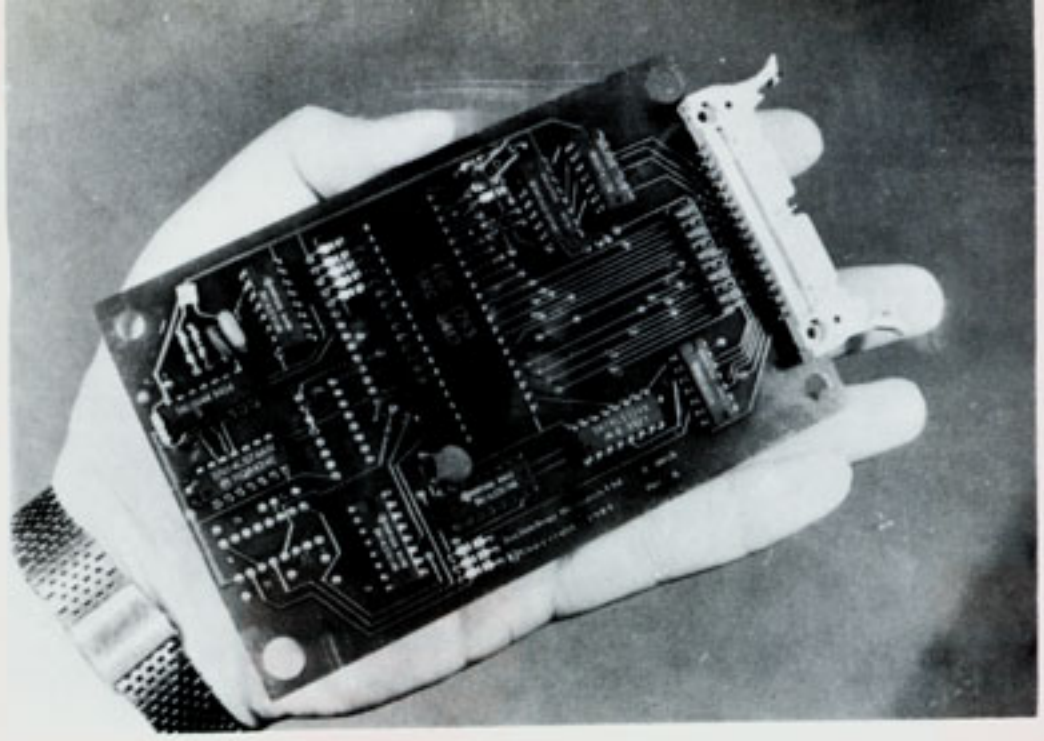
# MIKRO BÁZE

1989 **3**

**technický zpravodaj svazarmu pro zájemce o mikropočítače**

Cena 12 Kčs

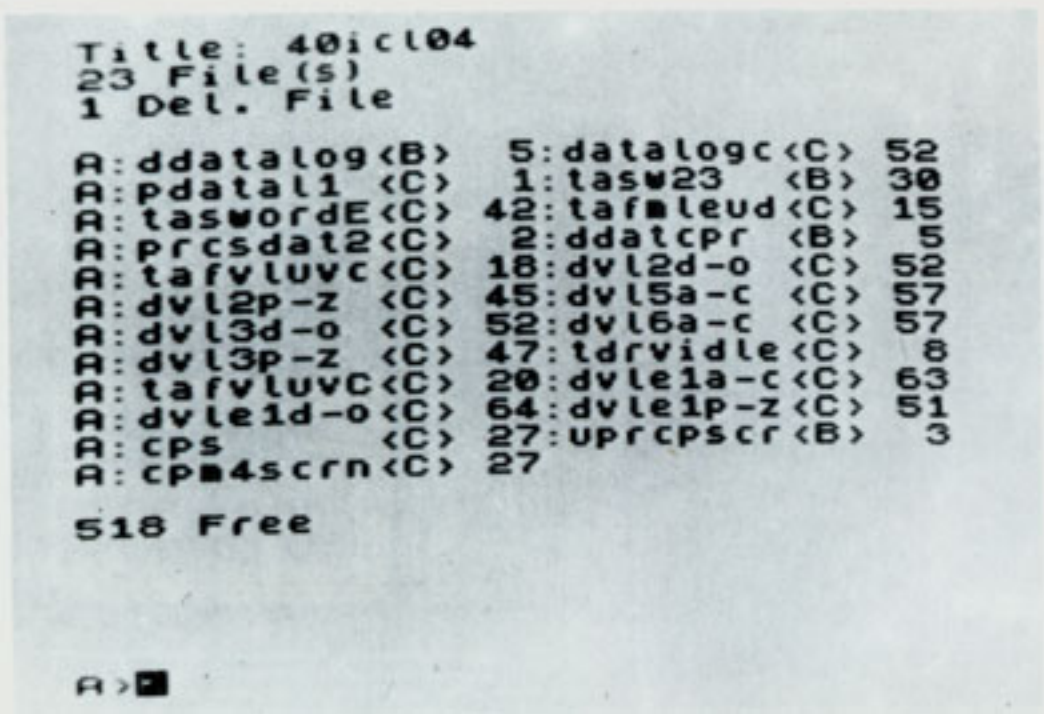



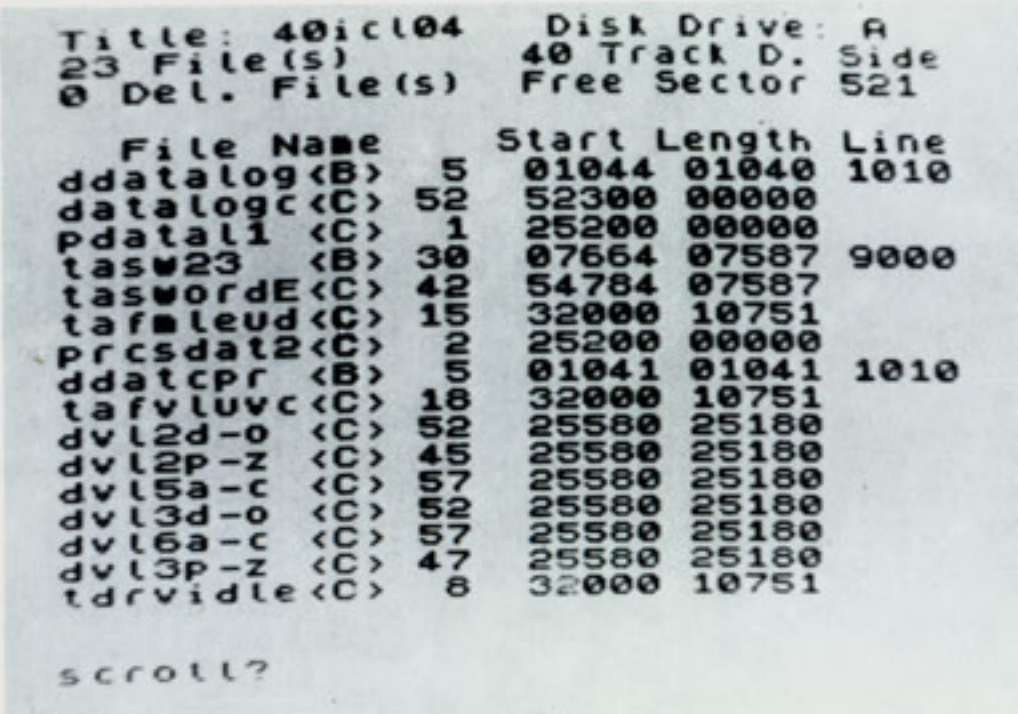
# POSTAVTE SI S NÁMI ŘADIČ FLOPPYDISKU

Po získání disketové mechaniky vyvstane každému další problém – jak ji vlastně připojit k počítači, kde sehnat potřebný řadič. K některým počítačům existují sice firemní konstrukce, ale ne každý si může něco takového dovolit koupit. A tak se většina novopečených majitelů mechanik začne pídit po zapojení vhodného řadiče. Jediný u nás dostupný návod vydala 666. ZO Svazarmu. Používá se zde IO Intel 8272 v celkem běžném zapojení. Je počítáno hlavně s připojením k ZX Spectru, ale neexistuje vlastní diskový operační systém (DOS).

Pro ZX Spectrum existuje zajímavý firemní řadič s vlastním DOSem – Betadisk, od anglické firmy Technology Research. Jeho zapojení je velice jednoduché, přesto se však ukázal v provozu velice spolehlivý. Je použit IO WD 1793. Vestavěný DOS dává řadu zajímavých možností nejen v Basicu, ale i ve strojovém kódu. Krátká rutinka v assembleru, spolu s kouskem Basicu, umožňuje například čtení libovolného formátu diskety, včetně disket z IBM PC. Pomocí služeb DOSu bylo též možno nejjednodušším možným způsobem nainstalovat CP/M. Mimo jiné dovede DOS zobrazit katalog disku (obr. vlevo dole), nebo podrobnou informaci o všech souborech na disku (obr. vpravo dole).

Na obrázku vlevo nahoře je vidět celý přístroj v krabičce. Uvnitř jsou dvě desky – řadič a operační systém. Deska řadiče je použitelná univerzálně, bez ohledu na typ počítače. Obrázek vpravo nahoře vás jistě přesvědčí, že je tato deska tak jednoduchá, že by neměl být problém ji postavit amatérsky. Sympaticky vypadá i to, že zde nejsou žádné nastavovací prvky. Protože mne k tomu nouze dohnala, pokusil jsem se postavit si celý Betadisk. Jak to dopadlo můžete vidět na 3. straně obálky.



1989/3

Mikro báze

technický zpravodaj svazarmu pro zájemce o mikropočítače

Obsah



Technický zpravodaj Svazarmu pro zájemce o mikropočítače. Vydává 602.ZO Svazarmu ve spolupráci s redakcí časopisu Amatérské radio. Povoleno ÚVTEI pod evidenčním číslem 87 007. Zodpovědný redaktor ing. Jan Klabal. Sestavil Ladislav Zajíček.

Grafická úprava Yvetta Dunděrová. Sekretářka redakce Božena Divišová. Redakční rada: ing. Petr Horský, ing. Jan Klabal, ing. Petr Kratochvíl, Josef Kroupa, Daniel Meca, ing. Alois Myslík, ing. Josef Truxa, Ladislav Zajíček. Za původnost a správnost příspěvků ručí autoři. Ročně vyjde 10 čísel.
Cena výtisku 12 Kčs podle ČCÚ a SCÚ č. 1030/202/86. Roční předplatné 120 Kčs. Objednávky přijímá a zpravodaj rozšiřuje 602.ZO Svazarmu, Wintrova 8, 160 41 Praha 6.

SVAZARM

602.ZO

&

Amatérské RADIO

# ŽE BY?

Nevěřil jsem vlastnímu uchu na telefonním sluchátku, když mi v něm šéfredaktor zvěstoval:

"Honoráře za články jsou zvýšeny na rozmezí od 50 do 90 korun za stránku strojopisu. V každém čísle to musí vyjít v průměru na 70 korun."

Někdo by mohl ironicky uznamenat: "Vida, jak málo stačí člověku k radosti." Jistě, skákat radostí nad tím, že místo jedné děravé ponožky mám teď dvě, může vyhlížet poněkud prostoduše. Ale kdo by se aspoň trochu nepotěšil tím, že v Mikrobázi konečně došlo k odskoku z tuhnoucího systému platby 45 Kčs za stránku? I s obrázky se to zlepšilo - rozpětí je od 15 do 500 korun, podle jejich "hustoty".

Otázka Že by? není jen nadějeplnou reakcí na zvýšení honorářů. Je adresována i potenciálním autorům kvalitních materiálů. Že by se konečně začali ucházet o naši přízeň a přetrhli zarputilé mlčení redakčního telefonu?

V Metakomunikacích minulého čísla jsem se zamýšlel nad problémem míry. Uvaha se týkala i mikrobázího poměřování. Mikrobáze se honorářovou úpravou dostala aspoň na spodní úroveň honorářů většiny časopisů. Průměrných 70 korun je o 55 procent víc než bývalých maximálních 45 Kčs. Na otázku, zda to přinese i stejné zvýšení kvality článků, by nedokázala odpovědět ani chytrá horákyně. Jednou z podstatných věcí, jakou úprava přináší, je o něco širší akční rádius redakce v jednání s renomovanějšími autory. Skýtá i možnost zvrácení současného stavu, kdy Mikrobáze skoro nemá co tisknout (překlady se nemohou stát těžištěm naší práce). A po nějaké době si snad dokonce budeme moci vybírat ve prospěch čtenářů. Právě tady lze tušit perspektivu postupného zvyšování kvality obsahu časopisu. Relativní naplnění jedné z logických podmínek pro spuštění cyklu autoři/redakce/čtenáři tomu dává určitou šanci.

Redakce konečně bude moci přistoupit ke slovně nadnesenému "zápasu o autora". Můžeme mu za jeho práci už aspoň něco nabídnout. Nebudeme muset stydlivě klopit oči při poznámce: "Víte, ale je za to jen 45 za stránku..." V odvetu klopí oči autor a tlumeně dí: "Jo, jo, jen kdybych tak někde našel ten čas na psaní..."- obligátní závěr většiny dosavadních jednání (hudebně řečeno - fade away, česky - do ztracena).

Jistě, že pro řadu pracovně vytížených autorů nebude 70, ba ani limitních 90 korun dostatečným stimulem k usednutí za počítač. S jejich znalostmi se na stránkách Mikrobáze teď ještě nesetkáme. Ale mám za to, že pro dost velkou obec schopných (a dosud váhajících) autorů bude tato míra přijatelnou hranicí, za kterou stojí za to k počítači přece jen zasednout.

Karel Velebný říkává s cimrmanovskou logikou: "Nejdůležitější jsou peníze, zdraví pak už přijde samo." Taky se říká - peníze nejsou všechno (říkají ti, co jich mají dost). Jistě, peníze nejsou vším. Ale jsou tím, čím jsou - směnnou hodnotou vynaložené práce. Alespoň mezi námi by to tak mělo být. Chtěl bych věřit, že v Mikrobázi pomalu končí nezdravý fenomém tzv. zájmové činnosti zadarmo. Tím nechci ani v nejmenším podcenit práci, kterou četní aktivisté v této oblasti dělají. Já totiž nechci, aby byli jakkoli podceňováni. Oceněním není sada diplomů zasunutá do prázdné kapsy. Kdo poskytuje smysl počínání druhých, má právo na jejich smysluplnou reflexi. Nutit dávat zdarma i brát zdarma má stejně neblahé důsledky. A netýká se to jen finančních vztahů. Obecně platný fakt, že všechno souvisí se vším, nelze obejít.

Do nového jara vykračujeme "sedmdesátikorunovým krokem". Snad se nám po té o něco zelenější trávě půjde měkčeji. Záleží to i na vás. Mikrobáze chce živě reflektovat aktuální stav výpočetní techniky. Toho nedosáhneme, když nás svým mlčením uzavřete do skleněného zámku. Ba co víc - vy nás nesmíte nechat ani vydechnout! Informujte, nabízejte i lajte... Ale hlavně - pište, pište, pište! Ať máme co zajímavého a původního tisknout. Když budete psát moc, zvýšíme počet tiskových stránek. Ale pozor - o co narostl honorář, o to víc budeme sedět na kvalitě!

...vtom zazvonil telefon... Že by...?

-elzet-

# HOVORY O PROGRAMOVÁNÍ

# NOVÝ DATALOG NA LETECKÝ BENZÍN

Poslední rozhovor s Petrem Adámkem jsem dělal před více než rokem. Tehdy už se počítačovou domovinou šířil jeho Datalog pro ZX Spectrum. Mezitím jsem se Petra několikrát ptal, jestli pokračuje v dalším vývoji své databáze. Vždy jsem dostal kladnou odpověď. Moje dotěrnost slavila úspěch - na podzim se u mě Petr objevil s kazetou v ruce, zasedl ke Spectru a živou demonstraci proložil instruktáží o nové, tryskové verzi Datalogu. Hned mi došlo, že by byla škoda, abych to poslouchal jen já. Zapnul jsem mikrofon a požádal Petra, aby to vzal popořádku.

**"Jaká byla reakce počítačového obecenstva na první verzi Datalogu?"**

"Spousta telefonických dotazů i dopisů. Zpočátku jsem to vítal, protože mě to ujišťovalo o tom, že se Datalog ujímá. Když se písemný dotaz vztahoval ke konkrétnímu použití Datalogu, odpověděl jsem. Nemálo lidí však tuhle cestu chtělo využít zcela jiným směrem. Nemohu odpovídat na dotazy typu "Víte, já mám dělat nějakou knihovnu na počítání s maticemi... Napište mi o tom něco!"

**"To znám. Nevím, kde se v některých lidech bere představa, že mohou na druhém žádat, aby jim věnoval čtvrt roku života. Oni snad ani nevědí, co činí."**

"Ale je i mnoho takových, kterým bych rád odpověděl, jenže nemohu, protože neumějí formulovat svůj dotaz. V jednom Bytu psali o tom, co všechno by měl mít tazatel předem připraveno, aby se dozvěděl, co chce vědět. Prostě jak popsat konkrétní problémovou situaci. Právě při dotazech uživatelů Datalogu jsem pochopil, jak je tohle důležité. Když se tazatele zeptám: "A co máte v ten moment na obrazovce?", tak neví. Pak chce, abych počkal se sluchátkem na uchu, že si Datalog nahraje, pak že nahraje data, pak že se pokusí tu situaci vyvolat atd., atp... Tak takhle tedy určitě ne. Ptát se je potřeba umět."

**"Když tě tak poslouchám, napadá mě řešení vrátit honorář a tak se z těch dotazů vymanit..."**

"Ne že bych o tomhle někdy uvažoval, ale když už jsi to vyslovil, skoro něco na tom je..." (oba se smějeme, anžto to vůbec nemyslíme vážně).

**"Přišly ti nějaké kritické odezvy?"**

"Zpočátku mě zarazilo, že veškerá odezva byla až příliš jásavá. Skoro jsem začal redakci podezírat, že kritiku přede mnou schovává. Nakonec mi přišel jeden dopis, který celý Datalog roztrhal na kusy. Při jeho čtení jsem brzy pochopil, o co jde. Pisatel byl zvyklý na Master File a všechno, čím se Datalog od něj navenek liší, považoval za jeho nedostatek. On prostě žádal, aby Datalog na obrazovce vypadal stejně jako Master File a aby vůbec všechno dělal jako Master File. Psal něco v tom smyslu jako "v Datalogu nikdy nevím, kde jsem, nikdy nevím, co udělat dál" a podobně. V jediném měl pravdu. Asice že první verze Datalogu neumí udělat záhlaví výpisu. To jsem v nové verzi dodělal.

Několik lidí mě upozornilo na nedostatek při tisku na tiskárnu. Datalog mezi jednotlivými obrazovkami vynechával jednu řádku. Všem jsem poslal opravu osmi bajtů, které tohle odstraňují. Samozřejmě, že nová verze to má v pořádku.

Zajímavé je, že jen jeden člověk mi napsal o jedné vyložené chybě programu. Když zadáš výběr a hledáš desetinné číslo na rovnost, pak se Datalog může zachovat zmateně, ba dokonce i zhroutit. Spojil jsem se s pisatelem dopisu a nechal si popsat konkrétní postup, jakým se k této situaci dostal. Díky tomu jsem chybu hned našel a opravil."

**"Máš nějaký přehled o tom, k čemu lidé Datalog používají? Objevily se nějaké opravdu užitečné aplikace?"**

"Až jsem se podivil. Původně jsem předpokládal, že Datalog budou využívat především sběratelé čehokoli. Co jsem třeba pokládal za bujnou fantazii, je aplikace ve skladovém hospodářství. Ale i tady Datalog našel uplatnění. Takhle nahonem si vzpomenu ještě na využití v jednom dabingovém studiu, ve výzkumném ústavu kovů... Zajímavé je, že všichni, kdo používají Datalog k seriózním účelům, požadují ruštinu. Jako jasný příklad mohu uvést třeba jazykový ústav v Bratislavě. Nový Datalog proto ruštinu má. Statisticky vzato - ze všech dopisů, co jsem dostal, jich asi 15 procent mluví o seriózních aplikacích, které jsem ani nepředpokládal."

**"Zpět do historie Datalogu. V době, kdy jsi ho odevzdal, měl jsi pocit, že dílo je hotovo a už se k němu nevrátíš?"**

"No...měl. Datalog jsem udělal podle svého nejlepšího svědomí a k obrazu svému, abych ho mohl používat i já sám. Ale čím déle člověk databázi používá, tím víc přichází na to, co jí chybí, co by stálo zato ještě upravit, vylepšit... Takže jednak já sám jsem dostával spoustu nápadů, co by se ještě dalo přidat, a pak tu byla řada podnětů i od ostatních uživatelů.

V době, kdy jsem program dokončil, mi vůbec nepřipadalo, že by byl nějak pomalý. Nejdelší soubor, jaký jsem v Datalogu měl, byla tenkrát ta Evropa, která se s ním dodává na pásku. Až když jsem paměť naplnil po okraj, zjistil jsem, že třeba listování u konce souboru je hodně pomalé. Zvláštní je, že stížností na tohle bylo minimálně. Lidi zřejmě vycházejí z toho, že na tak malém počítači to holt jinak nejde. Ale ono to jde. Nová verze nejen že má zachovaný komfort obsluhy, ale je ve všem mnohem rychlejší. Při všech úpravách a dodatcích jsem dbal na to, aby nová verze nebyla ani o bajt delší než stará. Tedy aby se všechny soubory vyhotovené ve staré verzi vešly do uživatelské paměti verze nové. Všechny nové příkazy jsou ke staré verzi přidány - původní nejsou nijak změněny. Tak může každý k nové verzi zasednout, aniž by se musel jakkoli "předělávat".

V nové verzi jsem se snažil uplatnit jen opravdu užitečné náměty. Do Datalogu jsem nedal třeba ovládání joystickem, protože je mi líto paměti, kterou by to zabralo."

**"Co je tedy v novém Datalogu nového?"**

"Především už zmíněné zrychlení všech programových funkcí. Samozřejmě včetně třídění."

**"O jaký typ třídění jde?"**

"Programátorskou hantýrkou se mu říká window sort. Není nikde nijak definován. Je to něco jako manipulace s okénky paměti. Upravil jsem algoritmus tohoto třídění a zrychlil jeho prováděcí smyčku na té nejnižší úrovni. Tam jsem ušetřil nějaký ten takt, což ve výsledku přineslo dost podstatné zrychlení. Ještě jsem provedl průzkum, s jakými daty lidi v databázi nejčastěji manipulují. Pak jsem před vlastní řazení umístil rozhodování, v němž privileguji častá data. Jde o optimalizaci rychlosti řazení. Konkrétně - texty a celá čísla se řadí plnou rychlostí. Jen desetinná čísla jsou na tom podobně jako v první verzi. Zrychlení práce s nimi je dáno pouze změnou základního algoritmu. Ještě mnohem rychlejší operace by přineslo použití standardních celých čísel v rozsahu dvou bajtů. Ale rozhodl jsem se použít tři bajty, už proto, že jednou z častých aplikací databáze je telefonní seznam.

Zmíněné rutiny nejsou jednoúčelové, vztahují se i k dalším operacím, jako je třeba listování. Jeho zvýšená rychlost je v kterémkoli místě souboru jednotná.

Dál jsem doplnil funkci výběru. Původní aparát výběru je sice silný, ale až příliš složitý. Dost často se stává, že člověk nepotřebuje vybírat položku, ale potřebuje ji rychle najít. Tak jsem do menu přidal "Najdi". Tenhle příkaz jsem poprvé uplatnil při práci se slovníkem, který pracuje oběma směry. Nic víc si už od toho nepřejuobousměrný rusko-český slovník, kde máš ruštinu v jednom sloupečku, češtinu ve druhém a hledat můžeš v obou.

Další novinkou jsou fragmenty. Začal jsem je používat při sestavování obrázků. V podstatě jde o zkrácený zápis často se opakujících částí pseudografiky nebo textu - něco jako makra. Předem si nadefinuješ určitý fragment a přiřadíš mu písmeno A až Z. Fragment pak můžeš používat pro jeho zobrazení v záznamu. Třeba kdybys měl pokaždé malovat obrázek pouzdra integrovaného obvodu, každý záznam by ti sebral přes 100 bajtů. Když použiješ fragment, v záznamu budeš mít jen čtyři bajty, které se na jeden či více fragmentů odvolávají. Jejich skladbou můžeš zobrazovat obrázky, které jsou si v mnohém podobné a liší se jen v některých částech - což je právě případ různých typů pouzder integrovaných obvodů.

Pro uživatele, kteří dosud používali Master File, mám hotový a ověřený konverzní program, který data Master Filu převádí na formát Datalogu. Jsou tam ošetřeny i takové situace, kdy je zpracovávaný soubor pro operační paměť příliš dlouhý. V takovém případě se rozdělí na dva výsledné. Grafika je pochopitelně nepřevoditelná, každý si ji pak v Datalogu upraví podle svého.

Ze všeho, o čem jsem teď mluvil, považuju za nejpodstatnější zvýšení rychlosti práce Datalogu a kompatibilitu zdola."

**"Proč vlastně nový Datalog ještě není mezi lidmi?"**

"To je právě to, co mi není jasné. Já jsem se v minulém roce dvakrát písemně obracel na 602.ZO a informoval ji o nové verzi Datalogu i o některých svých jiných programech. Porvé to bylo v září. Když nic nepřišlo a nikdo ani nezavolal, nabídku jsem zopakoval a připsal, že když znova neodpovědí, pustím novou verzi Datalogu mezi lidi. Za dva měsíce by ho byla plná republika. Brání mi v tom jedině pomyšlení, že je tu organizace, která mi zaplatila za první Datalog. Kdybych to udělal, neměli by co prodávat."

**"A proč to tedy nenabídneš někam jinam?"**

"Než tady pátrat po někom bůhvíkom a ještě mu to vnucovat, to bych to radši pustil jako všechno předtím. Každý týden se mi ozývá několik zájemců; takových těch tvrdošíjných je asi deset. Ptají se, jak to s tím je, jestli už to někde bude k mání a přemlouvají mne, abych jim to dal, ba dokonce prodal, což už vůbec nemíním. No a já jim každý týden říkám, že se to ještě nevyjasnilo a pořád čekám, kdy se konečně někdo od 602.ZO ozve. Za vrchol všeho považuju, že mi vůbec neodpovídají. Kdyby řekli ne, tak ne. Když ano, fajn. Ale oni jednoduše mlčí!"

**"O tempora, o morales... Vraťme se ke skutečné práci. Jaký je tvůj další osobní vývoj coby programátora?"**

"Odpověď je jednoduchá - IBM PC, bez něhož se profesionální software dnes nedá dělat."

**"Na Spectru si pořádný programátor bez assembleru neškrtne. Jak se díváš na vztah assembler-péčéčko?"**

"To není to nejlepší. Taková cesta je dokonce skoro neschůdná. Assembler se ve vztahu k péčéčku konečně dostává do pozice, jaká mu náleží - je to strojový jazyk a jako takový se využívá pro psaní operačního systému."

**"Ale Unix je psán v Céčku."**

"To ano, jenže původně byl psán pro velké počítače s vysokou rychlostí a velkou pamětí. To už se netýká třeba MS-DOSu. Ale bavme se o obecném programování. Když programátor přejde z malého počítače na péčéčko a bude se na něm snažit programovat assemblerem, je možné, že za rok, za dva vytvoří hvězdu. Jenže svět za stejnou dobu vyprodukuje stovky programů, které sice nebudou tak zářit, ale posunou celou oblast mnohem víc dopředu. Tady je cena za assembler veliká. A kdyby v něm programovali všichni, byl by dnes svět o pár let zpátky.

Program napsaný ve vyšším jazyku není optimální, je delší, pomalejší - ale je. Vše teď značnou měrou závisí na pokroku v oblasti hardwaru. Čím bude rychlejší, čím rozsáhlejší budou paměti počítačů, tím méně se projeví nevýhody programů napsaných ve vyšších jazycích.

Co nás vlastně nutilo psát na Spectru v assembleru? To, že je v něm pevný operační systém, ve kterém chybí spousta funkcí. Když se podíváš na MS-DOS, tam je všechno hotové. Pro ilustraci: Jeden můj přítel přešel ze Spectra na péčéčko. Když jsem se ho ptal, co na péčéčku dělá, odpověděl: "Nic. Tam už je všechno hotový." Považoval jsem to za přehnané, ale čím dál víc mu dávám zapravdu. U péčéčka se konečně můžeme zabývat využitím počítače, tedy aplikačními programy. Nemusíme se zabývat tvorbou programů pro to, aby ten počítač vůbec něco uměl - čili nepsat systémy (to už za nás udělali jiní) a začít psát užitečné aplikační programy. Tady je použití assembleru raritou.

Navíc, jazyk C (můj favorit) neklade žádná omezení z hlediska možností využití počítače a bez jakékoli výjimky dovoluje použít libovolnou jeho funkci tak, jako v assembleru. Rozdíl je jen v optimálnosti kódu, v jeho délce a rychlosti - to je nutně horší. Ale Céčko mi nezabraňuje využít i to, co se v jiných jazycích musí nutně napsat v assembleru. Céčko dovoluje úplně vše. Dnes dokonce i jazyky, které tradičně nic takového nedovolovaly, jsou vybavovány možnostmi, jaké má Céčko. Tak třeba nové verze Pascalu mají rozšíření, která jsou jeho duchu zcela cizí. Když porovnáme poslední dva produkty firmy Borland - Turbo C 2.0 a Turbo Pascal 5.0 - je na nich vidět, že vycházejí z jednoho. Spoustu rutin na té nejnižší úrovni mají společných, jsou poskládány z modulů, společných oběma jazykům. Přirozený vývoj nakonec přinutil Pascal, aby se vzdálil své matematické čistotě, a Céčko se čím dál víc blíží pascalovské pohodlnosti. Firmy dnes do kompilátorů Céčka zasazují taková makra či takové předdefinované skupiny znaků, případně operace, aby se tím programátor už nemusel zabývat. Tak se jazyky sobě blíží, jdou k jakémusi optimálnímu středu.

A ještě jedna zajímavost. Především na větších počítačích, ale týká se to i péčéček, přetrvává

situace, kdy se pořád nejvíc používají knihovny psané ve Fortranu. Je v tom určitý paradox. Tak když potřebuješ něco udělat, nejspolehlivější a nejjednodušší je sáhnout po 30 let starém Fortranu. Ne snad že bys to v něm napsal, ale použiješ hotové rutiny, které jsou za těch 30 let dokonale ověřeny praxí, a slinkuješ je se svým programem. Největší roli v tom hraje prvořadý požadavek-spolehlivost. Sice není výjimkou, že i po deseti letech provozu se objeví chyba, ale 30 let už je přece jen dost dlouhá doba na to, aby se všechny blechy vychytaly. A pak je tu samozřejmě ušetřený čas - rutiny už jsou hotové.

Hele, máš ještě něco? Já totiž jedu do práce..."

**"Cože? Teď večer? A v sobotu?"**

"Vždyť říkám, že jsme blázni. To je jako droga. Tak co?"

**"Ještě chvilku. Kdyby se ti poštěstilo mít doma péčéčko, hodil bys Spectrum do koše?"**

"To rozhodně ne..."

**"Nemaluješ si to moc?"**

"Nemaluju. Mám syna... Totiž - na to, aby se někdo mohl zabývat programováním, musí mít svůj počítač. Zdůrazňuju - svůj! Ne že ho má tatínek nebo Svazarm. Musí mít svůj. Nejen na sobě, ale i u známých jsem si ověřil, že tu jde o kvalitativní i kvantitativní rozdíl. Teprve když počítač člověku opravdu patří, začne fungovat."

**"Jako že k němu má libovolný přístup?"**

"Nejde jen o přístup, ale o vztah. Když není jeho, dělá všechno tak nějak do vzduchu. Teprve když mu patří, začne na něm dělat doopravdy."

**"Ptám se znova - až budeš mít doma péčéčko, budeš dál dělat i se Spectrem a pro Spectrum?"**

"Já jsem takových přechodů už pár prožil. Vždycky jsem byl přesvědčen, že po příchodu lepšího počítače využiju i ten starý. Třeba že z nich udělám tandem, že na tom novém budu vyvíjet programy pro ten starší a podobně. Ale nikdy k ničemu takovému nedošlo. I teď mám podobné cukání-sháním péčéčkové kompilátory pro Z80. Mám však obavu, že to zase skončí stejně. Zůstane u úmyslu a já budu sedět u péčéčka. Ale co zcela určitě udělám, je přenos dat z Datalogu do databází péčéčka."

**"Jaké jazyky používáš na péčéčku v práci?"**

"Téměř výhradně Céčko, pro jeho univerzálnost. Už na jiných větších počítačích se mi mockrát stalo, že jsem si říkal - tohle je jednoduchá úloha, pohodlně a rychle ji napíšu třeba v Pascalu. Takřka bez výjimky jsem se pak dostal do stádia, kdy mě Pascal omezoval. Třeba co jsem támhle ušetřil, tady musím složitě obcházet, tady zase linkovat assembler... Po předchozím zdánlivém ulehčení práce vždycky přišel moment nějaké ztráty. Kdyby šlo o programy, které bych si dělal sám pro sebe, nejspíš by nic takového nenastalo. Ale když děláš aplikační programy pro druhé, vždy se na tebe obrátí s tím, aby to tady a onde dělalo trošku něco jiného a podobně. Pak musím provést změnu a tam vždycky narazím. Vznikají neohrabanosti a zdržovačky, které nakonec odplaví všechen původně ušetřený čas. Z tohoto hlediska je naprosto jisté a bezpečné Céčko, kde k něčemu podobnému dojít nemůže. I když přiznávám, že je méně pohodlné (nemyslím tím obtížnější) než Pascal.

Céčko vítězí všude ve světě. Jeden můj známý se teď vrátil ze zahraničí. Když se tam ptal, jak programují, řekli mu: "To je jednoduché - amatéři programují v Pascalu, profesionálové v Céčku." To není žádný snobismus, ale nutnost."

**Rozmlouval -elzet-**

# DŘU, DŘEŠ, DŘEME ... CÉČKO (3)

**Třídy pamětí alias třídy objektů**
**aneb**
**různé způsoby existence dat**

V závěru minulé části seriálu jsme se poprvé setkali s pojmem globální proměnná. Pro připomenutí:

```
int jáglobál;
main ()
{int jálokál;
...}
```

Deklarované číslo jáglobál má globální působnost. Může se na něj odvolávat jakákoli další funkce programu. Číslo lok má působnost lokální, místní, vnitřní. Existuje jen uvnitř funkce v rozsahu od nejbližší levé svorky až po jí odpovídající pravou. Přiřazená hodnota proměnné existuje až od momentu její inicializace. Např.:

```
main()
{int lok1; /* deklarace lok1 */
 lok=1;    /* inicializace lok1 */
...
{int lok2; /* deklarace lok2 */
 lok=2;    /* inicializace lok2 */
...}       /* konec působnosti lok2 */
...}       /* konec působnosti lok1 */
```

Jak už tomu bývá, i tady jsou implementační rozdílnosti. Např. v Hisoftu C pro ZX Spectrum lze lokální proměnné deklarovat jen na začátku těla funkce.

Běžné lokální proměnné se v Céčku označují jménem auto (zkr. automatic). Jejich deklarace (označení typu, názvu proměnné a třídy paměti) proběhne při prvním průběhu funkcí "jednou provždy". Ale inicializují se vždy znova při každém zavolání funkce, jejíž součástí jsou, a za hranicí své působnosti svou hodnotu zase ztrácejí. Jinými slovy - při inicializaci je automatické lokální proměnné vyhrazena paměť pro uložení její hodnoty na zásobník proměnných této třídy. Po opuštění funkce je tato paměť volně k dispozici dalším akcím programu. Existence i působnost automatických proměnných jsou tedy výrazně ohraničeny.

Při deklaraci automatické lokální proměnné by měl být uveden název třídy paměti auto. Abychom však z opakovaného psaní slova auto nedostali písařskou křeč (a zbytečně nezačmárali text i paměť), Céčko dovoluje tuto identifikaci vynechat (zamlčet). Pak je ovšem každá deklarace v těle

funkce bez uvedení třídy paměti považována za automatickou. Ale neuděláte chybu, když název auto uvedete (např. auto int lok;). Někdy je takové označení dobré pro orientaci v delším textu.

Chceme-li se ve funkci odvolat na globální proměnnou, která je vůči tělu funkce proměnnou vnější, musíme v deklaraci uvést název **extern** (zkr. external - vnější):

```
int glob;
main()
{extern int glob;
...}
```

Pokud bychom zde název extern nepoužili, byla by deklarována automatická lokální proměnná, zcela nezávislá na stejnojmenné globální proměnné.

Při tvorbě delších programů je zdrojový text na paměťovém médiu rozdělen do více částí, připojujeme k němu již hotové moduly, vše různě kombinujeme, spojujeme atd. Odvolávka extern si dokáže najít "tu svou" proměnnou v kterékoli části textu, v níž byla deklarována. Samozřejmým předpokladem úspěšnosti nálezu je, že spojovaný text někde takovou deklaraci obsahuje. Někdy ovšem nebudeme chtít, aby se odvolávka vztáhla na globální proměnnou v jiné části textu, než v té, kde je deklarována. Jde o zabránění vzniku vztahu k ev. stejnojmenné proměnné v jiné části textu. Pak při deklaraci globální proměnné použijeme slůvko **static** (statická):

```
static int jáglobál;
main()
{...}
```

Tak zajistíme platnost globální proměnné jen v rozsahu dané části textu. V případě těchto tří částí (souborů) textu:

```
/* část 1 */          |  /* část 2 */
static int glob;      |  int glob;
funkce1()             |  funkce3()
...                   |  ...
funkce2()             |  funkce4()
...                   |  ...
-------------------------------------------
/* část 3 */          |
funkce5()             |
...                   |
```

se bude každá odvolávka **extern int glob** v části 1 vztahovat jen k číslu glob v této části textu. Stejná odvolávka v části 2 i v části 3 jen ke stejnojmennému číslu deklarovanému v části 2.

Když při deklaraci globální proměnné přiřadíme hodnotu, nemusíme se ve funkci již na tuto proměnnou odvolávat, můžeme s ní pracovat přímo:

```
int glob=5;
main()
{printf("%d",  glob);}
```

Výsledný výpis bude 5.

Slůvko static má ještě jednu významnou funkci. Když je použijeme při deklaraci lokální proměnné uvnitř funkce, bude se taková proměnná chovat zcela odlišně, než je tomu v případě lokální proměnné automatické. **Statická lokální svou hodnotu** poté, co programové řízení opustí "její" funkci, **neztratí!** Pro tuto svou vlastnost se často používá jako čítač různě opakovaných průběhů. Pro úplnost - u statické proměnné se při její deklaraci inicializuje i ukazatel adresy jejího uložení v paměti. Oproti tomu uložení automatických proměnných v jejich zásobníku je určeno ofsetem. Způsob manipulace s globálními proměnnými je závislý na konkrétní implementaci kompilátoru. Většinou se ukládají do statické paměti.

Abychom se mohli blíže podívat i na další odlišnosti chování obou druhů lokálních proměnných, rozšíříme si naše znalosti.

**Podmínka IF ELSE**
**a navrch != == -- ++**

Bezpochyby každý zná basicové sousloví IF THEN. Některé Basicy pracují i se slůvkem ELSE (česky jinak). Céčko z tohoto tria vynechává slovo THEN, protože je v jeho syntaxi zbytečné. Základní schéma podmínky:

```
if(výraz)
  příkazA;
else
  příkazB;
```

Když je výraz v závorce pravdivý (různý od nuly), provede se jen příkazA. Jinak (else) se provede jen příkazB. V obou případech pak program pokračuje dál za příkazemB, pokud mu samotný příkaz nevelí jinak.

Samozřejmě můžeme v programu používat i jen samotné IF:

```
if(čítač!=0) return (1);
if(čítač==0) return (0);
```

Basicovým fandům možná pomůže, když pomyslně vydechnou své THEN před příkazy return. Obě řádky můžeme přepsat i do tvaru:

```
if(čítač!=0) return (1); /* != je operátor */
else return (0);         /* nerovnosti */
```

Mimochodem - tady by místo druhé řádky stačilo samotné return (0), protože s jinou, než nulovou hodnotou k němu proměnná čítač už nedorazí.

Právě jsme narazili na dvě céčkové specialitky. První != není nic jiného, než osobitý zápis operátoru nerovnosti (<>). Ke druhé - dvěma rovnítkům vedle sebe - si řekneme něco více. Symbolicky můžeme == nahradit dotazem: Je rovno?, resp. Rovná se?. Zatímco jedno rovnítko vždy znamená přiřazení hodnoty. V řádce:

```
if(čítač==0) return (0); /* == je operátor */
                         /* rovnosti */
```

bude výsledek výrazu za if pravdivý, když bude čítač nulový. Pak se provede i příkaz return (0). Kdybychom však omylem napsali:

```
if(čítač=0) return (0);
```

bude mít výraz hodnotu přiřazenou, tedy nulovou. Výraz bude nepravdivý a příkaz return (0) se neprovede.

A na druhou stranu - přiřazení jakékoli nenulové hodnoty činí výraz pravdivým děj se co děj. Proto **pozor na rozdíl mezi = a ==!**

Podmínky IF ELSE můžeme libovolně řetězit, např:

```
if(a==4) b=44;
else if(a==3) b=33;
else if(a==2) b=22;
else if(a==1) b=11;
else b=0;
return b;
```

Jakmile je podmínka některého výrazu splněna, následné else zabrání dalším testům a programové řízení přejde na příkaz return b. Kdybychom dali pod sebe jen podmínky if (bez else), všechny testy by se postupně provedly (což ale někdy může mít svůj smysl).

Dost často se stává, že za podmínkou příkazu IF

se nám nahromadí víc příkazů. Je-li jich více než jeden, umístíme je mezi svorky, do tzv. bloku:

```
if(výraz)
  {příkaz1;
   příkaz2;
   ...}
```

V případě pravivosti výrazu se provedou všechny příkazy umístěné v bloku. Totéž platí pro else.

Než se pustíme do zkoumání rozdílů v chování automatické a statické lokální proměnné, zásobíme se ještě jednou specialitou céčkového psaní:

```
++prom; --prom;  /* prefix */
```

Basicově vyjádřeno: LET prom=prom+1, resp. LET prom=prom-1. Nebo assemblerově INC prom, resp. DEC prom. Jde tedy o zvětšení, resp. zmenšení obsahu proměnné o 1. Tím ale zvláštnost tohoto zápisu nekončí. Je tu i možnost:

```
prom++; prom--;  /* postfix */
```

Rozdíl je v momentu provedení zvětšení/zmenšení obsahu proměnné. Když jsou znaménka před ní, změní se její obsah před provedením operace, jíž se proměnná účastní. Když jsou znaménka za ní, změní se její obsah až po provedení operace. Např.:

```
main()
{printf("%d",proměnná1();
 printf(", %d",proměnná2();}

proměnná1()
{int prom; prom=1;
 if(prom--) return prom;
 return (100+prom);}

proměnná2()
{int prom; prom=1;
 if(--prom) return prom;
 return (200+prom);}
```

Výsledný výpis:

```
0, 200
```

Z toho je patrno, že ve funkci **proměnná1()** byl výraz (prom--) nenulový, ale přesto je vrácená hodnota proměnné nulová. To proto, že se zmenšila o 1 až po testu výrazu. Ve druhém případě byl testovaný výraz nulový, proto se příkaz return prom; neprovedl. Proto při použití tohoto typu zvětšování a zmenšování obsahu proměnné musíme dávat pozor na umístění znamének. Využití prefixu či postfixu má svůj význam hlavně v čítání průběhů smyčky.

Teď už k našim proměnným

**a k tomu ještě || && !**

```
main()
{
int a;
static int b=5;
a=5;
printf("%d, %d,",--a,--b);
if(!b||!a) return;    /* ! je NOT, || je OR */
main();
}
```

Napřed k jedné zásadní novince - závěrečným příkazem **main();** funkce volá sebe samu. Takovou rekurzi lze v Céčku velmi snadno vytvořit. Jakákoli funkce může volat samu sebe. Rekurze má uplatnění v matematických výpočtech, při třídění dat atd. U rekurze ovšem nesmíme zapomenout na vložení nějaké splnitelné podmínky, která nás ze smyčky vyvede. Jinak bychom se z ní už nedostali.

V řádce s poznámkou je zvláštní céčkový zápis logických operátorů. V Céčku jsou celkem tři:

```
&&    AND
||    OR
!     NOT
```

Tyto operátory mají stejnou funkci jako v Basicu, tedy nic zvláštního:

```
(výraz1 && výraz2 && výraz3... && výrazN)
```

Celý výraz v závorkách je pravdivý jen tehdy, jsou-li pravdivé všechny v něm uvedené výrazy 1..N.

```
(výraz1 || výraz2 || výraz3... || výrazN)
```

Celý výraz v závorkách je pravdivý jen tehdy, je-li pravdivý alespoň jeden z výrazů 1..N.

```
!výraz
```

To je negace výrazu - pravdivý se změní v nepravdivý a naopak.

Příkaz return v posledním programu bude proveden, jen když výraz (!b||!a) bude pravdivý. To nastane tehdy, když buď b nebo a nabyde nulové hodnoty. Celý řádek bychom mohli o něco srozumitelněji přepsat do tvaru if(b==0||a==0). Příkaz return nás vrátí tam, odkud byla funkce **main()** volána.

Jaký je výpis diskutovaného programu?

```
4 4,4 3,4 2,4 1,4 0
```

Tak máme potvrzeno, že automatická proměnná se inicializuje vždy znova, zatímco statická jen poprvé. Kdybychom v příkazu printf místo --a,--b napsali a--,b--, dostaneme výpis:

```
5 5,5 4,5 3,5 2,5 1
```

To dokazuje, že při prefixu se hodnota proměnné mění ihned, kdežto při postfixu až po skončení operace, jíž se proměnná účastní.

Použití rekurze v uvedeném případě je sice hezké, ale zdaleka ne vždy můžeme provedení smyček řešit tak rozmáchle. Prostě - je nejvyšší čas poznat

**Cyklus WHILE**

Jeho základní schéma:

```
while(výraz)
  příkaz;
```

Připojený příkaz se opakovaně provádí, dokud je výraz pravdivý. Výraz samozřejmě obsahuje určitou podmínku, která se v průběhu opakovaného provádění příkazu nakonec stane nepravdivou. Pak se smyčka zakončí a program pokračuje dál "za nosem". Výraz pochopitelně může být nepravdivý už při vstupu do smyčky. V takovém případě se příkaz neprovede ani jednou.

Podobně jako u IF, i u WHILE může být připojených příkazů víc. Pak je musíme uzavřít svorkami do bloku:

```
while(výraz)
  {příkaz1;
   příkaz2;
   ...}
```

Teď přepíšeme rekurzní program nerekurzně s použitím cyklu WHILE:

```
main()
{int a; a=5
while(a--) test();}  /* dokud a!=0, volej funkci
                                        test() */
test()
{static int x=5;
int y; y=5;
printf("%d %d,",--y,--x);}
```

Výsledný výpis bude naprosto stejný jako v předchozím programu (adekvátně i při změně prefixu na postfix). Ovšem přibyl nám tu čítač a. I když jde o automatickou proměnnou, neztrácí svou hodnotu, protože funkce main() se neuzavřela. Ta se ostatně uzavírá jako vůbec poslední.

Malá poznámka k inicializacím. Některé kompilátory Céčka dovolují inicializovat automatickou proměnnou hned při její deklaraci, např. int a=5. Hisoft C pro ZX Spectrum ale nikoli.

Zásadně si pamatujte, že statickou proměnnou musíte inicializovat hned při deklaraci. Pokud byste napsali třeba:

```
static int x;  /* nedobře */
x=5;
```

pak by proměnná x nabývala hodnoty 5 vždy znova a veškerý půvab by byl ten tam.

A ještě něco z počítačovštiny - název proměnné, pole a vůbec se nazývá identifikátor. To proto, aby se architekti super programů domluvili mezi sebou i sami se sebou. Upozorňuji na to pro případ, že byste se náhodou vyskytli ve vyšší datistické společnosti (každá má svůj bonton).

Předposlední třídou paměti, potažmo objektů, je register. Označení této třídy v zápisu je např.:

```
register int a;
```

Musím hned předeslat, že u osmibitových počítačů se tato třída zpravidla nevyskytuje. Důvodem jsou přetížené registry procesoru. Takováto proměnná se umisťuje do uvolněného registru, takže operace s její účastí pak probíhají rychleji. Proměnných této třídy lze efektivně deklarovat jen pár. Pokud na některou žádný registr nevybyde, Céčko s ní zachází jako s běžnou proměnnou. O omezeních při práci s registrovými proměnnými se dozvíte z programového manuálu.

Zcela poslední třídou objektů je typedef. Vzhledem k tomu, že svým pojetím spadá do pokročilejší látky, necháme si ho "na pak".

Ale abyste se necítili sebemíň ošizeni, hned přistoupíme k základům komunikace s počítačem prostřednictvím klávesnice.

## Funkce getchar() a putchar()

jsou standardními knihovními funkcemi. Getchar() odebírá jeden znak z aktuálního vstupního souboru a putchar() vkládá znak do aktuálního výstupního souboru. S jednou podobnou funkcí jste se už seznámili - printf() posílá formátovaný výstup na obrazovku. Funkce getchar() a putchar() standardně slouží pro odebrání znaku z klávesnice a jeho výstup na obrazovku a do programu. Je tu však jeden zásadní "zádrhel". Některé kompilátory ke čtení klávesnice a současnému výstupu přečteného znaku na obrazovku i do programu používají jen getchar() (např. Hisoft ZX Spectra nebo Turbo C pro péčéčko). U jiných getchar() dělá jen část práce - znaky se ukládají napřed do bufferu a teprve putchar() je odtamtud po stisku Enteru apod. pošle na obrazovku.

A dále - Hisoft C má getchar() i putchar() v základním programu. Většina jiných kompilátorů má obě funkce v knihovně zvané stdio.h (stdio znamená standard input/output).

Pro bufferované přečtení znaku z klávesnice, jeho odebrání z bufferu a vypsání na obrazovku je potřebný takovýto program:

```
#include <stdio.h>
main()
{char c;
c=getchar();
putchar(c);}
```

Tělo funkce můžeme napsat i "céčkovatěji":

```
{putchar(getchar());}
```

V první řádce je povel pro načtení knihovního modulu, který obsahuje obě požadované funkce. Hisoftu C nebo Turbo C stačí:

```
main()
{char c;
c=getchar();}
```

Na každý pád je třeba se podívat do manuálu konkrétního kompilátoru, jak pracuje s klávesnicí. V dalším textu budu používat "spectrovskou" formu (jednak Specter je u nás zkameněle historicky nejvíc a nakonec není žádný problém zápis doplnit podle individuálních potřeb).

Ve výpisu se objevila nová proměnná char (zkr. character - znak). Je to znaková proměnná o délce jednoho bajtu (číselná int má dva). Další rozdíl je v možnostech inicializace. Když budeme chtít proměnné přidělit hodnotu ASCII kódu 65 (písmeno A), můžeme tak učinit dvěma způsoby:

```
int a; a=65;      nebo:      char a; a='A';
```

Ale i ve druhém případě bude faktickým obsahem proměnné b číslo 65. To už napovídá, že tu není žádný zvláštní problém konverze. V určité míře jde o to, co se nám v kterém momentu hodí, s čím se nám bude lépe psát. Tak třeba výše uvedený program můžeme přepsat na práci s číselnou proměnnou:

```
main()
{int c;
c=getchar();}
```

Samotným příkazem putchar (ch) či putchar (c) (kde ch je znaková proměnná, c číselná) můžeme poslat na obrazovku jakýkoli jeden znak (tedy i řídící kód). Např.:

```
putchar(32)          neboli          putchar(' ')
putchar(`\n`)
putchar('\007')      apod.
```

Dobré kompilátory (zvláště v implementaci s Unixem) umožňují přepínat vstupy a výstupy mezi jednotlivými soubory. Tuto symfonii si mohou movitější majitelé přehrávat podle svých mnohasetstránkových manuálů.

Před uvedením závěrečného programu ještě malý syntaktický krůček. Již víme, že okamžité zvětšení proměnné o 1 můžeme napsat ve tvaru:

```
++prom
```

Totéž můžeme napsat i zcela obligátně:

```
prom=prom+1      ale i      prom+=1
```

Zvláštní? Porovnejte následující a vyjasní se:

```
prom=prom*5        totéž, co        prom*=5
b=b/(a-c)            "      "       b/=a-c
kýbl=kýbl*hadr       "      "       kýbl*=hadr
abc=abc-*(px++)      "      "       abc-=*(px++)
```

Jde opět o zkrácení a zpřehlednění zápisu. A jde

i o zvyk. Schématicky jsou obě další řádky shodné (op znamená operátor):

```
výraz1 = (výraz1) op (výraz2)
výraz1 op= výraz2
```

Operátorem zde může být: + - * / % << >> & ^ |.

K zatím neznámým operátorům se dostaneme příště. Uvedený zápis je velmi výhodný zvláště při použití dlouhých, komplikovaných výrazů.

V posledním programu minulé části jste se seznámili s definicí číselného pole. Dnes přibyde znakové. Opět je berte tak, jak je servírováno. Všemu kolem polí se ještě budeme velmi podrobně věnovat. Teď je nejdůležitější, abyste si na něčem ověřili učební látku této části.

Program určuje název dne v týdnu podle zadaného data (číslo dne a číslo měsíce) v roce 1989.

```
int rok89[]={(31),(28),(31),(30),(31),(30),
             (31),(31),(30),(31),(30),(31)};
char *týden[]={"sobota","neděle","pondělí",
               "úterý","středa","čtvrtek",
               "pátek",};
main()
{int b,c,čdne,čměsíce;b=0;
printf("\nUveďte číslo dne: ");
if((čdne=klávesnice())==-1)
  return;
printf("Uveďte číslo měsíce: ");
if((čměsíce=klávesnice())==-1)
  return;
if(čměsíce<1||čměsíce>12)
  {chyba();return;}
if(čdne<1||čdne>rok89[--čměsíce])
  {chyba();return;}
while(čměsíce>0)
  {c=rok89[--čměsíce];
   b+=c;}
b+= čdne;
if(b>6)
  {c=b/7;b-=c*7;}
printf("\n%s",týden[b]);}

klávesnice()
{char s[2];
int c,i;i=0;
while((c=getchar())!='\n')
   {if(c==' ') return(-1);
   if(c<'0'||c>'9'||i>1)
     {chyba();return(-1);}
   s[i++]=c;}
if(i==1) return(s[0]-'0');
  else return(10*(s[0]-'0')+s[1]-'0');}

chyba()
{printf("\nChyba v zadání\n");}
```

Když třeba chceme zjistit, jaký den připadne na štědrý večer, na obrazovce proběhne dialog:

```
Uveďte číslo dne: 24
Uveďte číslo měsíce: 12

neděle
```

(Bohužel se to týká i Silvestra.)

Pole **rok89** obsahuje počty dnů v jednotlivých měsících. Pole **týden** začíná sobotou jakožto nultým prvkem (alias sobota 31.12.88). Neděle je prvek[1], protože 1.1.89 neděle byla. V řádce:

```
if((čdne=klávesnice())==-1)
```

je nejdříve zavolána funkce **klávesnice()**. Po návratu je číslo dne přiřazeno proměnné **čdne**. To se při špatném zadání rovná -1 (viz příkazy **return(-1)** ve funkci **klávesnice()**). Nastane-li tento nevítaný případ, provede se následující return a řízení programu se vrátí tam, odkud byl volán (skončí). Totéž je o řádku dál pro **čměsíce**.

V testu:

```
if(čdne<1||čdne>rok89[--čměsíce])
```

je zjištěno, zda číslo dne není menší než 1 nebo zda není větší než nejvyšší datum zadaného měsíce. Proměnná **čměsíce** tu hraje roli pořadového čísla elementu v poli **rok89**. Elementy se počítají od nuly nahoru, proto je v testu **--čměsíce**.

V dalším cyklu **while** se postupně sečítají dny všech celých měsíců směrem k lednu a příkaz **b+= čdne** k výsledku připočte zadané číslo dne. Po vydělení sumy dnů sedmi se vypočte zůstatek a v poli **týden** se najde název dne. V odvolávce na pole **týden[b]** se přímo určí název dne v týdnu.

Ve funkci **klávesnice()** je deklarováno znakové pole se dvěma prvky. Cyklus:

```
while((c=getchar())!='\n')
```

přijímá znaky z klávesnice, dokud není stisknut Enter. Ten má kód symbolizovaný znakem n - new line. Jako každému řídícímu kódu mu předchází nerozlučný \ escape. Protože jde o **znakové vyjádření**, je **mezi apostrofy**. Escape jen upozorňuje, že za ním bude řídící kód. Výsledkem tohoto složeného vyjádření bude jen samotný řídící kód.

První řádka bloku cyklu zjišťuje, zda nebyla stisknuta mezera. Zařadil jsem ji tam pro případ, že by někdo mermomocí chtěl dát program do smyčky (aby z ní bylo kudy "vypadnout").

Druhá řádka testuje, zda je zadaný znak číslem a zda není činěn pokus zadat více prvků pole s než dva. Když není zadáno číslo nebo je zadáván třetí prvek, zavolá se funkce **chyba()** a následuje návrat s hodnotou -1. U polí musíme vždy dbát na to, abychom je "nepřefoukli". Jejich dimenzi zadáváme buď inicializací jejich prvků, nebo uvedením počtů prvků v hranatých závorkách. Tím poli zároveň vyhrazujeme rozsah paměti. Když při cyklickém či jiném obsazování pole jeho dimenzi přetáhneme, zcela jistě tak zajedeme někam, kam vjíždět neradno. A jak to končívá, víme.

Čtvrtá řádka bloku přiřazuje hodnotu znaku prvku pole **s** (nejdříve prvku **s[0]**, pak ev. ještě prvku **s[i++]**, tedy **s[1]**).

Když je zadání v pořádku, pak se při jednom zadaném znaku vrací hodnota čísla jen z jednoho prvku **s[0]**. Při páru zadaných znaků je nutno **s[0]** vynásobit deseti a přičíst jednotky z **s[1]**. Protože **main()** nepracuje s ASCII kódy čísel, je nutno od ASCII kódů v rozsahu 48..57 odečíst ASCII kód nuly ('0'), abychom dostali "čistá" čísla v rozsahu 0..9.

V příkazu printf na konci main() je novinka-formát **%s** (zkr. string) pro výpis řetězců (více o něm příště).

Když budete chtít ozkoušet program třeba na péčéčku v Turbo C, na úplně první řádku musíte dát **#include <stdio.h>**. Při sjetí programu v Turbu C jsem narazil na zvláštní štěnici. V Hisoftu C jsem nemusel inicializovat proměnnou b - automaticky jí byla přiřazena nula. Nikoli tak v Turbu C, kde nabývala nesmyslných hodnot. Proto je v programu inicializována nulou.

Nepochybuji, že se mezi vámi najdou - v dobrém smyslu slova - ti praví datističtí fanatici, kterým to nedá, a rozvinou program pro určení názvu dne od roku raz-dva až do doby, kdy zítra znamená včera nebo jak je to. Každopádně byste si program měli natukat do počítače a zkusit si s ním pohrát. A nějaký si vymyslete a pošlete - ať už pro otisknutí nebo inspiraci.

Příště nás čekají zbývající dva cykly do while a for, všechny formáty tisku, notný zbytek operátorů, návrat k hrátkám s ukazateli a obvyklých pár programků. Céčkových.

-elzet-

# FILOSOFICKÉ ASPEKTY STROJOVÉHO MYŠLENÍ (6)

**Tarského věta**

Mluvíme o pravdivosti a nepravdivosti. Co kdybychom se pokusili, tak jako jsme zformalizovali pojem dokazatelnosti, zformalizovat i pojem pravdy - konec konců, je to přece vlastnost sentencí (resp. jejich čísel).

Nechť True(x) je nějaká formule, řekněme v PA, která vyjadřuje pravdu: True(a) je pravdivá tehdy a jen tehdy, je-li a číslo pravdivé formule. Nechť T je formule analogická G, ale s True místo Tm (T má ovšem i jinou Tetu). T říká: "T není pravdivá". Čili "Lžu!" - Epimenidův paradox v plné síle.

Důsledek: Předpoklad, že něco jako True existuje, padá. Pravdu nelze ani vyjádřit formálně, tím méně ji lze reprezentovat, t.j. klást rovnítko mezi odhalování pravdy a dokazování v systému PA.

Obr.17. Teta si nese sebe sama (dle Steinbergra)

**Kouzlo autoreference**

"Nejsem teorém!" prozrazuje sama na sebe Gödelova sentence. Co slovo, to autoreference! Každý si spíše všimne té, která je vyjádřena zájmenem "nejsem" - kvůli ní se ostatně říká, že Gödelova sentence je autoreferenční. Trik, který je přitom použit, (PA nemá osobní nebo ukazovací zájmena) připomíná způsob autoreference, se kterým jsme se setkali již v odstavci "kouzlo autoreference" (příklad 8.) - Hofstadter jej nazývá Quineova metoda, čili quinování. Tentokrát má podobu

"Připojeno za citaci sebe sama není teorém"
připojeno za citaci sebe sama není teorém

Věta bez podmětu (uvnitř uvozovek) je vlastně Teta, táž věta včetně uvozovek odpovídá Gödelovu číslu Tety a celá věta pak Tetě použité na své vlastní číslo - tedy Gödelově sentenci.

Je tu však ještě druhá autoreference, méně nápadná, ale zato důležitější. Je obsažena ve slově "teorém" - mělo by totiž správně být "teorém Peanovy aritmetiky" nebo obecněji "teorém systému S". Gödelovou sentencí totiž hovoří celý formální systém o sobě samém. Systém je autoreferenční už tím, že si dovede číslovat své formule a důkazy a prostřednictvím těchto čísel o nich mluvit. Pak jako by Gödelovou sentencí o sobě říkal:

"Tuto sentenci nedovedu dokázat."

nebo

"Právě na této mnou nedokazatelné sentenci si můžete demonstrovat moji neúplnost."

Můžeme však jít ještě dál, překročit hranice formálního systému a představit si, že o sobě mluví vlastně sám Kurt Gödel:

"Právě na této v S nedokazatelné sentenci demonstruji, že dovedu dokázat neúplnost S."

(Anebo snad ústy Gödelovými o sobě mluví samotná věda matematická?)

**Autoreference ve formálním jazyce**

Nenápadně jsme se dotkli důležitého tématu: autoreference ve formálním jazyce má jiný charakter než v přirozeném jazyce. Každá vyslovená věta přirozeného jazyka je svým způsobem autoreferenční, nikoli však k sobě, ale ke svému mluvčímu. Při užívání přirozeného jazyka se jeho sémantika a syntaxe (v širším slova smyslu, tj. jednak co a jednak jak se něco říká) vzájemně prolínají; je to tím, že jazyk může výpovídat i o sobě, a tím, že na smysl věty má vliv i to, jak, kdy a v jakém kontextu byla vyřčena (mimochodem, je 'kontext' syntaktická nebo sémantická kategorie?). Pokusy vybudovat logickou sémantiku jazyka, jakkoli důmyslné a jakkoli užitečné, vždy více či méně jazyk znehybňují a formalizují, zbavujíce jej tak jeho přirozené sebetranscedence.

Formální jazyk logiky a matematiky se podstatně liší od přirozeného jazyka: za prvé odpadá subjekt mluvčího a status sdělovacích aktů. Platí tudíž zákon identity výroků, místo genidentity má výrok

logickou identitu, tj. je stále plně totožný sám se sebou. Za druhé existuje jasné odlišení syntaxe a sémantiky (už jsme o tom mluvili). Sémantika je přitom fixována, je jednoznačná a neexistuje významová zpětná vazba (význam závislý na tématu).

Autoreference ve formálním jazyce má pak ovšem jiný charakter. Autoreferenční výrok, nemaje jiného mluvčího, bere zodpovědnost za svůj obsah na sebe, stává se kvázisubjektem. Přitom však se tím, co říká, nemůže zpětně měnit - podléhá své logické identitě, nezná čas. Situace může vyústit i do logického sporu. Totéž se může stát i výrokům přirozeného jazyka, pokud je jim odebrán, byť jen částečně, jejich přirozený status řečových aktů. Logický spor má v takových případech podobu kouzelných aporií a sémantických paradoxů.

Autoreference v této podobě na nás dělá někdy dojem čistě destruktivní (paradox lháře), jindy zase jakoby produkovala nové poznatky: pravdivost nějakého výroku se "vynucuje", a to někdy i tam, kde nevyplývá z podstaty věci (obr. 18; srov. též úvodní test).

Rozdíl je obecnější. Některé typy autoreference totiž snižují počet logicky přípustných řešení (někdy až na nulu), jiné je zvyšují. Ve formálních systémech lze ty první použít v důkazech sporem- pokud najdeme (a to správně) ten předpoklad, který je možno změnit. To je případ Gödelovy sentence. Jako příklad opačný uveďme inverzi G' ke Gödelově sentenci: "Jsem teorém." G' není negace sentence G: aby G' byla autoreferenční, musí vyjít z negace Tety. (¬G je "G je teorém", kdežto G' je "G' je teorém". Mimochodem: negace výroků v první osobě je problematická; je negace výroku "Mám rýmu" výrok "Nemám rýmu" anebo "Ten, kdo řekl 'Mám rýmu' nemá rýmu"? Logická negace totiž předpokládá svět, kde platí logická identita.) Nyní, je-li G' dokazatelná, je pravdivá, není-li dokazatelná, není pravdivá - obě možnosti projdou a my musíme skutečně hledat důkaz. (Problém, zda G' je dokazatelná, položil v padesátých letech Henkin a rozřešil jej kladně Löb: G' je dokazatelná. Löbův důkaz významně ovlivnil studium autoreference v jazyce.) Důkaz Gödelovy věty je vlastně důkaz "vynucením" (jako na obr. 18), a i když zde je to formálně v pořádku (nikde nebyly překročeny hranice formálního systému ani jeho jazyka), rádi bychom věděli, zda autoreference je v zásadě jedinou metodou, jak dokázat např. neúplnost aritmetiky. Nedávno bylo ukázáno, že nikoli. Existují sentence, které mají konkrétní matematický význam a o nichž lze dokázat jinak než autoreferenčním trikem, že ač pravdivé, nejsou v PA dokazatelné (výsledek Parise a Harringtona).

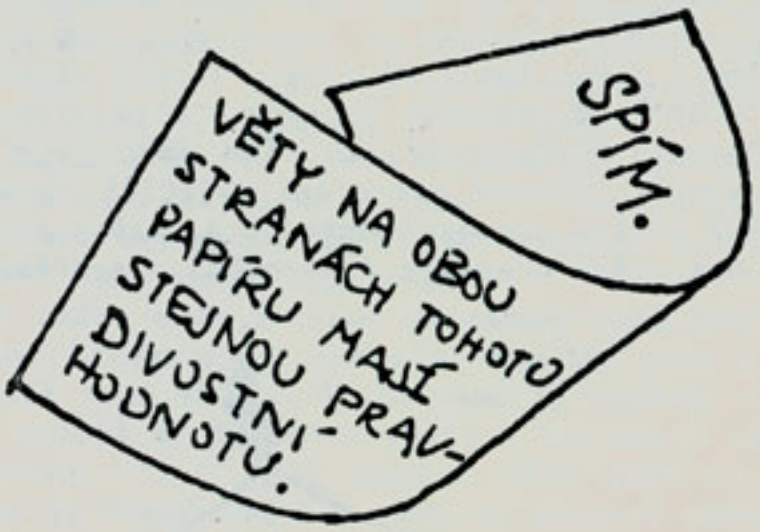
## Formální systém a počítač

Na jednoduchém formálním systému, jako byl náš neinterpretovaný DIA-systém, jsme si uvědomili, že jde vlastně o stroj. Charakterizován konečností a striktně vymezenými pravidly odvozování má každý (neinterpretovaný) formální systém charakter mechanismu, který svým chováním realizuje odvozování. Předpokladem ovšem je, že u takového stroje je dobře rozlišeno to důležité pro tuto funkci od jeho ostatních projevů (od toho, že se třeba třese, vypouští páru a stříká olej). A naopak: máme-li stroj a zaměříme se na určité stránky jeho chování (charakterizované konečným počtem prvků, vlastností a pravidel změny), lze toto chování reprezentovat symbolicky, tužkou na papíře, jako odvozování v nějakém formálním systému. Předpoklad určitého omezeného nedeterminismu stroje (připuštění konečného počtu alternativ- podobně jako to umožňuje formální systém) je zde namístě a nemění nic na jeho "mechaničnosti".

Pokud je příslušný formální systém interpretovaný, musíme, chceme-li rozvíjet paralelu mezi ním a příslušným strojem, použít téže interpretace pro prvky a stavy stroje jako pro odpovídající symboly a formule formálního systému. Teprve za takového předpokladu (který ovšem vyžaduje vnějšího pozorovatele) má smysl přenášet některé obecné vlastnosti formálního systému (jako např. úplnost) na stroj.

Počítač není nic jiného než velice složitý stroj. Počítačem zde ovšem rozumíme počítač plus program. Pro naše účely je dokonce vhodné dohodnout se na určitém (dostatečně univerzálním) počítači a rozlišovat pak už jen programy, které tak budou tvořit jakousi speciální kategorii strojů. Opět tedy můžeme srovnávat programy a formální systémy. Navíc je zřejmé, že každý formální systém lze takto realizovat programem- např. takovým, který manipuluje se symboly a tiskne to, co bychom my psali tužkou na papíře.

Jak by vypadal takový program pro Peanovu aritmetiku? Programátor by se mohl rozhodnout buď realizovat operace s formulemi jako manipulaci s řetězy anebo jako aritmetické operace s Gödelovými čísly formulí (přechod jedné realizace do druhé by byl realizován celkem snadno konstruovatelným




Obr.18. Silné hypnotikum

kompilátorem). Program sám by nebyl náročný: stačilo by, kdyby uměl rozpoznat axiómy PA a aplikovat inferenční pravidla. Pak by mohl třeba generovat všechny důkazy současně tím, že by vždy provedl všechny alternativy odvození.

Peanovu aritmetiku (a kterýkoli jiný formální systém v rámci predikátové logiky) by bylo možno pomocí jednoduché modifikace zmíněného programu reprezentovat též programem odpovídajícím na otázky. Takový program by měl jeden vstup. Kdybychom na tento vstup vložili sentenci PA (nebo její číslo) jako "otázku", program by odpověděl "ano" (výstup 1) v případě, že vstupní sentence je teorémem PA, nebo "ne" (výstup 0) v případě, že její negace je teorémem PA. Výpočet by ovšem nemusel nikdy skončit v případě, že sentence není ani teorémem ani negací teorému.

Takovému programu budeme říkat rozhodovací program (zde pro PA).

## Výpočty, které nekončí

Rozhodovacího programu lze použít i ke speciálním účelům. Představme si např., že A je formule PA s jednou volnou proměnnou y. Tuto formuli lze reprezentovat rozhodovacím programem, který pro daný vstup (nyní budeme předpokládat jen číselné vstupy) dá výstup 1 ("ano") právě tehdy, je-li sentence, která vznikne dosazením vstupního čísla do formule A (za y), teorémem, a výstup 0 ("ne"), když je negace této sentence teorémem. Formule A má své Gödelovo číslo a protože popsaný rozhodovací program na ní jednoznačně závisí, budeme tímto číslem označovat i tento program. Označme $\varphi_x$ funkci (z přirozených čísel do $\{0,1\}$), realizovanou programem s číslem x.

Vzpomeňme si na Tetu Gödelovy sentence. Je to formule s jednou volnou proměnnou a její rozhodovací program provede vlastně toto: vstupní číslo y jednak zkompiluje na příslušný program pro $\varphi_y$ a jednak je použije jako vstupní hodnotu pro $\varphi_y$. Pak simuluje výpočet $\varphi_y(y)$. Vyjde-li $\varphi_y(y) = 1$, dá Teta výstup 0 (Teta, jako formule, popírá existenci takového případu).

Obdobně lze vyrobit i Antitetu, která při $\varphi_y(y) = 0$ dá výstup 1. Nyní můžeme provést syntézu Tety a Antitety, totiž Syntetu. Synteta je opět program (řekněme s číslem s) a počítá parciální funkci $\varphi_s$:

$$\varphi_s(y) = \begin{cases} 0 \text{ když } \varphi_y(y) = 1 \\ 1 \text{ když } \varphi_y(y) = 0 \end{cases}$$

(nedefinováno jinak). Co se stane, dáme-li Syntetě na vstup její vlastní číslo? Máme

$$\varphi_s(s) = \begin{cases} 0 \text{ když } \varphi_s(s) = 1 \\ 1 \text{ když } \varphi_s(s) = 0 \end{cases}$$

takže zbývá jediná možnost: nikdy neskončit. Takto vypadá užití kouzla autoreference v programování. (Popsaná konstrukce je v podstatě důkazem nerozhodnutelnosti problému zastavení Turingova stroje a základem řady dalších důkazů teorie vyčíslitelnosti).

## Pauza

V tomto místě nechť si čtenář opět na chvíli oddechne. Přejdeme totiž zvolna na problémy, které se budou týkat i jeho samotného. Budeme se ptát, zda je či není schopen určitých duševních výkonů, které jsou počítači nedostupné.

## Počítač jako model mozku

Jedním z motivů, kterými je veden jeden směr výzkumu v oblasti umělé inteligence (budeme pro ni používat vžitou anglickou zkratku AI), je vyzkoušet počítač - onen nejdůmyslnější vynález dvacátého století - jako možný nástroj modelování lidské duševní činnosti. Co tato, snad až příliš obecná formulace, pokrývá?

Každé konání, ať už fyzické či duševní, má řadu aspektů: co se koná, jak se to koná, kdo je konatelem, jaký je konatelův vztah ke konání (např. otázka motivace) apod. Řekneme-li např. 'počítač násobí', neřekli jsme mnoho, protože jsme neuvedli, které aspekty uvažujeme a na které z nich klademe důraz. V případě násobení ovšem nevznikají nedorozumění, protože se lze opřít o běžný způsob chápání slova 'násobit' s důrazem na ono "co", teprve potom na "jak" (a to se míní jen postup, nikoli např. rychlost), ale tím to končí. Nikdo nebude argumentovat, že počítač nemůže násobit, protože násobení je přece schopnost člověka, zatímco stroj jen generuje součiny.

Situace se mění, když řekneme 'počítač myslí' (či z opatrnosti: 'počítač "myslí"'). Řada autorů (např. /19/) jasně odlišuje počítačovou simulaci duševních výkonů od jejich počítačového modelování. V prvním případě nám jde o "co", tedy o vnější vzhled a výsledek, v druhém též o "jak" - modelování postupů, metod, principů.

I kdybychom se shodli, že počítačem simulovat myšlení lze, ba dokonce, že je lze i modelovat, ještě by to neznamenalo, že počítač může myslet. V případě mentálních pojmů jako myšlení, vědění, sebeuvědomování, touha, vztek apod. jsou tři možnosti, jak se na tento problém dívat.

První z nich je mezi teoretiky AI nejběžnější (viz např. /11/) a lze ji považovat za pohled funkcionalismu (blíže o něm později). Tvrdí se, že mentální stavy a procedury je možno popisovat čistě přes jejich vzájemné vztahy, bez potřeby brát v úvahu, jaký "hardware" je realizuje. Na skákání míče a na skákání klokana je něco společného, totiž skákání.

Druhá možnost - zdůrazňovaná Hofstadterem - je vlastně zpřesněním funkcionalismu. Spočívá v hierarchizaci úrovní popisu, na kterých daný jev studujeme. Dá-li se na určité úrovni nalézt nějaký "hrubý" izomorfismus zkoumaných dějů, jsme oprávněni říci, že počítač i člověk realizují tutéž činnost vzhledem k dané úrovni a úrovním vyšším.

Rádi bychom zde upozornili na třetí možnost, která je ještě obecnější a spočívá v rozvinutí metaforického pohledu a metody abdukce. Řekneme-li, že počítač myslí, chceme především říci, že "myslí" (uvozovky zdůrazňují metaforičnost tohoto pojmu) a tím i naznačit, že je tu cosi společného (s myšlením člověka), ale ovšem též mnoho odlišného. To nemá být jen opatrný alibismus. Užiji-li metaforu, vyzývám tím nejen sebe, ale i druhé, aby se snažili vytušit to společné cosi, které může být nečekanou shodou stejně jako hlubokou souvislostí, izomorfismem na některé deskripční úrovni stejně jako nějakou zvláštností táhnoucí se skrze všechny úrovně.

""Nejsem teorém!" prozrazuje sama na sebe Gödelova sentence" - tohle je příklad metaforického použití vazby 'prozrazuje sama na sebe', kde to "cosi" společného s touž vazbou užívanou lidmi není nic méně než samovztažnost, zde v podobě autoreference sentence formálního systému. Kouzlo metafory však není jen v tom, že poukazuje na podobnost, ale současně i v tom, že intenzívně připomíná rozdílnost. Nejedno nedorozumění v interpretaci výsledků AI by se vyřešilo, kdyby obě tyto stránky byly vždy reflektovány.

Za výrazný příklad metaforického přenosu- dokonce abdukce - některých náhledů na kognitivní funkce člověka do počítačové podoby lze považovat Minského rámcovou koncepci reprezentace znalostí /12/, často vzpomínanou v literatuře z AI (pravděpodobně zde sehrálo svou roli i introspektivní poznání, např. u schématu tzv. "default assignments"). Přenos je možný i v opačném směru, od počítače k člověku. Příkladem může být model motivace Slomana a Crouchera /21/ (viz o ní též /136/), založený na abdukci metody paralelního programování pro vytvoření modelu motivace člověka (určeného ovšem k realizaci opět na počítači). Symbióza AI a kognitivní psychologie je založena právě na abdukcích od počítače k člověku. Takto lze pohlížet i na tzv. výpočtové modelování (v kontrapozici k modelování pomocí počítače - srov. /6/).

**(pokračování)**

**Ivan M.Havel, Petr Hájek**

---

Skončilo období, kdy roční tempo růstu odbytu osobních počítačů bylo 50-procentní. Podle výsledků konzultantské firmy IDC má do roku 1991 v západoevropských zemích tento růst klesnout na pouhých 8 procent ročně. Osobní počítače instalované v domácnostech i na pracovištích představují 60 procent z celkového prodeje počítačů. U prodeje profesionálních počítačů se očekávají roční růsty kolem 20 procent pro oblast finanční a organizačně-podnikovou; pro vědecké výpočty to má být o něco méně. Na tomto profesionálním sektoru se mikropočítače v roce 1985 podílely 68 procenty z celkového objemu prodeje počítačů. Tento podíl má v roce 1991 vzrůst dokonce na 72 procent. V roce 1991 bude trh z 54 procent patřit 32-bitovým počítačům typu IBM PC/RT, u 16-bitových počítačů (jako např. IBM PC/XT/AT) bude tempo odbytu vysoké, počítá se s 36 procenty. V roce 1985 bylo více než 93 procent prodaných počítačů v hodnotě vyšší než $1500.

(Jiří Kaplan)

# ROZŠÍŘENÍ PAMĚTI ZX SPECTRA (DOKONČENÍ)

**ad 9) Stručný popis modulu BIOS**
**(verze s pružnými disky 4.X - SINSOFT)**

Tento modul se využije v případě implementace OS CP/M na ZX-Sp 80K s řadičem pružných disků, popsaným v kapitole 8.

Modul BIOS, který můžete obdržet, když na naši adresu zašlete disketu, je základem implementace OS CP/M 2.2 na tomto počítači. Zajišťuje základní funkce zbývající části systému. Je složen v podstatě ze tří relativně samostatných částí: obsluhy klávesnice, zobrazovače a diskových jednotek.

### 1.1 Klávesnice

Klávesnice je navržena tak, aby vyhověla i 'gumovému' Spectru. Základním nedostatkem jeho klávesnice je absence CTRL-kódu (až na malé výjimky). Klávesu CTRL v této verzi nahrazuje EXTENDED MODE. Jeho aktivace se projeví změnou barvy borderu. Ta signalizuje, že kód dalšího tlačítka bude vyhodnocen jako řídící. Při stisknutém tlačítku dochází k periodickému generování řídícího kódu. K normální funkci se klávesnice vrátí po uvolnění všech kláves. Tak lze generovat libovolné řídící kódy v rozsahu 00-1FH i bez použití CTRL:

| tlačítka | řídící kód |
|---|---|
| ENTER | 0DH (CR) |
| SS+ENTER | 1BH (ESC) |
| CS+SPACE | .S |
| EDIT | .C |
| TRUE VIDEO | .. |
| INVERSE VID. | .- |
| SIPKY | příslušné řídící kódy směru |
| GRAPHICS | .I (TAB) |
| DELETE | 7FH (DEL) |

Některá tlačítka mají vnitřní funkci v BIOSu a nevracejí žádný kód:

| tlačítka | funkce |
|---|---|
| CAPS LOCK | vyplývá z označení |
| SS+Q | Návrat do Sinclair Basicu |
| SS+W | nucený restart BIOSu (studený start) |
| SS+E | pozastavení programu s čekáním na stisk ENTERu |
| SS+I | vypnutí/zapnutí akustické signalizace klávesnice |

Všechny tyto funkce se provedou okamžitě, bez ohledu na to, zda program žádá vstup z klávesnice. Jejich neuvážené použití (zejména SS+W) může natropit mnoho škod, ale mnohdy zachrání jinak již zcela zhroucený systém. Kombinace SS+E byla implementována pro pozastavení výpisu či jinou činnost u programů, které to normálně neumožňují. Po dobu aktivace této služby systém zcela stojí včetně zablokovaného přerušení, takže ani kurzor nebliká.

### 1.2 Zobrazovač

Pro BIOS byl zvolen formát 64 znaků ve 24 řádcích. Je použita částečně přepracovaná znaková sada z programu THE WRITER, která se vyznačuje příjemným vzhledem a dobrou čitelností. Rutiny však jsou nové a vyznačují se relativně velkou rychlostí výpisu. Byla implementována základní sada řídících znaků, která sice není vybavena některými možnostmi (např. INSERT/DELETE LINE), ale všem programům s rozšířenou činností plně vyhovuje (např. Turbo, VEDIT, WS):

| znak | mnemonika | význam |
|---|---|---|
| 07H | BELL | akustická signalizace |
| 08H | BS | kurzor vlevo |
| 0AH | LF | nový řádek/kurzor dolů |
| 0DH | CR | kurzor na začátek řádku |
| 18H | RIGHT | kurzor vpravo |
| 1AH | UP | kurzor nahoru |
| 1BH | LEAD-IN | adresace kurzoru (následující 2 znaky reprezentují binárně číslo sloupce a řádku s kurzorem) |
| 1DH | HOME | kurzor na pozici 0,0 |
| 1EH | EREOL | výmaz od kurzoru do konce řádku |
| 1FH | EREOS | výmaz od kurzoru do konce obrazovky |
| 0EH | SETATT | nastavení aktuálních atributů (následující znak představuje novou hodnotu atributu) |

Je samozřejmé, že atributy nemohou překročit své fyzické hranice, takže je možné rozlišení minimálně dvou znaků. Pro aplikaci funkce HIGHLIGHT doporučuji užívat (při standardním zobrazení bílých písmen na černém pozadí) sekvence 0E47H, pro návrat k normálnímu jasu 0E07H.

Ještě ve verzi 4.X se počítá s implementací českých znaků. Tato verze bude dodávána samostatně spolu s českým textovým editorem.

### 1.3 Diskové periférie

V současné době existuje mnoho verzí BIOSu pro různé kombinace mechanik, formáty záznamu atd. V zásadě platí, že BIOS může obsloužit až 8 diskových zařízení A-H. Zařízení A-G jsou vyhrazena pro pružné disky, zařízení H je ramdisk. V základním stavu je obsazen pouze určitý počet logických zařízení, závislý na počáteční konfiguraci BIOSu. Nové zařízení, formát atd. lze do systému zavést systémovým programem DRINST. Všechny diskové periférie lze navzájem zaměňovat a měnit jejich přiřazení systémovým programem DRIVE. Ten umožňuje libovolnou logickou jednotku zablokovat, odblokovat, zduplikovat do jiné, nebo zaměnit dvě mezi sebou (tyto programy spolu s krátkým popisem budou dodány na disketě). Běžné jsou různé druhy formátů 5.25" SS/DS, DD, 40 i 80 stop s různými délkami sektoru. Jsou navrženy tak, aby odpovídaly různým typům mikropočítačů (Robotron, Slušovice). U "firemního" formátu SINSOFT je na disketě DD

5.25" kapacita stopy 5,5K. BIOS 4.0 zatím neumožňuje spolupráci s disketami 8" DD nebo 5,25" HD kvůli obvodu DMA, s nímž se v BIOSu nepočítá. Maximální kapacita diskety je 900K, u jiných formátů se pohybuje od 170K do 820K.

Standardní součástí BIOSu je i handler pro obsluhu ramdisku 64K. Handler provádí i výpočet kontrolního součtu, takže při použití ramdisku je stále kontrolována správnost jeho obsahu. Když BIOS při svém studeném startu zjistí, že ramdisk není instalován, nebude se na něj již odvolávat a zařízení H se stane nedostupným.

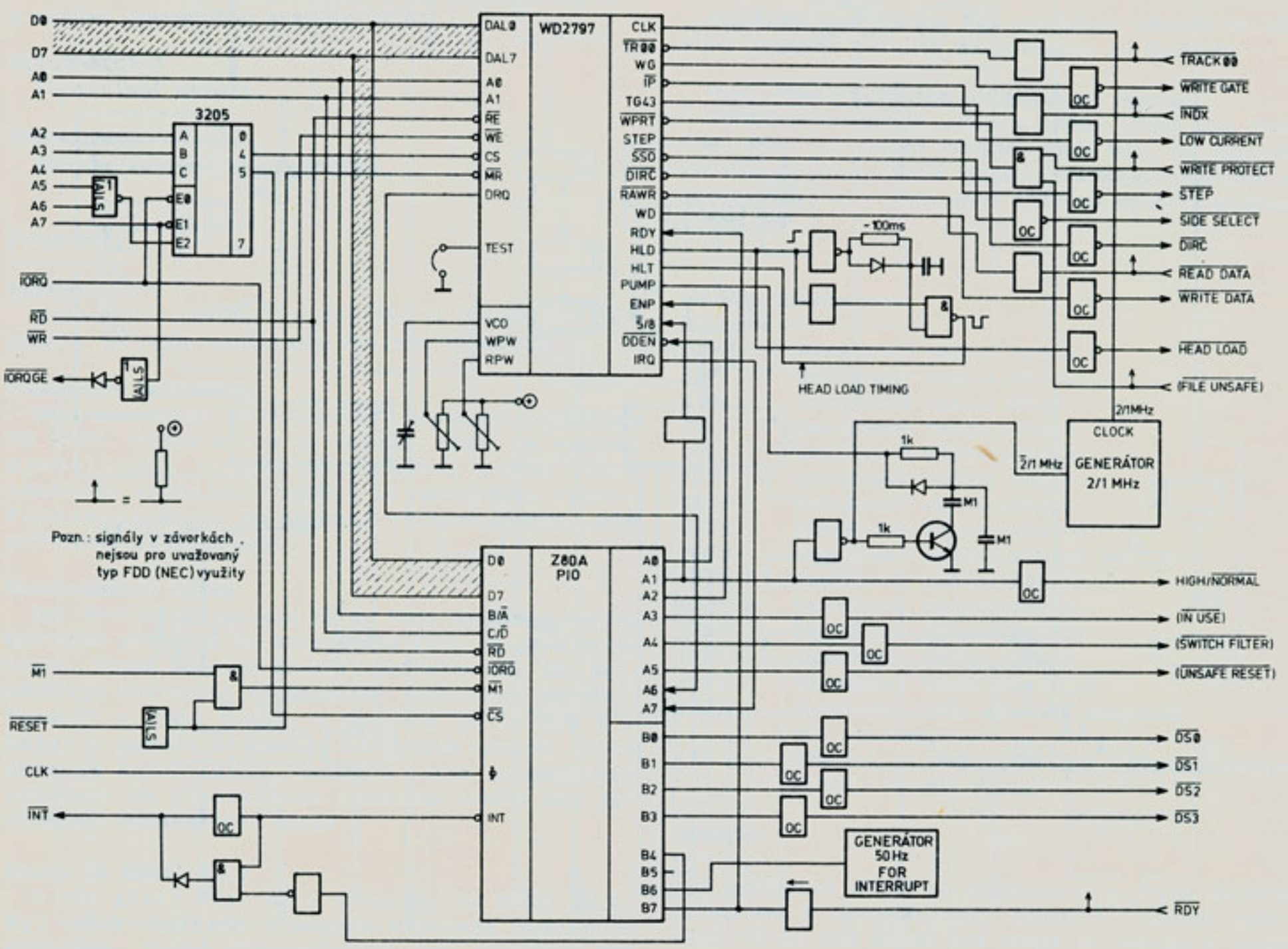
## 2. Ostatní vlastnosti

Začátek BIOSu leží na adrese EA00H, což odpovídá konvencím CP/M o kapacitě 60K.

BIOS má definován standardní IOBYTE, jak jej používá CP/M 2.2. Kromě výše popsaných znakových periférií ovládá ještě tiskárnu (driver tiskárny může být instalován dle požadavků uživatele) a zařízení PUNCH a READER, která však nejsou implementována. Detaily o instalaci těchto zařízení budou dodány v souboru spolu se systémem na disketě.

## 3.Zavedení systému

Modul BIOS je zásadně zaváděn do paměti ze systémových stop na disku systémovým zaváděcím programem BOOT (u disku 8" je uložen ve zvláštním souboru na nejvnitřnějších stopách disku). Tento program slouží k zavedení libovolného operačního systému, neboť pouze prečte 1. sektor 0. stopy na disku (v libovolném formátu i délce) a předá mu řízení. Pod kontrolou takto načteného programu proběhne načtení zbytku příslušného OS do paměti a jeho spuštění. BOOT sám bývá uložen v paměti ROM a po splnění své funkce se odepíná. Tato varianta byla zvolena i u ZX-Sp 80K. Protože konzole, realizující kromě jiných i tuto funkci, nebyla dosud zdokumentována, je k dispozici program BOOT i na kazetě. Po zavedení a spuštění tohoto programu je systém z disku načten už automaticky.



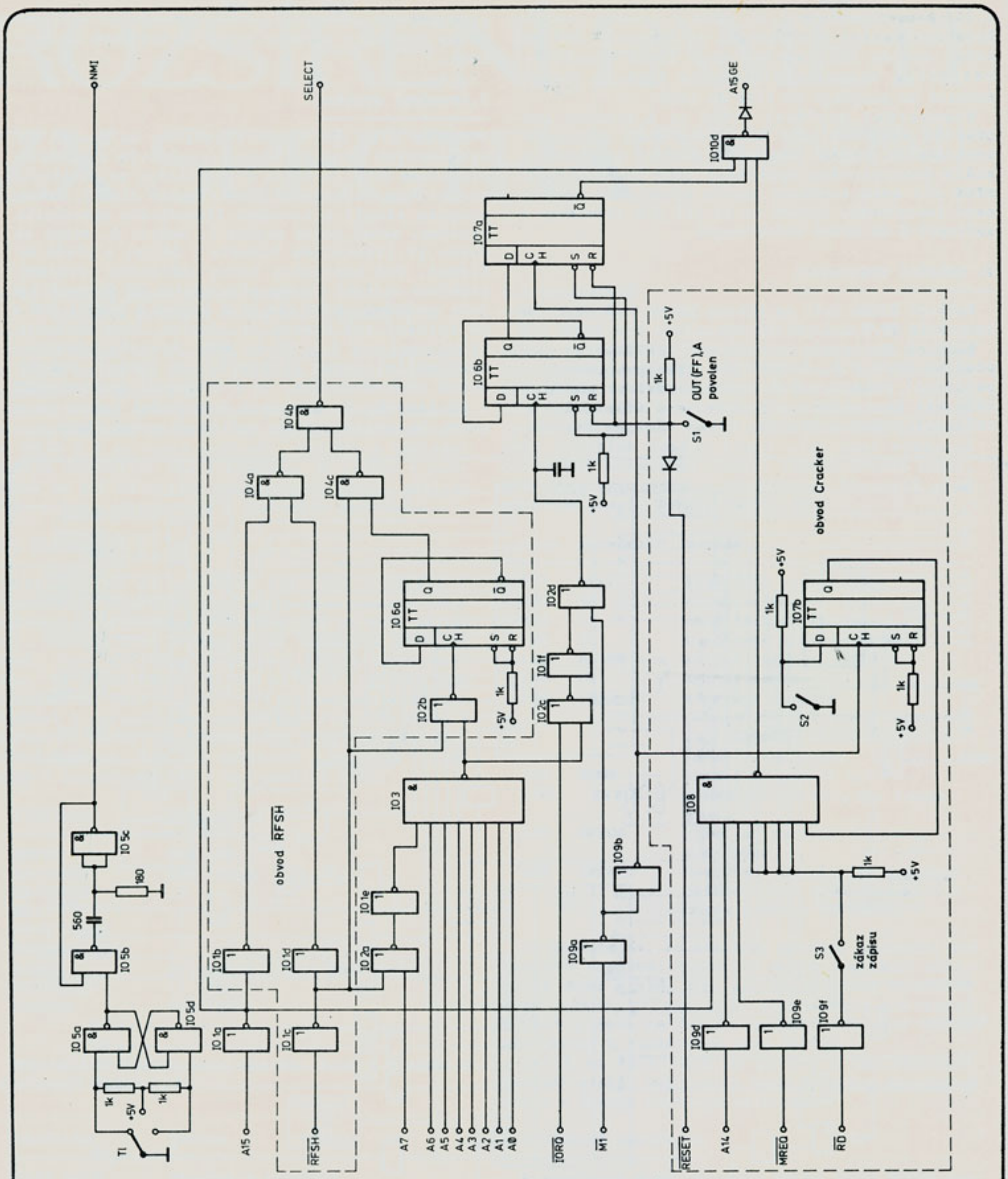

**Řadič floppy disku, připojitelný přímo ke sběrnici ZX-Sp 80K bez použití posilovače sběrnice. U většiny typů mechanik nejsou signály v závorkách využity. Proto hradla označená * není třeba osazovat. Pozn.: Signály v závorkách nejsou pro uvažovaný typ FDD (NEC) využity. Prosíme opravte si: Na vstupu do RDY WD2797 chybí invertor, který neguje signál /RDY.**

## 4. Informace potřebné pro konfiguraci systému

Pokud budete mít o tento systém zájem, spolu s disketou nám zašlete základní údaje o konkrétní konfiguraci vašeho technického vybavení:

- Počet a druh mechanik pružných disků:
  -SS/DS
  -SD/DD
  -3,5", 5,25", 8"
- Přiřazení mechanik jednotlivým jednotkám v okamžiku startu systému
- Druh formátu mechaniky (s vysokým využitím nebo některý z formátů používaný na jiných počítačích, ev. formát dle vašich požadavků, avšak podrobně popsaný)

Podle těchto údajů pro vás vytvoříme systémovou disketu s vlastním systémem i dalšími programy pro jeho použití (SYSGEN, FORMAT, DRINST, DRIVE,




▲

**Jedna z možností zapojení ZX-Sp 80K. Má 4 části:**
**1/ Stránkovací mechanismus (povinný)**
**2/ Obvod refresh (nepovinný - pouze pro 8-bitový refresh)**
**3/ Obvod NMI**
**4/ Obvod Cracker**
**Vše lze umístit i do "gumového" ZX Spectra.**
**Význam spínačů:**
**S1 - zákaz přestránkování (klasický mód)**
**S2 - mód Cracker**
**S3 - zákaz zápisu do ramky (pro adresy 0-4000H)**

**EXITCPM atd.) a se standardní sestavou programů OS CP/M (PIP, STAT apod.).**

**ad 10) Přehled programů nahraných na kazetě č.2**

**Některé zde uváděné programy malá část uživatelů ZX-Sp 80K již vlastní. Většina programů a manuálů je však zcela nová.**

**Manuály:**

**CRACKERtsw**
Stručný přehled použití programu CRACKER.

**DICT80 tsw**
Stručný návod k používání slovníku DICT 80K.

**SAPI/tsw.1-4**
Popis systémového simulátoru počítače SAPI 1, verze 4.0 pro ZX-Sp 80K. Z příručky k SAPI 1 je převzat popis MIKOSu a textového editoru PEDIT pro tvorbu zdrojových textů.

**TOOL48tsw**
Popis ovládání křížového assembleru pro jednočipové mikropočítače řady 48.

**LASER80tsw**
Doplněk k manuálu LASER GENIUS MONITOR v úpravě pro ZX-Sp 80K.

**DISAXtsw**
Podrobný manuál k disassembleru DISAX.

**WD2797tsw**
Hardwarový popis řadiče floppy disku WD2797.

**CP/M tsw.1-12**
Příručka programátora. Obsahuje popis struktury a služeb operačního systému CP/M.

**M(Z)BASIC1-3**
Prozatímní manuál.

**HPS+COPYt.**
Manuál ke kopíráku HPS+COPY.

**Programy:**

**Schémata/1-6**
Stavební návody na doplňky k ZX-Sp 80K. 1K paměti je rezervováno pro obsluhu tiskárny.

**Assembler EDTASM80k**
Makroassembler fy Microsoft, převzatý z počítače EG 3003. Práce s ním je snadná, navíc je vybaven symbolickým ladícím prostředkem Z-BUG, který umožňuje efektivní ladění ve zdrojovém tvaru. Funkce editoru je podobná editoru GENS3, ale podstatně zdokonalená.

**Disassembler DISAX**
Byl vyvinut přímo pro ZX-Sp 80K. Umožňuje tvorbu zpětných překladů ze strojového kódu do zdrojového tvaru. Kromě běžných funkcí (definice DB, DW, DS), umožňuje definovat uživatelská symbolická návěští, "makra" v kódu (např. u Spectra je to RST 8 + bajt) a generuje tabulky křížových odkazů. Je-li program úplně zadán, umožňuje zcela automatickou relokaci do libovolného prostoru paměti bez jakýchkoli dodatečných úprav.

**Basic MZ BASIC V3.0**
O něm platí vše, co o programu MZ BASIC V2.0. Byl však doplněn vlastním diskovým operačním systémem. Kromě progresívních rysů standardu MSX má i prvky operačního systému MS DOS (např. stromovou strukturu adresáře aj.). Uvedená implementace je směrem dolů kompatibilní. Znamená to, že všechny programy napsané na páskové verzi MZ BASIC V2.0 lze spustit i na verzi diskové.

**Operační systém CP/M**
Disková verze. Podrobnosti viz kapitola č.9.

**Operační systém CP/M mdv.**
Verze pro microdrive. Zapsáno na dvou kazetkách. Na jedné je vlastní OS CP/M v klasickém formátu microdrive, druhá je naformátována speciálně pro použití v CP/M. Před vložením kazetky do microdrivu je nutné provést hardwarovou úpravu IF1.

**BOOT V1.4**
Krátký zaváděcí program pro zavedení libovolného operačního systému z disku do paměti. Po připojení konzole na (e) STD BUS bude nahrazen pamětí EPROM.

**DICT 80k**
Slovník cizích slov.

**SAPI 1**
Systémový simulátor počítače SAPI 1, verze 4.0.

**Assembler TOOL48**
Pro jednočipový mikropočítač (pro provoz na SAPI 1). Protože SAPI 1 má jiný formát záznamu než klasické Spectrum, nelze k jeho kopírování použít běžné kopíráky. Kopii lze pořídit pomocí OS MIKOS (součást programu SAPI 1). TOOL48 je dlouhý 3B bloků. Chceme-li provést jeho kopii, po natažení do paměti jej nesmíme spustit, ale nejdříve provést kopii na pásek.

**Disassembler LASER80k**
Upravená verze programu LASER GENIUS MONITOR.
HPS+COPY

**Kopírák pro ZX-Sp 80K** umožňuje záznam ve většině známých formátů používaných na Spectru.

**Sada demonstračních programů MZ Basicu.**
Některé programy nebyly psány přímo pro ZX-Sp 80K. Mezi ně patří TOOL48, CP/M mdw. Na kazetách rovněž nejsou žádné další EPROMky, protože se domníváme, že v nich není těžiště využití možností ZX-Sp 80K.

## ad 10) Literatura


(18) Amatérské radio 1988/9
(19) Amatérské radio 1985/9
(20) Amatérské radio 1986/1
(21) Amatérské radio 1986/4
(22) Amatérské radio 1986/6
(23) Amatérské radio 1986/7
(24) Amatérské radio 1986/8
(25) Amatérské radio 1986/10
(26) Amatérské radio 1986/11
(27) Amatérské radio 1986/12
(28) Amatérské radio 1987/1
(29) Amatérské radio 1987/12
(30) Amatérské radio 1984/9
(31) Zpravodaj Karolinky č.12
(32) Spectrum 1/88, klubový zpravodaj 602.ZO Svazarmu
(33...) CPM-1...x, řada příruček vydaných Teslou ELTOS, VN MON Praha


## Závěrem

V současné době je rozpracována řada dalších programů a hardwarových doplňků. Budeme rádi, když ZX-Sp 80K najde své další uživatele a spolupracovníky v rozvoji jeho technického i programového vybavení.

**Ladislav Sieger**

---

**DiskDoubler firmy Datran Corp.** minimálně zdvojnásobuje kapacitu paměťového média - floppy disku, hard disku, ramdisku. Dosahuje toho specifickou komprimací. Kartu DiskDoubleru lze připojit k jakémukoli péčéčku. Pracuje s DOS 2.0 a výše. Interface je interaktivní, komprese a expanze probíhá současně se zápisem a čtením dat. Tato automatika nepracuje jen u systémových souborů .EXE, .COM, .SYS, .BAT, .BIN, protože DOS by je v komprimovaném stavu nepoznal. Lze však provést jejich "manuální" komprimaci. Cena $189.

# AMSTRAD CPC 6128 S PAMĚTÍ 384 K

Mnozí příznivci 8-bitových domácích počítačů určitě zatoužili vybavit své stroje větší pamětí. Dnes, kdy existují paměťové obvody s kapacitou až 1 Mbit, to může zvládnout i zkušený amatér. Otázkou ale zůstane, jak takto rozšířenou paměť využít. Systém ani profesionální programy o námi rozšířené paměti nic netuší. Chceme-li tedy paměť skutečně využívat, je to zpravidla možné jen ve vlastních programech.

U počítačů s diskovým operačním systémem je asi nejužitečnější využít přídavnou paměť jako ramdisk. Operační systém (pro 8-bitové počítače je nejrozšířenější CP/M) pak komunikuje s pamětí jako s normální disketovou jednotkou. Cena, za jakou můžeme ramdisk pořídit, není větší než cena disketové jednotky s porovnatelnou kapacitou. A rychlost, s jakou ramdisk zapisuje a čte, je až překvapující.

Pokud jste již v CP/M překládali delší program nebo třídili rozsáhlejší databázový soubor na disku, dobře víte, jak dlouho to někdy, za neustálého bzučení disketové jednotky trvá. Na ramdisku to proběhne za pár sekund a v naprostém tichu. Programy pod CP/M velmi často vyžadují přístup na disk. Proto je rychlejší si program a ostatní potřebné soubory nejdříve překopírovat na ramdisk a pracovat pak s ním. Samozřejmě, že data z ramdisku po vypnutí počítače zmizí. To se na první pohled může jevit jako nevýhoda. Při vývoji nových programů si floppy disk zpravidla zaplníme různými polotovary a za určitou dobu již sami nevíme, co vlastně platí. Při práci s ramdiskem zapíšeme na disketu jen kopii konečné verze a ostatní se nám po vypnutí samy vymažou.

Počítače Amstrad/Schneider CPC 6128 jsou standardně vybaveny pamětí RAM 128K a operačním systémem CP/M Plus. Rozhodneme-li se paměť našeho počítače rozšířit, máme dvě možnosti. Buď přídavnou paměť vestavět přímo do počítače, nebo ji připojit přes rozšiřovací konektor jako externí jednotku. První varianta je - pokud jde o vlastní zapojení - jednodušší, ale znamená značný zásah do počítače se všemi riziky. Druhá varianta přináší o něco složitější zapojení (musíme znovu vytvořit obvody pro ovládání paměti), ale počítač nemusíme ani otevřít. Já jsem se rozhodl příliš neriskovat a navrhnout přídavnou paměť 256K jako zvláštní jednotku.

Použitý mikroprocesor Z80 má 16 adresových vodičů, takže může přímo adresovat pouze 64K. Vnitřní paměť 128K je proto rozdělena na 8 bloků po 16K. Označíme je jako bloky 0 až 7. Z těchto osmi bloků jsou vždy současně vybrány jen 4 bloky, ke kterým má procesor přístup. Tak existuje 8 konfigurací paměti, které můžeme také očíslovat od 0 do 7. Každá přepíná různou sestavu čtyř bloků do paměťového prostoru.

Bloky jsou přístupné v jednotlivých konfiguracích následovně:

| Konfigurace | Bloky na adresách | | | | Konf.bajt |
|---|---|---|---|---|---|
| | 0000h | 4000h | 8000h | C000h | |
| 0 | 0 | 1 | 2 | 3 | C0 |
| 1 | 0 | 1 | 2 | 7 | C1 |
| 2 | 4 | 5 | 6 | 7 | C2 |
| 3 | 0 | 3 | 2 | 7 | C3 |
| 4 | 0 | 4 | 2 | 3 | C4 |
| 5 | 0 | 5 | 2 | 3 | C5 |
| 6 | 0 | 6 | 2 | 3 | C6 |
| 7 | 0 | 7 | 2 | 3 | C7 |

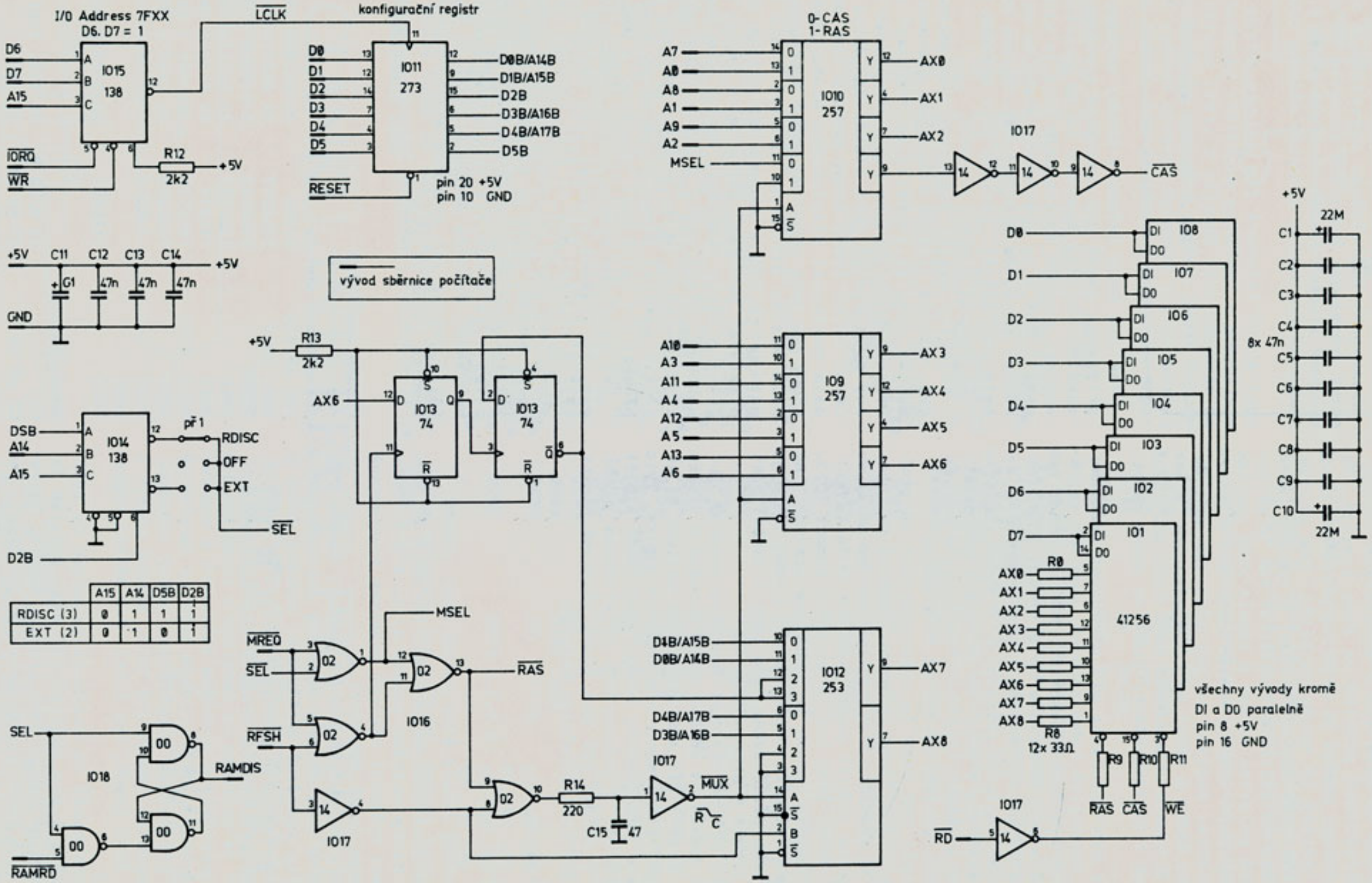
Přepínání konfigurací v CPC 6128 probíhá tak, že na výstupní port 7FXXh je vyslán konfigurační bajt, jehož tři spodní bity znamenají číslo konfigurace a bity 6 a 7 jsou vždy v log.1.

Přídavná paměť 256K, to je dalších 16 bloků (bloky 8 až 23) po 16K. Protože chceme připínat vždy jen jeden z těchto bloků, bude to vyžadovat dalších 16 možností konfigurace paměti. Přitom musíme v případě, že bude adresován blok přídavné paměti, znemožnit současný přístup do vnitřní paměti. Naše nové konfigurace zvolíme tak, že se budou krýt s vnitřními konfiguracemi 4 až 7, při nichž je bit 2 konfiguračního bajtu vždy roven 1.

Zavedeme tedy tyto další konfigurace:

| Konfigurace | Bloky na adresách | | | | Konf.bajt |
|---|---|---|---|---|---|
| | 0000h | 4000h | 8000h | C000h | |
| 8 | 0 | 8 | 2 | 3 | E4 |
| 9 | 0 | 9 | 2 | 3 | E5 |
| 10 | 0 | 10 | 2 | 3 | E6 |
| 11 | 0 | 11 | 2 | 3 | E7 |
| 12 | 0 | 12 | 2 | 3 | EC |
| 13 | 0 | 13 | 2 | 3 | ED |
| 14 | 0 | 14 | 2 | 3 | EE |
| 15 | 0 | 15 | 2 | 3 | EF |
| 16 | 0 | 16 | 2 | 3 | F4 |
| 17 | 0 | 17 | 2 | 3 | F5 |
| 18 | 0 | 18 | 2 | 3 | F6 |
| 19 | 0 | 19 | 2 | 3 | F7 |
| 20 | 0 | 20 | 2 | 3 | FC |
| 21 | 0 | 21 | 2 | 3 | FD |
| 22 | 0 | 22 | 2 | 3 | FE |
| 23 | 0 | 23 | 2 | 3 | FF |

Bity 0 až 5 těchto konfiguračních bajtů se zaznamenávají do registru konfigurací IO11. Při zvolených dodatečných konfiguracích bude bit 2 a bit 5 konfiguračního bajtu vždy roven 1, a protože blok vnější paměti bude na adresách 4000h až 7FFFh, bude adresový bit A14=1 a A15=0. Tuto podmínku vyhodnotí dekodér IO14 a na jeho výstupu se objeví signál /SEL, který umožní, aby se signály /RAS a /CAS aktivovala vnější paměť; zároveň je signálem RAMDIS vypnuta vnitřní paměť. Pro adresování v rámci bloku 16K stačí adresové bity A0 až A13. Pro výběr jednoho ze 16 vnějších bloků mohou být použity přímo bity D0B, D1B, D3B a D4B z registru konfigurací.


I/O Address 7FXX
D6. D7 = 1
IO15
138
LCLK
konfigurační registr
IO11
273
D0B/A14B
D1B/A15B
D2B
D3B/A16B
D4B/A17B
D5B
pin 20 +5V
pin 10 GND
IORQ
WR
R12
2k2
+5V
RESET
+5V C11 C12 C13 C14 +5V
G1 47n 47n 47n
GND
vývod sběrnice počítače
R13
2k2
AX6
IO13
74
IO14
138
DSB
A14
A15
př 1
RDISC
OFF
EXT
SEL
D2B
A15 A14 D5B D2B
RDISC (3) 0 1 1 1
EXT (2) 0 1 0 1
MSEL
MREQ
SEL
RFSH
IO16
RAS
IO18
RAMDIS
RAMRD
R14
220
C15 47
MUX
R/C
0-CAS
1-RAS
IO10
257
A7
A6
A8
A1
A9
A2
MSEL
AX0
AX1
AX2
IO17
CAS
IO9
257
A10
A3
A11
A4
A12
A5
A13
A6
AX3
AX4
AX5
AX6
IO12
253
D4B/A15B
D0B/A14B
D4B/A17B
D3B/A16B
AX7
AX8
IO8
IO7
IO6
IO5
IO4
IO3
IO2
IO1
D0
D1
D2
D3
D4
D5
D6
D7
41256
R0
12x 33Ω
AX0
AX1
AX2
AX3
AX4
AX5
AX6
AX7
AX8
R8
R9
R10
R11
RAS
CAS
WE
RD
+5V
22M
8x 47n
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
22M
všechny vývody kromě
DI a DO paralelně
pin 8 +5V
pin 16 GND

Použité paměti 41256 jsou dynamické. Každá jejich paměťová buňka je tvořena jedním kondenzátorem, který je buď nabitý nebo vybitý. Jednotlivé buňky jsou uspořádány do matice z 2**8 řádků a 2**10 sloupců. Paměť 256K je adresována 18 bity. Paměťový čip má však jen 9 adresových vstupů. Proto adresování probíhá ve dvou krocích. Na adresové vstupy pamětí je nejdříve přivedena první polovina adresy (8 spodních bitů). Pak přijde signál /RAS (Row Address Strobe) a vybraný řádek je přenesen do vnitřního registru. Pak je přivedena druhá polovina adresy a impuls /CAS (Column Address Strobe) - tím je vybrán jeden bit z vnitřního registru. Registr je pak přenesen zpět do paměťového řádku a náboj na kondenzátorech je tak obnoven. K tomuto občerstvení musí dojít přibližně po každých 2 milisekundách, jinak se obsah paměti ztrácí.

Protože mikroprocesor při provádění programu adresuje jednotlivé řádky náhodně, musí být zajištěno jejich cyklické adresování. Mikroprocesor Z80 zajišťuje a synchronizuje občerstvení signálem /RFSH. Na adresovou sběrnici posílá cyklicky se měnící obsah registru (čítače) R. V tomto registru se však mění jen 7 bitů. To postačuje pro paměti 64 Kbit ale paměti 256 Kbit vyžadují 8-bitové občerstvení, protože matice má 2**8 řádků.

V počítači CPC 6128 je občerstvování vnitřní paměti zajištěno jiným způsobem. Využívá se toho, že paměť je pravidelně adresována řadičem obrazovky. Přídavná vnější paměť využívá občertsvení signálem /RFSH. Osmý bit je vytvořen pomocí IO13. V prvním klopném obvodu se zachytává stav bitu A6 při příchodu signálu /RFSH a druhý klopný obvod je zapojen jako dělič dvěma. Jde tedy o jakési prodloužení čítače R mikroprocesoru.

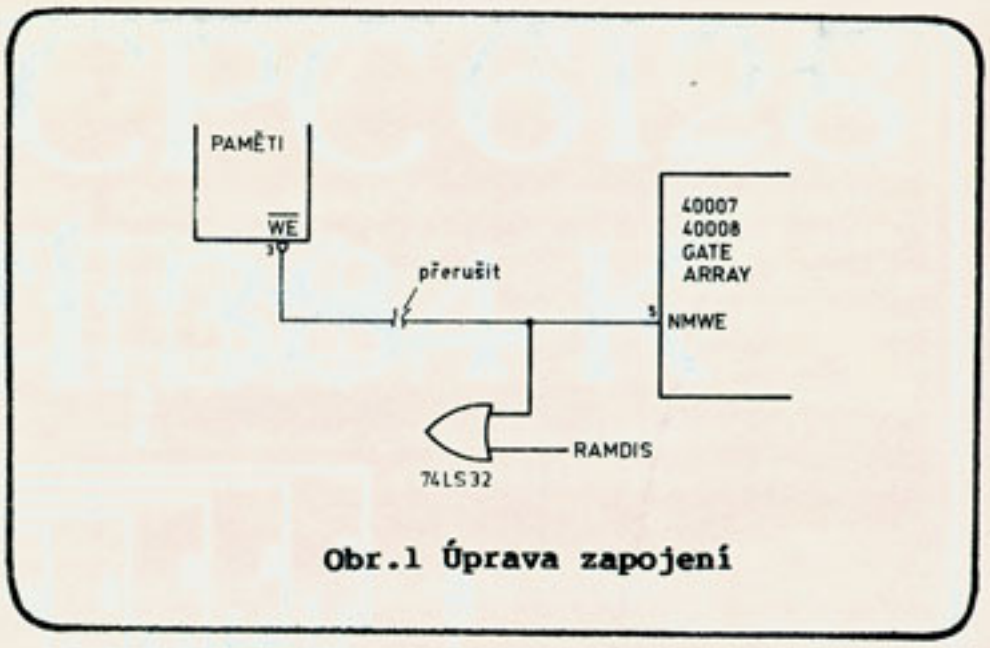
Přepínač Př1 na výstupu dekodéru IO14 má následující funkci: V poloze RDISK dochází k výběru přídavné paměti při konfiguračních bajtech E4 až FF tak, jak bylo uvedeno v předchozím popisu. Tyto konfigurace jsou použity tehdy, když paměť slouží jako ramdisk. V poloze VYP je přídavná paměť vypnuta. V poloze EXT jsou bloky přídavné paměti vybírané při konfiguračních bajtech:

C4, C5, C6, C7
CC, CD, CE, CF
D4, D5, D6, D7
DC, DD, DE, DF

a při konfiguračních bajtech E4, E5, E6 a E7 jsou vybírané bloky vnitřní paměti 4, 5, 6 a 7. Tuto konfiguraci vyžadují některé programy provozované pod Amsdosem (např. Pyradev).

Několik poznámek ke stavbě paměti: Obvody TTL musejí být z řady LS nebo ALS, aby sběrnice počítače nebyla nadměrně zatěžována. Dodržte počet a hodnoty blokovacích kondenzátorů. Pro poměrně drahé paměťové obvody použijte raději objímky. Celá paměť je umístěna na destičce o rozměrech 115 x 98 mm, uzavřena v krabičce zhotovené z cuprextitu a spojená krátkým plochým kabelem (maximálně 15 cm) s příslušným konektorem. Pokud nebudete mít k dispozici paměti 256 Kbit, můžete použít i paměti 64 Kbit (např. typ 4164). Pak můžete vypustit i IO13.

Přídavnou paměť na uvedeném principu by bylo možné použít i pro počítač CPC 464. Problém je však v rozdílné funkci jeho signálu RAMDIS. Zatímco u CPC 6128 tento signál zakazuje jak čtení, tak i zápis do vnitřní paměti RAM, u typu CPC 464 tento znemožní pouze čtení, ale nezabrání tomu, aby se zapisovalo současně do vnitřní i přídavné paměti. Stálo by možná zato, vyzkoušet úpravu dle obr.1. Po této úpravě zapojení by pak signál RAMDIS měl zabránit zápisu do paměti RAM. Po připojení přídavné paměti by pak i na CPC 464 mohly chodit některé programy určené pouze pro CPC 6128.




**Obr.1 Úprava zapojení**

Jak již bylo řečeno v úvodu, nejlépe využijeme přídavnou paměť pod operačním systémem CP/M jako ramdisk. Uprava systému se dá provést poměrně snadno, protože systém CP/M Plus je uložen celý v paměti RAM a na systémovém disku ho najdeme jako soubor s názvem C10CPM3.EMS. Úprava CP/M 2.2 by byla obtížnější, protože část systému je u počítačů Amstrad uložena v romce.

Nejdříve musíme definovat blok parametrů disku DPB. Protože vlastně simulujeme diskovou jednotku, musíme ramdisk rozdělit na "sektory" a "stopy". Délku sektoru zvolíme pro jednoduchost 128 bajtů, aby byla stejná jako standardní délka záznamu v CP/M. Délku jedné stopy zvolíme 16K, aby odpovídala velikosti připínaného bloku paměti. Změna stopy pak bude odpovídat změně konfigurace paměti. Dále musíme definovat záhlaví bloku diskových parametrů DPH a adresu DPH uložit do tabulky diskových jednotek. Zbývá ještě napsat rutiny pro čtení a zápis ramdisku. Rutiny jsou poměrně jednoduché. Stačí určit z čísla stopy odpovídající konfigurační bajt, z čísla sektoru vypočítat adresu paměti, přepnout na danou konfiguraci a přenést 128 bajtů mezi "sektorem" a DMA.

Při studeném startu musíme ramdisk inicializovat, to znamená vymazat jeho adresář. Není ale příliš vhodné, aby se disk vymazal tehdy, když startujeme znovu po resetu počítače známým tříprstovým hmatem. Proto je zavedena rutina, která před inicializací nejprve zkoumá celý adresář, a pokud zjistí, že v něm již jsou názvy souborů, tak ho nevymaže. Protože přídavná paměť je odpojitelná, další rutina při studeném startu testuje, zda je ramdisk vůbec připojen.

**Ing. František Langmaier**

**Seznam součástek**

| | | |
|---|---|---|
| **Integrované obvody** | | |
| IO1 až IO8 | 41256 | dynam.RAM 256x1 Kbit |
| IO9, IO10 | 74LS257 | 4x multiplexer 2 na 1 |
| IO11 | 74LS273 | 8-bitový střadač D |
| IO12 | 74LS253 | 2x multiplexer 4 na 1 |
| IO13 | 74LS74 | 2x klopný obvod D |
| IO14, IO15 | 74LS138 | dekodér 3 na 8 |
| IO16 | 74ALS02 | 4x hradlo NOR |
| IO17 | 74LS14 | 6x invertor s hysterezí |
| IO18 | 74ALS00 | 4x hradlo NAND |
| **Kondenzátory** | | |
| C1 | 22 M/10 V | TE 132, tantal.kapka |
| C2 až C9 | 47-68 nF | TK 782 |
| C10 | 22 M/10 V | TE 132 |
| C11 | 100 M/10 V | TF 007 |
| C12 až C14 | 47 nF | TK 782 |
| C15 | 47 pF | TK 754 |
| **Rezistory** | | |
| R0 až R11 | 33 Ohmů | TR 191 |
| R12, R13 | 2k2 až 10k | TR 191 |
| R14 | 220 Ohmů | TR 191 |

# CHYBY V TURBO PASCALU

V počítačových časopisech občas proskočí zmínka o chybách v některých osvědčených programech. Chybičkám se nevyhnul ani známý cépéemkový Turbo Pascal 3.0 firmy Borland. Polský časopis Komputer 10/88 tento problém shrnul a hlavně ukázal způsob nápravy.

První chyba vzniká při konverzi čísla typu Integer hodnoty -32768 ve vyjádření s pohyblivou řádovou čárkou (FP). V případě této konverze se program dostává do nekonečné smyčky. Jacek Gwiazdka navrhl způsob opravy. Opravnou rutinu umístil na adresu 0106H, kde je původně Borlandův copyright:

```
0106 OPRAVA: LD   A,H       ;Je číslo integer
0107         OR   L         ;rovno 0?
0108         JP   Z,0B72H   ;Je - skoč na FP=0
010B         CP   80H       ;Je číslo rovno 8000H,
010D         JP   NZ,100DH  ;čili -32768?
0110         LD   A,L       ;Ne - skoč do normální
0111         AND  A         ;procedury Borlandů a
0112         JP   NZ,100DH  ;pokračuj v ní.
0115         CALL 0B72H     ;Ano-vynuluj HL,DE,BC
0118         LD   L,90H     ;a ulož do nich -32768
011A         LD   B,80H     ;v reprezentaci FP
011C         RET            ;Návrat z KONVER


1008 KONVER: JP   OPRAVA    ;Skok do oprav.rutiny
```

Princip nápravy je zřejmý z komentáře. Na místo začátku původní rutiny KONVER, která provádí konverzi, je umístěn skok do opravné rutiny. Ta napřed oddělí číslo 0 a pak všechny hodnoty různé od -32768 vrátí ke zpracování do původní rutiny. V případě hodnoty -32768 zavolá rutinu pro zpracování FP=0, která hodnotu vynuluje a nakonec do registrů vnutí hodnotu -32768 v reprezentaci FP.

Na tuto chybu navazuje další, méně závažná. Spočívá v tom, že hodnotu -32768 nelze umístit v textu programu, napsaného v Turbo Pascalu, protože při kompilaci vyvolá chybové hlášení. Tuto hodnotu však můžeme zadat šestnáctkově. Stejná chyba se vyskytuje v Turbo Pascalu 3.0 pro MS DOS. Ve verzi 4.0 je už odstraněna.

Druhá chyba nemá až tak dalekosáhlé následky-nezpůsobuje zablokování programu. Projevuje se ve funkci modulo, pokud je její první argument záporný.

Příklad:

```
-2 mod 5  dává výsledek 2 (správně má být -2)
-8 mod 5  dává ale správný výsledek -3
```

Způsob opravy podle c't Magazinu 10/86:

```
0745         PUSH DE
0746         CALL 070FH
0749         POP  AF
074A         EX   DE,HL
074B         RET  P
```

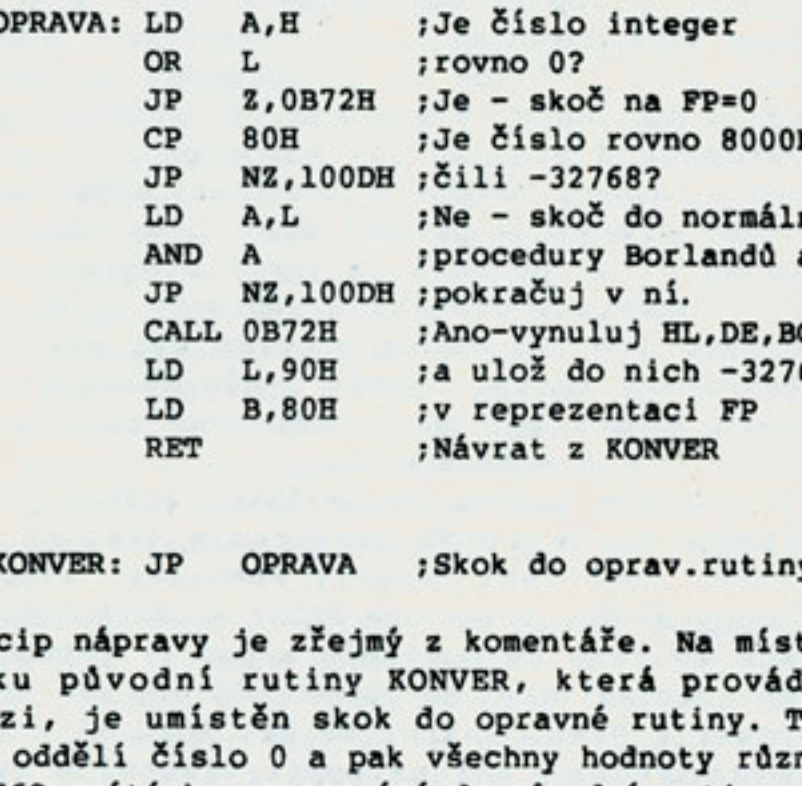
Třetí chyba se projevuje při použití Turbo Pascalu s operačním systémem CP/M Plus. V tom případě Turbo dává nesprávnou informaci o volném místě na disku (příkaz DIR z hlavního menu). Vyplývá to z rozdílných funkcí BIOSu v CP/M Plus a CP/M 2.2. Nápravu nabízí zvláštní verze Turbo Pascalu, upravená firmou Borland pro CP/M Plus:

```
2C41         LD   C,1AH     ;Nastavení DMA na
2C43         LD   DE,2C74H  ;adresu 2C74H.
2C46         CALL 0005H
2C49         LD   C,19H     ;Zjištění čísla
2C4B         CALL 0005H     ;aktuálního disku.
2C4E         LD   C,2EH     ;Zjištění volného
2C50         LD   E,A       ;místa (č.disku do E)
2C51         CALL 0005H     ;na disku.
2C54         LD   HL,(2C74H);Přenos hodnot do HL
2C57         LD   A,(2C76H) ;a do A.
2C5A         LD   C,A       ;Přesun do C.
2C5B         LD   B,3       ;Nastavení čítače.
2C5D LOOP:   XOR  A         ;Nulování CY.
2C5E         RR   C         ;Tříbajtové číslo je
2C60         RR   H         ;děleno osmi (převod
2C62         RR   L         ;počtu sektorů na KB).
2C64         DJNZ LOOP
2C66         JP   2C77H     ;Do původní procedury.

2C91         DEFB 0,0,0,0   ;Čtyři NOPy.
```

Pozor, tato úprava volá službu BDOSu číslo 46, která existuje jen v CP/M+. Nebude tedy fungovat v CP/M 2.2!

Dan Meca

# KE ČLÁNKU INTERFACE PRO PŘIPOJENÍ BODOVÉ TISKÁRNY Z ARA 10/87




Obr.1 Interface pro PMD 85

V uvedeném článku je popsán obvod pro interface IMS 2 počítače PMD 85. Má-li tento interface sloužit pouze k připojení tiskárny, jak je v záhlaví článku uvedeno, je zbytečně komplikovaný. V našem podniku (Chemoprojekt) používáme PMD 85-1 s tiskárnou DZM 180 připojenou podle přiloženého schématu, které je ověřeno dvouletým provozem a je součástkově naprosto nenáročné. Použité IO jsou typu 7400 a 7404 (nebo 2 x 7400). Princip činnosti je obdobný popisu IMS 2 v uvedeném článku. Napájení využívá +5 V z tiskárny.

Václav Horčička

# KČEMU JE CP/M?

Někdo se možná pozastaví nad tím, proč hodláme věnovat pravidelnou rubriku operačnímu systému CP/M, jehož sláva ve světě pohasíná. Je sice pravda, že verze CP/M 86 pro šestnáctibitové počítače se - oproti PC DOS - příliš neprosadila, ale v oblasti osmibitových počítačů i dnes drží primát. O tomto nejslavnějším a nejrozšířenějším operačním systému pro osmibitové pocítače se toho u nás mnoho nedočtete. Občas nějaká ta zmínka v Amatérském radiu...a tím to prakticky končí. Ostatně, operační systém, který se u nás příliš nepoužívá, si ani víc nezaslouží... Nebo snad ano? Určitě ano! Právě v Amatérském radiu se čtenář mohl dozvědět, že CP/M 2.2 se u nás dovedně skrývá pod pseudonymem MIKROS. I operační systém výborných počítačů Robotron 1715, nazývaný SCP, je zase CP/M. Je pravda, že ČSVTS v roce 1986 vydala publikaci o CP/M, ta však byla obecně nedostupná. A že by o ni zájem byl, se ukázalo při doprodeji zbytků brožury v prodejnách Tesly Eltos.

Citelnou informační mezeru se pokusíme částečně zaplnit naší novou rubrikou. Co si od toho slibujeme? Především překonání propasti programové nekompatibility mezi některými typy mikropočítačů na našem trhu. Vždyť CP/M lze implementovat na každý mikropočítač s procesory 8080 nebo Z80. Týká se to počítačů TNS, SAPI 1, ROBOTRON 1715, ale i PMD 85, IQ 151, PP 01, Didaktik Gama, ZX Spectrum, Sharp MZ 811 a 821, Amstrad/Schneider 664 a 6128, Sord M5, Commodore 128 a dalších. Tyto počítače, ať už mají CP/M implementován přímo výrobcem nebo až dodatečně, mohou s uvedeným operačním systémem používat stejné programy. A nejsou to programy ledajaké. Pod CP/M pracují například známé programy WordStar, TURBO-Pascal, MACRO 80, dBase II, Fortran a pod. Jejich přímí následníci pracují s MS-DOS na populárních pécéčkách. Protože CP/M byl prvním (a velice kvalitním) operačním systémem pro mikropočítače, vycházejí z něj všechny následující, včetně MS-DOSu. Kdo se tedy naučí pracovat s CP/M, ve velice krátké době pak zvládne PC-DOS na pécéčku. Navíc může i nadále používat novější verze svých oblíbených programů, přičemž jejich data a zdrojové texty jsou přenositelné.

Doufáme, že se nám do práce na rubrice podaří zapojit i cépéemkové profesionály, kteří mezi sebou živě komunikují, ale veřejnost o nich prakticky neví. Přitom by ji výsledky jejich práce určitě zajímaly. Třeba takový vývojový operační systém CP/X pro počítač Robotron 1715 od firmy Huvole (Hudec, Vojáček a Loeffler) svými možnostmi značně převyšuje původní SCPX firmy Robotron. Ze stejné dílny pochází zajímavý program, který s CP/M formátuje diskety a zapisuje i čte data ve formátu IBM PC. Tento program byl přijat jako zlepšovací návrh - jenže o něm málokdo ví. I v tomto směru bychom v naší nové rubrice mohli udělat kus prospěšné práce.

V rubrice budeme uveřejňovat jednak návody k různým hardwarovým rozšířením (řadiče floppy disku, rozšíření paměti, RAMdisky, simulátory floppy disků, interface RS232 a CENTRONICS, zaváděče systému do počítače a pod.), jednak popisy implementací CP/M na různé počítače (ZX Spectrum, Didaktik Gama, Sharp, PP 01, Sord M5 a další). Nezapomeneme ani na popis vlastního systému, včetně systémových programů (standardních i nestandardních) a jejich ovládání. Postupně přineseme řadu informací o zajímavých programech tuzemských i zahraničních autorů. Přitom budeme úzce spolupracovat s klubem CP/M, který nedávno vznikl v pražské 602.ZO (viz informace v minulém čísle). Autor tohoto textu a redaktor rubriky je jedním z jeho zakládajících členů. Máme ovšem zájem spolupracovat i se všemi dalšími uživateli CP/M, amatérskými i profesionálními. Spojit se s námi můžete přímo v klubu nebo písemně na adresu redakce.

**Daniel Meca**

# NOVÁ IMPLEMENTACE CP/M NA SHARP MZ-800 ▷▷▷

Jde o standardní operační systém CP/M 2.2 (též Mikros či SCP), pracující se 60K základní paměti RAM. Celý systém ideově vychází z původního operačního systému P-CP/M, ale jsou v něm odstraněny některé chyby a v mnoha směrech je vylepšen. Na rozdíl od jiných implementací CP/M pro Sharp MZ-800, které dosud v Československu vznikly, se popisovaný systém prakticky ve všech detailech chová stejně jako standardní P-CP/M, takže uživatelé zvyklí na původní programové vybavení a ovládání počítače si nemusejí zvykat na žádné novinky. Ale pozor - i když nová CP/M vypadá na první pohled stejně jako P-CP/M, je třeba si uvědomit, že jde o programový produkt vytvořený zcela od začátku znovu. To se samozřejmě týká i všech příslušejících systémových programů. Tyto programy se jmenují stejně jako původní, navenek se tak i chovají, ale pracují jinak.

Nový operační systém zachovává všechny přednosti původního řešení. Především umí obsluhovat všechny originální periférie Sharpu, počínaje standardními floppy disky přes různé varianty ramdisků, tiskárnu Centronics a konče sériovým RS232. Systém zachovává původní systémový řádek, možnost využití funkčních kláves, zobrazení módu klávesnice a hodiny reálného času. Hlavní předností nově vytvořené implementace je její neobyčejná pracovní rychlost. Spolupráce s disky je asi o 70 procent rychlejší než původní, obrazové operace jsou rychlejší více než 23 krát (!).

(Pozn.red.: Protože se nám tvrzení autora zdálo přece jen trochu nadsazené, nechali jsme si nový systém předvést. Celá řada testů dokázala pravdivost tvrzení. Nezbylo, než konstatovat, že zřejmě jde o vůbec nejrychlejší systém pro Sharp MZ-800.)

Přitom je pamatováno na tuzemské poměry, v nichž každý používá, co zrovna sežene. Pomocí systémového programu SETUP lze nadefinovat nejrůznější formáty připojených diskových jednotek. Základní formát diskety odpovídá standardu počítačů IBM PC XT/AT. Lze zvolit jednostranný i oboustranný záznam, 40 i 80 stop, na 80-stopé mechanice může být simulována 40-stopá. Dokonce i na každé připojené mechanice lze nastavit jiný formát záznamu. S ohledem na různorodost používaných mechanik lze nastavit všechny jejich časové konstanty s rozlišením jedné milisekundy. To umožňuje používat i perspektivní třímilisekundový krok diskové hlavy (to původní P-CP/M neumí). I způsob ovládání disků je dokonaleji propracován - díky úplné kontrole adresace na disku je znemožněn vznik chyb při zápisu a čtení. Zvláštní pozornost je věnována ošetření různých chyb, přičemž systém obvykle detekuje i příčinu závady a umožňuje zvolit způsob pokračování programu (Retry/Abort/Ignore) bez porušení původní činnosti.

Systém umožňuje použít disky z jiných počítačů. Pomocí programu DISKDEF lze pozměnit parametry libovolné mechaniky - např. počet stop, počet sektorů na stopě, velikost alokačního bloku, velikost adresáře a počet systémových stop. Jediný neměnný parametr je délka sektoru 512 bajtů.

Bios má zabudovanou obsluhu standardního sériového interfacu RS232C, u něhož můžeme nastavovat všechny parametry (délku slova, počet stop-bitů, paritu i způsob komunikace - handshaking).

Vedle standardních ramdisků lze nastavit i nestandardní ramdisk do kapacity 512K po jednotkách 16K.

V průběhu činnosti systému lze pomocí programu DEFKEY změnit řetězce nadefinované na funkčních klávesách. Původní P-CP/M tuto možnost neměla.

Zavaděč operačního systému je standardní, umožňuje zavedení CP/M z disku pouze pomocí původní romky v počítači - tak není nutno používat pomocný zaváděcí program. Součástí spouštěcí procedury jsou testy přítomné diskety, takže se systém správně zavede z jedno- i oboustranně formátované diskety, i když je připojena mechanika oboustranná.

Použité algoritmy obrazových operací dovolují nejen obvyklý výpis znaků, ale i volbu typu písma (dva různé znakové generátory a vypínatelná inverze znaků), kreslení oken, pseudografiku a rolování oken po znakovém nebo grafickém řádku (pomalu nebo rychle). Pomalé rolování obrazovky po grafickém řádku je velmi pohodlné, prohlížení textu neunavuje. Bios zpracovává také řídicí sekvence pro vložení a vypuštění řádku, což výrazně urychluje činnost textových editorů. Samozřejmě lze používat československou abecedu s diakritickými znaménky podle normy KOI 8 čs2. Tomu je podřízena i klávesnice se třemi přeřazovači (SHIFT, CTRL a GRAPH) - lze vložit všechny znaky s kódy 00H-FFH. Klávesnice je čtena v módu přerušení IM 2, což minimálně zdržuje běh programu. Kódy stisknutých kláves jsou ukládány do vyrovnávací paměti, takže systémové příkazy lze psát i v době vykonávání předchozích. Kromě toho se lze vracet k již odeslaným příkazům - např. při chybně vloženém řádku není třeba jej psát znova - stačí jej vyvolat a opravit.

Součástí systému jsou tyto tranzientní systémové programy:

**FORMAT** - formátování a verifikace disket v různých formátech.

**COPYSYS** - zkopírování operačního systému na různé formáty disket.

**DEFKEY** - definice funkčních kláves F1 až F10. Definici lze provádět kdykoli v průběhu činnosti systému, může být i součástí dávky (SUBMIT).

**TIME** - nastavení údaje systémových hodin.

**SETUP** - definice parametrů diskových jednotek, barev na obrazovce, pípání kláves (CLICK), parametry sériového styku, spouštění programu při zapnutí počítače (AUTOEXEC) a dalších.

**DISKDEF** - pro práci s disketami z jiných počítačů. Po zadání parametrů umožní přenos programů pomocí diskety.

**MLOAD**

**MSAVE** - pro práci s kazetovým magnetofonem. Vhodné k zálohování souborů nebo pro přenos programů mezi Sharpem a jinými pčítači. Použitý záznam je ve standardním formátu Sinclair. Lze volit i dvoj- a trojnásobnou rychlost. Delší soubory se automaticky rozdělí na kratší bloky. Obdobné programy již existují pro řadu jiných počítačů, včetně IBM PC.

**COPYDISK** - fyzické kopírování diskety. Zvláštností programu je jeho naprostá systémová nezávislost. Obdobné programy obvykle manipulují přímo s řadičem.

**BOOT** - ukončení práce operačního systému a návrat do monitoru Sharpu.

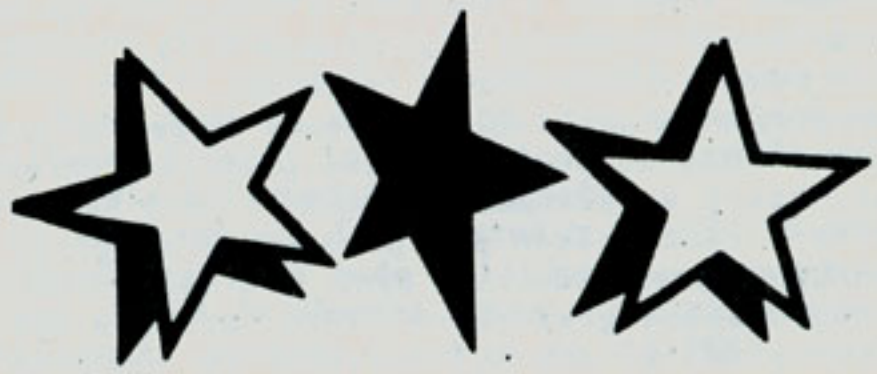
**Jiří Lamač**

---

* V Kabinetu elektroniky MV Svazarmu, Veleslavínská 42, Praha 6, zahájil v lednu svou činnost Klub elektroniky. Jeho činnost je výrazně zaměřena na oblast výpočetní techniky. Pořádají se zde kursy, ve kterých se zájemci mohou na počítačích PP 01 naučit základům elektroniky, informatiky a výpočetní techniky. Výsledkem kursu by měly být znalosti programování a výpočetní techniky přibližně na úrovni výuky středních škol. Kabinet se zaměřuje také na usnadnění přechodu adeptů výpočetní techniky na moderní počítače. A co je zvlášť zajímavé - chystá se na výuku operačního systému CP/M. Pro ten účel se připravuje unikátní učebna, kde bude v počítačové síti za řízení SAPI pracovat několik počítačů PP 01. Podrobnosti o tomto projektu se pokusíme přinést v některém z příštích čísel Mikrobáze. (Meca)

* OV Svazarmu v Praze 4 pořádá v Domě pohybové výchovy na Brumlovce kursy programování v Basicu. Jsou určeny dětem od 9 let i dospělým. Bližší informace na pražském telefonním čísle 42 45 32. (Meca)

* Jistě jste si všimli, že naprostá většina našich drobných zpráv je z Prahy, nebo se Prahy bezprostředně týká. Příčina je v tom, že jen velmi obtížně získáváme informace o mimopražském dění. Proto vyzýváme všechny, v jejichž okolí se něco počítačového děje - dejte nám o tom vědět! Stačí zaslat třeba jen korespondenční lístek na adresu naší redakce. Nezapomeňte uvést zpáteční adresu-pokud to bude něco zvlášť zajímavého, budeme v korespondenci pokračovat, nebo se za vámi přijedeme podívat. (Meca)



# SCREEN DUMP PRO TISKÁRNU D-100 A ZX SPECTRUM


**Ukázka tisku obrázku ze ZX Spectra na tiskárně D100 s uvedeným programem**

Příspěvek se věnuje přenosu obrazových bajtů ze ZX Spectra na tiskárnu D100 s interfacem IRPR. Vlastní zapojení je přesně podle /1/:

```
data  přes PA0-PA7
ACK   přes PB7
SC    přes PC0
```

Rutina pro vyslání obsahu reg.A na tiskárnu:

```
outa    cpl         ;Vyžadují se negovaná data
        out (31),a  ;Data na bránu PA
        xor a
        out (95),a  ;Vyslání signálu STROBE
        dec a
        out (95),a  ;STROBE se zruší
ack     in a,(63)   ;Je tiskárna schopna přijmout
        rla         ;nová data?
        jr c,ack    ;NE - čekej na potvrzení
        ret
```

V pramenu /2/ je sice uveden program COPY pro D100, ale je dlouhý a dost nepřehledný. Můj program je podstatně kratší a nepotřebuje buffer, v němž je sám uložen (tak v něm zůstává místo i na program LPRINT, který tu neuvádím). Navíc pouhými změnami řídících kódů můžeme program použít pro ovládání jakékoli 7-jehličkové tiskárny. Základní algoritmus je shodný s programem v /3/, který se však zabývá 8-jehličkovou tiskárnou.

Registrovým párem HL adresujeme obrazovou paměť. Rotací (HL) se do CY dostane příslušný obrazový bit, který se přesune do reg.A. Zjistí se adresa bajtu v řádku těsně pod předchozím a postup se opakuje celkem sedmkrát. Nyní je v reg A grafický sloupec sedmi grafických řádků. Protože systém D100 vyžaduje, aby byl nejvyšší bit nastaven na log.1 (pro rozlišení grafického bajtu od řídícího kódu) a log.0 určuje tisk příslušné jehly, provede se instrukce AND 01111111 a vstoupí se do rutiny OUTA bez provedení instrukce CPL.

Protože počet řádků je 192 a 192=27*7+3, musíme na závěr vytisknout jen 3 grafické řádky. Když test zjistí, že je překročen řádek 192, bere se každý další řádek jako 0. Maska se zároveň změní na 01110000, a tak se 4 poslední řádky ingorují (192=27*7+7-4).

Nyní zbývá vyřešit zjištění obrazové adresy. V romce ZX Spectra je na adrese 22AAH rutina PIXEL-ADD. Při vstupu musí být v reg.B y a v reg.C x souřadnice grafického bodu. Výstup ukládá adresu bodu do reg.HL.

Rutina však začíná instrukcemi, které jsou pro nás zbytečné:

```
22aa    ld  a,afh
        sub b
        jp  c,24f9h,REPORT-B
```

Úlohou této části je uložení čísla grafického řádku do reg.B. Řádek je ale počítán od vrchu obrazu. První dvě instrukce jsem nahradil takto:

```
        ld  a,195  ;Počet graf.řádků
        sub b
        cp  192    ;Již poslední?
        jr  c,ok   ;NE, pokračuj normálně
        xor a      ;ANO, použij řádek 0
ok      call 22b0h ;Zjisti adresu bodu (souř.C,B)
```

Tím se vyřešil poslední problém. Konečná podoba celého programu je přiložena.

Nyní už záleží jen na uživateli, jak si program dále upraví (doplnění řídících kódů apod.).

**Pavel Kováč**

**Literatura**


/1/ Soldát,J.: Paralelní připojení tiskárny Centronics k ZX Spectru, AR 8/86, str.300
/2/ Formánek,P.: Tiskárna D100 a ZX Spectrum, AR 7/87, str.257
/3/ SCREEN DUMP pro ZX Spectrum, Mikrobáze č.2
/4/ Instrukce k obsluze D100, 2.díl
/5/ Logan,J., O'Hara,F: Complete Spectrum ROM Disassembly, Melbourne House 1983


```
;-----------------------------------------------
;SCREEN DUMP pro tiskárnu D-100 a ZX Spectrum
;         (7-jehličková grafika)
;-----------------------------------------------

        ORG  #5B00        ;Rutina je umístěna
                          ;v tiskovém bufferu
ESC     EQU  #1B
LF      EQU  #0A
CR      EQU  #0D

START   CALL INI          ;Inicializace interfacu
        DEFB 5            ;a tiskárny (graf.mód)
        DEFB ESC,#31
        DEFB ESC,#30
        DEFB #0F
```

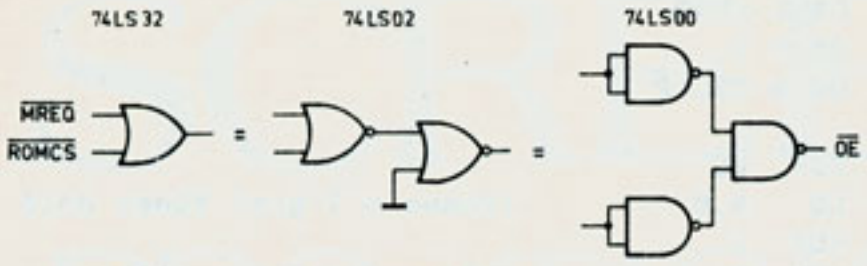
```
        LD   BC,49920    ;195*256
        LD   A,127
        LD   (MASK),A

;Hlavní cyklus

LOOP    PUSH BC

;Vytisknutí řádky (7*256 bodů)

LINE    PUSH BC
        LD   E,7

;zjištění adresy bodu v ramce

COL     PUSH BC
        PUSH AF
        LD   A,195       ;V reg.A číslo řádky počí-

        SUB  B           ;tané od vrchu obrazovky
        CP   192         ;Přesahuje SCREEN?
        JR   C,OK        ;NE - pokračuj normálně
        XOR  A           ;ANO - použij řádek č.0
OK      CALL #22B0       ;Rutina romky; v HL adr.
        POP  AF          ;bodu ve video-ramce
        POP  BC

;Vytvoření grafického bajtu

        RLC  (HL)
        RLA
        DEC  B
        DEC  E
        JR   NZ,COL

;Maska je normálně 01111111, pro poslední řádku
;01110000; bit 7 musí mít log.0 (označení grafic-
;kého bajtu)

        DEFB #E6         ;Kód AND n
MASK    DEFB 0
        CALL PRTGB

        POP  BC
        INC  C           ;Přechod na další sloupec
        JR   NZ,LINE

;Přechod na další řádku (LF na tiskárnu)

        CALL SEND
        DEFB 2
        DEFB CR,LF

        POP  BC
        LD   A,B         ;Posun o 7 graf.řádek dolů
        SUB  7
        RET  C           ;Vše vytisknuto, návrat
        LD   B,A
        CP   6           ;Poslední řádka?
        JR   NZ,LOOP     ;NE - pokračuj
        LD   A,112       ;ANO - masku na 01110000
        LD   (MASK),A
        JR   LOOP        ;a pokračuj dál

;Inicializace 8255

INI     LD   A,130
        OUT  (127),A

;Odeslání znaků na tiskárnu
;n je bajt těsně za instrukcí CALL SEND

SEND    POP  HL          ;Návratová adr.=adr.s n
        LD   B,(HL)
DOIT    INC  HL
        LD   A,(HL)      ;Další znak pro odeslání
        CALL OUTA        ;Vytiskni jej
        DJNZ DOIT        ;Pokračuj
        INC  HL
        JP   (HL)        ;"RET"

;Vytisknutí grafického bajtu

PRTGB   CALL OUTA2       ;Odešli graf.bajt
        LD   A,255       ;Teď bude prázdný sloupec
                         ;vlastně 00000000
OUTA    CPL

;Vytisknutí znaku (bez negace)

OUTA2   OUT  (31),A      ;Data na tiskárnu
        XOR  A
        OUT  (95),A      ;STROBE
        DEC  A
        OUT  (95),A      ;Zruš STROBE
ACK     IN   A,(63)      ;Test připravenosti
        RLA              ;tiskárny
        JR   C,ACK
        RET
```

# VESTAVĚNÍ ISO-ROM DO ZX SPECTRA

Tento článek navazuje na popis ISO-ROM z minulého čísla, ale týká se obecně způsobu připojení pozměněné ROM do ZX Spectra.

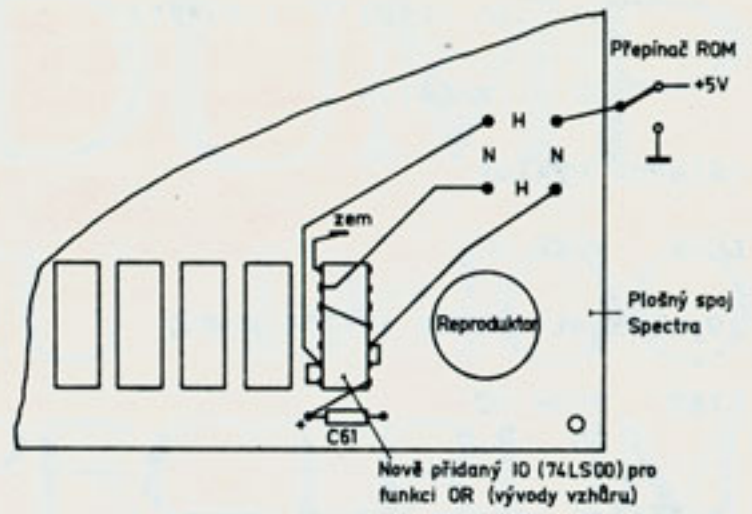
Z důvodu kompatibility programů je nejméně vhodnou cestou nahrazení původní ROMky novou. Vhodnější je zajistit přepínání obou ROMek-původní a nové. Zde se nabízejí dva zásadně rozdílné způsoby umístění přídavné ROM - uvnitř, nebo vně počítače. Umístění přídavné ROMky mimo počítač bylo popsáno např. v Mikrobázi 6/88 (popis modulu pro TURBO-ROM). Jeho velkou výhodou je, že nemusíme zasahovat do počítače. Zmíněné zapojení však znemožňuje současné použití jakýchkoli periferních zařízení. Doplněk by musel být řešen jako průchozí a zapojení by se muselo doplnit blokováním původní ROMky současně s EPROMkou, ovládanou signálem /ROMCS z periférií. Umístění přídavné ROMky uvnitř počítače je jednodušší a praktičtější. Mimo jiné je pak i sběrnice kapacitně méně zatížena. Přináší to však nutnost demontáže skříňky a vestavění vhodného přepínače.

Po zkušenostech lze doporučit montáž ISO-ROM do počítače tak, že bude umístěna shora na původní ROMce. Pokud je původní ROMka v patici, musíme ji odstranit, protože jinak by se obě ROM nevešly na výšku. Zapojení doporučované výrobcem ISO-ROM je příliš komplikované. Existuje jiný, mnohem jednodušší způsob doporučovaný tuzemskými propagátory ISO-ROM (JJn). Je založen na tom, že ROMky mívají na vývodu 27 negovaný CS, zatímco většinu




**Obr.1 Realizace funkce OR z dostupných IO ▲**

**Obr.2 Zapojovací plánek přepínání ROMek ▶**




EPROMek lze na tomto vývodu logickou nulou naopak odselektovat. Tento princip umožňuje připájet EPROMku všemi vývody na ROMku a vlastní přepnutí provést změnou potenciálu jediného vývodu. Přes nestandardní zapojení funguje EPROM bezvadně. Zbytek zapojení je obvykle řešen pomocí dvou diod a odporu, což - jako zcela nevyhovující - už bylo příčinou mnoha zklamání. Paralelně zapojené vstupy /OE dvou EPROMek totiž rozhodně nestačí uzemnit odporem doporučované hodnoty 10 kiloohmů. Ani výrazně nižší hodnota není zárukou bezvadné funkce, navíc by příliš zatížila sběrnici.

Stalo se, že takto zapojená EPROMka půl roku pracovala bezvadně a až po této době začal systém nevysvětlitelně náhodně vypadávat. Testovací programy nic neodhalily. Až osciloskop zapojený na /OE ROM+EPROM ukázal, že úroveň se tu pohybuje v zakázané oblasti a navíc se s časem poněkud mění. Proto byla tato část zapojení nahrazena jednoduchým hradlem OR (dostupný IO 74LS00 zapojený jako OR; lze použít i 74LS32 nebo 74LS02). Problémy rázem zmizely a vše funguje už další rok bez závad. Schéma i zapojovací plánek jsou na připojených obrázcích.

A ještě drobnost k otázce určení vhodného okamžiku přepnutí ROMek. U vnějšího připojení TURBO-ROM v Mikrobázi 6/88 je hardwarově zajištěno, aby k přepnutí nemohlo dojít během práce procesoru s ROMkou. Ve všech dalších popisech přídavných ROMek je vždy upozorňováno na nebezpečí zhroucení systému při přepnutí ROM v době jejího čtení. V praxi se ukázalo, že při práci s popsaným jednoduchým připojením ISO-ROM k žádným poruchám během přepnutí nedochází, pokud program zrovna pracuje v rutinách shodných v ROM i EPROM. Hardwarovým specialistům se při tomto tvrzení možná zježí vlasy, ale je to tak.

Po odevzdání rukopisu s popisem vlastností ISO-ROM jsem si vzpomněl ještě na jednu její příjemnou maličkost. V tabulce parametrů basicových příkazů je u příkazu PAUSE změněna třída parametru 06 na 03. Původní třída 06 předepisovala, že vždy musí následovat číselný výraz, zatímco třída 03 automaticky dosadí nulu, když tento výraz schází. V praxi to znamená, že při použití ISO-ROM nemusíme v případě čekání na tlačítko vypisovat PAUSE 0, ale jenom PAUSE. Znalci už vědí, že nejde jen o ušetřený stisk klávesy "0", ale že i v paměti se pokaždé ušetří sedm bajtů. Stejnou výhodu nabízí ISO-ROM také v příkazu SAVE "jméno" LINE n. Pro autostart od řádku 0 není třeba nulu po LINE zapisovat díky tomu, že v rutině SA-LINE-1 na adrese 0723H je volán podprogram FETCH-NUM místo EXPT-1NUM. Vzhledem k menší četnosti tohoto příkazu v programech to však zase není až tak velká výhoda.

**Daniel Meca**

# SPECTRISTICKÉ FINESY

Po přečtení článku "Co v manuálu nenajdete" (Mikrobáze 1/88) jsem se rozhodl navázat na zmíněný článek. Uvedené finesy ocení zvláště uživatelé, kteří ve svých programech často používají text.

**Využití operačního systému**

| | |
|---|---|
| RANDOMIZE USR 3330 | scroll strany (21 řádků) rychle nahoru |
| RANDOMIZE USR 3583 | scroll spodní poloviny obrazu nahoru o 1 řádek (spodní část se zasouvá do horní včetně atributů) |
| RANDOMIZE USR 3652 | spodní polovina obrazu rychle zmizí |
| RANDOMIZE USR 3280 | pomalý posuv celé strany nahoru o 1 řádek (lze využít smyčky FOR-n-NEXT pro posuv /n/ řádků bez otázky SCROLL?) |
| RANDOMIZE USR 3190 | obdoba předchozího příkazu včetně atributů |
| RANDOMIZE USR 3212 | příkaz vyvolá otázku SCROLL? |
| RANDOMIZE USR 4757 | vypíše se hlášení (C) Sinclair Research Ltd. |
| RANDOMIZE USR 0 | příkaz provede RESET počítače |
| RANDOMIZE USR 1551 | příkaz ekvivalentní VERIFY |
| RANDOMIZE USR 6040 | příkaz ekvivalentní ENTER |
| RANDOMIZE USR 7032 | příkaz ekvivalentní BREAK |
| RANDOMIZE USR 8000 | příkaz ekvivalentní PAUSE 0 |

Pro tvorbu borderových efektů (podobných práci s magnetofonem) vyzkoušejte tyto adresy:

se zvukem (1267, 1269, 1278, 1331, 1333, 1334)
bez zvuku (1248, 1251, 1290, 1314-1325, 1327)

PRINT 7997-USR 7997 vytiskne čas od stisknutí poslední klávesy

```
ANTIMERGE:
LET X=256*PEEK 23636+PEEK 23635:
POKE X,60:POKE X+1,0

ANTIBREAK:
POKE 23613,PEEK 23730-5
```

**Simulace psacího stroje**

```
 90 LET W$=" ZDE VLOZTE LIBOVOLNY TEXT "
100 FOR K=PI/PI TO LEN W$:PRINT">";:PRINT CHR$
    VAL"8";W$(K);: BEEP.01,10:PAUSE PI/PI:NEXT K
```

Poznámka: pro ">" užijte BRIGHT v přímém módu, tj. EM+9; pro zrušení zesvětlení užijte EM+8

**Dvoubarevný border**

```
FOR X=1 TO 100:BORDER 7:BORDER 0:PAUSE 1:
BORDER 0: BORDER 7:NEXT X
```

**Pohyb libovolného textu v libovolném řádku**

```
 1 LET B$="              32 znaků              ":
   LET A$="              32 znaků              ":
   FOR A=1 TO 32:PRINT AT X,0;INK 0;BRIGHT 1;
   A$(33-A TO 32):PRINT AT X+1,32-A;INK 0;
   BRIGHT 1;B$(1 TO A):NEXT A
```

Program je možno užít k pohybu textu na obrazovce. V našem případě se text uložený v A$ pohybuje na řádku X zleva a text uložený v B$ se pohybuje na řádku X+1 zprava.

**Několik užitečných pouků**

| | |
|---|---|
| POKE 23613,82 | odpojí rutinu BREAK |
| POKE 23613,84 | odpojí klávesu BREAK |
| POKE 23570,10 | odpojí klávesu EDIT |
| POKE 23570,16 | zastaví listing při stisku klávesy EDIT |
| POKE 23736,181 | autoSAVE (nahrávání bez systémového hlášení) |
| POKE 23743,80 (mimo ed. řádek) | na obrazovku nelze nic psát - např. pro zabránění poškození obrázku (zákaz nutno v programu odvolat) |
| POKE 23743,83 | zrušení zákazu psaní na obrazovku |
| POKE 23624,120 | způsobí BRIGHT řádků 22 a 23 |
| POKE 23693,120 | způsobí BRIGHT řádků 0-22 |

**Ing.Petr Wodak**

# K UNIVERZÁLNÍ TISKOVÉ RUTINĚ Z MIKROBÁZE 7/88

Se značným zpožděním, ale přece se nám dostal do rukou dopis ing. Pavla Jáneše, který svou původní rutinu T-3.1 pro ZX Spectrum vylepšil v nové verzi T-3.3. Rádi ji zveřejňujeme, protože se týká jedné z nejčastějších oblastí využití počítače. Průvodní text z Mikrobáze 7/88 zůstává v podstatě beze změny. V rutině pro přenos dat paralelním interfacem s obvodem 8255A je nově zařazen test stisku BREAKu pro případné přerušení tisku.

```
;------------------------
;      Univerzální
;     relokovatelná
;    tisková  rutina
;------------------------
;Pro ovládání grafiky tis-
;kárny: POKE start+102,198
;       POKE start+103,165
;Zpět na značení neznámého
;znaku: POKE start+102,62
;       POKE start+102,42
;Délka řádku:
;       POKE start+185,(x)
;Délka programu: 206 bajtů
;------------------------

        ORG  23300

CWR     EQU  130
PA      EQU  #1F
PB      EQU  #3F
PC      EQU  #5F
RR      EQU  #7F
STAT    EQU  #5CC5

INIT    LD   A,CWR
        OUT  (RR),A
        LD   HL,TOK
        ADD  HL,BC
        LD   (STAT),HL
        DEC  HL
        IN   A,(PB)
        RLA
        LD   A,0
        LD   (HL),A
        JR   C,NAST
        INC  (HL)
        RET
NAST    CPL
        OUT  (PC),A
        RET

SAT     DEFB 0,0,0

;STA+0...INDIK.TAB
;STA+1...POCITADLO
;SAT+2...DETEKTOR
```

```
ZACAT   PUSH AF
        LD   IX,(STAT)
        DEC  IX
        DEC  IX
        DEC  IX
        LD   A,(IX+0)
        CP   23
        JR   Z,TAB
        POP  AF
        CP   23
        JR   Z,NAVR
        CP   6
        JR   Z,CARKA
        CP   12
        JR   NZ,ENTER
        LD   (IX+1),0
        JR   TISK
ENTER   CP   13
        JR   NZ,DAL
KON     LD   (IX+1),0
        LD   A,13
        CALL #0C40
        JR   PTISK
        LD   (IX+1),0
        LD   A,10
        JR   TISK
DAL     CP   32
        JR   C,TISK
        CP   127
        JR   C,TISK
        SUB  165
        JP   P,#0C10
        LD   A,42
        JR   TISK
NAVR    LD   (IX+0),23
        DEC  (IX+1)
        RET
TAB     POP  AF
        DEC  A
        SUB  (IX+1)
        JR   TAM
CARKA   LD   A,(IX+1)
        LD   B,A
        ADD  A,16
        AND  #F0
        SUB  B
TAM     LD   B,A
```

```
MEZ     LD   A,32
        CALL #0C40
        JR   PTISK
        DJNZ MEZ
        LD   (IX+0),0
        RET
PTISK   DEC  SP
        DEC  SP
        POP  DE
        INC  DE
        INC  DE
        PUSH DE
TISK    PUSH AF
        LD   A,(IX+2)
        PUSH AF
        CALL #1F54
        JR   NC,BREAK
        POP  AF
        OR   A
        IN   A,(PB)
        JR   NZ,ACK
        RRA
        JR   C,TISK+1
        POP  AF
        OUT  (PA),A
        XOR  A
ZPET    OUT  (PC),A
        CPL
        NOP
        OUT  (PC),A
        LD   A,(IX+1)
        INC  A
        CP   65
        JR   NC,KON
        LD   (IX+1),A
        RET
ACK     RLA
        JR   C,TISK+1
        POP  AF
        CPL
        OUT  (PA),A
        LD   A,#FF
        JR   ZPET
BREAK   RST  8
        DEFB 20

TOK     EQU  ZACAT-INIT
```

# PÉCÉČKOVÝ DĚJEPIS

Naše nová rubrika je věnována osobním počítačům kompatibilním s IBM PC. Nic naplat - v celém počítačovém světě je to žhavá současnost. Nelze vycházet z toho, že u nás je tato oblast pro amatéry (žel mnohdy nejen pro ně) hudbou budoucnosti. Vše směřuje k tomu, že se s něčím podobným všichni setkáme, i když "někdo dříve a někdo později" (a kdo ho ještě nemá, ať neztrácí naději...).

Na čtenáře různých časopisů či jiných publikací věnovaných problematice výpočetní techniky se ze všech stran valí záplava různých odborných termínů jako IBM PC, PC/XT, PC/AT, hard disk, kompatibilita s IBM, CGA, EGA, VGA atd. Pro člověka zasvěceného do tajů světa počítačů IBM PC a kompatibilních (a je to tu zas) jsou to samozřejmě slova plně srozumitelná Ale přiznejme si - kolik z nás dokázalo překročit stín svého ZX Spectra, Atari 800, Sharpa či jiného osmibitového počítače tak, že všem těm slovům plně rozumí? Vyjděme z velmi reálného předpokladu, že dosavadní okruh čtenářů Mikrobáze se z velké části rekrutoval právě z řad majitelů domácích počítačů výše uvedené kategorie. I ti však, v naprostém souladu s požadavky doby, pošilhávají po výkonnějším zařízení. Přestože ceny osobních počítačů kategorie IBM PC se za poslední období opět posunuly směrem dolů, jsou to sumy pro našince stále ještě úctyhodné. Když si navíc představíme anabázi, kterou zájemci o získání této techniky musejí podstoupit, je zřejmé, že budou předem vyhledávat dostupné informace o předmětu svého zájmu, aby si ujasnili, po čem to vlastně touží - jaké to má technické parametry, jaké programy lze na takových počítačích použít...

Právě jim chceme v nové rubrice PC nabídnout seriál článků na témata spojená s touto kategorií počítačů. Dalšími pozornými čtenáři budou jistě i ti, kteří se do styku s těmito počítači dostanou třeba v zaměstnání a zatouží se o svém pracovním nástroji dozvědět něco víc, resp. si ujasnit doposud jen tušené (i manuály bývají k dispozici jen v jazycích, mezi nimiž naše mateřština nebývá).

Seriál volně navazujících článků o problematice počítačů kategorie IBM PC a slučitelných postupně přinese informace o tom, jak a proč vlastně počítač této kategorie vznikl, co bylo příčinou jeho rychlého rozšíření do celého světa a konečně jaké jsou trendy dalšího vývoje této oblasti. Řeč bude i o tom, z čeho se takový počítač skládá (o jeho hardwaru) a samozřejmě i o tom, co mu vdechuje život - o softwaru.

## Skromný začátek - IBM PC

Úvodem si povíme, jak to všechno začalo. V roce 1981 se zdálo, že počítačový trh je stabilní a že nikdo v dohledné době neohrozí postavení osmibitových mikropočítačů pracujících s diskově orientovaným operačním systémem CP/M. Za této situace přišel přední světový výrobce tzv. velké výpočetní techniky, americká firma IBM, s novým počítačem označeným zkratkou PC (Personal Computer) na bázi 16-bitového mikroprocesoru Intel 8088 s 8-bitovou datovou a 20-bitovou adresovou sběrnicí.

Základní koncepce PC se udržela dodnes. Mikroprocesor je spolu s dalšími obvody pro jeho styk s okolím na tzv. hlavní desce (main, resp. mother board), která obsahuje i základní paměť RAM a ROM, obvody pro styk s klávesnicí, obvody přerušení a krystalem pro řízení pracovního taktu mikroprocesoru. Na desce jsou dále konektory pro připojení rozšiřujících desek (slots) s doplňujícím technickým vybavením. Jsou to např. další pamětové obvody pro rozšíření základní paměti, deska s obvody zajišťujícími V/V operace, sériová, ev. paralelní komunikace s tiskárnou apod. Další rozšiřující desky obsahují obvody pro tvorbu grafiky, řadič (controller) disketových jednotek a případně i pro další volitelné funkce, rozšiřující základní schopnosti počítače. Základní deska je spolu s jednou nebo dvěma mechanikami pro pružné disky 5,25" a zdrojem potřebných napájecích napětí - umístěna v plechovém pouzdru, které slouží jako podstavec pro monitor. Klávesnice je k počítači připojena krouceným kabelem.

V době uvedení prvních péčéček to však zdaleka nebyly ony výkonné stroje, jak je známe dnes. Tehdy obvykle měly 64K paměti RAM, malou paměť ROM, monochromatický (zelený) monitor jen s textovým módem (25 řádků po 80 znacích), jednu disketovou mechaniku s kapacitou 160K, pracovní takt 4,77 MHz a operační systém PC DOS 1.0. Péčéčko tenkrát zdaleka nebylo nejvýkonnějším počítačem. Přineslo však několik zajímavých a hlavně užitečných novinek. Základní výhodou nového systému byla především jeho velmi snadná rozšiřitelnost a modifikovatelnost jednoduchým zasunutím přídavných desek do volných pozic na základní desce. Promyšlená konstrukce, kompaktnost, výkonnost, dobrá klavesnice a v neposlední řadě i zvuk jména výrobce nakonec vedly k tomu, že mnohá zavedená firma začala napodobovat produkt IBM. Zvyšující se počet prodaných péčéček a jeho kopií přiměl softwarové producenty zvýšit produkci pro jejich uživatele. To zpětně vedlo ke zvýšené poptávce o samotný počítač. Nakonec z toho byl celosvětový úspěch "Modrého giganta" či "Velké modré" (Big Blue), jak je IBM přezdívána.

## Pod tlakem konkurence - IBM PC/XT

Dosažený úspěch nepřinesl IBM jen aktiva, ale i nemalé komplikace. K zavedeným výrobcům výpočetní techniky se totiž začali řadit dosud úplně neznámí producenti (zejména ze zemí Dálného Východu), kteří s využitím levné pracovní síly a levnějších součástek (zprvu japonských, posléze vlastních) zaplavovali svět více či méně dokonalými kopiemi typu PC. Jenže - stejně výkonné kopie se nabízely za podstatně nižší cenu než originál. Objevily se i kopie s vyšším výkonem, než jaký mělo originální péčéčko, ale s cenou jen nepatrně vyšší. Zpočátku měli dálněvýchodní výrobci problémy z plnou slučitelností svých produktů. Po jejich překonání a po dosažení odpovídající spolehlivosti se kompatibilní počítače staly více než mocnou konkurencí IBM.

Reakce Big Blue na sebe nedala dlouho čekat. Vylepšený model IBM PC XT (eXtended Technology-rozšířená technologie) přinesl hned několik zásadních změn - kartu barevného grafického zobrazení CGA, jednotku floppy disků s kapacitou 360K, novou klávesnici typu XT, 256K paměti RAM (rozšiřitelné až do 640K), možnost doplnění hlavní desky matema-

tickým koprocesorem pro zrychlení operací a hlavně možnost připojení hard disku.

Nástup konkurenčních výrobců však byl ještě rychlejší než v případě původního pécéčka, brzy se objevily počítače shodné s novým vzorem. Nové kopie opět začaly nepříjemně konkurovat IBM jak cenově, tak i výkonnostně. Zbavovaly ji výlučného postvení na trhu a připravovaly ji o nemalé zisky. IBM se samozřejmě snažila nepříjemný tlak konkurence eliminovat nejen na poli ochrany autorských práv, ale i vývojem nových typů svého úspěšného počítače.

## Další kolo zápasu - IBM PC/AT

V krátké době byl na počítačový trh uveden model IBM PC AT (Advanced Technology - pokročilá technologie). Přinesl zásadní změnu - použití nového, plně 16-bitového mikroprocesoru Intel 80286, který znamenal nejen podstatné zvýšení výkonu, ale i možnost současného běhu několika programů najednou - multitasking (samozřejmě jen s odpovídajícím programovým vybavením). Dalším rozšířením oproti PC XT byl nový standard zobrazení v barevné grafice EGA (Enhanced Graphics Adaptor), slučitelný s předchozím grafickým adaptérem CGA (Color Graphics Adaptor). Novinkou byla také paměť RAM vyrobená technologií CMOS, zálohovaná miniaturní baterií pro uchování informace o základním nastavení PC i po vypnutí počítače. Disketová mechanika floppy 5,25" měla volitelnou kapacitou 360K nebo 1,2 MB pro zajištění kompatibility (slučitelnosti) s předchozími modely pécéčka na úrovni diskového pamětového media. Klávesnice typu AT měla více tlačítek a lepší ergonomii.

Nový typ byl standardně dodáván s 512K paměti RAM, jednou disketovou mechanikou, základními obvody pro V/V operace, grafickou kartou EGA, klávesnicí typu AT a výkonnějším zdrojem provozních napětí - až do 200 W. Architektura stavby počítače byla zachována při dodržení kompatibility nejen programového vybavení, ale i na úrovni rozšiřujících a doplňujících desek s dalším technickým vybavením. V případě těchto desek jde o slučitelnost směrem dolů - to znamená, že vše z PC XT lze použít na AT, ale opačně lze použít jen ty rozšiřující desky, které využívají sloty typu XT. V PC AT jsou z těchto důvodů vestavěny sloty obou typů.

Stejně jako předchozí modely, i PC AT se v krátké době dočkalo řady více či méně úspěšných kopií. Pro IBM je to jistě zarmucující, ale pro našince nesporná výhoda, protože průvodním jevem tohoto napodobování je stálý pokles cen všech kategorií pécéček, a to originálů i kopií.

## A jedeme dál - IBM PS/2

Vývoj zastavit nelze, model PC AT nemohl zůstat posledním. Vzhledem k produkovanému množství počítačů IBM/XT/AT po celém světě lze o nich hovořit jako o celosvětovém standardu. V nových pécéčkách je už 32-bitový mikroprocesor Intel 80386. V tomto případě však nositelem vlajky pokroku nebyla IBM, ale její konkuret Compaq se svým modelem Compaq Deskpro 386. Vývoj pokračuje směrem k neustále výkonnějším počítačům s velkou vnitřní pamětí RAM a ještě větší vnější pamětí na hard discích, dnes i s optickým záznamem. Zatím posledním želízkem v ohni firmy IBM je řada mikropočítačů označená zkratkou PS/2 - Personal System/2. Zachovává základní vnější koncepci řady PC, ale vnitřně je poněkud odlišná. S předchozí řadou je slučitelná na úrovni programového vybavení. Jestli se opět stane základem celosvětového standardu, ukáže čas.

## Na nic jsme nezapoměli - PC/Jr a PC Portable

V uvedené stručné genezi počítačů IBM nebyly úmyslně uvedeny "nepovedené" modely. IBM PC/Jr (Junior) byl levnější, méně vybavenou variantou pécéčka. IBM Portable představoval přenosný model. Oba nepřinesly firmě podstatný obchodní úspěch, IBM zde v podstatě vyklidila pole jiným výrobcům. Přes veškeré polemiky o výkonnosti či přínosu počítačů kategorie PC/XT/AT je zřejmé, že IBM svým rozhodnutím vstoupit na mikropočítačový trh, vytvořila nový průmyslový standard v malé výpočetní technice. A to nejen v principech činnosti ale i v designu a způsobu nasazení.

Popisem činnosti jednotlivých prvků a skupin PC/XT/AT se budeme podrobněji zabývat v samostatných článcích. Nyní opustíme oblast hardwaru a podíváme se na to, bez čeho by sebelepší počítač zůstal jen hromádkou kovu, plastů a křemíku. Zaměříme se na software.

## Operační systém MS DOS alias PC DOS

Začneme základním programovým vybavením - operačním systémem, který je opět zaměřen diskově (DOS - Disc Operating System). Jeho základ vytvořil Tim Paterson v roce 1979. Všechna práva na tento systém má od roku 1981 vyhrazena firma Microsoft- proto název MS DOS. IBM jej zvolila jako základní programové vybavení pro počítače řady PC. První prodejní verze pro počítače IBM PC (PC DOS verze 1.0) zdaleka ještě nedosahovala dokonalosti a propracovanosti těch dnešních. Od jejího uvedení na trh se objevilo sedm dalších verzí - průměrně jedna ročně. Všechny jsou neustále vylepšované, ale také zabírají stále více paměti počítače. Vývoj je to zcela pochopitelný. Vždyť od původních 64K paměti se postupně přešlo k 640K a od původních 160K na floppy disku až k dnešním 1,44 MB na disketě formátu 3,5".

Operační systém PC DOS má čtyři základní části. Tři z nich jsou dodávány na disketě a do počítače se zavádějí při každém spuštění. Jsou to - IBMBIO.COM, IBMDOS.COM a COMMAND.COM. Tytéž programy pro použití v kompatibilních počítačích Microsoft dodává pod názvy IO.SYS, MSDOS.SYS a COMMAND.COM.

IBMBIO.COM obsahuje podprogramy pro zabezpečení V/V operací. Vlastní operační systém IBMDOS.COM zabezpečuje činnosti uživatelského programového vybavení a realizuje práci s diskovými soubory. COMMAND.COM je tzv. interpreter příkazů - vykonává příkazy zadávané z klávesnice poté, kdy se systém ohlásí na monitoru znaky A:. Poslední částí operačního systému je program BIOS, trvale umístěný v paměti ROM. Obsahuje programové rutiny pro inicializaci počítače po zapnutí a prostřednictvím IBMBIO.COM zajišťuje styk programového vybavení s V/V zařízeními počítače. Analogie se systémem CP/M, který má ve stejných funkcích obdobné části BIOS, BDOS a CCP, není náhodná. CP/M dala základ většině dosud používaných operačních systémů.

S každou verzí operačního systému se dodává sada programů se souborem systémových služeb, které se používají příležitostně, pročež nemusejí být stále přítomné v operační paměti. Je to např. program pro formátování disket, programy pro modifikaci klávesnice, pro nastavení způsobu komunikace počítače s okolím a další (viz odpovídající tranzientní systémové programy v CP/M).

Vyšší verze operačního systému PC DOS se objevily s příchodem nových modelů pécéček. Tak PC DOS 2.0 byl dodáván společně s IBM PC/XT, PC DOS 3.0 nastoupil s typem IBM PC/AT. Dílčí verze (jako PC DOS 2.1) byly vytvořeny pro IBM PC/JR a IBM PC Portable.

### Nový systém - PC DOS 2.0

Zásadní zlom v pojetí operačního systému PC DOS přinesla jeho verze 2.0. Dnes lze spíše jen vyjímečně potkat programový produkt pracující s PC DOS 1.0 nebo 1.1. Vzhledem k tomu, že PC DOS 2.0 přinesl tolik podstatných změn, mluví se o něm jako o úplně novém operačním systému, vlastně jako o základním systému počítačů třídy IBM PC a následujících. Ve srovnání s předchozími verzemi má nová tyto základní odlišnosti:

- Stromovou strukturu katalogu disku - systém logického, hierarchického seskupování souborů na disku do podkatalogů. Přínos spočívá v rychlé a přesné orientaci v obsahu disku.
- Změnu způsobu formátování disku. To m.j. přineslo zvýšení kapacity ze 320 na 360K (9 sektorů po 512 bajtech místo 16 sektorů po 256 bajtech).
- Mnoho drobnějších úprav rozšiřujících možnosti operačního systému.

### Blíž k uživateli - PC DOS 4.0

Vývoj operačního systému pokračoval přes verze 3.0 pro IBM PC/AT a modifikace ve verzích 3.1, 3.2 a 3.3 k zatím poslední verzi PC DOS 4.0. Ta přišla vlastně jaksi navíc a přináší do operačního systému prvky grafiky a řízení pomocí myši.

Podrobnější popis jednotlivých verzí operačního systému PC DOS alias MS DOS, včetně popisu systémových příkazů, bude obsahem některého z dalších článků v rubrice PC.

### Budoucnost - OS/2?

Úplně novou a samostatnou kapitolou je operační systém OS/2 pro IBM PS/2 s mikroprocesorem 80286, resp. 80386. To je ovšem už úplně jiný příběh.

**Judr.Štěpán Buranyč**

## Error

V článku **Předejděte katastrofám pevného disku (Mikrobáze 7/88)** nám nedopatřením vypadlo jméno spoluautora textu. Tímto se Ladislavovi Jeriemu omlouváme.

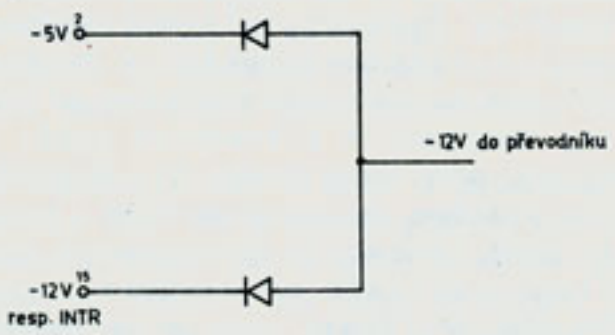
Po otisknutí článku **Osmibitové převodníky A/D a D/A pro PMD-85 (Mikrobáze 8/88)** jsme od jeho autorů dostali dopis s několika opravami (tentokrát je v tom redakce nevinně):

1) Ve schématu chybí kousek vodiče spojujícího vývod 1 dekodéru 3205 a vývod 5 obvodu 8255 (vodič končí vlevo dole bez označení).

2) V označení přívodů napájecího napětí v tomtéž schématu je chyba: napětí +12 V má mít správně číslo 30 (nikoli 2).

3) Pro zvýšení bezpečnosti a odolnosti zařízení k pozdějšímu připojení -12 V doporučujeme do přívodního konektoru FRB umístit dvě křemíkové diody (např. KA 20X) podle uvedeného obrázku. Tak zabezpečíme připojení alespoň -5 V a znemožníme průnik -12 V do počítače, když není upraven.

Autorům za dodatečnou opravu chyb děkujeme.




Další chůze po tenkém ledě této rubriky se netýká redakčních šotků, ale **způsobu zasílání příspěvků do Mikrobáze.** Alespoň pár těch zcela a úplně hlavních zásad, které byste měli při komunikaci s redakcí dodržet:

1) **Každý příspěvek musí obsahovat tyto údaje: jméno, adresu bydliště, číslo OP a rodné číslo autora.** Když chybí jméno, musíme u příspěvku uvést "autor neznámý". Když chybějí ostatní údaje, nemůžeme příspěvek hladce proplatit (a po jeho uveřejnění následuje zbytečná korespondence a zdržení).

2) Článek pište jako článek, nikoli jako dopis, nad kterým pak redaktor dostává jazykové křeče. Někdy takový dopis ani nejde rozumně přepsat do formy článku, takže nevyjde. Uvědomte si, že ve vztahu k časopisu jste autoři, tak jako skuteční autoři postupujte.

3) Programové výpisy se snažte vytisknout opravdu černě na bílém a hezky (raději velmi hezky) upravené. Nelze přetisknout výpisy, které pokračují přes ohyby perforovaného papíru nebo když konce řádek výpisu zůstaly na válci vaší tiskárny. Když vám zešedla páska, tiskněte bez ní přes uhlový papír, zakrytý jedním tenkým papírem. Pěkným výpisem se všichni vyhneme nepříjemným chybám plynoucím z přepisu. Někdy i vícenásobné kotrole nějaká ta chybička unikne. Pochází-li váš text, resp. i program ze ZX Spectra, pošlete nám je na kazetě. My si je převedeme přímo do péécéčka a kazetu vám pošleme zpátky. To se týká i všech textů připravených na standardu IBM PC (disketu zkopírujeme a pošleme obratem zpět).

4) Obrázky kreslete čím chcete, ale hlavně výrazně. Naprostá většina obrázků se překresluje. U žádného však nesmí chybět jeho název a číslo, na něž se odvoláváte v textu.

5) Když nám pošlete neotřelé hardwarové zapojení, takřka stoprocentně je otiskneme. Totéž se týká opravdu nápaditých a krátkých programů v jakémkoli jazyku. Nemůžeme otiskovat dlouhé "klády" nebo začátečnické prográmky jako opsané z manuálu počítače. Vždy máte možnost se předem dohodnout.

6) Zvláštní pozici mezi všemi materiály mají články, které neobsahují žádný software ani hardware. Pokud informují o něčem opravdu zajímavém nebo podnětném, jsou vítány. Vzhledem k malé tiskové ploše však nemůžeme dávat přednost dlouhým článkům o tom, jak by mělo předpisově vypadat pracoviště počítačového amatéra nebo kanceláře, neopodstatněným futurologickým statím o budoucnosti světa, nic neříkajícím překladům o zahraničních veletrzích atd. apod. Naopak uvítáme jakýkoli článek, který se na počítačovou problematiku dokáže podívat z profesionálního nadhledu a něco nám všem opravdu dá. Budete-li se chystat psát o čemkoli "mimopočítačovém", raději se s námi napřed domluvte, ať zbytečně neztrácíte čas.

7) Nakonec k překladům. Nebudeme se jim úplně vyhýbat, ale absolutní přednost mají původní materiály - tak jako v každé redakci, která nemá překlady ve svém erbu. Zde opět platí - raději se domluvte předem.

Věříme, že při dodržení těchto nezbytných zásad se naše společná tvorba časopisu projeví i jeho stoupající kvalitou.

# POČÍTAČ VĚZNĚM UČITELŮ

2. února 1989 jsem sledoval televizní pořad Logicky vzato. Měl trochu jiný kabát než obvykle. Dlužno říci, že koncepčně mu celkem slušel, i když pod ním štípal mráz. Moderátor a spolutvůrce cyklu se obrátil na zástupce středoškolských učitelů informatiky a vyzval je, aby se vyjádřili k úrovni pořadu. Beseda se však dotkla témat mnohem širších.

Pod titulem A co dál? byla proložena několika rozhovory se žáky tříd zaměřených na informatiku. Tak jsem se dozvěděl, že žáci programují vlastně jen tužkou po papíru. Ve školním roce 87/88 je pustili k počítači třikrát! A v tomto školním roce ještě ani jednou!!! A to přesto, že škola počítače má. Někde se dokonce učí Pascal, který ale na školních počítačích není implementován. Kocourkov jak vymalovaný.

Moderátor se logicky snažil na učitelích vyzvědět, proč školní počítače nejsou žákům volně přístupné. Z odpovědí mi šel mráz po zádech:

Prvním důvodem držení počítačů pod zámkem je prý zlodějna. Čtete dobře - žáci tříd specializovaných na informatiku prý jsou pobertové. A zřejmě nejraději kradou vybavení počítačů IQ 151. Třeba- jak naznačila jedna učitelka - by určitě ukradli nějaký koaxiální kabel. V případě zpřístupnění počítačové učebny by proto učitel musel být s nimi a dávat na ty gymnazijní kleptomany pozor (ne snad že by se žáky pracoval, on by prostě bachařil). A tohle říkají učitelé, kteří se na výchově svých žáků denodenně přímo podílejí!

Druhý důvod tkví v tom, že by učitel se žáky v učebně vůbec musel pobývat. Když on přece má svoje "padla", odhodí cihlu (pardon - učebnici) a frčí domů. Již od věků je povolání lékaře a učitele zcela přirozeně považováno za poslání. U dnešních učitelů gymnázií je to zřejmě jinak. Vím, že to neplatí pro všechny, ale ti besedující mezi ně zřejmě nepatří. Nabyl jsem z nich dojem, že výuka informatiky je dost obtěžuje. Potvrdili to svou výpovědí, když se je moderátor ptal, jak si počínají ve třídě s basicovými démony (žáky, kteří jsou svými znalostmi před ostatními). Přítomní se vyjádřili v tom smyslu, že takový případ nesmírně zvyšuje nároky na učitele (!). I když - jak prohlásila jedna učitelka - takovým démonům sice létají prsty po klávesnici, ale žádný větší program nedokáží vytvořit (připomínám výpověď žákůprsty ji "létaly" po klávesnici v loňském školním roce třikrát a v letošním ještě neměly šanci). Dotyčná učitelka měla zřejmě na mysli létání prstů po tatínkově počítači doma. Ale jaksi nikoho nevzrušilo, že nevyzrálost démona studujícího v počítačové třídě vůbec není jeho vinou (natož jeho tatínka). Zabývat se tím, proč se takovým démonům (zvláště jsou-li opravdu nadaní) nevěnuje individuální péče, šlo mimo rámec bontonu přítomných.

Musel jsem vzpomenout některých svých učitelů, kteří mně i jiným spolužákům věnovali spoustu času, když vycítili opravdový zájem o svůj předmět. Třeba jak mě fyzikář "prohodil" dveřmi kabinetu, když jsem mu (nerad) rozbil filmovou převiječku. A jak si mne druhý den zavolal, načrtl mi, jak to v ní funguje a nechal mne s ní, dokud ji neopravím. A jak mě pak hřála jeho pochvala. Nebo jak mne s několika dalšími spolužáky vodila učitelka kreslení po výstavách a zasvěcovala nás do tajů výtvarného umění. Jak naše skvělá češtinářka naučila celou třídu rozumět literárnímu jazyku i mimo osnovy... Tihle učitelé uměli tisíckrát víc než předepisovala školní látka. Byli to pedagogové tělem i duší, zaníceně rozdávali znalosti a cit. A nacházeli v tom smysl svého životního poslání. Žeby mizející zákoutí zdejší pedagogické historie? Určitě ne všude.

Dvouapůltý důvod uvalení vazby na počítače- žáci by si do volně přístupné učebny mohli přinést hry! Jaká svatokrádež pro bedlivě střeženou třináctou komnatu! Jak si to vlastně ten Komenský nebojaksejmenoval tím svým "Škola hrou" představoval?

Třetí důvod pro zamřížování počítačů - jde o elektrický spotřebič hrozící úrazem. Další alibi vlastní neschopnosti myslet a konat. Není problém všechny zásuvky znepřístupnit, takže žáci budou manipulovat jen vypínačem na televizoru a na počítači. Celý kabelový rozvod může být rovněž ukryt. Pak i kdyby přišel žák z pole s kosou na rameni, nemohlo by k úrazu proudem (nikoli kosou) dojít.

Ono je jednodušší vymýšlet neměnné důvody, pro které nelze žáky k počítačům pustit. Tak se i s celou informační revolucí dají pod zámek a basta. Zase o jednu učitelskou starost míň, vybyde i nějaká ta hodina volného času navíc, prostě paráda. Kdo by se staral o nějaké růžové perspektivy budoucnosti národa, když to ještě ke všemu jsou pobertové, před kterými je dobré mít se na pozoru...

Jenže - kdo tu obírá koho o co... Zkuste si na to, vy učitelé, odpovědět. A zkuste si odpovědět i na to, jak se k vám a ke svému okolí budou chovat (nejen) vaši žáci, které v televizi veřejně označíte za zloděje. A vůbec se zkuste tak nějak retrospektivně dobrat toho, odkud to všechno, co jste v besedě uvedli, pramení. Kde to vůbec začalo. A kam to vede. To, jak mládež vychováváte a učíte, je totiž věc nadmíru veřejná.

Situace ve vybavení škol počítači, programy a učebnicemi je jen další partie celého toho letitého Kocourkova. Samozřejmě to ztěžuje podmínky procesu výuky. Ale to vás ani v nejmenším nezbavuje odpovědnosti v ničem. Neukazujte jen kolem sebe, ale především na sebe. Jakákoli sebetěžší situace nemůže sloužit jako zdroj osobních alibistických výmluv, proč nebudu plnit své poslání, které jsem na sebe vzal. Didaktika není parabolou fňukání, ani synonymem výplatní pásky. Když už ale dojdete až sem, pak je snad opravdu lepší jít od toho.

-elzet-

# Z DOMOVA

* V Kabinetu elektroniky MV Svazarmu, Veleslavínská 42, Praha 6, zahájil v lednu svou činnost Klub elektroniky. Jeho činnost je výrazně zaměřena na oblast výpočetní techniky. Pořádají se zde kursy, ve kterých se zájemci mohou na počítačích PP 01 naučit základům elektroniky, informatiky a výpočetní techniky. Výsledkem kursu by měly být znalosti programování a výpočetní techniky přibližně na úrovni výuky středních škol. Kabinet se zaměřuje také na usnadnění přechodu adeptů výpočetní techniky na moderní počítače. A co je zvlášť zajímavé - chystá se na výuku operačního systému CP/M. Pro ten účel se připravuje unikátní učebna, kde bude v počítačové síti za řízení SAPI pracovat několik počítačů PP 01. Podrobnosti o tomto projektu se pokusíme přinést v některém z příštích čísel Mikrobáze. (Meca)

* OV Svazarmu v Praze 4 pořádá v Domě pohybové výchovy na Brumlovce kursy programování v Basicu. Jsou určeny dětem od 9 let i dospělým. Bližší informace na pražském telefonním čísle 42 45 32. (Meca)

* Jistě jste si všimli, že naprostá většina našich drobných zpráv je z Prahy, nebo se Prahy bezprostředně týká. Příčina je v tom, že jen velmi obtížně získáváme informace o mimopražském dění. Proto vyzýváme všechny, v jejichž okolí se něco počítačového děje - dejte nám o tom vědět! Stačí zaslat třeba jen korespondenční lístek na adresu naší redakce. Nezapomeňte uvést zpáteční adresu- pokud to bude něco zvlášť zajímavého, budeme v korespondenci pokračovat, nebo se za vámi přijedeme podívat. (Meca)

Mark Minasi se v měsíčníku BYTE 12/88 rozčiluje nad **nesolidností, až prolhaností computerových producentů.** Uvádí příklad nejmenované firmy, která ve svých propagačních materiálech uvádí, že vyrábí klon EtherLink Plus pro OS/2 za pouhých $300. M.Minasi se nadchl a firmě zavolal: "Je vaše deska kompatibilní s EtherLinkem Plus? Bude fungovat v LAN Manageru OS/2 s drivery EtherLinku Plus?" Fy: "Hned se na to podíváme... Ne, tyhle drivery nemůžete použít. Naše deska podporuje drivery většiny sítí, ale LAN Manageru ještě ne." MM: "Ale když vám to nefunguje s těmi drivery a navíc ani s vaší deskou pro LAN Manager nedodáváte vlastní driver, jak můžete tvrdit, že podporujete LAN Manager?" Fy: "Jaksi, řekl bych, že podporujeme LAN Manager asi v tom smyslu, jako podporujeme Nelsona Mandelu."

Do značné míry si s M.Minasim notuje **Phillippe Kahn, president firmy Borland:** "Jestli chceme dát počítačovou sílu lidem, kteří ji potřebují, musíme je zbavit panujícího zmatku mnohonásobných standardů v softwaru i hardwaru. Hrůza je, že to došlo až tak daleko, že když si něco koupíte, tak vám to vůbec nemusí fungovat. Jako by se dělalo ne proto, aby věci byly funkční, ale kvůli hvězdičkám v novinových recenzích." Veškerá podobnost s čímkoli, co by vám to mohlo připomínat, je samozřejmě čistě náhodná.

Řada firem se chystá zlepšit UNIX grafickým uživatelským interfacem. **Open Software Foundation** prohlásila, že během dvou let bude mít hotovou tzv. **otevřenou verzi UNIXu.**

# PŘEHLED POČÍTAČŮ TYPU PC

Pokračujeme přehledem počítačů kompatibilních s IBM PC a jeho klony, jak je nabízel britský trh v době loňských letních prázdnin. Tentokrát se věnujeme počítačům typu AT.

**Použité zkratky:**

**zLstg** - základní cena bez daně
**uP** - mikroprocesor
**Paměť** - standardní paměť
**Diskj** - vestavěné diskové jednotky
**+flop** - doplnitelné floppy jednotky
**+HD** - doplnitelný hard disk
**Her** - monitor Hercules a cena
**CGA, EGA** - typy monitorů a cena
**Mo** - monochromatický monitor
**P** - paralelní interface (Centronics)
**S** - sériový interface (RS232)
**E** - (expansion) konektory pro rozšíření
**tLstg** - typická prodejní cena:
v horní řádce cena základní konfigurace,
v dolní cena plné konfigurace
**+** - číslo za "+" je cena v Lstg;
když za "+" není číslo, je možno přikoupit výrobek jiných producentů (různé ceny)
***** - výrobek je součástí nabízené konfigurace

Počítače kompatibilní s IBM AT

| Název počítače | zLstg | uP | Paměť | Diskj | +flop | +HD | Her | CGA | EGA | P | S | E | tLstg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AES 286 AT | +2200 | 80286 | 1M 16M | 1x1,2M 20M | 1x360K 2x1,2M | 2x20M 51-85M | * | | Mo+ +345 | 1 | 2 | 28 616 | +2200 |
| ALCATEL XTRA/Prof.400 | +2715 | 80286 | 1M 16M | 1x1,44 20M | 1x1,44 1x360 | 40M 72M | * | +375 | +475 | 1 | 1 | | +2715 +3915 |
| APRICOT Xen I HD and XD | +2998 | 80286 | 1M 8M | 1x1,2M 20M | | 40M | * | | | 1 | 1 | | +2998 |
| APRICOT Xen-I 286 | +1599 | 80286 | 1M 2M | 2x1,44 | 1x1,2M | 30M 45M | * | | | 1 | 1 | | +1999 |
| AST Premium/286 | +1725 | 80286 | 512K 1M | 1x1,2M | | 20-70 | * | | Mo+ | 1 | 1 | | +1755 +2995 |
| CANON A200 EX | +4250 | 80286 | 640K 15,5M | 1x1,2M 40M | | 2x40 | | Mo+ | | 1 | 1 | | +4250 |

## Klony IBM AT

| Název počítače | zLstg | uP | Paměť | Diskj | +flop | +HD | Her | CGA | EGA | P | S | E | tLstg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMMODORE PC1 | +499 | 8088 | 512Kb | 1x360K | 2x360K 1x720K | | | Mo+ | | 1 | 1 | | |
| COMMODORE PC40 | +1599 | 80286 | 1M | 1x1,2M 20M | | 40M | * | +150 | | 1 | 1 | | +1599 |
| COMPAQ Deskpro 286 | +2209 | 80286 | 256K 640K | 1x1,2M | 1x360K 1x1,44 | 20M 40-130M | | Mo+ | +420 | 1 | 1 | 7 | |
| COMPUCORP Computext FS | +2700 | 80286 | 640K | 1x360K 1x20M | | 20M 40M | | * | | 1 | 2 | 5 | |
| DATA GENERAL Dasher/286 | +2990 | 80286 | 640K 4,8M | 1x1,2M 20M | 1x360K 1x720K | 2x20 | * | | | 1 | 2 | 18 516 | +2990 |
| DEC VAXmate | +3055 | 80286 | 1M 3M | 1x1,2M | | 20M | | | Mo+ | 1 | 1 | | +4415 |
| EPSON AX2 | +1495 | 80286 | 640K | 1x1,2M 20M | | | * | | Mo+ | 1 | 1 | 6 | +1495 |
| EPSON PC AX | +1995 | 80286 | 640K 15,5M | 1x1,2M 40M | 1x360K 1x720K | 80M | * | | | 1 | 1 | 9 | |
| ERICSSON | +2990 | 80286 | 512K | 1x1,2M 20M | 1x360K | 40M | | | Mo+ | 1 | 1 | 8 | |
| FERRANTI | +2525 | 80286 | 640K | 1x1,2M 20M | | 40M 76M | * | +159 | +488 | 1 | 1 | 6 | +2525 +4268 |
| FUTURE COMPUTERS FX 50 | +4395 | 80286 | 640K | 1x1,2M 40M | | | * | | | 1 | 2 | | +4395 |
| GOUPIL 65 286 | +2795 | 80286 | 1M | 1x20M | 1x720K 1x1,44 | 40M | * | | +500 | 1 | 1 | 4 | |
| HERMES PC200 | +3071 | 80286 | 1M 7M | 2x360K | 720K 1,44M | 20-68M | * | | | 1 | 1 | 7 | |
| HEWLETT PACKARD | +1756 | 80286 | 640K 1M | 1x1,2M 20M | 1x360K 1x1,44K | 40M | | | Mo+ | 1 | 1 | 7 | +1756 +2073 |
| HONEYWELL PC AP | +2795 | 80286 | 640K | 1x1,2M 20M | | | * | | | | | | |
| IBM AT Expanded | +4124 | 80286 | 512K 640K | 1x1,2M 30M | 1x360K | 2x30 | * | | | 1 | 1 | | |
| IBM PC XT286 | +2909 | 80286 | 640K 6,8M | 1x1,2M 20M | 1x360K 1x720K | | * | | | 1 | 1 | | |
| IBM PS/2 Model 50 | +3048 | 80286 | 1M 7M | 1x1,44 20M | 2x1,44 1x360K | | | | Mo+ +239 | 1 | 1 | 3MC | +3118 +3357 |
| IBM PS/2 Model 60 | +3693 | 80286 | 1M | 1x1,44 44M | | 70M | | | Mo+ +239 | 1 | 1 | 7MC | +3763 +4002 |
| KAYPRO 2861 | +1080 | 80286 | 640K 15M | 1x1,2M 20M | | 40M | * | | Mo+ | 1 | 1 | 8 | +2586 |
| MITSUBISHI MP286 | +1999 | 80286 | 640K 7,5M | 1x1,2M 20M | 1x360K 1x1,2M | 40M | * | | Mo+ | 1 | 1 | 6 | +1600 +2499 |
| NCR PC8 | +2760 | 80286 | 256K 16M | 1x1,2M | | 20M 30M | | Mo+ +305 | | 1 | 1 | | |
| NCR PC810 | +2860 | 80286 | 640K 16M | 1x1,2M 20M | 1x360K 1x1,44 | 70M | | Mo+ +290 | +485 | 1 | 1 | 8 | +2860 |
| NEC Powermate | +2695 | 80286 | 640K | 1x1,2M 20M | 1x720K | 40M 68M | | | * | 1 | 1 | 18 516 | +2695 +2995 |
| OLIVETTI M280 | +3090 | 80286 | 1M | 1x1,2M 20M | 1x360K 1x720K | 40M | | Mo+ +296 | +589 | 1 | 1 | 38 416 | +3071 +4160 |
| PANASONIC FX-800 | +3299 | 80286 | 512K 15M | 1x1,2M 40M | 1x360K | | | Mo+ | | 1 | | | |
| RESEARCH MACHINES RM Nimbus AX-286/2 | +2395 | 80286 | 1M 16M | 1x1,44 | 2x1,44 1n2M | 40M 20-115M | * | | | 1 | 1 | | |
| SAMSUNG MFC-6000 | +1399 | 80286 | 1M | 1x1,2M | | 20M | * | | | 1 | 1 | | +1699 |
| SANYO MBC 17 | +945 | 80286 | 512K 1M | 1x1,2M | | 20M 65M | | +200 +360 | | 1 | 1 | 5 | +1675 |
| SANYO MBC 990 HD20 | +2399 | 80286 | 512K 2M | 1x1,2M 20M | | 40M | * | | | 1 | 1 | | +2399 |
| SHARP 7500AT | +1995 | 80286 | 512K 1M | 1x1,2M 20M | | 40M | | Mo+ +200 | | 1 | 1 | 6 | |
| TANDON PAC 286 | +1595 | 80286 | 1M | | 1x1,2M | 30M 2x30M | * | | +395 | 1 | 1 | | +1595 +1900 |
| TANDON PCA | +1495 | 80286 | 1M | 1x1,2M 20M | | 40M 70M | * | | +395 | 1 | 1 | | +1495 +2695 |
| TANDON Target | +1695 | 80286 | 1M | 1x1,2M 20M | | 40M | * | | +395 | 1 | 1 | | +1695 +2295 |
| TANDY 1000TX | +895 | 80286 | 640K 768K | 1x720K | 2x720K 1x360K | 20M | | Mo+ +200 | | 1 | 1 | 58 | +1394 |
| TANDY 3000HL | +995 | 80286 | 512K 640K | 1x360K | 1x1,2M | 20M 40M | * | | | 1 | | 48 316 | +1534 |
| TATUNG | +1993 | 80286 | 640K | 1x1,2M | | 20M 40M | * | | | | | | |
| TELEVIDEO TeleCAT-286 | +2195 | 80286 | 512K 1M | 1x1,2M 20M | 1x360K | 40M 70M | | Mo+ | | 1 | 1 | 5 | |
| TEXAS INSTRUMENTS Business-PRO | +4000 | 80286 | 512K 15M | 1x1,2M | 1x360K | 20-120M | * | | | 1 | 1 | | |
| THORN EMI MPC | +1395 | 80286 | 512K 1M | | 1x360K 1x1,2M | 20M 40M | | * | * | 1 | 1 | 4 | +1395 |
| TULIP AT Compact | +1650 | 80286 | 640K | 2x1,2M | | 20M 40M | * | | | 1 | 1 | | +2045 |
| UNISYS PC-IT | +3640 | 80286 | 512K 5M | 1,2M 40M | 1x360K 1x720K | 69-117M | * | | | 1 | 2 | 68/ 16 | |
| UNISYS PC/microIT | +2995 | 80286 | 512K 3,5M | 1x1,2M 20M | 1x360K | | * | | +783 | 1 | 1 | | |
| UNISYS PW 300/500 | +2435 | 80286 | 640K 1,5M | 1,2M 20M | 1x360K 1x720K | 40M | * | | +610 | 1 | 1 | 28 | |
| VICTOR V286 | +2499 | 80286 | 512K 1M | 1x1,2M 20M | 1x360K 1x720K | 40M 70M | * | +300 | +500 | 1 | 1 | | +2549 +3549 |
| VICTOR VPCIII | +2499 | 80286 | 640K 1M | 1x1,2M 30M | 1x720K | 60M | * | +300 | +500 | 1 | 1 | | +2499 +3499 |
| WANG APC | +2535 | 80286 | 512K 2M | 1x1,2M | 1x1,2M 1x360K | 20M 30M | | | | 1 | 1 | | |
| WANG PC 240 | +1455 | 80286 | 640K 6,5M | 1x1,2M | 1x720K | 20M 42M | + 395 | | +395 +960 | 1 | 1 | 18 316 | +2275 +3365 |
| WANG PC 280 | +2630 | 80286 | 640K 10,5M | 1x1,2M | 1x360K 1x720K | 20M 32M | + 200 | | +585 | 1 | 1 | 28 616 | |
| WYSE PC 286 | +1815 | 80286 | 640K 15M | 1x1,2M | 2x1,2M | 20M 40M | * | +180 | +570 | 1 | 1 | 28 516 | |
| ZENITH Z286 | +1695 | 80286 | 512K | 2x360K | | 20M | * | * | * | 1 | 1 | 4 | +1975 +2425 |
| ZENITH Z248 | +2395 | 80286 | 512K 15M | 1x1,2M 20M | 1x360K | 40M | * | | | 1 | 1 | | |

| Název počítače | zLstg | uP | Paměť | Diskj | +flop | +HD | Her | CGA | EGA | P | S | E | tLstg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ACER PC900 | +1725 | 80286 | 512K | 1x1,2M | 2x1,2M | 32M | * | | +400 | 2 | 1 | | +1985 +2685 |
| ACER PC910 | +1780 | 80286 | 640K | 1x1,2M | 2x1,2M | 40M 80M | * | | +400 | 2 | 1 | | +2340 +2740 |
| AESIR Odin | +1999 | 80286 | 1M | 1x1,2M 30M | 1x1,44 | 40M | * | +350 | +595 | 1 | 1 | | +2030 +3224 |
| AJWAD Baby AT | +799 | 80286 | 640K 1M | 1x360K | 1x1,2M | 20M 40M | | | Mo+ | 1 | 1 | | |
| AMT 286/10 | +1695 | 80286 | 512K 640K | 1x1,2M | | | * | | | * | * | 6 | +1695 +2045 |
| AMT 286-20 | +2145 | 80286 | 1M 4M | 1x1,2M 40M | | | * | | +350 | 1 | 1 | 6 | +2145 +2495 |
| ARC 286 Turbo | +1599 | 80286 | 1M | 1x1,2M 20M | | 30M | * | | | 1 | 2 | | +1599 |
| ARC 286 Turbo 12 | +1899 | 80286 | 1M | 1x1,2M 20M | | 40M | * | | | 1 | 2 | | +1899 +2899 |
| BARBATEN AT | +1495 | 80286 | 640K 1M | 1x1,2M 20M | 1x360K | | * | +180 | | 1 | 1 | 8 | +1495 |
| BETOS PC 286 Turbo | +1249 | 80286 | 1M | 1x1,2M 1x30M | 1,2M 1,44M | 40M 90M | * | +195 | +399 | 1 | 1 | 28 616 | +1249 +2047 |
| BONDWELL BW39 | +1399 | 80286 | 1M | 1x1,2M 20M | | | * | +229 | | 1 | 1 | 5 | +1399 |
| BONDWELL BW63 | +1349 | 80286 | 640K | 1x1,2M 20M | | 35M | * | +299 | | 1 | 1 | 8 | +1349 |
| CAS 286DSC Turbo AT | +950 | 80286 | 1M | 1x1,2M 20M | | 40M 80M | * | | +400 | 1 | 1 | | +1015 +1525 |
| CDATA PC 501 AT | +1895 | 80286 | 512K 1M | 1x1,2M 20M | | 80M | | Mo+ | | 1 | 1 | | +1995 |
| CYBERCOM Baby-AT 286 Turbo | +2095 | 80286 | 1M | 1x1,2M 1x360K,20M | | 30M 40M | | * | | 1 | 1 | | +2095 |
| CYBERCOM PC/AT | +2095 | 80286 | 1M | 1x360K 1x1,2M | | 30M 40M | | * | | 1 | 1 | | +2095 |
| DELL System 200 | +1299 | 80286 | 640K 4,8M | 1x1,2M 20M | 1,2M 1,44M | 40M 70M | * | | +400 | 1 | 2 | 6 | +1299 +3199 |
| DIAMOND 286 | +1899 | 80286 | | 1x1,2M | | 40M 83M | * | +185 | +395 | 2 | 2 | 8 | +1899 +2804 |
| DIGITASK Axiom AT | +1099 | 80286 | 1M | 1x1,2M 1x1,44 | 2x1,2M | 20M 40M | * | +245 | +450 | 2 | 1 | 28 616 | +1344 +2049 |
| ELONEX PC 286 | +1295 | 80286 | 1M | 1x1,2M 40M | | 44M 72M | * | +180 | +380 | 1 | 2 | 8 | +1295 |
| EUROMICRO AT/286-Turbo | +1995 | 80286 | 1M | 1x1,2M 20M | 1x360K | 33M 70M | * | +240 | +645 | 1 | | | +1995 +3340 |
| EUROMICRO Dart 12 | +2795 | 80286 | 1,5M 3M | 1x1,2M 20M | 1x360K | 30M 80M | * | +240 | +645 | 1 | 2 | | +2795 +4140 |
| INTER ORIENT 2408 AT Turbo | +699 | 80286 | 512K 1M | 1x1,2M | 2x1,2M | 20M -40M | * | +200 | +300 | 2 | 1 | 8 | +699 +995 |
| ISIS 400 | +1125 | 80286 | 1M | 1x1,2M | 2x1,2M 1x720K | 20M 40M | * | | | 2 | 1 | 8 | +1430 |
| MATRIX AT-II | +1293 | 80286 | 512K 1M | 1x1,2M 20M | | 30M 40M | * | +219 | +395 | 1 | 1 | 8 | +1348 |
| MATRIX AT-III | +1681 | 80286 | 512K 1M | 1x1,2M | 720K | 40M | * | | +323 | 1 | 1 | 8 | +2152 |
| META-DYNE AT80/100 | +949 | 80286 | 1M | 1x1,2M 20M | 1x360K 1x720K | 20M 40M | * | +165 | +524 | 2 | 1 | 28 616 | +1038 |
| MICRONIX 286 Marvel | +1299 | 80286 | 640K 1M | 1x1,2M 20M | 1x360K | 40M | * | | | 1 | 1 | | +1299 |
| MISSION 286 Desktop | +1290 | 80286 | 512K 1M | 1x1,2M | | 30M 40M | * | | | 1 | 1 | 6 | +1290 +2190 |
| MISSION 286 Plus | +1290 | 80286 | 512K 2M | 1x1,2M | | 40DM 80M | * | | | 1 | 1 | 8 | +1290 |
| MITAC Paragon 286 | +1579 | 80286 | 640K 1M | 1,2M 40M | | | * | +120 | +270 | 1 | 1 | 28 216 | +1659 +1929 |
| MONROE 3-286 | +1875 | 80286 | 512K | 1x1,2M 20M | | 100M | * | | | 1 | 1 | 8 | |
| NESS PC-286S | +995 | 80286 | 512K 1M | 1x1,2M 20M | | 40M 80M | * | | | 1 | 1 | 8 | |
| NTS 286 | +1199 | 80286 | 1M 7M | 1x1,2M 20M | 1x720K | 40M 80M | * | +200 | | 1 | 2 | | +1199 |
| NTS Heat 286 | +1499 | 80286 | 1M 8M | 1x1,2M 20M | 1x360K 1x1,44 | 80M 160M | * | | | 1 | 2 | 8 | +1499 +2799 |
| OPUS PC V | +1295 | 80286 | 1M | 1x1,2M 30M | | | * | | +300 | 1 | 1 | 38 516 | +1345 |
| PACIFIC 286 | +849 | 80286 | 640K | 1x1,2M | | 20M 40-71M | * | | +389 | 1 | 1 | | |
| PACKARD BELL VT286 | +1360 | 80286 | 640K | 1x1,2M 20M | | 40M 44M | * | | +350 | 1 | 1 | | +1360 +2125 |
| PCC AT | +1459 | 80286 | 640K | 1x1,2M 20M | 2x1,2M | | | | * | 2 | 2 | 38 416 | +1524 |
| PEACOCK PC/AT | +1600 | 80286 | 512K | 1x1,2M 20M | | 30-40M | * | | | 1 | 2 | | |
| Q-DATA Turbo AT | +1741 | 80286 | 1M | 1x1,2M 20M | | | * | +356 | +497 | 1 | 2 | 8 | +1811 |
| QUBIE ATurbo | +1195 | 80286 | 640K | 1x360K 20M | 1x1,2M 720K | 42M | * | | +325 | 2 | 1 | 8 | +1270 +1770 |
| SBC 286AT | +999 | 80286 | 1M | 1x1,2M | 2x1,2M 1x360K | 20M 40M | * | +200 | +400 | * | * | 8 | +1399 +1999 |
| SCICON AT2 | +1264 | 80286 | 1M | 1x1,2M 20M | | 40M | +16 | Mo+ +126 | +510 | 1 | 1 | 28 616 | +1335 +2059 |
| SCREENS CL AT | +1095 | 80286 | 1M | 1x1,2M 20M | | 40M | * | +199 | +399 | 2 | 2 | 28 616 | +1095 +1694 |
| SCREENS ST 320 Baby | +1299 | 80286 | 1M | 1x1,2M 1x30M | | 40M | * | +199 | +399 | 2 | 1 | 5 | +1299 +1898 |
| SYSTEM ZONE Atel | +850 | 80286 | 640K 2M | 1x1,2M | 1x720K 1x360K | 20M 40M | * | | Mono +575 | 1 | 1 | 8 | +850 +1665 |
| VIGLEN VPC AT 286M | +1399 | 80286 | 1M | 1x1,2M 20M | | 40M | * | | | 1 | 2 | 28 616 | +1399 |
| WALTERS AT | +960 | 80286 | 640K | 1x1,2M | 1x360K 1x720K | 20M 40M | * | +200 | +450 | 1 | +26 | 28 616 | +1020 +1815 |
| WALTERS Baby AT | +910 | 80286 | 640K | 1x1,2M | 1x360K 1x720K | 20M 40M | * | +200 | +450 | 1 | +26 | 28 616 | +1355 +2150 |
| WELLS AMERICAN A*Star II PC AT | +2780 | 80286 | 1M | 1x1,2M 20M | | 30M 80M | * | | +570 | 2 | 2 | | +2780 |


# PROGRAMOVÁ NABÍDKA


## POKYNY K OBJEDNÁVÁNÍ PROGRAMŮ

**Nabízené programy si zájemci objednávají výhradně na korespondenčních lístcích adresovaných na 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6. Programy zasíláme na dobírku, je ale možný přímý nákup ve středisku VTI v Martinské 5, Praha 1.**

## PROGRAMY ZÁKLADNÍ NABÍDKY PRO ZX SPECTRUM

**CP/M** **191 Kčs**
Vstupenka do světa profesionálních osmibitových počítačů; možnost využívání množství programů, které jsou tímto systémem řízeny.

**ASSEMBLER 80** **198 Kčs**
Původní program, výkonný pomocník při programování ve strojovém kódu.

**PROFESOR** **170 Kčs**
Univerzální výukový program, základ pro instalaci dodávaných znalostních bází typu STUDENT.

**TESTEDITOR** **468 Kčs**
Program pro vlastní tvorbu znalostních bází typu STUDENT.

**TEMPERAMENT** **99 Kčs**
Zábavný psychologický test k určení typu temperamentu, kdy se můžete dozvědět, "kdo" vlastně jste.

## OD 1.6.1989 NABÍZÍME: NOVINKA!!

**ODA** **160 Kčs**
Osobní databázový systém s jednoduchým názorným ovládáním, volbou formátu zobrazení a tisku.

**PROGRAF** **139 Kčs**
Prostorové grafy pro zobrazování prostorových funkcí, s viditelností, volbou natočení a řezů.

**MULTITASKING** **149 Kčs**
Operační systém umožňující současný běh i více programů na jednom počítači.

**ZEMĚPIS** **147 Kčs**
Encyklopedický výukový program zeměpisu ČSSR s poskytováním řady informací a možností zkoušení.

**REMBRANDT** **170 Kčs**
Grafický program pro kreslení libovolných barevných obrazů s novými funkcemi.

**PLAYTAPE** **130 Kčs**
Program pro kontrolu nastavení magnetofonu, zaznamenaných dat, k měření kmitočtu a komprese obrazovky.

## PROGRAMY ZÁKLADNÍ NABÍDKY PRO POČÍTAČ SHARP MZ

**KANTOR I** **485 Kčs**
Výukový program pro studium deskriptivní geometrie - prvních 5 lekcí.

**KANTOR Ia** **291 Kčs**
Tři procvičovací programy k 1.dílu souboru.

## OD 1.6.1989 NABÍZÍME: NOVINKA!!

**ZX MONITOR** **190 Kčs**
Program pro práci ve strojovém kódu mikroprocesoru Z 80 s příkazy pro přenos programů z počítače ZX Spectrum na počítač SHARP MZ-800.

**KANTOR II** **588 Kčs**
Výukový program pro studium deskriptivní geometrie - lekce 6 až 11.

## PROGRAMY ZÁKLADNÍ NABÍDKY PRO POČÍTAČE PMD 85 (od 1.6.1989)

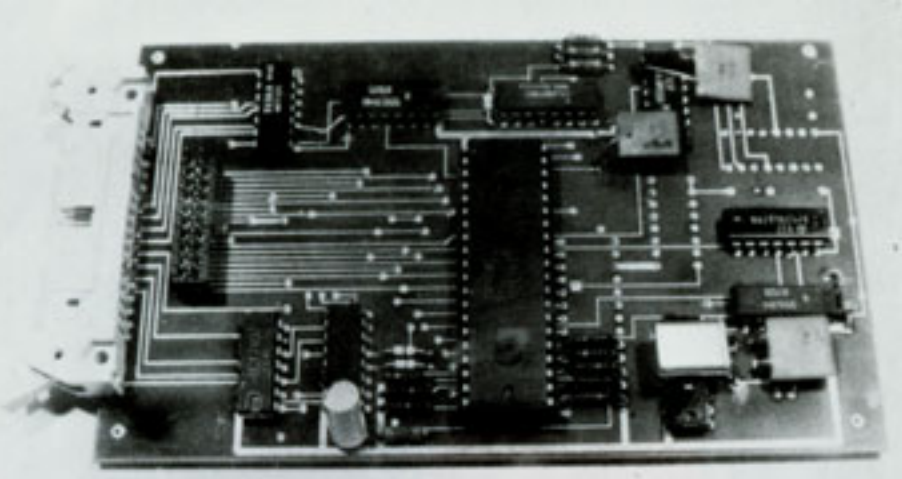
**DAM 0000 V2.0** **cca 160 Kčs**
Assembler pro PMD 85-1 s řádkově orientovaným editorem, překladačem, debuggerem a disassemblerem.

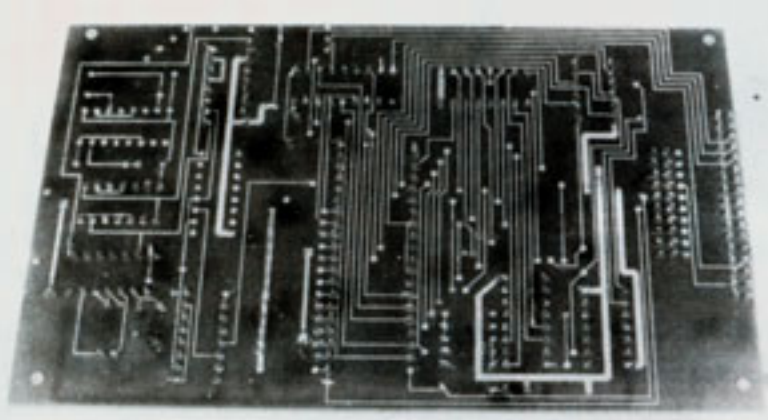
**DAM+2 V 1989** **cca 155 Kčs**
Assembler pro PMD 85-2 s celostránkovým editorem, překladačem, debuggerem, s krokováním programu a s disassemblerem.

**KASWORD V3.0** **cca 170 Kčs**
Textový editor s blokovými operacemi, formátováním řádek i odstavců a s diakritickými znaménky.

**KAREL V2.2** **cca 190 Kčs**
Výukový jazyk pro PMD 85, který umožňuje paralelní programování, užívání klíčových kláves. Obsahuje editor, slovník i rozklady slov.

Nabídky z minulých čísel zůstávají v platnosti





## POSTAVTE SI S NÁMI ŘADIČ FLOPPYDISKU *(pokračování ze 2. strany obálky)*

Obrázky nahoře ukazují jak dopadla amatérská stavba desky řadiče podle Betadisku. Na obrázcích dole je vidět, že i desku s DOSem pro ZX Spectrum lze postavit. Nebýt chyby v obrazci plošného spoje, mohlo vše fungovat na první zapojení. Jen se ukázalo být vhodné pro zvýšení stability doplnit keramický kondenzátor, blokující napájení. To je velmi povzbudivý výsledek. Protože vím, že podobnou zkušenost už přede mnou udělalo několik jiných, mohu toto zapojení vřele doporučit k amatérské stavbě. O své zkušenosti se stavbou, ale i s několikaletým používání profesionálního výrobku, se s vámi rozdělím v obsáhlejším článku. Jeho úvodní část bude již v příštím čísle Mikrobáze. Protože však bude jistě největším problémem shánění součástek, uvádím rozpisku součástek pro obě desky již teď. K některým IO se dovážejí sovětské ekvivalenty. WD 1793 a EPROM se dají občas získat na burze, případně z inzerátu. Místo klasického křemenného krystalu je možno použít též keramického rezonátoru, který bývá často používán pro jednočipové mikropočítače.

Daniel Meca

Deska řadiče

IO 1 – 7417 (07)
IO 2 – 74LS139
IO 3 – 7405
IO 4 – 74LS174
IO 5 – WD 1793 (SAB 1793)
IO 6 – 74LS367
IO 7 – 74LS161
IO 8 – 74LS04
IO 9 – 74LS74
IO 10 – 74LS161

D1 až D3 – KAS22

C1 – 10M/15V
C2 a C3 – M1 keram.

R1 až R6 – 5k6
R7, R8, R10 až R12 – 1k
R9 – 220
R13 až R15 – 10k

X1 – krystal 4MHz

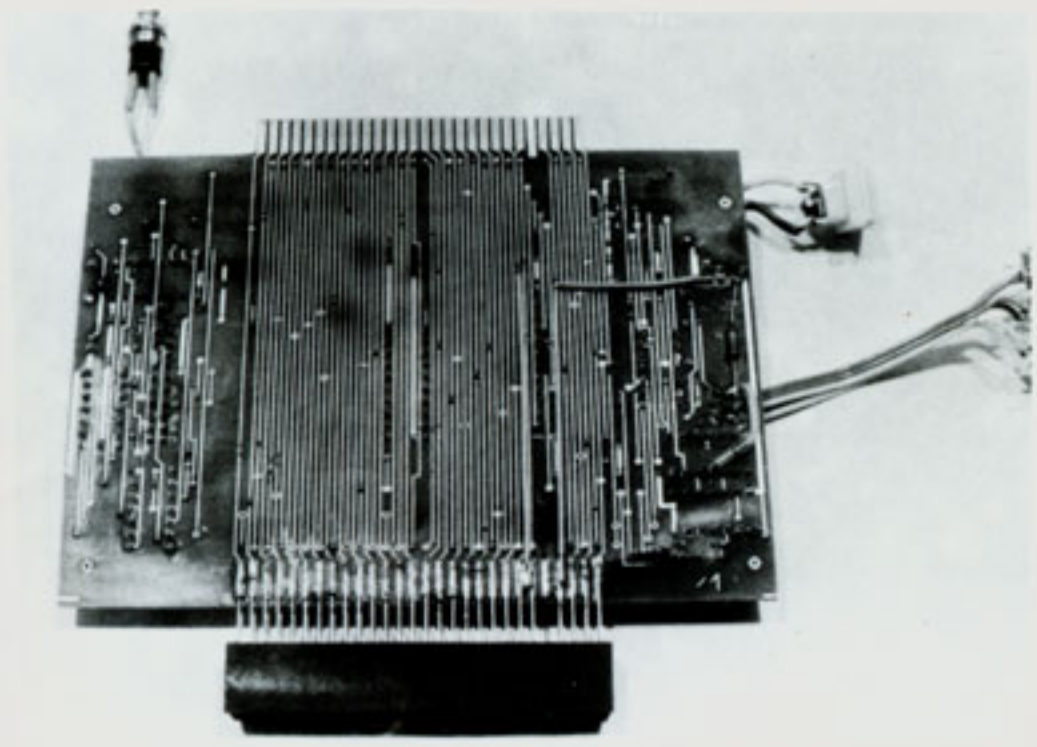
Deska DOSu

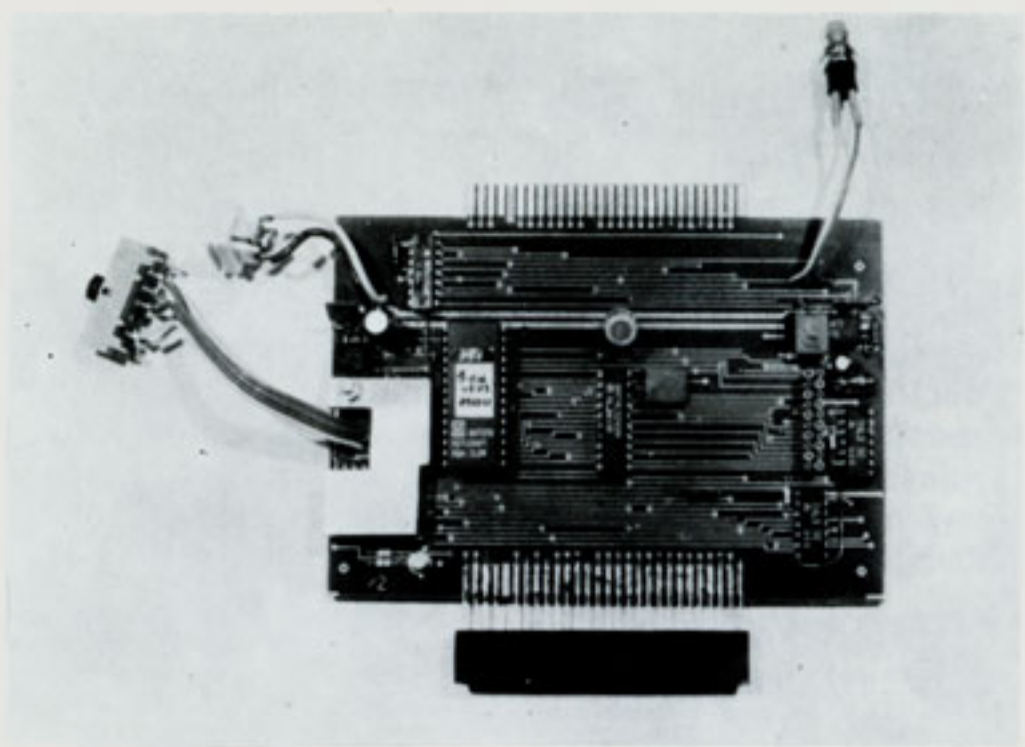
IO 1 – 74LS04
IO 2 – 74LS30
IO 3 – 74LS32
IO 4 – 74LS32
IO 5 – 74LS74
IO 6 – 74LS32
IO 7 – 74LS245
IO 8 – 27128 – 25 (EPROM 16KB)
IO 9 – 74LS123
IO 10 – 7805 (v plastu)

D1 až D8 – KAS22

C1 – 50M/15V
C2, C4, C5 – M1 keram.
C3 – 100M/10V
C6 – 10M/15V

R1 – 1K
R2 – M12
R3 – 470
R4 a R5 – 4K7
R6 – 5K6
R7 – 1K2
R8 – 820





Začátkem roku 1987 přinesla Mikrobáze první informaci o rozšíření RAM v ZX Spectru na zcela novém principu. Jednalo se o stránkování spodní části adresního prostoru. Autorem tohoto (na svoji dobu převratného) řešení je Jiří Lamač, pracující pod značkou Lecsoft. Z programů, které pro takto upravený počítač napsal, připomínáme jednak ty zcela první – MONS80 a LC+, jednak ten úplně poslední – implementaci operačního systému CP/M. Dnes již existují verze pro 272/528KB paměti, které byly popsány v ARA 9/88. Protože však od odevzdání rukopisu do AR vznikla řada dalších zajímavých variant, přineseme jejich popis v některém z příštích čísel Mikrobáze.


MONS80 – upravený disassembler MONS3 byl prvním programem pro nově rozšířené ZX Spectrum. Uživatel měl k dispozici volný prostor od 2000H do FFFFH.


LC+ – luxusní kopírovací program s volnou pamětí 64KB a průběžnou indikací zbývajícího prostoru v RAM. Umožňuje získání různých informací o kopírovaných programech. Ty je pak možno prohlížet a modifikovat. Praktická je možnost kontroly, případně opravy paritního bitu. Mezi Spectristy je rozšířená též verze, která byla upravena pro zapojení podle Sinsoftu (schema je uvnitř tohoto čísla).



CP/M v 2.2 – vrchol všeho snažení. Tento program umožňuje Spectristům přístup k tolika kvalitním programům, že už vlastně ani není třeba psát nové. Slogan „Vstupenka do světa profesionálních osmibitových počítačů", používaný v programové nabídce, není nijak přehnaný. Ze Spectra se skutečně stává profesionálně použitelný počítač.


Na titulní straně obálky je úvodní obrázek programu dodávaného 602. ZO Svazarmu.