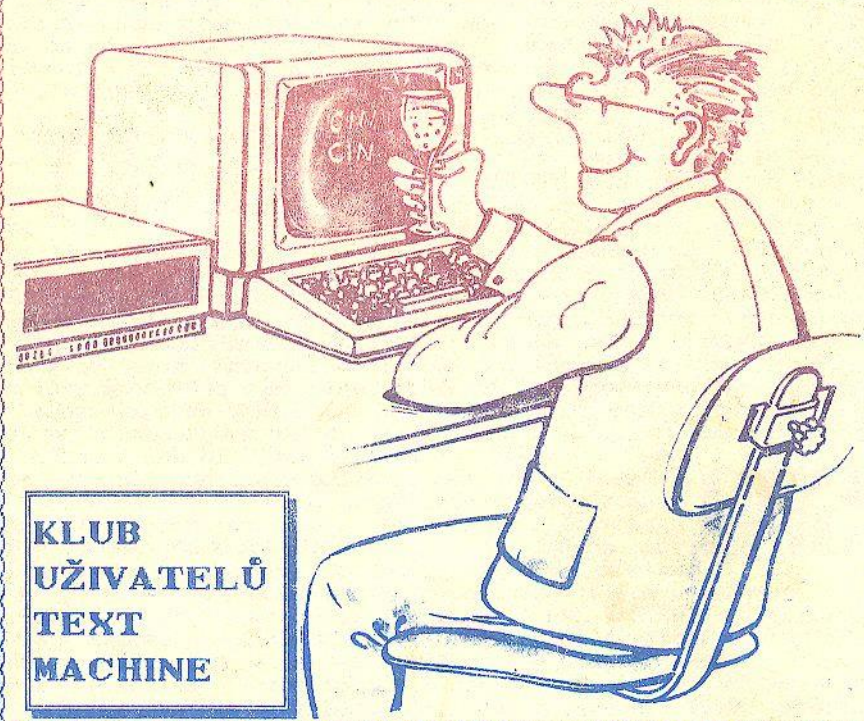
# AMATÉRSKÝ PROGRAMÁTOR

ČÍSLO 4 - ROČNÍK 4 - SRPEN 1993

## VYUŽITÍ NMI U DIDAKTIKU GAMA



KLUB
UŽIVATELŮ
TEXT
MACHINE

# Z REDAKCE

**Ahoj ápičkáři!**

Tak jsem tu opět se svým čtvrtým číslem Amatérského programátora, které i tentokrát vychází se značným zpožděním, **ale hlavně, že vůbec vyšlo!**

Důvody zpožděného vydání minulého čísla jsou Vám známy(bylo to uvedeno na obálce). Tentokrát je vina zcela na mé straně, ale není to proto, že bych snad neměl o čem psát (ba právě naopak, příspěvků je stále dost). Tentokrát jsem zase čekal až budu mít peníze na zakoupení papíru a na poštovné pro odeslání předplatitelům. Říkám Vám to zcela otevřeně a na nic se nevymlouvám.Chci jen,abyste znali skutečnou situaci,ve které se nacházím. Do konce roku bych měl vydat ještě dvě čísla. To se pochopitelně pokusím dodržet ať to dopadne jakkoliv. Jak to bude příští rok, však dosud nevím (?). Rád bych AP vydával zase každý měsíc, ale pouze o 28mi stránkách za cenu 14,-Kč. Kdo z Vás v něčem podnikáte, jistě víte, že naše (?) **česká vláda** na drobné podnikatele tak trochu (spíše dosti) s..., neboť si její členové musejí pořádně namastit především vlastní konta ve Švýcarech a jinde, no a nějaký malý český živnostníček jim může být ukradenej (ať si třeba pojde). Ale dosti již nářku nad smysluplností malého českého Absurdistánu a jejich slavné ptákostrany se všemi ptáky i neptáky...

## SPD

Pokud si posíláte diskety pro nahrání programů, **prosím Vás snažně, dobře si zabalte diskety, než je svěříte poště**! Často nám přicházejí vložené jen tak do obálky a dovedete si jistě představit v jakém jsou stavu. Na disketu, do které je vytlačeno razítko Vaší pošty, nebo je ohnutá, se dá už velice těžko nahrát i při sebevětším úsilí. Navíc nám ještě celou situaci zkomplikoval zase jeden z těch ptáků, který přišel na nápad zavést místo papírových bankovek kovové mince. Takže nyní již nemůžete k disketě přiložit příslušný poplatek, neboť kovová mince by mohla obálku protrhnout. Zbývá tedy jediné řešení; předplatit si i SPD. Na Vaši adres by pak bylo uvedeno, kolik ještě máte v naší minibance korun a nikoliv do kterého čísla máte předplaceno jak tomu je dosud. Podle toho, jak byste si **vybírali** programy by se Vám tato částka snižovala a snižovala... až by se Vám tam jednoho dne objevilo varování "KONTO JE VYČERPÁNO!".

## DOBÍRKY

již také nezasílám, neboť cena mnohdy značně převyšuje skutečnou cenu programu nebo publikace. Při odesílání zboží touto cestou, jsem nucen Vám připočítat poštovné, které se skládá z poplatku v průměrné výši asi 22,-Kč. Někdy se také stane, že si zásilku nevyzvednete ve stanovené lhůtě 14 dní a pošta pak zásilku vrací zpět.Já pak jsem nucen poště hradit dalších 12,-Kč za toto doručení.**Proto, když si budete něco objednávat z mé nabídky**, *uhraďte nejdříve danou částku složenkou typu "C" a na zadní stranu v kolonce "Zpráva pro příjemce" uveďte co si objednáváte.* Po obdržení Vaší platby Vám objednané zboží odešlu jako doporučenou zásilku na mé vlastní náklady.


# klub uživatelů TEXT MACHINE

## PSANÍ TEXTU

Po nahrání programu si vybereme vhodný psací font, kterých může být nejvíce čtyři. Vložíme disketu s fonty a z roletky **Style** vybereme funkci **Load**. Postupně si nahrajeme všechny fonty, které chceme v textu používat, volbou **Font 1** až **Font 4**.

V roletce **Misc.** si nastavíme stav **Video** tak, abychom na obrazovce viděli typ fontu, kterým píšeme. Je to stav, ve kterém na obrazovce vidíme pouze 32 znaků na řádek a při překročení se obrazovka začne posouvat směrem vlevo. Proč se o tom zmiňuji? Jestliže si **Video** nastavíme na 64 znaků na řádek, budeme mít sice přehled o celé šíři textu, ale nebudeme mít vizuelní kontrolu použitého fontu, tedy ani jeho **případné ztučnění** *nebo položení* a nebude vidět ani semigrafika. Z toho tedy plyne, že tento stav (64zn) nám bude sloužit pouze pro případnou vizuelní kontrolu textu. Jinak budeme při psaní používat stav **Video** při 32 zn. na řádek. Stav **Video** však můžeme měnit bez ohledu na to, která roletka je právě vyvolána, pouhým stiskem klávesy "V" (libovolnou roletku vyvoláme klávesou EXTEND).

Rovněž i změnu fontu můžeme provádět stiskem klávesy SYMBOL SHIFT a ‹ (F) kdy postupujeme k nižšímu číslu fontu, nebo SYMBOL SHIFT a › kdy postupujeme k vyššímu číslu fontu. Tento font byl v textu umístěn jako 4. Prvním stiskem SS+} se písmo ztuční, druhým stiskem SS+} *se font položí*, třetím stiskem SS+} *se položený font ztuční* a protože tento font je v pořadí 4. (tedy poslední) přejde se dalším stiskem SS+} do režimu, kde budou všechny fonty podtrženy. Postupně tedy stiskneme kl. SS+} až se dostaneme na font 4., který bude nyní podtržený. Dále se opakuje vše jako jsem uvedl výše, tzn. *podtržený font se nejprve ztuční, pak položí, položený se ztuční*. Jinak se toho dá také docílit z roletky **Style** volbou B-Bold, I-Italic, U-Underline.

## ZAROVNÁNÍ OKRAJŮ

Pokud budeme psát na celou šíři t.j. 64 znaků není třeba nastavovat levý a pravý okraj textu. Při zapnutém Wordwrap

bude text, který se již na řádek nevejde automaticky přetahován na další řádek. To však neznamená, že bude pravý okraj ihned i zarovnáván, jak tomu bývá u některých textových editorů. Není to však nedostatek, jak by se mohlo na první pohled zdát. Já osobně tento stav vítám, neboť vzhledem k tomu, že píšu dosti rychle, se často **překlepnu**, a v případě, že by byl již pravý okraj zarovnán, těžko bych již v textu prováděl opravy. Takto si mohu nejprve text znovu pročíst a teprve potom se rozhodnout pro zarovnání.

Nejprve postavím kurzor na začátek textu a vyvolám roletku **Block** ve které si zvolím funkci **9-Begin**, čímž označím začátek bloku. Pak nastavím kurzor na konec textu a znovu v roletce **9-Block** si zvolím funkci **0-End**. Tím si označím konec bloku. Pak znovu vyvolám **9-Block** a vyberu si příkaz **--Justify**, který provede zarovnání takto označeného bloku.

Zarovnání textu však nemusí být jen k pravému nebo levému okraji. Lze ho také tzv. **vycentrovat**, docílit efektu, který se používá převážně v reklamě.

Postup zde bude úplně stejný, jako jsem uvedl u zarovnání u pravého kraje. Pouze místo příkazu Justify zvolíme **?-Center**.

# DALŠÍ MOŽNOSTI PRÁCE S BLOKEM

Může se nám také stát, že se dodatečně rozhodneme u textu, který jsme původně psali od levého okraje a jehož šíře není na celou šíři, tedy 64 znaků na řádek, že po jeho levé straně budeme chtít umístit rámeček, nebo nějaký symbol a pod. Znamená to, že musíme pracně celý text přepsat? Nikoliv.

Opět si nastavíme začátek a konec bloku, eventuelně si text nejprve ještě i necháme zarovnat do okrajů, a pak se v roletce **Block** nastavíme na příkaz **+Left** pokud chceme blok odsunout vlevo nebo **+Right** pokud požadujeme posunutí bloku vpravo.

RIGHT

V každém případě to však lze použít skutečně jen tehdy, jestliže píšeme text např. do sloupců, či jiných užších útvarů než je celá šíře textu 64 zn. V opačném případě je snad jasné že se ani jedna z funkcí neprovede, neboť nebude místo kam by blok mohl být odsouván.

LEFT

Další funkcí, kterou lze s blokem provádět je jeho kopie na určité místo. Opět si nastavíme začátek a konec bloku a kurzor postavíme na místo, kam chceme blok zpokírovat. Tuto funkci jistě oceníme v případě, kdy chceme tisknout nějaký kratší text vícekrát a na jeden list papíru se také vejde vícekrát.

Tak např. tento krátký blok je kopírován čtyřikrát pod sebe.
Tak např. tento krátký blok je kopírován čtyřikrát pod sebe.
Tak např. tento krátký blok je kopírován čtyřikrát pod sebe.
Tak např. tento krátký blok je kopírován čtyřikrát pod sebe.

# DODATEČNÁ ZMĚNA PÍSMA

I po napsání celé stránky je možné velmi jednoduše změnit typ písma celé stránky, nebo části označené jako blok. Nejprve si nastavíme ten font

kterým chceme aby byl text vytištěn, pak si označíme začátek a konec bloku, a v roletce **Style** si zvolíme příkaz **Set style**. Celý blok textu bude přeměněn na zvolený font.

Zbývá nám už jen vysvětlit příkaz **Unjustify**, který zruší zarovnání textu. To se může hodit v případě, kdy máme již text zarovnán, ale rozhodneme se dodatečně pro nějaké změny. Po tomto příkazu se text opět vrátí do původní podoby než jsme provedli jeho zarovnání příkazem Justify.

Jestliže chceme blok textu označit jen v rámci jednoho odstavce, pak nastavíme kurzor pouze na jeho začátek a volbou příkazu **P-Paragraph** nám program sám najde jeho konec a ten bude také koncem bloku. Je snad jasné, že konec odstavce je o řádek výš, než řádek, kde začíná nový odstavec.

Poslední příkaz **Scratch** nám také značně urychlí práci s blokem, neboť po jeho volbě zruší označení jeho začátku a konce.

Na závěr se ještě zmíním o příkazu **/Reform**, který využijeme v případě, že text je původně napsán např. v šíři 64 znaků na řádek a potřebujeme ho **přepsat** na šíři třeba 60 znaků na řádek.

O příkazu **Delete** snad není třeba se nějak podrobně zmiňovat, úplně vymaže nastavený blok.

# RYCHLÁ ZMĚNA FONTU

Teprve, když je text již hotový můžeme usoudit, že některé jeho části by měly být podtrženy, ztučněny nebo položeny. Není třeba tyto části nebo slova zase znovu přepisovat. Již známým způsobem si nastavíme font nebo jeho podobu SS+J a pak si prohlížíme text. Ve chvíli, kdy nalezneme v textu slovo nebo větu, která má být např. tučně vytištěna (předem jsme si klávesou SS+J nastavili tučnost) stiskneme SS+D a **držíme je tak dlouho dokud kurzor nepřejede po celém slovu nebo větě, kterou chcem takto upravit.** Abychom nyní nemuseli pracně znovu volit původní nastavení fontu, nastavíme si kurzor na nějaký znak původního fontu a stiskneme klávesu SS+A. Kurzor si z tohoto znaku vezme vzorek a nastaví podle něho font.

# ULOŽENÍ TEXTU NA DISKETU

Celý text si uložíme na disketu z roletky **File** volbou funkce **Save**. Protože v textu také máme asi různé fonty a možná i semigrafiku, musíme si i toto vše uložit na disk. Vyvoláme roletku **Style** a zvolíme funkci **All fonts**, která nám na disk uloží všechny 4 fonty v jednom bloku. Pokud máme v textu i semigrafiku a předpokládáme, že s ní budeme ještě někdy pracovat, uložíme si ji také. Přejdeme do semigrafického editoru, kde šipkou zvolíme symbol ▣ a uvedeme název semigr.rutiny.

Takto uložený text pak můžeme pro opětovné použití nahrát do programu tak, že vyvoláme roletku **File**, zvolíme **Load**, uvedeme název pod kterým je text na disketě uložen. Pak vyvoláme roletku **Style**, opět zvolíme **Load** a zadáme příkaz **All fonts**. Pokud je v textu i semigrafika a chceme s ní dále pracovat, přejdeme do semigrafického editoru, zvolíme symbol ▣, uvedeme název semigrafické

rutiny a pak výběrem ikony ◪ ze semigrafického editoru můžeme se semigrafikou pracovat, resp. ji vkládat do textu.

# TISK STRÁNKY

Pokud chceme mít pro každou tištěnou stránku vždy stejnou hlavičku, využijeme k tomu funkce **EDIT HEADER**. Z roletky **Print** se po volbě této funkce dostaneme do části ve které můžeme využít jednoho řádku pro vytvoření hlavičky. Ta se pak automaticky vytiskne před tiskem textu. Header je vysoký celkem 24 bodů, to jsou dva řádky z nichž první je pro hlavičku a druhý zůstává vždy volný (nelze do něj psát). Zpět do textového editoru se vrátíme klávesou ENTER. Stejně můžeme využít i **EDIT FOOTER** ve kterém si uložíme zakončení stránky a to bude také vždy po vytištění každé stránky automaticky též vytištěno. Pokud nechceme ani z funkcí využít, zvolíme je postupně a pokaždé jejich obsah vymažeme klávesou SS+W.

Za hlavičkou (Edit header) následuje ještě při tisku několik prázdných řádků než začne být tištěna stránka. Počet těchto řádků, z nichž každý se skládá z osmi bodů, si můžeme nastavit volbou funkce **H-T Space X**.

Stejnou funkci má i volba **T-F Space X**, kterou určíme, kolik osmic bodů má následovat za Footerem.

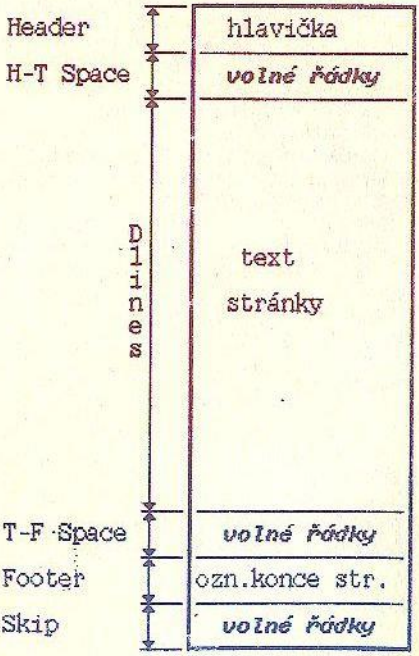
Pokud požadujeme aby byly stránky při tisku také číslovány postupně za sebou, označíme si nejprve volbou funkce **1st page X** číslo první stránky. V záhlaví (funkce Edit header) si označíme znakem # místo, kde se má tisknout číslo stránky.

Velikost stránky si nastavíme volbou funkce **Dlines XX, ale POZOR! nenastavujeme zde počet skutečných řádků, ale počet dvojic řádků na stránku.** Máme-li např. na stránku 60 řádků, pak uvedeme 30 dvojic.

Nyní si vysvětlíme jaké jsou možnosti volby tisku stránek. Pokud chceme a můžeme (to záleží zda naše tiskárna může tisknout na nekonečný papír) provést tisk na nekonečný papír, pak v roletce Print zvolíme funkci **Start** a tisk pak poběží nepřetžitě.

Vytisknout si však můžeme i předem definovanou část textu volbou funkce **Block**. To však předpokládá, že jsme si označili začátek a konec bloku.

Poslední možností je tisk na jednotlivé papíry. V tomto případě se po vytištění každé stránky tisk zastaví a čeká na stlačení libovolné klávesy.




*V příštím čísle si trochu podrobněji vysvětlíme jak pracovat se semigrafikou.*

-pc-

# ZAJÍMAVOSTI V OBLASTI ROM ZNAMÉ JAKO kalkulátor

Pro "AP" píše Jiří Brossmann

Většina lidí si pod pojmem kalkulátor představí nejspíš kapesní kalkulačku s tlačítky a displejem.

V lepším případě si vybaví nějaký kalkulátor integrovaný do programu, který více, či méně připomíná onu kapesní kalkulačku. Málokdo si však uvědomí, že i ve Spectru je takový kalkulátor. Není to samozřejmě nic jiného něž kus programu v ROMce ale přesto umí tento kalkulátor věle potřebné.Jeho kvality můžete nejlépe ocenin v Basicu, protože veškeré operace s čísly i s řetězci se provádějí právě za pomoci tohoto kalkulátoru.

S kalkulátorem úzce souvisí také některé podprogramy zajišťující zpracování výrazu v Basicovém řádku. Zvládnutí kalkulátoru je určitě důležité,protože se vám tak otevře možnost vypočítat ve strojáku prakticky cokoliv. Než však přejdeme k funkcím kalkulátoru,musíme si ujasnit několik věcí, které s tím úzce souvisí.

## 1.zpracování čísel

V počítači se kromě jedno a dvoubajtových čísel,které používá stroják, objevují také celá čísla (tzv.**integer**) a desetinná (buď **reálná** nebo také tzv. čísla **floating-point**, neboli čísla s plovoucí řádkovou čárkou (dále jen FP). K zápisu čísel integer i FP se používá pěti bajtů. Při zápisu v Basicu se číslo zapíše takto: nejprve se zapíší bajty, které je potřeba k vyjádření čísla při LISTu.Jsou to vlastně ASCII kódy daných čísel. Za nimy následuje bajt 14, který označuje, že jde o číslo, a za ním je 5 bajtů s vyjádřením čísla. Těchto 5 bajtů se při LISTu nezobrazí, ale počítá se právě s nimi (to, že se tyto bajty nevytisknou,zajistí právě ten bajt 14, takže podobně můžete schovat kus řádku). Při editaci se těch 6 bajtů (bajt 14 a 5 významových) vymaže a po stisku ENTERu se opět vypočte (další finta).

Teď se podívejme jak se zapisují čísla typu integer a FP:

### 1.1.-čísla integer

Tato čísla se zapisují takto:

1.bajt je 0.

2.bajt se používá jako znaménko.Je-li nulový, pak je číslo kladné, jestliže má hodnotu #FF, pak je číslo záporné.

3.bajt obsahuje nižší bajt čís.

4.bajt obsahuje vyšší bajt čís.

5.bajt je opět 0.

Takže např. číslo -20 je vyjá-

dřeno takto: 00 FF 14 00 00 (samozřejmě v hexa tvaru).

## 1.2.čísla floating-point

Tady je to trochu (hodně) složitější. Vyjádření je založeno na tom, že každé číslo lze vyjádřit jak X*Y↑n. Protože počítač pracuje s dvojkovými čísly, je Y vždy 2. Další postup při převodu je následující:

Nejprve číslo podělíme nejbližší vyšší mocninou 2. Získáme tak exponent a mantisu. K exponentu přičteme 128 (nastavený nejvyšší bit). Mantisy pak postupně odčítáme mocniny 2, přičemž postupujeme od -1 k -31 (31 bitů jsou 4 bajty) a do nejvyššího bitu prvního bajtu mantisy pak uložíme znaménko (je-li tento bit 0, pak je mantisa kladná, je-li 1, je záporná). Takto převedené číslo zapíšeme. Protože když to člověk čte, je to (skoro) nepochopitelné, uvedu příklad:

Dejme tomu, že máme číslo 14.38. Podle Pinďova expresu (řada ...512,128,64,32,16,8,4,2,1...) je nejbližší vyšší mocnina 16 (2↑4). Podělíme tedy;

14.38 / 16 = 0.89875

(pro kontrolu: 0.89875*2↑4=14.38)

Máme tedy exponent 4 a mantisu 0.89875. K exponentu přičteme 128+4=132.

Dále už operujeme s mantisou. Nejvyšší bit je volný pro znaménko, budu ho tedy nahrazovat "."

Mantisa je: 0.89875
Odečteme: -2↑-1
Mezivýsledek: 0.39875
což je větší než 0, takže nejvyšší bit je nastaven .1
Dále:

```
0.39875
-2↑-2
0.14875   =>.11
-2↑-3
0.02375   =>.111
-2↑-4
```

výsledný rozdíl je menší než 0, takže .110

a mezivýsledek opakujeme

```
0.02375
-2↑-5
0.02375   =>.11100
-2↑-6
0.008125  =>.111001
-2↑-7
0.0003125=>.1110011
```

a dále až do

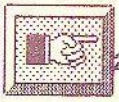
```
-2↑-12
0.000068359
      =>.1110011000001
```

**a tímto způsobem pokračujeme dále.**

Když máme mantisu převedenou, nastavíme znaménko (je kladná, čili 0) a číslo zapíšeme.

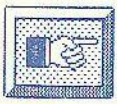
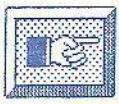
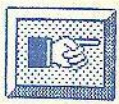
Samozřejmě se tak nevyhneme určité chybě, bude-li číslo mít mantisu, která se nedá dost dobře převést na binární formu (v manuálu prý stojí, že maximální číslo, které se zpracuje bez chyby je 4 294 967 295). Jak vypadá FP číslo v praxi vidíte na následujících příkladech:

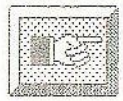
| dekad. | hexa |
|---|---|
| 0 | = 00 00 00 00 00 |
| 1 | = 81 00 00 00 00 |
| 1.111 | = 81 0E 35 EF 7D |
| -1.111 | = 81 8E 35 EF 7D |



## 2.proměnné kalkulátoru

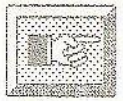
Možná jste si už prohlíželi popis systémových proměnných. Můžete při tom narazit na několik proměnných, které se odkazují na kalkulátor. Jsou to tyto: STK_BOT, STK_END, BREG, MEM a MEMBOT. Jsou v nich údaje týkající se zásobníku, kalkulačkových pamětí a čítače. Nejprve tedy k zásobníku kalkulátoru.



## 2.1. zásobník kalkulátoru

Tento zásobník slouží k uchování čísel, se kterými se má operovat. Každý zápis ho prodlouží o 5 bajtů (no a čtení ho o těch 5 bajtů zase zkrátí). Zásobník kalkulátoru (dále jen zásobník) se od uživatelského zásobníku (tak budeme nazývat zásobník adres) liší nejen typem ukládaných dat, ale také směrem růstu. Uživatelský zásobník má vrchol na nejnižší adrese, ale zásobník kalkulátoru ho má na adrese nejvyšší. Začátek (dno) zásobníku je v proměnné STK_BOT a jeho vrchol je v STK_END.

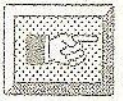


## 2.2. paměti kalkulátoru

Stane se, že při výpočtech potřebujeme uschovat mezivýsledek. K tomuto účelu je na kapesní kalkulačce paměť "M". U kalkulátoru ve Spectru existuje něco podobného. Pamětí je celkem 6 (MEM 0-MEM 5) a jejich obsah je uložen od adresy MEM. Tato proměnná se nastavuje na začátek proměnné MEMBOT, která má celkem 30 bajtů (logicky se to dá odvodit: máme 6 pamětí do kterých se ukládají 5-ti bajtová čísla. Z toho tedy plyne: 5*6=30).
Do pamětí se dá zapisovat, nebo z nich číst (k čemu by jinak taky byly, že...)

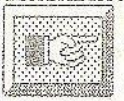
## 2.3. čítač

Při práci s kalkulátorem se občas používají smyčky a skoky. Aby se dala realizovat smyčka, potřebujeme čítač. U DJNZ se používá jako čítače registru B, ale ten by kalkulátor nerněl měnit, takže byla zavedena systémová proměnná BREG (b-register). Smyčka se používá např. při generování řad u "series generatoru".
Pokud jste se prokousali jednoduchou a nudnou úvodní pasáží, můžeme teď začít pronikat do tajů kalkulátoru.



## 3. kalkulátor

Kalkulátor je, jak jsem již řekl, část programu v paměti ROM počítače, kde se provádí aritmetické a řetězcové operace. Kalkulátor se volá pomocí restartu a protože si nejsem jist, jestli o nich v AP již něco vyšlo, ve zkratce se o nich zmíním.



## 3.1. restarty

Restarty jsou krátké programy na začátku paměti, které se volají pomocí příkazu RST. Jsou to vlastně podprogramy, které je možno volat také pomocí instrukce CALL, ale příkaz RST má na rozdíl od CALL pouze jeden bajt.
**Úplný přehled restartů byl již zveřejněn v AP číslo 1,2 a 3 ročník 1991** *(včetně funkcí kalkulátoru - pozn. redakce)*

Pro přiblížení se podívejme na příklad:

**Načtěte číslo X z Basicu a vypočtěte INT(X*PI/2)*X.**

```
LD  A,2       ;nastavení
CALL #1601    ;kanálu
LD  A,#16     ;tisková pozice
RST  #10      ;AT 0,0
LD  A,0       ;
RST #10       ;
LD  A,0       ;
RST #10       ;
RST #20       ;přisuň další znak
              ;(",")
```

```
CALL #24FB     ;vyhodnoť výraz
LD HL,#5C3B    ;proměnná FLAGS
BIT 6,(HL)     ;je číslo?
JR  Z,ERROR    ;ne - pak chyba

místo předchozího by mohlo být
       RST  #20
       CALL #1C82

RST   #28      ;X
               spusť kalkulátor
DB    #31      ;X,X
               zdvoj X
DB    #A3      ;X,X,PI/2
               přidej konstantu
DB    #04      ;X,X*PI/2
               vynásob
DB    #27      ;X,INT(X*PI/2)
               proveď INT
DB    #01      ;INT(X*PI/2),X
               vyměň
DB    #0F      ;INT(X*PI/2)+X
               sečti
DB    #38      ;opusť kalkulátor
CALL #2DE3     ;vytiskni číslo
LD HL,(#5C3D);nevypisovat
DEC HL         ;registr BC
DEC HL         ;při návratu do
LD SP,HL       ;Basicu
RET            ;Basic

ERROR RST #08;chybové hlášení
      DB  #0B;"Nonsence in..."

Program spustíme příkazem
PRINT USR start,X
```

Ještě bych vás chtěl upozornit, že podrobné informace o kalkulátoru jakož i dalších programech můžete získat z komentovaného výpisu ROM.

A na závěr ještě jednu radu: pokud budete s kalkulátorem pracovat, měli byste si vypsat všechny jeho funkce na speciální papír a každý večer před spaním si je dvakrát zopakovat...

Použitá literatuta:

(1) Programování ve strojím kódu díl 3.
(2) Komenotvaný výpis ROM.

-BST-

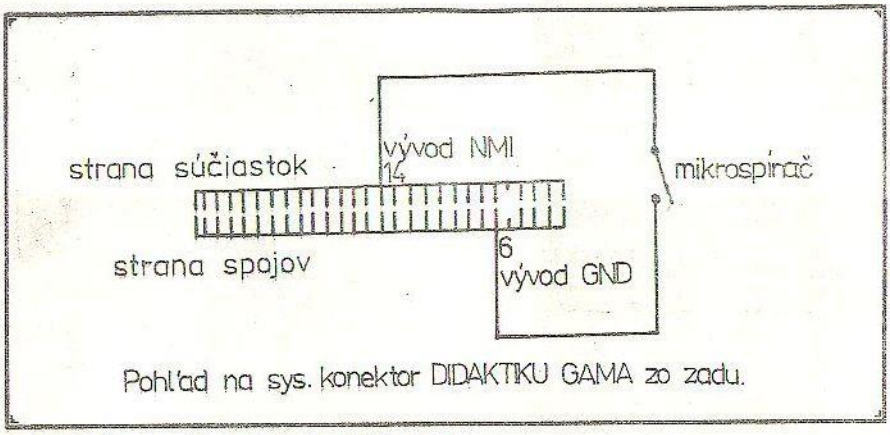
# VYUŽITIE NMI U DIDAKTIK GAMA

Možno každý majiteľ počítača po vzhiadnutí niektorých úvodných obrazoviek v hrách zatúžil mať obrázok, ktorý vidí na obrazovke i na papieri. Na to, aby sme mohli obrázok vytlačiť, musíme ho najprv uložiť na kazetu, ako súbor dát, ktorý môžeme ďalej spracovávať (dokreslovať) alebo vytlačiť. Nie vždy je to však možné preto, že hry väčšinou používajú špeciálne loadery.

Takéto ochrany je možné prekonať iba zásahom do hardware počítača. Jednou takou jednoduchou úpravou je využitie NMI, **nemaskovaného prerušenia.** Ide vlastne o využitie systému ZX Spectra, ktoré po privedení impulzu na vývod NMI systémového konektora preruší beh programu, ktorý práve vykonáva, uloží návratovú adresu na zásobník a skočí na adresu, ktorú vložíme do systémových premenných (23728 a 23729).

Adresa, na ktorú ukazujú tieto systémové premenné musí obsahovať rutinu na obsluhu NMI. To čo vykoná táto rutina závisí iba od vašej fantázie. Ak ukončíte túto rutinu inštrukciou RETN, vyberie sa zo zásobníka návratová adresa a program bude pokračovať tam, kde došlo k prerušeniu, privedením impulzu NMI.

Na konštrukciu jednoduchého




Pohľad na sys. konektor DIDAKTIKU GAMA zo zadu.

prípravku na generovanie NMI inpulzu potrebujete iba konektor WK 46580 s 2x28 vývodmi a mikrospínač, ktoré zapojíte podľa schémy na náčrtku.

Ďalej uvádzam vlastnú rutinu, ktorá po spustení zo štartovacej adresy nastaví syst. premenné tak, aby po privedení NMI inpulzu došlo k jej spusteniu. Program testuje stlačenie kláves a po ich stlačení vykonáva nasledujúce činnosti:

SAVE (S) uloží na kazetu hlavičky obsah obrazovky.

VERIFY (V) verifikuje nahrávku a v prípadě chyby vydáva zvukový signál až do stlačenia lubovolnej klávesy.

B prevedie návrat do Basicu, ak však došlo k prepísaniu syst. premenných, môže dôjsť k zamrznutiu počítača alebo k resetu.

CONTINUE (C) pokračovanie po vykonávaní programu (*obsah použitých registrov, povolí sa prerušenie a prevedie sa skok do RAM do programu na obsluhu maskovateľného prerušenia, ktorá vyzdvihne zo zásobníka návratovú adresu a skočí na ňu*). Rutina nie je relokovateľná.

**Pár poznámok:**

*NMI nie je možné používať na počítačoch ZX SPECTRUM a DIDAKTIK M z dôvodu chyby v ROM u týchto typov. Avšak u počítača DIDAKTIK GAMA je táto chyba již odstránená a preto NMI pracuje u tohoto počítača správne.*

*Rutinu na obsluhu NMI musíte umiestniť v RAM tak, aby nedošlo k jej prepísaniu (odporúčam ju umiestniť na koniec RAM, alebo do buferu tlačiarne 23296).*

*Taktiež nesmie dôjsť k prepísaniu systémových premenných.*

Rutinu na obsluhu NMI ve zdrojovém textu pro Prometheus najdete na další straně.

Příspěvek nám poslal Slavomír [illegible] se [illegible] mjak gramatick[illegible]

Redakce neručí za případné chyby ve výpisech programů od přispěvatelů ani za kvalitu jejich čitelnosti, pokud se jedná o kopie.

```
          ent   $              ;zaciatok programu
          org   $
AOSTART   ld    hl,RUN         ;inicializacia sys.prem.NMI
          ld    (23728),hl
          ret

RUN       push  af             ;rozskok podla stlacenia klaves
          exx
          di
          call  BEEP
UP        ld    a,2
          out   (254),a
          call  KEY
          cp    "s"
          call  z,SAVESCR
          cp    "v"
          call  z,VERSCR
          cp    "b"
          jr    z,BASIC
          cp    "c"
          jr    z,CONT
          jr    UP

KEY       ld    iy,23610       ;nacitanie stlacenia klavesy
          ld    (iy+1),204
          ld    (iy+7),0
          ld    (iy+48),0
          ei
UP2       halt
          bit   5,(iy+1)
          jr    z,UP2
          res   5,(iy+1)
          ld    a,(23560)
          di
          ret

BEEP      ld    hl,1000        ;zvukovy signal
          ld    de,20
          call  #03B5
          ret

SAVESCR   call  PARAM          ;ulozenie obrazovky na kazetu
          call  #04C6
          call  BEEP
          ret

VERSCR    call  PARAM          ;verify obsahu obrazovky
          or    a
          inc   d
          ex    af,af'
          dec   d
          ld    a,12
          out   (254),a
          call  #0562
          ret   c
```

```
UP3     call  BEEP
        xor   a
        in    a,(254)
        cpl
        and   31
        jr    z,UP3

        ret

BASIC   ld    a,7               ;navrat do Basicu
        out   (254),a
        ld    iy,23610
        ei
        rst   #08
        defb  255

CONT    pop   af                ;pokracovanie v prerusenom progr.
        exx
        jp    #38

PARAM   ld    ix,16384          ;nastavenie param.pre kaz.operacie
        ld    de,6912
        ld    a,255
        ret
```

# RUTINY NA OVLÁDÁNÍ JEDNOTKY

**D40 a stroják**

**Jako většina uživatelů počítače Didaktik i já mám disketovou jednotku D40. Jakožto zvědavý člověk jsem se také pokoušel dostat tomuto zařízení na kobylku, ale dlouho jsem nevěděl jak lze tuto periferii ovládat prostřednictvím strojového kódu. Štýral jsem se tedy v některých programech, které si v klidu nahrávaly své dohrávky z diskety a to prostřednictvím jakési záhadné rutiny na kterou jsem nemohl přijít.**

Zvláště mě vždy odradila instrukce **rst#00** na začátku těchto programů. Tak jsem toho nechal a začal předělávat programy pomocí zajímavého fígle (?) a to z Basicu. Po návratu do Basicu po instrukci **ret** jsem místo příkazu **STOP** v Basicu umístil vyvolání obslužné rutiny pro výběr MENU, která zjistila co chcete uskutečnit (LOAD*, SAVE*,...) a tuto instrukci zapsala na zvolený řádek v Basicovém programu. To bylo u programu SCREEN MACHINE od firmy CYBEXLAB. Dále pak zjistila jméno a dosadila jej na určené místo v Basicovém programu (obvykle mezi uvozovky) a také si vypočítala adresu nahrávání pokud šlo o fonty a umístila ji na příslušnou pozici. Nakonec se tato rutina opět vrátila zpět do Basicu a vykonala příkaz podle zadání z rutiny. Tato rutina je velice jednoduchá a proto ji nebudu ani uvádět (můžete si ji vytvořit v Basicu a pak ji zkompilovat). Doufám, že jste tento postup pochopili. Já vím, je to trochu stručné, ale přece nebudu plýtvat místem v tak efektivním časopise. Tak pokud máte opravdu zájem o podrobnosti, napište, určitě vám odpovím (přiložte ofrankovanou obálku).

**A nyní něco pro ty chytřejší..** Po přečtení AP 2-3/93 jsem konečně zjistil funkci instrukce **rst#00** (*chytrý*)no a začal jsem znovu se šťouráním do různých programů. Netrvalo to dlouho a zjistil jsem to, o čem jsem tak dlouho snil. **Rutiny na ovládání D40 ze strojového kódu.** A to jak pro LOAD tak pro SAVE. Jsou velice důmyslně vytvořeny a umístěny v RAM D40 (*opět chytré*). Pomocí této rutiny jsem si již upravil několik her (R-TYPE nebo HEROQEST, který je na DG80, protože se v tom *rýpal TESIL a pěkně to pro tento typ počítače upravil bez dohrávek*). Nebudu vás tedy již napínat a konečně vám tyto rutiny prozradím.

**Zavedení do RAM D40** (nezapomeňte **POKE #247,Z9**)

Tento program použijte např. když máte nedostatek místa v paměti, to znamenná, že není povinný. Od adresy 16128 si můžete uložit cokoli, nejen obslužné programy pro D40.

```
rst #00 (aktivace D40 ROM/RAM)
di
ld hl,adresa na které máte
      program pro uložení.
      Např.program LOAD.
ld de,16128
```

```
ld bc,délka tohoto programu
ldir
call 5888 (aktivace Basic ROM)
ei
.
.
program vyvoláte rst s00, di,
call 16128, ei, ...

Program LOAD
na adrese 16010 v D40 RAM musí
být název o délce 10ti znaků.
Např.název VISOFT.B uložte jako
86,73,83,79,70,84,0,0,0,0,66

      rst #00
      call 7311
L1    call 8491
      jr nz,L1
      push hl
      ld(15986),hl
      pop ix
      call 6625
      ld hl,adresa uložení(CODEadr
      ld(15988),hl
      call 6578
      call 9526
      di
      ld a,79
      ld(16119),a
      call 5888(aktivace BASIC ROM
      .

Program SAVE
(stejná pozn.jako pro LOAD)

rst #08 (aktivace D40 ROM/RAM)
ld hl,adresa začátku pro uložení
ld(15992),hl
ld(16088),hl
call 8609
ld de,délka pro uložení
call 6656
call 9526
di
ld a,79
ld(16119),a
call 5888(aktivace
.         Basic ROM)
```

D40 a stroják

D40 a stroják

Pokud vyvoláte rutiny z D40 RAM instrukce rst s00 a call 5888 použijte ve vyvolávacím programu.

-vid-

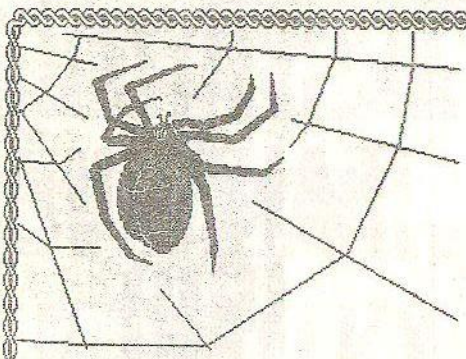
# S P D TAKÉ JIŽ PRO KOMPAKT

Díky dohodě mezi firmou Didaktik a.s. ve Skalici mám již možnost poskytovat programy ze sítě SPD také na malých disketách 3,5".

Protože redakce AT v podstatě již vlastně bankrotovala a dále je časopis vydáván podle toho, jak se podaří sehnat peníze na jednotlivé číslo, mohl jsem si s novým počítačí Kompakt nechat jen zdát.

Nyní jsem ze Skalice obdržel zcela nový počítač Kompakt a tak mohu vyhovět všem jeho uživatelům a rozšiřovat programy i na 3,5" disketách.

V době, kdy píšu tyto řádky byla již strana 26 a 27 tohoto čísla vytištěna, a proto není v nabídky SPD uvedeno, že programy lze objednávat i na disketách 3,5". Tato možnost platí od této chvíle pro veškerou nabídku programů, které si lze objednávat jak na disketách 5,25" tak i na 3,5". Týká se tedy i programů pro účetnictví a fakturaci, stejně jako i nové verze kompletu FOS verze 05 od WESOFTu. Závěrem připomínám, že je nutné zaslat si vlastní naformátované diskety a doklad o zaplacení příslušného poplatku.



# KURZ PROGRÁMOVÁNÍ STROJOVÉHO JAZYKA

pro AP píše Pavel Macek

Jak se vám líbí zobrazovat na obrazovce obsahy registrů nebo paměťových míst? Že je to zajímavé a užitečné? Bezpochyby. Zkusil se někdo zamyslet, jakým způsobem se zobrazí na obrazojku číslo šestnáctibodové? tedy takové, které vyjadřuje adresu paměťového místa a vyskytuje se velmi často v registrových párech (BC,HL,DE). Chceme-li obsah reg. párů zobrazit způsoby jaké jsme si dosud ukazovali, tedy dvojkově či šestnáctkově, nebude to žádný problém. Nejprve totiž zobrazíme obsah vyššího z registrů (pokud se jedná o HL tak nejprve reg.H) a potom reg. nižší (tedy L, v případě HL). Získáme tak číslo, složené ze šestnácti znaků nula nebo jedna v případě dvojkového vyjádření, nebo číslo složené ze čtyř znaků od 0000 do FFFF, v případě šestnáctkového vyjádření.

## DESÍTKOVÉ (DEKADICKÉ) ČÍSLO

Desítková číselná soustava má za základ, jak víme, deset číslic

0 1 2 3 4 5 6 7 8 9

Jestliže chceme zobrazit desítkově osmibitové číslo, musíme zjistit kolik obsahuje stovek, desítek a jednotek. Číslo pak zobrazíme na třech znacích, které představují počet jednotlivých řádků. Jakým způsobem se to provádí si ukážeme na konkrétním programu, který dokáže zobrazit desítkově vyjádřené číslo z registru A. Na počátku programu uložíme do registru A číslo, které pak budeme zobrazovat. Do reg.páru DE připravíme adresu do obrazovky, na níž přenášíme později tvary znaků vyjadřujících číslo. Registr C nulujeme, protože ho budeme používat jako čítač řádů. Začneme řádkem nejvyšším, tedy stovkami. Od čísla v reg.A začínáme odečítat sto. Jestliže od menšího čísla odečteme větší, nastaví se CARRY v reg.F na jedna. V takovém případě víme, že stovka již nelze od čísla odečíst a musíme ji tedy k obsahu reg. vrátit tak, že ji opět přičteme. Potom zobrazíme obsah reg.C a začneme odečítat desítky. Registr C je totiž při každém úspěšném odečtení řádu zvětšen o jedna a obsahuje tedy po skončení odečítání počet řádků, které číslo obsahuje. Když spočteme kolik má číslo desítek, přičteme poslední odečtenou desítku, zobrazíme počet desítek z reg.C na obrazovku a potom odečítáme od zbytku čís.v reg.A jednotky. Po zobrazení počtu jednotek máme desítkově zobrazeno 8mi

hitové číslo, které může nabývat hodnot od 000 do 255. Od návěští dek8 je prakticky zobrazovací část programu, která podle čísla v reg. C najde tvar potřebného číslicového znaku a u návěští dek8 jej pak zobrazí.

Když budeme chtít zobrazit desítkově číslo šestnáctibitové, bude postup stejný, ale delší o řád tisíců a desetitisíců. Desítkové číslo se bude skládat z 5ti číselných znaků a bude nabývat hodnot od 00000 do 65535.
Dávejte pozor na nastavení CARRY Raději jej hned na začátku takového programu nulujte.

```
ZK      ld  a,125
DEKA    ld de,18432+10
        ld  c,0
DEK1    sub 100
        jr  c,DEK2
        inc c
        jr  DEK1
DEK2    add a,100
        push af
        call DEK5
        pop af
        ld  c,0
DEK3    sub 10
        jr  c,DEK4
        inc c
        jr  DEK3
DEK4    add a,10
        push af
        call DEK6
        pop af
        ld  c,0
DEK5    sub 1
        jr  c,DEK6
        inc c
        jr DEK5
DEK6    ld  a,c
DEK7    ld bc,15616+128
        ld  h,0
        ld  l,a
        add hl,hl
        add hl,hl
        add hl,hl
        add hl,hl
        ex  de,hl
        push hl
        ld  b,8
DEK8    ld  a,(de)
        ld (hl),a
        inc de
        inc h
        djnz DEK8
        pop de
        inc de
        ret
```

## ZNAKY

Číslo, které se ukrývá na některém paměťovém místě, může také vyjadřovat kód nějakého znaku. Bývá proto zvykem vypisovat nejenom číselnou hodnotu bajtu, ale také znak, který může bajt eventuálně představovat. Jaké kódy jsou vlastně přiřazeny znakům? Pokud to nevíte, můžete si vzít k ruce příručku k počítači, která by měla tyto údaje někde obsahovat. Dozvíte se, že kódy znaků začínají od čísla 32 (mezera) a končí u čísla 127 (což je znak COPYRIGHT). Ve znakovém výpisu bajtů budeme tedy muset všechna ostatní čísla ignorovat a psát na místo nich nějaký předem zvolený, neutrální znak (např. tečku). Celý postup si vysvětlíme opět na konkrétním programu.

U návěští ZK vkládáme do reg. A vypisovaný bajt a do reg. páru DE připravujeme adresu do obrazovky, na níž bude probíhat výpis. U návěští aski se do reg. páru HL připravuje adresa tvaru znaku (v našem případě se jedná o tečku). Nyní testujem zda vypisovaný bajt není menší než nejnižší kód znaku, tj. 32. Jestli je bajt nižší, vypíše se na obrazovku tečka. Zbývá zjistit zda bajt není naopak větší než nejvyšší kód znaku, tj. 127. Pokud větší je pak se opět na obrazovku vypíše tečka. Když ovšem bajt má hodnotu jako některý kód znaku, zjistí se podle tohoto kódu adresa tvaru znaku a znak se vypíše na obrazovku.

Program, jak vidíte obsahuje části, které již dobře známe, totiž podprogram pro hledání adresy tvaru znaku nebo podprogram pro zobrazení znaku. Tady máte důkaz, že všechno nové je z velké části využití toho, co jsme se už naučili.

# Znaky

```
ZK      ld  a,25
        ld de,16432+10

ASK1    ld hl,15616+112
        cp  32
        jr  c,ASK1
        cp 128
        jr  nc,ASK1
        sub 32
        ld  bc,15616
        ld  h,0
        ld  l,a
        add hl,hl
        add hl,hl
        add hl,hl
        add hl,bc
ASK1    ex  de,hl
        push hl
        ld  b,8
ASK2    ld  a,(de)
        ld (hl),a
        inc de
        inc  h
        djnz ASK2
        pop de
        inc de
        ret
DELKA   equ $-ZK
```

Číslo již na obrazovku zobrazit umíme a dokonce třemi způsoby. Dostáváme se k problému, kdy potřebujeme naopak převést číslo ze znakové podoby do registru. Taková situace může nastat, budeme-li chtít například zadávat nějaká čísla z klávesnice. Po stisku klávesy získáme zpravidla kód jeho znakové podoby. Podle kódu můžeme nalést adresu jeho tvaru a zobrazit jej na obrazovce. Jak však dostaneme jeho hodnotu do registru, abychom ho mohli uložit na některé paměťové místo?

Začneme číslem osmibitovým, vyjádřeným šestnáctkově (hexadecimálně). Takové číslo se skládá ze dvou znaků. V našem programu máme například u návěští TYPE znakovou podobu čísla 42A. Na počátku programu u návěští POKEX ukládáme adresu znakové podoby čísla do reg.páru HL. Vezmeme první znak čísla do reg.A. Nyní musíme zjistit, zda je znak v relaci 0 až 9 nebo A až F. Když je znak v prvním rozmezí, stačí od jeho kódu odečíst kód znaku nula a máme v reg.A hodnotu čísla. Jestliže je znak ve druhém rozmezí, potom odečteme od jeho kódu takové číslo, které vyjadřuje kód znaku A zmenšený o 10 neboť znak A znázorňuje číslo 10. Máme-li hodnotu prvního znaku v reg.A, nulujeme CARRY a posuneme tuto hodnotu do vyšších čtyřech bitů. Tento stav uschováme do registru C. Zvětšíme HL na další znak čísla a ten dáme do reg. A. Znovu otestujeme kód znaku a odečteme od něho příslušné číslice, abychom v registru A získali jeho hodnotu. Tato bude na nižších čtyřech bitech, neboť se jedná o hodnoty 0 až 15.

Instrukcí OR C přidáme horní čtyři bity, jejichž hodnotu jsme získali z prvního znaku čísla. V reg.A je tedy po návratu z tohoto programu hodnota osmibitového čísla, vyjádřeného pomocí kódů znaků jako číslo hexadecimální.

Zkuste se zamyslet jak by to bylo z číslem 16tibitovým.

Šestnáctibitové číslo jsou vlastně dvě čísla po osmi bitech, takže není potřeba žádných zvláštních úprav, neboť stačí to celé nechat proběhnout nejprve pro první dva znaky (vyšší bajt) a pak pro druhé dva znaky (nižší bajt).

```
POKEX   ld  hl,TYPE

POKE1   ld   a,(hl)
        call POKE4
        or   a
        rla
        rla
        rla
        rla
        ld   c,a
        inc hl
        ld   a,(hl)
        call POKE4
        or   c
        ret

POKE4   cp   48
        ret  c
        cp   65
        jr   c,POKE2
        sub  55
        ret
POKE2   sub  48
        ret

TYPE    defm "2A"
```

Dokázali byste vymyslet postup jakým způsobem by se převáděl do registru 8mibitové číslo vyjádřené znaky jako desítkové (dekadické) ?

Pavel Rak

# systémové proměnné

## dokončení z minulého čísla

◆ ◆ ◆ ◆ **OLDPPC 23662** ◆ ◆ ◆ ◆ ◆ ◆
číslo řádku kterým bude pokračovat program po CONTINUE

◆ ◆ ◆ ◆ **OSPPC 23664** ◆ ◆ ◆ ◆ ◆ ◆
číslo příkazu na který skočí CONTINUE

◆ ◆ ◆ ◆ **FLAGX 23665** ◆ ◆ ◆ ◆ ◆ ◆
různé parametry (není uvedeno)

◆ ◆ ◆ ◆ **STRLEN 23666** ◆ ◆ ◆ ◆ ◆ ◆
délka právě vykonávaného řetězce

◆ ◆ ◆ ◆ **T_ADDR 23668** ◆ ◆ ◆ ◆ ◆ ◆
adresa další položky v syntaktické tabulce

◆ ◆ ◆ ◆ **SEED 23670** ◆ ◆ ◆ ◆ ◆ ◆
slouží pro generování náhodných čísel (náhodná čísla se negenerují náhodně, ale podle pevně daného programu). Příkaz RANDOMIZE číslo (0-65535) rozloží na dvě osmibitové, výsledné číslo je pak dáno vztahem číslo=PEEK 23670+256*PEEK 23671

◆ ◆ ◆ ◆ **FRAMES 23672** ◆ ◆ ◆ ◆ ◆
při každém přerušení (50x za sekundu) se obsah FRAMES zvýší o 1, po RESETu se nastaví na 0. Lze spočítat kolik sekund je počítač zapnutý a to výrazem: TIME=(PEEK 23672+256*PEEK 23673 *PEEK 23674)/50. Přesnost je závislá na kmitočtu v el. síti, který se mění den ze dne, z hodiny na hodinu, dělením to lze převést na formát HH:MM:SS pokud je ale zakázáno přerušení (BEEP,SAVE,LOAD...) tak k inkrementaci nedochází.

◆ ◆ ◆ ◆ **UDG 23675** ◆ ◆ ◆ ◆ ◆
adresa prvního znaku uživatelské grafiky (UDG), po RESETu se nastaví na 65368 (#ff58), font pro UDG je v ROM na #3eaf

◆ ◆ ◆ ◆ **COORDS 23677** ◆ ◆ ◆ ◆ ◆
obsahuje X-ovou souřadnici posledního bodu nakresleném příkazem PLOT, DRAW

**23678** ◆ ◆ ◆ ◆ ◆
totéž pro Y-ovou souřadnici

◆ ◆ ◆ ◆ **P_POSN 23679** ◆ ◆ ◆ ◆ ◆
číslo sloupce pozice pro LPRINT

**PR_CC 23680** bajt příští pozice tisku při LPRINT, méně významný

**nonarme 23681** většinou volný, občas je používán jako počítadlo vytištěných řádků na tiskárně

**ECHO_E 23682** adresa ve spodní části obrazovky za kterou nejde dál posunout kurzor směrem doprava

**DF_CC 23684** adresa 1 bajtu psaného na obrazovku

**DF_CCL 23686** platí totéž co bylo uvedeno výše pro spodní část obrazovky

**S_POSN 23688** obsahuje číslo sloupce 23558 (#5c88) a číslo řádku v 23689 (#5c89) běžné pozice pro PRINT. **POZOR : řádky a sloupce mají vnitřně jiné číslování než je zvykem v Basicu !**

**SYSTÉMOVÉ PROMĚNNÉ**

**S_POSNL 23690** totéž, ale pro spodní část obrazovky

**SCR_CT 23692** počet řádků vypsaných příkazem LIST dokud se neobjeví hlášení; "scroll ?". Zkuste tam dát 255 (nebo -1, jak je libo) a zadat příkaz LIST.

**ATTR_P 23693** barevná informace pro právě tištěný znak, nastavení v ROM #1265

**MASK_P 23694** obsahuje masku pro ATTR_P

**ATTR_T 23695** barevná informace o právě vytištěném znaku, nastavení v ROM na #128a

**MASK_T 23696** maska pro ATTR_T, pouze dočasně

**P_FLAG 23697** další info systém

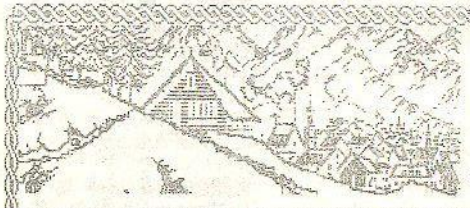
**MEMBOT 23698** zásobník kalkulátoru pro uložení hodnot se kterými pracuje, "more information in"

**NMI 23720** vektor pro NMI, lze využít pouze u Didaktiku Gama, protože Spectrum (Didaktik M) mají špatně v ROM ošetřenou obsluhu tohoto vektoru, jinak při přijetí signálu NMI procesor uloží poslední adresu hlavního programu z PC do zásobníku a PC natvrdo nastaví adresu 102 (#66), na této adrese začíná program pro obsluhu nemaskovaného přerušení, který vezme adresu z SP NMI a skočí na tuto adresu. Spectristi tyto dva bajty mohou použít pro své potřeby

**RAMTOP 23730** adresa posledního bajtu paměti využitelné pro Basic

**P_RAMT 23732** adresa posledního fyzického bytu v RAM.

*(dokončení příště)*



**Také kreslíte na počítači ?**

Pochlubte se nám! Svůj obrázek pošlete na disketě nebo na kazetě do redakce

# komprese dat

LEPŠÍ NEŽ PSANÍ DO AP JE POUZE VÝROBA VIRŮ

Komprimování dat je činnost podnětná a zajímavá. Kromě úspory místa na kazetách, disketách a tapetách (tady něco neštimuje) je zkomprimovaný program dobře chráněn proti různým nenechavcům typu **agent W24C**, kteří hodnotí program podle odolnosti ochrany(hlázni) Další výhodou je,že když člověk vidí program, který je zapakovaný,řekne si: **Ten si s tím pohrál!"** (myslí tím autora, který program - většinou hru - programoval 3 dny a byl rád,že se ho zbavil).

Ke komprimaci lze použít nej(h)různějších postupů. Já osobně jsem slyšel asi o 10ti druzích,mezi nimiž dominovaly komprese **bitové** a **aritmetické**.Je to zvláštní, ale princip těchto kompresí jsem,přes veškerou snahu nepochopil...

Z DG možná znáte kompresi R.Gemrota, nebo různé kyselé deště v BORDERu po nahrání programů.

Komprese se kterou vás chci (doufám,že zájem je obousranný) seznámit, je někde mezi Gemrodem a hustým mrholením.

Komprese se provede pouze tam, kde je účinná, eventuelně tam, kde to chcete.

Ke kompresi slouží dva programy.Jednomu se obvykle říká **kompresor** - protože zajišťuje kompresi (pakovaní, balení, archivaci,stlačení,sražení...), druhému **dekomprese**,který zase uvede data do původního stavu (rozbalí, rozpakuje)

Podobně jsou nazvány i dva programy v Basicu, které umožňují snadnou a celkem rychlou (ale po kompilaci, jak jinak) kompresi a dekompresi.

## JAK PROGRAMY FUNGUJÍ?

(taky by mě to zajímalo)

Při kompresi se nejprve projde úsek, který chceme stlačit a určí se, které byty se vyplatí komprimovat,jestliže je za bytem stejný byt,pak se do pole B() přičte jednička, je-li za aktuálním bytem jiný byt, pak se jedna odečte.Pole B() má 256 prvků (víc bytů se mi zatím nepodařilo najít) a na začátku se do něho vloží hodnota -1, protože každý komprimovaný byt potřebuje ještě jednu pozici v tabulce.

Potom se pole projde a tam, kde je hodnota větší než **nula** se vloží **jedna** (to jenom pro pořádek, nemá to žádný závažnější význam) a založí se další prvek do tabulky. Na jejich začátek se pak zavede její délka. Bude-li tedy komprimovaných 5 bytů 1-5, pak v tabulce budou hodnoty 05 01 02 03 04 05.

Dále se znovu prochází komprimovaná oblast s tím, že se vytváří zapakovaný blok (v jiné části paměti).Postupuje se tak, že nepakovaný byt se přenese, za pakovaný se zapíše počet -1. Najde-li program např. deset nul a nula bude pakovatelná,

zkrátí se oněch deset nul na dva byty 00 09.
Jestliže však nula nebude pakovatelná, přepíšou se nuly tak, jak jsou.
Při dekompresi se postupuje obdobně. Nejprve se načte tabulka a pak se spakovaný blok rozbaluje (nalezne-li nepakované byty, pak je přenese, pakovatelné uvede do původního stavu).
Tento postup předpokládá při dekompresi dva bloky v paměti Pokud však chcete mít pouze jeden, můžete si zapakovaný blok posouvat v paměti nahoru, vždy o potřebný počet zkrácený (pozor na to, že musíte posouvat odshora dolů - tedy nejprve nejvyšší byt, pak nižší, jinak se zapakovaný blok přepíše!). Vždy pak musíte změnit adresu zapakovaného bloku. To je však již spíše pro assembler, protože Basic, ač kompilovaný, by byl nechutně pomalý (což už je tak). Berte to tedy jako výzvu.
Chcete-li pakovat jen některé byty, stačí, když si v poli B() tyto označíte 1 (samozřejmě, že pro daný byt je určen prvek byt+1).

Na závěr ještě heslo dne:

**Žádná komprese není dost účinná, aby nemohla být ještě účinnější.**

Ještě bych chtěl poděkovat Romanu Střížkovi, který celý princip komprimace navrhl.

-BST-

*Výpis obou programů najdete na dalších stránkách*

## NA ZÁVĚR JEŠTĚ NĚKOLIK RAD PRO PROGRAMÁTORY

1. Nikdy nepouštět do bytu návštěvy, které nenesou nový software
2. Komunikaci s okolím provádět pouze přes port A
3. Neopouštět počítač, pokud to není nezbytně nutné
4. U počítače pobývat od 17,00 do 05,00 a spát v zaměstnání
5. Všechny časopisy číst 10x
6. Kupovat pouze počítačové časopisy
7. Rozbíjet cizí programy všemi dostupnými prostředky
8. Naučit se z paměti hexa vyjádření celé ASCII tabulky, včetně řídících kódů
9. Mít doma záložní zdroj, pro případ výpadku el.proudu
10. Přečíst si tato pravidla vždy, než usednete k počítači.

Mimochodem, který autor je na obálce posledního čísla AP?
(obrázek v rámečku vpravo nahoře)

```
    2 REM ***KOMPRES***
    4 REM
    5 REM :INT #ADR,CIL,E,F,I,G
    6 REM :INT BC),DEL
   10 REM : OPEN 0
   20 DIM B(256)
   30 LET ADR=16384: LET DEL=6912: LET CIL=58600
   40 FOR I=1 TO 256: LET B(I)=-1: NEXT I
  100 LET E=ADR: LET F=DEL
  110 BORDER 0: LET E=E+1: LET F=F-1
  120 IF PEEK (E-1)=PEEK E THEN  LET B(1+PEEK E)=B(1+PEEK E)+1: GO TO 140
  130 LET B(1+PEEK E)=B(1+PEEK E)-1
  140 BORDER 7: IF F THEN  GO TO 110
  150 REM
  160 LET E=0
  170 FOR I=1 TO 256: IF B(I)>0 THEN  LET E=E+1: LET B(I)=1: POKE CIL+E,I-1
  180 NEXT I
  185 IF E=0 THEN  PRINT AT 21,0;"NELZE KOMPRIMOVAT.": STOP
  190 POKE CIL,E: LET F=CIL+E+1: LET E=ADR
  210 IF B(1+PEEK E)>0 THEN  POKE F,PEEK E: GO SUB 1000: POKE F+1,G: LET F=F+
1: GO TO 240
  230 POKE F,PEEK E: LET E=E+1
  240 LET F=F+1: LET DEL=DEL-1
  250 BORDER 7: BORDER 0: IF DEL>0 THEN  GO TO 210
  260 PRINT AT 21,0;"DELKA:";F-CIL: BEEP .1,10
  270 STOP
 1000 REM
 1010 LET E=E+1: LET G=0
 1020 FOR I=1 TO 254
 1030 IF PEEK (E-1)=PEEK E THEN  LET DEL=DEL-1: LET G=G+1: LET E=E+1: IF DEL
THEN  NEXT I
 1040 RETURN
```

ADR ... adresa nespakovaného
CIL .... adresa spakovaného
DEL .... délka nespakovaného

```
    2 REM ***DEKOMPRES***
    4 REM
    5 REM :INT #ADR,CIL,E,F,I
    6 REM :INT BC),DEL
   10 REM : OPEN 0
   20 DIM B(256)
   30 LET ADR=16384: LET DEL=6912: LET CIL=58600
   40 FOR I=1 TO 256: LET B(I)=-1: NEXT I
  300 REM
  310 LET F=ADR: LET E=1+CIL+PEEK CIL
  320 FOR I=1 TO PEEK CIL: LET B(1+PEEK (I+CIL))=1: NEXT I
  330 IF B(1+PEEK E)>0 THEN  GO SUB 2000: GO TO 360
  340 POKE F,PEEK E: LET E=E+1: LET F=F+1
  350 LET DEL=DEL-1
  360 BORDER 7: BORDER 0: IF DEL>0 THEN  GO TO 330
  370 STOP
 2000 REM
 2010 FOR I=0 TO PEEK (E+1): POKE F,PEEK E: LET DEL=DEL-1: LET F=F+1: NEXT I:
 LET E=E+2
 2020 RETURN
```

# CHYBA V MDOSU

Zvláště proto, že při programování pracují moje nervy naplno, jsem zvyklý neprovádět zbytečné testy nejen strojových rutin, ale i disket.

Již několikrát se mi podařilo oddělat pár disket s docela důležitými informacemi. Bylo to však způsobeno chybou v MDOSu.

Totiž, když zadáte SAVE s nějakými parametry a budete pak chtít soubor zapsat na disketu, která je chráněna proti zápisu, tak je lepší již nemačkat >R< pro Retry. Nejlépe to pochopíte tak, když objetujete nějakou disketu.

## POKUS ČÍSLO JEDNA:

Vemte si disketu s nejdůležitějšími informacemi (a pro jistotu smažte i všechny svoje archivní kopie), zalepte proti zápisu. Pak zadejte třeba SAVE #"nečum"CODE 30000,1000 Zasuňte naši disketu do mechaniky a stlačte ENTER. Po chvíli se logicky objeví hlášení > DISC IS WRITER PROTECTED (Retry=R) nebo tak nějak... Protože jsme inteligenti, tak vyndáme disketu a drasticky strhneme nálepku tak, aby disketa vypadala jako po zápase s PC XT/AT.

Zasuneme tedy disketu zpět, stlačíme >R< a můžeme si být na 100% jisti, že kus diskety se totálně a perfektně zničil.

To je jedna věc. A nyní...

## POKUS ČÍSLO DVĚ:

Pokud máte rádi zázraky (tak jako já), tak to trošku vylepšíme.

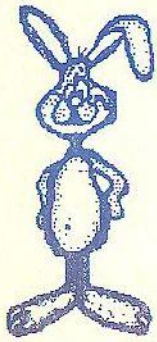
Postupujte stejně až do doby, než se objeví známé hlášení >DISK IS WRITE...atd. Pak vsuňte do mechaniky jinou disketu, která není chráněná proti zápisu. Když stisknete R, tak se ani moc neví co se stane. Když potom provedete CATalog u této diskety, tak zjistíte, že se zkopírovali různé soubory z té první diskety. O.K. V případě, že se však pokusíte nějaký tento soubor nahrát, tak bude počítač protestovat, ale nejzajímavější na tom je to, že bude protestovat i proti nahrání jiných programů... Z pravidla si vyberete ty nejdůležitější a soubory, třeba jako jsou obrázky vám nechá, jako vaši poslední radost...

Ověřena je pouze ta druhá metoda, ale ta první bude fachčit taky, na to vemte klidně i $H_2SO_4$.

Pokud také přijdete na nějakou podobnou lahůdku, kterou vám MDOS poskytl, pak neváhejte a napište nám to.

Agent 24C

V minulém čísle byl chybně vysázen pátý řádek, ve druhém sloupci, kde mělo být správně uvedeno; "a proto to nešlapalo i když byl tento řádek vymazán." Za tuto chybu se omlouvám.



# ani drobný živnostník nemusí být pozadu

## posudte sami

Pokud budete chtít vést své jednoduché účetnictví a ostatní agendu na počítači a podlehnete všeobecnému názoru, že je na to potřeba jedině počítač řady PC, pak si musíte připravit:

25.000 až 40.000 Kč na počítač,
7.000 až 20.000 Kč na tiskárnu
2.000 až 10.000 Kč na programy

celkem Vás to tedy bude stát 34.000 až 70.000 Kč.

## přesto je tu i mnohem levnější řešení

Programy na vedení jednoduchého účetnictví pro neplátce DPH a další agendy totiž existují také pro kapesní počítače

# didaktik m a didaktik kompakt

Počítač DIDAKTIK KOMPAKT pořídíte zhruba za 7.000,-Kč
nebo DIDAKTIK M pořídíte do 3.000,-Kč
devítijehličkovou tiskárnu zhruba za 2.500 až 6.000,-Kč
disketovou jednotku asi za 4.000 až 5.000 Kč

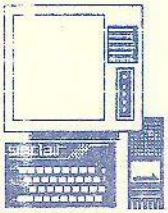
program DENÍK JEDNODUCHÉHO ÚČETNICTVÍ za 190,-Kč

program FORMULÁŘE také za 190,-Kč

**Celkem tedy zhruba za 13.400,- Kč !!!**

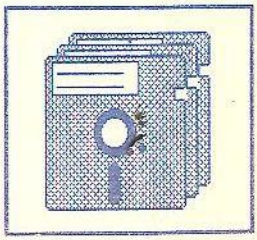
## přemýšlejte o tom

a informujte se na naší adrese: S E C O M
Staré Křečany
407 61



# SECOM

## NABÍDKA PROGRAMŮ A MANUÁLŮ PRO KAZETU A DISKETU 5.25"



Manuály jsou převážně ve formátu pro D-TEXT a pokud jsou pro R-TEXT je označeno zkratkou RTX. Programy a data lze objednat samostatně a i zde platí, že je nutno si poslat vlastní kazetu nebo disketu. Za každé **tři programy** nebo **soubory dat** je nutné přiložit **20,-Kč.**

### SPD9

FORMAT
MORSE 9
HLAS
PŘEVOD
TEST/PROG
CONTO
S-LOADER9
ZNALOSTI
T-MONITOR+
LIE-TEST
BIORYTMUS
TEST/IQ
TEST 7
TEST 1
TEST 2
TEST 3
TEST 4
TEST 5
TEST 6
TEST 8
TEST 9
TEST 10
TEST 11

### SPD

JEDNOTLIVÉ PROGRAMY

K=kazeta
D=disketa

| | | |
|---|---|---|
| R-TEXT/BT | R | |
| DIAGRAMY | R | |
| TAPE COVER | R | |
| TELESEA PB | DR | |
| ARTes48BT | R | |
| KAL-MES | DR | |
| T-MONITOR+ | R | |
| S-LOADER9 | R | |
| D-WRITER | D | |
| VU-FILE | D | |

### MANUÁLY

M6504
BT100
OVLADAČE TISK.
O-COPY
DTEXT35
PASCAL
EDITAS ASSM
MEGA BASIC
DATALOG
LOGO
SUPERCODE
NávodBB9.1 RTX
R-TEXT/BT

### 50.- Kč

### SECOM CODE 1

Obsahuje strojní rutiny a programy, které byly zveřejněny v AP za uplynulé období.

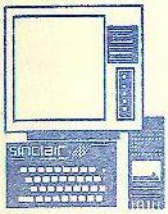
**Nutno zaslat vlastní disketu**

### SOFTWAROVÉ DROBNOSTI

POSTŘEHY
TUTOR
OBRAZOVKA
SOFTWAROVÉ DROBNOSTI

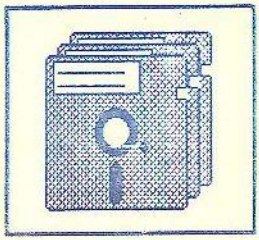
### BRO-SOFT

AIR LINES
C.WORLD
ZX MLUVÍ
SHADES
SLOVNÍK
SPEEDELOAD
zn.sady 8x8p



# SECOM

## NABÍDKA PROGRAMŮ NA DISKETÁCH 5.25"



Opět i v tomto čísle přináším přehled programů, které můžete získat za poplatek 20,-Kč za jednu disketu, kterou si naformátovanou na klasický formát sami zašlete. Přiložte řádně vyplněný objednací lístek, který v tomto čísle přiložen. Pokud některý komplet je i ve verzi pro magnetofon, je označen symbolem magnetofonu a pro disketu symbolem diskety.

PUBLIC DOMAIN

### SPD1

3DGRAF
LINLOM
VRH
VYPTROJ
HYDRLIS
PRINTER
SUPERCODE

### SPD3

SPRITE
12X8
GEN/UDG
GUCH
5xSCREEN$
ARTIST II
R-TEXT/BT

### SPD5

KVADR
PREVOD
PRINT
POPISKA

### SPD6

LOGO
PASCALL
PROLOG
FORTHRAN
MEGA-BASIC
BETA BASIC3

### SPD2

PÍSMA
DLAŇ48K
VAST
OBSAH MP/T
DISASSM(ALFI
TASW/ALFI
ASM80 (ALFI)
2xBT
DIAGRAMY3
SPECTRAMON
SPRITE

### SPD4

VERIFY/BT
VERIFY
AUTOLINE
DEFKEYS
LLIST
HLAVIČKY
PIRÁT
MC2
COLT
RENAMEDISK
TRACE
SUPERENUM
COMPRES
KURZOR
OREM

### SPD7

WRITER
ARTIST II
ARTSTUDIO AW
DTEXT/DaS
RTEXT GMC
TW BT 4GR
DTEXT PRT
DTEXT D10b
DTEXT D100

### SPD8

COMPILER
COMPILER2
HiBasic
MCODER


# AMATÉRSKÝ PROGRAMÁTOR

Soukromý a zcela nezávislý časopis pro amatérské programátory na počítačích ZX SPECTRUM a DIDAKTIK. Vychází každý druhý měsíc v rozsahu 28mi stran. Cena jednoho čísla je 15,-Kč/16,-sK. Předplatné ve výši 132,-Kč nebo 132,-sK lze zajistit na adrese redakce. Sazba a předlohy stránek byly vyrobeny programem DTP MACHINE za podpory utilit SMGCA / FRAMES a programu TM-FOS2. Nevyžádané rukopisy a jiné příspěvky nevracím. Do tohoto čísla přispěli: D.Wenzel, P.Macek, P.Rak, J.Brossmann, Slav. Kondáš,
**Vydává Petr Černý**, Staré Křečany, psč. 407 61.
Tiskne soukromá maloofsetová tiskárna SECOM, St.Křečany

| Adresa redakce | AP-SECOM Staré Křečany | Toto číslo vyšlo v srpnu 1993 |
|---|---|---|