SRPEN

FUN with Commodore

1992

časopis uživatelů Commodore 64/128 1. ČÍSLO

BASIC

ŽIVOTOPIS C 64

GEOS

TIPY A TRIKY

ASSEMBLER

COMMOTRONIC

Vydává Commotronic klub Šumperk

Obsah 1. čísla časopisu FUN:

Vážení uživatelé Commodore 64!

Konečně se Vám dostal do rukou netrpělivě očekávaný klubový časopis FUN. Proč se jmenuje zrovna FUN? Původně to byl jen pracovní název, "Fandovské univerzální noviny" a nakonec jsme si řekli, že práce s počítačem C64 je vlastně ohromná zábava a legrace, prostě fun. A zatím u toho zůstalo, protože jsme nevymysleli nic originálního.

Je nám jasné, že kritika hned na první číslo bude neúprosná, laťka totiž byla časopisem Počítač aktívne pro nás postavena velmi vysoko a dlouho nebudeme mít na to, abychom se jí přiblížili. I to byl jeden z důvodů, proč je FUN vydáván jen jako interní, klubová tiskovina. Ale dát dohromady schopnou redakční partu, jak se to podařilo Romanu Kučerovi, je obrovský výkon a taky štěstí. To se ale musí nejdříve dlouho pokoušet, než se dá přemluvit.

Desítky dopisů, které od Vás denně dostáváme, neobsahují jen Vaše přání a objednávky, ale v mnoha případech i velice konkrétní dotazy, které by bylo záhodno zodpovídat, ale zatím to jaksi není možné. Museli bychom k tomu účelu vyčlenit jednoho spolupracovníka, který by se o nic jiného nestaral.

Abychom alespoň trošku napravili tento nedostatek, snažili jsme se první číslo FUNu přiblížit škále dotazů v dopisech a našim znalostem toho, co Vás zajímá. Ale není na světě člověk ten, aby se zavděčil lidem všem. To se týká i nás i FUNu. Ale my máme jednu velkou výhodu. Jsme s Vámi neustále ve styku. Vaše připomínky, které jak očekáváme budou hojné a umožní nám korigovat obsah dalšího čísla k Vaší větší spokojenosti. Pokud bude o FUN zájem nebude pak problém rozšířit počet výtisků a tím pak i počet stran při stejné ceně, pokud neporostou ceny papíru rychleji než počet čtenářů.

FUN by měl obsahovat i rubriku čtenářu a inzerci. Předpokládáme, že se uvedené rubriky objeví ve druhém čísle, které se vám dostane do rukou na přelomu září a října.

Dík neustále se hromadícím problémum, které zvládáme pomaleji, než bychom si přáli, musíme Vám oznámit také jednu méně příjemnou zprávu. Kniha TOOL 64, ohlášená na srpen, bude spíše vánočním dárkem, než společnikem na konec prázdnin a začátek školního roku.

Jak jste zaregistrovali v našem ceníku, nebyla v něm cena TOOL 64 uvedena, protože nebyla známá. Dnes Vám můžeme oznámit že bude mít cca 300 stran formátu A5 a její cena bude do 120 korun. Objednávky TOOL 64, které jste nám zaslali, jsou řádně registrované a knihu Vám ihned po vujití dodáme.

Pochopitelně se snažíme získat další schopné spolupracovníky, kteří by nám pomohli dříve plnit Vaše přání, ale jde to pomalu, mnohem pomaleji, než byste Vy i my chtěli.

Přesto se domnníváme, že jsme od loňského léta, tedy za necelý rok vážné činnosti ve prospěch Commodoristů udělali slušný krok k Vaší spokojenosti.

Rozšířili jsme sortiment na více než dvojnásobek položek, při výrobě používáme většinou profesionální plošné spoje s nepájivou maskou, které zajišťují spolehlivost vyráběných doplňku. Tím klesl na minimum počet reklamací. Zahájili jsme dovoz značkových doplňků z NSR, budování databanky uživatelů počítačů C64 / 128 a Amiga, která již brzo bude obsahovat údaje o 10 000 uživatelích. Vydali jsem pro Vás první souhrnný katalog, připravili vydávání časopisu FUN, rozšířili kolektiv těch, kteří se ze všech svých sil snaží co nejlépe Vám vyhovět. Přestěhovali jsem se do nového působiště, kde jsou konečně pod jednou střechou všechny činnosti firmy, a kde můžeme důstojně přijímat Vaše návštěvy. V těchto dnech otevíráme v našem novém sídle i prodejnu. Zdá se, že i naše problémy s tiskem budou již definitivně vyřešeny a m.j. se dočkáte během měsíce i příručky Grafika na C64. Před Vánocemi Vám nabídneme novinky v programovém vybavení a to i na kazetách. Počítáme se zkvalitněním a rozšířením vydávaných příruček. A protože Basic V2.0 a Simons· Basic jsou již opravdu starštího data, připravujeme pro Vás systémové moduly s několika rozšířeními Basicu, jako je např. Basic V3.5, který Vám umožní spolupracovat s počítači C128, C16 a P4. Ale hlavně Vám dáme k dispozici monitor strojového kódu a jemnou grafiku, které uživatelé C64 nejvíce postrádají. O všech novinkách Vás budeme prostřednictvím FUNu i inzerce včas informovat.

Ale dost již slov na úvod. Podívejte se, co jsme pro Vás připravili a sdělte nám, jak jste byli spokojeni.

PROGRAMOVÁNÍ V BASICU

1. Teorie

1.1 Natahování programu z programu

Ačkoliv se v zásadě jedná o celkem jednoduchou a zdánlivě jasnou záležitost, není natahování programu z programu tak jednoduché. Dále si proto vysvětlíme, jaké překážky a proč nás při řešení tohoto programu mohou potkat.

Když píšete program, který pracuje s množstvím dat, často zjišťujete, že Basic-paměť počítače C64, kterou máte k dispozici, je docela zaplněná. V takovém případě se nabízí možnost rozdělit program na více částí a ty pak podle potřeby natahovat do počítače. Stává se také, že v Basic-programu potřebujete strojové rutiny které se musí uložit do paměti počítače před natažením hlavního programu do paměti. Přitom také mohou vzniknout problémy. V zásadě je nutno rozlišovat mezi natahováním programu v Basicu a strojovém kódu.

První případ rozšiřuje téměř neomezeně možnosti Basicu V2.0, neboť rozdělení programu na moduly, natahované do počítače postupně, umožňuje tvořit programy s téměř neomezenou délkou. Programové moduly zabírají málo místa v paměti, takže je více prostoru pro data i grafiku. Programy rozdělené na do sebe uzavřené moduly se navíc snáze opravují a mění, což také není zanedbatelné. Jednotlivé programové moduly se pak z programu jednoduše volají programovým řádkem

LOAD "název programu",8.

Nicméně se může stát, že dotažený programový modul bude zkomolený, to znamená, že ze začátku bude vše OK, ale od jistého programového řádku budou v programu kromě textu i různé grafické znaky, příkazy a jiné věci, které nás sice udivují, ale nejsou nám k ničemu. Jak se to může stát? K vysvětlení si vezmeme na pomoc schema rozdělení paměti počítače C64.

Paměť pro Basic u C64 obvykle začíná od adresy $0801, nebo dekadicky 2049. Končí na adrese $A000, nebo 40960. Tento rozsah si mezi sebe musí rozdělit vlastní program, proměnné, pole a řetězce. K tomuto rozdělení potřebuje operační systém počítače informace, kde končí program, kde p[roměné, kde pole atd. O tyto informace - adresy se starají tzv. vektory (ukazatele) v nulté stránce (zero page). Přesněji jsou to vektory, uložené na adresách od $2B až $38 (dekadicky 43 - 56). Pro dotahování programů jsou důležité hlavně začátek Basic-paměti, protože od něj se začínají ukládat všechny Basic-programy a začátek proměnných, který je shodný s koncem Basic-programu. Tyto vektory jsou uloženy na adresách 43, 44 a 45, 46. K tomu jednoduchý příklad:

Dejme tomu, že jste napsali program, který slouží ke zpracování adres, řízený z menu a sestávající z programových modulů, které se podle potřeby dotahují do počítače z diskety. Menu je přitom vlastně samostatný starovací program, který dotahuje do paměti všechny ostatní moduly programu. Dohromady takové menu nemá víc než 1 kB, vystačí proto s paměťovým prostorem $0801 - $0C00. Ostatní části programu, které načítají, třídí, vyhledávají, editují adresy jsou pochopitelně delší, najdelší z nich zabere cca 4 kB.

Když spustíte menu a zvolíte některou funkci (zadání nové adresy), natáhne menu tento programový díl z diskety. Ale proměnné Basicu začínají na konci programu, t.j. od adresy $0C00 a dotahovaný program bude se svou délkou 4 kB potřebovat paměť až po $1800. Aby nedošlo k vymazání proměnných v paměti, přeruší počítač natahování programu po naplnění paměti po adresu $0C00, t.j. po dosažení vektoru začátku proměnných. Samozřejmě k tomu dojde uprostřed některého řádku programu, což vede k tomu, že se nám nahednou v listingu objeví nesmysly, které jsou vlastně proměnné a další data v paměti.

Pokud známe příčinu nějakého problému, dokážeme jej řešit. Zde máme dvě možnosti:

První je zajistit, aby programové moduly byly vždy kratší než úvodní startovní menu. To je ale těžko realizovatelné. Druhou možností je upravit menu tak, aby bylo stejně dlouhé jako nejdelší programový modul. Pak dotahovaný program nemůže dosáhnout do proměnných a je vše OK. Pokud má nejdelší programový modul 10 kB, prodloužíme menu taky na 10 kB.

Prodloužení se provede následovně:

1. Do paměti se natáhne úvodní menu pomocí příkazu LOAD "menu",8.

2. Pomocí příkazu PRINT PEEK(45),PEEK(46) se vypíší oba bajty adresy počátku proměnných Tyto hodnoty si poznamenáme jako hodnoty 1a a 1b.

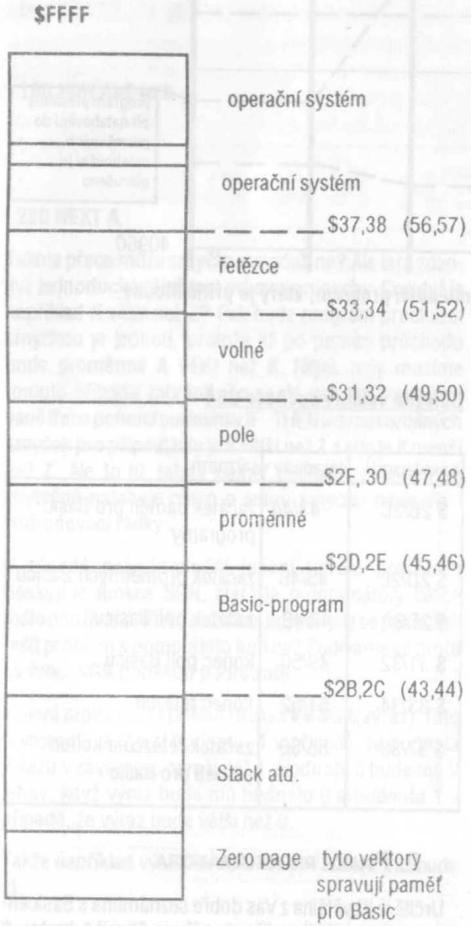
3. Do počítače se natáhne nejdelší programový modul příkazem LOAD"modul",8.

4. Pomocí příkazu PRINT PEEK(45),PEEK(46) se vypíší oba bajty adresy počátku proměnných. Tyto hodnoty si poznamenáme jako hodnoty 2a a 2b.

5. Nakone se znova natáhne do počítače startovní menu příkazem LOAD"menu",8 a změní se vektory počátku proměnných zadáním příkazového řádku v přímém módu: **POKE 45,2a: POKE 46,2b <RETURN>**

Hodnoty 1a a 1b si uschovejte, budete je potřebovat pokud budete chtít program ještě jednou měnit.

Rozdělení paměti C64



Nyní zbývá již jen uložit program znovu na disketu a přesvědčit se, že má po uložení na disk stejnou délku, jako nejdelší programový modul. Uvědomte si však, že při opravách či doplňování modulů se může změnit jejich délka. Potom je nezbytně nutné opravit délku menu tak, aby opět odpovídala nejdelšímu programovému modulu.

Strojové programy

Druhý případ dotahování programů nastává při natahování strojových programů, které se ukládají na určitou adresu uvnitř paměti počítače. Takové programy se natahují pomocí příkazu LOAD"název strojové rutiny",8,1. Jednička zcela na konci příkazu označuje tzv. absolutní natahování, při kterém se program uloží na přesně stejnou adresu, ze které se ukládal na vnější paměťové médium příkazem SAVE.

Nicméně má příkaz LOAD malý zádrhel. Po jeho provedení počítač provede ještě automaticky příkaz RUN, což nám způsobí poněkud nepříjemné komplikace. Když budeme například potřebovat natáhnout několik rutin pomocí přibližně takovéhoto postupu:

```
10 LOAD"rutina 1",8,1
20 LOAD"rutina 2",8,1
```

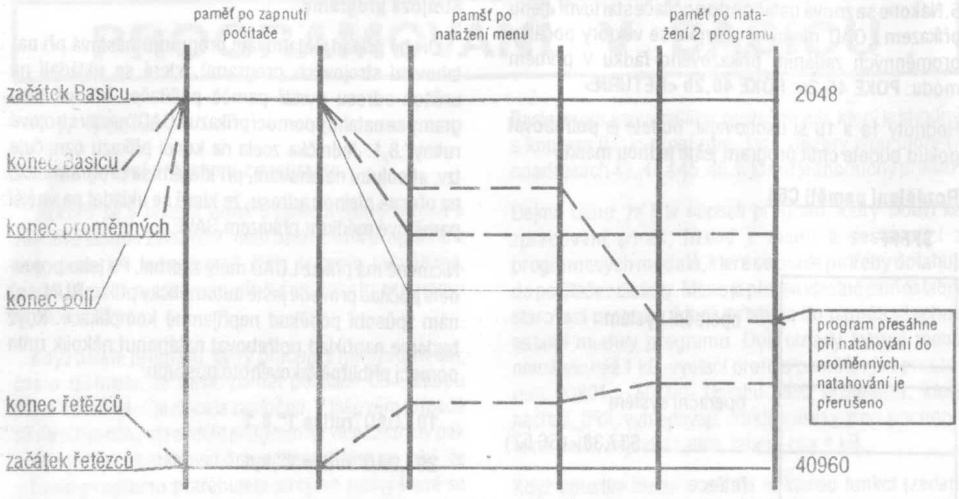
zaměstná počítač disketovou jednotku, ale jinak se nic nestane. Je to způsobeno tím, že při provádění programu se v řádku 10 natáhne rutina 1 a poté se automaticky provede příkaz RUN, který vede k průběhu programu opět od začátku, t.j. znovu se natahuje rutina 1. To se opakuje stále dokola, dokud ten nesmysl nepřerušíme. Musíme tedy program upravit tak, aby počítač po novém automatickém startu již věděl, že je první rutina již natažena a že má natahovat až druhou. Ale jak?

Není to tak složité. Naštěstí odpovídá automaticky prováděný příkaz RUN příkazu RUN provedenému z klávesnice jen částečně, takže se při něm nemažou proměné. To nám dává možnost stanovit při natahování, co se již do paměti natáhlo a co ještě nikoliv.

Upravené programové řádky pak vypadají takto:

```
10 IF A=0 THEN A=1: LOAD:rutina 1",8,1
20 IF A=1 THEN A=2: LOAD"rutina 2",8,1
```

Když se nyní program odstartuje, obsahuje proměnná A hodnotu 0. Tím je zajištěno, že se na řádku 10 provede příkaz za podmínkou IF, to je změna proměnné A na 1 a příkaz k natažení rutiny 1. Po opětovném automatickém spuštění programu příkazem RUN od začátku nebude provedena posloupnost příkazů v řádku 10, protože není splněna podmínka A=0. První řádek se proto přeskočí a provede se řádek 20. Protože jsme v prvním průchodu řádkem 10 nastavili proměnnou A na 1, provedou se příkazy uvedené za podmínkou IF v řádku 20, takže se proměnná A změní na 2 a natáhne se rutina 2. Opětovné automatické spuštění programu povede k vynechání příkazů za podmínkami IF v řádcích 10 a 20, protože proměnná A má hodnotu 2. Program pak bude pokračovat dalším řádkem Vašeho programu.



Toto se děje v paměti počítače, když natahujete z programu další program, který je příliš dlouhý.

Ach, ty vektory!

Vektory, které jsou často také označovány jako ukazatele, jsou bajtové páry, obsahující důležité adresy z paměti počítače. Adresy se počítají podle následujícího vzorce:

Adresa = PEEK (vektor A) + 256 * PEEK(vektor B)

Přitom odpovídá vektor A nižšímu batu adresy (LO-bajt) a vektor B vyššímu bajtu (HI-bajt) adresy. Počítač používá vektory k označování důležitých, ale proměnných adres, jako je například začátek polí. Vektory, které se používají velice často, jsou uloženy v ZERO PAGE. ZERO PAGE (nulta stránka paměti) představuje prvních 256 bajtů RAM a je velmi důležitá pro procesor 6502/6510, neboť strojové příkazy pracují s adresami v nulté stránce rychleji, což přináší významnou časovou úsporu. Ta je důležitá již proto, že interpreter Basicu V 2.0 počítače C64 nepatří mezi nejrychlejší a každá časová úspora je při jeho běhu důležitá. Tyto vektory v RAM mají tu výhodu, že je můžeme snadno měnit. Například při práci s Basic-programy můžeme pomocí výše uvedených vektorů na adresách $2B - $38 měnit rozdělení paměti pro Basic dle vlastní potřeby, posouvat začátek paměti pro program i pro proměnné. Při posouvání začátku paměti pro Basic je však nutno vynechat první 2 kB paměti, které využívá operační systém počítače a kam si procesor ukládá data důležitá pro jeho práci. Dále pak nemá význam pokoušet se využívat rozsah paměti nad adresou 40960, kde jsou umístěny ROM - pevné paměti, obsahující operační systém KERNAL, interpret Basicu a vstupy/výstupy.

Důležité vektory počítače C64

hexadecimál.	dekadicky	význam
$ 2B/2C	43/44	začátek paměti pro Basic programy
$ 2D/2E	45/46	začátek proměnných Basicu
$ 2F/30	47/48	začátek polí Basicu
$ 31/32	49/50	konec polí Basicu
$ 33/34	51/52	konec řetězců
$ 37/38	55/56	začátek řetězců a konec paměti pro Basic

2. Praxe programátora

Určitě je již většina z Vás dobře seznámena s Basicem V2.0 a má za sebou první programátorské kroky. A pokud jste nováčci, učíte se rychle a za pár měsíců z Vás budou ostřílení programátoři. Každý programátor má poněkud odlišný přístup k řešení problémů a taky soubor příkazů, který používá bývá charakteristický. Nicméně všichni se setkávají s řadou společných problémů, které řeší svým osobitým způsobem. Ale proč vymýšlet vymyšlené, je lepší věnovat vlastní energii na řešení vlastních speciálních problémů a ty obecné programové postupy převzít od zkušenějších. K tomu účelu Vám nabízíme následující rady.

2.1 FOR - TO - NEXT zase jinak

Představte si, že potřebujete zabudovat do programu smyčku FOR - TO - NEXT, ale tak, aby počáteční a koncová hodnota byly voleny nebo počítány v programu. Jistě před sebou většina z Vás vidí jednoduché řešení asi takového typu:

```
10 INPUT"zadejte počáteční a konc. hodnotu";Z,K
.
.
100 FOR A=Z to K
.
.
.
200 NEXT A
```

Takhle přece může smyčka vypadat, ne? Ale tato zdánlivě jednoduchá záležitost má svoje mouchy. Co když je například **K** větší než **Z**? Pak bude program procházet smyčkou je jednou, protože již po prvním průchodu bude proměnná **A** větší než **K**. Nějak tedy musíme tomuto případu zabránit. To se dá provést komplikovaně třeba pomocí podmínky IF - THEN a dvou rozdílných smyček pro případ, kdy je **K** větší než **Z** a kdy je **K** menší než **Z**. Ale to již zabírá paměť počítače, program se zbytečně natahuje nejen o jednu smyčku navíc ale i rozhodovací řádky.

V zásadě nejjednodušší řešení tohoto problému poskytuje funkce SGN, která je programátory velice málo používaná. Ruku na srdce, kdypak jste se pokoušeli řešit problém s pomocí této funkce? Podíváme se proto na funci **SGN** poněkud podrobněji.

Pokud zapíšeme například rovnici **V = SGN(výraz)**. Tato rovnice dává pro V hodnotu -1 v případě, že hodnota výrazu v závorce je menší než 1. Hodnotu 0 bude mít V tehdy, když výraz bude mít hodnotu 0 a hodnutu 1 v případě, že výraz bude větší než 0.

Takže například výsledek operace V = SGN(4-12) bude -1.

V našem případě využijeme příkaz SGN ke stanovení, zda naše smyčková proměná má klesat nebo stoupat. K tomu účelu musíme smyčku doplnit ještě o přídavek STEP, asi v této podobě:

```
10 FOR A = Z to K STEP SGN(K-Z)
```

Když nyní dojde k případu, že koncová hodnota bude menší než počáteční, bude kroku STEP přiřazena hodnota -1. To znamená, že smyčka bude odečítat směrem dolů. V opačném případě, kdy bude Z menší než K, bude kroku přiřazena hodnota +1 a smyčka bude počítat směrem nahoru. Nyní zbývá již jen maličkost, aby programovaná smyčka byla opravdu univerzální. Ne vždy totiž potřebujeme smyčku s krokem 1. Proto je nutno smyčku ještě doplnit do definitivní podoby

```
10 FOR A = Z to K STEP SGN(K-Z)*F (faktor)
```

Takto upravená smyčka je již univerzálně použitelná a jistě si najde ve Vašich programech místo.

2.2 Definice proměnných

Ve velice málo programech se také objevuje příkaz DEF FN, ačkoliv se jedná o mocný programový prostředek, s pomocí kterého můžeme rozšiřovat počet funkcí Basicu V2.0. Jeho syntaxe:

```
DEF FN proměnná1(proměnná2) = aritmetický výraz
```

je jednoduchá , takže proč vlastně není využíván? Asi proto, že si programátoři bez příkladu neuvědomují, jak dobrým pomocníkem by jim tento příkaz mohl být. Proto si uvedeme několik příkladů použití příkazu **DEF FN.**

Vezmeme například matematický program, ve kterém se vícekrát opakuje matematický výraz a*b + (c/2). Není nic jednoduššího, než pro tento případ použít definici

```
DEF FN F(X) = a*b + (c/2)
```

Okamžitě máme k dispozici definovanou funkci, kterou použijete jednoduše např. v programovém řádku

```
10 P = FN F(X)
```

X je v tomto případě tzv. slepá proměnná, která je potřebná jen k provedení definované funkce a slouží k tomu, aby Basic interpert nereptal. Lze také definovat funkci typu

```
F = X^2 * ¶
```

a to programovým řádkem

```
10 DEF FN F(x) = X * X * ¶
```

Takto definovaným vzorcem můžete počítat plochu kruhu s poloměrem x. Když zavoláte funkci například příkazem PRINT FN F (6), dostanete jako výsledek plochu kruhu s poloměrem 6. Lze jen litovat, že tato funkce, rozšiřující Basic o nejčastěji používané funkce, je tak málo používána. Ale to není ještě vše, co příkaz DEF FN umožňuje. Lze definovat funkce, které obsahují jiné, již definované funkce. Pokud například potřebujete počítat plochu válce, je nutno spočítat plochu pláště a obou kruhových podstav. Ty se počítají z průměru nebo poloměru a výšky. Bude tedy nutno definovat jednu

funkci, která poslouží pro výpočet plochy podstavy a druhou, která bude počítat plochu pláště. Ty se dají definovat takto:

```
10 DEF FN A(X) = X * X * ¶ : REM PLOCHA podstavy

20 DEF FN B(X) = 2 * x * ¶ * V: REM OBVOD
```

Nyní již nebude problém definovat funkci, počítající plochu vaice při daném poloměru X a výšce V:

```
30 DEF FN C(X) = 2 * FN A(X) + FN B(X)
```

Zkuste nyní zadat příkazový řádek

```
X = 3 : V = 8 : PRINT C(X)
```

Vidíte, funguje to! A přitom je to tak jednoduché.

Při své práci narazíte jistě na mnoho případů, kde se definice funkce bude hodit. Můžete si tak vytvořit celou knihovnu pomocných funkcí. Když se budete snažit, aby Vaše programy byly nejen funční, ale i přehledné, budete nacházet stále nové případy, kdy DEF FN bude vítaným pomocníkem.

(pokračování)

ASSEMBLER NA C64

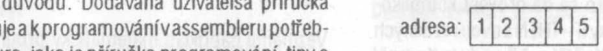
1. Úvod

Programování v assembleru není o tolik komplikovanější, než programování v Basicu. Není však tak rozšířeno mezi uživateli C64 jako programování v Basicu z více důvodů. Dodávaná uživatelsá příručka vůbec nezmiňuje a k programování v assembleru potřebná další literatura, jako je příručka programování, tipy a triky pro programátory v assemnbleru, komentovaný výpis ROM atd. nejsou, nebo spíše nebyly dosud na našem trhu dostupné. Situace se však zlepšuje a programátorům v assembleru se postupně vytváří podmínky pro vážnou práci i v českém jazyce. A abychom podnítili zájem začínajících programátorům a poskytli jim první informace, které jim umožní se rozhodnout se, zda jim assembler "chutná" nebo ne, museli jsme do nového časopisu zařadit i tuto rubriku pro začátečníky. Od příštího čísla pak budeme v rubrice Assembler uvádět i krátké programy ve strojovém kódu, aby si i zkušenější programátoři mohli doplnit svoji knihovnu.

2. Základní pojmy - početní soustavy

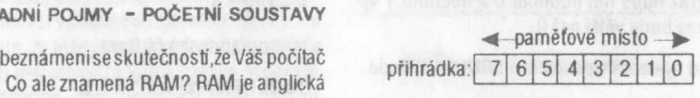
Určitě jste již obeznámeni se skutečností, že Váš počítač má 64kB RAM. Co ale znamená RAM? RAM je anglická zkratka názvu RANDOM ACCESS MEMORY, což přeloženo do češtiny znamená paměť s volitelným přístupem. Jedná se tedy o označení paměti. Paměť u počítače slouží k ukládání různých informací, nutných pro jeho práci. Pokud má paměť 64 kB, znamená to, že má celkem 65 535 "zásuvek", do nichž lze informace uložit. Pokud chceme, můžeme pak do těchto "zásuvek" také nahlížet, co v nich je, případně jejich obsah vyměnit za jiný. V řeči počítačových odborníků se tyto zásuvky nazývají paměťová místa nebo paměťový prostor. Pokud si představíte, že zásuvky jsou uspořádány jedna za druhou, můžete je také opatřit čísly nebo adresami. Paměťové místo 3 pak bude 3-tí zásuvka z leva.

Symbolicky si můžete představit toto uspořádání takto:



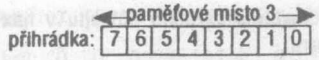
Protože používáme počítač a počítač je vlastně stroj na počítání, je logické, že paměť počítače nesloužík ukládání čehokoliv, ale jen čísel. Do každého paměťového místa lze uložit číslo od 0 do 255, tedy např. 26 nebo 148. Číslo mezi 0 a 255 se nazývá bajt (byte). K bajtu si toho musíme říci více, budeme to dále potřebovat.

Jak již bylo dříve řečeno, představuje bajt číslo 0 až 255. Proč ale ne 0 až 10 000 ? Odpověď na tuto otázku je poněkud komplikovaná. Představte si ,že každá zásuvka - paměťové místo je ještě rozdělena na 8 přihrádek. Paměťové místo pak bude vypadat následovně:



Každá z těchto přihrádek se bude nazývat odborně bit. U přihrádek budeme rozlišovat jen dva definované stavy, zdali jsou plné nebo prázdné. Oba tyto stavy - prázdná/plná - označíme v počítačové řeči jako pravdivý a nepravdivý, nebo jako 1 (pravda), nebo 0 (nepravda), místo plná a prázdná zásuvka. K tomu příklad:

Vezmeme např. paměťové místo 3. Vyplníme jeho přihr. 1, 3, 5, 7. Naše paměťové místo pak bude vypadat asi takto:



nebo s označeními 0 a 1:

a nyní ještě zbývá vysvětlit, jak se z této posloupnosti nul a jedniček získá číslo v rozsahu 0 až 255 a obráceně, aby si je počítač mohl uložit do svých paměťových míst. K tomu účelu si musíme nejdříve vysvětlit, jak funfuje náš známý dekadický početní systém.

Dekadický početní systém

V dekadickém systému existuje 10 čísel, odtud název dekadický systém. Podíváme se, jak se dekadické číslo spočte z jednotlivých cifer. Vezme si např. číslo 1987:

Místo:	3	2	1	0
Cifra:	1	9	8	7

Jak vidíte, na nultém místě jsou jednotky, na prvním desítky, na druhém stovky a na třetím tisíce. Rozdíl mezi jednolivými za sebou následujícími místy je dán stále stejným faktorem deset (1, 10, 100, 1000 ...), nebo lépe řečeno násobkem deseti. Tyto hodnoty (1, 10, 100) odpovídají mocninám 10 (10^0, 10^1, 10^2). Hodnota 1987 se spočítá dle tohoto postupu:

číslo na 0. místě krát 1	(10^0)
číslo na 1. místě krát 10	(10^1)
číslo na 2. místě krát 100	(10^2)
číslo na 3. místě krát 1000	(10^3)

= 7*1 + 8*10 + 9*100 + 1*1000 = 1987

Binární početní systém

Analogicky funguje tento systém v případě, že jsou k dispozici jen dvě

čísla (0 a 1). Tento systém, pouze se dvěma čísly se nazývá binární (dvojkový) systém. Obdobně jako u dekadického čísla máme za sebou jednotky, desítky, stovky atd., tak i binárního čísla následují za sebou čísla, která se však neliší o desetinásobek, ale jen o dvojnásobek. Nepočítáme tedy v násobcích deseti, ale dvou.

Nulté místo představuje dvojkové (binární) číslo 2^0 = 1, první místo 2^1 = 2, druhé místo 2^2 = 4, atd.

Dekadickou hodnotu binárního čísla z dříve uvedeného bajtu spočteme tímto postupem:

místo: 7 6 5 4 3 2 1 0
binární hodnota: 1 0 1 0 1 0 1 0 - binární číslo (bajt)

výpočet hodnoty:

číslo na 0. místě krát 1	(2^0)
číslo na 1. místě krát 2	(2^1)
číslo na 2. místě krát 4	(2^2)
číslo na 3. místě krát 8	(2^3)
číslo na 4. místě krát 16	(2^4)
číslo na 5. místě krát 32	(2^5)
číslo na 6. místě krát 64	(2^6)
číslo na 7. místě krát 128	(2^7)

= 0*1 + 1*2 + 0*4 + 1*8 + 0*16 + 1*32 + 0*64 + 1*128
= 2 + 8 + 32 + 128
= 170

Určitě teď chápete i způsob, jak se dá do jednoho bajtu zakódovat číslo 0 - 255. Pokud jsou všechny bity nastaveny na 0, je hodnota bajtu 0. Pokud jsou nastaveny všechny bity na 1, je hodnota bajtu

1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255.

Pomocí jednoho bajtu se tedy dá zobrazit 256 čísel (0 - 255).

Ještě jednou shrnuto

Paměť počítače se skládá z paměťových míst umístěných v řadě za sebou. Číslo paměťového místa v řadě se označuje jako adresa paměťového místa. Do každého paměťového místa se vejde 1 bajt (byte). Jelikož bajt sestává z 8 bitů, které mohou nabývat buď hodnotu 0 nebo 1. Každému bitu odpovídá určitá mocnina dvou, např. třetí místo pak odpovídá 2^3 = 8, pokud je odpovídající bit nastaven na 1. Pokud se sečte hodnota všech nastavených bitů v bajtu, obdrží se největší hodnota celého bajtu.

Hexadecimální početní sytém

Jistě Vás už napadá, že práce s binárními čísly je poněkud pracná a zdlouhavá. Proto se používá ve výpočetní technice ještě jeden početní systém, šestnáctkový, neboli hexadecimální, který je svým základem 16 příbuzný binárního systému. Tato příbuznost nám usnadní některé přepočty, jak si dále ukážeme. Hexadecimální systém získáme rozdělením jednotlivých bajt; na dvě části - půlbajty, neboli tzv. nibbly. Jeden bajt pak vypadá následovně:

	půlbajt				půlbajt			
bit	7	6	5	4	3	2	1	0

Pokud se rozdělí i přiřazované mocniny dvou, odpovídající jednotlivým bitům, získáme takovéto rozdělení:

bit	3	2	1	0	3	2	1	0
hodnota	8	4	2	1	8	4	2	1
=	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0

Z toho vyplývá, že pomocí půlbajtu (nibblu) můžeme zobrazovat čísla 0 - 15 (1 + 2 + 4 + 8 = 15). Abychom mohli hodnotu obou půlbajtů zabrazovat jednou cifrou, jedním znakem, musí být k dispozici početní systém s 16 čísly. Poněvadž však známe pouze čísla 0 - 9, musíme místo ostatních chybějících použít písmena abecedy. V takovém početním systému, který se nazývá hexadecimální podle čísla 16, které je jeho základem, je používáno dále uvedené označení jednotlivých čísel:

dekadicky	binárně	hexadecimálně
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Pokud je třeba převést například dekadické číslo 114 do hexadecimálního tvaru, postupujeme následovně:

1. číslo se převede do binárního tvaru:

bit:	7	6	5	4	3	2	1	0
hodnota:	0	1	1	1	0	0	1	0

2. bajt se rozdělí na dva půlbajty - nibbly:

bit:	3	2	1	0	3	2	1	0
hodnota:	0	1	1	1	0	0	1	0

3. spočítá se hodnota každého půlbajtu v hexadecimálním tvaru:

hodnota:	0	1	1	1	0	0	1	0
	0+4+2+1				0+0+2+0			
	7				2			

4. nakonec se dekadická hodnota obou nibblů převede na hexadecimální a sečte se:

113 = $72

Obdobně se převádí čísla z hexadecimálního tvaru do dekadického. Přitom je nutno pamatovat, že v dekadickém systému má každé místo hodnotu odpovídající mocnině 10, u binárního systému má každé místo hodnotu odpovídající mocnině 2. Obdobně u hexadecimálního systému bude mít každé místo hodnotu odpovídající mocnině 16. Dekadické číslo přepočteme z hexadecimálního (např. $D3) takto:

D * 16^1 + 3 * 16^0 = 13 * 16 + 3 + 1 = 208 + 3 = 211

Jak vyplývá z výše uvedeného zápisu, používáme pro označení hexadecimálních čísel symbol dolaru - $. Obodbně pak pro binární čísla existuje označení % (procenta).

Například: 255 = $FF = %11111111

3. Od Basicu ke strojovému kódu

Funkce paměti počítače a převody mezi binární - dekadickou - hexadecimální soustavou jsou základem pro práci ve strojovém jazyku nebo assembleru. Nejdříve si však probereme Basicovský příkaz POKE. Zadání parametrů u příkazu POKE je následující:

POKE adresa, hodnota

Adresa musí ležet v rozsahu 0 - 65535 a hodnota mezi 0 - 255. Jistě si teď dokážete představit, co je míněno tou adresou, neboť jsme si tento pojem již dříve objasnili. Je tím míněno číslo paměťového místa nebo - jako v našem případě - číslo zásuvky. Druhý parametr, hodnota, může být jakýkoliv matematický výraz, jehož hodnota leží v rozsahu 0 - 255, tedy může být zapsána jedním bajtem. Pomocí příkazu POKE lze pak uložit do určitého paměťového místa definovanou hodnotu. Pokud zadáte počítači například příkaz:

POKE 288,17 <RETURN>,

uloží se do paměťového místa s dekadickou adresou 288 hodnota 17. Bajt vypadá v binárním vyjádření následovně:

```
bity:  7  6  5  4  3  2  1  0
bajt:  0  0  0  1  0  0  0  1

hodnota:
 0*2^7+0*2^6+0*2^5+1*2^4+0*2^3+0*2^2+0*2^1+1*2^0
=0   +0   +0   +16  +0   +0   +0   +1
=                16                 +1
=                                   17
=$               1                  1
=$                                  11
```

Známe tedy již jeden příkaz, pomocí kterého můžeme vytvořit spojení mezi Basicem a strojovým jazykem.

K příkazu POKE, sloužícímu k ukládání čísel do paměti počítače existuje analogicky i opačný příkaz, pomocí kterého lze do paměti počítače nahlížet a kontrolovat, co je v jednotlivých paměťových místech uloženo. K tomu účelu zadejte příkaz

PRINT PEEK (288) <RETURN>.

Číslo za příkazem PEEK je adresa paměťového místa, kterého obsah chceme znát. Jako výsledek tohoto příkazu se na obrazovce zobrazí číslo 17. Zapište výše uvedený příkaz ještě jednou a proveďte jej stiskem RETURN. Objeví se opět číslo 17. To znamená, že se obsah paměti při čtení nemění.

A shrnutí na závěr

Pomocí příkazu POKE je možno určité číslo uložit (zapsat) do stanoveného paměťového místa. Analogicky je možno pomocí příkazu PEEK číst hodnoty, uložené v paměti počítače, aniž se touto operací obsah paměti změní.

4. MONITOR STROJOVÉHO KÓDU

Zápis strojového programu do paměti počítače vyžaduje změnit jednotlivé bity paměťových míst počítače, což je velmi pracné. Takovým způsobem prováděné programování by bylo velice zdlouhavé, nehledě na množství chyb, které by se při tomto postupu vyskytovaly. Člověk by musel z tabulky vyhledávat číslené kódy jednotlivých instrukcí procesoru a zapisovat je, relativní skoky by bylo nutno dopředu počítat, atd. Z toho důvodu byla vynalezena pomůcka, které se říka ASSEMBLER. ASSEMBLER zjednodušuje zapisování kódů strojových instrukcí pomocí tzv. mnemokódů. Je to překladač mnemokódů do číselných kódů. Mnemokódy jsou prakticky symboly jednotlivých strojových instrukcí. Mnemokód LDA #$10 odpovídá ve strojovém kódu posloupnosti bajtů S A9 10.

V dalším popisu se budeme důsledně držet těchto mnemokódů, které zápis strojového programu výrazně zjednodušují. A k dodržení úplnosti budeme kromě mnemokódu uvádět i jeho hexadecimální tvar.

Aby bylo možno na počítači Commodore 64 programovat ve strojovém kódu, doporučuje se použít tzv. MONITOR strojového kódu. MONITORem strojového kódu není přirozeně myšlen žádný přístroj k zobrazení dat, ale program, pomocí kterého je možno strojový program zapisovat pomocí uvedených mnemokódů. Kromě toho lze pomocí monitoru prohlížet obsah paměti, ten pak měnit, porovnávat, ukládat či natahovat na a ze záznamových médií (kazety, disku). Dále je možno strojové programy zobrazovat ve formě disassemblovaných listingů, t.j. ve formě mnemokódů a instrukcí.

Ale vraťme se k mnemokódům. Mnemokód je skupina tří písmen, která jsou zkratkou instrukce, jíž zastupují. Jako příklad použijeme mnemokód **TAX**. Tato zkratka nahrazuje příkaz Transfer Accumulator to X-Register. Jak vidíte, pod mnemokódem si může programátor představit více než pod číslem SAA, které je hexadecimálním kódem instrukce **TAX**.

Jednotlivými příkazy assembleru se budeme zabývat později. Ještě však pár slov k funkcím, které monitor strojového kódu poskytuje. Monitor je program. Je tedy nutno s ním jako s programem zacházet. Commodore 64 nemá bohužel na rozdíl od svých mladších bratrů C16, P/4 nebo C128 tento program v systému zabudovaný. Je proto nutno jej do počítače natahovat z kazety, diskety nebo modulu. Monitor existuje v mnoha variantách, majících shodný základ a lišících se hlavně v adresovém prostoru, do kterého se ukládají, dále pak ve vedlejších funkcích, které poskytují (například přepočty čísel mezi soustavami).

Zaujímaný adresový prostor je důležitý zvláště u delších programů. Program nemůže být ukládán do stejného adresového prostoru který je obsazen monitorem, což může být někdy rozhodující. Známé jsou například monitory **PROFI-MON 64, ASSEMBLER, HES, MON 64, MONITOR, MES MON**, atd. Jejich základní funkce jsou ve všech případech shodné a lze je shrnou do následujícího přehledu:

A -	Assembler	assemblování, překlad mnemokódů do hex. tvaru,
C -	Compare	porovnání,
D -	Disassemble	disassemblování,
F -	Fill	vyplnění,
G -	Go	start programu,
H -	Hunt	hledání,
L -	Load	natahování programu,

S - Save ukládání programu,
T - Transfer kopírování
V - Verify porovnání,
X - eXit výstup z monitoru.

Kromě těchto příkazů mají v monitorech svůj význam i tečka, středník, otazník, závorka, mezera. Je nutno je respektovat při zápisu programu i práci s monitorem. Dále se seznámíme podrobněji s jednotlivými příkazy assembleru.

Popis nejdůležitějších příkazů monitoru

A - Assemble

Pomocí příkazu "A" je možno zapisovat programy ve formě mnemokódů. Za příkazem A musí být zapsána adresa v 16ti bitovém formátu, t.j. jako čtyřmístné hexadecimální číslo. Všechny zápisy čísel se v monitoru provádějí v hexadecimálním tvaru. V případě zápisu adresy však odpadá nutnost zapisovat znak $ před číslem. Za adresou následuje mnemokód a za ním parametr. Například:

```
A 3000 LDA #$50 <RETURN>
▲ ▲    ▲   ▲
a b    c   d
```

a - příkaz, v našem případě A - Assemble,

b - adresa, od které má být program v assembleru do paměti ukládán,

c - mnemokód instrukce, která se má provést,

d - parametr instrukce, který bude popsán dále, kdy si budeme význam instrukcí vysvětlovat.

Poté, co je řádek zapsán například dle vzoru, je nutno stisknout klávesu RETURN a tak příkazový řádek ukončit. Po stisku klávesy RETURN monitor přeloží mnemokód do číselného kódu instrukce, zapíše jej spolu s parametrem do paměti počítače v pořadí instrukce, parametr. Dále přepíše zápis na obrazovce číselným kódem instrukce a parametrem a náš zápis posune až za ně. Nakonec na nový řádek napíše A a další volnou adresu, například:

```
A 3002.
```

Kurzor stojí za adresou a čeká na zápis další instrukce v mnemokódu. Mnemokódy je nutno zapisovat přesně. V případě chybného zápisu nebo jiné syntaktické chyby vypíše monitor na obrazovku otazník (?) v místě, kte si již dál neví rady.

C - Compare

Příkazem pro porovnávání lze snadno porovnat dva různé paměťové rozsahy. K tomu účelu se za příkaz C zadá jako parametr počáteční a koncová adresa prvního rozsahu a počáteční adresa druhého rozsahu. Po stisku RETURN se vypíší všechny adresy druhého rozsahu, které mají obsah odlišný od odpovídajících adres prvního rozsahu.

Chceme například srovnat rozsah $E000 až $E003 s rozsahem od $F000.

Oba rozsahy, pokud se na ně podíváme do paměti počítače, vypadají takto:

```
$E000: $20 A4 CA 91
$F000: $50 8C 37 05
```

Nyní zapište následující posloupnost příkazů :

```
C E000 E003 F000
```

a uzavřete ji stiskem RETURN. Na obrazovce se vypíší adresy

```
E000 E001 E002 E003.
```

Jak vidíte, nejsou na žádných dvojicích odpovídajících adres shodné hodnoty. Proto byly po porovnání vypsány všechny adresy, které byly přezkoušeny.

D - Disassemble

Díky příkazu D má uživatel k dispozici přesně opačnou funkci, než je funkce A - Assemble. Prostřednictvím přípazu D je dle možnosti obsah paměti zobrazen ve tvaru mnemokódu strojových instrukcí a jejich parametrů. Pokud dané číslo nemá význam jako kód strojové instrukce, je tato nahrazena trojicí otazníků. Protože disassemblovaný výpis je svým tvarem vhodný pro editaci programů v assembleru, je každý řádek výpisu možno editovat. K tomu opět příklad:

```
D A785 A795 <RETURN>
```

Po stisku RETURN se na obrazovce objeví výpis:

```
A785 0F    ???
A786 85 95 STA $95
A788 90 30 BCC $A7BA
A78A B0 3C BCS $A7C8
A78C A5 95 LDA $95
A78E C9 91 CMP #$91
A790 F0 18 BEQ $A7AA
A792 C9 20 CMP #$20
A794 B0 BC BCS $A752
```

Koncová adresa rozsahu, která má být disassemblován, nemusí být uvedena. Pokud ji nezapíšeme, vypisuje se disassemblovaný obsah paměti až do dosažení adresy FFFF nebo stisku klávesy RUN-STOP.

F - Fill

Pomocí příkazu F lze vyplnit zadaný paměťový rozsah určenou hodnotou. Zápis příkazu a parametrů vypadá obvykle takto:

```
F 0801 0901 00
  ▲    ▲    ▲
  a    b    c
```

a - počáteční adresa paměťového rozsahu, který má být vyplněn určenou hodnotou,

b - konec paměťového rozsahu, který má být vyplněn určenou hodnotu,

c - hodnota, kterou má být definovaný paměťový rozsah vyplněn.

Tento příkaz se používá hlavně v těch případech, kdy má být vyvíjený program vybaven určenou výstupní podmínkou, nebo při přípravě programů pro EPROM.

G - Go

Pomocí příkazu G lze spouštět strojové programy. J však nutné, aby byly ukončeny instrukcí BRK. Tato instrukce způsobí skok zpět do monitoru. Programový ukazatel PC ukazuje po takovém ukončení programu na adresu, následující ihned za instrukcí BRK, která program přerušila. Proto je poměrně jednoduché testovat pomocí monitoru strojové programy. Je pouze nutné na kličových místech zabudovat instrukce BRK, aby se dal program testovat po částech. Předávání parametrů je u příkazu G následující:

Za příkazem G se zapisuje pouze adresa, od které má být strojový program spouštěn. Při startu programu pomocí příkazu G se obsah registrů procesoru nastaví shodně se zobrazovaným (viz příkaz R). K příkazu G ještě příklad:

```
G 9DE6 <RETURN>
```

Jelikož na adrese 9DE6 je uložena instrukce BRK s kódem $00, dojde po zapsání příkazu a stisku <RETURN> okamžitě ke skoku zpět do monitoru. Na obrazovce se objeví toto hlášení:

```
B* (BREAK)
  PC   IRQ  SR AC XR YR SP
 9DE7  EA31  31 42 FA 9D FE
```

Jak vidíte, stojí programový čítač na adrese $9DE7, což je adresa následující ihned za adresou $9DE6, na které byl předtím ukončen strojový program (nebo správně řečeno zahájen a ukončen) instrukcí BRK.

Je to tedy zařízeno tak, aby bylo možno v programu pokračovat ihned za místem kde došlo k přerušení. Pokud se totiž zadá pouze příkaz G bez parametru (adresy), je program odstartovnán od adresy která je uložena v programovém čítači. V našem případě je to adresa následující za příkazem BRK.

H - Hunt

Tento příkaz je užitečný k tomu, abychom v paměti mohli vyhledávat určitou posloupnost bajtů, případně textové řetězce. Jako parametry příkazu se zadávají počáteční a koncová adresa paměťového rozsahu, který má být prohledáván a posloupnost bajtů, případně textový řetězec, po kterých pátráme.

Pokud chceme například v paměťovém rozsahu $A000 až $AFF hledat posloupnost bajtů $0A 0B 0C, zadáme:

```
H A000 AFFF 0A 0B 0C <RETURN>
```

Okamžik po stisku RETURN se na obrazovce objeví na novém řádku adresa $A418 a na dalším řádku kurzor. Pokud si prohlédneme paměť počítače například dále popisovaným příkazem M, najdeme skutečně od adresy $A418 posloupnost bajtů 0A 0B 0C 0D 0F ....

V případě, že pomocí příkazu H hledáme v paměti texový řetězec, je nutno jej označit uvozovkou nebo apostrofem, takže zápis pak vypadá následovně:

```
H 8000 EFFF "BASIC <RETURN>
```

Za hledaným řetězcem uvozovky nebo apostrof nezapisujeme, monitor by je považoval za součást hledaného řetězce. Po chvíli hledání se na novém řádku obrazovky vypíše:

```
E461 E48B
```

Na uvedených adresách jsou totiž uloženy části textu úvodního hlášení systému počítače.

L - Load

K přenášení programu z externího záznamového média (v našem případě nejspíše magnetofonu nebo disketové jednotky) do paměti počítače slouží právě příkaz L. Zadání parametrů odpovídá příkazu LOAD v Basicu V2.0. Název programu musí být v uvozovkách, číslo periferie, které je uváděno vždy hned za názvem

programu musí být dvoumístné hexadecimální číslo. Vzhledem k tomu, že naše obvyklé periferie mají nízká čísla (magnetogon 01 a disketová jednotka 08), jsou vlastně shodná s dekadickými.

Příklady:

```
L"TEST",08 <RETURN>
```

způsobí natažení programu s názvem TEST z diskety v disketové jednotce 08 do paměti počítače. Program se natáhne do stejného adresového prostoru, ze kterého byl na disketu uložen. Pokud zápis změníme na

```
L"TEST",01 <RETURN>
```

natáhne se program z kazetového záznamníku.

Ne vždy potřebujeme uložit program na stejné místo, ze kterého byl na externí paměť ukládán. Pro tento případ většina monitorů umožňuje ještě jeden tvar zápisu příkazu L, doplněný nakonec adresou, od které se má program ukládat. Zápis pak vypadá takto:

```
L"TEST",08,2A00 <RETURN>
```

a způsobí natažení programu TEST z disketové jednotky a uložení od adresy $2A00.

M - memory (modify)

Pokud chceme prohlížet obsah paměti počítače v hexadecimálním tvaru nebo ASCII znacích, případně měnit její obsah, je k dispozici příkaz M. Zadání parametrů je stejné jako u příkazu D, to znamená, že zapisujeme buď dvě adresy, nebo jen jednu, případně žádnou. Podíváme se na paměťový rozsah $E460 až $E470. K tomu účelu zapíšeme:

```
.M E460 E470 <RETURN>
```

Na obrazovce se objeví výpis:

```
:E460 20 42 41 53 49 43 20 42  BASIC B
:E468 59 54 45 53 20 46 52 45  YTES FRE
:E470 45 0D 00 93 0D 20 20 20  EM@♥M
```

a b c

a - hexadecimální adresa prvního pamětového místa v řádku,

b - obsah osmi paměťových míst v řadě za sebou v hexadecimálním tvaru,

c - obsah osmi paměťových míst vypsaný jako ASCII znaky.

Znaky, které jsou podtržené, se zobrazují inverzně. Pokud chcete obsah paměti změnit, je to možné v části obsahující hexadecimální výpis paměti. Stačí pouze kurzorem najet na některé číslo a přepsat je. Po ukončení úprav na řádku není nutné zajíždět kurzorem na konec řádku. Bez ohledu na postavení kurzoru na řádku stiskněte RETURN a opravený zápis se uloží do paměti počítače. Poznáte to podle změněného zápisu v části ASCII.

Ne všechna čísla od 0 po 255 mají vyjádření jako ASCII znaky. Řada kódů má například význam jako řídící znaky (HOME, CLEAR SCREEN, RVS ON, RVS OFF, CURSOR DOWN, CURSOR UP, RETURN atd.) Proto se ve sloupci, obsahujícím výpis paměti v ASCII vyjádření často objevují tečky (.), kterými monitor nahrazuje kódy, nezobrazitelné ve formě znaku.

(pokračování)

GEOS

Proč právě GEOS?

Nezáleží na tom, zda s počítači pracujete již déle, nebo zda C64 představuje na tomto širokém poli Váš prvý krok. Sami brzy přijdete na to, že GEOS je nejjednodušším řešením, které maximálně využívá možnosti C64.

Jak je to možné? Už sám název Graphic Environment Operating System napovídá, že GEOS je orientován spíše graficky, než textově. Otevře Vám cestou myší, piktogramů, okének a roletových menu svět známý z PC. Potřebujete pouze jednoduché odkliknutí myší či joystickem.

Objekty jako datové soubory nebo diskety, ale i činnosti jako tištění dokumentů nebo listování jsou zobrazeny formou malých obrázků (angl. Icons). Ostatní povely jsou pohodlně dosažitelné z roletových menu, které jsou umístěny na horním okraji obrazovky.

GEOS je tak senzační a různorodý, že jeho mnohostrannost správně oceníte až v praxi. Stejně jako ostatní softwarové produkty, doznal GEOS řadu zlepšení. Od verze I.I přes I.2 a I.3 k verzi 2.0. Zvláště posledně jmenovaná verze z výkonného hardware C64 nebo C128 díky geniálnímu software, který vůbec pro tuto kategorii počítačů existuje, vyždímá maximum. Tyto možnosti, jež dlouhá léta dřímaly v pouzdrech

integrovaných obvodů C64 a C128 probudil až GEOS. Pokrok představoval už ve své první verzi. Software nové generace. GEOS 2.0 je korunou úsilí programátorů. Jeho nasazením se C64, C128 nemusí stydět před žádným PC, Macintoshem či Amigou. Pokud se spočítá poměr cena-výkon, vychází GEOS jako jasný vítěz.

Řada uživatelů práci s GEOSem špatně chápe. Vidí jen výhody pohodlné práce aniž by si uvědomila, že systém GEOS s sebou přináší řadu novinek. Zjednodušeně je lze charakterizovat následovně:

- podpora všech disketových jednotek a RAM modulů, lepší a rychlejší než bez GEOS.

- pohodlnější práce s disketami a datovými soubory, protože GEOS je automaticky pozná a reaguje na ně.

- princip WYSIWYG - na obrazovce je vidět všechno tak, jak bude později vytištěno: What You See is What You Get.

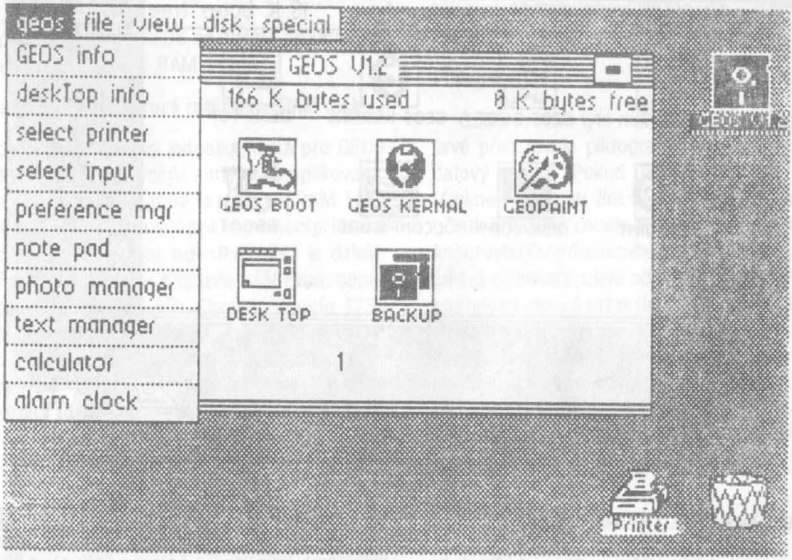
- GEOS nastavil laťku značně vysoko při výměně dat (př. spojení grafiky a textu, sestavování serlových dopisů).

Jistě by se dala snést ještě řada dalších argumentů, která by hovořila ve prospěch GEOSu.

Jak vypadá dodávka.

Po zdůraznění těchto důležitých vlastností by se čtenář mohl domnívat, že GEOS je rozsáhlé hardwarové rozšíření pro C64 nebo C128. Zcela obráceně. GEOS je softwarový produkt, program. Přesněji řečeno GEOS sestává z řady programů a datových souborů, jež jsou dodávány na disketách. Spolu s disketami, které jsou čitelné obvyklými floppydiskovými jednotkami se dodává obsáhlý manuál (německy, anglicky).

GEOS 1.1
představoval jednu stranu diskety.



- pružné možnosti tisku

- veškeré místo na datovém nosiči může být využito jako virtuální paměť

- GEOS se orientuje konsekvenčně na profesionální systémy. Tedy ten, kdo jej ovládá může okamžitě pracovat s Apple Macintosh a obráceně

- na bázi GEOSu byly vytvořeny mnohé produkty, které ve svém oboru obtížně nacházejí konkurenci(př. Geo Publish v sektoru DTP)

GEOS 1.2
již zabral celou oboustranně nahranou disketu.

GEOS 1.3
spotřeboval dvě oboustranně nahrané diskety.

GEOS 2.0
je umístěn na čtyřech oboustranně nahraných disketách.

Jistě! Důležitější, než absolutní počet disket je jejich obsah. U GEOSu 64 2.0 mají diskety srovnatelný či dílem identický obsah. Prvá disketa obsahuje systémo-

vé programy a rutiny a jejich aplikace (Boot, Print Drivers, Geo Write 2.1, Geo Print 2.0, fotomanager, budík, poznámkový blok). Jedna strana druhé diskety je kopií systémových programů z prvé diskety. Na druhé straně jsou umístěny utility pro psaní (Write Utilities): Geo Merge, Geo Laser, Text Grabber, znakové sady pro laserový tisk, další tiskové budiče pro zmenšený tisk nebo pro vícenásobný tisk. Třetí disketa je německo-anglický Geo Spell. Na čtvrté disketě jsou k nalezení budiče pro různé typy tiskáren a programy GEOS-demo.

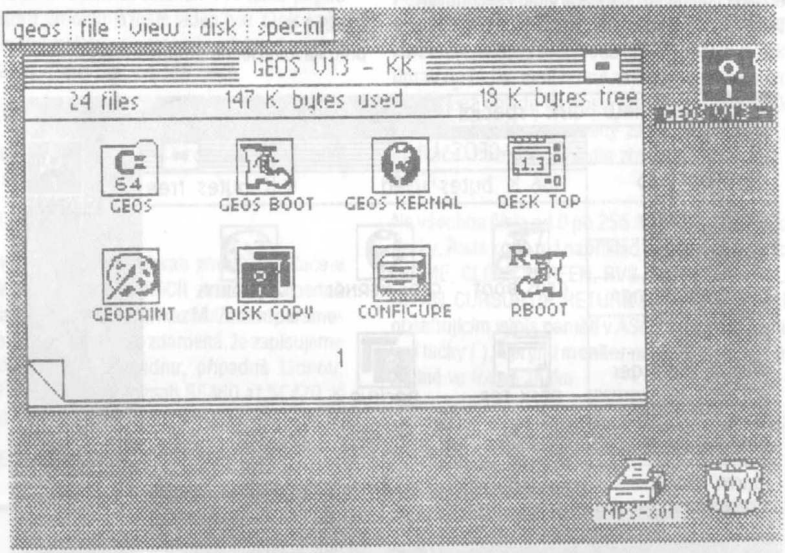
Zákazník, který se rozhodl pro GEOS a legálně jej zakoupil, může odeslat k registraci vyplněnou kartu s výrobním číslem a potom může využívat výhod služby zákazníkům.

Pro Amigu byly už vyvinuty emulátory činnosti C64. GEOS žel nepatří mezi programy, které by se pomocí nich daly na Amize provozovat.

Monitor.

Typ vhodného monitoru je vázán na konfiguraci celého počítačového systému (C64 . C128) a na odpovídající verzi GEOS.

GEOS 128 2.0 - většina vlastníků tohoto GEOSu vyřešila svůj problém s monitorem volbou typu (př. 1084S), který umožňuje přepínat mezi 40 a 80 znakovým módem. To je při nasazení GEOSu velmi výhodné, neboť aplikace pro C128 běží většinou v 80 znakovém módu, kdežto Geo Publish a Geo Chart pouze ve 40 znakovém módu.



Možný či nutný hardware?

V následujících odstavcích vysvětlím jaký hardware je pro činnost GEOSu nezbytný, či jaký se doporučuje.

Počítač.

Jelikož jste uživatelem C64 s určitostí na vašem počítači GEOS poběží. Lhostejno, zda máte starý C64 nebo novější C64II. Držitelé C128 se mohou u verse GEOS 2.0 rozhodnout a zvolit mezi GEOS 64 2.0 a GEOS 128 2.0. Speciální verzi pro C 128 lze jen doporučit, neboť akceptuje všechny výhody C 128: větší paměť, lepší možnosti grafiky, rychlejší procesor s taktem 2 MHz, klávesnici pro zadávání číslic.

GEOS 64 2.0 - postačuje monitor s kompozitním vstupem (CVBS). Použitelný je i televizor, ovšem nedoporučuje se. Pokud z jakýchkoli důvodů přece jen chcete použít TVP pamatujte na následující pravidlo: proporcionální písmo GEOSu je často na TVP s malou uhlopříčkou čitelnější, než na větších přístrojích.

Disketové jednotky a RAM moduly.

Centrální periferní jednotkou pro GEOS je disketová jednotka. Pro přečtení systémových disket 5 1/4" přicházejí do úvahy mechaniky 1541, 1541c, 1541II a 1571 stejně jako integrovaná jednotka C128D.

Všechny jmenované disketové jednotky jsou GEO-

Sem 5 až 7* zrychleny. Pozoruhodná je ta skutečnost, že pod GEOS 2.0 je drive 1571 spojený s C64 také schopen zapisovat oboustranně, což jde normálně jen s C128.

Jako druhý přístroj je možno použít 3 1/2" disketovou jednotku 1581. Tento přístroj se sice už nevyrábí ovšem občas bývá k mánía nabízí se jako velmi výkonné paměťové médium s kapacitou 790 kB. Rutiny GEOSu z něj dělají velmi spolehlivý přístoj, neboť obcházejí slabá místa operačního systému 1581.

Maximální počet disketových jednotek.

GEOS 2.0 pracuje maximálně se třemi "logickými" disketovými jednotkami. Jednou z disketových jednotek je některý z typů 1541, 1571 nebo 1581 nebo RAM disk 1750 eventuelně 1764. Pokud nepoužijete RAM - disk, obslouží GEOS pouze dvě disketové jednotky současně. Jsou-li tyto jednotky různých typů (např. 1541 a 1581) musí se celý systém takzvaně nakonfigurovat.

Principielně pracuje GEOS se dvěma disketovými jednotkami a eventuelně s RAM diskem.

Nejvýznamnější hardwarové rozšíření - RAM disk.

Každá druhá disketová jednotka nabízí pro GEOS řadu výhod. Ještě efektivnější a méně komplikované, než další disketová mechanika je rozšíření RAM 1750 nebo 1764. GEOS 2.0 simuluje potom pomocí přídavné paměti druhou disketovou jednotku, která je daleko rychlejší, než pravé floppy. Kapacita RAM disku odpovídá přitom disketě (165 kB). Pomocí modulu 1750 dosáhnete dvojnásobné kapacity 330 kB stejně jako u mechanik 1571. Zakoupení RAM disku pro práci s GEOSem vřele doporučuji. Před koupí však pečlivě rozvažte,které rozšíření se pro vás jeví jako výhodné. Neleňte a nechejte si je předvést ve spolupráci se svým C64 a GEOSem.

Ovládání.

GEOS se ovládá shodně s ostatními grafickými uživatelskými systémy bez použití klávesnice. Po obrazovce se pohybuje pomocí joysticku nebo myši malou šipkou.

Joystick - Každý začátečník s GEOSem používá k jeho ovládání joystick zasunutý do portu 1. Pozor, nezapomeňte vypnout auto-fire!

Myš 1350 - Používá se přesně jako joystick. Principielně se také o joystick v myším pouzdře jedná. Přesto má tato myš oproti joysticku výhodu v ovládání jednou rukou. Co se týče přesnosti a rychlosti nepřináší pseudomyš 1350 žádný pokrok.

Myš 1351 - Je optimálním ovladačem pro GEOS. Ostatní typy myší tlačí do pozadí. V přesnosti a rychlosti předčí myš 1351 i ten nejdražší joystick. Práce s ní přináší požitek.

Tiskárna.

Už z dřívějška víte, že "G" v názvu GEOS je vyhrazeno pro "GRAPHIC". GEOS tiskne z grafického módu. Program obsahuje budiče pro snad všechny tiskárny tisknoucí grafiku včetně tiskáren laserových.

Šipka.

Při práci s GEOSem potřebujete klávesnici jen zřídka. Příkazy zadáte jednoduše nastavením ukazovátka a odklepnutím. Ukazovátko má nejčastěji formu šipky, může mít však i jiný tvar a může jej měnit (viz. Geo Paint). Aby se šipka pohybovala rychle a přitom přesně, je její rychlost upravena. Při přesouvání na delší vzdálenosti je pohyb zrychlen, na kratších drahách pojíždí pomaleji. Tím je umožněno přesné najetí jednotlivých bodů. Tvar, barvu a rychlost ukazovátka si podle přání můžete sami měnit.

Piktogramy.

GEOS využívá tyto malé obrázky pro více učelů. Za prvé představuje piktogram objekty jako disketa či datový soubor. Pokud ukážete na takový piktogram (šipkou), sdělujete tím GEOSu, že s ním něco chcete učinit. Příkladně chcete tímto způsobem nechat vytisknout vytvořený dokument. Za druhé mohou být využity také pro činnosti, které od GEOSu žádáte. Pokud třeba ukážete na ohnutý rožek (levý dolní) a odklepnete, otočí GEOS list a přejde na další. V mnoha uživatelských programech představují piktogramy účinné nástroje. Příkladem může posloužit Geo Paint s celou řadou nástrojů pro kreslení, umístěných na levém okraji obrazovky. Pro volbu příslušného nástroje potřebujete opět najet šipkou na žádaný piktogram a odklepnout.

Menu.

Protože GEOS nemůže zobrazit každý příkaz jako piktogram, má v záloze připravenu řadu rolovacích menu. Místo toho abyste dlouze vypisovali název příkazu můžete opět ukázat šipkou na příslušný řádek menu. Jednotlivá menu (různá podle aktuálního programového okolí) je možno vyvolat pod soubory na horním okraji obrazovky. Pokud najedete šipkou na příslušný titulek menu a odklepnete, objeví se okamžitě další menu ("spadne roleta"). V něm zvolíte šipkou příkaz a k jeho provedení ještě jednou odklepnete.

V mnoha případech neobsahuje menu žádné příkazy, ale řadu možností. Dobrým příkladem je volba druhu písma v Geo Paint. Zvolíte druh písma, odklepnete, a nic

se neděje. Změnu zpozorujete teprve tehdy, když zadáte něco přes klávesnici.

Další druh menu obsahuje řadu piktogramů. Jednoduchým odklepnutím se dají změnit. Příkadem může být menu se značkami přístrojů a módů.

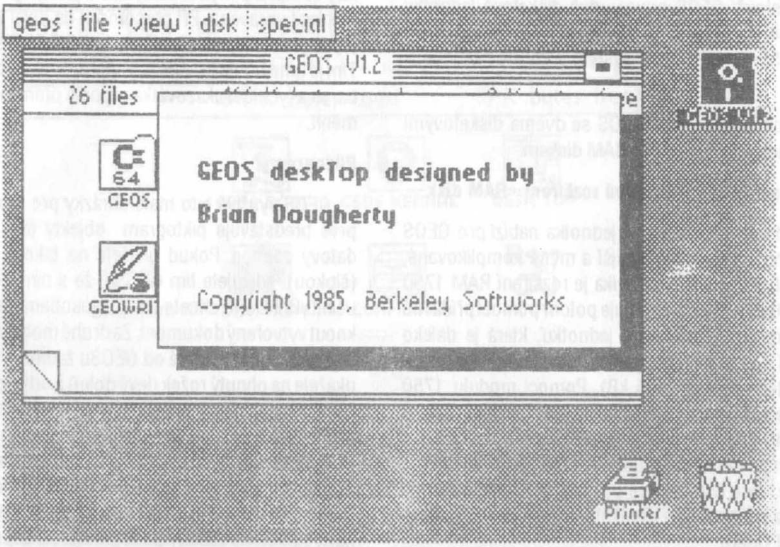
Vedle objasněných menu existuje řada submenu. Pokud zvolíte v Geo Paint nebo Geo Write druh písma, odklepnete nejprve v seznamu povelů fonts. Seznam druhů písma se objeví stejným způsobem jako už u známých menu. Nyní zvolíte některý z druhů písma (př. Cory). Soubor nebude okamžitě aktivován nýbrž se objeví submenu, které k tomuto druhu písma přiřadí jeho velikost. Teprve po této volbě budou příslušná data dotažena z diskety.

Zpravidla postačuje najetí šipkou a odklepnutí.

Pomocné programy.

Práce s GEOSem přináší ještě jeden zážitek. Můžete používat připravené pomocné programy, které plní jednoduché úlohy v rámci aplikačního programu. Příklad: během práce s Geo Write můžete vyvolat pro numerické výpočty pomocný program "Kalkulátor" a výsledky použít v textu, aniž byste museli vytvořený text ukládat na disketu a opouštět Geo Write. Seznam pomocných programů, které jsou k dispozici (totiž těch, které jsou na disketě) se nachází v menu GEOS v levém horním rohu.

Může se stát, že jste si vyvolali pomocný program,



Okna.

GEOS dává informace pomocí ikon. Pomůcka" zápisník (Notepad) otevírá své vlastní okno do kterého si můžete napsat svůj vlastní text a datový soubor uložit na disketu. Okno pro kreslení v Geo Paintu a pro text v Geo Write jsou dalšími příklady. Uvnitř těchto oken bývá zpravidla ukázána část celkové informace (text, obrázek). Pomocí nástrojů může být okno libovolně posunováno po obrazovce.

"Komunikační okna" jsou speciálním druhem oken. Pokud se objeví, musíte nejprve odpovědět a teprve potom smíte pracovat dál. Komunikační okno obsahuje přesné návěští nebo otázku na niž musíte odpovědět.

který však neběží. Ponejvíce je to proto, že disketa neobsahuje dostatek místa nutného zpravidla pro mezioperace.

Integrace uživatelských programů pod GEOSem.

Touto vlastností se rozumí, že všechny uživatelské programy užívají stejný systém a stejné ovládací prvky. Zjednodušuje to práci, neboť postačí naučit se zacházet s myší, piktogramy, menu a okny. Za druhé mohou programy běžící pod GEOSem jako Geo Print a Geo Write mezi sebou bez problémů vyměňovat data (pomocí podprogramů Textmanager a Fotomanager). Tím znáte to nejdůležitější pro to, abyste mohli okamžitě

pracovat s každým dalším budoucím uživatelským programem pod GEOSem.

Desktop - uspořádaný psací stůl.

Desktop se dá přirovnat k velínu z něhož je možno řídit všechny důležité funkce GEOSu. Ve středním okně jsou markantními piktogramy vyznačeny názvy programů aktuální diskety. Návěští >>LOAD "$"<< při používání GEOS můžete klidně zapomenout. Okno ukazuje najednou maximálně 8 piktogramů. K dalšímu prolistování directory musíte aktivovat ohnutý levý dolní rožek. Chcete-li se naopak vrátit o stránku zpět je zapotřebí aktivovat volný levý dolní roh. Na horním okraji je umístěna lišta s příkazy. Obsahuje názvy jednotlivých rolovacích menu v nichž je možno nalézt jednotlivé příkazy menu.

V horním pravém rohu je piktogram diskety s názvem právě vložené diskety, v tomto případě GEOS V 1.3. Tento piktogram je zobrazen černě. Znamená to, že disketa je otevřena. Okno podá i další důležité informace. Název diskety najdete v titulku ve středu nahoře. Pod ním se lze dozvědět počet datových souborů a volnou a obsazenou kapacitu diskety. Čtvereček vpravo nahoře slouží k deaktivaci (uzavření) diskety. Mimo leží další dva piktogramy. Tiskárna a odpadkový koš. Pokud odložíte datové soubory na tato místa budou vytištěny eventuelně vymazány.

Aplikační programy.

Textové editory běžící pod GEOSem zastupují programy Geo Write. Umožňují tvorbu jak krátkých dopisů, tak bohatých zpráv. Geo Write plní všechny funkce, které jsou od textových editorů očekávány (opět jsou podpořeny řadou menu v horní části obrazovky): pohyb kursoru, značení celých stran, odstavců nebo jednotlivých slov, použití různých typů písma, formátovaný text (podle levého okraje, pravého okraje, centrovaný) a výtisk textu.

Pomocným programem "Pait Drivers" se dá provozovat DTP. K tomu se musí napřed instalovat příslušný budící program pro tiskárnu.

Pro ty, kteří jsou držiteli C64 delší dobu a zpracovávali texty jinými editory je program "Text Grabber" schopen transformace dat do formátu Geo Write.

Programy pro kreslení - Geo Paint.

Ani ti, kteří mají grafiku jako své hobby nepřijdou s GEOSem zkrátka. Geo Paint je program pro kreslení 16-ti barvami. Zná nejen kružnici, elipsy, čáry, obdélníky a funkce sprej, ale i připojení libovolného textu do grafické obrazovky 320*200 pixelů.

Je jasné, že co se množství dat týká, C64 jich do své malé paměti uloží necelou stránku formátu A4. Stránku formátu A4 rozděluje Geo Paint na jednotlivé výřezy obrazu, které se mohou při pozdějším tisku automaticky spojit. Z celkového obrazu se dají vyříznout malé obrázky a uložit na disketu ("Fotoalbum"). Tyto malé obrázky se dají přilepit na různá místa jiných obrazů v Geo Paintu nebo Geo Write.

Další software pro GEOS.

Od dob vzniku po dnešek byla vytvořena řada aplikačních programů a utilit, které usnadňují práci se systémem. Ke zpracování privátních nebo služebních adres a podobných dat slouží programy Geo-File a Geo Dex. Mohou sloužit k vytvoření přehledných kartoték zákazníků, videofilmů či podobných dat. Geo Publish umožňuje sestavovat a tisknout školní noviny či reklamní letáky. Použitím International Fontpack a Mega-Pack 1 a 2 dostanete do ruky velké množství znakových sad již mohou být samozřejmě použity pro Geo Write a Geo Paint. Ten kdo chce tvořit vlastní programy pod GEOSem měl by se zabývat programovým paketem Geo - Basic a Geoprogrammer. Téměř každé přání milovníků dálkového přenosu dat vyplní Geoterm.

Velké plus produktů GEOS neleží jen v tom, že vypadají velmi podobně, ale běží stejným principem. Tím je zaručena velká kompatibilita různých aplikací. Pokrokem je, že všechny funkce a příkazy jsou vyvolatelné z menu a icon. Uživatelé, kteří se v počítačích celkem nevyznají mohou lehce a bez stresů ovládat jednotlivé programy.

Podobným vývojem neprošly jen domácí počítače, ale i PC (Windows).

(pokračování)

Tiskárna D100M

V minulých měsících nabízela firma Commotronic polské tiskárny **D-100M** za velice výhodnou cenu 2 890,- Kčs. Pro ty, kteří tiskárnu neznají, uvádíme její základní technická data:

Počet jehel :	9
Rychlost tisku :	80 zn./sek.
	12 zn./sek. NLQ
počet zn. na ř. :	max. 130
download + bafr :	1.0 kB
šířka papíru :	max. 250 mm
druh papíru :	listový, perforovaný, rolovaný
max. počet kopií:	3
typ písma :	úzký, normální, široký, tučný
barvicí páska :	normální, do psacího stroje
grafický mód :	max. 1920 bodů na řádek
interfejs :	Centronics
emulace :	EPSON FX 80

Tiskárna D-100M byla testována úspěšně s FC III, kde tiskne v grafickém i textovém módu. S FC II a DTP programem PRINTFOX spolupracuje též bez problémů. S použitím Centronics Spooleru je použitelná pro výpisy listingů, textové editory v Basicu, Wizawrite a grafické hardcopy.

Výhodná cena byla výprodejová a tiskárny byly dodávány pouze se 14ti denní zárukou.
V současné době jsou již tyto tiskárny vyprodány.
Přesto se podařilo zajistit další dodávky uvedených tiskáren, v tomto případě s roční zárukou. **Cena** je však díky záruce vyšší a činí **4090,- Kčs.**

PROMĚNY HARDWARE C64

Od roku 1982 bylo sestaveno a prodáno několik milionů exemplářů C64. Vlastně vše začalo už v roce 1978. PET 2001 (Personal Elektronic Transactor) začal dobývat trh a spustil tím boom v oblasti domácích počítačů. Tento přístroj byl takřkajíc dědečkem C64. Byl koncipován jako stolní přístroj. Na desce stolu zabral tudíž místo jako dnešní PC. Vybavení bylo ovšem podstatně chudší. S klávesnicí připomínající hračku a vestavěným kazetovým záznamníkem byl špičkou tehdejších domácích počítačů. Měl osazeno 8 pamětí a implementovaný Basic, který se příliš nelišil od dnešního Basicu 2.0.

Kazetovoý záznamník je dnes u mnoha uživatelů C64 nahrazen datasetem. Brzy po zavedení byl název PET změněn na CBM 2001 (Commodore Business Machine). Krátce nato přišel na trh další model CBM 3008. Současně se serií 3000 se objevil také tatínek C64: počítač VIC 20. Se svou 3,5 kB pamětí RAM a tehdy extrémně nízkou prodejní cenou (v Německu "pouhých" 800 DM) se postaral o čerstvý vítr na tehdy ještě mladé počítačové scéně. Během krátké doby bylo v Německu prodáno několik set tisíc přístrojů. V roce 1982 vstoupil na scénu C64. Tenkrát byl senzací. S pamětí 64kB a svými fantastickými grafickými možnostmi zastínil vše co dosud zaplnilo obchody s počítači. Možnosti a paměť vysvětlují dnes už těžko pochopitelnou cenu 1400 DM. Deska hustě osazená integrovanými obvody zcela vyplňovala dole pouzdro počítače. Nahoře bylo ještě dostatek místa pro rozšíření. Člověk si mohl pájet co srdce ráčí, aniž by musel mít strach, že se jeho hardwarové rozšíření nepodaří umístit dovnitř. Až do roku 1986 byl C64 vyráběn a dodáván s nezměněnou elektronikou. Jako zvláštnost vytvořil Commodore přenositelný C64 pod typovým označením S64. Byl produkován pouze v období 1983 až 1986. Malý barevný monitor a floppy 1541 byly s deskou C64 smontovány v jednom pouzdru.

V roce 1986 dostal počítač nový desing. C64 se mohl prezentovat plochým pouzdrem. Bohužel s touto novou formou se ztratila spousta místa nad deskou. Stavba vlastních doplňků dovnitř počítače se stala téměř nemožná. Protože se však nezměnila elektronika, byl nový model zcela kompatibilní se starou verzí.

V dalším roce zmizel tento tvar z regálů obchodů a Commodore se prezentoval starou formou. Při odšroubování víka byla hned zřejmá zcela nová deska. Co se jejich rozměrů týče, byla o třetinu menší oproti staré. Commodore silně zredukoval i počet integrovaných obvodů. Bylo to umožněno sloučením logických funkcí. Multiplexní logika k obsluze paměti potřebuje spoustu TTL obvodů. U této verze byly téměř všechny tyto obvody sloučeny do jednoho speciálního.

Současně se objevila i verze, kdy ve starém obalu byla osazena nová deska. Na user-portu tohoto počítače ovšem chybělo střídavé napětí 9 V. Proto bylo nemožné provozovat na tomto portu např. epromer.

V roce 1991 přišla na trh nepozorovaně další verze C64. Objevila se opět v plochém pouzdře. Zvenku se počítač nelišil od již zavedeného. Deska uvnitř doznala opět změny.

Celkem existuje pět rozdílných verzí C64. Během času byly jednotlivé verze desek neustále nepatrně měněny. Tak změnil modulátor i u jedné serie svou velikost. Jedenkrát byl s přepínačem kanalů, jedenkrát bez něj. popřípadě byl opatřen zvenčí přístupným trimrem pro naladění televizního kanálu. Také VIC byl opatřen nejrůznějšími periferiemi. Jeho okolí se měnilo častěji, nikdy však nevznikly vážnější softwarové problémy.

Verze 1

Na desce tištěných spojů tohoto počítače byla řada logických funkcí realizována TTL obvody. Několik blokovacích kondenzátorů se staralo o čisté napájení jednotlivých obvodů.

Podívejme se na hardware důkladněji.

Nahoře vlevo se nacházejí oba CIA (Complexe Interface Adapter). Jejich úlohou je řízení přenosu dat z C64 do vnějšího světa. Ihned vedle je umístěna pevná paměť počítače. Je tvořena třemi obvody ROM (Read Only Memory). První obsahuje Basic-Interpreter, druhý pro-

vozní systém a konečně znakovou sadu C64. Vedle je mikroprocesor. Pod těmito obvody se nachází operační paměť počítače. Tenkrát právě tyto obvody byly velmi drahé, odráželo se to v celkové ceně počítače. Jeden obvod může mít uloženu jednobitovou informaci, ale 65 536*. Protože náš počítač pracuje s šířkou dat 8 bitů, bylo potřeba nárokům procesoru vyhovět osmi těmito malými obvody. Vedle procesoru je umístěn obvod PLA (Programmable Logic Array). Pomáhá správně obsluhovat paměť. Bez tohoto obvodu by byla deska osazena navíc řadou TTL obvodů. O dobrý zvuk se stará SID (Sound Interface Device). Vedle komplexního syntetizéru přechovává ještě převodník A/D. Přes něj může být C64 ovládán regulátory typu paddle.

Nad SIDem je konektor seriového portu a videovýstupu, v odstíněném pouzdru se nachází modulátor. V principu nejde o nic jiného, než o televizní vysílač s malým výkonem. Přes antenní zdířku spojuje C64 s televizním přístrojem. Přesně pod modulátorem opět v odstíněném krytu je umístěn videočip. O správné přiřazení barev se stará vlastní RAM. Protože VIC dává jen 16 barev postačuje k jeho obsluze čtyřbitová paměť.

Vpravo nahoře vidíte jistě expanzní port a pod ním vnitřní zdroj C64. Starší obvody potřebovaly ještě k bezproblémovému provozu pomocné napětí 12 volt. To je zde vyráběno ze střídavého napětí 9V.

Verze 2

SX64 obsahuje přibližně shodné obvody jako stará deska C64. Zkoušelo se s pokud možno malými změnami hardware nalisovat přístroj do menšího pouzdra. Přece však došlo k několika mechanickým změnám. User-port byl pomocí úhlové destičky přeložen nahoru. Na konektorech, které spojují jednotlivé desky přenosného počítače, Commodore velmi ušetřil. Neustálými otřesy při přenášení se spoje uvolňovaly a nic nefungovalo. Muselo se odšroubovat víko, na spoje se muselo silně zatlačit a přístroj opět fungoval.

Verze 3

Deska je osazena mnoha novými obvody. Commodore nahradil skoro všechny TTL obvody. Okamžitě rozeznatelný speciální obvod s mnoha vývody převzal jejich funkce, především multiplex s obsluhou paměti. Také ostatní funkce byly rozděleny do několika obvodů. Paměť tvoří pouze dva obvody s organizací 64k*4. Toto řešení bylo cenově příznivější, než osazení 8 švábů. Charakter ROM (znaková sada) zůstala, zatímco basic a provozní systém byly sloučeny do 16 kB ROM. Procesor 6510 byl nahrazen typem 8500. Oba obvody CIA zůstaly, ale změnila se jejich poloha na desce. Ten, který obsluhoval user-port zůstal na svém místě, zatímco druhý, pro řízení klávesnice a joysticku se přesunul napravo dolů.

Na desce nahoře vlevo se nachází CIA zodpovědná za chod user-portu. Bezprostředně vpravo od něj je umístěna znaková ROM. Vedle potom 16 kB ROM basic + provozní systém. Pod mikroprocesorem jsou osazeny oba paměťové obvody. Následuje velký 64 vývodový speciální obvod MMU (Memory Management Unit). Commodore se u této verze zřekl veškerého stínění u VIC. Menší potřeba proudu tohoto nového obvodu dovolila též jeho provoz bez chladiče. Ihned vedle je oscilátor, následuje SID. Nad ním se nachází modulátor ve zmenšeném stínícím krytu. Vedle SID je zasazen druhý obvod CIA (klávesnice, joystick). Zbytek místa na desce zabral vnitřní zdroj.

Přes tyto zásadní změny v hardware zůstal nový C64 velmi softwarově kompatibilní k svému předchůdci. Jen některé programy vypověděly na této verzi svou službu. S hardwarovou přenositelností to bylo úplně jinak. Řada hardwarových přístavků nefunguje.

Hlubší zásahy byly provedeny i u nových obvodů. Technická výhoda: SID pracuje pouze s jedním napájecím napětím. Nevýhodná je naopak změna filtračních kondenzátorů. Na starém C64 perfektně propracovaný zvuk přinesl na novém přístroji jen unavené skřípění.

Zmíněné kondenzátory podstatným způsobem ovlivňují zvuk. Proto musí být kondenzátory buďto změněny nebo se musí programově nastavit jiné parametry.

Verze 4

Ke cti přišlo staré pouzdro. Deska tištěných spojů odpovídá předchozímu modelu. Na user-port se opět vrátilo 9V střídavého napětí. Hardwarová rozšíření, která toto napětí potřebují, opět mohla fungovat.

Verze 5

Tato dosud poslední verze dostala opět ploché pouzdro. Pokud však člověk vezme lupu zjistí, že chybí jeden obvod. Paměť barev byla integrována do multifunkčního obvodu. Mimo to jsou vpravo na joystickovém portu nápadné malé žluté prvky, které chrání hodnotné obvody CIA před zničením. Svou úlohu plní téměř dokonale. CIA vydrží teď podstatně déle, ovšem občas dojde ke zničení ochranného prvku. Prvky jsou tvořeny spojením odporu a kondenzátoru. Kondenzátory mají nepříjemnou vlastnost, že se prorážejí. Tím se stáhne vstup k zemi. Jinak počítač nebyl měněn. Commodore díky vysoké integraci ušetřil další obvody. Po chybějící RAM barev působí deska se zbylými obvody velmi prázdně. Její další zmenšování ovšem nemá smysl.

Pohled do budoucna.

Mezi tím uběhlo pár let a C64 je stále ještě softwarově kompatibilní se svými předchůdci. Největší problém

dalšího vývoje C64 leží v jeho softwarové kompatibilitě. Mezitím bylo vytvořeno pro náš počítač přes 100 000 programů. Každý hlubší zásah do hardwaru by odsoudil tyto programy do odpadkových košů.

Změny tohoto systému se nedají lehce provést. Protože však integrace postupuje stále dále k hustějším obvodům, třeba nás jednoho dne Commodore překvapí.

TIPY A TRIKY

Tuto rubriku bychom v budoucnu a to co nejbližším rádi věnovali Vašim vlastním nápadům z programátorské praxe. Věříme, že budou nejméně stejně zajímavé jako dále uvedené, získané z nejrůznějších literárních zdrojů.

Zatím nejjednodušší rutina MERGE

Pro spojování dvou programů v Basicu existuje řada rutin. Dále uvedená rutina představuje zatím nejjednodušší nám známé řešení. Postup při spojování dvou programů je následující:

Nejprve se do počítače natáhne 1. program a aniž se spustí, zadá se v přímém módu příkazový řádek:

```
POKE 43, PEEK (45)-2: POKE 44, PEEK (46)
```

a uzavře se stiskem klávesy RETURN.

Poté se natáhne druhý program (pozor na to, aby měl čísla řádků vyšší než první program!) a zase se v přímém módu zapíše příkazový řádek:

```
POKE 43,1: POKE 44,8
```

Tím je vše skončeno a program složený ze dvou původních je možno editovat nebo spouštět.

Natažení programu z programu

Mnoha uživatelům by ulehčilo práci, kdyby mohli z jednoho programu natáhnout další, který by původní program přepsal. I když Basic V2.0 tuto možnost přímo nenabízí, existuje dále popsané řešení. V uvedeném podprogramu v Basicu je nutné pouze za proměnnou NP$ dosadit název programu, který chcete do počítače natáhnout:

```
10 POKE 828,0

20 PRINT CHR$(147)CHR$(17)

30 PRINT "LOAD"CHR$(34);NP$;CHR$(34);",8"

40 PRINT CHR$(17) CHR$(17) CHR$(17) CHR$(17)
"RUN"

50 POKE 631,13: POKE 632, 13: POKE 198,2

60 PRINT CHR$(19);:END
```

Zkušenější programátoři jistě poznají, že podprogram pracuje v simulovaném přímém módu s využitím klávesnicového bafru.

Tento podprogram je určen pro natahování programů z diskety. Pro kazetu bude nutno jen na řádku 30 změnit označení periferie z 8 na 1. Ale pozor! V TURBU to nepůjde!

S pomocí tohoto podprogramu není problém rozdělit velký program na více částí, které se do paměti natahují dle potřeby. Tím je tento podprogram zvláště užitečný pro modulární programátory. Zbývá ještě upozornit, že natahování programu z programu může mít při práci s daty další zádrhele, které blíže vysvětluje Basic koutek v tomto čísle.

Datassette a řízení

Normálně přebírá po zadání příkazu LOAD nebo SAVE řízení motoru magnetofonu přímo procesor počítače, který je pro tuto práci speciálně přizpůsoben. To ale neznamená, že by jej nebylo možno řídit samostatně z programu, pokud to potřebujeme. Práci s Datassettem Vám mohou v mnoha případech ulehčit následující příkazy:

```
POKE 192,2: POKE 1,55
```

způsobí, že motor záznamníku bude při převíjení pásky vypnut.

```
POKE 192,1: POKE 1,23
```

způsobí opětovné zapnutí motoru.

```
WAIT 1,16
```

způsobí čekání záznamníku na stisk klávesy STOP.

```
100 IF PEEK (1) = 55 THEN 100
```

čeká na stisk klávesy PLAY, nebo jedné z převíjecích kláves.

```
PRINT PEEK(150)
```

vypíše FLAG (návěští) záznamníku. Při nule (0) je motor vypnut, při jedné (1) je motor zapnut.

Problémy s ochranou proti zápisu

Kdo používá programy, které spolupracují s disketovou jednotkou nebo tiskárnou, si jistě vzpomene na řadu případů, kdy díky nalepené ochraně proti zápisu přišel o data v počítači, protože došlo k přerušení programu díky výpisu chybového hlášení nebo "device not present error". Část problémů v tomto případě řeší dále uvedený podprogram, určený pro práci s disketovou jednotkou.

```
10  OPEN 1,8,15,"M_R" + CHR$(0) + chr$(28):
GET#1,A$: CLOSE1

20 A = ASC(A$+CHR$(0))AND16: IF A<>0 THEN 100

30  PRINT"ODSTRANTE PROSIM OCHRANU PROTI
ZAPISU!":

   POKE 198,0:WAIT 198,1:GOTO 10

100 REM ***** POKRACOVANI PROGRAMU *****
```

Tato rutina přiblíží Váš program profesionálnímu provedení.

SAVE s komentářem

Při ukládání programů na disketu se dají k názvu programu připojit poznámky, které není nutno při natahování programu (zápisu názvu) uvádět, které však jsou vypsány ve vylistovaném adresáři (directory).

Je zde samozřejmě omezení, dané celkovou délkou názvu 16 znaků, do kterých se musí vlézt název i poznámka, ale vzhledem k jednoduchosti zápisu stojí tato pomůcka za to.

Při ukládání programu zapíšeme:

```
SAVE "NÁZEV PROGRAMU<SHIFT + SPACE>POZNÁMKA",8
```

a řádek uzavřeme stiskem RETURN. Po uložení programu si na obrazovku vylistujeme adresář diskety a zjistíme, že v adresáři je název programu uzavřen uvozovkami a poznámka je až za nimi, to znamená, že ji nemusíme při natahování programu uvádět. Maličkost, ale potěší, co říkáte?

Vypínání obrazovky

Kdo měl někdy příležitost pracovat s počítačem PC, jistě zatoužil mít na svém Commodore 64 řadu funkcí, kterými se PC honosí. Mezi ty opravdu užitečné patří na PC funkce, která v případě, že obsluha odejde od počítače a nepoužívá klávesnici, vypne obrazovku, aby se zbytečně nevypalovala. Opětovný stisk libovolné klávesy pak způsobí rozsvícení obrazovky a je možno pokračovat v přerušené práci.

Dále uvedený podprogram v Basicu má shodnopu funkci. Navíc má výhodu, že není jako obvykle psán ve strojovém kódu, ale v Basicu, což rozšiřuje možnost jeho využití programátory v Basicu.

```
10 REM *** HLAVNI PROGRAM ***

20 C = 15: T = T/60: PRINT"CLR"

30 GET A$: GOSUB 100: REM DOTAZ NA KLAVESNICI
+ VOLANI PODPROGR.

40 PRINT A$: GOTO 30: REM OVERENI DOTAZU

100 REM PODPROGRAM VYPNUTI OBRAZOVKY

110 IF A$ <> "" THEN T = TI/60:RETURN

120 IF T+C > TI/60 THEN RETURN

130 BA=PEEK(53280): POKE 53265, PEEK(53265) AND
239: POKE 53280,0

140 GET AU$: IF AU$="" THEN 140

150 POKE 53265,PEEK(53265)OR16: POKE 53280,BA:
T=TI/60: RETURN
```

Proměnná C v programu znamená čas v sekundách, po kterém se obrazovka vypíná. Jak vyplývá z listingu, je po příkazu GET A$ v hlavním programu volán podprogram pro vypínání obrazovky. Pokud nikdo nestiskne klávesu, zůstane program v podprogramu a čeká na řádku 140 na stisk klávesy. Teprve po jejím stisku se program vrátí do hlavního programu. Vlastní podprogram je na řádcích 100 - 150. Řádky 10 - 40 slouží jen pro demonstrační účely.

DEVICE NOT PRESENT

Ukončení běhu programu, do kterého jsem předtím hodinu cpali data výše uvedeným hlášením je infarktovou záležitostí. Zažil jej určitě každý, kdo má za sebou nějaký ten týden práce s počítačem. Je způsobeno tím, že jsme zapomněli zapnout tiskárnu nebo disketovou jednotku a patří mezi nejméně příjemné programátorské zážitky. Přitom ochrana proti takovému přerušení

není nic složitého a vyplatí se zabudovat ji do každého programu, který s disketovou jednotkou nebo tiskárnou spolupracuje. Je jen zapotřebí zabudovat do programu následující řádky:

```
10 OPEN 2,8,2
```

Tento příkaz způsobí otevření datového kanálu k disketové jednotce. Pokud chcete testovat kanál k tiskárně, změňte programový řádek na:

```
10 OPEN 2,4,7
```

Tento datový kanál je pak nutno opět rychle uzavřít, takže nevznikne důvod k chybovému hlášení. K tomu účelu zapište ještě jeden programový řádek:

```
20 CLOSE 2
```

Až potud je vše jasné. Kanál jsme otevřeli a zase zavřeli, ale nevíme, zda byla periferie na datovou sběrnici připojena. K tomu účelu se musíme počítače dotázat na status periferie ST. Pokud je periferie připojena, má systémová proměnná ST, obsahující hlášení o statusu periferie hodnotu 0. V případě, že připojena není, nebo je na ní jiná závada, je různá od nuly. Další řádky programu slouží již jen k dotazu na status a reakci na výsledek. Tento podprogram by ve vašich programech neměl chybět. Opravdu je velice amatérské, když se běh programu přeruší jen proto, že jsme opomněli zapnout disketovou jednotku nebo tiskárnu.

```
30 IF ST = 0 THEN 70

40 PRINT "ZAPNENTE PERIFERII A STISKNETE KLAVESU!"

50 GET A$: IF A$ ="" THEN 50

60 GOTO 10

70 REM *** POKRACOVANI PROGRAMU ***
```

Několik stále aktuálních rad pro programátory

Využívání systémových rutin zefektivňuje nejen práci programátora v assembleru, ale i v Basicu. To, co v Basicu vyžaduje několik programových řádků, se dá mnohdy vyřešit jedním SYSem:

SYS 66409 - nová iniciace VICu

SYS 59626 - skroluje obsah obrazovky o jeden řádek nahoru.

SYS 64767 - studený start Basicu

SYS 58592 - čeká 8,5 sekund na stisk C= klávesy (využívá se při natahování programu z Datassettu)

POKE 214,X

POKE 211,Y

SYS 58732 - nastaví kurzor na řádek X a sloupec Y

POKE 781,X

SYS 59903 - vymaže řádek X z obrazovky

POKE 808,230 - vypne klávesu RUN-STOP

POKE 649,0 - zablokuje klávesnici, takže nereaguje na žádnou klávesu

POKE 649,10 - odblokuje klávesnici

Komfortní INPUT

Otazník, který označuje v Basic - programech příkaz INPUT nepůsobí vždy dobře a každý z nás mnohdy zatoužil po jeho náhradě jiným znakem, nebo jeho odstranění. Je více možností, jak tento problém řešit. Jednou z nich je například tato:

```
10 OPEN 1,0: INPUT#1,A$: CLOSE 1
```

nebo:

```
10 POKE 19,0.
```

Po návěští INPUT však musí být tato změna ihned odstaněna příkazem POKE 19,0.

Tato dvě řešení odstraňují otazník z obrazovky. Ale občas je nutné použít jiný znak než otazník. Tento problém se dá řešit takto:

```
10 POKE 631,20: POKE 632,20: POKE 633, ASC(žádaný znak):

20 POKE 634,32: POKE 198,4
```

Protože nastavená změna "vydrží" jen pro jedno provedení návěští INPUT, je vhodné udělat z těchto dvou řádků podprogram.

Všechno je strukturované!

Basic opravdu není jazyk, pro který by se s nadšením rozhodovali příznivci strukturovaného programování. Ale co zbývá začátečníkovi na C64 jiného, než Basic V2.0? Pro tyto případy je určen následující trik, který umožní tvořit Basic-programy alespoň trošku strukturované. Následující programový řádek nevypadá přeh-

ledně:

```
10 FOR I=1 TO 20:FOR J=1 TO I: A=4: GOSUB 400:NEXT J,I
```

Syntakticky mu nelze nic vytknout, ale vyznat se v něm vyžaduje zapnout mozkové závity. Rozhodně bude vypadat lépe, když místo tohoto řádku zapíšete následující:

```
10 FOR I = 1 TO 20
20 FOR J = 1 TO I
30 A = 4
40 GOSUB 400
50 NEXT J
60 NEXT I
```

Každému příkazu je přidělen jeden řádek. Bylo by fajn, kdyby ještě bylo možno smyčky i vzhledově vnořit do sebe, aby co nejvíce znázorňovaly skutečnost. Pak program vypadá následovně:

```
10 FOR I = 1 TO 20 20 : FOR J = 1 TO I
30 :   A = 4
40 :   GOSUB 400
50 : NEXT J
60 NEXT I
```

Jistě uznáte, že tento zápis je nebe a dudy ve srovnání s původním jednořádkovým zápisem. Ale ty dvojtečky ještě nepředstavují ideální řešení. Proto Vám na závěr nabízíme skutečně profesionální vzhled listingu s využitím následujícího triku.

Obrazovkový editor při zápisu program ignoruje mezery, ponechané za číslem řádku a standartně nastavuje za číslem řádku jednu mezeru. Aby se při zápisu programu choval rozumněji, použijeme proti němu malý trik. Programový řádek zapíšeme tímto způsobem:

```
20 <SHIFT/C><tři mezery>FOR J = 1 TO I <RETURN>.
```

Asi si řeknete "no, tenhle grafický znak je horší, než dvojtečka". Ale zkuste si řádek znovu vylistovat a budete spokojenější. Tímto postupem jsme přesvědčili editor, aby při zápisu neignoroval mezery a díky tomu pak zápis vypadá tak, jak jsme jej chtěli od začátku mít:

```
10 FOR I = 1 TO 20 20   FOR J = 1 TO I
30      A = 4
40      GOSUB 400
50   NEXT J
60 NEXT I
```

Takže, programátoři, jste již se zápisem spokojeni?

Hexadecimální na dekadické a zpět

Je mnoho důvodů, pro které je vhodné mít k dispozici programový přepočet mezi hexadecimální a dekadickou soustavou. K tomu účelu jsou určeny dvě následující rutiny v Basicu:

```
10 HX$ = ""
20 INPUT"dekadické číslo";D
30 H$ = "0123456789ABCDEF"
40 T = 12 : FOR I = 1 TO 4
50 DC = D/(2^T)
60 D = D - INT (DC) * (2^T)
70 HX$ = HX$ + MID$(H$,DC+1,1)
80 PRINT HX$
90 GOTO 10
```

Další rutina slouží k obrácenému převodu mezi hexadecimálním a dekadickým číslem:

```
10 INPUT"hexadecimální číslo";H$
20 D = 0: FOR I = 1 TO LEN(H$)
30 DT = ASC(MID$(H$,I,1)
40 D = 16 * D + DT + 48 + 7 * (DT>64)
50 NEXT
60 PRINT D
70 GOTO 10
```

Autostart programu

Uživatelé kazetového záznamníku Datassette to mají alespoň v něčem pohodlnější, než uživatelé disketové jednotky. Pokud chtějí, aby se program po natažení z Datassettu ihned spustil (autostart), stačí, když stiskou klávesy <SHIFT/RUN-STOP> a spustí magnetofon stiskem klávesy PLAY. Program se po natažení do počítače automaticky spustí. Tento trik je použitelný i při natahování programů z diskety. Je však nutno dodržet tento zápis:

LOAD:název programu",8,1 <SHIFT/RUN-STOP>.

Takto natažený program z diskety se také ihned po natažení spustí.

Kopírování ROM do RAM

Následující rutina umožňuje snadno a rychle zkopírovat ROM do RAM:

```
START   SEI
        LDX #4
LOOP    STY $57,X
        DEX
        BNE LOOP
        JSR $A3E8
        JMP $FDDD
```

Umístění rutiny na libovolné místo paměti je vzhledem k její relokovatelnosti jednoduché a nebude jistě ani začínajícím programátorům v assembleru činit potíže.

OK na obrazovce

Pokud potřebujeme v našem programu hlášeni OK, můžeme je vyvolat jednoduše příkazem

SYS 63529.

Přerušení na příkaz

Pokud do programu zabudujeme příkaz JSR $A82C, přeruší program práci po stisku klávesy <RUN-STOP>.

PRESS PLAY ON TAPE

Po zadání příkazu SYS 63 **544 (JSR $F838)** se na obrazovce objeví text

PRESS PLAY ON TAPE a počítač čeká na stisk uvedené klávesy.

Žádný RESET

Pokud potřebujeme zprogramu provést RESET počítače, použijeme příkaz SYS 64738. Spustíte-li ale dále uvedený krátký program, bude RESET zablokován, jak to známe u profesionálních programů:

```
100 FOR I = 32770 TO 32 778
110 READ A: POKE I,A
120 NEXT I
130 DATA 10,128,195,194,205,56,48,88,0
```

Žádný LISTING

Jeden z postupů, jak zabránit výpisu programů na obrazovku je tento: POKE 775 hodnota. Všechny hodnoty s výjimkou 167 zamezí vylistování programu.

Žádný SAVE

Po zadání příkazového řádku

POKE 801,0: POKE 802,0: POKE 818, 165

nejde nadále žádný program pomocí příkazu SAVE uložit. Stejného výsledku dosáhneme také s pomocí příkazových řádků:

POKE 818, 116: POKE 819, 196 nebo

POKE 918, 34 : POKE 819, 253

(pokračování)

Autorské příspěvky

Redakce FUNu přijímá příspěvky čtenářů do všech rubrik časopisu. Příspěvky do rubriky Dopisy čtenářů nejsou honorovány.

Příspěvky do ostatních rubrik jsou honorovány částkou 200,- Kčs za uveřejněnou tiskovou stánku.

Nevyžádané rukopisy se nevracejí.

Příspěvky do časopisu označujte v rohu obálky nápisem FUN.

Nezapomeňte uvést jméno, celou adresu a číslo OP kvůli honoráři!

Inzerce ve FUNu

Osobní inzerce

Inzertní stránky jsou děleny na 3 sloupce po 28 znacích. Za jeden řádek v jednom sloupci se platí 15,- Kčs. Inzeráty se platí složenkou, kterou zašle redakce FUNu žadateli o inzerát na základě propočtu z dodaného textu inzerátu. Nezaplacené inzeráty se neuveřejňují.

Podnikatelská inzerce

Cena za jeden inzertní řádek je 30,- Kčs. V případě plošného inzerátu je možno tisknout velikost 1/9, 1/6, 1/3, 1/2 a 1 stránka. Cena za plošnou inzerci je 5000,- Kčs za stránku.

Inzerce je zatím pouze jednobarevná. Může být otištěna buď dodaná předloha, nebo je inzerát sestaven redakcí na základě dodaného textu. Inzerát se platí fakturou.

Redakce si vyhrazuje právo rozhodovat o vhodnosti inzerátu pro otištění.

Nové ceny počítačů Amiga

S platností od 01.08.92 prodává firma Commotronic konečně počítače Amiga za cenu, kterou slibovala již v loňském roce, ale kterou se nakonec nepodařilo dodržet. Cena za Amigu 500 je rovných 14 000,- Kčs.

Součástí dodávky je i český překlad Amiga DOSu!

V příštím čísle najdete:

Assembler na C64 (pokračování)
Programování v Basicu (pokračování)
GEOS (pokračování)
Tipy a triky
Představujeme Vám: Turbo Cartridge 16kB
256 kB RAM disk
Ze světa C64
Různé
Dopisy čtenářů

Prodejna firmy Commodore

Jak již bylo uvedeno v úvodu, připravuje firma Commotronic otevření prodejny v novém působišti na Dolnomlýnské 2, Šumperk. Prodejna se sortimentem Commodore 64, Amiga a příslušenství bude mít do konce září prodejní dobu:

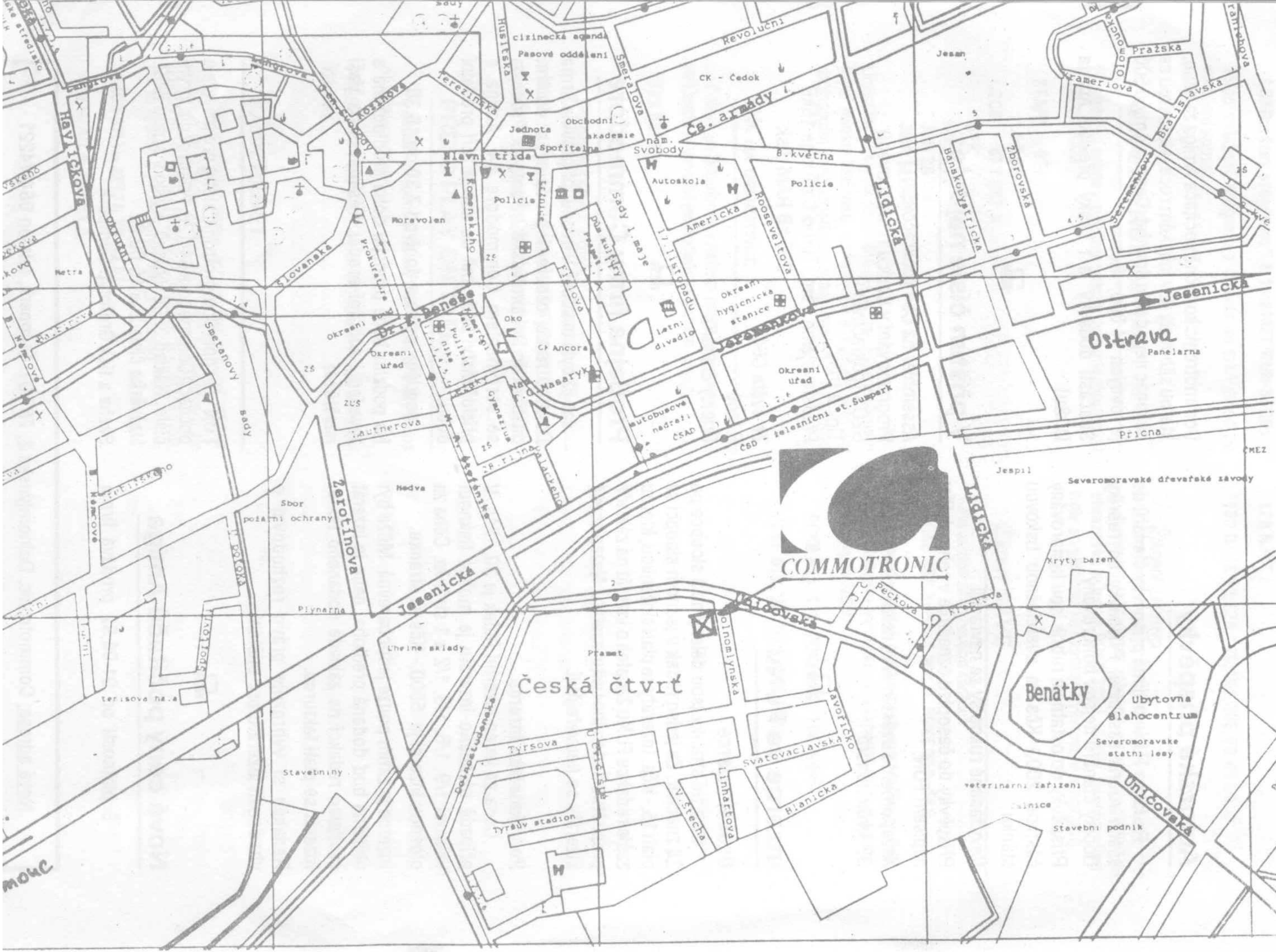
pondělí - pátek v době od 7.30 do 16.30

Na podzim bude prodejní doba prodloužena a zaveden i sobotní prodej. Těšíme se na Vaši návštěvu!

FUN - časopis Commotronic klubu pro uživatele počítačů Commodore 64/128. První ročník, první číslo. Náklad 3000 výtisků. Vyšlo v srpnu 1992. Uzávěrka čísla 2 je 15.09.92

Sazba a tisk Grafotyp Šumperk

Nová adresa: Commotronic, Dolnomlýnská 2, 787 01 Šumperk, telefon 0649/4221


COMMOTRONIC
Ostrava
Jesenická
Lidická
Česká čtvrť
Benátky
Uničovská
Revoluční
Čs. armády
8.května
Rooseveltova
Americká
17.listopadu
Sady 1.máje
Hlavní třída
Komenského
Dr.E.Beneše
Jeremenkova
Žerotínova
Smetanovy sady
Havlíčkova
Husitská
Terezínská
Smetalova
Zborovská
Banskobystrická
Jeremenkova
Bratislavská
Pražská
Olomoucká
Kramerova
Příčná
Jablonského
Pecková
Krapkova
Olnomlýnská
Javoříčko
Svatováclavská
Hlanická
Linhartova
V.Štecha
Tyršova
Tyršův stadion
U cvičiště
Dolnostudénská
Okresní úřad
Okresní hygienická stanice
Policie
Autoskola
CK - Čedok
Jesan
cizinecká agenda
Pasové oddělení
Jednota
Obchodní akademie
Spořitelna
nám. Svobody
Moravolen
Prokuratura
Slovanská
Okresní soud
Oko
CK Ancora
Nám. T.G.Masaryka
letní divadlo
Dům kultury
Gymnasium
Lautnerova
Medvа
Sbor požární ochrany
Plynárna
Uhelné sklady
Pramet
Stavebniny
Metra
ZUŠ
ZŠ
autobusové nádraží ČSAD
ČSD - železniční st.Šumperk
Panelárna
Jespil
Severomoravské dřevařské závody
ČMEZ
Krytý bazén
Ubytovna Blahocentrum
Severomoravské státní lesy
Veterinární zařízení
Stavební podnik
B. Němcové
Okružní