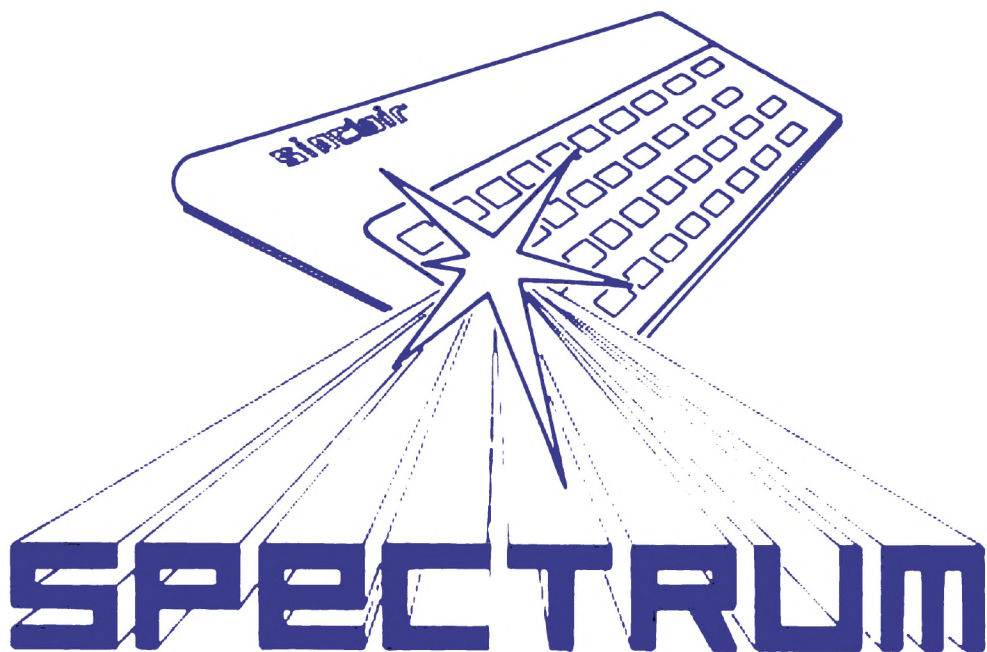




MEMORY RESIDENT SYSTEM

verzia 08E



MEMORY RESIDENT SYSTEM

verzia 08E

UŽIVATELSKÁ PRÍRUČKA MRS

RNDr. Ivan Jedlička, Slavomír Lábsky

O B S A H

1.0	Úvod	5
2.0	Popis práce s jednotlivými modulmi	6
2.1	Spustenie systému	6
2.2	Umiestnenie systému v pamäti	6
2.3	Editor	7
2.3.1	Príkazový režim	7
2.3.2	Obrazovkový režim	15
2.3.3	Písanie riadku	17
2.3.4	Chybové hlásenia	20
2.4	Prekladač zostavovacieho jazyka	21
2.4.1	Spustenie prekladu	21
2.4.2	Vyhodnotenie výrazov	22
2.4.3	Riadenie prekladu pseudoinštrukciami a direktívami	22
2.4.4	Použitie tlačiarne	24
2.4.5	Chybové oznamy	24
2.4.6	Použitie relatívnych skokov	26
2.5	Ladiaci program	26
2.5.1	Úvod	26
2.5.2	Zadávanie hodnoty veličiny pre ladiaci program	27
2.5.3	Príkazy pre ladiaci program	28
2.5.4	Ladenie programov	38
2.5.5	Niektoré dôležité adresy a systémové premenné	40
3.0	Príklad použitia systému MRS v praxi	42

1.0 ÚVOD

MRS (memory resident system) je programový prostriedok určený na vývoj programového vybavenia v zostavovacom jazyku mikroprocesora Zilog 80. Poskytuje všetky bežné funkcie bez toho, aby využíval vonkajšie zariadenie typu pružný disk, a preto je obzvlášť vhodný na prácu na malých osobných počítačoch. Jeho použitie je však opodstatnené i na výkonnejších systémoch, pretože práca s ním je veľmi pohodlná a rýchla.

Systém sa skladá z nasledujúcich modulov:

- **editor (edi)** - je určený na písanie a opravovanie zdrojových programov v zostavovacom jazyku mikroprocesora Z80
- **zostavovací program (asm)** - prekladá zdrojový text pripravený editorom do binárnych modulov a tieto ukladá do knižnice
- **ladiaci program (dbg)** - slúži na ladenie programov
- **spätný prekladač (dis)** - vytvára z binárneho kódu zdrojový text, ktorý je možné ďalšími modulmi spracovať

Systém je napísaný v zostavovacom jazyku mikroprocesora Zilog 80 a realizovaný na osobných počítačoch MO8X. Pri návrhu bol však kladený veľký dôraz na prenositeľnosť systému do iného technického prostredia a už bol realizovaný prenos na domáci počítač ZX Spectrum, kde boli doplnené mnohé nové funkcie a tým vznikla verzia 08.

Ďalší opis systému sa týka tejto konkrétnej implementácie.

2.0 POPIS PRÁCE S JEDNOTLIVÝMI MODULMI

2.1 SPUSTENIE SYSTÉMU

Systém sa nahrá v basicu príkazom

```
load ""
```

a riadenie sa odovzdá príkazovému režimu editora.

Po nahratí sa vykoná studený štart systému od adresy 54876. Pritom sa vyčistí oblasť pamäte pre zdrojový text, nastaví sa niektoré dôležité systémové premenné a inicializuje sa zoznam ilegálnych inštrukcií. Ak systém spustíme od adresy 54885, tak sa obsah systémových premenných nemení (warm start). Túto druhú adresu môžete použiť, ak ste v basicu a chcete skočiť do systému bez toho, aby sa vám vymazal zdrojový text.

2.2 UMIESTNENIE SYSTÉMU V PAMÄTI

Systém využíva pamäť od #c800 do #ff00. Priestor od adresy #c7ff smerom dole sa využíva na zdrojový text užívateľa. Užívateľský stack pointer je pri spustení systému inicializovaný na hodnotu #ff40. Teda ak užívateľovi v jeho vlastných programoch nestačí 64 bytov (#ff00-#ff40) na stack, musí si inicializovať stack pointer sám.

Na požiadanie môžu byť vytvorené i verzie umiestnené v pamäti na iných adresách, ale len v oddôvodnených prípadoch.

Na adrese #fe00 je pracovný buffer systému, ktorý používa aj užívateľ a v ďalšom texte sa odvolávky na premennú bufer týkajú tejto adresy.

2.3 EDITOR

2.3.1 PRÍKAZOVÝ REŽIM

Editor umožňuje prácu v dvoch režimoch: príkazovom a obrazkovom. Po vyvolaní je v režime príkazovom, ktorý je indikovaný výpisom

edi>

do posledného riadku obrazovky. Tento riadok je rezervovaný pre styk užívateľa so systémom a v ďalšom texte sa nazýva dialógový riadok. Užívateľ píše príkazy do dialógového riadku bežným spôsobom, pričom na opravu preklepov využíva tieto špeciálne klávesy:

Šipka vľavo - spôsobí presun jazdca na predchádzajúci znak

Šipka vpravo - spôsobí presun jazdca na nasledujúci znak

Graph/ins - vloží medzeru na pozíciu označenú jazdcom a zvyšok textu posunie doprava

Delete - vymaže znak označený jazdcom a zvyšok textu posunie doľava. Ak je jazdec za posledným znakom riadku, tak sa posunie doľava a posledný znak sa vymaže.

Enter - ukončí zadávanie textu

Všetky znaky, které systém MRS akýmkoľvek spôsobom spracúva, musia byť napísané malými písmenami. Jedinú výnimku tvorí príkaz INI, ktorý sa píše veľkými písmenami. Systém však pozná aj veľké písmená, ktoré užívateľ môže použiť napríklad na poznámky alebo ako textové konštanty (pozri modul edi).

Pri písaní textu sa sledujú hranice riadku a pokus o ich prekročenie, ako aj stlačenie nedefinovanej špeciálnej klávesy oznámi zvukovým signálom. Dialógový riadok má kapacitu 64 znakov.

Všetky príkazy, ktoré editor pozná, sú v nasledujúcej tabuľke (výrazy v zátvorkách sú nepovinné):

asm - prekladá zdrojový text priamo do binárneho kódu v pamäti

dbg - odovzdá riadenie ladiacemu programu

mon - návrat do basicu. Realizuje sa kompletným obnovením zásobníka a skokom na adresu #1303. Je to vhodné vtedy, keď sa zásobník náhodou zmaže a vy stratíte návratové adresy.

mon* - návrat do basicu inštrukciou ret. Zásobník sa neinicializuje. Toto umožní, že basicový program bude po návrate pokračovať ďalším príkazom. Alebo, ak systém MRS zavoláte z nejakého iného systému, po mon* sa vráti do tohto iného systému.

run adresa - spustí užívateľský program od tejto adresy. Ak adresu ne zadáte, platí naposledy zadaná.

INI - inicializuje pracovnú oblasť editora (zmaže zdrojový text) a pripraví ju na vstup nového textu. Obrazovka sa vymaže, jazdec sa nastaví na prvú pozíciu a editor prejde do obrazovkového režimu. ***Pozor! Toto je jediný príkaz editora, ktorý treba písať veľkými písmenami, pretože po vymazaní textu sa tento už nedá vrátiť.***

ln= - posúva ukazovateľ po texte, a síce ln-N posunie ukazovateľ o N ln- riadkov smerom k začiatku textu, ln+N o N riadkov smerom ku koncu ln+ textu a príkaz ln=N nastaví ukazovateľ na n-tý riadok. Príkaz ln-0 nastaví ukazovateľ na prvý riadok, príkaz ln+0 za posledný riadok a príkaz ln=0 na naposledy opravovaný riadok. Príkaz ln= povoľuje ako ďalší parameter tiež refazec znakov. Ukazovateľ sa nastaví na prvý riadok, ktorý v poli návestia obsahuje daný text. Ak taký riadok neexistuje, považuje sa to za chybu a užívateľ môže text opraviť. Ak za príkazom nasleduje žiadny parameter, vezme sa N=0. Vykonaním príkazu sa vypíšu na obrazovku riadky od vyhľadaneho po koniec obrazovky alebo textu, jazdec sa nastaví na prvú pozíciu a editor prejde do obrazovkového režimu (pozri ďalej). Poznámka: N je desiatkové číslo.

dlb M N - zmazanie riadkov s poradovými číslami v intervale M-N včítane

cpb M N - skopírovanie riadkov s poradovými číslami M-N pred riadok, na ktorom bol jazdec pri prechode do dialógového režimu. Ak je tento riadok v intervale M-N považuje sa to za chybu. (pozri tiež obrazovkový režim).

sav meno - slúži na uloženie zdrojového textu na magnetofón. Za príkazom sav ako ďalší parameter nasleduje meno súboru. Od príkazu sav musí byť oddelené jednou medzerou. Meno súboru je ľubovoľný reťazec maximálne 10 znakov, ktorý sa na túto dĺžku v prípade potreby doplní medzerami. Ak meno súboru chýba považuje sa to za chybu. Zaznie zvukový signál a užívateľ môže chybný príkaz opraviť.

Editor sám určí rozsah pamäte, ktorý je potrebné uložiť na pásku; zdrojový text je uložený so štandardnou hlavičkou, ktorá v posledných 2 bytoch má zapísanú hodnotu 65535, podľa čoho editor rozpoznáva zdrojové texty.

Pred odoslaním príkazu klavesom enter je nutné zapnúť magnetofón na nahrávanie. Po nahratí textu (alebo po prerušení nahrávania klavesom break) sa opäť prihlási editor.

svd meno - tento príkaz má rovnakú funkciu ako príkaz sav, ale platí len pre disketovú verziu programu, spolupracujúcu s disketovou jednotkou D40.

mer - slúži na nahranie zdrojového textu z magnetofónu. Za príkazom mer nasleduje meno požadovaného textu, ktoré tvorí reťazec maximálne 10 znakov od príkazu mer oddelený medzerou. Ak meno chýba, nahrá sa prvý modul, ktorý editor na páske nájde. Editor rozoznáva vlastné súbory a prípadný súbor s rovnakým menom, ktorý ale nevytvoril editor, sa ignoruje.

Po odoslaní príkazu užívateľ zapne magnetofón a editor vyhľadá na páske program s daným menom. Počas vyhľadávania vypisuje editor do dialógového riadku informácie o prečítaných moduloch v tvare

meno

kde meno je meno nájdeneho modulu. Ak sa na páske nájde požadovaný modul, za jeho meno sa do dialógového riadku vypíše text

loading

a modul sa nahráva do pamäti. Vyhľadávanie a nahrávanie modulu je možné prerušiť klávesom break. Vtedy sa riadenie vráti editoru.

Ak sa pri nahraní modulu zistí chyba nahrávania, zozbrazí sa posledne zadaný príkaz, takže ak ho chce užívateľ zopakovať, stačí ho už len odoslať enterom.

Ak editor zistí, že sa hľadaný text už do pamäti nezmestí, vypíše do dialógového riadku

mem full

a po stlačení klávesu enter sa riadenie vráti editoru.

Úspešne nahratý text sa ukladá do pamäti pred riadok, na ktorom bol nastavený jazdec vtedy, keď užívateľ prešiel z obrazovkového režimu do príkazového. Toto ukladanie môže trvať rôzne dlho a počas neho je vypísaný v dialógovom riadku text

walt

Po vložení nahratého textu sa opäť prihlási editor výpisom edi.

loa - tento príkaz má rovnakú funkciu ako príkaz mer, len s tým rozdielom, že text nahraný z pásky prepíše text v pamäti. Preto o ňom platí všetko, čo bolo napísané o príkaze mer, len po nahratí textu sa riadenie hneď vráti editoru. Ak hneď za príkazom nebude jedna medzera, bude sa vykonávať príkaz "hea".

ldd meno - tento príkaz má rovnakú funkciu ako príkaz loa, ale platí len pre disketovú verziu programu, spolupracujúcu s disketovou jednotkou D40.

DÔLEŽITÉ UPOZORNENIE! Príkazy *svd* a *ldd* využívajú pri svojej práci interpret *BASICu*, preto pri ich používaní nesmiete zmazať pôvodný zavádzací program v *BASICu*, ani zmazať systémové premenné *BASICu*.

ver - kontrola nahratého zdrojového textu. Tento príkaz sa dá použiť aj na kontrolu ľubovlného bloku bajtov so štandardnou hlavičkou (Bytes:). V prípade chyby to isté, ako pri príkaze "mer".

hea - vypisuje obsahy hlavičiek nájdených na kazete vo formáte skladajúcom sa z týchto častí:

- identifikačný bajt (0=basic, 1=čís.pole, 2=zn.pole, 3=bajty)
- meno súboru (znaky menšie ako 32 sa zobrazia ako '?', väčšie ako 127 sa vypíšu ako '.')
- 1.číslo - dĺžka súboru
- 2.číslo - basic: štartovací riadok
pole: zakódované meno poľa
bajty: začiatková adresa
- 3.číslo - basic: dĺžka programu bez premenných
bajty: pôvod súboru (65535 je zdrojový text)

Jednotlivé časti formátu sú od seba oddelené dvojbodkami.

Pri disketovej verzii programu bol tento príkaz vypustený, ale dá sa plnohodnotne nahradiť príkazom loa (viď príkaz loa).

spd výraz - nastavenie rýchlosti nahrávania pre príkaz sav. Hodnota 0 znamená štandardnú rýchlosť 1500 Bd, hodnota 1 znamená dvojnásobnú rýchlosť 3000 Bd. Príkazy hea, loa, ver a mer sú vybavené dvojrychlostným loaderom, ktorý si sám zistí rýchlosť nahrávky.

ald - špeciálny príkaz, pomocou ktorého užívateľ spustí postupnosť volaní jednotlivých modulov systému a síce asm a dbg. Pokiaľ preklad skončí s chybami volanie ladiaceho programu sa neuskutoční. Bližšie pozri pri popise jednotlivých modulov.

new výraz - nastavenie basicovej systémovej premennej RAM-TOP. Ak za príkazom nie je žiadna hodnota, príkaz vráti riadenie basicu jeho kompletnou inicializáciou. (ako NEW v basicu).

sys výraz - nastavenie systémovej premennej MEMORY. Táto premenná ukazuje, kam sa bude ukladať preložený binárny kód, ak v texte nie je pseudoinštrukcia "org" a zároveň vyhradzuje maximálnu pamäť pre zdrojový text. Celý zdrojový text môže byť

uložený nad hodnotou MEMORY. Ak by ste zadali príliš veľkú hodnotu, systém ju neprijme. Ak nezadáte nič, hodnota MEMORY sa nezmení.

Potom príkaz sys spôsobí výpis týchto systémových premenných:

- RAMTOP (viď príkaz new)
- MEMORY (viď príkaz sys)
- MEMTOP (skutočná adresa, odkiaľ začína zdrojový text)
- LENGTH (dĺžka zdrojového textu)

mod výraz - nastavenie módov práce externých modulov systému. Ak výraz nezadáte, zobrazí sa posledne zadaná hodnota. Pre samotný systém nemá tento príkaz priamy význam.

cls - zmazanie obrazovky. Ak za príkazom nasleduje jedno číslo, nastaví sa ním border a ak nasleduje ešte jedno číslo, bude znamenať hodnotu atribútov na obrazovke. (Je to vhodné najmä vtedy, keď si budete chcieť zmeniť farby písma a podkladu.)

val výraz - zobrazí hodnotu výrazu vo formáte:

XXXX XXXX XXXX XXXX YYYY:ZZZZ

kde X znamená binárny tvar, Y hexadecimálny a Z dekadický.

let návěstie=výraz - priradí návěstiu hodnotu výrazu. Za příkazom musí nasledovať návěstie oddelené jednou medzerou. Zadaná hodnota návěstia platí po najbližší preklad textu (asm) alebo jeho vymazanie (INI).

ref reťazec - hľadá ľubovoľný výskyt reťazca znakov v texte. Reťazec je postupnosť ľubovoľných znakov oddelená od príkazu jednou medzerou. Teda pomocou tohto príkazu je možné hľadať návěstia, inštrukcie, poznámky, ich jednotlivé časti alebo aj celé kombinácie znakov.

Napríklad:

ref call #

hľadá všetky inštrukcie nepodmieneného volania s adresou zadanou hexadecimálnym číslom. Po nájdení príslušného reťazca sa vykoná príkaz `ln=` na riadok, kde sa reťazec našiel. Ak nie je v príkaze reťazec zadaný, hľadá sa ďalší výskyt naposledy zadaného reťazca od riadku, ktorý je za kurzorom.

alt návěstie1 návěstie2 - premenovanie návěstia. Všade tam, kde sa vyskytuje návěstie1 (či už v poli návěstia alebo v poli adresy) sa namiesto neho dosadí návěstie2. Hodnota, ktorú malo návěstie1 sa priradí návěstiu2. Teda návěstie1 celkom prestane existovať. Ak návěstie1 neexistuje alebo návěstie2 už existuje alebo je syntaxne nespravne zadané, systém to považuje za chybu. Užitočné je to vtedy, ak namiesto návěstia "aaa" (ktoré už máte v texte napísané tisíckrát) chcete napísať výstižnejšie "inkey".

dis M N - vytvorenie zdrojového textu z binárneho kódu v pamäti. Číslo M znamená adresu začiatku prekladaného kódu, N je koniec kódu. Vytvorený text sa ukladá pred riadok, na ktorom bol jazdec, rovnako ako keby ho užívateľ písal sám. Ak chceme získať aj hodnoty čítača adres, môžeme takto získaný zdrojový text normálne preložiť a na základe výpisu prekladu doplniť výrazy v poli adresy o návěstia.

Príkaz `dis` má niekoľko rôznych podôb:

`dis` - prekladá kód do tvaru inštrukcií Z80. Kódy, ktoré systém nevie spracovať, sa zapisujú pomocou pseudoinštrukcie `db`.

`dis*` - prekladá kód pomocou pseudoinštrukcie `db`.

`dis:` - prekladá kód pomocou pseudoinštrukcie `dw`.

`dis$` - prekladá kód ako `dis*`, ale všetky texty dá do apostrofov.

Príkaz `dis` sa využíva ak chceme pokračovať v práci na programe, ktorý sme doteraz vytvárali v strojovom kóde (alebo pod iným systémom) pod systémom `mrs`, alebo ak chceme do odladeného programu vložiť niektoré časti systému `mrs`.

Ak sa pri vykonávaní spätného prekladu preplní pamäť, systém vypíše

`mem full`

a riadenie sa po stlačení enteru vráti editoru.

ins M N - toto je špeciálna obdoba príkazu `dis` a slúži na vloženie zdrojového textu, ktorý nebol vytvorený systémom `mrs`.

Za príkazom nasledujú dve konštanty rovnako ako za príkazom `dis`. Prvá konštanta obsahuje adresu užívateľského podprogramu, na hodnotu druhej konštanty je inicializovaný register `hl` pri prvom volaní užívateľského podprogramu. Tento register je k dispozícii užívateľskému modulu a pri ďalších volaniach má hodnotu, ktorú mal pri odchode z podprogramu.

Úlohou užívateľského podprogramu je pri každom zavolaní uložiť do bufra od adresy `#fe00` ďalší riadok zdrojového textu a nasadiť `carry flag = 0`. Ak už nie je k dispozícii ďalší riadok, užívateľský podprogram nasadí `carry flag = 1` a riadenie sa vráti editoru.

Riadky textu sa syntakticky kontrolujú a správne riadky sa ukladajú do textu pred riadok, na ktorom bol jazdec (pozri príkaz `dis`).

Chybné riadky sa zobrazia do prvého riadku obrazovky a v pravom dolnom rohu je ich poradové číslo. Riadky je potom možné opravovať tak, ako je to popísané v odseku 2.3.3. Oprava sa ukončí klávesom `enter`. V prípade chyby zaznie zvukový signál a cyklus sa opakuje. Príkaz `ins` je možné prerušiť klávesami `caps shift/medzera`. Do dialógového riadku sa vypíše

break

a po stlačení klávesu `enter` sa riadenie vráti systému `mrs`. Ak sa pri vkladaní textu preplní pamäť, systém reaguje rovnako ako pri príkaze `dis`. Na kláves `caps shift/medzera` reaguje systém krátko po ukončení opravy klávesom `enter`.

***Poznámka:** Tento príkaz používajú aj externé moduly, ktoré nie sú súčasťou systému MRS.*

Ostatné texty sa považujú za chybné príkazy. V prípade ľubovoľnej syntaxnej chyby je užívateľ na ňu upozornený zvukovým signálom, jazdec sa nastaví na prvý znak príkazu a chybu je možné opraviť.

Niekoľko poznámok:

Čísla v príkazoch `dlb`, `cpb`, `ins`, `dis`, `cls` môžu byť dekadické alebo hexadecimálne a musia byť oddelené od seba ľubovoľným nečíselným znakom. Výraz je postupnosť dekadických a hexadecimálnych čísel alebo návestí (ktoré majú definovanú hodnotu) oddelených od seba znamienkami plus a mínus. Namiesto výrazu sa

môže napísať aj jedno binárne číslo. Binárne čísla začínajú znakom '%' za ktorým je postupnosť jednotiek a núl. Binárne čísla nemôžu byť súčasťou výrazu. Hexadecimálne čísla začínajú znakom '#' za ktorým je postupnosť číslíc 0-9 a malých písmen a-f. Dekadické čísla nemajú žiadny špeciálny začiatkový znak. Návestie je postupnosť max. 6 malých písmen a číslíc začínajúca písmenom.

2.3.2 OBRAZOVKOVÝ REŽIM

Po vykonaní príkazov INI, ref a ln editor prejde do obrazovkového módu. To znamená, že na obrazovku vypíše text od riadku nastaveného príkazom ln alebo ref alebo je obrazovka prázdna (po príkaze INI), jazdec je v ľavom hornom rohu a editor očakáva text.

Pre pochopenie obrazovkového režimu je nutné si uvedomiť, že editor už pripravuje text pre zostavovací jazyk, a to tak, že ho vhodne predspracuje. Preto už pri písaní textu musíme dbať na správny syntax a v ďalšom texte sa preto prelínajú informácie o editore s informáciami o zostavovacom jazyku.

Všeobecný tvar riadku textu je nasledovný:

label ins adr pozn

vidíme, že riadok je rozdelený na 4 polia s týmto obsahom:

label - pole návestia. Začína na nulte pozícii a má dĺžku 7 znakov. Je buď vyplnené medzerami, alebo obsahuje návestie. Návestie môže byť reťazec maximálne 6 písmen a číslíc, ktorý začína písmenom. Zbytok poľa musí byť vyplnený medzerami.

ins - pole inštrukcie. Začína na siedmej pozícii a má dĺžku 5 znakov. Buď je vyplnené medzerami, alebo obsahuje platnú inštrukciu mikroprocesora Zilog 80, prípadne platnú pseudo-inštrukciu zostavovacieho jazyka. Zbytok poľa musí byť vyplnený medzerami.

adr - pole adresy. Začína na 12.pozícii a má dĺžku 20 znakov.

Ak je pole ins vyplnené medzerami, musí byť aj pole adr vyplnené medzerami. Ináč obsahuje platnú adresu pre inštrukciu alebo pseudoinštrukciu zapísanú v poli ins. Pole adresy musí byť zakončené medzerou a zbytok poľa musí byť vyplnený medzerami.

pozn - pole poznámky. Začína na 32.pozícii a má dĺžku 32 znakov. Môže obsahovať ľubovoľné znaky, keďže ho zostavovací jazyk ignoruje. Pole poznámky nesmie byť v riadku, ktorý nemá inštrukciu. Počas písania poznámky sa sleduje dĺžka riadku.

Okrem takéhoto riadku pozná systém riadok poznámky a riadok na riadenie prekladu. Riadok poznámky začína znakom ; a môže obsahovať ľubovoľné znaky, keďže ho prekladač ignoruje. Riadok na riadenie prekladu začína znakom *, jeho obsah sa syntakticky kontroluje. Platné príkazy sú popísané v odseku 2.4.3 a zbytok riadku musí byť vyplnený medzerami.

Tieto riadky nie sú členené na polia a stlačenie kláves True video alebo Inv video v nich sa považuje za chybu (pozri odsek 2.3.3.).

Pole adresy môže obsahovať okrem prvkov predpísaných presne ištrukciou aj prvky, ktoré sa vyhodnocujú počas prekladu, a ktoré nazývame výrazy. Výraz v systéme mrs môže mať jeden z týchto tvarov:

- symbol
- symbol+symbol
- symbol-symbol
- symbol*symbol
- symbol/symbol
- symbol!symbol operácia or
- symbol&symbol operácia and
- symbol@symbol operácia xor
- symbol (záporná hodnota)
- symbol (vyšší byte 2-bytovej konštanty)
- symbol (nižší byte 2-bytovej konštanty)

pričom symbol môže mať jeden z týchto tvarov:

- návestie
- decimálna konštanta, čo je reťazec číslíc
- hexadecimálna konštanta, čo je reťazec číslíc a písmen a-f, pred ktorými je znak #.
- znaková konštanta, čo je ľubovoľný zobraziteľný znak uzavretý v apostrofoch.
- znak \$ - označuje čítač inštrukcií.

okrem inštrukcií môžu byť v poli ins tieto pseudoinštrukcie:

org výraz - riadi čítač inštrukcií

ds výraz - rezervovanie miesta v pamäti pri preklade

db výraz,výraz... - jednobajtové hodnoty

dw výraz,výraz... - dvojbajtové hodnoty

nav. equ výraz - definovanie hodnoty navestia

end - koniec prekladaného textu

Podrobnejšie viď v časti Riadenie prekladu.

2.3.3 PÍSANIE RIADKU

Riadok sa píše bežným spôsobom, pričom sa využívajú tieto špeciálne klávesy (SS = symbol shift, CS = casp shift):

Šipka vpravo - spôsobí presun jazdca na nasledujúci znak, hranice riadku sa kontrolujú.

Šipka vľavo - spôsobí presun jazdca na predchádzajúci znak, hranice riadku sa kontrolujú.

Graph/ins - vloží medzeru na pozíciu označenú jazdcom a zbytok poľa posunie doprava.

Delete - vymaže znak označený jazdcom a zbytok poľa posunie doľava. Ak je jazdec za posledným znakom riadku, posunie sa o jeden znak doľava a posledný znak sa vymaže.

Inv video - posunie jazdca na začiatok ďalšieho poľa, hranice riadku sa kontrolujú.

True video - presunie jazdca na začiatok poľa alebo na začiatok predchádzajúceho poľa, hranice riadku sa kontrolujú.

SS + medzera - nastaví jazdca na posledný znak v poli adresy

medzera - vyplní zbytok poľa medzerami a jazdca nastaví na začiatok ďalšieho poľa. Táto funkcia je potlačená v poli poznámky, v riadku poznámky, alebo keď je medzera súčasťou reťazca alebo znakovkej konštanty. V týchto prípadoch sa medzera normálne uloží do textu.

Ďalšie riadiace znaky spôsobujú ukončenie písania riadku. Riadok sa syntakticky skontroluje. Ak niektoré pole nevyhovuje predpisom, jazdec sa nastaví na jeho začiatok, vydá sa zvukový signál a užívateľ musí riadok opraviť.

Syntakticky správny riadok sa pomerne komplikovane komprimuje a opäť sa prevedie do zdrojovej formy. Potom sa porovná s pôvodným riadkom a pokiaľ sa tieto dva riadky nerovnajú, vypíše sa riadok, ktorý vytvoril systém namiesto pôvodného riadku a zaznie zvukový signál. Ak je užívateľ s novým riadkom spokojný stlačí enter a pokračuje v práci, ináč ho môže opraviť.

Na prvý pohľad komplikovaný systém je pre užívateľa prakticky neviditeľný. Ak sú totiž riadky rovnaké (a to sú vždy, keď je riadok správne napísaný), pokračuje normálne v práci.

Pomocou tohto mechanizmu sa objavlia chyby

- pretečenia (príliš veľká konštanta sa komprimovaním zmení)
- neformalizovaný zápis konštanty (hexadecimálna konštanta musí mať párny počet bytov, pred decimálnou konstantou nesmú byť nuly)
- formálne správne, ale neexistujúce inštrukcie (napr. inštrukcia add hl,ix sa pretvorí na inštrukciu add ix,ix a tým sa zistí chyba)

Systém mrs (moduly edi, asm a dbg) správne spracúva aj niektoré skryté inštrukcie. Sú to jednak inštrukcie sll (operačný znak cb30 až cb37) a inštrukcie ekvivalentné inštrukciám, ktoré pracujú s registrami h a l, ale s prefixom dd resp. fd). Tieto potom pracujú s vyšším alebo nižším bytom indexregistra ix resp iy. Tieto inštrukcie sa zapisujú tak, že na mieste registra h sa píše xh alebo yh a na mieste registra l sa píše xl alebo yl. Nie každá náhrada

napr. registra h registrom xh je možná (napr. ak už inštrukcia má prefix cb tak h sa xh nedá nahradiť). Bližšie o skrytých inštrukciách pozri v odbornej literatúre. Ináč jediná odchylka od štandardného zápisu inštrukcií je, že inštrukcia ex af,af' sa píše bez apostrofu, teda ako ex af,af

V prípade správneho riadku ďalšia akcia závisí od použitého riadiaceho znaku.

Šipka hore - jazdec sa nastaví na začiatok predchádzajúceho riadku. Ak je jazdec na prvom riadku obrazovky, celá obrazovka sa posunie o riadok dole. Ak je jazdec na prvom riadku textu, zaznie zvukový signál a nič sa nevykoná.

Šipka dole - jazdec sa nastaví na začiatok nasledujúceho riadku. Ak je jazdec na poslednom riadku obrazovky, posunie sa celá obrazovka o riadok hore. Ak je jazdec za posledným riadkom textu, ozve sa zvukový signál a nič sa nevykoná.

Edit - vypíše sa predchádzajúca obrazovka textu a jazdec je na začiatku prvého riadku obrazovky.

Caps lock - vypíše sa nasledujúca obrazovka textu a jazdec je na začiatku prvého riadku obrazovky.

SS + enter - vypíše sa nová obrazovka textu tak, že riadok, na ktorom bol jazdec, sa stane prvým riadkom obrazovky

Extend mode - vymaže sa riadok, na ktorom bol jazdec a zvyšok obrazovky sa posunie hore. V tomto prípade sa neanalyzuje syntaktická správnosť riadku. Ak je jazdec za posledným riadkom textu ozve sa zvukový signál a nič sa nevykoná.

CS + enter - za riadok, na ktorom je nastavený jazdec sa vytvorí jeho kopia.

Enter - text na obrazovke počnúc riadkom, na ktorom je jazdec, sa posunie smerom dole, čím sa vytvorí miesto pre vloženie nového riadku. Ak je jazdec na poslednom riadku obrazovky, toto miesto sa vytvorí posunom celého textu o riadok hore. Ak je jazdec za posledným riadkom textu, ozve sa zvukový signál a nič sa nevykoná.

Break - ukončenie práce v obrazovkovom režime a návrat.

Odoslanie iných riadiacich znakov má za následok zvukový signál a nič sa nevykoná. Každý kláves má opakovaciu schopnosť, to znamená, že jeho podržanie má rovnaký výsledok, ako keby sa opätovne stláčal.

2.3.4 CHYBOVÉ HLÁSENIA

Na veľkú väčšinu chybných akcií reaguje editor zvukovým signálom a prípadne nastavením jazdca na chybnú položku. Existujú však dva chybové stavy, následkom ktorých editor vypíše chybové hlásenie do dialógového riadku a po stlačení klávesu enter sa riadenie vráti príkazovému režimu.

mem full - tento výpis znamená, že pamäť, určená na zápis textu sa zaplnila. Nutné je text skrátiť, prípadne nastaviť menšiu hodnotu premennej memory. Prakticky je ale tento stav zriedkavý, pretože veľkosť voľnej pamäti stačí na cca 6000 riadkov zdrojového textu.

Iný dôvod na výpis tohto oznamu je, keď sa zaplní tabuľka symbolov. Táto tabuľka je uložená za textom a dynamicky sa rozširuje. Jej maximálna kapacita je však 255 návestí. Návestia, ktoré sa nepoužijú pri preklade sa automaticky vymažú po skončení prekladu. (pozri popis prekladača zostavovacieho jazyka). Firma Busy soft však pripravuje ďalšiu, deviatu verziu systému MRS, v ktorom bude mať tabuľka návestí kapacitu až 65535 návestí a teda vždy dôjde skôr k preplneniu pamäte ako tejto tabuľky.

2.4 PREKLADAČ ZOSTAVOVACIEHO JAZYKA

2.4.1 SPUSTENIE PREKLADU

Prekladač zostavovacieho jazyka (v ďalšom texte tiež assembler) sa vyvolá príkazom `asm` (alebo `ald`). Assembler v každom prípade sleduje hranice pamäti, ktorá je k dispozícii až po koniec voľnej pamäti (premenná `memtop`).

Assembler prekladá text pripravený editorom do príslušnej pamäti a na obrazovke vytvára výpis prekladu. Riadok výpisu sa vypíše do jedného až dvoch riadkov obrazovky nasledovne:

- | | |
|-----------------|--|
| 1. až 5. znak | -poradové číslo riadku |
| 7. znak | -obsahuje medzeru alebo kód chyby. Chybové kódy sú popísané ďalej. |
| 9. až 12. znak | -hexadecimálny výpis hodnoty čítača adries. |
| 14. až 21. znak | -hexadecimálny výpis kódu |

Druhý riadok obrazovky obsahuje výpis zdrojového textu. Pri výpise na iné výstupné zariadenie sa všetko vypisuje do jedného riadku.

Výpis je možné kedykoľvek zastaviť klávesom `CS+S` a potom opätovne spustiť klávesom `CS+Q`. Po skončení prekladu vypíše assembler informáciu o počte chýb

errors: n

kde `n` je počet zistených chýb.

Pokiaľ sa preložil celý text (to znamená, že pseudoinštrukcia `end` je posledným riadkom textu alebo užívateľ ju vôbec nepoužil), prezrie sa tabuľka symbolov a v preklade nepoužité symboly sa vymažú.

Na záver prekladu, ak bol požadovaný výstup všetkých riadkov príkazom `*a` (pozri odsek 2.4.3), sa vypíše tabuľka použitých návěstí. Návestia sú usporiadané podľa abecedy a za menom návestia nasleduje jeho hodnota ako hexadecimálne číslo.

2.4.2 VYHODNOTENIE VÝRAZOV

Výraz v poli adresy sa vyhodnotí bežným spôsobom. To znamená, že decimálna a hexadecimálna konštanta má hodnotu danú svojim zápisom, znaková konštanta má hodnotu danú hodnotou ascii znaku, ktorý ju reprezentuje a hodnota návestia je daná veľkosťou čítača adres v okamihu, keď sa návestie vyskytlo v poli návestia.

Špeciálne unárne výrazy sú symbol, ktorý definuje hodnotu vyšších 8 bitov zo 16-bitovej hodnoty a symbol, ktorý definuje hodnotu nižších 8 bitov 16-bitovej hodnoty. Napr. ak chceme zistiť, či 1 register sa rovná nižším 8 bitom návestia adresy, možno to vykonať nasledovne:

ld a,1

cp adresa

Symbol \$ nadobúda počas prekladu hodnotu čítača inštrukcií.

2.4.3 RIADENIE PREKLADU PSEUDOINŠTRUKCIAMI A DIREKTÍVAMI

Pseudoinštrukcie riadia preklad zdrojového textu a direktívy riadia vlastnú činnosť prekladača. Pseudoinštrukcie už boli spomenuté v časti venovanej editoru. Tu rozoberieme ich funkciu vzhľadom k assembleru.

org výraz - riadi veľkosť čítača adres. Výraz v poli adresy sa vyhodnotí a jeho hodnota sa priradí čítaču adres. Prvky výrazu musia byť známe už pri prvom prechode. To znamená, že súčasťou výrazu nesmie byť návestie, ktoré sa predtým nevyskytlo v poli návestia.

ds - riadi veľkosť čítača adres. Čítač adres sa zvýši o hodnotu danú výrazom v poli adresy, čím sa vlastne v programe rezervuje pamäť. Podobne ako v prípade pseudoinštrukcie org musí byť hodnota výrazu známa už v prvom prechode.

db - ukladá do pamäti hodnoty určené výrazmi v poli adresy. Tieto hodnoty sa musia zmestiť do 1 byte. Ak je v adresnej časti refazec znakov v apostrofoch, tak sa do pamäti postupne uložia ich ascii hodnoty. Samozrejme je možné ukladať aj záporné hodnoty.

dw - ukladá do pamäti hodnoty určené výrazmi v poli adresy, každá hodnota zaberá dva byty.

equ - priradí návestiu v poli návestia hodnotu danú výrazom v poli adresy. Aj táto hodnota musí byť známa už v prvom prechode. Namiesto equ stačí napísať len jeden znak =.

end - ukončuje prekladanú časť textu, teda text sa prekladá od začiatku po pseudoinštrukciu end. V prípade, že sa táto pseudoinštrukcia v texte nevyskytuje, assembler si ju pri preklade sám dopíše na koniec zdrojového textu. Preto možno povedať, že táto pseudoinštrukcia je nepovinná.

Na riadenie práce prekladača sa využívajú špeciálne riadky, a síce také, ktoré začínajú znakom *. V tomto prípade assembler predpokladá, že za znakom * nasleduje niektorý z týchto príkazov:

a - znamená, že sa má na výstupnom zariadení vytvárať výpis všetkých riadkov. Ak chce užívateľ získať len výpis tabuľky symbolov (pozri odsek 2.4.1), musí *a vložiť pred riadok s pseudoinštrukciou end alebo na posledný riadok, ak end v texte nie je.

e - znamená, že sa má na výstupnom zariadení vytvárať len výpis chybných riadkov. Táto hodnota je preddefinovaná na začiatku prekladu.

l - znamená, že počas prekladu sa vytvára výpis na tlačiareň (pozri bližšie v odseku o použití tlačiarne).

p - má zmysel len v spojení s direktívou l. pomocou direktívy p vyšleme na tlačiareň riadiaci znak nová strana (#0c). Ak za znakom p nasleduje decimálna konštanta, tak sa znak nová strana vyšle vždy po vytlačení daného počtu riadkov.

t - výpis sa bude vytvárať na obrazovku. Táto hodnota je nastavená na začiatku prekladu.

Platnosť direktív začína riadkom s touto direktívou a trvá po výskyte ďalšieho príkazu.

cXXXX - (kde XXXX je hexadecimálna konštanta) znamená, že hoci preklad pokračuje normálnym spôsobom, výsledný kód sa začne ukladať v pamäti od adresy XXXX. Týmto spôsobom možno vytvoriť v pamäti kód, ktorý je schopný práce až po presunutí na patričné miesto v pamäti. Ak XXXX=0, binárny kód sa v pamäti nevytvára vôbec. Tak je možné získať výpis programu bez narušenia pamäti. Platnosť príkazu cXXXX sa zruší pri výskyte pseudoinštrukcie org.

2.4.4 POUŽITIE TLAČIARNE

Ako už bolo spomenuté, výskyt riadku *l spôsobí, že sa začne vytvárať výpis na pripojenej tlačiarňi. Systém je dodávaný bez obslužného programu pre tlačiareň, pretože spôsoby pripojenia tlačiarní k počítaču ZX Spectrum sú veľmi rozmanité. Na adrese #dd77 je však inštrukcia skoku (c3 XX XX). Normálne je v jej adresnej časti skok na inštrukciu ret. Ak si užívateľ napíše vlastný obslužný podprogram pre vytlačenie jedného znaku (a prípadne aj pre inicializáciu tlačiarne) a zodpovedajúcim spôsobom modifikuje adresnú časť inštrukcie tohto skoku, získa možnosť tlačíť výstup z prekladača. Užívateľský obslužný podprogram (na ktorý smeruje skok na #dd77) preberá kód znaku v registri A. Všetky registre môže meniť. Ako riadiace znaky sa vyskytujú kódy #0d (cr), #0a (lf) a #0c (ff - prechod na novú stranu).

2.4.5 CHYBOVÉ OZNAMY

Chybové oznamy možno rozdeliť na dve skupiny. Jednak sú to vážne chyby, ktoré okamžite ukončia preklad a síce:

mem full - pokus o uloženie kódu mimo pamäť vyhradenú preň. Buď je preložený binárny kód príliš dlhý, alebo sa nesprávne použila pseudoinštrukcia org, prípadne direktíva *cXXXX.

Táto chyba vznikne až pri skutočnom pokuse o zápis do pamäti. Vhodným použitím direktívy *cXXXX a následným posunutím binárneho kódu na správne miesto je možné preložiť kód na ľubovoľné miesto pamäti.

prerušenie prekladu - ak užívateľ počas prekladu stlačí break, preklad sa preruší, do dialógového riadku sa vypíše text

break

a po stlačení klávesy enter sa riadenie vráti editoru.

Druhú skupinu tvoria chyby zistené počas prekladu. Tieto chyby sú indikované kódom chyby v znaku pred poľom návestia. Chybné riadky sa zobrazia na obrazovke vždy, nie je možné to potlačiť.

u - v adresnej časti sa vyskytuje návestie, ktoré nebolo definované, teda nevyskytlo sa v poli návestia ani nebolo definované pseudoinštrukciou ext. Návestia v adresnej časti pseudonštrukcií musia byť definované prv, ako sa vyskytnú v adresnej časti, teda musia byť definované v prvom prechode prekladu. Návestie, ktoré bolo raz označené ako nedefinované, je tak označené pri každom výskyte, aj keď bolo prípadne neskôr v texte nájdené v poli návestia.

m - v poli návestia sa vyskytuje návestie, ktoré už bolo definované, teda už sa vyskytlo v poli návestia. Týmto chybovým kódom je označený jeho každý výskyt.

d - v adresnej časti sa vyskytlo návestie, ktoré bolo viackrát definované, teda jeho výskyt v poli návestia bol označený chybovým kódom m. Tento oznam má v podstate informatívny charakter. Uľahčuje užívateľovi opravu chyby m.

r - výraz v adresnom poli inštrukcie, ktorá pripúšťa len 8-bitovú hodnotu, prekročil rozsah 1 bytu (-256 až +255, pri relatívnych inštrukciách -128 až +127).

z - pri použití operácie delenia došlo k pokusu o delenie 0.

Ak nie je protokol o preklade presmerovaný na tlačiareň a pri preklade dôjde ku chybám R alebo Z, zaznie zvukový signál a

automaticky sa skočí do obrazovkového režimu editora na riadok, na ktorom nastala táto chyba, aby ju užívateľ mohol hneď opraviť.

Výraz, pri spracovaní ktorého bola zistená chyba má hodnotu 0.

2.4.6 POUŽITIE RELATÍVNYCH SKOKOV

Pokiaľ sa v adresnej časti inštrukcie relatívneho skoku vyskytne výraz obsahujúci návestie, tak výsledná hodnota sa automaticky upraví. Ak sa vo výraze návestie nevyskytuje, výsledná hodnota výrazu sa priamo zapíše do adresnej časti inštrukcie. Teda napr. inštrukcia `jr label+4` má v adresnom poli hodnotu `label+4-$-2`, kde `$` je hodnota čítača inštrukcií v čase prekladu inštrukcie. Inštrukcia `jr #10` má po preklade v adresnej časti hodnotu `#10`.

2.5 LADIACI PROGRAM

2.5.1 ÚVOD

Ladiaci program sa spúšťa príkazom `dbg` bez parametra. Návrhu modulu `dbg` bola venovaná veľká starostlivosť, pretože ladenie programov je najčastejšia činnosť programátora. Po spustení modulu `dbg` vypíše základnú stavovú informáciu do spodných dvoch riadkov v tvare

X pcpc inštrukcia SZAPC
aa bbcc ddee hhll xhxl yhyl spsp

X - status. status je jeden znak, ktorým je užívateľ informovaný o stave ladiaceho programu, spravidla `x` je medzera

pcpc - hexadecimálna hodnota registra `pc`

inštrukcia vytvorí výpis inštrukcie, ktorá sa má vykonať, v

symbolickom tvare. Hodnota adresnej časti je vyjadrená ako hexadecimálna konštanta. Toto pole je na obrazovke vypísané inverzne.

SZAPC - indikátory (sign, zero, auxiliari cary, parity, carry). príslušné písmeno znamená, že zodpovedajúci indikátor je rovný 1, ináč je na danej pozícii znak -.

aa - hexadecimálna hodnota registra a

bbcc - hexadecimálna hodnota registra bc (v tomto poradí)

ddee - hexadecimálna hodnota registra de (v tomto poradí)

hhll - hexadecimálna hodnota registra hl (v tomto poradí)

xhxl - hexadecimálna hodnota registra ix (v tomto poradí)

yhyl - hexadecimálna hodnota registra iy (v tomto poradí)

spsp - hexadecimálna hodnota registra sp

Základná informácia sa vypisuje vždy do dialógového riadku. Užívateľ však môže zabezpečiť opis dialógového riadku do obrazovky pomocou klávesy I tak, ako je to popísané ďalej, alebo ešte jednoduchšie môže opis jednorázovo skopírovať o dva riadky vyššie klávesou extend mode.

2.5.2 ZADÁVANIE HODNOTY VELIČINY PRE LADIACI PROGRAM

Skoro vo všetkých prípadoch, keď užívateľ zadáva hodnotu nejakej veličiny má k dispozícii tieto možnosti:

- decimálna konštanta
- hexadecimálna konštanta (začína znakom #)

- návěstie s definovanou hodnotou
- ľubovolná kombinácia uvedených troch prvkov spájaných operáciami + alebo -
- binárna konštanta (len v niektorých príkazoch)
- nič (len pri niektorých príkazoch)

Pritom na syntax zápisu sú kladené rovnaké požiadavky ako pri písaní symbolu v editore. Syntakticky nesprávny zápis ladiaci program odmietne zvukovým signálom a dá možnosť ho opraviť.

Ak užívateľ udal hodnotu ako návěstie, ladiaci program prezrie tabuľku symbolov posledne prekladaného zdrojového textu a ak tam daný symbol nenájde (alebo nemá definovanú hodnotu), zaznie zvukový signál a je možné chybný symbol opraviť. Daná veličina získa hodnotu príslušného symbolu, inými slovami napríklad register pc môžeme nastaviť na adresu ľubovolného návěstia v posledne prekladanom module a bez toho, aby sme sa museli starať o uloženie programu v pamäti.

Na editovanie vstupu možno využiť rovnaké prostriedky ako pri práci s dialógovým riadkom.

Ak sa odošle prázdny riadok, príslušná veličina sa nemení (pozri napr. príkaz pre modifikovanie pamäte m).

2.5.3 PRÍKAZY PRE LADIACI PROGRAM

Príkazy pre ladiaci program sú jednopísmenové. Po napísaní príslušného znaku sa príkaz hneď vykoná, nie je teda možné opraviť chybný príkaz. Odoslanie znaku, ktorému nie je priradený žiaden príkaz nevykoná nič.

Nasleduje popis príkazov. Spôsob ich použitia pozri v odseku 2.5.4.

R - príkaz pre nastavenie hodnôt 16 bitových registrov. Po jeho napísaní systém očakáva meno registra, teda jeden zo znakov p, a, b, d, x, y, s. Ak napíše iný znak, ladiaci program reaguje zvukovým signálom a očakáva ďalší príkaz. Po napísaní správneho mena registra sa dialógový riadok vymaže, vypíše sa

rn:

kde n je meno registra, ktorý sa bude modifikovať. Týmto príkazom možno modifikovať len 16-bitové registre (napr. nie len v register, ale vždy celú dvojicu bc). Ak sa odošle prázdny riadok, hodnota príslušného registra sa nezmení.

Pokiaľ užívateľ inicializuje register p, automaticky sa inicializuje aj register sp na hodnotu, ktorú mal pri spustení ladiaceho programu.

E - príkaz pre nastavenie hodnôt 8 bitových registrov. Po stlačení E sa vypíše "8 bit" a očakáva sa jeden z týchto znakov: A,B,C,D,E,F,H,L,R,I, X,Y. Po zadaní X alebo Y sa ešte očakáva písmeno H alebo L. Potom sa do predposledného riadku vypíše pôvodná hodnota príslušného registra a v poslednom riadku systém očakáva zadanie novej hodnoty. Príkaz E dovoľuje zadávať aj binárne čísla.

I - príkaz pre nastavenie bodu prerušenia. Ladiaci program očakáva hodnotu zadanú podľa bodu 2.5.2 a na adresu danú touto hodnotou nasadí bod prerušenia. Potom do dialógového riadku vypíše

n:

a očakáva decimálne číslo, ktoré udáva koľkokrát musí program prejsť cez bod prerušenia, kým sa jeho vykonávanie zastaví. Ak sa hodnota neudá, program sa preruší hneď, keď narazí na bod prerušenia. Ak sa udá napr. 1000 tak sa program preruší, keď narazí na bod prerušenia 1001 krát. Okrem toho sa pri každom prechode bodom prerušenia testuje kláves break a ak je stlačený, vykonávanie programu sa preruší.

Pri prerušení sa riadenie odovzdá ladiacemu programu, ktorý pokračuje výpisom základného stavu a čítaním príkazu. Pri výpise základného stavu je na prvej pozícii znak 'B' (break), ktorý signalizuje, že program narazil na bod prerušenia. Vo vykonávaní programu možno pokračovať buď po odstránení bodu prerušenia príkazom o alebo po vykonaní jednej inštrukcie príkazom s (pozri ďalej). V druhom prípade bod prerušenia v programe ostal a ak program pri ďalšom vykonávaní naň narazí, opäť sa preruší jeho vykonávanie. Počet prechodov bodom prerušenia musí byť opäť špecifikovať, pretože ostal nastavený na hodnotu 0.

Užívateľ môže špecifikovať najviac jeden bod prerušenia. Pri zadaní nového sa starý automaticky vymaže.

Na bod prerušenia program reaguje bez ohľadu na spôsob, akým bol spustený, pravda pri príkaze g len pre programy, ktoré nie sú v pamäti ROM.

Bod prerušenia je realizovaný pomocou inštrukcie rst 10 a využíva systémovú premennú CURCHL na adrese #5c51. Pre užívateľa je teda táto premenná pri práci s ladiacim programom nepoužiteľná. Toto je jediná vec, pri ktorej systém mrs využíva pamäť ROM.

O - príkaz pre odstránenie bodu prerušenia.

G - príkaz na spustenie programu. Registre sa inicializujú na hodnoty, ktoré vidno pri výpise základného stavu a program sa odštartuje. Pokiaľ program úspešne prebehne až po poslednú inštrukciu návratu (dosť zriedkavý jav), riadenie sa vráti ladiacemu programu. Register pc je nastavený opäť na začiatok programu, čiže program možno opätovne spustiť príkazom G.

CS+G - príkaz má rovnakú funkciu ako príkaz G, ale vykonávanie programu sa preruší pri prvom prechode bodom prerušenia bez ohľadu na počet prechodov.

S - príkaz na vykonanie jednej inštrukcie. Vykoná sa jedna inštrukcia s hodnotami registrov ako vidno pri výpise základného stavu. Po jej vykonaní sa riadenie vráti ladiacemu programu. Výpis základného stavu začína znakom 'S' (step), ktorý signalizuje krokovací režim. Ak užívateľ zabezpečí opis dialógového riadku do obrazovky (príkazom L), získa podrobný prehľad o vykonávaní svojho programu.

Pred vykonaním sa inštrukcia analyzuje, či neporušuje niektorú, užívateľom stanovenú ochranu (pozri príkaz W).

Analyzovanie inštrukcie typu ldir môže trvať dosť dlho, je to možné odstrániť krokováním pomocou príkazu CS+S (pozri ďalej).

Krokováť možno aj programy v pamäti ROM (čo väčšina podobných systémov neumožňuje).

CS+S - krokovanie programu s ignorovaním ochrán (pozri odsek 2.5.4.).

C - príkaz má rovnakú funkciu ako príkaz S ale krokovanie programu sa vykoná tak, že za inštrukciu sa vloží bod prerušenia a

inštrukcia sa vykoná normálne. Tento interný bod prerušenia nesúvisí s bodom prerušenia, ktorý zadal užívateľ okrem prípadu, že bod prerušenia zadaný užívateľom je v podprograme, ktorý krokuje príkazom C. V tomto prípade sa vykonávanie programu preruší v podprograme, ale už nie po výstupe z neho.

Význam príkazu je predovšetkým v tom, že odladené podprogramy volané inštrukciou call sa vykonávajú v normálnom rýchlo režime. Podobne ako pri príkaze I (insert break) sa funkcia nedá použiť na programy v pamäti rom.

Rovnako ako pri krokovaní príkazom S sa inštrukcia pred vykonaním analyzuje.

CS+C - má rovnakú funkciu ako príkaz C, ale inštrukcia sa vykoná aj keď analýza zistila narušenie niektorej ochrany.

CS+J - týmto príkazom sa inštrukcie vypisujú, ale nevykonávajú sa (v tomto prípade sa samozrejme ani nič nekontroluje). Register pc sa nastaví na ďalšiu inštrukciu. Príkaz CS+J slúži na preskočenie inštrukcií, ktoré sa nemajú vykonať (napr. chceme ukončiť predčasne príkaz cyklu, teda nevykonáme inštrukciu djnz, ale ju preskočíme).

T - program sa vykonáva v sledovacom režime. Jedná sa vlastne o opakovaný krokovací režim. Ladiaci program každú inštrukciu pred vykonaním interpretuje a zistí či je prípustná. Prípustné inštrukcie vykoná, vypíše základný stav do dialógového riadku, ale bez výpisu inštrukcie v symbolickom stave, pretože túto informáciu v sledovacom režime nestihne užívateľ v dialógovom riadku sledovať. Pravda ak požiada o opis dialógového riadku do obrazovky (príkazom L), začínajú tieto riadky znakom 'T' (trace), ktorý signalizuje sledovací režim a obsahujú aj výpis inštrukcie v symbolickom tvare. Popis možností, ktoré poskytuje sledovací režim pozri v odseku 2.5.4

CS+T - to isté, ale s ignorovaním ochrán

N - program sa vykonáva v zrýchlenom sledovacom režime. Možnosti tohto režimu sú rovnaké ako pri príkaze T, len sa po vykonaní inštrukcie nevypisuje základný stav, čím sa značne urýchli vykonávanie sledovaného programu.

CS+N - to isté, ale s ignorovaním ochrán

W - príkaz umožňuje nastaviť pamäťové okná a okná pre register pc. Po zadaní príkazu ladiaci program očakáva znak 'm' (memory), ak sa budú definovať pamäťové okná alebo znak 'p' (program counter), ak sa budú definovať okná pre register pc. Ak užívateľ napíše iný znak, ladiaci program prejde do základného stavu, ináč vypíše hodnoty príslušných okien - štyri dvojice hexadecimálnych čísel, kde každá dvojica definuje jedno okno. Ak chce užívateľ zmeniť niektoré z týchto ôsmich čísel, zadá jeho poradové číslo pomocou číslice v rozsahu 1-8. Po napísaní iného znaku prejde ladiaci program do základného stavu, ináč očakáva hodnotu príslušnej hranice, ktorá sa zadá podľa odseku 2.5.2. Po jej napísaní prejde ladiaci program do základného stavu.

Funkcia pamäťových okien je vysvetlená v odseku 2.5.4.

P - príkaz na zobrazenie pamäťového okna. Ladiaci program vypíše do dialógového riadku text

p:

a čaká hodnotu podľa zásad popísaných v odseku 2.5.2. potom vypíše

l:

a očakáva desiatkové číslo. Adresa definuje začiatok pamäťového okna a číslo l udáva jeho dĺžku v štvoriciach bytov (a v počte riadkov výpisu pamäte). Ak sa v ľubovolnom sledovacom režime zmení obsah tohto pamäťového okna zobrazí sa na obra-zovku výpis pamäte od danej adresy l riadkov (pozri tiež príkaz d).

Ak sa namiesto čísla stlačí len enter ruší sa funkcia zobrazovania pamäťového okna.

M - príkaz na modifikovanie obsahu pamäti. Ladiaci program vypíše do dialógového riadku text

m:

a čaká hodnotu podľa zásad popísaných v odseku 2.5.2. Potom do dialógového riadku vypíše adresu a obsah 4 bytov od tejto adresy vrátane. Výpis je jednak hexadecimálny a jednak znakový. Užívateľ môže modifikovať pamäť zadávaním hexadecimálnych konštánt a využíva nasledujúce riadiace klávesy:

- **šipka vľavo** - prechod na predchádzajúci znak. Medzery, ktoré oddeľujú byty sa automaticky skáču, ak je jazdec na prvom

znaku posunie sa výpis o jeden znak smerom k nižšej adrese.

- **šipka vpravo** - prechod na nasledujúci znak. Medzery, ktoré oddeľujú byty sa automaticky skáču. Ak je jazdec na poslednom znaku, posunie sa výpis o jeden znak smerom k vyšším adresám

- **true video** - zobrazia a modifikujú sa predchádzajúce 4 byty.

- **inv video** - zobrazia a modifikujú sa nasledujúce 4 byty.

- **extend mode** - byty modifikované na obrazovke sa vrátia do pôvodného stavu. Všeobecne do pamäti sa zapíše modifikovaná štvorica znakov až vtedy, keď sa posunú po obrazovke tak, že niektorý z nich zmizne z obrazovky.

- **enter** - modifikované byty sa uložia do pamäti a riadenie sa vráti ladiacemu programu, ktorý vypíše základný stav

- **ascii znak** - reprezentujúci hexadecimálnu číslicu (0-9 a-f) prepíše číslicu, na ktorej je jazdec. Jazdec sa posunie o jednu pozíciu, pričom sa medzery skáču.

- **medzera** - tento znak je akceptovaný len keď je jazdec na prvom znaku dvojice. Systém po ňom očakáva ďalší znak. Jeho kód sa uloží na miesto označené jazdcom.

- *stlačenie inej klávesy je chyba.* Zaznie zvukový signál a nič sa nevykoná.

Príkaz **M** (a príkaz **CS+M**) najčastejšie využíva tú vlastnosť vstupu, že odoslanie prázdneho riadku nezmení čítanú hodnotu. Inými slovami ak napr. príkazom **m:#4000** prezeráme pamäť od adresy **#4000**, tak ak najbližšie príkaz **m:** ukončíme hneď klávesom **enter**, opäť sa zobrazí pamäť od **#4000**.

CS+M - to isté ako **M**, ale pamätá si svoju adresu, ktorá je nezávislá od adresy príkazu **M**. Je to dobré na to, keď potrebujete viackrát meniť naraz dve pamäťové bunky na úplne rôznych adresách. Jednu meníte pomocou **M**, druhú pomocou **CS+M**.

D - príkaz na zobrazenie pamäte. Systém vypíše do dialógového riadku

d:

a zobrazí na obrazovku hexadecimálny a znakový výpis 192 bytov od adresy, ktorú zadal užívateľ. Ak odoslal prázdny riadok, tak sa použije naposledy zadaná adresa. Výpis riadi užívateľ týmito klávesami

klávesom **šipka hore** sa zobrazí nasledujúcich 192 bytov

klávesom **šipka dole** sa zobrazí predchádzajúcich 192 bytov

klávesom **A** špecifikuje znakový reťazec (pozri ďalej)

klávesom **H** špecifikuje reťazec hex. konštánt (pozri ďalej)

klávesom **enter** sa riadenie vráti ladiacemu programu

Iné klávesy sú považované za chybné.

Ak užívateľ použije príkaz **A** (resp. **H**) systém vypíše do dialógového riadku

a: (resp. **h:**)

a očakáva reťazec znakov (resp. hexadecimálnych konštánt). Pri ich zadávaní je možné opravovať chyby klávesom delete, reťazec sa ukončí klávesom enter. Pri chybné zadanom reťazci zaznie zvukový signál.

Editor vyhľadá reťazec v pamäti a urobí jéj výpis na obrazovku. Ak sa daný reťazec v pamäti nenájde, zaznie zvukový signál. Ak sa pri hľadaní reťazca zistí koniec pamäte, systém automaticky pokračuje od začiatku.

Ak sa zadá prázdny reťazec systém vyhľadáva nový výskyt prv zadaného reťazca. Ak ešte nebol reťazec zadaný, hľadá sa výskyt prvého bytu výpisu pamäte.

CS+D - ako **D**, ale pamätá si svoju vlastnú adresu. Takže môžete pomocou príkazov **D** a **CS+D** prezerať naraz dve oblasti pamäte.

A - príkaz na spätný preklad s výpisom na obrazovku. Do dialógového riadku sa vypíše text

a:

a na obrazovku sa zobrazí spätný preklad 24 riadkov od adresy, ktorú zadá užívateľ. Na ďalšie listovanie môžete použiť tieto klávesy:

Caps lock: ďalšia stránka výpisu

A: ďalší riadok výpisu

Edit: o #70 bajtov späť

Enter: návrat

CS+A - ako A, a ostatné viď príkaz CS+D.

X - tento kláves slúži ako prepínač. Jeho opätovným stláčaním vymieňame sadu registrov za alternatívne registre. Inými slovami stlačením klávesu x vykonáme inštrukcie exx aj ex af,af'.

L - týmto príkazom určí užívateľ, koľko riadkov sa má zobraziť na obrazovke. Systém do dialógového riadku vypíše

n:

a očakáva počet riadkov. Pred výpisom základného stavu sa potom posúva obrazovka o 2 riadky hore, čím sa predchádzajúci stavový riadok opíše do obrazovky, pričom sa na to využije n dvojriadkov obrazovky. Ak n=0 používa sa len dialógový riadok, ale v režime T sa vypisuje aj inštrukcia v zdrojovom tvare. Ak sa namiesto čísla stlačí enter, tak sa táto vlastnosť zruší.

Q - skok do dialógového režimu editora

CS+Q - skok do basicu - ako príkaz "mon" v editore

B - pre výpis stavovej informácie sa bude používať znakový súbor z ROM. **CS+B** - bude sa používať vlastný súbor 4*8 bodov

V - (val) Očakáva zadanie výrazu, ktorý sa zobrazí ako v príkaze "val" v editore. Možno zadávať aj binárne čísla.

Z - Skok do obrazovkového režimu editora. Ide vlastne o normálny príkaz editora "ln=" a aj syntax nasledujúcich parametrov, ktoré zadáte, musí vyhovovať tomuto príkazu. Ak zadáte len nulu, nikam sa skákať nebude. Break stlačený v editore vás tentoraz nevráti do dialógového režimu, ale priamo do ladiaceho programu.

CS+Z - to isté ako Z, ale po breaku v editore sa nevrátite do ladiaceho programu ale skočíte na príkaz "ald". Čiže zmeny, ktoré urobíte v zdrojovom texte sa vám hneď prenesú do binárneho kódu v pamäti.

SS+Z - skok priamo na príkaz "ald". Toto sa vám zídete vtedy, keď ste v ladiacom programe, potrebujete preložiť zdrojový text a nechce sa vám ísť do editora a napísať tam príkaz "ald".

F - príkaz OUT. Po zadaní si systém vypýta adresu (16 bitové číslo) a údaj (osembitový), ktorý treba poslať na port s príslušnou adresou. Prázdne riadky budú predstavovať posledné zadané hodnoty. Po odoslaní oboch hodnôt treba stlačiť enter pre potvrdenie a vykonanie príkazu.

Stlačením **medzery** sa príkaz nevykoná.

CS+K - príkaz "Save key" - uloženie hodnôt registrov do kľúča. Systém MRS má k dispozícii štyri kľúče, do ktorých si môžete odkladať hodnoty všetkých registrov, ktoré vidíte vypísané v základnej stavovej informácii. Po zadaní príkazu systém očakáva číslo kľúča (1 až 4), do ktorého budete registre ukladať.

K - príkaz "Load key" - vybratie hodnôt registrov z kľúča. Tieto príkazy majú význam vtedy, keď potrebujete viackrát volať nejakú rutinu s rovnakým alebo podobným začiatočným nastavením registrov.

Y - zákaz prerušenia (vykonanie inštrukcie DI)

CS+Y - povolenie prerušenia (inštrukcia EI)

SS+Y - nastavenie módu prerušenia. Systém očakáva jednu z číslic 0,1,2 a podľa toho vykoná inštrukciu im0,im1 alebo im2.

***Poznámka:** Systém vôbec nevyužíva ROM BEEP rutinu, preto pri pípaní ani nezakazuje ani nepovoľuje prerušenie. Prerušenie sa zakáže iba v príkazoch na obsluhu magnetofónu (loa,sav,mer,ver,hea.spd).*

H - bezhlavičkový load. Po zadaní príkazu treba zadať adresu pamäti, kam sa bude nahrávať, dĺžku, koľko sa bude nahrávať a flagbyte bloku, ktorý sa bude nahrávať. Vo všetkých prípadoch platí, že odoslanie prázdneho riadku znamená naposledy zadanú príslušnú hodnotu. Po zadaní poslednej hodnoty začne príkaz nahrávať. V prípade úspešného nahratia sa príkaz vráti do ladiaceho programu, no pri chybe sa znovu spustí.

SS+H - bezhlavičkové verify. Ináč to isté, ako príkaz H.

CS+H - bezhlavičkové save. Zadávanie parametrov je také isté, ako v príkazoch H a SS+H. Po zadaní parametrov príkaz ešte čaká na stlačenie klávesy enter a až potom sa začne vykonávať. Ak stlačíte medzeru, príkaz sa nevykoná.

J - normálne save s hlavičkou. Po zadaní adresy a dĺžky (ako v predchádzajúcich príkazoch) si systém pýta meno. Meno (maximálne desaťznakové) musíte vždy zadať (inak to bude 10 medzier). Tento príkaz vygeneruje normálny štandardný súbor typu "Bytes:".

***Poznámka:** Pre všetky tieto príkazy sa pamätá jedna spoločná adresa. To isté platí aj o dĺžke a flagbajte. Toto usporiadanie umožňuje pomerne pohodlnú prácu: Po príkaze save (CS+H) si zvolíte verify (SS+H), do ktorého už nemusíte písať tie isté parametre ako do save.*

SS+S - príkaz spd. Týmto príkazom sa zadáva rýchlosť nahrávania magnetofónových príkazov v ladiacom programe. Po zadaní príkazu sa očakáva jedna z kláves 0 a 1. Ak stlačíte "0", bude rýchlosť štandardná (1500 Bd), pre "1" bude dvojnásobná (3000 Bd). Všeobecne platí, že môžete stlačiť ľubovoľnú klávesu, kontroluje sa z jej ASCII kódu len nultý bit.

***Poznámka.** Neplatí to pre príkaz header (CS+enter), ktorý si sám zistí rýchlosť nahrávky.*

CS+U - príkaz run ako v editore. Spúšťa strojový kód od adresy zadanej v príkaze run v editore. Po návrate sa riadenie odovzdá ladiacemu programu.

CS+enter - príkaz hea. Platí o ňom všetko, čo o rovnomenom príkaze v editore. Po stlačení klávesy break sa riadenie vráti do ladiaceho programu.

Extend mode - skopírovanie základnej stavovej informácie o dva riadky vyššie. Je to vhodné vtedy, keď pri zadávaní parametrov potrebujete túto stavovú informáciu vidieť.

Delete - zmazanie obrazovky. (ako cls v editore). Zíde sa vám to, keď bude obrazovka od atribútov tak zamazaná, že skoro nič nebude vidieť.

Šipka hore: $PC=PC-1$

Šipka dole: $PC=PC+1$

Tieto dva príkazy dovoľujú veľmi pohodlne nastavovať register PC v malom rozsahu pamäti. Napríklad keď chcete viackrát vykonať jednu inštrukciu, nemusíte zakaždým PC nastavovať, ale stačí sa vrátiť šipkou o príslušný počet bajtov.

2.5.4 LADENIE PROGRAMOV

Sledovací režim (príkazy t alebo n) je veľmi silný prostriedok na ladenie programov. Pri jeho návrhu bol sledovaný hlavný cieľ - užívateľ v sledovacom režime nesmie stratiť vládu nad svojim programom. Kontrolu nad programom môže stratiť štyrmi spôsobmi:

- prepísaním pamäti
- vykonaním zakázanej inštrukcie
- zacyklením programu
- skokom do neznáma

Ladiaci program teda musí poskytnúť užívateľovi možnosť

chrániť ľubovoľnú časť pamäte pred prepísaním, filtrovať niektoré inštrukcie a umožniť prerušenie programu. Toto je dosiahnuté nasledovne:

- v sledovacom režime nie je možné prepísať adresy pamäte, ktoré sa nachádzajú v niektorom pamäťovom okne. Ak teda máme niektoré pamäťové okno nastavené ako interval aaaa bbbb, tak adresy od aaaa do bbbb včítane nie je možné v sledovacom režime prepísať. Pri pokuse o ich modifikovanie sa riadenie vráti ladiacemu programu, ktorý vypíše základný stav. Tento výpis začína znakom 'M', (memory), ktorý signalizuje, že došlo k pokusu modifikovať chránenú pamäť.

V editore sa tretie pamäťové okno automaticky inicializuje tak, že je chránený samotný systém a zdrojový text užívateľa.

- Na adrese #ff40 je zoznam, pomocou ktorého užívateľ môže definovať ilegálne inštrukcie. Ak pri vykonávaní programu príkazom s,c,t alebo n ladiaci program zistí, že má vykonať inštrukciu, ktorá je v tomto zozname, vykonávanie programu sa preruší a do dialógového riadku sa vypíše základný stav, pričom sa nastaví status='I' (illegal). Zoznam ilegálnych inštrukcií musí byť zakončený inštrukciou nop (00), ktorá teda nemôže byť ilegálna.

Inštrukcie sa zadávajú jedným bytom (operačným znakom), okrem inštrukcií s prefixom #ed, ktoré sa zadávajú dvoma bytmi (teda #ed a ešte jeden byte). Teda ako ilegálnu inštrukciu možno definovať celú triedu inštrukcií s prefixom #cb (rotácie, posuny bitové operácie), inštrukcie, ktoré pracujú s registrom ix (prefix #dd), ale aj napr. konkrétne inštrukciu ldір (byty #ed,#b0) alebo inštrukciu exx (byte #d9)

Inštrukcie definované ako ilegálne možno vykonať bežnými krokovacími príkazmi stlačenými spolu s caps shiftom.

Okrem ilegálnych inštrukcií definovaných užívateľom sa rovnakým spôsobom vyhodnocujú byty, ktorých kombinácia nemá pre procesor Z80 význam (napr. byty #dd,#fd). V takomto prípade je tiež status vo výpise ladiaceho programu 'I' (illegal), ale krokovaním s vypustením kontroly (príkazmi cs/s alebo cs/c) sa takáto ilegálna inštrukcia nevykoná, ale sa preskočí jej prvý byte.

- pred vykonaním každej inštrukcie v sledovacom režime zisťu-

je ladiaci program, či nebol stlačený kláves break. Ak áno, vykonávanie programu sa preruší a ladiaci program prejde do základného stavu. Týmto klávesom je teda možné prerušiť program počas vykonávania a pokračovať ďalej krokováním alebo spustiť normálne program príkazom G.

- obdobnú funkciu ako pamäťové okná majú aj okná pre register pc. Ich využitie nie je také kritické ako využitie pamäťových okien, keďže ladiaci program vie sledovať alebo krokováť aj programy v pamäti rom. To znamená, že aj keď má register pc akúkoľvek hodnotu, neznamená to stratu kontroly nad programom. Aj tak je použitie okien pre pc užitočné, užívateľ pomocou nich môže realizovať ďalšie body, ba dokonca celé oblasti prerušenia. Na rozdiel od pamäťových okien, ktoré zakazujú vykonanie inštrukcie, okná pre register pc ich naopak povoľujú. To znamená, že inštrukcia sa vykoná len vtedy, ak hodnota registra pc padne aspoň do jedného okna. Pri spustení programu je prvé okno pre register pc inicializované na interval 0008-ffff, teda register pc môže nadobúdať akékoľvek hodnoty okrem prvých ôsmich adries.

2.5.5 NIEKTORÉ DÔLEŽITÉ ADRESY A SYSTÉMOVÉ PREMENNÉ

memtop-#c7ff tu je uložený zdrojový text. Adresu memtop vám ukáže príkaz "sys" v editore.

#c800-#c83f buffer pre príkaz "ref" v editore.

#c840-#c87f registrové kľúče - priestor pre odklad registrov príkazmi "Save key" a "Load key" (K,CS+K).

#c880 začiatočná adresa samotného systému MRS. Odtiaľto sa nahráva do pamäti.

#d50e-#d50f systémová premenná "mode" (2 bajty). Sem sa ukladá hodnota napísaná v príkaze "mod" v editore.

#d65c studený štart systému. Vymaže sa zdrojový text,

premenná memory sa nastaví na #8000 a inicializuje sa zoznam ilegálnych inštrukcií. (dec 54876)

#d665 teplý štart systému. Zmaže sa obrazovka a vypíše sa úvodná správa. (dec 54885)

#d668 skok priamo do editora. Nič sa nemaže, ani sa nevypisuje žiadna správa. (dec 54888)

#dd77 Na tejto adrese je skok na rutinu, ktorá vytlačí na tlačiarňu jeden znak. Podrobnejšie v časti o pripojení tlačiarne na systém MRS.

#f4b3 konečná adresa všetkých programov. Ak sa chcete z programu vrátiť a v zásobníku máte neporiadok, môžete to vykonať inštrukciou "jp #f4b3".

#fde8 koniec samotného systému MRS. Potiaľto sa nahráva do pamäti.

#fe00-#fe3f textový buffer systému. Sem sa píšú všetky texty.

#fe40-#feff rôzne systémové premenné. Tu sa nachádza aj zásobník systému.

#ff00-#ff3f užívateľský zásobník. Akonáhle by užívateľovi nestačilo 64 bajtov, musí si zásobník inicializovať sám niekde inde v pamäti.

#ff40 Tu začína zoznam ilegálnych inštrukcií. Teoreticky je neobmedzený, no prakticky môže byť len po koniec pamäte (#ffff) s musí končiť nulou.

3.0 PRÍKLAD POUŽITIA SYSTÉMU MRS V PRAXI

Ukážme si použitie strojového kódu a systému MRS na jednom ucelenom príklade. Pri jeho výbere som sa snažil zohľadniť niekoľko protirečivých kritérií. Program by mal priniesť praktický úžitok, mal by dokumentovať výhody strojového kódu pred Basicom, mal by byť dostatočne jednoduchý, ale zároveň aj primerane zložitý.

Po dlhších úvahách som nakoniec vybral program na triedenie textových reťazcov podľa abecedy. To je problém, ktorý sa pomerne často vyskytuje a triedenie napísané v jazyku BASIC trvá nezniesiteľne dlho. Existujú rôzne algoritmy, ale najväčšie zrýchlenie dosiahneme ak daný modul napíšeme v strojovom kóde. V ďalšom texte je niekoľko málo úmyselných chýb, aby som mohol ukázať postup pri ich odhaľovaní a opravovaní.

Vývoj každého programu prebieha v troch etapách: návrh algoritmu, jeho zápis vo zvolenom programovacom jazyku a jeho napísanie a odladenie na príslušnom počítači. Náš cieľ je vytvoriť modul, ktorý bude volateľný z jazyka BASIC a bude triediť údaje, ktoré budú v tomto jazyku vytvorené. Predpokladáme, že je definované pole reťazcov príkazom DIM A\$(N,M), čím je definované pole N textových reťazcov, každý má dĺžku M znakov. Podľa informácií z príručky pre ZX Spectrum (kapitola 24), sú premenné v pamäti Spectra uložené od adresy, ktorej hodnota je uložená na adrese VARS=23627. Bez ujmy na všeobecnosti môžeme predpokladať, že premenná A\$(N,M) je uložená ako prvá, stačí ak ju ako prvú definujeme v našom programe. Z tvaru uloženia tejto premennej, tak ako je popísaný v kapitole 24 vyplýva, že na adrese (VARS)+4 je v dvoch bytoch uložená hodnota N=počet reťazcov a na adrese (VARS)+6 je v dvoch bytoch uložená hodnota M=dĺžka reťazca. Od adresy (VARS)+8 sú uložené za sebou jednotlivé reťazce. Pripomínam ešte, že zápisom (VARS) označujem 2-bytovú hodnotu uloženú na adrese VARS.

Na triedenie reťazcov použijeme veľmi jednoduchý algoritmus. Prechádzame postupne množinu reťazcov a porovnáme vždy dva susedné. Pritom sa využije skutočnosť, že ASCII kód znaku je priamo úmerný jeho poradiu v abecede, porovnáваме vlastne numerické hodnoty dvojíc znakov na rovnakých pozíciách v porovnáwanej dvojici reťazcov. Prvá dvojica vzájomne odlišných znakov udá zároveň aj porovnanie reťazcov v abecednom usporiadaní.

Ak pre porovnávanú dvojicu reťazcov zistíme inverziu, tak tieto dva reťazce vzájomne vymeníme, tento fakt si poznačíme do

nejakej premennej a pokračujeme v porovnávanie nasledujúcej dvojice reťazcov. Keď prejdeme celú množinu reťazcov skontrolujeme, či sme v tomto prechode vykonali nejakú výmenu. Ak áno zopakujeme celý cyklus znova, ináč sú reťazce zotriedené.

Tento veľmi jednoduchý postup doplníme jednou fintoou, ktorá náš program trochu skomplikuje, ale najmä triedenie dlhších reťazcov sa významne zrýchli. Pre dané reťazce si vytvoríme zoznam adries, na ktorých sa nachádzajú. Prechádzať budeme potom tento zoznam adries a namiesto výmeny dvoch reťazcov (čo môžu byť i stovky bytov) budeme vymieňať len dvojbytové adresy. Počas triedenia sa teda poprehadzujú len tieto adresy, ktoré na konci obsahujú informácie.

V zozname adries je vlastne adresa i -teho reťazca v abecednom triedení. Na základe týchto informácií poprehadzujeme reťazce v poli $A\$ (N, M)$. Ak nechceme pre zotriedené reťazce rezervovať nové miesto v pamäti, musíme si trochu ponamáhať rozum a vymyslieť algoritmus, ako ich vymieňať v pôvodnom poli, s použitím len jedného pomocného reťazca. Táto časť algoritmu je z celého programu najzložitejšia, doporučujem čitateľovi, aby sa pokúsil problém pochopiť a vyriešiť samostatne a až potom čítať ďalej.

Body 1 - 4 opakujem pre $i=1$ až $N-1$, kde N je počet reťazcov.

1. Vezmem i -tu hodnotu so zoznamu adries, táto hodnota udáva teda adresu reťazca, ktorý je na i -tej pozícii v abecednom triedení.

2. Reťazec na tejto adrese vymením s i -tym reťazcom v poli $A\$ (N, M)$.

3. V zostávajúcich adresách v zozname adries, teda na pozíciách $i+1$ až N najdem adresu i -teho reťazca z poľa $A\$ (N, M)$.

4. Namiesto tejto hodnoty zapíšem i -tu hodnotu zo zoznamu adries. Takým spôsobom zoznam adries zachytí stav, ktorý vznikol v poli $A\$ (N, M)$ výmenou dvojice reťazcov.

Nesmieme zabudnúť, že vždy musí platiť, že hodnota na i -tej pozícii v zozname adries musí udávať adresu i -teho reťazca v abecednom triedení.

Napíšme teraz uvedený algoritmus v zostavovacom jazyku procesora Z80. Hlavný program sa skladá z troch častí a jednej samostatnej procedúry na porovnávanie reťazcov. Ako pracovné pamäťové bunky potrebuje premenné FLAG, LENGTH, COUNT a ADDSTR, ktoré definujeme príkazmi prekladača DS. Premenné v jazyku BASIC sú uložené od adresy, ktorej hodnota je na adrese VARS. Zoznam adries budeme vytvárať na začiatku voľnej pamäti, adresa voľnej pamäti je na adrese RAMTOP a ako pracovnú pamäť použijeme pamäť používanú na prácu s tlačiarňou, ktorá je na adrese 23296 a má dĺžku 256 bytov. Na túto dĺžku teda obmedzíme dĺžku sortovaných reťazcov. V ďalšom budeme vlastný text programu písať veľkými písmenami, ak ho chce čitateľ prepísať do Spectra musí použiť malé písmená, lebo len tieto písmená systém MRS pozná. Teda úvod nášho programu môže vyzeráť takto:

```

;
;ssort -   usporiadanie reťazcov v poli a$(n,m) podľa abecedy
;
;vstup:   pole a$(n,m), ktoré musí byť definované ako prvá
;         premenná v programe v jazyku basic, dĺžka reťazca
;         (hodnota m) maximálne 256.
;
;vystup:  reťazce v poli a$(n,m) sú zoradené podľa abecedy
;
tmpbuf    equ    23296           adresa pracovnej pamäte
vars      equ    23627          adresa adresy premenných
ramtop    equ    23730          adresa adresy voľnej pamäte
flag      ds     1              indikátor výmeny reťazcov
length    ds     2              dĺžka reťazca
count     ds     2              počet reťazcov
addstr    ds     2              adresa uloženia reťazcov

```

Napíšme najprv procedúru porovnávanie reťazcov s názvom CMPS. Vstup do procedúry sú adresy dvoch porovnávaných reťazcov v registrových pároch BC, DE, dĺžka reťazca je na adrese LENGTH a výstup procedúry bude v príznaku CY.

```

;
;cmps    - procedúra na porovnanie dvojice reťazcov
;
;vstup: <bc>,<de> - adresy reťazcov
;výstup: cy=1 reťazec na adrese je za reťazcom na adrese
;          <bc>v abecednom usporiadaní
;mení sa: <af>
;
cmps     push hl      uchovanie registrov
         push de
         push bc
         ld  hl,(length)
         ex  de,hl
;
;<bc>,<hl>- adresy reťazcov, <de> - dĺžka reťazca
;
d1       ld    a,(bc)
         cp    (hl)      porovnaj dvojice znakov
         jr    nz,d1      znaky sa nerovnajú
         inc   hl         <hl> a <bc> ukazujú
         inc   bc         na ďalšiu dvojicu znakov.
         dec   de         zníž a otestuj počítadlo
         ld    a,d        porovnaných znakov
         or    e          <de>= 0 ?
         jr    nz,d1      nie, pokračuj ďalšou dvojicou
d2       pop   bc         obnov registre
         pop   de
         pop   hl
;
;cy = 1   ak hodnota znaku na adrese <hl> je väčšia ako
;          hodnota znaku na adrese <bc> .
;cy = 0   ak sú reťazce rovnaké alebo hodnota znaku na adrese
;          <hl> je menšia ako hodnota znaku na adrese <bc>.
ret

```

Prvá časť programu pripraví pole adries a prevezme informácie o počte a dĺžke reťazcov. Zoznam adries reťazcov zakončíme dvoma nulovými bytmi (teda adresou 0), ktorá bude slúžiť ako zarážka. Ak pri prezeraní zoznamu adries narazíme na adresu 0 znamená to, že sme prešli celý zoznam.

```

ssort    ld    hl,(ramtop)
         inc   hl
         push  hl          tu sa vytvorí zoznam adries
         ld    hl,(vars+4)
         ld    (count),hl   počet reťazcov
         ld    hl,(vars+6)
         ld    (length),hl  dĺžka reťazca
         ld    hl,vars+8
         ld    (addstr),hl  adresa prvého reťazca
;
;stack - adresa zoznamu adries, bc - dĺžka reťazca
;hl - adresa prvého reťazca, de - počet reťazcov
;
a1       ex    de,hl        hl-počet, de-adresa reťazcov
         ex    (sp),hl      hl-adresa zoznamu, stack-počet
         ld    (hl),e       do zoznamu sa uloží
         inc   hl          adresa ďalšieho reťazca
         ld    (hl),d       najprv nižší, potom vyšší byte
         inc   hl
         ex    (sp),hl      hl-počet, stack-adresa zoznamu
         ex    de,hl        de-počet, hl-adresa reťazcov
         add   hl,bc        hl na ďalší reťazec
         dec   de           zníž počítadlo reťazcov
         ld    a,d          testuj počítadlo de
         or    e            na 0
         jr    nz,a1        zoznam adries neúplný
         pop   hl           hl-adresa konca zoznamu adries
         ld    (hl),a       zoznam adries zakončíme
         inc   hl           dvoma nulovými bytmi

```

ld (hl),a využijeme, že a=0

Druhá časť programu vykoná vlastné triedenie reťazcov, pričom ich usporiadanie sa zachytí v poli adries.

b1	xor	a	vynuluj a
	ld	(flag),a	nastav indikátor výmeny
	ld	hl,(ramtop)	
	inc	hl	hl-adresa zoznamu adries
	ld	c,(hl)	bc-adresa prvého reťazca
	inc	hl	
	ld	b,(hl)	
	inc	hl	
b2	ld	d,b	de-adresa prvého
	ld	e,c	porovnávaného reťazca
b3	ld	c,(hl)	bc-adresa druhého
	inc	hl	
	ld	b,(hl)	porovnávaného reťazca
	inc	hl	
	ld	a,b	ak adresa druhého reťazca=0
	or	c	tak je zoznam adries vyčerpaný
	jr	z,b4	pokračuj testom premennej flag
	call	cmps	ináč porovnaj reťazce

;

;ak cy=0 výmena adries reťazcov v zozname adries sa nevykoná

;a program pokračuje na návěští b2, kde sa adresa druhého

;reťazca (obsah registra bc) presunie na miesto adresy prvého

;reťazca (do registra de), register bc sa naplní adresou

;ďalšieho reťazca a pokračuje sa v cykle porovnávaní.

;ináč sa reťazce v zozname vymenia, register de tým obsahuje

;adresu prvého reťazca a stačí pokračovať na návěští b3, kde

;sa register bc naplní adresou ďalšieho reťazca.

;

jr	nc,b2	výmena adries sa nevykoná
ld	a,l	

	ld	(flag),a	poznač výmenu adries reťazcov
	push	hl	ulož adresu do zoznamu adries
	dec	hl	
	ld	(hl),d	ulož adresy do zoznamu
	dec	hl	v obrátenom poradí, najprv de
	ld	(hl),e	potom bc.
	dec	hl	keďže sa do pamäti ukladá
	ld	(hl),b	zhora dole ukladá sa
	dec	hl	najprv vyšší a potom nižší byte
	ld	(hl),c	
	pop	hl	obnov adresu do zoznamu adries
	jr	b3	a pokračuj
b4	ld	a,(flag)	koniec jedného prechodu
	or	a	bola v ňom vykonaná výmena ?
	jr	nz,b1	áno - pokračuj novým prechodom

V tretej časti programu sa usporiadajú reťazce v poli A\$(N,M) podľa informácií v obsiahnutých v zozname adries, realizuje sa teda algoritmus popísaný bodmi 1. až 4.

	ld	bc,(addstr)	bc ukazuje na reťazce v poli a\$
	ld	hl,(ramtop)	hl ukazuje na adresy reťazcov
	inc	hl	v zozname adries
c1	ld	e,(hl)	de-adresa reťazca, ktorý je
	inc	hl	na i-tom mieste v usporiadaní
	ld	d,(hl)	reťazcov podľa abecedy
	inc	hl	ak je nasledujúca adresa 0
	ld	a,(hl)	tak sa algoritmus vykonal
	inc	hl	pre i=1 až n-1
	or	(hl)	a reťazce sú usporiadané
	ret	z	riadenie sa vráti do basicu
	push	hl	ulož adresu do zoznamu adries

;

;nasleduje výmena reťazcov, ktorých adresy sú v registroch bc
;a de (bod 2. algoritmu). na výmenu sa použije pracovná pamäť

;na adrese tmpbuf. na začiatku sa uchová obsah registrov hl, bc
; a de, ktoré sú potrebné pri realizácii bodov 3. a 4.
;vlastná výmena reťazcov ďalej nebude komentovaná, doporučujem
;čitateľovi, aby starostlivo sledoval obsahy registrov pred
;použitím inštrukcie ldir.

;

```

push hl
push de
push bc
push de          tu začína výmena reťazcov
ld    bc,(length)
push bc
ld    hl,tmpbuf
ex    de,hl
ldir
pop    bc
pop    de
pop    hl
push   hl
push   bc
ldir
pop    bc
pop    de
push   de
ld    hl,tmpbuf
ldir          výmena reťazcov skončená
ex    de,hl
pop    bc
pop    de
ex    (sp),hl

```

;

;register bc obsahuje adresu reťazca v poli a\$, register de
;obsahuje adresu reťazca so zoznamu adries, register hl
;obsahuje adresu nasledujúcej adresy zo zoznamu adries (de

;je hodnota so zoznamu a hl je ukazovateľ do zoznamu).
;v stacku sú uložené adresy na ďalší reťazec v poli a\$ a
;adresa na ďalšiu adresu v zozname adries. tieto hodnoty sú
;potrebne pre opakovanie bodov 1. až 4.
; realizovať body 3.a 4. znamená nájsť v zozname adries,
;ktorého začiatok je v hl hodnotu, ktorá je v bc a prepísať
;ju hodnotou, ktorá je v de
;

c3	ld	a,(hl)	
	cp	c	porovnanie nižších bytov
	inc	hl	prv než sa vykoná prípadný skok
	ld	a,(hl)	treba prečítať aj vyšší byte,
	inc	hl	aby hl bol na ďalšej adrese
	jr	nz,c3	nižšie byty sú rôzne
	cp	b	ináč porovnaj vyššie byty
	jr	nz,c3	pokračuj v hľadaní
	dec	hl	ináč prepíš nájdenú hodnotu
	ld	(hl),d	hodnotu v registri de
	dec	hl	
	ld	(hl),e	
	pop	bc	bc-adresa na ďalší reťazec v a\$
	pop	hl	hl-na ďalšiu adresu v zozname
	jr	c1	pokračuj bodmi 1.až 4.
	end		koniec textu programu

Program je na papieri, nasleduje tretia časť - jeho uloženie do počítača. Z kazety nahráme program MRS, príkazom INI prejdeme do obrazovkového režimu a uvedený text starostlivo opíšeme (komentáre môžeme vynechať).

Väčšinu chýb, ktoré urobíme pri opise systém zachytí, ak sa ďalej vyskytnú iné ťažkosti, než tu popísané, je potrebné odstrániť chyby, ktoré vznikli pri prepise.

Po napísaní celého textu sa vrátíme z príkazového režimu (klávesom BREAK) a program preložíme príkazom ASM. Prípadné chyby, ktoré modul ASM odhalí opravíme a príkazom SAV SSORT uložíme zdrojový text na pásku, aby sme ho počas ladenia náhodou nevymazali.

Ladenie začneme od procedúry CMPS. Vstupné dáta pre túto procedúru pripravíme niekde v pamäti príkazom M modulu DBG. Vyvoláme modul DBG a napíšeme text

```
m #8000
```

```
j a n o j o z o
```

tak od adresy #8000 uložíme ASCII kódy dvoch reťazcov 'jano' a 'jozo'.

Príkazmi

```
rb#8000
```

```
rd#8004
```

```
rpcmps
```

```
m length 0400
```

uložíme do registrov BC a DE adresy reťazcov, do registra PC adresu procedúry CMPS a do premennej LENGTH dĺžku porovnávaných reťazcov (4). Príkazom N spustíme procedúru CMPS a ladiaci modul vypíše za chvíľu základný stav začínajúci znakom E. Procedúra teda prebehla úspešne, skontrolujeme ešte, že sa registre BC, DE, HL nezmenili a že CY flag obsahuje správnu hodnotu. V našom prípade musí byť CY=1 lebo reťazec na adrese DE 'jozo' je za reťazcom na adrese BC 'jano'. Rovnakým spôsobom overíme funkciu procedúry v prípade, že sú reťazce vymenené alebo sa porovnávajú dva totožné reťazce. Keďže všetky testy sú úspešné, považujeme procedúru za odladenú.

Aby sme mohli odskúšať prvú časť programu musíme pripraviť pole A\$ v jazyku BASIC. Príkazom mon vráti riadenie systém MRS programu BASIC.

Výraz za príkazom CLEAR opravíme na 6*4096+7*256-1, čím vytvoríme na začiatku voľnej pamäti priestor 256 bytov používaný v programe SSORT na zoznam adries. V basicu napíšeme napr:

```
20 DIM A$(5,4):LET A$(1)="IVAN":LET A$(2)="RUDO":
    LET A$(3)="JANO":LET A$(4)="PALO":
    LET A$(5)="ADAM"
```

Potom systém MRS opäť spustíme pomocou RUN a príkazom DBG vyvoláme ladiaci program. Príkazmi:

```
rpssort
sssssssss
```

vykonáme prvých deväť inštrukcií, ktorými by sa mali prevziať parametre poľa A\$. Register BC by mal mať hodnotu 4 (dĺžka reťazca) a register DE hodnotu 5 (počet reťazcov). Pohľad na obrazovku nás však presvedčí, že v prvej časti programu je niečo chybné, registre BC a DE majú veľmi zvláštne hodnoty.

Keď sa bližšie pozrieme napr. na inštrukciu LD BC,(VARS+4) vidíme, že do registra BC dávame hodnotu z adresy (VARS+4), ale požadovaná hodnota je na adrese (VARS)+4, čo je rozdiel. Táto veľmi typická chyba sa týka rovnako obsahov registrov DE a HL. Prvé tri riadky počnúc návestím SSORT sú v poriadku, ale ďalších päť riadkov vymažeme a nahradíme ich riadkami

```
ld    hl,(vars)
inc   hl
inc   hl
inc   hl
inc   hl
ld    e,(hl)
inc   hl
ld    d,(hl)
inc   hl
ld    c,(hl)
inc   hl
ld    b,(hl)
inc   hl
ld    (length),bc
ld    (count),de
```

Príkazom q ukončíme prácu ladiaceho modulu. Príkazom ln=ssort sa nastavíme na začiatok programu a opravíme ho. Program opäť preložíme, aby sa oprava premietla aj do binárneho kódu a opäť ho vyskúšame. Pre zmenu to môžeme vykonať tak, že príkazom DBG vyvoláme ladiaci modul, a príkazmi:

rpssort

ib1

n

nastavíme PC register na začiatok programu, bod prerušenia na návěstie b1 a program po tento bod vykonáme v sledovacom režime. Príkazmi

mlength

mcount

maddstr

skontrolujeme postupne obsah premenných count, length a addstr. V prvom prípade je obsah prvých dvoch bytov 04 00, v druhom prípade 05 00 a v treťom prípade 26 5f. Teda length=4, count=5 a skontrolujeme ešte hodnoty na adrese #5f26. Klávesom

ukončíme príkaz m a príkazom

d#5f26

zobrazíme obsah pamäte od adresy #5f26. Aj tu je všetko v poriadku, na adrese #5f26 sú naše reťazce definované v programe BASIC. Ostáva ešte overiť, či od adresy (ramtop)+1 sú uložené adresy týchto reťazcov zakončené dvoma nulami. Napíšeme teda

d#6700

a skutočne od adresy #6700 sú hodnoty 26 5f 2a 5f 2e 5f 32 5f 36 5f 00 00 tak, ako to má byť.

Rovnakým spôsobom overme činnosť programu po návěstie c1, teda vlastné triedenie. Príkazmi

icl

n

vykonáme program po návstevie c1 v sledovacom režime. Program sa úspešne na toto miesto dostane a nám ostáva overiť, či hodnoty na adrese #6700 obsahujú informácie nutné na usporiadanie reťazcov podľa abecedy. Keďže adresa #6700 bola ostatná, ktorú sme príkazom d zobrazili stačí napísať príkaz

d

Na adrese #6700 sú hodnoty 36 5f 26 5f 2e 5f 32 5f 2a 5f 00. Príkazom m#5f36

vidíme, že text na adrese #5f36 je ADAM. Rovnakým spôsobom overíme, aj ostatné adresy, čím si potvrdíme, že adresa na i-tej pozícii je adresa i-teho reťazca v usporiadaní podľa abecedy. Ladenie programu teda úspešne pokračuje, ostáva otestovať záverečnú časť programu. Nie je nutné nastaviť bod prerušenia, pretože posledná inštrukcia RET vráti riadenie modulu DBG. Pustíme program ďalej príkazom

n

ale program sa ani po 10 sekundách (čo je ozaj dosť pre tak jednoduchý kód) neprihlási. Preručíme jeho vykonávanie klávesom BREAK a pokračujeme v jeho vykonávaní krokováním príkazom s, aby sme zistili, kde sa zamotal.

Program stále vykonáva inštrukcie počnúc návstevím c3, ktorými hľadá hodnotu registra BC v zozname adries, ako ukazovateľ slúži register HL. A register HL má už hodnotu cez #7000, hoci zoznam adries je v intervale #6700-#670b. Teda z nejakého dôvodu sa hodnota v BC registri nenašla. Pohľad na obrazovku nám prezradí, že BC register má veľkosť #5f2e a príkazom

m#5f2e

zistíme, že sa jedná o reťazec JANO. Čím je tento reťazec zvláštny? Mnoho rozmýšľania si ušetríme ak zbadáme, že register DE, ktorého hodnotou máme nájdenú hodnotu v zozname prepísať má tiež hodnotu #5f2e. To vlastne znamená, že reťazec JANO sa nikam neposúva. Tento reťazec bol pôvodne na treťom mieste a po usporiadaní na treťom mieste ostane. Keďže jeho adresu hľadáme až od štvrtej pozície, je jasné, že ju nemôžeme nájsť. Bod 3. opravíme na (???)

V zostávajúcich adresách v zozname adries, teda na pozíciách i až N nájdeme adresu i-teho reťazca z poľa A\$(N,M).

Pre program v zostavovacom jazyku to znamená, že pred prehľadávaním zoznamu adries, ukazovateľ (hodnotu HL) znížime o 2. Príkazom q ukončíme prácu modulu DBG, príkazom edi vyvoláme editor a príkazom ln=c3 sa nastavíme na riadok s návěstím c3. Pred tento riadok vložíme dvakrát riadok dec hl a program opäť preložíme. Keďže časť reťazcov už bola posunutá, môžeme pole A\$ obnoviť tak, že príkazom mon vrátime riadenie do BASI-Cu a program spustíme príkazom RUN. Potom príkazom dbg zavoláme ladiaci modul a príkazmi

```
rpssort
```

```
n
```

vykonáme celý program v sledovacom režime. Program po malej chvíli skončí návratom do ladiaceho modulu a nám ostáva príkazom

```
d#5f26
```

overiť, že reťazce sú naozaj usporiadané. Predpokladajme, že program je týmto odladený, hoci v praxi by to chcelo dôkladnejšie testovanie. Ostáva už len jedno, pripraviť program tak, aby slúžil ako univerzálny program bez prítomnosti systému MRS. Ako logické miesto v pamäti sa núka priestor od adresy #fe00 a tak zdrojový text upravíme tak, že za prvý riadok (obsahujúci len ;) vložíme riadok ORG #fe00. Aby sme zistili dĺžku programu doplníme ešte posledný riadok s textom END ľubovoľným návěstím napr. na KONIEC END.

Ak sa však program pokúsime preložiť príkazom asm, preklad skončí výpisom mem full. Na adrese #fe00 je totiž kód systému MRS a prekladač ho nedovolí prepísať. Na šťastie MRS poskytuje prostriedok proti tejto komplikácii. Za riadok s textom ORG vložíme inštrukciu riadenia prekladu v tvare napr. *c8e00. Ak preložíme takto upravený program, tak sa vlastný kód uloží od adresy #8e00, ale vykonávať sa dá až po jeho presunutí na adresu #fe00. Ako ukážeme, tento presun možno urobiť pomocou jazyka BASIC. Po preklade ešte určíme rozsah a štartovaciu adresu programu. Vieme, že program bude začínať na adrese #fe00=65024 koncovú adresu a štartovaciu adresu zistíme pomocou modulu dbg príkazmi

```
mkoniec
```

```
mssort
```

V našom prípade to je koniec=#fec2 program má teda dĺžku #1c=194 bytov a ssort=#feld=65053 čo je hodnota štartovacej adresy. Klávesom

ukončíme príkaz m a príkazmi q a mon vrátime riadenie programu BASIC a uložíme program na pásku. Pritom musíme zohľadniť, že vlastný kód je uložený od adresy #8e00=36352. Uložíme ho teda príkazom

```
SAVE "SSORT"CODE 36352,194
```

Pred jeho použitím musíme nastaviť RAMTOP aspoň na hodnotu #fe00-2*počet reťazcov-1, pretože od (RAMTOP)+1 sa ukladá zoznam adries. Potom nahráme binárny kód do pamäti príkazom

```
LOAD "SSORT"CODE 65024
```

a pokiaľ je premenná A\$(N,M) prvá v našom programe tak jej reťazce zoradíme podľa abecedy príkazom

```
RANDOMIZE USR 65053
```

Na záver ešte jedno upozornenie. AK pole A\$ nahrávame z pásky, tak sa pôvodná definícia príkazom DIM vymaže a pole už nebude prvá premenná. Znamená to, že v tomto prípade musí byť príkaz LOAD "MENO"STRING A\$() prv, ako sa definuje iná premenná.

Schopný čitateľ s pomocou príručky k počítaču Spectrum ľahko upraví program tak, aby vedel vyhľadať danú premennú v zozname premenných. Potom môže byť meno poľa vstupným parametrom pre podprogram, toto meno sa môže uložiť príkazom POKE na pevné miesto v pamäti a podprogram SSORT ho odtiaľ prečíta.

