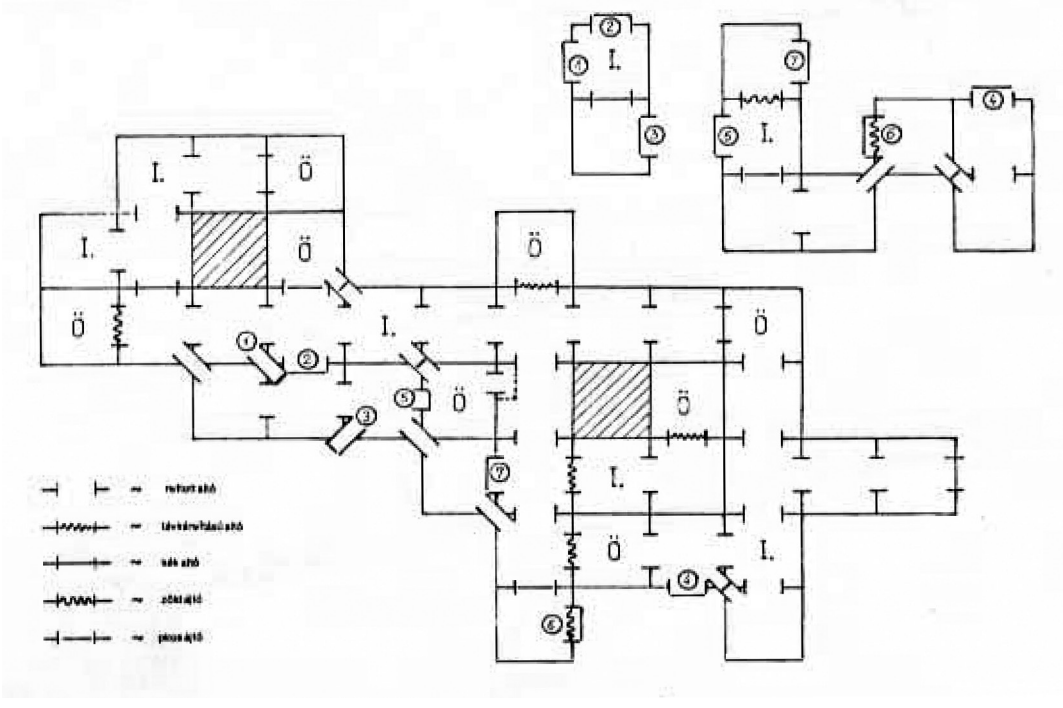
Gránát - kacsa, kutya (amely inkább hasonlít egy négy lábon járó lapostetűhöz)

Virág - a robotlányok elcsábítására szolgál. (Úgy látszik csak az embereknél igaz az a közmondás, hogy aki szereti a virágot, az rossz ember nem lehet.)

Borosüveg - a robotfiúk szeretik nagyon a bort, hát adjuk oda nekik.

A játék nehezítésére szolgál az is, hogy néhány ajtó csak kódkulccsal nyílik. (Nehogy már véletlenül könnyebb legyen a játék!). Természetesen az ajtó színének megfelelő színű kódkulccsal nyílnak az ajtók (lásd térkép!). Ha már szenvedtünk két órát a játékkal, s meggyógyítottunk minden öregembert, akkor menjünk a térkép jobb szélén lévő szobába. Itt elkezd villogni az első kód. Menjünk rá valamelyik négyzetre. Ha szerencsénk van, nem esik ránk semmi. Ha nincs? Kezdjük újra a játékot! Talán sikerül egyszer. Vagy nem is ezt kell csinálni? Ha valaki tud jobbat, írja meg! Előre is köszi! És végül egy kis segítség a játékhoz. Nyomjuk le mind a nyolc funkcióbillentyűt egyszerre. (Gyengébbek kedvéért a gép billentyűzetének tetején lévő kék billentyűket.) Lássunk csodát! Ha azt nem is látunk, de a keret színe átváltozott. Jé, és az energiánk se fogy. Így talán könnyebb lesz végigjátszani eme bonyolult játékot.

Jó szórakozást minden billentyűzet koptatónak. És még egyszer. Kérem azokat, akiknek van valami használható ötletük a játék végéhez, írják meg. A címet, gondolom, már minden ENTERPRISE-os tudja.



ÖRÖKÉLETKÓDOK

Az első számban leírtuk, hogy az örökéletesítést milyen formában adjuk meg. Azoknak, akik nem jutottak volna hozzá, most ismét leírom a lényeget. Az átalakításhoz az ASMON (SIMON) című assemblert használjuk. Szögletes zárójelben az egyes funkciókat elindító billentyűk vannak, a számokat és a címet a megjelenő kérdésre, illetve a memória dumpba kell beírni. A „last address" üzenetet az ASMON írja ki, megadva a betöltés végcímét. Ha nem a megadott címet kapja, akkor az egy másik verzió. Arra a továbbiak nem érvényesek.
A használt funkciók a következek: [R] - betöltés, [M] – módosítás, [S] – felvétel.

CAULDRON
[R] 10F0 [ENTER] BFFF [ENTER] CAULDRON [ENTER]
"Last address: 3EEF"
-[M] 15C6 [ENTER] 18 [ESC]
-[M] 1AB8 [ENTER] 18 [ESC] ij -[M] 2463 [ENTER] B7 [ESC]
-[M] 246C [ENTER] B7 [ESC]
-[M] 256B [ENTER] 00 [ESC]
-[S] 10F0 [ENTER] 3EEF [ENTER] CAULDRON [ENTER]

UNICUM
-[R] 10F0 [ENTER] BFFF [ENTER] UNICUM [ENTER]
"Last address: 15EF"
-[M] 10F2 [ENTER] FA [ESC]
-[M] 116C [ENTER] F0 05 [ESC]
-[M] 1270 [ENTER] F5 3E B7 32 35 13 FI C3 00 10 [ESC]
-[S] 10F0 [ENTER] 15F9 [ENTER] UNICUM [ENTER]

KING OF THE CASTLE
-[R] 10F0 [ENTER] BFFF [ENTER] KING
"Last address: 1242"
-[M] 1148 [ENTER] AF 32 EF 22 C3 00 01 00 [ESC]
-[M] 1136 [ENTER] 5C [ESC]
-[S] 10F0 [ENTER] 1242 [ENTER] KING [ENTER]

HUNCH BACK
-[R] 10F0 [ENTER] BFFF [ENTER] HUNCHBA [ENTER]
"Last address: 131F"
-[M] 12FE [ENTER] 20 03 [ESC]
-[M] 1320 [ENTER] AF 32 08 69 C3 7C 60
-[M] 10F2 [ENTER] 27 [ESC]
-[S] 10F0 [ENTER] 1326 [ENTER] HUNCH BA [ENTER]

JETMAN1
-[R] 10F0 [ENTER] BFFF [ENTER] JETMAN1 [ENTER]
"Last address: 131D"
-[M] 11FF [ENTER] 20 03 [ESC]
-[M] 1320 [ENTER] 3E E0 32 66 90 3E 03 32 51 90 C3 00 80 [ESC]
-[M] 10F2 [ENTER] 2D [ESC]
-[S] 10F0 [ENTER] 132C [ENTER] JETMAN1 [ENTER]

GAME OVER 2
-[R] 10F0 [ENTER] BFFF [ENTER] GAMEOVER2 [ENTER]
"Last adress: 13B5"
-[M] 1236 [ENTER] B6 03 [ESC]
-[M] 13B6 [ENTER] AF 32 FE 95 32 24 97 C3 84 7B [ESC]
-[M] 10F2 [ENTER] C2 [ESC]
-[S] 10F0 [ENTER] 13BF [ENTER] GAME OVER 2 [ENTER]

GAME OVER 1
-[R] 10F0 [ENTER] BFFF [ENTER] GAME OVER l [ENTER]
"Last address: 13B7"
-[M] 1236 [ENTER] B8 03 [ESC]
-[M] 13B8 [ENTER] AF 32 AD 98 32 A6 99 C3 84 7B [ESC]
-[M] 10F2 [ENTER] C2 [ESC]
-[S] 10F0 [ENTER] 13C1 [ENTER] GAME OVER 1 [ENTER]

EXOLON1
-[R] 10F0 [ENTER] BFFF [ENTER] EXOLON1 [ENTER]
"Last address: 12A5"
-[M] 11ED [ENTER] A8 02 [ESC]
-[M] 12A8 [ENTER] F5 AF 32 1D 9D FI C3 60 6D [ESC]
-[M] 10F2 [ENTER] B1 [ESC]
-[S] 10F0 [ENTER] 12B0 [ENTER] EXOLON1 [ENTER]

RAMBO
-[R] 10F0 [ENTER] BFFF [ENTER] RAMBO [ENTER]
"Last address: 157F"
-[M] 1217 [ENTER] 80 05 [ESC]
-[M] 1580 [ENTER] AF 32 09 6B 32 0A 6B C3 00 67 [ESC]
-[M] 10F2 [ENTER] 8A [ESC]
-[S] 10F0 [ENTER] 1589 [ENTER] RAMBO [ENTER]

BOBBY BEARING
-[R] 10F0 [ENTER] BFFF [ENTER] BOBBY [ENTER]
"Last address: 14EF"
-[M] 114E [ENTER] F0 04 [ESC]
-[M] 14F0 [ENTER] F5 3E AF 32 F8 73 FI C3 7E 01 [ESC]
-[M] 10F2 [ENTER] FA [ESC]
-[S] 10F0 [ENTER] 14F9 [ENTER1 BOBBY [ENTER]

ACADEMY
-[R] 10F0 [ENTER] BFFF [ENTER] ACADEMY [ENTER]
"Last address: 19EF"
-[M] 1399 [ENTER] F0 09 [ESC]
-[M] 19F0 [ENTER] F5 AF 32 AC AE 32 D0 AE 32 FA AE 32 1F AF 3D 32 92 7A 32 99 7A 32 11 7A 32 49 7A FI C3 00 5B [ESC]
-[M] 10F2 [ENTER] 0F 09 [ESC]
-[S] 10F0 [ENTER] 1A0E [ENTER] ACADEMY [ENTER]

BOULDER DASH 2
-[R] 10F0 [ENTER] BFFF [ENTER] BOULDER [ENTER]
"Last address: 1BAD"
-[M] 1254 [ENTER] B0 0B [ESC]
-[M] 1BB0 [ENTER] F5 AF 32 F8 7A 32 F9 7A 32 FA 7A FI C3 A6 7E [ESC]
-[M] 10F2 [ENTER] BF [ESC]
-[S] 10F0 [ENTER] 1BBE [ENTER] BOULDER [ENTER]

ROCKY HORROR SHOW
[R] 10F0 [ENTER] BFFF [ENTER] RHS [ENTER]
"Last address: 1300"
-[M] 11 EB [ENTER] 08 03 [ESC]
-[M] 1308 [ENTER] F5 AF 32 AC BE 32 A0 BE 32 B5 BE FI C3 00 5B [ESC]
-[M] 10F2 [ENTER] 17 [ESC]
-[S] 10F0 [ENTER] 1316 [ENTER] RHS [ENTER]

HEARTLAND
-[R] 10F0 [ENTER] BFFF [ENTER] HEARTLAND [ENTER]
"Last address: 1289
-[M] 11 Dl [ENTER] 90 02 [ESC]
-[M] 1290 [ENTER] F5 3E C9 32 E9 A7 FI C3 A8 61 [ESC]
-[M] 10F2 [ENTER] 9A [ESC]
-[S] 10F0 [ENTER] 1299 [ENTER] HEARTLAND [ENTER]

OLLIE&LISSA
-[R] 10F0 [ENTER] BFFF [ENTER] OLLIE&LISSA [ENTER]
"Last address: 16EA"
-[M] 1258 [ENTER] F0 06 [ESC]
-[M] 16F0 [ENTER] F5 AF 32 09 87 32 0A 87 32 0B 87 32 27 86 32 28 86 32 29 86 3E 18 32 F3 92 FI C3 3C F9 [ESC]
-[M] 10F2 [ENTER] 0D 06 [ESC]
-[S] 10F0 [ENTER] 170C [ENTER] OLLIE&LISSA [ENTER]

Játék Basic módra

Az alábbi listához nem lehet sokat hozzáfűzni. Aki ismeri a Basic nyelvet, az könnyen megérti a program működését. A játékban egy autóverseny izgalmaiba élhetjük bele magunkat. Minden pályán 5 kört kell megtennünk meghatározott idő alatt. Ha az összes pályát teljesítettük, a gép gratulál és kiírja az elért pontszámunkat. Akár versenyeket is lehet rendezni, hogy ki tud nagyobb pontszámot elérni.
A program érdekessége, hogy az autó hangját sztereóban hallgathatjuk. A hang onnan jön, ahol éppen tartunk a pályán. A programban láthatunk két példát arra, hogyan tehetünk gyorsabbá egy programot. Az egyik megoldás az „OUT 191,12", amely felgyorsítja a memóriakezelést. A mások a „POKE 56,201". Ez utóbbi a megszakításokat tiltja le. Emiatt nem használhatjuk az INKEY$ függvényt, és a SOUND utasításnak sem lesz hatása.

```
100 PROGRAM „StockCar.bas"
110 SET 26,1:SET 7,1
120 CLEAR FONT
150 SET 22,1:SET 23,2:SET 24,40:SET 25,20:SET 8,1
160 OPEN #1:"video:"
170 OPEN #2:"video:"
180 SET #1:PALETTE 0,25,182,246,190,254,183,247
190 SET #2:PALETTE 0,25,182,246,190,254,183,247
200 SET #1:LINE MODE 3
210 SET #2:LINE MODE 3
220 SET 22,0:SET 23,0:SET 25,4
230 OPEN #3:"video:"
240 SET #3:PALETTE 0,219,0,0
250 LET CON=0
260 CALL CHARS
270 TEXT 40
290 CALL USR(56,0)
300 SET #102:PALETTE 0,0,0,0
310 PRINT #102,AT 2,10:"ENTERPRESS PROGRAMS"
315 SET #102:INK 3
320 PRINT #102,AT 4,13:"STOCK CAR RACE"
325 SET #102:INK 1
330 PRINT #102,AT 8,1:"SELECT CONTROL:"
340 PRINT #102,AT 11,5:"1 - INTERNAL"
350 PRINT #102,AT 14,5:"2 - EXTERNAL 1"
360 PRINT #102,AT 17,5:"3 - EXTERNAL 2"
370 PRINT #102,AT 22,10:"PRESS FIRE TO START"
380 CALL LIGHT
390 LET I$=INKEY$
400 SELECT CASE I$
410 CASE „1"
420 IF CON<>0 THEN
430 LET CON=0
440 SOUND PITCH 37,DURATION 5,SOURCE 2
450 END IF
460 CASE „2"
470 IF CON<>1 THEN
480 LET CON=1
490 SOUND PITCH 49,DURATION 5,SOURCE 2
500 END IF
510 CASE „3"
520 IF CON<>3 THEN
530 LET CON=2
540 SOUND PITCH 61,DURATION 5,SOURCE 2
550 END IF
560 CASE ELSE
570 END SELECT
580 IF JOY(CON)<>16 THEN 390
590 !POKE 56,201
600 OUT 166,9:OUT 168,0:OUT 172,0:OUT 167,0:OUT 181,7
610 LET LEV,SCORE=0
620 RESTORE 2220
630 LET TIM,CIRCL=0:LET AS=1:LET LIM=40:LET
    CX,CXB=640:LET CY,CYB=150:LET ASB=2:LET CH,
    CHB,CHB2=97:LET XP,YP=0
640 PRINT #102,AT 22,10:"   PLEASE WAIT!   „
650 CALL SCREEN
660 SELECT CASE JOY(CON)
670 CASE 1
680 LET CH=CH-1-8*(CH=97)
690 CASE 2
700 LET CH=CH+1+8*(CH=104)
710 CASE 4
720 LET XP=XP+4*(XP>3):LET YP=YP+4*(YP>3)
730 CASE 8
740 LET XP=XP-4*(XP<20):LET YP=YP-4*(YP<24)
750 CASE ELSE
760 END SELECT
770 PLOT #ASB:CXB,CYB,
780 PRINT #ASB:CHR$(CHB2);
790 LET CHB2=CHB:LET CXB=CX:LET CYB=CY:LET CHB=CH
800 SELECT CASE CH
810 CASE 97
820 LET CXN=CX+XP:LET CXC=CXN+60:
    LET CYN=CY:LET CYC=CYN
830 CASE 98
840 LET CXN=CX+XP:LET CYN=CY+YP:LET
    CXC=CXN+60:LET CYC=CYC+4
850 CASE 99
860  LET CYN=CY+YP:LET CXN=CX:LET
    CXC=CXN:LET CYC=CYN+4
870 CASE 100
880  LET CXN=CX-XP:LET CYN=CY+YP:LET
     CXC=CXN-4:LET CYC=CYN+4
890 CASE 101
900  LET CXN=CX-XP:LET CYN=CY:LET
     CXC=CXN-4:LET CYC=CYN
910 CASE 102
920  LET CYN=CY-YP:LET CXN=CX-XP:LET
     CXC=CXN-4:LET CYC=CYN-68
930 CASE 103
940  LET CYN=CY-YP:LET CXN=CX:LET
     CXC=CXN:LET CYC=CYN-68
950 CASE 104
960  LET CXN=CX+XP:LET CYN=CY-YP:LET
     CXC=CXN+60:LET CYC=CYN-68
970 CASE ELSE
980 END SELECT
990 PLOT #ASB:CXC,CYC,
1000 GET #ASB:CHEC$
1010 IF ORD(CHEC$)=0 THEN
1020 LET CX=CXN:LET CY=CYN
1030 PLOT #ASB:CX,CY,
1040 PRINT #ASB:CHR$(CH);
1050 ELSE
1060 LET XP,YP=0
1070 PLOT #ASB:CX,CY,
1080 PRINT #ASB:CHR$(CH);
1090 CALL BUMM
1100 END IF
1110 LET CAR=ASB:LET ASB=AS:LET AS=CAR
1120 DISPLAY #AS:AT 1 FROM 1 TO 20
1130 LET TIM=TIM+1
1140 PRINT #3,AT 4,6:TIM;
1150 LET CIRCL=CIRCL-(CX>640-XP/2 AND
     CX<640+XP/2 AND CY<400 AND(CH=97
     OR CH=98 OR CH=104) AND XP<>0)
1160 PRINT #3,AT 4,37:CIRCL;
1170 OUT 160,76-XP*2:OUT 171,63-CX/21
1180 OUT 175,CX/21:OUT 181,7
```

```
1190 LET STP=IN(181)
1200 IF (STP BAND 1)=0 THEN 270
1210 IF CIRCL<5 THEN 660
1220 IF TIM>LIM THEN
1230 POKE 56,245
1240 CALL USR(56,0)
1250 TEXT
1260 SET #102:PALETTE 0,0,0,0
1270 PRINT #102,AT 5,17:"SORRY!"
1280 PRINT #102,AT 9,13:"BUT YOU LOSE!"
1290 PRINT #102,AT 17,13:"YOUR SCORE:";SCORE
1300 PRINT #102,AT 20,13:"PRESS ANY KEY"
1310 CALL LIGHT
1320 SPOKE 255,16042,0
1330 IF INKEY$="" THEN 1330
1340 GOTO 270
1350 ELSE
1360 OUT 171,0:OUT 175,0
1370 LET LEV=LEV+1:LET SCORE=SCORE+(LIM-TIM)*10
1380 IF LEV<3 THEN 630
1390 END IF
1400 POKE 56,245
1410 CALL USR(56,0)
1420 TEXT
1430 SET #102:PALETTE 0,0,0,0
1440 PRINT #102,AT 5,12:"CONGRATULATIOS!"
1450 PRINT #102,AT 10,12:"YOU ARE THE BEST"
1460 PRINT #102,AT 12,12:"STOCKCAR RACER!"
1470 PRINT #102,AT 15,12:"YOUR SCORE:";SCORE
1480 PRINT #102,AT 20,13:"PRESS ANY KEY"
1490 CALL LIGHT
1500 SPOKE 255,16042,0
1510 IF INKEY$="" THEN 1510
1520 GOTO 270
1530 DEF BUMM
1540 FOR Z=0 TO 6
1550 OUT 160,80-10*Z:OUT 171,60-10*Z:OUT 175,10*Z
1560 CALL PAUSE(5)
1570 NEXT
1580 FOR Z=5 TO 0 STEP-1
1590 OUT 160,80-10*Z:OUT 171,60-10*Z:OUT 175,10*Z
1600 CALL PAUSE(5)
1610 NEXT
1620 END DEF
1630 DEF CHARS
1640 SET CHARACTER 97,0,36,126,251,189,251,126,36,0
1650 SET CHARACTER 98,0,12,62,115,251,222,124,48,0
1660 SET CHARACTER 99,56,108,214,124,124,254,108,56,0
1670 SET CHARACTER 100,0,48,124,206,223,123,30,12,0
1680 SET CHARACTER 101,0,36,126,223,189,223,126,36,0
1690 SET CHARACTER 102,0,12,62,123,223,206,120,48,0
1700 SET CHARACTER 103,28,54,127,62,62,107,54,28,0
1710 SET CHARACTER 104,0,48,124,222,251,115,62,12,0
1720 END DEF
1730 DEF LIGHT
1740 RESTORE 1810
1750 DO
1760 READ C
1770 SET #102:PALETTE 0,C,0,54
1780 CALL PAUSE(0)
1790 LOOP UNTIL C=201
1800 END DEF
1810 DATA 128,16,144,2,130,18,146,210,154,218,
     147,211,155,219,91,203,75,217,89,201
1820 DEF PAUSE(T)
1830 FOR I=0 TO T
1840 NEXT
1850 END DEF
1860 DEF SCREEN
1870 LET SZIN=2
1880 CLEAR #1
1890 CLEAR #2
1900 CLEAR #3
1910 DO
1920 SET #1:INK SZIN
1930 SET #2:INK SZIN
1940 READ X,Y,P,M
1950 SELECT CASE M
1960 CASE 0
1970 PLOT #1:X,Y,:PLOT #2:X,Y,
1980 FOR I=1 TO P
1990 PRINT #1:CHR$(159);:PRINT #2:CHR$(159);
2000 NEXT
2010 CASE 1
2020 FOR I=Y TO Y+P*72 STEP 72
2030 PLOT #1:X,I,:PLOT #2:X,I,
2040 PRINT #1:CHR$(159);:PRINT #2:CHR$(159)
2050 NEXT
2060 CASE ELSE
2070 LET LIM=M
2080 END SELECT
2090 LET SZIN=SZIN+1+6*(SZIN=7)
2100 LOOP UNTIL P=0
2110 SET #1:INK 1
2120 SET #2:INK 1
2130 PLOT #AS:CX,CY,:PLOT #ASB:CXB,CYB
2140 PRINT #AS:"a";:PRINT #ASB:"a";
2150 PRINT #3,AT 2,6:"LEVEL";LEV:PRINT
      #3,AT 2,26:"LIMIT:";LIM;"S";
2160 PRINT #3,AT 4,1:"TIME:";TIM;
2170 PRINT #3,AT 4,15:"SCORE:";SCORE;
2180 PRINT #3,AT 4,30:"CIRCLE:";CIRCL;
2190 DISPLAY #3:AT 21 FROM 1 TO 4
2200 DISPLAY #AS:AT 1 FROM 1 TO 20
2210 END DEF
2220 DATA 0,68,20,0,0,716,20,0,0,140,7,1,1216,
     140,7,1,256,352,12,0,256,424,12,0,0,0,0,600
2230 DATA 192,68,14,0,0,716,7,0,704,716,9,0,
     192,428,14,0,64,212,2,0,128,68,1,1,0,212,
     6,1,368,644,6,0,1088,68,2,1,1216,212,6,1,320,
     356,8,0,1152,212,1,0,0,0,0,700
2240 DATA 0,68,20,0,0,716,20,0,64,392,7,0,768,
     392,7,0,0,140,7,1,1216,140,7,1,160,554,15,0,
     160,230,15,0,608,284,3,1,0,0,0,1200
```

Tippek – trükkök (folytatás)

Hol a hiba?

Titokzatos hiba bujkál valahol az EXDOS - IS-DOS csatlakozásánál. Néha, ha RAM-diszket használva akarjuk az IS- DOS-t elindítani, az EXDOS
File not found
hibaüzenettel utasítja el a próbálkozást. Ezután, ha megszüntetjük a RAM-diszket, és újra megpróbáljuk, a rendszer No RAM disk (Retry, Abort)
üzenetet ad, akkor is, ha gondosan az A: lemezegységre váltottunk előtte. A hiba megelőzésére jobb a RAM-diszket csak az IS-DOS alatt létrehozni.

Hibás a DISKCOPY!

Az IS-DOS rendszer egyik fontos segédprogramja a lemezről-lemezre másoló DISKCOPY. Sajnos, ez a program sem mentes a hibától.
A felhasználók többségének csak egy lemezegysége van. Semmi baj, a forrás- és a céllemez cserélgetésével megoldható a másolás. Gyors előzetes fejszámolás: a gép memóriája 128 K, ebből persze lejön valamennyi az EXOS, EXDOS, IS-DOS számára, marad mondjuk közel 100 K. A 180 K-s lemezt 2 részletben, a 360 K-st 4 részletben, a 720 K-st 8 részletben lehet másolni... elviselhető.
A valóság sokkal kiábrándítóbb. A program valamiért csak a belapozott RAM szegmenseket használja, így a lemezmásolás 7, 14 illetve 28 részletben történik (ez 14, 28 illetve 56 lemezcserét jelent). Akkor is, ha történetesen 256 K-ra vagy 1 megára bővített RAM van a gépemben. Kétségbeejtő.
Még egy hasznos tanács: 4 blokknyi RAM-diszk mellett a program nem működik, és ezt nem is jelzi, csak csendesen beragad. Ügyeljünk tehát a használatakor.

APRÓHIRDETÉSEK

ENTERPRISE-osok figyelem! Különböző EPROM-ba égetett programok
- ASMON, GEN, MON, SPEEDTEST, SPEEDLOADER stb.
valamint hardver bővítések : memória 256 KB-tól 2 MB-ig,
óra kártya, EPROM cartridge stb. kaphatók.
Válaszborítékért részletes tájékoztató!
Györfi Attila, 1076 Budapest, Dózsa Gy. út 6.

ENTERPRISE belső memóriabővítés 320 KB-ra, hibátlan RAM frissítéssel.
Átkapcsolás 128 KB/320 KB üzemmód között
Bozai Gábor, 8000 Székesfehérvár, Budai út 92. fsz. 5. Üzenet: (22) 20-857

Alig használt Spectrum emulátor programokkal együtt eladó. Irányár: 5000 Ft
Titi László, Tel.: munkaidőben 121-35-26/80

ROCKSOCK négycsatornás sztereó zeneszerkesztő program magyar nyelvű leírással utánvéttel megrendelhető lemezen vagy kazettán.
Ára: 150 Ft + a hordozó ára.
Devilsoft, 1399 Budapest, Pf. 701/334

TALLGRAAS (eredeti amerikai)
40 MB -os STREAMER
1 éves garanciával eladó.
Külön vezérlő nem szükséges!
Ára 43.000 Ft + ÁFA
Érdeklődni munkanapokon 15 - 18 óra között a 1128 - 356 -os Bp. -i telefonszámon

---

ENTERPRISE számítógép floppyval, lemezekkel, programokkal ELADÓ.
Bp. X. kér. Juhar u. 19.

Számítástechnikát oktató intézmények figyelmébe ajánljuk:
IBM PC-n használható tárrezidens DOS OKTATÓ SEGÉDPROGRAMUNKAT,
Iskolák számára 50%-os kedvezménnyel,
ára : 3000 Ft + ÁFA
Érdeklődés, megrendelés:
MÁTRIX Kft
8000 Székesfehérvár,
Dózsa Gy. tér 10.
Tel.: (22) 16-520 /141, 145
Fax: (22) 11- 585

hirdetések, felhívások

Az ENTERPRESS előző száma korlátozott számban még megvásárolható a kiadónál.
Előfizetési megrendeléseiket a POSTA HELIR BUDAPEST 1900 címére adhatják fel.

ENTERPRISE klubok,
ENTERPRISE köré szerveződött baráti társaságok,
ENTERPRISE témákban levelezni kívánók címeit várjuk!
A címeket lapunkban rendszeresen közöljük.

ENTERPRESS

Hirdetésfelvétel

Az apróhirdetések ára: 1 Ft karakterenként. A szöveget és a befizetést igazoló nyugtát (rózsaszín postautalványon) az alábbi címre kérjük elküldeni:

MÁTRIX Kft.
ENTERPRESS
8000 Székesfehérvár
Dózsa György tér 10.
Bankszámlaszám: OTP 679-022096-9

Tisztelt leendő Szerzőtársak!
Kérjük Önöket, hogy anyagaik elkészítése, beküldése előtt feltétlenül kérjenek ingyenes tájékoztatót a feltételekről, a szerkesztőség által felállított tartalmi és formai elvárásokról.
Az érdeklődők leveleit levélcímünkre várjuk:
ENTERPRESS
1399 Budapest
Pf. 701/334