MIKROBÁZE

06 _ PASCAL (2)

_ paralelní přenos dat (1)

_ paralelní interfejs (1)

_ 80K RAM

pro ZX Spectrum

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

OBSAH

Dalibor výpočetní techniky 2

Pascal (2.část) 4

Mikroprocesor Z80 (4. část) 10

Paralelní přenos dat (1. část) 15

Paralelní interfejs 23

● INFORMAČNÍ SBĚRNICE

Commodore C64 32

Jaké bude dnes počasí 34

● SPECTRUM

Obrazová paměť ZX Spectra (3. část) 36

80 K RAM pro ZX Spectrum 41

Hry pro ZX Spectrum 44

● IQ 151

Rady uživatelům IQ 151 49

● PROGRAMOVÁ NABÍDKA MIKROBÁZE

ZX Spectrum 51

Anstrad/Schneider 54

Pokyny k objednávání programů 56

Slovo k náhodným čtenářům 56

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

Praha, duben 1987

Dalibor výpočetní techniky

V kontrapunktu s názvem úvodníku nebudu nikterak zasahovat do hájemství Večerníčků ČST. Vida - první věta, a už se do ní vloudila zkratka. Všude zkratek, jako když člověk nahlédne do computerové dokumentace. Když už jsme u nich, měl bych jednu novou - DODO. Nejedná se o žádného vyhynulého ptáka do křížovky, i když bych neměl nic proti tomu, kdyby to, co mám na mysli, potkal stejný osud. Ať dlouho nenapínám (ale nakonec tu odyseu známe všichni) - probíháte oky obchodní sítě, sháníte, a nakonec to něco, či obdobu toho třeba i seženete...jenže to nemá potřebné plánované parametry. Je tedy nutno se v klidu posadit a promyslet, jak upravit to či ono, aby celek byl funkční. Tím momentem se ocitáme v životní situaci s indexem DODO - DOdělej DOma.

Nic proti inovačním trendům, ale DODO je jaksi pořád na svém místě. Pravda, posunuje se. Tu je z jedné strany produkce vystrnaděno, ale hned se zase šikovně usídlí v jiné. To, jak si libuje v oblasti výroby a distribuce naší elektroniky, jistě netřeba popisovat. Stanu-li se dodomákem koupí magnetofonu či televizoru, nemusím ještě tolik zoufat. Když ne já, tak kamarád, specializovaný na dodomistiku určitého výseku produkce, bude přesně vědět, kam sáhnout, čím pohnout, co dodat či ubrat, aby přístroj vydechl požadované parametry.

Jenže - DODO má svůj strop. Kdybych se sebevíc snažil, doma mi integrovaný obvod VLSI nevykouzlí nikdo. Byť bych byl sebegeniálnějším programátorem, nemohl bych přece věnovat zbytech života tomu, abych se samozásobil potřebným softwarem, díky němuž by se mi počítač konečně stal opravdovým pomocníkem. A teď DODO rad! Ani "baborač" nepomůže, neb na soft/hardwarový bol léčivé bylinky není.

I kozojedský vladyka byl svého času v nouzi, jež ho, jak historie praví, naučila housti. Kdyby už tenkrát byly počítače a Dalibor se jim ve věži věnoval, jistě by mu lid pražský do košíčků i kazety přikládal, aby na ně některou ze svých nejnovejších assemblerových kompozic přičinil. Znám jednoho takového Dalibora (ve věži ale není), an mnohé ze svých kompozic poskytl lidem všem. A bez košíčků. Až Mikrobáze přišla s košem proseb - tak se jeden z jeho (tolik potřebných) titulů skví v jejím softwarovém menu. Lid však se o Daliborově kompozici s předstihem dozvěděl, takto jeho sídlo s košíčky obléhaje, pokoje mu nedaje. I nejednou volal ku mně Dalibor o pomoc: "Je tu nějaký zeman, košíčkem mává, kompozice se silou mocí domáhá. Objasni mu, že takto počínati sobě nemá, item ani mně jemu vstřícným býti nepřísluší. Předávám ho k telefonu."

Zmíněný programátor však není jediným, koho softwarové nouze přinutila jak k dodomistice (zde výstižněji - dodomystice), tak k původní tvorbě krátkých pomocných programů. Daliboři výpočetní techniky. Objevují se i ve sféře hardwaru, kde dominují různá dodomistická zapojení interfejsů a vedlejších pamětí.

Zde je však na místě položit si několik otázek. Lze takový potenciál nadále nechávat ležet ladem? Neudělit impuls rozšíření jeho početní základny? Razit heslo "Ono si to nějak pomůže samo, co je nám vlastně po tom"? nebo "Máme své předpisy a tam o tomhle díkybohu nic není, tak co si budem pod sebou pálit křesla? My přece žádnou odpovědnost nikde zakotvenou nemáme! Prosím vás, jaké programy, interfejsy,

modemy? K čemu a na co? A navrch ještě literaturu a časopisy?" Ale ptejme se i zcela konkrétně. Udělali by naši Daliboři programy, které budete moci používat, kdyby za nimi nepřišla Mikrobáze? Na tuto otázku lze odpovědět jednoznačně - NE, tyto programy by nevznikly. Jejich autoři by neměli proč trávit jejich tvorbou dlouhé měsíce. I oni sami, jako většina z nás, museli předtím vystačit se zahraničními, mnohdy i nepříliš dokonalými náhražkami, které o naší rodné řeči nemají ani ponětí. Prostě jsme si na tento stav museli všichni zvyknout - z uzamknutého kruhu nevedla žádná cesta ven. A tady se hned vnuká neodbytná noticka - i když Mikrobáze takovou první cestou je, neměly by se co nejrychleji objevit další? Tady žádné DODO nepomůže.

Není to tak dlouho, co se v pražském TKM ze své bolesti vyznal Eda Smutný, autor mikropočítače ONDRA. Dozvěděli jsme se o tom, jak si zase jednou smlsla hloupá, ignorancí a společenskou netečností zaštítěná byrokracie. ONDRA jí neprošel. Máme snad v obchodech tak široký výběr tuzemských mikropočítačů, k nimž by vznikalo tuzemské softwarové zázemí? K nimž by byla dodávána jednoduchá externí zařízení? Ani jeden. Tím základem se mohl stát ONDRA - předevčírem černobílý, včera už barevný. Není žádný. Nějaké jedno procento z tržních fondů jednoho podniku nic - ani Ondru, ani Dalibora - nespasí. Moderátorka TKM po vyslechnutí monologu otce Ondry konstatovala: "Smutný, rozhněvaný muž". Ani tady žádné DODO nepomůže.

Zkuste si v prodejně SNTL zakoupit obecnému lidu srozumitelnou a praxi blízkou příručku programování počítače nebo stavby hardwaru. Nic. Ticho. Nezájem. Lokální redakční hráz se jeví býti neprodyšnou. Vydávejte si doma!? Ani tady se DODO neprosadí.

A časopisy? Zelená příloha AR je jediným, čtete správně - jediným (!) - specializovaným kouskem periodika, které se mikropočítačům věnuje. Kdeže sliby, kdeže loňské sněhy jsou! Co na to DODO? I ono se diví a rozpačitě ukazuje na všechny světové strany kolem nás.

Ve smutném výčtu by bylo možno pokračovat. To jsou však už "jen" důsledky chybného stavu. Chybného i v tom, že zlepšovatelé zlepšují buď zastaralé stroje, nebo zlepšují, co mělo být od počátku dobré. Má-li někdo produkovat, je přece bez diskuze, že jeho produkt má a musí být dobrý od svého zrodu. Tak dobrý, aby se na něj pozornost zlepšovatelů nesoustředila. Je jistě pozoruhodné, že tam, kde zástupy zlepšovatelů nejsou, se produkuje z větší části na světové úrovni, na níž žádných následných zlepšovatelů netřeba - úroveň kvality produkce je řízena vývojem vědeckého poznání.

Tato evoluce má své jasné zákonitosti, ale i potřeby, resp. podmínky, za nichž může probíhat. V nich není místo pro jakoukoli dodomistiku. V nich jsou tvůrčí síly zapojeny do společenské tvorby. A té se musí uvolnit prostor pro její přímý odraz ve společnosti. Netečnost, nezodpovědnost, laxnost, zkostnatělost, byrokratická procedurálnost, omezování akčního rádiusu invence - příčiny disfunkce toho, co by v opačném případě mohlo být funkční, společensky přínosné, evoluční.

Setrváváním ve stavu, který otevírá pole působnosti dodomismu nejen doma, ale často už i v zaměstnání, se nikam nedojde. Nakonec máme na to i jedno moudré přísloví - devět řemesel, desátá bída. Významné rozhodnutí vrcholných orgánů Svazarmu přiklonit se ke koncepci Mikrobáze jako zdravě rizikovému experimentu otevření pole působnosti původní tvorbě v mikropočítačové technice by mělo sloužit jako vůdčí

ideový příklad všem, kteří mají rozhodnutí podobného obsahu přímo v náplni své práce.

...aby počalo ubývat Daliborů výpočetní techniky i smutných rozhněvaných mužů, aby se nám z ní DODO odstěhovalo jednou provždy. Jen na jedno při vší kritice a hledání nových cest nesmíme zapomenout - že z nebe samo nic nepadá, že za nás nikdo nic neudělá. Zvolna se zvyšující počet Mikrobázi zasílaných příspěvků a námětů nasvědčuje tomu, že řada těch, kteří si to uvědomují, roste. A to je příslib do budoucna.

-kš/elzet-

Pascal

2. část

Pokračujeme v našem rychlokursu jazyka Pascal. Laické čtenáře (i odborníky) upozorňujeme, že pro velký rozsah látky zde nemůžeme poskytnout vyčerpávající informace o všech detailech Pascalu. Zájemce o vážnější práci s tímto jazykem odkazujeme na knihu Programování v jazyku Pascal trojice autorů Jinoch, MÜller, Vogel (vydalo SNTL v roce 1985). Nemáte-li ji, zkuste se zeptat v některé knihovně.

Na úvod pár krátkých definic pro doplnění základních pojmů:

Pascal je jazyk s blokovou strukturou programu. Blok může obsahovat dílčí, vnořené bloky.

Identifikátor je posloupnost písmen a číslic začínající písmenem. Je jím např. jméno proměnné, konstanty, typu, procedury apod. Některé identifikátory jsou standardní, jazykem předdefinované (REAL, BOOLEAN, WRITELN atd.), jiné musejí být definovány (deklarovány) - např. proměnné.

Konstanty -symbolická jména korkrétních hodnot. Slouží především rychlé orientaci v programu. Konkrétní hodnota (literál) 3.14 může mít deklarované jméno PI, které pak symbolizuje uvedenou hodnotu při jeho použití v programu.

Proměnná-datový objekt, jehož hodnota se (na rozdíl od konstanty) může v průběhu programu měnit. Každá proměnná je určitého typu, kterým je určena množina přípustných hodnot proměnné. Její typ a identifikátor se určují deklarací.

Výrazy - jimi se předepisují posloupnosti operací. Výrazy obsahují operandy (např. proměnné a konstanty), operátory (+, -, and apod.) a okrouhlé závorky. Jednotlivé operace se provádějí zleva doprava s ohledem na prioritu operátorů a na závorky.

Priorita operátorů:

1) not - nejvyšší priorita
2) * / div mod and
3) + - or
4) znaménka rovnosti a nerovnosti - nejnižší priorita

STANDARDNÍ TYPY PROMĚNNÝCH

Jak už víme, v Pascalu musíme deklarovat názvy a typy v programu použitých proměnných. Pascal obsahuje celkem čtyři typy proměnných:

INTEGER

Je konečnou souvislou podmnožinou celých čísel. Konečnou podmnožinou proto, že u každého počítače je nejvyšší vyjádřitelné číslo determinováno velikostí použitého slova mikroprocesoru i celkovým počtem bitů paměti. Limit této podmnožiny je určen m.j. i způsobem, jakým překladač s čísly pracuje. INTEGER je typ celých čísel. Deklarace typu INTEGER mohou být např. takovéto:

```
var PočetHousek : integer;

var Procenta : 0..100;
```

Ve druhém případě může proměnná Procenta nabývat hodnoty z intervalu 0-100.

REAL

Množina hodnot typu REAL je konečnou podmnožinou reálných čísel. Platí pro ni omezení uvedená u typu INTEGER. REAL je tedy typ reálných čísel. S touto proměnnou jsme se již seznámili:

```
var Plocha : real;
```

BOOLEAN

Množina logických hodnot, které se označují předdefinovanými identifikátory TRUE a FALSE (standardní identifikátory konstant typu BOOLEAN). Množina logických hodnot je uspořádána tak, že platí FALSE < TRUE. Deklarovaná proměnná:

```
var Podmínka : boolean;
```

může být užita např. takto:

```
Podmínka :=true;
Podmínka :=(2=3);
Podmínka :=Podmínka and not Podmínka;
```

CHAR

Hodnotami typu CHAR jsou znaky (obvykle podle ASCII). Umísťují se mezi apostrofy. Např. po deklaraci:

```
var Odpověď : char;
```

můžeme v programu přidělit:

```
Odpověď :='s';
```

Podobně jako u typu INTEGER, i zde můžeme zavést interval:

```
var VelkáPísmena : 'A'.. 'Z';
```

Velmi důležitým pravidlem Pascalu je zásada shodnosti typů. Není dovoleno přidělovat proměnné určitého typu hodnoty, které jí nenáleží. Tak proměnným typu BOOLEAN nelze přidělovat hodnoty celých čísel (typ INTEGER). Jako vždy, i zde platí pravidlo výjimky (je výjimkou jedinou): je přípustné přidělovat hodnoty celých čísel proměnné

typu REAL. Tento převod se nazývá konverze mezi typy INTEGER a REAL.

ARITMETICKÉ OPERACE

S proměnnými typu REAL můžeme provádět nám dobře známé čtyři základní operace se znaménky + - * /.

Tři první operátory (vyjma lomítka) můžeme použít při výpočtech s proměnnými typu INTEGER. Operátor dělení/ je pro ně "nahrazen" těmito operátory:

a div b - celočíselná část výsledku dělení a b
a mod b - zůstatek výsledku dělení a b

A div B je obdobou basicové operace INT(A/B). A mod B obdobou basicové operace A/B-INT(A/B).

Při výpočtech můžete dále používat tyto standardní funkce:

abs (x), sqr (x), sin (x), cos (x), arctan (x), exp (x), ln (x), které jsou obecně známé. Možná nebude někdo z vás znát následující, proto je uvádíme s vysvětlivkami:

sqrt (x) - druhá odmocnina x (zatímco sqr je druhá mocnina)
trunc(x) - celá část čísla x ("zaokrouhlení dolů")
round(x) - trunc (x+.5) pro nezáporná x
= trunc (x-.5) pro záporná x
odd (i) - parita; true pro lichou, false pro sudou

LOGICKÉ OPERACE

Pro logické proměnné a výrazy (typ BOOLEAN) Pascal užívá tyto jednoduché operace s operátory and, or, not:

a and b - konjunkce (log. součin)
a or b - disjunkce (log. součet)
not a - negace

OPERACE SE ZNAKOVÝMI PROMĚNNÝMI

ord (x) - kódové číslo znaku x
chr (x) - znak kódu x
succ(x) - znak s kódem o 1 vyšší než kód x
pred(x) - znak s kódem o 1 nižší než kód x
Z toho např. vyplývá, že succ /zn/=chr(ord/zn/+1), tedy např. succ ('A') = 'B'

STAVBA ALGORITMU

Známe již podmíněné instrukce "IF podmínka THEN instrukce" a "IF podmínka THEN instrukce ELSE instrukce." Podmínka je typu BOOLEAN, instrukce může být buďto jednoduchá nebo složená. Ta druhá představuje sled instrukcí (mohou být opět složené) strukturálně vymezených klíčovými slovy BEGIN a END (k nim dále patří IF, THEN, REPEAT a WHILE). K posledním dvěma si brzy řekneme víc.

Představme si, že v programu je takovýto sled instrukcí:

```
   if x = 0 then write ('špatně')
   if x = 1 then write ('středně')
   if x = 2 then write ('dobře')
```

Podle hodnoty, jakou nabyde x, se pak v programu odehraje náležité. Všimněte si, že podmíněnou instrukci THEN jsme zde užili třikrát. Tento a jemu podobné zápisy můžeme zjednodušit:

```
case x of
   0:write ('špatně');
   1:write ('středně');
   2:write ('dobře')
end.
```

CASE x OF bychom mohli trochu delším opisem přeložit jako "v případě, že x nabyde některé z hodnot před dvojtečkou, proveď instrukci umístěnou za ní". V uvedeném zápisu je ovšem nutné, aby x skutečně nabylo jedné z uvedených hodnot. Jinak nelze předpovědět výsledek operace. Tomu však můžeme předejít takto:

```
case x of
     0:write ('špatně');
     1:write ('středně');
     2:write ('dobře')
     else write ('hrozně')
end.
```

Nezapomeňte dodržet pravidla syntaxe Pascalu - před ELSE není středník! Operace instrukce CASE musí být zakončena slovem END.

SMYČKY

jsou jedním ze základních kamenů jakéhokoli programování. Pro jejich stavbu nám Pascal nabízí několik možností. Např.:

WHILE podmínka DO instrukce

Znamená - dokud platí uvedená podmínka, prováděj danou instrukci. Příklad programového užití:

```
   i :=0;
   while i < 1000 do
     begin
       writeln (i);
       i :=i+1;
end.
```

Uvedená ukázka vypíše sloupec čísel od 0 do 999. Další možnost stavby smyčky:

REPEAT instrukce UNTIL podmínka

Jinými slovy - opakuj danou instrukci dotud, dokud není splněna uvedená podmínka (resp. dokud má hodnotu false). Např.:

```
   i :=0;
   repeat
     writeln (i);
     i :=i+1;
   until  i > =1000;
```

Protože slova REPEAT a UNTIL ohraničují smyčku, nevyžaduje zde Pascal použití slov BEGIN a END.

Výsledný efekt obou prográmků je stejný. Přesto oba uvedené typy tvorby smyček nejsou zdaleka identické. Jeden z velmi důležitých rozdílů je v tom, že WHILE zjišťuje platnost podmínky před provedením instrukce ve smyčce, zatímco REPEAT až po jejím provedení. Proto je instrukce smyčky REPEAT provedena vždy nejméně jednou, zatímco u WHILE nemusí dojít k žádnému provedení. Na to nikdy nezapomeňte!

A jsme u možnosti třetí, poslední:

FOR proměnná :=počáteční hodnota TO koncová hodn. DO instrukce

Instrukce se provádí, dokud proměnná nenabyde koncové hodnoty. Je to obdoba všem basicovým znalcům dobře známé smyčky FOR...NEXT. Použijeme jí pro vytvoření funkčního ekvivalentu předchozích dvou smyček:

```
for Počet :=0 to 999 do writeln (Počet);
```

Proměnná Počet musí být typu INTEGER. Místo slova TO můžeme použít DOWNTO pro sestupný odpočet:

```
for Počet :=999 downto 0 do writeln (Počet);
```

Na rozdíl od Basicu nemůžeme volit krok (STEP) smyčky. Krok je vždy roven 1 (resp. -1). Pro jiný průběh musíme použít jednu z prvních dvou možností.

Tak už jsme probrali tři ze čtyř druhů strukturovaných příkazů Pascalu: složené, podmíněné (IF a CASE) a cyklu (REPEAT, WHILE, FOR). V dalším na nás čeká poslední z nich, příkaz WITH.

Následující čtyři programové ukázky demonstrují, co jste se dosud z Pascalu naučili.

```
program Minčíslo;

{Nejmenší číslo, jaké může
počítač reprezentovat}

const Nula = 0.0;
var Iks1, Iks2 : real:,

begin
   Iks1 :=1;
   writeln ('Moment');
   while Iks1 <> Nula do
     begin
       Iks2 :=Iks1;
       Iks1 :=Iks1/2;
     end;
   writeln ('Nejmenší číslo je: ', Iks2)
end.
```

```
program Otázka;
var Roky: integer;
begin
   writeln ('Kolik let zbývá do konce století');
   writeln;
   repeat
     read (Roky);
     write ('- ');
     if Roky=13 then
        writeln  ('Správně!')
     else
        writeln ('Chyba! 'Zkus znova!');
   until Roky=13
end.
```

```
program ASCII;

  {Tabulka ASCII kódů znaků od 32 do 255
   při použití standardní funkce chr }

const Radky = 20;
      Sloupce = 5;
   var Index : integer;
      Enter  : char;

   begin
     writeln ('Tabulka kódů ASCII');
     writeln;
     for Index :=32 to 255
       begin
         write (Index, ':', chr (Index),' ');
         if Index mod Sloupce=0 then
          writeln;
         if Index mod (Radky*Sloupce)=0 then
          begin
           writeln ('Stiskni ENTER...');
           read (Enter)
         end
     end
end.
```

```
program Test;

var Odpověď : char;

begin
   writeln ('Otázka:');
   writeln;
   writeln ('Co je to RAM?');
   writeln;
   writeln ('A - Dřevěný lem obrazu                              ');
   writeln ('B - Volně přístupná paměť                           ');
   writeln ('C - Anglicky beran                                  ');
   writeln ('D - Restaurace a menzy                              ');
   writeln ('Vyber jednu z odpovědí A, B, C nebo D!              ');
   read (Odpověď);
   writeln;

   case Odpověď of
    'A', 'a' : writeln (Bez čárky nad A?!');
    'B', 'b' ,
    'C', 'c' : writeln ('Výborně!          ');
    'D', 'd' : writeln ('Nikolivěk         ')
else
   writeln ('To neznám')
 end
end.
```

BAJTEK 11/86
-elzet-

Mikroprocesor Z80

AUTONOMNÍ ASSEMBLEROVÉ RUTINY

4. část

V této části si probereme rutinu RELOC, která přemístí program ve strojovém kódu s automatickým přetransponováním všech absolutních adres v instrukcích volání a skoků, takže se o ně nemusíme starat. Různé varianty této funkce mají zahrnuty všechny assemblery a profesionálnější disassemblery. Součástí rutiny RELOC je rutina IBT, kterou si probereme současně, a rutina LENGTH (tu najdete ve 4. č. zpravodaje).

RELOC transponuje jen adresy, které adresují jakékoli místo paměti uvnitř transponovaného bloku. Ostatní zůstávají nezměněny. Na to je třeba pamatovat. Stejně tak je transponován 16tibitový operand každé instrukce typu LD, což může někdy vadit. A nakonec je třeba si uvědomit, že touto rutinou nemůžeme přenášet definovaná data, která by při přenosu bloku byla pojímána jako instrukce. Na ně stačí samotná rutina IBT pro tzv. inteligentní přenos.

RELOC — Přemístění kódu v paměti

READR — Přičtení ofsetu k absolutní adrese instrukce

:Činnost — RELOC - přenesení bloku a určení ofsetu.
READR - transpozice přičtením ofsetu, pokud instrukce adresuje místo uvnitř bloku.

:Akce — Přenos bloku subrutinou IBT
Pokud instrukce obsahuje absolutní adresu, pak ji vyjmi a přičti k ní ofset. Když je výsledek v mezích adres bloku, proveď změnu, jinak ponech.

:CPU — Z80

:Hardware — RAM

:Software — IBT pro čistý přenos bloku
LENGTH pro vyčlenění instrukcí připadajících v úvahu pro transpozici

:Vstup — RELOC: BC - první adr. přenášeného bloku
HL - poslední adr. přenášeného bloku
DE - první adr. nového umístění bloku
READR: HL - " " " " "
DE - poslední adr. " " "
BC - ofset

:Výstup — Všechny registry kromě IX a AF´ změní obsah

:Chyby — Definovaná data a 16tibitové operandy instrukcí LD jsou rovněž zváženy pro transpozici

:Registry — AF, BC, DE, HL, IY, BC´, DE´, HL´

:Zásobník — 8(včetně volání IBT a LENGTH)

:RAM

:Délka — 121

:Cykly — Neuvedeny

:Třída 2 — -diskrétní — -přerušitelná — *promovatelná

--*-*- — -opakovatelná — *relokovatelná — -robustní

```
:
:... BC=1.adr. zdroje, HL=posl. adr.zdroje, DE=1. adr. nového umíst.
:
RELOC   OR    A         ;Nastavení CY na 0 a výpočet      B7
        SBC   HL,BC     ;bajtů v bloku -1                 ED 42
        PUSH  BC        ;Výměna HL s BC přes zásobník,    C5
        EX    (SP),HL   ;takže HL=1. adr. zdroje a        E3
        POP   BC        ;BC=počet bajtů-1                 C1
:
        PUSH  BC        ;Uložení počtu bajtů-1,           C5
```

```
        PUSH  DE        ;1. adr. nového umístění a          D5
        PUSH  HL        ;1. adr. zdroje                     E5
        EXX             ;Převedení obsahů HL a DE           D9
        POP   HL        ;do altern. registrů                E1
        POP   DE                                            D1
        OR    A         ;Nastavení CY na 0 a výpočet        B7
        SBC   HL,DE     ;ofsetu: zdroj-nové umístění        ED 52
        EX    (SP),HL   ;Počet bajtů-1 do HL a              E3
        POP   BC        ;ofset do BC                        C1
        ADD   HL,DE     ;Posl. adr. nového umístění         19
        EX    DE,HL     ;do DE, HL=1. adr. nového umístění  EB
        EXX             ;Jejich uložení do alt. banky       D9

:
        INC   BC        ;Skutečný počet bajtů               D3
        CALL  IBT       ;a čistý přenos bloku               CD lo hi
        EXX             ;Přepnutí alter. registrů           D9
        RET   Z         ;Zpět, když přenos nebyl žádný      C8

:
:...HL=1.adr.bloku, DE=posl.adr.bloku, BC=ofset
:
READR   INC   DE        ;Adr. bloku+1                       13
        PUSH  HL        ;1. adr. do IY                      E5
        POP   IY                                            FD E1
        PUSH  HL        ;1.adr. do altern. reg. HL          E5
        EXX             ;Přepnutí reg. bank                 D9
        POP   HL        ;Ukazatel adr. instrukce            E1

:
LOOP    PUSH  HL        ;Jeho uložení, dokud trvá LOOP      E5
        PUSH  HL        ;  "     "   ,  "   nedostane      E5
        CALL  LENGTH    ;délku  instrukce do reg. E         CD lo hi
        POP   HL        ;Obnovení ukazatele                 E1
        PUSH  DE        ;Uložení délky, dokud trvá LOOP     D5

:
        LD    A,E       ;Test délky instrukce, když je      7B
        CP    3         ;menší než 3,                       FE 03
        JR    C,NXTI    ;nebude se transponovat             38 3D
        LD    A,(HL)    ;1.bajt instr. do reg. A            7E
        JR    Z,INDX    ;Když délka=3, pak test IX, IY      28 10

:
:
        INC   HL        ;Délka=4, proto ukazatel+1          23
```

```
        CP    #ED       ;Když je to #ED,                FE ED
        JR    Z,ADDR    ;zjisti adresu                  28 13
        LD    A,(HL)    ;Druhý bajt do A                7E
        CP    #CB       ;Když je to #CB, pak bez        FE CB
        JR    Z,NXTI    ;transpozice, jdi zpět          28 30
        CP    #36       ;Stejně tak, když jde o         FE 36
        JR    Z,NXTI    ;instr. LD (X/Y+d)              28 2C
        JR    ADDR      ;Jinak zjisti adresu pro tran.  18 08

:
INDX    CP    #DD       ;3-bajt. instrukce              FE DD
        JR    Z,NXTI    ;Ty mají adresy pro trasp.,     28 26
        CP    #FD       ;pokud to ovšem nejsou          FE FD
        JR    Z,NXTI    ;instr. s IX nebo IY            28 22

:
ADDR    INC   HL        ;Nižší bajt adresy instrukce    23
        LD    C,(HL)    ;do reg. C                      4E
        INC   HL        ;Vyšší bajt adresy              23
        LD    B,(HL)    ;do reg. B                      46

:
        PUSH  BC        ;Přenos  adresy                 C5
        EXX             ;do                             D9
        POP   HL        ;altern. reg. HL a přičtení     E1
        ADD   HL,BC     ;ofsetu z altern. reg. BC       09
        PUSH  HL        ;Přenos nové adresy             E5
        EXX             ;do                             D9
        POP   BC        ;norm. reg. BC                  C1

:
        PUSH  IY        ;1. adr. bloku do HL přes zásob. FD E5
        EX    (SP),HL   ;Ukazatel do zásobníku          E3
        OR    A         ;Nastavení CY na nulu a test    B7
        SBC   HL,BC     ;1.adr. bl. se změněnou adresou ED 42
        POP   HL        ;Obnovení ukazatele             E1
        JR    Z,RPLC    ;Jdi na změnu adr., když je na  28 0B
        JR    NC,NXTI   ;zač. bloku, je-li níž, tak ne  30 0C

:
        PUSH  BC        ;Nová adresa                    C5
        EXX             ;do                             D9
        POP   HL        ;altern. reg. HL                E1
        OR    A         ;                               B7
        SBC   HL,DE     ;Test, zda nová adresa          ED 52
```

```
        EXX             ;není výš, než konec bloku          D9
        JR    NC,NXTI   ;Je-li výš, žádná transpozice       30 03
:
RPLC    LD    (HL),B    ;Výměna staré abs. adresy           70
        DEC   HL                                            2B
        LD    (HL),C    ;novou adresou                      71
:
NXTI    POP   DE        ;Obnovení délky instrukce           D1
        POP   HL        ;a ukazatele                        E1
        LD    D,0                                           16 00
        ADD   HL,DE     ;Ukazatel na další instrukci        19
:
        PUSH  HL        ;Ukazatel                           E5
        EXX             ;do                                 D9
        POP   HL        ;altern. reg. HL                    E1
        OR    A                                             B7
        SBC   HL,DE     ;porovnání s adr. bloku+1           ED 52
        EXX             ;Přepnutí reg. bank                 D9
        JR    C,LOOP    ;Celý blok hotov? NE, pak LOOP      38 A9
        RET             ;ANO, pak konečný návrat            C9
:
:
```

```
:...HL=1. adr. zdroje, DE=1. adr. nového místa,  BC=počet bajtů bloku
:
IBT     OR    A         ;Nastavení CY na 0                  B7
        SBC   HL,DE     ;Rozdíl 1. adres zdr. a nov. bl.    ED 52
        ADD   HL,DE     ;Obnovení HL beze změny indik.      19
        RET   Z         ;Zpět, když 1. adresy shodné        C8
:
        JR    C,BACK    ;Skok, když DE větší než HL         38 03
        LDIR            ;Jinak proveď přenos                ED B0
        RET             ;a vrať se                          C9
:
BACK    ADD   HL,BC     ;Převedení 1. ukazatele             09
        DEC   HL        ;na posl. adr. zdroje               2B
        EX    DE,HL     ;Převedení 2. ukazatele             EB
        ADD   HL,BC     ;na posl. adr.                      09
        DEC   HL                                            2B
        EX    DE,HL     ;nového bloku                       EB
```

```
LDDR        ;Provedení přenosu                  ED B8
SCF         ;CY=1; nová adr. výš než zdroj      37
RET         ;Návrat zpět                        C9
```

Rutinu IBT (Intelligent Block Transfer) můžete užít i samostatně pro čistý přenos bloku bajtů bez jakékoli změny jejich obsahu. Rutina je ve třídě 1 (******) - diskrétní, přerušitelná, promovatelná, opakovatelná, relokovatelná, robustní. Když je na výstupu CY=0, proběhl přenos na nižší adresy, při CY=1 na vyšší. V prvním případě je na výstupu HL=zdroj+1 a DE=nový blok+1, ve druhém HL=zdroj-1 a DE=nový blok-1. V obou pak Z=0 i BC=0. Je-li na vstupu HL=DE, na výstupu je Z=1. Délka IBT je 20 bajtů. Počet cyklů v případě HL=DE je 41. Při přenosu na nižší adresy je počet cyklů 47+BC*21, přenos na adresy vyšší zabere 89+BC*21 cyklů.

Příště obrátíme svou pozornost na zajímavou rutinu NEWPC, která umožňuje krokovat programem. Tato fukce je nezbytná pro ladění programů ve strojovém kódu a je základním stavebním kamenem monitorů, které mají schopnost provádět anylýzu programů.

Assembler Routines for Z80
D. Barrow, 1985
přeložil a upravil
-elzet-

Paralelní přenos dat

Část 1.

Tento článek se zaměřuje na použití obvodů s nižší hustotou integrace (non-LSI). Jako příklady zapojení budou uvedeny paralelní interfejsy pro klávesnici a Centronics pro tiskárnu. V další části budou demonstrovány různé obvody PIA a programovatelné periferní interfejsy pro obecné užití. Konkrétní příklady obvodů zahrnou i verzi s obvody LSI pro klávesnici a interfejs tiskárny a stykovou jednotku PIA pro Apple II, zajišťující řízení přenosu analogových dat.

ZÁKLADY

Mikroprocesory a interfejsové adaptéry obsahují obvody NMOS, nověji CMOS, jejichž tranzistory pracují s negativní či pozitivní logikou. Vyznačují se velmi nízkou spotřebou energie.

Většina známých mikroprocesorů jsou typu NMOS. Např. 8080A, 8086, 6502, 6809 apod. Z řady CMOS to jsou např. 65C02, 68020, HD64180, 80C85 a další. Hradla,střadače a buffery - "tmely" systémů - pracují s logikou TTL, která je ve spojení s obvyklými systémy počítačů nejefektivnější.

DVOJČINNÝ VÝSTUP

Skupina obvodů TTL se dělí na několik podskupin s tímto značením: LS (nízkovýkonové Schottky), F (rychlé), ALS (zlepšené nízkovýkonové Schottky) a další. Různé typy těchto obvodů mají výstup typu dvojčinného výstupu s otevřeným kolektorem nebo

třístavový výstup.

Dvojčinný výstup je jako standardní užit ve většině čipů hradel, čítačů, násobičů a dekodérů. Jeho anglický název totem-pole vychází z faktu, že se skládá ze dvou tranzistorů (někdy z párů Darlingtonových tranzistorů) postavených na sobě jako totemový sloup. Výstup je umístěn mezi oběma tranzistory. Když je horní tranzistor sepnut, na výstupu je vyšší napětí, než je-li sepnut tranzistor dolní (viz **obrázek 1a**). Výhodou tohoto výstupu je, že nízká impedance řídícího zdroje umožňuje ovládat zátěže s vysokou kapacitou bez výrazného snížení délky spínacích časů.

Tranzistory dvojčinného výstupu nejsou sepnuty ve stejném okamžiku, kromě momentu překlopení logických úrovní, kdy celý pár zkratuje napájecí napětí +5 voltů proti zemi. Výsledkem je pro obvody TTL typický šumový jehlový impuls. Z tohoto důvodu je v TTL logice zapotřebí tolika kondenzátorů omezujících šum.

VÝSTUP S OTEVŘENÝM KOLEKTOREM

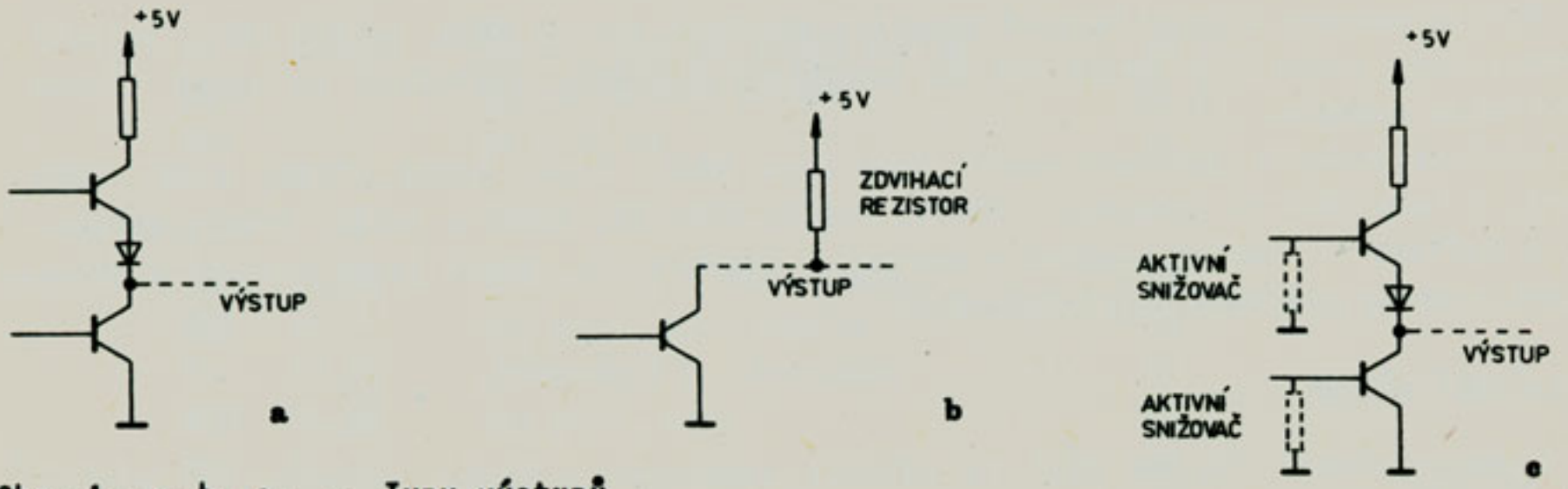
Někdy potřebujeme spojit výstupy dvou nebo více obvodů dohromady. Pro ten případ použijeme výstupu třístavového nebo s otevřeným kolektorem.

Obrázek 1b ukazuje typický výstup s otevřeným kolektorem. Od zapojení na **obrázku 1a** se liší tím, že horní tranzistor "totemového sloupu" chybí, kolektor dolního tranzistoru je otevřený, tedy nezapojený, resp. připojený k výstupnímu kolíku. Protože žádné aktivní prvky obvodu nespojují výstup s kladným napájením, tento typ výstupu může být překlopen na vyšší úroveň pouze na vnější podnět (např. rezistorem připojeným ke kladnému pólu zdroje).

Zatímco výstup s otevřeným kolektorem umožňuje propojení mnoha prvků obvodu, nedostatek aktivních komponentů na "vysoké" straně obvykle prodlužuje dobu přepínacího času. Použití vnějších rezistorů má za následek "unikání" proudu přes tyto rezistory. Snížením hodnoty rezistorů můžeme čas překlopení zkrátit, ovšem za cenu zvýšení úniku proudu.

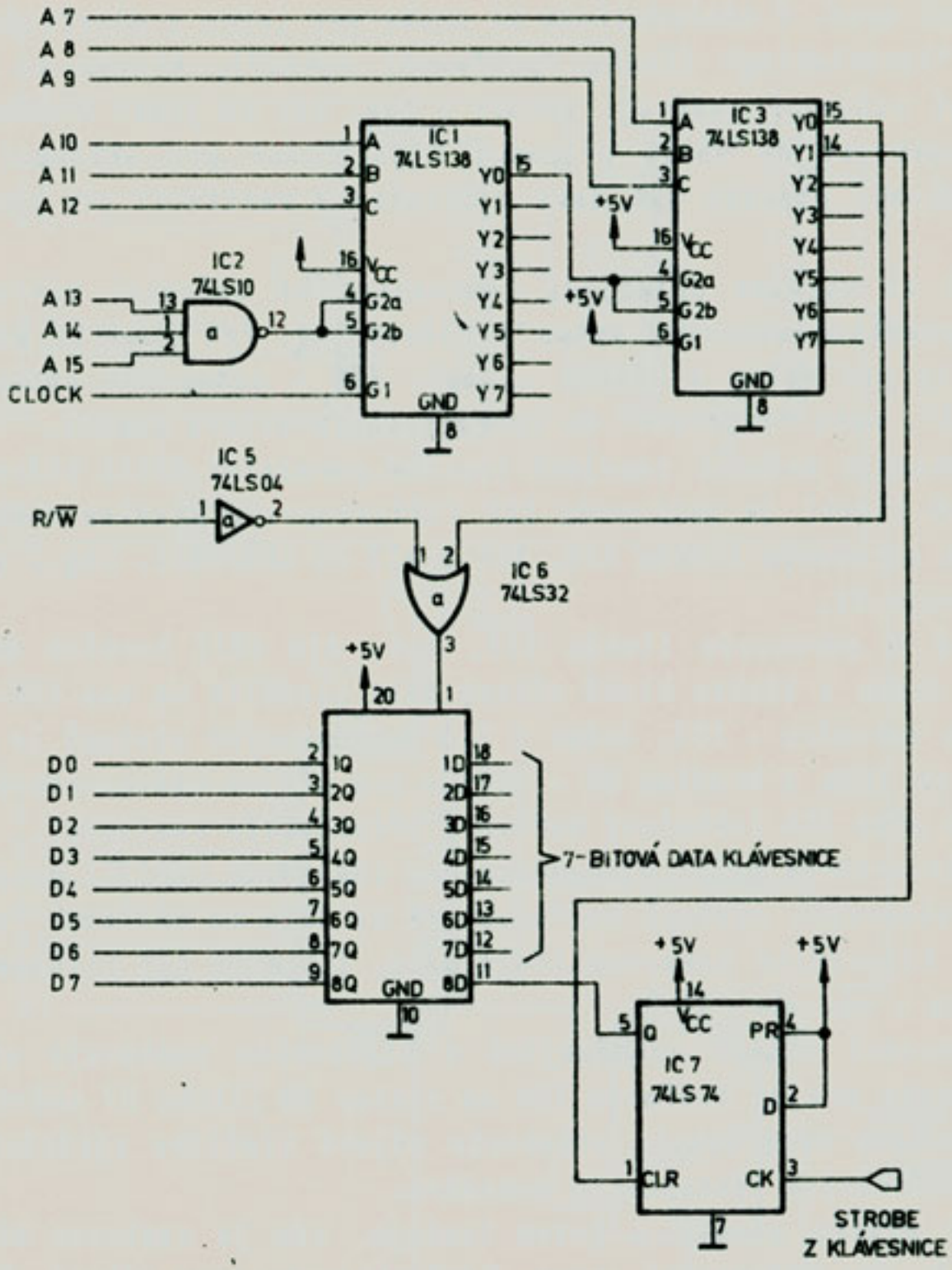
TŘÍSTAVOVÝ VÝSTUP

Tento typ zapojení (viz **obr. 1c**) pracuje při "vypnutí" aktivních vstupních obvodů



Obr. 1. a, b, c, - Typy výstupů

a) Dvojčinný výstup **b) Výstup s otevřeným kolektorem** **c) Třístavový výstup**



Obr. 2 a - Klávesnicový vstup s Bufferem 74LS541

stejně jako zapojení dvojčinného výstupu. Když jsou vstupní obvody "zapnuty", oba tranzistory se vypnou a na jejich bázích je velmi malý potenciál. Tento třetí stav se vyznačuje vysokou výstupní impedancí.

Uvedené zapojení kombinuje vysokou přepínací rychlost "totemového sloupu" s hlavní dispozicí zapojení s otevřeným kolektorem - tak s jeho pomocí lze vzájemně propojovat různé prvky zařízení.

Zapojení s otevřeným kolektorem a třístavovým výstupem se používá nejčastěji tam, kde je propojeno dohromady mnoho výstupů, jako např. u datových a řídících linek mikropočítačových systémů.

PARALELNÍ INTERFEJS KLÁVESNICE

Mikropočítač s jeho mikroprocesorem a pamětí (RAM a ROM či EPROM) musíme "naučit" domluvit se s vnějším světem - různými periferními zařízeními, jako diskovými jedno-

tkami, tiskárnami apod. V následujícím si probereme jednu z důležitých pasáží interfejsingu, tzv. "handshaking" (česky "potřesení rukou"), kterým si mikroprocesor a externí zařízení vzájemně podávají zprávy o tom, kdy má jeden druhému předávat data a kdy přenos zastavit do další případné signalizace pro aktivaci přenosu. Jedná se tedy o koordinaci přenosu dat mezi počítačem a periférií při paralelním přenosu dat.

Jako příklad si vezmeme připojení klávesnice ke sběrnici mikroprocesoru. Dále bude zvažován paměťově mapovaný vstup z klávesnice. Při použití mikroprocesoru se separátními porty pro vstup a výstup - např. Z80 - mohou být vstupy a výstupy buď přímé, "průchozí", nebo "mapované" do vyhrazeného prostoru paměti.

Obrázek 2a ukazuje možnost zapojení typické paralelní ASCII klávesnice. IC1 (dekodér matice 3x8 linek) je aktivován IC2 (třívstupovým hradlem NAND), jehož vstupy jsou připojeny k adresovým linkám A15, A14 a A13 na sběrnici. Log.1 signálu systémových hodin aktivuje G1.

IC1 připojený k adresovým linkám A12, A11, A10 dělí horních 8K paměti 64K do osmi 1K bloků. Jeden z těchto bloků můžeme dále rozdělit na osm 128-bajtových částí přidáním druhého dekodéru - IC3. Jeden z IC3 dekódovaných výstupů je použit v kombinaci s invertorem IC5a a hradlem OR - IC6a - pro příjem klávesnicových dat klíčovaných na sběrnici přes IC4 (oktalový buffer s třístavovým výstupem).

Hradlo OR s invertorem brání mikroprocesoru zapisovat (WRITE) na adresu klávesnice aktivací třístavových výstupů bufferu, což by mělo za následek "zmatek" na sběrnici. (Jestliže klávesnice sama neklíčuje data, IC4 může být klíčovací obvod s třístavovým výstupem - viz **obr. 2b.**)

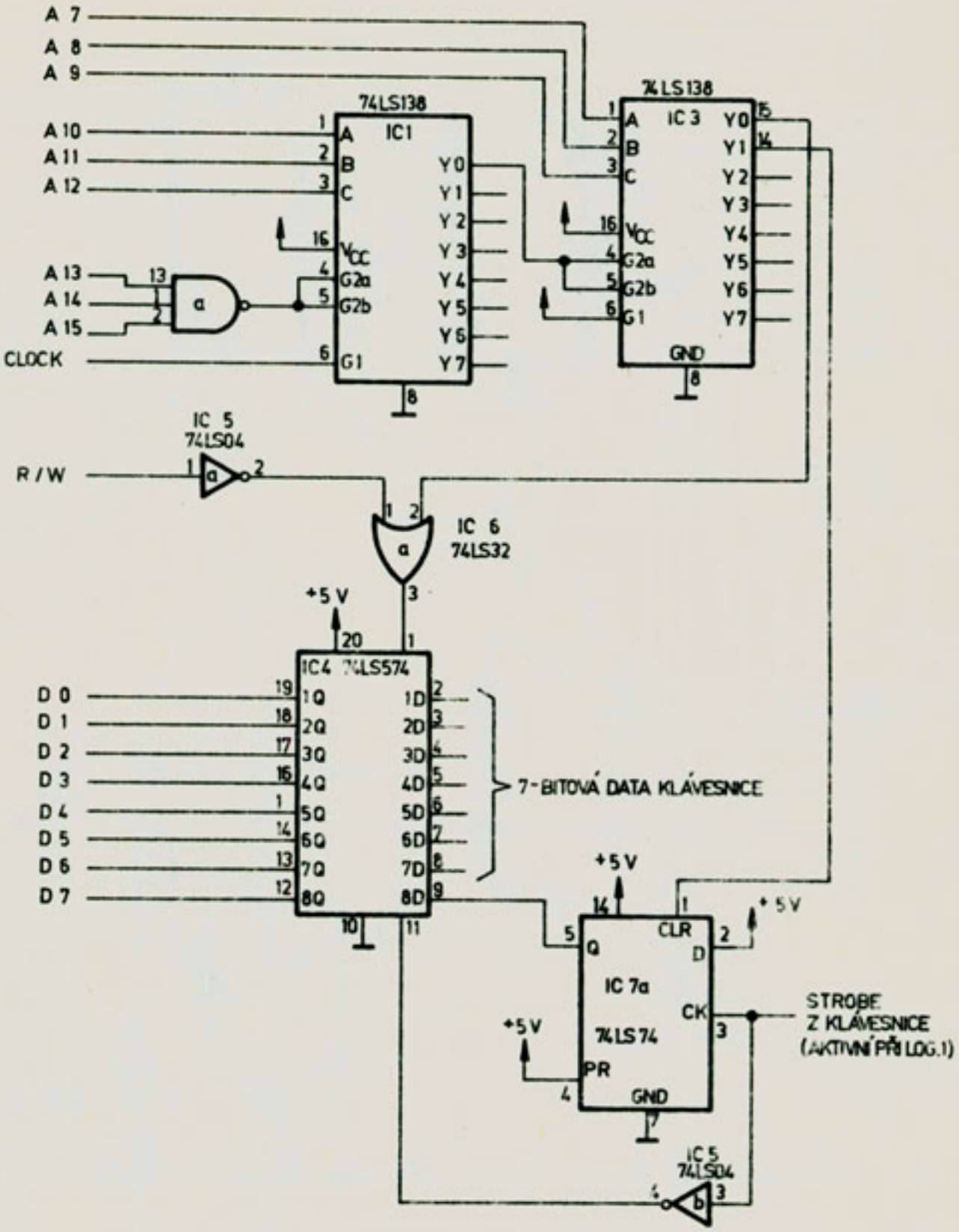
Aktivní strobovací signál vycházející z klávesnice nastavuje výstup Q IC7 na log.1. Jeho hodnota projde bufferem IC4 na výstup 8Q (linku D7 sběrnice). Mikroprocesor určí, zda bylo nebo nebylo stisknuto nějaké tlačítko zjištěním obsahu jedné ze 128 adres dekódovaných výstupem Y0 dekoděru.

Mikroprocesor přečte adresu a potom testuje, zda je nastaven bit D7. Pokud je D7 ve stavu log.0, čte klávesnici znova. Když mikroprocesor zjistí, že nějaká klávesa byla stisknuta, zjistí si její adresu z dekodéru a jeho výstupem Y1 převrátí bit D7 do stavu log.0 (přes CLR IC7).

Celý proces si můžeme popsat assemblerem 6502 takto:

```
FETCH:   LDA KEYB      ;Přečti adresu klávesnice
         BPL FETCH     ;Když je D7=0, vrať se na FETCH
         AND #$7f      ;Maskuj nejvyšší bit, zbylé obsahují ASCII
                       ;kód (v reg.A)
         BIT CLEAR     ;Překlop D7 do log.0
```

Alternativní metodou je vstup řízený přerušením - mikroprocesor je přerušován klávesnicí po každém jejím stisku. **Obrázek 3** ukazuje obvod z **obrázku 2a** upravený pro přerušením řízený vstup. Jedinou změnou je přidání invertoru s otevřeným kolektorem - IC5b - který řídí přerušení mikroprocesoru. Vstup invertoru je řízen výstupem Q IC7a. Invertor a IC5a mohou být ve stejném pouzdře - v tom případě by měl být na výstupu IC5a připojen rezistor v hodnotě 2 až 3 kiloohmy. Pro jednoduchost budeme předpokládat, že mikroprocesor má jeden vstup pro přerušení, které je aktivo-



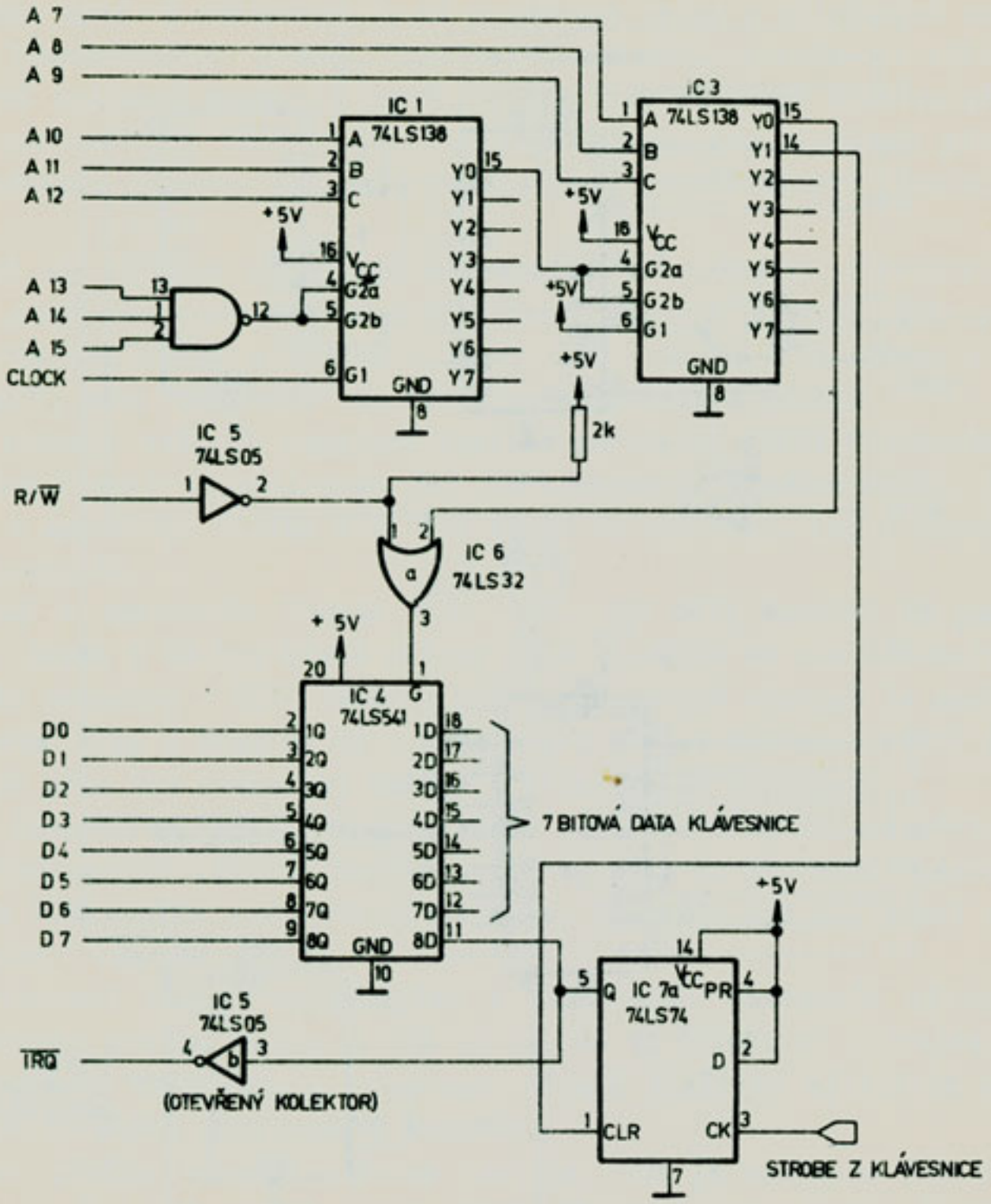
Obr. 2 b - Klávesnicový vstup s osmičkovým střádačem 74ALS574

váno úrovní log.0. Výklad o tom, jak různé mikroprocesory a operační systémy pracují s přerušeními, by zaplnil velkou knihu.

Všimněte si, že bit D7 je stále ovlivňován IC7a. To je proto, aby mikroprocesor mohl zjistit, zda zdrojem přerušení je klávesnice.

PARALELNÍ INTERFEJS TISKÁRNY

Standard Centronics většinou vyžaduje, aby impuls požadavku tiskárny o vyslání dat byl stabilní po dobu minimálně 1 mikrosekundy. Poté je vyslán z počítače min. 1 mikrosekundu široký strobovací signál DATA (log.0). Výstupní data musejí opět zůstat stabilní po dobu alespoň i mikrosekundy poté, co náběžná hrana strobovacího impulsu DATA dosáhne úrovně log.1.

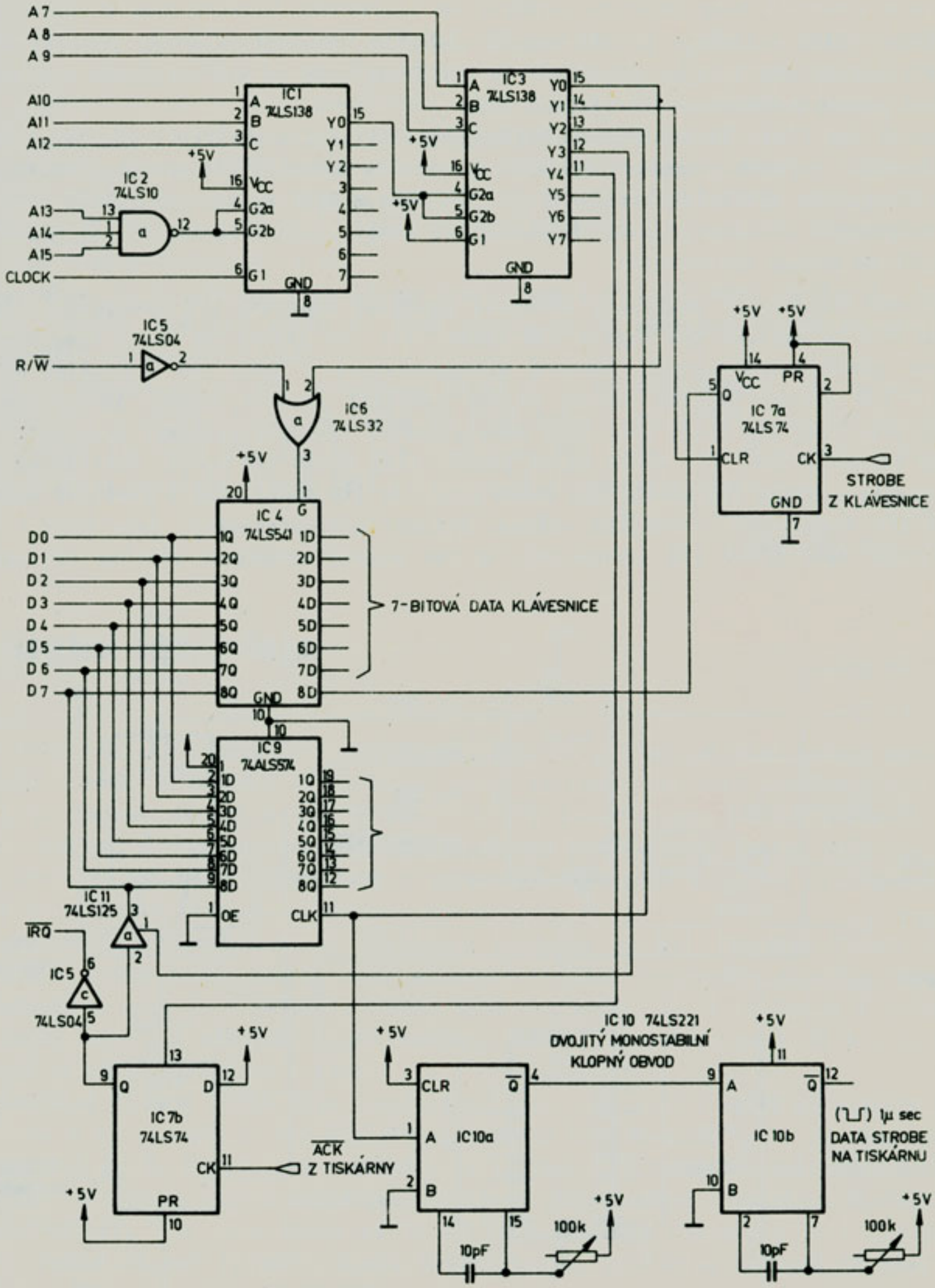


Obr. 3 . Klávesnicový vstup řízený přerušením (s Bufferem 74LS541)

Když je tiskárna připravena pro příjem dat, posílá počítači mikrosekundový strobovací signál ACKnowledge (uvědomovací) na úrovni log.0. Ačkoli pro paralelní komunikaci počítače s tiskárnou jsou předepsány ještě další signály (BUSY, DEMAND, ON-LINE, atd.), užívají se málokdy. Firma Centronics vyvinula svůj interfejs v dobách, kdy ještě neexistovaly mikropočítače, jak je známe dnes. Původní interfejsy se připojovaly k velkým počítačům, které se bez těchto přídavných signálů neobešly. Dnešní mikropočítače většinu těchto signálů pro úspěšný přenos dat na své tiskárny nepotřebují.

Obrázek 4 je vlastně jen malým rozšířením zapojení z **obr.2 a 3.** Náběžná hrana výstupu Y2 dekodéru IC3 časuje IC9 a spouští IC10a - ten generuje výstupní puls 1 mikrosekundu široký. Sestupná hrana výstupu IC10a spouští IC10b, který vysílá na tiskárnu strobovací signál DATA (log.0).

Pro případ potřeby použití portu tiskárny jako zdroje aktivního přerušení, je IC7b časován signálem ACK tiskárny. Ten aktivuje IC5c, který generuje přerušení,



Obr. 4. Klávesnice a tiskárna se střídačem 74ALS574

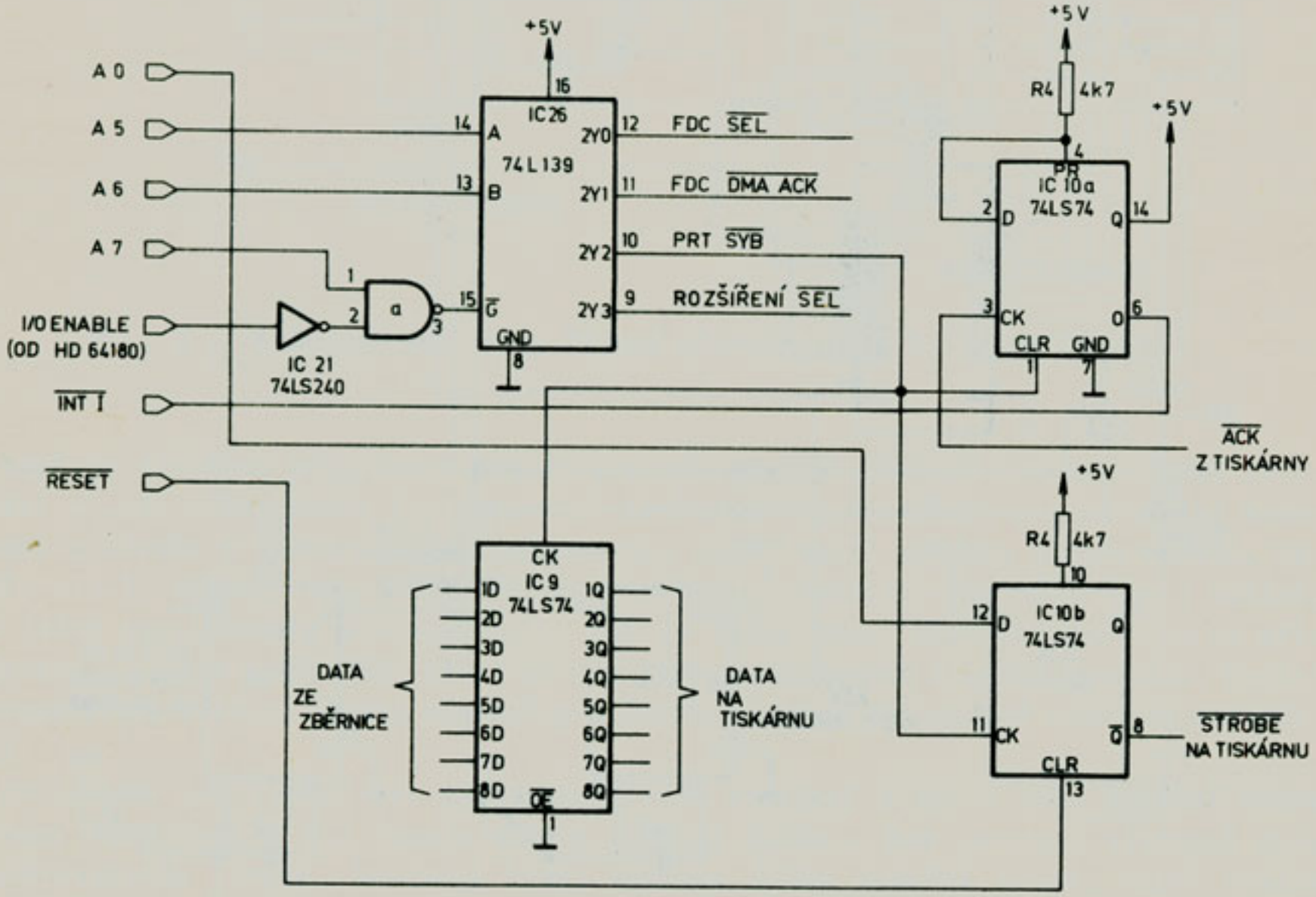
signalizující, že tiskárna je připravena pro příjem dalších dat. IC7b řídí rovněž vstup třístavového bufferu - IC11a - aktivovaného jedním z dekódovaných výstupů IC3.

Výstup IC11a ovlivňuje bit D7 datové sběrnice. Mikroprocesor si může na adrese IC11a zjistit, zda zdrojem přerušení je tiskárna. IC7b je pak vynulován a přerušení zrušeno jedním z dekódovaných výstupů IC3.

Obecně lze říci, že systémy s větším počtem aktivních periférií používají řízení přerušeními, jejichž prostřednictvím se periférie dožadují své obsluhy. Pro systémy s menším počtem aktivních periférií je někdy výhodnější způsob cyklického dotazování. To platí i v našem případě. Proto bychom mohli zcela vynechat řízení přerušením a přejít na způsob dotazování.

Popsané interfejsy používají adresové paměťově mapované I/O dekódování. (Každý výstup IC3 je aktivní pro rozsah 128 adres. Ale jen zřídka je nutné kompletně dekódovat všech 8x128 adres). Některé mikroprocesory - jako Z80 a novější HD64180 - mají I/O porty, jejichž adresy nejsou součástí rozsahu paměti adresovatelného mikroprocesorem, ale tvoří speciální adresovatelný "vstupně-výstupní prostor".

Z80 adresuje 64K, tj. 65536 bajtů paměti plus 65536 I/O portů. HD64180 adresuje 512K paměti a 65536 I/O portů. Oba mohou používat jak porty, tak i paměťově mapovaná místa pro komunikaci s vnějším světem. Ostatní mikroprocesory, jako 6502 a 6809, postrádají specifické I/O adresování, a tak musejí vykonávat všechny I/O operace



Obr. 5. Port Tiskárny SB180

pomocí paměťově mapy v rámci svého 64K adresovatelného rozsahu.

Na závěr a pro celkové dokreslení si ukážeme výstupní port tiskárny v systému SB180 (obr. 5). Používá oktalový střadač IC9 pro datový výstup a dva "D" střadače (z jednoho pouzdra 74LS74) pro koordinování přenosu dat (handshaking). Je ukázkou užití výstupního portu místo paměťového mapování. Rozdíl v zapojení mezi ním a interfejsem na obr. 4 vyžaduje i trochu odlišné softwarové řízení.

Strobovací impuls ENABLE mikroprocesoru HD64180 spouští invertor IC21. Výstup invertoru řídí jeden vstup hradla NAND - IC22. Další vstup IC22 je připojen k adresové lince A7. Výstup IC22 řídí signál (log.0) ENABLE dekodéru IC26. Adresové vstupy IC26 jsou řízeny linkami A5 a A6.

Výsledkem je to, že když je A7 na úrovni log.1 a I/O STROBE je log.0, dekodér rozdělí horních 128 I/O adres do čtyř skupin po 32 adresách. Každá ze skupin má svůj vlastní strobovací signál.

Aktivní strobovací signál pro adresy CD až DF je PRINTER STROBE. Ten nuluje IC10a a časuje IC9 a IC10b. Adresová linka A0 řídí datový vstup IC10 a určuje úroveň signálu STROBE (aktivní při log.0) pro tiskárnu. ACK (akt. log.0) signál z tiskárny časuje IC10a a generuje signál INTI (akt.log.0) do mikroprocesoru.

BYTE 7/86

Přeložil Rad. PLETKA

Paralelní interfejs

1. část

Na toto téma se publikuje řada článků. Většina z nich je však určena pouze zasvěceným - autor u čtenářů předpokládá aspoň základní znalosti problematiky. Tento seriál se naopak obrací k těm z vás, kteří o interfacingu takové znalosti zatím nemáte. Takže začneme od základů (i když sem tam budete muset leccos vzít "za hotové"). Znalosti si můžete doplňovat z materiálů Mikrobází souběžně otiskovaných na stejné či příbuzné téma.

Elementární informační prvek číslicové techniky je bit (zkratka BInary digiT) - označuje jednu buňku paměti nebo elementární část dvojkového signálu. Sám může mít hodnotu 0 nebo 1. Z několika těchto bitů (zpravidla 8) je složen BYTE (slabika, dále česky bajt). Z několika bajtů vznikají slova, zpravidla ze 2 až 4 - ty potom obsahují 16 až 32 bitů (ale i více).

Jednotlivé bity v bajtu jsou označovány indexy n=0....7 a každý má hodnotu 2**n, bit číslo 0 má hodnotu 1, bit číslo 7 má hodnotu 128. Hodnota bajtu je dána součtem hodnot všech bitů, které jsou nastaveny na 1. Např.: 01100101B = 0+64+32+0+0+4+0+1= 101. Tato hodnota se nazývá dekadická. Běžně se však používá i soustava šestnáctková (hexadecimální), kde jsou čísla od 10 do 15 označována písmeny A až F (výjimečně jinak). Pokud chceme vyjádřit číslo šestnáctkově, přiřadíme hodnotě 10 znak "A", hodnotě 11 "B",.....,hodnotě 15 "F".

Např.:	0011B	odpovídá	3 HEXA
	0101B		5 HEXA
	1100B		C HEXA

Takto se převádějí čísla 0..15. Hodnoty větší než 15 se převádějí analogicky. Např.: 11000101B (1100 0101B) odpovídá C5H. Hexadecimální čísla budu dále značit písmenem H na konci, čísla binární B na konci (pokud to bude nutné, písmenem D čísla dekadická).

Tím jsme si popsali, jak je číslo v počítači zaznamenáno. Pokud na číslicovém zařízení (např. počítači) požadujeme nějakou činnost, musíme informace do něj vpustit (vložit) nebo je získat na výstupu (odebrat). Hlavními částmi počítače jsou procesor, paměti ROM (Read Only Memories), RAM (Random Access Memories) a porty pro styk s vnějším zařízením. Tím může být třeba klávesnice, disk, další pamětí, tiskárna atd. A právě komunikace s vnější periférií je tématem tohoto článku.

Přenos dat může probíhat různými způsoby. Přenos světelnými paprsky, magnetickým polem, zvukovým vlněním, ale také po klasických vodičích. Dva stavy jsou zde reprezentovány napětím (např. když napětí je, znamená to log 1; když není, pak jde o log 0) nebo frekvencí (vyšší tón = log 0, nižší tón = log 1). My se budeme zabývat prvním případem, a to změnou napětí na vodičích. Hodnota pro jednotlivé logické stavy je přesně daná:

pro TTL vstup:	log 0.....	U<= 0.8 V
	log 1.....	U>= 2.0 V
pro TTL výstup:	log 0.....	U<= 0.4 V
	log 1.....	U>= 2.4 V

Pro hodnoty napětí mezi log 0 a log 1 není logické stav definován.

Přenos dat můžeme rozdělit i podle jiného kritéria. Nebude to podle druhu přenosové cesty, ale podle vlastního obsahu dat na přenosové cestě. Je to přenos sériový a přenos paralelní. Oba tyto způsoby mají společnou jednu věc: každý program (nebo také data) je složen z jednotlivých bajtů, a po těchto bajtech se také vysílá. U sériového přenosu se vezme daný bajt určený k přenosu a jeho postupně odebírané bity se vysílají. Směr odebírání není vždy stejný. Tento sled bitů je často doprovázen tzv. start-impulsem a stop-impulsem nebo i paritním bitem (logický stav a délka trvání také nejsou jednotné). Někdy je přenos doprovázen dalšími řídícími vodiči.

Vzhledem k tomu, že pro vyslání jednoho bajtu potřebujeme min. 8 taktů (pro každý bit, start, stop a paritu), je ve srovnání s paralelním přenosem pomalejší. Důležitá je i otázka synchronizace činností vstupu a výstupu. Ale přesto má proti paralelnímu přenosu jednu obrovskou výhodu. Místo osmi datových vodičů mu stačí pouze jeden. Z toho vyplývá, že se používá hlavně pro přenos dat na velké vzdálenosti nebo když si nemůžeme z nějakého důvodu dovolit připojit 8 vodičů. My se budeme zabývat přenosem paralelním.

Jak už bylo řečeno, při paralelním přenosu jsou data přenášena po osmi datových vodičích (budeme je označovat DATA a jednotlivé vodiče DATA0...DATA7). Na jedné straně vodičů je vysílač, na druhé přijímač. Když chceme přenést bajt z vysílače do přijímače, vysílačem náš bajt vyšleme a na druhém konci vodičů si je přijímač přečte. Ale co se stane, když data na vstupu odpojím nebo změním? Přijímač bude vždy číst pro něj platnou hodnotu (ta je v těchto případech nedefinována). Uděláme tedy nepatrnou úpravu našeho zapojení - k osmi vodičům DATA připojíme další (výstupní) - jeho logický stav indikuje:

log 0 - na vodičích DATA jsou platná data
log 1 - na vodičích DATA jsou neplatná data

Tento vodič se značí různě, DAV, jako zkratka slov DAta Valid (platná data). Vzhledem k tomu, že nás pouze informuje o stavu vodičů DATA, bude patřit do skupiny řídících vodičů.

Tím se situace změnila a pro vyslání jednoho bajtu z vysílače do přijímače nestačí pouze připojit data, ale musíme:

a) na vodiče DATA připojit z vysílače vysílaný bajt
b) na vodič DAV připojit log 0.

Před vlastním přenosem (při inicializaci) musí být vodič DAV nastaven na hodnotu log 1. Po připojení DAV na log 0 začne přijímač přijímat data. Po odpojení dat z vodičů DATA se na výstupu objeví něco nedefinovaného. Bylo by tedy vhodné k bodu a) a b) připojit další:

c) na vodič DAV připojit log 1

Co se ale stane, když přijímač nestačí data zpracovat mezi body b) a c)? Zcela jistě by vznikla kolize. Jednou z možností, jak tomu předejít, je mezi body b) a c) počkat. Ale jak zjistit dobu čekání? Pokud dochází u tiskárny k posunu válce, trvá to někdy dokonce i pár sekund. To by znamenalo čekat po vyslání každého bajtu po tuto dobu. Tak by náš přenos dat nebyl právě nejrychlejší. Proto zvolíme jinou možnost - připojíme další vodič. Ten nás bude informovat o tom, zda přijímač data už převzal. Budeme jej označovat NDAC (No Data ACcepted - data nebyla převzata) - je opět aktivní v log 0. Bude také patřit do skupiny řídících vodičů, ale na rozdíl od vodiče DAV bude vstupní (tj. informaci podává přijímač vysílači). Přenos jednoho bajtu proběhne v tomto sledu:

a) na vodič DATA připoj vysílaný bajt
b) na vodič DAV připoj log 0 (data platná)
c) čekej, dokud je na NDAC log 0 (data nepřijata)
d) na vodič DAV připoj log 1

Nyní byl bajt přijat do přijímače. Může ale dojít k takovéto situaci: podle pravidel a) až d) vyšlu bajt do přijímače a jdu znovu na bod a). Pokud ale přijímač ještě předchozí bajt nezpracoval, pokouším se signálem DAV do něj znovu zapisovat - tak ale může dojít ke kolizi. Teorii s vyčkáváním jsme už jednou zavrhli a ani teď ji nepoužijeme. Připojíme další vodič, který bude opět vstupní a bude nás informovat o připravenosti přijímače. Budeme jej označovat NRFD (Not Ready For Data= = nepřipraven pro data) - i ten bude aktivní v log 0.

Tak jsme vytvořili vše potřebné - přenos jednoho bajtu z vysílače do přijímače bude vypadat takto:

a) čekej, dokud NRFD = log 0; jakmile NRDF = log 1, skok na b)
b) na vodič DATA přípoj vysílaný bajt
c) na vodič DAV připoj log 0 (data platná)
d) čekej, dokud NDAC = log 0
e) na vodič DAV připoj log 1 (data neplatná)

To je celý algoritmus přenosu dat (resp. jednoho bajtu) od vysílače k přijímači. Nyní jej můžeme označit za bezkolizní. U přijímače se odehrává něco podobného (opět předpokládáme, že byly vodiče NRFD a NDAC nastaveny na log 0):

a) na vodič NRFD připoj log 1 (jsem připraven)
b) čekej, dokud DAV = log 1; jakmile DAV = log 0, skok na c)
c) přijmi data z vodičů DAT
d) na NRFD připoj log 0
e) na NDAC připoj log 1 (data převzata)
f) čekej, dokud DAV = log 0; při log 1 skok na f)
g) na NDAC připoj log 0

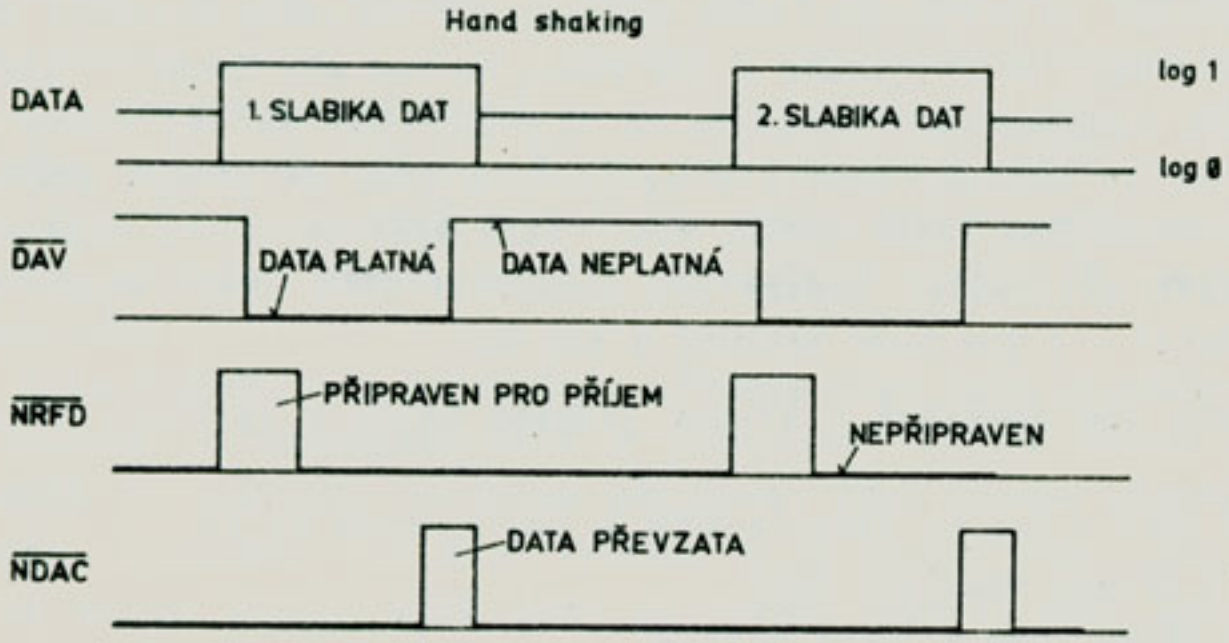
Tento přenos dat se nazývá "handshaking" (potřásání rukou); viz **obr. 1.** Vodiče pro řízení bývají označovány různě; DAV, NDAC a NRFD jsou použity u sběrnice HP--IB (IMS-2).

Dnešní mikropočítače mají společnou datovou sběrnici, ke které je připojen mikroprocesor, dále paměti typu RAM i ROM a vstupně-výstupní porty. Data přenášená po datové sběrnici jsou doprovázena řídícími signály, které informují o činnosti procesoru. V další části budu vše demonstrovat na mikroprocesoru Zilog Z80. Pro připojování vstupních a výstupních portů musíme znát význam těchto vodičů:

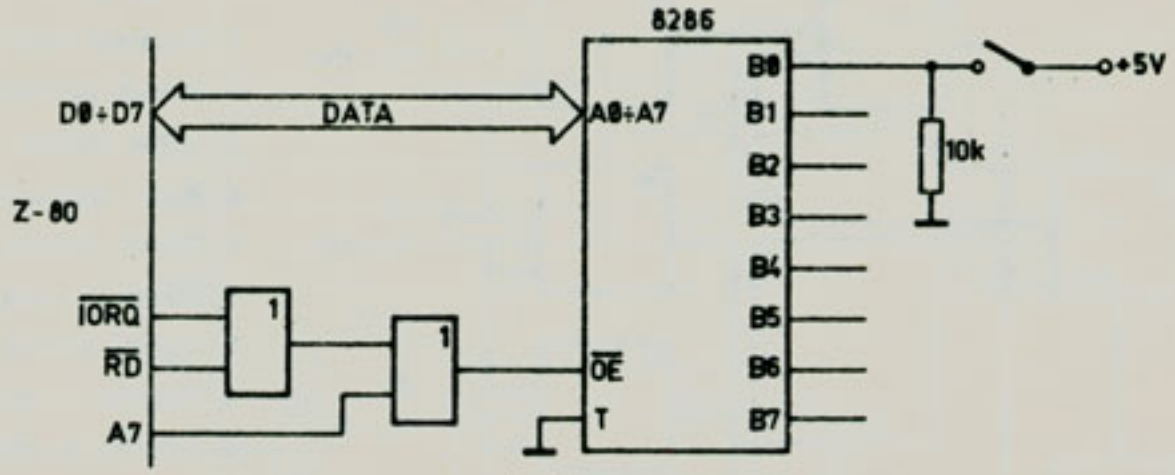
D0-D7 datová sběrnice
A0-A15 adresová sběrnice
a vodičů řídících:

RD Read-čtení z portu nebo z paměti
WR Write-zápis do portu nebo do paměti
IORQ Input-Output Requesl-žádost o I/O port
MREQ Memory Request-žádost o paměti
M1 Machine Cycle One - čtení instrukce
(Všechny řídící signály jsou aktivní v log 0!)

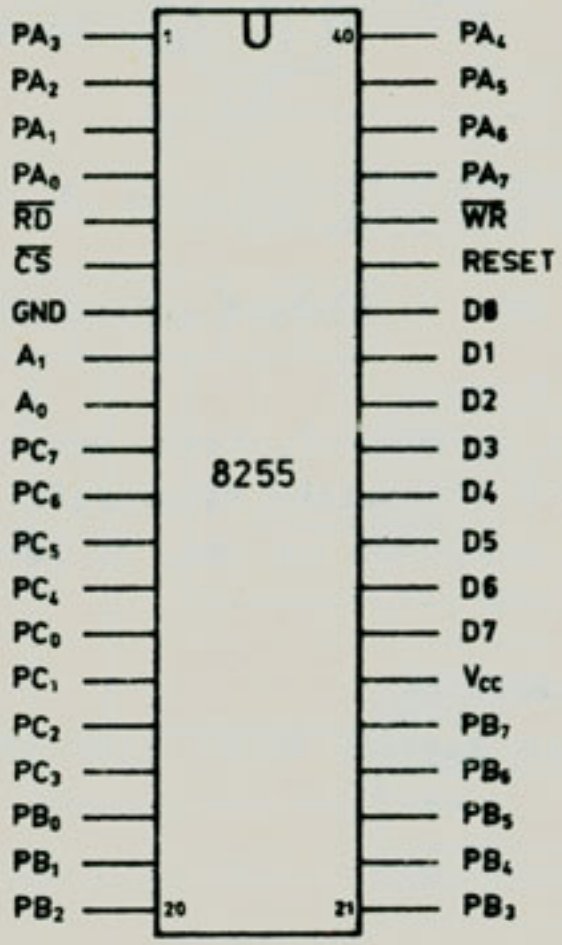
Pokud jsou na jedné sběrnici připojeny paměti, procesor, porty a další zařízení,



Obr. 1. Handshaking



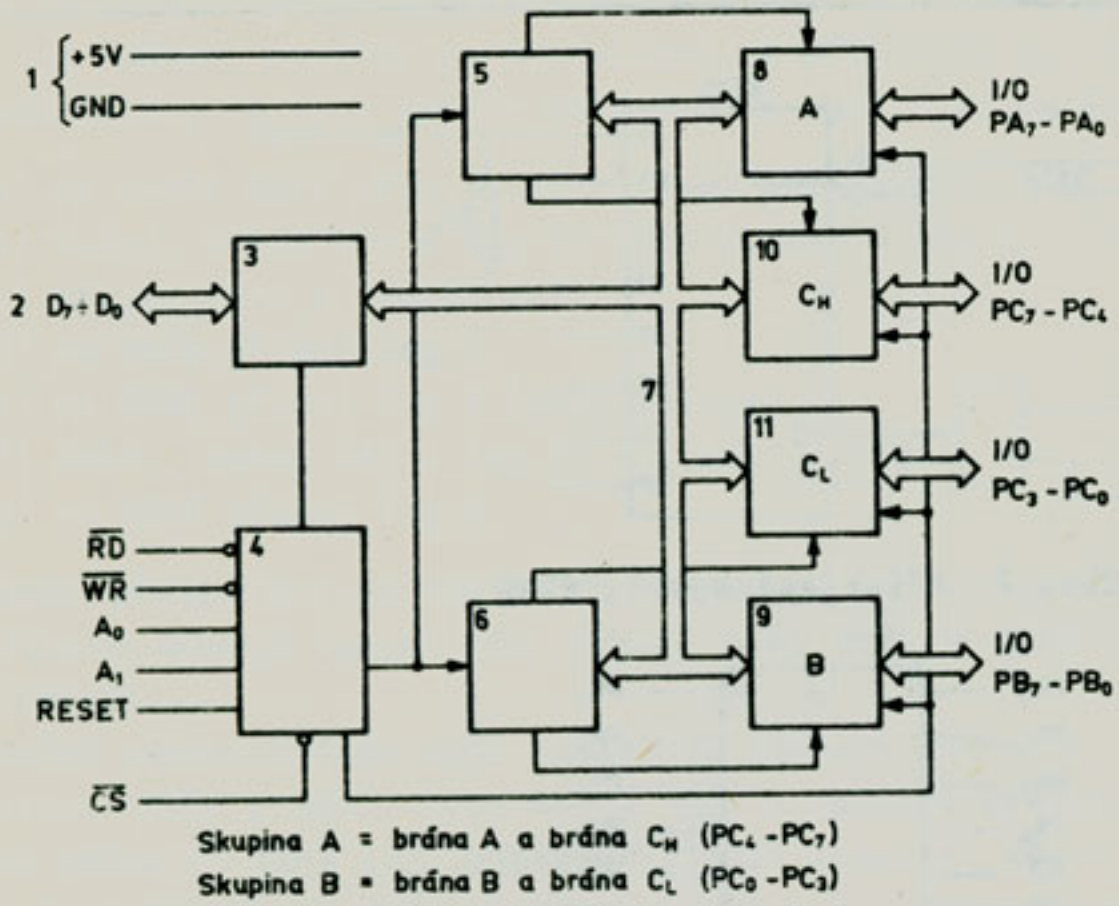
Obr. 2. Připojení budiče 8286



Obr. 3. Zapojení pouzdra 8255

mohly by se na této sběrnici jejich úrovně "poprat". Mikroprocesor funguje jako vůdčí element - na základě jeho signálů jsou generovány povely pro činnost ostatních zařízení. Každé zařízení obsahuje alespoň jeden vodič povolující připojení. V jeho označení se zpravidla vyskytuje písmeno S (Select - vybrat) nebo E (Enable - umožnit). Pokud na těchto vstupech není aktivní hodnota, chová se zařízení jako nepřipojené (výstupy jsou v tzv. třetím stavu). Když má zařízení takových vodičů více, musí být zpravidla na všech aktivní hodnota (většinou log 0).

Pro demonstraci uvádím obvod MHB 8286, což je obousměrný budič. Vstupem T rozhodujeme, zda bude obvod fungovat jako vstupní nebo výstupní; vstupem OE jej připojujeme na sběrnici. Jako nejjednodušší příklad si můžeme ukázat připojení vstupního portu ke sběrnici počítače ZX-Spectrum (k němu můžeme připojit třeba joystick). Schéma je na obr. 2; včetně generování signálu OE. Ten bude mít hodnotu log 0, pokud bude logická 0 na lince RD, IORQ a A7 (joystick Kempston). Tak jsme vytvořili jeden vstupní port. Pokud potřebujeme portů více, je vhodné použít obvod 8255 vyvinutý firmou Intel (Tesla MHB 8255). Obsahuje 24 vstupních a výstupních linek (brány A,B



1. Napájecí napětí
2. Datová sběrnice
3. Buffer datové sběrnice
4. Řídící logika pro čtení a zápis
5. Řízení skupiny A
6. Řízení skupiny B
7. Vnitřní datová sběrnice
8. Skupina A, obvod A
9. Skupina B, obvod B
10. Skupina A, obvod C_H
11. Skupina B, obvod C_L

Obr. 4. Blokové schema 8255

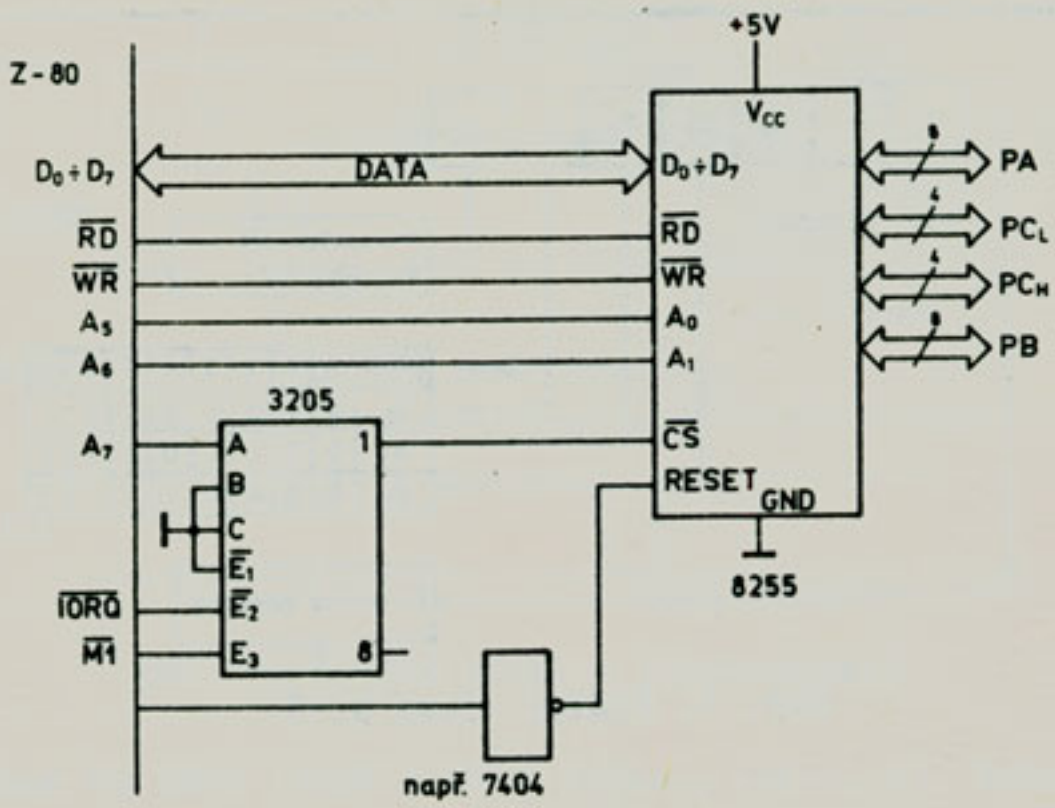
a C), směr přenosu lze řídit softwarově, navíc obsahuje řídicí vodiče pro handshaking (ve srovnání s 8286 je i ekonomicky výhodnější). Jeho připojení budu opět demonstrovat na Z80. Obvod je znázorněn na obr. 3, jeho vývody mají následující funkce:

D0-D7 obousměrná datová sběrnice
RESET nulování vstupu (po signálu RESET jsou brány A, B i C nastaveny jako vstupní)
CS aktivace obvodu
RD čtení
WR zápis
A0,A1 adresa bran nebo CWR
PA brána A
PB brána B
PC brána C

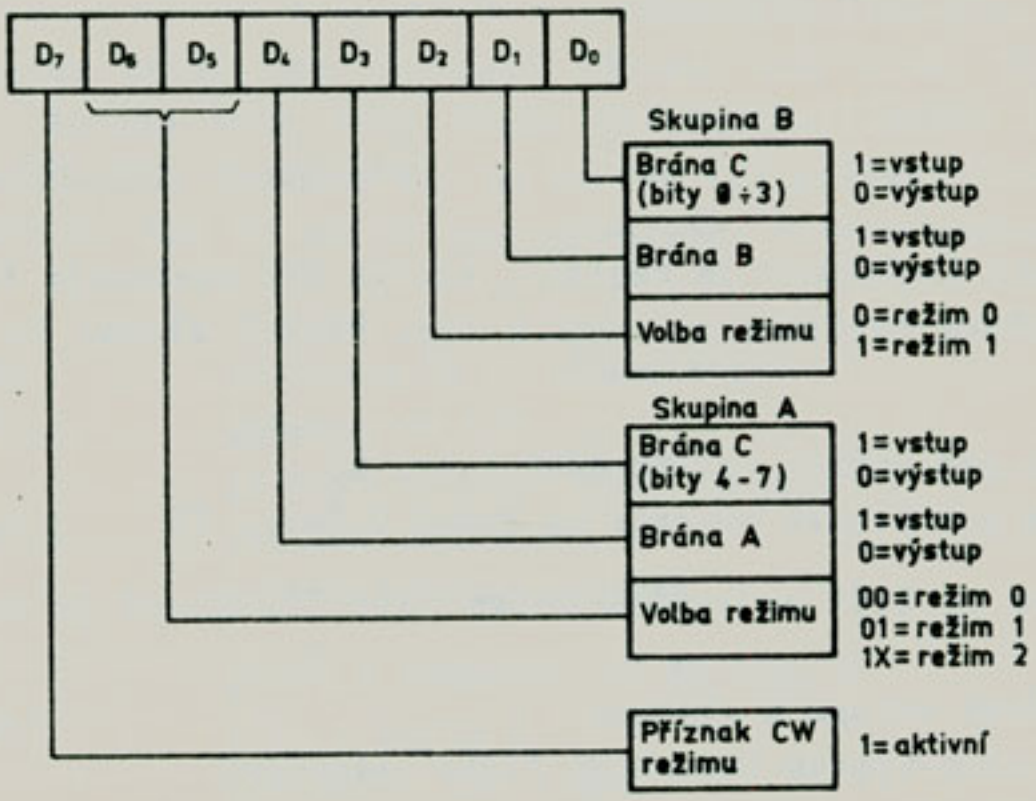
Blokové schéma 8255 je na obr. 4, připojení obvodu k mikroprocesoru Z80 na obr. 5. Vodiče RD, WR propojíme. Signál RESET musíme pro 8255 invertovat např. pomocí obvodu 74LS00 nebo 74LS04 (obvod musí být typu LS, aby nedocházelo k velkému zatížení sběrnice). Datové vodiče spojíme a ze signálů M1, IORQ a A7 (linka A7 je zvolená a je pro aktivaci 8255 nejpoužívanější) generujeme signál CS (chip select) pro aktivaci obvodu. Na vstupy A0, A1 připojíme třeba linky A5, A6 Z80 (zvoleno).

Nejprve musíme vygenerovat signál CS. To lze učinit pomocí obvodu 74LS00 (viz AR), nebo pomocí dekodéru 1 z 8 (MH-3205, viz obr. 5). Tak získáme aktivovací signál "I/O aktivní při A7=0".

Zde je důležitá linka M1. Procesor Z80 při požadavku přerušení generuje signály IORQ a M1; na adresové sběrnici se objeví čítač adres, zatímco při vykonávání in-



Obr. 5. Připojení Z-80 k 8255



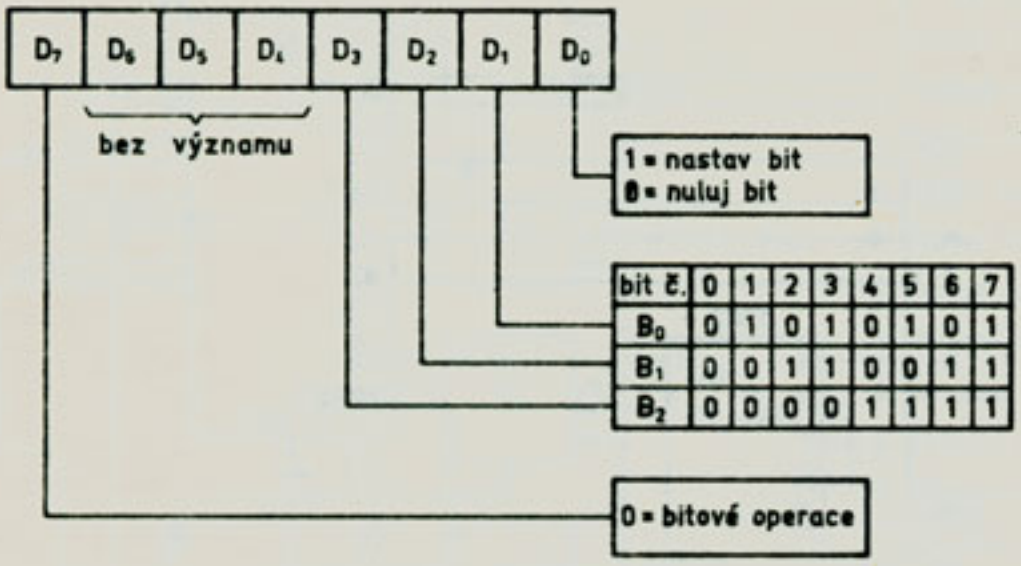
Obr. 6. Řídící slovo režimu

strukce pro vstup a výstup není M1 v aktivním stavu. Pokud by se na adresovém vodiči A7 objevila log 0, mohl by být generován signál CS (z obvodu 3205 i při přerušení. Obvod 8255 vyžaduje kromě CS ještě alespoň signál RD nebo WR, takže nebezpečí nehrozí. Ale jsou obvody, které se na základě tohoto signálu aktivují. Já se ale přikloním k 3205, protože je po určité úpravě zapojení schopen generovat signály pro více zařízení (např. více obvodů 8255).

Z teoretického hlediska zde ale vzniká jiná kolize: u obvodu 8255 je pro komunikaci s procesorem uváděn tento sled signálů:

a) na vodičích A0, A1 se objeví adresa brány a CS přejde do aktivního stavu,
b) po čase $\geq$=0 ns přichází signál RD nebo WR

Z firemní dokumentace Z80 ale vyplývá, že může přijít RD, WR dříve než IORQ (tudíž dříve než CS). Přesto ale toto zapojení s obvodem 8255 funguje.



Obr. 7. Řídící slovo bitů

Pokud budeme chtít s obvodem komunikovat, musí být adresový vodič A7 Z80 v log 0 (->gen. CS) a na A5, A6 adresa brány, kombinace A5A6 odpovídají:

00 brána A
01 brána B
10 brána C
11 CWR.

CWR (Control Word Register) je registr, kterým CPU nastavuje funkční konfiguraci každé brány. Do tohoto registru se posílá tzv. řídící slovo. To může mít dva formáty:

a) řídící slovo pro určení režimu 8255 (obr. 6)
b) řídící slovo pro bitové operace na bráně C (obr. 7)

Obě řídící slova se liší nejvyšším bitem (bit 7). Obvod je schopen pracovat ve třech režimech:

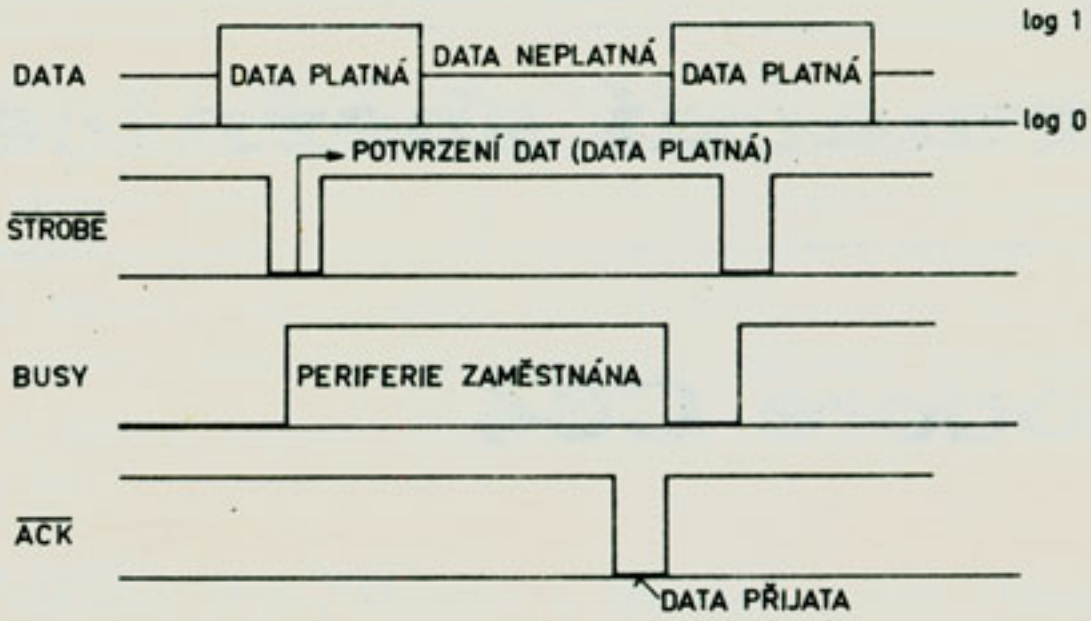
a) základní způsob vstupu a výstupu
b) strobovaný vstup-výstup (obsahuje řídící signály)
c) obousměrná sběrnice

Nyní k vlastnímu programování 8255. Předpokládejme, že potřebujeme připojit tiskárnu (samozřejmě musí být vybavena paralelním vstupem). Obvod 8255 použijeme v režimu 0. Režim 0 je definován takto: dvě I/O osmibitové brány (A,B), dvě I/O čtyřbitové brány (spodní polovina C=CL, horní polovina C=CH), výstupy jsou s pamětí, vstupy bez paměti.

Při připojování tiskárny se budeme držet zásady použití co možná nejmenšího počtu bran. Tiskárna s paralelním vstupem (bývá nazýván Centronics; viz obr. 8) má vodiče označeny:

DATA	datový vstup
STROBE	potvrzení platnosti dat na datovém vstupu
BUSY	tiskárna zaměstnána (nemohu přivést další data)
ACK	potvrzení příjmu

(signály STROBE, ACK jsou aktivní v log 0, BUSY v log 1)



Obr. 8. Časové průběhy (Centronics)

Bránu A přivedeme na DATA (bude tvořit datovou sběrnici). Bránu C naprogramujeme tak, aby byla spodní polovina vstupní a horní polovina výstupní. Bránu B necháme třeba ve vstupu. Na bit 3 brány C připojíme vodič BUSY, na bit 4 STROBE. V tabulce na obr. 6 najdeme řídící slovo: 1 00 0 0 0 1 1=10000011B=131D=83H.

Tuto hodnotu musíme poslat na výstupní adresu, kde A7=0, A5=A6=1, zbylé bity budou rovny log 1. Adresa bude tedy číslo 01111111B=127D=7FH (analogicky adresa brány A=31D=1FH, B=63D=3FH, C=95D=5FH). Na adresu 127 pošleme číslo 131 instrukcemi LD A,131 a OUT (127),A. Tak je obvod naprogramován. Přenos jednoho bajtu bude vypadat následovně:

```
INIT   LD   A,131        ;hodnota CW do akumulátoru
       OUT  (127),A      ;přenos do CWR
       LD   A,9          ;bitová operace (PC4 nastavíme)
       OUT  (127),A      ;STROBE neaktivní
       RET               ;obvod inicializován

BYTE   LD   C,A          ;bajt z akumulátoru do reg. C
B2     IN   A,(95)       ;čtu z portu C
       BIT  3,A          ;je BUSY? log 1=ano
       JR   NZ,B2        ;musím čekat
       LD   A,C          ;bajt zpět do akumulátoru
       OUT  (31),A       ;jeho vyslání na bránu A
       LD   A,B          ;bitová operace (PC4 vynulujeme)
       OUT  (127),A      ;STROBE=log 0 (data platná)
       LD   A,9          ;bitová operace (PC4 nastavíme)
       OUT  (127),A      ;STROBE=log 1 (konec impulsu)
       RET               ;bajt byl přenesen
```

Rutinou INIT nastavíme režim 8255 a vodič STROBE do neaktivního stavu. Rutina BYTE nejdříve čeká na připravenost tiskárny. Pak připojí na bránu A data, nuluje bit č.4 v bráně C (STROBE je aktivní), vzápětí tento bit nastavuje (STROBE je neaktivní). Při tomto impulsu tiskárna přijme data, okamžitě přivádí na BUSY log 1 (aktivní) a začíná data zpracovávat.

Richard LUKEŠ

Informační sběrnice

Commodore C64

Tento počítač patří mezi nejznámější a nejrozšířenější počítače na světě. Patří mezi ně pro své dobré vlastnosti, snadnou rozšiřitelnost, širokou rodinu periférií a dobré programové vybavení. A to vše i přes nyní již nedokonalý mikroprocesor 6510 z velké rodiny Motorola 65.... Pokud se tedy uživatel bude chtít orientovat na programování ve strojovém kódu, musí zvládnout tento procesor strukturálně odlišný od typu Z80. Není to nutné pro kompilátory, které C64 nabízí z řady programovacích jazyků.

C64 je počítač určený pro mnoho druhů a oblastí použití, a proto bylo zvoleno robustní mechanické provedení s velmi dobrou klávesnicí, která obsahuje standardní QWERTY klávesnici s doplňkovými funkčními klávesami a klávesami pro editaci obrazovky - C64 má celoobrazovkový editor. Konektory určené pro propojení počítače s periferiemi jsou umístěny na pravém boku počítače - zdroj a joystiky, a na zadní stěně - port uživatele, rozšíření paměti, magnetofon, sériový port, video a audio, RF TV signál. Bohužel je použita řada konektorů, které nejsou běžně dostupné. Zdroj C64 je umístěn mimo počítač na dlouhém přívodním kabelu. Je bohužel zapotřebí speciálního magnetofonu ovládaného i napájeného počítačem. Disketová jednotka na 5.25" (palcové) diskety se připojuje do sériového portu a sama umožňuje napojení další jednotky. Kapacita disket je 170K a je možno použít diskety dvojstranné. Základní paměť C64 je sice 64K, ale ty nejsou nikdy zcela přístupné uživateli. Basic, základní programové vybavení C64, nabízí zhruba 38K volné paměti, je ovšem nutno upozornit na fakt, že tento Basic je poměrně "hloupý". Neobsahuje ani jemnou grafiku, kterou C64 nabízí v 16ti barvách na rastru 320x200 bodů, ani příkazy pro sprity, kterých má C64 osm o maximální velikosti 24x21 bodů. Proto se pro aplikace s nutností využití těchto prvků používají některé z dalších Basiců na C64. Nejznámější - a snad i nejlepší z nich - je Simon´s Basic, který nabízí 30K volné paměti a řadu příkazů pro jemnou grafiku, (PLOT, LINE, BLOCK, DRAW, ROT, CIRCLE), hudbu (VOL, SOUND, PLAY), sprity, ale i procedury, uživatelské funkce a různá další vylepšení. Z dalších Basiců jmenujme: ULTRA Basic a EXBASIC-LEVEL-II. Nelze opomenout vynikající graficky zaměřený kompilátor G-PASCAL s řadou příkazů pro grafiku, sprity a hudbu. Uživateli se nabízí nepřeberné množství her, na kazetě a na disku (nebo pouze na disku). Tyto celodiskové hry dosahují neuvěřitelné kvality, olympijskými hrami se všemi disciplinami v komfortní animované grafice počínaje a složitými kombinačními hrami zabírajícími jeden až dva celé disky konče. Řada her je známá i z jiných počítačů: BLUE MAX, AIRWOLF, FLIGHT SIMULATOR (existuje několik verzí, ta nejdokonalejší zabírá celý disk a umožňuje nevídané věci), ARC OF YESOD, YESOD OF NODES, SPY vs SPY I i II, SPY HUNTER, BASE BALL, WORLD CUP, IMPOSSIBLE MISSION.

Nechme ale výčtu her a zaměřme se na užitkové programy. Vynikající grafický editor KOALA PAINTER, zpracovaný metodou ikon a ovládaný pouze joystikem, umožňuje rychlé kreslení obrázků potřebných pro hry i uživatelské programy. Pro hudebníky je určen automatický bubeník RYTHM ROCKER. Z dalších jmenujeme evidenční program LAGES, adresáře a automaty na vytváření dopisů, programy pro testy počítače, disket a periférií, assemblery, monitory, disassemblery, vyučovací programy, textové procesory, databanky a další; záplava dalších užitkových programů se již staví téměř na úroveň záplavy her.

Výsledky testů na rychlost pro Basic:

Testy provedeny v Simons Basicu:

BM1	prostý cykl 0 - 1000	1.90
BM2	podmínkový cykl	10.64
BM3	podmínkový cykl výpočtů prom.	19.60
BM4	podmínkový cykl výpočtů konst.	21.39
BM5	BM4 + volání GOSUB	23.13
BM6	BM5 + prázdný cykl 1 až 5	34.10
BM7	BM6 a v cyklu ind. prom	53.50
BM8	aritmetika a funkce	117.23

Grafické testy

BMG1	PLOT útvaru 100 x 100	64.60
BMG2	DRAW útvaru 100 x 100	7.30
BMG3	16 kružnic od r = 0 do 80 krok 5	39.52
BMG4	vybarvení čtverce 100 x 100	14.84
BMG5	nepoužit	
BMG6	příkaz plného čtverce 100 x 100	2.30
BMG7	rychlost tisku řet. funkce	3.69
BMG8	rychlost tisku řet. konstanty	2.26

Pro srovnání uvádím BM1-8 od Commodore PLUS 4

BM1	BM2	BM3	BM4	BM5	BM6
1.4	9.4	17.9	18.5	20.9	34.2

BM7	BM8
54.6	100.6

tento výrobek firmy Commodore má CPU 7501 a je pokračováním počítače C64 s již novým Basicem.

Daniel Dočekal

Jaké bude dnes počasí

"Mám si vzít deštník nebo mini?"
"Na to Vám stačí mikro!"
...a přijímač UHF, patřičná anténa a může být i tiskárna. Proč? Protože i vy můžete mít doma malou zájmovou meteorologickou stanici. Družice sledující počasí rozvěšovat po obloze nemusíte, je jich tam dost - a všechny vysílají signály, jejichž podstatnou část tvoří kódované videosnímky naší planety. Jako třeba obrázky, které můžete vidět při televizní předpovědi počasí... tak nějak by mohla znít reklama nové oblasti zájmové činnosti, v níž se snoubí vhf technika s výpočetní technikou.

Je k tomu mimo vyjmenovanou aparaturu samozřejmě potřeba spousty vědomostí a umu. Autor článku "Spojení s vesmírem" z měsíčníku Computing Age 11/85 se příjmem signálů z družic zabývá po 12 let. Ve svém vyprávění sice nepodává žádný podrobný návod k realizaci své zájmové činnosti, ale přesto stojí za to si ho vyposlechnout. Ke své aparatuře má připojenu i počítačovou konfiguraci. Další slova už patří jemu:

První meteorologický snímek vyslaný z vesmíru na Zemi přišel před 25 lety ze satelitu TIROS. Tím momentem nastal v práci meteorologů tolik potřebný a očekávaný zlom. Do té doby se shromažďovaly informace z jednotlivých pozemních stanic, které však ve svém součtu nemohly poskytnout komplexní údaje pro vytvoření přesnější předpovědi. Nehledě už na složitost příjmu i zpracování informací a z toho plynoucí časové prodlevy... Dnešní satelity podávají meteorologům kontinuální informace o momentální situaci a jejím vývoji na kterémkoli místě na světě.

Před 12 lety jsem si postavil přijímač signálu meteo-družic...měl jsem k němu připojen stařičký osciloskop..byl to přímo pytel součástek! Dnes vám stačí mikropočítač (mám BBC) a nějakých 300 liber na celou aparaturu včetně přijímače fy Timestep Electronics, který umožní zachycení signálů pro jejich zobrazení na monitoru, případně i na tiskárně. Někteří lidé se snaží utrácet tisíce za různá super zařízení. Zapomínají však, že družice vysílají poměrně jednoduchý signál, pro jehož zpracování relativně dobře poslouží i to, co jsem vyjmenoval.

Satelity lze rozdělit do dvou skupin: orbitální a geostacionární. Ty první jsou "velmi nepokojné" - dráha jejich letu vůči Zemi je výslednicí pohybu po jejich vlastní dráze (u meteorologických obvykle od pólu k pólu) a otáčení Země "pod satelitem". Příkladem takové družice je americká NOAA9, kterou rovněž přijímám. Oblet Země jí trvá sice jen 102 minut, ale protože se během každého z nich Země pootočí asi o 25,5 stupně, objeví se nad vámi jen šestkrát denně - z toho se mi její signál podaří zachytit jen jednou či dvakrát. Vysílá současně dva snímky - jeden v oblasti viditelného světla, druhý infra. Oba jsou vedle sebe v jednom, na 120 TV řádkách. Každý řádek sestává z půlky pro viditelné a půlky pro infra světlo. Jedna obrazovka se naplní řádku po řádce až za jednu minutu - tedy velmi pomalu.

Protože je družice ve stálém, poměrně rychlém pohybu, je pro pevné zaměření kamery na "právě snímaný řádek" užito zařízení Scanning Radiometer, sestávající z rotujícího zrcadla a fotocitlivého snímače. Obraz nemá to, co si představujeme pod pojmem začátek, ale je vysílán jako "kontinuální proužek". FM signál je typu APT (Automatic Picture Transmi-

ssion), který používají i sovětské družice. Interfejs mezi přijímačem a počítačem převede signál z upravených 2,4 kHz (AM) konvertorem A/D na digitální tvar, v němž je zakódována úroveň jasu obrazových bodů. Příjem probíhá na 137,62 MHz.

Druhý typ satelitů - geostacionární - je z hlediska jejich zaměření mnohem výhodnější, protože "visí" nad rovníkem a pohybují se rychlostí i směrem shodnými s otáčením Země. Na rozdíl od svých neposedných kolegů, kterým nic neunikne, hledí kamery stacionárních družic na stále stejné "místo" - šířka záběru je samozřejmě značná - 4 takové družice stačí na pokrytí celé planety. Evropu má v záběru METEOSAT 2. Zde již obrázek má svůj začátek. Sestává z 800 řádek, z nichž jsou postupně vyslány vždy 4 za vteřinu. Zatímco z tohoto satelitu máte k dispozici obrázek stále, je jeho příjem mnohem složitější, potřebné zařízení dražší. V kombinaci Timestepu s BBC je obrázek z NOAA9 mnohem lepší. METEOSAT 2 vysílá na 1,6 GHz, proto je signál hned za anténou převeden na 137,62 MHz. Pro jeho dobré zachycení je zapotřebí parabolické antény v ceně 335 liber.

Interfejs fy Timestep obsahuje vnitřní hodiny 2 Hz, které jej synchronizují s přijímaným signálem a vyrovnávají úniky NOAA9, vznikající vlivem jeho pohybu (Dopplerův efekt). Tyto hodiny rovněž napomáhají orientaci začátku řádky a pulsu, který rozděluje obrázek spektra viditelného a infra světla. Tyto pulsy musí být přesně pod sebou, aby obraz byl lineární.

Protože BBC má obrazovou paměť složenou z 320 x 256 bodů, neukazuje plnou plochu vysílaného obrazu (něco se "vynechává"). Napomáhá tomu software téže firmy (37,50 liber), které m.j. umožňuje s obrazem všelijak pracovat. Jednou z pěkných funkcí programu je možnost skokového zvětšování libovolných míst zaznamenaného obrazu. U obrázku z NOAA9 může od sebe oddělit obě složky a promítnout buď jen půlku "viditelnou" nebo jen infra. Lze si rovněž hrát s vybarvováním obrazu. Program dále spolupracuje se záznamem, který byl pořízen třeba ve vaší nepřítomnosti (přijímač při zachycení silného signálu automaticky spustí magnetofon) - na jedné stopě je totiž signál družice, na druhé impulsy hodin interfejsu, zabraňující rozjetí obrázku při jeho načítání z pásku. Software zajišťuje i tisk celého nebo zvolené části obrazu.

Tolik autor. Článek je doprovázen ukázkami zachycených obrázků, jak byly vytisknuty na běžné maticové tiskárně. Je to pohled, který stojí za to.

Lze snad dodat, že "dálnopisně kódované" zprávy o počasí (samozřejmě bez obrázku) můžete zachytit na krátkovlnném přijímači a jednoduše je dekódovat počítačem, který vám zprávu vypíše. Ovšem k obrazu skoro půlky planety to má přece jen trošku daleko. K zachycení a zpracování signálů meteo-družic samozřejmě nemusíme mít zrovna přijímač nebo počítač uvedené značky. Bylo by zajímavé zjistit, zda u nás existuje podobně computerizovaný amatérský meteorolog - už proto, aby prostřednictvím Mikrobáze mohl předat své poznatky a zkušenosti nejen těm, jejichž ženy nevědí,co by si měly zrovna vzít na sebe. Ozvete se?

-elzet-

Obrazová paměť ZX Spectra

3. část

ORGANIZACE PAMĚTI ATRIBUTŮ

je mnohem jednodušší, než je tomu u paměti kresby. První místo paměti je nultý znakový čtverec nultého znakového řádku. Má adresu 22528. Následující adresa 22529 odpovídá 1.znakovému čtverci v 0.řádku atd.

Prostě - začátek je vlevo nahoře a jednotlivé adresy postupují po řádcích, jako když čteme řádky na stránce knihy. Protože víme, že ZX Spectrum má 24 znakových řádků po 32 znakových čtvercích, celkem 768, známe i délku paměti atributů. Paměť začíná na adrese 22528 a končí na adrese 23295. Následující adresy (od 23296) již náleží tiskovému bufferu.

Také pro určení adresy znakového čtverce používám tabulku, abych nemusel adresu při tvorbě programu vypočítávat. Nebudeme ji kreslit celou, je jednoduché si ji podle následujícího fragmentu doplnit:

	: číslo znakového sloupce								
22528	0	1	2	3	4	5	6	7	. . .
číslo znak. řádku	0	1	2	3	4	5	6	7	-
	32	33	34	35	36	37	38	39	-
	64	65	66	67	68	69	70	71	-
	96	97	98	99	100	101	102	103	-
	128	129	130	131	132	133	134	135	-
	160	161	162	163	164	165	166	167	-

.

. (tabulka pokračuje až do 31. znakového sloupce

. a do 24. znakového řádku)

V jednotlivých čtvercích jsou čísla, která po přičtení k základní adrese 0.znakového čtverce (tedy k hodnotě 22528) udají adresu stejně umístěného čtverce.

Např. chci-li pracovat s atributem zvýrazněného čtverce, jeho adresa bude:

22528 + 102 = 22630

Ale přiznám se, že jsem mnohdy trochu líný to sečítat, a tak klidně takový příkaz zadávám ve formě:

100 POKE 22528 + 102, xxx

Asi bych se za to měl trochu stydět. Nicméně faktem zůstává, že práce s tabulkou ušetří spoustu rutinního počítání.

ADRESOVÁNÍ PAMĚTI ATRIBUTŮ BINÁRNĚ

I na adresování atributů je vhodné se podívat binárně. Nebudeme zde dokazovat, jak jsme k jednotlivým číslům došli - zvídavý čtenář si to hravě odvodí sám. Podíváme se přímo na výsledek.

Adresa atributů 0 a znaku 0 na nultém znakovém řádku bude 22528. Binárně bude vypadat takto:

```
: 15 14 13 : 12 11 : 10 : 9 8 :: 7 6 5 : 4 3 2 1 0 :
:          :       :    :    ::       :           :
: 0  1  0  : 1  1  : 0  : 0 0 :: 0 0 0 : 0 0 0 0 0 :
:          :       :    :    ::       :           :
:     A    :  A1   : A2 :  B  ::   D   :     E     :
:          :       :    :    ::       :           :
                          :             :
                          :     BD      :
```

A,A1,A2 - číslo určující atributovou část paměti; konstanta
B - určení třetiny; počítáno shora
BD - určení znakového řádku; počítáno shora na celé obrazovce
D - určení řádku; počítáno shora v jedné třetině
E - určení znakového čtverce; počítáno zleva

Když se na tento rozpis podíváme, vidíme, že i zde vlastně platí všechno, co jsme si řekli o paměti kresby. Tedy např. v části E (bereme-li ji zcela samostatně) máme binární formou zapsána čísla 0-31, přímo určující pořadové číslo příslušného čtverce v jednom znakovém řádku.

Sloučením částí B a D vznikne samostatná část BD, která v binární formě přímo zobrazuje pořadové číslo znakového řádku (počítané od horní části obrazovky dolů, bez ohledu na třetiny).

Když se vyskytne potřeba přebarvovat celé řádky, lze toto "počítadlo" s výhodou využít. Např. jeden z disassemblerů označuje různé skupiny instrukcí povelů vždy stejnou barvou - skoky jsou červené, přesuny v registrech modré apod. Pro stavbu programu s podobným "vybarvovacím" požadavkem je "počítadlo" BD jako stvořené.

POROVNÁNÍ ADRESY KRESBY A ADRESY ATRIBUTU

V kapitole o adresování obrazovky jsem uvedl, že jsem se konstruktérům ROMky v duchu omluvil za svá prvotní neuctivá slova. Když jsem však pátral po tom, jak k nějaké kresbě připojit odpovídající barvu, omluvil jsem se nahlas. Ostatně posuďte sami.

Řekněme, že jsme kreslili ve čtveřici s těmito souřadnicemi:

Třetina	1	01
Bodový řádek ve znaku	7	111
Znakový řádek ve třetině	3	011
Znakový čtverec v řádku	6	001100

Když předřadíme konstantu 010, tak podle předchozí kapitolky hravě složíme adresu tohoto místa binárně: 010 01 111 011 001100
a to je 20326 dekadicky.

Adresu atributu zjistíme zatím ještě z tabulky, se kterou již umíme pracovat. Tak zjistíme, že připočítávat v tomto případě budeme 358 - celá adresa atributu bude:

22528 + 358 = 22886

Skutečně zajímavé to začne být v okamžiku, kdy obě tyto adresy přepíšeme do binární formy a napíšeme je pod sebe:

```
          : 15 14 13 12 11 10 9 8 : 7 6 5 4 3 2 1 0 :
          ----------------------------------------
KR.20326 = : 0  1  0  0  1  1 1 1 : 0 1 1 0 0 1 1 0 :
          :                       :                 :
AT.22886 = : 0  1  0  1  1  0 0 0 : 0 1 1 0 0 1 1 0 :
```

Podobnost je vidět na první pohled. Bity 0-7 jsou naprosto stejné a pouze v bitech 8-15 vidíme rozdíly. Část A v adrese kresby KR vlastně odpovídá části B. A stejné to je i s částmi D a E.

Kdo se jen trochu vyzná ve strojovém kódu, rázem si uvědomí, že adresy píšeme do dvou registrů - a tady vidíme, že dolní bajt adresy je stejný. To je také jeden z důvodů mé omluvy konstruktérům ROMky. Celé toto hraní si s čísly není nijak samoúčelné. Vše využijeme při kresbě a jejím vybarvování.

Jinak řečeno - z adresy kresby potřebujeme vypočítat adresu atributů. A protože je dolní bajt stejný (bity 0-7), máme ušetřenu polovinu práce. Ale co s horními bajty? Zopakujme si je - budou vypadat takto:

```
            : 15  14  13  12  11  10  9  8 :
            :                               :
  KRESBA  - : 0   1   0 : 0   1 : 1  1  1  :
            :             : .   . :         :
            :               .   . . . . . .:
            :               .               .:
            :               . . . . . . . .:
            :                       :  .  .:
 ATRIBUTY - : 0  1   0    1   1   0 : 0  1 :
```

Bity 8-10 v adrese kresby určují, o který bodový řádek jde. Všechny bodové řádky však spadají vždy pod jediný znakový čtverec, tedy i pod jediný čtverec atributový - tak nás vlastně nezajímají, protože je budeme přebarvovat všechny najednou. Bity 13-15 tvoří konstantu. Takže jediné, co nás na adrese kresby zajímá, jsou bity 11--12, určující třetinu.

U adresy atributu tvoří konstantu bity 10-15. Pro nás podstatná část adresy - číslo třetiny - leží v bitech 8-9. Vidíme, že jediná informace horního bajtu, která nás eminentně zajímá - číslo třetiny - se "stěhuje" tak, jak je naznačeno tečkami. Ostatní jsou konstanty.

Uvedený posun se přímo nabízí k využití a je nasnadě, že takovouto změnu velmi lehce provedeme přímo ve strojovém kódu. Je několik způsobů, jak takový program udělat. Jedním z nich může být třeba tento:

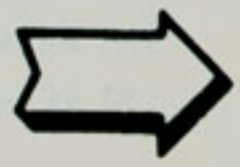
1 - adresu kresby zapsanou v registru třikrát "zrotujeme"
2 - bity 10-15 doplníme na požadovanou konstantu

To je vše, co potřebujeme udělat, abychom z adresy kresby vypočítali adresu atributu. Bod 1 můžeme realizovat trojnásobným provedením instrukce SRL (rotace doprava s doplněním nulou zleva). Tím dostaneme bity 11 až 12 na pozice 8 a 9, přičemž se log. 1 z bitu 14 dostane na bit 11, kde ji také potřebujeme mít. Zbývá doplnit levou část registru na požadovaný tvar:

```
     máme   0 0 0 0 1 0 0 1
"přidáme"   0 1 0 1 0 0 0 0
 získáme    0 1 0 1 1 0 0 1
```

což je požadovaný tvar horního bajtu správné adresy atributového čtverce. Mimochodem - binárních 01010000 je totéž, co 80 dekadicky.

Máme-li adresu kresby v registrovém páru BC, reg. C ponecháme beze změny. Reg. B můžeme modifikovat např. takovouto krátkou rutinou:



```
        SRL  B        ; rotace doprava
        SRL  B
        SRL  B
        LD A, 80      ; maska 01010000
        XOR  B        ; maskování reg.A
        LD B, A       ; výsledek do reg.B
```

V reg. BC bude odpovídající adresa atributového čtverce, příslušejícího vstupní adrese čtverce kresby.

Z uvedeného příkladu je patrné, že přepočet adresy je velmi jednoduchý. Kdykoli budeme potřebovat vybarvit nějaké právě kreslené místo, zapíšeme adresu kresby do reg. BC a necháme program projít uvedenou rutinou. Na jejím výstupu bude v reg. BC adresa atributu, kterou můžeme ihned použít pro vybarvení čtverce kresby.

Pokud bychom chtěli celou záležitost řešit Basicem, bude to o trochu složitější. Způsobů, jak věc řešit, je více. Nabízím tento:

-Nejdříve musíme modifikovat rutinu tak, aby spolupracovala s Basicem. Pro uložení adresy kresby můžeme využít volné bajty v oblasti systémových proměnných - např. adresy 23728 a 23729.

-Provedeme zápis dvou částí programu - první uloží rutinu ve strojovém kódu do paměti, do druhé budeme z klávesnice vkládat adresu kresby, a nakonec bude vypsána žádaná adresa atributu.

Modifikace rutiny není nijak složitá. Můžeme ji umístit třeba od adresy 60000:

```
 ORG 60000      ; adresa uložení rutiny
 LD BC, (23728); vyzvednutí adresy
 SRL B          ; dále totéž jako výše
 SRL B
 SRL B
 LD A,80
 XOR B
 LD B,A
 RET            ; návrat do Basicu
```

Pro uložení rutiny do paměti použijeme tento podprogram:

```
 110 RESTORE 160
 120 FOR N=0 TO 14
 130 READ A
 140 POKE 60000+N,A
 150 NEXT N
 160 DATA 237, 75,176,92,203,56,203,56,203,56,62,80,168,71,201
 170 RETURN
```

Přepočtu adres bude sloužit tento podprogram:

```
 210 INPUT "Vlož adresu kresby";A
 220 IF A ≤ 16384 OR A ≥ 22527 THEN BEEP .1,35: GOTO 210
 230 POKE 23728,A-256*INT(A/256)
 240 POKE 23729,INT(A/256)

 310 PRINT "ADR.KRESBY=";A
 320 PRINT "ADR.ATRIB.=";USR 60000
 330 RETURN
```

Na řádce 320 je příkaz PRINT USR xxxxx, který volá rutinu na adrese xxxxx. Po návratu do Basicu je na obrazovku vypsán obsah reg.BC (to je zvláštnost uvedeného příkazu).

Dosud jsme se zabývali jen formou zápisu binárního a dekadického. Zájemcům o práci s obrazovou pamětí ZX Spectra doporučuji, aby si ji prošli i s čísly hexadekadickými. Přijdou na leccos zajímavého.

V příští, poslední části, si na praktickém příkladu ukážeme, jak podstatu kresby "šikmých" barevných ploch za použití Basicu i strojového kódu.

Jiří POBŘÍSLO

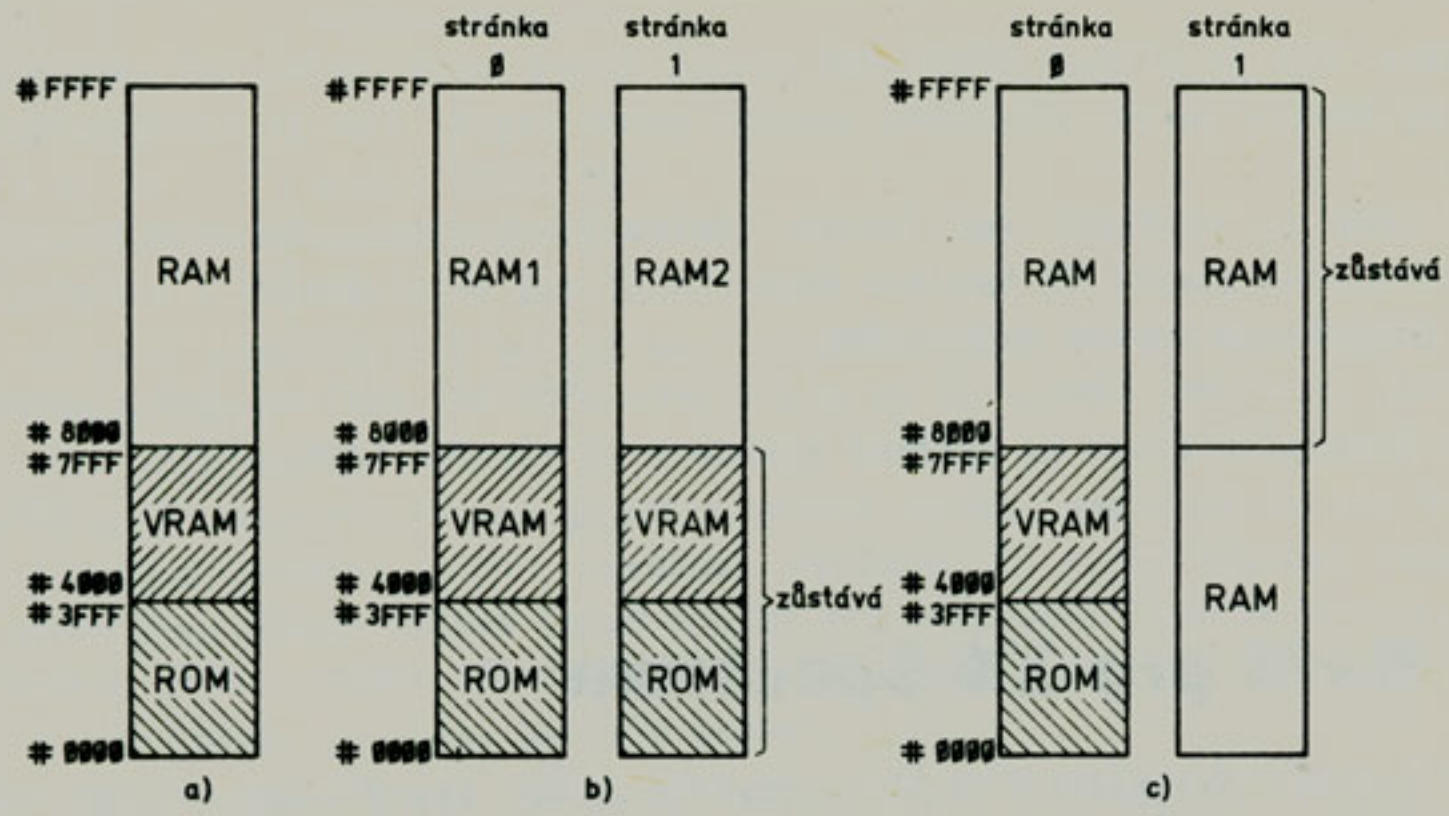
80 K RAM pro ZX Spectrum

Každý majitel ZX-Spectra jistě napsal program, který nemohl plnit funkci jen proto, že pracoval s datovým souborem tak velkým, že se prostě nevešel do paměti. Nebo uživatel potřeboval na Spectru napsat program pro jiný počítač, který má paměť RAM umístěnou od začátku adresového prostoru. Mnoho lidí, kteří pracují v asembleru, bezesporu trápí skutečnost, že při zhroucení Spectra musí chtě-nechtě stisknout RESET a nahrát třeba celých 40 K z kazety znovu. Nedostatek paměti rovněž pocítíme, chceme-li si monitorem prohlédnout nějakou osmačtyřicetikilobajtovou hru. Dlouhé hry se navíc obtížně kopírují, musí se půlit nebo jinak upravovat. A přitom stačí si uvědomit, že případné zvětšení operační paměti počítače má každý ve svých rukou.

Nejjednodušší cesta, která se nabízí, je nahradit stávající paměti 32K×1 (4532) na adresách # 8000.. # FFFF u Spectra 48K paměti 64K×1 (4164). Při této úpravě není totiž nutno zhotovovat desku s plošnými spoji pro nové paměti. Paměti 64K×1 jsou již relativně dobře dostupné i u nás. Spíše jde o to, kam tyto paměti umístit v adresovém prostoru Spectra. Podíváme-li se na adresaci paměti (obr. 1a), vidíme, že zaujímá celých 64K. To je největší možná paměť, kterou je mikroprocesor Z80 schopen přímo adresovat. Paměť je proto nutno stránkovat. To znamená, že v určitém okamžiku má procesor přístup jen do části celé operační paměti (do jedné stránky), v jiné chvíli zas do stránky jiné. Výběr jednotlivých stránek se provádí buďto hardwarově (dospěje-li provádění programu na určitou adresu), nebo softwarově (procesor si instrukcí OUT sám stránky přepne). Naše Spectrum tedy bude mít dvě stránky paměti přepínané softwarově (abychom mohli používat standardní programy beze změn, nemluvě o snadnější hardwarové realizaci). Protože však nesmíme zapomenout na ROM a VRAM (VideoRAM), které jsou také na sběrnici, je nutné uspořádání obou stránek zvolit co nejvýhodněji. V zahraniční literatuře pro uživatele počítačů Sinclair jsme se mohli před časem dočíst o takovém stránkování, kde se přepínají dvě poloviny nově vzniklé paměti 64 K RAM, a to na místě původní paměti RAM (viz obr. 1b). Toto řešení ale není pro nás příliš výhodné, a to hned ze dvou důvodů:

1. Rozsah paměti na adresách # 0000.. # 3FFF není možno pro program využít (je zde nepotřebná paměť ROM), stejně tak jako oblast obrazovky.
2. Celá paměť RAM není umístěna v jedné stránce, tzn. během provádění programu je nutné stále přestránkovávat.

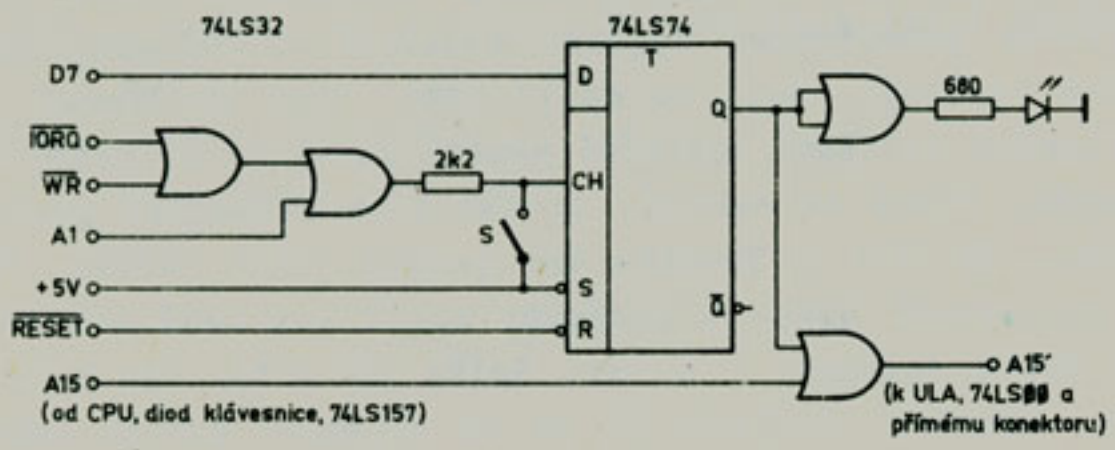
Uvedené nedostatky odstraňuje stránkování nově vzniklé paměti přes ROM a VRAM.



Obr. 1a Adresace paměti obyčejného Spectra

Obr. 1b Adresacě paměti podle zahr. časopisu

Obr. 1c Adresace paměti podle uvedeného zapojení



Obr. 2. Schéma zapojení stránkovacího portu

Paměť 64 K RAM je celá přístupná v první stránce, obrazovka a ROM jsou ve stránce nulté a tudíž zbytečně nezabírají prostor v paměti pro program (obr. 1c). V praxi také oceníme fakt, že prostor v takto vzniklé stránce paměti běží rychleji než v obyčejném Spectru, protože nenastává sdílení času mezi procesorem a obvodem ULA při přístupu do paměti na adresách #4000.. #7FFF.

Schéma zapojení portu, který umožní stránkování paměti, je velmi jednoduché (obr. 2). Nastavení nulté nebo první stránky určuje stav klopného obvodu D (polovina 74LS74). Zápis do tohoto obvodu ze sběrnice je umožněn dvěma hradly OR (74LS32). Výběr nastane při $\overline{IORQ}$=$\overline{WR}$=A1=L, tomu odpovídá adresa #FD, kterou nevyužívá žádná ze standardně vyráběných periférií. Aktivace paměti se provádí 7.bitem datové sběrnice. Tedy provedením OUT (#FD),A se připojí stránka 0, je-li v akumulátoru #00, nebo stránka 1, je-li v akumulátoru #80. Činnost stránkování je následující: je-li klopný obvod D ve stavu L, vodič A15' je funkčně shodný s vodičem A15 a Spectrum

pracuje normálním způsobem. Zákaz výběru RAM na adresách #0000..#7FFF v první stránce provádí hradla NAND generující signály $\overline{RAS}$ a $\overline{CAS}$ (viz zapojení Spectra), neboť tvorba těchto signálů je závislá na stavu A15'. Aktivujeme-li první stránku, překlopí se obvod D do stavu H a způsobí, že signál A15' je trvale na úrovni H. Tím je znemožněn výběr ROM a VRAM (výběr těchto pamětí obstarává ULA v závislosti na A15' a A14). Přístup obvodu ULA do VRAM je uskutečňován obvyklým způsobem. Naopak výběr RAM 64K probíhá v celém adresovém prostoru, neboť A15' je ve stavu H jako kdyby se vybírala paměť 32K u obyčejného Spectra. Dále je třeba zajistit, aby při zapnutí Spectra se nastavila nultá stránka. To řeší spojení resetovacího vstupu obvodu D se signálem RESET. Pro případ, že by nějaký program přece jen manipuloval s portem #FD, je možno stránkování vypnout ručně pomocí miniaturního spínače.

Čtvrté hradlo OR z pouzdra 74LS32 spíná diodu LED, která indikuje aktivaci stránky 1.

ZX Interface 1 stránkuje svou ROM přes ROM ve Spectru. Na tomto místě se tedy stránkují vlastně tři různé paměti. Ke kolizi však zde dojít nemůže, protože Interface 1 se aktivuje hardwarově od adresy #0008 a této hodnoty na adresové sběrnici jdoucí ze Spectra nelze dosáhnout, protože v případě přestránkování je A15' trvale v úrovni H.

Mechanická realizace rozšíření paměti nenaráží na větší problémy. Nejdříve vypájíme vybrané paměti a na jejich místo bez jakékoliv změny zapojení zapájíme paměti nové. Zkušenosti ukázaly, že dokonce není ani třeba používat odsávačku. Na malou destičku umístíme dva zbývající integrované obvody a vodiči je propojíme s patřičnými body na základní desce. V klasickém Spectru se destička po menších problémech dá umístit pod chladič stabilizátoru napětí, u Spectra + ji přišroubujeme k nálitku na vnitřní straně skříňky (vpravo dole, jako kdyby pan Sinclair tušil, že sem budeme něco dávat). Je ale třeba dát pozor při zapojování vodičů A15 a A 15'. Na správných místech přeškrábneme plošný spoj a zkontrolujeme ohmmetrem. Dále vypájíme propojku u multiplexeru 74LS157 a jeho nejvyšší adresový vstup spojíme s A15. Od původní A15 se tímto zásahem odpojí vstup A15 obvodu ULA, vstup 74LS00 a také kontakt A15 přímého konektoru Spectra. Spojením těchto bodů dostaneme vodič A15'. Po zapojení všech obvodů a kontrole zapojení vizuálně a ohmmetrem zapneme Spectrum do sítě. Mělo by se nastartovat známým testem paměti a zobrazit hlášku "© 1982 Sinclair Research Ltd". Pak zkusíme dát z Basicu OUT 253,128. Při správné funkci se rozsvítí indikační LED a Spectrum přestane reagovat na klávesnici. Stisknutím RESET dioda LED zhasne a Spectrum se znovu nastartuje.

Několik poznámek k realizaci:

1. Obvody 74LS74 a 74LS32 nelze nahradit typy 7474 a 7432 (popř. 74S74 a 74S32) pro velký odběr proudu. Nejlepší by bylo použít obvody ALS nebo HCT.

2. Paměti 4164 musejí mít dobu přístupu maximálně 250 ns a sedmibitový občerstvovací cyklus.

3. Po zapojení paměti se u některých Specter objeví o něco větší neklid obrazu (sněžení). Lze odstranit blokovacím keramickým kondenzátorem 47K, který připájíme přímo k napájecím vývodům každé paměti. Obraz je pak kvalitnější, než jaký byl před úpravou.

4. Různé amatérské konstrukce stavěné ke Spectru a používající A15 nebudou po popsané úpravě pracovat při přestránkování do stránky 1, což by ovšem nemuselo nikomu příliš vadit.

5. Po vyresetování Spectra nezůstane zachován obsah paměti z důvodu zastavení občerstvovací činnosti procesoru. Chceme-li využívat teplého resetu, musíme reset zapojit tak, jak se má vlastně zapojovat správně (viz např. zapojení MIKRO-AR).

Specter s 80 K RAM je již postaveno několik desítek nejen v Praze, ale i v jiných městech, a přibývají další. Existují na ně již i některé programy, i když jich zatím není mnoho (komfortní kopírovací program s volným pracovním prostorem 65535 bajtů, průběžným zobrazováním velikosti volné paměti během nahrávání a možností kopírovat přes NET, a také monitor a zpětný asembler MONS3M21 začínající od adresy # 70, který umožňuje pracovat s programem od adresy # 2000 do # FFFF (nebo po jednoduché relokaci od # 006F do # E000), a to bez ztráty jediného bajtu v tomto prostoru, neboť stránkuje skrz ROM (!). Mimoto existuje úsporný kompresor dat, hodící se především pro archivaci her, grafických dat a textů). Na Spectrum 80K je možno také implementovat CP/M, což ovšem předpokládá vlastnit alespoň vnější RAM-disk a Microdrive nebo disketovou mechaniku. Čeká se na další dobrodince, ochotné napsat nějaké zajímavé programy.

Jiří LAMAČ

Hry pro ZX Spectrum

Mikrobáze ustoupila od šíření zahraničního softwaru, což nezasáhne ty, kteří chtějí počítač používat jako zdatného pomocníka - pro tyto uživatele jsou připravovány původní české a slovenské programy, jejichž nabídka bude průběžně doplňována. Citelněji jsou zasaženi ti, kterým počítač slouží jen pro zábavu. Hry se sice výměnnými sítěmi šíří rychlostí blesku, ale - ouha - bez manuálů se těžko zjišťuje nejen jejich ovládání - leckdy zůstává neproniknutelnou tmou zahalen i sám cíl hry. Proto v této rubrice budeme postupně uvádět stručné informace alespoň o těch nejpopulárnějších a nejkvalitnějších. Samozřejmě začneme "zkraje", rokem 1983.

CHEQUERED FLAG (48K)

Chequered Flag je simulace jízdy závodního auta Formule 1, při níž projíždíte některé z nejznámějších závodních okruhů. Na vaší přístrojové desce je rychloměr, ukazatel otáček, paliva a teploty motoru, řadící páka, počitadlo kol a hodiny. Vidíte i volant, který se otáčí podle vašich příkazů. Závody Formule 1 nejsou jednoduché. Kromě toho, že nesmíte vyletět ze silnice, budete se muset vyhýbat i různým překážkám:

OLEJ/VODA - Přejetím přes olej nebo vodu na silnici se dostanete do smyku s porušením vlády nad vozem.

SKLO - Po přejetí skla vám praskne pneumatika, což je provázeno značným poklesem rychlosti vozu.

PŘEHŘÁTÍ MOTORU - Dojde-li k přehřátí, vaše auto exploduje. Proto nepřeřazujte prudce na nižší rychlostní stupně, případně zaparkujte v boxech, kde se váš motor okamžitě ochladí.

PALIVO - Při velmi dlouhých závodech musíte sledovat palivoměr. Vaše nádrž se naplní při zastavení v boxech.

BOXY - Přejete-li si zastavit v boxech (vodorovné čáry po straně závodní dráhy), stačí zastavit v jejich blízkosti. V boxech bude provedena dokonalá údržba vašeho vozu (nové pneumatiky, doplnění paliva a ochlazení motoru).

OPUŠTĚNÍ SILNICE - Jestliže váš vůz opustí silnici, jeho výkonnost značně klesne. Když vyjedete příliš daleko, můžete havarovat!

Zvolte závodní okruh, počet kol a vůz. Pak už jen počkáte, až se rozsvítí zelené startovní světlo.

OVLÁDÁNÍ VOZU

Zrychlování:	**klávesa 'O'**	**(nula)**
Brzdění:	klávesa 'I'	(písmeno I)
Vyšší rychlostní stupeň:	klávesa 'M'	nebo jakákoliv napravo od 'M'
Nižší rychlostní stupeň:	klávesa 'N'	nebo jakákoliv nalevo od 'N'
Zatáčení prudce nalevo:	klávesa 'A'	
jemně nalevo:	klávesa 'S'	
jemně napravo:	klávesa 'D'	
prudce napravo:	klávesa 'F'	
Pauza:	klávesa 'H'	
Předčasné ukončení jízdy:	klávesy 'H' a 'T' dohromady	

OKRUHY

Brands Hatch	Monaco
Psion Park	Cambridge Ring
Silverstone	Paul Ricard
Micro Drive	Saturn Sands
Osterreichring	Monza

VOZY

FERETTI TURBO

Velmi silný vůz - 640 koňských sil mezi 8000 a 10000 otáčkami. Má vynikající akceleraci, ale je těžko zvládnutelný, proto je vhodný pouze pro zkušené řidiče.

PSION PEGASUS

Nový vůz, vyvinutý s použitím špičkových technologií. Síla jeho motoru je 560 koňských sil mezi 5000 a 10000 otáčkami. Je velmi rychlý, vhodný pro průměrné řidiče.

McFASTER SPECIAL

Tento vůz má automatické řazení, proto je ideální pro nezkušené řidiče. Přesto je velmi rychlý, síla jeho motoru je 500 koňských sil.

MANIC MINER (48K)

Horník Willy je polapen v podzemí. Aby se dostal ven, musí projít 20 strašlivých jeskyní. V každé jeskyni je několik blikajících klíčů, které musí sebrat. Poté projde blikajícím východem do další jeskyně. Willy má 3 životy, o které přichází tím, že:

1. Spadne z příliš velké výšky
2. Dotkne se jakékoliv pohyblivé příšerky
3. Dotkne se křoví, pavouka, krápníku nebo bomby
4. Dojde mu kyslík (zásoba kyslíku je zobrazována ve spodní části obrazovky vedle nápisu AIR)

Body získáváte jednak sbíráním klíčů, jednak tím, že opustíte místnost (bodová prémie přímo úměrná množství kyslíku, který vám zbyl). Za každých 10000 bodů získá Willy jeden život navíc.

OVLÁDÁNÍ

Start hry:	Enter
Chůze doprava:	'P'
doleva:	'O'
Skok:	'Z'
Pauza:	'A'
Zapnutí/vypnutí hudby:	'L'
Předčasné ukončení hry:	Caps Shift a SPACE dohromady

Nekonečný počet životů:
POKE 35136,35 (umístěte do prvního Basicu).

COOKIE (16/48K)

Kuchař Charlie má všechny suroviny zamknuté v kredenci, odkud je vyjímá, jen když vaří. To se neposedným surovinám nezamlouvá, proto jakmile k tomu mají příležitost, z kredence utíkají. Charlie musí suroviny zasáhnout pytlíky mouky a srazit je do mixéru. Když spadnou do popelnice, snědí je krysy, které nemají Charlieho rády - proto po něm házejí různé smetí. Na mixéru je počitadlo, které udává, kolik surovin se ještě musí do mixéru dostat. Když do něj spadne nějaké smetí, stav počitadla se zvýší. Charlie ztratí jeden ze svých životů, když se dotkne jakékoli suroviny nebo smetí.

OVLÁDÁNÍ

Pohyb doprava:	'Q
doleva:	'W
dolů:	'E
nahoru:	'R'
Házení mouky:	'T'
Pauza:	'Caps Shift'

JET SET WILLY (48K)

Po svém dobrodružství v podzemí se horník Willy vrací domů. Ale ani tady mu není dopřán klid. Jeho manželka Maria ho nepustí spát, dokud neuklidí celý dům, což není jednoduché. Dům má 60 místností, ve kterých je rozházeno přes 80 předmětů, které musí Willy posbírat. Na druhou stranu je to poněkud jednodušší než sbírat poklady v podzemí hry MANIC MINER. Zde může Willy libovolně procházet z jedné místnosti do druhé. O jeden ze svých životů přichází tím, že:

1. Dotkne se některého pohyblivého objektu.
2. Dotkne se miny.
3. Spadne z příliš velké výšky.

Jediné po čem Willy touží, je - vyspat se. Maria ale stojí ve dveřích ložnice (MASTER BEDROM) a nepustí Willyho k posteli, dokud nesebere všechny předměty!

OVLÁDÁNÍ

Start hry:	'Enter'
Chůze doleva:	'O'
doprava:	'P'
Skok:	'Z'
Pauza:	'A'
Hudba:	'Enter'

Nekonečný počet životů: POKE 35899,0 (pozn. ostatní úpravy převzít od Bahenského a umístit sem!)

SABRE WULF (48K)

V této hře jste odvážným lovcem, jehož úkolem je uniknout z obrovské džungle (16 x 16 obrazovek). Musíte však napřed najít čtyři části ztraceného amuletu. Pak vás domorodci pustí do jeskyně, která vede z džungle ven. Kromě blikajících částí amuletu najdete v džungli také meče, přilby, bedny, šperky atd. Ty vám pouze zvyšují skóre. Malé červené sošky vám přidávají jeden život, proto se snažte je sebrat.

V džungli je mnoho nepřátel. Některé z nich můžete zničit jedinou ranou mačety, jiné (velká zvířata a domorodce) můžete mačetou pouze zahnat. Jestliže jste delší dobu v jedné obrazovce, objeví se plamen, který je nezničitelný. Přímé setkání s ním znamená ztrátu života. V jižní části džungle pobíhá šakal - i toho nemůžete zneškodnit (on vás však ano).

Někdy na stezce uvidíte malý trs trávy. Z trsu čas od času vyrůstá tropická květina. Když se dotknete ROZKVETLÉ květiny, bude na vás mít zajímavé účinky, které se liší podle toho, jakou barvou květina rozkvetla (tatáž květina rozkvétá pokaždé jinou barvou). Když se květiny dotkne, nabyde lovec téže barvy a jeho chování se na určitou dobu změní (před skončením tohoto vlivu začne lovec blikat). Např. dotknete-li se žluté květiny, nemůžete se pohybovat, světle modrá květina zrychlí vaši chůzi, fialová přehodí smysl ovládacích prvků. Bílá květina předčasně ruší vliv jakékoli jiné květiny. Pokud je lovec pod vlivem květiny, je NEZRANITELNÝ - může volně procházet nepřáteli, ale nemůže používat mačetu.

Když posbíráte všechny čtyři části amuletu, vyhledejte jeskyni, před níž sedí

domorodec v masce. Vstupte dovnitř - tak zvítězíte. Když se pokusíte vstoupit do jeskyně bez amuletu, domorodec se rozzlobí. K dispozici máte 5 životů (nové, jak bylo uvedeno výše, získáte sbíráním malých červených sošek).

OVLÁDÁNÍ

Start hry:	´O´ (nula)
Pohyb doleva:	´Q´
doprava:	´W´
nahoru:	´E´
dolů:	´R´
Šermování mačetou:	´T´
Pauza:	´Caps Shift´

Nekonečný počet životů:
POKE 43575, 255: POKE 41725, 255

MOON ALERT (48K)

Astronaut havaroval na Měsíci. Musí se dostat na nejbližší měsíční základnu. Naštěstí se mu podařilo z trosek rakety zachránit speciální měsíční vozidlo. To je vybaveno horizontálním a vertikálním paprskometem a zařízením, které mu umožňuje přeskakovat terénní překážky. Nad povrchem Měsíce poletují ufoni, kteří našeho astronauta nemají příliš v lásce. Kromě toho, že na něj házejí bomby, připravili mu na cestě mnoho dalších nástrah. (Bomby nepřátel nelze likvidovat. Jedinou záchranou je včasná změna rychlosti.) Změna rychlosti je důležitá i proto, že ovlivňuje délku skoku. V pokročilejších fázích hry se můžete často zachránit jen přesně vypočítaným skokem. Stiskem tlačítka určeného ke střelbě vystřelí oba paprskomety současně. Střelbou můžete ničit jak nepřátele, tak i NĚKTERÉ překážky na cestě.

Měsíční vozidlo se během jízdy nesmí ani jedním kolem dotknout jakékoli nerovnosti a nesmí být zasaženo nepřítelem - jinak ztrácíte život. Jeden prémiový získáte za každých 10 000 bodů. Pokud vlastníte CURRAH MICROSPEECH SYNTHETIZER, bude vaše hra provázena hlasovými efekty.

OVLÁDÁNÍ

Zrychlování:	´C´
Zpomalování:	´X´
Střelba:	´J´
Skok:	´I´

Nekonečný počet životů:
POKE 39754,0

(Dokončení příště)

Rady uživatelům IQ 151

POZOR NA TLAČÍTKO CTRL

Pro zastavení výpisu programu a pro zastavení běžícího programu v Basicu používáme tlačítko CTRL, pro následující brejk pak tlačítko F3. Nepoužívejte nikdy v příručce uvedené kombinace CTRL C !. Jestliže totiž omylem stisknete místo CTRL sousední tlačítko FA, přepíšete část programu na nepoužitelný úsek. Funkce FA byla do verze BASICu přidána až dodatečně a tvůrci programového vybavení opomněli ošetření tohoto místa.

Tlačítko F1 obdobně nahradí CTRL A, F2 použijeme při odblokování klávesnice místo CTRL B. Naučme se používat F1 pro odřádkování místo klávesy CR, pokud nechceme vložit započatý programový nebo příkazový řádek. Často se tak vyhneme hlášení chyby.

Označíme-li symbolem "kt" kód stisknutého tlačítka KT, pak při kombinaci CTRL KT se generuje kód "kt - 64", který bývá kolem řídícího znaku. Proto např.

CTRL A se vyhodnotí jako 65 - 64 = 1, zruší započatý řádek a odřádkuje,
CTRL B má kód 66 - 64 = 2, zablokuje nebo odblokuje klávesnici,
CTRL C je 67 - 64 = 3, brejk zastaveného výpisu nebo programu,
CTRL O je 79 - 64 = 15, přepne do grafického režimu,
CTRL _ dá 95 - 64 = 31, smaže obrazovku ap.
Funkce klávesy CTRL zůstává zachována i v režimu MONITOR.

OBNOVA SMAZANÉHO PROGRAMU V BASICU

Vlivem nedokonalé funkce řadiče přerušení 8259 dochází někdy samovolně, někdy po stisku tlačítka BR, ke zdánlivému vymazání programu v Basicu jako při inicializaci počítače červeným tlačítkem RES. Pro tyto případy se vyplatí mít na začátku kazety nahrán krátký obnovovací program ve strojovém kódu:

```
0100 21 6E 01 7E 23 A7 C2 03

0108 01 22 6A 01 5E 23 56 7B

0110 B2 CA 18 01 EB C3 0C 01

0118 23 22 D0 00 C3 D6 CA
```

Program nahrajeme příkazem W 100,11E,100.

Obnovení programu proběhne okamžitě po přehrání kazety do počítače.

Závadu řadiče přerušení odstraníme tak, že adresy skoků v tabulce přerušení JMP F800 (skok na inicializaci počítače) nahradíme skokem JMP CAD6 (skok na teplý start BASICU). Konkrétně na adresách

7FE0 H, 7FE4 H, 7FE8 H, 7FEC H, 7FF0 H a 7FFC H

nahradíme instrukci C3 00 F8 instrukcí C3 D6 CA.

V žádném případě nestačí nahradit JMP instrukcí RET (C9), protože žádosti o přerušení nižší úrovně pak zůstanou potlačeny a dojde k zablokování klávesnice řízené přerušením 50 Hz (skok na adrese 7FF8).

Jiří JEŽEK

ROZŠÍŘENÍ PŘÍKAZŮ BASICU 6

V příkazech GOTO, GOSUB, RESTORE, LIST a RUN může následovat pouze číslo řádku, nelze na místě čísla použít výraz.

Tento nedostatek můžeme odstranit příkazem CALL, volajícím podprogram ve strojovém kódu, takto:

```
CALL HEX(D03B), výraz ... RESTORE výraz
CALL HEX(CF35), výraz ... LIST výraz .
```

Tak např.

```
CALL HEX(D03B),3980+50xN
```

nastaví ukazatel příkazu DATA na řádek 4030 pro N=1, na řádek 4080 pro N=2 ap.

Příkaz GOTO výraz lze částečně nahradit příkazem ON . GOTO s uvedením čísel všech cílových řádků, nebo elegantněji

```
CALL HEX(a),výraz ,
```

kde na adresu a umístíme krátký podprogram ve strojovém kódu

```
POP H
JMP D181 .
```

V Basicu by mohl tento podprogram vypadat např. takto:

```
... POKE 256,225:POKE 257,195:POKE 258,129:POKE 259,209:
CALL 256,výraz  .
```

Pro RUN výraz bude podprogram složitější:

```
CALL HEX(a),výraz  ,
```

kde na adrese a je nutno mít k dispozici

```
XCHG
CALL CD80
XCHG
CALL D190
JMP  CFDA  .
```

Nejdelší bude podprogram pro příkaz GOSUB výraz; podprogram uvádím pouze ve strojovém kódu:

```
EB CD 80 CD 11 DA CF C5 EB C3 85 D1 EB 22 02 00 CD 29 CC 03 C1
D1 2A C2 00 E5 2A C6 00 E5 26 8C E5 33 D5 2A 02 00 EB C3 81 D1
```

Uživatele, kteří programují pouze v BASICU 6, mohou použít navíc tyto příkazy:

```
CALL 53307, výraz   ...   RESTORE výraz
CALL 53045, výraz   ...   LIST výraz
CALL 256,výraz      ...   GOTO výraz
CALL 260,výraz      ...   RUN výraz
CALL 271,výraz      ...   GOSUB výraz
```

Potřebné podprogramy si připraví příkazem GOSUB 60000 takto:

```
60000 RESTORE 60050:FOR I=256 TO 300
60010 READ A:POKE I,A:NEXT:RETURN
60050 DATA 225,195,129,209,235,205,128,205,235,205,144,209,195
60060 DATA 218,207,235,34,2,0,205,41,204,3,193,209,42,194,0
60070 DATA 229,42,198,0,229,38,140,229,51,213,42,2,0,235,195,129,209
```

Lubomír JEŽEK

Programová nabídka MIKROBÁZE

ZX Spectrum

DrMG

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

DIAPEN

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

/uB-PASCAL

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

DATALOG

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

mikROMkód

Velmi žádaný, kompletní přehled rutin ROMky ZX Spectra 48K (tedy i ZX Spectra+ a ROMky, která je funkční v módu 48K u nových verzí ZX Spectra 128K). Kazeta bude obsahovat jednoduché rutiny, které budou voláním subrutin ROMky vykonávat požadované funkce i díky možnosti vkládání různých vstupních parametrů do hlavních rutin. Jedná se tedy o obdobu známého programu Supercode, ovšem s tím, že struktura rutin bude zcela odlišná, funkčně variabilnější a bude se plně soustředit na využití rutin paměti ROM. Celé dílo bude korunováno plným, komentovaným assemblerovým výpisem celých 16K ROMky. Pochopitelně nebude chybět ani popis hlavních programových rutin kazety. Oproti mnohým z vás známému originálnímu výpisu ROMky bude mikROMkód doplněn informacemi o možnostech práce s jejími stěžejními částmi (především kalkulátorem, rutinami obrazovky, klávesnice, ovládáním kanálů, zvuku, tisku, atd.). Program

mikROMkód se svým velice rozsáhlým manuálem stane nezbytností každému, kdo bude chtít plně využít schopností svého počítače - především znalcům assembleru mikroprocesoru Z80. Ale i stoupenci jiných jazyků budou moci využít znalostí, které jim mikROMkód poskytne.

PLOŠNÍK

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

KAREL

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

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

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

KAREL má své opodstatnění i při přípravě k programování v jazyce Basic, který má nejširší uplatnění v oblasti mikropočítačů. Je holou skutečností, že Basic nemá dostatek prvků podporujících tvorbu strukturovaných a modulárních programů. Kdo však zvládnul KARLA, má i v jazyce Basic předpoklady k vytváření účinných, přehledných a dobře modifikovatelných programů. Mikropočítače a roboty sice směřují k takové dokonalosti, že je nebude třeba programovat speciálními programovacími jazyky, ale algoritmizace je dovednost využitelná obecně, nejen při programování počítačů.

V našem programu je Karel jméno robota, který je nakreslen na obrazovce mikropočítače. Karel má na obrazovce svoje město ohraničené zdí a v tomto městě vykonává

funkci dopravní služby. Může se přesouvat z křižovatky na křižovatku a ukládat i sbírat na křižovatkách dopravní kužely, značky.

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

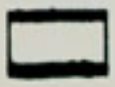
Karlovi se dávají povely běžnými českými slovy. Pochopení nových poznatků proto není ztěžováno současným učením anglických slov. Karel je také úslužný robot, všechno, co si může domyslet, udělá nebo napíše sám. S formální stránkou svého učení v nejvyšší míře sám napomáhá. A tak se stává z trpělivého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

AMSTRAD/SCHNEIDER

TRAN-SP-AM

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

Všechny programy budou distribuovány pouze na kazetách.



V minulém zpravodaji uvedená informace o programu ING.INK pro AMSTRAD/SCHNEIDER je zrušena po dohodě s autorem i klubem uživatelů tohoto typu počítače. Jeho členové se z důvodu potřeby rychlé aplikace česky a slovensky píšícího slovního procesoru rozhodli dát přednost úpravě zahraničního programu.

Tajemník klubů výpočetní techniky 602. ZO Svazarmu Dan Dočekal přišel s nabídkou přímé účasti členů klubu na převodu programů Mikrobáze na jiné typy počítačů. Protože programy jsou tvořeny v assembleru Z80, připadají v úvahu počítače se stejným typem mikroprocesoru. Jedná se především o SORD M3, SHARP MZ-821, Amstrad/Schneider. O tom, jak se tato spolupráce bude vyvíjet, vás budeme průběžně informovat.

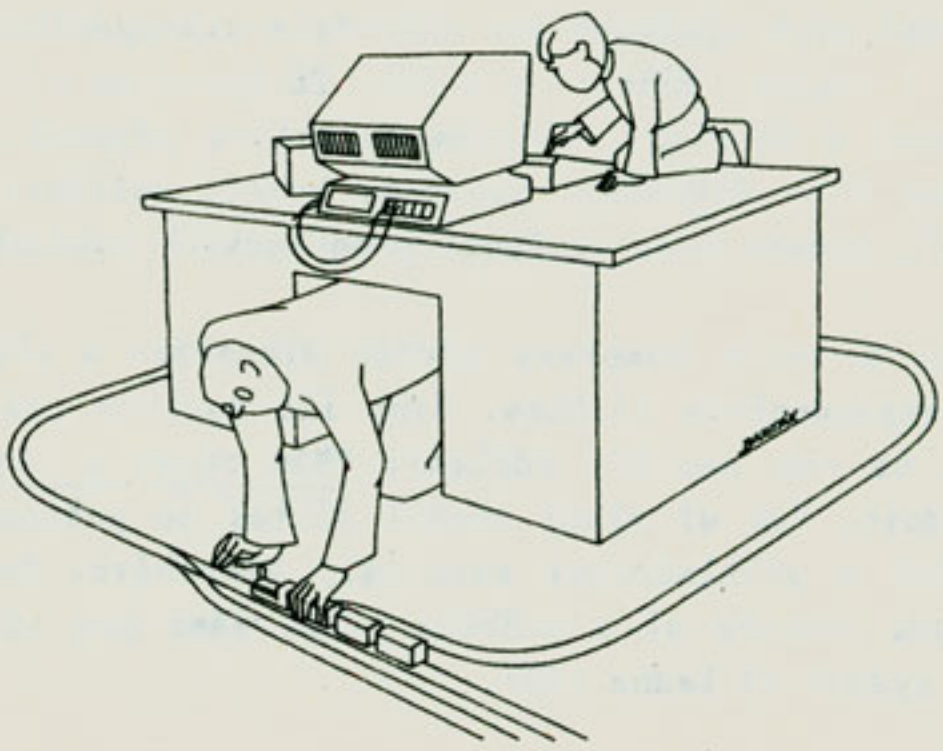
Informace o aktuálním vývoji tvorby programů pro nabídku Mikrobáze a jejich distribuci:

Jak jste si mohli povšimnout, u některých programů dochází k časovým posunům. Programová tvorba je velmi náročnou tvůrčí činností. Stává se, že i když je program hotov, přijde nápad na vám prospěšnou změnu některé jeho funkce, či se objeví nepatrná "kosmetická" vada - a pak se musí počkat na realizaci potřebné úpravy. Velmi náročnou součástí nabízeného softwaru jsou manuály. V mnoha případech svým velmi hutným rozsahem překračují 100 stran strojopisu, což je už docela slušná kniha. Manuály jdou k redakčnímu zpracování a do výroby pochopitelně až po všech "vyladěních" programu. Jistě mnozí z vás víte, jaké problémy mají zahraniční výrobci užitkového softwaru ve vztahu k dodržení slíbených lhůt jejich distribuce. Oproti leckdy velmi odbytým manuálům těchto firem chceme, aby naše manuály byly tím, čím být mají - kvalitní, vyčerpávající informací o obsluze programu. Péče, kterou jim z tohoto hlediska věnujeme, si - i s ohledem na jejich rozsah - žádá svůj čas. Jakmile Mikrobáze dostane hotový program i manuál od tvůrce, je vše už věcí výroby, která, jak dokazuje pravidelnost vydávání zpravodaje, běží na plné obrátky. Co však nemůže nikdo urychlit (ani autoři sami), je tvůrčí proces vytvářených programů. Proto vás prosíme, abyste termíny uvedené u ještě nedistribuovaných programů brali jako plán, v němž může výše uvedenými vlivy dojít k posunu.

ZX Spectrum:
Dr. MG a µB-Pascal se již distribuují.
DATALOG a KAREL - manuál ve výrobě, distribuce v květnu.
PLOŠNÍK - probíhají úpravy komunikace, distribuce v červnu.
DIAPEN a mikROMkód ve tvorbě.

Amstrad/Schneider:
TRAN-SP-AM - úpravy komunikace, distribuce v květnu.

Pro zajímavost - počet objednávek k 6. 4. 87:
Dr. MG-167, DATALOG-137, DIAPEN-134, µB-PASCAL-105, mikROMkód-97, KAREL-67



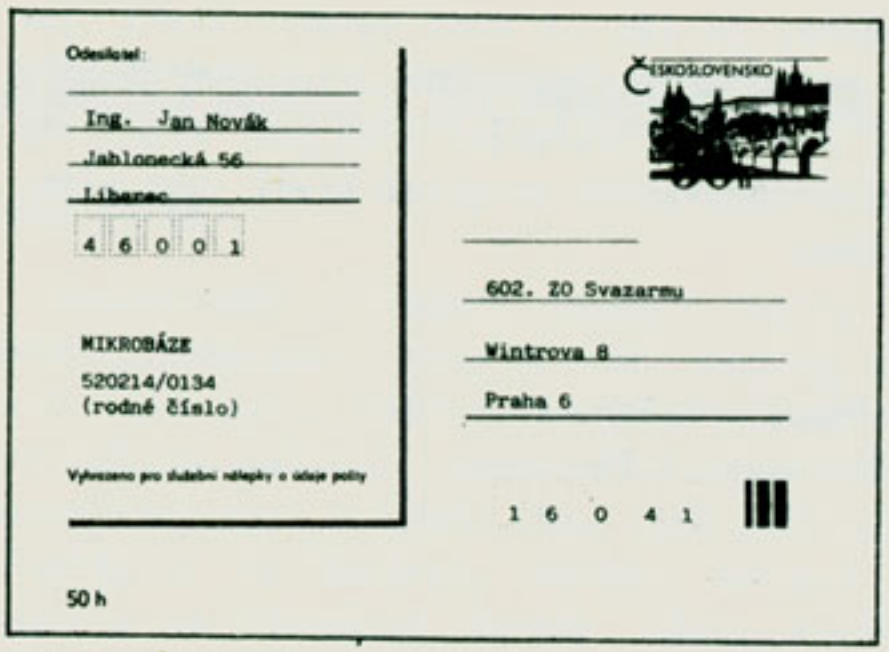
Pokyny k objednávání programů

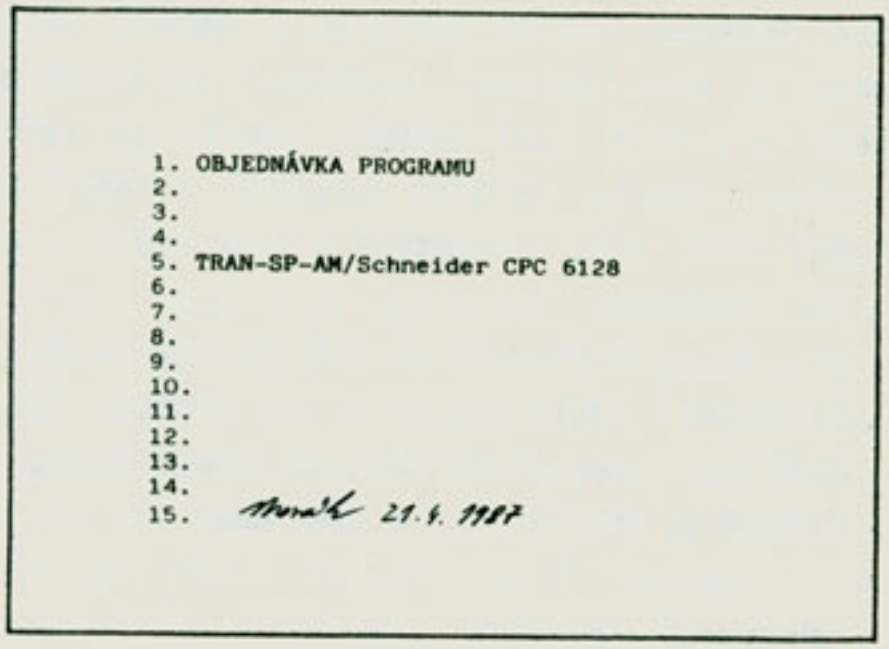
Samozřejmě, v platnosti zůstávají "pravidla hry" zveřejněná jak v Amatérském radiu (naposled v č. 5. ročníku 1985), tak v tiskovině s organizačními pokyny, kterou dostal každý, kdo projevil korespondenčním lístkem zájem o členství v Mikrobázi. Pro jistotu otiskujeme vzory vyplnění líce i rubu korespondenčního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

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

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

5. Diapen/Sinclair Spectrum





Slovo k náhodným čtenářům

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

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