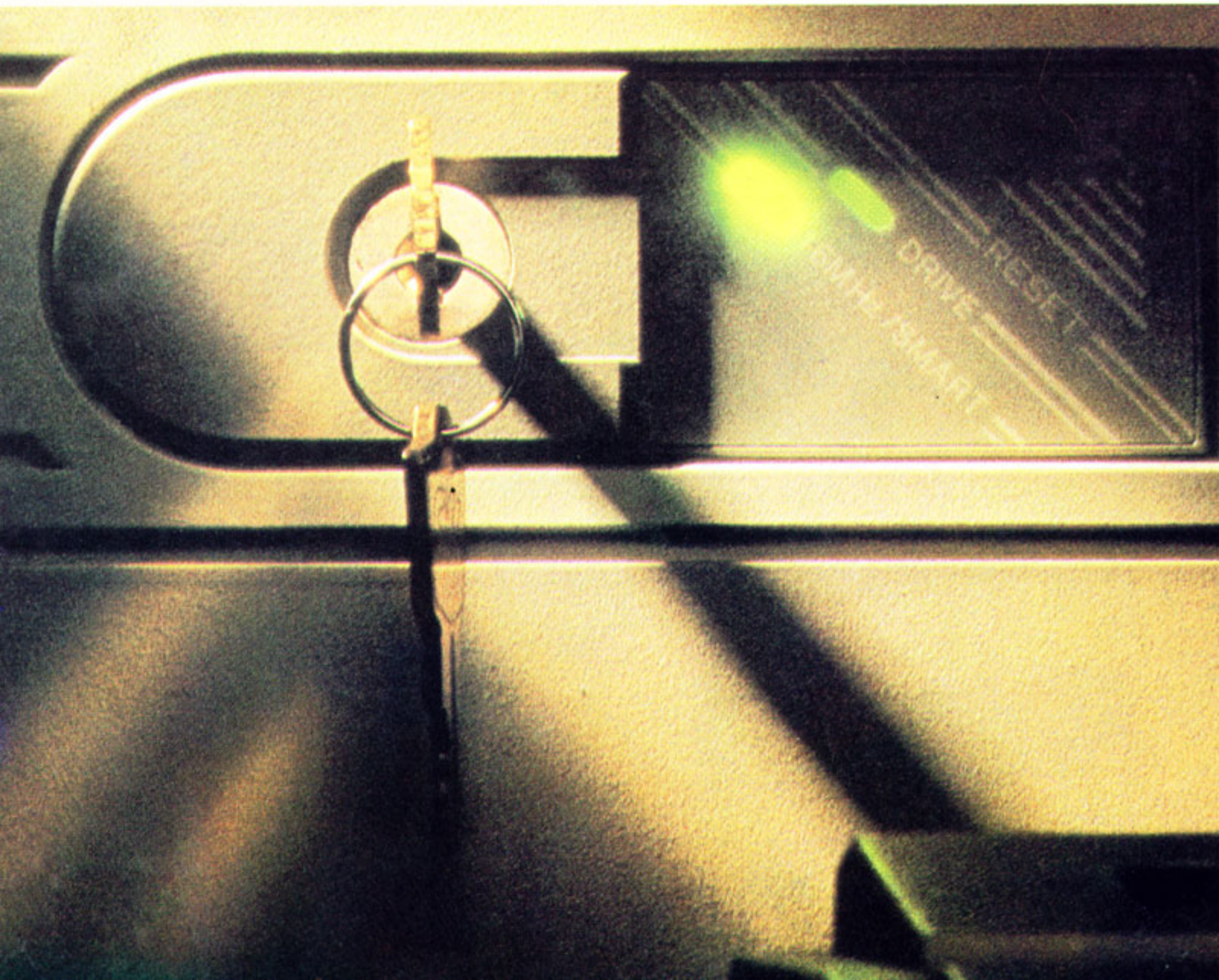
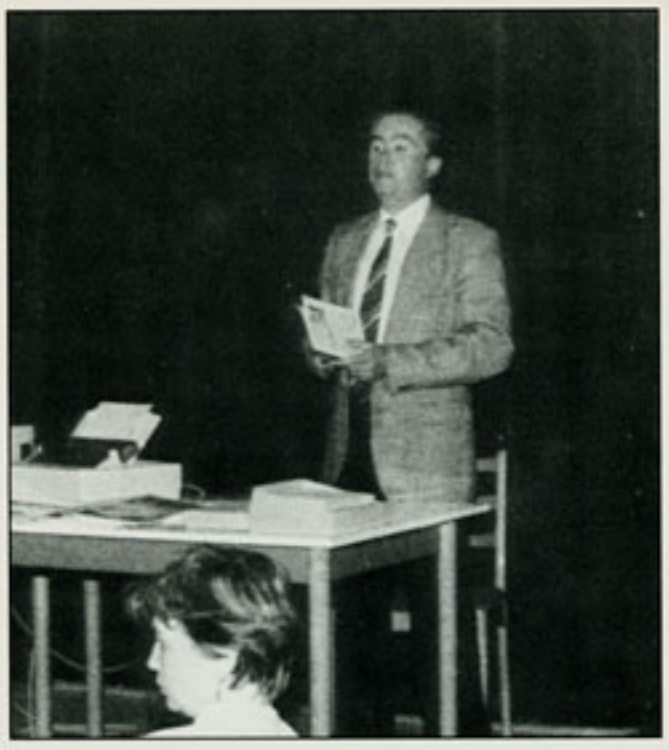
MIKRO BÁZE

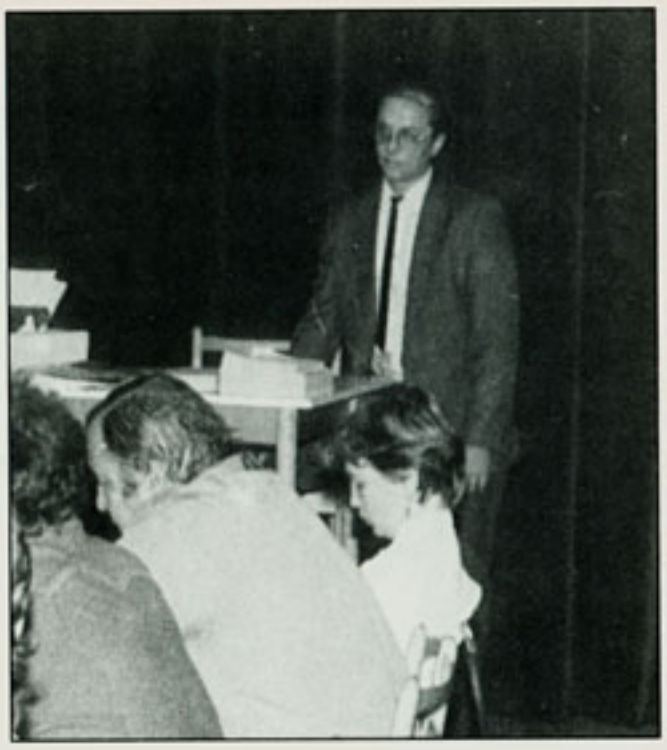
1989 6

technický
zpravodaj
svazarmu
pro zájemce o
mikropočítače

Cena 12 Kčs







TEXTOVÝ EDITOR

TEXT 602

SE PŘEDSTAVUJE

Lépe řečeno byl představen novinářům 22.6.1989. Malý sál paláce Kotva, kde se předvádění konalo, byl téměř zaplněn. Byli zde přítomni zástupci redakcí celé řady časopisů a novin. Početně snad nejvíc byla zastoupena redakce 100+1 ZZ. Úvodem krátce promluvil tajemník 602. ZO, Josef Kroupa (nahoře uprostřed) o rozvoji činnosti této ZO v oboru malé výpočetní techniky. P. Dien (vlevo) informoval o dílně dBase a O. Luňák (vpravo) pohovořil o činnosti střediska vědeckotechnických informací. Pak už se ujal slova ing. Richard Kaucký, vedoucí střediska programového vybavení pro šestnáctibitové počítače a jeden z autorů nového editoru v jedné osobě (obr. dole).

(pokračování na 3. straně obálky)

MIKRO BÁZE

1989/6

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 vedoucí redaktor Daniel Meca. Obálka ak. grafik Jiří Blažek, grafická úprava Lenka Chromková. Sekretářka redakce Zdeňka Válková. Redakční rada: Petr Horský, ing. Jan Klabal, ing. Petr Kratochvíl, Josef Kroupa, Rudolf Mach, Daniel Meca, ing. Alois Myslík, ing. Josef Truxa. Za původnodt 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

Hodně štěstí, Láďo!

Jak si mohl každý všimnout, na stránkách Mikrobáze se už od jejího vzniku objevovala pod řadou článků šifra -elzet-. Že se pod ní skrývá jméno Ladislav Zajíček, to časem asi pochopili i ti méně vnímaví. Ne, nelekejte se, není to nekrolog. Láďa je živý a zdravý. Jenom po těch letech opustil naši redakci. Tak, a je to venku. Já jsem zastáncem toho, aby se špatné zprávy říkaly hned a bez příkras.

Ladislav Zajíček stál už u zrodu našeho časopisu a s malou přestávkou ho dělal až doposud. Jestli dobře, nebo špatně - to mohou posoudit spíš čtenáři. Protože já sám jsem byl ještě vloni jen čtenářem Mikrobáze, mohu zde říci, že asi spíš dobře. Ne že bych byl vždy a nekriticky nadšen obsahem i formou, ale vždy jsem se na nové číslo těšil. Později, už jako redaktor Mikrobáze, jsem měl možnost poznat na vlastní kůži, co to je dělat takový časopis. A tak jsem Láďovi dodatečně ještě ledacos odpustil. Ono totiž dělat odborný časopis, který by zaujal, a co víc, dělat ho prakticky sám, to je velké sousto, kterým by se možná zadusil nejeden profesionální redaktor. A vidíte, Láďa to dokázal. Přitom vlastně měl u nás jen část úvazku. Jiný časopis, podobného rozsahu, má obvykle tak asi šest redaktorů na plný úvazek.

Bylo by pokrytectví tvrdit, že nám to spolu v redakci vždy ideálně klapalo, tomu by asi stejně nikdo nevěřil. Když se něco dělá s plným zápalem, člověk se nevyhne nějaké té rozepři, ale jinak jsme si myslím docela rozuměli. Proto mě zpráva, že kolega Zajíček naši redakci opouští, docela zaskočila. Prý má něco lepšího. Mezi lidmi se cosi proslýchá, že snad je to časopis Elektronika. Pokud je tomu opravdu tak, alespoň vím, kde hledat ty typicky "zajíčkovské" články, které mi někdy mluvily z duše a jindy mě zas popouzely. Nevím jak komu, ale mě budou v Mikrobázi scházet.

U příležitosti převzetí štafety mi nezbývá, než jménem čtenářů Mikrobáze, i jménem všech spolupracovníků, poděkovat Ladislavu Zajíčkovi za vše, co pro náš časopis udělal a popřát mu mnoho úspěchů v novém působišti.

Tak tedy - Hodně štěstí, Láďo!

Daniel Meca

Výzva čtenářům

Protože v obsazení naší redakce zůstala po odchodu Ladislava Zajíčka citelná mezera, vyvstává ještě naléhavěji potřeba nových spolupracovníků. Je to příležitost pro ty, kteří mají zájem o obor a kteří přitom nejsou na štíru s češtinou. Práce redaktora je zajímavá. Znamená to chodit mezi lidi, zajímat se o novinky a zajímavosti z oboru, být všude tam, kde se něco zajímavého děje. Přitom není nutné (a snad ani žádoucí) o všem sám psát, jde spíš o získání příspěvků a jejich zpracování.

Zájemci o tuto práci, která může být vykonávána jako vedlejší pracovní činnost, spojte se nezávazně s vedoucím střediska Mikrobáze. Adresa je:

Daniel Meca
Jihlavská 76
140 00 Praha 4

A ještě něco. Kdo nechce, nebo nemůže pracovat jako redaktor, a přitom má zájem nám pomoci, může se stát i jakýmsi krajským zpravodajem. Stačí nám, když příspěvky dodá v nezpracované formě. Máme zájem informovat nejen o tom, co zajímavého vzniklo v Praze, ale ve všech krajích republiky.

hovory o programování

ŘEČ O TEXTU

O dnešní rozhovor jsem požádal ing. Richarua Kauckého, vedoucího střediska programů pro šestnáctibitové počítače 602. ZO Svazarmu Praha 6.

Software - to slovo se v poslední době skloňuje ve všech pádech. Nástup domácích počítačů a zejména profesionálních osobních počítačů třídy PC způsobil nejen revoluci v našem životě, ale změnil se i způsob výroby programových prostředků. Tato oblast se stala doménou velkých zahraničních softwarových firem. Je vůbec možné konkurovat těmto firmám a vytvořit u nás moderní původní program?

Původní české programové vybavení je zatím řídkým jevem na našem, zatím se tvořícím, trhu programového vybavení. Dosavadní snahy soukromých uživatelů domácích počítačů se často omezují jen na shánění co největšího počtu zahraničních programů. Nic nevadí, že k nim nevlastní příslušný manuál a ve většině případů neovládá ani příslušný jazyk. Bezuzdné kopírování zahraničních programů je však nejen nemorální, ale i nezákoné. Ve své vlastní podstatě znamená bezprostřední ohrožení domácích programátorů. Ten, kdo bez skrupulí zcizí zahraniční program, aniž je schopen jej efektivně využít, se jistě nezastaví ani před zcizením programu domácí provenience. To pak bude mít za následek, že zisk z prodeje programů nepokryje ani výrobní náklady a trh programového vybavení bude postupně paralyzován. Tím se domácí programátoři budou omezovat na nenáročné úpravy zahraničních programů (tvorba takzvaných mutací) a nevytvoří se potřebné kádry pro tvorbu původních rozsáhlejších programových systémů. Nebezpečné názory typu "nic lepšího než <jméno programu> již nelze vytvořit, doděláme tam češtinu a to stačí..." jsou pouze dokladem omezenosti. Naši programátoři nejsou o nic horší, než jejich kolegové na západě. Mají horší podmínky pro svoji práci a chybí jim manažéři pro rozsáhlejší projekty. Proces tvorby programového produktu je totiž náročný nejen na čas, finanční prostředky, ale i na vzájemnou kordinaci zúčastněných odborníků v souladu s požadovanými výstupy projektu. Navíc musí existovat trh programového vybavení, který je schopen zajistit jistý zisk i pro organizaci finančně kryjící celý projekt. Časy, kdy oficiální organizace investovaly do programového vybavení pouze náklady na překlad cizojazyčného manuálu a změnou názvu zcizených programů spolu s přemrštěnou cenou, chtěly docílit světovosti, jsou snad již za námi. Naštěstí zde dochází k tlaku, kterým jsme nuceni respektovat mezinárodní právo, jinak se vystavujeme možnosti žaloby a následných finančních postihů ze strany poškozené firmy.

Činnost naší organizace na poli tvorby programového vybavení je dnes už všeobecně známa. Jedním z úkolů střediska vědeckotechnických informací v Martinské ulici v Praze je přímé přiblížení výsledků činnosti "šestsetdvojky". A že těchto výsledků není málo, o tom svědčí neutuchající zájem široké veřejnosti o nabídku původního programového vybavení pro osmibitové počítače v rozsahu prozatím asi dvaceti titulů. Je to zároveň odpověď na pochybnosti o prospěšnosti původních programů.

V současné době uvádíme na trh také původní programové vybavení pro šestnáctibitové počítače, kompatibilní s IBM PC/XT/AT. Jedná se především o originální český textový editor Text602, ale připravili jsme také interaktivní kursy MS-DOSu a dBASE III Plus, sadu programových nástrojů pro programátory a novou službu - zakázkové programové vybavení.

Děkuji za opravdu vyčerpávající odpověď na moji první otázku. Trochu to sice připomíná rozhovor s Vladimírem Menšíkem, který už obvykle ke druhé otázce ani nikoho nepustil, ale ten tvůj zápal pro věc je nádherný.

Te bych ale rád stočil řeč na zmíněné programy pro IBM PC, to je v nabídce šestsetdvojky novinka. A protože jsi spoluautorem editoru Text602, pověz nám něco o něm. Textový editor je bezesporu nejpoužívanějším programovým prostředkem. Pro PC jich existuje celá řada, ale ...

Ano je tu několik ALE, která brání v efektivním používání zahraničních produktů. Jak každý ví, zásadním problémem je naše mateřština. Tuto problematiku je možno zásadně rozdělit do třech oblastí: klávesnice, obrazovka a tiskárna.

S kávesnicí úzce související kódování, třídění a dělení češtiny i slovenštiny je kapitolou samo o sobě. Nejznámější a nejrozšířenější kód je americký standard ASCII, který je definován v rozmezí 0 až 127, kde v oblasti 0 až 31 jsou řídící kódy a v oblasti 32 až 127 je kompletní latinka, ale bez jakýchkoli akcentů. Pro češtinu a slovenštinu zbývají kódy 128 až 255. Tak byla definována norma KOI-8čs pro JSEP, která je odvozena z normy ISO-8. Tato norma umožňuje jednotlivým zemím definovat v rozsahu 192 až 255, tedy o 128 výše, než písmena v ASCII, svoji národní kódovou tabulku. Ta by měla mít svůj řád a měla by logicky navazovat na množinu ASCII.

Jenže IBM se na snahy International Organization for Standardization podívala svrchu a definovala si svůj znakový generátor pro PC tak, že od 128 do 175 jsou písmena z nejrůznějších abeced západoevropských jazyků, v oblasti 176 až 223 jsou semigrafické symboly a od 224 do 254 matematické symboly.

Člověk by si řekl, že tedy stačí definovat vlastní abecedu v oblasti kódů 128 až 175 a bude po starostech. Tak je definován rezidentní klávesnicový driver KEYBCS2, bratří Kamenických. Ti v době kdy grafická karta EGA byla malým technickým zázrakem, přiřadili písmenům západoevropských jazyků naší abecedu tak, aby alespoň trochu připomínala naše písmena s diakritikou. Tento kód samozřejmě nemá žádný řád, ale stal se jakýmsi neoficiálním standardem, protože neruší semigrafické symboly, které jsou v programech hojně využívány.

Aby to nebylo zas tak jednoduché, stanovila IBM normu LATIN 2 pro východoevropské jazyky, která svým chaosem připomíná KEYBCS2 v celé oblasti kódů 128 až 255. A aby ještě nebyl našemu utrpení konec, v RVHP se připravuje norma KOI-8L2, která opět není kompatibilní s oficiální normou LATIN 2. A teď si drazí uživatelé vyberte a zkuste

mezi sebou přenášet informace!

Samozřejmě, že rezidentních klávesnicových driverů je vyvinuto více. Jsou si víceméně podobné a přepínají klávesnici ČSR, SSR, standardní a kombinovanou pomocí funkčních kláves F1 až F10, spolu s kombinací kláves CTRL, SHIFT a ALT. Naštěstí většina z nich ctí kódování, které zavedli bratři Kameničtí.

Ruku v ruce s kódováním jde i třídění a dělení slov. U třídění lze snad v budoucnu předpokládat kladný posun, postavený na normě LATIN 2. Lze jistě úspěšně pochybovat o tom, že by u Microsoftů měli ponětí o našich složitých startovacích kriteriích. To ovšem tak úzce nesouvisí se zpracováním textu, ale spíše se zpracováním dat v různých databázích.

A co dělení slov?

To je problém nejen pro textový editor, ale hlavně pro DTP, který bude asi ještě dlouho řešen německou verzí programu a jakýnsi reprezentativním slovníčkem vyjímek. Pokud ovšem nedojde v budoucnu k rozumnější politice ze strany uživatelů DTP.

Začínám tušit, že ani s tou obrazovkou to nebude žádná legrace.

To tedy ne. Zobrazení češtiny a slovenštiny je jednoduché jenom pokud je v našem PC grafická karta EGA, nebo VGA, protože obě mají možnost definice vlastní znakové množiny v paměti RAM. Tyto karty jsou však stále ještě proklatě drahé. U ostatních, levných grafických karet - mám na mysli CGA a Hercules, které jsou pro řadu aplikací plně postačující - se jedná o netriviální problém úpravy již jednou daného programu. Také je možné vyměnit ROM za EPROM s předefinovaným znakovým generátorem. Nelze předpokládat, že by výrobci v dohledné době dodávali jako jeden ze znakových genrátorů i LATIN 2.

Zbývají ještě tiskárny. O těch už ledacos víme od osmibitových počítačů. Když je download ...

Tak tedy budu stručný. Ani download není bez problémů. Řada tiskáren má download jen pro oblast 33 až 126, tedy tam kde leží ASCII. A dalším problémem je volba správného kompromisu mezi umístěním diakritických znamének nad písmeny. Jde hlavně o znaménka nad velkými písmeny, protože ta se do standardní matice nevejdou. S tím potom souvisí také rychlost tisku. Buď lze tisknou rychle - ve standardní matici, ale s deformovanými velkými písmeny - nebo se separátním tiskem písmen a znamének, což však trvá dvakrát tak dlouho. Optimalizovaný způsob tiskne separátně pouze znaménka nad velkými písmeny, tedy tisk je zpomalen jen když se v řádce vyskytne velké písmeno s diakritikou.

Teď začínám být opravdu zvědav, jak jste se se vším tímhle vyrovnali.

Text602 je původní český textový editor, který byl vytvořen českými programátory. Proto respektuje všechny zvláštnosti českého jazyka. Pracuje na IBM PC/XT/AT kompatibilních počítačích s operační pamětí alespoň 384 kB a minimálně s jedním floppydiskem. Editor umí samozřejmě všechny standardní funkce, jako je práce s bloky textu, formátování odstavců, nalezení a nahrazení, skokové příkazy, definice formátu stránky atd. Velmi snadné je jeho ovládání pomocí systému inteligentních "pull down menu", který je navíc vybaven nápovědou, stručně vysvětlující každou položku v menu. Je to takzvaný "on line help". Text602 je možno ovládat také pomocí myši podle standardu Microsoft, popřípadě zkrácenými příkazy. Uživatelé WordStaru, SideKicku, XTree, Turbo Pascalu a dalších programů ocení jistě i možnost ovládání pomocí příkazů WordStaru.

Bez jakékoliv instalace je možno editor spustit na vších grafických kartách, tj. CGA, Hercules, EGA i VGA. Přitom, protože se jedná o editor typu WYSIWYG, bude psaný text na všech těchto kartách zobrazen téměř v takové podobě, v jaké bude vytištěn, včetně různých druhů písma a řádkování.

Jak je to s klávesnicí?

Náš editor nabízí pět typů klávesnic. ČSR a SSR jsou totožné s psacím strojem CONSUL a to i způsobem zadávání háčků a čárek nad velká písmena, což jistě ocení sekretářky a písařky. SPC je klávesnice vybavená speciálními symboly. Tím mám na mysli třeba matematické symboly, některá písmena řecké abecedy, linky a rámečky. Klávesnice ČSa je určena hlavně pro programátory a všechny ostatní, kteří nejsou zvyklí na klávesnici psacího stroje. Poslední možností je standardní klávesnice IBM. Všechny typy klávesnice jsou přepínány zkrácenými příkazy, nebo pomocí menu.

Máte tam i dělení slov?

Dělení slov podle pravidel českého pravopisu je možno zařadit při přeformátování odstavce nebo bloku. Dokonce je pamatováno i na přetahování krátkých spojek a předložek z konce řádku. Je možno si zvolit dělení automatické i s potvrzením. Pro dělení je použit originální úsporný algoritmus. Vzhledem k záludnostem češtiny by však byl nutný pro stoprocentní úspěšnost rozsáhlý slovník vyjímek. Je tedy jistější používat při dělení slov individuální potvrzení volby.

Velký problém bývá kolem kompatibility s ostatními programy.

Uživatel si může zvolit vstupně/výstupní kód KEYBCS2, LATIN 2, nebo KOI-8čs pro kódování češtiny a dále si může zvolit vhodný "export", tj. výstupní formát. Jedna z možností výstupního formátu je formát WordStar, což umožňuje například přenos textu do DTP systémů.

Nakonec se ještě pochlub tím, jak se Text602 vyrovnává s nepřehlednou džunglí kolem tiskáren.

To bude asi jednou z nejsilnějších stránek našeho editoru. Dokáže totiž tisknout prakticky na všem, ať už to jsou tiskárny s devíti, nebo čtyřiadvaceti jehličkami, s typovým kolečkem, s českou pamětí ROM, laserové tiskárny a další. Je to zajištěno pomocí tzv. DST souborů, neboli definičních souborů tiskárny. Standardně dodáváme DST soubory pro tisk v grafickém módu. Lze však přikoupit DST soubory pro download všech běžnějších maticových tiskáren, které se vyskytují v ČSSR, ale také DST soubory pro download laserových tiskáren Hewlett-Packard HP LaserJet Plus/II/IID a kompatibilních s emulací jazyka PCL. Teprve s downloadem je možné plně využít všech možností editoru při plné rychlosti tisku. Jsme schopni nadefinovat příslušné DST soubory i pro další typy tiskáren na přání zákazníka.

Tak se mi zdá, že tenhle nový editor by vyřešil celou řadu problémů, se kterými se potýkáme v naší redakci.

A to ještě není všechno. Kromě uvedených funkcí nabízí Text602 ještě definování a práci s makry, interaktivní práci s adresáři a soubory, možnost provádění příkazů MS-DOSu přímo z editoru, odchod

o operačního systému se zpětným návratem (tzv. OS shell), vestavěný elektronický kalkulátor s možností vložení výsledku do textu, přepínání grafického režimu, zobrazení aktuálního času, vkládání data a času do editoru...

Zaraž, už jsi mě přesvedčil, budeme asi jedním z prvních zákazníků. Jak je to s možností získání editoru?

Text602 je možno objenat na adrese: 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6. Informace je možno získat i osobně ve středisku VTI Svazarmu, Martinská 5, Praha 1, kde si také zájemci mohou zdarma nakopírovat demoverzi. Ta ovšem neukládá na disk a netiskne. Editor se dodává na disketách 5¼ palce. Součástí dodávky je i obsažná příručka, která byla zpracována a vytištěna v editoru Text602 na laserové tiskárně HP LaserJet IID.

Děkuji za rozhovor.

Rozmlouval Daniel Meca

Nakonec jsme se dohodli, že dostanu editor k testování v redakci. Richard mi dokonce upravil download pro laserovou tiskárnu tak, aby pracoval v potřebné šířce písma. A tak je, počínaje tímto číslem, Mikrobáze kompletně zpracovávána editorem Text602.

PROBLÉMY KOLEM KLÁVESNICE ZX SPECTRA

Herbert Urbanec

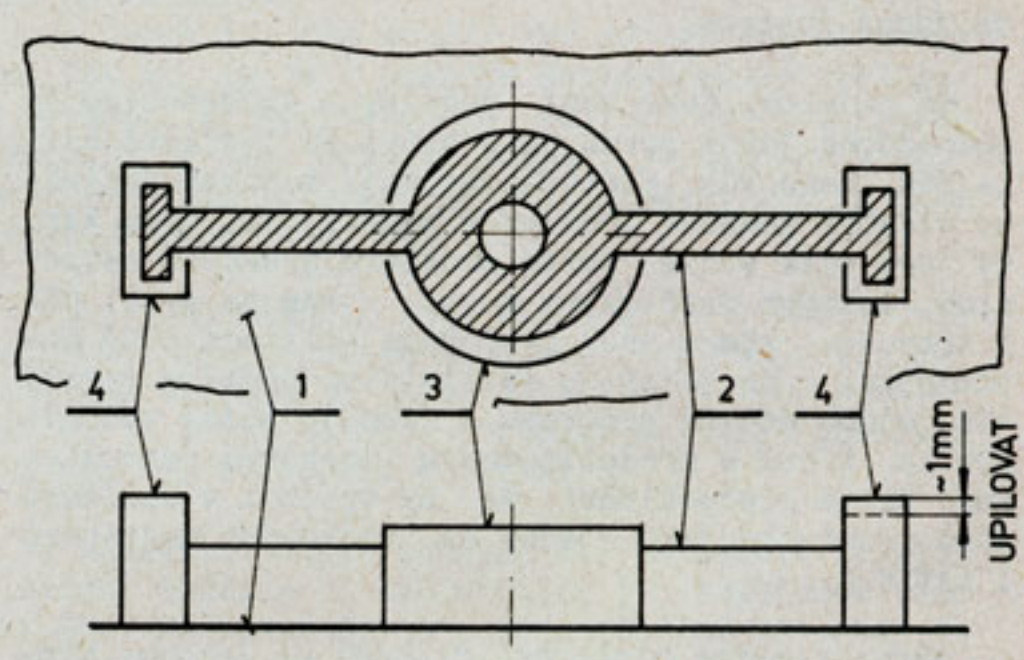
Snad každý majitel Spectra narazí po kratší či delší době na problém funkčnosti (lépe řečeno nefunkčnosti) klávesnice. Vzhledem k tomu, že se pan Sinclair opravdu snažil "ušetřit" kde se dá, je to bohužel vbrzku znát i zde.

Rád bych čtenáře seznámil s několika drobnostmi a poznatky, které jsem získal při opravách Specter a také jejich klávesnic.

První problém se týká pouze majitelů Spectra+ a souvisí s jeho vylepšenou klávesnicí. Různých, většinou preventivních úprav, zabývajících se chodem mechanické klávesnice Spectra+, bylo na stránkách různých časopisů již uveřejněno několik. Většinou jde o různé podložky, ať již jednotlivě pod každou klávesu, nebo v pásech pod celé řady, které tvoří jakési dorazy a zabraňují prostřižení membrány při necitlivém zacházení (většinou při hraní různých her). Ovšem po několika opravách klávesnice jsem zjistil, že některé série Specter mají špatně provedené výlisky horního krytu, což se projevuje tím, že pomocná vodítka u obou kláves CAPS SHIFT a u klávesy ENTER viditelně převyšují ostatní hlavní vodítka a především otvory pro samořezné šrouby. Po sestavení klávesnice, při trochu silnějším utažení šroubů dojde k deformaci a často i k následnému poškození membrány a to v místech, kterými jsou vedeny spoje. Při rozebrání klávesnice je tato závada patrná již na první pohled viditelnými otlaky, jak na membráně, tak i na gumové podložce, která membránu odděluje od vodítek a jezdců kláves. Nejčastější jsou deformace (a bohužel především prasklé spoje) v mezerách mezi CAPS SHIFT a CAPS LOCK a mezi tečkou a CAPS SHIFT.

Při provozu se tato závada projevuje buď tím, že některé (většinou zdvojené) klávesy odmítají pracovat, nebo naopak některá klávesa spíná, i když nemá - v drtivé většině CAPS SHIFT, takže potom místo malých písmen píšeme velká, místo nuly DELETE, při použití SYMBOL SHIFTU neustále naskakuje EXTEND MODE - a práce s takovouto klávesnicí jistě není příliš pohodlná.

Úprava, odstraňující tuto závadu, je velmi jednoduchá a spočívá v ubrání výšky pomocných vodítek zhruba o jeden milimetr. Případné větší zkrácení nemůže být na škodu, spíše naopak, jelikož vodící šachta je i po úpravě stále dostatečně dlouhá a zabezpečuje spolehlivý chod klávesy. (Viz obr. 1)



Obr.1

1. Základna (horní kryt počítače)
2. Jezdec, který při stisku tlačítka dosedá přes pružnou gumovou odložku na membránu klávesnice
3. Hlavní vodítko jezdce (u všech kláves)
4. Pomocná vodítka - u CAPS SHIFTů dvě
   u ENTER tři

Druhý problém se týká především těch majitelů Spectra, kteří ve svém počítači neustále něco kutí, předělávají či opravují a nevyhnou se tedy občasnému rozebrání počítače a tím také manipulaci s přívody klávesnice (tzv. "kšandami").

Tyto přívody jsou vlastně přímým pokračováním membrány klávesnice a jsou tedy tvořeny tenkou plastickou fólií s napařenou vodivou vrstvou stříbra. Z tohoto důvodu je zřejmé, že je-li životnost klávesnice dosti omezena, je životnost přívodů k ní prakticky nulová. Stačí několikrát (byť i sebeopatrněji) vyndat a zandat konec přívodů do konektorů v plošném spoji a kšanda se ohne a posléze i nalomí, čímž se ovšem přeruší napařený spoj a obvykle přestane fungovat část klávesnice. Pokud to ještě délka přívodů dovoluje (neboť kšandy mají zpočátku jisté délkové rezervy), ustřihneme prostě přívod na místech zlomu, vodivou stranu folie pečlivě očistíme (nejvhodnější je tvrdá gumovací pryž - modrá, šedivá - kterou již delší dobu úspěšně používám na čištění stříbrných vodičů, ale i přímých konektorů na plošném spoji) až do světlého lesku a přendáme tenkou destičku z ustřižené části, sloužící ke zpevnění konce přívodu, který se zasouvá do konektoru (u některých sérií Specter chybí dokonce i tato vymoženost!).

Po několika podobných opravách však dospějeme ke stavu, kdy jeden z přívodů (popřípadě oba) je již natolik zkrácen, že prostě ve složeném stavu nedosáhne ke konektoru. Viděl jsem již několik úprav, spočívajících v různém nastavování drátky, lepením proužků staniolu apod., avšak žádná z nich mne nepřesvědčila, jak svým vzhledem a funkčností, tak především trvanlivostí. Každý, kdo se již tímto nemilým problémem zabýval, jistě došel k závěru, že folie se prostě pájet nedá a že jakékoli různě vsunuté drátky a spojky při nejbližší příležitosti upadnou - takže jsme takřikajíc tam, kde jsme byli. Protože jsem i já, asi po tříletém (a to dost razantním) používání klávesnice, narazil na "kšandový" problém a bylo mi líto vyhodit jinak úplné funkční membránu, rozhodl jsem se pro jednoduchou, funkční a především spolehlivou úpravu. Ta umožňuje využít jinak dobré membrány, které jistě zahálí u mnoha majitelů Specter v šuplíku (pokud je už rovnou nevyhodili).

Úpravu provedeme následujícím způsobem:

- v plošném spoji vypájíme oba přímé konektory od přívodů klávesnice

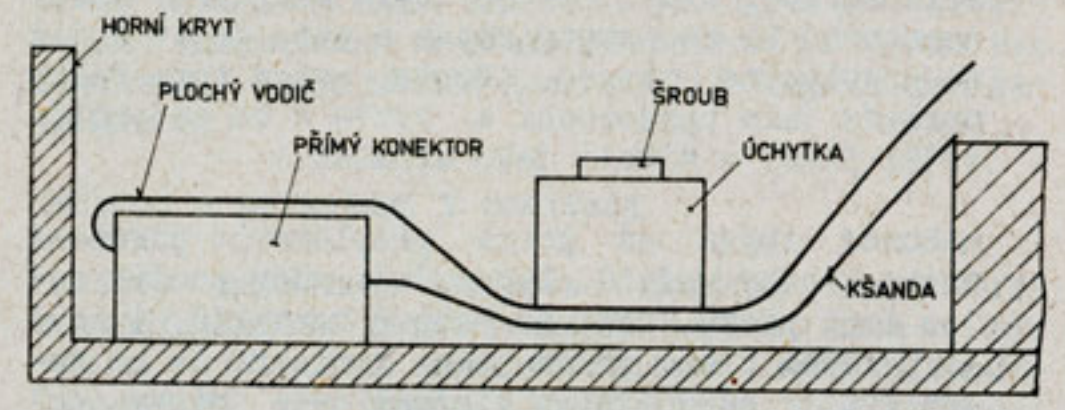
- nyní je potřeba nahradit tyto konektory jinými o 5 a 8 kontaktech, ke kterým vlastníme i protikusy. Mě se nejvíc osvědčil konektor FRB, zkrácený na dva kusy po 5-ti a po 8-mi kontaktech a potom podélně rozříznutých na jednořadé. Já jsem tyto úkony provedl pouze za pomoci jehlového pilníku a pilky na železo a mohu říci, že výsledek byl více než uspokojivý. Do plošného spoje zapojíme konektory osazené dutinkami s kolmými vývody pro plošné spoje, které však před osazením konektoru ohneme asi o 40 stupňů tak, aby konektor mířil šikmo dozadu (je to velmi nutné vzhledem k výšce klávesnice). Protikus použijeme nejlépe v provedení pro ovíjené spoje, přičemž vývody opět asi o 40 stupňů ohneme tak, aby byly po zasunutí konektoru konce vývodů zhruba vodorovně dozadu (popř. nepatrně šikmo nahoru). Je možné, že u obyčejných Specter ("gumáků") bude i tak celková výška obou konektorů stále ještě bránit sestavení počítače. Potom je možné opatrně trochu ohnout i kolíčky protikusů.

- na protikusy konektorů připájíme asi desetičentimetrové kousky plochého vodiče (nezapomeňte na bužírky, aby nedošlo ke zkratu o plechový kryt klávesnice) a na druhý konec připájíme původní přímé konektory, vyjmuté z plošného spoje. Pozor na kontakty u těchto konektorů! Jsou pouze jednostranné a u každého konektoru jinak orientované. Správnou orientaci určíme podle vodivé vrstvy na kšandách.

- nyní obě kšandy uvolníme odšroubováním obou příchytek na horním krytu a zkrátíme přibližně na 33-36 mm (u Spectra+) od hrany membrány.

- přesvědčíme se, zda není žádná část přívodů na tomto úseku poškozená či nalomená (nejčastěji bývá kšanda natržená těsně u hrany membrány) a pro jistotu ještě očistíme všechny spoje tvrdou pryží - viz výše. To se týká především Specter+, kde je navíc ještě jeden pásek pro zdvojené klávesy. V případě, že je vše v pořádku, můžeme pokračovat dále. Jinak zbývá ještě možnost obnovy poškozených přívodů, kterou popíši později.

- nakonec obě "prodlužovačky", které jsme si připravili, nasuneme na zkrácené kšandy, plochý vodič předtím ohneme zpět přes okraj konektoru tak, aby po nasunutí vedl vrchem přes kšandu zpět směrem dopředu a uchytíme jej zároveň s kšandami pod úchytky (viz obr. 2). Tím zabráníme při tahu za vodič namáhání pájeného místa a případnému utržení.



Obr.2

- klávesnici (zatím bez sešroubování počítače) zapojíme pomocí prodlužovaček a vyzkoušíme všechny (!) klávesy - nezapomeňte na oba SHIFTY. V případě, že některé klávesy (většinou zdvojené) nefungují, ještě jednou velmi pečlivě očistíme přívody a popř. zkusíme kšandy pod úchytkami vhodně vypodložit, abychom zajistili spolehlivý kontakt mezi všemi vrstvami přívodů.

- pokud nenastaly již žádné další problémy, můžeme nyní počítač sešroubovat a opět vyzkoušíme klávesnici.

Tato úprava má výhodu nejen ve využití původní "nefunkční" membrány, ale i v tom, že při dalším rozebírání již prakticky nemůžeme kšandy poškodit, a případný upadlý pájený spoj, či snad přerušený plošný vodič (neumím si ovšem vůbec představit, jak by bylo možné tento vodič nějak neúmyslně poškodit) lze kdykoli snadno opravit.

Snad jedinou nevýhodou této úpravy je, že v případě získání nové membrány není možno použít původního způsobu připojení. Jedinou možností je tedy zkrácení přívodu (což ovšem já nepovažuji ani tak za nedostatek, jako spíš za preventivní opatření, které nás zbaví všech následných, již popsaných problémů). Myslím si však, že tato otázka bude tížit jen nepatrnou část majitelů Specter, jelikož nedostatek membrán je více než kritický.

Na závěr bych chtěl ještě popsat jednu úpravu, která umožňuje, v návaznosti na předchozí, použít i ty membrány, které mají poškozené přívody těsně u okraje membrány, či v dané délce zkrácených přívodů.

Upozorňuji předem, že tato úprava je poněkud obtížnější a zdlouhavější, než by se mohlo při pouhém čtení zdát, vyžaduje značnou dávku trpělivosti a jistou pečlivost.

Po odstřižení přívodů nám zbydou v podstatě celé neporušené pásky vodičů. Tyto pásky zásadně nevyhazujte! Je možné je použít buďto v nyní popisované úpravě, nebo kdykoli později k opravě přerušeného spoje na přívodech. Je to tedy dost cenný materiál. Nyní již vlastní postup úpravy zlomeného, či přerušeného spoje:

- oprava se provádí přeplátováním porušeného místa kouskem dobrého vodiče ze zbytku kšandy a to tak, že oba vodiče musí být orientovány vodivými vrstvami k sobě.

- před přeplátováním je nutno jak "plát" tak i původní vodič velice pečlivě očistit. Opět vřele doporučuji již výše zmíněnou tvrdou gumovací pryž, pomocí které nejen odstraníme zoxidovaný povrch vodičů, ale především lepidlo, které na původním spoji zůstalo po odtržení (samozřejmě velmi pečlivém, abychom nezničili ještě něco dalšího) další vrstvy v případě,

že opravovaný spoj se nachází již v místě, kde jsou jednotlivé vrstvy membrány slepeny. Pozor! Tento bod je nejdůležitější na celé opravě a na jeho pečlivém provedení závisí nejen úspěšnost celé opravy, ale i "život" celé membrány. Stačí nepozorný či neopatrný zásah a může dojít k již neopravitelné závadě. (Berte ovšem tyto řádky opravdu jako upozornění a výzvu k co nejpečlivější práci a nikoli jako strašák.)

nakonec zbývá už pouze přeplátovat porušená másta kousky vodičů. Doporučuji vždy použít celou šíři pásku, tedy 5, nebo 8 kontaktů, i když je porušen třeba jenom jeden kontakt. Je to nejen kvůli snadnějšímu vložení celé pásky, než jen úzkého proužku, ale především proto, aby opravený přívod měl na celé své šířce stejnou tloušťku. Tak bude zajištěn po přitažení pod úchytky (a případném podložení) co nejlepší kontakt pro všechny vodiče a vrstvy. Potom již postupujeme dle předchozího návodu, jako s neporušenou membránou.

A úplně na konec ještě jedna poznámka. Popsaný způsob opravy se dá použít i k opravě některých přerušených spojů přímo v membráně, je však velmi náročný a doporučuji ho pouze jako nouzové řešení. V takovéto situaci by však bylo již lepší pořídit si externí klávesnici. O této problematice rovněž vyšlo několik příspěvků v časopisech. Například tak trochu aprílové řešení v [2]. Za nejlepší ovšem pokládám [3], kde se zabývají nejenom problémem klávesnice, ale popisem Spectra vůbec (je to zatím nejlepší a nejkomplexnější popis, který jsem kdy u nás viděl). Jako další náměty na provedení klávesnice viz [4] - [6].

K otázce hardwarového připojení klávesnice bych se rád vrátil někdy příště.

Literatura:

[1] Amaterské radio, řada A, č. 8/88, str. 302
[2] Amaterské radio, řada A, č. 3/88, str. 104
[3] Amaterské radio, řada B, č. 1/89, téměř celé číslo
[4] Amaterské radio, konstrukční příloha 1986, str. 13-15
[5] Amaterské radio, příloha Mikroelektronika 1987, str. 3-13
[6] Elektronika, č. 7/87, str. 27

PŘEVODNÍK CENTRONICS —› RS 232C

ing. Jiří Loskot

Tak jako řada dalších uživatelů počítače, jsem se dostal do situace, kdy člověk sežene periferii, kterou není možno přímo připojit ke stávajícímu systému. V mém případě to byla tiskárna se sériovým rozhraním RS 232C, zatímco můj Sharp MZ 821 má výstup na tiskárnu definován rozhraním CENTRONICS.

Tento problém lze v zásadě řešit třemi způsoby:

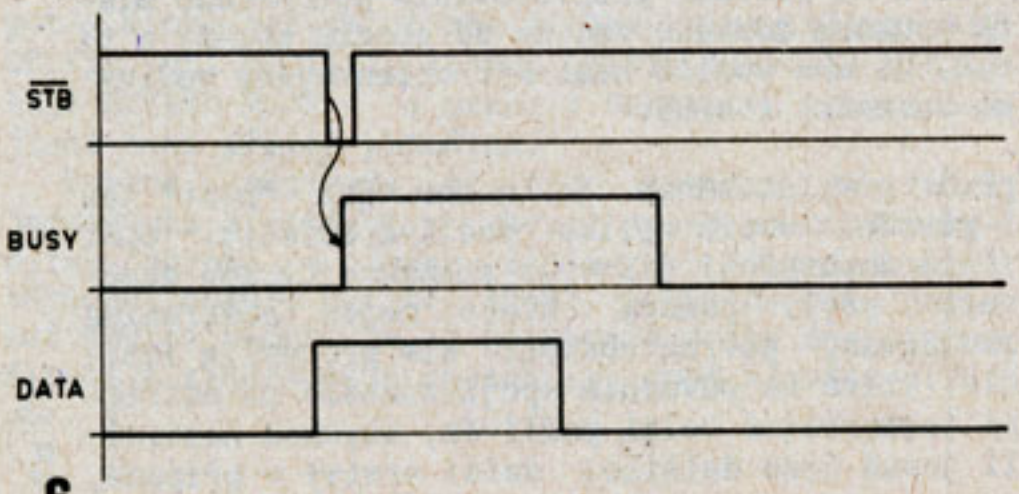
1) Použít čistě programové řešení - vznikne problém s neslučitelnými napěťovými úrovněmi TTL a RS 232C.

2) Připojit univerzální převodník USART - výhoda šestranné komunikace, ale nevýhoda pro uživatele, kteří jsou nuceni předělat stávající programové vybavení na obsluhu USART.

3) Použít čistě obvodové řešení, které sice není univerzální, ale umožňuje použití veškerého stávajícího vybavení. Lze je též beze změny použít i na jiném mikropočítači, vybaveném paralelním rozhraním CENTRONICS či podobným.

Třetí způsob se v mém případě zdál nejvýhodnější. Nejprve se stručně zmíním o definici signálů pro rozhraní CENTRONICS a RS 232C.

Pro základní komunikaci pomocí rozhraní CENTRONICS jsou nutné tři signály, jejichž vzájemné vztahy vyplývají z následujícího časového diagramu:



kde /STB je signál z počítače oznamující tiskárně platnost dat k převzetí. Délka minimálně 0,5 mikrosekundy.
BUSY je signál od tiskárny oznamující, že data nebyla ještě přezata a zpracována.
DATA data vyslaná počítačem na osmibitovou sběrnici ke zpracování tiskárnou. Musí být v předstihu před signálem STB nejméně o 0,5 mikrosekundy.

Rozhraní RS 232C je sériové s následujícími napěťovými úrovněmi:

-3V až -15V	log I
+3V až +15V	log 0
-3V až +3V	zakázaný stav

RS232C tedy pracuje v takzvané negativní logice s úrovněmi neslučitelnými s logikou TTL. Posloupnost přenášeného signálu je dána časovým diagramem:

Z obou obrázků je zřejmé, že je nejprve nutno převést slovo z paralelního tvaru na sériový a též generovat signál BUSY po celou dobu vysílání dat v sériovém kódu. Dále je nutné upravit napěťové úrovně z TTL na RS 232C. Při konstrukci převodníku jsem dále vycházel z požadavku napájení pouze +5V a použití minima součástek.

Popis a realizace převodníku

Celý převodník lze rozdělit na čtyři části:

1) vlastní převodník paralelního kódu na sériový.

2) generátor hodinového kmitočtu pro zadanou modulační rychlost.

3) zdroj napětí -12V.

4) převodník napěťových úrovní.

Pro převodník kódu byl použit obvod Tesla MHB 1012, který umožňuje obvodové nastavení jednotlivých parametrů výstupního signálu podle této tabulky:

vývod	funkce	log. úroveň
NP 35	vyslání paritního bitu	I=ne 0=ano
TSB 36	počet vyslaných STOP bitů	I=dva 0=jeden
NB1 37	počet vyslaných	0-5 0-6 I-7 I-8
NB2 38	datových bitů	0/ I/ 0/ I/
EPS 39	typ paritního bitu	I=sudá 0=lichá

kde log. I = DIL rozpojen, log. 0 = DIL spojen.

Platnost nastavení vývodu EPS závisí na nastavení vývodu NP. Nastavení těchto parametrů je nutné provést před připojením převodníku k napájení.

Strobovací vstup /DS je ošetřen monostabilním klopným obvodem, takže je schopen komunikovat s obvody, které nesplňují přímo normu CENTRONICS (např. Z80-PIO).

Vývod TMBM posílá potvrzovací signál BUSY pro rozhraní CENTRONICS. V případě, že při sériovém přenosu nebudeme využívat signál /CTS, který idikuje nepřipravenost tiskárny k příjmu dat, stačí diodu D6 nahradit drátovou propojkou a součástky D7, D8, R6 a R11 neosadit. Odpor R11 osadíme v závislosti na napěťových úrovních cignálu /CTS. Má-li tento signál napěťové úrovně 12 až 15V, bude jeho hodnota 4700 ohmů. Při úrovních TTL lze odpor nahradit drátovou spojkou.

Vývod TCP slouží pro připojení generátoru hodinového kmitočtu modulační rychlosti. Tento kmitočet musí mít hodnotu šestnáctkrát větší než je požadovaná modulační rychlost (např. pro 300Bd bude f= 4800Hz). Generátor tohoto kmitočtu je tvořen standardním zapojením s astabilním obvodem typu BE 555. Kmitočet je určen hodnotami kapacity C a odporu R podle přibližného vzorce

$$f = \frac{1,49}{C*(R+2*R3)}$$

Modulační rychlosti 300Bd odpovídají hodnoty: R = 27k a C = 5,6 nF. S ohledem na přesné nastavení je vhodné nahradit odpor R kombinací odporu a odporového trimru. Kondenzátor C použijeme nejlépe styroflexový.

Jelikož obvod MHB 1012 potřebuje napájecí napětí -12V a též pro přenos pomocí RS 232C je nutné záporné napětí, použil jsem v převodníku nepřímo buzený jednočinný měnič, který generuje záporné napětí -10 až -12V. Měnič je buzen periodickým signálem o kmitočtu 40 - 60 kHz, vytvářeným jednoduchým generátorem z hradel TTL.

Vlastní transformátor je navinut na ferritovém jádře o průměru 14 - 18 mm z hmoty H22 nebo H12. Tranzistor KF 508 je vhodné vybrat s co největším zesílením a opatřit chladičem. V případě malého výstupního napětí je nutno zmenšit odpor R5. Zdroj musí být schopen dodat prcud minimálně 15 mA.

Poslední částí převodníku je obvod upravující úroveň výstupního signálu na hodnoty odpovídající rozhraní RS 232C. Pro jednoduchost jsem zvolil napěťové úrovně +5V a -10V. Vlastní obvod je realizován pomocí IO typu MAA 748, použitém jako invertující komparátor s překlápěcí úrovní +1,4V. Pro odstranění zakmitávání jsem použil kompenzační kapacitu C6.

Oživení a připojení k počítači

Při oživování nejprve osadíme zdroj -12V a zkontrolujeme jeho výstupní napětí. Mělo by být -10 až -12V. Poté překontrolujeme logickou sondou činnost MKO pro úpravu signálu /STB. Po překlopení STB z log. I na log. 0 musí na výstupu generovat krátký impuls úrovně log. 0.

Hodinový kmitočet nastavíme nejlépe pomocí čítače, případně osciloskopu na odpovídající hodnotu. Jemné doladění provádíme trimrem o hodnotě přibližně 1/10 R, zapojeným v sérii s odporem R. Po osazení ostatních součástek připojíme převodník k počítači. Připojení je nutné provést při vyputém počítači!

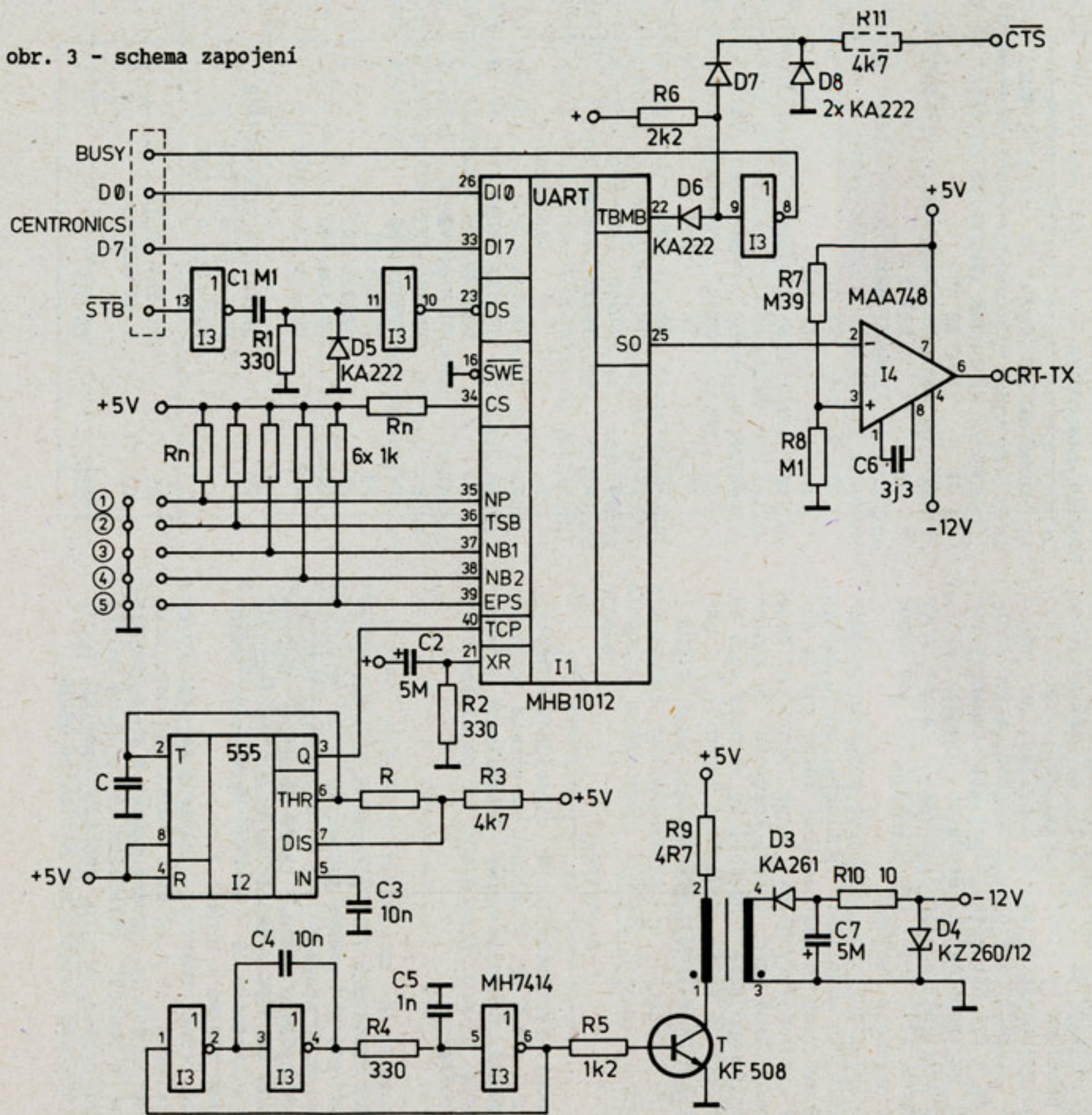
Po zapnutí napájení můžeme vyvolat rutinu tisku na LPT a na výstupu převodníku bychom měli dostat signál v sériovém tvaru, podle nastavení spinačů DIL.

Převodník je možné připojit i k jinému typu počítače než je SHARP MZ-821. Plošný spoj je navržen pro snadnou úpravu pořadí signálů proškrábnutím a propojením drátovou spojkou.

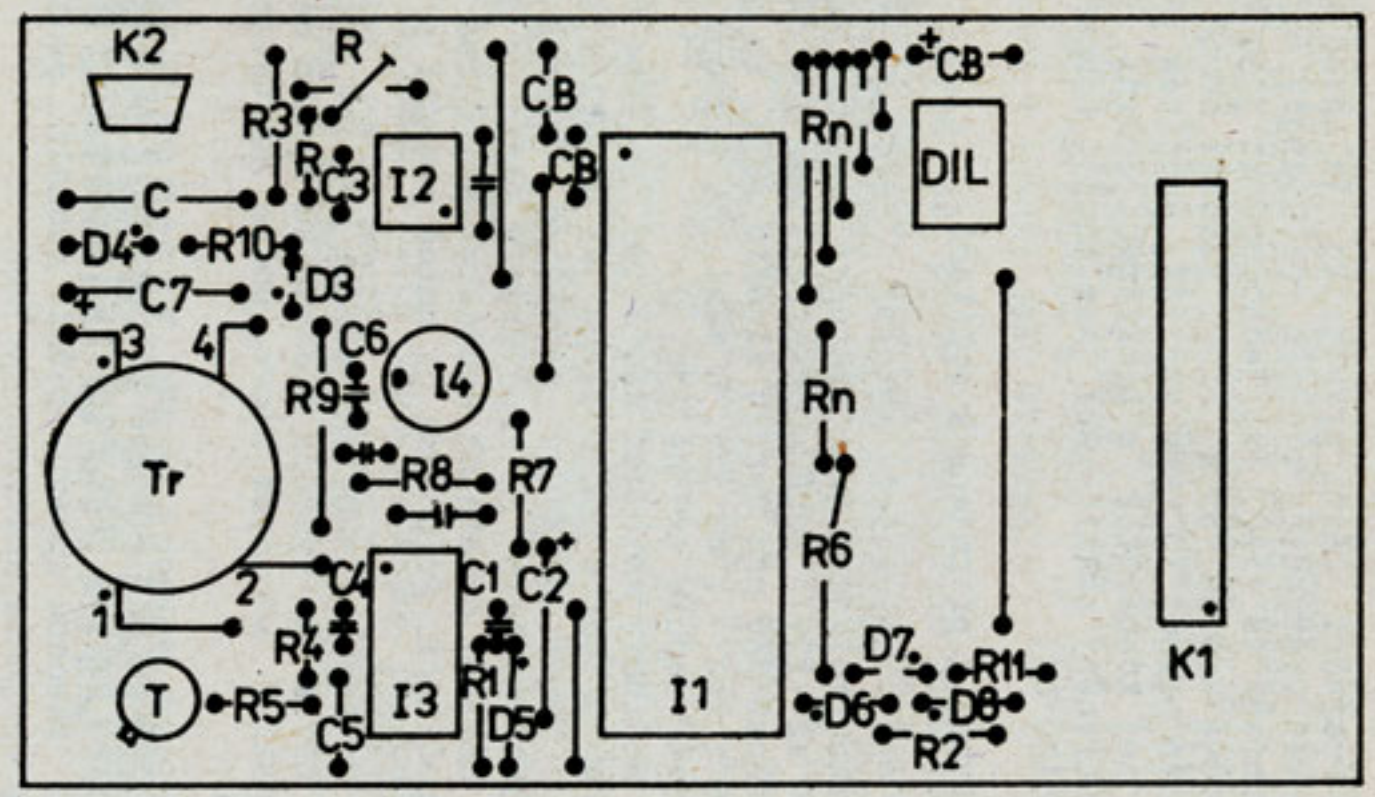
Použité součástky

Označení	Typ		Kusů
I1	MHB 1012		1
I2	BE 555		1
I3	UCY 7414		1
I4	MAA 748		1
T	KF 508		1
D1, 2, 5-9	KA 222		7
D3	KA 261		1
D4	KZ 260/12		1
C1, B	TK	M1	4
C2	TE982	5M	1
C3,4	TK	10K	2
C5	TK	1K	1
C6	TK	3j3	1
C7	TE986	5M	1
C	TC215	viz.popis	
R		viz.popis	
Rn	TR112a	1K	6
R1,2,4	TR112a	330R	3
R3	TR112a	4K7	1
R5	TR112a	1K2	1
R6	TR112a	2K2	1
R7	TR112a	M39	1
R8	TR112a	M1	1
R9	TR112a	4R7	1
R10	TR112a	10R	1
R11	TR112a	viz.popis	
Tr	jádro H22 prům.14*8*8 N1 - 12z 0,2 CuL N2 - 50z 0,2 CuL		
K1	TY 513 30 11		1
K2	2 WK 462 33		1
DIL	TS 501		1

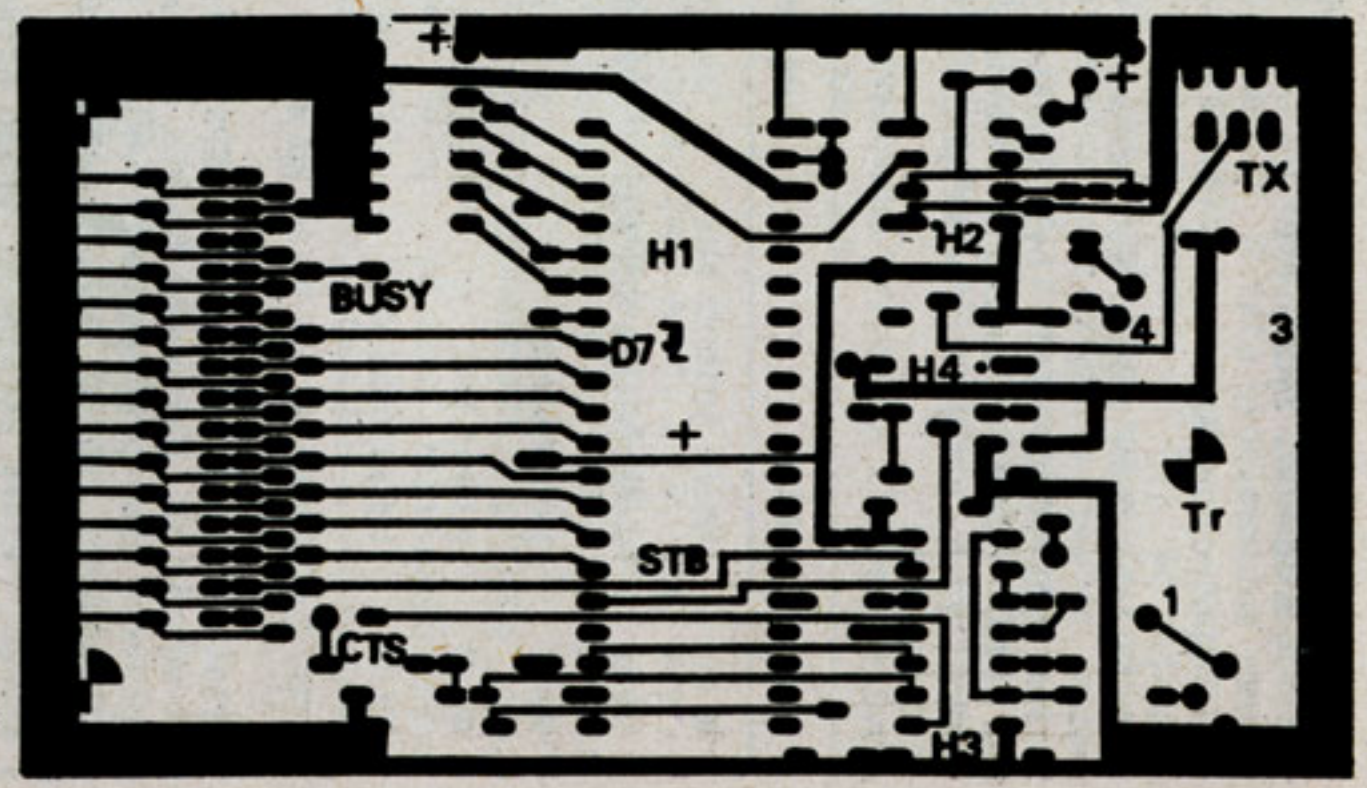
obr. 3 - schema zapojení



obr. 5 - rozložení součástek



obr. 4 - plošné spoje



CP/M · CP/M · CP/M

Postavte si s námi diskový řadič

/2/

Daniel Meca

Jak jsem již slíbil minule, teď potěším hlavně spektristy, protože následující popis systému TR-DOS 5.xx, kterým je Betadisk vybaven, je určen pro ně. Nic zlého se však nestane ani majitelům jiných počítačů, když si následující řádky přečtou. Naopak, mohou se inspirovat pro vlastní vývoj něčeho podobného.

Tak tedy hned úvodem to tajemné zapojení, o němž jste měli od minule přemýšlet. No ovšem, teď už je to jasné i těm, kteří tak trochu váhali. A pro ty, kterým to nic neříká ani teď, doplním vysvětlení. Při normální činnosti ZX Spectra se z paměťové oblasti znakového generátoru čtou jenom data. Pokud v této oblasti voláte podprogram, jsou zde čteny také instrukce procesoru. Přitom procesor generuje vedle /MRQ ještě signál /M1. No a na to právě číhá hradlo 1 v IO3. Jestliže tedy IO2 vydekóduje příslušnou adresu a zmíněné hradlo 1 potvrdí že se čte instrukce, překlopí výstup 6 hradla 2 v IO3 klopný obvod 2 v IO5. Tím dojde k přepnutí EPROM na místo ROM a zároveň se hradlem 2 v IO6 zablokuje signál /IORQ' pro další periférie. Pak je při /IORQ a adresových vodičích A0, nebo A1 na úrovni log. 0 odblokován také řadič. TR-DOS je tedy akceschopný. O odstránkování se stará zase jiná část zapojení. Pokud je A14, nebo A15 na log. 1 (to pozná hradlo 4 v IO4), stačí už jen číst instrukci a výstup 11 hradla 4 v IO3 způsobí zpětné překlopení klopného obvodu 2 v IO5. Tím se opět odpojí Eprom, zablokuje se deska řadiče a uvolní se průchod signálu /IORQ na vývod /IORQ'. Všimněte si, že i při odstránkování je podmínkou čtení instrukce v oblasti RAM. DOS používá toiž část RAM pro své systémové proměnné a při práci s nimi nesmí dojít k odstránkování.

Zvlášť pěkné na celé věci je to, že pokud není přestránkováno a TR-DOS není přímo v činnosti, nemůže program běžící na Spectru poznat, že je něco připojeného. Obsah ROM je původní a na V/V adresách nic nevisí. To je přesně ta správná chvilka, kdy můžeme takový nic netušící běžící program uschovat na disk. Ochrana neochrana. Na to je totiž u Betadisku tlačítko (firma TR ho krásně pojmenovala "Magic button"), které stačí zmáčknout a je to. Co že se přitom děje uvnitř? Při první žádosti procesoru o RAM (/MRQ na nule a adresa A14, nebo A15 na log. 1), je aktivován monostabilní klopný obvod 2 v IO9 a od něho MKO1 v IO9. Tak je jednak přestránkováno na EPROM (překlopený klopný obvod 2 v IO5), a jednak je vyvoláno nemaskovatelné přerušení.

Příslušná rutina v EPROM pak nejprve ze všeho uloží všechny registry a nahraje na disk prvních 512 bajtů obrazové paměti. Protože v tu dobu ještě DOS nedisponuje žádnou RAM pro práci s adresářem, jsou použity sektory 10 a 11 na stopě 0, které jsou vždy vyhraženy pro práci systému. Uvolněnou část VIDEORAM pak DOS používá celkem běžným způsobem pro své účely při uložení obsahu celé RAM na disk. Pochopitelně se po skončení ukládání do vypůjčené části RAM zase vrátí její původní obsah z disku, takže program pokračuje tam, kde byl přerušen.

Jenže to už máme na disku, pod provizorním názvem "@" jeho kopii. Kdykoli dáme DOSu příkaz GOTO "@" CODE, načte se obdobným způsobem do počítače kompletní obsah RAM, včetně stacku, takže po obnovení obsahu uložených registrů běží uložený program od toho místa, kde byl přerušen stiskem tlačítka. To je vlastně celé to kouzlo.

TR-DOS má ještě jednu příjemnost. Při prvním startu je totiž automaticky spuštěn program v Basicu, nazvaný "boot". Pochopitelně, je-li na disku (když není, systém se nijak nebrání). Program "boot" je možno spustit z DOSu kdykoli, pouhým RUN. K čemu je to dobré, to snad ani nemusím psát.

Teď také nějakou nevýhodu TR-DOSu. Tak předně - jsou v něm chyby. Jenže, ve kterém systému vlastně nejsou. Ostatně, alespoň je co vylepšovat. Další nevýhodou TR-DOSu je to, že si v RAM potřebuje uložit své systémové proměnné. Udělá to obdobně jako Microdrive - posune začátek Basicu o těch potřebných 112 bajtů nahoru (Basic pak tedy začíná na 23867). Někdy to vadí, někdy ne. Až budu popisovat různé maličkosti a finty kolem Betadisku, přidám k lepšímu krátkou rutinku, která to po načtení programu zase napraví, takže poběží i "stroják" v řádku 0. Horší je, že se posunutím Basicu nahoru zmenší prostor pod RAMTOPem. A aby byla v RAM tlačenice pod RAMTOPem ještě větší, využívá TR-DOS 5.xx pro práci s adresářem ještě 256 bajtů ve WORKSPACE, nebo pod Basicem. Je to jen na chvilku při čtení či zápisu, ale může to vyvolat hlášení "Out of RAM" a příkaz se neprovede. Příkazy LIST a MOVE mají ještě větší nároky na paměť.

Dvoupólový třípolohový přepínač slouží jednak k resetu, jednak umožňuje zvolit druh provozu. Střední poloha je určena pro spolupráci s normálním Spectrem 48k. Navíc je v této poloze pomocí R7, R8 a D8 umožněna kompatibilita s programy, upravenými pro starší verze TR-DOSu. Zde je nutná malá odbočka. Verze 5.xx se volá na adresách 15616 a 15619, zatímco předchozí verze používaly adresy 15360 a 15363. Pro zachování alespoň částečné kompatibility programů jsou v DOSu 5.xx na starých adresách umístěny skoky na nové adresy. Tento způsob volání však nejde použít ve spojení s některými periférierni, které do původně prázdné oblasti ROM vkládají svůj program. V takovém případě může pomoci už jen hardwarová úprava. Odporem R8 se přes diodu D8 podrží vstup 4 adresového dekodéru IO2 na log. 1. Protože je na tento vstup normálně přiváděna A8, dekódují se adresy vždy, jako by A8 byla na jedničce. Při volání 15360 je dekódováno 15616 atd.

Zbývající krajní poloha přepínače je určena pro provoz ve spojení se Spectrem 128. V této poloze je jednak vyřazená výše zmíněná hardwarová úprava adresování, jednak je při resetu blokováno vyvolání TR-DOSu. Systém se pak musí volat softwarově.

Při prvním volání si TR-DOS inicializuje systémové proměnné a testuje si připojenou mechaniku (to způsobuje ty strašlivé zvuky, které při inicializaci vydává mechanika). Je schopen sám určit, zda mechanika má 40, nebo 80 stop. Tento údaj si uloží a po srovnání s druhem záznamu na disku (tento údaj si poznamenal na disk při jeho formátování) sám zvolí jednoduchý, nebo dvojitý krok. Dále si sám otestuje nejvyšší použitelnou rychlost krokování. Jak to všecno dělá, to si povíme později. Neodpustím si zde k tomu poznámku. Jestliže používání různých automatických funkcí je pro

laika požehnání, pro zkušenějšího je to mnohdy hrob. Dáte mi jistě zapravdu, až se dost vyvzteká-te například s kombinací Betadisku a mechaniky, která umožňuje hardwarové přepnutí na dvojitý krok. Když budete používat střídavě 40-ti a 80-ti stopý záznam, uvidíte sami. Pokud po zavolání TR-DOSU máte vyndanou disketu a po jeho přihlášení zadáte 40, nebo 80, ušetříte si mrazení v zádech při části zvuků mechaniky. Já si ve svém Betadisku vyřadím v rámci úprav i to automatické zkoušení rychlosti krokování.

TR-DOS má (moderní je teď říkat podporuje) tento formát záznamu na disk:

1. Dvojitá hustota záznamu.
2. Jednostranný i oboustranný záznam.
3. 40, nebo 80 stop.
4. 16 sektorů na stopu.
5. 256 bajtů na sektor.
6. Nepoužívá žádné přečíslení sektorů. Aby se dosáhlo urychleného čtení stopy, jsou sektory přečíslovány už při formátování (tzv. skew faktor je zde 2, tzn. že stopa bude čtena na 2 otáčky. Starší verze TR-DOSu měly tento faktor 3).

Z toho vyplývají celkové kapacity diskety 180, 360, nebo 720 KB.

Jak si vysvětlíme později, můžeme pomocí TR-DOSu jednoduše zaznamenávat a číst i sektory jiných délek, ale to už bude ve strojovém kódu. Samotný řadič pak umožňuje navíc také jednoduchou hustotu záznamu a dokonce i High density.

TR-DOS 5.xx lze využívat jak z Basicu, tak ze strojového kódu. Jeho funkci nevadí současné připojení Interface I a Microdrive - docela si s nimi rozumí. V Basicu pracuje hlavně jako rychlý magnetofon. Pro jednotlivé příkazy má TR-DOS vlastní interpreter. Následujících několik adres se vztahuje pouze k verzi 5.03. Interpreter začíná na adrese 031AH. Standardní spectrovské tokeny se nejprve převedou do interních kódů. Tabulka pro převedení na interní kódy je na 2FF3H. Tabulka adres jednotlivých rutin je na 3008H. Pro použití se Spectrem 128 je na adrese 30A9H tokenizační rutina. Její činnost si můžete vyzkoušet, když dáte příkaz např.

THEN list a pak vymažete **THEN**.
Když nyní vypsaný příkaz **list** odešlete, je napřed tokenizován a pak normálně zpracován DOSem.

Karolínka sice prý vydala kompletní český překlad návodu k Betadisku, ale přesto nemůže škodit stručný přehled basicových příkazů:

INT. KÓD	BASIC. PŘÍKAZ	ADR. V5.03	
01	CAT	0433H	Výpis katalogu disku
02	*	1018H	*"D:" přepne na disk D
03	FORMAT	1EC2H	FORMAT "název" formátuje
04	MOVE	16ABH	"setřese" obsah disku
05	NEW	053AH	NEW "nový","starý" rename
06	ERASE	0787H	ERASE "název" vymazání
07	LOAD	1815H	Jako v Basicu
08	SAVE	1AD0H	Jako v Basicu
09	RETURN	1CFBH	Návrat do Basicu
10	PEEK	19A5H	PEEK "název"adrbuf,sektor
11	POKE	19A9H	POKE "název"adrbuf,sektor
12	MERGE	19B1H	Jako v Basicu
13	RUN	1D4DH	Natáhne a spustí (i CODE)
14	OPEN#	2182H	Jako v Basicu
15	CLOSE#	2656H	Jako v Basicu
16	COPY	0690H	COPY "nový","starý"
17	4	2997H	40 nastavení na 40 stop
18	GOTO	2DA1H	GOTO "název" CODE magic
19	8	29AEH	80 nastavení na 80 stop
20	LIST	11CEH	Vypíše rozšířený katalog
21	VERIFY	1810H	Jako v Basicu

Pokud chceme formátovat jednostranně, musí začínat název disku znakem "$". Příkazy **POKE** a **PEEK** umožňují načíst, či uložit sektor daného pořadí v souboru od adresy **adrbuf**. **RUN** spouští program s autostartem od řádku autostartu, ostatní od řádku 0. Tímto příkazem je možno spustit i program ve strojovém kódu od jeho počáteční adresy. **OPEN#** a **CLOSE#** slouží k sekvenčnímu přístupu k datům. Zde je však v systému nějaká chyba - při práci s těmito příkazy se systém často hroutí, přičemž dochází mnohdy i k poškození záznamu na disku. **COPY** slouží ke kopírování z mechaniky na mechaniku. **COPY s** umožňuje kopírování na jedné mechanice. **LIST** vypisuje údaje o délkách, startovacích adresách a autostartech.

Jednotlivé příkazy je možno volat i v programových řádcích Basicu. V takovém případě musí být každý příkaz vždy jako poslední na řádku a musí být předznamenán sekvencí:

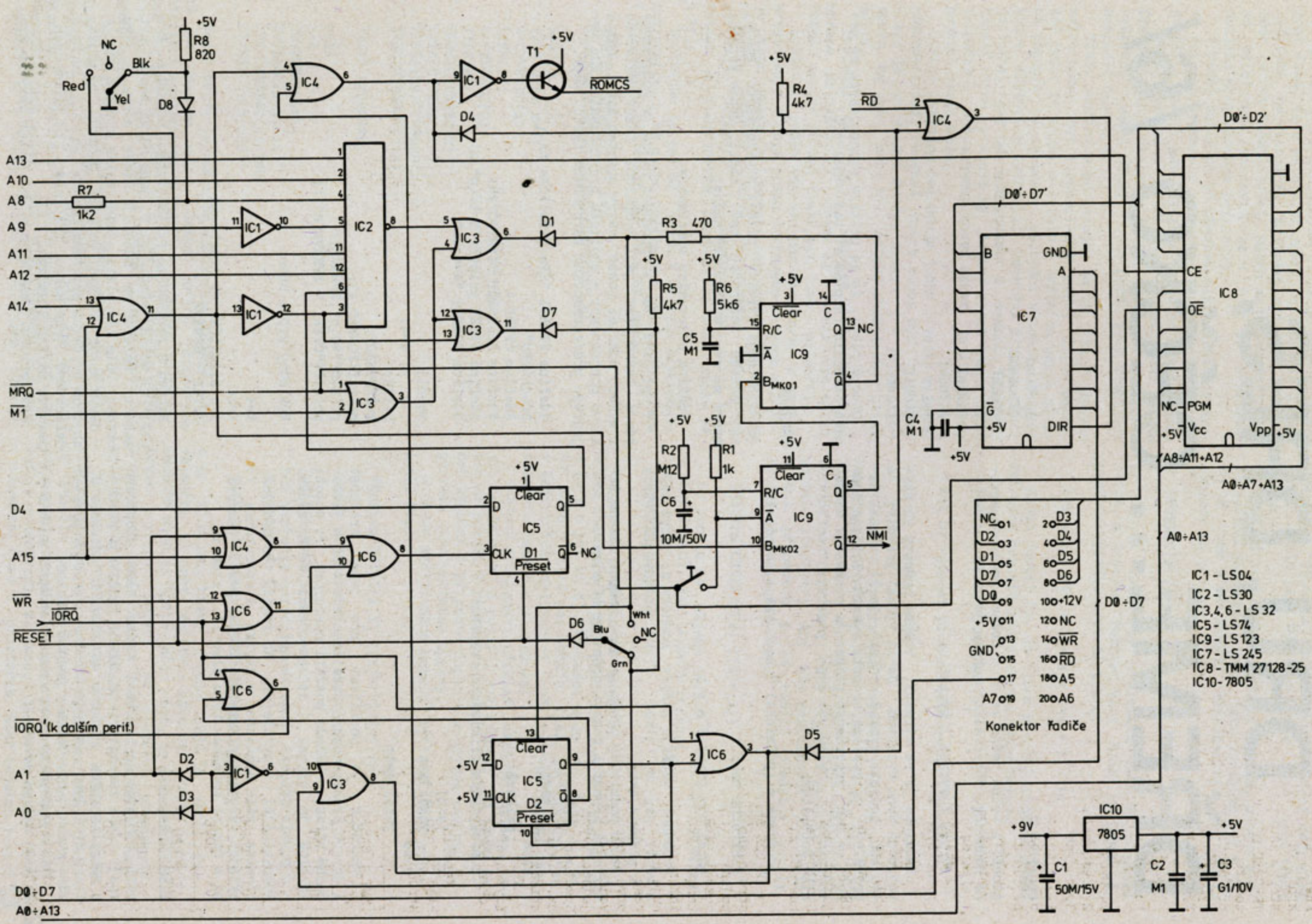
RANDOMIZE USR 15619:REM:, nebo

LET ERR = USR 15619:REM:

Druhá varianta volání nám v proměnné **ERR** vrací chybový kód:

```
 0 = žádná chyba
 1 = soubor nenalezen
 2 = soubor již existuje
 3 = není místo
 4 = katalog je plný
 5 = počet záznamů překročen
 6 = není disk
 7 = chyba disku
 8 = chyba syntaxe
 9   ---
10 = proud je již otevřen
11 = ???
12 = proud není otevřen
```

(pokračování příště)



Obr. 2 - Zapojení desky operačního systému 5.xx

DŘU, DŘEŠ, DŘEME... CÉČKO /6/

Operátor ?:

slouží k volbě jedné ze dvou hodnot podle stanovené podmínky. "Oficiální" schéma je:

```
výraz1 ? výraz2 : výraz3
```

Podmínka je ve **výrazu1**, volené hodnoty ve zbývajících dvou výrazech. Je-li **výraz1 pravdivý**, je zvolena hodnota **výraz2**, jinak (je-li **výraz1 nepravdivý**), je zvolena hodnota **výraz3**. Zvolená hodnota se pak programově přiřazuje proměnné. Mnemotechnicky si celou věc lépe pamatuji "neoficiálním" schématem:

```
proměnná = (podmínka) ? hodnota1 : hodnota2
```

Konkrétně např.:

```
většíje = (a>b) ? a : b
```

Proměnná **většíje** nabyde hodnoty **a**, když bude **a** větší než **b**, jinak získá hodnotu **b** (tedy i v případě a=b).

Tento podmíněný výběr ze dvou hodnot můžeme samozřejmě zapsat i pomocí podmínky **if-else**:

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

s převedením příkladu do zápisu:

```
if   (a>b)
     většíje=a;
else
     většíje=b;
```

Operátor **?:** použijeme pro jeho krátkost i dobrou přehlednost všude tam, kde jde o jednoduché výrazy. Výhoda použití operátoru **?:** oproti **if-else** je zřejmá i z porovnání délky zápisu - v prvním případě bude zkompilovaný strojový kód kompaktnější i rychlejší. Na rozdíl od podmínky **if-else** však nemůžeme v zápisu s operátorem **?:** skládat výrazy (příkazy) do bloku, použít vnořování a řazení podmínek typu **else-if...else-if**.

Preprocesor

není výhradní specialitou Céčka. Něco obdobného znají třeba assembleristé, kteří používají pseudo- a makroinstrukce. Oproti překladači assembleru, který bývá dvou- až tříprůchodový, probíhá překlad zdrojového textu Céčka v jednom průchodu.

Všechny příkazy preprocesoru začínají znakem **#**. Je jich celkem devět:

```
#define
#include
#undef
#if
#ifdef
#ifndef
#else
#endif
#line
```

Obecně platí, že za příkazy preprocesoru se nepíše středník (nepíše se tam vůbec nic).

#define

je nejčetněji užívaný příkaz preprocesoru. Pomocí něj můžeme definovat tzv. makra. Obsah každého makra má svůj symbolický název:

```
#define názevmakra obsahmakra
```

V případě, že preprocesor najde v programové řádce těsně před její kompilací název některého z maker, nahradí tento název obsahem makra. Odtud slovo preprocesor - anglické preprocessing zde vyjadřuje předzpracování zdrojového textu. Např.:

```
#define ZPRÁVA "Šla žížalička na \
procházku"
#define ČTYŘI 4
#define ŠESTNÁCT ČTYŘI*ČTYŘI
#define TISKX printf("X=%d\n",x)
#define FORMÁT "X=%d\n"

main()
(int x; x=4;
TISKX;
x=ŠESTNÁCT;
printf(FORMÁT,x);
printf("%s\n",ZPRÁVA);
printf("ČTYŘI - ZPRÁVA\n");}
```

Výsledný výpis bude:

```
X=4
X=16
Šla žížalička na procházku
ČTYŘI - ZPRÁVA
```

Napřed k základnímu syntaktickému pravidlu. Pro definici makra platí, že v jeho názvu nesmí být mezera. Mezerou se odděluje název makra od jeho obsahu. Dále zde platí jedna spíše zvyková regule - názvy maker se píší velkými písmeny. To proto, abychom v textu programu snadno poznali, co bylo definováno jako makro (na rozdíl od proměnných, psaných malými písmeny).

Pro definici makra je v editorech obvykle vy-

hrazena jen jedna řádka. Pro definici znakového řetězce přesahujícího řádku můžeme na její konec zařadit znak \ a pokračovat na řádce další. Zkušenost říká, že u ostatních maker není příliš dlouhá definice nejvhodnější, protože spíš než pomocí, může leccos zkomplikovat. Makra mají své rafinované záludnosti (zastavím se u nich po osvětlení dalšího programového příkladu).

V první definici je obsahem makra **ZPRÁVA** znakový řetězec, vymezený uvozovkami. Pro ilustraci je použit i znak \ pro navázání textu na další řádce.

Druhá definice přiděluje symbolickému názvu **ČTYŘI** číselný obsah 4. Všude, kde se v programu vyskytne symbol **ČTYŘI**, bude před svou kompilací nahrazen číslem 4.

Třetí definice poukazuje na to, že v obsahu makra mohou být i symbolické názvy jiných maker. Zde pozor na další pravidlo - definice makra začíná působit až od svého umístění v textu do jeho konce. Kdybychom prohodili umístění druhé a třetí definice, pak by preprocesor u symbolu **ŠESTNÁCT** hlásil chybu, protože by neznal obsah makra **ČTYŘI** (hledání a nahrazování názvů maker jejich obsahy probíhá shora dolů).

Čtvrtá definice naznačuje jednu z velkých výhod makrodefinicí. Obsahem makra mohou být i příkazy - zde je to volání funkce tisku. I v jejím argumentu by opět mohl být nějaký symbolický název makra.

Poslední definice je ukázkou toho, že makrem můžeme definovat i argumenty funkcí. Zde makro **FORMÁT** určuje formát tisku pro funkci **printf()**.

Poslední řádka výpisu nás seznamuje s dalším pravidlem - i když obsah textového řetězce obsahuje název makra, k žádnému nahrazení jeho obsahem nedojde. Textové řetězce (či znakové konstanty) zůstávají netknuty (to pochopitelně platí i ve vztahu k programovým komentářům mezi znaky **/*** a ***/**).

Podívejme se teď na jednu z velkých šikovností makrodefinic. Má svého vzdáleného příbuzného třeba v basicových příkazech **DEF FN** a **FN**. Např.:

```
#define ČTVEREC(x) x*x
#define TISK(x) printf("x=%d\n",x)

main()
{int x,z; x=4;
z=ČTVEREC(x);
TISK(z);
z=ČTVEREC(2);
TISK(z);
TISK(ČTVEREC(x));
TISK(ČTVEREC(x+2));
TISK(100/ČTVEREC(2));
TISK(ČTVEREC(++x));}
```

Výpis programu:

```
z=16
z=4
ČTVEREC(x)=16
ČTVEREC(x+2)=14
100/ČTVEREC(2)=100
ČTVEREC(++x)=30
```

Název makra je zakončen argumentem funkce, která je obsahem makra. Ještě jednou připomínám striktní syntaktické pravidlo - v názvu makra nesmí být mezera. Proto je argument v závorce připojen těsně k názvu (uvnitř závorky rovněž nesmí být žádná mezera).

Název argumentu funkce u jména makra se nevztahuje jen ke stejnojmenným argumentům. Zde může být argument **x** nahrazen jakýmkoli jiným. Jím bude nahrazena i proměnná **x** v obsahu makra. Tak např. při převádění programové řádky **TISK(ČTEVEREC(x);** bude argument **x** z druhé definice nahrazen výrazem **ČTVEREC(x)**. Ovšem to je zase makrofunkce, která si žádá řešení. Protože v ten moment je **x=4**, bude výsledkem výpočtu funkce podle první definice makra číslo 16. Výsledný výpis je v jeho třetí řádce.

Když si prohlédnete výpis programu, budete možná leckde zaskočeni, jako jsem zpočátku byl i já. To už jsme u záludností makrodefinic. Při prvním pohledu na příkaz

```
ČTVEREC(x+2);
```

by člověk řekl, že výsledkem bude 36, protože 4+2=6 a 6*6=36. Počítač však hlásí výsledek 14, protože výpočet provedl takto:

```
x*x = x + 2*x + 2 = 4 + 2*4 + 2 = 14
```

Abych dostal to, co jsem původně očekával, musel bych upravit zápis obsahu makrofunkce:

```
#define CTVEREC(x) (x)*(x)
```

Pak by výpočet proběhl očekávaným způsobem:

```
(4+2) * (4+2) = 36
```

Podobně u výpočtu **100/ČTVEREC(2)** nebude výsledkem 25 (jakožto 100/4), ale 100/2*2=100. Pro očekávaný výsledek 25 je třeba obsah makrofunkce upravit na tvar (x*x), aby výpočet proběhl podle zápisu 100/(2*2). Aby nám makrofunkce pracovala v obou případech podle našich představ, museli bychom ji definovat takto:

```
#define CTVEREC(x) ((x)*(x))
```

Do omylu se můžeme dostat i použitím prefixu v makrofunkci, jak ukažuje příkaz:

```
ČTVEREC(++x);
```

Výpočet proběhne ve sledu ++x * ++x. Tím se ovšem hodnota **x** zvýší postupně dvakrát (napřed ze 4 na 5 a pak na 6). Výsledek bude 5*6, tedy 30. To už samo o sobě nasvědčuje tomu, že prefixům se v definicích makrofunkcí radši vyhneme (pokud něco takového není naším záměrem).

Uvedený programový příklad názorně dokládá, že při definicích a použití makrofunkcí je třeba se mít na pozoru a radši si rozepsat, jak bude výpočet vypadat a případně doplnit nezbytné závorky či provést jiné potřebné úpravy.

U názvu makrofunkce nemusí být jen jeden argument. Např.:

```
#define VĚTŠÍ(X,Y) ((X)>(Y)?(X):(Y))
```

Podle něj pak bude příkaz třeba:

```
většísud=VĚTŠÍ(sud1,sud2);
```

převeden do tvaru:

```
většísud=((sud1)>(sud2))?(sud1):(sud2)
```

a my zjistíme, který z obou sudů je větší. Povšimněte si opět umístění závorek. Místo jednoduchého sudu se totiž může stát, že budeme potřebovat porovnat výrazy se spoustou proměnných a operátorů. Bez závorek v makru bychom pak nejspíš dostali chybný výsledek.

V této souvislosti se vynořuje otázka - proč vlastně stavět programovou strukturu funkcí, když by stačilo jen nadefinovat makrofunkce s jejich argumenty? Tuto otázku musí programátor řešit sám podle momentální situace. Jádro problému leží v soupeření času s prostorem. Když nějakou funkci

voláme z různých míst programu, pak je skoro vždy lepší zařadit ji sólově přímo do programu, protože po překladu bude její strojový kód v paměti jen jednou. Když totiž do všech míst volání funkce místo toho zařadíte název makra, jeho obsah se v každém takovém místě přeloží do strojového kódu, který se tak ve výsledku notně "natáhne". U počítače s hodně velkou pamětí by to nemuselo až tak vadit, spíš naopak - funkce, která nevychyluje programové řízení, proběhne rychleji, než když se napřed volá a pak se ještě čeká na návrat z jejího těla.

Na závěr povídání o příkazu **#define** jeden oblíbený céčkový bonbónek pro pascalské:

```
#define then
#define begin {
#define end   ;}
```

Céčkový program pak můžete psát syntaxí Pascalu, např.:

```
if (a=b) then
   begin
      a=0;
      b=1
   end
```

Makro **then** v těle funkce před překladem textu jednoduše zmizí, **begin** bude nahrazeno levou svorkou, **end** středníkem s pravou svorkou. A tomu už kompilátor Céčka bude rozumět.

#include

S tímto příkazem preprocesoru jsme se již setkali. Pro zopakování - v průběhu překladu nahradí preprocesor řádku

```
#include "název"
```

datovým souborem s uvedeným názvem (do programu je vložen hledaný soubor ze vstupního zařízení - disku apod.). Když je název v uvozovkách, je napřed prohledáván adresář zdrojových textů. Když tam není nalezen, hledání pokračuje ve standardní knihovně. Příkaz

```
#include <název>
```

hledá soubor jen ve standardní knihovně.

Vložený soubor je okamžitě dál překládán. I vkládané soubory mohou obsahovat příkazy **#include** (tehdy se označují jako vnořené).

Bez vkládání souborů se prakticky neobejdeme - jednak bez použití rozsáhlé standardní knihovny Céčka nic moc nevytvoříme a za druhé se nám naše zdrojové texty obvykle nevejdou do paměti najednou (a i kdyby, překlad dlouhého textu trvá nepříjemně dlouho). Při žonglování se soubory musíme dávat pozor na oblast působnosti globálních proměnných (viz Mikrobáze 3/89) a na návaznost makrodefinic.

V předpolí - aneb - Ještě jednou o ukazatelích

Než se začneme zabývat céčkovými polnostmi, vrátím se ještě na chvilku k ukazatelům, protože polní operace se jimi jen hemží. A bez plného pochopení ukazatelů s poli nic nepořídíme.

I když mám tři knížky o Céčku, na ukazatele jsem dlouho koukal, jako ta příslovečná husa do flašky. Určitě nejsem sám, komu nestačí povrchně memorovat syntaxi a kdo nepovažuje věc za pochopenou, dokud nevidí víc "dovnitř", neobjeví vnitřní souvislosti. Možná jde o assemblerový stihomam, ale porozumění operacím na té nejnižší úrovni poskytuje širší rozhled i možnosti využití dispozic jazyka.

První problém, na který jsem při hrátkách s ukazateli narazil, bylo čtení obsahu adres a zapisování do nich. Např. jsem chtěl "začernit" celou obrazovku ZX Spectra cyklickým ukládáním bajtu FFH na všechny adresy obrazové paměti (je to celkem 6144 adres od adresy 16384 nahoru). Obrátil jsem se na znalce Céčka, Petra Adámka. Rovnou z telefonu na mě vychrlil tuto funkci:

```
main()
(typedef char *p;
char *adr;
int a,c; a=16384;
for(c=0;c<6144;c++)
  (adr=cast(p)a++;
   *adr=255;)}
```

Pro mne v ní byly hned dvě nové věci - **typedef** a **cast**. Typedef se používá nejčastěji tehdy, když chceme získat ještě lepší "čtecí parametry" výpisu programu. Např. po deklaraci:

```
typedef double početkřesel;
```

můžeme kdykoli místo deklarace typu **double** (v případě, kdy jde o **početkřesel**) použít jako typ **početkřesel** místo **double**. Tím nijak nevytváříme nový typ, ale deklarací **početkřesel** dáváme proměnné typ **double**, ovšem s tím, že ji navíc máme ještě symbolicky popsanou. Např po deklaraci

```
početkřesel čalounění;
```

bude proměnná **čalounění** typu **double int**. Čili nejde o nic jiného, než o syntaktickou synonymitu. Jako začátečníci jí zatím nemusíme věnovat zvláštní pozornost, protože se krásně obejdeme bez ní. Navíc by nám někdy mohla i zamotat hlavu. Třeba po deklaraci:

```
typedef char *ukazatel;
```

a využití synonyma v další deklaraci:

```
ukazatel špekáčky;
```

nám může snadno uniknout, že **špekáčky** nejsou jen prostou proměnnou, ale ukazatelem na typ **int** (čili ***špekáčky**). Větší význam mohou mít podobné synonymické deklarace až u struktur, které jsou ještě před námi.

V našem případě se začernáním obrazovky má **typedef** trochu jiný účel a vztahuje se tu přímo k použití dalšího neznámého operátoru **cast**. V HiSoftu C ZX Spectra je přímé použití slova **cast** nestandardní. Toto slovíčko se ve standardním Céčku ve své slovní podobě vůbec nevyskytuje. Podstatné je, že se jím označuje převod jednoho typu na jiný. Ve výše uvedeném programu jde o převod proměnné **a** typu **int** do ukazatele **p** typu **char** a přidělení výsledné hodnoty ukazateli **adr** rovněž typu **char**, čímž je vyhověno potřebám typové shodnosti při přiřazování hodnot. Ve standardním Céčku by "tento cast" byl přímější:

```
adr=(char *)a++;
```

a nemuseli bychom předtím deklarovat žádný ukazatel ***p**. Takhle to ovšem HiSoft C neumí.

Proč tento převod vůbec děláme? Protože při použití ukazatele na **int** bychom pracovali se dvěma bajty (**int** je dvoubajtová hodnota), ale my potřebujeme ovlivnit jen obsah jedné adresy, tedy jednoho bajtu. A to nejde jinak, než že náš ukazatel **adr** bude ukazovat na typ **char**, který je v Céčku jediný jednobajtový.

Podívejme se teď na celou záležitost zevrubněji

a hned bude vše jasné. Když jsem nad Petrovým prográmkem bloumal, zrovna moc chytrý jsem z něj nebyl. Zkusil jsem napsat něco, co by stejně začernávalo obrazovku, i když by to pracovalo s ukazatelem na typ int:

```
main()
{int adr; adr=16384;
poke (adr);}

poke(adr)
int *adr;
{int c;
for(c=0;c<3072;c++)
  *adr=65535;}
```

Musel jsem použít subfunkci poke(adr), protože snahu o konstrukci s ukazatelem a přidělení mu hodnoty 16384 přímo ve funkci main(), kompilátor odmítl. Přenos parametru adr do subfunkce a vytvoření ukazatele až v ní, se mu konečně zalíbilo. Protože tu jde o ukazatele na typ int, tedy na dva bajty, musel jsem na ně poslat hodnotu 65535 neboli FFFFH. Jimi se v prvním průchodu cyklu for obsadily adresy 16384 a 16385. Atd. Proto je počet průchodů o polovinu menší (3072), než je celkový počet adres obrazové paměti (6144).

Obrazovka se hezky začernala, jenže spokojený jsem dvakrát nebyl. Co když budu chtít něco poslat jen na jednu adresu, nikoli na dvě sousedící? Pak ale nějak musím použít ukazatele na typ char. Zkusil jsem malou obměnu:

```
main()
{int adr; adr=16384;
poke(adr);}

poke(adr)
char *adr;
{int c;
for(c=0;c<6144;c++)
  *adr=255;}
```

Fungovalo to. Jenže teď jsem stál před hádankou, co a jak se vlastně v paměti počítače s těmito ukazateli odehrává, resp. kdo kam na co ukazuje a co kam posílá. Napsal jsem si "osvícenskou" funkci, protože osvícení mi bylo víc než třeba:

```
main()
{int adr; adr=16384;
printf("%u\n",&adr);
poke(adr);}

poke(adr)
int *adr;
{printf("%u,%u,%u,%u",&adr,&(*adr),adr,*adr);}
```

Na obrazovce se objevilo:

```
65359
65357,16384,16384,0
```

Po chvilce mi svitlo. Adresa 65359 je adresa uložení proměnné adr z funkce main(). Na adresu 65357 se uložila hodnota parametru adr přeneseného z main() do poke(). Samozřejmě jde o uložení lokálních proměnných do automatické paměti. Na obou zmíněných adresách leží hodnota 16384 (tedy přesněji - na těchto adresách a adrese o jednu vyšší; jde přece o dvoubajtový int).

&(*adr) podává informaci o tom, na jakou adresu ukazatel ukazuje, tedy obsah jaké adresy bude případně měnit nebo z ní číst. A ta se shoduje s proměnnou adr. Poslední výpis *adr informuje o obsahu této adresy (program z ní čte). Obsah adresy 16384 je tu nulový.

Kdybych místo deklarace int *adr použil deklaraci char *adr, nic by se v tomto výpisu nezměnilo, pokud by obsah adresy adr byl nadále nulový. Jenže i tak tu jeden podstatný rozdíl je - obsah adresy *adr by mi podal informaci jen o jednom bajtu na adrese adr, nikoli o dvou bajtech na adresách adr a adr+1, jako v případě předešlém. Takže při int *adr jde de facto o "dvoubajtovou nulu" 0000H, při char *adr o "jednobajtovou" 00H, což je ve výsledku stále táž dekadická nula.

To vše záleží na tom, na jaký typ ukazatel ukazuje. Zde je dobré si uvědomit něco, co mě zpočátku dost mátlo - když deklarujeme ukazatele char *adr, sama hodnota adr zůstává typu int, tedy (u malého počítače) dvoubajtová, aby dosáhla na jakoukoli adresu v rozpětí adresovacího prostoru mikroprocesoru. Typ char zde znamená jen to, že nač bude ukazatel ukazovat, bude mít rozsah jednoho bajtu. Podobně deklarace double *adr ponechá adresu standardní, ale ukazatel bude zaobcházet se čtyřmi bajty od adresy, na kterou ukazuje apod.

Abych věc rozvedl ještě více, poprvé jsem si zahrál s ukazatelem na ukazatele, který má dvouhvězdičkovou podobu. Můžeme tu hovořit o dvoustupňovém nebo dvouúrovňovém ukazateli. Svůj pátrací program jsem trošku pozměnil:

```
main()
{int adr; adr=16384;
poke(adr);}

poke(adr)
int **adr;
{printf("%u,%u,%u,%u,%u",
        &adr,*adr,&(*adr),**adr,&(**adr);}
```

Výpis:

```
65357,0,16384,45043,0
```

Konečně se mi vyjevilo sladké tajemství, o kterém jsem se v žádné knížce srozumitelně nedočetl. Adresa 65357 je nám už jasná - je to opět uložení hodnoty přeneseného parametru adr z funkce main(). Čili - na adrese 65357 a 65358 leží dva bajty, které dohromady dávají adresu 16384. Výpis *adr opět říká, že na této adrese i adrese o jednu vyšší (neb v deklaraci ukazatele je int) leží nula (16 bitů s hodnotou nula). Výpis 16384 je zase adresou, na niž ukazatel prvního stupně ukazuje.

Ale co tu dělá číslo 45043? Rychle jsem si zjistil, co obsahují adresy 0000H a 0001H ZX Spectra (v posledním prográmku změňte deklaraci adr v main() na adr=0 a přečtěte si, co říká *adr). Výsledek - 45043! Poslední číslo výpisu informuje o tom, z jaké adresy (zde nula) bude číst nebo do ní psát ukazatel druhého stupně. Přehledněji:

Na adresách 65357 a 65358 je číslo:

16384	obsah adres 16384 a 16385 = 0 *adr=0

Na adresách 16384 a 16385 je číslo:

0	obsah adres 0 a 1 = 45043 **adr=45043,

protože na adresách 0 a 1 je číslo:

45043

Tak můžeme měnit a číst nejen obsah adresy

16384, ale i adresy 0 "přes" druhý stupeň ukazatele na ukazatele. Čili **adr nás odkazuje přes obsah ukazatele adr (*adr) na adresu, která je dána tímto obsahem (dvoustupňový adresový vektor). Znalejším konstruktérům programů jistě netřeba říkat, že se tu nabízí možnost provázaného indexování, resp. odkazování do různých částí paměti s určitou skladbou dat (databázová architektura, konstrukce rozličných datových a adresových tabulek jako třeba look-up a jump tables, řazení indexů při třídění dat místo dat samotných apod.). Toto indexování je nesmírně užitečné právě při práci s poli, hlavně s těmi, které mají nestejnou délku prvků, jak je typické pro pole znakových řetězců. Pro Céčko je příznačné vytváření polí ukazatelů i polí ukazatelů na ukazatele.

Když jsem prohlédl záhady ukazatelů, pokusil jsem se obměnit funkci, kterou mi formou první pomoci poskytl Petr Adámek. Uvádím ji bez komentáře, pokuste se v ní orientovat už sami:

```
main()
{typedef char *charptr;
int adr;
for(adr=16384;adr<22528;adr++)
  *cast(charptr)adr=255;}
```

Pokud vám po prvním čtení textu není něco jasné, zapište si prográmky do počítače, měňte jejich parametry, sledujte výsledné změny a uveďte si je do souvislosti s výše řečeným. Tuto část musíte bezpodmínečně vstřebat, protože bez absolutního pochopení práce s ukazateli se v Céčku dál nedostanete. Resp. byste se odsoudili k prostému memorování syntaxe a předem se tak zbavili tvořivého přístupu k řešeným problémům. To je důvod, proč jsem se ukazateli zabýval poněkud šířeji a opět se nedostal k vlastnímu tématu polí. Ale o to lépe se nám po nich bude chodit v pokračování příštím.

-elzet-

KONVERZACE DAT A PROGRAMŮ MEZI SYSTÉMY MS DOS A CP/M

Boris Grinas

Operační systém CP/M, vytvořený v roce 1972 firmou Digital Research Corporation, se stal prvním světovým standardem operačního systému pro osobní počítače. Díky osobním počítačům Robotron 1715, TNS a dalším je tento systém široce rozšířen v CSSR. V poslední době se ve světě rozšířily osobní počítače slučitelné s IBM PC nebo PS/2. Tyto počítače používají 16-ti, nebo 32-bitové mikroprocesory firmy Intel a pracují s operačním systémem MS DOS, případně i s novým operačním systémem OS/2. Postupně se zavádějí i u nás. Tak vzniká potřeba převádět data mezi počítači se systémy MS DOS a CP/M.

Data mezi počítači můžeme převádět přes vyměnitelná média (nejčastěji pružné disky o průměru 5.25"), nebo propojením dvou počítačů kabelem přímo, případně přes modem. Problém přenosu dat však nekončí jejich zápisem na medium počítače s jiným operačním systémem. Data musí být často upravena tak, aby bylaa použitel nás programovým vybavením provozovaným pod daným operačním systémem.

Pro propojení počítačů kabelem používáme nejčastěji sériové rozhraní V.24 (podle obdobné americké normy se jmenuje RS-232C). Tímto rozhraním je vybavena většina u nás dostupných počítačů. Výměnu dat provedeme pomocí jednoho z komunikačních programů, které existují pro oba typy počítačů a operačních systémů. Jsou to například programy MoveIt, nebo Kermit. Kermit je zajímavý tím, že patří k programům, které nezištně rozšiřují americké skupiny uživatelů. Program se dodává ve formě dobře okomentovaných zdrojových textů v Assembleru, nebo v jazyku C. Kermit existuje rovněž pro počítače s architekturou DEC, nebo IBM 370/4361. V ČSSR je Kermit upraven po počítače typu IBM PC/XT/AT, Robotron 1715, Robotron A5120/30 a minipočítače řady SMEP. Pro počítače Robotron a IBM PC existují modifikace s poloduplexním provozem pro práci s modemy MDS-1200.

Jednodušší metodou převodu dat je přenos vyměnitelných médií. Osmibitové počítače dostupné v ČSSR používají obvykle pružné disky 8", nebo minidisky 5.25". Osobní počítače typu IBM PC používají minidisky 5.25" (někdy též mikrodisky 3.5"). Pro přenos dat můžeme tedy použít minidisky 5.25". Oba počítače musí mít alespoň jednu mechaniku pro čtení těchto disket. Pro konverzi dat potřebujeme program, který umí číst a psát na diskety cizího operačního systému. Znamená to, že do-mácí operační systém musí podporovat čtení a zápis fyzických bloků dat na diskety cizího operačního systému. Tento požadavek se pro osmibitové počítače realizuje velmi těžce. Naopak, šestnáctibitové počítače mají programovatelný řadič pružného disku, který je dobře popsán ve firemní dokumentaci. Nicméně konverzní programy existují jak pro osmibitové, tak i pro šestnáctibitové počítače.

Na straně počítače se systémem CP/M byl pro konverzi dat vyvinut program REFM-DOS. Program nezávisí na použité implementaci operačního systému CP/M. Jedinou podmínkou je, aby jeho BIOS dokázal číst diskety s fyzickou délkou sektoru 512B, jak je tomu u disket systému MS DOS. To umožňuje např. systém CP/A od zlepšovatelů Hudce, Vojáčka a Loefflera pro počítače Robotron 1715. Program REFM-DOS umožňuje číst adresář disku CP/M a MS DOS, zobrazovat sektory, alokační bloky a alokační tabulku disku MS DOS. Samozřejmě může provádět oboustrannou konverzi dat. Program umí sám poznat formát disku MS DOS a pracuje s disky MS DOS, formátovanými na 180K, 360K a 720K. Program lze používat jak konverzačním, tak i dávkovým způsobem. V dávkovém způsobu program pracuje s parametry na příkazovém řádku, které definují směr konverze a jména převáděných souborů. Výhodou tohoto programu je to, že pracuje v prostředí CP/M a zapisuje na diskety přímo ve formátu používaného počítače. Logické a fyzické formáty disku nebyly v systému CP/M (na rozdíl od MS DOS) standardizovány, takže volba správného formátu je často u konverzních programů problémem.

Na straně počítačů se systémem MS DOS existuje pro konverzi několik programů, z nichž se v ČSSR nejvíce rozšířil program XenoCopy. Tento program zvládá širokou škálu logických a fyzických formátů disku CP/M a poskytuje uživateli velké pohodlí při práci. Zůstává problémem, že musíme přesně vědět

nejen počet stop a počet sektorů na stopě svého disku CP/M, ale také znát i logickou organizaci disku, např. kolik je rezervovaných stop, kolik sektorů zabírá adresář disku a délku alokačního bloku.

Jak při konverzi na počítačích se systémem MS DOS, tak i na osmibitových počítačích, vznikají při konverzi dat soubory o délce, která je celým násobkem 128. To proto, že systém CP/M definuje délku souboru s přesností jednoho logického sektoru, kdežto MS DOS s přesností jednoho bytu. Proto se při konverzi do formátu MS DOS zkonvertují soubory správně, kdežto zpětná konverze přidá k délce souboru MS DOS několik bytů navíc. Pro většinu aplikací toto prodloužení nevadí, záleží na dalším použití zkonvertovaných souborů. Vyzkoušel jsem konverzi souborů *.COM a *.EXE do CP/M a zpět. Zůstaly provozuschopny pod systémem MS DOS. Prodloužení určitě nevadí textovým souborům, jejichž logický konec je označen znakem 1AH (^Z).

Ve většině případů konverze dat nekončí pouhým převedením souboru do jiného operačního systému. Podíváme se podrobněji na některé konkrétní případy konverze dat a programů do systému MS DOS.

Převod programu

Obecné soubory *.COM systému CP/M nejsou použitelné na 16-bitových počítačích. V raných etapách vývoje systému MS DOS existoval v MS DOSu emulátor systému CP/M, který vytvářel pro *.COM soubory systému CP/M domácí prostředí. Má mnoho omezení, a proto na tento prostředek zapomeneme. Programy převádíme tak, že převedeme jejich zdrojové texty, tyto texty upravíme, zkompilujeme a odladíme program znovu na 16-bitovém stroji.

DBASE II <=> DBASE III Plus

Firma Ashton Tate dodává konverzní program dCONVERT pro konverzi souboru *.DBF, *.MEM, *.FRM, *.FMT a *.CMD z DBASE II do DBASE III+. Program má jednoduchou obsluhu a může se používat jak konverzačním, tak i dávkovým způsobem. Soubory *.DBF, *.MEM, *.FRM a *.FMT konvertuje rovnou tak, že staré soubory přejmenuje na *.DBB, *.MEB atd. Soubory *.CMD musíme nejdříve přejmenovat na *.PRG. Od programových souborů *.PRG se nedělají záložní kopie. Tyto soubory jsou po konverzi proloženy řádkami komentářů s označením nejednoznačných, nebo nedovolených konstrukcí. Změny provedené konverzním programem jsou rovněž označeny. dCONVERT nepřevádí soubory *.NDX, indexy vytvoříme pod DBASE III znovu. Zpětná konverze z DBASE III do DBASE II je možná pouze pro datové soubory *.DBF. Příklad konverze:

1. Zkonvertujeme všechny soubory celé aplikace do formátu MS DOS jedním z výše popsaných konverzních programů a zapíšeme je do adresáře C:\dbase\prg

2. Pomocí programu XTREE přejmenujeme všechny soubory *.CMD na *.PRG

3. Spustíme program dCONVERT dávkovým způsobem:

```
C:\dbase\prg>dconvert *.*
```

Předpokládejme, že program dCONVERT je nalezen. Začne se konverze a na obrazovce se objevují jména konvertovaných souborů. U souboru *.PRG se vypisuje i počet řádků, které potřebují dodatečný zásah programátora. Musím upozornit na to, že dCONVERT nepřevede celý program bezchybně. Jsou konstrukce, které jsou syntakticky správné v DBASE II i v DBASE III, ale chovají se v každém prostředí jinak. Proto převedená aplikace musí být pro DBASE III dodatečně otestována a upravena.

Turbo Pascal 3.X <=> Turbo Pascal 4.0 a 5.0

Zde je situace obdobná. Firma Borland International dodává konverzní program UPGRADE. Ten dosadí do převedených zdrojových textů klauzuli USES tak, aby se vytvořilo prostředí slučitelné s Turbo 3. Na požádání program Upgrade umí rozdělit celý zdrojový text programu na moduly. Na rozdíl od DBASE II a DBASE III může program v Pascalu po konverzi pracovat hned bez závad. I zde však není situace úplně ideální. Za prvé - program Upgrade špatně zpracovává deklarace FORWARD. Za druhé - pro dobré využití všech možností nové verze Pascalu musí programátor program podstatně změnit. Konečně - mezi osmi a šestnáctibitovými stroji jsou podstatné rozdíly a v dobrém programu musí být použity i jiné přístupy k tvorbě programového vybavení. Proto prakticky ve většině případů musíme programy sestavit znovu v nové kvalitě a ze starých k tomu využijeme pouze odladěné kousky zdrojových textů. Nepotřebujeme k tomu ani program UPGRADE. Příjemné je, že firma BORLAND dodává k novému Turbo Pascalu (poslední verze ohlášená v říjnu 1988 je 5.0) systém Turbo Access, který je stoprocentně slučitelný s populárními toolboxy pro osmibitové počítače. Pro převod dat, vytvořených Pascalským programem pod systémem CP/M, do formátu šestnáctibitové implementace, použijeme ten poznatek, že soubory CP/M implementace mají oproti implementaci MS DOS na začátku 4 byty navíc. Uděláme krátký program, který kopíruje původní soubor větu po větě do jiného souboru a vypustí 4 byty na začátku. Data pořízená systémem Turbo Access se konvertují obdobně. Použijeme k tomu samotný systém Turbo Access a nezapomeneme přitom, že čísla vět u nové implementace jsou 4 bytová. Indexní soubory nejsou slučitelné, protože se používá jiná, rychlejší odrůda stromu (B+ stromy) a musí být vytvořeny znovu.

WordStar a jiné textové procesory

Texty psané ve WordStaru jsou plně přenositelné. Horší je to s jinými textovými procesory, které nejsou známé na osmibitových počítačích. Některé z nich však umí pracovat se soubory ve formátu WordStaru, jiné dovedou tyto soubory do svého formátu překonvertovat. Například WordPerfect umí zkonvertovat soubor ve formátu WordStar do svého formátu a naopak svoje soubory převést do WordStaru. V každém případě většina textových procesorů umí pracovat s čistými ASCII soubory.

Pozn. red. - Přenos dat mezi CP/M a MS DOS, včetně příslušného překódování, umožňují také programy VFORM a KODING, které dodává 602. ZO Svazarmu jako součást programového vybavení pro počítače PC/XT/AT. Při přípravě rukopisů do Mikrobáze oba tyto programy bohatě využíváme. Podrobnosti o těchto a dalších programech najdete v některém z příštích čísel našeho zpravodaje.

●●●

Následující konstrukce není už tak docela žhavou novinkou. Její autoři se totiž dlouho zdráhali ji uveřejnit. Po velkém úspěchu již druhé konstrukce obdobného mikropočítače na stránkách Amatérského radia, nakonec přece jen popis konstrukce dodali. Nepředpokládáme, že si po uveřejnění bude každý hned počítač stavět. Věříme však, že přímo ukázkově propracovaná konstrukce (autoři jsou profesionálové), spolu se srozumitelným popisem funkce, vás v mnohém poučí a inspiruje. Upozorňujeme, že nejde jen o kopii ZX Spectra, ale o zcela nový počítač, který se jako Spectrum chová. Dovede však také ledacos navíc. Zvláštní pozornost si zaslouží zobrazovací část, na jejímž principu připravují její autoři pro Mikrobázi novou konstrukci. Bude to doplněk pro ZX Spectrum, který umožní dvojnásobnou hustotu grafiky, tj. 512x256 bodů. Připojí se mezi obvod ULA a jeho patici. Jaké to přinese výhody, například při CP/M, kde bude možno zobrazit plných 80 zn. na řádek, to snad ani nemusíme zdůrazňovat.

BOBO 64 K (1)

Ing. Josef Blahůt

ing. Václav Daněček

BOBO 64k není mikropočítač pro profesionální použití. Je to hračka, kterou jsme navrhli jako variantu k - u nás tak oblíbenému - mikropočítači ZX Spectrum 48k. BOBO je, použijeme-li paměť EPROM se shodným obsahem, s tímto mikropočítačem plně programově slučitelný a navíc má několik drobných vylepšení. Na konektor "4K" je možné připojit externí klávesnici, nebo ovladač her Spectrum. Adresy pro paralelní rozhraní jsou voleny tak, aby se brána A (konektor "3K") chovala jako ovladač typu Kempston. Všechny programy lze s použitím signálu WAIT libovolně zpomalovat, protože obnovování paměti RAM neprovádí mikroprocesor. Paměť EPROM překrývá 16k RAM, do kterých lze kdykoliv zapisovat a po RESETu zůstává jejich obsah zachován. Můžeme tam např. nahrát kopírovací program (nemůže volat podprogramy z paměti EPROM), který bude mít po odpojení EPROM k dispozici více jak 50k volné paměti RAM.

Mikropočítač BOBO nemusí pracovat jen jako Spectrum. Pro tento účel jsme přidali sériové a paralelní rozhraní (tiskárna, sériový styk) a 3 další režimy zobrazování. Tak je možné používat mikropočítač i pod systémem CP/M. Při připojování dalších periferií přes systémové konektory "1K" a "2K" (pružné disky, RAM disk) je třeba dodržet zásadu, aby každý výstup byl zatížen maximálně jedním vstupem LS.

BOBO 64k není vybaven kodérem PAL. Výstup VHF je pouze černobílý, v osmi stupních šedi. Pro barevné zobrazení je nutné používat barevný monitor, popřípadě upravit barevný TV přijímač. Úprava BTV Oravan je v příloze číslo 7. Takto je možné upravit prakticky každý z moderních československých BTV.

1. Základní vlastnosti počítače

Mikropočítač byl navržen jako jednodeskový, na desce o rozměrech (285 x 250 mm). Je řízen mikroprocesorem Z80A s hodinovou frekvencí 3,5 MHz. Paměť RAM má kapacitu 64 kbyte a je společná pro procesor i obvody zobrazení. Paměť ROM má kapacitu 16 kbyte a překrývá spodní část RAM.

Mikropočítač je připojitelný k libovolnému TV přijímači, případně k černobílému nebo barevnému monitoru. Dále obsahuje jednoduché obvody pro připojení běžného magnetofonu, který slouží k záznamu dat a programů. Pro připojení dalších periferií je na desce paralelní a seriové rozhraní.

1.1 Specifikace mikropočítače

Procesor:	- Z80A CPU řízený frekvencí 3,5 MHz, aresovatelný prostor 64 kbyte, délka slova 8 bitů, počet instrukcí 158
Paměť RAM:	- 64 kbyte společná pro procesor i zobrazení s délkou slova 8 bitů
Paměť ROM:	- 16 kbyte, překrývá spodní část paměti RAM. Programově odpojitelná.
Zobrazení:	3 způsoby zobrazení - rozlišení 256x192 bodů, 8 barev, každé skupině 8x8 bodů jsou přiřazeny 2 barvy. Kapacita obrazové paměti je cca 7 kbyte. Programově lze obrazovou paměť umístit buď na konec paměti RAM, nebo bezprostředně paměť ROM 16k. - rozlišení 256x256 bodů, 8 barev, každé skupině 8-mi následujících bodů jsou přiřazeny 2 barvy. Kapacita obrazové paměti 16 kbyte. Je umístěna na konci paměti RAM. - rozlišení 512x256 bodů, 1 barva na černém pozadí. Kapacita obrazové paměti 16 kbyte. Umístěna na konci paměti RAM.
Vstupy a výstupy:	- Vstup a výstup pro přenos dat na magnetofon - vstup dat z klávesnice - sériový vstup a výstup s obvodem MHB 8251 - paralelní vstup a výstup s obvodem MHB 8255A - zvukový výstup na reproduktor - výstup pro RGB monitor - výstup úplného televizního signálu (video) - výstup televizního signálu VHF
Přerušení:	- maskovatelné přerušení každých 20 ms - nemaskovatelné přerušení při příjmu znaku ze sériové linky
Napájení:	- 5V, odběr cca 1A

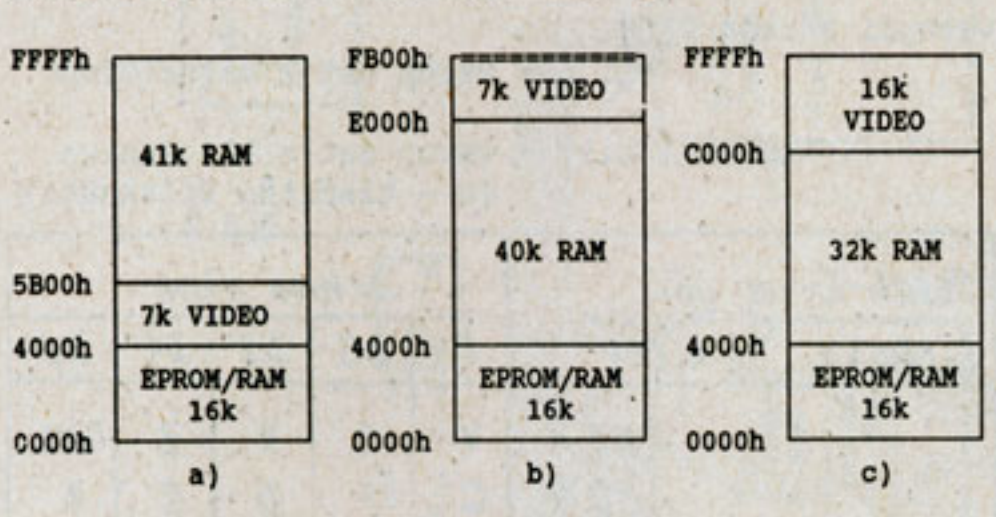
Po zapnutí mikropočítače nebo stisknutí tlačítka RESET se připojí paměť ROM, zvolí se zobrazovací režim 256x192 bodů a obrazová paměť se připojí za paměť ROM.

1.2 Paměť

Celý adresový prostor paměti je rozdělen na tři bloky. Blok paměti ROM, blok paměti RAM pro data a programy a blok paměti RAM pro videoprocesor. Paměť ROM je umístěna na lokacích 0-3FFFH, kde blokuje paměť RAM. Paměť RAM je však nepřístupná pouze při čtení, zápis do této oblasti je možný vždy, a po odpojení paměti ROM jsou data k dispozici. Odpojení paměti ROM se provádí instrukcí OUT na adresu 0FCH. Nastavení bitu 7 na této adrese do jedničky (80H) odpojí paměť ROM. Připojení se provede po vynulování bitu 7, což je možné provést buď programově nebo stisknutím tlačítka RESET.

Umístění bloku paměti pro videoprocesor závisí na zvoleném typu zobrazení. Při zobrazení 256x192 bodů může být obrazová paměť umístěna na lokacích 4000H až 5AFFH, nebo E000H až FAFFH. Při zobrazení 256x256, nebo 512x256 bodů je obrazová paměť umístěna na lokacích C000H až FFFFH. Řídící příkazy pro přemisťování obrazové paměti jsou uvedeny v následující části.

Po zapnutí mikropočítače, nebo stisknutí tlačítka RESET, je obrazová paměť umístěna na lokacích 4000H až 5AFFH. Možnosti rozdělení adresového prostoru paměti znázorňuje obr. 1.



Obr. 1 - rozdělení adresového prostoru paměti

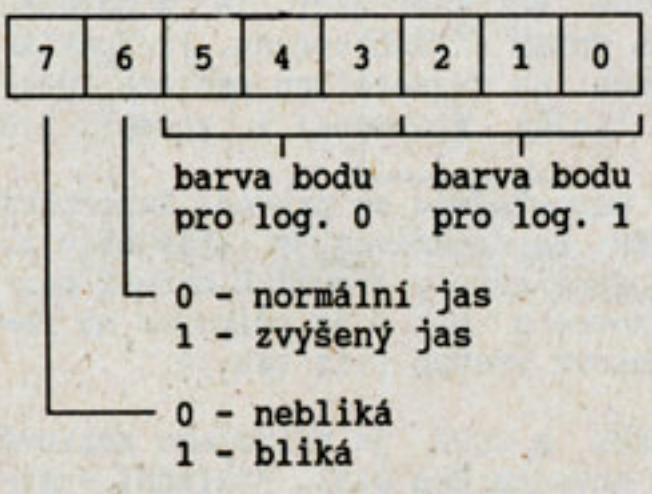
1.3 Režimy zobrazení

Obrazová paměť je společná pro procesor i videoprocesor. Do ní jsou programem zapsána data, která čte videoprocesor a zobrazuje je na obrazovce.

V obrazové paměti odpovídá každému bodu na stínítku obrazovky 1 bit. Paměť je obrazovým procesorem cyklicky čtena a data jsou vysílána v sériovém tvaru na obrazovku. Vždy se přečte najednou informace o osmi po sobě následujících bodech, tj. 1 bajt paměti a vyšle se na výstup. Jako první vystupuje na obrazovku bit 7 (je tedy zobrazen vlevo), jako poslední bit 0. V režimech 256x192 a 256x256 bodů je současně čten ještě druhý bajt (tzv. atribut), ve kterém je obsažena informace o barvě každého bodu. Proto je obrazová paměť rozdělena na dvě části. V první je uložena informace o stavu každého bodu na obrazovce (0 nebo 1), ve druhé jeho barva.

Protože pracovní plocha žádného ze zobrazovacích režimů nepokrývá celou plochu obrazovky, je možno okolí pracovní plochy (pozadí obrazu) obarvit některou z osmi barev. Změnu pozadí provádí instrukce OUT na adresu 0FEH. Barvu určují bity 0-2 výstupního bajtu a její kódování je stejné jako u atributu v tab.2.

Ve všech režimech je nultý sloupec nultého řádku umístěn na obrazovce vlevo nahoře.



Tab. 2 - tvar atributu

1.3.1 Zobrazení 256x192 bodů

V tomto režimu je zobrazováno celkem 256x192=49152 bodů. V paměti jsou uloženy po osmi, takže obsadí 6144 bajtů. V druhé části obrazové paměti jsou uloženy atributy. Každý je společný pro skupinu 8x8 bodů a tvoří ho 1 bajt paměti. Celkem tedy zabírají 32x24=768 bajtů. To znamená, že celá obrazová paměť obsadí 6912 bajtů. Tento počet se zdá být rozumným kompromisem mezi velikostí obsazené paměti RAM a možnostmi grafiky tohoto režimu. Obrazová paměť RAM je po zapnutí umístěna těsně za paměť ROM, tj. od adresy 4000H. Pro některé aplikace je vhodné umístit obrazovou paměť až na konec paměti RAM, tj. od adresy E000H. Toto přepínání provádí instrukce OUT na adresu 0FCH. Nastavení bitu 6 na této adrese do jedničky (40H) připojí obrazovou paměť na konec paměti RAM a vyslání 00H přesune obrazovou paměť za paměť ROM.

Jeden řádek na obrazovce (256 bodů) reprezentuje 32 bajtů v paměti. Jednotlivé řádky však nejsou v paměti uspořádány za sebou. Jejich umístění znázorňuje tab.1. Dosazením čísla řádku (v binárním tvaru) za R0-R7 dostaneme počáteční adresu řádku v paměti RAM. Připočtením čísla sloupce (S0-S4) získáme adresu některé z 32 osmic v jednom řádku (32 sloupců x 8 bodů = 256 bodů na řádek). Adresu atributu získáme dosazením čísla řádku za R3-R7 a čísla sloupce za S0-S4. Symboly R0-R2 se v adrese nevyskytují, protože atribut je stejný pro čtvereček 8x8 bodů. Toto, na první pohled složité, rozmístění obrazových řádků v paměti je z hlediska programování výhodné, uvažujeme-li zobrazení jednoho ASCII znaku v rastru 8x8 bodů. Pak bude na obrazovce umístěno 24 řádků po 32 znacích a pro jeden znak vystačíme s jediným atributem. Počáteční adresa obrazové paměti (označuje místo, kde bude umístěn vrchol znaku) může být v registrech HL a vlastní tvar znaku bude uložen v osmi bajtech v paměti ROM (s počáteční adresou v registrech DE). Zobrazení znaku pak provedeme jeho postupným čtením z paměti ROM (s inkrementem DE) a vysláním každého bajtu do obrazové paměti s následným inkrementem registru H. Počáteční adresa znaku v obrazové paměti musí být násobkem osmi. Znak může být tedy umístěn pouze na řádcích 0-7, 8-15, 16-23 atd. Při pokusu použít tento algoritmus k zobrazení znaku, např. na řádcích 5-12, dojde k rozdělené znaku na dvě části (5-7 a 8-12) a každá z nich bude zobrazena v jiném místě obrazovky (viz tab.1).

Typ zobrazení		ADRESA PAMĚTI															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
256 x 192 od adr. 4000H	body	0	1	0	R7	R6	R5	R4	R3	S4	S3	S2	S1	S0	R2	R1	R0
	atribut	0	1	0	1	1	0	R7	R6	S4	S3	S2	S1	S0	R2	R1	R0
256 x 192 od adr. E000H	body	1	1	1	R7	R6	R5	R4	R3	S4	S3	S2	S1	S0	R2	R1	R0
	atribut	1	1	1	1	1	0	R7	R6	S4	S3	S2	S1	S0	R2	R1	R0
256 x 256 od adr. C000H	body	1	1	0	R7	R6	R5	R4	R3	S4	S3	S2	S1	S0	R2	R1	R0
	atribut	1	1	1	R7	R6	R5	R4	R3	S4	S3	S2	S1	S0	R2	R1	R0
512 x 256 od adr. C000H	body 1	1	1	0	R7	R6	R5	R4	R3	S4	S3	S2	S1	S0	R2	R1	R0
	body 2	1	1	1	R7	R6	R5	R4	R3	S4	S3	S2	S1	S0	R2	R1	R0

R0 - R7 : vyjádření čísla řádku v binárním tvaru
S0 - S4 : vyjádření čísla sloupce v binárním tvaru

Tab. 1 - umístění obrazových řádků v paměti RAM

Atribut je rozdělen na 4 části. V bitech 0-2 je informace o barvě bodů (ve čtverečku 8x8), které jsou uložené v obrazové paměti jako 1 (inkoust); v bitech 3-5 je barva bodů uložených v paměti jako 0 (papír). Barvy příslušející jednotlivým kombinacím bitů jsou v tab. 2. Jednička v bitu 6 značí

zvýšení jasu příslušného čtverečku a jednička z bitu 7 označuje jeho blikání s frekvencí 1,6 Hz.

1.3.2 Zobrazení 256x256 a 512x512 bodů

V režimu 256x256 je zobrazováno 65 536 bodů. První část obrazové paměti zabírá tedy 8 192 bajtů. V druhé části jsou uloženy atributy a každému bajtu z první části náleží jeden. Celková kapacita obrazové paměti je tedy 16 384 bajtů. Je umístěna od adresy C000H. Každý atribut nese informaci o barvě, jasu a blikání osmi bodů. Tvar atributu je stejný jako v režimu 256x192 bodů. I umístění jednotlivých řádků v paměti je obdobné a znázorňuje ho tab.1.

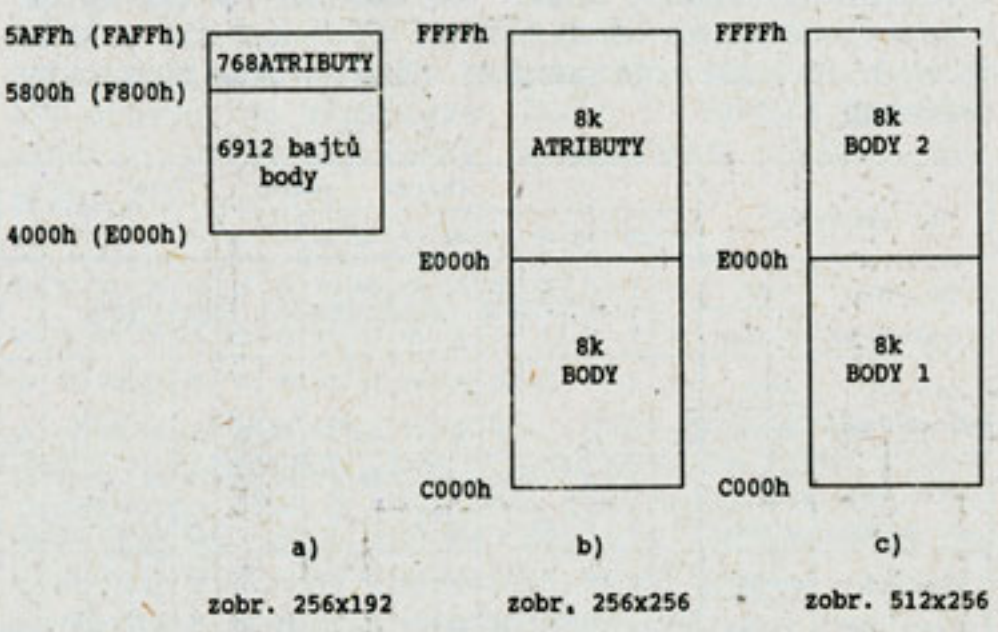
Přepnutí na zobrazení 256x256 bodů provádí opět instrukce OUT na adresu 0FCH, a to vysláním 20H (nastavení bitu 5 do jedničky).

I když grafické možnosti tohoto režimu jsou bohatší, neumožňují zobrazit 64 resp. 80 ASCII znaků na jednom řádku. Proto je možno použít ještě třetí typ zobrazení s rozlišením 512x256 bodů. Jednomu řádku je v obrazové paměti vyhrazeno 64 bajtů (64x8=512 bodů). Těchto 64 bajtů je rozděleno na 2x32 a každá polovina řádku je uložena v jiné polovině paměti, jak je uvedeno v tab.1. Do této tabulky dosazujeme za S0-S4 čísla 0-31. Ovšem každý sloupec není složen z osmi bodů (jako v předcházejících režimech), ale z šestnácti (32 sloupců x 16 bodů = 512 bodů). Při zobrazování se přečte nejprve bajt z adresy VIDEO1, pak z adresy VIDEO2 a v tomto pořadí se zobrazí v příslušném sloupci. To znamená, že při zobrazování každého řádku je každá osmice bodů čtena z jiné poloviny obrazové paměti.

Režim 512x256 bodů nepoužívá atributy (na jejich místě je VIDEO2). Obrazová paměť má kapacitu 16 384 bajtů a je umístěna od adresy C000H. Bodu, který je uložen v obrazové paměti jako 1, odpovídá černá barva a bodu uloženému jako 0 barva pozadí, určená instrukcí OUT na adresu 0FEH.

Přepnutí na zobrazení 512x256 bodů provádí instrukce OUT na adresu 0FCH, a to vysláním 60H (nastavení bitů 5 a 6 do jedničky).

Rozdělení obrazové paměti v různých režimech znázorňuje obr. 2.



Obr. 2 - rozdělení obrazové paměti RAM

1.4 Vstupy a výstupy

Mikropočítač je ovládán z klávesnice, která může být složena až z 56 tlačítek. Jsou uspořádána v matici 8 řad x 7 tlačítek. Při čtení dat z klávesnice se využívá skutečnosti, že při instrukci IN A, (C) vysílá procesor Z80 na adresové vodiče A8 až A15 data z registru B. Klávesnice se čtę instrukcí IN z adresy 0FEH a před jejím provedením se zapíše nula na jeden z osmi bitů registru B. Po provedení vstupní instrukce bude v registru A uložen stav jedné z osmi řad tlačítek. Vstupní instrukci provedeme 8 krát a současně vystřídáme nulu na všech bitech registru B (nula je vždy jen na jednom bitu registru B, ostatní bity musí zůstat v jedničce). Tak přečteme všech 8 řad tlačítek. Je-li některé z nich stisknuto, nastaví se příslušný bit registru A do nuly. Příklad přiřazení znaků pro klávesnici OWERTY s maticí 8 řad x 5 tlačítek je v tab.3

Tab. 3 - vstupní a výstupní adresy

vstupní adresa **0FEH**

bit 6 - vstup dat z magnetofonu

ostatní bity - vstup dat z klávesnice (0 - tlačítko stisknuto)

Obsah B-reg při instruke IN A,(C)	Datové slovo				
	D4	D3	D2	D1	Dø
FE H	V	C	X	Z	CAPS
FD H	G	F	D	S	A
FB H	T	R	E	W	Q
F7 H	5	4	3	2	1
EF H	6	7	8	9	ø
DF H	Y	U	I	O	P
BF H	H	J	K	L	ENT.
7F H	B	N	M	S.SH.	SPACE

Příklad obsazení klávesnice s maticí 8x5 tlačítek

Výstupni adresa **0FEH**

bit 4 - zvukový výstup
bit 3 - výstup dat na magnetofon
bity 0÷2 - barva pozadí (kódování jako v tab. 2)

Výstupní adresa **0FCH**

bit 7 - 0 -> EPROM připojena
1 -> EPROM odpojena

bit 6 5

0 0 - zobr. 256 x 192 bodů od adr. 4000H
0 1 - zobr. 256 x 256 bodů od adr. C000H
1 0 - zobr. 256 x 192 bodů od adr. E000H
1 1 - zobr. 512 x 256 bodů od adr. C000H

Paralelní rozhraní

adresa 7FH - řídící slovo
1FH - data brána A
3FH - data brána B
5FH - data brána C

Sériové rozhraní

adresa 0FDH - stavové slovo
0DDH - přenos dat

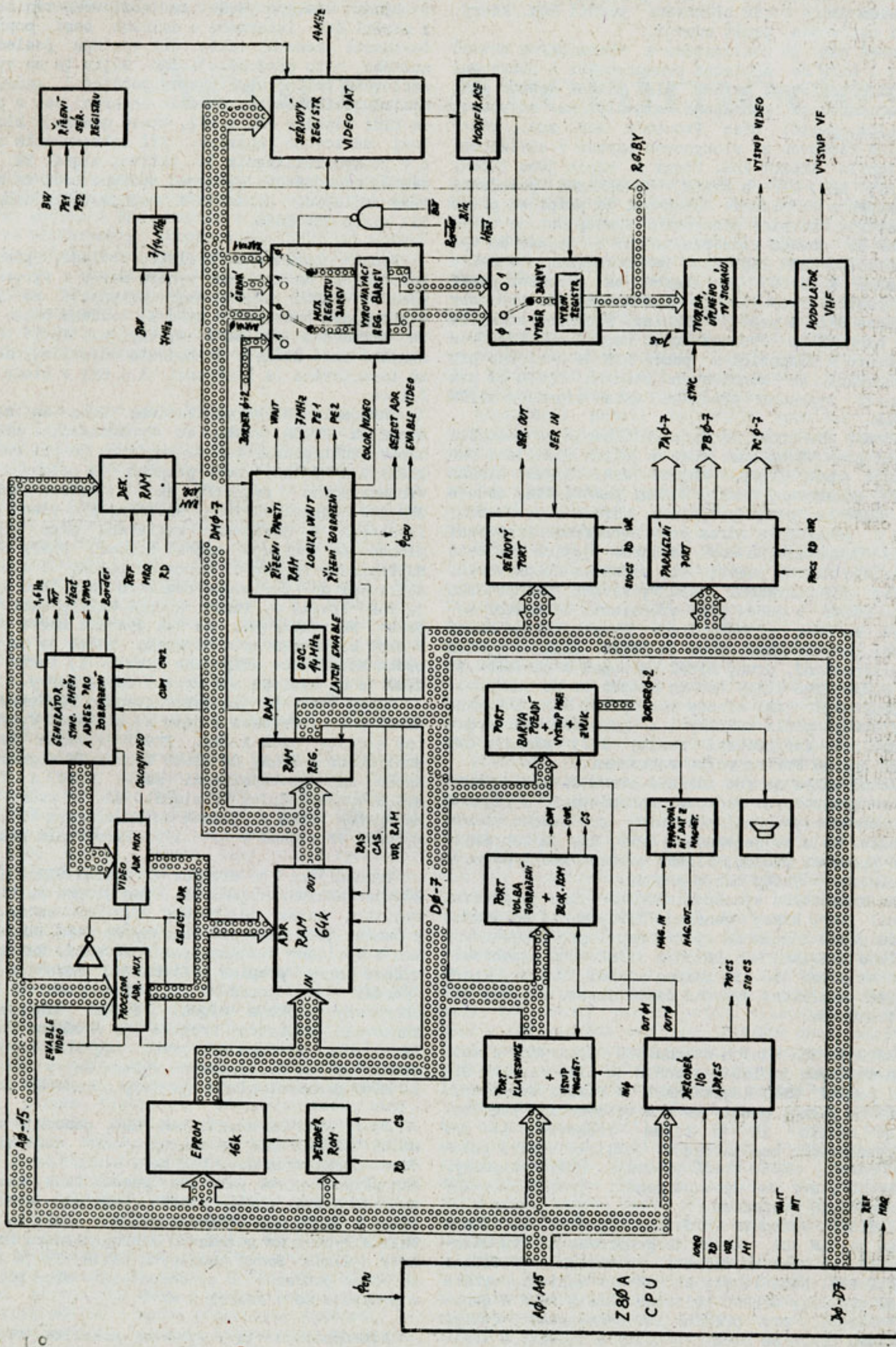
Kromě klávesnice se instrukcí IN s adresou 0FEH čte na bitu 6 vstup z magnetofonu. Předpokládá se, že data jsou na magnetofonu nahrána jako pulsy se střídou 1:1 a frekvencí v rozmezí 0,5 - 5 kHz.

Výstup dat do magnetofonu se provádí instrukcí OUT s adresou 0FEH. Zaznamenáván je stav bitu 3. Bity 0-2 této výstupní adresy slouží k určení barvy pozadí na obrazovce a bit 4 je připojen na reproduktor jako zvukový výstup (viz tab.3).

Na adresách 0FDH a 0DDH je připojeno sériové rozhraní, tvořené obvodem MHB 8251. Základní vysílací a přijímací frekvence je 76,8 kHz. To představuje, při konstantě 64, přenosovou rychlost

1200 bitů za sekundu. Adresa 0FDH slouží pro čtení stavu a zápis řídicího slova; adresa 0DDH je určena pro přenos dat. Sériový styk je realizován proudovou smyčkou 20 mA. Po příjmu každého znaku se generuje nemaskovatelné přerušení.

K ovládání paralelních periferií je mikropočítač vybaven třemi výstupními bránami, které jsou realizované obvodem MHB 8255A. K jeho ovládání slouží čtyři adresy. Zápis řídicího slova se provádí na adresu 7FH. Pro bránu A je určena adresa 1FH, pro bránu B adresa 3FH a brána C je ovládána adresou 5FH. Všechny tři brány jsou připojeny přímo na výstupní konektor.



Obr. 3 - blokové schema

2. Popis technického řešení

Na obr. 3 je blokové schema mikropočítače. Jeho základem je mikroprocesor Z80A. K němu je připojeno 16k paměti EPROM a 64k RAM. Dekodér ROM umísťuje paměť EPROM na začátek paměťového prostoru. Je ovládán signály RD a MRQ z procesoru a řídicím signálem CS z Portu blokování paměti ROM, který, je-li aktivován, paměť odpojí.

Paměť RAM je realizována z dynamických obvodů 64k x 1 bit. Je společná pro procesor i zobrazení a její přidělování ovládá blok Řízení paměti RAM. Z procesoru se přenášejí signály na adresovou sběrnici paměti přes Procesor adr. mux., který přepíná řádkovou a sloupcovou adresu v závislosti na signálu SELECT ADR. Čtení i zápis jsou řízeny signály RAS, CAS a WRRAM, na základě požadavků procesoru a zobrazení. Vstup dat je připojen přímo na datovou sběrnici procesoru. Výstup dat je veden jednak do obvodů videoprocesoru, a jednak do Vyrovnávacího RAM registru. Tento registr je nutný vzhledem k době, po kterou má procesor paměť k dispozici. Tato doba je tak krátká, že procesor nestihne data z paměti přečíst. Proto se signálem RAM nahrají do vyrovnávacího registru, kde jsou procesoru k dispozici a paměť RAM je uvolněna pro zobrazování. Na konci svého čtecího cyklu si pak procesor vyzvedne přečtená data signálem LATCH ENABLE.

Adresy pro obrazová data přicházejí na sběrnici paměti přes Video adr. mux. a jejich rozdělení zajišťuje opět signál SELECT ADR. Signál ENABLE VIDEO, generovaný obvody Řízení paměti RAM, určuje zda bude na paměť připojena adresa z procesoru, nebo z generátoru adres pro zobrazení. Přepínání adres jednotlivých bodů a adres jejich atributů zajišťuje Řízení paměti RAM signálem COLOR/VIDEO. Adresy pro zobrazení jsou vytvářeny Generátorem sync. směsi a adres pro zobrazení. Generátor sestává ze dvou čítačů, které synchronizuje obvod Řízení paměti RAM. První je čítač řádků s cyklem 64 mikrosekund, druhý čítač sloupců s periodou 29 ms. Jednotlivé stavy těchto čítačů se po zakódování používají jako adresy pro paměť RAM. Kódování se provádí podle tabulky 1 a přepínání adres pro jednotlivé zobrazovací režimy řídí signály CW1 a CW2 určené Portem volby zobrazení.

Kromě adres se (na základě stavu čítačů) vytváří v generátoru řádkové zatemňovací a řádkové a snímkové synchronizační pulsy pro tvorbu videosignálu. Dále se generuje v době, kdy je zobrazení mimo pracovní plochu, signál Border, který připojí na obrazový výstup barvu pozadí.

Ze snímkového synchronizačního pulsu je odvozen signál INT, který generuje každých 20 ms žádost o přerušení a může se využít např. pro programovou obsluhu klávesnice. Dělením snímkového synchronizačního pulsu 16-ti vznikne signál 1.6 Hz, který zajišťuje blikání vybrané části obrazu v rytmu této frekvence.

Zdrojem základních ovládacích frekvencí a dalších časových řídicích signálů je blok nazvaný Řízení paměti RAM, logika WAIT a řízení zobrazení. Tvoří ho čítač, paměť PROM a vyrovnávací registr. Čítač modulo 16 je řízen frekvencí 14 MHz z krystalového oscilátoru a cyklicky vybírá polovinu paměti PROM, která generuje řídicí signály. Hazardní stavy na výstupu paměti jsou filtrovány vyrovnávacím registrem.

Jediným vstupním řídicím signálem do tohoto bloku je RAM ADR, který je generován blokem Dekodér RAM na základě žádosti procesoru o přidělené paměti RAM. Není-li signál RAM ADR aktivní, vybírá čítač cyklicky druhou polovinu paměti PROM a generuje signály pro probíhající zobrazovací cykly. V každém cyklu se čtou dva bajty z paměti. V režimech 256x192 a 256x256 bodů se první bajt nahraje do sériového registru video dat, druhý bajt (atribut) do Vyrovnávacího registru barev a 8 takto přečtených bodů je sériově, frekvencí 7 MHz, vysláno na obrazovku. V režimu 512x256 bodů jsou oba přečtené bajty nahrány do sériového registru video dat a jako 16 bodů vyslány frekvencí 14 MHz na obrazovku.

Během výstupu bodu ze sériového registru se z paměti čte informace o dalších osmi, popřípadě šestnácti bodech, takže při výstupu posledního, nultého, bitu předcházejícího bajtu je na výstupu sériového registru připraven počáteční, sedmý, bit následujícího bajtu. Okamžik nahrání dat z paměti se řídí signály PE1 a PE2, které generuje blok Řízení sériového registru. Při zobrazovacím režimu s v ysokým rozlišením je aktivní signál BW. Tento signál zdvojnásobí rychlost výstupu bodů ze sériového registru, takže je v jednom řádku zobrazeno 512 místo 256 bodů.

Výstup Sériového registru ovládá, přes blok Modifikace, Multiplexer výběru barvy s vyrovnávací pamětí. Tento multiplexer připojuje na výstup barvu podle hodnoty (0 nebo 1) každého bitu, který ze sériového registru vystupuje. Má-li některá skupina bodů blikat, je hodnota odpovídajících bitů invertována (s frekvencí 1,6 Hz) v bloku Modifikace.

Informace o barvě každého bodu se nahrává signálem COLOR VIDEO do Vyrovnávacího registru barev. Multiplexer tohoto registru je při režimech 256x192 a 256x256 bodů přepnut do polohy 0 a do vyrovnávacího registru se z datové sběrnice DM0-DM7 nahrává atribut, společný pro všech 8 bodů zapsaných v Sériovém registru. Mimo pracovní plochu se aktivuje signál Border, který přepíná Multiplexer registru barev do polohy 1 a blokuje nahrávání do Sériového registru. Na jeho výstupu je pak trvalá 0, která přepne Multiplexer výběru barev. Na výstup RGB se tak dostává barva pozadí. V době řádkového zatemňovacího pulsu je na výstup bloku Modifikace připojena trvalá 1, takže v této době je na výstupu RGB úroveň černé barvy.

Kromě barvy se do Vyrovnávacího registru barev nahrává i informace o jasu a blikání. Vytváří se tak signály Blik a Jas. První je veden do bloku Modifikace a druhý do bloku Tvorba TV signálu, kde spolu se synchronizační směsí SYNC a signály R,G,B vytváří úplný TV signál. Ten je veden jednak na výstup VIDEO a jednak do Modulátoru VHF, který vytváří VF signál v I. - III. televizním pásmu.

Trochu jiná je situace v režimu 512x256 bodů, kdy je aktivní signál BW. Ten přepne Multiplexer registru barev do polohy 1. Oba bajty čtené z paměti jsou postupně nahrány do Sériového registru a vysílány dvojnásobnou frekvencí. Multiplexer výběru barvy přepíná mezi barvou pozadí a černou (pozadí = 0, černá = 1). Mimo pracovní plochu obrazovky blokuje signál Border nahrávání do Sériového registru. Protože je současně aktivní BW, je na výstupu Sériového registru trvalá 1, která přepne Multiplexer výběru barvy do polohy 1. Mimo pracovní plochu je tedy obrazovka černá.

Je-li aktivní signál RAM ADR, generuje se okamžitě signál WAIT a mikroprocesor čeká na konec čtení obrazových dat. Pak se v bloku Řízení paměti RAM připojí první polovina paměti PROM, kam je do zobrazovacích cyklů vložen paměťový cyklus pro čtení a zápis dat mikroprocesorem. Zruší se signál WAIT a vyhoví se požadavku mikroprocesoru na paměťový cyklus. Počet vložených taktů TW je závislý na době potřebné k synchronizaci obou procesorů a pohybuje se v mezích 0 až 3.

Mikropočítač je vybaven několika vstupními a výstupními zařízeními. Všechny výběrové adresy

se dekódují v bloku Dekodér IO adres. Pro řízení vnitřních funkcí slouží Port volby zobrazení a blokování paměti ROM. Bity 5 a 6 tohoto registru generují řídící signály CW1 a CW2 pro určení režimu zobrazení a bit 7 odpojuje paměť ROM.

Barvu pozadí určují bity 0-2 bloku Port barvy pozadí. Součástí tohoto registru jsou ještě dvě jednotlivé brány. Bit 3 ovládá výstup na magnetofon a bit 4 je připojen na zvukový výstup. Vstup magnetofonu je spolu s klávesnicí čten v bloku Port klávesnice a magnetofonu. Magnetofon je připojen na bit 6, ostatní bity slouží pro klávesnici. Poslední dva bloky tvoří sériové a paralelní rozhraní s obvody MHB 8251 a MHB 8255A.

(pokračování)

Acorn Archimedes

Radek Solar

Archimedes je zatím posledním produktem britské firmy ACORN, která se proslavila v Anglii již dříve svými počítači BBC, Master a Compact.

Je to ultrarychlý stoprocentně 32bitový osobní počítač, založený na nové technologii RISC, jehož srdcem je speciální RISC - procesor od Acornu-Arm. Ve své cenové kategorii drží absolutní primát v rychlosti, grafice a zvuku. Podle časopisu CHIP dosahuje výkonu workstations a srovnatelné počítače jsou na trhu za cenu až přes 20 000 DM.

V současné době se volně prodávají tři modely Archimeda: 305, 310 a 440. Acorn inzeruje ještě model 410, který však není ve skladech. Nejskromnějším je 305, který má 512 KB RAM. Následuje nejrozšířenější model 310 s 1MB RAM, nejlepší 440 má 4MB RAM a 20MB harddisk. Všechny modely jsou standardně dodávány s jedním vestavěným 3,5" drivem, barevným RGB monitorem, myší, klávesnicí totožnou s počítači PC-AT, sériovým (RS423) i paralelním (Centronics) rozhraním. Všechny mají ROM 512KB s operačním systémem Arthur 1.2 a mnoha užitečnými moduly včetně Basicu, desktopu, zvukového syntetizeru, mnoha znakových generátorů (včetně znaků české i slovenské abecedy), překladače assembleru a mnoha dalších. Model 440 (a i 410) mají navíc zabudovány tzv. "module backplane" pro připojování "modulů" (karet). K ostatním modelům se musí přikoupit zvlášť.

Hlavní předností Archimeda je určitě obrovská rychlost, kterou ilustruje např. úplný (čistě softwarový) emulátor IBM PC-XT s CGA kartou, na němž PCTOOLS ukazuje rychlost 200% proti IBM. Mezi triumfy patří také vynikající grafika (1280 x 976 bodu, 4096 barev), která díky vysoké rychlosti umožňuje bezkonkurenční animaci a zvuk (8 nezávislých stereofonních kanálů, každý může hrát libovolné nasamplované sekvence).

Velmi příjemně působí, že většina toho, co uživatel k práci potřebuje, je již v ROM. Pokud je počítač nastaven (Archimedes si pamatuje mnoho informací i při odpojení od sítě), přivítá nás při zapojení dobře udělaný desktop (něco jako GEM na Atari ST), který velmi usnadňuje práci s disky a navíc obsahuje různé kalkulačky, zápisníky a kalendáře. Dále ROM obsahuje několik desítek dalších modulů a aplikací pro práci s grafikou, zvukem, písmem, tiskárnou, sítí Archimedu, okny, sprity a mnoho dalších. Za zmínku stojí aspoň interpretační BASIC V, který podle CHIPu ruší všechny dosavadní záporné představy o Basicu. Je pravda, že s klasickým Basicem má společného už velmi málo. Jeho rychlost a obrovské možnosti způsobují, že mu většina ostatních počítačů nemůže konkurovat ani ve strojovém kódu. BASIC V obsahuje i úplný překladač assembleru ARM - procesoru, takže příkazy Basicu se mohou libovolně kombinovat s částmi strojového kódu. Samozřejmostí je celoobrazovkový editor s mnoha pohotovými funkcemi.

Přes vynikající Basic je Archimedes přece jen více využit ze strojového kódu. V podobě ARM - procesoru se programátorovi dostává do ruky nástroj neslýchaných možností. Není třeba připomínat závratnou rychlost systému RISC, který systémem pipelining zpracovává najednou vždy 3 (32bitové) instrukce. Jeho instrukční soubor je oproti očekávání (RISC = Reduced Instruction SET Computer - Počítač s omezeným instrukčním souborem) velmi výkonný a efektivní. Základních instrukcí je 44 a většina má několik desítek obměn, když se ovšem za obměnu nepočítá použití jiných registrů. ARM má 27 32bitových registrů, každý kromě PC je úplně univerzální a lze ho použít všude.

Organizaci paměti zajišťuje speciální memory controller chip(MEMC), který rozdělí fyzickou paměť do částí a ty promítá na uživatelem zvolené adresy celkového adresovaného prostoru 64 MB. To značně zjednodušuje práci s pamětí, protože všechny části začínají na pevných adresách, bez ohledu na svou proměnnou délku.

Uživatel má k dispozici také blok bateriemi zálohované CMOS paměti, která se nevymaže ani při odpojení od sítě. Obsahuje všechny údaje o konfiguraci počítače, grafickém režimu, tiskárně, informaci o tom, co se má spustit při zapnutí počítače, kalendář, hodiny a další užitečné věci. Zbytek může využít pro své účely.

Archimedes má patici pro připojení matematického Floatingpoint koprocesoru, pokud však není připojen, simuluje ho OS emulátorem. S Archimedem je dodáván zdarma také emulátor mikroprocesoru 6502.

Přestože již nyní existuje pro Archimedes mnoho vynikajícího softwaru, mnoho lidí asi potěší softwarový PC-XT emulátor, který sice zatím simuluje jen CGA (ne EGA), ale zato je velmi spolehlivý. Dosud nebyl zjištěn program, který by na něm nechodil. Koncem roku 1988 by se měl začít prodávat pro Archimedes operační systém UNIX a již se pracuje na implementaci OS/2. Z jazyků jsou nyní k dispozici např. Pascal, C, Fortran, Logo.

Z hardwarových doplňků firmy nabízejí různé přídavné 3,5" a 5,25" drajvy, buffery pro tiskárny, 8b i 16b zvukový sampler, digitizer obrazu, zařízení pro mix obrazu z videa s grafikou Archimeda, I/O obvody pro připojení vlastních digitálních i analogových zařízení, RAM/ROM desky umožňující vsazení vlastních pamětí s programy a další zařízení, např. multisync-monitory, které umožní lépe využít grafických možností počítače.

Přes svou bezkonkurenčnost není dosud Archimedes na kontinentě příliš rozšířen. Vinu na tom má podivná obchodní politika Acornu, který se snaží systém prodávat hlavně do školství (jako dříve BBC). Očekává se však, že nejlepší má Archimedes ještě před sebou. Ještě dříve než UNIX a OS/2 se má objevit nový Arthur 2 s výborně udělaným multitaskingem a mnoha vylepšeními.

Přestože Archimedes není nejlevnější, v poměru k výkonu je jeho cena více než dobrá. Cenově je totiž na úrovni COMMODORE AMIGA 2000, která se mu ovšem svými schopnostmi nemůže rovnat. Archimedes 310 stojí pod 1000 Lstg, model 1440 se 4MB RAM a harddiskem se prodává zhruba za 2400 Lstg.

IBM PC
z pohledu programátora

/3/

RNDr. Januš Drózd

3. ASSEMBLER

Programátor v assembleru si musí uvědomit, že plná kontrola nad segmentovými registry mu neposkytuje libovůli v manipulaci s jejich obsahem. Chce-li totiž v instrukci odkazovat na místo v paměti, pak druh instrukce (při odkazech do kódu) nebo i možnost použít instrukci (při odkazech do dat) závisejí na běhové hodnotě v segmentových registrech. Protože tedy generovaný kód závisí na hodnotách při běhu, nezbývá, než při překladu oznámit assembleru, že při běhu na tom a tom místě programu budou v segmentových registrech adresy těch a těch segmentů.

Abychom mohli stanovit způsob přístupu k paměťovým objektům, musíme kód a data rozdělit do segmentů. Všechny objekty definované v jednom segmentu mají stejnou segmentovou adresu. Efektivní adresa se počítá od začátku segmentu od nuly. Typický program v assembleru má tuto strukturu (vyhrazená slova jsou tučně):

```
DATA  SEGMENT word public 'DATA'; začátek sg. DATA
      EXTRN   ... defin. externích dat ze sg. DATA
ALFA  DW      1A5DH       ; definice dat v sg. DATA
DATA  ENDS                ; konec sg. DATA

TEXT  SEGMENT word public 'CODE'; začátek sg. TEXT
      EXTRN   ...defin. ext. návěští ze sg. TEXT
START LABEL   far         ; vchod do programu
      ASSUME  CS:TEXT, SS:STACK  ; zajistil systém
      MOV     AX,DATA     ; (*)
      MOV     DS,AX       ; do DS sg. adresa DATA
      ASSUME  DS:DATA     ; informace o obsahu DS
  ... vlastní kód programu .....................
TEXT  ENDS                ; konec segmentu TEXT

STACK SEGMENT para public 'STACK'; zač. sg.STACK
      DW      100 dup(?) ; rez. místa pro stack
STACK ENDS                ; konec sg. STACK
      END     START  ; startovací adresa programu
```

Pseudoinstrukce **SEGMENT** a **ENDS** vymezují hranice segmentu. Před nimi je uvedeno **jméno segmentu**. Návěští START odpovídá efektivní adresa 0 v segmentu TEXT, proměná ALFA je na efektivní adrese 0 v segmentu DATA.

Pseudoinstrukce:
ASSUME segmentový_registr : jméno_segmentu
sděluje assembleru, že v uvedeném segm. registru bude při běhu segm. adresa uvedeného segmentu. Assembler pak při generování kódu instrukce např.:
MOV BX, ALFA
ví, že pro přístup k proměnné ALFA ze segmentu DATA může použít registr DS.

Obsah segm. registru, ohlášený pseudoinstrukcí ASSUME, platí až do ohlášení jiného obsahu stejného registru nebo do výskytu pseudoinstrukce:
ASSUME segmentový_registr : NOTHING

ASSUME je tedy obdobou pseudoinstrukce USING na IBM 360. Na rozdíl od ní však ASSUME lze použít pouze pro segmentové registry, a nikoliv pro bázové nebo indexové registry.

Při spouštění programu je do CS:IP zavedena vzdálená adresa START, do SS segmentová adresa segmentu STACK a do SP efektivní adresa konce segmentu STACK (neboli velikost segmentu) - zásobník totiž roste dolů. V instrukci označené (*) zaváděč DOSu 2 upraví hodnotu segmentové adresové konstanty podle zaváděcí adresy segmentu DATA.

Pomocí pseudoinstrukce
grupa GROUP segment, segment, ...
lze zřetězit uvedené segmenty do jediného segmentu. Všechny pak budou mít společnou segm. adresu "grupa" a od ní se bude počítat efektivní adresa. Celková délka spojených segmentů však nesmí přesáhnout 64 KB.

Symboly označující adresy dat nesou trojí informaci:

- segment, v němž jsou data definována;
- efektivní adresu v tomto segmentu;
- typ dat (byte, slovo, dvojslovo, ...).

Při přístupu k datům označeným symbolem se segment, v němž jsou data definována, hledá mezi ohlášenými obsahy segmentových registrů (pomocí v místě výskytu instrukce platných pseudoinstrukcí ASSUME). Nenajde-li se, je ohlášena chyba. Jinak se generuje instrukce přistupující k paměti přes určený segm. registr.

Mnemonika mnoha instrukcí je navržena tak, že k vygenerování správného kódu je třeba znát i typ dat. Proto v instrukci nestačí uvést adresu, s níž se má pracovat, nýbrž buď symbol, který má svůj typ i segment, nebo explicitní údaj o typu a segmentu. Např. v instrukci:
INC WORD PTR DS:[2D6H]
zápis WORD PTR sděluje, že to, co se má inkrementovat na efektivní adrese 2D6H v segmentu zpřístupněném registrem DS, je slovo.

Externí symboly označující adresy dat se definují pseudoinstrukcí:
EXTRN symbol : typ, symbol : typ, ...
Tento zápis poskytuje asembleru potřebnou informaci o typu symbolu, hodnotu efektivní adresy symbolu doplní linker. Schází však jméno segmentu, na něž se v původním návrhu patrně zapomnělo. Proto byl zaveden předpoklad, že je-li pseudoinstrukce EXTRN umístěna v nějakém segmentu, jsou jí zavedené symboly deklarovány ve stejném segmentu. Je-li EXTRN mimo segmenty, nelze počítat s přítomností segm. adresy symbolu v segm. registru a je třeba

ji explicitně zavést např. instrukcemi (do segm. registru nelze přímo zavést konstantu):

```
MOV     AX, SEGMENT symbol
MOV     ES, AX
```

Návěští, tj. symboly označující místa v kódu, mají jednu zvláštnost. Na rozdíl od symbolů označujících data obsahují v sobě informaci (vedle segmentu a efektivní adresy), zda jde o blízkou nebo vzdálenou adresu. Díky tomu např. instrukce:

```
JMP     NAVESTI
```

nevyžaduje žádnou dodatečnou specifikaci, zda se má generovat skok na blízkou nebo vzdálenou adresu. Assembler pak ale nemůže volit blízkou nebo vzdálenou adresu automaticky podle toho, zda skok vede přes hranici segmentu. Místo v programu, na něž vedou odkazy ze stejného i z jiných segmentů, je nutno označit dvěma návěštími.

Externí návěští definujeme pseudoinstrukcí:

EXTRN label : vzdal, label : vzdal, ...

kde **"vzdal"** je FAR nebo NEAR a určuje, zda jde o vzdálenou nebo blízkou adresu. Pro blízké adresy je tento popis nedostatečný, a tak stejně jako u dat záleží na umístění pseudoinstrukce dovnitř správného segmentu.

Kód v assemblerském programu je rozdělen do procedur vymezených pseudoinstrukcemi:

jméno_procedury PROC vzdal a
jméno_procedury ENDP.

Pojem procedura by však neměl navozovat představu samostatného podprogramu s jedním vchodem a jedním východem, jako ve vyšších PJ. Procedura má pouze tyto dvě funkce:

- definuje návěští **jméno_procedury** jako blízké, nebo vzdálené, podle obsahu **vzdal**;
- zajišťuje, že všechny instrukce RET použité v proceduře budou odpovídat **vzdal**.

Návěští lze definovat tak, že je umístíme před instrukci a oddělíme dvojtečkou. Pak půjde o **blízké** návěští. Univerzálnější způsob poskytuje pseudoinstrukce:

návěští LABEL vzdal.

O dalších atributech, které se vyskytly v ukázkovém programu v assembleru, se zmíníme v části 4.5 a na začátku 5. kapitoly.

Od assembleru se očekává, že pokud nezpracovává makra, měl by být nejvýše dvouprůchodový. Nešťastná architektura 8086/88 způsobuje, že kód lze generovat ve dvou průchodech pouze za cenu jistých omezení. Problém je v tom, že v případě dopředných odkazů (ať už na návěští nebo na data) závisí nejen obsah, ale i počet bytů generovaného kódu na segmentu, do něhož symbol patří. Proto assembler, chce-li v prvním průchodu přidělit návěštím efektivní adresy, musí u návěští uhodnout, zda je blízké nebo vzdálené (hádá blízké) a u dat, ve kterém segmentu jsou definována. Hádá-li špatně, dojde k chybě ohlášené jako "Phase error between passes" a programátor má smůlu. Proto se v assembleru doporučuje:

- definovat všechna data před prvním použitím;
- dopředné odkazy na vzdálená návěští označit ve zdrojovém kódu atributem FAR.

Podobná chyba, způsobená špatným umístěním pseudoinstrukce EXTRN, bude ohlášena až při linkování a hledá se ještě hůř.

Nejrozšířenějším assemblerem na PC je MASM firmy Microsoft. Jeho manuály jsou natolik nepřesné, že je mnohdy těžké odlišit, co je chybou a co kuriozitou assembleru. Jeho kvalitu ilustrují níže uvedené příklady.

Instrukce SCAS slouží k hledání hodnoty v řetězci. V manuálu je explicitně uvedeno, že to, zda instrukce hodnotu našla, poznáme z nulového resp. nenulového obsahu registru CX. Není to pravda, CX bude nulové jak v případě neúspěšného hledání, tak v případě nalezení hodnoty na poslední možné adrese.

Začínající programátor zpravidla neví, jak assembleru sdělit, zda instrukce RET označuje blízký nebo vzdálený návrat. Zkusí obvykle napsat:

```
RET FAR ; obdoba správného JMP FAR XX,
```

což asembler akceptuje, ale vygeneruje jinou instrukci.

Až do čtvrté verze byl popis struktury segmentů v MASMu dosti složitý a operátor OFFSET (vracející efektivní adresu symbolu) byl navržen nešťastně. Ve verzi 5 byl zaveden alternativní, zjednodušený popis segmentů a při té příležitosti byl změněn význam operátoru OFFSET. Aby byla zachována slučitelnost s předchozími verzemi, bylo stanoveno, že nový význam OFFSET platí pouze tehdy, když se používají zjednodušené popisy segmentů. Celá potíž je v tom, že nové popisy segmentů nemají vyjadřovací sílu starých. Proto je nutno kombinovat nové i staré. Programátor pak neví, kde mu platí nový a kde starý význam OFFSET, a v manuálu řešení nenajde.

Direktivou

```
.RADIX 16
```

lze změnit implicitní číselnou soustavu na hexadecimální. Nicméně asembler pak vyhodnotí konstanty 49D a 10B jako (dekadicky) 49 a 2, protože hexadecimální číslice D a B bude považovat za sufixy měnící číselnou soustavu.

Je-li symbol definován pseudoinstrukcí:

```
ALFA EQU  BETA
```

a zveřejníme-li jeho hodnotu pseudoinstrukcí:

```
PUBLIC ALFA,
```

pak assembler bude pseudoinstrukci PUBLIC ignorovat. Experimentálně zjištěná "správná" definice ALFA je:

```
ALFA EQU  BETA + 0
```

při níž se chyba neprojeví.

Naštěstí jiné SW produkty firmy Microsoft mají v průměru lepší kvalitu. Jinak by bylo v zájmu programátorů vytvořit databázi chyb SW nástrojů.

4. VYŠŠÍ PROGRAMOVACÍ JAZYKY

Při prvním seznámení s architekturou procesoru 8086/88 vzniká dojem, že je pro implementaci vyšších PJ velmi vhodná. Podporuje např. vytváření aktivačních záznamů na zásobníku a snadný přístup k lokálním datům. Ve skutečnosti se však architektura Intel opírá o tuto koncepci:

"Umožnit snadnou a efektivní práci s malými objekty; s velkými objekty pracovat jinak a složitěji."

Myšlenka, že se jinak pracuje s malými daty a programy a jinak s velkými je však v přímém rozporu s idejemi vyšších PJ.

Implementace vyšších PJ na PC mají řadu společných rysů. Povšimneme si nejběžnějších překladačů: MS Pascalu, MS C, Turbo Pascalu, Turbo C a vyjímečně také MS Fortranu a MS Basicu. Všechny informace o těchto systémech rychle zastarávají, a tak je možné, že v blízké době budou dostupné novější verze těchto produktů, než se podařilo získat autorovi příspěvku, a některé zde uvedené údaje už nebudou platné.

4.1 Základní datové typy

V celočíselných typech je situace přehledná:

délka	znam.	jazyk C	MS Pascal	Turbo Pascal
4 byty	ano	signed long	INTEGER4	LONGINT
4 byty	ne	unsigned long	-	-
2 byty	ano	int	INTEGER	INTEGER
2 byty	ne	unsigned int	WORD	WORD
1 byte	ano	signed char	SINT	SHORTINT
1 byte	ne	unsigned char	BYTE	BYTE

S typem real je jedna pricipiální potíž. Procesory řady 80x86 nemají instrukce pro provádění operací nad čísly v pohyblivé čárce. Reálné operace místo nich provádějí koprocesory 80x87. Avšak zájemců o počítání s reálnými čísly je podstatně více než těch, kteří jsou ochotni investovat do (dosti drahého) koprocesoru. Pak nezbývá, než provádět operace nad reálnými čísly programově. Popíšeme, jak těmto uživatelům vyšel vstříc Turbo Pascal.

Tento překladač může pracovat ve dvou režimech. Ve standardním má uživatel k dispozici 6-bytový typ real a knihovnu podprogramů, které nad nim (nepříliš rychle) provádějí aritmetické operace. V rozšířeném režimu jsou navíc definovány typy single (4 byty), double (8 bytů) a extended (10 bytů). Operace rozšířeného režimu může provádět buď koprocesor, nebo podprogramy emulační knihovny. Zvolíme-li emulační knihovnu, přeložený program ji bude obsahovat, ale použije ji, jen pokud koprocesor není připojen.

Drobnou nevýhodou je, že v emulačním režimu je zpracování typu real méně efektivní, než zpracování ostatních tří typů. Koprocesor ani emulační programy neumějí pracovat přímo s typem real, a tak ho převádějí na extended.

V MS Pascalu je typ real 4 nebo 8-bytový (real, real8), stejně jako v jazyku C (float, double).

Oba Pascaly používají pro reprezentaci hodnot výčtových typů 1 nebo 2 byty podle mohutnosti typu (do 256 hodnot 1 byte). V Turbo C jsou v souladu s normou všechny konstanty definované výčtem typu int.

Typ ukazatel je v MS Pascalu reprezentován 2-bytovou blízkou adresou, v Turbo Pascalu 4-bytovou vzdálenou adresou (!). V C jsou k dispozici blízké i vzdálené ukazatele a jejich použití jednak závisí na paměťovém modelu, jednak může být ovlivněno programátorem. Blíže o tom v dalších kapitolách.

4.2 Standardní aktivační záznam

Jak se ve vyšších programovacích jazycích přiděluje paměť proměnným? To závisí na povaze jazyka. V jazycích jako Fortran nebo Basic se každé proměnné přidělí úsek paměti, jehož velikost je určena typem proměnné. Toto uspořádání je velmi jednoduché, ale má dvě nevýhody:

- V průběhu celého výpočtu je paměť přidělena všem proměnným. Tím vzrůstají celkové nároky programu na paměť.
- Nelze používat rekurzivní volání podprogramů.

Jazyky algolského typu, tedy např. Pascal, C, Simula, Modula, ADA, dělí proměnné na globální a lokální. Globální proměnné jsou deklarovány v hlavním programu, lokální v podprogramech. Globální proměnné existují po celou dobu provádění programu, zatímco lokální proměnné vzniknou při zavolání podprogramu a zaniknou při jeho dokončení.

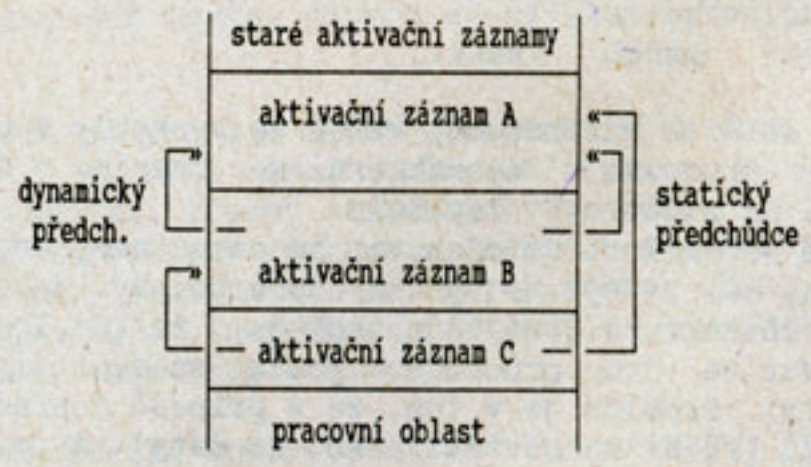
Implementace mechanismu přidělování paměti lokálním proměnným vychází ze skutečnosti, že proměnná, která dřív vznikla, později zanikne. Proto je přirozené vymezovat prostor pro lokální proměnné (alokovat proměnné) na zásobníku.

Při zavolání podprogramu se zmenší hodnota ukazatele zásobníku tak, že na vrcholu zásobníku vznikne volná oblast potřebné velikosti pro tzv. **aktivační záznam**. V aktivačním záznamu budou uloženy:

- hodnoty lokálních proměnných podprogramu;
- hodnoty pomocných proměnných, které vygeneroval překladač;
- hodnoty parametrů podprogramu.

Aby se po skončení podprogramu výpočet mohl vrátit k předchozímu aktivačnímu záznamu, je v AZ uložen také ukazatel na něj. Říkáme mu ukazatel na **dynamického předchůdce**, protože ukazuje na AZ podprogramu, který byl dříve zavolán. Kromě toho AZ podprogramu X musí obsahovat také ukazatel na AZ podprogramu, v němž je X deklarován, obsažen. Tomu říkáme ukazatel na **statického předchůdce**. Výjimkou je ovšem jakyk C, v němž podprogramy nemohou být obsaženy v jiných podprogramech.

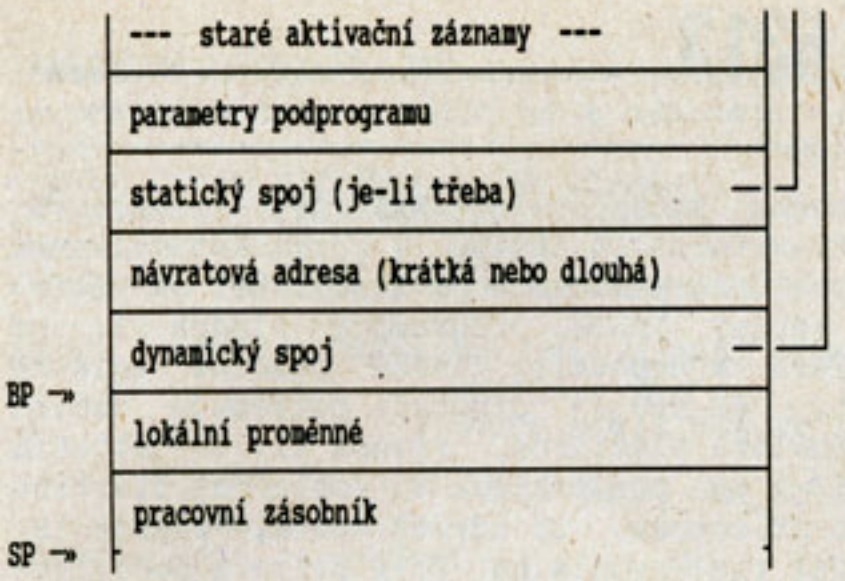
Příklad: Předpokládejme, že v podprogramu A v Pascalu jsou deklarovány podprogramy B a C. Když A zavolá B a B zavolá C, vypadají aktivační záznamy na zásobníku takto:



Všechny autorovi známé implementace vyšších PJ ukládají aktivační záznamy na HW zásobníku, na jehož aktuální vrchol v segmentu SS ukazuje registr SP. Velikost HW zásobníku je omezena velikostí segmentu, tj. 64 KB. Takto jsou omezeny paměťové nároky lokálních objektů v současně aktivních podprogramech i hloubka rekurze.

Registr SP vždy ukazuje na nižší byte slova nacházejícího se na vrcholu HW zásobníku. Při ukládání dat se hodnota SP zmenšuje. Na níže uvedených obrazcích se zásobník bude zaplňovat směrem dolů.

Standardní tvar aktivačního záznamu byl zaveden firmou Microsoft a v základních rysech je všeobecně dodržován. O odchylkách se zmíníme dále. Pokud implementace PJ akceptuje tento standard, lze pak ladícími prostředky jako CodeView, nebo TurboDebug analyzovat, zobrazovat a zpřístupňovat obsah běhového zásobníku.



Volání podprogramu probíhá takto:

- volající program umístí na zásobník hodnoty parametrů (tím začne vytvářet nový AZ);
- je-li volaný program staticky vnořen do jiného podprogramu X, volající uloží na zásobník odkaz na aktivační záznam X (statický spoj);
- instrukce volání podprogramu CALL zapíše na zásobník návratovou adresu;
- volaný podprogram obdrží v BP adresu aktivačního záznamu volajícího a uloží ji na zásobník (dynamický spoj);
- do BP umístí adresu svého (právě vytvářeného) aktivačního záznamu;
- na vrcholu zásobníku vymezí oblast pro své lokální proměnné.

Standardní začátek a konec podprogramu vypadá takto:

```
PODPR:  PUSH BP     ; odkaz na dynamického předchůdce
        MOV  BP, SP ; do BP adresa vlastního akt.zázn.
        SUB  BP, rozsah_lokálních_proměnných
        PUSH SI     ; jen pokud podpr. mění reg. SI
        PUSH DI     ; jen pokud podpr. mění reg. DI
; vlastní podprogram
        POP  DI     ; jen pokud bylo PUSH DI
        POP  SI     ; jen pokud bylo PUSH SI
        MOV  SP, BP ; likvidace místa pro lok. prom.
        POP  BP     ; obnovení ukazatele na akt.zázn.
        RET         ; návrat
```

(Uschovávání a obnovování SI a DI se v Turbo Pascalu 4 nepoužívá.)

Pro přístup ke svým parametrům a lokálním proměnným podprogram používá bázování registrem BP. Při použití BP totiž procesor implicitně pracuje se zásobníkovým segmentem (viz kap. 1). K obsahu BP se přičítá posunutí, které je pro parametry kladné, pro lokální proměnné záporné. Pořebujeme-li přistoupit k objektu s posunutím XX na předchozí statické úrovni, lze použít např. obratu:

```
MOV DI, [BP+4]; DI=adresa stat. nadř. akt. zázn.
MOV AX, SS:[DI+XX]; AX:=hodnota proměnné
```

(příklad předpokládá blízkou návratovou adresu).

Tento způsob přístupu k proměnným deklarovaným ve staticky nadřazených blocích vyžaduje ve srovnání s udržováním displeje (tj. pole odkazů na statické nadřízené) méně organizačních akcí a méně paměti. Doba přístupu k proměnné je horší pouze při překračování více než jedné statické úrovně. Další výhodou použité metody je, že předáváme-li podprogram jako parametr, stačí s ním předat adresu aktivačního záznamu statického nadřízeného a nikoliv celý displej.

4.3 Volací konvence

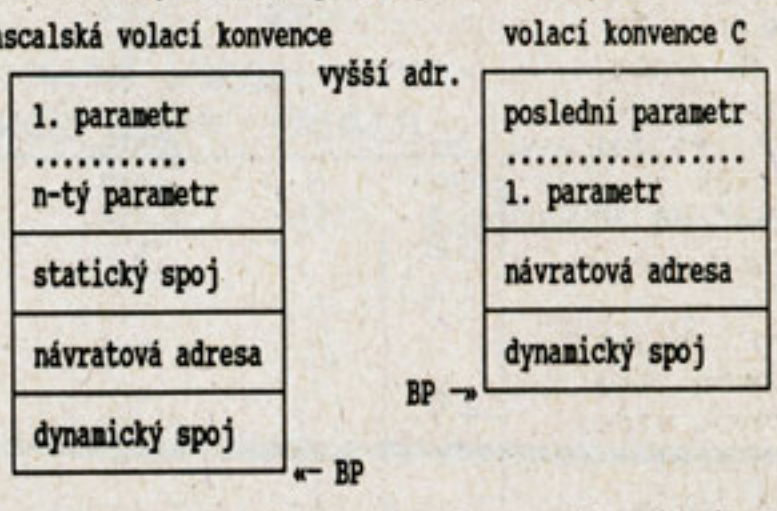
V předchozím odstavci jsme pomlčeli o tom, v jakém pořadí se parametry zapisují na zásobník a kdo je ze zásobníku odstraní. Je zřejmé, že i v této otázce musí nastat shoda mezi podprogramem a každým modulem, který jej volá. Takové dohodě se říká **volací konvence**.

Při volání podprogramu se jeví jako nejpřirozenější ukládat parametry v tom pořadí, v jakém jsou uvedeny v hlavičce. Pak ve vytvořeném aktivačním záznamu bude mít první parametr největší posunutí a poslední nejmenší.

Chceme-li však používat podprogramy s proměnným počtem parametrů, popsaný postup se nehodí. Počet parametrů se pak obvykle určuje z hodnoty prvního nebo několika prvních parametrů. K tomu potřebujeme, aby posunutí prvního parametru nezáviselo na počtu parametrů. Toho dosáhneme, uložíme-li parametry na zásobník v **opačném** pořadí, než v jakém jsou uvedeny v hlavičce.

Ukládání pevného počtu parametrů ve stejném pořadí, v jakém jsou v hlavičce, nazýváme **pascalskou volací konvencí**. Parametry pak odstraní volaný podprogram instrukcí RET (RET na 8086/88 může mít parametr určující, o kolik se má změnit hodnota SP).

Ukládání potenciálně proměnného počtu parametrů v obráceném pořadí nazýváme **volací konvence C**. Parametry pak odstraňuje volající. Pouze ten totiž s jistotou ví, kolik jich je.



Pascalskou volací konvenci používají i překladače Fortranu a Basicu. Je poněkud efektivnější než konvence C.

(pokračování)

Informace ze zahraničí

ATARI ST předpovídá počasí?

Zajímavým systémem pro připojení k počítači ATARI ST je TIMESTEP (cena 850 GBP). Jedná se o kompletní sestavu pro příjem snímků z meterologických družic Meteostat-3, GOES a NOAA. Sestává se z parabolické antény, předzesilovače, přijímače a dekodéru. Chcete-li si prohlédnout Evropu, Afriku a oba americké kontinenty z výšky - máte možnost.

TAS AMSTRAD

Tato zkratka označuje novou službu 602. ZO SVAZARMU. Ve spolupráci se ZOZO Media bylo zřízeno Technické poradenství A Servis (odtud zkratka TAS) pro výrobky britské firmy AMSTRAD. Zároveň TAS provádí zprostředkování prodeje za devizy. Zboží je po potvrzení provedené platby vydáno z konsignačního skladu v Praze. Informace ve středisku VTI, Martinská 5, tel. 228774.

HARDCOPY SAPI–1 NA TLAČIARNI K6313

Ing. Miroslav Ondruška

Mikropočítačová stavebnica SAPI-1 bola v poslednej dobe rozšírená o dosku DGD-1, ktorá spolu s Basicom V5.0/G umožňuje počítačovú grafiku. Pre užívateľa je ale dôležité z vykreslených obrázkov na obrazovke získať trvalý záznam. Tlačiareň K6313 v spolupráci s príslušným podprogramom toto umožňuje.

Doska DGD-1 pracuje s rastrom 320x240 bodov a obsadzuje blok 16KB v ľubovolnej štvrtine adresného priestoru operačnej pamäte počítača. Obsah bytu z tohoto priestoru sa zobrazí vo forme ôsmych susediacich bodov v obrazovom riadku, byt na prvej adrese v ľavom hornom rohu. Každému bitu je pridelený 1 bod, prvému bodu bit s najvyššou váhou. Obrazovému riadku prislúcha 40 bytov. Ďalší obrazový riadok ovládajú byty o 64 adries vyššie. Z cekového priestoru 16KB sa využíva len 42x240 bytov.

Tlačiareň K6313 umožňuje okrem tlače bežných znakov aj prepnutie do grafického módu, ktorý je použitý pri HARDCOPY. Tlačiareň používa 8 ihličiek pričom bitu b0 prislúcha najspodnejšia a bitu b7 najvrchnejšia ihlička. Vzdialenosť medzi ihličkami je 1/72 palca, z toho potom vyplýva potreba nastaviť vzdialenosť riadkov na hodnotu 8/72 palca. Poloha prepínačov na paneli pod krytom tlačiarne musí byť nasledovná: 7-1 off, 7-2 on, poloha ďalších prepínačov nie je kritická.

Podprogram zabezpečujúci HARDCOPY, tabuľka 1, vykoná transformáciu obsahu z grafickej videoram na dáta postupne posielané na tlačiareň. Je preložený od adresy 1F00H, programový riadok 15, po adresu 1FA6H a nepoužíva žiadne pomocné registre RAM-pamäte. Ku svojej činnosti potrebuje vedieť adresu handleru tlačiarne, riadok 11. Umiestnenie dosky DGD-1 sa predpokladá v poslednej štvrtine adresného priestoru, od adresy C000H, riadok 18. Po prepnutí riadkovania na 8/72 palca a počiatočnom nastavení registrov, riadok 17 až 20, nasleduje úsek, ktorý zabezpečí vykreslenie 240 obrazových riadkov, riadok 24 až 43. Na začiatku úseku sa K6313 prepne do grafického módu, riadok 24, posunie obrázok do stredu papiera, riadok 25, a vytlačí 8 obrazových riadkov, riadok 28 až 30. Táto činnosť sa v cykle vykoná pre všetky ďalšie osmice obrazových riadkov. Na záver podprogramu nasleduje ešte prepnutie riadkovania na 1/6 palca, riadok 44. Ak pracujete s iným riadkovaním buď zmeníte túto časť podprogramu, alebo po návrate z podprogramu môžte zmeniť riadkovanie basicovým príkazom.

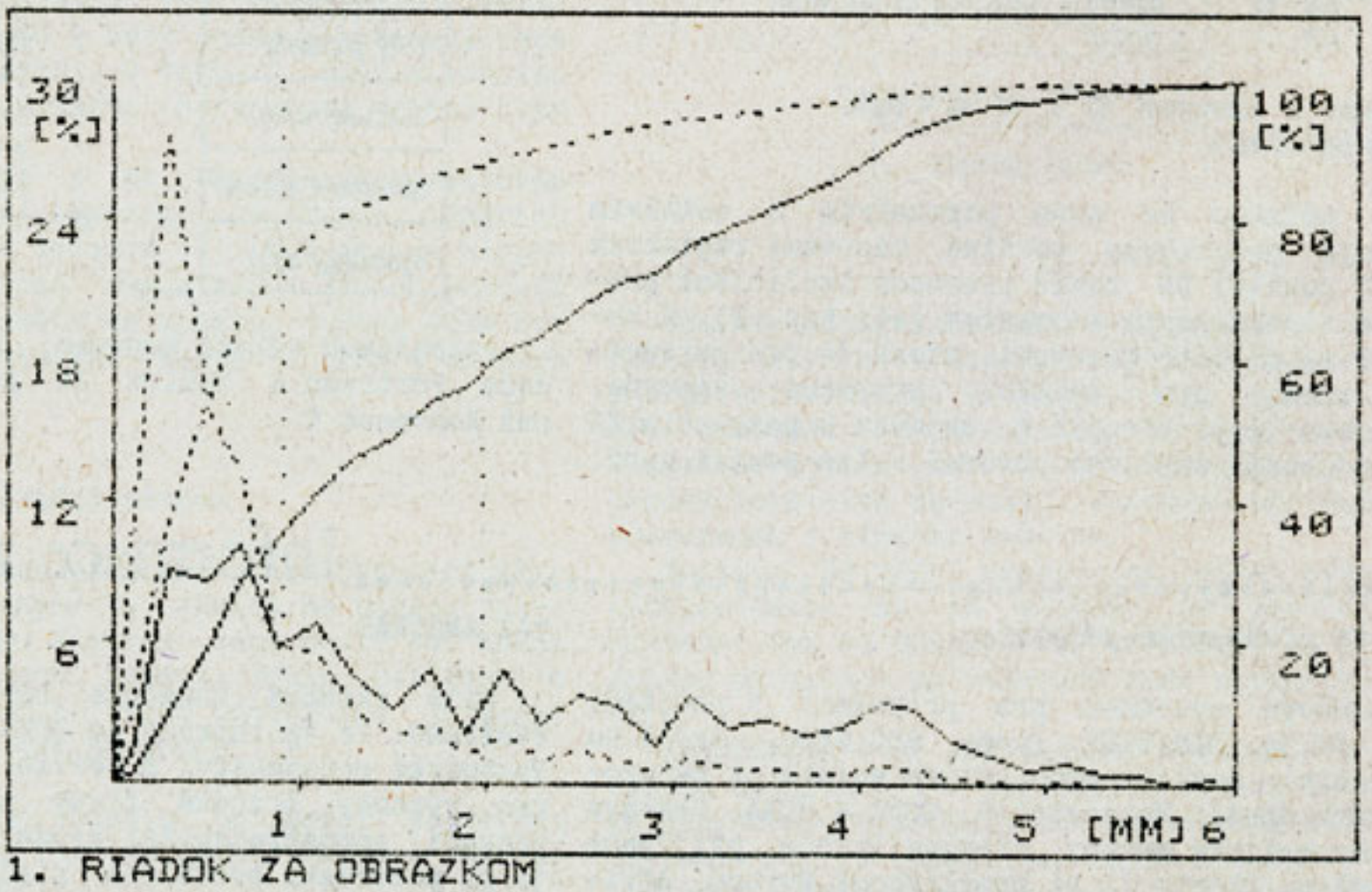
Uvedený podprogram sa z užívateľského programu spustí bežným volaním podprogramu. V tabuľke 2 je ukážka skopírovanej už dopredu pripravenej videoram spolu s listingom krátkeho basicového programu. Skopírovanie trvá niečo vyše minúty a obrázok má rozmery 135x84,5 mm.

Tab. 2 - ukážka volania podprogramu HARDCOPY a výsledok činnosti

```
:LLIST

10 PRINT#L"                    UKAZKA HCOPY GRAFICKEJ VIDEORAM SAPI-1 NA K6313"
20 PRINT#L"                    1. RIADOK PRED OBRAZKOM"
30 X=CALL(HEX(1F00),0)
40 PRINT#L"                    1. RIADOK ZA OBRAZKOM"
50 PRINT#L"                    2. RIADOK ZA OBRAZKOM"

READY
:RUN
                UKAZKA HCOPY GRAFICKEJ VIDEORAM SAPI-1 NA K6313
                1. RIADOK PRED OBRAZKOM
```




```
                1. RIADOK ZA OBRAZKOM
                2. RIADOK ZA OBRAZKOM
```

Tab. 1 - výpis programu

```
;*****************************************
;* HARDCOPY SAPI-1/BASIC V5.0 ==> K6313 *
;*****************************************
;VSTUP:OBSAH VIDEORAM (ADR. 0C000H)
;VYSTUP:TLACIAREN K6313
;MENI:VSETKY REG.
;VYZADUJE:PRAZDNY BUFFER TLACIARNE
;.....
K6313   EQU     7B9H;ADR. OBSLUZ. PROG. TLACIARNE
                    ;-VSTUP:C REG.
                    ;-MOZE MENIT:PSW,BC REG.
;.....
        ORG     1F00H ;UKLADACIA ADR. DO EPROM
        PUBLIC  HCOPY
HCOPY:  CALL    YPOSN1
        LXI     H,0C000H;POC. ADR. VIDEORAM
        LXI     D,64    ;VZDIALENOST 2 BODOV
        MVI     B,30    ;30=240/8     NAD SEBOU
                        ;240-POCET OBR. RIADKOV
                        ;8-POCET POUZITYCH
                                       IHLICIEK
STLPC1: CALL    GMOD    ;VYTLACI (78+320)*240
        CALL    XPOSUN                 BODOV
        PUSH    H
        MVI     C,40    ;40*8 BODOV V RIADKU
RIADOK: CALL    BYT
        DCR     C
        JNZ     RIADOK

        CALL    CR
        POP     H
        DAD     D
        DAD     D
        DAD     D
        DAD     D
        DAD     D
        DAD     D
        DAD     D
        DAD     D
        DCR     B
        JNZ     STLPC1
        CALL    YPOSN2
        RET

BYT:    PUSH    B       ;VYTLACI 8*8 BODOV
        MVI     C,8     ;REG. C-KTORY BIT BYTU
Z1:     MVI     B,8     ;REG. B=8-POCET BODOV
        CALL    BIT
        CALL    ZNAK    ;TLACI 8 BODOV NAD SEBOU
        DCR     C       ;DALSI ZAUJIMAVY BOD
        JNZ     Z1
        INX     H       ;POSUN O 8 BODOV VPRAVO
        POP     B
        RET

BIT:    PUSH    H       ;ZISKA SKUPINU 8 BODOV
        XRA     A       ;NAD SEBOU
S2:     PUSH    B
        PUSH    PSW
        MOV     A,M
S1:     RAR             ;OSAMOSTATNIT ZAUJIMAVY BOD
        DCR     C
        JNZ     S1
        XTHL
        MOV     A,H
        POP     H
        RAL
        DAD     D       ;POSUN O 1 BOD DOLU
        POP     B       ;OBNOVA C REG.
        DCR     B
        JNZ     S2

        POP     H
        RET             ;V A REG. ZIADANE BODY

GMOD:   MVI     A,1BH   ;GRAFIC. MOD K6313
        CALL    ZNAK
        MVI     A,4BH   ;CHR$(27);"K";CHR$(N1);
        CALL    ZNAK                   CHR$(N2);
        MVI     A,8EH   ;78+320=18EH BODOV V
        CALL    ZNAK                       RIADKU
        MVI     A,1
        CALL    ZNAK
        RET

;.....
YPOSN1: MVI     A,1BH   ;NASTAV POSUN VOZIKA O
        CALL    ZNAK                   8/72 PALCA
        MVI     A,41H   ;CHR$(27);"A";CHR$(8)
        CALL    ZNAK
        MVI     A,8
        CALL    ZNAK
        RET

;.....
YPOSN2: MVI     A,1BH   ;NASTAV POSUN VOZIKA O
        CALL    ZNAK                   1/6 PALCA
        MVI     A,32H   ;CHR$(27);"2";
        CALL    ZNAK
        RET

;.....
XPOSUN: PUSH    B       ;POSUN OBRAZKU DO STREDU
        MVI     C,78    ;O 78 BODOV
XPOSN1: MVI     A,0
        CALL    ZNAK
        DCR     C
        JNZ     XPOSN1
        POP     B
        RET

;.....
CR:     MVI     A,0AH   ;NOVY RIADOK+NAVRAT
        CALL    ZNAK                   VOZIKA
        MVI     A,0DH
        CALL    ZNAK
        RET

;.....
ZNAK:   PUSH    B       ;VSTUP:A REG.
        MOV     C,A     ;MENI:PSW
        CALL    K6313   ;POSLE ZNAK NA TLACIAREN
        POP     B
        RET

        END
```

CELOSTÁTNÍ SOUTĚŽNÍ PŘEHLÍDKA ERA'89 TRENČÍN

se uskuteční ve dnech 18. - 25.11.1989. Bude v těchto dnech veřejnosti přístupná denně od 9:00 do 18:00 hod. Místo dlouhého povídání připojujeme seznam plánovaných doprovodných akcí:

- Vyhlášení nejúspěšnějších sportovců Svazarmu SSR
- Přehlídka vítězných prací festivalu audiovizuální tvorby
- Přehlídka prací SVOČ
- Odborné přednášky z elektroniky, výpočetní techniky, audio a videotechniky pro veřejnost
- Odborný seminář "Programování minirobotů a ministrojů" ve spolupráci s DT ČSVTS Bratislava
- Školení instruktorů elektroniky II. třídy
- Prodejní stánky DOSS, Tesla ELTOS, Tesla II. jakost, TUZEX
- Slovenská kniha, Supraphon, Opus a pod.
- Výstava profesionálních výrobků čsl. výrobců
- Promítání odborných a zábavných videoprogramů
- Soutěže pro návštěvníky
- konzultační a předvádécí středisko pro návštěvníky
- Elektronická dílna mládeže
- Kolektivní radioamatérská stanice OK3KTN
- Amatérské videostudio
- Burza programů k počítačům a elektroniky
- Videodisko ples a galakoncert na počest přehlídky ERA'89

MONITOR informací střediska VTI SVAZARMU

Středisko
vědeckotechnických informací Svazarmu
pro elektroniku
Martinská 5, 110 00 Praha 1

Služby střediska

Jsou poskytovány pouze osobně. Vyřizování členství a hostování v 602.ZO Svazarmu, přístup ke knihovně časopisů na mikrofiších, pořizování ozalitových kopií z knihovny časopisů, prodej programových produktů Mikrobáze, zpravodaje Mikrobáze, nepájivých kontaktních polí, technické poradenství pro výrobky firmy AMSTRAD a poskytování informací o odborných akcích Svazarmu.

Pracovní doba

pondělí		zavřeno	
úterý až čtvrtek		10 - 12	14 - 17
pátek		10 - 12	14 - 16
AMSTRAD	pondělí až pátek	8 - 10	

>>>>>>> telefon 22 87 74 <<<<<<<

Seznam časopisů na mikrofiších

64'ER-Das Mag. fuer comp.fans (DE), ACM Siggraph: Comp. Graphics (US), Applied Mathematical. Modelling (GB), Bajtek (PL), Byte (US), Chip-Das Mikrocomp. Magazin (DE), Communication News (US), Communications of the ACM (US), Computer (IEEE) (US), Computer Aided Design (GB), Computer Design (US), Computer Graphics And Applicat. (US), Computer Journal (GB), Computer Networks (NL), Computing Reviews (US), Comsat Technical Review (US), Desktop publishing World (GB), Datamation (GB), Electr. Sound+Rte (CH), Electr. and Wirel. World (GB), Electri - onics (US), Electronics - Int. Edition (US), Elektor (DE), Elektronik (DE), Elektronique Praktique (F), Elo (DE), Elrad (DE), Ezermester (MLR), Funkamateur (NDR), Funkschau (DE), Happy Computer (DE), Hifi News and Rec. Review (GB), Hobby (Magazin der Technik) (DE), IBM Journal of R & D (US), IEEE Trans. Circ. Systems (US), IEEE Trans. Syst. Man Cybern. (US), IEEE Trans. on Softw. Engineering (US), Industrial Robot (GB), Industrial and Proc. Control Mag. (GB), Information and softw. technol. (GB), Intern. Business Equipment (BE), Journ. Acous. Soc. Amer. (US), Journ. Parall. Programming (US), Journ. of The Aud. Eng. Soc. (US), Kompjutr za vas (BG), Komputer (PL), Laser & Applications (US), MC - Die Mikrocomp. Zeitschrift (DE), Microelectr. and Reliability (GB), Mikromagazin (MLR), Mikrodok (DE), Mikroklan (PL), Mikroprocess. Sredstva I Sist. (SU), Moj Mikro (YU), Nachricht. Elektr. + Telematik (DE), Office Equip. and Products (JP), Personal Computer World (GB), PC Magazin (DE), Practical Computing (GB), Practical Electronics (GB), Practical Wireless (GB), RE - Radioelektronik (PL), Radio (SU), Radio Electronics (US), Radio-Amater (YU), Radio und Fernsehen Elektronik (NDR), Radio-Televizija-Elektronika (BG), Radioelektronik (PL), Rádiótechnika (MLR), Revija za mala računala (YU), Robòtica (GB), Siemens R&D Reports (DE), Simulation (US), Software-Practice and Exper. (GB), Solid State Communications (GB), Solid State Technology (US), TB - Report (DE), Techniky Komputerowe (PL), The Office (GB), Toshiba Review (JP), ZX Computing Monthly (GB)

Použité zkratky:

US-USA, GB-Velká Británie, YU-Jugoslávie, JP-Japonsko, CH-Švýcarsko, BG-Bulharská lid. republika, F-Francie, BE-Belgie, DE-NSR, SU-SSSR

Knihy na mikrofiších

Forth (programovací jazyk - manuál v češtině) * Příručka jazyka Basic pro ZX Spectrum (kniha v češtině) * Przewodnik pro ZX Spectrum (v polštině) * Úvod do programování ve strojovém jazyku ZX 81, ZX Spectrum (kniha v češtině) * ZX Microdrive + Interface I (kniha v angličtině) * ZX Spectrum I + II (kniha v angličtině)

Informace ze zahraničí

POKE někam, něco

je noční můrou mnohých herních fanatiků. Zajímavé "pouky" najdete v jugoslávském časopisu Moj mikro. Časopis je mikrofišován 602. ZO Svazarmu a je k dispozici členům Svazarmu ve středisku VTI, Martinská 5, tel. 228774.

ZX SPECTRUM / FISHER TECHNIC

Název nenaznačuje spojení firem, ale hardware. Potřebný popis modulu styku včetně schematu, seznamu součástek a desky tištěných spojů najdete v jugoslávském časopisu Moj mikro 2/1989, str.12.

HYPERPAD

Slovo "hyper" se váže k dnes velmi citovanému výrazu "hypermedia", slovo "pad" pak je zkratkou "Personal Application Developer", neboli - volně přeloženo - prostředek pro vývoj osobních aplikací. HYPERPAD dovoluje nejen úpravu, ale i realizaci programového vybavení a to pomocí vlastního jazyka HYPERCAD, jazyka C či asembleru. HyperPad vám umožní ovládat například CD-ROM, WROM nebo jiný prostředek. Všechny funkce lze realizovat při kapacitě pamětí 512 kilobajtů - je možné tedy využít běžný počítač standardu IBM PC. Bližší informace: PCW 5/89 - str. 118.

PROGRAMOVÁ NABÍDKA

Ve snaze usnadnit vám orientaci v naší nové, rozsáhlejší nabídce, uvádíme napřed všechny programy v přehledné tabulce. V ní snadno naleznete informaci o tom, pro které typy počítačů je ten který program napsán či upraven, jaká je jeho cena a kdy bude v prodeji. **Upozorňujeme, že můžete posílat objednávky na všechny programy, na které je stanovena cena,** bez ohledu na to, zda jsou již dány do prodeje. Připravované programy totiž budeme dávat postupně do prodeje v pořadí podle vašeho zájmu. U některých programů je dokonce dostatečný počet objednávek přímo podmínkou pro zahájení tisku návodu.

Počínaje příštím číslem budeme uveřejňovat stručné specifikace jednotlivých programů. O těch zvlášť zajímavých se pokusíme získat od autorů podrobný popis.

Podtržené ceny značí programy, které již jsou v prodeji. Hvězdička za cenou značí, že program bude v prodeji v nejbližší době.

Jednotlivé typy počítačů, pro něž jsou programy určeny, jsou v tabulce značeny takto:

ZX Spectrum - ZX Spectrum (+,+128,+2,+3), Delta, Didaktik Gama
Atari - Atari 800/130
C64 - Commodore 64/128
Sharp - řada MZ 800
Sord - Sord M5
PMD - PMD-85 (1,2,2a)
IQ - IQ-151
PC - počítače slučitelné s IBM-PC

V ceně každého programu je již započtena cena příslušného nosiče (kazeta, disketa) a obsáhlého tištěného návodu. K této ceně se připočítává případné poštovné.

Všechny nabízené programy můžete osobně zakoupit v naší prodejně **Praha 1, Martinská 5, telefon 22 87 74** (středisko VTEI), kde lze získat též podrobnější informace, případně demoverze některých programů. Další možnost je objednat si programy korespondenčním lístkem na adrese **602. ZO Svazarmu, Dr. Z. Wintra 8, 160 41 Praha 6.** Na objednávce nezapomeňte uvést **typ počítače a jeho verzi!**

	ZX Sp.	Atari	C64	PMD	IQ	SHARP	SORD	PC
PROFESOR II	161,-	169,-	169,-	429,-	429,-			479,-
STUDENT 1	137,-	145,-	145,-	299,-	299,-			399,-
STUDENT 2	137,-	145,-	145,-	299,-	299,-			399,-
TESTEDITOR	457,-	480,-						
ZX MULTITASKING	139,-							
GROS	119,-	129,-	129,-	299,-	299,-			439,-
ODA	149,-							
TEMPERAMENT	90,-							
PROGRAF	130,-	139,-	139,-	329,-	329,-			479,-
SONDA 4D	99,-							399,-
STOPKY	150,-							
DR.MG	124,-							
DATALOG	175,-							
MIKROBÁZE PASCAL	194,-							
CP/M	180,-							
ASSEMBLER 80	189,-							
BASIC S	110,-							
TELETEXT	330,-					330,-	330,-	

(*dokončení tabulky najdete na straně 32*)

PROGRAMOVÁ NABÍDKA

(*dokončení tabulky ze strany 31*)

	ZX Sp.	Atari	C64	PMD	IQ	SHARP	SORD	PC
PLAYTAPE	109,-							
REMBRANDT	170,-							
DATALOG 2	175,-							
MFLOG	97,-							
TAPELOG	109,-							
SOS COPY							96,-	
DAM 0000 V2.0				190,-				
DAM +2 V1989				169,-				
KASWORD V3.0				188,-				
KAREL V2.2				135,-				
GRUTI				95,-				
WELLCOPY				190,-				
GRED				175,-				
GREP				165,-				
MUSICA				180,-				
COLOSS MON						210,-		
KANTOR I						485,-		
KANTOR Ia						295,-		
KANTOR II						470,-		
KANTOR III						370,-		
GEORUT I						130,-		
KURS MS-DOS								998,-
KURS DBASE III+								998,-
TEXT602								2998,-
DISKETA #1								798,-
DISKETA #2								798,-

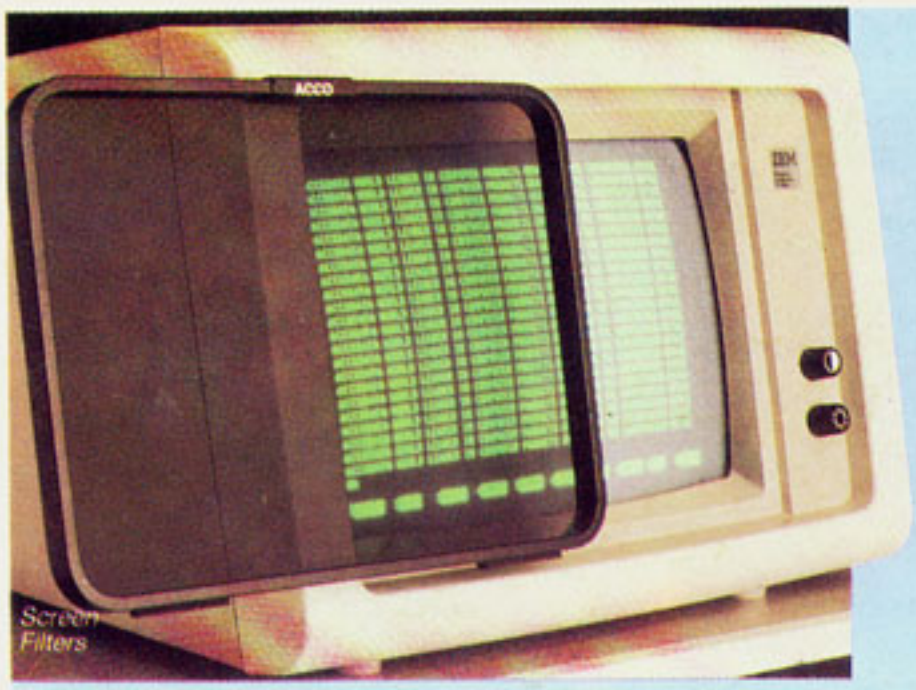
Každý program je dovoleno v jednom čase používat pouze na jednom počítači a také jakékoli pořizování kopií programů, jejich dat nebo příruček dalším osobám není dovoleno.

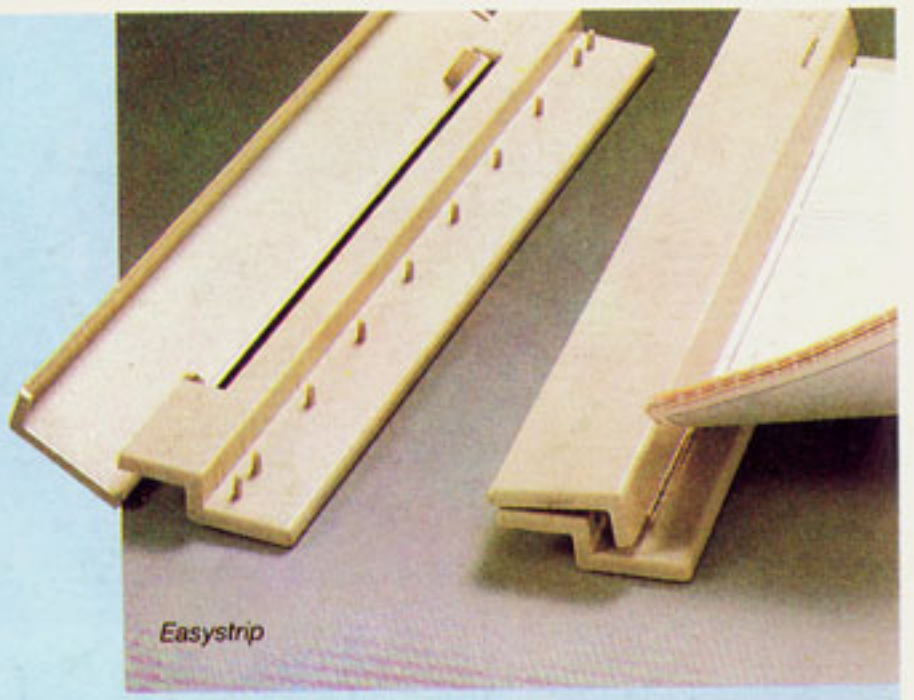
TEXT 602 SE PŘEDSTAVUJE

(dokončení ze 2. strany obálky)

Ve své přednášce poukázal ing. Kaucký na úskalí, se kterými se setkali tvůrci původního českého textového editoru a vysvětlil, jak se s nimi vyrovnali. Během krátké přestávky poskytl rozhovor redaktorovi ČS Televize. Následovalo předvádění možností editoru, během kterého bylo zodpovězeno nepřeberné množství otázek. Ještě dlouho po skončení oficiální části programu probíhaly diskuse. Zdá se, že TEXT 602 vykročil do života úspěšně. Vy, čtenáři Mikrobáze, se o něm dozvíte řadu podrobností v rozhovoru s ing. Kauckým uvnitř tohoto čísla.

Text i foto: D. MECA





ACCODATA

COMPUTER ACCESSORIES

Tato anglická firma se zaměřila na nejrůznější doplňky a pomůcky pro výpočetní techniku. Z jejího bohatého katalogu jsme vybrali jen několik ukázek různých praktických maličkostí. Nad řadou z nich se nám přitom nabízí otázka, zda by něco takového nedokázali také naši výrobci. Zatím je nám známo pouze to, že síťku podle obrázku vlevo nahoře vyrábí u nás nějaké JZD. Výrobci, ozvěte se!