MIKROBÁZE

08 -PASCAL (4)
-desktop publishing
-digitalizace obrazu
-jak naučit PMD česky

SPOLEČNÁ SLUŽBA AMATÉRSKÉHO RADIA A 602.ZO SVAZARMU PRO UŽIVATELE MIKROPOČÍTAČŮ

AR-602

OBSAH

Mikrobáze, technický zpravodaj pro mikroelektroniku a výpočetní techniku. Vydává 602. ZO Svazarmu pro potřeby vlastního aktivu. Zodpovědný redaktor ing. Jan Klabal, sestavení rukopisu Ladislav Zajíček. Adresa redakce: 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6, telefon 32 85 63. Neprodejné! Povoleno ÚVTEI pod evidenčním číslem 87 007. Náklad 4000 výtisků. Praha, červenec 1987

ERA 1987 Žďár nad Sázavou

Celostátní přehlídka "ERA 87 Žďár nad Sázavou" je pořádána na počest 70. výročí VŘSR. Svým zaměřením na propagaci úspěchů socialistické společnosti při plnění úkolů závěrů XVII. sjezdu KSČ a VII. sjezdu Svazarmu, přehlídka dokumentuje rozvoj polytechnické výchovy mládeže, branně-politické výchovy a činnosti, technického vzdělávání a technické propagandy ve svazarmovské elektronice.

Cílem přehlídky "ERA 87 Žďár nad Sázavou", jako účinné vzdělávací, výchovné a propagační akce pro veřejnost, je přispět k získávání návštěvníků, zejména mládeže, pro elektronizaci národního hospodářství, jako významného intenzifikačního faktoru výstavby socialistické společnosti.

V neposlední řadě si přehlídka klade za cíl rozvíjení polytechnické výchovy, technické tvořivosti a branně technické činnosti v elektronice ve všech odbornostech Svazu pro spolupráci s armádou. Umožní aktivizovat zlepšovatelské a vynálezecké hnutí na pomoc národnímu hospodářství i vlastní organizaci. Dále umožní prohloubení propagace této činnosti na veřejnosti.

Přehlídka se koná ve dnech 19. až 29. listopadu 1987 v Domě kultury ROH Žďár nad Sázavou.

Součástí přehlídky bude kromě konstrukční a branně sportovní a technické činnosti, včetně různých akcí radioamatérů, i **Mikroden malé výpočetní techniky.** Uskuteční se v sobotu 28. 11. 1987 a semináře i přednášky, které v tento den proběhnou, budou zaměřeny na hardware a software z oblasti mikropočítačů. Nad celou akcí má patronát "ikrobáze 602. ZO Svazarmu. Cílem Mikrodne je veřejná propagace malé výpočetní techniky a její užitkové vlastnosti, a zvýšení zájmu o hardwarovou a softwarovou oblast. Očekává se také výměna již získaných zkušeností mezi uživateli, a to jak z provozu mikropočítačů, tak i periferních zařízení a programového vybavení.

Všechny zájemce o malou výpočetní techniku zveme na tuto jedinečnou celostátní akci.

Desktop publishing

Desktop - deska stolu, publishing - vydávání. Nový termín nového způsobu elektronického zpracování textu a grafiky. Desktop publishing (dále jen DP) je zároveň novým komerčním hitem soft/hardwarových producentů v oblasti osobních počítačů (týká se především IBM PC se všemi jeho typy a klony).

DP představuje integraci tradičně dlouhého řetězce výroby tiskovin do jedné počítačové konfigurace, kterou pohodlně umístíte na desce stolu. Po dlouhá staletí měl tento řetězec dva konce - autora na začátku a tiskárnu na konci. Uprostřed seděl vydavatel-nakladatel, který rozhodoval, co vydá, přesněji - do čeho investuje (co koupí a bude pak prodávat). Před autorem stála nejen tvorba sama, ale i zeď komerčních tlaků působících na vydavatele s jeho edičním plánem, kapacitami, lhůtami atd. Proto se v souvislosti s DP hovoří o demokratizaci vydavatelské činnosti.

Podobně, jako se v případě CAI (Computer Aided Instruction - výuka počítačem) mluví o demokratizaci přístupu ke vzdělání.

Vpád počítače do redakcí a tiskáren není ničím zvlášť novým. EPS - Electronic Publishing System existuje v mnoha podobách už řadu let. Jenže - podobně jako ještě před 15 lety byste za počítač se schopnostmi ZX Spectra museli dát svůj celoživotní plat, do minulého roku neměl ekonomicky průměrně si žijící člověk nejmenší šanci stát se svým vlastním elektronickým vydavatelem.

Potřebná hardwarová DP konfigurace sestává z počítače se 16ti-nebo 32tibitovým mikroprocesorem, pamětí min. 1 MB, jednoho, lépe však dvou monitorů, floppy, hard disku a laserové tiskárny. Vše možno pořídit už v ceně kolem 2500 dolarů. Cena konfigurace je přímo závislá na kvalitě tiskárny (1200-60000 dolarů). Nemalou položku tvoří i nezbytný software - ceny se pohybují v rozpětí od 200 do 8000 dolarů.

Uživatelům mikropočítačů je znám v podstatě jen jeden způsob práce se znaky a jejich tiskem. Textovými editory (třeba populárním Taswordem) do paměti uložené znaky v jejich ASCII reprezentaci (1 znak - 1 bajt) nejsou počítačem dále nijak typograficky zpracovány. Takové zpracování může provést až teprve bodová (např. jehličková) tiskárna v závislosti na řídících kódech, které na ni před tiskem nebo během něj pošleme. Tak můžeme tisknout některé základní typy písma, ev. několik jejich kombinací. Nepříjemným omezením osmibitových textových editorů je nemožnost kombinace zpracování, tedy i průběžného tisku znaků a grafiky (písma a obrázků). Přesněji - příprava takového materiálu i jeho tisku by byla velmi (a zbytečně) komplikovaná, navíc s četnými omezeními.

Konfigurace DP vybavená náležitým programem pracuje s připravovaným tiskovým materiálem zásadně odlišným způsobem. Řídící kódy pro tisk textu (resp. grafiky) nejsou nutně vysílány spolu s ním na tiskárnu, na níž uživatel teprve vidí, "jak to vlastně tak asi bude vypadat", ale řízen je všechen budoucí skutečný typo/grafický vzhled materiálu už přímo při jeho zpracování v počítači, tedy ještě před tiskem. Laserová tiskárna pak už "jen okopíruje", co uživatel skutečně vidí na obrazovce. Tomuto zpracování se anglicky říká WYSIWYG (what you see is what you get - co vydíte, to dostanete).

Pro to, abychom opravdu viděli, co dostaneme, je nezbytný monitor s velmi hustou budovou sítí. I počítač musí mít adekvátně velkou obrazovou paměť. Protože průběžná (interní) programová příprava obsahu tak rozsáhlé paměti je trochu těžkopádná, je vhodné v celé konfiguraci mít ještě jeden monitor, kde se text připravuje "po staru", podobně jako třeba při práci s Taswordem. Při pohledu na takto připravovaný text na monitoru uvidíte jeden typ znaků, protkaných houštinou "čínského písma", které je množinou řídících kódů budoucího vzhledu tiskového materiálu. Kdykoli ovšem můžete dát počítači příkaz, aby vám na druhém monitoru ukazál, jak bude vše ve výsledku vypadat. Zapojení dvou monitorů přineslo s sebou nutnost soft/hardwarového řešení jejich "přepínání" i volby interaktivnosti - tj. na kterém monitoru kdy a jaké (a zda vůbec na obou) budeme moci provádět přímé zásahy.

Obrázky pro jejich následné importování do programu DP lze tvořit známými způsoby. Uživatelům mikropočítačů nejbližší je "kresba kurzorem" pomocí grafického programu. Tzv. scannerem můžete převést do paměti (digitalizovat) na papíru či jiném povrchu už vytisknutý nebo nakreslený materiál, ev. jej sejmout z diapozitivu apod.

A pomocí digitalizace obrazu sejmutého TV kamerou pak můžete dát do paměti opravdu co oko ráčí. Obdržený obrázek je samozřejmě vždy možné nějak dále upravovat. K tomu je potřebné další programové vybavení.

Programy DP mají příjemnou vymoženost. Obrázek si můžete doslova naprogramovat pomocí příkazů jednoduchého jazyka, který je v programu implementován (často na bázi FORTHu). Nese obecné označení DPL - Description Page Language - jazyk pro tvorbu stránky. Podstatně rozšiřuje nejen možnosti grafické úpravy materiálu trochu vzdělanějším grafikem, ale také dává všem inženýrům a vědcům možnost okamžitého, "bezpracného" grafického zobrazení funkcí, o nichž v textu pojednávají. S výsledným obrázkem je pak možno si opět hrát dle libosti.

V této souvislosti profesor D. G. Pelli, autor jednoho ze série článků věnovaných DP v měsíčníku BYTE 5/87, uvádí žertovnou historku o tom, jak s ním přestala mluvit jedna americká firma. Profesor se zabývá zkoumáním schopností lidského vizuálního systému (jednoduše - co, proč, jak a kdy naše oko vidí či nevidí). Pro ten účel si vytvořil řadu matematických programů v jazykových implementacích DPL. Protože zřejmě neměl zbytečných 60000 dolarů, chtěl využít služeb jedné z firem, které nabízejí vytisknutí čehokoli na drahých laserových tiskárnách podle zákazníkem dodaného programu. Za jednu vytisknutou stránku pak požadují 10 dolarů. Profesorův program byl tak náročný, že pro vytisknutí jedné grafické stránky obsadil firemní tiskárnu na 4 hodiny. Firma mu k dodané stránce připsala vzkaz: "Never more!" (Nikdy více!).

Doba tisku takto programově připravených obrázků je samozřejmě přímo závislá na schopnostech programátora. Tak např. v jednom časopise uveřejněný test uvedl, že program Post Script tiskl naprogramovanou podrobnější mapu USA na velmi kvalitní laserové tiskárně 8 hodin. Producenti Post Scriptu se okamžitě ozvali, examinátorův program překódovali a dosáhli času 4 hodiny. Zde složitost věcí tkví v kombinaci komplikované grafiky se spoustou textu. Samozřejmě, že tu nešlo o čistý čas tiskárny, který je velmi krátký, ale o převedení programem zakódované informace do bitové grafické mapy, která se teprve tiskne. Kdybyste další kopie tiskli z této výsledné mapy (umístěné v bufferu přístupném tiskárně), vše už by pak šlo laserovou rychlostí. Pokud se snad ptáte, k čemu tedy DPL, když použitím bitové mapy to jde ráz na ráz, pak jsou pro to dva důvody. Prvním je rozsah mapy - kolem 1 MB. Tak by se vám i na hard disk vešlo dost málo obrázků (stály by opravdu za všechny peníze). Proto i text je kódován v ASCII. Druhým důvodem, navazujícím na předchozí, je to, že velmi efektní grafické obrazce, stejně jako průběhy funkcí, lze snadno vyjádřit několika jednoduchými programovými vztahy. Nehledě už na to, že kdybyste takový obrázek měli kreslit kurzorem, nikdy by se vám to nepodařilo.

Protože tradiční formát stránky je obdélník - z pohledu čtenáře "protáhlý vertikálně" - monitor pro kontrolu konečného vzhledu stránky nemá obrazovku postavenu klasicky "na širší stranu", ale na užší. Při zobrazení stránky můžeme používat několikastupňový elektronický zoom (transfokátor), kterým lze prohlížet jakoukoli část stránky jako pod lupou a "nad stránkou" s ní přejíždět.

Jednou z nejobtížnějších fází přípravy tisku materiálu je jeho lámání do stran včetně montáže obrázků do textu. Tato fáze je u dobrých DPL zajímavou a příjemnou hrou. Obrázek na stránce můžete umístit kamkoli libo, v jakékoli velikosti, jakkoli

jej "seříznout" apod. Text se pak automaticky rozmístí kolem něj na zbývající volné ploše stránky podle pravidel vámi zadaných. Lámat můžete jednu stránku nebo obě sousedící najednou. Při lomu můžete vzhled písma (především titulků) i jeho rozmístění jakkoli upravovat, doplňovat grafikou atd.

Vše dosud uvedené vypadá celkem nekomplikovaně. Komplikace však každému, kdo se do DP míní pustit, nastanou, když se má rozhodnout při výběru potřebného softwaru. Každý z nich něco umí, ale také neumí, může obsluhovat omezený rejstřík typů tiskáren atd. Jedním z nejdůležitějších momentů pro výběr softwaru je, zda jím budeme chtít připravovat a zpracovávat obsáhlejší materiály, které budou na jedné stránce kombinovat různé typy písma s grafikou a obrázky při variabilním rozvržení "novinových" sloupců na tiskové straně apod. V tom případě musíme sáhnout po dražším produktu, který nám vše zmíněné umožní bez komplikací, s vysokým stupněm automatizace operací při rozmisťování textu na každé stránce a všech možných dodatečných úpravách a změnách. Při této vysoce náročné práci se bez dvou monitorů obejít nelze. Pokud nebudeme tak nároční, vystačíme s levnějším softwarem a jedním interaktivním monitorem.

Jednodušší programy DP dávají přednost zadávání povelů výběrem funkcí z menu - jsou určeny méně zdatným, nenáročným uživatelům. U ostatních se povely zadávají funkčními tlačítky v kombinaci s možností využití příkazů implementovaného jazyka DPL. První jsou pomalejší, mají méně funkcí, druhé jsou rychlé, vybavené vším možným, ale bývá dost obtížné se s nimi naučit pracovat rychle, při využití všeho, co umějí.

Při výběru DPL je třeba dát pozor i na to, zda do nich lze importovat text a grafiku z jiných programů. Prakticky žádný software DP neobsahuje tak dokonalý grafický program, aby předstihl některý z těch, které jsou pro ten který počítač již vyvinuty. A také pro to není žádný důvod. Při práci s rychlou vedlejší pamětí není problém si zavolat grafický program, na který jsme navíc už zvyklí, obrázek si v něm vytvořit a pak jej kdykoli namontovat do programu DP. Nejde však jen o tento druh obrázků, ale např. i o grafy výsledků z kalkulačních programů, přenos různých tabulek i s texty apod. Zde jde hlavně o šetření časem a variabilitu zpracování. Proto zásadně platí, že ke každému softwaru DP potřebujeme dále nejméně jeden program pro práci s grafikou a jeden slovní procesor. Plejádu těchto programů můžeme rozšiřovat podle potřeby, ovšem s ohledem na to, co je náš DP software schopen importovat.

Kromě uvedených, jsou mezi DP programy ještě další četné rozdílnosti. Např. při montáži obrázku do textu je pro něj u některých programů nutno předem vytvořit "díru" na stránce a text rozmístit kolem. při změnách rozměru nebo umístění obrázku je tuto operaci nutno opakovat stále dokola, což někdy může připravovat o čas i o nervy. Vyskytují se dokonce i programy, které neumějí provést některé globální změny v již vytvořeném materiálu nebo i měnit vzdálenost řádek! Některé mají limitující potíže při tvorbě komplikovanějších kombinací textu s grafikou. Výhodou při tvorbě prvního čísla nějakého periodika je, když můžeme uchovat kódy jeho formátu, umístění a tvaru titulů apod. pro přípravu čísel dalších. To umí jen málo programů. Nejen z těchto důvodů je nutno jejich výběru věnovat zvýšenou pozornost.

Jeden z významných rozdílů mezi programem DP a běžným slovním procesorem je prů-

běžná tvorba proporčního písma. Něco podobného dokáží i lepší maticové tiskárny, ale ne v provedení DP. U DP se zcela ztrácí původní význam tiskových sloupců, jejichž počtem se udával konstantní počet úhozů na řádce, včetně neměnné šířky mezery (space). V DP programu třeba písmena VA napsaná vedle sebe zasahují každé do bitové mapy svého souseda. Podobně je tomu u mnohých sousedství písmen psaných italikou (ležatě) - jde např. o provázání dvou písmen f. I když se tak vlastně "prolne" sloupec bajtů, tvořících druhé písmeno do prvního, program si s tím dokáže poradit. Této funkci se anglicky říká kernel. Mezery mezi slovy mají proměnnou bitovou šířku v přímé závislosti na počtu slov na řádce - jejich celková bodová délka se odečte od stanovené bodové šířky řádky a výsledek se vydělí počtem slov, zmenšeným o 1. Perfektní proporcionalita písma je jednou z hlavních předností DP z hlediska estetiky tisku.

Další odlišnou specialitou DP softwaru je způsob uložení bitové mapy znaků. U slovních procesorů je tato mapa uložena "natvrdo" někde v paměti počítače, bajt po bajtu. Při psaní znaku je jemu odpovídající sestava bajtů odebrána (z oblasti zvané character set) ke zobrazení na monitoru. U DP však každému znaku přísluší jeden program, který, v závislosti na zadaných vstupních parametrech, vypočítá jeho konečný vzhled. Aby nebylo nutno nechávat znaky vypočítávat stále dokola, mapy znaků, se kterými pracujeme, se ukládají do volné paměti - oblasti dat znakových sestav. DP program po stisku tlačítka napřed zjistí, zda už znak sestavil. Když ano, mapu převede na monitor. Když ne, vypočítá ji, uloží "napropříště" do datové oblasti a zobrazí. Takto vytvořené znaky jsou pochopitelně používány jen při psaní stejného typu znaků. Závisí na konstrukci programu, zda při změně typu bude tato oblast vymazána, či uchována pro případné pozdější užití. Další zvláštností je, že DP program může při tvorbě textu využívat i předprogramovaných znakových sad umístěných v laserové tiskárně.

Jinak velká část DPL umí v podstatě vše, co ovládají vysoce kvalitní slovní procesory, včetně oprav gramatických chyb či dělení slov na konci řádek (obojí je v angličtině poměrně jednoduché). Pro tyto účely DPL obsahují komprimovaný slovník, s nímž jsou slova textu porovnána.

Výsledná kvalita tisku polotónů (různých stupňů šedi) je přímo závislá na použité laserové tiskárně. Technologie tohoto tisku přinesla nový tiskový rermín - buňku (cell), která obsahuje týž polotón. Např. má-li tiskárna grafickou hustotu podání tisku 300 bodů/palec a 18,75 buněk/palec, znamená to, že pro vyjádření jednoho stupně šedi potřebujeme 300:18,75=16 bodů. Jinými slovy - na jednom palci můžeme podat max. 18 stupňů šedi, přičemž devatenáctý bude v tomto případě zasahovat už do "sousedního palce". Tento poměr analogicky platí i dvourozměrně - jeden stupeň šedi vyžaduje plochu 16x16=256 bodů. Uvedené parametry platí pro nejlacinější laserové tiskárny (např. AppleLaserWriter). Ty nejkvalitnější (např. Linotype L300) mají podání výrazně lepší - 2540 bodů a 120 buněk. Tisk s těmito parametry je od fotografického podání prostým okem nerozeznatelný. Některé firmy vyrábějí (dost drahá) zařízení, na nichž můžeme tisknout v podobně vysoké kvalitě barevně. Tak např. digitalizovaný barevný obraz nějakého mistra malíře je ve výsledném barevném podání přesnou kopií originálu.

I když máte monitor s nižší rozlišovací schopností než má ke konfiguraci připo-

jená laserová tiskárna, vůbec to neznamená, že by se nechala o něco ošidit. Jemnost podání bude vždy adekvátní parametrům tiskárny. O to se stará hardwarové zapojení, zvané RIP (Raster Image Processor - rastrový grafický procesor). Díky tomu, že je napojen přímo na video signál, sleduje průběh signálu každé TV řádky a rozkládá jej na počet bodů, který tvoří jedna grafická řádka laserové tiskárny. Podle úrovně jasu v tom kterém momentu RIP rozhodne, zda laser "zhasne" nebo "bude svítit", tedy i zda barvící toner v tom místě přilne na tiskovou plochu (papír apod.) či nikoli. Jestli při svitu laseru toner přilne či naopak, závisí na typu tiskárny. Jeden z posledních typů zapojení RIP využívá paměť 3 MB pro to, aby během tisku jedné stránky mohl průběžně a s velkou operační rychlostí připravovat bitovou mapu stránky další. Umožňuje i využití části této paměti pro uložení (down load) vzhledu specifických znaků, které se v té které bitové mapě objeví, využívá se i pro další služby (pak tento RIP zpracovává jen jednu stránku). V souvislosti s přenosem TV obrazu na laserovou tiskárnu je vhodné se zmínit o tom, že se vlastně jedná o obdobu digitalizace TV obrazu. Při něm je obrázek, snímaný např. video kamerou, rozložen na jednotlivé bity paměti podobným způsobem. A protože většinou má kamera vyšší rozlišovací schopnost, než s jakou pracuje obrazová paměť počítače, jedná se o opačný poměr - video signál je rozložen na menší počet bodů (bitů obrazové paměti), aby počítačem byl reprodukovatelný. Tak ovšem z obrázku leccos zmizí. Podobně by zmizelo i z laserové tiskárny, kdyby měla nižší rozlišovací schopnost než obrazová paměť počítače. Z toho plyne, že rozlišovací schopnost této paměti musí být stejná nebo nižší než jakou má připojená laserová tiskárna.

Některé laserové tiskárny vyšší třídy mohou připravovat filmové pozitivy pro následný fotografický převod zobrazení na tiskové matrice. Objevily se onyxové fólie, které jsou po vyjmutí z laserové tiskárny prakticky připraveny pro maloofsetový tisk s životností cca 25000 kopií.

"Před použitím naší tiskárny se prosím, připoutejte! Poletíme laserovou rychlostí!" - zní slogan jednoho výrobce. Rychlosti tisku se u laserových tiskáren spodní cenové hranice pohybují v rozmezí 5-10 stran za minutu. Horní hranice dosahuje 120 str./min. Tisk je takřka neslyšný (dokonce tak, že slyším hluboký povzdech majitelů jehličkových tiskáren). Tiskárny bývají vybaveny rozsáhlým bufferem pro přijetí hotového vzhledu stránky, aby během tisku nebyl blokován počítač pro jinou činnost. Pochopitelně můžete také předem stanovit, kolik kopií té které stránky chcete vyhotovit. Na vstup tiskárny lze vložit najednou 200-250 kusů papíru, které si tiskárna sama odebírá. Jinak jsou tiskárny - podobně jako maticové - vybaveny různými předprogramovanými sadami znaků. Tak lze na nich tisknout i text připravený tradičním způsobem.

Jednou ze stinných stránek každého zařízení, v němž se něco hýbe, je jeho omezená životnost. Životnost mechaniky laserové tiskárny se pohybuje kolem dvou set (vyjímečně až šesti set) tisíc vytisknutých stran. V poměru k její ceně to není počet příliš efektivní. K nákladům je třeba připočítat nutnou výměnu (nákup) kazety s tonerem po každých cca 3000-5000 vytisknutých stranách. Tam kde by užití laserové tiskárny bylo vyloženě samoúčelné, bylo by i neekonomické. Jako u každého nasazení dražšího zařízení, i u laserové tiskárny je nutno mít předem zajištěnu návratnost, resp. opodstatněnost vynaložených prostředků.

Tak, jako se u nás začíná pomalu, ale jistě projevovat přestup uživatelů osmibitových počítačů na šestnáctibitové, lze předpokládat, že budou své nové majitele inspirovat ke konfiguračnímu rozšíření na systém DP. Určitě tomu napomohou i stále klesající ceny hardwaru. Publikující autoři tak budou moci dovést své publikace do konečné podoby na svém vlastním stole. Tento nový způsob práce změní i tradici styku autora s redakcí - místo dosavadního putování autorů do redakcí s proškrtanými svazky papíru pod paží budou redaktoři a grafici chodit za autory, aby těmto novodobým "pilotům" konfigurací DP sdělili, kde by si přáli učinit jaké úpravy a hned se na ně také mohli podívat. Po konečném zpracování celého materiálu autor přinese do tiskárny buď jen krabičku disket nebo vytisknuté předlohy pro tisk. Půjde-li o menší počet tenčích výtisků, zhotoví je přímo doma. Co víc si může autor - ale především čtenář - přát?

-elzet-

Pascal /4/

I když je tato část našeho rychlokursu Pascalu poslední, budeme se k němu na stránkách Zpravodaje Mikrobáze dále vracet.

Systematický přístup k algoritmizaci složitějších problémů spočívá v jejich rozkladu na řadu problémů a jejich samostatné algoritmizaci v podprogramech. To platí pro programování obecně. V Pascalu se podprogramy dělí na procedury a funkce:

Procedura - algoritmus, který nějakým způsobem transformuje data

Funkce - algoritmus výpočtu jedné hodnoty, která může být použita jako operand ve výrazu

PROCEDURY

Až dosud jsme se seznámili s několika procedurami a (matematickými) funkcemi, které jsou v Pascalu implementovány - např. procedury READ a WRITE obsluhují vstup a výstup programu. Pro zopakování příklady jejich vyvolání:

```
read (Počet);
Prvek := sqrt(Počet)
```

Způsob deklarace programových procedur si ukážeme na jednoduchém příkladu. Představme si, že v několika místech programu potřebujeme na chvilku zastavit jeho běh. Můžeme toho dosáhnout zařazením cyklu FOR:

```
begin
  for n := i to 1000 do;
end.
```

Pro to, abychom nemuseli tento cyklus psát do všech míst programu, kde budeme chtít programový běh pozdržet, můžeme nadeklarovat proceduru a z jednotlivých programových míst ji pak volat. Jedná se tedy o techniku volání podprogramů umístěných

v programové struktuře. Požadovaná procedura bude např. takováto:

```
procedure Počkej;
begin
  for n := i to 1000 do;
end;
```

Deklarace procedury, podobně jako programu, začíná její hlavičkou (identifikátorem), zakončenou středníkem. Jednoduchá procedura má tvar:

```
procedure identifikátor;
begin
  posloupnost příkazů
end;
```

Procedury musejí být ve struktuře programu deklarovány před operační částí programu. Struktura zápisu pascalovského programu je tedy v zásadě tato:

1. Hlavička programu
2. Definice konstant
3. Definice typů dat
4. Deklarace proměnných
5. Deklarace procedur
6. Deklarace funkcí
7. Operační část programu

Procedura (i funkce) má podobnou strukturu. Uvnitř procedury můžeme rovněž deklarovat konstanty, typy dat, proměnné, ale i další procedury. Všechny tzv. objekty, které definujeme a deklarujeme uvnitř procedury, se ve vztahu k ní nazývají lokálními objekty. Jejich zvláštností i předností je, že se nestávají součástí globálních objektů hlavního programu. Tak šetří prostor paměti i čas potřebný k jejich programovému vyhledávání. Kromě uvedených existují ještě objekty nelokální, které vznikají při vnořování dvou či více procedur do sebe. Lokální objekty jedné procedury se stávají nelokálními objekty následující. Lokální (samozřejmě i nelokální) objekty tedy existují jen během exekuce procedury, resp. procedur vnořených, mimo ni ne.

Po skončení běhu procedury (vnořených procedur) je paměť vyhrazená pro práci s (ne)lokálními objekty uvolněna. Pro procedury však nepřestávají být dostupné všechny globální objekty, které byly do momentu její exekuce nadefinovány. Jak z uvedeného vyplývá, vyznačuje se Pascal tou zvláštností, že globální i (ne)lokální objekt může mít shodné jméno. V tom případě dává procedura přednost (ne)lokálnímu objektu a svého globálního jmenovce se netkne (zůstává nezměněn).

Tak jsme se dostali k rozšířenému tvaru procedury:

```
procedure identifikátor;
  lokální deklarace
begin
  posloupnost příkazů
end;
```

Naši předchozí proceduru Počkej můžeme podle toho rozšířit:

```
procedure Počkej;
  var n : integer;
begin
  for n := i to 1000 do
end;
```

Tak pro tuto proceduru již nemusíme deklarovat proměnnou n v hlavním programu.

Nyní zasadíme naši proceduru do programu:

```
program Odpočítávání;
var k : integer;

procedure Počkej; (JakDlouho : integer)
var k : integer;
begin
  for k := i to JakDlouho do;
end;

begin
  for k :=  10 downto 0 do;
    begin
      writeln (' --- ',k);
      Počkej (32000)
    end;
  writeln  ('S T A R T')
end.
```

Provedením tohoto programu se přesvědčíme, že změny lokální proměnné k nemají vliv na stejnojmennou globální proměnnou.

PARAMETRY PROCEDUR

Naše první deklarace procedury Počkej nebyly nijak univerzální. Jejich provedení trvalo vždy cyklem FOR stanovený časový úsek. V předchozím programu jsme proceduru obohatili o parametr. Voláním procedury z programu můžeme vždy stanovit jeho hodnotu (zde to bylo 32000). Provedení procedury Počkej pak bude trvat po dobu nutnou k provedení 32000 jejích cyklů. Parametr nemusíme stanovit jako absolutní hodnotu, ale můžeme použít i proměnné z programu, např. Počkej (x).

Parametry procedur se definují v jejich hlavičce jako seznam specifikací formálních parametrů. Jednotlivé parametry se oddělují středníkem. Znázorníme si obecně i toto rozšíření:

```
procedure identifikátor (seznam specifikací parametrů);
  lokální deklarace
begin
  posloupnost příkazů
end;
```

Příklad hlavičky procedury s parametry:

```
procedure PROC (FX, FY:integer; var FZ:real;
                procedure FP (F:integer));
```

FX a FY - form.parametry typu integer volané hodnotou
FZ - form.parametr typu real volaný odkazem
FP - form.procedura s jedním parametrem typu integer volaným hodnotou

V Pascalu jsou zavedeny dva mechanismy volání parametrů - hodnotou a odkazem. Parametr volaný hodnotou je lokální proměnnou procedury (je to tedy její vstupní parametr). Parametr volaný odkazem představuje v těle procedury vždy konkrétní proměnnou určenou skutečným parametrem. S ní pak v proceduře pracujeme jako s její budoucí výstupní proměnnou, vypočítanou funkcí procedury.

V deklaraci parametrů musejí být uvedeny názvy typů, nikoli však jejich plné definice. Chybná je např. tato deklarace:

```
procedure Výpočet (Tabule : array [ 1..100 ] of real);
```

Správný postup řešení tohoto problému - napřed v patřičné části programu provedeme definici typu, např.:

```
type Kódy = array [ 1..100 ] of real;
```

a pak (někde dále v programu) při deklaraci procedury píšeme:

```
procedure Výpočet (Tabule : Kódy);
```

Představme si následující deklaraci:

```
procedure RozměrTab = 48;
type JednaTab = array [ 1..RozměrTab ] of integer;
var Tab1, Tab2, Tab3, Tab4 : JednaTab
Počet : integer;
```

Dohodněme se ještě na tom, že náš algoritmus vyžaduje, aby počáteční hodnota všech elementů tabulek byla 1. To můžeme zajistit pomocí procedury. Dále však chceme, aby s její pomocí byla možná změna libovolné ze čtyř proměnných typu JednaTab:

```
procedure Inicial (var NějakáTabulka : JednaTab;
                               Hodnota : integer);

var k : integer;
begin
  for k := 1 to RozměrTab do
    NějakáTabulka [ k ] := Hodnota;
end;
```

Klíčové slovo VAR před deklarací parametru NějakáTabulka znamená, že v momentu aktivace procedury Inicial parametr převezme hodnotu JednaTab. Příklady volání procedury:

```
Inicial (Tab1, 1);                nebo:   Počet := 0;
                                          Inicial (Tab4, Počet)
```

Příkladem chybného volání je např.:

```
Inicial (1000, 0);
```

mimo jiné i proto, že 1000 není názvem proměnné typu JednaTab.

Kromě toho, že procedura může volat jakoukoli jinou, dříve deklarovanou proceduru, může volat i sebe samu - pak jde o proceduru rekurzívní.

FUNKCE

Jak uvedeno výše, funkce jsou algoritmy pro výpočet hodnoty určitého typu. Deklarujeme je podobně jako procedury, ovšem s tím, že v hlavičce uvádíme typ funkce (jejího výsledku) a v její části operační přiřazujeme získaný výsledek jejímu názvu.

Schématické znázornění deklarace funkce:

```
function identifikátor (sezn. specifikací param.): typ funkce;
  lokální deklarace
begin
  posloupnost příkazů
end;
```

Typ funkce nesmí být strukturovaný a musí být označen identifikátorem typu. V těle funkce musí být přiřazovací příkaz, kterým se identifikátoru funkce přiřazuje hodnota a jehož provedením se definuje výsledná funkční hodnota.

Volání funkce se provádí tzv. zápisem funkce, zatímco volání procedury je příkaz. Příklad:

```
function PiKrát (k : real) : real;
begin
  PiKrát := 3.14 * k
end;
```

Volat tuto funkci můžeme např. těmito zápisy funkcí:

```
x := sin (PiKrát (2));          nebo:     writeln (PiKrát(3/2));
```

kde číselné parametry (v prvním případě 2, ve druhém 3/2) jsou přiřazeny proměnné k.

Procedury s jejich parametry i funkce pochopíte ihned, když si uvedené příkazy vyzkoušíte na počítači. A hned vás jistě napadnou jejich aplikace ve vašich vlastních programech (sestavených pomocí uB-PASCALu?).

Čtyři krátké části rychlokursu programování v jazyku PASCAL pochopitelně nezahrnují vše, čím se tento jazyk vyznačuje. Redakce zpravodaje Mikrobáze se bude obracet na jí známé odborníky s prosbou o poskytnutí dalších studijních materiálů.

Pokud jsou takoví i mezi vámi, čtenáři, nechť se nám přihlásí. Rádi jejich dobře zpracované příspěvky otiskneme.

Na závěr tohoto rychlokursu je zařazen delší program, který je modifikací známé "Hry na život", v níž je simulováno rození a zánik kolonií buněk. V programu je obsaženo vše, co jste se v našich čtyřech částech rychlokursu mohli naučit.

```
program HraNaZivot

const XPole = 10;
      YPole = 10;
type  StavPole = (Zive Mrtve):
      Plocha = array [ 1..Xpole,1..YPole ] of StavPole;
      OdODo8 = 0..8;
var   Pole, NovePole : Plocha;    {Plocha poli }
      Pokolení       : integer;   {Cislo pokoleni}
      Sousedstvi     : OdODo8;    {Pocet sousedu }
      Konec          : Boolean;   {Konec hry }
      i, j           : integer;
      znak           : char;

procedure Instrukce;
var i : integer;
begin
   for i := 1 to 24 do writeln:,
   writeln ('H R A  N A  Z I V O T');
   writeln;
   writeln ('Na obdelnikovem poli zije kolonie');
   writeln ('bunek. V dalsim pokoleni prezivaji');
   writeln ('bunky, ktere maji 2 nebo 3 sousedy');
   writeln ('Nova bunka se rodi na prazdnem');
   writeln ('mistu, ktere ma 3 sousedy. Ostatni');
   writeln ('bunky hynou. Program umoznuje sle-');
   writeln ('dovani stavu kolonie v jednotlivych');
   writeln ('pokolenich.');
   writeln (                        Stiskni ENTER');
   read Znak;
end;

procedure Inicial (var Tabule : Plocha:
                       Stav   : Stav Pole);
var i,j  : integer;
begin
   for i := 1 to XPole do    { Elementy Tabule budou
   for j := 1 to YPole do      mit stejnou hodnotu Stav }
     Tabule [ i,j ] := Stav
end;

procedure Kresli (Tabule : Plocha);
var i,j  : integer;
begin
   writeln;
   writeln ('          ',Pokolení'-I POKOLENÍ');
   writeln;
```

```
  write  (´          +´);
  for i := 1 to XPole do write (´- ´);
  writeln (´+ ´);

  for j := 1 to YPole do
    begin
      write (´          |´);
      for i := 1 to XPole do
        if Tabule [ i, j ] = Zive then
        write (´O´)
        else                    { Tisk grafiky Tabule }
        write (´ ´);
      writeln (´ ´);
    end;
  write (´          +´);
  for i := 1 to XPole do write (´- ´);
  writeln (´+´);
  writeln;
 end;

procedure Osidleni (var Tabule : Plocha);
var Konec : boolean;          { Vyplneni Tabule
    i, j  : integer;            zivymi bunkami }
begin
  writeln (´ Zadej dalsi dvojice souradnic ´);
  writeln (´   zivych bunek tak, aby       ´);
  writeln (´     0 < x < ´, XPole+1        ´);
  writeln (´     0 < y < ´, YPOle+1        ´);
  writeln (´    Stisk 0 konci zadani.      ´);
 Konec := false;
 while not Konec do
   begin
     write (´         X= ´); read (i);
     if i <> 0 then
       begin
         write (´           y= ´);
         read (j); writeln;
         Tabule [ i, j ] := Zive;
       end;
     Konec := (i=0);
   end;
 for i := 1 to 24 do writeln;
end;
  function PocetSousedu (Tabule : Plocha;
           PX, PY: integer):  0d0Do8;
  var i, j  : integer;
      Pocet : 0d0Do8;
```

```
begin                    {Vypocet poctu
  Pocet := 0;             sousedu bunky}
  for i := PX-1 to PX+1 do
  for j := PY-1 to PY+1 do
    if (i> 0) and (i< =XPole) then
    if (j> 0) and (j< =YPole) then
    if Tabule [ i, j ] = Zive then
      Pocet := succ (Pocet);
  PocetSousedu := Pocet
end;
begin
  Instrukce;
  Konec := false;        {Hlavni cast
  Pokoleni := 0           programu }
  Inicial (Pole, Mrtve);
  Kresli (Pole);
  while not Konec do
    begin
      Pokoleni := Pokoleni+1;
      Kresli (Pole);
      Inicial (NovePole, Mrtve);
      for i := 1 to XPole do
      for j := 1 to YPole do              {Hlavni cyklus
      begin                               programu }
       Sousedstvi :=
       PocetSousedu (Pole, i, j);
       if Sousedstvi = 3 then
         NovePole [ i, j ] := Zive;
       if (Sousedstvi = 4) and
          (Pole [ i, j ] = Zive) then
          NovePole [ i, j ] := Zive;
        end;
      writeln ('ENTER - dalsi pokoleni,');
      writeln ('K+ENTER        - konec.');
      read (Znak) ; writeln;
      Konec := Znak in [ 'K, 'k' ];
      for i := 1 to PoleX do
      for j := 1 to PoleY do
        Pole [ i, j ] := NovePole [ i, j ];
    end;
end.
```

BAJTEK 1/87, s použitím učebnice
Programování v jazyku PASCALU (SNTL 85)
-elzet-

Assembler Z 80

Atonomní rutiny

V seriálu ukázek rutin uděláme malou odbočku. Čtenář našeho zpravodaje, G. Jordanov, nám zaslal svou rutinu pro relokaci assemblerového programu. Protože přímo navazuje na dosud probíranou tématiku, rozhodli jsme se ji uvést na tomto místě.

Rutina relok přepočítává adresy na předem definovaných místech programu. Tato místa musíme umístit jako DEFW do relokační tabulky, která musí být zakončena dvěma nulami. Samozřejmě, že v tabulce budou jen taková místa, u nichž změna programového řízení zasahuje adresy uvnitř relokovaného programu, tj. ty, které se po relokaci změní. Nebudeme tedy měnit adresy skoků JP a volání CALL např. tehdy, když se jedná o skoky do rutin ROMky apod. Uvedené platí i pro odchylky v instrukcích JR a DJNZ.

Vše názorně vysvětluje za rutinou RELOK vypsaný příklad relokace několika krátkých rutin.

```
0000                00010 ;*********************************
0000                00020 ;
0000                00030 ;        RELOKACNA RUTINA
0000                00040 ;     (c)by G.Jordanov  1987
0000                00050 ;
0000                00060 ;*********************************
0000                00070
0000                00080
0000                00090 ; Relokacna cast zabezpecuje
0000                00100 ; aby bol program relokovatelny.
0000                00110 ; po prelozeni nahrame pomocou:
0000                00120 ;
0000                00130 ;     LOAD "" CODE xxxxx (adresa)
0000                00140
0000                00150 ; a spustime:
0000                00160
0000                00170 ;    RANDOMIZE USR xxxxx (adresa)
0000                00180 ;
0000                00190 ; vhodne je predtym pouzit:
0000                00200
0000                00210 ;    CLEAR xxxxx-1 (adresa-1)
0000                00220 ;-------------------------------
0000                00230
0000 214E00         00240 RELOK   ld    hl,reltab
0003 09             00250         add   hl,bc
0004 5E             00260 KONEC?  ld    e,(hl)
0005 23             00270         inc   hl
```

```
0006 56          00280          ld    d,(hl)
0007 23          00290          inc   hl
0008 7A          00300          ld    a,d
0009 B3          00310          or    e
000A 2811        00320          jr    z,ASCII
000C             00330
000C EB          00340          ex    de,hl
000D 09          00350          add   hl,bc
000E D5          00360          push  de
000F E5          00370          push  hl
0010 5E          00380          ld    e,(hl)
0011 23          00390          inc   hl
0012 56          00400          ld    d,(hl)
0013             00410
0013 EB          00420          ex    de,hl
0014 09          00430          add   hl,bc
0015 EB          00440          ex    de,hl
0016             00450
0016 E1          00460          pop   hl
0017 73          00470          ld    (hl),e
0018 23          00480          inc   hl
0019 72          00490          ld    (hl),d
001A E1          00500          pop   hl
001B 18E7        00510          jr    konec?
001D             00520
001D             00530
001D             00540 ;-------------------------------
001D             00550 ; Priklad pre relokacnu rutinu
001D             00560
001D             00570
001D             00580
001D 3E02        00590 ASCII    ld    a,2
001F CD0116      00600          call 1601h;otvor kanal
0022             00610
0022             00620 ;-------------------------------
0022             00630
0022 CD3200      00640 p1       call ppoz;prva pozicia
0025 0603        00650          ld    b,3 ;opakuj 3x
0027 CD4200      00660 rotuj    call znaky;vypis znaky
002A C5          00670          push bc
002B CD3A00      00680 p2       call line ;volny riadok
002E C1          00690          pop   bc
002F 10F6        00700          djnz rotuj
0031 C9          00710          ret   ;koniec
```

```
0032            00720
0032            00730 ;-------------------------------
0032            00740
0032 0618       00750 ppoz    ld    b,24; AT 0,0
0034 0E21       00760         ld    c,33
0036 CDD90D     00770         call  0dd9h
0039 C9         00780         ret
003A            00790
003A            00800 ;-------------------------------
003A            00810
003A 0620       00820 line    ld    b,32
003C 3E20       00830 space   ld    a,32 ;medzery
003E D7         00840         rst   10h
003F 10FB       00850         djnz  space
0041 C9         00860         ret
0042            00870
0042            00880 ;-------------------------------
0042            00890
0042 3E41       00900 znaky   ld    a,65
0044 F5         00910 opakuj  push  af
0045 D7         00920         rst   10h
0046 F1         00930         pop   af
0047 3C         00940         inc   a
0048 FE61       00950         cp    97
004A C24400     00960 p3      jp    nz,opakuj
004D C9         00970         ret
004E            00980
004E            00990 ;-------------------------------
004E            01000
004E 2300       01010 reltab  defw  p1+1
0050 2800       01020         defw  rotuj+1
0052 2C00       01030         defw  p2+1
0054 4B00       01040         defw  p3+1
0056 0000       01050         defw  0000
0058            01060
0058            01070 ;===============================
0058            01080 END     END

00000 TOTAL ERRORS
```

Informační sběrnice

Digitalizace statického obrazu

Namáhavou a zdlouhavou tvorbu počítačové grafiky pomocí různých grafických programů můžeme nahradit konfigurací kamera-videodigitizér-počítač, která vše vyřeší "jednou ranou". Bohužel se celá věc bez kamery neobejde, proto není nijak laciná. Elektronika digitizéru však není náročná ani finančně, ani realizačně. Cena použitých zahraničních integrovaných obvodů nepřevyšuje 14 Lstg. Pokud nejsou pro vás dostupné, budiž vám popis zařízení informací pro jeho převod na naši současnou základnu.

Možné aplikace zapojení jsou různorodé: lze "číst" výkresy, kresby, digitalizovat portréty, předměty a výsledný obraz v případě potřeby ještě dodatečně upravit počítačem. Obraz můžeme použít při práci se záznamovým zařízením, zobrazovat jej na tiskárně, plotteru, zachycovat na diasnímacích apod. Možnosti umělců-elektroniků jsou téměř neomezené. Též amatérskému televiznímu vysílání se otevírají nová pole působnosti - třeba při zpracovávání map nebo přenosu obrázků prostřednictvím RTTY a SSTV. I profesionálové mohou z navrženého zapojení čerpat inspiraci. Možnosti využití digitizéru jsou opravdu široké a záleží jen na uživateli, které z nich zvolí.

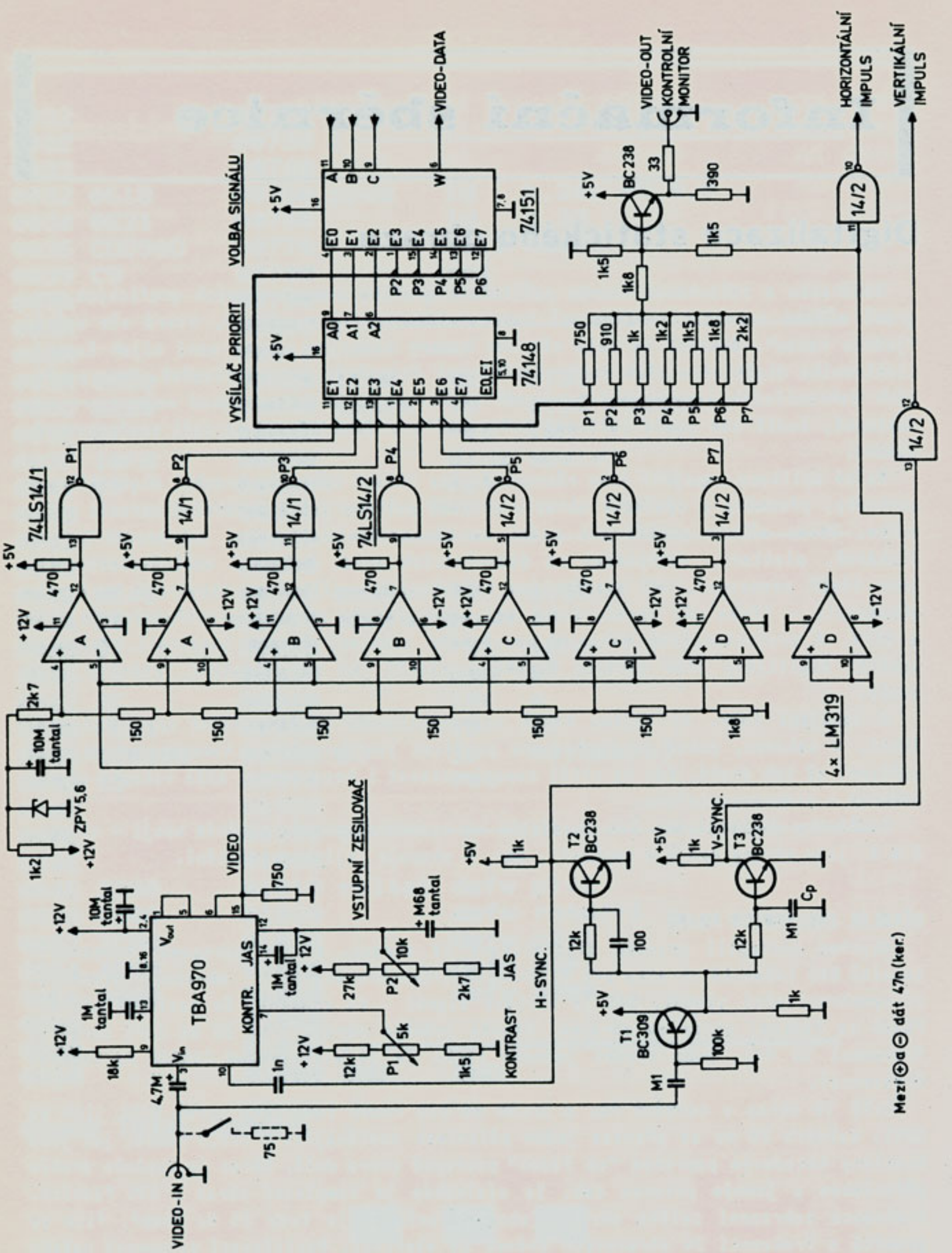
V zapojení (obr. 1) nejsou kromě videozesilovače použity žádné speciální součásti. A nakonec, pokud se rozhodnete používat jen černobílou kameru, nebude celé zařízení až tak drahé. Kdo by chtěl experimentovat i bez kamery, může vstupní signál odebrat z videomagnetofonu.

Součástky digitalizace lze rozmístit na standardizovaném plošném spoji - na obr. 2 a 3 jsou strany spojů a součástí, obr. 4 ukazuje jejich rozmístění.

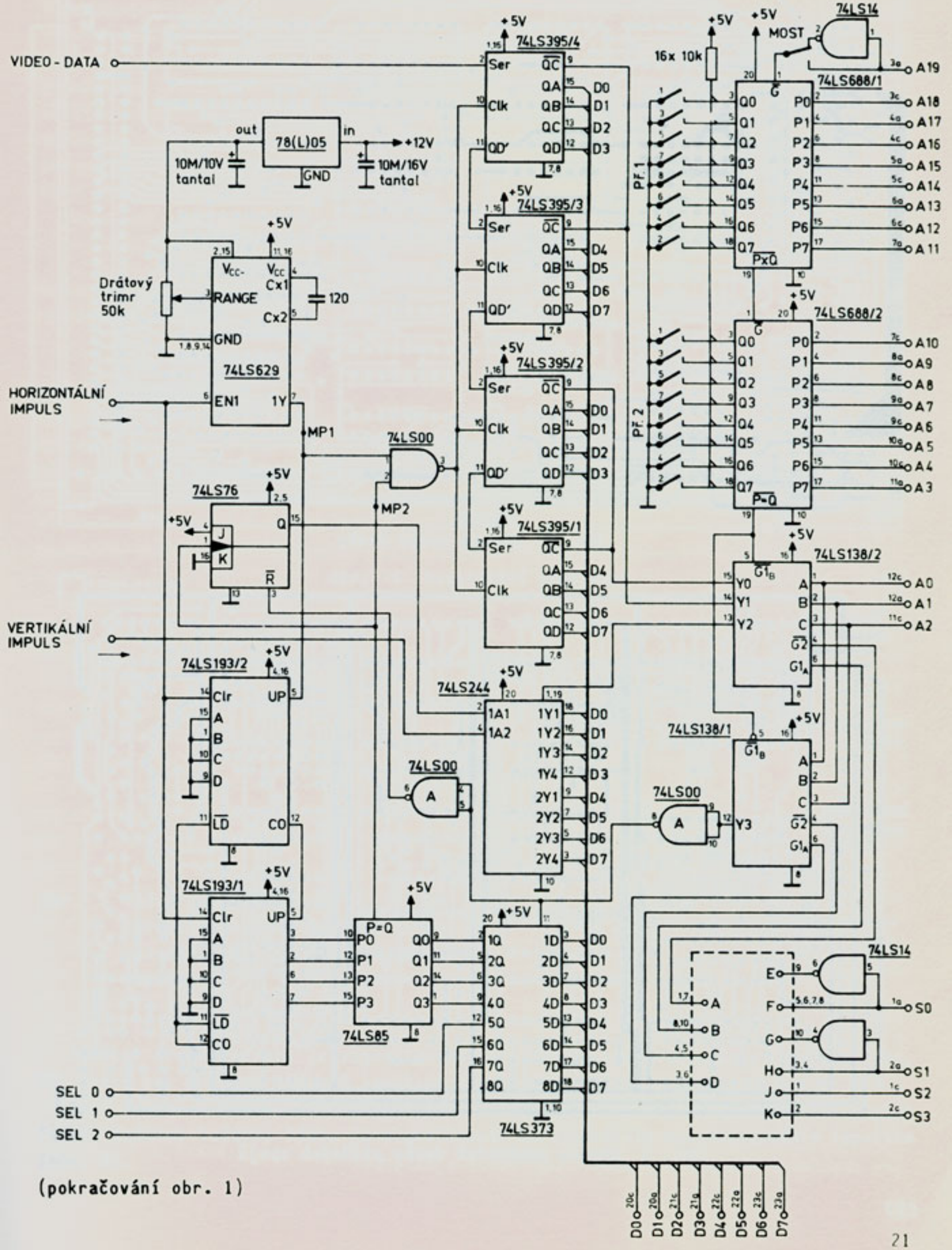
NĚKOLIK ZÁKLADNÍCH POJMŮ

Abychom pochopili princip digitalizace TV obrazu, musíme porozumět konstrukci impulsů BAS (německá zkratka pro Bild-Austast-Synchronsignal): obrazový-zatemňovací-synchronizační signál. BAS je sériový signál - to znamená, že postupně přenáší řádek za řádkem, obraz za obrazem. Stavba jednoho obrazu trvá 20 ms (při obrazové frekvenci 50 Hz). Doba trvání jednoho řádku je 64 mikrosekund (15625 Hz řádkové frekvence).

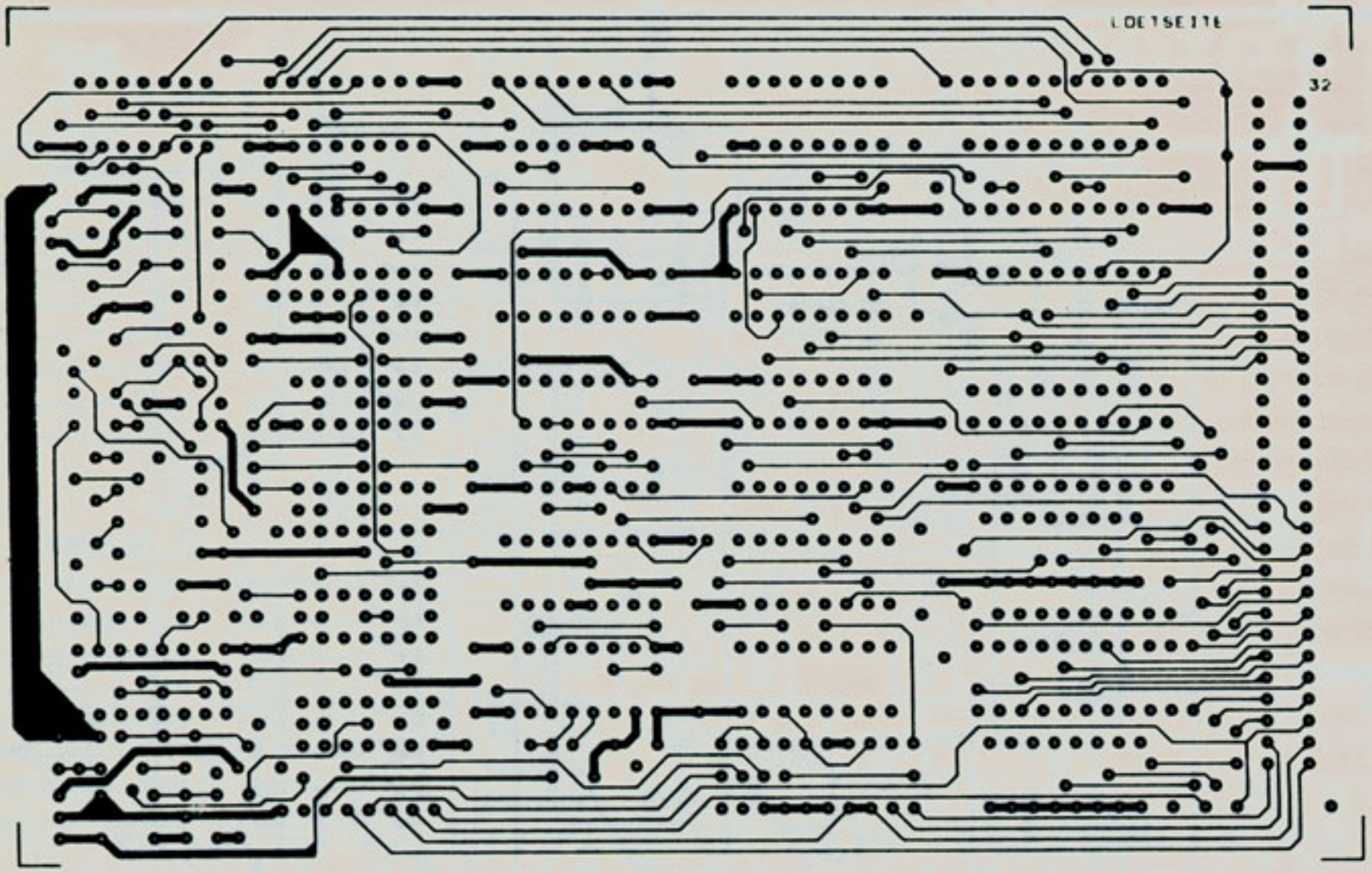
Celý TV obraz v normách PAL a SECAM má 625 řádků (americká norma NTSC jich má 525). Každých 20 ms se přenese jeden půlsnímek sestávající z 312,5 řádků, jež jsou kresleny katodovým paprskem "ob řádek" postupně zhora dolů. Obraz je doplněn druhým půlsnímkem s jemným posunem tak, že výsledkem je kompletní obraz o 625 řádcích. Frekvence zobrazení celého obrazu (obou půlsnímků) je tedy 25 Hz (např. film se promítá s frekvencí 24 obrazových okének za vteřinu). Rozlišovací schopnost našeho digitizéru je 256 řádků. Formát jednoho půlsnímku se 312,5 řádky je proto celkem vhodný.



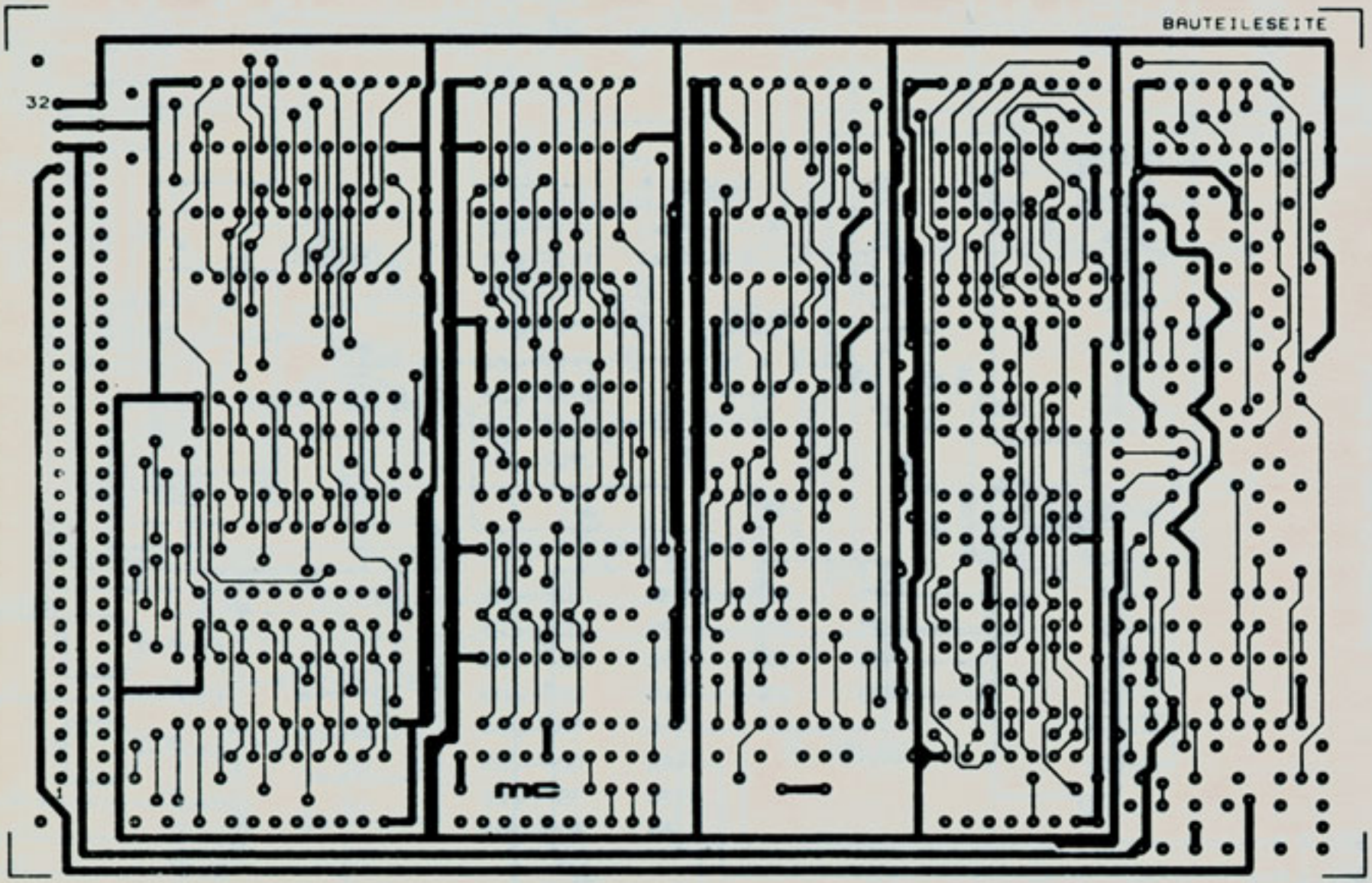
Obr. 1 Na plošný spoj evropského standardu se vejde celý digitalizér. Povšimněte si toho, že spínače DIL pro dekódování adres nejsou uspořádány podle svých hodnot.



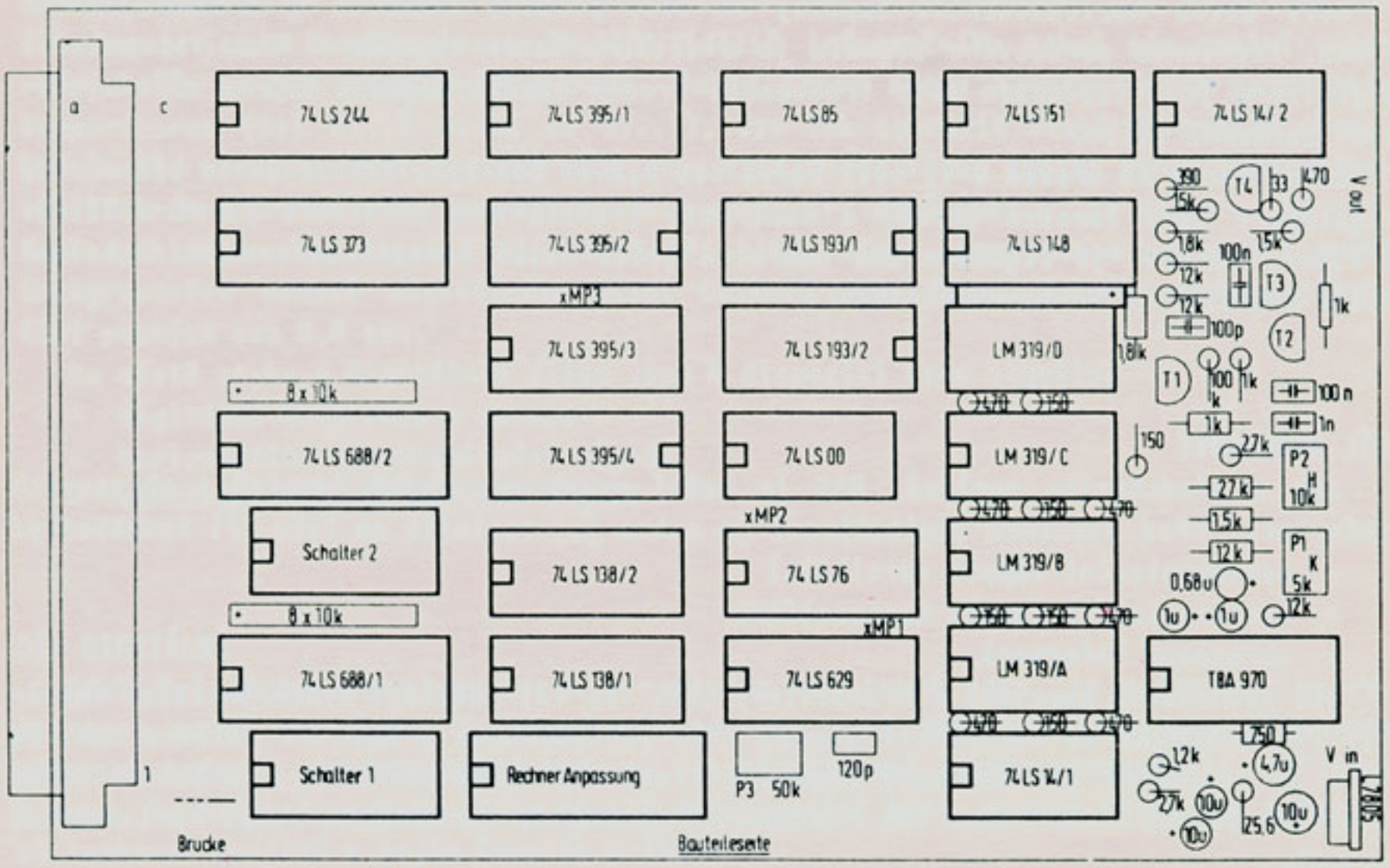
(pokračování obr. 1)



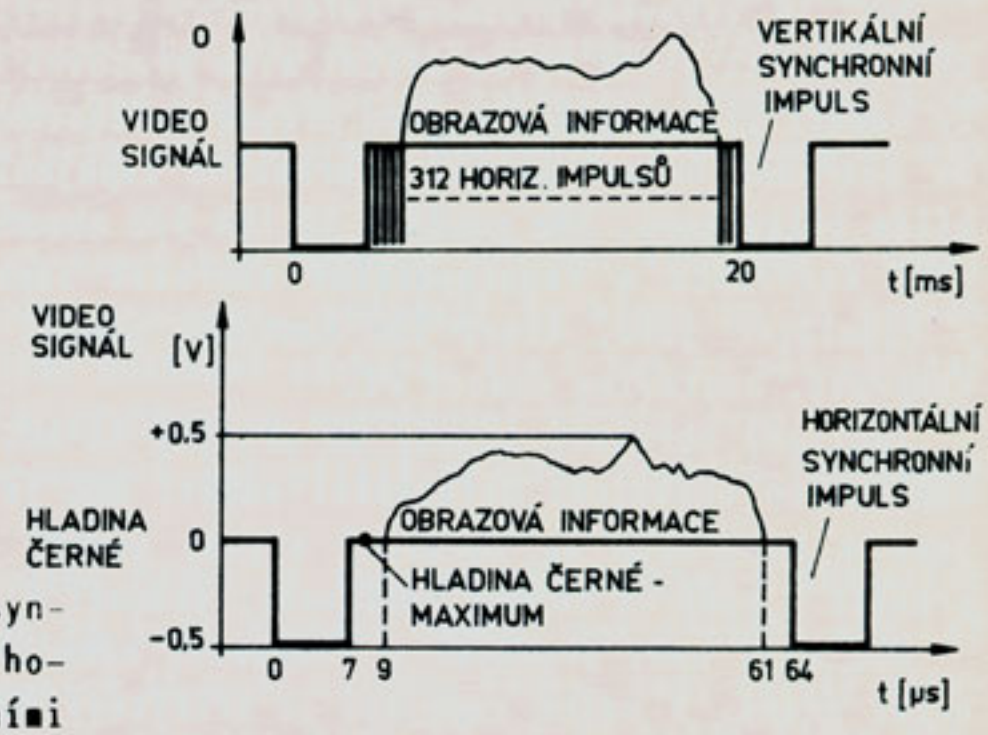
Obr. 2 Strana spojů desky



Obr. 3 Strana součástek desky plošných spojů



Obr. 4 Rozmístění součástek



Obr. 5 Každý jednotlivý obraz je synchronizován vertikálními a horizontálními synchronizačními impulsy

Počátek kresby obrazu je signalizován vertikálním synchronizačním impulsem (obr. 5), který pro nás bude zároveň impulsem startu digitalizace obrazu.

Mezi každými dvěma vertikálními synchronizačními impulsy se přenese 312 plných řádků. Začátek každého řádku je signalizován horizontálním synchronizačním impulsem (obr. 5).

Mezi dvěma horizontálními impulsy se tedy přenese obrazová informace jednoho řádku. Frekvence obrazového signálu je u většiny kamer nejvýše 4 MHz. Vstupní díl digitizéru je prizpůsoben této frekvenci. Úroveň napětí signálů BAS se může u různých zařízení lišit.

Černá barva má úroveň 0 V, bílá barva cca 0,5 V. Obě tyto úrovně mohou být měněny regulátory jasu (úroveň černé) a kontrastu (rozdíl mezi úrovněmi černé a bílé). U kamery barevné je signálům BAS nadřazen ještě nosič barevné informace. Ten však naším digitizérem vyhodnocován není.

DÍL ANALOGOVÝ

Jak z předchozího vyplývá, BAS je směsí synchronizačních impulsů a obrazových informací. Před samotnou digitalizací je nutno jej rozložit na tyto tři komponenty:

- synchronizační impuls vertikální
- synchronizační impuls horizontální
- obrazovou informaci

Dvě jednoduchá zapojení s tranzistory (T1, T2, T3) oddělují synchronizační impulsy. Při přepnutí na vertikální signál (T3) je frekvence pomocí Cf snížena tak, že dále je zesilován jen obrazový signál s frekvencí vyšší než 50 Hz. Za Schmittovými klopnými obvody 64LS14 máme pak k dispozici pravoúhlý vertikální, resp. horizontální signál.

Úpravu obrazové informace zajišťuje předzesilovač a analogově-digitální převodník (dále značený ADC).

ADC je vybaven přesnou referenční úrovní. Obrazový signál se musí pohybovat v této úrovni; jinak by se dostal mimo pracovní rozsah ADC. Vzhledem k tomu, že hodnoty napětí videosignálu nejsou přesně definovány, musí být obrazová informace zvedána do potřebné úrovně pomocí svorkového zapojení, které se vyskytuje v každém černobílém televizoru. Zde použitý IO TBA 970 obsahuje regulovatelný zesilovač k nastavování kontrastu a regulovatelné svorkové zapojení k nastavení úrovně černé barvy- - regulátor jasu, který lze nastavit pomocí P2. Rozdíl mezi černou a bílou, tedy kontrast, nastavíme pomocí P1.

Zesílený obrazový signál se přivede na ADC, který signál otestuje v osmi různých napěťových úrovních. Referenční napětí ADC je pevně nastaveno. Pomocí potenciometrů jasu a kontrastu se obrazový signál umístí do pracovního rozsahu ADC. Lze ovšem provést i záměrně chybná nastavení, vedoucí k vyvolání zvláštních efektů.

ADC musí umět pracovat s frekvencí několika MHz. Takový stavební kámen je však velmi drahý. V našem zapojení je tříbitový ADC nahrazen sedmi rychlými komparátory.

Překlápěcí (či přepínací) referenční napětí jsou předem napevno nastavena sítí odporů. Obrazový signál je optimální tehdy, když úroveň černé barvy leží těsně pod překlápěcím napětím nejnižšího (cca 1.8 V) a hladina bílé barvy těsně nad překlápěcím napětím nejvyššího komparátoru (cca 2.5 V). Nastavování je bez problému - k němu stačí monitor, na němž zobrazíme digitalizovaný obraz.

Na pomocném monitoru můžeme pouhou digitalizací obrazu (i bez použití počítače) vyvolávat zajímavé efekty. Proto se video-fanouškům může vyplatit postavit si i jen analogový díl pro rozličné experimenty s obrazem.

Komparátorům je přiřazen prioritní dekodér, na jehož výstupu jsou třibitové informace o jasu. Demultiplexer 74LS151 (dělič informačního toku) nám umožní volbu různých obrazových signálních zdrojů. Pokud čteme obraz bezprostředně za nějakým komparátorem (Sel 0...2 rovná se 3...7), lze tak zobrazit čistě černobílý obraz. Pokud máme v úmyslu číst binárně kódovanou tříbitovou větu, musíme obraz digitalizovat třikrát a přitom vždy provést selekci oněch tří bitů (Sel 0...2 rovná se 0...2). Nepředpokládá se současné čtení všech tří bitů. Tímto kompromisem se uspoří mnoho integrovaných obvodů digitálního dílu. Celková doba čtení úplného obrazu přesto činí pouhých cca 0,8 sec.

Na místě styku s digitálním dílem jsou k dispozici tyto signály:

- digitální informace o obraze
- horizontální synchronizační signál
- vertikální synchronizační signál

DÍl DIGITÁLNÍ

Když byl obraz v analogovém dílu rozložen na stupně jasu, je v tomto dílu vyhodnocen podél osy času. U televizního obrazu není obrazová informace přenášena přímo po bodech, ale jednotlivé body se v sobě vzájemně jakoby rozpouštějí. Vzhledem k tomu, že počítač pracuje jen s přesně definovanými body, musíme obrazovou informaci testovat spojitě s jejím časovým průběhem.

Horizontální rozlišení digitizéru je 256 bodů na řádek. Jeden řádek trvá 64 mikrosekund. Po uplynutí doby působení horizontálního synchronizačního impulsu zůstává ještě 57 "volných" mikrosekund využitelných pro testování obrazové informace řádku (obr. 5). Z toho vyplývá čas pro testování každého bodu v rozsahu 57/256 mikrosekund, což představuje 223 ns na každý bod (to odpovídá taktovací frekvenci 4.48 MHz). Uvedená frekvence je řízena taktovým oscilátorem. Oscilátor křemíkový se sem nehodí, proto byl zvolen obvod 74LS629.

Počítač by tak měl přečíst jeden obrazový bod každých 223 ns. Vzhledem k tomu, že žádný mikropočítač neumí tak rychle pracovat, střádají se bloky po 16 bodech do posuvného registru, z nějž je počítač postupně odebírá. Ale i tak by měl počítač k dispozici pouhých 3,57 mikrosekund k přečtení a nastřádání těchto dvou bajtů do paměti - a to je pro většinu mikroprocesorů čas příliš krátký. Problém řeší malý trik:

Přečteme pouze prvních 16 bodů prvního řádku. K tomu má náš počítač k dispozici 64 mikrosekund. Potom přečteme prvních 16 bodů druhého řádku, pak třetího atd. až k řádku 256. Tak počítač zpracoval prvních 16 bodů každého řádku, tedy svislý sloupec o šířce 16 bodů. Následuje přečtení sloupce bodů 17 až 32 každé řádky atd. až je nakonec přečten poslední, 16. sloupec.

Obraz se tedy rozloží na 16 sloupců po 16 bodech všech 256 řádků. Pro přečtení každého slova po 16 bitech má počítač k dispozici 64 mikrosekund, což zvládnou i pomalejší CPU. Doba přečtení obrazu obnáší 64 mikrosekund x 256 řádků x 16 sloupců - celkem 0,262 s.

Právě díky popsanému rozdělení obrazu na sloupce bodů je uvedené zapojení tak jednoduché a levné. Čtení obrazu v reálném čase, obvyklé u profesionálních přístro-

jů, si vyžaduje komplikované řízení vedlejších pamětí a podstatné zvýšení pořizovacích nákladů.

Počítač řídí digitizér tak, že čte určený svislý sloupec v šířce po 16 bodech. Číslo příslušného sloupce můžeme libovolně zadávat (tak lze číst i jen svislý výřez obrazu). 16bitový čítač počítá body obrazu a komparátor signalizuje, kdy příslušný sloupec ve videosignálu začíná. Tento signál aktivuje posuvný registr k načítání požadovaného sloupce.

Horizontální synchronizační signál je sledován program - startuje taktový oscilátor a vymazává čítač bodů. Obzvláštní důraz je kladen na čistotu a bezvadnost tvaru bočních hran tohoto signálu.

Pokud čteme bílý sloupec na černém pozadí, jsou nutná určitá opatření (aby svislice zůstala rovná a neroztřepaná):

Oscilátor musí po horizontálním synchronizačním impulsu nakmitat fázově naprosto shodně. Jinak by vznikla odchylka v rozmezí plus minus 1 bodu. Obvod 74LS629 tuto podmínku splňuje.

Na počátku každého řádku uplyne určitý čas, než započne předávání obrazové informace (přední černé ohraničení). Ukázalo se, že není účelné vykrývat tento čas monostabilním klopným obvodem. Je lépe počítat se ztrátou několika původních TV bodů a získat tak bezvadnou výslednou kvalitu obrazu.

Stejně tak i po vertikálním synchronizačním impulsu uběhne určitý čas, než dorazí řádek s informací o obrazu. Uvedenou dobu lze lehce vykrýt časovací smyčkou z programu počítače.

INTERFEJS POČÍTAČE

obsahuje:

- 8 linek datových informací
- 20 linek informací
- linky pro řídící signály

Z **tab. 1** je zřejmá volba registrů a funkcí pro řízení adres a řídících cest. Jak připojit digitizér na různé CPU, ukazuje **tab. 2.**

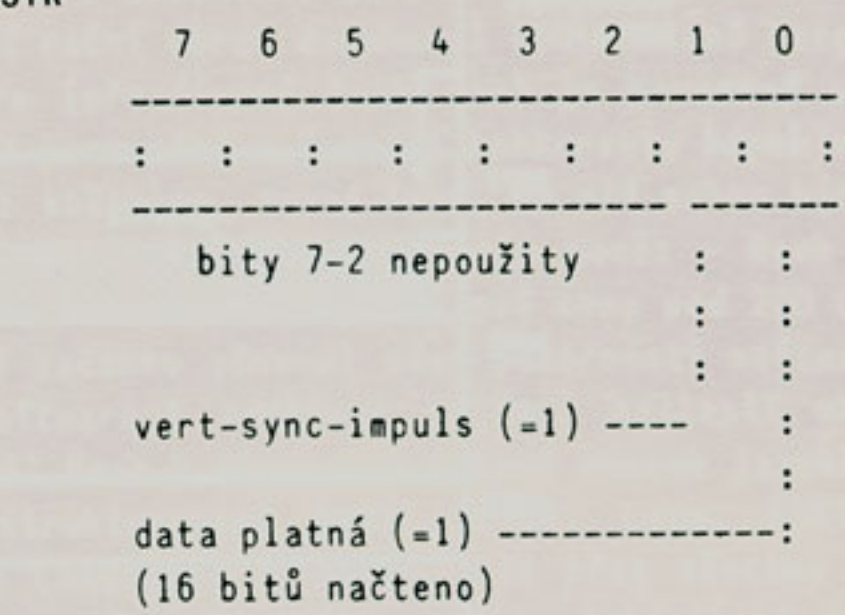
74LS373 hraje roli řídící paměti. Na tomto portu se nastavují čísla sloupců a selektor obrazových dat. K indikaci signálu o stavech digitizéru slouží vstupní port 74LS244. Zde mohou být čteny signály "vertikální synchronizační impuls" a "data platná". Obrazová data jsou čtena přímo z posuvných registrů.

ADRESY REGISTRŮ A VÝZNAMY JEDNOTLIVÝCH BITŮ

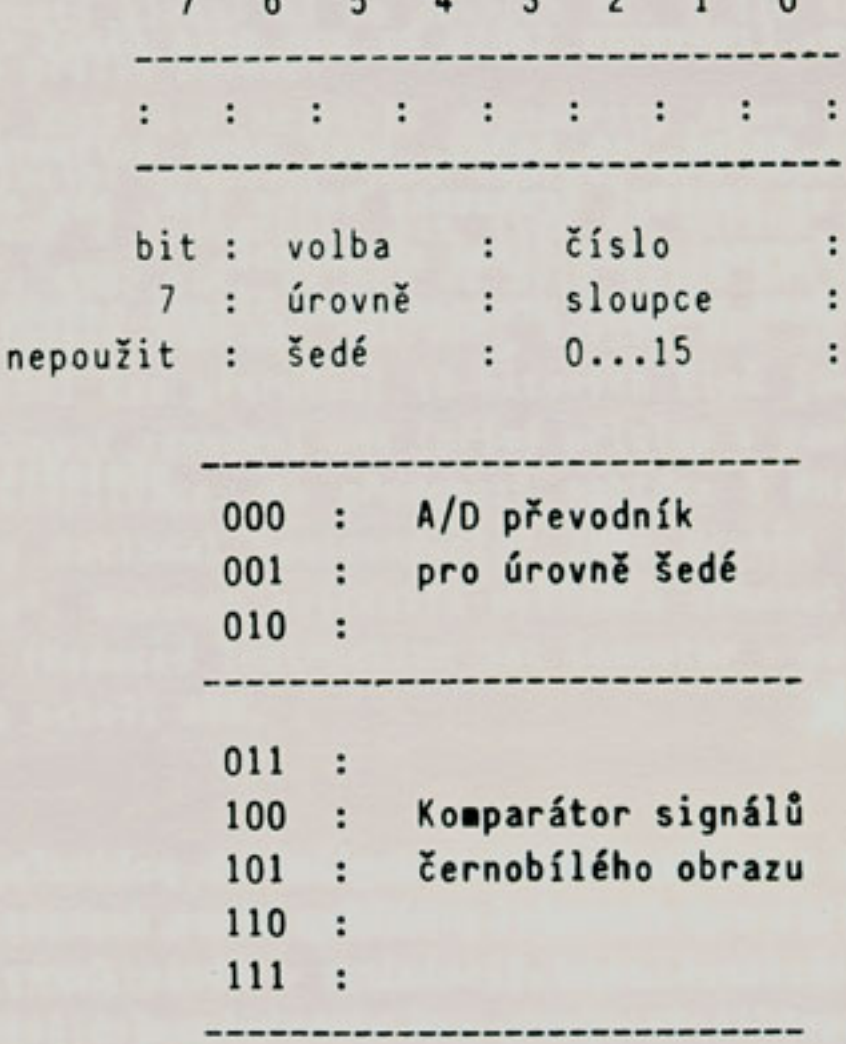
Adresa	Funkce
AD	nižší bajt obr. dat
AD+1	vyšší " "
AD+2	stavový registr
AD+3	řídící registr

AD- základní adresa nastavená spínači DIL

STAVOVÝ REGISTR



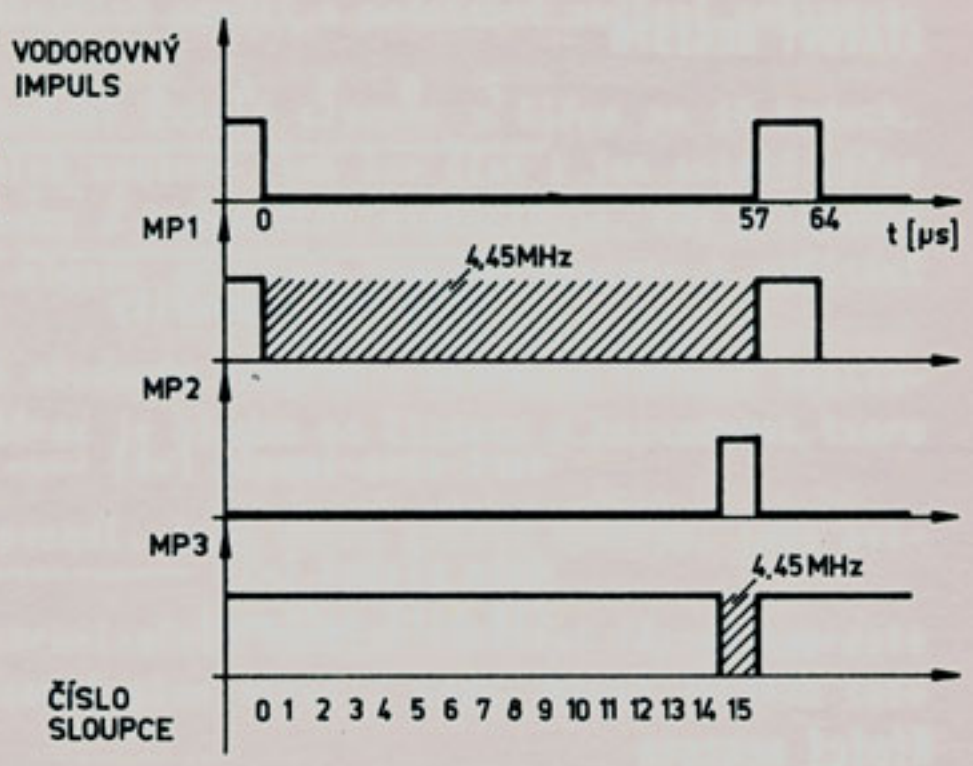
ŘÍDÍCÍ REGISTR



STAVBA A VYLAĎOVÁNÍ

Po sestavení digitizéru musíme vyladit frekvenci taktového oscilátoru. Osciloskop připojte na měřící místo 3, v němž dochází k překlápění podle horizontálního synchronizačního signálu. Signál taktu posuvného registru je ovládán synchronizačním impulsem v závislosti na výběru daného sloupce bodů. Nastavíme poslední sloupec a frekvenci taktu odladíme tak, aby signál na měřícím bodu 3 nebyl ještě střižen

Obr. 6 Diagram impulsů k vyrovnání oscilátoru takto:
Zde je sloupec č. 15 selektován, t.zn. 01...04 74LS373 jsou na úrovni log.1



následným synchronizačním impulsem. Frekvence taktu by pak měla činit cca 4.48 MHz (obr. 6).

Bez osciloskopu je nastavování možné jen tehdy, když máme software pro čtení a výstup grafiky již hotov. Pak se nastaví frekvence taktu tak, aby obraz dosáhl maximální šíře a nevykazoval interferenci nebo šikmé zkreslení.

V příští, poslední části se budeme věnovat programovému řízení našeho obrazového digitizéru.

mc 10/85

přeložil ing. KRAUS

ACORN RISC - je zisk

Acorn Computers Ltd. je jedna z nejúspěšnějších anglických počítačových firem. Ale jako mnoho dalších, i ona zažila své finanční problémy během deprese roku 1985. V roce 1979 ji založili dva odchovanci sira Sinclaira - Chris Curry a Hermann Hauser. V Cambridgi se společně pustili do výroby sad stavebnicových jednodeskových řídících jednotek s procesorem 6502. Tyto malé desky spojovali do komplexních průmyslových řídících systémů. V následujícím roce firma zahájila výrobu osobního, rozšiřitelného počítače Atom, vycházejícího ze zkušeností se systémy s CPU 6502. Po nějakou dobu byl Atom nejlevnějším počítačem v Anglii (za cca 200 Lstg). Získal mnoho následovníků, hlavně mezi těmi, kteří umějí zacházet s pájkou stejně dobře jako s assemblerem. Vylepšené Atomy najdete leckde ještě i dnes.

Další výrobek Acornu, zpočátku nazývaný Proton, byl vytvořen tak, aby splňoval vysoké požadavky konkursu BBC na osobní počítač pro potřeby vzdělávacího televizního seriálu. Soutěž se Sinclair Research, jejichž barevné Spectrum 48K tehdy již bylo na trhu, Acorn nakonec vyhrál.

Po dlouhém zdržení, zaviněném problémy s kvalitou ULA obvodů, byl počítač BBC

uveden na trh a zaváděn i do škol a univerzit. Z Acornu se stala velmi prosperující firma, jejíž roční obrat dosahoval 100 miliónů Lstg.

BBC Micro (přezdívaný Beeb, což je hovorový výraz pro BBC) je stále obstojným počítačem. V porovnání s četnými konkurenty má velmi slušnou grafiku. Jeho předností je i 10 MHz sběrnice, zvaná Tube, na niž mohou být připojeny další procesory. Acorn si za tuto výlučnost účtuje spoustu peněz - Beeb se držel na 400 Lstg ještě dlouho poté, co jeho konkurenti snížili své ceny pod 200 Lstg.

Acorn od začátku věnoval více pozornosti softwaru než většina ostatních výrobců. Pro své softwarové oddělení najímal nejlepší absolventy Cambridgské univerzity. Díky tomu měl Beeb k dispozici takový rozsah jazyků jako takřka žádný jiný počítač - rozvinutý strukturovaný BASIC, LISP, Logo, FORTH, Pascal, BCPL (Basic Combined Programming Language) a další. Ale přes všechny tyto klady měl Beeb jeden hlavní nedostatek - malou kapacitu paměti. Ambiciózní parametry, kombinované s omezenými adresovacími schopnostmi procesoru 6502, poskytovaly maximálně 32K pracovního prostoru (teprve v roce 1986 zvětšeno na 64K), který v grafickém módu s vyšší rozlišovací schopností klesal až na pouhých 8K. S tím se při práci s jazyky LISP či Logo nedostanete příliš daleko.

A tak v době své největší prosperity Acorn založil tým, který měl v tajnosti vytvořit vlastní procesor jako náhradu CPU 6502. Rozhodnutí, které se může jevit jako unáhlené a riskantní. Ale lidé od Acronu fandili jednoduchosti a rychlosti architektury 6502 tak, že pro ně bylo těžké přijmout kteroukoli z komerčně dostupných 16-bitových náhrad. Operační systém BBC je z velké části řízen přerušením. Pomalý reakční čas přerušení 16-bitových čipů jako Intel 8086 a Motorola 68000 by znamenal nutnost zavedení DMA (přímý přístup k paměti) - to by však přineslo některé nežádoucí komplikace. Po předchozí nabídce 3 MHz 6502 Acorn sáhl ještě po obvodu 32016 firmy National Semiconductor jako druhém procesoru pro Beeb. Ale pořád to nebylo "ono". Nakonec zvýtězila myšlenka vrhnout se na tvorbu vlastního procesoru Acorn RISC (Reduced Instruction Set Computer - počítač s redukovanou sadou instrukcí) Machine neboli ARM.

A R M

Tajemství ARM bylo skutečně dobře uchováno. Do oznámení v srpnu 1985, kdy již byl ARM vytvořen a k dispozici ve vzorcích, se o něm v počítačovém tisku neobjevilo ani slovo. Konstruktéři Acornu spolupracovali s americkou firmou VLSI Technology Inc., která jim dodala příslušný CAD (software pro návrh obvodu s použitím počítače), pracovní stanice a pak vyrobila i čipy. Tým Acornu měl sice již nějaké zkušenosti s VLSI obvody z práce na obvodu ULA pro BBC, ale žádnou praxi s navrhováním procesorů. Jejich ULA ovšem byla v té době jedním z nejkomplexnějších zákaznických obvodů, jaký kdy byl vyroben.

Za pouhých 18 měsíců tým Acornu navrhl ARM se vším všudy - a fungoval, jak měl, od prvního čipu. To není jen velký důkaz konstruktérské zručnosti Acornu, ale také potvrzením správnosti konstrukční filozofie RISC. Dosažení výkonu, jaký podává RISC Acornu, je při použití jakékoli ze současných kolosálních architektur (jako např. 68020) velmi nepravděpodobné.

Lidé v konstrukčním týmu Acornu byli inspirováni účinností zastaralého 6502 a ostatními výzkumnými týmy v oblasti obvodů RISC, jakými jsou třeba týmy ve Stanfordu

a Berkeley. Nicméně se v několika směrech odchýlili od dříve používaných technik. Konstrukčními cíli, které si uložili, byla vysoká prováděcí rychlost, malá a pravidelná sada instrukcí a velmi krátký reakční čas na přerušení (zvláště ve srovnání se současnými 16-bitovými čipy). Navíc tu byl požadavek, aby čip byl schopen podporovat virtuální paměťové operace. Protože Acorn zahájil práci v roce 1984, logicky šel přímo ke 32-bitovým procesorům, protože 16-bitová éra se již chýlila ke konci.

Navrhování ARM začalo zcela správně instrukční sadou místo hardwarem. Navržení, vychytání chyb, zkoušky čipu, to vše se provádělo softwarovou simulací. Některé z nich běžely na mikropočítačích BBC s druhým procesorem 6502 - bez jakéhokoli hardwarového prototypu. První vyrobené čipy byly také teprve první hardwarovou realizací projektu!

ARM používá výrazně linkovanou architekturu pro dosažení výkonu 3 MIPS (milión instrukcí za sekundu) z malého (7 mm čtverečních) čipu obsahujícího 25000 tranzistorů. Pro porovnání: čip 68020 s 9 mm čtverečními obsahuje 192000 tranzistorů a dosahuje cca 2,5 MIPS. Při časovacím ekvivalentu 5 MHz ARM provádí standardní testy v Basicu téměř desetkrát rychleji než IBM PC AT a značně rychleji než TDI Pinnacie se svým 12 MHz CPU 68000. První verze ARM používá celkem konzervativní 3-mikronovou CMOS (komplementární struktura MOS) a konstrukci s dvojitou kovovou úrovní. Potřebuje tak malý příkon, že se zahřívá pouze neznatelně. Celé zařízení je umístěno na čtvercovém 84-kolíkovém nosiči čipů Jedec.

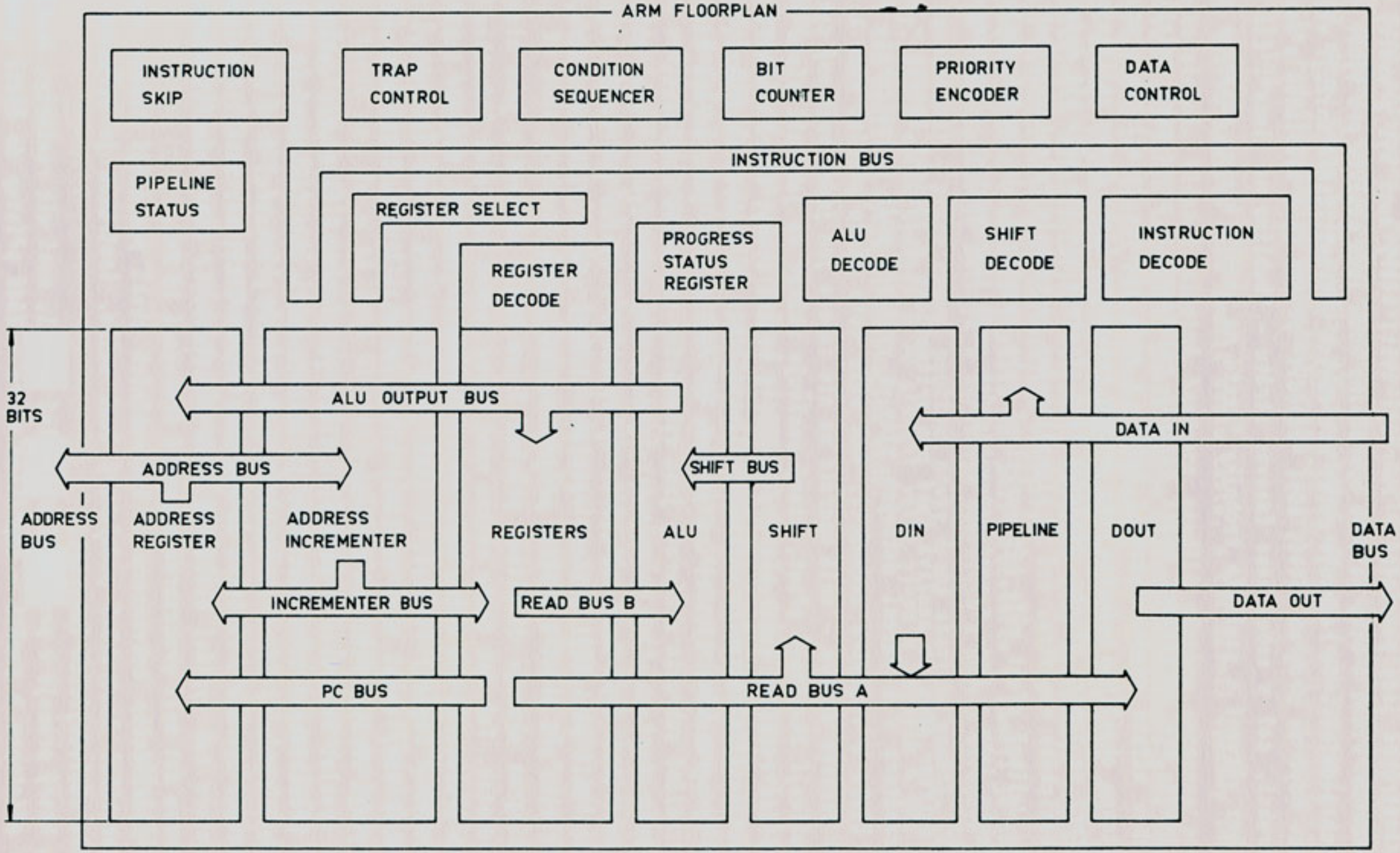
Další verze přejdou na 2,4 mikronu - tak budou čipy ještě menší. A menší čipy znamenají větší zisk. Acorn odhaduje, že jejich výroba bude více než čtyřikrát levnější než u megačipů ze sérií 80X86 a 680X0. Acorn tak směřuje k nabídce svého RISCu výrobcům levných osobních počítačů, stejně jako sektoru výzkumu umělé inteligence.

ARM má 25 plných 32-bitových registrů, 32-bitovou datovou a 26-bitovou adresovou sběrnici, která umožňuje adresovat 64 MB paměti. Programátorovi je normálně dostupných pouze 16 registrů. Během přerušení se extra registry stanou přístupné procesoru pro simulaci DMA kanálu bez nutnosti uchování kteréhokoli z uživatelských registrů. Registr 15 je programovým čítačem - ve svých prvních šesti bitech obsahuje stavové indikátory (RISC pro ně nemá zvláštní registr).

Všechny instrukce jsou 32-bitová slova, rozdělená do několika polí a vyvolatelná v jednom časovém cyklu. Všechny operace jsou prováděny ve 32-bitových blocích. Určité instrukce typu load-and-store (přenos dat) doplňují bajty nulami do 32 bitů při využití "barelového shifteru" čipu. Výhoda je v tom, že stejná délka zjednodušuje práci s instrukcemi a posloupnostmi. Potencionální nevýhodou je, že minimálně 4-bajtová instrukce generuje mnoho kódu. Ale vzhledem k dnešním cenám pamětí to není až tak důležité.

ARM má 44 základních instrukčních kódů, které můžeme rozdělit do pěti kategorií: load-and-store jednoduchého registru, load-and-store multiregistru, aritmetické a logické operace, skoky a instrukce softwarového přerušení. ARM nemá žádné instrukce násobení a dělení. Každý typ instrukce má několik polí - nastavením patřičných bitů můžete ze základní sady vytvořit velké množství různých instrukcí.

Podobně jako modely ze Stanfordu a Berkeley, i ARM má architekturu load-and-store. Pouze instrukce tohoto typu mají přístup k paměti. Všechny datové operace jsou prováděny "z registru do registru".



Blokový diagram 32-bitového průchodu dat čipem ARM a některé z jeho hlavních funkčních jednotek. Názvy byly výjimečně ponechány v angličtině, protože pro některé z nich je velmi těžké najít významově adekvátní český ekvivalent.

Všechny instrukce jsou podmíněné - zahrnují test, který musí být pravdivý před jejich provedením. První 4 bity každého operačního kódu jsou použity pro výběr jedné ze 16 možných podmínek. To redukuje počet potřebných větvení, které snižují efektivitu systému. Když se program větví, musí "odhodit" příští (již vyvolanou a dekódovanou) instrukci, což s sebou nese časovou ztrátu - "bublinu na lince". Lze psát programy i bez větvení s použitím techniky "testu na skok", kterou obsahuje každá instrukce.

ARM má pouhé dva adresovací módy: bázicky relativní a programově relativní. Nicméně z obou můžete snadno vytvořit další. V módu bázicky relativním pracujete buď s 12-bitovou okamžitou hodnotou nebo druhým registrem (jeho posun zajišťuje barelový shifter) ve funkci ofsetu. Výsledek této ofsetové operace může být dodatečně přepsán do bázického registru, což je signalizováno nastavením "přepisovacího" bitu v instrukci. Jelikož ofset může být pozitivní nebo negativní, je snadné dosáhnout stejného efektu jako u pre- a post-automatických módů inkrementace a dekrementace obvodu 68000.

Barelový shifter ARMu se používá také pro aritmetické a logické posuny - a bez zásahu programátora - k řazení datových slov a vydělení polí z instrukcí. Např. násobíme-li číslo v registru sedmnácti, ARM může toto číslo přičíst samo k sobě čtyřnásobným posuvem doleva - celá operace mu ale zabere jen jeden hodinový cyklus.

Skoky používají 24-bitový ofset, který umožňuje skákat na jakékoli místo v paměti. Nejsou zde rozdílné instrukce pro dlouhé a krátké skoky, které neušetří prostor ani čas. Jestliže nastavíte volitelný "linkový" bit ve skokové instrukci, obsah registru 15 (programového čítače) se přenese do registru 14 jako adresa návratu. Takže skoky, volání subrutin a návraty jsou vykonávány tou samou instrukcí.

Všechny instrukce mohou být provedeny v jednom hodinovém cyklu, vyjma multiregistrových instrukcí typu load-and-store. Ty vyžadují 1 cyklus pro každý registr. Tyto instrukce dále poskytují možnost rychlého uložení stavu celého procesoru, což zvyšuje efektivitu při práci s vyššími programovacími jazyky.

Blokový diagram ukazuje 32-bitový průchod čipem ARM a některé z jeho hlavních funkčních jednotek. Tok dat touto cestou není řízen jednou kontrolní jednotkou jako u konvenčních procesorů, ale řadou separátních funkčních jednotek. Dekodér instrukcí je programovatelný logický řetězec, v němž jsou instrukce pevně zakotveny. Není zde mikrokódová ROM, resp. není tu žádný mikrokód. Bity v aktuálních slovech instrukcí poskytují většinu řídících informací.

ARM může vyvolat instrukci z paměti, zatímco je předchozí instrukce dekódována a provedení předchůdkyně této instrukce je dokončováno v ALU (aritmetické logické jednotce). Tato situace, která maximalizuje průchodnost procesoru, trvá po tu dobu, dokud ARM provádí operace z registru do registru nepřetržitě, bez skoků. Architektura load-and-store tak vyplácí dividendy svou efektivností. Acorn naměřil maximální rychlost přenosu dat mezi procesorem a pamětí frekvencí 18 MHz.

Jednotky podmíněného sekvenceru a instrukčních skoků provádějí hlavní testy. Jestliže momentální instrukční podmínkový test selže, instrukce je vyřazena bez přerušení běhu následujících instrukcí.

Procesor, který pracuje v prostředí virtuální paměti, musí mít restarty. Jestliže instrukce s přístupem k paměti - například store - žádá přístup do části paměti,

která není přístupná, "paměťový manažer" nařídí její zrušení. Když procesor obdrží signál zrušení, musí provést restart, obnovit stav procesoru a provést některé další opravné akce. Kdyby bylo použito tzv. zpožděné větvení (model Berkeley - instrukce skoku je redefinována po provedení další instrukce), byla by uvedená operace hůře proveditelná.

Tým Acornu namísto toho zvolil cestu testů, a tak učinil všechny instrukce ARM restartovatelné. Nicméně samotný hardware neudělá všechnu opravnou práci; pouze konzervuje potřebnou informaci, takže teprve uživatelův software může znovunastavit stav procesoru za využití konzervované informace.

ARM dosáhl požadované krátké reakční doby přerušení částečně díky absenci nepřerušitelných multicyklových instrukcí, částečně díky systémovým registrům, které zbavují nutnosti uchovávat uživatelské registry.

Extrémně vysoká frekvence přenosu dat mezi procesorem a pamětí je dosažena díky širokým nemultiplexovým datovým a adresovým sběrnicím a docela skromným časem cyklu (150 nanosekund) - proto nemusí používat prvky drahé statické paměti. Čip má řídící signály, které mohou vydobýt o 30 procent větší výkon z levných 4 MHz dynamických pamětí RAM použitím cyklů stránkovacího módu.

SOFTWARE

Na rozdíl od mnoha nových čipů má ARM již docela dobré softwarové zázemí - např. překladač BBC Basicu, LISP compiler s celoobrazovkovým editorem, kompilátory pro BCPL a Modula-2. Pascal, C, FORTRAN a Prolog jsou ve výrobě pro potřeby vědců a výzkum umělé inteligence.

Obchodní oddělení Acornu, jedno z nově vytvořených poté, co Acorn převzal koncern Olivetti, je zodpovědné za návrh a výrobu ARM. Ale je značně skoupé na informace. Zatím není známo, v jakém výrobku bude nasazen první ARM. ACORN sdělil zatím jen tolik, že bude prodávat čipy jiným výrobcům, z nichž několik už "něco" připravuje.

ZÁVĚR

Acorn bezmála zkrachoval počátkem roku 1985, kdy Londýn stlačil cenu akcií (tedy i ceny zboží) k podlaze během povánočního výprodejního "masakru" počítačů. Italská společnost Olivetti firmu zachránila a nyní je nejvyšším držitelem jejích akcií. Tvrdí se, že v té době Olivetti ještě nic nevěděla o projektu ARM. Je-li to pravda, bylo později vedení Olivetti určitě příjemně překvapeno.

V ARM má Acorn/Olivetti jeden z prvních, navíc výjimečných komerčních RISC procesorů na světě. ARM by mohl přinést revoluční změny do oblasti levných domácích počítačů. Je překvapujícím potvrzením správnosti filozofie RISC v termínech výkonů, času potřebného k vývoji a nízkých výrobních nákladů.

BYTE 6/86

Přeložil R. PLETKA

Herbář nápadů a zkušeností

Jak naučit PMD 85 - 1 česky

Mikropočítač PMD 85-1 neumí zobrazovat písmena malé abecedy. V některých případech je navíc nezbytné mikropočítačem zpracovávat i české texty - jde o přípravu nejrůznějších písemností, sestavení jmenných seznamů řazených podle příjmení osob atd. Tato potřeba se projevuje zejména při spojení mikropočítače s tiskárnou, která v souboru znaků kompletní českou abecedu má. V našem případě to byl elektronický psací stroj.

Problém vyřešili před časem svazáci z městského výboru SSM v Praze. Jejich způsob je však chráněn přihláškou ZN a technické podrobnosti nejsou známy. Nabízíme vlastní, vyzkoušené řešení.

K němu stačí 4 kusy pamětí EPROM typu K573RF1 nebo MHB8708, možnost jejich naprogramování a kousek drátu. Tímto vybavením upravíte operační systém tak, že na obrazovce uvidíte texty psané v češtině, které pak můžete výstupním portem posílat do vhodné tiskárny - celý tento článek je tímto jednoduchým editorem zpracován a vypsán psacím strojem ROBOTRON S 6011. Texty je rovněž možné nahrát na mgf kazetu a archivovat.

Klávesnice: Asi těžko seženete jinou než má PMD. Psaní malých písmen umožňuje i stávající provedení se současným stiskem klávesy SHIFT, i když se zobrazují "čínské znaky". Chybějí však klávesy pro malá písmena české abecedy - ě, š, č atd. Aby rozdíly mezi psaním na psacích strojích a PMD byly co nejmenší, jsou znaky: ˇ, ě, š, č, ř, ž, ý, á, í, é, ú, ů přiřazeny klíčům K12 až K23 (tedy K0 až K11 se současným stiskem klávesy SHIFT). V důsledku toho není psaní sice úplně stejné jako u psacího stroje, ale je vyhovující. Nespornou předností je, že nemusíme vůbec nic na klávesnici měnit.

Kódování: Pro kompletní českou abecedu je určen kód KOI 8-čs, který je osmibitový. Ten v PMD nemůžeme využít, protože pro znaky a základní řídící příkazy je k dispozici jen sedmibitový kód ASCII. Zvolili jsme (nebyli jsme ničím vázáni) toto přiřazení (HEX):
ˇ 05, ě 06, š 07, č 08, ř 09, ž 10, ý 11, á 12, í 13, é 14, ú 15, ů 16.
Při tomto rozhodování musíte vzít v úvahu možnosti vaší tiskárny a nezapomeňte, že kódy 0A, 0D a 1C využívá monitor jako řídící (LF, CR, CLS).

Monitor: Originální monitor je ve čtyřech pamětech PROM v objímkách, s adresací (HEX) 8000 - 83FF č.I, 8400 - 87FF č.II, 8800 - 8BFF č.III, 8C00 - 8FFF č.IV. Generátor znaků je umístěn od adresy 8600 do 87FF, místo pro rozšíření o malá písmena

není. Proto použijeme další paměť, kterou naprogramujeme kompletní sestavou všech potřebných znaků (na 1 znak 8 bajtů, celkem tedy 1 KB). Zapojíme ji paralelně nasunutím na některou ze čtyř pamětí, kromě vývodu č. 20 - ten odehneme a zapojíme kouskem drátu k vývodu č. 11 dekodéru 3205 (je blíže k mikroprocesoru) - paměť bude aktivní na adresách 9000H - 93FFH. Správnou funkci monitoru zajistíme výměnou pamětí PROM č.II, III a IV za paměti EPROM naprogramované stejně jako původní, až na tyto změny (HEX):

paměť č.II od adr. 8451: 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16
paměť č.III adr. 8ABC: 90
paměť č.IV adr. 8FFB: 90

A to je všechno. Přeji příjemnou a užitečnou práci v mateřské řeči s mikropočítačem.

Ing. Švitorka

.....

ADR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
9010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
9020	FF	FF	FF	FF	FF	FF	FF	FF	0A	04	00	00	00	00	00	00
9030	0A	04	00	0E	11	1F	01	0E	0A	04	00	1E	01	0E	10	0F
9040	0A	04	00	1E	01	01	01	1E	0A	04	00	1A	06	02	02	02
9050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
9060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
9070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
9080	0A	04	00	1F	08	04	02	1F	08	04	11	11	11	1E	10	0C
9090	08	04	00	0E	10	1E	11	1E	08	04	00	06	04	04	04	0E
90A0	08	04	00	0E	11	1F	01	0E	08	04	00	11	11	11	11	1E
90B0	04	0A	04	11	11	11	11	1E	FF	FF	FF	FF	FF	FF	FF	FF
90C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
9100	00	00	00	00	00	00	00	00	04	04	04	04	04	00	04	00
9110	0A	0A	00	00	00	00	00	00	0A	0A	1F	0A	1F	0A	0A	00
9120	0E	15	15	0E	14	15	0E	00	03	13	08	04	02	19	18	00
9130	04	0A	0A	06	15	09	16	00	04	04	02	00	00	00	00	00
9140	08	04	02	02	02	04	08	00	02	04	08	08	08	04	02	00
9150	00	04	15	0E	15	04	00	00	00	04	04	1F	04	04	00	00
9160	00	00	00	00	02	02	01	00	00	00	00	1F	00	00	00	00
9170	00	00	00	00	00	00	02	00	00	10	08	04	02	01	00	00
9180	0E	11	19	15	13	11	0E	00	04	06	04	04	04	04	0E	00
9190	0E	11	10	0C	02	01	1F	00	1F	10	08	1C	10	11	0E	00

```
91A0    08 0C 0A 09 1F 08 08 00 1F 01 0F 10 10 11 0E 00
91B0    1C 02 01 0F 11 11 0E 00 1F 10 08 04 02 02 02 00
91C0    0E 11 11 0E 11 11 0E 00 0E 11 11 1E 10 08 07 00
91D0    00 00 00 00 02 00 02 00 00 00 04 00 04 04 02 00
91E0    08 04 02 01 02 04 08 00 00 00 1F 00 1F 00 00 00
91F0    02 04 08 10 08 04 02 00 0E 11 10 08 04 00 04 00
9200    06 11 19 15 1D 01 0E 00 04 0A 11 11 1F 11 11 00
9210    0F 11 11 0F 11 11 0F 00 CE 11 01 01 01 11 0E 00
9220    0F 12 12 12 12 12 0F 00 1F 01 01 0F 01 01 1F 00
9230    1F 01 01 0F 01 01 01 00 0E 01 01 01 19 11 1E 00
9240    11 11 11 1F 11 11 11 00 0E 04 04 04 04 04 0E 00
9250    10 10 10 10 11 11 0E 00 11 09 05 03 05 09 11 00
9260    01 01 01 01 01 01 0F 00 11 1B 15 15 11 11 11 00
9270    11 11 13 15 19 11 11 00 0E 11 11 11 11 11 0E 00
9280    0F 11 11 0F 01 01 01 00 0E 11 11 11 15 09 16 00
9290    0F 11 11 0F 05 09 11 00 0E 11 01 0E 10 11 0E 00
92A0    1F 04 04 04 04 04 04 00 11 11 11 11 11 11 0E 00
92B0    11 11 11 0A 0A 04 04 00 11 11 11 15 15 15 0A 00
92C0    11 11 0A 04 0A 11 11 00 11 11 0A 04 04 04 04 00
92D0    1F 10 08 04 02 01 1F 00 07 01 01 01 01 01 07 00
92E0    00 01 02 04 08 10 00 00 1C 10 10 10 10 10 1C 00
92F0    0E 11 00 00 00 00 00 C0 00 00 00 00 00 00 1F 00
9300    02 04 08 00 00 00 00 00 00 00 0E 10 1E 11 1E 00
9310    01 01 0F 11 11 11 0F 00 00 00 1E 01 01 01 1E 00
9320    10 10 1E 11 11 11 1E 00 00 00 0E 11 1F 01 0E 00
9330    08 04 04 0E 04 04 04 02 00 00 1E 11 11 1E 10 0C

.........

9340    01 01 0F 11 11 11 11 00 04 00 06 04 04 04 0E 00
9350    08 00 08 08 08 08 09 06 02 02 12 0A 06 0A 12 00
9360    06 04 04 04 04 04 0E 00 00 00 0B 15 15 15 15 00
9370    00 00 0F 11 11 11 11 00 00 00 0E 11 11 11 0E 00
9380    00 00 0F 11 11 0F 01 01 00 00 1E 11 11 1E 10 10
9390    00 00 1A 06 02 02 02 00 00 00 1E 01 0E 10 0F 00
93A0    00 04 0E 04 04 04 08 00 00 00 11 11 11 11 1E 00
93B0    00 00 11 11 0A 0A 04 00 00 00 11 11 15 15 0A 00
93C0    00 00 11 0A 04 0A 11 00 00 00 11 11 11 1E 10 0C
93D0    00 00 1F 08 04 02 1F 00 08 04 04 02 04 04 08 00
93E0    04 04 04 00 04 04 04 00 02 04 04 08 04 04 02 00
93F0    0C 10 0E 01 00 C0 00 00 0A 15 0A 15 0A 15 0A 15
```

Úprava obrázků ze ZX-Spectra pro IQ 151

Od začátku roku 1987 je dodáván jako příslušenství k IQ151 modul GRAFIK. Tento modul doplňuje počítač o grafiku 512x256 bodů. Pro práci s modulem je dodáván BASIC G, což je původní BASIC 6 rozšířený o 17 grafických příkazů. Modul neslouží k zobrazování textů, takže spolu s ním musíme použít videopaměť. Použitá VIDEORAM pracuje nezávisle na modulu GRAFIK.

S modulem lze pracovat ve strojovém kódu za použití instrukcí IN a OUT. Pro styk s počítačem obsahuje modul jeden obvod 8255 a jeden 3212. Z toho plyne, že nejprve musíme inicializovat obvod 8255:

```
MVI A 8Ø
OUT D3
```

Modul pracuje ve dvou režimech - přístupu bitovém a bajtovém. Jak plyne z názvů, režimy se liší způsobem adresování obrazových elementů. Volba režimu se provádí na portu D2. Jednotlivé bity D2 plní tyto funkce:

bit 7 - spolupráce GRAFIK-8Ø8Ø: Ø = zakázána; 1 = povolena
bity 6-4 - rezervovány pro řízení barvy
bit 3 - výstup videosignálu: Ø = obrázek zakázán; 1 = povolen
bit 2 - priorita: Ø = procesor; 1 = video
bit 1 - bod: Ø = svítí; 1 = nesvítí
bit Ø - přístup: Ø = bitový; 1 = bajtový

Hodnota souřadnice y se zadává na port D1, souřadnice x na DØ (bajtový přístup), nebo na DØ a D4 (bitový přístup). Při bajtovém přístupu je D4 bajt na [x,y].

Pro převod grafiky ze Spectra je použit bajtový přístup. Program postupně načítá jednotlivé bajty do střadače a "rozdvojuje" je do registrového páru BC. Takto upravená data vysílá do modulu GRAFIK. Samozřejmě bylo nutné počítat s rozdělením videoram Spectra na třetiny, takže se program skládá z několika do sebe vnořených částí.

Program jsem umístil od adresy 2ØØ, ale pomocí některého z unifikované řady assemblerů si jej lze přeložit kamkoli do volné RAM.

Pokud máme program úspěšně přeložen, a např. na adrese 2ØØØ pomocí "přetahováku" umístěnu "obrazovku" ze ZX-Spectra, můžeme v režimu MONITOR zavelet C2ØØ. Zadáme adresu 2ØØØ a po potvrzení se nám obrázek vyjeví v celé své černobílé kráse. Jelikož si program nevšímá paměti atributů, vypadá obrázek tak, jako když na Spectru nastavíme pro celou obrazovku PAPER černý a INK bílý.

Doufám, že se můj program stane vítaným pomocníkem všem uživatelům IQ151.

- autor neznámý -

PŘEVOD ZX OBRAZOVKY NA MODUL GRAFIK

```
   Skok na začátek vlastního programu
EX Ø2ØØ C3 ST      JMP ST
TABULKA MASEK PRO PŘEVOD JEDNOTLIVÝCH BAJTŮ
MA Ø2Ø3 FC Ø3 F3 ØC CF 3Ø 3F CØ
```

```
INICIALIZACE UKAZOVATKA A ČÍTAČE
SI Ø2ØB 21 MA     LXI H MA          ; do HL adresa tabulky masek
   Ø2ØE 2B         DCX H             : zmenšená o 1
   Ø2ØF 16 Ø4      MVI D Ø4  D       ; počáteční stav čítače bitů
TEST BITU JE-li 1, NULUJ ODPOVÍDAJÍCÍ BIT V B
B  Ø211 Ø7         RLC               ; přesuv bitu do CY
   Ø212 F5         PUSH PSW          ; uložit střadač
   Ø213 DA A1      JC A1             ; test bitu
   Ø216 23         INX H             ; nastavení ukazatele na odpovídající masku
   Ø217 78         MOV A,B           ;
   Ø218 Aô         ANA M             ; nulování odpovídajících bitů ve výsledku
   Ø219 23         INX H             ; nastavení ukazatele na novou masku
   Ø21A C3 AF      JMP AF            ; skok na konec cyklu
BIT JE Ø.  NASTAV ODPOVÍDAJÍCÍ BITY V B
A1 Ø21D 23         INX H             ;
   Ø21E 23         INX H             ; nastavení odpovídající masky
   Ø21F 78         MOV A,B           ;
   Ø22Ø B6         ORA M             ; nastavení odpovídajících bitů ve výsledku
TEST, JE-LI DOKONČEN PŘEVOD PRVNÍCH ČTYŘ BITŮ
AF Ø221 47         MOV B,A           ; uschování výsledku
   Ø222 F1         POP PSW           ; návrat zpracovávaného bajtu
   Ø223 15         DCR D             ; dekrementace čítače
   Ø224 C2 B       JNZ B             ; není-li převod skončen, převádí dále
ZNOVUNASTAVENÍ REGISTRU HL a D
C  Ø227 21 MA     LXI H MA          ; do HL adresa tabulky masek
   Ø22A 2B         DCX H             ; zmenšená o 1
   Ø22B 16 Ø4      MVI D Ø4  D       ; počáteční stav čítače bitů
TESTUJE BITY V DRUHÉ ČÁSTI
A PROVÁDÍ JEJICH NULOVÁNÍ
CX Ø22D Ø7         RLC               ; přesun bitu do CY
   Ø22E F5         PUSH PSW          ; uložení zpracovávaného bajtu
   Ø22F DA B1      JC B1             ; test bitu
   Ø232 23         INX H             ; nastavení masky
   Ø233 79         MOV A,C           ;
   Ø234 A6         ANA M             ; nulování odpovídajících bitů ve výsledku
   Ø235 23         INX H             ; nová maska
   Ø236 C3 BF      JMP BF            ; konec převodu
PROVÁDÍ NASTAVENÍ BITU V D REGISTRU
B1 Ø239 23         INX H             ;
   Ø23A 23         INX H             ; ukazatel na žádanou masku
   Ø23B 79         MOV A,C           ;
   Ø23C B6         ORA M             ? nastavení odpovídajících bitů
TESTUJE KONEC PŘEVODU
BF Ø23D 4F         MOV C,A           ; uschování výsledku
   Ø23E F1         POP PSW           ; návrat zpracovávaného bajtu
```

```
   Ø23F 15        DCR D           ; dekrementace čítače bitů
   Ø24Ø C2 CX     JNZ CX          ; není-li převod skončen, převáděj dále
   Ø243 C9        RET             ; návrat z podprogramu

PROVÁDÍ INICIALIZACI MODULU GRAFIK
ST Ø244 3E 8Ø     MVI A 8Ø C @    ; maska pro CWR 8255
   Ø246 D3 D3     OUT D3  S       ; na odpovídající port
   Ø248 3E 87     MVI A 87 G      ; maska pro nastavení funkce modulu GRAFIK
   Ø24A D3 D2     OUT D2  R       ; na port D2
   Ø24C 21 FF 3F  LXI H 3FFF      ; počáteční stav čítače souřadnic
SMAZÁNÍ GRAFIKU, ZADÁNÍ POČÁTEČNÍ ADRESY PŘEVODU
Ø1 Ø24F 7D        MOV A,L         ; y-ová souřadnice do střadače
   Ø25Ø D3 D1     OUT D1  Q       ; nastavení této souřadnice v modulu GRAFIK
   Ø252 7C        MOV A,H         ; x-ová souřadnice do střadače
   Ø253 D3 DØ     OUT DØ  P       ; vyslání do modulu GRAFIK
   Ø255 97        SUB A           ; nulování střadače
   Ø256 D3 D4     OUT D4  T       ; nulování bajtu na souřadnicích x,y
   Ø258 2D        DCR L   ,       ; nová souřadnice y
   Ø259 7D        MOV A,L         ;
   Ø25A FE FF     CPI FF  ●       ; test konce jednoho sloupce
   Ø25C C2 Ø1     JNZ Ø1          ; není-li konec, pokračuj v mazání sloupce
   Ø25F 25        DCR H           ; nová souřadnice x
   Ø26Ø 7C        MOV A,H         ;
   Ø261 FE FF     CPI FF  ●       ; test konce mazání
   Ø263 C2 Ø1     JNZ Ø1          ; není-li smazáno, pokračuj v mazání
   Ø266 3E 8F     MVI 8F    C     ; maska pro nastavení funkce modulu GRAFIK
   Ø268 D3 D2     OUT D2  R       ; na port D2
   Ø26A 21 TX     LXI H TX        ; adresa textu do HL
   Ø26D CD 88 F4  CALL F488       ; tisk textu
   Ø27Ø CD C3 F4  CALL F4C3       ; vstup H' z klávesnice
   Ø273 E5        PUSH H          ; uložit do zásobníku
   Ø274 CD 47 F6  CALL F647       ; smazání videoprogram
   Ø277 1F                        ; kód pro smazání
   Ø278 21 8ØEF   LXI H EF8Ø      ; adresa kurzoru
   Ø27B 22 ØC ØØ  SHLD ØØØC       ; nastavena
   Ø27E E1        POP H           ; návrat adresy převodu
   Ø27F 11 FF ØØ  LXI D ØØFF      ; souřadnice x,y
POSTUPNĚ ZOBRAZUJE TŘI TŘETINY OBRAZU, NÁSLEDUJE NÁVRAT DO MONITORU
SM Ø282 CD 13     CALL 13         ; zobrazení horní třetiny obrázku
   Ø285 1E BF     MVI E BF  ?     ; y-ová souřadnice střední třetiny
   Ø287 CD 13     CALL 13         ; zobrazení střední třetiny
   Ø28A 1E 7F     MVI E 7F  ◆     ; y-ová souřadnice dolní třetiny
   Ø28C CD 13     CALL 13         ; zobrazení dolní třetiny
   Ø28F C9        RET             ; návrat do monitoru
ZOBRAZÍ JEDEN BODOVÝ ŘÁDEK
A JE POČET BAJTŮ V ŘÁDKU
```

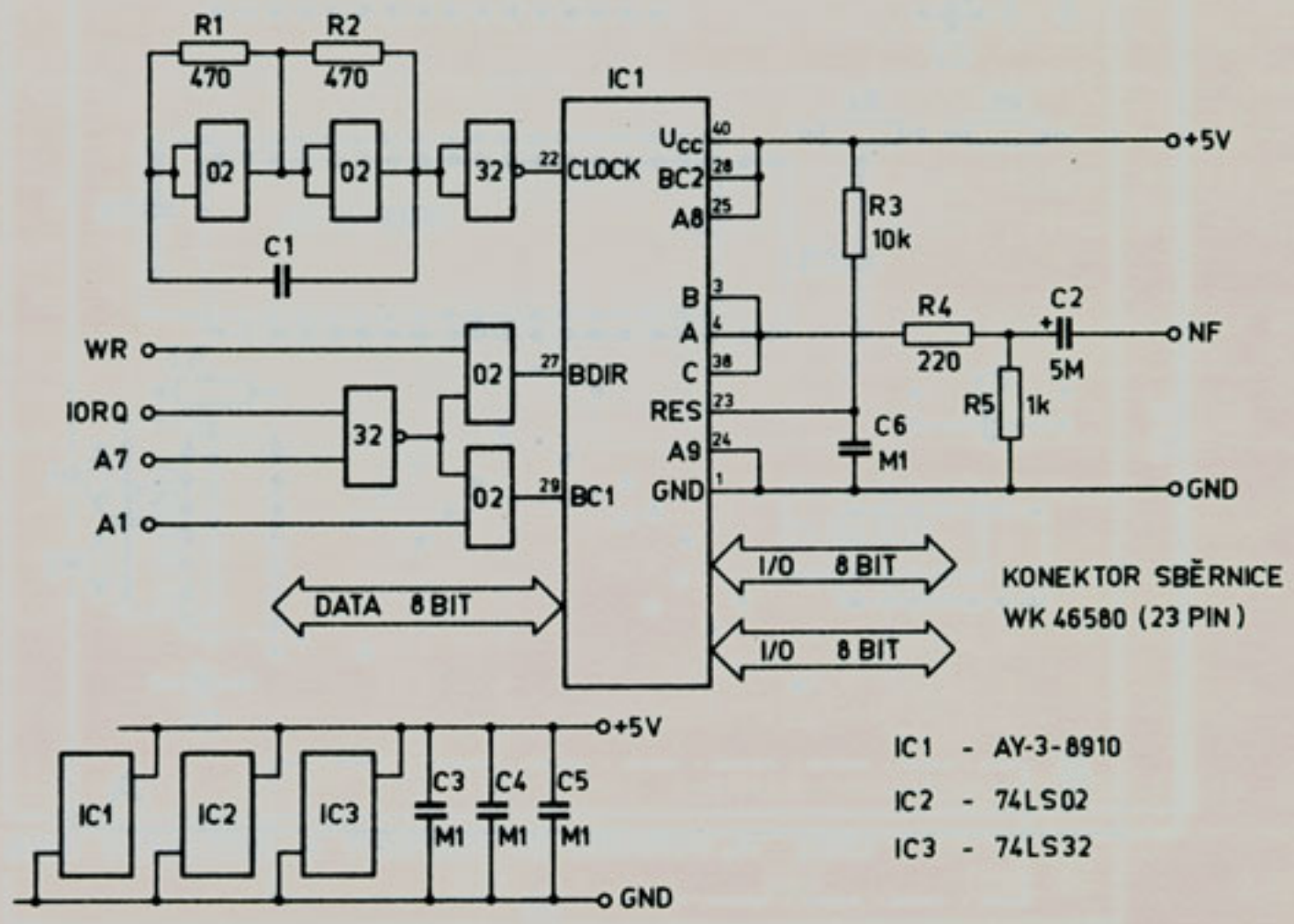
```
OR 0290 3E 20     MVI A 20        ; počet bajtů
   0292 16 00     MVI D 00        ;
POSTUPNĚ PROVÁDÍ EXPANZI JEDNOTLIVÝCH BAJTU A UMISŤUJE JE DO MODULU GRAFIK
1R 0294 F5        PUSH PSW        ;
   0295 7E        MOV A,M         ; vyzvednutí originálu obrazového bajtu
   0296 E5        PUSH H          ; uložení adresy
   0297 D5        PUSH D          ; uložení souřadnic x,y
   0298 CD SI     CALL SI         ; převod jednoho bajtu
   029B D1        POP D           ; návrat souřadnic
   029C 7B        MOV A,E         ; nastavení y-ové souřadnice v modulu
   029D D3 01     OUT D1 Q        ;
   029F 7A        MOV A,D         ; nastavení x-ové souřadnice v modulu
   02A0 D3 D0     OUT D0  P       ;
   02A2 78        MOV A,B         ; 1, bajt výsledku do střadače
   02A3 D3 D4     OUT D4  T       ; a do modulu GRAFIK
   02A5 14        INR D           ; x= x + 1
   02A6 7A        MOV A,D         ; x-ová souřadnice do střadače
   02A7 D3 D0     OUT D0  P       ; a do modulu GRAFIK
   02A9 79        MOV A,C         ; 2.bajt výsledku do střadače
   02AA D3 D4     OUT D4  T       ; a do modulu GRAFIK
   02AC 14        INR D           ; x= x + 1
   02AD E1        POP H           ; návrat adresy
   02AE 23        INX H           ; inkrementace adresy
   02AF F1        POP PSW         ;
   02B0 3D        DCR A           ;
   02B1 C2 1R     JNZ 1R          ; test konce zobrazování jednoho bodového řádku
   02B4 C9        REI             ; návrat z podprogramu
VYKRESLÍ JEDNU TŘETINU OBRÁZKU
13 02B5 3E 08     MVI A 08  H     ; počet znakových řádků

HG 02B7 F5        PUSH PSW        ;
   02B8 CD  8     CALL 8          ; zobrazení osmi bodových řádků
   02BB 7B        MOV A,E         ; y-ová souřadnice do střadače
   02BC C6 3F     ADI 3F  ?       ; posuv na další bodový řádek
   02BE 5F        MOV E,A         ; souřadnice do registru E
   02BF F1        POP PSW         ;
   02C0 3D        DCR A           ; další znakový řádek
   02C1 C2 HG     JNZ HG          ; test konce třetiny
   02C4 C9        RET             ; návrat z podprogramu
VYKRESLÍ 8 BODOVÝCH ŘÁDKŮ JAKO NA SPECTRU
 8 02C5 3E 08     MVI A 08  H     ; počet bodových řádků
NU 02C7 F5        PUSH PSW        ;
02C8 CD OR        CALL OR         ; tisk jednoho bodového řádku
   02CB 7B        MOV A,E         ; y-ová souřadnice do střadače
   02CC D6 08     SUI 08  H       ; posuv o 8 dolů
   02CE 5F        MOV E,A         ; y-ová souřadnice do registru E
```

```
   02CF F1        POP PSW          ;
   02D0 3D        DCR A            ; další bodový řádek
   02D1 C2 NU     JNZ NU           ; test konce zobrazování
   02D4 C9        RET              ; návrat z podprogramu
TABULKA TEXTU
TX 02D5 ZADEJ ADRESU: ■
99 02E4 00        NOP
```

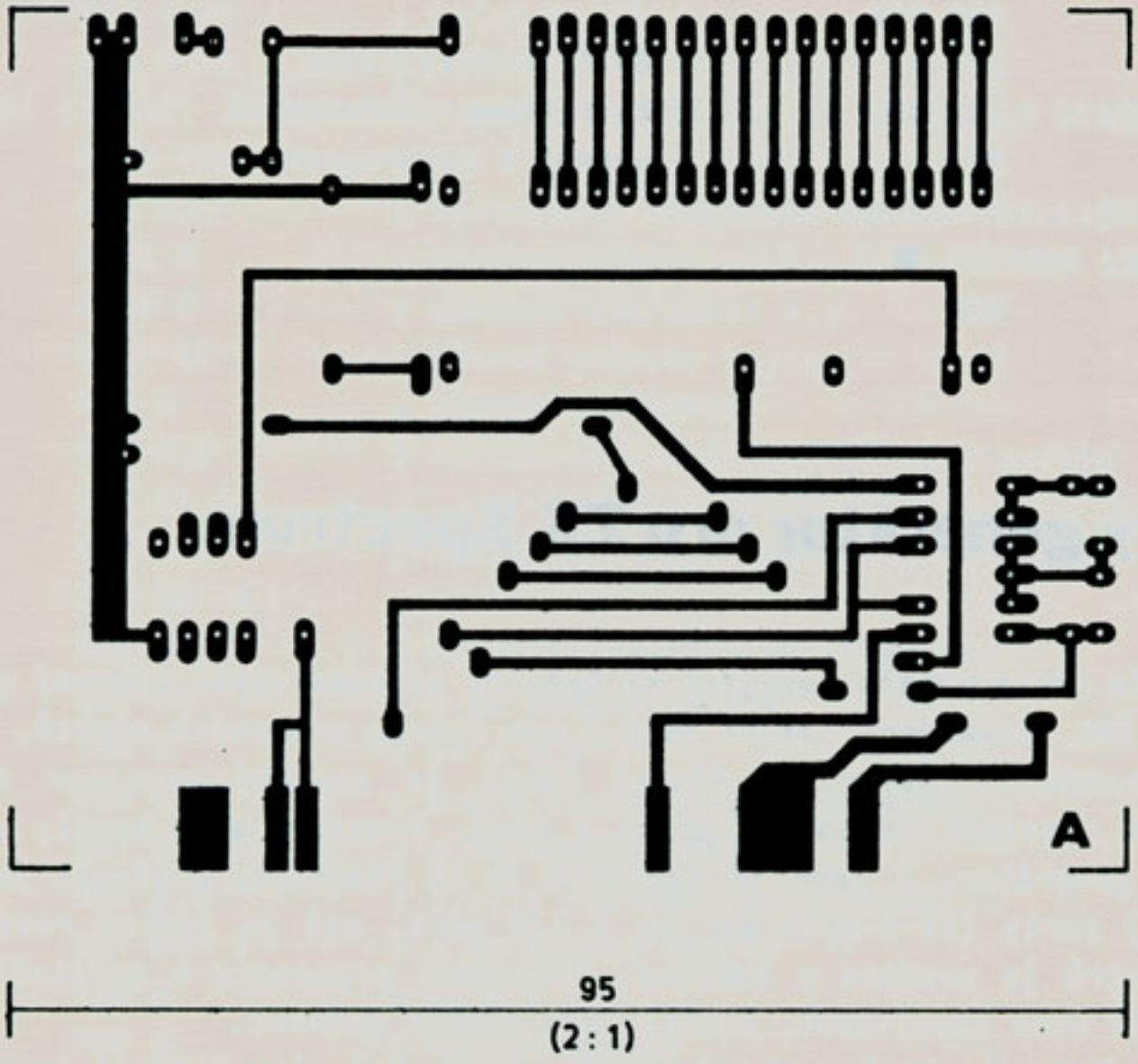
Zvukový generátor pro ZX Spectrum

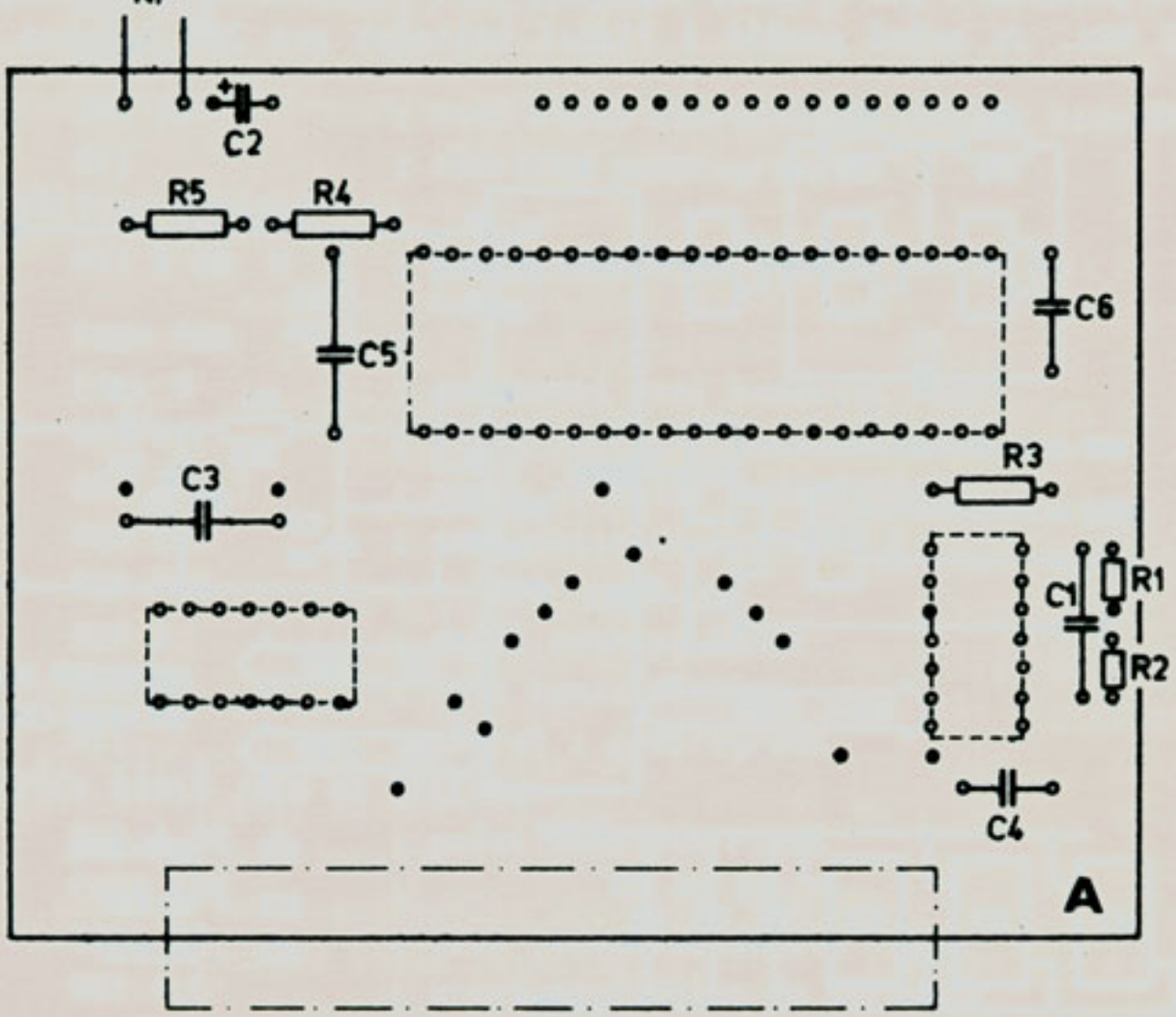
Připojením popisovaného zapojení ke Spectru získáme velmi výkonný generátor (jako u Spectra 128K, Commodoru, Atari apod.) pro tvorbu zvukových efektů, resp. simulaci zvukových hudebních nástrojů. Hlavní částí je zvukový procesor AY-3-8910 nebo AY-3-8912. V zapojení je použit první z nich. Procesor má 3 kanály pro tvorbu tónů a 1 šumový generátor. Z jeho 16 registrů je 14 řídících, 2 slouží jako vstupně/výstupní (I/O) porty (AY-3-8912 má jen 1 I/O port, což je jediný rozdíl mezi nimi). Funkce registrů je uvedena v tabulce. Připojení procesoru k počítači je možné buď přes PIO 8255 nebo samostatně. Zvolil jsem druhou možnost.

Dekodér adresy je tvořen obvody 7432 a 7402. Polovina 7402 je generátor řídícího kmitočtu pro procesor. Dvě hradla, zapojená jako multivibrátor, kmitají na frekvenci přibližně 2 MHz. Je možné použít i řídící kmitočet pro Z80 Spectra (je vyveden na

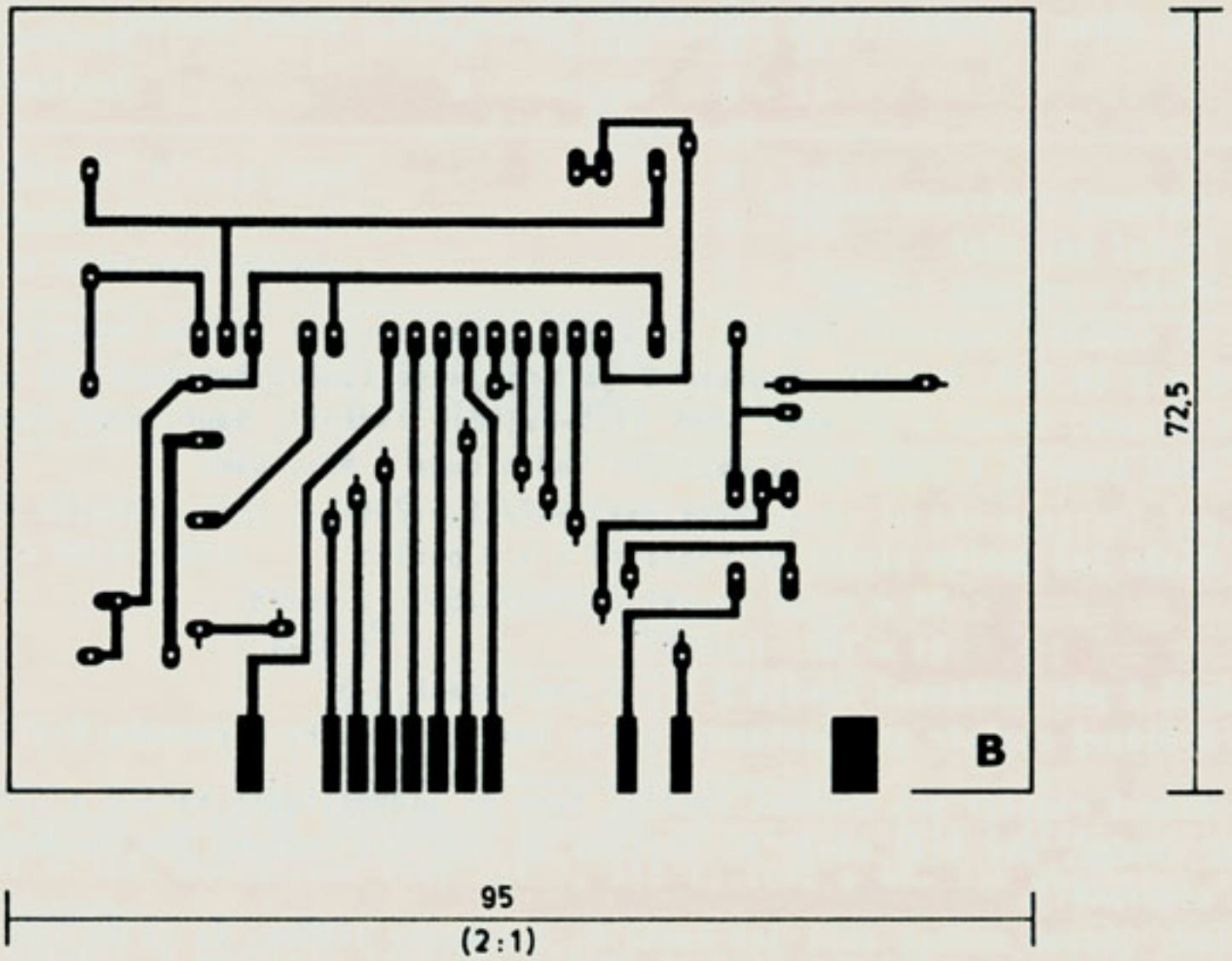


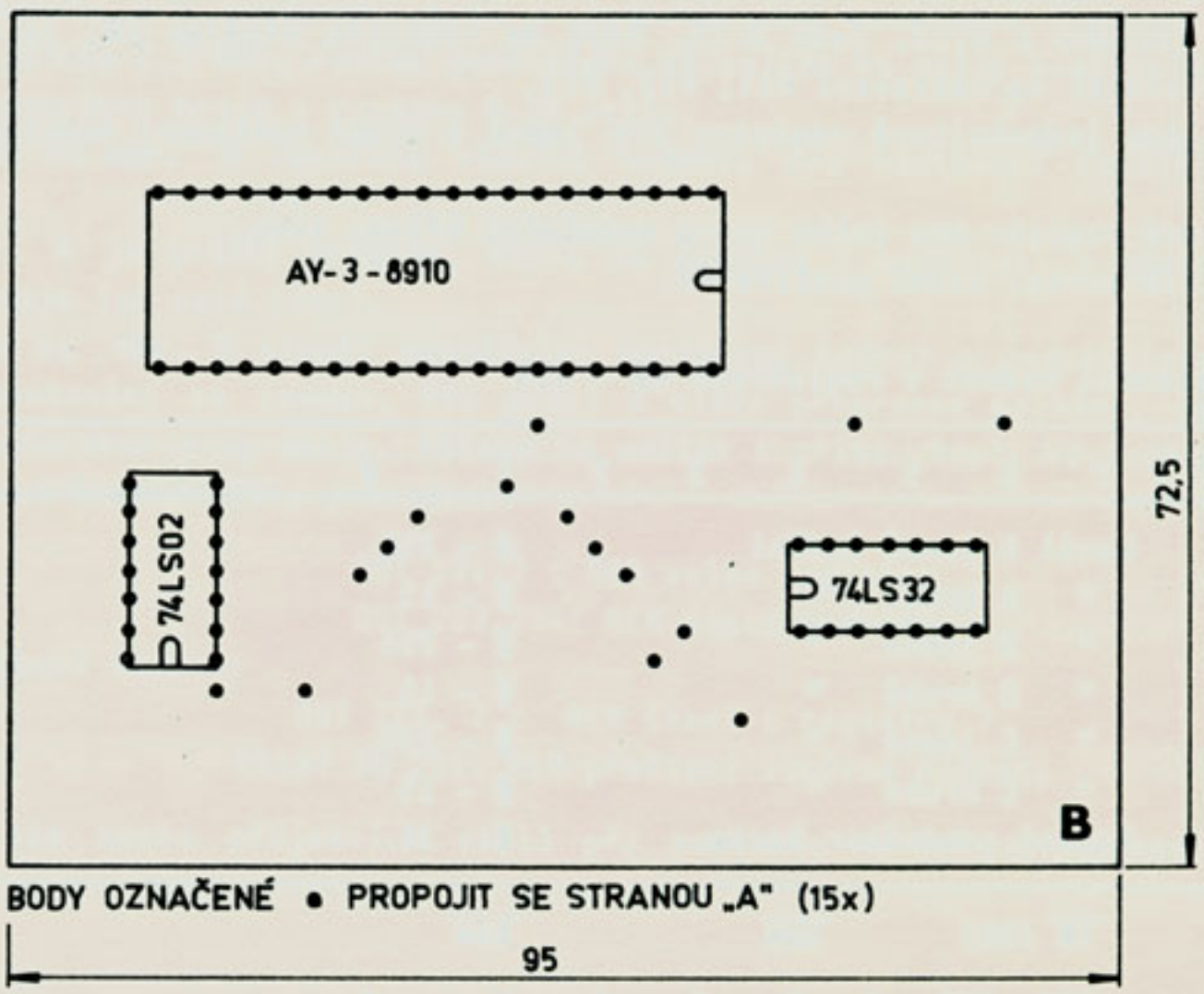
Obr. 1





Obr. 2a Zapojení plošného spoje - strana "A"





Obr. 2b Zapojení plošného spoje - strana "B"

sběrnici), ale musíme počítat s tím, že ULA "občas" hodiny procesoru blokuje. Řídící kmitočet je dělen v jednotlivých registrech kanálů A,B,C podle vzorce:

$$Fk = \frac{Fh}{16*n}$$

Fk - výstupní frekvence kanálu

Fh - řídící kmitočet

n - číslo v rozsahu 1-4096

R0 - z tabulky registrů vidíme, že kanál A nastavujeme změnou hodnoty v registrech 0 a 1 (B 2-3, C 4-5). Registr 0 je pro jemné ladění 8-bitové, R1 pro hrubé ladění 4-bitové. Proto n je max. 4096 (1,5-bajtové ladění). Např. když chceme, aby kanál A hrál komorní A (440 Hz) a kmitočet hodin bude 2 MHz, pak:

n = 2000*10**3/440/16 = 284

Toto číslo teď musíme vložit do registrů 1 a 0 podle:

```
R0 = n-INT (n-INT (n/256)*256)  ; R0=28
R1 = INT  (n/256)               ; R1=1
```

To platí i pro nastavení kanálů B a C v registrech 3-5.

R6 - 5-bitový registr pro nastavení barvy šumu (dekadicky v rozsahu 0-31).

R7 - hlavní registr pro míchání kanálů, šumu do jednotlivých kanálů a I/O zařízení. Když chceme, aby hrál kanál A, vložíme do registru hodnotu 254 (hodnoty se nastavují negovaně!).

Příklad: Chceme nechat znít jen kanál A. Z tabulky vidíme, že je to bit 0, který má dekadickou hodnotu 1. Pak hodnota vložená do R7 bude 255-1=254.

R8, R9, R10 - nastavování hlasitosti a volba modulační křivky tónu. Hlasitost nastavujeme v rozsahu 4 bitů, tedy 0-15. Když do registru vložíme 16, je umožněna modulace v závislosti na nastavení R11, R12, R13.

R11, R12 - dva jednobajtové registry pro jemné a hrubé nastavení modulace podle křivky v R13.

R13 - 4-bitový registr pro volbu tvaru modulační křivky tónu.

R14, R15 - I/O porty, volně použitelné.

Tabulka nastavení hodnot v registrech RØ - R15

REG.	BIT - 7	6	5	4	3	2	1	Ø
RØ	kanál A jemné ladění							Ø - 255
R1	" A hrubé "							Ø - 15
R2	" B jemné "							Ø - 255
R3	" B hrubé "							Ø - 15
R4	" C jemné "							Ø - 255
R5	" C hrubé "							Ø - 15
R6	šumový generátor barva							Ø - 31
R7 Mix	$\overline{IN}$ /OUT			$\overline{sum}$			$\overline{ton}$	
	IO1	IO2	C	B	A	C	B	A

R8	hlasitost	M	rozsah	Ø - 15
R9	hlasitost	M	rozsah	Ø - 15
R1Ø	hlasitost	M	rozsah	Ø - 15
R11	modulační křivka jemně			Ø - 255
R12	modulační křivka hrubě			Ø - 255
R13	Tvar modulační křivky			Ø - 15
R14	8 bitový I/O port			
R15	8 bitový I/O port			

Dále je důležité znát způsob zápisu a čtení procesoru. To nám prozradí log. stavy na vývodech BDIR a BC1:

Funkce	BDIR	BC1
čtení	0	1
zápis	1	0

Z toho vychází poměrně jednoduchý dekodér adresy pro zápis i čtení. Zvolil jsem adresu 221 a 223. Dekódujeme tedy bit 7, bit 1, IORQ a WR. Tak obsáhneme všechny kombinace tabulky.

Pro sestavení a kontrolu zapojení můžeme zvukový generátor připojit ke Spectru a programovat. Kdo má možnost, změří si ještě předtím frekvenci hodin, z níž pak odvodí hodnoty pro nastavení tónů v jednotlivých kanálech. Frekvence v uvedeném zapojení je cca 2,2 MHz. Výstup generátoru připojíme na vstup kvalitního zesilovače, protože generátor hraje v rozsahu 8-10 oktáv.

Program na **obr. 3** vypočítá všechny hodnoty pro nastavení kanálů v rozsahu 8 oktáv.

Na **obr. 4** je krátký program, který ze Spectra v kombinaci s generátorem udělá klavír (hraje jen celé tóny).

Program na **obr. 5** slouží manuálnímu nastavení hodnot v jednotlivých registrech a experimentování se zvukem. Při práci s tímto programem poznáme, jak se vlastně zvukový generátor programuje. Můžeme to udělat i přímo. Např. když chceme do registru 7 zapsat hodnotu 254, provedeme:

```
OUT 221,7 : REM volba registru
OUT 223,254: REM zápis hodnoty
```

A pro čtení hodnot:

```
OUT 221, 7 :  REM čtení z registru 7
PRINT IN 221: REM přečtení vložené hodnoty
```

Přeji vám mnoho příjemných chvil při poslechu nových a zajímavých zvukových efektů.

G. JORDANOV

```
  10 BORDER 6: PAPER 6: INK 0: CLS : POKE 23658,8
  15 LET t$="C C#D D#E F F#G G#A A#H "
  20 INPUT "F-OSCILATORA=? (KHz) ";f
  25 LET pc=0: FOR n=0 TO 7: PRINT ''' INVERSE 1; BRIGHT 1;"PC
NOTA/OKT FREK   L-BYT   H-BYT": PRINT  PAPER 3; INK 7; INVERSE
1;"                                    "
  30 FOR m=1 TO 24 STEP 2: LET pc=pc+1: READ t
  50 LET x=(f*1e3)/t/16
  60 LET x1=INT (x-INT (x/256)*256): LET xh=INT (x/256)
  70 LET h=xh: GO SUB 200: LET j$=h$: LET h=x1: GO SUB 200: LET
 k$=h$
  80 PRINT pc;TAB 4;t$(m TO (m+1));"-";n;TAB 12;t;TAB (21-LEN (
STR$ x1));x1;"/";k$;TAB (29-LEN (STR$ xh));xh;"/";j$
  90 NEXT m: PRINT  BRIGHT 1;"                                    "
: PRINT #1;TAB 10; FLASH 1;"NIECO STLAC!": PAUSE 0: CLS : NEXT
n: PRINT AT 10,11; FLASH 1;" OPAKOVAT ?": INPUT x$: IF x$="A" T
HEN  CLS : RESTORE 100: GO TO 25
  95 RUN
 100 DATA 16.4,17.3,18.4,19.4,20.6,21.8,23.1,24.5,26.0,27.5,29.
1,30.9
 110 DATA 32.7,34.6,36.7,38.9,41.2,43.7,46.2,49,51.9,55,58.3,61
.7
 120 DATA 65.4,69.3,73.4,77.8,82.4,87.3,92.5,98.0,103.8,110.0,1
16.5,123.5
 130 DATA 130.8,138.6,146.8,155.6,164.8,174.6,185.0,196.0,207.7
,220.0,233.1,246.9
 140 DATA 261.6,277.2,293.7,311.1,329.6,349.2,370.0,392.0,415.3
,440.0,466.2,493.9
 150 DATA 523.3,554.4,587.3,622.3,659.3,698.5,740.0,784.0,830.6
,880.0,932.3,987.8
 160 DATA 1046.5,1108.7,1174.7,1244.5,1318.5,1396.5,1480.0,1568
.0,1661.2,1760.0,1864.7,1975.5
 170 DATA 2093.0,2217.5,2349.3,2489.0,2637.0,2793.8,2960.0,3136
.0,3322.4,3520.0,3729.3,3951.1
 200 LET h$="  "
 210 LET h1=h: FOR o=LEN h$ TO 1 STEP -1: LET x1=h1-INT (h1/16)
*16: LET h$(o)=CHR$ (x1+CODE "0"+7*(x1>9)): LET h1=INT (h1/16):
 NEXT o
 220 RETURN
 300 SAVE "SOUND/DATA" LINE 10
```

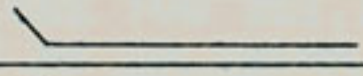
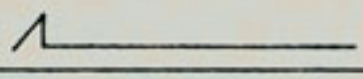
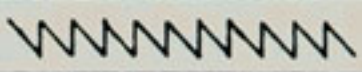
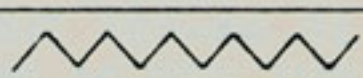
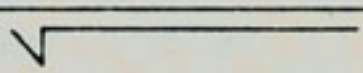
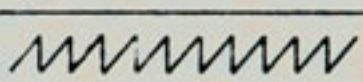
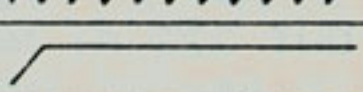
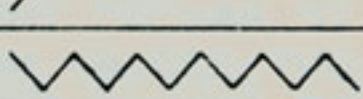
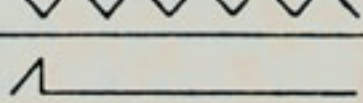
OBR.3 - PROGRAM VYRATA HODNOTY REGISTROV PRE JENNOTLIVE TONY

```
  10 BORDER 6: PAPER 6: INK 0: CLS : POKE 23658,8
  20 OUT 221,7: OUT 223,248: OUT 221,8: OUT 223,16: OUT 221,9:
OUT 223,16: OUT 221,10: OUT 223,16: OUT 221,11: OUT 223,255: OU
T 221,12: OUT 223,40
  50 PRINT AT 10,10;"KLAVESY A-K"
 100 LET A$=INKEY$: IF A$="" THEN  GO TO 100
 110 IF A$="A" THEN  RESTORE 2000: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 120 IF A$="S" THEN  RESTORE 2010: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 130 IF A$="D" THEN  RESTORE 2020: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 140 IF A$="F" THEN  RESTORE 2030: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 150 IF A$="G" THEN  RESTORE 2040: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 160 IF A$="H" THEN  RESTORE 2050: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 170 IF A$="J" THEN  RESTORE 2060: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
 180 IF A$="K" THEN  RESTORE 2070: FOR N=0 TO 5: READ A: OUT 22
1,N: OUT 223,A: NEXT N: GO TO 1000
1000 OUT 221,13: OUT 223,0: GO TO 100
2000 DATA 81,8,40,4,20,2
2010 DATA 105,7,180,3,218,1
2020 DATA 153,6,76,3,166,1
2030 DATA 59,6,29,3,142,1
2040 DATA 140,5,198,2,99,1
2050 DATA 241,4,120,2,60,1
2060 DATA 103,4,51,2,25,1
2070 DATA 40,4,20,2,10,1
9000 SAVE "SOUND/DEMO" LINE 10
```

OBR.4 DEMONSTRACNY PROGRAM - KLAVIR

```
  10 BORDER 6: PAPER 6: INK 0
  40 INPUT "REGISTER ? ";R: IF R>13 THEN  GO TO 100
  50 OUT 221,R
  60 INPUT "DATA ? ";D: OUT 223,D
  70 CLS
  80 FOR N=0 TO 15: OUT 221,N: PRINT AT N,0;N;AT N,4;IN 221: NE
XT N
  90 GO TO 10
 100 FOR N=0 TO 13: OUT 221,N: OUT 223,0: NEXT N: CLS : GO TO 8
0
 200 SAVE "SOUND/REG" LINE 10
```

OBR.5 MANUALNE NASTAVENIE REGISTROV

DEC.	CONTINUE	ATTACK	ALTERNATE	HOLD	TVAR
Ø	Ø	Ø	X	X	
4	Ø	1	X	X	
8	1	Ø	Ø	Ø	
10	1	Ø	1	Ø	
11	1	Ø	1	1	
12	1	1	Ø	Ø	
13	1	1	Ø	1	
14	1	1	1	Ø	
15	1	1	1	1	

Tabulka průběhů registru R13

MIKROPOČÍTAČE

Hry pro ZX Spectrum

2. část

ATIC ATAC (48K)

Jste ve strašidelném hradu, z něhož se musíte dostat ven. V pravé stěně první místnosti uvidíte dveře s nápisem A.C.G. To je východ. Abyste se jím dostali ven, musíte k němu najít velký zlatý klíč s písmeny ACG. Tento klíč musíte složit ze tří částí. Ty jsou poschovávány v různých místnostech hradu. Kromě částí velkého klíče najdete jídlo, malé barevné klíče a jiné drobnosti. Jídlo seberete tak, že se jej dotknete - tím se zvýší vaše energie (znázorněna na pravé straně obrazovky ubývajícím pečeným kuřetem). Ostatní předměty (včetně částí zlatého klíče) berete i pokládáte stisknutím tlačítka 'Z'. Tyto předměty můžete používat jako orientační značky (některé z nich mají i lepší použití, ale to musíte objevit sami).

V hradu je mnoho různých dveří. Bílé dveře se otvírají a zavírají samy od sebe v náhodných časových intervalech. Barevné mřížované dveře lze otevřít (natrvalo)

pouze klíčem shodné barvy. V podlaze jsou padací dveře, které se také zavírají a otevírají náhodně. Jestliže na ně vstoupíte, propadnete se o patro níž.

Když vstoupíte do nějaké místnosti, po chvíli se v ní objeví barevné obláčky. Z nich se rodí strašidélka, která můžete zneškodnit. Při přímém doteku vám užírají kuře (energii). V některých místnostech jsou strašidla o něco větší a NEZNIČITELNÁ (Hratě Dracula, mumie, Frankensteinovo monstrum, Zombie atd.). Dojde-li vám energie tj. z kuřete zbyde jenom kostra), přijdete o jeden ze svých čtyř životů. Na začátku hry si vyberete postavu, kterou budete ovládat. Každá z těchto postav má jiné specifické schopnosti. Rytíř může procházet velkými stojacími hodinami, kouzelník knihovnami a otrok sudy s vínem. A každá z nich ostřeluje strašidélka jinými zbraněmi.

OVLÁDÁNÍ

Start hry:	'O' -Předtím tlačítky 4,5,6 zvolit postavu
Pohyb doleva:	'Q'
doprava:	'W'
dolů:	'E'
nahoru:	'R'
Střelba:	'T'
Braní a odkládání věcí:	'Z'
Pauza:	'Caps Shift'

Nekonečný počet životů:
POKE 36519,0: POKE 35353,0

UNDERWURLDE (48K)

Poté, co našel amulet ztracený v džungli (v SABRE WULF), ocitl se náš lovec v začarovaném zámku, z nějž se musí dostat ven. Zámek má přes 500 místností. Asi dvě třetiny tvoří podzemí. V levém rohu obrazovky 'DEPTH xx' vidíte, ve kterém patře se právě nacházíte. Jednička udává nejvyšší patro; čím je číslo větší, tím hlouběji jste. Zámek má tři východy. Všechny jsou v nejvyšším patře (DEPTH 01). V každém z nich vás očekává reklama na některou z dalších her firmy ULTIMATE. Abyste se dostali k východu, MUSÍTE projít podzemím.

Po zámku poletuje mnoho různých příšerek, které sice lovce nesnědí, ale když se ho dotknou, odhodí ho stranou. To je nepříjemné zvláště tehdy, stojí-li lovec na okraji propasti, nebo visí-li na laně. Když lovec spadne z větší výšky než asi jeden a půl obrazovky, přijde o jeden ze svých životů (pokud ovšem nespadne na bublinu). Počet zbývajících životů je znázorněn vedle bílé figurky v pravém rohu obrazovky. Jestliže se vám podaří někde v zámku tuto figurku najít a sebrat ji, přibyde vám jeden život navíc.

Lovec ale není zcela bezbranný. Může po nepřátelích střílet některou ze zbraní, které v zámku najde. V místnosti, ve které hru začínáte, leží první zbraň - prak. Další zbraně (luk, meč a pochodeň) jsou v hradu rozmístěny pokaždé jinak. Lovec může mít u sebe maximálně tři zbraně. Ty, které má u sebe, vidíte v pravé horní části obrazovky. Jestliže má lovec u sebe více než jednu zbraň, používá při střelbě

tu, kterou sebral jako poslední.

Pokud se v podzemí dostanete do místnosti se stropem, ale bez podlahy, ve stropě je zaručeně lano. Když v takovém případě skočíte směrem ke stropu, zavěsíte se automaticky na lano. S jeho pomocí se můžete spouštět dolů. Jste-li na laně, hrozí nebezpečí, že z místa, v němž je lano ukotveno, může neočekávaně vypadnout špičatý kámen, který vám neudělá nic dobrého. Zkuste se na laně houpat, třeba vás kámen mine.

V podzemí najdete tři speciální strašidla, která stojí pod velkým krápníkem a hlídají vstup do další části podzemí. Na každé z nich platí jenom JEDNA URČITÁ zbraň:

STRAŠIDLO:	ZBRAŇ:
Šestinohý brouk	mečem
Rohatý čert	šípem vystřeleným z luku
Démon mávající křídly	pochodní

Hra je koncipována tak, že předtím, než uniknete, musíte zneškodnit všechna tři strašidla. V podzemí leží modré diamanty. Když se lovec některého z nich dotkne, zmodrá a po určitou dobu je nesmrtelný. (Zbývající doba nesmrtelnosti je zobrazena vedle nápisu GEMS v levé horní části obrazovky.) V některých podzemních místnostech narazíte na bahenní sopky, z nichž unikají bubliny plynu. Když na některou z nich vyskočíte, vynese vás nahoru. Ale pozor - příšerka, která se lovce dotkne, může ho srazit jak z lana, tak i z bubliny.

OVLÁDÁNÍ

Start hry:	'O'
Chůze doleva:	'Q'
doprava:	'W'
Skok:	'R'
Střelba:	'T'
Braní a odkládání zbraní:	'Space'
Pauza:	'Enter'

VISÍ-LI LOVEC NA LANĚ

Rozhoupání doleva:	'Q
doprava:	'W'
Natahování lana:	'E'
Zkracování lana:	'R'
Seskočení z lana:	'Caps Shift

Nekonečný počet životů:
POKE 59376,0: POKE 59380,0

PINBALL (16K)

V této hře musíte pomocí pálek (flipperů) udržet ve hře kuličku létající po hrací

ploše a získat co nejvíce bodů. Body přibývají pokaždé, když se kulička dotkne různých cílů v hracím poli. Bodový zisk nezávisí přímo na tom, jak dlouho udržíte kuličku v hracím poli. Ke hře patří i to, že musíte zjistit, jak se mění vaše skóre po zasažení toho kterého cíle a jaké to má vedlejší efekty (písmena S,A,G,I,T,T,A,R, I,A,N, BONUS, atd.). Kulička je mimo hru, jakmile propadne mezi pálkami. K dispozici máte celkem pět kuliček. Po rozsvícení celého nápisu SAGITTARIAN (latinsky STŘELEC) získáte vždy jednu prémiovou.

OVLÁDÁNÍ

Začátek hry:	'Enter'
Prvotní odpálení kuličky:	'O' (je možno určit sílu odpálení)
Levý flipper:	'Q'
Pravý flipper :	'P'

TORNADO LOW LEVEL (48K)

Vaším úkolem je odstartovat s tryskovým letadlem, za letu posbírat pět kruhových cílů ze země, a potom jemně přistát. Jestliže tento úkol splníte, pokračujete v další fázi se stejným terénem, ale s obtížněji rozloženými cíli. Za letu se musíte vyhýbat budovám, telegrafnímu vedení, sloupům veřejného osvětlení, stromům a břehům (to když letíte pod úrovní pevniny). Navíc je váš úkol časově omezen. Vaše letadlo může letět jednou ze dvou rychlostí vlivem změny sklonu křídel.

V pravém horním rohu obrazovky je umístěn radar zobrazující vaše blízké okolí. Cíle, které musíte sebrat přízemním letem, jsou na radaru zobrazeny jako tečky. Pod radarem umístěné ukazatele udávají vaši výšku, množství paliva, počet zbývajících cílů a čas. Dále na obrazovce vidíte, kolik letadel ještě máte k dispozici (začínáte se třemi), skóre momentální a rekordní.

Kromě toho, že musíte přistát po splnění úkolu, je přistání nezbytné vždy, kdykoli potřebujete doplnit palivo nebo se podívat na mapu (tlačítko 'M'). Na mapě jsou cíle znázorněny červeně blikajícími čtverci.

Při přistávání musíte:

1. mít roztažená křídla (nižší rychlost)
2. dosednout JEMNĚ
3. po dosednutí držet tlačítko 'Klesání', dokud letadlo nezastaví

OVLÁDÁNÍ

Začátek hry:	'3'
Start/změna rychlosti:	'X'
Stoupání:	'1'
Klesání:	'Q'
Zatáčení doleva:	'G'
doprava:	'H'
Mapa:	'M'

Fr. FUKA

Programová nabídka MIKROBÁZE

ZX Spectrum

DrMG

Na bázi známé kombinace programů GENS3 a MONS3 postavená úprava, která umožňuje např. jednodušší spolupráci mezi oběma částmi programu, odpadají starosti se studenými a teplými starty, lze měnit začátek pracovní oblasti, při disassemblování se monitor neptá na adresu, kam překlad uložit, ale sám si vyhledá konec zdrojového textu generátoru, uloží překlad za něj a upraví příslušné parametry generátoru, dále je přidáno tolik potřebné "pípání" tlačítek, průvodní texty jsou slovenské, přidaný modul provádí přepočty mezi různými číselnými soustavami atd. Původní funkce obou základních programů zůstávají zachovány. Doplněno dvěma svazky bohatého manuálu. DrMG je jedinou kompilací původního materiálu se zahraničním vzhledem k jeho určení pro průběžné doplňování znalostí assembleru Z80 v návaznosti na didaktickou činnost Mikrobáze, směrovanou na její členskou základnu. **Cena 124,- Kčs, poštou na dobírku včetně balení 135,- Kčs.**

DIAPEN

Slovní procesor pro editaci textu v českém a slovenském jazyce. Název je složen ze dvou slov - PEN je dnes již mezinárodním označením mnoha druhů písátek, DIA je zkratkou slova diakritický (rozlišovací), které ve spojení se znaménky označuje čárky, háčky, tečky, kroužky apod. u písmen mnoha abeced různých jazyků. DIAPEN počítá i s velmi jednoduchou úpravou pro editaci jakékoli abecedy mnoha dalších jazyků (od azbuky po norštinu). Na rozdíl od všelijakých úprav anglických editorů pro editaci diakritických znamének dosahuje DIAPEN zvláštní úpravou toho, že všechny původní znaky ASCII kódu zůstávají zachovány a vůbec se nemění jejich pozice na klávesnici. Výhoda tohoto řešení je m.j. v tom, že na DIAPENU napsaná např. česká slova se na jiném editoru promítnou nikoli jako změť nesrozumitelných znaků, ale budou u nich chybět jen dia-znaménka. Rovněž pro tisk českého textu z jiného editoru nebude třeba provádět žádné úpravy - text se vytiskne jen bez dia-znamének. Manuál DIAPENu bude obsahovat instruktáž provedení tisku písmen s těmito znaménky na tiskárnách, které mají buď grafický mód, nebo tzv. down-load do volné paměti tiskárny. Přímo na kazetě budou softwarové bloky pro práci s některými typy tiskáren a interfaců. DIAPEN bude obsahovat všechny funkce pro práci s textem, jak je tomu např. u Taswordu nebo Spectral Writeru, a některé funkce nové; ovládání tiskárny je rozšířeno. Obecně lze říci, že DIAPEN vyplňuje výraznou mezeru, která zeje v oblasti

editace češtiny a slovenštiny z klávesnic zahraničních mikropočítačů. Doplněno bohatým manuálem. **Cena není dosud stanovena, protože program není dokončen.**

/uB-PASCAL

Integrovaný systém umožňující editaci, překlad a provádění programů v jazyce PASCAL. Obrazovkový systém editoru pracuje se 64 znaky na řádce. Použitá verze jazyka PASCAL je velmi blízká mezinárodní normě ISO 7185 (úroveň Ø) a implementacím DC-PASCALU na mikropočítačích IQ 151 a PP 01. Jazyk obsahuje řadu rozšíření. Překladač je navržen tak, aby byl vhodným prostředkem i pro výuku programování. Poskytuje detailní chybovou diagnostiku a možnost přísných běhových kontrol. Programy mohou pracovat na logické úrovni se soubory, které fyzicky vstupují nebo vystupují přes klávesnici, obrazovku, tiskárnu, magnetofon, microdrive. Přiložen přehledně zpracovaný manuál. **Cena 194,- Kčs, poštou na dobírku včetně balení 205,- Kčs.**

DATALOG

Svým uživatelským komfortem v mnoha směrech výrazně převyšuje obdobné databázové programy pro ZX Spectrum. Založení databanky a formátu výpisu všech zpráv a položek je snadné (řešeno přímým grafickým navrhováním formátu s průběžnou kontrolou jeho vzhledu na obrazovce) a velmi variabilní. To platí i pro provedení jakékoli změny nebo opravy. Uživatel je při práci s DATALOGEM veden jednoznačnou volbou funkcí z posloupnosti přehledných menu. Kterákoli položka zprávy může být vypisována ve formátu 64 nebo 32 znaků/ř. Novinkou, kterou uživatelé ZX Spectra ocení, je možnost dělení souborů dat databanky podle uživatelem stanoveného výběru s následným individuálním zápisem vybraných částí souborů na vnější paměťové médium. Takto vzniklé "dílčí soubory" mohou být do databanky načteny jak samy o sobě, tak i přičleněny k souboru v databance přítomnému, tedy spojovány. Komunikace se záznamovými zařízeními je zajištěna příkazy Basicu, které jsou uživateli přístupné, přizpůsobitelné jakémukoli záznamovému zařízení. Dodávaná verze obsahuje příkazy pro magnetofon a microdrive. Přenos dat na tiskárnu neprobíhá pomocí funkce COPY, ale ve formě znakových kódů. DATALOG pracuje s českou a slovenskou abecedou, implementována jsou i jinojazyčná písmena, vyskytující se např. v příjmeních. Velmi detailně zpracovaný manuál DATALOGu má dvě části. První je určena běžným uživatelům, druhá poskytuje programátorům informace především o možnosti provedení změn některých parametrů DATALOGu. **Cena 175,- Kčs, poštou na dobírku včetně balení 186,- Kčs.**

mikROMkód

Velmi žádaný, kompletní přehled rutin ROMky ZX Spectra 48K (tedy i ZX Spectra+ a ROMky, která je funkční v módu 48K u nových verzí ZX Spectra 128K). Kazeta bude obsahovat jednoduché rutiny, které budou voláním subrutin ROMky vykonávat požadované funkce i díky možnosti vkládání různých vstupních parametrů do hlavních rutin. Jedná se tedy o obdobu známého programu Supercode, ovšem s tím, že struktura rutin bude zcela odlišná, funkčně variabilnější a bude se plně soustředit na využití rutin paměti ROM. Celé dílo bude korunováno plným, komentovaným assemblerovým výpisem

celých 16K ROMky. Pochopitelně nebude chybět ani popis hlavních programových rutin kazety. Oproti mnohým z vás známému originálnímu výpisu ROMky bude mikROMkód doplněn informacemi o možnostech práce s jejími stěžejními částmi (především kalkulátorem, rutinami obrazovky, klávesnice, ovládáním kanálů, zvuku, tisku, atd.). Program mikROMkód se svým velice rozsáhlým manuálem stane nezbytností každému, kdo bude chtít plně využít schopností svého počítače - především znalcům assembleru mikroprocesoru Z80. Ale i stoupenci jiných jazyků budou moci využít znalostí, které jim mikROMkód poskytne. **Cena není dosud stanovena, protože program není dokončen.**

PLOŠNÍK

Se stane vítanou pomůckou každého konstruktéra - elektronika. S jeho pomocí lze snadno a rychle vytvářet návrhy plošných spojů, osazených až 100 obvodových prvků (včetně integrovaných obvodů). Jako vstupní parametry jsou do PLOŠNÍKu vkládány typy součástek, očíslovaná místa jejich spojů atd. Na výstupu je zpracován grafický návrh plošného spoje, který je možno dodatečně optimalizovat jak programově, tak přímo graficky na obrazovce. Výsledný návrh lze vytisknout na tiskárně s grafickým módem nebo s pomocí definovaných znaků tiskárny (download). Uživatelé bez přístupu k tiskárně mohou použít cestu fotografické kopie obrazovky. **Cena není dosud stanovena, protože program není dokončen.**

KAREL

Tento populární programovací jazyk vám Mikrobáze nabízí hned ve třech provedeních pro tři různé mikropočítače. Z toho verze pro ZX Spectrum je zcela novou modifikací programu. Programový manuál se zabývá nejen programovacími povely, jeho rozsah je významně rozšířen o celou metodiku programování s tímto typem jazyka. Ve své objednávce nezapomeňte uvést, pro jaký typ počítače program objednáváte. Dále si krátce připomeneme základní charakteristiku KARLA.

KAREL je mikropočítačový program, který je současně zábavnou hrou i seriózní učební pomůckou. Pochází ze Stanfordské univerzity, kde byl původně koncipován jako předstupeň výuky programovacího jazyka Pascal. V naší implementaci je do jisté míry setřena jednostranná orientace na Pascal a přidání některých nových prvků činí z tohoto programu univerzální prostředek pro počáteční fáze výuky moderního programování.

Niklaus Wirth, autor programovacího jazyka Pascal, uvádí, že program - algoritmy + datové struktury. V systému KAREL jsou datové struktury potlačeny, což umožňuje snadnější chápání základních pojmů a postupů používaných při sestavování algoritmických struktur programů. Získané poznatky a dovednosti jsou pak využitelné ve většině ostatních programovacích jazyků.

KAREL má své opodstatnění i při přípravě k programování v jazyce Basic, který má nejširší uplatnění v oblasti mikropočítačů. Je holou skutečností, že Basic nemá

dostatek prvků podporujících tvorbu strukturovaných a modulárních programů. Kdo však zvládnul KARLA, má i v jazyce Basic předpoklady k vytváření účinných, přehledných a dobře modifikovatelných programů. Mikropočítače a roboty sice směřují k takové dokonalosti, že je nebude třeba programovat speciálními programovacími jazyky, ale algoritmizace je dovednost využitelná obecně, nejen při programování počítačů.

V našem programu je Karel jméno robota, který je nakreslen na obrazovce mikropočítače. Karel má na obrazovce svoje město ohraničené zdí a v tomto městě vykonává funkci dopravní služby. Může se přesouvat z křižovatky na křižovatku a ukládat i sbírat na křižovatkách dopravní kužely, značky.

Program KAREL pro mikropočítač je dělán tak, aby sám dával návod k další činnosti obsluhy; lze s ním pracovat, aniž by uživatel potřeboval jakoukoliv příručku. V případech, kdy pro stručnost není jeho pokyn jednoznačný, lze správný postup nalézt metodou pokusů a omylů. Přesto je vhodné, či spíše nutné doplnit práci s programováním Karla také vysvětlením základních pojmů strukturovaného a modulárního programování. K tomu slouží tištěný instrukční materiál.

Karlovi se dávají povely běžnými českými slovy. Pochopení nových poznatků proto není ztěžováno současným učením anglických slov. Karel je také úslužný robot, všechno, co si může domyslet, udělá nebo napíše sám. S formální stránkou svého učení v nejvyšší míře sám napomáhá. A tak se stává z trpělivého žáka ještě trpělivějším učitelem. (A pro zkušené programátory se z učitele stává zábavný společník!)

AMSTRAD/SCHNEIDER

TRAN-SP-AM

Oboustranný převodník formátů zápisu a čtení dat mezi počítači ZX Spectrum a Amstrad/Schneider. Program např. umožní, aby text zapsaný na kterémkoli z obou počítačů bylo možno načíst "do jeho kolegy" a jakkoli s ním dále pracovat. Převod se provádí na počítači Amstrad/Schneider. To znamená, že TRAN-SP-AM umožní vzájemnou komunikaci mezi držiteli obou typů počítačů, stejně jako těm z vás, kteří ze Spectra přecházíte na Amstrad, nabídne možnost převodu všech vašich textů zapsaných na Spectru na záznamový formát vašeho nového počítače. Součástí programu je modul, který umožní konverzi znakových kódů, pokud jí bude třeba. Pokročilejší uživatelé výpočetní techniky správně tuší, že převádět lze nejen texty psané na slovních procesorech, ale i jakékoli jiné (zdrojový text assembleru, databázové záznamy apod.). Převod se provádí na počítači Amstrad/Schneider. Obsluha programu je vedena logicky řazenými dotazy menu, je proto velmi jednoduchá. Pro uživatele, kterým je formát dat ještě trochu hádankou, je připojen informativní manuál.
Všechny programy budou distribuovány pouze na kazetách. **Cena není dosud stanovena, protože program není dokončen.**

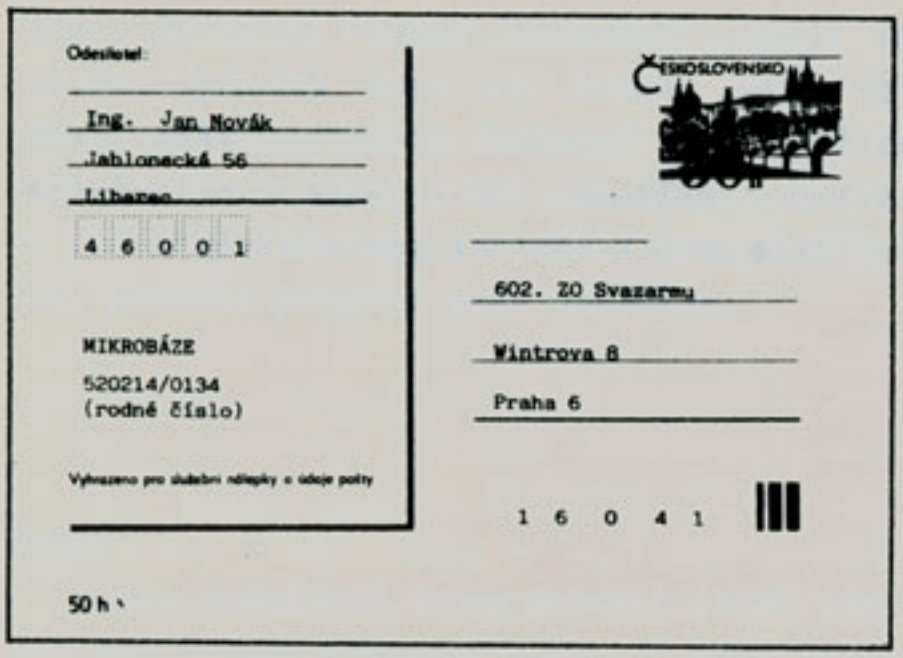
Pokyny k objednávání programů

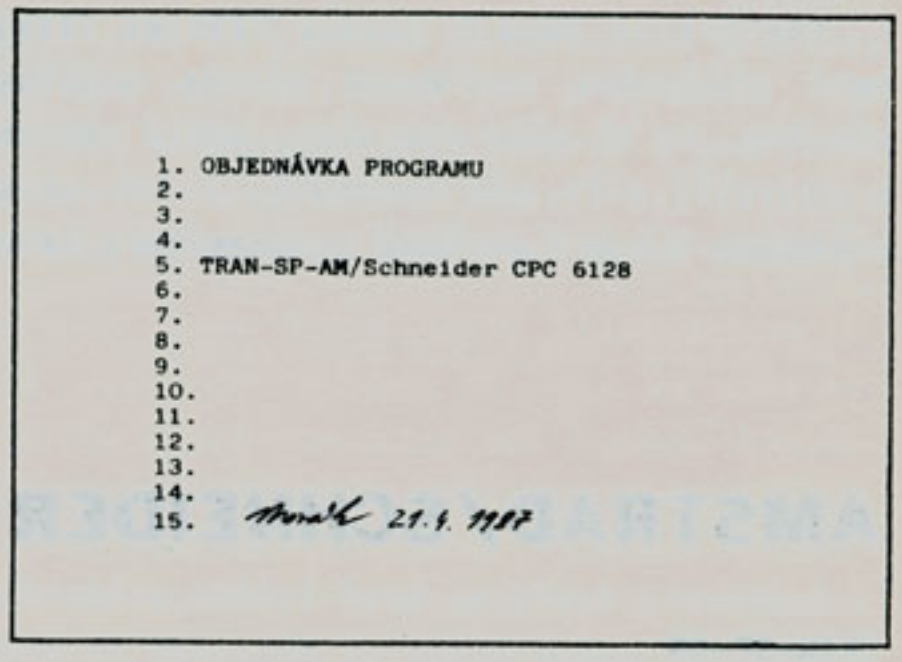
Samozřejmě, v platnosti zůstávají "pravidla hry" zveřejněná jak v Amatérském radiu (naposled v č. 5. ročníku 1985), tak v tiskovině s organizačními pokyny, kterou dostal každý, kdo projevil korespondenčním lístkem zájem o členství v Mikrobázi. Pro jistotu otiskujeme vzory vyplnění líce i rubu korespondenčního lístku k objednání programu z naší nabídky znovu.

Rub lístku budete vyplňovat v řádcích 1, 5 a 15. Do řádku 1 napíšete OBJEDNÁVKA PROGRAMU, do řádku 5 označení (název) programu podle nabídky.

POZOR! Programy ještě nemají přesná katologová označení, v nichž budou v budoucnu zakódovány typy počítačů. Proto zatím označení programu na řádku č. 5 uvádějte i s udáním typu počítače, pro který program objednáváte. Příklad:

5. Diapen/Sinclair Spectrum





Slovo k náhodným čtenářům

Dostal se vám tento zpravodaj Mikrobáze do rukou a zaujala vás aktivita rozvíjená v programových a technických službách pro uživatele mikropočítačů? Redakce časopisu Amatérské radio jako iniciátor Mikrobáze a 602. ZO Svazarmu v Praze 6 jako realizátor vás zvou k členství v několikatisícovém kolektivu zájemců o výpočetní techniku a její aplikace. Jako člen Mikrobáze Svazarmu budete dostávat zpravodaje (v roce 1987 celkem pět čísel), budete moci využívat programových nabídek a dalších plánovaných služeb.

O bližší informace o celém komplexu služeb Mikrobáze a přihlašovací materiály je třeba požádat korespondenčním lístkem. Jeho líc vyplňte (zásadně strojem) podle vzoru na této straně, na rub napište sdělení: "Mám zájem o členství v Mikrobázi", připojte datum a podpis. Pak už stačí vhodit lístek do poštovní schránky a čekat na podrobné informační a přihlašovací materiály Mikrobáze. Dostanete je obratem. I když se naším členem stanete až v průběhu roku, máme pro vás připraveny všechny Zpravodaje Mikrobáze vydané od ledna 1987.