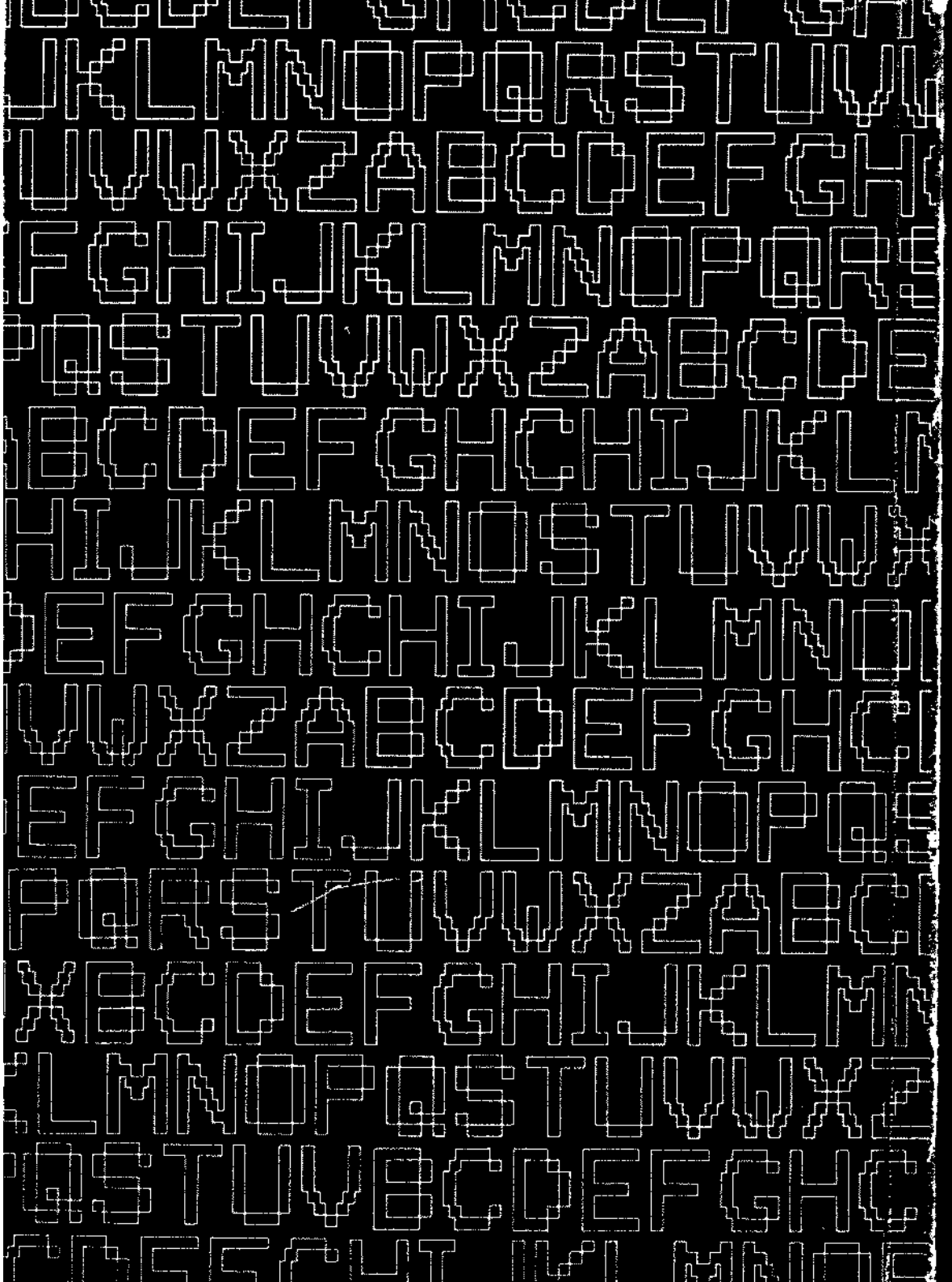


GRAMOVÉ UTINY IQ 151

RNDr. Miloslav Feil, CSc., RNDr. František Lustig,
RNDr. Jaroslav Markvart, Alan Kuběna, Petr Láf

PROGRAMOVÉ RUTINY

IQ 151

Komenium, n. p., Praha
1986

ÚVOD

Tato příručka je určena zvláště čtenářům z řad studentů a učitelů středních škol, kteří již ovládají význam klíčových slov jazyka Basic školního počítače IQ 151 a dovedou samostatně tvořit jednoduché programy. Rovněž je určena těm, kteří mají zájem zdokonalit své znalosti z programování uvedeného počítače ve strojovém kódu.

Text příručky má za úkol zdokonalit a doplnit tyto dovednosti tím, že uvádí a vykládá nejčastěji využívané části /tzv. fragmenty/ programů s určitými vlastnostmi, které lze využít v rozsáhlejších uživatelských programech. Nejedná se tedy o sbírku příkladů v běžném smyslu, ale o přehled častých programových částí a vysvětlením jejich funkce v rozsáhlém programu. V příručce se rovněž uvádějí i některé složitější příkazy zadávané v přímém režimu, které se mohou uplatnit při manipulaci s vytvářeným nebo hotovým programem v paměti počítače.

V příručce se však vyskytují i některé hotové programy servisního charakteru - například program pro přečíslování programových řádků jiného programu apod. Takové programy jsou pro odlišení od programových fragmentů na prvním programovém

řádku označeny slovem REM, za nímž následuje název programu v uvozovkách.

V příručce by mohl najít poučení začátečník, který se snaží například o tvorbu jednoduchých grafických her, ale i pokročilý, který se zajímá o využití některých podprogramů monitoru při programování v jazyce Basic nebo ve strojovém kódu, při spolupráci s magnetofonem apod.

Příklady jednoduchých částí programů /fragmenty/ se záměrně podřizují jediné podmínce - aby jasně a názorně ukazovaly určitý vykládaný jev. Jiný smysl tyto fragmenty nemají. Je na uživateli, aby na základě pochopení daného jevu na takovém příkladu dovedl jev využít ve své programátorské praxi. Vysvětlení některých jevů je popisné, jejich přesné zdůvodnění spočívá totiž až v detailním rozboru podprogramů monitoru, které však začátečník většinou úspěšně studovat nemůže. Přesto však na základě tohoto textu je schopen některé podprogramy monitoru použít.

Vzhledem k tomu, že příručka je určena jednak začátečníkům, jednak i pokročilejším, je dynamicky přizpůsobováno tempo výkladu různých částí příručky. Jedná-li se například o grafické fragmenty v jazyce Basic, předpokládá se, že je bude studovat začátečník a tedy tempo výkladu je pomalejší. Jedná-li se naopak o výklad funkce některých fragmentů ve strojovém kódu, předpokládá se, že je bude studovat již pokročilejší čtenář a tedy tempo výkladu je vyšší.

Úspěšné studium této příručky podmiňuje:

a/ znalost publikací

Jedlička, Z. - Feil, M.:" Basic pro začátečníky "

Feil, M.: " Monitor IQ 151 " ,

b/ znalost předchozích publikací a alespoň dobrá orientační znalost publikace

Feil, M.: " Strojový kód IQ 151 " ,

pokud uživatel studuje fragmenty ve strojovém kódu, volání a využití podprogramů monitoru.

Je samozřejmé, že příručka s tímto rozsahem nemůže plně pokrýt všechny možnosti na tomto poli, domníváme se však, že široké učitelské i žákovské veřejnosti může usnadnit další kroky v dovednostech a práci se školním počítačem IQ 151.

Autoři.

Důležité upozornění

Některé periferie počítače / jako například Minigraf 0507 apod./ poněkud modifikují rozdělení oblastí paměti RAM počítače, takže oblast, do níž se ukládá program v jazyce Basic nezačíná na paměťovém místě s hexadecimální adresou 16A a případně se objevují i další rozdíly. Tato příručka předpokládá, že počítač je spojen pouze se základními periferiemi /tj. televizorem a magnetofonem/ a že není k němu nic dalšího připojeno. Pochopí-li uživatel princip některého jevu podle tohoto textu a bude-li mít k počítači navíc trvale připojenou některou další periferii, je nutno, aby samostatně prostudoval rozdíly v rozdělení paměti počítače, které vznikly připojením periferie a modifikoval příslušné fragmenty na tuto situaci, což není obtížné, pokud pochopí správně vykládaný jev a činnost toho kterého příkladu.

Důsledkem změny v rozdělení částí paměti RAM je i to, že přehráváme-li program v jazyce Basic prostřednictvím příkazu L v režimu MONITOR z počítače bez dalších periferií na magnetofonový pásek a pak zpětně do paměti počítače, ke kterému je například připojen Minigraf 0507, program v jazyce Basic nejde spustit, protože počítač s touto periferií vyžaduje začátek záznamu programu v paměti jinde - nikoliv na hexadecimální adrese 16A.

Rozdělení mezi jednotlivé autory

Feil : 1/ veškeré textové zpracování;
2/ náměty a příklady kromě níže uvedených.

Kuběna: XII.1. , XII.7

Láf: VI.2.

Lustig: VIII.4., VIII.6., XI.2.1., XII.8., XII.11., XII.13

Markvart: III./f, VII.2., VII.3., VII.5., IX.1., X.,
XI.3., XII.2.a/, XIII.1.a/

I. Mazání obrazovky

Nejčastěji používané příkazy pro mazání textů na obrazovce jsou následující:

```
1/  FOR I = Ø TO 32 : PRINT : NEXT I
2/  CLS
3/  PRINT CHRØ (31)
4/  CALL HEX (D65Ø)
5/  C D65Ø
6/  CALL D65Ø
```

První čtyři příkazy je možno použít v režimu BASIC, přičemž v případě 1/ dojde k posunu původního textu na obrazovce směrem nahoru a nový text se objevuje odspodu. V případech 2/, 3/ a 4/ nedochází k posuvu původního textu vzhůru a nový text se na obrazovce objevuje od horního tiskového řádku.

Příkaz 5/ je možno použít pouze v režimu MONITOR. Případ 6/ je možno používat pouze v programech ve strojovém kódu /je to vlastně instrukce pro strojové programování/. V konkrétním strojovém programu se tato instrukce uplatní prostřednictvím trojice hexadecimálních čísel

```
..... CD  5Ø  D6 .....
```

II. Výměna textů na obrazovce

V mnohých programech v jazyce BASIC je potřebné zajistit postupné vystřídání částí delšího textu na televizní obrazovce. To se provádí v programech nejčastěji těmito způsoby:

a/ Čekání na vložení libovolného čísla. Všimněte si struktury následujícího ukázkového programu. Řádky 1Ø a 4ØØ

zastupují nějaké rozsáhlejší texty, které mají být postupně zobrazeny. Příkazy na řádcích 5 a 3ØØ mažou původní texty, které na obrazovce byly. Příkaz INPUT na řádku 2ØØ způsobí, že první část textu je na obrazovce po dobu, dokud uživatel nevloží z klávesnice do počítače číslo 1 /nebo jiné číslo/ a nestiskne následně tlačítko CR. Pak dojde ke zobrazení druhé části textu. Za příkazem INPUT je v úvozovkách text, který instruuje uživatele při postupu.

```
  5  CLS
 1Ø  PRINT " 1. cast textu "
2ØØ  INPUT " Stiskni 1 " ; A
3ØØ  CLS
4ØØ  PRINT " 2. cast textu "
```

b/ Čekání na příkaz CONT. V tomto případě dojde k vystřídání textů na obrazovce, pokud uživatel vloží v přímém režimu příkaz CONT a stiskne následně tlačítko CR. Řádek 2ØØ předchozího programu, který způsobuje čekání mezi zobrazením jednotlivých částí textů je nyní takovýto:

```
2ØØ  STOP
```

Příkaz STOP však způsobí na obrazovce hlášení o přerušení programu a hlášení READY s blikajícím kurzorem, což může někdy narušovat informace na obrazovce.

c/ Čekání na stisk libovolné černé klávesy. Je to velmi často využívaný způsob, řádek 2ØØ našeho ukázkového programu je v tomto případě

```
2ØØ  IF INKEYØ = " " THEN 2ØØ
```

Mezi uvozovkami není žádný znak ani mezera. Jestliže uživatel

nestiskne žádné černé tlačítko, příkaz na řádku 2ØØ neustále vrací chod programu opět na řádek 2ØØ. Programové řádky s vyšším číslem než 2ØØ se mohou zpracovávat až tehdy, když uživatel stiskne libovolné černé tlačítko na klávesnici. Odpadá následné stisknutí tlačítka CR nerozdíl od minulých případů.

d/ Čekání po určitou dobu s možností jejího prodloužení.

Jestliže v ukázkovém programu zaměníme programový řádek 2ØØ za řádek

```
2ØØ  WAIT ( 5Ø )
```

,

první část textu zůstane na obrazovce po dobu 5Ø krát Ø.1 s a pak bude automaticky vystřídána druhou částí textu, uživatel nemusí při střídání textů s počítačem manipulovat vůbec. Pokud se zdá doba mezi vystřídáním textů uživateli příliš krátká, stiskne po objevení daného textu na obrazovce tlačítko CTRL a drží ho stisknuté do té doby, dokud studuje text na obrazovce. Objeví-li se na obrazovce blikající kurzor, vystřídání textů nastane po stisku libovolného černého tlačítka. Pokud se kurzor na obrazovce neobjeví během doby, kdy je tlačítko CTRL stisknuté, výměna textů proběhne automaticky /uživatel v posledním případě studoval text kratší dobu, než je doba pauzy určená příkazem WAIT a k pozastavení chodu programu prostřednictvím stisknutého tlačítka CTRL nedošlo/.

Je samozřejmé, že délku "povinné" pauzy mezi jednotlivými částmi textu je možno regulovat pomocí čísla v argumentu příkazu WAIT.

Poznámka:

Všechny výše uvedené možnosti mohou být použity nejen při střídání textů na obrazovce, ale i mezi jednotlivými částmi

výpočetního programu apod.

III. Způsoby větvení programu

Většina programů v jazyce BASIC využívá větvení programu, tj. přechodu na jednotlivé části programu nebo podprogramy podle hodnoty určitého parametru. Nejčastěji se používá následujících struktur větvícího programu:

a/ Větvení pomocí vložení parametru příkazem INPUT.

Prostudujte si následující ukázkový program.

```
1ØØ INPUT A
11Ø IF A = 1 THEN 5ØØ
12Ø IF A = 2 THEN 6ØØ
13Ø IF A = 3 THEN 7ØØ
14Ø GOTO 1ØØ
5ØØ PRINT 5
51Ø END
6ØØ PRINT 6
61Ø END
7ØØ PRINT 7
71Ø END
```

Programové řádky 5ØØ až 71Ø symbolizují určité větve programu, na řádku 1ØØ je pomocí příkazu INPUT vloženo nějaké číslo a označeno identifikátorem A. Podle této číselné hodnoty programové řádky 11Ø až 13Ø zajistí přechod na zvolenou programovou větev. Řádek 14Ø má důležitou funkci spočívající v tom, že pokud vložené číslo je jiné než 1 nebo 2 nebo 3, nedojde k přechodu na nějakou větev programu, ale počítač vyžaduje na uživateli opětné zadání vhodného čísla. Jde tedy o jakési

jištění programu proti nevhodnému parametru. Uvedený způsob větvení má tu nevýhodu, že uživatel musí jednak stisknout číslo, které způsobí přechod na zvolenou programovou větev, jednak musí ještě následně stisknout tlačítko CR. Naproti tomu jeho výhoda spočívá v tom, že je možno větvit program pomocí řetězců /slov - žákových odpovědí apod./. Pak je v programu místo identifikátoru A na řádcích 1ØØ, 11Ø, 12Ø a 13Ø stringový identifikátor A$ a za rovnítky na řádcích 11Ø, 12Ø a 13Ø v uvozovkách řetězce /slova/, s nimiž se A$ porovnává. Je vybrána ta větev programu, kde se A$ shoduje s řetězcem v úvozovkách.

b/ Větvení jednoduchým použitím slova INKEY$. Programové řádky 1ØØ až 13Ø v předchozím případu zaměníme na

```
1ØØ IF INKEY$ = " " THEN 1ØØ
11Ø IF INKEY$ = " 1 " THEN 5ØØ
12Ø IF INKEY$ = " 2 " THEN 6ØØ
13Ø IF INKEY$ = " 3 " THEN 7ØØ .
```

Programové řádky 14Ø až 71Ø jsou ponechány beze změn. Na řádku 1ØØ mezi uvozovkami není žádný znak ani mezera. Uvedený řádek způsobuje čekání na stisk libovolného černého tlačítka uživatelem. Jestliže uživatel stiskne některé z tlačítek 1, 2 nebo 3, podle programových řádků 11Ø až 13Ø dojde k přechodu na zvolenou programovou větev. Pokud stiskne jiné tlačítko, pak podle programového řádku 14Ø /shodného s minulým případem/ se počítač vrátí na řádek 1ØØ a vyžaduje zadání některé ze "správných" hodnot parametru.

☐ Poznámka:

1/ Pro větvení založeném na tomto schematu lze použít

nejen čísel, ale i písmen , případně dalších znaků. Záleží pouze na tom, jaké znaky jsou uvedeny v uvozovkách na programových řádcích 11Ø až 13Ø.

2/ Programový řádek 1ØØ, který způsobuje čekání, lze vynechat. Pak je ale bezpodmínečně důležitý programový řádek 14Ø, který je nyní ve tvaru

```
14Ø GOTO 11Ø .
```

c/ Větvení s využitím slova ON. Všimněme si nyní upraveného ukázkového programu, v němž jsou programové řádky od 1ØØ do 14Ø následující:

```
1ØØ IF INKEY$ = " " THEN 1ØØ
11Ø V = VAL ( INKEY$ )
12Ø ON V GOTO 5ØØ, 6ØØ, 7ØØ
14Ø GOTO 1ØØ .
```

Ostatní programové řádky od čísla 5ØØ do 71Ø jsou shodné s uvedenými v minulých dvou případech. Mezi uvozovkami na řádku 1ØØ není žádný znak ani mezera. Řádek 1ØØ má opět význam vyčkávací. Jestliže uživatel stiskne nějaké číselné tlačítko na počítači, funkce INKEY$ nabude v tomto okamžiku hodnoty jednoznakového řetězce, který je tvořen znakem čísla právě stisknutého. Řádek 11Ø převede tento jednoznakový řetězec na odpovídající číslo, s nímž lze již narozdíl od řetězce počítat. Vzniklé číslo je označeno identifikátorem V. Podle hodnoty tohoto čísla /1 nebo 2 nebo 3/ vybere příkaz na řádku 12Ø odpovídající programovou větev, která začíná na programovém řádku uvedeném jako první nebo druhý nebo třetí v pořadí za GOTO. Pokud uživatel stiskne jiné číslo než některé ze tří uvedených, případně písmeno, příkaz na řádku 14Ø vrátí počítač zpět na řádek 1ØØ jako v minulých případech.

Programový řádek 1ØØ lze opět vynechat, pak je nutný řádek 14Ø s příkazem GOTO 11Ø.

d/ Zkrácení předchozího případu. Příklad způsobu větvení uvedený v c/ je možno ještě zkrátit takto:

```
12Ø ON VAL ( INKEY$ ) GOTO 5ØØ, 6ØØ, 7ØØ
14Ø GOTO 12Ø
```

Programové řádky 5ØØ až 71Ø jsou beze změn, programové řádky 1ØØ, 11Ø jsou vypuštěny. Protože se jedná pouze o zkrácení předchozího případu, není třeba dalších komentářů.

e/ Větvení pomocí příkazů INPUT a ON. Je používáno poměrně často, i když připomíná spíše případ a/ pouze s tím rozdílem, že pro větvení je možno použít jen tlačítek 1, 2 nebo 3, nelze používat písmen nebo řetězců. Ukázkový program je následující:

```
1ØØ INPUT V
12Ø ON V GOTO 5ØØ, 6ØØ, 7ØØ
14Ø GOTO 1ØØ
```

Programové řádky od 5ØØ do 71Ø jsou stejné jako v minulých případech.

f/ Větvení s využitím kódů tlačítek. Každé tlačítko klávesnice počítače má svůj decimální kód. Existuje podprogram monitoru, začínající na decimální adrese 63658 /tj. na hexadecimální adrese F8AA/, který čeká na stisknutí libovolného tlačítka klávesnice. Pokud uživatel některé tlačítko stiskne, zmíněný podprogram monitoru obsadí střadač A kódem stisknutého tlačítka. Pomocí funkce USR jde toto číslo ze střadače převést do režimu BASIC a dále s ním v tomto režimu pracovat. Program,

který nám ukáže kódy jednotlivých tlačítek, je například následující:

```
1 PRINT USR (63658)
2 GOTO 1
```

Po startu tohoto programu příkazem RUN se objeví blikající kurzor, stiskneme-li libovolné tlačítko, na obrazovce se objeví jeho decimální kód. Tak zjistíme například, že kód tlačítka 1 je 49, tlačítka 2 je 5Ø, tlačítka 3 je 51.

Ukázkový program, využívající k větvení kódů tlačítek, je následující:

```
1Ø5 B = USR (63658)
11Ø IF B = 49 THEN 5ØØ
12Ø IF B = 5Ø THEN 6ØØ
13Ø IF B = 51 THEN 7ØØ
```

programové řádky 5ØØ až 71Ø jsou totožné s předchozími případy, na programovém řádku 14Ø je opět "kontrolní" příkaz GOTO 1Ø5. Větvení programu nastává po stisku některého čísla z 1, 2 a 3, při stisku jiného čísla se počítač vrátí z řádku 14Ø na řádek 1Ø5 a čeká na stisk některého ze "správných" tlačítek. Řádek 1Ø5 může mít samozřejmě i tvar

```
1Ø5 B = USR ( HEX ( F8AA ) )
```

Pro větvení programu v tomto případě je možno používat tlačítek číselných i písmenových, rovněž i tlačítek F1 až F5 - záleží pouze na jejich kódech uvedených na řádcích 11Ø až 13Ø. Případně i tlačítek ovládajících pohyb kurzoru.

☐ Závěrečné poznámky:

1/ V případech a/ a e/ je nutno po vložení větvícího

čísla nebo řetězce stisknout následně tlačítko CR, u ostatních případů nikoliv.

2/ Popsanými metodami je možno zajistit i volání různých podprogramů. Konkrétně modifikace případu d/ pro volbu různých podprogramů je následující:

```
1ØØ IF INKEY$ = " " THEN 1ØØ
12Ø ON VAL ( INKEY$) GOSUB 5ØØ, 6ØØ, 7ØØ
4ØØ END
5ØØ PRINT 5
51Ø RETURN
6ØØ PRINT 6
61Ø RETURN
7ØØ PRINT 7
71Ø RETURN
```

Všimněte si, že v tomto případě nesmíme použít "kontrolní" řádek

```
14Ø GOTO 1ØØ
```

protože chod programu po návratu z některého podprogramu musí pokračovat až ke konci hlavního programu, který symbolizuje řádek 4ØØ. Použití řádku 1ØØ pro čekání na stisk klávesy je naopak bezpodmínečně nutné.
Pokud stiskneme jiná čísla než 1, 2,nebo 3, nedojde k přechodu na žádný podprogram, protože kontrolní řádek není k dispozici. Program tedy není zajištěn proti vložení nevhodných hodnot.

IV. Zajištění proti zadání chybné hodnoty nebo proti nesprávnému programovému řádku

IV.1. Zajištění proti zadání chybné hodnoty

Hodnoty vkládáme do počítače za chodu programu nejčastěji pomocí slova INPUT. Další příkaz může být kontrolní, který při nevhodném vloženém čísle vrátí počítač opět na předchozí INPUT. Na obrazovce se to projeví ale dalším tiskovým řádkem, případně se text na obrazovce posune celý vzhůru. Abychom zabránili těmto rušivým grafickým projevům opakovaného slova INPUT, použijeme následující struktury programu:

```
1ØØ CLS
11Ø PRINT & 15, 7
12Ø INPUT " Vloz kladne cislo " ; A
13Ø IF A < Ø THEN 11Ø
14Ø PRINT A
```

Příkaz CLS není nutno v konkrétním případě použít, řádek 11Ø přesune tiskovou pozici na vhodné místo obrazovky /třeba pod předchozí text apod./, řádek 13Ø zjistí, zda uživatel vložil správné číslo. Pokud je vložené číslo chybné, tisková pozice se přesune opět do původního místa /nikoliv o řádek níže/ a k posunům původních textů nebo vytvoření dalšího tiskového řádku při opakovaném vkládání hodnoty nedojde.

IV.2. Zajištění proti nevloženému programovému řádku

V některých programech v jazyce BASIC vkládá uživatel dodatečně na určitý programový řádek tvar konkrétní funkce. Program dostane uživatel do rukou v takové formě, že na řádku, kam má doplnit tvar konkrétní závislosti, je pouze slovo REM.

Program lze doplnit tak, že sám zjistí, zde uživatel na místo tohoto příkazu REM dosadil nějakou závislost - pokud závislost vložena nebyla, program se přeruší. Názorně nám to ukazuje následující jednoduchý příklad tabelačního programu.

```
1Ø REM"TABELACE"
2Ø FOR X = 1 TO 1Ø
3Ø REM
35 IF PEEK(HEX(18A)) = HEX(8E) THEN STOP
4Ø PRINT X, Y
5Ø NEXT X
```

Program nutno napsat bez mezer. Závislost má být vložena na řádek 3Ø místo slova REM. Podíváme-li se na záznam tohoto programu v paměti počítače, vidíme, že REM na řádku 3Ø se projeví svým hexadecimálním kódem 8E na hexadecimální adrese 18A. Jestliže je tento program spuštěn bez předchozího doplnění o konkrétné závislost na řádku 3Ø, příkaz na programovém řádku 35 ho automaticky zastaví.

Jestliže však uživatel doplní před startem programu řádek 3Ø třeba takto:

```
3Ø Y = SQR(X)
```

pak na hexadecimální adrese 18A již není kód slova REM a tedy programový řádek 35 nezpůsobí zastavení chodu programu.

Poznámka:

Pokud bychom na programové řádky předcházející řádku 3Ø doplnili nějaké mezery nebo kdybychom před řádek 3Ø zařadili ještě jiné programové řádky, změní se hexadecimální adresa, na níž je kód slova REM a je nutno odpovídajím způsobem změnit i řádek 35.

V. Grafické programy - pohyby znaků na obrazovce

Zopakujte si, že rohová místa na obrazovce mají hexadecimální adresy

V.1. Pohyb grafického znaku zleva doprava přes celou obrazovku bez záznamu stopy

Tento proces nám ukazuje následující program.

```
 5 CLS
1Ø FOR I = HEX(EDØØ) TO HEX(ED1F)
15 POKE I, Ø
2Ø WAIT(5)
25 POKE I, 32
3Ø NEXT I
```

Řádek 15 takového programu namaluje grafický znak na obrazovku do místa, jehož decimální adresa je určena I, řádek 25 jej vymaže /dá na jeho místo mezeru, jejíž decimální kód je 32/ a řádek 15 opět grafický znak na obrazovku namaluje, ale do pozice s adresou o 1 vyšší. To se opakuje, dokud znak neproběhne celou šířku obrazovky. Regulace rychlosti pohybu znaku provádí argument příkazu WAIT na řádku 2Ø. Kdybychom chtěli získat posuv jiného znaku než znaku s kódem Ø, pak na řádek 15 bychom doplnili decimální kód tohoto jiného znaku jako druhý parametr za příkaz POKE místo původní Ø.

V.2. Modifikace příkladu pro pohyby v jiných směrech

a/ pohyb zprava doleva - řádek 1Ø bude mít nyní tvar:

```
1Ø FOR I = HEX(ED1F) TO HEX(ED ØØ) STEP -1
```

b/ svislý pohyb shora dolů - nutno si uvědomit, že sousední místa při vertikálním pohybu dolů mají adresu zvětšující se po 32 decimálně, což je 2Ø hexadecimálně. Proto řádek 1Ø má nyní tvar:

```
1Ø FOR I = HEX(EC1Ø) TO HEX(EFFØ) STEP HEX(2Ø)
```

c/ svislý pohyb zdola nahoru - krok je tak velký jako v předchozím případě, ale je záporný, protože adresy míst obrazovky se směrem vzhůru snižují. Řádek 1Ø bude tedy tvaru:

```
1Ø FOR I = HEX(EFFØ) TO HEX(EC1Ø) STEP -HEX(2Ø)
```

d/ pohyb šikmý dolů - krok bude nyní decimálně 33, což je hexadecimálně 21. Pokud bude pohyb probíhat po vedlejší úhlopříčce obrazovky, řádek 1Ø bude:

```
1Ø FOR I = HEX(ECØØ) TO HEX(EFFF) STEP HEX(21)
```

e/ pohyb šikmý vzhůru podle této úhlopříčky obrazovky - krok bude nyní záporný a roven decimálně -33, tedy hexadecimálně -21, řádek 1Ø má nyní tvar:

```
1Ø FOR I = HEX(EFFF) TO HEX(ECØØ) STEP - HEX(21)
```

f/ pohyb po hlavní úhlopříčce nahoru - řádek 1Ø má nyní tvar:

```
1Ø FOR I = HEX(EFEØ) TO HEX(EC1F) STEP -HEX(1F)
```

Všimněte si, že krok je nyní decimálně -31, což je hexadecimálně -1F.

g/ pohyb po hlavní úhlopříčce dolů - řádek 1Ø má nyní tvar:

```
1Ø FOR I = HEX(EC1F) TO HEX(EFEØ) STEP HEX(1F)
```


h/ pohyby v jiných směrech - krok je jiné celé decimální číslo než 32, -32, 31, -31, 33, -33. Nutno samozřejmě odpovídajícím způsobem upravit obě hexadecimální meze pro I na řádku 1Ø. Liší-li se však absolutní hodnota kroku příliš od 32 decimálně, nevychází pohyb na obrazovce již názorně - jsou velké vzdálenosti mezi místy, na nichž se grafický znak postupně objevuje.

V.3. Pohyb znaku zleva doprava se záznamem stopy

Program je analogický jako v případě uvedeném v V.1. , pouze na řádku 25 není jako druhý parametr decimální kód mezery, která přemaže původní polohu posunujícího se znaku, ale decimální kód znaku, který vytváří stopu. Je-li tedy stopou tečka, pak řádek 25 v programu uvedeném v V.1. má tvar:

```
25 POKE I, 46
```

Poznámka:

Analogickým způsobem a podle výkladu v V.2. bychom získali pohyby se stopou v různých směrech na obrazovce.

V.4. Další způsob řešení programu pro pohyb znaku na obrazovce

Následující způsob je vhodnější v případech, kdy je nutno při každém posuvu znaku na obrazovce testovat, zda již grafický znak došel do jistého místa na obrazovce. Pokud nikoliv, program pokračuje opakováním jeho určité části. Následuje jednoduchý příklad pohybu grafického znaku od okraje obrazovky až do místa s hexadecimální adresou ED15.

```
 5 CLS
1Ø A = HEX(EDØØ)
15 POKE A, Ø
2Ø WAIT(5) : POKE A, 32
```

```
25 A = A + 1
3Ø IF A = > HEX(ED16) THEN END
35 GOTO 15
```

Příkaz na řádku 15 vytiskne grafický znak v místě, jenož decimalní adresa je určena A, řádek 2Ø způsobí čekání po určitou dobu a pak grafický znak na původní poloze vymaže. Řádek 25 zvýší adresu místa na obrazovce o 1 a následující řadek 3Ø testuje, zda již nebyla překročena mez, na níž se má pohyb zastavit. Pokud je překročena, program se zastaví, pokud ne, řádek 35 vrátí chod programu do místa, kde se zobrazí grafický znak na sousedním místě - tedy jeho pohyb pokračuje.

Další varianta tohoto programu je následující, využívající dvou proměnných A a B pro adresy, na nichž se tiskne grafický znak nebo mezera /zhášecí znak pro předchozí polohu grafického znaku/:

```
 5 CLS
1Ø A = HEX(EDØØ)
15 POKE A, Ø : WAIT(5)
2Ø B = A
25 A = A + 1
3Ø IF A = > HEX(ED16)THEN END
35 POKE B, 32
4Ø GOTO 15
```

Pokud čtenář porozuměl minulému programu, není tento třeba zvláště komentovat.

☐ Poznámka:

Pro pohyby v jiných směrech není na řádku 25 v obou předchozích případech příkaz A = A + 1, ale přičítá se hodnota

kroku /decimální/ v souladu s V.2. Rovněž nutno odpovídajícím způsobem upravit adresy míst, na nichž pohyb začíná a končí. Rovněž také změnami druhých čísel za příkazy POKE lze dosáhnout pohyby různých grafických znaků bez stopy a se stopou.

V.5. Pohyb znaku až k jinému grafickému znaku na obrazovce

Následující program ukazuje řešení pohybu grafického znaku od levého kraje obrazovky až před jiný grafický znak, na obrazovce předem umístěný. Srovnejte tento program s programem uvedeným v V.4.

```
1Ø A = HEX(EDØØ)
15 POKE A, Ø
2Ø WAIT(5)
25 POKE A, 32
3Ø A = A + 1
35 IF PEEK(A) = 72 THEN END
4Ø GOTO 15
```

Program startujeme příkazem

```
CLS : POKE HEX(ED17), 72 : RUN
```

odeslaným tlačítkem CR.

První příkaz startovacího složeného příkazu vyčistí obrazovku, druhý příkaz nakreslí na obrazovku písmeno H /jehož decimální kód je právě 72/ na místo o hexadecimální adrese ED17, třetí příkaz spustí program. Na programovém řádku 35 se při každém posunu grafického znaku testuje, zda následující poloha již není obsazena písmenem H. Pokud ano, program končí, pokud ne, dochází k dalšímu posuvu grafického znaku na obrazovce.

Poznámky:

1/ Pozorný čtenář na základě předchozích článků sestaví již samostatně analogické programy pro pohyby v různých směrech, různě rychlé a pro pohyb různých grafických znaků se stopou a bez ní.

2/ Dáme-li v předchozím případě na obrazovku jiné písmeno nebo znak než H, je nutno odpovídajícím způsobem změnit i decimální kód na řádku 35, jinak k zastavení pohybu před tímto novým znakem nedojde a znak bude pohybujícím se grafickým znakem "smeten".

V.6. Varianta předchozího případu s využitím akustických signálů

Změníme-li v předchozím programu řádek 35 za:

```
35 IF PEEK(A) = 72 THEN PRINT CHR$(7) : END
```

ozve se při "nárazu" pohybujícího se grafického znaku akustický signál.

Změníme-li řádek 35 za řádek:

```
35 IF PEEK(A) = 72 THEN POKE 23, 9 : POKE 24, 121 :
      CALL HEX(F973) : END
```

ozve se při nárazu pohybujícího se grafického znaku akustický signál, jehož výška a doba je určena tímto způsobem na decimálních adresách 23 a 24. Jiný akustický signál dostaneme, změníme-li druhé parametry v příkazech POKE na řádku 35.

Opět je možno známými způsoby uvedené programy rozšířit na pohyby v různých směrech, se stopou, bez stopy, různě rychlé apod.

V.7. Periodický pohyb grafického znaku v daných mezích

Je to kombinace pohybu ke značce a zpět ke druhé. Meze pohybu budou opět znázorněny pomocí písmen H na obrazovce. Program je následující:

```
10 A = HEX(ED01)
15 POKE A, 0
20 WAIT(5)
25 POKE A, 32
30 A = A + 1
35 IF PEEK(A) = 72 THEN CALL HEX(F973) : GOTO 110
40 GOTO 15
110 A = A - 1
115 POKE A, 0
120 WAIT(5)
125 POKE A, 32
130 A = A - 1
135 IF PEEK(A) = 72 THEN CALL HEX(F973) : GOTO 10
140 GOTO 115
```

Program má dvě analogické části - řádky 10 až 40 pro pohyb doprava a 110 až 140 pro pohyb zpět. Po dosažení maximální amplitudy se ozve akustický signál /viz řádky 35 a 135/.

Program startujeme pomocí příkazu:

```
CLS : POKE HEX(ED00), 72 : POKE HEX(ED17), 72 : RUN
```

který odešleme tlačítkem CR. Tento složený příkaz nejprve vymaže obrazovku, pak na ní umístí pomocí dvou příkazů POKE obě hranice tvořené písmenem H a nakonec spustí náš program.

V.8. Mazání znaků z obrazovky s akustickým signálem

Činnost následující ukázky programu spočívá v tom, že pokud pohybující se grafický znak narazí na jiný znak, vymaže ho a současně se ozve akustický signál. Pohybující se grafický znak pokračuje ve své dráze. Program je následující:

```
1Ø FOR I = HEX(EDØØ) TO HEX(ED19)
2Ø POKE I, Ø
3Ø WAIT(5)
4Ø IF PEEK(I + 1) < > 32 THEN CALL HEX(F973)
5Ø POKE I, 32
6Ø NEXT I
```

Kromě programového řádku 4Ø si čtenář snadno již zdůvodní všechny programové řádky. Smysl řádku 4Ø je následující:

Jestliže na dalším místě, na nějž se má grafický znak přesunout, není mezera s decimálním kódem 32, pak se ozve akustický signál a při dalším cyklu programu se na uvedené místo dostane mezera podle řádku 5Ø. **Tím původní grafický znak zmizí.**

Program startujeme následujícím příkazem:

```
CLS : POKE HEX(EDØ5), 15 : POKE HEX(EDØ9), 34 : RUN
```

který odešleme tlačítkem CR. Prostřední dva příkazy POKE nastaví do cesty pohybujícího se grafického znaku jiné grafické symboly.

VI. Řízení pohybu grafického znaku na obrazovce pomocí tlačítek

Dosud pohyb grafického znaku na obrazovce byl řízen pouze programově, nyní si ukážeme, jak je možno jeho pohyb řídit některými tlačítky počítače. Nejdříve ho budeme řídit určitými černými tlačítky /speciálně tlačítky s čísly 1, 2, 3 a 4/, dále si ukážeme možnost řízení jeho pohybů pomocí bílých

tlačítek ovládajících normálně pohyb kurzoru.

VI.1. Pohyb grafického znaku na obrazovce řízený černými tlačítky

Všimněte si struktury tohoto programu:

```
1Ø A = HEX(ED1Ø)
2Ø POKE A, Ø
3Ø B = A
5Ø IF INKEY$ = " 1 " THEN B = A + 1
6Ø IF INKEY$ = " 2 " THEN B = A - 1
7Ø IF INKEY$ = " 3 " THEN B = A + 32
8Ø IF INKEY$ = " 4 " THEN B = A - 32
9Ø POKE A, 32
1ØØ A = B
11Ø GOTO 2Ø
```

Uvědomte si, že pohyb grafického znaku ve svislém směru je nutno provádět tak, že k adrese jeho původní polohy přičítáme nebo odečítáme decimální číslo 32. Programové řádky 5Ø až 8Ø zajišťují, aby při stisku některého z tlačítek 1, 2, 3,nebo 4 se grafický znak posunul v určeném směru /1 - doprava, 2 - doleva, 3 - dolů, 4 - nahoru/. K pohybu grafického znaku dochází pouze tehdy, když je nějaké z jmenovaných tlačítek stisknuto. Pohyb je poměrně rychlý, program neobsahuje žádnou časovou pauzu.

Program startujeme odesláním příkazu

```
CLS : RUN
```

tlačítkem CR.

Varianty tohoto případu:

a/ Zaměníme-li druhé parametry příkazů POKE na řádcích

2Ø a 9Ø, dostaneme pohyb jiného grafického znaku a případně také stopu jeho dráhy.

b/ Jestliže řádek 9Ø upravíme na tvar

```
9Ø IF A < > B THEN POKE A, 32
```

zamezíme blikání grafického znaku, pokud se neposunuje.

c/ Pokud do programu doplníme řádek

```
95 IF PEEK(B) = 65 THEN CALL HEX(F973)
```

pak při pohybu grafického znaku po obrazovce se ozve akustický signál v tom případě, že "přejede" písmeno A kdekoliv na obrazovce /decimální kód písmena A je právě 65/. Pokud pohybující se grafický znak přejede jiný znak než A, akustický signál se neozve. Všechny druhy znaků však při "přejetí" z obrazovky zmizí. Program je však nutno startovat pouze příkazem RUN, případně si připravit na obrazovku různé znaky a písmena.

VI.2. Využití kurzorových šipek pro řízení pohybu znaku

Každému tlačítku klávesnice počítače je přiřazeno určité číslo, nazvané kódem tohoto tlačítka. Podprogram monitoru, který začíná na hexadecimální adrese F8C9, vkládá kódy bílých tlačítek do akumulátoru A a kódy černých tlačítek do registru C. Není-li stisknuto žádné tlačítko, je v akumulátoru A decimální číslo 138. Kódy jednotlivých bílých tlačítek nám vypisuje tento jednoduchý program:

```
1 PRINT USR(HEX(F8C9))
2 GOTO 1
```


Startujeme ho příkazem RUN a podle něho zjistíme, že bílému tlačítku posouvajícímu kurzor nahoru odpovídá kód 25, posouvajícímu kurzor dolů kód 26, posouvajícímu kurzor vlevo kód 8, posouvajícímu kurzor vpravo kód 32 a posouvajícímu kurzor do levého horního rohu kód 12. Všech pět čísel je decimálních.

Ukázkový program pro posuv grafického znaku na obrazovce pomocí tlačítek s kurzorovými šipkami je následující:

```
1Ø A = HEX(ED1Ø)
2Ø POKE A, Ø
3Ø B = A
4Ø N = USR(HEX(F8C9))
5Ø IF N = 8 THEN B = A - 1
6Ø IF N = 32 THEN B = A + 1
7Ø IF N = 25 THEN B = A - 32
8Ø IF N = 26 THEN B = A + 32
9Ø POKE A, 32
1ØØ A = B
11Ø GOTO 2Ø
```

Vzhledem k předchozímu výkladu není nutno tento program zvlášť komentovat. Startuje se opět příkazem CLS : RUN .

Poznámka:

a/ Tlačítka s kurzorovou šipkou vlevo nahoru je možno využít ještě k dalším účelům. Například doplníme-li řádek

```
95 IF N = 12 THEN 1Ø
```

vrací se při stisku uvedeného tlačítka pohybující se grafický znak do výchozí polohy.

b/ Pokud chceme použít v programu jiná tlačítka - například

černá, je nutno zajistit, že po vyvolání příslušného podprogramu monitoru dojde ještě k přesunu obsahu registru C do registru A /instrukce MOV A, C/. Znamená to tedy použít krátkého strojového programu, který bude začínat třeba na adrese 2ØØØ /hexadecimálně/. Je následující:

adresa	kód	instrukce
2ØØØ	CD C9 F8	CALL F8C9
2ØØ3	79	MOV A, C
2ØØ4	C9	RET

Jednotlivá čísla, která odpovídají stisku libovolné klávesy, budou nyní ve střadači A, jeho obsah je již převoditelný do režimu BASIC pomocí funkce USR. Kódy libovolného tlačítka zobrazí tedy program

```
1 PRINT USR(HEX(2ØØØ))
2 GOTO 1
```

pokud toto tlačítko držíme stisknuté.

VII. Víceznakové grafické procesy na obrazovce

Jistě jste si všimli, že dosud jsme posouvali pouze jedním znakem po obrazovce. V této kapitole si ukážeme, jak je možno posouvat na obrazovce celým útvarem, ale nejdříve si objasníme provedení inverze nebo blikání různých částí obrazovky nebo textů a kapitolu uzavřeme ukázkou posouvání textů a možností stabilních obrázků.

VII.1. Blikání nápisů

Následující dva krátké programy ukazují provedení blikání nápisu "IQ 151" . V prvním případě se k přepínání mezi normálním a inverzním režimem používá přepínacího znaku


```
PRINT CHR$(19)
```

ve druhém případě přepínání zajistíme vhodným obsazením řídící adresy 17. Oba případy jsou poměrně jednoduché, takže nepotřebují zvláštního výkladu. Oba programy startujeme odesláním příkazu CLS : RUN tlačítkem CR, rychlost blikání řídí argument příkazu WAIT.

```
1Ø PRINT&7, 7 ; CHR$(19) " IQ 151 "
2Ø PRINT&7, 7 ; " IQ 151 "
3Ø GOTO 1Ø
```

```
1Ø POKE 17, 1 : PRINT&7, 7   " IQ 151 " : WAIT(5)
2Ø POKE 17, Ø : PRINT&7, 7   " IQ 151 " : WAIT(5)
3Ø GOTO 1Ø
```

VII.2. Inverze části obrazovky

Následující program provede inverzi znaků v horní části obrazovky /od hexadecimální adresy ECØØ do EFØØ/. Probíhá jako konečný cyklus, pro každou adresu zjistí pomocí příkazu na řádku 3Ø, jakým znakem je obsazena. Invertování znaku provede tak, že ke kódu původního znaku přičte decimální číslo 128 v případě, že původní kód byl menší než 128, nebo odečte decimální číslo 128, pokud kód původního znaku je větší než 127. Znak s takto upraveným kódem je inverzní k původnímu. Kód nového znaku se přesune na místo původního "neinvertovaného" znaku - viz řádek 6Ø. Celý program je následující:

```
2Ø FOR I = HEX(ECØØ) TO HEX(EPØØ)
3Ø A = PEEK(I)
4Ø IF A > 127 THEN B = A - 128 : GOTO 6Ø
5Ø B = A + 128
6Ø POKE I, B
7Ø NEXT I
```

Program startujeme příkazem RUN. Rozmyslete si, proč lze řádky 3Ø až 6Ø nahradit: 45 POKE I, PEEK(I) + 128 AND 255 .

VII.3. Inverze části obrazovky pomocí strojového programu

Princip invertování znaků na obrazovce pomocí strojového programu spočívá ve využití instrukce

XRI, 8Ø .

Všimněte si, že je-li ve střadači libovolné hexadecimální číslo z intervalu <ØØ; FF> , pak jmenovaná instrukce přičte k tomuto číslu hexadecimální číslo 8Ø /což je decimálně 128/ v případě, že číslo ve střadači bylo menší než 8Ø hexadecimálně /což je 128 decimálně/ a odečte hexadecimální číslo 8Ø v případě, že ve střadači bylo číslo větší než 7F hexadecimálně /tj. 127 decimálně/. Je to vlastně analogie minulého příkladu. Pomocí strojového programu, který umístíme do paměti počítače od hexadecimální adresy 5ØØØ, budeme invertovat část obrazovky hexadecimální adresou ECØØ počínaje a ECFF konče. Program je následující:

adresa	kód	instrukce	význam
5ØØØ	21 ØØ EC	LXI H, ECØØ	počát. adresa oblasti do HL
5ØØ3	7E	MOV A, M	přesun obsahu místa s adr. v HL do střadače A
5ØØ4	EE 8Ø	XRI, 8Ø	invertování ve střadači

adresa	kód	instrukce	význam
5ØØ6	77	MOV M, A	přesun invertovaného znaku na původní adresu
5ØØ7	23	INX H	vzrůst čísla v HL o 1
5ØØ8	3E FF	MVI A, FF	přesun čísla FF do střadače
5ØØA	BD	CMP L	srovnání hodnot v A a L
5ØØB	C8	RZ	konec, jsou-li porovnávané hodnoty stejné
5ØØC	C3 Ø3 5Ø	JMP 5ØØ3	skok na adresu 5ØØ3

Pokud bychom chtěli, aby po ukončení jednoho invertování obrazovky se uskutečňovaly automaticky další inverze, upravíme konec našeho strojového programu takto:

adresa	kód	instrukce	význam
5ØØB	CA ØØ 5Ø	JZ 5ØØØ	jsou-li porovnávané hodnoty stejné, skok na začátek strojového programu
5ØØE	C3 Ø3 5Ø	JMP 5ØØ3	skok na adresu 5ØØ3

Invertování části obrazovky pomocí programu ve strojovém kódu probíhá však natolik rychle, že působí rušivě řádkování televizoru.

VII.4. Posuvy víceznakových útvarů na obrazovce

Pokud bychom chtěli posunovat po obrazovce nějakým víceznakovým útvarem pouze pomocí programu v jazyce BASIC /nejčastěji s využitím příkazů PRINT & /, pak bychom zjistili, že v daném časovém okamžiku lze posunout pouze jedním znakem celého víceznakového útvaru - obrazce. Chod programu v jazyce BASIC je však poměrně pomalý, takže by během posunu celého obrazce

docházelo k jeho rušivé deformaci. Je tedy nezbytné při řešení pohybu víceznakového obrazce použít programu ve strojovém kódu, který umožňuje nesrovnatelně rychlejší zakreslení všech znaků celého útvaru do předepsaných míst. Jako příklad si uvedeme posuv obrazce složeného ze 6 znaků, rozložených těsně vedle sebe ve dvou řadách a třech sloupcích. Obrazec je následující:

Po prostudování principu sestavení tohoto ukazkového programu si čtenář již snadno vytvoří analogické složitější případy. Náš program bude mít část strojovou, která provede rychlé umístění celého obrazce do určitého místa na obrazovce nebo jeho vymazání a část v jazyce BASIC, která bude určovat polohu obrazce na obrazovce, případně jeho přesuny. Poloha celého obrazce bude určována adresou jeho horního levého znaku v oblasti VIDEO-RAM.

Program ve strojovém kódu budeme vkládat do paměti od hexadecimální adresy 1ØØØ. Nejdříve na prvních 6 paměťových míst vložíme po řadě hexadecimální kódy znaků, vytvářejících celý obrazec /začínáme vlevo nahoře, po třech horních znacích přejdeme na levý spodní znak a skončíme na pravém dolním znaku/. Na následujících 6 paměťových míst vložíme hexadecimální kódy znaků, které mají na obrazovce zůstat po posunutí obrazce do jiné polohy - jsou to vlastně kódy znaků, vytvářejících stopu obrazce při jeho pohybu. Pokud je pohyb obrazce bez stopy, pak na zmíněných šesti paměťových místech je hexadecimální kód mezery, tedy 2Ø.

Jednotlivé znaky obrazce se přesouvají na určené místo obrazovky tak, že se po řadě jednotlivé hexadecimální kódy přesunou do střadače pomocí instrukce LDAX B /hexadecimální adresa paměťového místa, z něhož se kód přenáší, je v BC/ a následně se ze střadače vyšlou na určené místo obrazovky pomocí instrukce STAX D /hexadecimální adresa tohoto místa je v DE/. Adresy v obou dvojitých registrech zvyšujeme postupně pomocí instrukcí INX B a INX D. Až budeme přecházet od dokončené první řady znaků na druhou, nesmíme zapomenout na odpovídající zvýšení hodnoty hexadecimální adresy v DE /viz instrukce na adresách 1Ø2Ø až 1Ø24 v programu/. Při posunu obrazce na jiné místo se nejdříve původní vymaže nebo nahradí znaky stopy /všimněte si při té příležitosti, že čtení adres pro dvojitý registr BC začíná od prvního znaku stopy nebo mezery/ a pak se celý obrazec zakreslí v posunuté pozici - prostudujte si důkladně působení instrukcí na hexadecimálních adresách 1ØØC až 1Ø14 v programu. Program spuštěný od hexadecimální adresy 1Ø12 obrazec maluje, program spuštěný od hexadecimální adresy 1ØØC obrazec maže nebo jej nahradí znaky stopy. Celý strojový program je následující:

adresa	kód	instrukce	význam
1ØØØ	1Ø		jednotlivé znaky obrazce
1ØØ1	11		
1ØØ2	1F		
1ØØ3	ØØ		
1ØØ4	11		
1ØØ5	ØØ		
1ØØ6	2Ø		znaky stopy nebo mezery
1ØØ7	2Ø		

adresa	kód	instrukce	význam
1ØØ8	2Ø		
1ØØ9	2Ø		znaky stopy nebo mezery
1ØØA	2Ø		
1ØØB	2Ø		
1ØØC	Ø1 Ø6 1Ø	LXI B, 1ØØ6	nastavení čtení znaků stopy
1ØØF	C3 15 1Ø	JMP 1Ø15	skok na tisk obrazce
1Ø12	Ø1 ØØ 1Ø	LXI B, 1ØØØ	nastavení čtení znaků obrazce
1Ø15	ØA	LDAX B	přesun kódu 1. znaku do střadače
1Ø16	12	STAX D	přesun kódu 1. znaku ze střadače na obrazovku
1Ø17	Ø3	INX B	zvýšení obou adres
1Ø18	13	INX D	v BC a DE o 1
1Ø19	ØA	LDAX B	
1Ø1A	12	STAX D	
1Ø1B	Ø3	INX B	analogický přesun kódu
1Ø1C	13	INX D	2. a 3. znaku obrazce
1Ø1D	ØA	LDAX B	
1Ø1E	12	STAX D	
1Ø1F	Ø3	INX B	zvýšení v BC o 1
1Ø2Ø	21 1E ØØ	LXI H, ØØ1E	vytvoření adresy 1. zna-
1Ø23	19	DAD D	ku druhého řádku obrazce
1Ø24	EB	XCHG	na obrazovce
1Ø25	ØA	LDAX B	
1Ø26	12	STAX D	zakreslení 3 znaků
1Ø27	Ø3	INX B	2. řádku obrazce

adresa	kód	instrukce	význam
1Ø28	13	INX D	
1Ø29	ØA	LDAX B	viz minulý komentář
1Ø2A	12	STAX D	
1Ø2B	Ø3	INX B	
1Ø2C	13	INX D	
1Ø2D	ØA	LDAX B	
1Ø2E	12	STAX D	
1Ø2F	C9	RET	konec

Program v jazyce BASIC, který bude řídit posuvy celého obrazce, je následující:

```
 Ø CLS
 3 D1 = HEX(EE1Ø)
 5 D2 = D1
 7 CALL HEX(1Ø12), D1
1Ø A = USR(HEX(F8C9))
13 IF A = 32 THEN D1 = D1 + 1
16 IF A = 8 THEN D1 = D1 - 1
19 IF A = 26 THEN D1 = D1 + 32
21 IF A = 25 THEN D1 = D1 - 32
24 WAIT(1)
27 IF D1 <> D2 THEN CALL HEX(1ØØC), D2
3Ø GOTO 5
```

Pomocí příkazu na řádku 7 se vyvolá strojový program pro nakreslení obrazce od adresy D1, která přejde do dvojitého registru DE, neboť je uvedena za příkazem CALL jako parametr - viz výklad vlastností příkazu CALL v minulých příručkách. Programové řádky 1Ø až 21 řídí změnu polohy obrazce /tedy adresy

D1, od které se obrazec začíná malovat/ na základě stisku bílých tlačítek s kurzorovými šipkami. Na řádku 24 je pauza, která pro náš program není podstatná, pouze může jeho chod vhodně zpomalovat. Příkaz na řádku 27 provede vymazání obrazce v původní poloze, pokud došlo ke změně hodnoty adresy D1. Řádek 3Ø pak zajistí opětné zakreslení obrazce v posunuté poloze svým odkazem na "kreslící" příkaz na řádku 7 při předchozím vyrovnání hodnot D1 a D2 na řádku 5.

Po startu tohoto programu příkazem RUN ovládáme posuv celého obrazce bílými tlačítky pro posuv kurzoru.

☐ Poznámka:

Všimněte si také, že celý program probíhá jako nekonečný cyklus a v každém cyklu se tiskne stejný obrazec znovu, i když se jeho poloha nezměnila. K výmazu obrazce dochází ale pouze pokud se má posunout na obrazovce. Těchto faktů lze využít dále takto:

Pokud bychom na konci strojové části programu doplnili další instrukce, které by programově a periodicky měnily kódy některých znaků, z nichž je obrazec složen /a které jsou na hexadecimálních adresách od 1ØØØ do 1ØØ5/, obrazec by "ožil" a tímto "živým"obrazcem bychom posunovali na obrazovce již dříve popsaným způsobem. Rozšíříte-li například původní strojový program o následující instrukce, pak se bude druhý znak obrazce periodocky měnit ve svou inverzi.

adresa	kód	instrukce	význam
1Ø2F	FE 2Ø	CPI, 2Ø	zjištění, zda se obrazec maže, nebo kreslí. Jako operand musí být poslední znak stopy nebo mezera a tyto znaky musí být odlišné od posledního znaku obrazce !
1Ø31	C8	RZ	konec, pokud se kreslí stopa
1Ø32 1Ø35	21 Ø1 1Ø 7E	LXI H, 1ØØ1 MOV A, M	vložení kódu 2. znaku obrazce do střadače A
1Ø36	EE 8Ø	XRI, 8Ø	vytvoření kódu inverzního znaku ve střadači
1Ø38	77	MOV M, A	přesun kódu inverzního znaku na adresu 1ØØ1
1Ø39	C9	RET	konec

☐ Poznámky:

a/ Dalším rozšířením části programu v jazyce BASIC bychom mohli dostat další varianty programu s akustickými signály, případně spolupracovat s dalšími znaky na obrazovce podle příkladů uvedených v předchozích statích.

b/ V případě, že obrazec má více znaků na více řádcích, pak část strojového programu analogická té, která je v našem příkladu rozložená od hexadecimální adresy 1Ø15 do 1Ø2E, je řešena formou konečného programového cyklu ve strojovém kódu nebo strojových podprogramů - jde totiž o stále se opakující řadu instrukcí LDAX, STAX, INX B, INX D.

VII.5 Pohyby textů na obrazovce

Následující program ukazuje princip, podle něhož lze získat na obrazovce text, posunující se zprava doleva. U programu záleží na tom, z kolika znaků se text skládá - v našem případě jich bude mít 6Ø. Text se doplní 29 mezerami, které jsou umístěny před ním - viz programové řádky 13Ø až 15Ø. Celý text i s doplněnými mezerami je vložen v jednorozměrném řetězcovém poli Z$ o 9Ø místech, deklarovaném na programovém řádku 11Ø. Programové řádky 13Ø až 15Ø a 16Ø až 2ØØ způsobují vesměs načtení jednotlivých znaků řetězce do tohoto pole ze seznamu dat na řádcích 3ØØ a 31Ø. Programový řádek 19Ø způsobuje postupný tisk celého textu na obrazovce. Příkaz na řádku 21Ø přemísťuje kurzor do levého horního rohu obrazovky před ukončením chodu programu. Celý program je následující:

```
11Ø DIM Z$(9Ø)
12Ø CLS
13Ø FOR J = 1 TO 29
14Ø Z$(J) = "␣"
15Ø NEXT J
16Ø FOR J = 3Ø TO 89
17Ø READ Z$(J)
18Ø FOR K = Ø TO 29
19Ø PRINT & 28, 29 - K ; Z$(J - K);
2ØØ NEXT K, J
21Ø PRINT CHR$(12)
22Ø END
3ØØ DATA -,-,-,U,K,A,Z,K,A,-,V,Y,S,T,U,P,U,-,T,E,X,T,U,
     -,V,E,-,F,O,R,M,E,-,
```



```
31Ø DATA P,O,S,U,N,O,V,A,N,I,-,Z,P,R,A,V,A,-,
     D,O,L,E,V,A,-,-,-
```

Nezapomeňte, že na programovém řádku 14Ø je mezi uvozovkami mezera /tlačítko SP na počítači/.

VII.6. Stabilní obrázky a texty na obrazovce

Z příručky " Monitor IQ 151 " str. 44 víme, že lze zmenšit počet řádků na obrazovce tak, aby na spodní části obrazovky byl vymezen prostor pro text nebo obrázek, který se při výpočtech nebo dalším chodu programu nemá posunovat vzhůru - má být na obrazovce stabilně. Nyní tento postup zobecníme tak, abychom mohli mít trvalý obrázek nebo text na horní i dolní části obrazovky a další výpočty aby probíhaly pouze ve vymezené střední části obrazovky.

Na hexadecimálních adresách 2Ø a 21 je uložena při zapnutí počítače adresa ECØØ, která zajišťuje, že první adresa oblasti VIDEORAM je totožná s adresou prvního místa prvního tiskového řádku na obrazovce. Jestliže dáme na paměťové místo s hexadecimální adresou hexadecimální číslo ED místo EC, pak se tisk na obrazovce bude objevovat počínaje místem s hexadecimální adresou EDØØ, tedy výpočty nebo chod programu nenaruší text ani obrázky, umístěné předtím na obrazovku od adresy ECØØ do ECFF. Nesmíme však zapomenout zmenšit počet řádků na obrazovce, abychom mohli k obdobným účelům využít i spodní část obrazovky. Celý postup dokumentuje následující ukázkový program:

```
1 CLS
2 PRINT & 3, 5 " 1. OBRAZEK "
4 POKE HEX(21), HEX(ED)
6 PRINT & 19,5 " 2. OBRAZEK "
```

```
 8 PRINT & Ø, Ø
1Ø POKE HEX(13), 8
12 PRINT " VYPOCTY "
14 GOTO 12
```

Programový řádek 1 maže obrazovku, programový řádek 2 zastupuje nějaký obrázek v horní části obrazovky, programový řádek 4 provede úpravu dalšího zobrazení tak, že začíná až od hexadecimální adresy ED$$. Řádek 6 zastupuje obrázek na spodní části obrazovky /všimněte si, že nyní musíme řádky počítat od nově zvoleného prvního řádku/. Řádek 8 vrací kurzor na začátek nově zvoleného počátečního tiskového řádku. Pak podle řádku 1Ø dochází k omezení délky stránky na malý počet řádků. Programové řádky 12 a 14 zastupují nějaké další výpočty na shora i zdola zkrácené stránce.

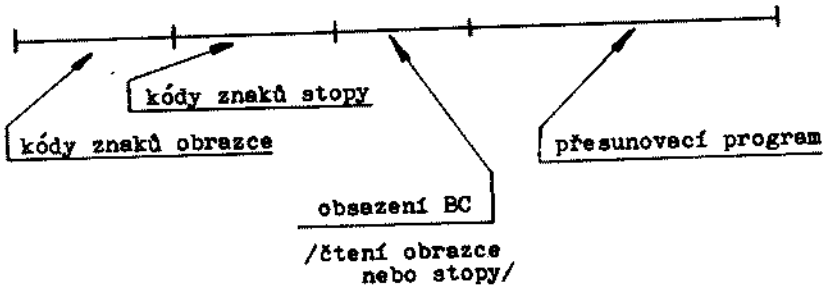
VII. 7. Úkoly a náměty ke kapitole VII.

a/ Prostudujte si sami působení příkazu CLS v případě shora i zdola zkrácené stránky /viz VII.6./.

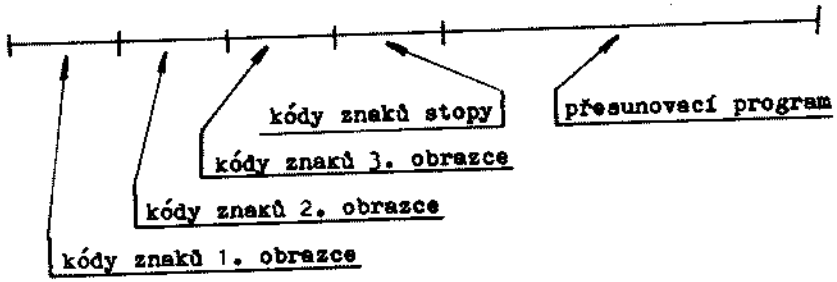
b/ Sestavte sami strojový program, který provádí posunování textů na obrazovce analogicky jako v VII.5. včetně obslužného programu v jazyce BASIC. Uděláte to velmi snadno, když si uvědomíte, že text je vlastně obrazec složený z jednoho řádku o určitém počtu znaků /konkrétně písmen/ a využijete principů vyložených v VII.4. Sestavte tedy nejdříve program pro posuv slova složeného z několika písmen, řízený tlačítky s kurzorovými šipkami, pak úpravou obslužného programu v jazyce BASIC vyřešte jeho automatický posuv. Dále zajistěte blikání jednoho nebo všech písmen textu - tj. periodické střídání s inverzními znaky. Nakonec zkuste vytvořit

strojový program pro posuv textu složeného z více slov včetně obslužného programu v jazyce BASIC.

c/ Uvědomte si, že struktura strojového programu v VII.4 je následující:



Tento vzorový program lze zobecnit takto:



Jednotlivé obrazce přesunujeme do vhodné polohy na obrazovce pomocí příkazů v jazyce BASIC typu

CALL a, P_1 , P_2 ,

přičemž podle vlastností příkazu CALL počítače IQ 151 víme, že P_2 se vkládá automaticky do dvojitého registru DE /a řídí tedy polohu obrazce nebo stopy na obrazovce/, P_1 se vkládá automaticky do dvojitého registru BC /a řídí tedy čtení znaků zvoleného obrazce nebo stopy/. Tento postup nám umožní posunovat po ploše obrazovky dynamické /periodicky proměnné/obrazce - třeba tančící panenku apod. Dynamický obrazec vzniká tak, že se na

dané místo obrazovky zobrazují postupně obrazce 1, 2 a 3, mezi nimi je vždy krátká časová pauza, takže výsledek je analogický střídání snímků při promítání filmu.

Šikovný čtenář si již nyní na tomto základě sestaví různé strojové programy včetně obslužných programů v jazyce BASIC, které mu umožní různými čtveřicemi tlačítek na klávesnici počítače posunovat různé statické víceznakové obrazce bez stopy, nebo také s různými stopami /každý obrazec zanechává jinou stopu/. Dále již snadno sestaví analogické programy pro posuv několika různých dynamických obrazců, každý dynamický obrazec bude posunovatelný jinou čtveřicí tlačítek a rovněž každý dynamický obrazec bude zanechávat na obrazovce při svém pohybu jinou stopu.

Poznámky:

a/ Parametr a v příkazu CALL a, p_1, p_2 je decimální adresa začátku přesunovacího programu.

b/ Oživení víceznakových obrazců se nazývá animací /obdobně jako ve filmové tvorbě/. Tvorba animovaných obrázků může být i atraktivní zájmovou činností mladších žáků - vytvářejí ovšem jen kódy znaků jednotlivých obrazců, přesunovací program a obslužný program snadno udělá již vedoucí zájmového kroužku.

VIII. Korigování programu během jeho chodu a využití časových informací

VIII.1. Programové mazání programu

a/ Dáme-li místo závěrečného programového příkazu END nebo STOP příkaz SCRATCH, program se automaticky vymaže po ukončení svého chodu. Nelze samozřejmě použít u programů probíhajících v nekonečném cyklu.

b/ Programové mazání programů probíhajících v nekonečném cyklu je možno provádět například tak, jak ukazuje následující příklad:

```
 5 PRINT 1
1Ø IF INKEY$ = " R " THEN SCRATCH
15 GOTO 5
```

Jestliže během chodu tohoto programu stiskneme tlačítko písmene R, program se automaticky vymaže v důsledku řádku 1Ø.

c/ Předchozí případy lze upravit tak, že místo příkazu SCRATCH dáme následující dva příkazy POKE:

```
POKE HEX(16A), Ø : POKE HEX(16B), Ø
```

Oba příkazy není třeba vysvětlovat, pokud si čtenář vzpomene na význam hexadecimálních adres 16A a 16B pro záznam programu v jazyce BASIC. Význam uvedených příkazů si však úplný začátečník nedovede v programu správně vysvětlit, příkaz SCRATCH ano.

VIII.2. Zrušení funkce tlačítka CTRL

Pomocí tlačítka CTRL, případně CTRL a C zastavujeme, případně ukončujeme chod programu. Jestliže však na decimální

adresu 254 vložíme nenulové číslo, ruší se účinek tlačítka CTRL. Ukazuje nám to názorně následující program:

```
1Ø POKE 254, 1
2Ø PRINT 1
3Ø GOTO 2Ø
```

Po jeho startu příkazem RUN zjistíte, že nejde pozastavit ani zastavit pomocí CTRL. Jde zastavit pouze tlačítkem BR /se současným skokem do režimu MONITOR/ nebo tlačítkem RES /se současným vymazáním programu/. Normální funkce tlačítka CTRL se obnoví po stisku RES nebo při hlášení READY na obrazovce.

VIII.3. Změny některých příkazů v programu za jeho chodu

VIII.3.1. Zkrácení programového cyklu během stisku tlačítka

Všimněte si struktury následujícího jednoduchého programu, který probíhá v nekonečném cyklu. Tento cyklus se zkrátí, pokud uživatel drží stisknuté tlačítko s číslem 1, kdy nekonečný cyklus probíhá pouze mezi řádky 1Ø, 2Ø, 3Ø, 6Ø a 7Ø.

```
1Ø PRINT 1 ;
2Ø PRINT 2 ;
3Ø IF INKEY$ = " 1 " THEN GOTO 6Ø
4Ø PRINT 3 ;
5Ø PRINT 4 ;
6Ø PRINT 5
7Ø GOTO 1Ø
```

VIII.3.2. Trvalé změny v programových příkazech po stisku určitého tlačítka

Následující program obsahuje na programovém řádku 64

příkaz GOTO, který má decimální kód 136 /hexadecimálně 88/ a tímto kódem je obsazena hexadecimální adresa 1A2 - o čemž se přesvědčíte pomocí příkazu D16A v režimu MONITOR. Jestliže v průběhu programu tento kód slova GOTO zaměníme pomocí příkazu

```
POKE HEX(1A2), 142
```

za kód slova REM, který je decimálně právě 142, pak provedete vlastně trvalou korekci celého programu.

Pozor !! Následující program vkládáme bez jakýchkoliv mezer mezi jednotlivými znaky a příkazy. Jestliže bychom doplnili do prvních 4 řádků nějaký znak, kód GOTO již není na adr. 1A2.

```
16 PRINT 1 ;
32 IF INKEY$ = " E " THEN POKE HEX(1A2), 142
48 IF INKEY$ = " R " THEN POKE HEX(1A2), 136
64 GOTO 16
8Ø PRINT 2
96 GOTO 16
```

Spustíte-li tento program, začne probíhat pouze mezi řádky 16 až 64. Stisknete-li tlačítko s písmenem E, pak na programovém řádku 64 se slovo GOTO změní na REM a program začne nadále probíhat mezi řádky 16 až 96, přičemž příkaz REM 16 na řádku 64 nemá účinek. Po stisku tlačítka s písmenem R se vrátí tvar i chod programu do původní verze, REM na řádku 64 se opět změní na GOTO.

VIII.4. Pauza provedená pomocí strojového programu

Časovou pauzu provádíme v programech v jazyce BASIC příkazem WAIT s vhodným argumentem. Ve strojových programech je ovšem tento způsob obtížně /nebo spíše složitě/ proveditel-

ný, proto je lépe využít podprogramu monitoru, který začíná na hexadecimální adrese F5A5, přičemž délka pauzy je řízena obsahem střadače A. Do střadače A je možno vkládat pro tento účel hexadecimální čísla z intervalu ⟨ØØ; 7F⟩ .

Ukázkový program pro pauzu provedenou pomocí strojového programu je následující:

adresa	kód	instrukce	význam
5ØØØ	3E 7Ø	MVI A, 7Ø	určení délky pauzy obsazením střadače A
5ØØ2	CD A5 F5	CALL F5A5	volání podprogramu
5ØØ5	C9	RET	konec

Dále vložíme v režimu BASIC následující jednoduchý program:

```
1 PRINT 1 ;
2 CALL HEX(5ØØØ)
3 PRINT 2
4 GOTO 1 .
```

Po startu tohoto programu příkazem RUN si všimneme, že mezi tiskem čísla 1 a 2 je časová prodleva, která je řiditelná hexadecimálním číslem na hexadecimální adrese 5ØØ1.

Poznámka:

V konkrétních strojových programech se doporučuje před použitím této časové prodlevy uschovat původní obsahy registrů /zvláště A a F/ do zásobníku pomocí instrukcí typu PUSH /zvláště PUSH PSW/ a po ukončení pauzy uschované údaje opět do příslušných registrů vložit pomocí instrukcí typu POP /zvláště POP PSW/. Čtenář si zajisté zdůvodní tento postup sám.

Tato časová prodleva se používá často v různých grafických strojových programech pro posuvy útvarů na obrazovce, protože posuvy bez časové prodlevy jsou příliš rychlé.

VIII.5. Zastavení programu časovačem

Následuje ukázka programu, který se sám zastaví po uplynutí doby 2ØØ krát 1/5Ø sekundy. Program využívá adresy 8, na níž jsou časové údaje popisované v předchozích příručkách. Analogické programy, využívající "pomalejších" adres 9 a 1Ø /decimálně/, si čtenář sestaví již sám.

```
 1Ø POKE 8, Ø
1ØØ PRINT 1 ;
2ØØ IF PEEK(8) > 2ØØ THEN END
21Ø GOTO 1ØØ .
```

Programový řádek 1Ø nejdříve nastaví výchozí hodnotu na adrese 8, program pak bude probíhat v nekonečném cyklu až do doby, kdy na adrese bude číslo větší než 2ØØ. Testování čísla na adrese 8 provádí příkaz na programovém řádku 2ØØ.

VIII.6. Příkazy GOTO, GOSUB a RESTORE s vypočteným parametrem

Víme, že za příkazy GOTO, GOSUB a RESTORE bývá parametr, který vyjadřuje decimální číslo programového řádku, na něž přejde zpracování programu. Za jmenovanými příkazy nesmí být však identifikátor - je tedy nepřípustný program, který by obsahoval třeba takovouto část:

```
515 INPUT A
527 GOTO A .
```

Pomocí nepříliš složitého programu v jazyce BASIC lze zajistit, že počítač akceptuje i příkaz uvedeného typu, za nímž je vypoč-

tený programový řádek. Uvědomíme-li si, že příkaz GOTO má hexadecimální kód 88 a na následujících paměťových místech jsou jednotlivé hexadecimální kódy cifer čísla řádku, pak stačí pomocí příkazu POKE programově změnit tyto kódy cifer v souladu s ciframi čísla, které je uvedeno pod nějakým identifikátorem. Následující dva ukázkové programy proto zavádějí na řádku 1Ø číslo požadovaného programového řádku, na které má program přejít, do paměti pod identifikátor A, řádek 2Ø udělá z tohoto vloženého čísla řetězec. Nyní se na programovém řádku 3Ø určí kód první cifry zleva /uvědomte si, že funkce ASC určuje jen kód prvního znaku řetězce/, řádky 4Ø a 5Ø určí kód druhé cifry zleva, řádky 6Ø a 7Ø kód třetí cifry zleva, řádky 8Ø a 9Ø kód čtvrté cifry zleva. Pomocí příkazů na programových řádcích 1ØØ až 13Ø se provede dosazení takto získaných kódu jednotlivých cifer do připraveného příkazu GOTO na programovém řádku 1. Řádek 14Ø předá řízení programu na takto získaný příkaz GOTO. Programové řádky 1ØØØ, 2ØØØ a 3ØØØ zastupují rozsáhlejší skutečné větve složitějšího programu. Program startujeme příkazem RUN.

Analogicky se postupuje pro případ příkazu GOSUB nebo RESTORE s vypočteným parametrem. V následujících dvou statích jsou ukázky takových programů.

VIII.6.1. Ukázka příkazu GOTO s vypočteným parametrem

Program je následující:

```
Ø GOTO 1Ø
1 GOTO XXXX
1Ø CLS : INPUT "Radek, kam chci skocit - 1ØØØ,2ØØØ,
  3ØØØ" ; A
```



```
2Ø A$ = STR$(A)
3Ø A1 = ASC(A$)
4Ø A$ = RIGHT$(A$, 3)
5Ø A2 = ASC(A$)
6Ø A$ = RIGHT$(A$, 2)
7Ø A3 = ASC(A$)
8Ø A$ = RIGHT$(A$, 1)
9Ø A4 = ASC(A$)
1ØØ POKE HEX(177), A1
11Ø POKE HEX(178), A2
12Ø POKE HEX(179), A3
13Ø POKE HEX(17A), A4
14Ø CLS : GOTO 1
1ØØØ CLS : PRINT " Proveden skok na radek 1ØØØ " : END
2ØØØ CLS : PRINT " Proveden skok na radek 2ØØØ " : END
3ØØØ CLS : PRINT " Proveden skok na radek 3ØØØ " : END
```

Upozornění: na programovém řádku 1 není mezi GOTO a prvním znakem X mezera. Dbejte rovněž upozornění z VIII.3.2.

Každý takto upravený příkaz GOTO vyžaduje od uživatele nalézt jeho umístění v paměti počítače. V našem případě je GOTO XXXX umístěn záměrně na začátku programu a "přeskočen" příkazem GOTO 1Ø na řádku Ø, aby ho bylo možné snadno v paměti najít včetně adres jednotlivých znaků X, které zastupují pozdější skutečné cifry.

Poznámky:

a/ Pokud číslo řádku takto za GOTO vkládané je výsledkem nějakých matematických operací, je nutno použít ještě funkce INT, aby číslo řádku bylo celočíselné.

b/ Po provedení výpisu programu následně po ukončení jeho chodu je místo GOTO XXXX již nějaký konkrétní příkaz GOTO 1ØØØ, nebo GOTO 2ØØØ, nebo GOTO 3ØØØ.

c/ Pokud čísla řádků za GOTO by měla být pětimístná, použijeme tvaru GOTO XXXXX a příslušné rozšíření té části programu, která počítá a dosazuje kódy všech cifer za tento příkaz.

d/ IQ 151 má sice příkaz ON ve spojení s GOTO p_1, p_2, ... , avšak pro vícenásobný programový přepínač, než se vejde na jeden programový řádek, je nutno použít uvedeného postupu.

VIII.6.2. Ukázka příkazu GOSUB s vypočteným parametrem

Příklad je analogický předchozímu, proto jej uvádíme již bez dalších vysvětlujících poznámek.

```
 Ø GOTO 1Ø
 1 GOSUB XXXX : END
 1Ø CLS : INPUT " Radek podprogramu, kam chci skočit -
    1ØØØ, 2ØØØ, 3ØØØ " ; A
 2Ø A$ = STR$(A)
 3Ø A1 = ASC(A$)
 4Ø A$ = RIGHT$(A$, 3)
 5Ø A2 = ASC(A$)
 6Ø A$ = RIGHT$(A$, 2)
 7Ø A3 = ASC(A$)
 8Ø A$ = RIGHT$(A$, 1)
 9Ø A4 = ASC(A$)
 1ØØ POKE HEX(177), A1
 11Ø POKE HEX(178), A2
 12Ø POKE HEX(179), A3
 13Ø POKE HEX(17A), A4
```



```
 14Ø CLS : GOTO 1
 1ØØØ CLS : PRINT " Proveden skok na podprogram na
      radku 1ØØØ " : RETURN
 2ØØØ CLS : PRINT " Proveden skok na podprogram na
      radku 2ØØØ " : RETURN
 3ØØØ CLS : PRINT " Proveden skok na podprogram na
      radku 3ØØØ " : RETURN
```

Analogický program pro příkaz RESTORE s vypočteným parametrem si sestaví čtenář již samostatně.

IX. Programové korekce daného programu

Úvodem si připomeneme, že záznam programu v paměti počítače začíná na hexadecimální adrese 16A, kterou na dekadický tvar převedeme

B1 = HEX(16A) .

Číslo prvního programového řádku je na decimálních adresách B1 + 2 a B1 + 3 a je rovno

B3 = PEEK(B1 + 2) + 256 * PEEK(B1 + 3)

decimálně.

Protože na prvních dvou místech v záznamu každého programového řádku je uložena adresa začátku záznamu následujícího programového řádku v paměti počítače, pak záznam druhého programového řádku začíná / u libovolného programu v jazyce BASIC/ na decimální adrese

B2 = PEEK(B1) + 256 * PEEK(B1 + 1) .

Jestliže na první dvě místa záznamu libovolného programového řádku dáme místo odkazu na adresu začátku záznamu následujícího řádku nuly, pak počítač již tento programový řádek nezpra-

cuje, rovněž tak i následující programové řádky. Nejsou ani ve výpisu programu, ani nejdou přehrát na magnetofonový pásek. /Připomeňte si v této souvislosti působení tlačítka RES./ Výše uvedeným způsobem lze provádět výmaz části programu počínaje zvoleným programovým řádkem. Postup ukazuje následující článek.

IX.1. Servisní program pro vymazání části programu

Předpokládáme, že v paměti počítače je program v jazyce BASIC v oblasti řádkových čísel Ø až 5Ø199. Za tento program vložíme servisní program, který je následující:

```
5Ø2ØØ REM"VYMAZ KONCE PROGRAMU"
5Ø21Ø INPUT " Zadej cislo radku " ; A1
5Ø22Ø B1 = HEX(16A)
5Ø23Ø B2 = PEEK(B1) + 256*PEEK(B1 + 1)
5Ø24Ø B3 = PEEK(B1 + 2) + 256*PEEK(B1 + 3)
5Ø25Ø IF B3 > = A1 THEN 5Ø28Ø
5Ø26Ø B1 = B2
5Ø27Ø GOTO 5Ø23Ø
5Ø28Ø POKE B1, Ø
5Ø29Ø POKE B1 + 1, Ø
5Ø3ØØ END
```

Servisní program startujeme příkazem RUN 5Ø2ØØ a pak zadáme číslo programového řádku, počínaje kterým chceme programové řádky vymazat. Počítač porovná nejdříve vložené číslo s číslem prvního programového řádku - viz řádky 5Ø22Ø až 5Ø24Ø, pokud je výsledek porovnání negativní, v souladu s příkazem na řádku 5Ø26Ø a 5Ø27Ø začne porovnávat číslo dalšího programového řádku s vloženým číslen atd. Pokud je výsledek porovnání na řádku 5Ø25Ø pozitivní, provede se nulování paměťových míst, odkazujících na adresu začátku následujícího řádku pomocí příkazů na řádcích 5Ø28Ø a 5Ø29Ø.

Poznámky:

1/ Servisní program provede současně s výmazem řádků obsluhovaného programu také automaticky výmaz sama sebe, pokud je obsluhovaný program před servisním programem. Proto je vhodné mít uvedený servisní program trvale nahraný na magnetofonovém pásku.

2/ Zdatnější uživatel si tento jednoduchý program doplní tak, aby na hexadecimální adresy DØ a D1 byla doplněna adresa nového konce programu.

Závěrečná poznámka:

Na vyložených principech lze sestavovat další servisní programy, které například automaticky změní příkaz PRINT za LPRINT /pro rychlé využití jiné periferie apod./. Zvláště jednoduché je to v případě, kdy zaměňovaný příkaz je hned za číslem programového řádku. Program pracuje tak, že projde všechny programové řádky a pokud na paměťovém místě pro první příkaz na řádku najde kód příkazu PRINT, změní ho na kód příkazu LPRINT. Jak najdeme jednoduše hexadecimální kód libovolného příkazu, nás poučí poznámky v závěru práce.

X. Poznámky k záměně proměnných a logice

a/ Víme, že záměnu hodnot dvou proměnných A a B lze snadno provádět pomocí třetí proměnné Z takto:

```
Z = A : A = B : B = Z
```

Abychom tuto třetí "vyrovnávací" proměnnou používat nemuseli, je možno záměnu hodnot A a B provést takto:

```
A = A + B : B = A - B : A = A - B
```

Existují i další analogické možnosti založené na dělení a násobení - pozor pouze na případné dělení nulou !

b/ Příkazu PRINT, resp. LET je možno využít k vyhodnocení pravdivosti výroků. Například

```
PRINT 1 = 1
```

dává výsledek -1 /pravda/,

```
PRINT 1 = 2
```

dává výsledek Ø /nepravda/. Rovněž tak lze použít

```
LET A = 1 = 1
LET B = 1 = 2
```

V prvním případě bude mít A hodnotu -1, ve druhém bude mít B hodnotu Ø.

Výše uvedených skutečností lze v programech využít při srovnávání dvou vícesložkových vektorů. Mějme například první vektor o složkách A1, A2, ... , A5 , druhý vektor o složkách B1, B2, ... , B5. Na otázku, kolik složek obou vektorů je stejných /například A3 = B3, A5 = B5 apod./, nám odpoví příkaz


```
PRINT - ((A1 = B1) + (A2 = B2) + (A3 = B3) +
        (A4 = B4) + (A5 = B5))
```

který odešleme tlačítkem CR. Uvědomte si, že za každou rovnost odpovídajících si složek získáte do součtu v závorce jedno číslo -1.

XI. Zaplnění a posuvy obsahu úseků paměti

XI.1. Naplnění paměťových míst pomocí strojového programu

Z práce v režimu MONITOR známe příkaz F, který obsazuje paměťová místa daného úseku paměti zvoleným číslem:

```
F 2A7, 3E2, 31
```

počáteční a koncová hexadecimální adresa obsazovaného úseku

hexadecimální číslo z intervalu <ØØ; FF> , kterým budou místa obsazena

Tento příkaz režimu MONITOR lze provádět i pomocí programu ve strojovém kódu tak, že

v registru C je číslo, kterým se úsek obsazuje
v DE je koncová adresa obsazovaného úseku paměti
v HL je počáteční adresa obsazovaného úseku paměti

a podprogram monitoru, který požadovaný úkon provede, začíná na hexadecimální adrese F258.

Například strojový program, který naplní obrazovku čísly 1 je následující - vkládáme ho do paměti od hexadecimální adresy 3ØØØ:

adresa	kód	instrukce	význam
3ØØØ	ØE 31	MVI C, 31	vložení čísla 31 do C
3ØØ2	21 ØØ EC	LXI H, ECØØ	počát. adresa do HL
3ØØ5	11 FF EF	LXI D, EFFF	koncová adresa do DE
3ØØ8	CD 58 F2	CALL F258	volání podprogramu
3ØØB	C9	RET	konec

V režimu BASIC vložíme nyní následující program, který pouze náš strojový program vyvolá a po návratu z něho čeká na stisk libovolného černého tlačítka.

```
 5 CALL HEX(3ØØØ)
1Ø IF INKEY$ = "" THEN 1Ø
15 END
```

Poznámka:

Uvědomte si, že obrazovku lze naplnit jedničkami také pouze v režimu BASIC tak, že řádek 5 má tvar

```
5 FOR I = HEX(ECØØ) TO HEX(EFFF) : POKE I, 49 : NEXT I
```

Pokud chceme uvedeným způsobem naplnit obrazovku číslem 2, pak analogický strojový program vložíme do paměti od hexadecimální adresy 32ØØ:

adresa	kód	instrukce
32ØØ	ØE 32	MVI C, 32
32Ø2	21 ØØ EC	LXI H, ECØØ
32Ø5	11 FF EF	LXI D, EFFF
32Ø8	CD 58 F2	CALL F258
32ØB	C9	RET

Rozdíl od předchozího strojového programu je pouze v hexadecimální kódu znaku, jímž se obrazovka zaplňuje.

XI.2. Posuv obsazení úseku paměti

Z práce v režimu MONITOR známe příkaz M, který posunuje obsazení paměťových míst daného úseku paměti na jiné místo.

```
M 3E1, 5A1, A1Ø
```

počáteční a koncová hexadecimální adresa úseku paměti

počáteční adresa úseku, na nějž se obsazení přesune

Tento příkaz lze také provádět pomocí strojového programu následujícím způsobem.

- v HL je počáteční adresa úseku
- v DE je koncová adresa úseku
- v BC je počáteční adresa úseku, na nějž bude obsazení přesunuto

přesun vyvolá podprogram monitoru, začínající na hexadecimální adrese F247.

Provádění přesunu obsazení úseků paměti pomocí programů ve strojovém kódu nám umožňuje:

1/ rychlé uschování momentálních údajů na obrazovce do paměti a zpětné vyvolání celé obrazovky z paměti ve vhodném okamžiku;

2/ uschování programu v jazyce BASIC do jiné části paměti, vložení dalšího programu, po jeho vymazání okamžitý přesun původního programu do operační paměti.

Oba případy popisují následující stati.

XI.2.1. Uschování obrazovky do paměti počítače

Z předcházejícího výkladu známe strojový program pro zaplnění obrazovky jedničkami. Program, který nám uschová takto zaplněnou obrazovku do paměti od hexadecimální adresy 5ØØØ, je následující / uložen v paměti od hexadecimální adresy 31ØØ /.

adresa	kód	instrukce	význam
31ØØ	21 ØØ EC	LXI H, ECØØ	vložení adres do
31Ø3	11 FF EF	LXI D, EFFF	dvojitých registrů
31Ø6	Ø1 ØØ 5Ø	LXI B, 5ØØØ	
31Ø9	CD 47 F2	CALL F247	volání podprogramu
31ØC	C9	RET	konec

Strojový program, který přesune obrazovku na úsek paměti, začínající na hexadecimální adrese 6ØØØ, vložíme od hexadecimální adresy 33ØØ a bude mít analogický tvar:

adresa	kód	instrukce	význam
33ØØ	21 ØØ EC	LXI H, ECØØ	vložení adres do
33Ø3	11 FF EF	LXI D, EFFF	dvojitých registrů
33Ø6	Ø1 ØØ 6Ø	LXI B, 6ØØØ	
33Ø9	CD 47 F2	CALL F247	volání podprogramu
33ØC	C9	RET	konec

Zpětný přesun obrazovky z úseku začínajícího na hexadecimální adrese 6ØØØ do oblasti VIDEORAM provádí následující strojový program začínající na hexadecimální adrese 34ØØ a zpětný přesun obrazovky z úseku začínajícího na hexadecimální adrese 5ØØØ do oblasti VIDEORAM provádí strojový program začínající na hexadecimální adrese 35ØØ - oba strojové programy

již zkráceně.

adresa	kód	instrukce
34ØØ	21 ØØ 6Ø	LXI H, 6ØØØ
34Ø3	11 FF 63	LXI D, 63FF
34Ø6	Ø1 ØØ EC	LXI B, ECØØ
34Ø9	CD 47 F2	CALL F247
34ØC	C9	RET
35ØØ	21 ØØ 5Ø	LXI H, 5ØØØ
35Ø3	11 FF 53	LXI D, 53FF
35Ø6	Ø1 ØØ EC	LXI B, ECØØ
35Ø9	CD 47 F2	CALL F247
35ØC	C9	RET

Všimněte si, že koncová adresa přesunované oblasti se od počáteční liší o 3FF /hexadecimálně/, což je právě počet paměťových míst oblasti VIDEORAM.

Na základě strojových programů, uvedených v XI.2. můžeme zajistit periodické střídání obrazovky zaplněné jedničkami a dvojkami. To provádí následující obslužný program v jazyce BASIC:

```
1Ø CALL HEX(3ØØØ)
2Ø CALL HEX(31ØØ)
3Ø CALL HEX(32ØØ)
4Ø CALL HEX(33ØØ)
5Ø WAIT(5)
6Ø CALL HEX(35ØØ)
7Ø WAIT(5)
8Ø CALL HEX(34ØØ)
9Ø GOTO 5Ø
```

Příkaz na programovém řádku 1Ø naplní obrazovku jedničkami, příkaz na řádku 2Ø takto zaplněnou obrazovku uschová do paměti od hexadecimální adresy 5ØØØ, příkaz na řádku 3Ø zaplní obrazovku dvojkami, příkaz na řádku 4Ø uschová toto obsazení do paměti od hexadecimální adresy 6ØØØ. Význam programových řádků 5Ø a 7Ø je samozřejmý. Příkaz na řádku 6Ø vyvolá uschovanou obrazovku zaplněnou jedničkami zpět na VIDEORAM, příkaz na řádku 8Ø vyvolá uschovanou obrazovku zaplněnou dvojkami zpět na VIDEORAM. Příkaz na řádku 9Ø způsobí periodické střídání přesunů obou uschovaných obrazovek na oblast VIDEORAM.

Je samozřejmé, že pomocí těchto metod nemusíme uschovávat do paměti pouze obrazovky ze stejných znaků, ale obrazovky zaplněné libovolným textem, výpočty nebo obrázkem.

XI.2.2. Úschova programu v jazyce BASIC do jiné části paměti

Víme, že záznam programu v jazyce BASIC začíná od hexadecimální adresy 16A a končí na hexadecimální adrese, uložené na paměťových místech s hexadecimálními adresami DØ a D1 - na D1 jsou vyšší dva řády, na DØ nižší dva řády adresy.

Servisní strojový program, který umožní přesunutí našeho programu v jazyce BASIC, bude vložen od hexadecimální adresy 3ØØØ, strojový program, který mám "uschovaný" program v jazyce BASIC vrátí na původní místo v paměti počítače, bude vložen od hexadecimální adresy 31ØØ. Program v jazyce BASIC budeme uschovávat do úseku paměti, který začíná hexadecimální adresou 5ØØØ. / S výhodou se někdy takový program přesouvá do oblasti USR vhodně zvoleného rozsahu./

Pro zjednodušení budeme přesouvat obsah větší části paměti, konkrétně od adresy Ø až do adresy konce programu. Na

hexadecimálních adresách 5ØØØ a 5ØØ1 uložíme koncovou adresu přesunutého programu, abychom ho mohli snadno přesunout zpět. Celý program pro uschování programu je následující:

adresa	kód	instrukce	význam
3ØØØ	2A DØ ØØ	LHLD ØØDØ	vložení koncové adresy programu do HL
3ØØ3	11 ØØ ØØ	LXI D, ØØØØ	vložení počáteční adresy do DE
3ØØ6	EB	XCHG	výměna obsahů HL a DE
3ØØ7	Ø1 Ø2 5Ø	LXI B, 5ØØ2	vložení adresy úseku, na který se přesouvá
3ØØA	CD 47 F2	CALL F247	volání podprogramu
3ØØD 3Ø1Ø 3Ø13	2A DØ ØØ Ø1 Ø2 5Ø Ø9	LHLD ØØDØ LXI B, 5ØØ2 DAD B	výpočet koncové adresy přesunutého programu
3Ø14	22 ØØ 5Ø	SHLD 5ØØØ	uložení koncové adresy posunutého programu na místa s adresami 5ØØØ a 5ØØ1
3Ø17	C9	RET	konec

Strojový program, který "vrátí" náš "uschovaný" program na původní místo, bude vložen od hexadecimální adresy 31ØØ a bude poněkud kratší, protože odpadá výpočet koncové adresy posunutého programu a její ukládání na vyhrazená místa v paměti počítače. Program je následující:

adresa	kód	instrukce	význam
31ØØ	2A ØØ 5Ø	LHLD 5ØØØ	koncová adresa vložená do HL
31Ø3	11 Ø2 5Ø	LXI D, 5ØØ2	vložení počáteční adresy do DE
31Ø6	EB	XCHG	výměna obsahů HL a DE
31Ø7	Ø1 ØØ ØØ	LXI B, ØØØØ	adresa, k níž bude úsek přesunut, vložená do BC
31ØA	CD 47 F2	CALL F247	volání podprogramu
31ØD	C9	RET	konec

Máme-li nyní v paměti počítače nějaký program v jazyce BASIC, pak odesláním příkazu

CALL HEX(3ØØØ)

tlačítkem CR v režimu přímého výpočtu se tento program okopíruje do části paměti, začínající na adrese 5ØØØ. Nyní můžeme stisknout tlačítko RES a pracovat se zcela jiným programem v jazyce BASIC. Pokud chceme vyvolat původní program, který je v paměti počítače "uschován", stačí v režimu přímého výpočtu odeslat tlačítkem CR příkaz

CALL HEX(31ØØ)

Místo přehrávání původního programu na magnetofonový pásek a jeho zpětného přehrání do paměti počítače je mnohdy výhodnější použít "úschovy" původního programu, které je podstatně méně náročné na čas, zvláště pokud máme servisní strojové programy v paměti počítače.

Poznámky:

1/ Tímto postupem "schováváme" do paměti počítače nejen program, ale i hodnoty všech systémových proměnných, jak byly nastaveny při chodu uschovavaného programu v jazyce BASIC - všimněte si, že se přesunuje celá oblast paměti od adresy ØØØØ až do konce záznamu programu, nikoliv jen od hexadecimální adresy DØ nebo 16A. Má to tu výhodu, že po návratu uschovaného programu se automaticky nastaví všechny systémové proměnné na hodnoty, které byly při chodu původního programu.

Stačilo by přesunovat oblast paměti od hexadecimální adresy DØ, tím by se však poněkud zkomplikovaly oba servisní strojové programy zvláště tím, že by se musela poněkud složitějším způsobem počítat koncová adresa posunutého programu - ta je nutná pro zpětný přesun programu na původní místo.

2/ Při přesunování konkrétního programu v jazyce BASIC je nutno vždy rozvážit, jaký je jeho rozsah a k jaké adrese je ho možno přesunout, aby přesunutý program nezasáhl program nepřesunutý, případně aby přesunutý program se nedostal do oblasti trvale obsazené překladačem programovacího jazyka. V naší ukázce jsme přesouvali k hexadecimální adrese 5ØØØ, pokud uživatel potřebuje realizovat přesun k jiné adrese, musí odpovídajícím způsobem servisní programy upravit. Rovněž musí provádět jejich úpravu, pokud chce servisní programy umístit od jiných adres, než 3ØØØ a 31ØØ v příkladu uvedených - viz adresy ve volacích příkazech CALL HEX ...

XI.3. Servisní program pro zpětné vyvolání programu v jazyce BASIC po stisku tlačítka RES

Víme, že tlačítko RES znehodnotí program v jazyce BASIC tak, že zruší platné údaje na paměťových místech s hexadecimál-

ními adresami DØ, D1, 16A a 16B. Sestavíme nyní jednoduchý servisní strojový program, který nám správné obsazení všech čtyř adres obnoví, takže s programem v jazyce BASIC můžeme ihned pracovat dále i po stisku RES.

Servisní program vložíme třeba od hexadecimální adresy 6ØØØ a bude následující:

adresa	kód	instrukce	význam
6ØØØ	2A DØ ØØ	LHLD ØØDØ	naplnění HL obsahem pam. míst DØ a D1
6ØØ3	22 ØD 6Ø	SHLD 6ØØD	uložení obsahu HL na pam. místa 6ØØD a 6ØØE
6ØØ6	2A 6A Ø1	LHLD Ø16A	naplnění HL obsahem pam. míst 16A a 16B
6ØØ9	22 ØF 6Ø	SHLD 6ØØF	uložení HL na 6ØØF a 6Ø1Ø
6ØØC	C9	RET	konec
6ØØD 6ØØE 6ØØF 6Ø1Ø	ØØ ØØ ØØ ØØ	NOP NOP NOP NOP	rezervovaná místa pro ukládání adres
6Ø11 6Ø14 6Ø17 6Ø1A	2A ØD 6Ø 22 DØ ØØ 2A ØF 6Ø 22 6A Ø1	LHLD 6ØØD SHLD ØØDØ LHLD 6ØØF SHLD Ø16A	zpětné přesuny obsahů pam. míst 6ØØD až 6Ø1Ø na místa s adresami DØ, D1, 16A a 16B
6Ø1D	C9	RET	konec

Servisní program používáme takto:
Do paměti vložíme program v jazyce BASIC a v režimu MONITOR uvedený servisní program. Pak v režimu BASIC odesláním příkazu

CALL HEX(6ØØØ)

tlačítkem CR nebo v režimu MONITOR pomocí C6ØØØ uložíme na předem rezervovaná paměťová místa s hexadecimálními adresami 6ØØD až 6Ø1Ø hodnoty z adres, na něž působí tlačítko RES. Nyní již pracujeme s programem v jazyce BASIC a pokud nás okolnosti donutí k tomu, abychom stiskli tlačítko RES, pak program v jazyce BASIC vyvoláme zpět pomocí příkazu

CALL HEX(6Ø11)

v režimu BASIC, případně

C6Ø11

v režimu MONITOR.

Poznámky:

1/ Servisní programy tohoto typu je možno ukládat do části paměti RAM, která je běžně nevyužitá /pokud nejsou připojeny některé další periferie/, konkrétně od hexadecimální adresy 1ØØ do adresy, na níž začíná záznam programu v jazyce BASIC - tedy 16A. Pokud uživatel vloží servisní program do této části paměti, musí odpovídajícím způsobem upravit příslušné adresy v jednotlivých instrukcích strojového programu - konkrétně za první a druhou instrukcí SHLD a za třetí a čtvrtou instrukcí LHLD. Změní se samozřejmě i volací adresy obou částí servisního programu.

2/ Pokud umístíme servisní program do uvedené oblasti, má to následující výhody:

a/ Přehráváme-li náš program v jazyce BASIC pomocí příkazu

W DØ, ... , 1ØØ

v režimu MONITOR /servisní program začíná na adrese 1ØØ/, pak se přehrává automaticky i servisní program, který je mezi hexadecimálními adresami DØ a 16A. Druhý parametr v příkazu W , vyznačený tečkami, je v konkrétním případě programu v jazyce BASIC koncovou adresou jeho záznamu v paměti.

b/ Při výše uvedeném způsobu nahrávky na magnetofonový pásek bude při zpětné přehrávce do paměti počítače automaticky spuštěn servisní strojový program, který uschová kopie obsahů hexadecimálních adres DØ, D1, 16A a 16B na vymezená místa - uživatel tedy nemusí startovat tuto první část servisního programu zvlášť. Je samozřejmé, že zpětná nahrávka do paměti počítače se provádí pomocí příkazu L v režimu MONITOR.

Poznámka ke kapitole XI.

Připomeňte si ještě jednou triviální skutečnost, že polohy programu v jazyce BASIC, servisních programů a případně přesunutých programů nesmí v paměti počítače incidovat. Jinak dojde k havarii obou incidujících programů. Při volbě umístění servisních programů v paměti počítače musíte odpovídajícím způsobem upravit i všechny údaje, které jsou na tomto umístění závislé /adresy apod./

XII. Spolupráce s magnetofonem

V této kapitole si podstatným způsobem rozšíříme své znalosti z oblasti spolupráce jedné základní periferie - magnetofonu s počítačem IQ 151. Uvidíme, že magnetofon nenahrává pouze programy, ale také umí přehrát hodnoty číselných a řetězcových proměnných, dokáže program v jazyce BASIC automaticky spustit, kreslit v průběhu přehrávky na obrazovku apod.

XII.1. Základní informace

V této kapitole hraje podstatnou roli obsazení decimální adresy 234:

- je-li obsazena nulou, přepne výstup počítače na vstup obrazovky;
- je-li obsazena dvojkou, přepne výstup počítače na vstup magnetofonu;
- je-li obsazena decimálním číslem 128, přepne výstup magnetofonu na vstup počítače a čtená data se budou průběžně zapisovat na obrazovku;
- je-li obsazena decimálním číslem 129, přepne výstup magnetofonu na vstup počítače a čtená data se průběžně zobrazovat nebudou.

Význam decimálních adres 25 a 26:

- příkaz

```
POKE 25, 255 : POKE 26, 1
```

nastaví v paměti počítače buffer nezbytný pro čtení z magnetofonu. Toto nastavení se ruší tlačítkem RES nebo při chybě čtení, provede se automaticky při MLOAD.

Význam portu s decimální adresou 137:

- pokud je na něm decimální číslo 159, inicializuje počítač opět na vstup z klávesnice.

Podprogram začínající na decimální adrese 51996, což je hexadecimálně CB1C:
- inicializuje magnetofon k nahrávání.

Podprogram začínající na decimální adrese 51992, což je hexadecimálně CB18:
- inicializuje magnetofon pro čtení z magnetofonového pásku.

XII.2. Nahrávka programu od zvoleného řádku

a/ Je-li provedeno obsazení decimální adresy číslem 2, pak je výstup počítače přepnut na magnetofon a například příkaz LIST s číslem programového řádku neprovede požadovaný výpis na obrazovku, ale provede zápis programových řádků počínaje zvoleným na zapnutý magnetofon. Snadno lze tedy sestavit servisní program, který pořídí zápis programových řádků na magnetofon počínaje řádkem se zvoleným číslem. /Rovněž MSAVE lze použít s parametrem - číslem řádku, od něhož se program nahraje./

```
1Ø REM"NAHRAVANI OD ZVOLENEHO CISLA RADKU"
2Ø CLS
4Ø PRINT " Zapni nahravani a stiskni cerne tlacitko "
5Ø IF INKEY$ = "" THEN 5Ø
6Ø POKE 234, 2
7Ø CALL 51996
8Ø LIST ....
9Ø END
```

Program umístíme do paměti před jiný program v jazyce BASIC, jehož výpis od zvoleného čísla řádku chceme provádět. Na řádku 8Ø nutno doplnit číslo programového řádku, od něhož chceme

výpis provést. Program startujeme příkazem RUN 1Ø, až se na obrazovce objeví pokyn v důsledku řádku 4Ø, zapneme magnetofon na nahrávání a po asi 5 s stiskneme černé tlačítko. Čekání na tento stisk je důsledkem řádku 5Ø. Řádky 6Ø a 7Ø provádějí příslušné přepnutí výstupu počítače a inicializaci magnetofonu.

b/ Abychom nemuseli vkládat do počítače dříve uvedený servisní program, je možno použít některého z následujících složitějších příkazů, které odesíláme v přímém režimu tlačítkem CR.

```
POKE 234, 2 : CALL HEX(CB1C) : LIST ....
```

nebo

```
POKE HEX(EA), 2 : CALL HEX(CB1C) : LIST ....
```

Protože podprogram jazyka BASIC, který provádí příkaz LIST začíná na hexadecimální adrese CF36, přičemž číslo programového řádku musí být vloženo do dvojitého registru DE, je možno použít ještě tohoto složitého příkazu v přímém režimu:

```
POKE HEX(EA), 2 : CALL HEX(CB1C) : CALL HEX(CF36), ....
```

Upozornění:

Čtyři tečky na konci každého ze tří uvedených příkazů zastupují konkrétní decimální číslo programového řádku obdobně, jako tomu bylo v XII.2.a/. Uvědomte si ještě, že

```
HEX(EA)  = 234
HEX(CB1C) = 51996
```

c/ Výpis programu v jazyce BASIC počínaje zvoleným programovým řádkem lze rovněž na základě předchozího výkladu zajistit pomocí krátkého strojového programu, který vložíme do paměti počítače třeba od hexadecimální adresy 1ØØ. Program je následující - komentářů již není třeba:

adresa	kód	instrukce	význam
1ØØ	3E Ø2	MVI A, Ø2	vložení čísla 2 na místo
1Ø2	32 EA ØØ	STA ØØEA	s hex. adresou EA
1Ø5	CD 1C CB	CALL CB1C	volání podprogramů
1Ø8	CD 36 CF	CALL CF36	
1ØB	C9	RET	konec

Tento strojový program lze volat z režimu BASIC příkazem CALL HEX(1ØØ), ...

Čtyři tečky mají již dříve uvedený význam.

Upozornění:

1/ Zopakujte si význam parametrů za příkazem CALL v jazyce BASIC.

2/ Pokud používáte některého ze způsobů uvedených v b/ a c/, nutno před odesláním složitých příkazů v režimu přímého výpočtu již zapnout nahrávání magnetofonu, protože zápis na magnetofon začíná okamžitě po jejich odeslání tlačítkem CR.

XII.3. Zápis dat na magnetofon

Přepneme-li výstup počítače pomocí obsazení decimální adresy 234 číslem 2 na magnetofon, pak rovněž příkaz PRINT nepíše výsledek nějaké číselné nebo řetězcové operace na obrazovku, ale na magnetofonový pásek. Tímto způsobem je možno ve formě magnetofonového záznamu uchovávat různá data. Celý proces nám ukáže následující jednoduchý příklad.

```
 5 I = 1Ø.56
1Ø POKE 234, 2
15 CALL 51996
```



```
2Ø PRINT I
25 END
```

Programový řádek 5 simuluje nějakou číselnou hodnotu zavedenou pod identifikátorem, vyskytujícím se i za příkazem PRINT. Řádky 1Ø a 15 provádějí příslušná přepnutí /viz XII.1./, programový řádek 2Ø provádí zápis hodnoty proměnné I na magnetofon.

Postupujeme takto:

Zapneme nahrávání magnetofonu, předtím si připravíme na obrazovce příkaz RUN a po asi 5 s nahrávání pilotního kmitočtu stiskneme tlačítko CR. Po ukončení nahrávky se na obrazovce objeví READY a je nutno zastavit nahrávání magnetofonu.

XII.4. Čtení zápisu dat z magnetofonu

Nyní si ukážeme opačný postup, kdy do počítače zavedeme hodnotu proměnné, zaznamenané na magnetofonovém pásku. Před vstupem údaje z magnetofonu /který se bude realizovat prostřednictvím slova INPUT/, je nutno provést:

- přepnutí výstupu magnetofonu na vstup počítače pomocí příkazu POKE 234, 128;
- vyvolat inicializaci čtení magnetofonu příkazem CALL 51992;
- připravit v počítači vstupní buffer pomocí příkazu POKE 25, 255 : POKE 26, 1 .

Ukazkový program bude následující /vložíte ho do počítače po jeho zapnutí, abyste měli jistotu, že počítač všechny dříve zavedené proměnné "zapomněl"/:

```
1ØØ POKE 234, 128
15Ø CALL 51992
2ØØ POKE 25, 255 : POKE 26, 1
```

```
25Ø INPUT E
29Ø END
```

Programové řádky 1ØØ až 2ØØ provádějí přepnutí a přípravu počítače pro čtení z magnetofonu, řádek 25Ø pak čtení realizuje.

Postup:

Do počítače vložíme tento program, na obrazovku napíšeme příkaz RUN a spustíme přehrávání magnetofonu. Jakmile se ozve pilotní tón před magnetickým záznamem nějaké číselné proměnné, stiskneme tlačítko CR. Ukončení přehrávky signalizuje počítač hlášením READY. Pomocí příkazu PRINT E v přímém režimu se můžeme přesvědčit, že hodnota zavedená pod identifikátorem E souhlasí s hodnotou, kterou jsme dříve na magnetofon nahráli. Všimněte si současně, že hlášení READY nastaví opět "běžný" režim práce počítače a také toho, že při přehrávání lze vlastně číselnou hodnotu přejmenovat /zavést pod jiným identifikátorem/, než pod kterým byla nahrána na magnetofon.

Poznámky:

1/ Změníte-li řádek 1ØØ předchozího programu na tvar

```
1ØØ POKE 234, 129
```

pak při popsaném způsobu přehrávání hodnoty proměnné zpět do počítače se načítaná hodnota nebude na obrazovce během chodu magnetofonu objevovat.

2/ Analogickým způsobem lze nahrávat hodnoty řetězcových proměnných, nutno pouze používat za příkazy PRINT a INPUT stringových identifikátorů.

3/ Pokud chceme nahrávat a přehrávat více proměnných, jak číselných, tak i řetězcových, můžeme buď v programech používat opakujících se příkazů PRINT, resp. INPUT, nebo dát za oba příkazy více identifikátorů, případně čísel a řetězců takto:

- úprava nahrávacího programu

```
2Ø PRINT 2Ø.32 " , " I " , " " AHOJ "
```

- úprava přehrávacího programu

```
25Ø INPUT E, F, C$
```

Všimněte si důležité věci, že aby příkaz PRINT souhlasil syntakticky s příkazem INPUT, jsou v příkazu PRINT v úvozovkách oddělovací čárky. Slovo AHOJ je řetězec, musí být rovněž v uvozovkách a proto jsou v horním řádku dvě uvozovky těsně vedle sebe.

XII.5.Úprava programu pro nahrávání

V konkrétních složitých programech v jazyce BASIC je nutno, aby se chod programu zastavil v okamžiku, kdy jsou požadované údaje pro přehrávku již vypočteny a připraveny. Počítač dá o této skutečnosti uživateli vědět napsáním pokynu pro spuštění nahrávání magnetofonu a čeká na stisk černého tlačítka. Ukázkový program tohoto typu je následující - uvádíme ho již bez komentáře:

```
 5 I = 1Ø.345
 8 PRINT " Zapni nahravani, stiskni tlacitko "
 9 IF INKEY$ = "" THEN 9
1Ø POKE 234, 2
15 CALL 51996
2Ø PRINT I
21 POKE 234, Ø
22 PRINT " Konec nahravky "
```

```
23 PRINT 1
24 GOTO 23
```

Poznámky:

1/ Všimněte si, že příkaz na řádku 21 provádí zpětné přepnutí tak, aby PRINT psalo opět na obrazovku.

2/ Řádky 23 a 24 zastupují nějaké další pokračování programu.

3/ Všimněte si, že program po ukončení nahrávky nejde zastavit tlačítkem CTRL. Funkce tlačítka CTRL se obnoví, pokud do programu po ukončení nahrávky dáme příkaz POKE 254, Ø třeba tak, že řádek 22 upravíme na tvar:

```
22 PRINT " Konec nahravky " : POKE 254, Ø
```

4/ Nahrávání magnetofonu v uvedeném případě zapínáte až se na obrazovce objeví příslušný pokyn v důsledku programového řádku 8.

XII.6. Úprava programu pro přehrávání

Analogické uživatelské úpravy popsané v předchozím článku lze provésti i u programů pro přehrávání dat z magnetofonu do paměti počítače. Program uvedeme proto již bez podrobného komentáře.

```
 8Ø PRINT " Zapni prehravani a stiskni tlacitko "
 9Ø IF INKEY$ = "" THEN 9Ø
1ØØ POKE 234, 128
15Ø CALL 51992
2ØØ POKE 25, 255 : POKE 26, 1
25Ø INPUT E
255 POKE 234, Ø
257 OUT 137, 159
```



```
26Ø POKE 254, Ø
263 PRINT " Vypni magnetofon a stiskni tlacitko "
265 PRINT 1
27Ø GOTO 265
```

Všimněte si, že programové řádky 255 až 26Ø přepínají počítač do "běžného" režimu. Programové řádky 257 a 26Ø zapojují tlačítka klávesnice, mají však ještě ten důsledek, že počítač po ukončení nahrávky automaticky čeká na stisk černého tlačítka, pak teprve pokračuje v práci podle části programu uvedené za přehrávkou. Nemusí se tedy explicitně uvádět čekací řádek se slovem INKEY$, jako při startu přehrávky.

XII.7. Ukázka přehrávek dat s návěštím

Pro náročnější čtenáře uvádíme ještě další vylepšení nahrávacích a přehrávacích programů, kdy jednotlivé sady dat na magnetofonovém pásku je možno označit vhodným řetězcem, který slouží jako návěští /nikoliv hlasové/. Při zpětném přehrávání do paměti počítače je nejdříve toto návěští porovnáno se slovem, které uživatel vloží z klávesnice do počítače a data jsou do počítače přehrána jen tehdy, když je shoda obou řetězců. V opačném případě počítač hlásí, že sadu dat s uvedeným návěštím nenalezl.

Protože z předchozích příkladů máme již dostatečné znalosti, abychom činnost následujícího programu pochopili, uvádíme jen krátce důležité technické údaje pro uživatele.

Následující příklad ilustruje jednoduchou situaci, kdy se na pásek nahrávají čísla 1, 4, 9, .. , 1ØØ a pak se čtou z magnetofonu, přičemž se právě čtené položky zobrazují na obrazovce. Vlastní data následují za návěštím. Přípravné i ukon-

čovací akce jsou řešeny pomocí podprogramů. V nich jsou použity identifikátory O$, OO$ a II. Záznam dat na magnetofon se provádí příkazem RUN nebo RUN 2, zpětná přehrávka se provádí pomocí příkazu RUN 3. Program je následující:

```
2 GOSUB 1ØØØØ:FORI=1TO1Ø:PRINTI*I:NEXTI:GOSUB1ØØ5Ø:END
3 GOSUB1Ø1ØØ:DIMA(1Ø):FORI=1TO1Ø:INPUTA(I):NEXTI:GOSUB1Ø15Ø:END
1ØØØØ REM ZACATEK ZAZNAMU NA MG
1ØØ1Ø INPUT"Zapni mg. na zaznam a zadej navesti";O$
1ØØ2Ø POKE234,2:CALL51996:PRINTO$:RETURN
1ØØ5Ø REM KONEC ZAZNAMU NA MG
1ØØ6Ø POKE 234,Ø:PRINT"Ukoncen zaznam s navestim ";O$;CHR$(7):
      POKE254,Ø:RETURN
1Ø1ØØ REM PREHRAVANI Z MG
1Ø11Ø INPUT"Zapni prehravani a zadej navesti zaznamu";OO$
1Ø12Ø POKE234,128:CALL51992:POKE25,255:POKE26,1
1Ø13Ø INPUTO$:II=II+1:IFII<9THEN1Ø14Ø
1Ø131 PRINT"Zaznam ";OO$;"nebyl nalezen":END
1Ø14Ø IFNOT(OO$=O$)THEN1Ø13Ø
1Ø145 RETURN
1Ø15Ø REM KONEC PREHRAVANI Z MG
1Ø16Ø POKE234,Ø:OUT137,159:PRINT"Ukončeno prehravani ";
      O$;CHR$(7)
1Ø17Ø POKE254,Ø:RETURN
```

XII.8. Přehrávání prvků libovolné matice

V předchozím jsme se naučili přehrávat v obou směrech mezi magnetofonem a počítačem hodnoty libovolných proměnných, ať již číselných nebo řetězcových. Nyní se naučíme přehrávat obdobně hodnoty prvků libovolné matice. Jako příklad si uvedeme přehrávání jednoduché číselné matice 3 x 3.

Program, který přehraje hodnoty prvků matice na magnetofonový pásek, je následující:

```
1Ø A(Ø,Ø) = 1 : A(Ø,1) = 2 : A(Ø,2) = 3
2Ø A(1,Ø) = 4 : A(1,1) = 5 : A(1,2) = 6
3Ø A(2,Ø) = 7 : A(2,1) = 8 : A(2,2) = 9
4Ø POKE 234, 2 : CALL 51996
5Ø FOR I = Ø TO 2
6Ø PRINT I "DATA" A(I,Ø)"," A(I,1)"," A(I,2)
7Ø NEXT I
8Ø END
```

Řádky 1Ø až 3Ø zastupují výpočty jednotlivých prvků matice, na řádku 4Ø jsou známé příkazy pro přepnutí počítače na magnetofon a pro inicializaci přehrávky. Na 5Ø až 7Ø programovém řádku dochází k zápisu hodnot prvků matice na magnetofon vždy po třech tak, aby před každou trojicí bylo slovo DATA a před tímto slovem ještě číslo programového řádku, které bude mít hodnotu od jedné do tří. Všimněte si opět zvláštní polohy uvozovek - jsou mezi nimi oddělovače a slovo DATA.

Pokud je magnetofon zapnut na nahrávání a je-li odeslán příkaz RUN tlačítkem CR, pak se na magnetofonový pásek nezaznamenávají pouze hodnoty jednotlivých prvků matice, ale zaznamenávají se celé programové řádky, které jsou ve tvaru

```
Ø DATA 1, 2, 3
1 DATA 4, 5, 6
2 DATA 7, 8, 9
```

Poznámka:

Pozorný čtenář tento postup jistě sám zobecní, což mu

dá možnost pořizovat na magnetofonový pásek nahrávku libovolného programového řádku nebo jejich sady, které si takto sestaví /viz i následující příklad/.

Nyní vypneme a zapneme počítač, abychom si byli jisti, že se zrušily všechny programy a proměnné v paměti počítače a běžným způsobem pomocí příkazu MLOAD lze tyto "vyrobené" řádky přehrát z magnetofonu do paměti počítače. K nim doplníme tento krátký program:

```
 5 DIM A(2,2)
1Ø FOR I = Ø TO 2
2Ø FOR J = Ø TO 2
3Ø READ A(I,J)
4Ø PRINT A(I,J)
5Ø NEXT J
6Ø NEXT I
7Ø END
```

Řádek 5 deklaruje číselné pole vhodných rozměrů, další řádky zavádějí číselné hodnoty jednotlivých prvků matice v souladu s hodnotami získanými při přehrávce předchozích programových řádků Ø, 1 a 2 do paměti počítače. Řádek 4Ø pouze pro kontrolu vypisuje hodnotu každého zaváděného prvku matice. Program s doplněnými řádky Ø, 1, 2 se startuje obvyklým způsobem pomocí RUN.

XII.9.Rovnání prvků do matice vhodných rozměrů

Jde o modifikaci předchozího případu pouze s tím rozdílem, že na magnetofon nahráváme data, která nebyla předtím indexována a netvořila tedy matici.

Program, který nahraje řádky s číselnými údaji, příkazem DATA a číslem nahrávaného řádku, je následující:


```
1Ø A = 1 : B = 2 : C = 3 : D = 4 : E = 5
2Ø F = 6 : G = 7 : H = 8 : K = 9
4Ø POKE 234,2 : CALL 51996
5Ø PRINT 1 "DATA" A "," B "," C "," D
6Ø PRINT 2 "DATA" E "," F "," G "," H "," K
7Ø END
```

Opět si připomeňte polohu uvozovek v takovýchto případech. Při nahrávce na magnetofon postupujeme opět tak, že připravíme na obrazovku slovo RUN, zapneme nahrávání magnetofonu a po asi 5 s stiskneme tlačítko CR. Programové řádky 1Ø a 2Ø simulují výpočty hodnot jednotlivých proměnných nějakým složitějším programem.

Po skončení chodu programu budou na magnetofonovém pásku záznamy těchto programových řádků:

```
1 DATA 1, 2, 3, 4
2 DATA 5, 6, 7, 8, 9
```

Opět vypneme a zapneme počítač, aby se odstranily všechny proměnné a programy z jeho paměti. Běžným způsobem pomocí příkazu MLOAD nahrajeme tyto dva řádky do paměti počítače a doplníme je následujícími:

```
 5 DIM A(2,2)
1Ø FOR I = Ø TO 2
2Ø FOR J = Ø TO 2
3Ø READ A(I,J)
4Ø PRINT A(I,J)
5Ø NEXT J
6Ø NEXT I
7Ø END
```

Spustíme-li tento kompletní program v režimu BASIC obvyklým způsobem, pak programové řádky 1Ø až 6Ø uloží přehrané hodnoty z řádku 1 a 2 jako prvky matice 3 x 3, příkaz na řádku 4Ø každý prvek pro kontrolu vypíše.

XII.10.Autostart programů v jazyce BASIC

Pokud přehráváme strojový program do paměti počítače v režimu MONITOR, dovedeme zajistit jeho automatický start - připomeneme si pouze význam třetího parametru v příkazu W v režimu MONITOR. Jedná-li se o program v jazyce BASIC, naučíme se nyní provádět takovou jeho nahrávku na magnetofonový pásek, aby se po zpětném přehrání do paměti počítače rovněž automaticky spustil, tedy bez použití příkazu RUN po ukončení nahrávky do paměti počítače.

Pro zjednodušení budeme předpokládat, že program v jazyce BASIC má programový řádek Ø - v tomto případě je autostart jednoduchý. Postup si ukážeme na následujícím příkladu. Mějme následující jednoduchý program:

```
 Ø CLS
1Ø PRINT 1 ;
2Ø PRINT 2 ;
3Ø PRINT 3 ;
4Ø PRINT 4
5Ø END
```

Celý program je opět psán bez mezer mezi příkazem a číslem, případně příkazem a číslem řádku a číslem a středníkem. Po vložení programu do paměti počítače přejdeme do režimu MONITOR tlačítkem BR a zjistíme, že záznam programu v paměti počítače končí na hexadecimální adrese 197. Od následující adresy pomocí příkazu

S v režimu MONITOR vložíme následující servisní strojový program:

adresa	kód	instrukce	význam
198	3E 98	MVI A, 98	obsazení portu s hexadec.
19A	D3 87	OUT 87	adresou 87 hex. číslem 98
19C	C3 68 D1	JMP D168	skok na podprogram, zajišťující autostart programu v jazyce BASIC

Nyní takto doplněný program nahrajeme na magnetofonový pásek v režimu MONITOR pomocí příkazu

W DØ, 19E, 198

- po zpětné přehrávce do paměti počítače v režimu MONITOR pomocí příkazu L bude po ukončení přehrávky automaticky spuštěn servisní strojový program od hexadecimální adresy 198 a tento zajistí autostart našeho programu v jazyce BASIC. Z předchozích příruček opět vyplývá zdůvodnění, proč je nutno přehrávat úsek paměti od hexadecimální adresy DØ, nikoliv jen od 16A.

Poznámky:

1/ Pokud bychom ze servisního strojového programu vypustili dvě první instrukce - MVI A, 98 a OUT 87 - program v jazyce BASIC by se po zpětném přehrání do paměti počítače rozběhl až po stisknutí libovolného černého tlačítka.

2/ Je-li v paměti počítače nějaký program v jazyce BASIC, který má programový řádek Ø, lze ho také spustit /kromě RUN nebo GOTO Ø/ v režimu BASIC příkazem CALL HEX(D168) nebo v režimu MONITOR příkazem CD168.

XII.11. Přehrávka obsazení úseku paměti na magnetofonový pásek v režimu BASIC nebo pomocí strojového programu

V předchozím jsme se naučili přehrávat v obou směrech hodnoty číselných a řetězcových proměnných v režimu BASIC. Nyní budeme v obou směrech přehrávat v režimu BASIC nebo pomocí strojového programu obsazení daného úseku paměti, tedy sadu hexadecimálních čísel tak, jak to provádí v režimu MONITOR příkaz W a L. Věnujeme se nejdříve přehrávce z paměti počítače na magnetofonový pásek. Příkaz W v režimu MONITOR vyžadoval tři parametry:

$$W\ p_1, p_2, p_3 \quad ,$$

kde p_1 je hexadecimální adresa počátku úseku paměti, p_2 je koncová hexadecimální adresa úseku a p_3 je hexadecimální startovací adresa. Funkci příkazu W provádí ve strojových programech podprogram monitoru, který začíná na hexadecimální adrese F267, přičemž hexadecimální adresa počátku úseku paměti musí být předtím vložena do HL, koncová do DE a startovací adresa do BC. Chceme-li rychlonahrávku, dáme rovněž předtím na hexadecimální adresu 1C číslo Ø2.

Jako konkrétní příklad si uvedeme nahrávku oblasti paměti od hexadecimální adresy 4FØØ do 51ØØ. Před dalším postupem si obsaďte tento paměťový úsek nějakými čísly - nejlépe nějakými částmi strojových programů apod, abychom měli kontrolu, že se údaje z této oblasti naším postupem přehrávají.

Od hexadecimální adresy 6ØØØ vložíme nyní následující jednoduchý strojový program. Po jeho vložení **zapneme nejdříve** nahrávání magnetofonu a pak tlačítkem CR odešleme v režimu BASIC příkaz

CALL HEX(6ØØØ) ,

který může být rovněž součástí vhodného programu. Jeho působením se na magnetofonový pásek zaznamená obsazení požadovaného úseku paměti. Případný chod další části programu v jazyce BASIC, která následuje za tímto příkazem, nastane po stisku libovolného černého tlačítka.

adresa	kód	instrukce	význam
6ØØØ	3E Ø2	MVI A, Ø2	vložení čísla Ø2 do A
6ØØ2	32 1C ØØ	STA ØØ1C	uložení Ø2 na adresu 1C
6ØØ5	21 ØØ 4F	LXI H, 4FØØ	naplnění dvojitých registrů
6ØØ8	11 ØØ 51	LXI D, 51ØØ	počáteční, koncovou a start-
6ØØB	Ø1 ØØ ØØ	LXI B, ØØØØ	tovací adresou /program ne-
			bude automaticky spuštěn,
			protože v BC je nula !/
6ØØE	CD 67 F2	CALL F267	volání podprogramu
6Ø11	C9	RET	konec

Zpětná přehrávka z pásku do paměti počítače:

Po opětném zapnutí počítače přejdeme do režimu MONITOR, na obrazovku napíšeme L, zapneme přehrávání magnetofonu a v době, kdy se ozve úvodní pilotní kmitočet před výše pořízenou nahrávkou, stiskneme tlačítko CR. Po ukončení přehrávky se můžeme pomocí příkazu D s vhodnou adresou přesvědčit, že se obnovilo dřívější obsazení úseku paměti mezi paměťovými místy od hexadecimální adresy 4FØØ do 51ØØ.

XII.12. Přehrávka obsazení paměťových míst z magnetofonového pásku do paměti počítače v režimu BASIC nebo pomocí strojového programu

Dalším naším úkolem je realizovat činnost příkazu L, který přísluší režimu MONITOR, pomocí strojového programu, jehož by bylo možno rovněž volat z režimu BASIC obdobně, jako jsme to již dříve udělali pro příkaz W. Připomeneme si, že L znamená totéž, co LØ, obecně za L může být hexadecimální parametr, mající význam popsaný v předchozích příručkách.

Příkaz L je realizován podprogramem monitoru, který začíná na hexadecimální adrese F3BA, přičemž zmíněný hexadecimální parametr musí být umístěn na vrcholu zásobníku, což se nejčastěji provede tak, že se nejdříve vloží do HL a pak se použije instrukce PUSH H. Fungování těchto relací si ukážeme opět na jednoduchém příkladu.

Obsaďte si nejdříve pomocí příkazu S v režimu MONITOR paměťová místa s hexadecimálními adresami od 5ØØ do 5Ø8 po řadě hexadecimálními čísly od 31 do 39 /tj. kódy cifer 1 až 9/. Jde opět jen o určité zaplnění daného úseku paměti pro naši ukázku, proto je konkrétní obsazení nepodstatné. Dále v režimu MONITOR pořídíme nahrávku tohoto obsazení pomocí příkazu

W 5ØØ, 5Ø8, Ø

/Kdybychom jako třetí parametr dali hexadecimální číslo CAD6, zajistili bychom si po zpětné přehrávce automatický návrat do režimu BASIC./

Nyní vypneme a zapneme počítač, aby obsazení uvedených adres bylo zrušeno a v režimu MONITOR vložíme od hexadecimální adresy 6ØØ následující strojový program. Tento strojový program můžeme spustit:

a/ z režimu BASIC příkazem CALL HEX(6ØØ);

b/ z režimu MONITOR příkazem C6ØØ,

ale před odesláním kteréhokoliv z obou příkazů tlačítkem CR za-

pneme přehrávání magnetofonu a teprve, když magnetická hlava začne snímat úvodní pilotní kmitočet, stiskneme CR. Po provedení přehrávky do paměti počítače se pomocí příkazu D5ØØ v režimu MONITOR můžeme přesvědčit, že došlo k obnovení původních obsahů paměťových míst zvoleného úseku paměti.

adresa	kód	instrukce	význam
6ØØ	21 ØØ ØØ	LXI H, ØØØØ	obsazení vrcholu zásob-
6Ø3	E5	PUSH H	níku hexadecim. parametrem
6Ø4	C3 BA F3	JMP F3BA	skok na podprogram

Změníme-li trochu náš strojový program tak, že místo LXI H, ØØØØ bude LXI H, ØØ4Ø , pak upravený strojový program začíná

21 4Ø ØØ .....

a pomocí něho přehrajeme naši nahrávku na paměťová místa od hexadecimální adresy 54Ø do 548.

XII.13. Potlačení kontrolního záznamu na obrazovce při nahrávání programu pomocí servisního strojového programu

V průběhu nahrávání programu do paměti počítače v režimu MONITOR nebo pomocí servisního programu uvedeného v minulém článku se v horní části obrazovky objevuje zápis programu. Ten lze potlačit tak, že poněkud rozšíříme náš nahrávací strojový program, který bude nyní vypadat takto:

adresa	kód	instrukce	význam
6ØØ	21 FF 7E	LXI H, 7EFF	odstranění záznamu
6Ø3	22 19 ØØ	SHLD ØØ19	z obrazovky
6Ø6	CD 47 F6	CALL F647	
6Ø9	3A		parametr

adresa	kód	instrukce	význam
6ØA	3E Ø3	MVI A, Ø3	obsazení portů
6ØC	D3 87	OUT 87	s hexadecimálními
6ØE	3E DF	MVI A, DF	adresami 87 a 89
61Ø	D3 89	OUT 89	
612	21 ØØ ØØ	LXI H, ØØØØ	
615	E5	PUSH H	program pro nahrá-
616	C3 BF F3	JMP F3BF	vání

Z minulého článku máte ještě záznam určitého obsazení paměťových míst na magnetofonovém pásku. Vyčistěte paměť počítače jeho krátkým vypnutím, pak od hexadecimální adresy 6ØØ vložíte výše uvedený strojový program, spustíte přehrávání magnetofonu a až začne být snímán úvodní pilotní kmitočet, odešlete tlačítkem CR v režimu MONITOR příkaz C6ØØ nebo v režimu BASIC příkaz CALL HEX(6ØØ). Po ukončení přehrávky se přesvědčíte, že došlo k požadovanému obsazení paměťových míst a navíc během přehrávání se kontrolní zápis na obrazovce neobjevoval.

XII.14. Přehrávání programů s titulky

Mnohé profesionální programy jsou dělány tak, aby při přehrávání do paměti počítače pomocí příkazu L v režimu MONITOR se nejdříve nahrál a automaticky spustil krátký strojový program, který vymaže obrazovku a potlačí kontrolní záznam programu na ní v další fázi přehrávání. V průběhu dalšího přehrávání se na obrazovce objeví titulek nebo hlavička nahrávaného programu, která na obrazovce zůstává po celou dobu přehrávání. Až je přehrávání programu do paměti ukončeno, program /ať strojový nebo v jazyce BASIC/ je spuštěn autostartem. Ukážeme si nyní na zjednodušeném příkladu, jak může být taková nahrávka programu uděláno. Do paměti počítače vložíme nejdříve vlastní program - vezmeme jednoduchý program v jazyce BASIC, který je uveden v článku XII.10. Jeho záznam v paměti počítače končí na hexadecimální adrese 197. Počínaje hexadecimální adresou 198 vložíme následující řadu servisních programů:

adresa	kód	instrukce	význam
198	3E 98	MVI A, 98	autostart programu
19A	D3 87	OUT 87	v jazyce BASIC
19C	C3 68 D1	JMP D168	
19F	CD 5Ø D6	CALL D65Ø	mazání obrazovky
1A2	3E 6F	MVI A, 6F	pauza
1A4	CD A5 F5	CALL F5A5	
1A7	21 FF 7E	LXI H, 7EFF	potlačení záznamu
1AA	22 19 ØØ	SHLD ØØ19	na obrazovce při
1AD	CD 47 F6	CALL F647	nahrávání programu
1BØ	3A		do paměti počítače
1B1	3E Ø3	MVI A, Ø3	
1B3	D3 87	OUT 87	
1B5	3E DF	MVI A, DF	
1B7	D3 89	OUT 89	
1B9	21 ØØ ØØ	LXI H, ØØØØ	příprava pro nahrávání
1BC	E5	PUSH H	další části programu
1BD	C3 BF F3	JMP F3BF	do paměti počítače
1CØ	54	T	titulek programu, který
1C1	49	I	se má objevit na obra-
1C2	54	T	zovce během nahrávání
1C3	55	U	programu do počítače
1C4	4C	L	

adresa	kód	instrukce	význam
1C5	45	E	dokončení textu
1C6	4B	K	titulku
1C7	21 CØ Ø1	LXI H, Ø1CØ	přesun titulku na
1CA	11 C6 Ø1	LXI D, Ø1C6	obrazovku od adresy
1CD	Ø1 ØC ED	LXI B, EDØC	EDØC
1DØ	CD 47 F2	CALL F247	
1D3	3E 6F	MVI A, 6F	pauza
1D5	CD A5 F5	CALL F5A5	
1D8	21 FF 7E	LXI H, 7EFF	potlačení kontrolního
1DB	22 19 ØØ	SHLD ØØ19	při delším nahrávání
1DE	CD 47 F6	CALL F647	programu do paměti
1E1	3A		počítače
1E2	3E Ø3	MVI A, Ø3	
1E4	D3 87	OUT 87	
1E6	3E DF	MVI A, DF	
1E8	D3 89	OUT 89	
1EA	21 ØØ ØØ	LXI H, ØØØØ	příprava pro nahrávání
1ED	E5	PUSH H	další části programu
1EE	C3 BF F3	JMP F3 BF	do paměti počítače

Nyní si někde ve volné paměti počítače - v našem případě třeba od hexadecimální adresy 5ØØ vytvoříme další servisní program, který nám umožní získat nahrávky na magnetofonový pásek jednotlivých částí takto vytvořeného kombinovaného programu a tyto části postupně a v průběhu zpětné přehrávky startovat. Servisní program pro pořízení záznamu našeho programu včetně doplňujících strojových programů na magnetofonový pásek je následující:

adresa	kód	instrukce	význam
5ØØ	21 9F Ø1	LXI H, Ø19F	přehrávka úseku paměti
5Ø3	11 BF Ø1	LXI D, Ø1BF	mezi hexadecimálními
5Ø6	Ø1 9F Ø1	LXI B, Ø19F	adresami 19F až 1BF
5Ø9	CD 67 F2	CALL F267	s autostartem od 19F
5ØC	3E 7F	MVI A, 7F	pauza
5ØE	CD A5 F5	CALL F5A5	
511	21 CØ Ø1	LXI H, Ø1CØ	přehrávka úseku paměti
514	11 FØ Ø1	LXI D, Ø1FØ	mezi hexadecimálními
517	Ø1 C7 Ø1	LXI B, Ø1C7	adresami 1CØ až 1FØ
51A	CD 67 F2	CALL F267	s autostartem od 1C7
51D	3E 7F	MVI A, 7F	pauza
51F	CD A5 F5	CALL F5A5	
522	21 DØ ØØ	LXI H, ØØDØ	přehrávka úseku paměti
525	11 9E Ø1	LXI D, Ø19E	mezi hexadecimálními
528	Ø1 98 Ø1	LXI B, Ø198	adresami DØ až 19E
52B	CD 67 F2	CALL F267	s autostartem od 198
52E	C9	RET	konec

Nyní pořídíme nahrávku našeho programu na magnetofonový pásek tak, že v režimu MONITOR napíšeme na obrazovku příkaz C 5ØØ , dále zapneme nahrávání magnetofonu a po asi 5 s stiskneme CR. /Nahrávka končí až po objevení se kurzoru!/ Takto získanou nahrávku přehráváme zpět do paměti počítače pomocí příkazu L v režimu MONITOR a vidíme, že po dobu nahrávání je na čisté obrazovce zvolený text nebo obrázek.

Všimněte si také korelace časových pauz, aby počítač byl připraven na přijetí nahrávky další části programu tehdy, když magnetická hlava snímá pilotní kmitočet před příslušnou

částí programu.

XIII. Blokování přehrávek a výpisů

Předpokládáme, že v paměti počítače je určitý program v jazyce BASIC.

a/ Víme, že mezibloková distance je uložena na hexadecimální adrese 1C. Pokud na tuto adresu dáme velké číslo - například EF - pak nejde provádět přehrávka programu z paměti počítače na magnetofonový pásek /respektive je bezcenná, protože ji není možno úspěšně přehrát zpět do paměti počítače/ ani v režimu MONITOR ani v režimu BASIC, pokud neprovedeme vhodné obsazení adresy 1C.

b/ Na hexadecimálních adresách 16A a 16B začíná v paměti záznam prvního programového řádku a konkrétně na těchto adresách je uložena adresa paměťového místa, kde začíná záznam druhého programového řádku. Jestliže na tyto adresy dosadíme hexadecimální číslo 16A pomocí příkazů

```
POKE HEX(16A), HEX(6A)
POKE HEX(16B), HEX(Ø1)
```

v režimu BASIC, pak první programový řádek "odkazuje stále sám na sebe". Důsledek je takový, že program jde běžným způsobem spustit /RUN, GOTO ... /, ale není možno získat na obrazovce výpis programu příkazem LIST /stále se vypisuje pouze první programový řádek/ Takto upravený program nelze ani v režimu BASIC přehrát na magnetofonový pásek, neboť se stále opakovaně nahrává pouze jeho první řádek. Rovněž nejde do programu vložit další programový řádek, ani nějaký řádek z programu vypustit. Pokud se někdo o nahrávku programu v režimu BASIC přesto pokouší, nesmyslné nahrávání stále stejného prvního programového řádku

může přerušit pouze stiskem tlačítka BR, ale chce-li se vrátit do režimu BASIC pomocí příkazu R, zjistí, že počítač tento příkaz nerespektuje, vypne navíc klávesnici a uživateli nezbývá než stisknout RES.

Program jde však bez obtíží přehrát v obou směrech běžným způsobem v režimu MONITOR. K výpisu programu se však ani potom nedostaneme, pokud neprovedeme odblokování. To spočívá ve zpětném správném obsazení hexadecimálních adres 16A a 16B nebo /jednodušeji/ odesláním čísla stále se opakujícího programového řádku tlačítkem CR v režimu BASIC. Tím se uvedený řádek z programu nevymaže, ale pouze se odblokuje výpis programu a možnost jeho přehrávání v režimu BASIC.

Poznámka:

Příkaz POKE HEX(16B), HEX(Ø1) většinou provádět nemusíme při blokování programu, protože hexadecimální adresa 16B bývá číslem Ø1 zpravidla obsazena.

c/ Výše uvedených skutečností je možno využít při zabezpečování daného programu proti kopírování /způsobů existuje však celá řada, toto je jen jednoduchý příklad/. Program je zabezpečen pouze před neodborníkem, odborník dovede většinou každý chráněný program odblokovat a dále s ním pracovat, ať je blokování provedeno tím či jiným způsobem. Jenom pro ukázku tedy uvedeme postup, kdy dochází k zablokování výpisu a přehrávky programu již v průběhu nahrávky originálu do paměti počítače.

Předpokládejme, že máme v počítači program v jazyce BASIC dosud nechráněný a nezablokovaný proti výpisu a přehrávce. Předpokládejme, že konec jeho záznamu v paměti počítače je na hexadecimální adrese 35F,což zjišťujeme známými postupy v režimu MONITOR /případně podle obsahů paměťových míst s adresami

DØ a D1. Za konec záznamu tohoto programu, tedy od hexadecimální adresy 36Ø vložíme v režimu MONITOR tento doplňující servisní program:

adresa	kód	instrukce	význam
36Ø	21 6A Ø1	LXI H, Ø16A	uložení adresy 16A do HL
363	22 6A Ø1	SHLD Ø16A	přesun čísla 16A na adresy 16A a 16B
366	21 1C ØØ	LXI H, ØØ1C	uložení adresy 1C do HL
369	36 EF	MVI M, EF	obsazení adresy 1C číslem EF
36B	CD D6 CA	CALL CAD6	skok do režimu BASIC
36E	C9	RET	konec

Nyní tento upravený program přehrajeme na magnetofonový pásek v režimu MONITOR pomocí příkazu

W DØ, 36E, 36Ø .

Všimněte si, že poslední parametr příkazu způsobí autostart doplněného servisního strojového programu a tento svým chodem provede výše uvedená zablokování a automatický přechod do režimu BASIC.

Nahrávka, kterou takto pořídíme, je již "zapečetěná" v tom smyslu, že kdokoliv ji dostane, může si ji přehrát do svého počítače, ale nemůže ji běžným způsobem přehrát na svůj magnetofonový pásek, pokud neprovede příslušná odblokování. Pořídí-li si však přesto svou nahrávku v režimu MONITOR nebo BASIC, je bezcenná, protože svou nahrávku již zpětně do počítače nenahraje.

Obdobně se dají "pečetit" i programy ve strojovém kódu - princip je z předchozího jasný. Zde však onen "pečetící"

servisní program může být uvnitř strojového programu a lze ho těžko najít.

XIII.1. Blokování startu programu v jazyce BASIC při jeho další přehrávce

a/ Obvykle automaticky předpokládáme, že"logický" začátek programu v jazyce BASIC je na programovém řádku s nejnižším řádkovým číslem. Obecnější případ spočívá v následující struktuře programu:

```
    Ø GOTO 1628
 1ØØØ ..........
  ... ..........
  ... ..........
 1628 "logický"začátek programu
  ... ..........
  ... ..........
  ... ..........
 2ØØØ ..........
```

Pokud takovýto program nahráváme v režimu MONITOR pomocí příkazu W na magnetofonový pásek, umíme již zajistit autostart nahrávaného programu v jazyce BASIC při zpětné přehrávce, protože má nultý řádek. Dále lze zajistit, aby při ukončení zpětné přehrávky do paměti počítače pomocí příkazu L v režimu MONITOR se pak přemazal automaticky příkaz GOTO 1628 třeba na REM AHOJ. /Pozor - musí mít stejný počet paměťových míst !/

To znamená, že pokud uživatel použije originální autorovu magnetofonovou nahrávku programu, program se mu sám spustí. Pokud si ale bude chtít pořídit vlastní kopii programu, na nultém řádku již nemá GOTO 1628, ale pouze REM AHOJ a nemůže již program sám spustit, leda že by jeho rozborem našel jeho logický začátek, což je mnohdy složitější, než obdobný program

samostatně sestavit.

V konkrétním případě se uvedený způsob blokování provádí následovně:

Předpokládáme, že v paměti počítače je nechráněný program v jazyce BASIC, jehož záznam v paměti končí na hexadecimální adrese 55F. Od hexadecimální adresy 6ØØ vložíme tento strojový servisní program:

adresa	kód	instrukce	význam
6ØØ	3E 98	MVI A, 98	autostart programu
6Ø2	D3 87	OUT 87	v jazyce BASIC
6Ø4	C3 68 D1	JMP D168	
6Ø7	C9	RET	konec
6Ø8	8E	8E - REM	
6Ø9	41	41 - A	REM AHOJ
6ØA	48	48 - H	
6ØB	4F	4F - O	
6ØC	4A	4A - J	
6ØD	21 Ø8 Ø6	LXI H, Ø6Ø8	přesun REM AHOJ na místo
61Ø	11 ØC Ø6	LXI D, Ø6ØC	GOTO 1628, které bylo
613	Ø1 6E Ø1	LXI B, Ø16E	od adresy 16E až do 173
616	CD 47 F2	CALL F247	
619	C9	RET	konec

Nezapomeňte, že v původním programu v jazyce BASIC není žádná zbytečná mezera zvláště na nultém řádku, který se zaměňuje. Dále se do zmíněného programu v jazyce BASIC doplňuje na vhodné místo příkaz CALL HEX(6ØD) , který právě způsobí přepsání nultého programového řádku, nebo ještě lépe CALL 1549, c[illegible]e totéž, ale je podstatně méně nápadný, zvláště pro nepříliš zdatného uživatele.

Nyní pořídíme nahrávku tohoto doplněného programu na magnetofonový pásek v režimu MONITOR pomocí příkazu

W DØ, 619, 6ØØ

a při zpětném přehrávání do paměti počítače pomocí příkazu L v režimu MONITOR je náš program chráněn výše popsaným způsobem.

b/ Program v jazyce BASIC lze samozřejmě doplnit i určitými "falešnými" začátky, které se uživateli nabízejí a znesnadňují mu nalezení správného logického začátku programu. Tyto začátky mohou mít následující vlastnosti:

1/ probíhají v nekonečných a nesmyslných cyklech;

2/ způsobí výmaz programu;

3/ způsobí zřícení programu v paměti počítače.

První dva případy jsou jasné nebo byly vyloženy v úvodních kapitolách příručky. Proto se zmíníme jen o třetí možnosti. Jistě se vám již stalo, že jste chtěli volat strojový program nebo podprogram pomocí příkazu CALL .... v režimu BASIC a spletli jste si přesnou adresu začátku tohoto strojového podprogramu. Často byl výsledek takový, že veškeré programy se v paměti počítače "zřítily", paměť byla obsazena pouze střídavě hexadecimálními čísly ØØ a 39. Tohoto efektu využijeme při ochraně programu - jeho falešné začátky obsahují takovéto příkazy CALL... , za nímž je "řítivá" adresa. Snadno si několik vlastních "řítivých" adres najdete sami, když budete zkusmo volat různé adresy v oblasti od HEX(F8ØØ).

XIII.2. Blokování startu strojového programu při jeho další přehrávce

Je to analogie předchozího způsobu, pro strojové pro-

gramy je poněkud jednodušší. Opět "logický" začátek strojového programu lze umístit nikoliv na jeho nejnižší adresu, ale někde uprostřed. Pouze výrobce programu zná jeho správnou startovací adresu a proto může pořídit jeho nahrávku včetně autostartu na magnetofonový pásek. Uživatel může tuto nahrávku kdykoliv do počítače nahrát, spustí se mu sama. Pokud ale pořídí vlastní nahrávku tohoto programu, neví, kde je jeho začátek a tedy mu není nic platná.

Je samozřejmé, že i strojový program jde doplnit "falešnými"začátky včetně vlastností uvedených v předchozím článku. Zvláště účinné jsou i v tomto případě "řítivé" adresy.

Uživatel si však může uvědomit, že startovací adresa se objevuje v závěru nahrávky programu do paměti počítače při kontrolním záznamu na obrazovce. Autor programu může situaci zkoplikovat tak, že kontrolní záznam při nahrávání potlačí - viz minulé kapitoly. Blokování se tedy přenáší na rovinu souboje znalostí autora programu a uživatele.

Seznam volacích adres použitých podprogramů

hexadecimální adresa	význam
CAD6	přechod do režimu BASIC
CB18	čtení dat z mg. pásku v režimu BASIC
CB1C	nahrávání dat na mg. pásek v režimu BASIC
CF36	volání výpisu programu, číslo řádku v DE
D168	autostart programu v jazyce BASIC
D65Ø	mazání obrazovky
F247	příkaz M, počátek v HL, konec v DE, nový začátek v BC
F258	příkaz F, číslo v C, počátek v HL, konec v DE
F267	příkaz W, počátek v HL, konec v DE, startovací adresa v BC
F3BA	příkaz L, adresa v HL a pak PUSH H rovněž i F3BF
F5A5	pauza, v A regulace délky
F8AA	čeká na stisk klávesy, kód tlačítka v A
F8C9	nečeká na stisk tlačítka, kód bílých tlačítek v A
F973	akustický signál nastavený na adresách 17,18 decimálně

Adresy a port používaný v příručce

decimální adresa	význam
25 26	nastavují buffer pro čtení z mg.
234 254	přepínání výstupů a vstupů počítače zapínání funkce tlačítka CTRL

Port o decimální adrese 137 - inicializuje počítač na vstup z klávesnice

☐ Poznámka:

Na závěr si ukážeme, jak je možno najít hexadecimální kód libovolného klíčového slova jazyka BASIC. Hledejme například kódy RUN, NEXT, GOTO.

V režimu BASIC vložíme tento jednoduchý / a na první pohled nesmyslný/program :

```
1 RUN : NEXT : GOTO
```

Program samozřejmě nemůžeme spustit, ale v režimu MONITOR pomocí příkazu D16A zjistíme, že obsazení paměťových míst je následující / po řadě od 16A /:

74 Ø1 Ø1 ØØ 89 3A 82 3A 88 ØØ ....

Odtud bezprostředně vidíme, že hexadecimální kód slova RUN je 89, kód NEXT je 82 a kód GOTO je 88. Za každou dvojtečku v programu je zde hexadecimální číslo 3A.

OBSAH

PROGRAMOVÉ RUTINY IQ 151

Autoři: RNDr. Miloslav Feil, CSc. a kol.

Recenzenti: Dr. Jiří Boltík, CSc.
PaedDr. Jan Kucheř

Schválilo ministerstvo školství České socialistické republiky dne 11. července 1986, č. j. 20 805/86 - 211 pro uživatele IQ 151 na středních školách.

Vydalo Komenium, n. p., Praha, 1986

1. vydání

Odpovědný redaktor: PhDr. Emílie Petráková

Technický redaktor: Martina Mášová

Náklad: 10 000 kusů