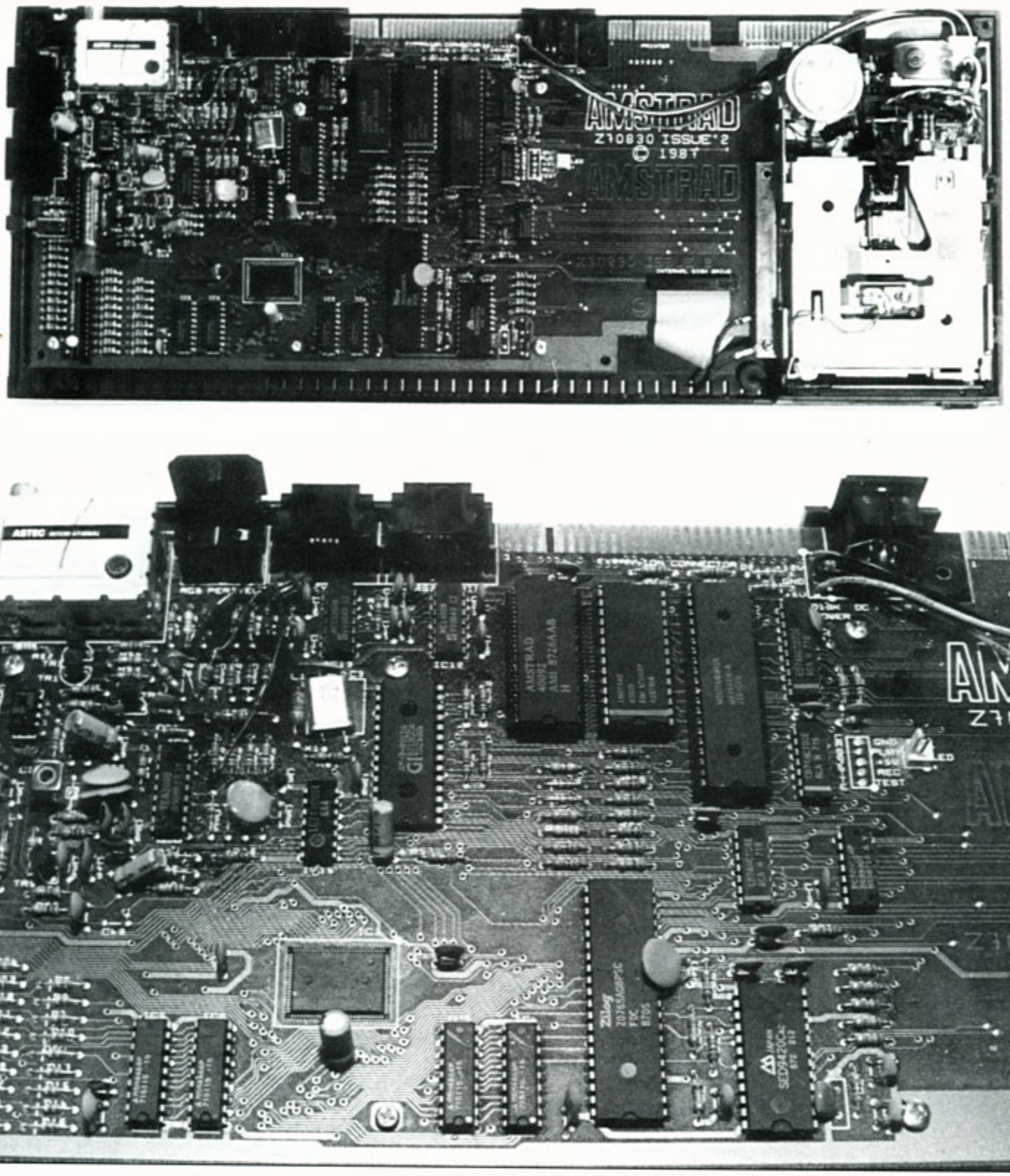
MIKRO BÁZE

1989 7

technický zpravodaj svazarmu pro zájemce o mikropočítače

Cena 12 Kčs


sinclair
128K
ZX Spectrum+3
128K
FDD
FLOPPY DISC DRIVE


AMSTRAD
Z70830 ISSUE 2
©1987
AMSTRAD
AMSTRAD
Z70830
PRINTER
Zilog
SED9420CA

MIKRO BÁZE

1989/7

OBSAH

Technický zpravodaj Svazarmu pro zájemce o mikropočítače. Vydává 602. ZO Svazarmu ve spolupráci s redakcí časopisu Amatérské radio. Povoleno ÚVTEI pod evidenčním číslem 87 007. Zodpovědný redaktor ing. Jan Klabal. Sestavil vedoucí redaktor Daniel Meca. Obálka ak. grafik Jiří Blažek, grafická úprava Lenka Chromková. Sekretářka redakce Zdeňka Válková. Redakční rada: Petr Horský, ing. Jan Klabal, ing. Petr Kratochvíl, Josef Kroupa, Rudolf Mach, Daniel Meca, ing. Alois Myslík, ing. Josef Truxa. Za původnodt a správnost příspěvků ručí autoři. Ročně vyjde 10 čísel. Cena výtisku 12 Kčs podle ČCÚ a SCÚ č. 1030/202/86. Roční předplatné 120 Kčs. Objednávky přijímá a zpravodaj rozšiřuje 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6.

SVAZARM

602.ZO

&

Amatérské RADIO

A JEDEME DÁL

Podle data, kdy se vám dostává do rukou toto číslo Mikrobáze, můžete snadno usoudit, že s termíny vydávání jsme letos opět na štíru. Je ironií, že v době, kdy se tiskl optimistický úvodník do čísla 5, ve kterém projevil J. Kroupa potěšení nad posílením redakčního kolektivu a jeho stabilizací, tento stav se nám opět povážlivě snížil. Po odchodu celé poloviny všech redaktorů (viz. minulé číslo) jsem totiž zůstal v redakci sám. Abych se tu nebál, je tu se mnou ještě naše nová sekretářka, i když jen na půl úvazku. No, vlastně to zas tak zlé není - máme něco kolem deseti tisíc čtenářů. To už je přece nějaký kolektiv.

Prostě, jednu chvíli jsme nevěděli kde nám hlava stojí. Ani teď si nemůžeme na nedostatek práce stěžovat, ale už jsme se dostali do trochu pravidelného tempa. Dokonce se snažíme Mikrobázi trochu vylepšit. Věřím, že i vy máte upřímnou snahu nám v tom pomoci. Soudím tak podle množství korespondence, která nám, hlavně teď po prázdninách, začíná do redakce chodit. A není to jenom kritika typu: "...to a to děláte špatně, měli byste to dělat nějak jinak...", případně trochu konstruktivnější návrhy jako: "...zařaďte články o...". Dostáváme konečně také takovéhle dopisy: "...a proto vám posílám příspěvek na toto téma...".

Je nabíledni, že ty poslední nás těší nejvíc, přestože často také kritizují, ale zároveň nám nabízejí pomocnou ruku. A to je důležité, dobrý zpravodaj můžeme udělat jen v úzké spolupráci se čtenáři. A my ho chceme dělat dobře.

Teď to vypadá, že nás všichni jen kritizují. Tak to není. Dostáváme i řadu pochvalných dopisů, které nám dodávají sílu a chuť do další práce. Jen nás v té souvislosti mrzí jedna věc - nestačíme při nejlepší vůli na všechny dopisy včas odpovídat. Dovolte, abych se touto cestou omluvil všem, kterým jsem dosud neodepsal. Časem se polepším. Teď mám ale jedinou možnost, - dělat Mikrobázi. A to i za cenu toho, že třeba nestihnu vyřídit část korespondence.

Ve snaze lépe informovat naše čtenáře, jezdíme po různých akcích, nejen svazarmovských. Fotografujeme, rozmlouváme a zapisujeme, uzavíráme smlouvy (to množné číslo tu není omylem, naše sekretářka zde obětavě přikládá ruce k dílu). K tomu práce v redakci, zajištění tisku, honoráře... Nemůžeme tedy při nejlepší vůli zodpovídat různé dotazy, se kterými se na nás obracíte. Vaše důvěra nás těší, na otázky bychom rádi odpověděli, ale nemůžem to dělat individuálně. Na ukázku dovolte citovat část jednoho, ne tak netypického dopisu:

"...Chtěl bych Vás tímto dopisem poprosit, abyste mi poradil, jak je možné připojit tiskárnu BT-100 k ZX Spectru přes univerzální paralelní interface UR-4. Prosím Vás taky, abyste při vysvětlování postupoval co nejsrozumitelněji (bez velkého užívání odborných termínů - nejlépe stylem: "Propojíme ten a ten otvor v konektoru FRB s tím a tím otvorem na tiskárně" a pod.), protože, jak už jsem říkal, jsem v oblasti elektrotechniky téměř absolutní laik... ...ale přesto Vás prosím, abyste si chvíli toho času našel a zbavil mě toho bezradného vzteku na sebe samotného při pohledu na dva, pro mě tak potřebné, kusy elektroniky, jejichž záruční doba se neúprosně krátí..."

Po přečtení takového dopisu se čtenářem sdílím vztek, zaměřil bych ho ale spíš na výrobce, kteří si stále nemohou zvyknout na to, že sebelepší výrobek je bez pořádného návodu pro řadu lidí bezcenný.

Ale proto jsem dopis necitoval. Jde totiž o to, že i když většina tazatelů přikládá ofrankovanou obálku, nebo korunovou známku na odpověď, ani tak nemají zaručenou odpověď v žádaném rozsahu. Stručná odpověď na takovýto dopis zabere řekněme půlhodinku. Jenže podobných je řada. To pak je půlhodinka k půlhodince... Už tak jsem přes léto pracoval možná dvanáct hodin denně, včetně řady víkendů. Tyto dopisy nás však inspirovaly k zavedení krátké rubriky pro začátečníky.

Nakonec z jiného soudku. Blíží se konec roku a bude pomalu čas předplatit si příští ročník. V této souvislosti vás můžeme uklidnit - nepočítáme se zvýšením ceny. To se týká pouze novin a časopisů, které byly dosud dotované státem. My už od samého začátku musíme hospodařit v přísném chozrasčotu (při našem nákladu a současných cenách v polygrafii to není nic jednoduchého). To také vysvětluje naší poměrně vysokou cenu. Teď se nám tedy v tomto ohledu řada časopisů přiblíží. Raději bychom se však byli přiblížili cenou my jim.

Daniel Meca

DŘU, DŘEŠ, DŘEME... CÉČKO /7/

Po minulé sprše hrátek s ukazateli, vás v této části čeká přímo povodeň. Coby rozvojový začátečník jsem nad céčkovou ekvilibristikou s ukazateli polí dost dlouho setrvával v roli pověstné husy nad lahví. Kolem mě spousta rozevřených knih, a já mezi nimi jako ten mnich...jen narůstající hora otázek bez kýžených odpovědí. Nakonec jsem literaturu opustil a ponořil se do vlastního bádání. Jeho výsledek předkládám v naději, že vám ulehčí pronikání do záhad causy "Rafinovaný ukazatel vs. člověk obecný".

Začnu hned tím nejsložitějším - poli znaků. Číselná vám pak už půjdou sama.

Na úrovni běžného programování se sice stává, že potřebujeme více než dvourozměrné pole znaků, ale pro další výklad si je zvolím jako limit. I proto, že Céčko má pro souvztažné operace s větším počtem polí a proměnných všeho druhu své struktury a uniony. Pokud byste se chtěli dostat ke třem a více rozměrům, nebude až tak nesnadné odvodit vše potřebné z dalšího textu.

Znakové řetězce se do "céčkové paměti" ukládají s nulou '.0' na konci (za posledním znakem řetězce). Na to musíme obecně pamatovat, když polím předem vyhrazujeme paměť. Pro další výklad budiž rozuměno, že řetězec je jednorozměrné znakové pole. Inicializovat řetězec můžeme např. takto:

```
char *řetězec="znaky";
```

což je totéž, jako:

```
char *řetězec={'z','n','a','k','y','.0'};
```

Jak vidíte, ve skutečnosti se do paměti uloží o jeden znak víc (z gruntu vzato, ono se tam uloží ještě leccos jiného, ale to je věc kompilátoru, mimo oblast našich momentálních zájmů).

Obě deklarace říkají, že vytvářejí ukazatel na typ char. Samotný ukazatel pak míří na první znak řetězce. Inicializaci můžeme provést ještě dalšími způsoby:

```
char řetězec[]="znaky";
char řetězec[1]="znaky";
char řetězec[1][5]="znaky";
char řetězec[ ][5]="znaky";
```

a totéž i s hvězdičkou za char. Bez ní inicializujeme jednorozměrné znakové pole přímo, bez prostřednictví ukazatele, tedy i bez jeho vytvoření. V deklaraci pole (bez uvedení jeho budoucích rozměrů), resp. v definici (s uvedením jeho rozměrů, tedy i s náležitou rezervací rozsahu paměti) musí být přítomna buď hvězdička, nebo hranaté závorky, nebo i obojí. Bez toho bychom definovali jen jeden znak (v případě int jedno číslo).

Deklarace na první řádce dává kompilátoru na vědomí, že jde o pole, jehož rozměry nejsou dány. Budou však okamžitě určeny uvedenou inicializací. V ten moment kompilátor uloží do paměti "znaky" a bude pole brát jako "1 x 5". Samozřejmě můžeme pole deklarovat bez uvedení jeho rozměrů i bez jeho současné inicializace. Jakmile však někde nějak takovému poli programově nezajistíme náležitý (třeba i jen předpokládaný) prostor a začneme do něj později ukládat data, pole nerušeně přepíše všechny důležité adresy ve svém okolí. (Jak uvádí M.Waite a spol. v knize **C Primer Plus**, taková situace je pro programátory velice zábavná, ale jen když se vyskytne v cizím programu.) Pro zajištění paměťového prostoru má Céčko standardní knihovní funkce **calloc()** a **malloc()** (character allocation a memory allocation). Jim podobnou funkci **alloc()** si předs tavíme v závěrečném programu.

Deklarace na druhé řádce uvádí, že půjde o jednorozměrné pole bez udání jeho délky. Třetí řádka definuje jednorozměrné pole beze zbytku. Deklarace na čtvrté řádce říká, že prvky pole budou mít délku pěti znaků, ale neví se, kolik jich bude.

Tyto deklarační úvahy platí jen v případě, že pole není současně inicializováno. Na uvedených řádkách je vše jasně dáno. Dále si ukážeme, jak využít jednu ze šikovných schopností kompilátoru při práci se znakovými poli. Dejme tomu, že chceme do paměti umístit dvourozměrné pole, jehož obsahem bude 5 jmen:

```
char *jména[]={"Eva","Jeremiáš","Evelýna",
               "Jan","Pavlína"};
```

V jediných hranatých závorkách, informujících jen o tom, že "to" bude pole, nic není a přesto se kompilátor nedurdí a vše uloží, jak náleží. Jaké je nejdelší jméno z oněch pěti? Jeremiáš má 8 znaků. Aby se do pole vešel, mohli bychom pole deklarovat třeba takhle:

```
char *jména[][9]={"---jména jako výše---"};
```

Číslo 9 je tam pro přidělení místa zakončovací nule. Ve výsledku jde o zásadní rozdíl v rozsahu zabrané paměti po inicializaci pole:

```
1.deklarace/inicializace:   Eva0
                            Jeremiáš0
                            Evelýna0
                            Jan0
                            Pavlína0

2.deklarace/inicializace:   Eva000000
                            Jeremiáš0
                            Evelýna00
                            Jan000000
                            Pavlína00
```

Jistě netřeba cokoli dodávat. Ve stylu druhé deklarace pracuje známý žrout paměti - basicový

příkaz DIM.

Následujícím rozborem rozmístění polí a ukazatelů v paměti vstoupíme do jejich tajemné říše. Z původního testovacího programu uvádím jen nejpodstatnější kousky:

```
main()
( ...
static char **m="mikro";
static char *n="mikro";
static char o[]="mikro";
static char *p[]="mikro";
...
printf("%s %c\n",m,*m);
printf("%u %u %u\n",&m,m,*m);
printf("%s %c\n",n,*n);
printf("%u %u\n",&n,n);
printf("%s %c\n",o,*o);
printf("%u\n",o);
printf("%s %c\n",*p,**p);
printf("%u %u\n",p,*p);
... .
```

Výpis:

```
mikro
65365 48524 27913
mikro m
65363 48534
mikro m
65357
mikro m
65355 48552
```

Uvedené adresy se vytvořily v Hisoftu C na ZX Spectru. To je však pro podstatu věci vedlejší, protože adresy poskytují zásadní informaci o rozvržení paměti a o práci s ukazateli. Jen pro tuto část si označme adresu, na níž sídlí ukazatel jako jeho sídlo a adresu obsahující cílový znak jako sídlo domovníka. Je-li tedy ukazatel přítomen, ukazuje na sídlo domovníka, který je povinen ukázat znak, na kterém "sedí", resp. přijmout místo něj jiný.

První deklaraci jsem uvedl jen pro úplnost, její částečnou zmatenost vyjevím později. Pro nás má svou důležitost z hlediska uložení ukazatele:

ukazatel na m	**domovník m alias ukazatel nižšího stupně**
65365 (2 bajty, obsahující adr.domovníka)	48524
ukazatel na n	**domovník n**
65363	48534
	domovník o
	65357
ukazatel na p	**domovník p**
65355	48552

Na první pohled je patrné, že kompilátor ukládá ukazatele za sebou na nejvyšší adresy paměti. Je tu ovšem jedna výjimka - **pole o[]** nemá ukazatele. Kompilátor to řeší tak, že pole "mikro" uloží přímo na nejvyšší adresy mezi ostatní ukazatele (povšimněte si většího rozdílu mezi uložením ukazatele na n a domovníkem o - právě tam leží řetězec "mikro" **pole o[]**).

A jsem u jedné z céčkových podivností, která mne dlohou dobu mátla. Céčko bere i domovníka o jako ukazatele a říká, že rozdíl mezi "ukazatelem" na **pole o** a ukazatelem na **pole n** je v tom, že první je konstantou a druhý proměnnou. Nemohu si pomoci , ale to mi připadá jako určitá výmluva pro chtěnou konstrukci, jíž se podřizují všechny související funkce Céčka. Domovník nemá proč být adresovou konstantou či proměnnou, protože to je adresa, kde začíná řetězec a jeho obsah můžeme měnit. Mnohem logičtější by bylo, kdyby ukazatel na **pole n** byl pojímán jako jednorozměrné pole ukazatelů s jedním prvkem a další operace s ním by podléhaly zákonitostem uvedeným u pole ukazatelů na řetězce **pole p**. Tak tomu v Céčku ovšem není.

Proč? Protože když zápis ***ukazatel** ve formátu **%u** beru jako výzvu pro vypsání obsahu ukazatele, kterýmžto obsahem je adresa domovníka, pak v případě ***o** ve stejném formátu dostanu nesmysl, resp. adresu složenou ze dvou prvních znaků **řetězce o**. Ve formátu **%c** odkazuje domovník na sebe samého, protože je tu poněkud schizofrenicky domovníkem i ukazatelem v jedné osobě. Přiznám se, že než jsem objevil tenhle logický paradox, byl jsem nepříjemně dlouho obětí logického přístupu, který jsem po seznání skutečnosti musel zahodit.

Zatímco ukazatele a pole bez ukazatelů rostou od vyšších adres dolů, pole vázaná na ukazatele se ukládají od spodních adres nahoru (rostou proti sobě). Na adresách domovníků je uloženo první písmeno **pole "mikro"**, proto se vyskytuje ve výpisu znakového formátu. Ale neobjevuje se u prvního **pole m**. Z minulého pokračování víte, že v případě proměnné typu **int** můžeme pracovat s dvoustupňovým ukazatelem. V případě typu **char** to (alespoň v Hisoftu C) jde jen "napolo". Požadavek o výpis řetězce je splněn správně. Ovšem pro výpis jednotlivých znaků si kompilátor ponechává adresovou aritmetiku pro **int** (tedy "skáče" nikoli po jednom bajtu, ale po dvou):

```
*++m či m[1] dá výpis i
      m[2] dá výpis r
```

což je v obou případech spodní bajt čísla uloženého na adresách:

48524	48525	48526	48527	48528	48529
9	m	i	k	r	o

Proto se ve výpisu ***m** nic neobjevilo - kód 9 není zobrazitelný. Požadavek na číselný výpis ****m** (nejnižší úroveň dvoustupňového ukazatele) dal 27913, což je 9+109*256 (109 je ASCII kód písmene m). Z uvedených důvodů bude jistě lepší se téhle deklaraci pro typ char vyhnout.

U ostatních deklarací funguje všechno "normálně". Tak třeba žádost o výpis znaku na pozici n[3] dá r.

Možná někoho z vás překvapilo, že pro formát výpisu řetězce **%s** je požadována vyšší úroveň reference než pro výpis znaku. Tak už to ve funkci printf() i v dalších chodí a je třeba to mít na paměti.

Zatímco reference pro výpis znaků (zvlášť) a řetězců (také zvlášť) jsou v prvních třech případech na stejné úrovni, poslední (pole p) se od nich liší. K tomu (i k významu unárního operátoru & v argumentech funkce tisku) se hned dostaneme.

Všechny řetězce byly inicializovány v těle funkce, proto musíte kompilátor žá dat o statickou paměť. V automatické mohou být řetězce i pole je n deklarovány nebo definovány.

Přejděme k analýze dvourozměrného pole:

```
char *data[]={"alfa","beta","gama","delta");
```

Pole je deklarováno jako vnější (mimo tělo funkce), proto není požadavek na statickou paměť nutný (ale být tam může).

V Céčku se deklarací polí týkají tři základní pravidla priorit. Nejvyšší mají závorky (), níž jsou hranaté [] a ještě níž jsou unární operátory & a *. Kromě toho platí, že co z toho je blíž identifikátoru, to má přednost. Např. definici

char *něco[3][4] vyluštíme následovně: identifikátor v objetí * a [3] říká, že se má vytvořit jed norozměrné pole tří ukazatelů (adresových), které budou ukazovat na adresy, na nichž budou uloženy první znaky tří znakových řetězců dlouhých 4 znaky. Jinak též - jde o vytvoření 3 ukazatelů na pole typu char.

Nebudu široce vypisovat obsah příkazů volání funkce **printf()** pro vyluštění výše uvedené deklarace, uvedu jen to nejnutnější. Pro zjištění adres, které nás zajímají, jsem po chopitelně použil formát **unsigned int**. K argumentům hned připojuji výsledný výpis:

```
data=65359          *data=48512
&*data=65359        &**data=48512
&data[0]=65359      data[0]=48512
                    &data[0][0]=48512
```

Ukazatel na pole data sídlí na adrese 65359 a svým obsahem ukazuje na domovníka dlícího na adrese 48512. Povšimněte si jedné zajímavé zákonitosti. Hvězdička nás vždy dostává na nižší úroveň, tedy tam, kam ukazatel ukazuje, což je sídlo domovníka. Operátor & nás oproti tomu vede k vyšší úrovni - od domovníka k ukazateli. Jsou-li oba operátory (tj. & i *) vedle sebe, navzájem se ruší. To platí i v případě sledu *&.

Další zákonitost - stejný vliv jako * tu má použití hranatých závorek s referencí na prvek 0. Povšimněte si toho ve druhém sloupci - z tohoto pohledu jsou páry na řádkách 1, 3 a 2, 4 identické. Je to proto, že tu platí zákonitost adresové aritmetiky. ***data** je totéž, co ***(data+0)**.

Uvedenou deklarací pole data se neuložilo jen inicializované dvourozměrné pole, ale vytvořilo se i pole ukazatelů na začátky jednotlivých řetězců (na jejich první znaky).

Jak už víte z předchozích částí, adresa názvu proměnné (pamatujete na zápis &proměnná?) je adresou jejího uložení v paměti. V případě naší deklarace pole **data** je už samotný název pole adresou uložení prvního ukazatele na znakové pole (ukazatele na první řádku). Název **data** je totiž totéž, co &data[0] (čili doslova a názorně adresa ukazatele na první řetězec pole). Názvem pole tak můžete předávat jeho adresu ostatním funkcím.

Když je pole data polem ukazate lů, kde je další?

&data[1]=65361 resp.: data+1

Kompilátor uložil ukazatel na 2. prvek jednoduše o 2 bajty výš. Atd. Tady už o vzájemném "rušení" unárního operátoru a hranatých závorek nemůže být řeči. Výraz:

***(data+1)**

poskytne obsah druhého ukazatele (adresu domovníka druhého řetězce). Kulaté závorky jsou tady nutné, protože zápis ***data+1** by jen přičetl jednu k adrese, která je obsahem ukazatele na prvního domovníka. Kdežto ***(data+1)** dle adresové aritmetiky Céčka přičte 2 (počet bajtů typu **int**, neboli **sizeof(int)**) k adrese **data** a pak dostaneme obsah adresy dalšího ukazatele v poli ukazatelů. (Adresová aritmetika viz Mikrobáze 2/89, str.10, 2.sloupec.)

Sousedi na 1.řádce obou sloupců mají tento význam - **data** poskytuje adresu uložení prvního ukazatele z celého pole ukazatelů, ***data** míří na (ev. čte) obsah této adresy.

Pokračujme dál ve výpisech obsahů adres, tentokrát ve formátu %c pro výpis znaků:

```
data[0][0]=a
*data[0]=a
**data=a
```

Ve všech případech jde o výpis prvního znaku prvního řetězce v poli. ***data** si můžete převést na tvar **(*(*data))**. Rozvoj probíhá zevnitř. ***data** požaduje obsah prvního ukazatele typu **int** (=48512). Pak nastoupí požadavek *48512 (teoretický zápis), čili obsah (zde - už typu **char**) adresy domovníka. A ten sedí na písmenu a (první znak řetězce alfa).

Pro výpis řetězce (formát %s) opět musíme o jednu úroveň zpět:

***data=alfa**

Zkuste uhodnout, co nastane po příkazu:

```
printf("%c",*++(*data));
```

Záhadný výraz opět rozvineme zevnitř. ***data=48512** coby obsah adresy ukazatele, **++48512=48513** a nakonec ***48513=1** jako 2. znak řetězce **alfa**. Stejný výsledek dá **printf()** s argumentem ***++data[0]** ve formátu **%c**. Z toho je patrno, že první operace přečtení obsahu obou adres, na nichž sedí ukazatel, je celočíselná (int). Teprve potom nastupuje přečtení obsahu jediné adresy domovníka.

Dejme tomu, že by v přímém sledu po předchozím příkazu následoval:

```
printf("%s",*data);
```

Uhádnete, co se stane? Vypíše se: lfa. Proč? Protože předchozí příkaz zvýšil obsah adresy ukazatele. Na adrese 65359 už není původních 48512, ale 48513. Můžete se o tom přesvědčit jedním z výše uvedených způsobů. Stejně jako **++(*data)** zvýší obsah prvního ukazatele **++data[0]**. Ovšem další ukazatele zvýšíme už jen příkazem **++data[1]**, **++data[2]** atd.

Takovéto přímé zvyšování obsahu ukazatelů může mít své neblahé následky. Proto je lépe používat proměnné v hranatých závorkách (např. **data[x][y]**). Tak program odkazujeme na určitý znak pole beze změny obsahu ukazatelů.

Jinou možností je vytvoření kopie obsahu pole s jinými ukazateli. S nimi pak můžeme provádět, co libo a opustit je třeba i ve stavu "značného nepořádku". Příklad:

```
#define POCPRVKU 4
char *data={"alfa","beta","gama","delta"};
main()
{char *a[POCPRVKU]; int i;
for(i=0;i<POCPRVKU;i++)
  a[i]=data[i];}
```

V poslední řádce programu jsou novému poli ukazatelů, přesněji - obsahům ukazatelů nového pole postupně předávány adresy domovníků všech čtyř řetězců pole data. Na důkaz:

```
data=65359      a=65345
  starý           nový
ukazatel na řetězec 0 pole data

*data=48489    *a=48489
```

Adjektivum starý tu neznamená, že by kopírování domovnických adres nepřežil; existuje i nadále. S ouhrnně: **a[0]** až **a[3]** jakož i **data[0]** až **data[3]** ukazují postupně na adresy těchže prvních znaků jednotlivých řetězců, uložených od domovnických adres: 48489, 48494, 48499, 48504.

Ukazatele **pole data** (čteno pomocí **&data[0]** až **&data[3]**) jsou na adresách 65359, 65361, 65363, 65365, ukazatele **pole a** na adresách 65345, 65347, 65349, 65351.

Tak byla vytvořena dvě rozdílná pole ukazatelů,

která odkazují na jedno a totéž znakové pole.

Před vstupem do praktičtější části výkladu upozorňuji na to, že ne vždy poskytne výpis obsahů adres pomocí operátorů & a * obraz skutečnosti. V případech nejistoty radši použijte funkci peek() pro čtení obsahu jednotlivých adres:

```
typedef char *charptr;
peek(adr)
int adr;
{return *cast(charptr)adr;}
```

Tuto funkci můžeme volat i s argumentem absolutní adresy - např. obsah=peek(60000);. Použití konverze cast a vytvoření nového názvu typu pomocí typedef viz Mikrobáze 6/89.

Jako intermezzo ukazatelové kalvárie malá hádanka na procvičení naběhlého:

```
main()
{static char *včera="Bejvávalo bejvávalo dobře";
static char *teď="Už ani ta budoucnost
                  není, co bývala.";
static char *zítra;
zítra=teď=včera;}
```

Co bude výsledkem příkazu:

```
printf ("%s\n%s\n%s\n", včera, teď, zítra);?
```

A proč? Jak se budou/nebudou lišit ukazatele a domovníci?

Další program je ukázkou prostého abecedního třídění dvourozměrného znakového pole. Program porovnává znakové řetězce tvořící řádky pole. Pro třídění je použit algoritmus bubble sortu. Sestupně jsou porovnávány vždy dvě sousední řádky, znak po znaku. Když má znak vyšší řádky nižší hodnotu, obě zůstávají na místě a algoritmus pokračuje porovnáváním nižších dvou řádek. V případě vyšší hodnoty znaku na vyšší řádce se obě řádky prohodí. Je-li počet řádek n, první počet porovnání je n-1. Protože "nejtěžší" řádka vždy probublá na své místo, každou další srovnávací sekvencí se počet porovnání (proměnná počpor) o 1 snižuje, dokud celá jedna sekvence neproběhne beze změny. Ta je indikována proměnnou indizměn (0 znamená beze změny, při 1 došlo ke změně).

Program úspěšně řeší i výskyt většího počtu stejných řetězců. Především však poukazuje na možnost výměny ukazatelů místo výměn celých řetězců. Zvláště v případě řetězců o nestejné délce by jejich výměny vyžadovaly mnohem víc operací a času programátorského o i programového.

```
#define MAXPRVEK 6 /* index nejvyšš. prvku */
char *jména[]={"Eva","Jana","Hanka","Blanka",
               "Manka","Fanka","Kunhuta"}
int indizměn; /* indikátor výměn řádek */
main()
{int počpor; počpor=MAXPRVEK;
while(počpor-- > 0)
  {indizměn=0;
  porovn(počpor);
  if(!indizměn) break;}
počpor=0;
while(počpor.=MAXPRVEK)
 printf("%s\n",jména(počpor);}

porovn(x)
int x;
{int i,j; i=0;
for(;x>=0;x--)
 (for(j=0;jména[i][j]==jména[i+1][j];j++)
  ;                          /* prázdný příkaz */
 if(jména[i][j]-jména[i+1][j]>0)
   prohoz(&jména[i],&jména[i+1]);  /* ukaza- */
 ++i;}}                            /* tele řádek */

prohoz(adr1,adr2)   /* výměna ukazatelů obou */
int *adr1,*adr2;                    /* řádek */
{int depo;
indizměn=1;
depo=*adr2;
*adr2=*adr1;
*adr1=depo;}
```

Řádky se porovnávají ve funkci porovn(). Když se objeví požadavek na výměnu řádek, program zavolá funkci prohoz(). Pomocí proměnné depo se vymění (doslova přepíší) ukazatele na oba řetězce pole jména. Program tedy pracuje tzv. destruktivně - původní sled obsahů sídel ukazatelů je nahrazen novým (ale samotné řetězce zůstávají v paměti na svých místech).

Při psaní programu jsem se dopustil chyby - ve funkci prohoz jsem deklaroval *depo a při výměně ukazatelů jsem používal *depo (správně jen depo). Na konci měly oba ukazatelé stejný obsah, ukazatel na jeden řetězec "zmizel". Víte proč?

Abychom nevyšli z tréninku i v jiných oblastech, připojuji obměnu funkce porovn() s použitím cyklu do-while (místo for):

```
porovn(x)
int x;
{char a,b; int i,j; i=0;
for(;x>=0;x--)
  {j=0;
  do {a=jména[i][j];
      b=jména[i+1][j];
      j++};
  while(a==b);
  if(a-b>0)
    prohoz(&jména[i],&jména[i+1];
  ++i;}}
```

Další program jsem převzal z céčkové bible Kerninghana a Ritchieho Programovací jazyk C, která byla v srpnu k dostání i v Praze. Myslím , že těžko hledat lepší krátký příklad pro procvičení pole ukazatelů. Program čte vstup z klávesnice funkcí getline(), volané z readline() na ř. 18. V úvodu programu (ř. 2) je definován max. počet tříděných řádek číslem 100 a jejich celková délka je stanovena před funkcí readline() na 1000 bajtů (ř. 11). Kdykoli dojde k přetečení těchto hodnot, program se vrátí s chybovým hlášením.

Na ř. 16 je deklarováno pole line[MAXLEN] (tedy bez ukazatele). Znaky z klávesnice toto pole postupně zaplňují ve funkci getline(), které je adresa domovníka předána hodnotou line z readlines() , aby ji v getline() převzala adresa s pole char s[]. Když je řádka ukončena znakem '\n' (new line - u Spectra ENTER), je zavolána funkce *alloc(), aby určila nový ukazatel na novou řádku (čili na její začátek) v bufferu allocbuf[ALLOCSIZE]. Poté je řádka z pole s, resp. line okopírována do zmíněného bufferu. A tak dále, dokud buď nestisknete dvakrát po sobě ENTER, nebo nepřejedete definované hraniční konstanty.

Třídění ve funkci sort() obstarává algoritmus pana Shella. Prohoz ukazatelů na konci funkce je v principu týž jako v předchozím programu. Setříděné řádky nakonec vypíše funkce writelines(). *lineptr++ na jejím konci postupně skáče po ukazatelích jednotlivých řádek a printf() tiskne jejich obsah.

Všude, kde se vyskytuje přenos adresy pole lineptr, je předávána přímo její hodnota - to znamená, že se nevytváří nový ukazatel na pole lineptr. Proto se také nikde nesetkáte se změnou obsahu tohoto ukazatele, aby nedošlo k nevítaným událostem. Na počátku je definován pouze maximální počet ukazatelů na stejnojmenné znakové pole (řádky 2 a 4). Obsah těchto ukazatelů je do nich postupně zapisován na ř. 26 po zkopírování ukonče-

ného řetězce z pole s v getline(), resp. z pole line v readlines() do pole allocbuf z funkce *alloc().

Jediné místo, kde je přepis obsahu ukazatelů žádoucí, je pochopitelně funkce sort().

Novinkou je tvar zápisu char *alloc() - vyjadřuje, že tato funkce vrací u kazatel na typ char, což už vlastně víme. K hvězdičkovaným funkcím blíže v dalším pokračování.

```
 1  #define NULL 0
 2  #define LINES 1000
 3  main()
 4  {char *lineptr[LINES];
 5  int lines;
 6  if((nlines=readlines(lineptr,LINES))>=0)
 7    {sort(lineptr,nlines);
 8     writelines(lineptr,nlines);}
 9  else printf("Vstup příliš velký"\n");}
10
11  #define MAXLEN 1000
12  readlines(lineptr,maxlines)
13  char *lineptr[];
14  int maxlines;
15  {int len,nlines;
16  char *p,*alloc();line[MAXLEN];
17  nlines=0;
18  while((len=getline(line,MAXLEN))>1)
19    if(nlines>=maxlines)
20      return(-1);
21    else if((p=alloc(len))==NULL)
22      return(-1);
23    else
24      {line[len-1]='\0';
25      strcpy(p,line);
26      lineptr[nlines++]=p;}
27    return(nlines);}
28
29  #define EOF -1
30  getline(s,lim)
31  char s[];
32  int lim;
33  {int c,i;
34  i=0;
35  while(--lim>0&&(c=getchar())!=EOF&&c!='\n')
36    s[i++]=c;
37  if (c=='\n')
38    s[i++]=c;
39  s[i]='\0';
40  return(i);}
41
42  #define ALLOCSIZE 1000
43  static char allocbuf[ALLOCSIZE];
44  static char *allocp=allocbuf;
45  char *alloc(n)
46  int n;
47  {if allocp+n<=allocbuf+ALLOCSIZE)
48    {allocp+=n;
49    return(allocp+=n;}
50  else
51    return(NULL);}
52
53  strcpy(s,t)
54  char *s,*t;
55  {while(*s++=*t++) ;}
56
57  sort(v,n)
58  char *v;
59  int n;
60  {int gap,i,j;
61  char *temp;
62  for(gap=n/2;gap>0;gap/=2)
63    for(i=gap;i<n;i++)
64      for(j=i-gap;j>=0;j-=gap)
65        {if(strcmp(v[j],v[j+gap])<=0) break;
66      temp=v[j];
67      v[j]=v[j+gap];
68      v[j+gap]=temp;}}
69
70  strcmp(s,t)
71  char *s,*t;
72  {for(;*s==*t;s++)
73    {t++;
74    if(*s=='\0')
75      return(0);}
76  return(*s-*t);}
77
78  writelines(lineptr,nlines)
79  char *lineptr[];
80  int nlines;
81  {while(--nlines.=0)
82    printf("%s\n",*lineptr++);}
```

Družkou funkce alloc() a jí podobných je funkce free(), která uvolňuje paměť obsazenou polem , na které směřuje jeho ukazatel. Ten je také argumentem volající funkce. Kdybychom např. chtěli uvolnit paměť posledně načteného řetězce lineptr[i], zavolali bychom funkci příkazem free(lineptr[i]). Můžeme zrušit jakýkoli řetězec i uvnitř pole allocbuf, protože díky použití pole ukazatelů a využití funkce *alloc() máme ukazatele na všechny řetězce v poli allocbuf. Ovšem pak nám v allocbuf zůstane díra. Některé kompilátory mají schopnost takovou díru příště zalepit. Ty nejlepší mají v knihovně funkci realloc(), která projevuje snahu přeuspořádat děravé pole allocbuf tak , aby v něm pokud možno nezůstaly žádné díry. Upozorňuji, že k funkci free() neoddělitelně patří definice na ř. 42-44.

```
free(p)
char *p;
{if(p.=allocbuf&&p<allocbuf+ALLOCSIZE)
  allocp=p;}
```

Pro doklad funkce programu přikládám malý rozbor. Po zadání série jednoznakových řetězců v pořadí z, x, t, r, e jsem zjistil tyto hodnoty:

lineptr[0] až [4]=64339 až 64347 (vždy 2 bajty)
***lineptr[0] až [4] v readline() postupně ukazuje na adresy domovníků:**

64367, 64369, 64371, 64373, 64375 s obsahem:
z0 x0 t0 r0 e0

Po setřídění jsou hodnoty odebrané ve funkci writelines() následující:

lineptr[0] až [4]	*lineptr[0] až [4]	znak
64339	64375	e
64341	64373	r
64343	64371	t
64345	64369	x
64347	64367	z

Samotné uložení pole ukazatelů i znaků je beze změny, ale obsahy ukazatelů se přepsaly hodnotami adres jejich uložení v opačném pořadí, aby vyhověly abecednímu seřazení.

To, že rozdíly mezi sídly domovníků jsou všude 2 bajty, je jen tím, že jsem do každého řetězce uložil po jednom znaku, ke kterému kompilátor automaticky přiřazuje ještě binární nulu. Kdybych zapisoval 10 znaků na řádku, byla by vzdálenost mezi sousedními domovníky 11 adres. Zapisovat ovšem můžete řetězce libovolné délky (až do mezního tisíce znaků - ale i tuhle mez můžete podle přání a paměti vašeho počítače změnit).

Když budete mít chuť a čas, pokuste se program upravit tak, aby mohl třídit neuspořádaná znaková pole již uložená v paměti (třeba načtená z pásku), tedy nejen pole znak po znaku čtené z klávesnice.

Pro Hisoft C jsem musel v programu udělat jednu důležitou změnu. Na ř. 18 je na konci číslo 1 (původně tam je 0). S originální nulou jsem nemohl program přinutit k závěrečnému třídění po dvojím stisku ENTERu. Hisoft C vrací 1 z funkce getline(), i když by asi neměl - i++ příliš brzy z výší svou hodnotu a return(i) tak nikdy nevrátí nulu (jakožto indikaci nulového, neboli prázdného řetězce). Pokud to váš kompilátor dělá tak, jak je původně psáno, opravte si to zpět na nulu.

Další drobná oprava pro Hisoft C je na řádce 73. Příkaz t++ je v originále přirozeně uvnitř závorky na ř. 72, hned za s++ (odděleno čárkou). Bohužel, tuhle eleganci Hisoft C nestrpí. Nuceně tak přibyly i blokové svorky.

I když už jsme v Céčku dost daleko (tak někde kousek za prostředkem), tanec kolem ukazatelů ještě neskončil. Ovšem na druhou stranu - tohle horké téma máme už nejmíň z 80 procent za sebou. Ale i kdyby ze sta - můžete si být jisti, že se z vás operace s ukazateli budou sypat jako zrní z pytle? Při studiu každého jazyka se člověk nakonec vždy dostane do stádia, kdy samotné čtení literatury nestačí. Přichází osudové údobí trpělivého vstřebávání vytvářených návyků (léta dřiny) - aby to člověku "vlezlo pod kůži" a nemusel pokaždé nad vším neefektivně hloubat. Fortel se tomu říká. Takové staré , nemoderní slovo, že?

Ještě ke zhuštěné podobě zápisu, jaký na stránkách Mikrobáze používám. Jak jsem předpokládal, už se objevila jeho kritika ze strany vlastníků péčéček. Uvedenou podobu jsem (ač nerad) zvolil ze dvou důvodů. První - aby se toho do každé části seriálu vešlo co nejvíc (aneb - Kolik stránek má Mikrobáze?). Druhý - v editoru ZX Spectra s délkou řádky 32 znaků je program psaný podle pravidel céčkové estetiky ještě nepřehlednější než v podobě zhuštěné. Všichni, kdož mají lepší počítače, mohou bez nejmenších problémů programy psát, jak nejhezčeji budou umět. A když do Mikrobáze přijde někdy nějaký céčkový prográmek, jistě bude jeho hezké podobě vyhrazena náležitá plocha. Tenhle seriál si takový luxus na pár stránkách nemůže dovolit (byli bychom teprve někde v jeho polovině). Věřím, že za nejpádnější argument funkce zhuštěnýzápis() bude mnohými bratry Čechy uznáno, že mi za víc muziky vrací stejné peníze.

Příště dorazíme pole jejich číselnými druhy, podíváme se na ukazatele ve spojitosti s funkcemi a položíme závěrečný díl svých céčkových základů zvaných struktury a uniony. Pro ně a pro příklady užití knihovních funkcí nám zbývají poslední tři pokračování.

-elzet-

CP/M·CP/M·CP/M

Úprava ZX SPECTRUM 128+A 128+2

Ing. Jiří Janoušek

Popisovaná úprava výrazně rozšiřuje možnosti využití těchto počítačů. Umožňuje přepnout Spectrum do módu ALLRAM, takže je možno mimo jiné implementovat operační systém CP/M. Dále umožňuje též používat různě upravené ROMky, například LECROM, ISOROM, TURBOROM a další.

Při návrhu úpravy jsem vycházel mimo jiné i z požadavku co nejmenších finančních nákladů. Celou úpravu se nakonec podařilo realizovat se čtyřmi integrovanými obvody v úhrnné ceně asi 100 korun.

Pro přepínání pamětí je standardně využíván stránkovací port #7FFD. Jeho jednotlivé bity mají následující význam:

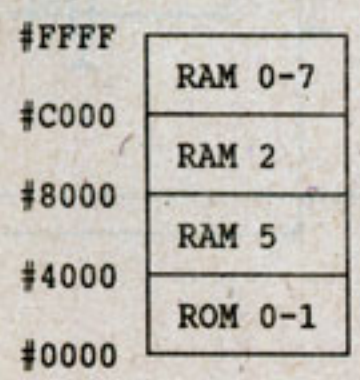
bity 0-2 : číslo RAM vhorní části paměti

bit 3 : použitá videoram 5 nebo 7. ULA zobrazuje tuto RAM ať je kdekoli (i mimo prostor adresovatelný procesorem.

bit 4 : přepíná ROM nová/stará

bit 5 : ochrana. Při nastavení zabrání přístupu na port #7FFD

Standardní mapa paměti vypadá takto:

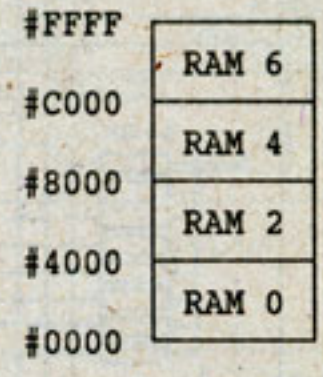


Nově přidané bity:

bit 6 : přepnutí do konfigurace ALLRAM

bit 7 : přepnutí RAM 0 do spodní části paměti místo ROM a zabránění jejího přepsání. Zbytek paměti je nastaven podle ostatních bitů v portu. Pouze bit 6 má větší prioritu.

Mapa paměti při nastaveném bitu 6 (ALLRAM):



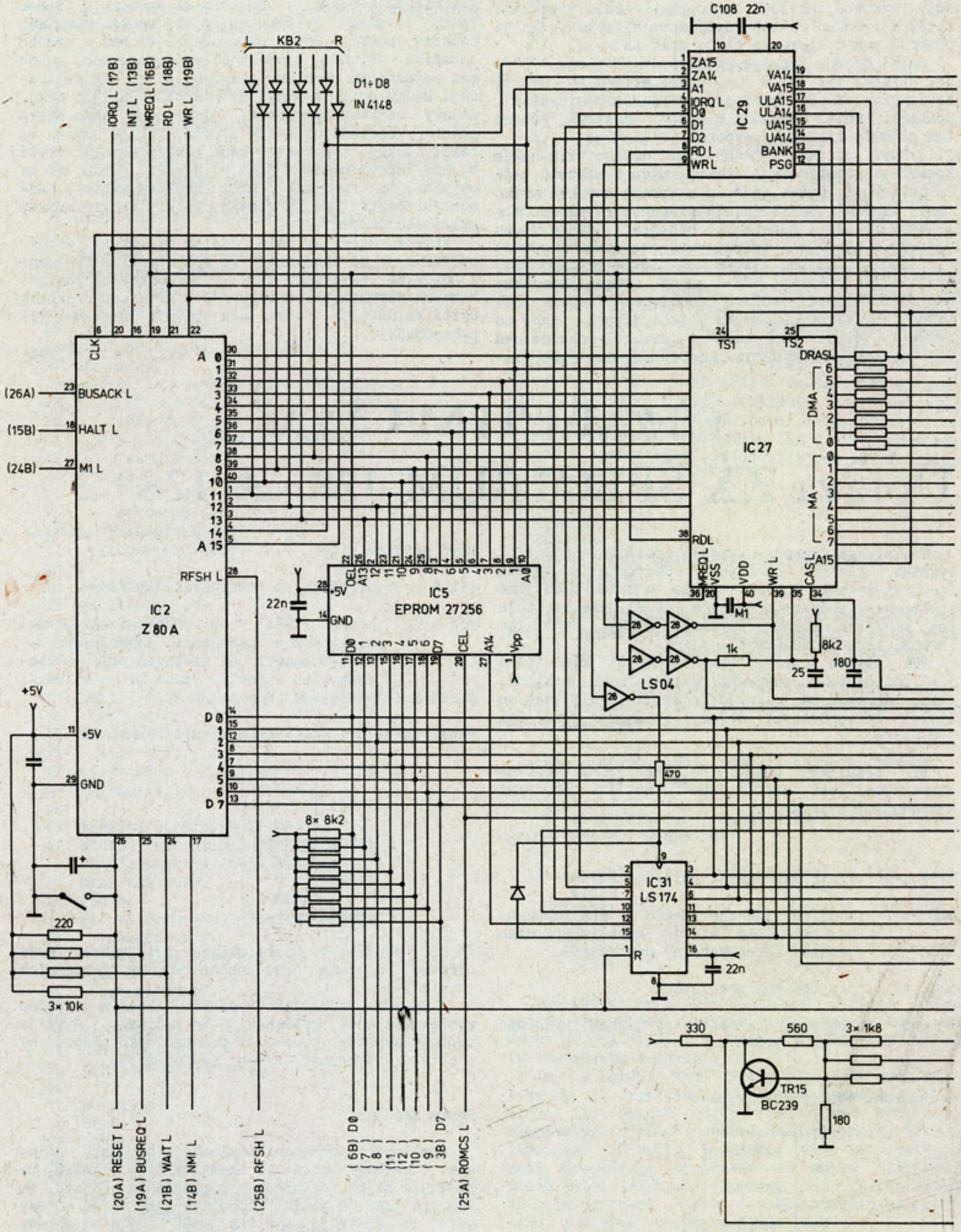
Pro použití v tomto režimu byly vybrány sudé stránky, protože jsou občerstvovány procesorem (viz dále).

V úpravě je použito vypínače, kterým je možno zablokovat nově přidané funkce počítače. Když je vypínač sepnutý, chová se počítač jako neupravený - bity 6 a 7 nemají na jeho funkci vliv.

Popis zapojení

Pro lepší pochopení následujícího textu je na obr. 1 celkové schema ZX Spectrum 128+. Myslím, že poslouží nejen zájemcům o popisovanou úpravu. Ve Spectru 128k je pamět rozdělena tak, že od adresy #0000 do #3FFF je umístěna pamět ROM a od adresy #4000 do #7FFF je umístěna videoram, stejně jako u Spectra 48k. Pro možnost implementace CP/M je nutno přepnout do celého prostoru paměti RAM.

obr. 1 - celkové schema počítače ZX SPECTRUM 128+



IC30 LS157
C109 22n
330
1N4147
A14
470
RDL
470
2× ZTX 313
180
1k
1k
100
100
470
470
IORQGE
PHI CPU
IORQ L
INT L
MREQ L
RD L
WR L
A15
A14
DMA7
RAS L
DMA6
DMA5
DMA4
DMA3
DMA2
DMA1
DMA0
ULA IC1
5×10k
KB0
KB4
KB1
L
R
VCC1
VCC2
15
GND
M1
22n
22M
22n
EAR
M1
680
EAR
10n
1k
MIC
M1
10k
MIC
M1
10k
4164
7 6 5 4 3 2 1 DI
DO
CASL WRL RASL
22n
4164
0 1 2 3 4 5 6 7
DI
DO
RAS L
WRL CASL
8×470
560
CAS L
DRAM WE
D0
1
2
3
4
5
6
D7
ROMCS L
VB
CL
R145
15k
C132
2n2
TR13
BC 239
R145
1k
28
VSYNC
R
G
B
SYNC
8M
8M
BRIGHT
17M
ANALOGUE CIRCUIT DIAGRAM
SOUND
IC32
AY-3-8912A
DA0
DA7
BC1
BDIR
BC2
+5V
A8
A9
A1
A2
A3
A4
A5
A6
A7
R
C
B
A
GND
22n
IC33
IC34
+12V (23A)
100
1k
KEYPAD
RS232
D23
47M/16V
47M
2k2
-12V
-8V
(20B)
1 - GND
2 - T×D (VSTUP)
3 - R×D (VYSTUP)
4 - DTR (VSTUP)
5 - CTS (VYSTUP)
6 - +12V

Adresový prostor se dělí na dva bloky paměti po 64kB. První blok (liché stránky) je občerstvován obvodem ULA. Ten při současném požadavku přístupu procesoru do této části paměti pozastavuje procesoru hodiny, stejně jako je tomu u Spectra 48k při přístupu do videoram. Sudé stránky jsou obhospodařovány výhradně procesorem, takže při jejich použití nehrozí nebezpečí porušeného časování. Jednotlivé 16k stránky jsou připínány v okamžiku, kdy A14 i A15 jsou v úrovni H, podle nastavení bitů 0-2 stránkovacího portu. Tato funkce je zajišténa zákaznickým logickým polem (IC29, nebo v ZX Spectrum 128+2 - IC7). Vlastní výběr provádí Sinclairův obvod (IC27) na základě signálů TS1 a TS2, které určují stránku v 64k bloku a vodiče ULA15, který vybírá 64k blok. Při ULA15 = H jsou vybrány stránky sudé, občerstvované procesorem. Vyplývá to z původní funkce obvodu ULA ve Spectru 48k, kdy vodič ULA15 vybírá videoram a ROM.

Princip úpravy spočívá v tom, že nově přidaný obvod překlene zákaznické logické pole, připojí A14 a A15 procesoru přímo na vodiče TS1 a TS2 Sinclairova obvodu a na vodič ULA15 přivede úroveň H. Tím procesor adresuje celý sudý 64k blok. To vše samozřejmě jen v případě nastavení bitu 6 stránkovacího portu.

Pro přepnutí RAM0 na místo ROM je použit stejný princip, ale musí být splněny následující podmínky:

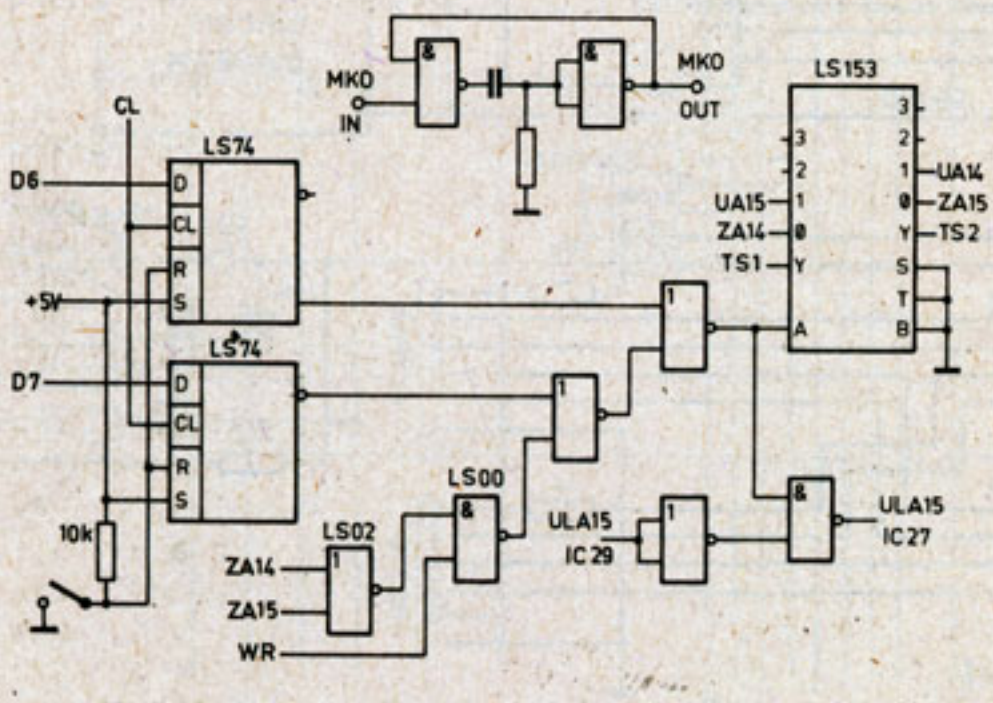
a) nastavený bit 7

b) A14 = L

c) A15 = L

d) WR L = H

Po splnění těchto podmínek se provede přepnutí do módu ALLRAM, takže RAM0 se objeví na adresách #0000 - #3FFF, ale při přístupu do jiných částí paměti, nebo při pokusu o zápis do prostoru #0000 - #3FFF zůstává počítač v původní konfiguraci pamětí, podle nastavení stránkovacího portu. V případě, že je RAM0 přepnuta stránkovacím portem do oblasi #C000 - #FFFF, není zde chráněna proti zápisu, takže je možné nahranou ROM upravovat za jejího běhu. Pro správnou funkci nahrané ROMky je třeba přepnout do horních 16k jinou stránku RAM, nejlépe RAM4 nebo RAM6.

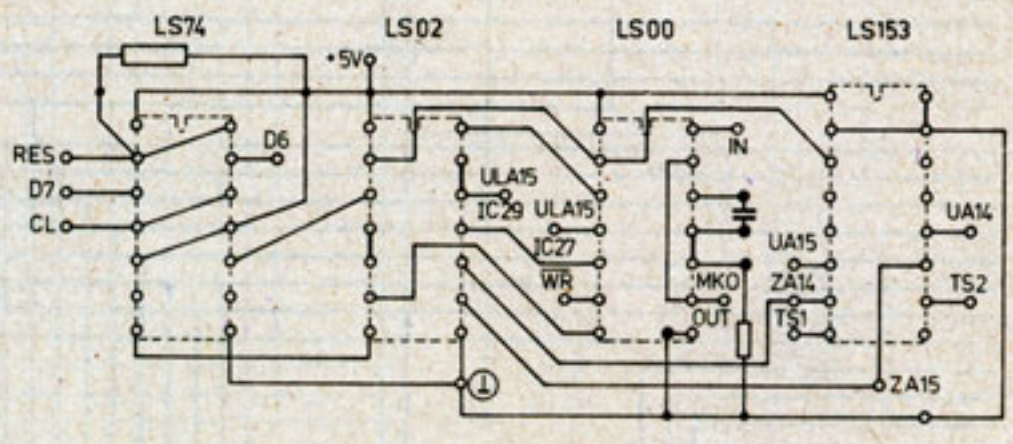


obr. 2 - shema zapojení doplňku

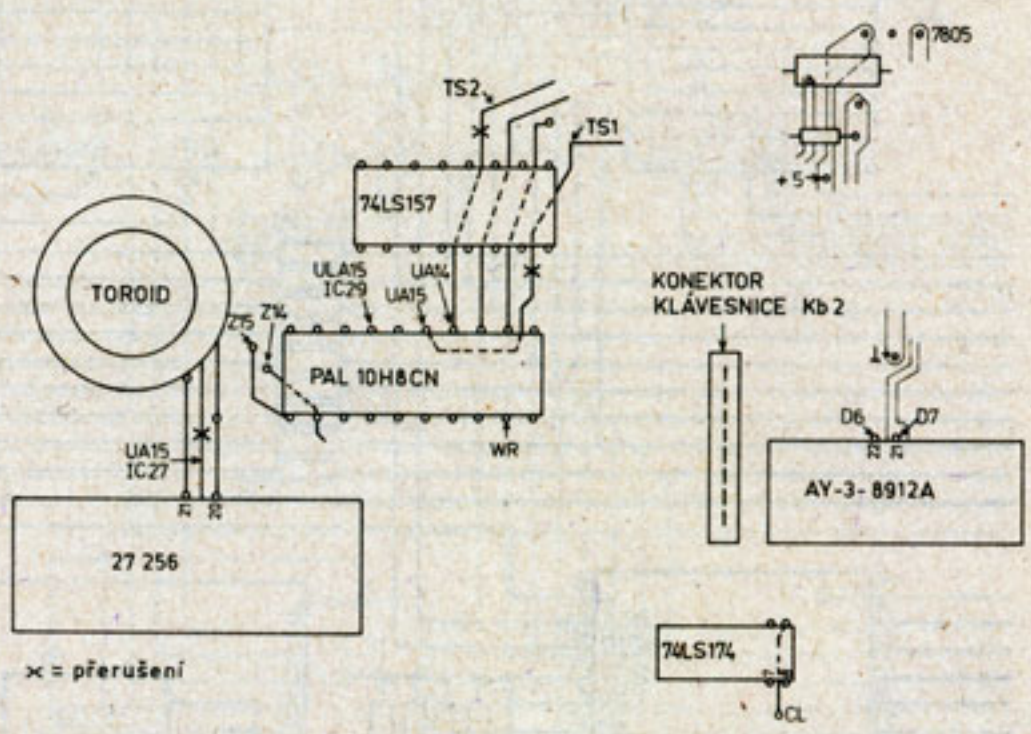
Schema zapojení doplňku je na obr. 2. Je zde použit obvod MH74LS74, který uchovává dva nejvyšší bity stránkovacího portu #7FFD. Jeho vstup hodinového signálu je spojen s hodinovým vstupem stávajícího portu, tvořeného obvodem 74LS174. Vývod reset je připojen na vypínač pro blokování funkce. Při sepnutí vypínače je obvod trvale nulován, čímž je zabráněno nastavení nejvyšších dvou bitů. Při použití tlačítka reset se počítač nevrací do původní konfigurace, ale zůstává přepnut podle nastavení bitů 6 a 7. Nedoporučuji však tlačítko reset použít, protože při resetu může dojít k přepsání části paměti v případě, že signál RESET přijde při aktivních signálech WR a MREQ.

Za obvodem 74LS74 je připojena logika, sloužící pro přepínání RAM0 na místo ROM a zabránění jejího přepsání. Další jednoduchá logika je použita pro přepínání buď původního signálu ULA15, nebo úrovně H. Zbývající dvě hradla jsou zapojena jako MKO pro RESET.

Pro přepínání UA14,15 nebo ZA14,15 na TS1,2 je použit obvod MH74LS153, který je na našem trhu běžně k dostání.

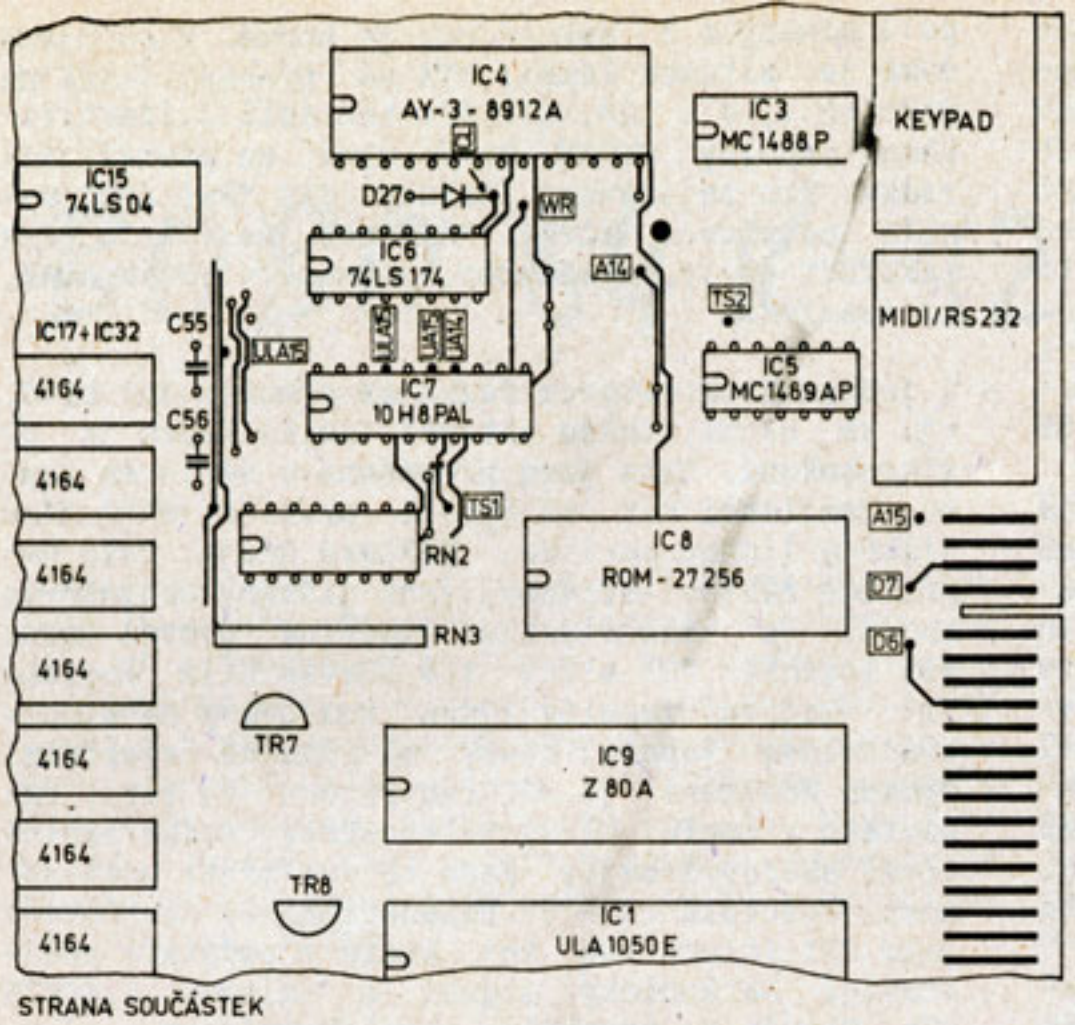


obr. 3 - návrh plošného spoje

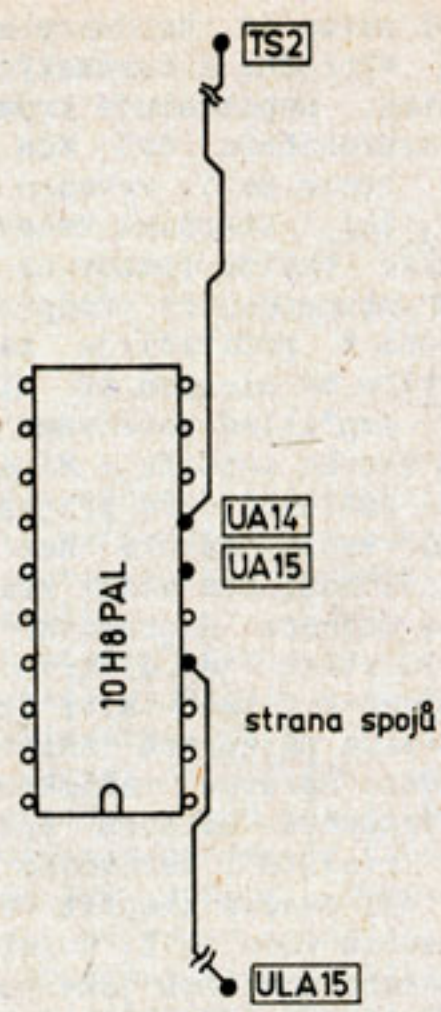


obr. 4 - plánek připojení do Spectra 128+

Ideový návrh plošného spoje je na obr. 3. Způsob montáže se liší podle toho, zda se jedná o 128+, nebo 128+2. Je nutno přeškrábnout 3 spoje a připájet 12 drátů. Pájíme přímo na nožičky IO, do prokovených otvorů a ve dvou případech přímo na spoje (ty je pak dobré zakápnout vhodným lepidlem). Na obr. 4 je plánek připojení do Spectra 128+, na obr. 5 a 6 do Spectra 128+2. Úprava chodila na první zapojení i s použitím partiových součástek. Používejte zásadně součástek LS (ALS), protože obyčejné TTL mají příliš velký vstupní proud a docházelo by k přetěžování obvodů počítače.



obr. 5 - připojení do Spectra 128+2



obr. 6 - připojení do Spectra 128+2

Celou úpravu je vhodné rozšířit o RESET, synchronizovaný s M1 procesoru. Toto zapojení však již bylo publikováno.

Pražský SINCLAIR klub 602. ZO připravuje pro členy Svazarmu montáž této úpravy odborníkem do počítačů zakoupených u nás, a to bez ztráty záruky.

Pozn. red. - Náš spolupracovník, Jiří Lamač, známý především svými úspěšnými implementacemi CP/M na ZX Spectrum a Sharp, pochopitelně ani v tomto případě neváhal a pohotově dodal verzi CP/M i pro tuto úpravu. Zatím je možno ji získat na schůzkách klubu uživatelů CP/M v Praze.

Příjem teletextu pomocí osobního počítače

Tomáš Laška (hardware) a Michal Matyska (software)

V Mikrobázi číslo 4 jsme vám slíbili popis nového adaptéru k počítači pro příjem Teletextu. Okamžitě jsme byli zasypáni spoustou dotazů, svědčících o velikém zájmu o tento obor. Vzhledem k současnému nevelkému početnímu stavu naší redakce (1 redaktor a ½ sekretářky) jsme nemohli na všechny dotazy odpovědět. Tímto se všem tazatelům omlouváme a místo odpovědi přinášíme slíbený popis. Autoři jsou studjící (na podzim už 4. ročníku) ČVUT FEL v Praze. Od předání čísla 4 do tisku se autoři se svou konstrukcí umístili jako druzí ve fakultním kole SVOČ a v celostátním kole byli třetí, čímž postoupili do mezinárodního kola SVOČ.

V souvislosti se zavedením vysílání Teletextu nás zaujala možnost využít pro zpracování teletextových informací počítače, který by tak umožnil vykonávat funkce zajišťované dekodérem (videoprocesoru a paměti ke zpracování a zobrazení textu). Je známo, že je u nás značný počet majitelů osobních počítačů a technické řešení příjmu signálu teletextu by jim tedy dalo možnost využít tuto službu a to s podstatně menšími náklady. Hlavně však o několik let dříve než bude k dispozici náš nový TVP či dekodér.

Využití počítače k příjmu Teletextu má několik výhod. Lze využít větší kapacitu paměti (dekodér má obvykle max. 8 kB), vybrané informace lze uchovat na vnější paměťové médium, jednoduchým programem lze informace převést do textového procesoru, nebo je rovnou vytisknout. To jsou výhody, které přináší sama hardwarová konfigurace počítače. Významné výhody přináší možnost softwarového zpracování signálu Teletextu. Protože na vstup počítače přichází informace v digitální formě, je možné pomocí vhodného software informace zpracovat. Jednou z hlavních výhod je možnost zobrazení znaků české aslovenské abecedy. Pokud dojde k sebemenší změně formátu dat, či k přechodu na jinou úroveň Teletextu, musí vlastník dekodéru vestavěného do televizoru (pokud dekodér už neumožňuje příjem vyšší úrovně) koupit nový dekodér, nebo TVP (pokud ovšem bude v prodeji). O tom se mohli přesvědčit majitelé televizorů Grundig, kterým se místo znaků s diakritickými znaménky zobrazují roztodivné "muří nohy". U počítače stačí drpobná úprava programu a je vše v pořádku.

Nemá cenu polemizovat nač vlastně Teletext přijímat - zdali to stojí za to. Výhody této služby už ocenila nejedna vyspělá země. Také u nás poskytuje Teletext velké množství zajímavých a aktuálních informací, přičemž se některé i několikrát denně aktualizují (i to lze vhodným programem zjistit a zajistit si aktualizaci archivovaných dat). Množství vysílaných strtánek se stále zvyšuje.

Obslužný program je dost obsáhlý a složitý. Byl napsán s důsledným využitím normy Teletextu. Zatím jsou hotové verze pro ZX Spectrum, IBM PC a Atari. Pracuje se však na dalších.

Náš adaptér Teletextu, který jsme navrhli, zkonstruovali a úspěšně vyzkoušeli koncem roku 1988, může spolupracovat s libovolným počítačem,

protože není vázán na hardware ani na OS. Adaptér je postaven výhradně z tuzemských, běžně dostupných součástek, nepotřebuje krystal a náklady na součástky nepřesáhnou 500 Kčs. U modernějších televizorů, které mají vyveden videovýstup, není třeba úprav. Ani u starších televizorů nedělá jeho vyvedení velké potíže (pozor na oddělení od sítě a příslušné bezpečnostní předpisy - pozn. red.). Pro připojení k počítači je zapotřebí paralelní rozhraní, které má alespoň 10 bitů vstupu a 1 bit výstupu. My například používáme interfejs s 8255 a na druhém vzorku interfejs Mirek.

Je nutné zdůraznit, že příjem teletextu klade na přijímač vyšší nároky než příjem barevného obrazu, signál musí být velmi kvalitní. Teletextový signál je zejména degradován odrazy a křížovou modulací. Teletext je distribuován sítí 2. TV programu, protože v této síti jsou modernější vysílače. Korekce parametrů těchto vysílačů, která byla provedena Správou radiokomunikací v součinnosti s Výzkumným ústavem spojů, přinesla též zlepšení v kvalitě barevného obrazu. Situaci v Praze by měl značně zlepšit nový vysílač Praha město (v provozu bude ve 2. pololetí 1990).

Pomocí našeho adaptéru jsme mohli podrobně analyzovat strukturu řádků použitých pro přenos teletextu. Zjistili jsme přitom některé zajímavé skutečnosti, které nejsou uvedeny v normě Teletext.

Následuje stručné shrnutí charakteristik Teletextu a z toho vyplývajících požadavků na adaptér.

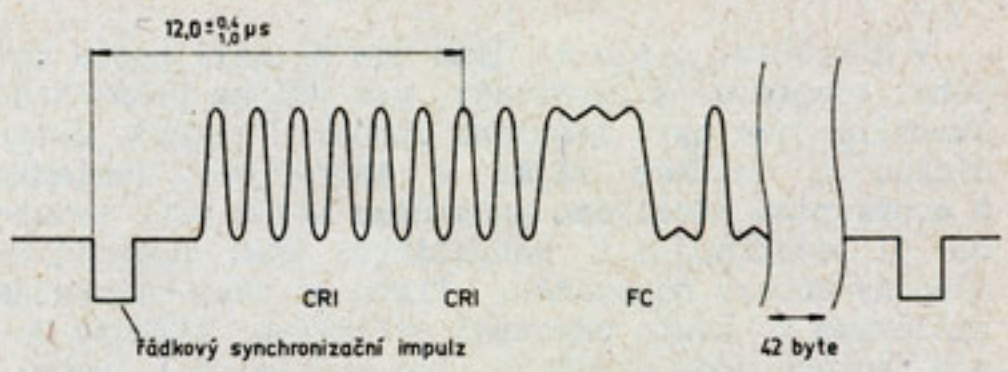
Základní charakteristika systému Teletext:

- přenosová rychlost TTX je 6 973 500 bit/s, což je srovnatelné s přenosovou rychlostí z pevného disku. Právě tato rychlost je při příjmu TTX zdrojem největších problémů (synchronizace, ukládání do paměti aj.). První datový bit trvá 144,44 ns. Když si uvědomíme, že data musí být vzorkována v polovině intervalu (optimálně) tj. 72,22 ns je patrné, že běžné integrované obvody TTL pracují na hranici zaručených parametrů. Při sekvenčním řazení obvodů tak snadno vznikají hazardy a jiné nepříjemné projevy nenulové doby zpoždění logických členů. Právě tomuto problému jsme věnovali při návrhu velkou pozornost.

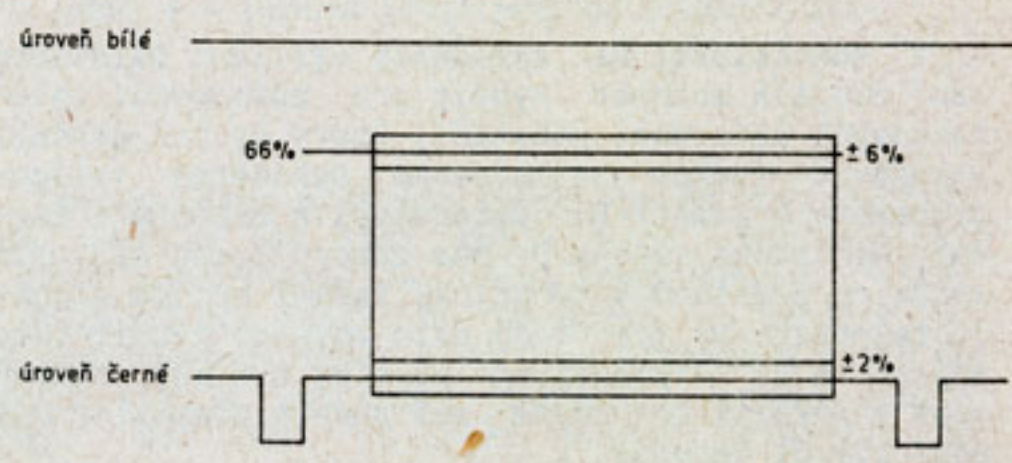
- Teletext se přenáší v zatemněných řádcích TV signálu, tudíž není při sledování pořadů vidět. Představu o signálu TTX můžeme získat pokud na 2. programu rozladíme vertikální (obrazovou) synchronizaci tak, aby se obraz zvolna pohyboval. Uvidíme mihotající se bílé body v černém předělu mezi snímky. Z přenosu TTX signálu vyplývá několik dalších údajů. V naší normě se obraz skládá z dvou půlsnímků. Jeden půlsnímek trvá 20 ms - to odpovídá frekvenci 50 půlsnímků za sekundu. Celý snímek se skládá z 625 televizních řádků. Frekvence řádků je 15 625 Hz to znamená, že jeden TV řádek trvá 64 mikrosekund. Teletext se vysílá v každém půlsnímku. Protože TTX se vysílá v zatemněných řádcích připadají v úvahu řádky 6÷23 a 318÷335. V praxi je ale část z volných řádků využita pro měrné účely (jsou na nich namodulovány měrné signály) a v soustavě SECAM, která je ČSSR používána, je 9 řádků použito pro identifikační impulzy SECAM. Tím se počet řádků použitelných pro přenos TTX značně zmenší. To je ale na úkor rychlosti přístupu k požadované informaci, neboť TTX stránky se vysílají cyklicky a s rostoucím počtem stran se prodlužuje doba přístupu k požadované informaci. V počátku vysílání se používaly pro přenos TTX pouze dva řádky v každém půlsnímku a to 19, 20 a 332, 333. Z toho můžeme spočítat že za sekundu se přenesly přibližně 4 stránky. Při vysílání například 500 stránek by v nejhorším případě byla doba přístupu 2 minuty a 5 sekund což je hodně. V současné době je situace lepší; TTX se přenáší ještě na řádcích 7, 8 a 320, 321 kde nahradil 2 identifikační impulzy SECAM, takže nyní se přenáší 4+4 řádky. Pro zajímavost Italský TTX "Rai uno" vysílá nejvíce - 12+12. Většinou se vysílá 8+8 (SKYTEXT aj.). To adaptér, popsaný v ročence AR, už nezvládá.

- V jednom Teletextovém řádku se přenáší 360 bitů, tj. 45 bajtů, takže aktivní délka řádku je 52 mikrosekund. Data jsou přenášena v sériovém tvaru, nejlehčí bit se vysílá první a data jsou jištěna lichou paritou. Některé údaje, jako například záhlaví stránky, jsou jištěny Hamingovým kódem. Ve videosignálu odpovídá úroveň černé ±2% logické "0" a 66% ±6% úrovně bílé logické "1". Datové impulzy jsou tvarovány na sinus-kvadrátový impulz, který má výhodné spektrum. Způsob modulace je NRZ (no return to zero) bez návratu k nule. Každý Teletextový řádek začíná třemi shodnými bajty. Jsou to dva bajty posloupnosti 10101010 tj. 55 hexadecimálně, označované jako CRI (clock run in), které v podstatě představují harmonický signál o kmitočtu 3,48675 MHz. Slouží k prvotnímu zasynchronizování adaptéru. Potom následuje důležitý byte 11100100 tj. 27 hexadecimálně označovaný jako FC (framing code), podle kterého je v podstatě identifikován platný teletextový řádek. Potom následují 2 bajty dalších informaie (pozice řádku ve straně aj.) a potom 40 bajtů textu. Podrobný význam jednotlivých informačních bajtů a jejich dekódování nejsou z hlediska hardware podstatné. Případné zvědavce bychom odkázali na březnovou přílohu AR Mikroelektronika, kde je vlastně velmi podrobně a přehledně popsána norma Teletextu, která o této problematice pojednává.

Na obrázku 1 je vidět struktura prvních tří bajtů teletextového řádku a na obrázku 2 jsou znázorněny tolerance úrovní v teletextovém řádku.



Obr. 1 - Struktura prvních tří bajtů TTX řádku



Obr. 2 - Tolerance úrovní TTX signálu

(pokračování příště)

BOBO 64 K (2)

Ing. Josef Blahůt

ing. Václav Daněček

2.1 Procesor

Mikroprocesor Z80A (pozice 602) má všechny kontakty vyvedeny na dva systémové konektory 1K a 2K, umístěné vzadu na desce. Pro snadnou výměnu v případě poruchy je zasazen v objímce. Hodinový signál 3,5 MHz se přivádí přes dva paralelně zapojené invertory s odporem na +5V, aby se dosáhlo požadované úrovně signálu v log 1. Nevyužité vstupní signály jsou ošetřeny odpory. Procesor je nulován po připojení napájecího napětí a po stisknutí tlačítka RESET.

2.2 Řízení pamětí

Mikropočítač je možno osadit dvěma paměťmi EPROM typu I2764 (603, 604), případně jednou paměti I27128 (do pozice 603). Jejich dekodér je v pozici 405. Propojka ve vstupu 1 určuje použitý typ paměti. V zapojení podle schematu je připojen adresový bit A13, což odpovídá použití paměti typu I2764. Do vstupu 6 dekodéru je připojen bit 7 registru pro vnitřní řízení (404) který, je-li v jedničce, odpojuje paměť EPROM. Dekodér může být blokován i vnějším signálem ROMCS ze systémového konektoru do vstupu 3. Obě paměti jsou v objímkách.

Paměti RAM jsou realizovány z dynamických obvodů 64k x 1 bit se sedmi, popřípadě osmibitovým obnovovacím cyklem. Jsou umístěny v pozicích 503-510 a jejich časování ovládá jednoduchý řadič složený z čítače 74LS193 (102) a paměti PROM MH74188 (103). Za pamětí je vyrovnávací registr 104, který odstraňuje hazardní stavy na jejím výstupu. Celý řadič je řízen frekvencí z krystalového oscilátoru 14 MHz. Průběhy řídících signálů na výstupu vyrovnávacího registru jsou na obr. 4. Čítač je inkrementován sestupnou hranou frekvence 14 MHz, tj. v čase t1, a nahrávání vyrovnávacího registru se provádí náběžnou hranou téhož signálu. Vzdálenost těchto dvou hran je 35 ns, zatímco zpoždění přes čítač a paměť dosahuje minimálně 50 ns. Proto se hodnota z adresy nastavená v čase t1 nahrává do vyrovnávacího registru až o 1,5 taktu později, v čase t2.

Nepožaduje-li mikroprocesor paměť RAM, je klopný obvod RAMFF (303/9) nastaven do jedničky, čítač 102 čte horní polovinu paměti PROM a probíhají zobrazovací cykly. Tato situace je na obr. 4 znázorněna plnou čarou a změny adres paměti PROM probíhají v pořadí, které je uvedeno pod bodem a). Na obrázku jsou dva zobrazovací cykly. Každý trvá 1,14 mikrosekundy (tolik času je třeba k zapsání osmi bodů na obrazovku). Během této doby se musí přečíst dva bajty z paměti (informace o hodnotě osmi bodů a atribut). Kromě toho musí být v každém zobrazovacím cyklu dostatek času na provedení paměťového cyklu mikroprocesoru. Proto se při čtení obrazových dat využívá stránkového režimu dynamických pamětí (Page mode). Předpokladem pro použití tohoto režimu je však shodná řádková adresa pro oba čtené byty. Proto jsou adresy jednotlivých řádků umístěny v paměti podle tab. 1. Z té je patrné, že spodních 8 bitů adresy (řádková adresa) je shodných pro oba čtené bajty (stav bodů i atribut). Další výhodou tohoto uspořádání je, že se během zobrazování mění řádková adresa v modulu 256, a tím se automaticky provádějí obnovovací cykly dynamické paměti.

Před začátkem čtení připojuje signál ENABLE VIDEO- (- značí negaci signálu) na adresovou sběrnici paměti multiplexery 208 a 309. Protože je současně v nule i signál SELECT ADR, připojí se na sběrnici řádková adresa. Potom signál RAS- aktivuje paměť. Se zpožděním daným RC členem R40, C8 přepíná signál SELECT ADR z invertoru 107/12 multiplexery a na adresové sběrnici se objeví sloupcová adresa. Zpoždění signálu SELECT ADR musí být takové, aby byla sloupcová adresa ustálená minimálně 10 ns před příchodem signálu CAS-. Ten přichází se zpožděním 70 ns za RAS. Na konci prvého ze dvou signálů CAS- se signálem PE1 nahrají data do sériového registru 310, 311 a 312. Signál VIDEO/COLOR připojí na sběrnici adresu atributu a na konci druhého pulsu CAS- se jeho sestupnou hranou nahrají data do vyrovnávacího registru barev (209, 210). Při režimu s vysokou rozlišovací schopností se signálem PE2 nahrává do sériového registru i druhý bajt dat. Signál RAS- končí 70 ns před koncem druhého CAS-. Před spuštěním dalšího čtení obrazových dat je prodleva 0,58 mikrosekundy, do které je možno vložit paměťový cyklus mikroprocesoru.

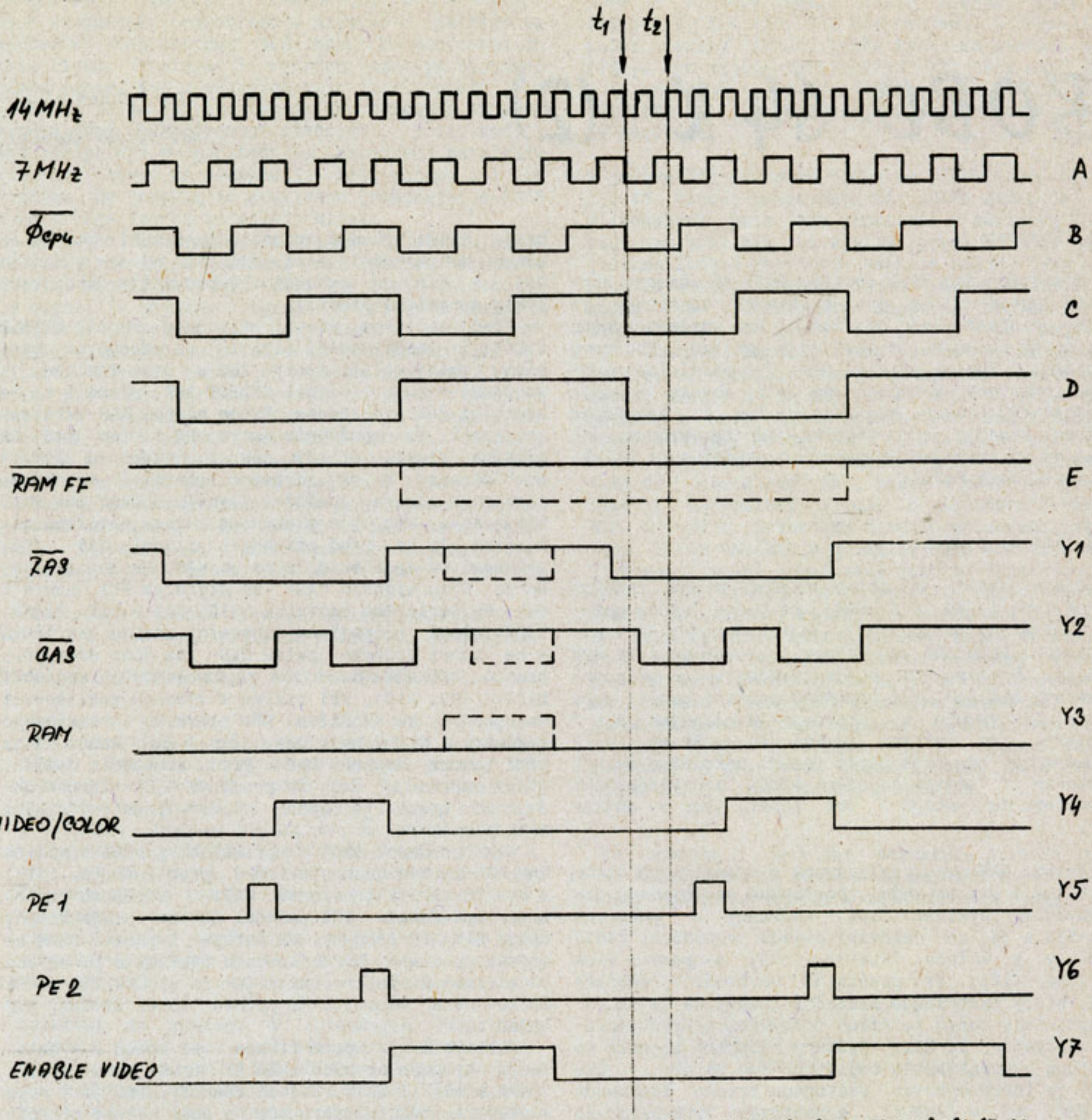
Mikroprocesor žádá o přidělení paměti signálem RAM ADR-. Ten generuje ihned přes invertor 101/2 a hradlo 105/1A1 signál WAIT-. Náběžnou hranou z výstupu čítače 102 (signál D) se nuluje klopný obvod RAM FF (303/9). Na vstupu E paměti PROM se objeví nula a v důsledku toho následuje po adrese 23 adresa 8. Tato situace je na obr. 4 vyznačena čárkovaně a odpovídá jí průběh adres uvedený pod bodem b).

Nastavení výstupu D čítače 102 spolu s vynulovaným klopným obvodem RAM FF zruší přes hradla 304/8 a 105/11 WAIT. Signál ENABLE VIDEO je v nule a otevírá multiplexery 502 a 605, které propojí adresovou sběrnici mikropočítače a paměti. Současně se generuje signál RAM, který otevře vyrovnávací registr 408. Signál RAM je aktivní při čtení i zápisu, ale data se z vyrovnávacího registru připojují na datovou sběrnici signálem LATCH ENABLE- z hradla 407/3, který se vytváří jen při čtení. Při zápisu generuje hradlo 105/6 signál WR RAM-.

Řadič vyšle posloupnost signálů RAS-, CAS- a SELECT ADR. Tím provede paměťový cyklus a po jeho skončení uzavře signál RAM vyrovnávací registr 408, do kterého se (v případě cyklu čtení) uložila přečtená data.

Čítač 102 pokračuje v adresování spodní poloviny paměti a provádí se další zobrazovací cyklus. Před příchodem náběžné hrany na výstupu D obvodu 102, zruší mikroprocesor žádost o paměť (signál RAM ADR- je 1) a nastaví se klopný obvod RAMFF 303/9. Na vstupu E paměti PROM se objeví jednička. V důsledku toho následuje po adrese 7 adresa 24 a celé řízení se vrací do druhé poloviny paměti PROM.

Přidělování paměti mikroprocesoru znázorňuje obr. 5, kde jsou naznačeny dva z možných způsobů synchronizace mikroprocesorových taktů.



a) posloupnost adres paměti PROM, když probíhá pouze zobrazování

b) posloupnost adres paměti PROM při žádosti procesoru o paměť RAM

Obr. 4 - řídící signály paměti PROM

2.3 Zobrazení

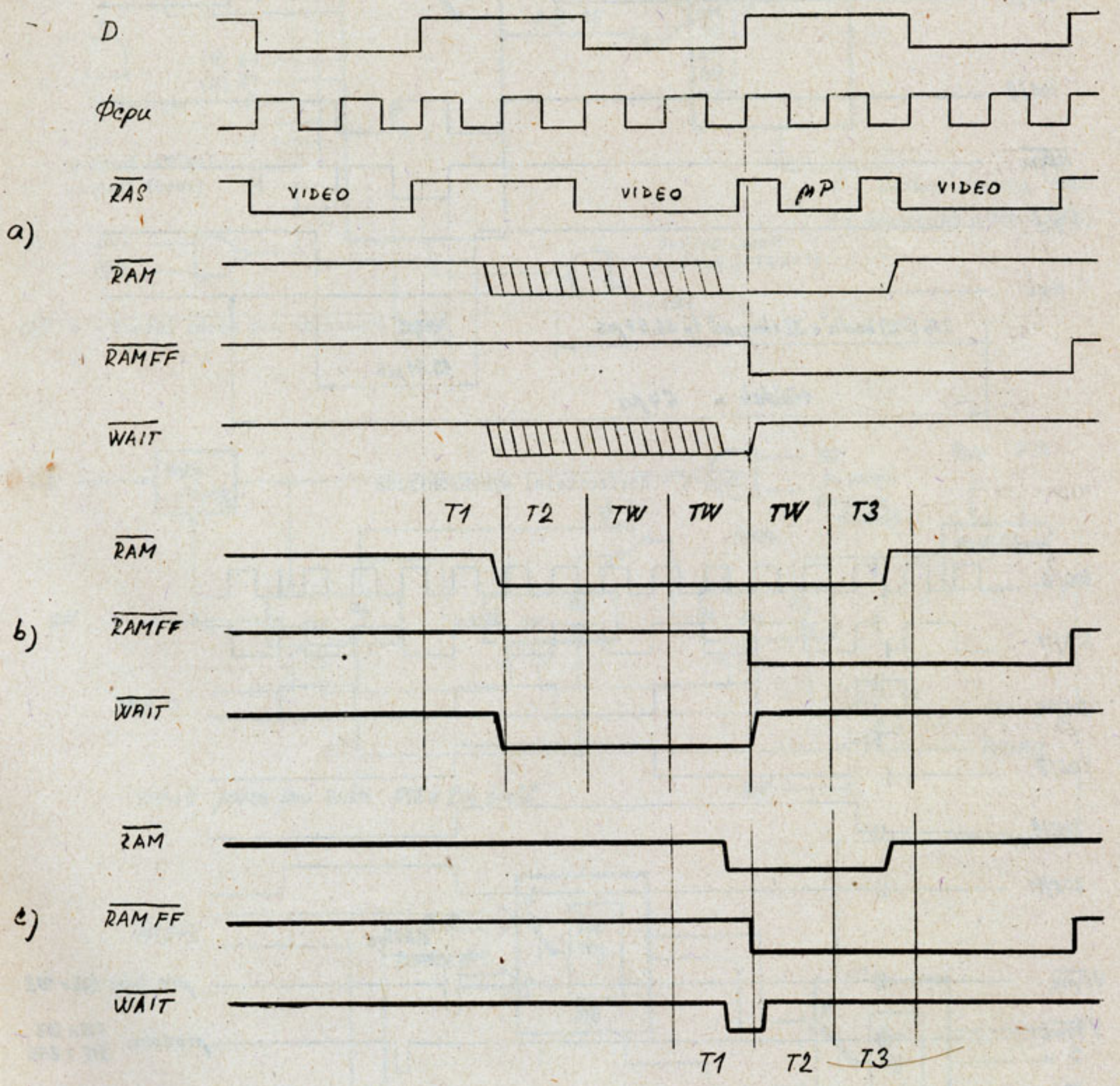
Úplný televizní signál tvoří směs synchronizačních pulsů a jasová složka. Synchronizační pulsy generují čítače 306, 307, 205 a 206 řízené signálem HCLK. Tento signál je jednou osminou základního zobrazovacího kmitočtu 7 MHz.

Čítač sloupců tvoří obvody 306 a 307. Jejich celkový dělicí poměr je 56, což představuje dobu cyklu 64 mikrosekund (jeden televizní řádek). Z výstupů čítačů jsou dekódovány horizontální zatemňovací (Hzat-) a synchronizační (Hsync-) pulsy. Jejich průběhy jsou na obr. 6.

Čítač řádků (205, 206 a nevyužitá dělička z čítače sloupců 307) pracuje s dělicím poměrem 312 (počet TV řádků). Tvar a způsob dekódování vertikálního synchronizačního pulsu je na obr. 7. Tranzistor T7 slouží k vystředění obrazu při režimu 512x256 bodů. Vertikální synchronizační puls se dále používá ke generování žádosti o přerušení (202/9) a po dělení šestnácti (201) vytváří signál 1,6 Hz, který se používá v bloku modifikace k invertování výstupního signálu (blikání vybraných částí obrazu).

Stav čítače řádků a sloupců se mění vždy po vyslání osmi (při zobrazení 512x256 bodů - šestnácti) bodů na obrazovku. Proto jejich výstupy slouží současně k adresování obrazové paměti. Z 56-ti stavů čítače sloupců se používají stavy 0-31 (32 sloupců po osmi, resp. šestnácti bodech) a ze 312-ti stavů čítače sloupců se, podle typu zobrazení, používají hodnoty 0-191, resp. 0-255. Mimo tyto vymezené stavy se generuje z hradla 304/12 signál BORDER-, který blokuje (na hradlech 106) nahrávání dat do sériového registru (/310, 311, 312) a na výstupy RGB připojí barvu pozadí. čtení dat z obrazové paměti probíhá stále (i když se čtená data dále nezpracovávají), protože slouží současně k obnovování dynamických pamětí.

Překódování jednotlivých stavů čítačů do adres pro obrazovou paměť provádějí multiplexery 208, 308, 309 a hradla 207 a 305, podle stavu bitů 5 a 6 (signály CW1 a CW2) vnitřního řídicího registru 404. Kódování se provádí podle tab. 1 a je názorně nakresleno na obr. 8. Signál VIDEO/COLOR přepíná před příchodem druhého pulsu CAS- multiplexery tak, aby se na sběrnici připojila adresa atributu.

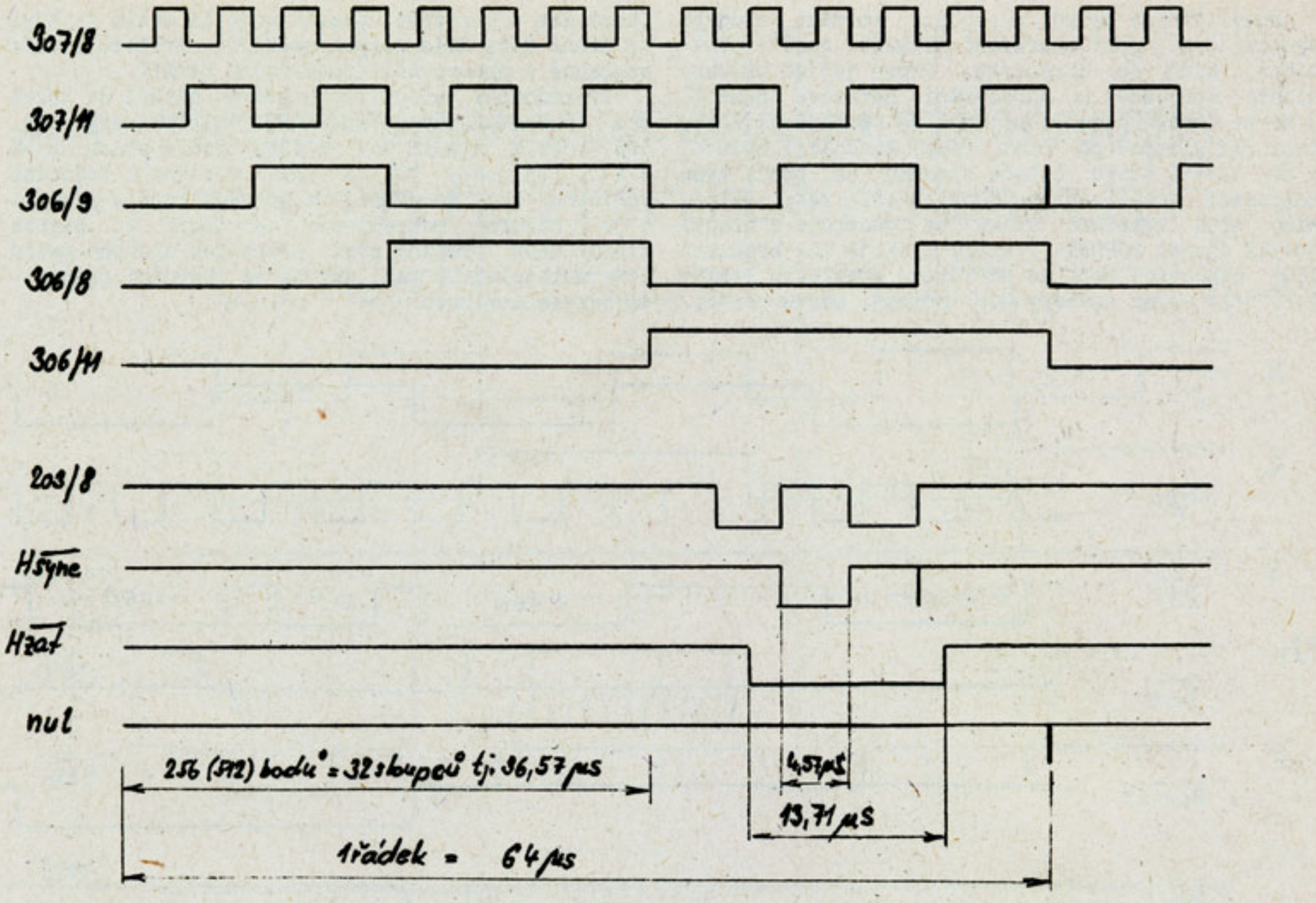


T1,T2,T3,TW - takty mikroprocesoru

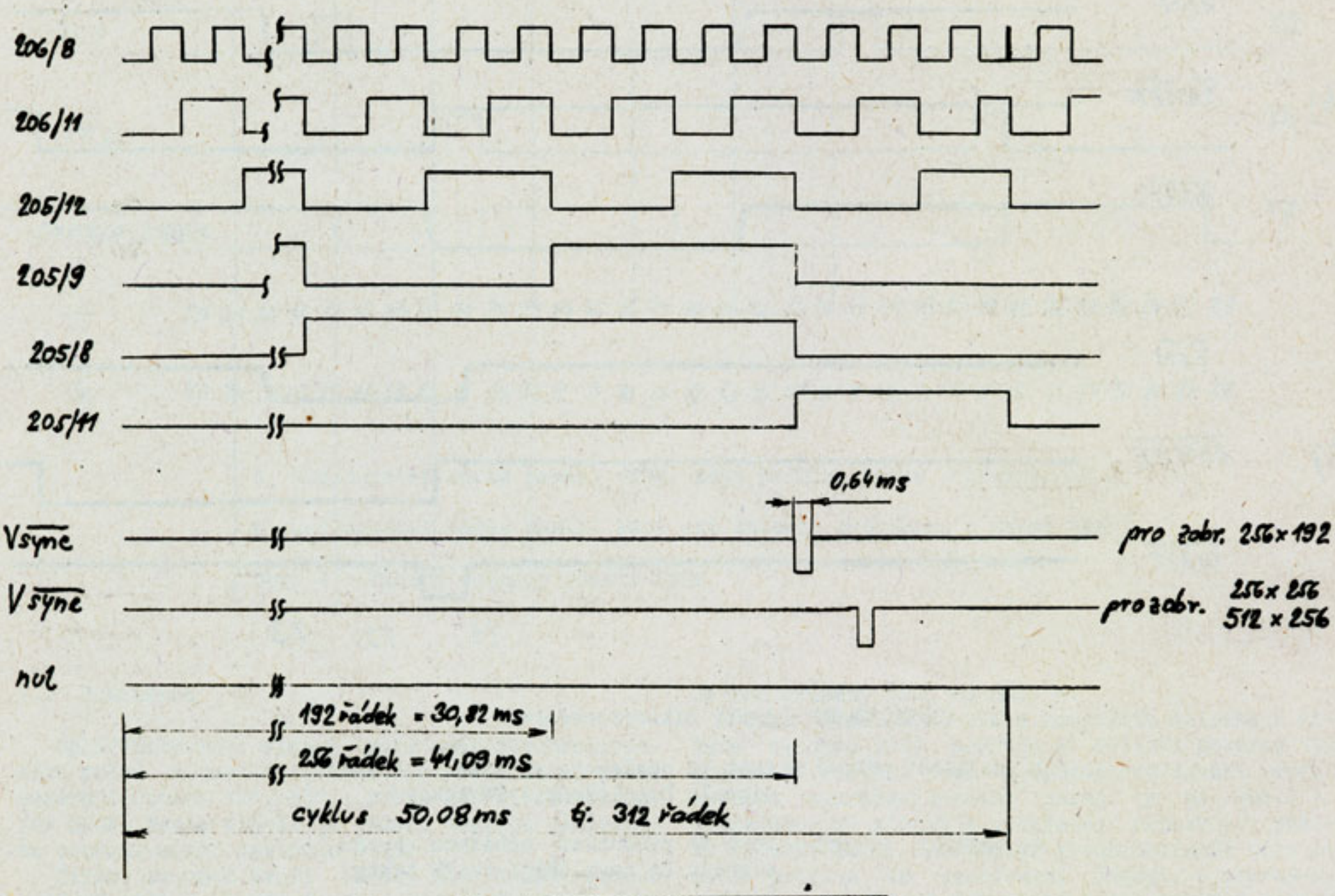
b) nejhorší případ - takt T1 následuje po náběžné hraně signálu D; vloženy 3 TW takty

c) ideální případ - takt T1 předchází náběžnou hranu signálu D; nevkládají se TW takty

Obr. 5 - přidělování paměti RAM

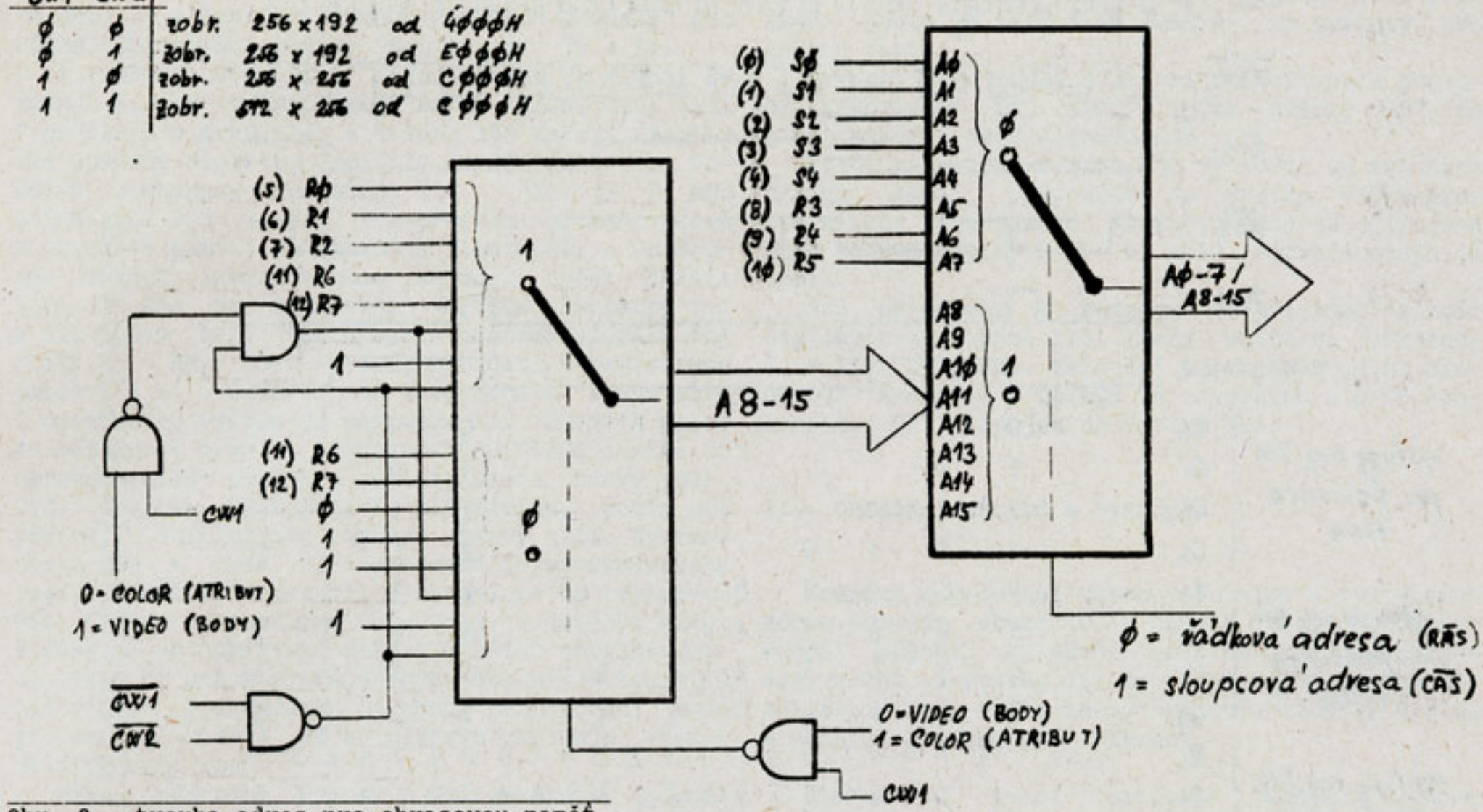


Obr. 6 - horizontální synchronizace



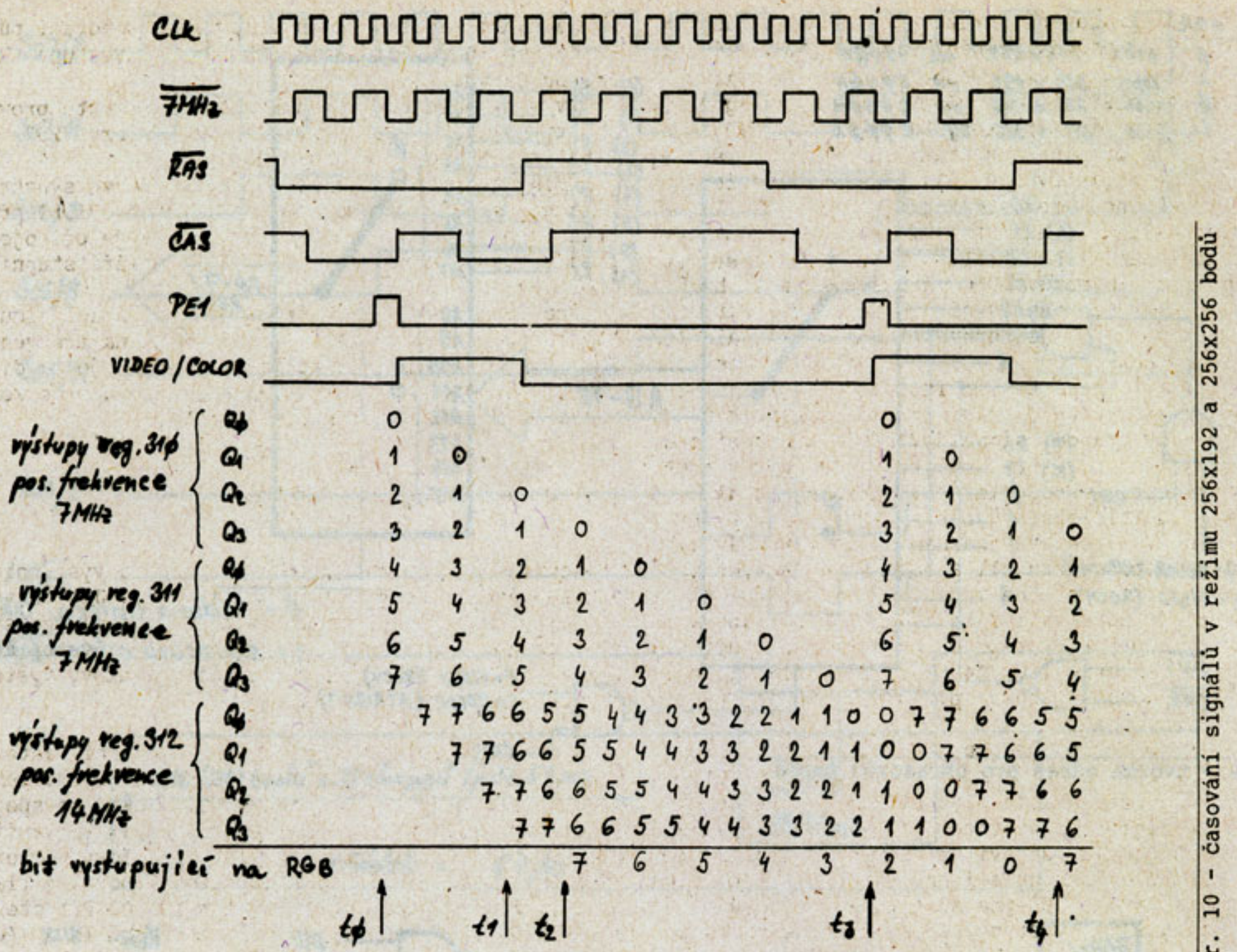
Obr. 7 - vertikální synchronizace

CW1	CW2	
0	0	zobr. 256 x 192 od 4000H
0	1	zobr. 256 x 192 od E000H
1	0	zobr. 256 x 256 od C000H
1	1	zobr. 512 x 256 od C000H

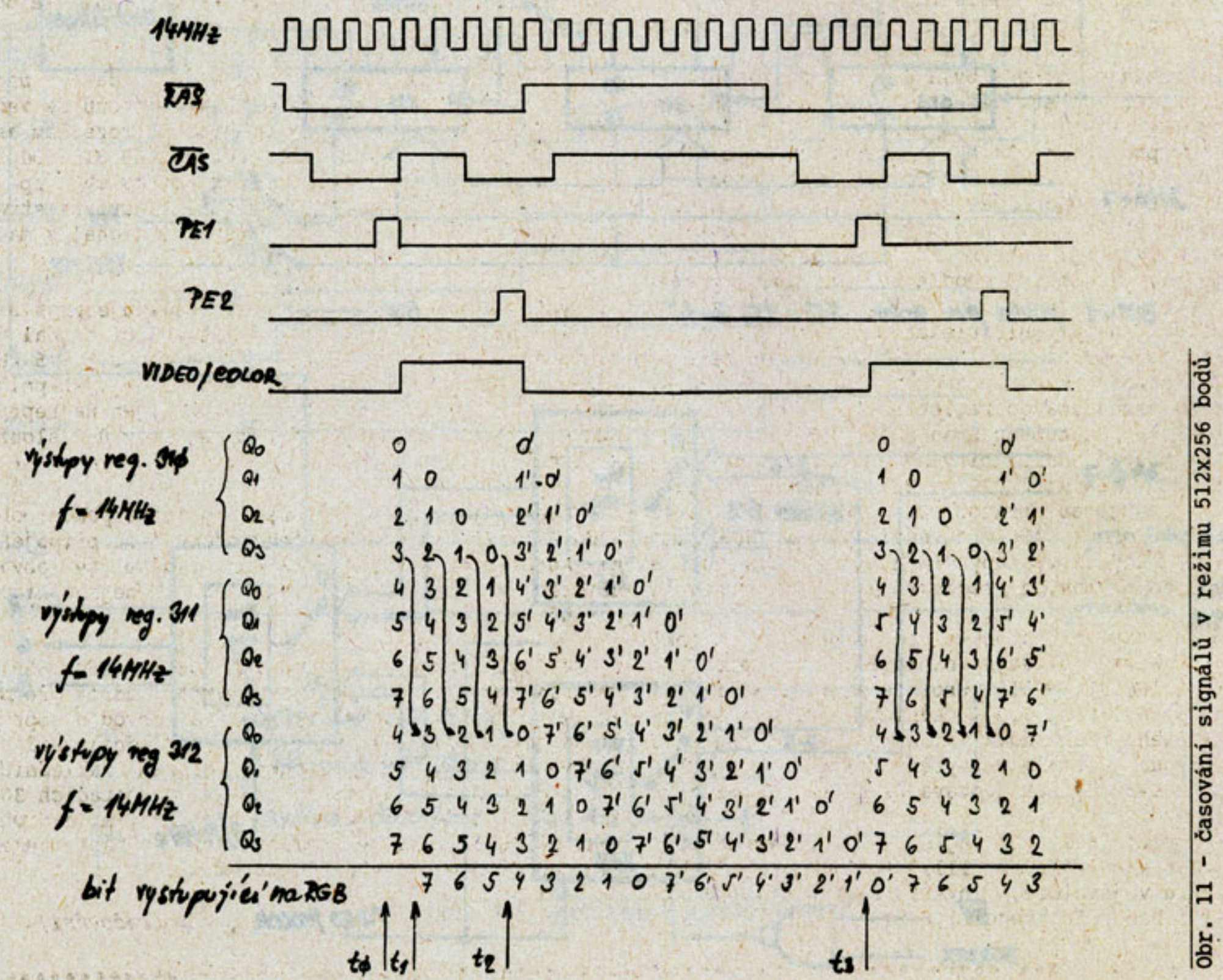


Obr. 8 - tvorba adres pro obrazovou paměť

Obr. 9 - zpracování obrazových dat



Obr. 10 - časování signálů v režimu 256x192 a 256x256 bodů



Obr. 11 - časování signálů v režimu 512x256 bodů

Způsob zpracování obrazových dat je znázorněn na obr. 9 a časové průběhy řídicích signálů pro různé zobrazovací režimy jsou na obr. 10 a 11.

V režimech 256x192 a 256x256 bodů je signál BW rovný 0, oba multiplexery barev (209, 210) jsou v poloze 0 a přepínač z hradel 108 zapojí všechny tři posuvné registry 310, 311 a 312 do série. Posuvná frekvence registrů 310, 311 je 7 MHz a registru 312 14 MHz. Po přečtení prvního slova z obrazové paměti se aktivuje signál PE1 a sestupnou hranou invertovaného signálu (přes hradla 110) 7 MHz se nahrávají sériové registry 310 a 311 (čas t0). Sestupnými hranami signálů CLK (312) a 7 MHz- (310 a 311) se data v registrech posouvají a v čase t1 se informace o stavu bitu 7 dostává na výstup Q3 registru 312. Okamžik potom se sestupnou hranou signálu VIDEO/COLOR zapíše do vyrovnávacího registru multiplexeru barvy (209, 210) atribut. Vystupující bit přepne, podle své hodnoty, multiplexer výběru barvy 211. Vybraná barva se v čase t2 zaznamená do vyrovnávacího registru multiplexeru 211 a objeví se na výstupech RGB. Takto se postupně vysouvají i ostatní bity. V čase t3 se nahrávají data z dalšího zobrazovacího cyklu do právě vyprázdněných sériových registrů 310 a 311. V okamžiku t4 končí zobrazování bodů přečtených v čase t0 a plynule navazuje výstup dalších osmi bodů.

Mimo pracovní plochu se generuje signál BORDER-, který přepojí multiplexery 210 a 211 do polohy 1 a přes hradla 106 blokuje nahrávání do sériových registrů. Protože je na sériovém vstupu registru 310 nula, objeví se po výstupu posledního bodu každého řádku nula i na výstupu Q3 registru 312, která trvale přepne multiplexer výběru barvy 211 do nuly a na výstupy RGB se dostane barva pozadí (viz obr. 9).

V režimu 512x256 bodů je signál BW roven 1 a posuvná frekvence registrů 310, 311a 312 je 14 MHz, protože pracuje zdvojovač kmitočtu z hradel 110. Zpoždění na R70 a C9 je nastaveno tak, aby střída frekvence 14 MHz byla 1 : 1 a sestupná hrana přicházela ve stejném okamžiku jako při frekvenci 7 MHz. První slovo z paměti se signálem PE1 (při sestupné hraně 14 MHz) nahrává v čase t0 do registrů 310, 311 a 312. Protože jsou BW i VIDEO/COLOR v jedničce, spojil přepínač z hradel 108 registry 310 a 311 do série a data v registru 311 se nevyužívají (obr.9). Na výstupu Q3 registru 312 je bit 7, který (podle své hodnoty) přepne multiplexer výběru barvy 211. Protože je BW v jedničce, jsou multiplexery barvy 210 a 211 přepnuté do polohy 1 a lze vybírat pouze ze dvou barev (černá a barva pozadí). Vybraná barva se v čase t2 zaznamená do registru 211 a objeví se na výstupu RGB. Sestupnou hranou 14 MHz se pak postupně vysouvají další bity. V t2 se do právě vyprázdněných registrů 310 a 311 nahrává signálem PE2 a sestupnou hranou 14 MHz druhé slovo z obrazové paměti. S frekvencí 14 MHz jsou bity dále vysouvány na výstupy RGB a v čase t3 navazuje zápisem do sériových registrů další zobrazovací cyklus.

Mimo pracovní plochu je generován signál BORDER-, který blokuje nahrávání do sériových registrů 310, 311 a 312. Protože signál BW přivádí na vstup registru 310/1 jedničku, objeví se na konci každého řádku nula i na výstupu Q3 registru 312 a přepne multiplexer výběru barvy 211 na nulu. Tím se na výstupy RGB dostává černá barva pozadí.

Bit vystupující z Q3 registru 312 je veden přes blok modifikace (110/3, 111/3 a 11/6). Je-li bit 7 atributu v jedničce, invertuje hradlo 110/3 vystupující bod s frekvencí 1,6 Hz.

Na konci každého řádku připojí zatemňovací puls Hzat- spolu se signálem BORDER- na výstupy RGB černou barvu.

Zvýšení jasu podle stavu bitu 6 atributu provádí tranzistor T4, který (přes odpory R57, 58 a 59) zvedá úroveň signálů RGB.

Úplný televizní signál je vytvořen ze synchronizační směsi, připojené na výstup VIDEO přes tranzistor T5 a jasové složky, která je připojena přes váhové odpory R60, 61 a 62 (vytváří stupnici šedi).

Pro připojení k běžnému TV přijímači slouží oscilátor z hradel 112, který kmitá na frekvenci I. - III. TV pásma. Kmitočet oscilátoru je na diodě D21 směšován s úplným TV signálem a přes kondenzátor C11 připojen na výstup VHF.

2.4 Ovládání vstupů a výstupů

Neúplné dekódování všech vstupních a výstupních adres provádí obvod 403 a hradla 402. Jejich výstupní signály ovládají vnitřní řídicí registr 404, vstup a výstup dat na magnetofon, vstup dat z klávesnice, volbu barvy pozadí, zvukový výstup a paralelní i seriové rozhraní.

Klávesnice tvoří maticové spojení datové a adresové sběrnice. Adresová sběrnice je oddělena diodami D10-D17. Při stisknutí tlačítka se spojí příslušný datový a adresový vodič. Je-li v okamžiku čtení klávesnice přes obvod 607 nulový některý adresový vodič A8-A15 a stisknuto libovolné tlačítko, přečte se na příslušném bitu 0. Při čtení se využívá skutečnosti, že při instrukci IN A, (C) vystupuje na vodičích A8-A15 adresové sběrnice B registr procesoru (viz čl. 1.4). Klávesnice je vyvedena na dva konektory, vnější a vnitřní. Adresová sběrnice vnějšího konektoru (4K) je oddělena obvodem 606.

Přes budič 607 se na bitu 6 čtou data z magnetofonu. Vstupní signál z magnetofonu se vede přes jednoduchou pásmovou propust s rozsahem asi 300 Hz - 5 KHz na vstup komparátoru MAB 311. Odpor R50 zavádí hysterezi komparátoru asi 40 mV a zpětná vazba s kondenzátorem C5 slouží k udržení střídy výstupního signálu 1:1. Výstupní signál z magnetofonu má mít úroveň asi 300 mV.

Výstup dat na magnetofon se realizuje přes bit 3 výstupního registru 313. Vystupující signál je tvarován pásmovou propustí s rozsahem 0,5 - 5 kHz a má úroveň asi 300 mV. Bit 4 tohoto výstupního registru je přes tranzistor T6 připojen na reproduktor, jako zvukový výstup a bity 0-2 slouží k určení barvy pozadí.

Paralelní rozhraní je realizováno s pomocí obvodu MHB 8255A a všechny jeho brány jsou připojeny na konektor 3K. Pro snadnou výměnu je obvod umístěn v objímce (signály na 3K nejsou nijak chráněny).

Sériové rozhraní používá obvod MHB 8251. Tranzisotry T3 a T4 na výstupu tvoří zdroj proudu 20 mA. Vstupní signál zpracovává obvod s optoelektrickým oddělovacím členem. Základní vysílací frekvence 76,8 KHz vznikne dělením základního zobrazovacího kmitočtu číslem 91 v čítačích 302 a 301. První je zapojen jako dělička 7-mi a druhý jako dělička 13-ti. Při příjmu každého bajtu generuje obvod nemaskovatelné přerušení.

(pokračování)

Postavte si s námi diskový řadič /3/

Daniel Meca

Minule jsme si povídali o TR-DOSu 5.xx a jeho využití v Basicu. Dnes vám nabízím novou věc - využití TR-DOSu ve strojovém kódu. Tímto způsobem se možnosti Betadisku zmnohonásobí.

Pokud je mi známo, nebyl dosud publikován přehled služeb DOSu. Původní výrobce dokonce ve svém návodu pečlivě zamlčuje, že vůbec tyto služby existují. Podle originálního návodu firmy Technology Research se má volat DOS ve strojovém kódu takto:

```
CHADD EQU  23645       ;Basicová syst. proměnná.
      ORG  xxxxx       ;Uložení tohoto kódu.
      LD   HL,(CHADD)  ;Dočasné uložení adresy
      LD   (TEMP),HL   ;z CHADD.
      LD   HL,SAVE     ;Adresa pro zápis
      LD   (CHADD),HL  ;do CHADD.
      CALL 15619       ;Volání DOSu.
      LD   HL,(TEMP)   ;Zrestaurování obsahu CHADD
      LD   (CHADD),HL  ;na původní hodnotu.
      RET

TEMP  DEFS 2           ;Místo pro dočasné uložení.

SAVE  DEFB 234         ;REM
      DEFB 58          ;:
      DEFB 248         ;SAVE
      DEFB 34          ;"
      DEFM "nazev"     ;název programu
      DEFB 34          ;"
      DEFB 13          ;ENTER
```

To sice v zásadě funguje, ale problémy vzniknou při pokusu zadat např. číslo řádku pro autostart, nebo adresu pro CODE.

Tento způsob přístupu ve strojovém kódu je mírně řečeno nepraktický. Při zkoumání DOSu jsem však zjistil, že jsou zde k dispozici přímo služby, jak tomu bývá zvykem u jiných systémů. Zde je jejich úplný seznam, doplněný adresami, na kterých ve verzi 5.03 začíná vlastní rutina (není to adresa pro volání služby!):

Č. Adresa. Popis funkce

00 3D98H Restore - hlava na stopu 0.

výstup: BC kód chyby

01 3DCBH Nastavení čísla mechaniky

vstup: A číslo mechaniky

výstup: BC kód chyby

02 3E63H Najede hlavou na určenou stopu

vstup: A číslo stopy

výstup: BC kód chyby

03 3F02H Uloží číslo sektoru

vstup: A číslo sektoru

výstup: na 5CFFH číslo sektoru
BC kód chyby

04 3F06H Uloží adresu bufferu

vstup: HL adresa bufferu

výstup: na 5D00H adresa bufferu
BC kód chyby

05 1E3DH Načtení sektoru

vstup: HL=buffer
D=stopa
E=sektor
B=počet sektorů

výstup: data v bufferu
BC kód chyby

06 1E4DH Zápis sektoru

vstup: HL buffer
D stopa
E sektor
B počet sektorů
data v bufferu

výstup: BC kód chyby

07 28D8H Vypsání katalogu disku

vstup: A kanál

výstup: katalog disku do daného kanálu
BC kód chyby

08 165CH Načtení zvolené hlavičky z adresáře

vstup: A pořadí

výstup: od 5CDDH hlavička
BC kód chyby

09 1664H Zapsání hlavičky do adresáře

vstup: od 5CDDH hlavička
A pořadové číslo hlavičky

výstup: BC kód chyby

10 1CF0H Vrátí číslo hlavičky (pořadí)

vstup: od 5CDDH hlavička

výstup: na 5D0FH číslo hlavičky,
nebo FFH
C číslo hlavičky, nebo 0
(při nálezu nast. Z, jinak NZ)

11 28FBH Uloží basicový blok CODE

vstup: od 5CDDH hlavička
HL začátek bloku
DE delka bloku

výstup: BC kód chyby

12 28F2H Uloží basicový program

vstup: od 5CDDH hlavička
na 5CD1H buď 0, nebo
číslo řádku autostartu

výstup: BC kód chyby

13 01D3H **Neobsazeno**

14 290FH **Načtení programu**

vstup: na 5CDDH hlavička
A=0 - start a délka podle hlavičky
A=3 - HL start
DE delka

výstup: BC kód chyby

15 01D3H **Neobsazeno**

16 01D3H **Neobsazeno**

17 01D3H **Neobsazeno**

18 2926H **Vymazání souboru**

vstup: od 5CDDH hlavička

výstup: BC kód chyby

19 28E0H **Přenesení hlavičky (16 bajtů) z adresy v HL na 5CDDH**

vstup: v HL adresa bufferu
v bufferu hlavička

výstup: na 5CDDH hlavička

20 28E3H **Přenesení hlavičky (16 bajtů) z 5CDDH na adresu v HL**

vstup: na 5CDDH hlavička
v HL adresa bufferu

výstup: v bufferu hlavička

21 2739H **Verifikace stopy (kontrola čitelnosti)**

vstup: D stopa

výstup: na 5CD6H počet chyb
BC kód chyby

22 1FEBH **Volba strany 0**

23 1FF6H **Volba strany 1**

24 0405H **Ověření značky TR-DOSu a nastavení druhu záznamu podle údajů na disku**

výstup: nastavené bity specifikace
BC kód chyby

POZOR! TR-DOS používá číslování sektorů od nuly. Je však zajímavé, že v chybových hlášeních jsou udávána čísla sektorů začínající od jedné. Při oboustranném záznamu ukládá systém stopy tak, že sudé stopy (0, 2, 4, atd.) jsou na straně 0 a liché (1, 3, 5, atd.) jsou na straně 1. V chybových hlášeních jsou však udávána skutečné přečtená čísla stop, takže stopy jsou číslovány na každé straně zvlášť, přičemž se čísla stran neuvádějí. Orientace v těchto hlášeních je pak dost obtížná.

Při volání služeb se připraví parametry na systémové proměnné a zadají se hodnoty do registrů. V registru C musí být vždy číslo požadované služby. Je také vhodné vynulovat obsah adresy 5D0F před každým voláním služby. Sem totiž systém ukládá kód chyby, ale sám ho obvykle nenuluje. Potom se volá adresa 3D13H (u všech verzí 5.xx stejná). Většina služeb vrací v BC kód chyby (pokud to má smysl). Chybové kódy jsem příliš nestudoval - stačilo že 0=žáná chyba. Vyjímka je ve službě 10, která vrací v C číslo hlavičky a 0 zde značí neúspěch.

Služby pro čtení a zápis sektoru jsou napsané tak, že umožňují i čtení a zápis sektorů jiných délek. V takovém případě však nejde využít multisektorový provoz, ale je nutno vždy volat službu jen pro jeden sektor. V dalších dílech tohoto seriálu najdete ukázky programů pro vzájemný převod různých formátů.

Adresář je uložen na stopě 0 a zabírá sektory 0÷7. Každá položka v adresáři je určena hlavičkou, délky 16 bajtů, jejíž struktura vypadá takto:

bajty 1÷8 - název
bajt 9 - typ ("B"=Basic, "C"=CODE, "D"=DATA "#"=sekvenčně uložená data)
bajty 10÷11 - start CODE, nebo délka Basicu
bajty 12÷13 - délka CODE, nebo dél. Bas. bez prom.
bajt 14 - počet sektorů
bajt 15 - na kterém sektoru začíná
bajt 16 - na které stopě začíná

Takových položek se do adresáře vejde max. 128. Soubor je vymazán tak, že se první znak názvu nahradí hodnotou 1. Od této chvíle se název již nevypisuje v katalogu, ale soubor je jinak neporušen. Ke skutečnému přepsání souboru může dojít až po "setřesení" souborů na disku příkazem MOVE. Tím se teprve fyzicky přemístí soubory tak, aby byl jeden těsně za druhým. Odpovídajícím způsobem se přemístí a aktualizují i záznamy v adresáři. Popis smazaných souborů tedy zůstane na konci adresáře. Tyto neplatné položky adresáře mají první znak názvu přepsaný nulou. V té době jsou smazané soubory zčásti přepsány a tedy nenávratně ztraceny. Pokud smažete poslední soubor v adresáři, za nímž už všechny položky začínají nulou, je první znak jeho názvu rovnou nahrazen nulou. Jím zabraný prostor na disku je dán k dispozici systému hned i bez MOVE.

Výše popsané principy je dobré znát. Umožní totiž mnohdy záchranu zdánlivě ztraceného souboru. Také je dobré si uvědomit, že důležité soubory jsou nejvíce ohroženy při zápisu do adresáře, tj. při SAVE a MOVE. Rušivým impulzem, nebo výpadkem napájení při těchto operacích, může dojít k vážným poškozením adresáře. To se pak projeví ztrátou souborů a to i těch, kterých se tato operace zdánlivě netýkala. Stačí třeba na začátek sektoru 0, stopy 0 dát nulu a v katalogu se od té chvíle nic nebude vypisovat. Všechny soubory na disku budou nedostupné. Přitom jde na disk ukládat nové soubory. Přestože budou v pořádku uloženy, budou také nedostupné a nevypíší se v adresáři. Stačí nulu zpětně nahradit kódem znaku a vše je rázem v pořádku.

Možná se vám poslední dva odstavce zdají zbytečně podrobné, ale oceníte je ve chvíli, kdy se vám do adresáře vloudí chybička. Já jsem si to neměl kde přečíst a tak jsem nad Betadiskem strávil mnoho nezapomenutelných chvil.

Sektor 8 nese informace o disku. Sektory 9÷15 jsou k dispozici systému. Uspořádání informací o disketě v sektoru 8 je následující:

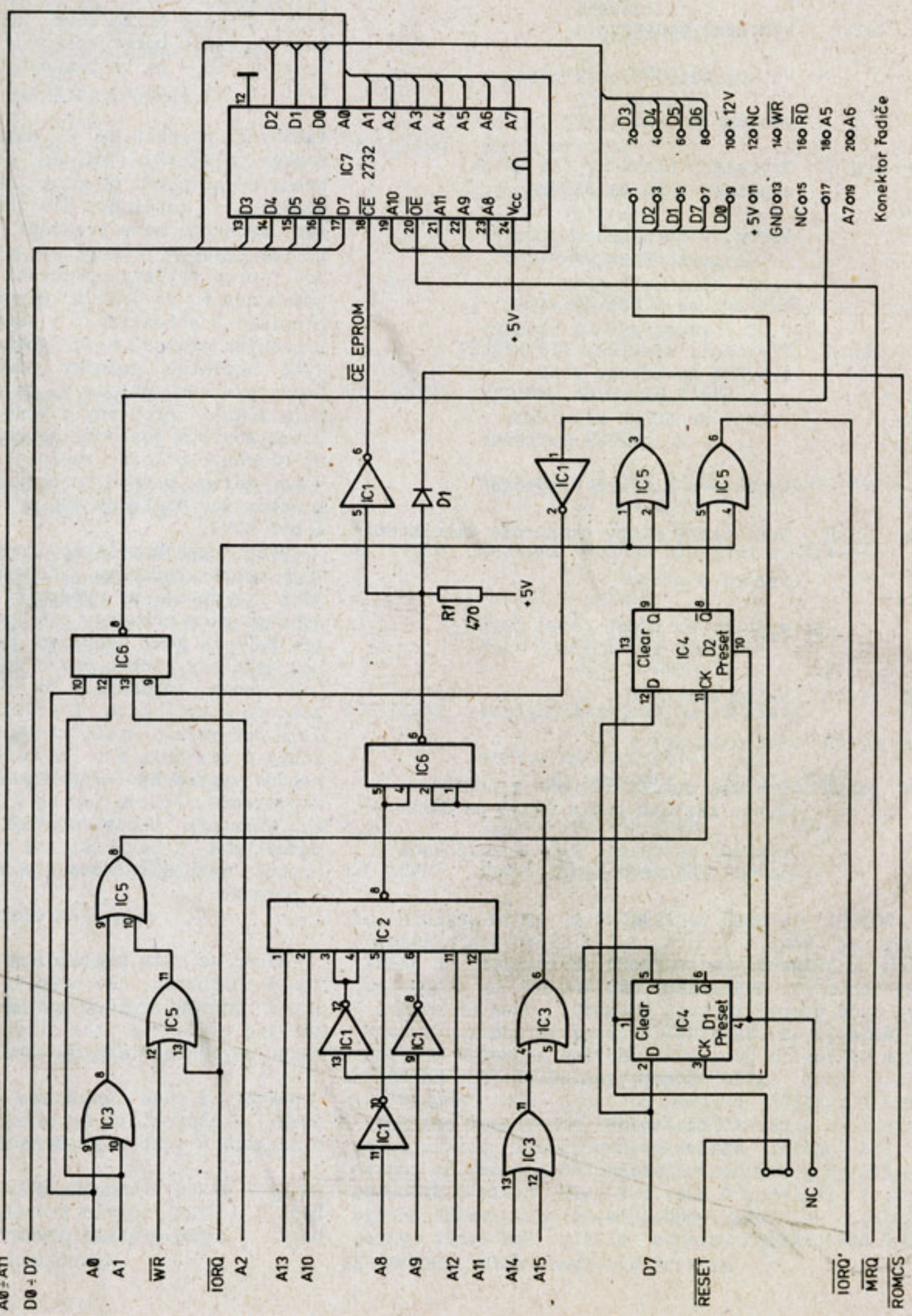
bajty 0÷224 - hodnota 0
bajt 225 - číslo prvního volného sektoru
bajt 226 - číslo první volné stopy

bajt	227	- označení druhu záznamu 22=80tr,DS 23=40tr,DS 24=80tr,SS 25=40tr,SS
bajt	228	- počet souborů na disku, včetně smazaných
bajty	229+230	- počet volných sektorů na disku
bajt	231	- 16 (identifikace TR-DOSu)
bajty	232+233	- 0
bajt	234÷242	- 0 (ve starších verzích heslo)
bajt	243	- 0
bajt	244	- počet smazaných souborů
bajt	245÷253	- 9 bajtů jméno diskety

Pro ty, kterým se zdá deska s operačním systémem TR-DOS 5.xx příliš složitá, nebo kteří nemají EPROM 16KB, ale také pro zajímavost, uvádím ještě schema desky TR-DOSu 3.0. Byla to vůbec první prodejná verze, zatížená ještě mnoha podstatnými chybami. Přesto mi pár let stačila ke štěstí. Používá EPROM jen 4KB. Přesto bylo dost nadlidské se v jeiim DOSu vyznat, protože ta čtyři kila se zrcadlí čtyřikrát nad sebou a program je adresován do různých zrcadel. Asi pro zmatení nepřítele. Část obsahu EPROM se dekóduje do volného adresového prostoru ROM od adresy 15360. Tam jsou umístěny především stránkovací rutiny. Jistou zajímavostí je, že systém vyžadoval v nejméně vhodných situacích heslo (Password) a počítal si kontrolní součet EPROM. Přístup k DOSu ve strojovém kódu byl dost obtížný, protože systém neměl jednotlivé služby. Zapojení je však tak jednoduché, že se může stát inspirací pro konstruktéry.

Pokud někdo máte nějakou verzi Betadisku, poradím vám jak dostat obsah EPROM na disk, pro účely vlastních výzkumů. Stačí dát příkaz SAVE "tr-dos" CODE 0,16384 a je to. Během činnosti DOSu je totiž přistránkována EPROM, takže je možno ji snadno nahrát. Že je to jasné a jednoduché? Také si to myslím. No vidíte, tehdy před čtyřmi lety mi trvalo snad měsíc, než jsem na to přišel. Moje chyba.

(pokračování příště)



Obr. 3 - Zapojení desky operačního systému 3.0

IBM PC
z pohledu programátora

/4/

RNDr. Januš Drózd

4.4 Vrácení hodnoty funkce

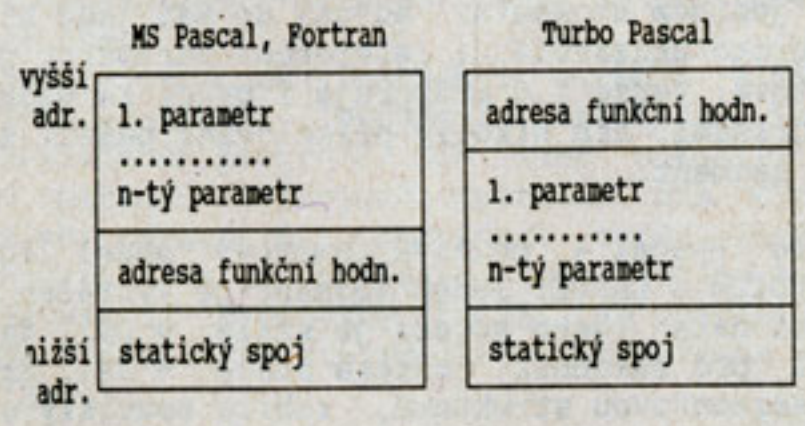
Funkční hodnoty jednoduchých typů se vracejí v závislosti na velikosti:

Vrácená hodnota	registr
jednobytová hodnota	AL
dvoubytová hodnota, blízká adresa	AX
čtyřbytová hodnota, vzdálená adresa	DX:AX

Je-li hodnota funkce strukturovaného typu, nevrací se v registrech, nýbrž v paměti. Adresa hodnoty je pak vrácena v AX (blízká) nebo v DX:AX (vzdálená adresa). Způsob vymezení paměťové oblasti pro funkční hodnotu závisí na překladači.

V překladačích MS C, Turbo C a Basicu si prostor pro svou funkční hodnotu alokuje volaná funkce ve statické oblasti paměti. Při každém volání vrátí (stejnou) adresu této oblasti.

Tento přístup nelze použít v Pascalu, protože paměť pro funkční hodnotu se musí alokovat dynamicky. Funkce totiž může napřed přiřadit svému identifikátoru výslednou hodnotu a potom se rekurzivně zavolat. Proto v MS Pascalu a MS Fortranu prostor pro funkční hodnotu vyhradí volající. Adresu této oblasti předá volané funkci jako poslední, dodatečný parametr. Turbo Pascal funguje podobně, ale adresu oblasti pro výsledek předává před prvním parametrem. Zásobník vypadá při volání takto:



Hodnotu typu real překladač Turbo C nevrací v registrech, ale na vrcholu floating point zásobníku. Tento zásobník je realizován buď v HW matematického koprocesoru nebo v SW emulační knihovně. Turbo Pascal vrací reálnou 6-bytovou hodnotu v DX:BX:AX, hodnoty ostatních reálných typů vrací stejně jako Turbo C. Překladače Microsoft vracejí 4-bytový real v DX:AX, 8-bytový real v paměti jako u datových struktur.

4.5 Standardní struktura segmentů

Firma Microsoft zavedla konvence pro jména segmentů, využití segmentů, spojování segmentů do grup a tříd. Dva moduly musí respektovat tytéž konvence, pokud:

- obsahují části stejného segmentu;
- obsahují segmenty patřící do stejné grupy;
- chtějí využít rozčlenění segmentů do tříd k souvislému uložení paměťových objektů podobného charakteru (kód, inicializovaná data, neinicializovaná data);
- chtějí mít společný zásobníkový segment.

Jména a atributy standardních segmentů jsou v tabulce:

Model	jméno sg.	align	combine	třída	grupa
všechny	DATA	WORD	PUBLIC	'DATA'	DGROUP
	CONST	WORD	PUBLIC	'CONST'	DGROUP
	BSS	WORD	PUBLIC	'BSS'	DGROUP
	STACK	PARA	STACK	'STACK'	DGROUP
Small	TEXT	WORD	PUBLIC	'CODE'	
Medium	jmTEXT	WORD	PUBLIC	'CODE'	
Compact	TEXT	WORD	PUBLIC	'CODE'	
	FARDATA	PARA		'FARDATA'	
	FARBSS	PARA		'FARBSS'	
Large & Huge	jmTEXT	WORD	PUBLIC	'CODE'	
	FARDATA	PARA		'FARDATA'	
	FARBSS	PARA		'FARBSS'	

Segmenty ve třídě 'CODE' obsahují kód. Segmenty ve třídách DATA a FARDATA obsahují data, jejichž iniciální hodnoty budou v přeloženém programu a budou zaváděny do paměti při spouštění. Segmenty ve třídách BSS a FARBSS obsahují definice dat, která nemají po zavedení programu do paměti definovanou hodnotu. Segment CONST slouží k uložení programových konstant (literálů). Segment STACK rezervuje místo pro HW zásobník.

Nejprve vysvětlíme význam atributu "combine" (prostřední sloupec tabulky). Pokud má hodnotu PUBLIC nebo STACK, pak linker spojí takto označený segment se segmenty stejného jména do jediného segmentu. Pokud atribut "combine" chybí, segment je považován za soukromý a není spojován.

Ve sloupci "jméno sg." je uvedeno jméno segmentu. Pokud má tvar "jmTEXT", znamená to, že jméno segmentu je zřetězením jména zdrojového souboru

a řetězce "TEXT". Díky tomu v každém zdrojovém modulu vznikne z takto označených segmentů zvláštní segment (přestože combine = PUBLIC).

V modelech Compact, Large a Huge potřebujeme více datových segmentů. Microsoft pro jejich odlišení používá jinou techniku než u kódových segmentů. Všem je přiděleno stejné jméno, ale nemají atribut PUBLIC. Výsledný efekt je stejný.

Údaj v sloupci "align" říká, že segment má být zaveden do paměti na sudou adresu (WORD) nebo na adresu dělitelnou 16 (PARA).

Jediná grupa, která se standardně vytváří, je DGROUP. Omezuje celkový rozsah segmentů DATA, CONST, BSS a STACK na 64 KB. Velké datové objekty je proto nutno definovat v segmentech FARDATA a FARBSS.

Struktura segmentů v produktech firmy Borland je podobná výše popsané, má však svoje zvláštnosti, které zde nebudeme popisovat. Upozorníme pouze na dvě skutečnosti. Především Turbo Pascal nevytváří linkovatelný object kód, který by vyhovoval konvencím MS DOSu a dal se zpracovávat programem LINK, nýbrž rovnou spustitelný program EXE. Proto spojování s externími moduly je jednostranné - do Pascalu lze začlenit object moduly se zcela specifickou strukturou popsanou v manuálech.

V Turbo C existuje paměťový model TINY, který dovoluje vyrábět programy spustitelné DOSem 1. V modelu TINY jsou všechny segmenty spojené do jediné grupy DGROUP. Pak při běhu programu všechny segmentové registry obsahují segm. adresu DGROUP. Z programu přeloženého v modelu TINY nejprve linker vyrobí EXE-program DOSu 2 a z něj pak program EXE2BIN vytvoří COM-program vyhovující požadavkům DOSu 1.

4.6 Pascal na IBM PC

Jak MS Pascal tak i Turbo Pascal používají jazyk blízký normě ISO 7185 obohacený o běžná rozšíření, jako:

- inicializace hodnot statických proměnných;
- strukturované konstanty;
- větev OTHERWISE v příkazu CASE;
- příkaz pro opuštění právě prováděného podprogramu;
- typ pro řetězce proměnné délky a procedury pro manipulaci s řetězci;
- změna typu výrazu;
- deklarace proměnné na explicitně uvedené adrese.

Překladače obsahují prostředky pro modulární programování odvozené z Moduly II. V Turbo Pascalu lze vytvářet samostatně kompilovatelné jednotky, tzv. units. Unit obsahuje deklarace konstant, typů, proměnných a podprogramů, z nichž část zveřejňuje a část skrývá. Unit má ve verzi 4 tuto obecnou strukturu:

```
unit <identifikátor jednotky>
interface                    ( definiční část )
uses <seznam identifikátorů importov. jednotek>
<deklarace veřejných objektů>
implementation               ( implementační část )
<deklarace soukromých objektů>
begin                        ( inicializační část )
<posloupnost inicializačních příkazů>
end.
```

V definiční části se uvádějí hlavičky všech veřejných podprogramů. V implementační části pak následují jejich identifikace a bloky.

Přístup programu k samostatně kompilovaným jednotkám zajistíme tak, že bezprostředně za hlavičku programu umístíme klauzuli:

uses <seznam identifikátorů jednotek>

Před provedením těla programu se provedou inicializační části uvedených jednotek. Všechny objekty, definované v definičních částech těchto jednotek, budou přístupné i v programu tak, jako by byly definovány v bloku obsahujícím program. Jejich identifikátory lze v programu předefinovat, pak ale pro přístup k objektům z jednotek je nutno použít kvalifikaci ve tvaru:

identifikátor_jednotky.identifikátor_objektu,

která explicitně stanovuje, že jde o odkaz na objekt definovaný v uvedené jednotce.

Povšimněte si, že nelze samostatně kompilovat definiční a implementační část. V důsledku toho byly ve verzi 4 zakázány cyklické odkazy mezi jednotkami (Vzdor tomu, že by se daly snadno implementovat. Stačilo by při překladu závislých jednotek číst definiční část jednotky z jejího zdrojového textu.)

Ve verzi 5 Turbo Pascalu se problém cyklických závislostí mezi moduly vyřešil jinak, způsobem inspirovaným Modulou II. Syntaxe jednotky je rozšířena tak, že klauzuli **uses** lze použít i v implementační části. Mezijednotkové odkazy umístěné v implementačních částech mohou tvořit cyklus.

MS Pascal implementuje podobný koncept jednotek. Liší se však v tom, že definiční část jednotky se musí (ve zdrojové formě) opsat do všech jednotek, které ji používají. Pak nic nebrání cyklickým závislostem mezi jednotkami.

Kromě jednotek lze v MS Pascalu vytvářet také tzv. **moduly**. Modul, na rozdíl od programu, nema tělo. Všechny podprogramy definované na nejvyšší statické úrovni v modulu jsou veřejné. V hlavním programu uvedeme hlavičky podprogramů definovaných v modulech následované direktivou EXTERNAL. Modul je slabší koncept než unit, stojí za ním patrné snaha o kompatibilitu se staršími implementacemi.

MS Pascal používá paměťový model MEDIUM. Každa kompilační jednotka vytváří zvláštní kódový segment. Se vzdálenou adresou se volají právě ty podprogramy, jejichž jména jsou veřejná. Všechna data jsou v jediném segmentu, adresy do dat jsou blízké, a proto ukazatelé i referenční parametry jsou dvoubytové. Verze 1.0 podporuje i práci se vzdálenými adresami, ale nikoliv přidělování paměti mimo datový segment.

Turbo Pascal 4 používá paměťový model ležící mezi MEDIUM a LARGE. Jeden segment je vyhražen pro statická data (jeho adresa je stále v DS), další segment pro zásobník (adresa stále v SS). Halda nemá segmentovou strukturu, zabírá souvislý úsek fyzické paměti. Jak ukazatelé, tak i všechny parametry předávané referencí, používají vzdálené adresy. Ve srovnání s MS Pascalem je tedy k dispozici více paměti, ale přístup k referenčním parametrům a k proměnným alokovaným na haldě je pomalejší.

4.7 Jazyk C na IBM PC

Popularita jazyka C v poslední době prudce roste. Osvědčil se jako jazyk vhodný pro systémové programování, poskytující vyšší produktivitu práce než asembler.

C však není nezáludný bezpečný jazyk typu Pascalu. K jeho používání je nutná solidní znalost. Svědčí o tom např. níže uvedený program:

```
... ... <stdio.h>
int i;
unsigned int ui;
long l;
unsigned long ul;
main()                  /* kolik je 3-7+1 ? */
{ ui=3;  i=3;  ul=3;  l=3;
  printf ("%ld,  %ld\n",    ui-7 +11,   i-7 +11 );
  printf ("%f,   %f \n",    ul-7l+1.0,  l-7l+1.0);
}

65533,   -3
4294967293.000000,   -3.000000
```

Na kvalitu implementací jazyka C měla příznivý vliv ANSI norma. Snad nejdůležitějším krokem ve vývoji jazyka bylo zavedení tzv. **prototypů funkcí**. Prototyp funkce má tvar:

typvysl idfce (typ idpar, typ idpar, ...);

Od místa uvedení prototypu v programu bude při každém volání funkce **idfce**:

- kontrolován počet a typy parametrů a typ výsledku;
- provedeny potřebné konverze hodnot skutečných parametrů na typ formálních parametrů;
- provedena konverze typu výsledku funkce dle kontextu v místě volání.

Prototyp funkce slouží k popsání funkce buď definované dále v programu (kterou chceme volat před deklarací) nebo externí funkce. Nahrazuje pascalskou hlavičku funkce následovanou direktivou (např. FORWARD, EXTERNAL).

V jazyku C je při psaní modulů zvykem vytvořit soubor obsahující prototypy všech funkcí, které modul zveřejňuje, a popisy veřejných proměnných (opatřené atributem EXTERN). Tento soubor, zvaný **header file**, je pak vtažen (include) do všech zdrojových programů, které modul používají. Zjednodušuje se tím popis interface modulů a zvyšuje bezpečnost.

Implementace Turbo C a MS C jsou blízké ANSI normě. Na PC existují i objektově orientovaná rozšíření C, např. C++ nebo Objective C. Rozšířený jazyk se buď překládá preprocesorem do standardního C (Advantage C++) nebo rovnou do strojového kódu (Zortech C++).

Velmi výhodnou vlastností Turbo C a MS C je jejich schopnost provádět sémantické kontroly v překládaném programu. Jde např. o kontrolu, zda typy operandů odpovídají operacím, zda typy skutečných parametrů odpovídají typům formálních parametrů, zda se přiřazují kompatibilní hodnoty. Používáme-li prototypy funkcí, je rozsah těchto kontrol přibližně stejný jako v UNIXovském programu LINT.

Typickým rysem jazyka C je poměrně jednoduché jádro a rozsáhlý soubor knihoven podprogramů. Část těchto knihoven je standartizována (ANSI), část slouží k využití specifických vlastností hostitelského OS a hardware. Implementačně závislé knihovny jsou rozsáhlé a není snadné se v nich orientovat. Např. programátor v Turbo C má pro práci se soubory k dispozici:

- knihovnu pro obsluhu "streamů" dle [5] (fopen, fread);
- knihovnu pro obsluhu souborů dle UNIXovských zvyklostí (open, read);
- knihovnu pro práci se soubory ve stylu MS DOSu (open, read).

Není však k dispozici zcela transparentní přístup k souborům na úrovni operačního systému, což u jazyka pro systémové programování překvapuje.

V MS C lze program překládat v kterémkoliv z 5 základních paměťových modelů. V Turbo C je k dispozici navíc šestý model - TINY (viz 4.5). Každý model používá svoje vlastní knihovny! V Turbo C existují v modelech s jedním datovým segmentem dvě haldy - blízká a vzdálená. Pro přístup k první z nich se používají 2-bytové ukazatele, k druhé 4-bytové.

4.8 Ladící prostředky na PC

Programátoři na PC mají v poslední době k dispozici kvalitní prostředky k ladění na úrovni zdrojových textů. Překladače PJ (pro MASM to platí od verze 5) jsou schopny vložit do generovaného kódu údaje o:

- jménu a umístění zdrojového souboru;
- pořadových číslech zdrojových řádek, z nichž vznikly jednotlivé úseky kódu;
- rozsazích platnosti, identifikátorech, typech a adresách všech zdrojových proměnných.

Program CodeView, standardní ladící prostředek firmy Microsoft, ladí spustitelný EXE program obohacený o uvedené informace. K tomuto programu si vyhledá příslušné zdrojové soubory a při ladění zobrazuje jejich úseky. Umožňuje:

- prohlížet a modfikovat obsah paměti a strojových registrů;
- prohlížet a modifikovat hodnoty programových proměnných;
- vyčíslovat hodnoty výrazů utvořených dle syntaxe některého zdrojového jazyka (Fortran, C, Pascal);
- krokovat program buď po zdrojových řádcích nebo po strojových instrukcích;
- umístit do zdrojového nebo přeloženého programu body zastavení (breakpoints) a spustit běh programu. Běh se zastaví na konci programu nebo po dosažení některého bodu zastavení;
- provádět program ve zpomaleném režimu (animace) s možností jeho zastavení z klávesnice;
- definovat výrazy (které mohou obsahovat programové proměnné), jejichž hodnoty se průběžně zobrazují při výpočtu;
- definovat podmínky zastavení běhu programu jako:
  - změnu hodnoty nebo
  - dosažení zadané hodnoty

  v programové proměnné nebo úseku paměti.

Logika ladění vyžaduje, aby byly k dispozici dvě obrazovky: obrazovka laděného a ladícího programu. Pokud uživatel nemá dva videoadaptéry a dva monitory, lze dvojici virtuálních obrazovek realizovat dvěma způsoby. Buď se využije fakt, že adaptéry mají 4 textové videostránky, nebo se pro obsah obrazovky vyhradí buffer v operační paměti. První způsob je rychlejší, ale lze jej použít jen tehdy, pokud laděný program nepoužívá více videostránek ani grafický režim adapteru.

Ve verzích 2.0 Turbo C a 5.0 Turbo Pascalu je zjednodušený ladící systém integrován spolu s překladačem a editorem do jediného programu. Mimo to byl uveden na trh samostatný Turbo Debugger. Pro programy vytvořené Turbo Pascalem 4.0 existuje také speciální externí debugger.

Problémem kvalitních debuggerů je, že jsou dost velké, a proto se nemusí vejít do paměti současně s laděným programem. Na to mysleli návrháři Turbo Debuggeru, když umožnili ladění na dvojici počítačů spojených linkou. Na prvním počítači běží laděný program a malý komunikační modul, na druhém pak ladící systém.

5. SPOLUPRÁCE MODULŮ V RŮZNÝCH PJ

Při vytváření rozsáhlých programových celků se lze dostat do situace, kdy v jednom programu se střetávají věci velmi rozdílné povahy. Může jít např. o:

- přímé ovládání hardware počítače s extrémními nároky na rychlost;
- řízení databáze;
- složité numerické výpočty vyžadující optimalizaci kódu;
- část programu s velmi složitou vnitřní logikou vyžadující modulární řešení.

Protože staří veleještěři jako PL/1 už vyhynuli a noví veleještěři jako ADA ještě nepřesvědčili o své životaschopnosti, programátor může:

- buď si vybrat svůj zamilovaný jazyk a na vlastní kůži se přesvědčit, že není zcela univerzální;
- nebo programovat každou část v jazyce, který je pro ni přirozený, a pak se pokusit vše spojit do jednoho celku.

Zatímco mnohé programovací jazyky poskytují solidní vnitřní modulární prostředí (viz Pascal), pro spojování modulů z různých jazyků musíme používat služby systémového linkeru. To je však možné pouze za předpokladu, že překladač PJ dokáže vyrobit standardní object modul. Tuto schopnost nemá např. Turbo Pascal, proto se připojování dalších modulů k němu musí podřídit specifickým požadavkům.

Linker obdrží na vstupu jeden nebo více object modulů (v souborech s příponou OBJ) a případně i knihovny (v souborech s příponou LIB). Z knihoven vybere jen ty moduly, do nichž vedou externí odkazy z object modulů nebo dříve vybraných knihovních modulů. Všechny vybrané moduly linker spojí, vyřeší odkazy mezi nimi a vytvoří z nich jeden EXE soubor.

V první fázi linker zpracuje stejně pojmenované segmenty v závislosti na atributu "combine". Segmenty označené:

- PUBLIC, STACK nebo MEMORY zřetězí za sebou a přepočte efektivní adresy jejich symbolů na společný začátek;
- COMMON překryje přes sebe, přičemž délka největšího z nich bude délkou výsledku;
- neoznačené segmenty považuje za soukromé a nespojuje.

Dále linker spojí do jednoho segmentu všechny segmenty patřící do stejné grupy. Spojené segmenty seřadí tak, aby segmenty patřící do stejné třídy byly souvisle za sebou. Výsledek, spolu s tabulkou segmentových adresových konstant a případnými ladícími informacemi, zapíše do souboru s příponou EXE.

Úspěšné spojení modulů vyžaduje, aby:

- objekty, které mají být považovány za stejné, měly stejná jména i z pohledu linkeru (viz 5.1);
- aby všechny spojované moduly dodržovaly stejné konvence pro strukturu a pojmenování segmentů (např. dle 4.5).

Pro úspěšnou funkci spojeného programu je třeba, aby se moduly shodly:

- ve volacích konvencích podprogramů (viz 4.3 a 5.3);
- ve vnitřní reprezentaci odpovídajících typů dat (viz 4.1);
- ve způsobu vrácení hodnoty funkce (viz 4.4);
- v paměťovém modelu nebo alespoň ve volbě blízkých, nebo vzdálených adres pro přístup ke sdíleným podprogramům.

Volaný podprogram musí zachovávat hodnoty registrů BP, DS, SI, DI (a pochopitelně obnovit hodnoty CS, IP, SS, SP). Registr BP ukazuje na platný aktivační záznam, v registru DS je segmentová adresa statických dat, registry SI a DI se v jazyku C využívají pro uchovávání registrových proměnných.

V zájmu zjednodušení dalšího výkladu nyní zavedeme umělý pojem "paskalizovaný objekt". Překladače jazyka C disponují přepínačem **"Pascal calling convention"** a umožňují přidat k deklaraci proměnné nebo funkce atribut **Pascal** nebo **Cdecl**. Je-li tento přepínač při překladu vypnutý, budou paskalizované ty objekty kompilační jednotky, které jsou označené atributem Pascal. Je-li přepínač zapnutý, jsou paskalizovány všechny objekty, které nejsou označeny atributem Cdecl.

5.1 Slučitelnost jmen sdílených objektů

Tvoříme-li program z více modulů, potřebujeme, aby objekt definovaný v jednom modulu byl přístupný v jiných modulech. V takovém případě jeden modul prohlásí svůj objekt za veřejný a ostatní moduly jej prohlásí za externí. Jména veřejných a externích objektů se zapíšou do object souboru a linker vyřeší odkazy mezi moduly.

Toto jednoduché schéma podstatně komplikují akce, které překladač provede se jménem dříve, než je zapíše do object souboru.

První úpravou je zkrácení jména na jistý maximální počet znaků. Tento počet může být menší, než je počet znaků vnitřně rozlišovaných v překladači.

Zkrácení jména	Microsoft	Borland (Turbo)
Basic	40	?
C	8	volitelně 1 - 32
Fortran	6	-
Pascal	8	netvoří object
Assembler	31	všechny

Druhou možnou úpravou je konverze malých písmen na velká. Provádějí ji překladače Pascalu, Fortranu a Basicu. Jazyk C, v němž se standardně rozlišuje mezi malými a velkými písmeny, ji dělá pouze u paskalizovaných objektů. Assembler převádí malá písmena na velká v závislosti na nastavení přepínače /MX. Komplikacím s malými a velkými písmeny se vyhneme, pokud linker bude ignorovat rozdíly mezi nimi. Je proto vhodné pamatovat na vhodné nastavení příslušného přepínače linkeru.

Překladače jazyka C předřazují jménům nepaskalizovaných objektů zapisovaným do object souboru podtržítko '_'. Proto objekt XYZ se zvenčí jeví jako _XYZ. Přidávání podtržítek lze sice vypnout přepínačem "generate underbars", ale takové počínání znemožní linkovat C s jeho vlastními knihovnami, proto jej nelze doporučit.

5.2 Definice externích a veřejných objektů

V jazyce C se externí a veřejné objekty popisují v souladu s normou.

V Pascalu se pro popis externích podprogramů z jiného jazykového prostředí a z modulů (MS Pascal) používá obvyklý způsob - direktiva EXTERN. Zpřístupnění externích objektů definovaých v jednotkách je popsáno v části 4.7.

Zvláštnosti assembleru byly rozebrány v kap. 3. Připomeňme, že špatné umístění pseudoinstrukcí EXTRN vede k obtížně odstranitelným chybám.

5.3 Přizpůsobení volacích konvencí

Spolupráce jazyka C na jedné straně a Pascalu, Fortranu nebo Basicu na straně druhé, by nebyla prakticky možná bez způsobu, jak změnit volací konvence.

V jazyce C se použije Pascalská volací konvence pro paskalizované funkce.

V MS Pascalu se použije volací konvence C pro podprogramy označené [C] bezprostředně před středníkem ukončujícím hlavičku..

5.4 Jak prakticky postupovat

Potřebujeme-li zajistit spolupráci modulů naprogramovaných v různých jazycích, doporučujeme se držet níže uvedených pravidel. Nejsou jediná možná, zato jsou jednoduchá.

1. Pro sdílené objekty nepoužíváme jména delší, než 8 znaků (je-li ve hře i Fortran, pak 6 znaků).
2. Zajistíme, aby linker nerozlišoval mezi velkými a malými písmeny.
3. Veřejné a externí objekty v C paskalizujeme.
4. Ve Fortranu označíme sdílené podprogramy atributem "Pascal".
5. Je-li některá část naprogramována v Pascalu, pak buď použijeme MS Pascal, nebo všechny ostatní moduly vygenerujeme dle specifických požadavků Turbo Pascalu.
6. Nekombinujeme překladače Microsoft a Borland (výjimku tvoří assembler).

6. ZÁVĚR

Poslední rada: Dříve, než začnete tvořit svůj původní a dokonalý program pro PC, zjistěte ve svém okolí, zda již není dávno hotový. Existují příklady programů, které byly s velkým úsilím nezávisle vyvinuty na desítkách pracovišť.

Závěrem bych chtěl poděkovat RNDr. Pavlovi Novákovi, A.C. a Petrovi Horskému za pečlivé pročtení článku a řadu podnětných připomínek.

Literatura

(Hvězdička je značkou mimořádné kvality)

[1]* Norton, P.: Programmer's Guide to the IBM PC. Microsoft Press, Washington 1985.
[2] Macro Assembler 5.0 Programmer's Guide. Microsoft 1987
[3] Mixed-Language Programming Guide. Microsoft 1987.
[4] The 8086 Family User's Manual. Intel 1979.
[5] Kernighan, B.W., Ritchie, D.M.: The C Programming Language, Prentice-Hall, Englewood Cliffs, N.J. 1978. Slovenský překlad "Programovací jazyk C", ALFA, Bratislava 1988
[6]* Holzner, S.: Advanced Assembly Language on the IBM PC, Prentice-Hall, New York 1987.
[7] Novák, P., Štěpán, P.: Počítač IBM PC s operačním systémem MS-DOS (PC-DOS)

Informace ze zahraničí

Co nového ve světě tiskáren?

PC WORLD FOCUS je přílohou britského časopisu PCW (Personal Computer World). Číslo 5/89 je věnováno tentokrát tiskárnám. Obsahem přílohy jsou nejen údaje o více než 200 typech tiskáren, ale i zajímavé články věnované problematice přenosu informací na papír. Nechybí ani soutěž s hlavní cenou - laserovou tiskárnou EPSON GQ3500. Stojíte-li tedy před problémem výběru vhodného typu tiskárny pro váš počítač, řešení najdete v časopisu PCW 5/89, který je dostupný na mikrofiších ve středisku VTI Svazarmu.

Magneticko-optický disk 500 Megabajtů

dodává britská firma 3M. Přenosová rychlost je více než 25 Megabitů za sekundu. Výhodou tohoto způsobu záznamu dat je minimální opotřebení záznamového média.

Lokální síť a PASCAL

Zajímá-li vás, jak lze využít NetBios pro komunikaci dvou "paskalovských" programů napříč sítí, podrobné informace najdete v britském časopise Personal Computer World (PCW) 5/89 - str. 216

PowerCad

je název programového vybavení dodávaného s digitizérem (tabletem) britské firmy Hegotron za celkem 300 GBP. Tablet je dodáván s kartou pro IBM PC XT/AT. Programové vybavení umožňuje export/import dat s programem AutoCad. Při zobrazování souboru (obligátní raketoplán) je však o poznání rychlejší než AutoCad. Program podporuje všechny tradiční grafické adaptéry. K dispozici je sada knihoven symbolů pro nejrůznější oblasti použití. Informace včetně demoverze jsou k dispozici ve středisku VTI Svazarmu.

Znáte počítače AMSTRAD?

Díky spolupráci ZOZO Media a 602. ZO Svazarmu, která vyústila ve zprosředkování prodeje výrobků firmy AMSTRAD na našem trhu, budeme tedy moci tyto osobní počítače, známé svoji kvalitou a příznivou cenou, zakoupit také v Československu. Zatím se jedná o zprostředkování prodeje za devizy, zaručen je však záruční i pozáruční servis, technické poradenství a pravidelné seznamování našich zákazníků s novými výrobky firmy, formou předváděcích dnů, seminářů a dalších akcí. Zveřejňujeme tedy základní přehled osobních počítačů Amstrad se základními technickými parametry. Počítače Sinclair zůstávají nadále v sortimentu firmy Amstrad. Uvedená cena platí pro konsignační sklad v Praze.

ZX Spectrum 128K+2

Paměť: 128 kilobajtů RAM, 32 kilobajtů ROM.
Mikroprocesor: Z80 / 3.5469 MHz.
Obrazovka: 256 x 192 pix, 8/8 barev.
Zvuk: 3 úrovně / 16 obálek.
Klávesnice: Qwerty / 58 tlačítek.
Záznam dat: vestavěný datakorder.
Firmware: 128K rozšířený Spectrum Basic.
V/V rozhraní: UHF PAL, RGB, RS232C, Centronics, MIDI, zvukový výstup, 2 ovladače, sběrnice Z-80.
Počítač je dodáván včetně zdroje 9V.

Cena: 113 GBP

ZX Spectrum 128K+3

Paměť: 128 kilobajtů RAM, 64 kilobajtů ROM.
Mikroprocesor: Z80 / 3.5469 MHz.
Obrazovka: 256 x 192 pix, 8/8 barev.
Zvuk: 3 úrovně / 16 obálek.
Klávesnice: Qwerty / 58 tlačítek.
Záznam dat: 3" disk. jednotka (40 stop, 9 sektorů, 512 bajtů na 1 sektor). Struktura kompatibilní s CP/M.
Firmware: 48K Spectrum Basic.
V/V rozhraní: UHF PAL, RGB, RS232C, Centronics, MIDI, zvukový výstup, 2 ovladače, druhá disketová jednotka, sběrnice Z-80.
Součástí počítače je následující programové vybavení: 128K+3 Basic, +3 DOS, CP/M 2.2 a 3.0.
Počítač je dodáván včetně zdroje 9V / 12V.

Cena: 167 GBP

SINCLAIR PC200

Paměť: 512 kilobajtů RAM, ROM BIOS kompatibilní s IBM.
Mikroprocesor: 8086 / 8 MHz, konektor pro 8087.
Obrazovka: CGA/MDA 640 x 200 pix, 2 barvy, lze připojit na TV přijímač bez úprav.
Zvuk: vestavěný reproduktor s regulací hlasitosti.
Klávesnice: typ AT / 102 tlačítek.
Záznam dat: vestavěná jednotka 3 1/2" / 720 kilobajtů.
Software: MS-DOS 3.3.
V/V rozhraní: 2. disk. jednotka, TV, 2 rozšiřující konektrory, seriový V/V, Centronics, 2 ovladače, zvukový výstup.
Počítač je dodáván včetně myši a software.

Cena: 327 GBP

CPC 464

Paměť: 64 kilobajtů RAM, 32 kilobajtů ROM.
Mikroprocesor: Z80 / 4 MHz.
Obrazovka: 640 x 200 pix, 16 barev (6845 CRT).
Zvuk: 3 úrovně / 8 oktáv (AY-3-8912).
Klávesnice: Qwerty / 74 tlačítek.
Záznam dat: integrovaný datakorder 1000/2000 baudů rychlost záznamu lze nastavit programově.
Firmware: Locomotive Basic, CP/M 2.2 a 3.0.
V/V rozhraní: 2. disk. jednotka, RGB, Z80-sběrnice, DDI-1, Centronics, 2 ovladače, zvukový výstup.
Počítač je dodáván včetně monitoru.

Cena: 107 GBP

CPC 6128

Paměť: 128 kilobajtů RAM, 48 kilobajtů ROM.
Mikroprocesor: Z80 / 4 MHz.
Obrazovka: 640 x 200 pix, 16 barev (6845 CRT).
Zvuk: 3 úrovně / 8 oktáv (AY-3-8912).
Klávesnice: Qwerty / 74 tlačítek.
Záznam dat: 3" disk. jednotka (765 floppy disc controller).
Firmware: Locomotive Basic CP/M 2.2, 3.0, Logo.
V/V rozhraní: kazetový mgf, Centronics, RGB, 2 ovladače, zvukový výstup, 2. disk.jednotka, Z80-sběrnice.
Počítač je dodáván včetně monitoru.

Cena: 210 GBP

K počítačům CPC je dodáván na přání TV tuner pro příjem televizního signálu a modul digitálních hodin s radiopřijímačem.

PCW 8256

Paměť: 256 kilobajtů RAM (512 u typu 8512).
Mikroprocesor: Z80.
Obrazovka: 90 sloup. / 32 řádek.
Klávesnice: Qwerty / 82 tlačítek.
Záznam dat: 3" disk. jednotka 2x160 Kb (8512: plus 1x1Mb).
V/V rozhraní: RGB, 2. disk. jednotka s kapacitou 1MB.
Software: Mallard Basic, CP/M 2.2, 3.0, Logo, GSX.
Počítač je dodáván včetně monitoru a jehličkové tiskárny.

Cena: 356 GBP (8512: 462 GBP)

PCW 9512

Paměť: 512 kilobajtů RAM.
Mikroprocesor: Z80.
Záznam dat: 3" disk. jednotka 1MB.
V/V rozhraní: RGB, 2. disk. jednotka

Znáte počítače AMSTRAD?

Obrazovka: 90 sloup. / 32 řádek.
Klávesnice: Qwerty / 82 tlačítek.
Software: Mallard Basic, CP/M 2.2, 3.0, Logo, GSX.
Počítač je dodáván včetně monitoru a tiskárny s typovým kolečkem (daisywheel).

Cena: 498 GBP

PC 1512

Paměť: 512 kilobajtů RAM.
Obrazovka: CGA 640 x 200 pix.
Mikroprocesor: 8086 / 8MHz.
Koprocesor : 8087.
Záznam dat: 5 1/4", pružný disk 1 kus (SD), 2 kusy (DD).
V/V rozhraní: sér./paralelní, světelné pero, myš, tablet, pevný disk (HD).
Klávesnice: Qwerty / 85 tlačítek.
Software: Locomotive Basic 2, GEM Desktop, GEM Paint.
Počítač je dodáván s možností volby monitoru Mono (MD) a Color (CD).

Cena: SD: 320 GBP, DD: 407 GBP, MD: 90 GBP, CD: 177 GBP

PC 1640

Paměť: 640 kilobajtů RAM.
Mikroprocesor: 8086 /8MHz (V30).
Koprocesor : 8087.
Záznam dat: 5 1/4", pružný disk 1 kus (SD), 2 kusy (DD).
V/V rozhraní: sér /paralelní, světelné pero, myš, tablet, pevný disk (HD).
Klávesnice: Qwerty / 85 tlačítek.
Obrazovka: CGA, EGA, MDA, Hercules.
Software: Locomotive Basic 2, GEM Desktop, GEM Paint.
Počítač lze dodat včetně monitoru Mono (MD), Color (CD) a Enhanced Color (ECD).

Cena: SD: 410 GBP, DD: 490 GBP, HD: 690 GBP, MD: 110 GBP, CD: 207 GBP, EGA: 309 GBP

PPC 512/640

Paměť: 512/640 kilobajtů RAM.
Mikroprocesor: 8086 /8MHz (V30).
Koprocesor : 8087.
Záznam dat: 3 1/5" pružný disk 720 KB.
V/V rozhraní: modul rozšíření (pevný disk, V/V par./ser.).
Obrazovka: CGA, MDA, LCD. CGA a MDA lze připojit externě.
Klávesnice: Qwerty / 102 tlačítek.
Software: Mirror II, PPC Organizer.
Počítač je dodáván včetně displeje LCD a modemu (V21-V23 modem pouze typ PPC 640).

Cena: 512 - SD: 440 GBP, DD: 516 GBP,
640 - SD: 511 GBP, DD: 588 GBP

PC 2086

Paměť: 640 kilobajtů RAM.
Mikroprocesor: 8086 /8MHz.
Koprocesor : 8087.
Záznam dat: disková jednotka 3 1/5" s kapacitou 720 Kb, pevný disk 30Mb s překladem 1:1.
V/V rozhraní: druhá disk. jednotka 3 1/2", 5 1/4",pevný disk, streamer, V/V par./ser.
Rozšiřující konektory: 3 x 8 bitů plus pevný disk.
Obrazovka: CGA, EGA, Herkules, VGA (256 barev).
Klávesnice: Qwerty / 102 tlačítek.
Software: MS-DOS 3.3, Windows 2.03, GW Basic.
Počítač lze připojit na síť NOVELL nebo AMSTRAD NETWORK jako pracovní stanici (workstation).

Cena: SD: 514 GBP, DD: 644 GBP, HD: 891GBP

PC 2286

Paměť: 1 megabajt RAM.
Mikroprocesor: 80286 / 12MHz.
Koprocesor : 80287.
Záznam dat: disková jednotka 3 1/5" s kapacitou 1.4 Mb, pevný disk 40Mb s překladem 1:1.
V/V rozhraní: druhá disk. jednotka 3 1/2", 5 1/4", streamer, V/V par./ser.
Rozšiřující konektory: 5 x 16 bitů.
Obrazovka: CGA, EGA, Herkules, VGA (256 barev).
Klávesnice: Qwerty / 102 tlačítek.
Software: MS-DOS 4.0, Windows 286, GW Basic.
Počítač lze připojit na síť NOVELL nebo AMSTRAD NETWORK jako řídící stanici (server).

PC 2386

Paměť: 4 megabajty RAM, stránkování 64 Kb/35 ns.
Mikroprocesor: 80386 / 20MHz.
Koprocesor : 80387.
Záznam dat: disková jednotka 3 1/5" s kapacitou 1.4 Mb, pevný disk 65Mb s překladem 1:1.
V/V rozhraní: druhá disk. jednotka 3 1/2", 5 1/4", streamer, V/V par./ser.
Rozšiřující konektory: 5 x 16 bitů.
Obrazovka: CGA, EGA, Herkules, VGA (256 barev).
Klávesnice: Qwerty / 102 tlačítek.
Software: MS-DOS 4.0, Windows 386, GW Basic.
Počítač lze připojit na síť NOVELL nebo AMSTRAD NETWORK jako řídící stanici (server).

Cena: 2548 GBP (HD)

AMSTRAD NETWORK

Verze: 1.2.
Operační systém: AMSNET.
Podpora OS: MS-DOS / PC-DOS 3.1 - 3.3.
Podpora HW: IBM PC XT/AT/OS/2.
Hardware: Omninet.
Maximum stanic: 6.
Nároky na řídící stanici: 512Kb min.
Nároky na pracovní stanici: 256Kb min.
Cena kompletní sítě pro 3 PC (HW + SW) je přibližně 400 GBP

MONITOR informací střediska VTI SVAZARMU

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

Služby střediska

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

Pracovní doba

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

>>>>>>> telefon 22 37 74 <<<<<<<

Seznam časopisů na mikrofiších

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

Použité zkratky:

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

Knihy na mikrofiších

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

Informace ze zahraničí

Bezpečnější monitory TAXAN

mají snížit úroveň elektromagnetického pole a elektrostatického náboje. Speciálním stíněním cívek obrazovky byl snížen elektromagnetický tok z 88mT/sek na 23mT/sek. Elektrostatické pole stínítka je sníženo speciálním povlakem z ± 6kV na ± 0.5kV. Cena těchto "ekologických" monitorů je o 50 GBP vyšší, než u standardních typů.

Textový procesor pro programátory

pracující v assembleru Z80 popisuje britský časopis PCW 5/89 - str. 240. Kompletní výpis programu jistě zaujme i naše programátory pracující s assemblerem Z80.

Laserové tiskárny

známých značek porovnával britský časopis Personal Computer Magazine. Jednalo se o následující typy:

Epson GQ-3500	HP LaserJet II
Honeywell Lasepage 801	Fujitsu RX7300E
Apple Laserwriter II NT	Qume ScripTen
Brother HL-8	Kyocera 2200
Facit P-7150	SBC Speedlaser
Star Laserprinter 8	IBM 4216

A výsledek testů? Časopis doporučuje dva typy tiskáren. Jsou jimi Qume ScripTen a Star Laserptinter 8. Bližší data najdete v PC Magazine 6-7/88.

PROGRAMOVÁ NABÍDKA

Naše organizace pro vás připravila bohatou nabídku zajímavých programů. Jejich přehled jste mohli najít v Mikrobázi 6/89. Dnes přinášíme stručné charakteristiky jednotlivých programů. V tabulce si pak už snadno najdete, zda příslušný program existuje i ve verzi pro váš počítač a jaká je jeho cena.

Protože se od doby, kdy jsme připravovali tabulku do čísla 6, došlo k jistým změnám, doplňte si laskavě do tabulky tyto údaje:

1) CP/M už máme i ve verzi pro SHARP.
2) K programům PLAYTAPE, REMBRANDT, DATALOG 2, MFLOG a TAPELOG pro ZX Spectrum si můžete přidělat hvězdičku, protože budou v prodeji už v nejbližší době.
3) V nejbližší době bude v prodeji též nový textový editor pro ZX Spectrum TOPTEXT. V tabulce uveden nebyl.
4) U programu GREP pro PMD si můžete podtrhnout cenu, protože už je v prodeji.
5) Pro počítač PMD jsou v prodeji programové soubory HRY1, HRY2 a HRY3 v cené 143,- Kčs za soubor. Tyto programy též nebyly v tabulce uvedeny.

Dodávka každého programového produktu je tvořena kazetou (disketou) s nahraným programem a uživatelskou příručkou.

UPOZORNĚNÍ: Každý program se smí provozovat v jednom čase pouze na jednom počítači. Kopírování programů, dat, nebo příruček dalším osobám není dovoleno.

Nabízené programy lze osobně zakoupit v prodejně 602. ZO Svazarmu (Praha 1, Martinská 5, telefon 22 87 74), nebo si je můžete na korespondenčním lístku objednat na adrese 602. ZO Svazarmu, Dr. Z. Wintra 8, 160 41 Praha 6. Na objednávce nezapomeňte uvést typ počítače a jeho verzi!

PROFESOR II

Univerzální výukový program, který je možno naplnit databázemi z nejrůznějších oborů. Zároveň s programem dostanete databáze "Naše hrady a zámky" (pouze Spectrum) a "Evropská města". Další báze lze přikoupit na kazetách STUDENT.

STUDENT 1

Pět znalostních bází pro program PROFESOR (Města ČSSR, Evropská pohoří, Světová moře a oceány, Slovní druhy (český jazyk), Souhvězdí)

STUDENT 2

Pět znalostních bází pro program PROFESOR (Naše pohoří, Významné vrcholy, Města světa, Křižovatky (dopravní výuka), Malá násobilka). Pro počítače PMD-85, IQ-151 a Atari je dodávaná modifikace STUDENT 2A.

TESTEDITOR

Program pro tvorbu bází k programu PROFESOR podle vlastních potřeb. Součástí dodávky jsou i tři ukázkové báze (Naše města, Evropská moře, Násobilka).

ZX MULTITASKING

Víceprocesový operační systém pro ZX Spectrum. Program umožňuje běh více programů na počítači zároveň a to bez jakýchkoliv hardwarových úprav! Součástí dodávky je i ukázkový program "Kalkulátor a zápisník".

GROS

Grafický systém pro podporu rozhodování je programem z oblasti umělé inteligence. Umožňuje volit nejvýhodnější variantu řešení z několika možných, mezi kterými se rozhodujeme. Součástí dodávky je i ukázková databáze.

ODA

Osobní databázový systém s jednoduchým a názorným ovládáním. Kromě obvyklých editačních a vyhledávacích možností dovoluje aritmetické výpočty, volbu formátu zobrazení i tisku atd. Dodávka pro ZX Spectrum obsahuje dvě modifikace vlastní báze, podpůrný program pro zakládání nových bází, ukázkovou databázi sklad a vzorové databáze formátů několika typických aplikací.

TEMPERAMENT - MOŽNOST VÝHRY!

Je nahlédnutím do světa psychologie. Zábavnou formou psychologického testu se můžete dozvědět, kdo vlastně jste a kam můžete zařadit své příbuzné a známé. Zjištěné výsledky jsou graficky znázorněny a slovně vysvětleny. Kazeta může obsahovat výherní lístek na prvních deset titulů této nabídky.

PROGRAF

Optimalizovaná a rozšířená verze programu "Prostorové grafy", který umožňuje názorné zobrazování funkcí o více proměnných jako prostorových ploch. Uživatel má možnost volby řady parametrů jako jsou úhly natočení a nadhledu, zneviditelnění zakrytých částí, detailnost kresby apod. Pro ZX Spectrum najdete na kazetě také rozsáhlý demonstrační program trojrozměrné animace v reálném čase, který byl vytvořen právě za pomoci programu PROGRAF.

PROGRAMOVÁ NABÍDKA

MIKRO BÁZE

SONDA 4D - POZOR SOUTĚŽ!

Zábavný program procvičující orientační schopnosti, logické myšlení a fantazii. Program vám umožní cestu do fantastického světa čtvrtého rozměru. Součástí dodávky je několik datových náplní pro simulaci různých prostorových úrovní prostředí. Na 2D úrovni se setkáte s klasickým bludištěm, ve 3D budete procházet prostorovým bludištěm a v úrovni 4D poznáte zakřivení prostoru, černé díry a další zajímavé jevy. V průběhu roku 1990 se vlastníci programu mohou zúčastnit soutěže se zajímavými cenami.

STOPKY

Časoměrný program umožňující náročná měření více paralelně běžících procesů a zpracování naměřených dat. Program je vybaven multitaskingem umožňujícím například paralelní odečítání a komentování časů editorem. Program se s výhodou uplatní při různých sportovních soutěžích a laboratorních měřeních, kde se odečítá větší počet časových údajů.

DR.MG

Upravená verze asembleru GENS 3 a monitoru MONS 3 pro ZX Spectrum.

DATALOG

Databázový systém pro počítač ZX Spectrum s mnoha možnostmi, včetně použití české a slovenské abecedy.

MIKROBÁZE PASCAL

Prostředek pro editaci, překlad a běh programů napsaných v jazyce Pascal. Jde o pomúcku vhodnou k výuce programování v Pascalu.

CP/M

Nejrozšířenější operační systém užívaný na profesionálních osmibitových počítačích. Použití tohoto operačního systému na ZX Spectrum vyžaduje hardwarovou úpravu počítače.

ASSEMBLER 80

Překladač asembleru umožňující programování na úrovni strojového kódu.

BASIC S

Výukový program seznamující se základními zásadami programování v jazyce BASIC počítače ZX Spectrum. Tento průvodce programováním je určen hlavně začátečníkům.

TELETEXT

Program, který umožní pomocí adaptéru příjem teletextu systému WST úrovně 1.5 (tj. teletext vysílaný též čs. televizí).

PLAYTAPE

Program pro správné seřízení a kontrolu magnetofonu, zaznamenaných dat, měření kmitočtu a komprese obrazovky.

REMBRANDT

Grafický program pro kreslení barevných obrázků s řadou rozšiřujících funkcí oproti známému programu ART STUDIO.

DATALOG 2

Nová verze databázového programu DATALOG s plnou kompatibilitou se starou verzí, zvýšenou rychlostí operací, formulářovým výpisem, příkazem "najdi" a novým typem dat s doprovodnými obrázky. DATALOG 2 lze objednat na společné kazetě s MFLOG a TAPELOG za 333,- Kčs.

MFLOG

Vhodný doplněk DATALOGU 2 pro konverzi dat z Masterfile do DATALOGU obou verzí. Vstup a výstup mezi páskou, microdrive i diskem.

TAPELOG

Automatický podrobný výpis obsahu pásky s přepisem do datového souboru DATALOGU a s údaji: jméno, délka, typ, chyba atd.

SOS COPY

Kopírovací program nové generace pro všechny typy záznamů. Soubory lze nahrávat v žádaném pořadí, s volitelnou mezerou, signálem apod. Při práci jsou uváděny informace o souboru a volné paměti.

DAM 0000 V2.0

Assembler pro PMD 85-1 včetně řádkového editoru, překladače, zpětného překladače a debuggeru.

>>> >>> >>> >>> >>> *(pokračování v příštím čísle)*

Ústřední výbor Svazarmu
a český výbor elektrotechnické společnosti ČSVTS

uspořádaly ve dnech 28. - 30.9.1989,
v Domě kultury ROH pracujících
Dopravních podniků v Praze 7,
již III. výstavu

SOFTWARE 89

Aktuálně zařazujeme několik záběrů z vernisáže, kde promluvili: Plk. ing. Šimek, Prof. Dr. Taraba DrSc. a ředitel KD DP, který také přestřihl pásku.

AMSTRAD

O nových mikropočítačích Amstrad řady 2000 naleznete podrobnou informaci v článku uvnitř tohoto čísla Mikrobáze. Pro lepší představu také připojujeme několik obrázků. Další podrobnosti se můžete dozvědět ve středisku VTI, Martinská 5, Praha 1.