MIKROBÁZE

Dr.MG

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

AR-602

OBSAH

ÚVOD

Program Dr.MG vznikl rozsáhlou úpravou a doplněním všeobecně známých a oblíbených programů firmy HISOFT - MONS3 a GENS3. Pro vážnější práci s mikropočítačem patří jakýkoliv monitor a assembler mezi nepostradatelné, MONS3 a GENS3 pro ZX Spectrum mezi nejoblíbenější. Navzdory široké škále jejich možností a výborných vlastností mají tyto programy některé nedostatky.

Mnozí uživatelé ZX Spectra neovládají angličtinu a často právě z tohoto důvodu nedokáží využít všechny přednosti a výhody MONS3 a GENS3. Proto jsme se v první řadě rozhodli přeložit texty původních programů do rodné slovenštiny (zvolili jsme úpravu bez diakritických znamének - abychom šetřili paměť). Nepříjemným nedostatkem byla i "němá" klávesnice - to programátory nutilo k necitlivému zacházení s křehkou kontaktní membránou klávesnice. Proto jsme MONS3 i GENS3 naučili "pípat".

Abychom předešli kolapsům programů vinou chybné volby tzv. horkého a studeného startu GENS3, zavedli jsme do našeho modulu automatiku, která sama rozpozná, jestli v GENS3 nějaký program je či není. Tak GENS3 startuje vždy ze stabilní adresy, kterou si předem můžeme sami určit.

Jednou jsme se pokusili generovat zdrojový text pro GENS3 pomocí MONS3 a znova to dopadlo neslavně. Bylo totiž potřebné přepočítat a "zapoukovat" více parametrů a my jsme se přirozeně zmýlili. Rozhodli jsme se proto, že se tyto i ostatní neduhy pokusíme vyléčit. Tak vznikl:

KAPITOLA 1 ŘÍDÍCÍ A SPOJOVACÍ MODUL DOKTOR

ve strojovém kódu (jeho moderační složka je basicová). Strojový kód je standardně uložený od adresy 28000 (6D60H) a má délku 3 kB. Modul zaručuje rychlou a jednoduchou manipulaci s programem.

Jednou z hlavních funkcí modulu je integrace činností MONS3 a GENS3. Zajišťuje jejich vzájemnou informovanost a řídí jejich spolupráci. Zároveň zabezpečuje jejich soběstačnost a nezávislost, takže je můžeme prakticky libovolně přemísťovat v paměti RAM. Modul přemísťování řídí a dokáže relokovat i sám sebe. Umožňuje uložit na pásek (a později použít) MONS3, GENS3

i sebe izolovaně nebo v jakékoli vzájemné konfiguraci.

Řídí vstupy do MONS3 a GENS3. Umožňuje jednoduchým způsobem odlaďovat assemblerové programy, poskytuje možnost vytvořit vlastní basicový program, který se pak stává součástí (přechodnou nebo trvalou) řídícího modulu. Vytvořený strojový program je možno jednoduchou manipulací uložit na pásek a nahrávku verifikovat. Pochopitelně za jeho součinnosti můžeme ukládat strojový program do volné paměti Spectra.

Informuje uživatele o aktuální lokalizaci jednotlivých součástí Dr.MG a také o rozsahu volné paměti.

Uživateli nabízí možnost jednoduchých a rychlých převodů mezi jednotlivými číselnými soustavami včetně pětibajtového vyjádření s plovoucí řádovou čárkou a číselného tvaru v syntaxi kalkulátoru Spectra.

1.1 Hlavní panel

Po vstupu do řídícího modulu se zobrazí hlavní panel.

```
Dr.MG-monitor(assembler)editor
MONS ... M                    d
GENS ... G                    h
LOAD ... L                    b
SAVE ... S
VERI ... V MEMORY  KALKUL
           hex dek  hex dek
PROG ... P
FARB ... F
RELO ... R

HEX ... H
BIN ... B

Dialogový řádek
```

Modul nerozlišuje malá a velká písmena. Příkazy mají okamžitý účinek (bez nutnosti tisknout ENTER). Při číselných převodech můžeme vložit až 20 znaků, přičemž vložení 20. znaku vykoná funkci ENTER. Při hexadecimálním vstupu se zobrazí jen čtyři nejnižší bajty vloženého čísla (respektive výsledku převodu). Binárním vstupem můžeme uložit až 16tibitovou hodnotu.

Když uložení číselného parametru ukončíme číselným znakem (0-9, A-F, desetinnou čárkou, znaménkem + a -), který nebude

platný pro aktuálně použitou číselnou soustavu, po stisku ENTERu program vklad ignoruje. Číselný vklad můžeme ukončit místo ENTERu jakoukoli klávesou s nečíselným znakem (s výjimkou BREAK a "X"). Dekadická čísla můžeme vkládat i v exponenciálním tvaru.

Při každém vstupu do řídícího modulu, i po použití příkazu, který očekává následné vložení hodnoty, je modul nastavený na vstup dekadických čísel. Když po vstupu do řídícího modulu vložíme dekadické číslo a vklad správně ukončíme, program to pochopí jako příkaz k převodu vloženého čísla do jiných číselných soustav.

Jakoukoli ukládanou hodnotu můžeme vložit i v hexadecimálním nebo binárním tvaru, a to tak, že před uložením čísla uložíme jako první znak H (tj.hexad.) nebo B (tj.bin.).

Vkládaný znak můžeme z dialogového řádku vymazat pomocí DELETE nebo stlačením klávesy "X" (funguje jako DELETE). Když si po vstupu do BASICu budete přát jeho výpis, zadejte LIST 100. K hlavnímu panelu se vrátíme příkazem RUN nebo GO TO 1.

1.2 Příkazy hlavního panelu

1.2.1 Příkaz "M"

Stlačením "M" vstoupíme do MONS3, který je v programu standardně uložený od adresy 31000 (78DCH); jeho délka je 6010 (177AH) bajtů. Pro příkaz "M" platí stejná pravidla jako pro příkaz "D" s tím rozdílem, že segment MONS3 můžeme pasivovat, když po zadání "M" vložíme znak "0".

1.2.2 Příkaz "G"

Stlačením "G" vstoupíme do GENS3, který je v programu standardně uložený od adresy 37010 (9056H); jeho délka je 8450 (2102H) bajtů.

1.2.3 Příkaz "L"

Umožňuje načítání strojového kódu do paměti ZX Spectra. Příkaz si vyžádá uložení názvu souboru, který vkládáme bez uvozovek. Když místo názvu vložíme ENTER, program se vrátí k hlavnímu panelu. Po uložení názvu souboru si příkaz vyžádá zadání adresy, od níž chceme uložit soubor do paměti, a jeho délku. Když místo hodnot vložíme samotný ENTER, budou použity hod-

noty, které obsahuje hlavička záznamu na pásku. I zde můžeme číselné hodnoty uložit ve tvaru dekadickém, hexadecimálním nebo binárním.

1.2.4 Příkaz "S" (SAVE)

Jím přeneseme strojový kód z paměti počítače na pásek. Příkaz si vyžádá uložení názvu, počáteční adresy a délky ukládaného souboru. Když místo požadovaných parametrů vložíme samotný ENTER, program se vrátí k hlavnímu panelu.

1.2.5 Příkaz "V" (VERIFY)

Po stisku "V" si program vyžádá název, počáteční adresu a délku strojového kódu, který chceme verifikovat. Uložení názvu souboru je povinné. Když místo požadovaných číselných hodnot vložíme samotný ENTER, uskuteční se verifikace souboru, který bude na pásce nalezen jako první.

1.2.6 Příkaz "P"

Stiskem "P" vystoupíme z řídícího modulu do BASICu. Zde můžeme napsat vlastní basicový program od řádky č. 1000. K hlavnímu panelu se vrátíme příkazem RUN nebo GO TO 1. Náš basicový program můžeme pak spustit stiskem "P".

1.2.7 Příkaz "F"

Překlopí barvy papíru a inkoustu do inverze.

1.2.8 Příkaz "R"

Stiskem "R" vyvoláme zobrazení relokačního menu.

1.3 Převody mezi číselnými soustavami

Vkládání číselných hodnot viz část 1.1. Převod mezi jednotlivými číselnými soustavami je velmi rychlý, výsledek okamžitě aktualizuje hlavní panel. Modul umožňuje práci s dekadickými čísly od 1E-38 do 1E+38. Na hexadecimální a binární tvar můžeme převádět celá dekadická čísla v rozpětí od -65535 do +65535. Výsledek převodu záporného dekadického čísla se zobrazí ve formě dvojkového doplňku, na což nás upozorní blikající písmeno c.

Výsledky všech převodů (včetně převodů desetinných dekadických čísel a záporných dekadických čísel menších než -65535) se vyobrazí i ve formě pětibajtového vyjádření s plovoucí řádovou čárkou a v číselném tvaru "dialektu" kalkulátoru ZX Spectra. Ptáte se, co je "dialekt" kalkulátoru? ZX Spectrum má ve své ROMce velmi užitečný podprogram - kalkulátor. Vykonává výpočty s plovoucí řádovou čárkou, manipulace s řetězci atd. Je pravděpodobně nejuniverzálnějším podprogramem ROMky. Ti, kdož znají činnost kalkulátoru podrobněji, vědí, že během jeho užití se může vyskytnout potřeba uložit na vrchol jeho zásobníku nějaké číslo. Voláním patřičných podprogramů se však nejen ztrácí čas, ale plýtvá se i místem. Proto má kalkulátor rutinu STK-DATA, která umožňuje vkládat desetinná čísla přímo v režimu kalkulátoru. Čísla se vkládají v tzv. kondenzované formě a podle tvaru čísla se skládají ze 2, 3, 4, 5 nebo 6 bajtů. Uvedená rutina si číslo správně upraví na pětibajtové číslo s plovoucí řádovou čárkou a uloží je na vrchol zásobníku kalkulátoru.

1.4 Relokační menu

Na něm jsou dekadicky znázorněny aktuální hranice jednotlivých segmentů Dr.MG:

```
LOKALIZACE SEGMENTU:
BASIC ... 23755-26868
CLEAR ... 27999
DOKTOR .. 28000-30999
MONS 3 .. 31000-37009
GENS 3 .. 37010-45459
PR.OBL .. 65000
SAVE
```

1.4.1 Příkazy relokačního menu

1.4.1.1 Příkaz "C"

Po stisku "C" se v dialogovém řádku znázorní "RAMTOP" a program očekává vstup nové hodnoty RAMTOPu. Při zadávání nového RAMTOPu je třeba dodržet určitou opatrnost, abychom (nechtěně) nezničili některý ze strojových programů Dr.MG.

1.4.1.2 Příkaz "D"

Po stisku "D" se v dialogovém řádku objeví výzva k zadání počáteční adresy, od níž chceme umístit strojový kód modulu DOKTOR. Když místo čísla vložíme samotný ENTER, modul zůstane na původní adrese. Modul nelze zlikvidovat zadáním hodnoty 0 (viz příkazy "M", "G").

Pokud bychom pro přemístění modulu DOKTOR zadali adresu, která by měla za následek jeho přesah do některé z oblastí jiných přítomných modulů Dr.MG (MONS3, GENS3, PR.OBL), příkaz se nevykoná.

Modul DOKTOR řídí relokaci všech segmentů programu. Přemístění má vlastnosti tzv. "inteligentní kopie", t.j. nová lokalizace segmentů může překrývat jakoukoli část své původní polohy.

1.4.1.3 Příkaz "M"

Po stisku "M" program čeká na uložení nové požadované počáteční adresy MONS3. Pro příkaz platí táž pravidla jako u příkazu "D" - s tím rozdílem, že je možné pasivovat segment MONS3, když vložíme hodnotu menší než je aktuální hodnota RAMTOPu; zároveň se pasivuje segment PR.OBL, který organicky souvisí s MONS3. K pasivovanému bloku nemáme pak přístup z hlavního panelu příkazem "M". Pasivované segmenty sice "fyzicky" zůstávají v paměti na svém původním místě, ale z hlediska programu Dr.MG představují volnou oblast paměti. Můžeme ji tedy použít k různým účelům. Po pasivaci segmentu zároveň z obrazovky zmizí údaj o jeho hranicích.

1.4.1.4 Příkaz "G"

Zásady vztahující se na příkaz "M" platí i pro příkaz "G" s tím rozdílem, že po pasivaci GENS3 se nepasivuje segment PR.OBL. Zdrojový text v GENS3 je organickou součástí celého segmentu, což se odrazí ve zobrazení poslední adresy GENS3 v relokačním menu.

1.4.1.5 Příkaz "P"

Pracovní oblast je segment v paměti, kterou používá MONS3 při dekompilaci a krokování následovně:

Při dekompilaci se do ní ukládá tabulka návěští, přičemž jedno návěští zabírá 2 bajty paměti. Kromě toho se do něj uklá-

dá hranice zadaných datových oblastí v rámci dekompilace ("Od:", "do:"). Každá definovaná datová oblast zabírá 4 bajty paměti. Při krokování se do této oblasti ukládá informace potřebná pro návrat obsahu paměti do tvaru, který byl před zadáním přerušení (příkaz MONS3 "W"). Jedno přerušení zabírá 5 bajtů pracovní oblasti.

Segment PR.OBL není možné pasivovat samostatně vložením hodnoty 0.

1.4.1.6 Příkaz "S"

Stiskem "S" vyvoláme funkci SAVE. Na pásek se zapíše přesná kopie Dr.MG v aktuální konfiguraci, zobrazené v relokačním menu. Po zadání příkazu program požaduje uložení názvu celého komplexu. Když místo názvu vložíme samotný ENTER, příkaz se nevykoná. Pasivované segmenty programu jsou ignorovány.

Program se na pásek uloží s autostartem, který zabezpečí správný a kompletní zápis všech částí programu Dr.MG. Když jsme měli před zadáním příkazu "S" v GENS3 jakýkoli zdrojový text, stává se organickou součástí nahrávky. Totéž platí pro segment Basicu, včetně jeho případně připsané části.

1.4.2 Poznámky k relokaci

Relokaci MONS3 a PR.OBL je možné realizovat jen před prvním přivoláním MONS3. Pro účely ochrany před přepsáním při relokaci program chápe PR.OBL jako 20tibajtový blok. Podobně lze relokovat GENS3 jen před jeho prvním přivoláním z hlavního panelu. Toto omezení se netýká segmentu DOKTOR, který je možno přesouvat libovolně.

Pasivováním a relokací jednotlivých segmentů je možné dosáhnout optimální konfigurace pro konkrétní potřeby konkrétního uživatele. Je možné vytvořit (a přirozeně i uložit na pásek) následující varianty:

DR.+MONS3+GENS3
DR.+MONS3
DR.+GENS3
DR.

Když požadujeme použití samotného programu MONS3 nebo GENS3, je nutno uložit jej na pásek příkazem "S" z hlavního panelu. Informace o počáteční adrese a délce získáme z relokač-

ního menu. I při samostatném použití MONS3 či GENS3 je jejich zaváděcí adresa v paměti současně jejich standardní startovací adresou.

Je třeba poznamenat, že adresa pracovní oblasti, kterou můžeme měnit v rámci relokačního menu, je hodnotou předvolenou, kterou MONS3 použije, když na jeho výzvu "adr.prac.obl.:" odpovíme stiskem tlačítka ENTER. Když v MONS3 při dekompilaci udáme jinou konkrétní hodnotu než je předvolená, systém uloženou hodnotu použije v rámci jedné dekompilace. Předvolená hodnota zůstane nezměněná.

KAPITOLA 2 SPUŠTĚNÍ

Pokud budete někdy chtít použít MONS3 samostatně, pak vězte, že i sám o sobě je relokovatelný. Do počítače jej umístěte od adresy xxxxx, kterou uvedete v příkazu LOAD ""CODE xxxxx. Spustíte jej voláním této adresy. (Pro případ, že byste narazili někdy na neupravený MONS3, vězte dále, že budete-li jej potom chtít spustit znova, musíte volat adresu o 29 větší, než byla původní.)

Příklad:

Řekněme, že chcete MONS3 umístit od adresy #C000 (49152 dekadicky) a ihned jej poprvé spustit. Provedete to takto:

```
LOAD "MONS3" CODE 49152
RANDOMIZE USR 49152
```

Do neupraveného programu MONS3 znovu vstoupíme příkazem RANDOMIZE USR 49181 - tím se vynechá část programu, která přemisťuje adresy v rámci MONS3 při prvním vstupu.

MONS3 má na pásku délku zhruba 5K. Po umístění do počítače je to však téměř 6K v důsledku rozmístění tabulek relokačních adres, které následují za hlavním programem. MONS3 má svou vlastní zásobníkovou paměť (machine stack), takže je zcela soběstačný. V okamžiku vstupu do MONS3 se na několik sekund objeví copyrightové hlášení. Pak naskočí obraz paměťového kompasu (viz příloha). Ten vypisuje registry a stavové indikátory (flags) mikroprocesoru Z80 včetně jejich obsahů, dále 24-bajtový výsek paměti, soustředěný kolem aktuální hodnoty střelky kompasu (MEMORY POINTER) - adresa střelky a její obsah jsou mezi znaky ">" a "<". Na počátku je nastavena na adresu 0. Na prvním řádku obrazovky je dekompilovaná instrukce, adresovaná střelkou kompasu.

Pozn.: Všechna hexadecimální čísla v textu manuálu jsou předznamenána znakem "#", jaký pro ten účel používají MONS3 i GENS3. Tak např. hexadecimální číslo 12FE je v manuálu vyjádřeno jako #12FE.

Dále jsou oproti originálu změněny některé výrazy. FRONT PANEL (čelní panel) nazýváme jednoduše KOMPAS a MEMORY POINTER (paměťový ukazatel) je převeden na výraz STŘELKA kompasu. Tuto metaforickou proměnu považujeme za jednodušší i relativně logičtější pro komunikaci uživatele s programem.

Při vstupu do MONS3 jsou všechny adresy i jejich obsahy vypsány v hexadecimálním tvaru. Stisknutím tlačítek SYMBOL SHIFT a 3 (viz další kapitola) lze měnit vypisování adres (ne však jejich obsah) na tvar dekadický. Nezapomeňte však, že adresy musíte vkládat vždy jen hexadecimálně! Příkazy se vkládají z klávesnice jako odpověď na systémový kurzor " " pod zobrazením obsahu paměti. Lze je psát malými i velkými písmeny (přepínají se obvyklým způsobem pomocí CAPS SHIFT 2). Některé příkazy, jejichž účinek by při chybném použití mohl být katastrofální, vyžadují stisknutí jak klávesy SYMBOL SHIFT, tak klávesy příkazového znaku. Použití klávesy SYMBOL SHIFT je dále v této příručce reprezentováno zkratkou "ss"; např. ssZ znamená současný stisk kláves SYMBOL SHIFT a Z. Podobně cs1 znamená současný stisk tlačítek CAPS SHIFT a 1 (majitelé ZX Spectra + vědí, že v tomto případě jim stačí stisknout samotný EDIT). Příkazy mají okamžitý účinek - není třeba je ukončovat stiskem ENTERu. Neplatné příkazy program ignoruje. Celý panel kompasu se aktualizuje po provedení každého příkazu, takže můžete sledovat okamžité změny příkazem vyvolané. Mnohé příkazy vyžadují vstup hexadecimálního čísla - můžete vložit libovolný počet hexadecimálních číslic (0-9, A-F nebo a-f) a ukončit je jakoukoli číslicí, která není hexadecimální. Je-li koncový znak platným příkazem, pak se provede vzápětí po předchozím příkazu. Je-li koncovým znakem "-" (znaménko mínus), pak po vložení dostaneme zápornou hodnotu vloženého čísla ve formě dvojkového doplňku (two´s complement); např. 1800- dá #E800. Vložíte-li více než 4 číslice hexadecimálního čísla, pak se uchovají a zobrazí pouze poslední 4 číslice. Jestliže si v kterémkoli okamžiku přejete návrat do překladače BASICu, stiskněte CAPS SHIFT 1 neboli EDIT (pokud pracujete s modulem DOKTOR, EDIT vás pochopitelně přenese do tohoto modulu).

Důležitá poznámka:

MONS3 blokuje přerušení (DI) pro zajištění své správné funkce. Při práci s ním nesmí být přerušení uvolněno (EI). To samozřejmě neplatí, když spouštíte své programy z Basicu (modulu DOKTOR).

KAPITOLA 3 POUŽÍVANÉ PŘÍKAZY

Kdekoli se v textu manuálu používá klávesa ENTER k ukončení hexadecimálního čísla, lze de facto použít libovolný nehexadecimální znak (viz kapitola 1). Také "-" lze užít pro vyznačení mezery.

SYMBOL SHIFT 3 neboli "#"

Přepíná (mění) základ číselné soustavy, v níž jsou adresy zobrazeny, mezi 16 a 10. Při spuštění MONS3 jsou adresy zobrazeny hexadecimálně. Použijte ss3 ke změně na dekadické zobrazení a opět ss3 pro návrat k hexadecimálnímu formátu. Tento postup ovlivní párové registry a všechny adresy včetně adres generovaných disassemblerem, ale nezmění zobrazení obsahů adres - ty jsou vyjádřeny vždy jen hexadecimálně.

SYMBOL SHIFT 4 neboli "$"

Zobrazí jednu dekompilovanou stranu v jazyku assembler od adresy obsažené ve střelce kompasu. Užívá se ke zjištění, jaké instrukce v programu následují. Při programování zápisem strojového kódu přímo do paměti počítače můžeme tak kontrolovat, zda jsme někde neudělali chybu. Opakovaným stisknutím ss4 se vrátíme ke zobrazení kompasu. Stiskem jiné klávesy si vyžádáme výpis další disassemblované stránky.

ENTER

Paměťový kompas posune zobrazení adres o 1 výše. Ve střelce bude adresa o 1 vyšší než byla před stiskem ENTERu. Jednoduše řečeno - výpis se posune o 1 adresu nahoru.

CAPS SHIFT 7 (kurzor nahoru)

Výpis se posune o 1 adresu dolů.

CAPS SHIFT 5 (kurzor doleva)

Výpis se posune o 8 adres dolů. Používá se k rychlému krokování vzad.

CAPS SHIFT 8 (kurzor doprava)

Výpis se posune o 8 adres nahoru. Používá se k rychlému

krokování vpřed.

"," (čárka)

Ve střelce se objeví obsah běžné (momentální) adresy zásobníku (indikovanou v SP). To je užitečné, chceme-li si prohlédnout okolí návratové adresy volaného programu atd.

"G" (Get - dostaň)

Vyhledání zadaného řetězce v paměti ("G" a řetězec).

Po systémovém náznaku ":" vložte první bajt ukončený ENTERem. Po každém dalším náznaku ":" pokračujte ve vkládání dalších bajtů, dokud nezadáte celý řetězec. Pak v odpověď na další ":" stiskněte jen ENTER. Tím je ukončeno definování řetězce. MONS3 prohledává paměť od adresy střelky vzhůru. Při nalezení řetězce se aktualizuje obsah kompasu tak, že jeho střelka obsahuje adresu prvního znaku řetězce.

Příklad:

Chceme zjistit výskyt řetězce #3EFF (2 bajty) počínaje adresou #8000. Budeme postupovat takto:

M:8000 ENTER	nastavení střelky kompasu na #8000
G:3E ENTER	určení 1. bajtu řetězce
FF ENTER	určení 2. bajtu řetězce
ENTER	ukončení řetězce

Po závěrečném ENTERu (nebo některém nehexadecimálním znaku) příkaz "G" hledá v paměti (od adresy #8000) první výskyt řetězce #3EFF. V okamžiku nalezení se aktualizuje zobrazení kompasu. V hledání dalších výskytů téhož řetězce můžeme pokračovat opakovaným stiskem tlačítka "N" (viz níže). Je nutno si uvědomit, že i když v paměti hledaný řetězec původně nebyl, MONS3 jej nalezne - tam, kam si jej sám při jeho definici uložil. Díky tomu můžeme posledně definovaný řetězec hledat kdykoli znova, aniž bychom jej museli znova definovat. Dále je důležité vědět, že prohledávání paměti probíhá cirkulačně, čili - poté, co MONS3 při prohledávání paměti narazí na adresu #FFFF, pokračuje od adresy #0000 opět vzhůru.

"H" (Hexadecimal)

Převádí dekadické číslo na jeho hexadecimální ekvivalent. K systémovému znaku ":" vložte dekadické číslo, které ukončíte

libovolným nečíselným znakem (třeba ENTERem). Jakmile číslo ukončíte, zobrazí se na stejném řádku znak "=" a za ním hexadecimální ekvivalent dekadického čísla. Stiskem jakékoli klávesy se vrátíte do příkazového režimu.

Příklad:

H:41472 =A200 (mezera je použita jako koncový znak)

"I"

Inteligentní kopie.

Příkaz kopíruje blok paměti z jedné pozice na jinou. Jeho inteligence spočívá v tom, že může překopírovat blok i na místo, kde překryje svou původní polohu. Příkaz "I" vás vyzve ke vložení počáteční a koncové adresy přenášeného bloku ("Od:, do:") a první adresy, od které bude celý přenášený blok nově umístěn ("kam"). V odpověď na tyto výzvy vložte hexadecimální čísla. Vložíte-li počáteční adresu přenášeného bloku větší než koncovou, příkaz se neprovede. V opačném případě se blok přesune dle příkazu. Nezapomeňte, že se jedná o kopírování bloku. Tzn., že nepřepsané adresy původního bloku si svůj obsah zachovají.

"J" (Jump - skoč)

Spuštění programu od dané adresy.

Tento příkaz si vyžaduje vložení hexadecimálního čísla. Poté se vynuluje vnitřní zásobníková paměť, vymaže kompas a provádění programu začne od dané adresy. Chcete-li se vrátit ke zobrazení kompasu po provedení programu (resp. kdykoli v jeho průběhu), nastavte bod přerušení (viz příkaz "W") na adresu, od které návrat požadujete.

Příklad:

J:B000 ENTER (spustí program od adresy #B000)

Před ukončením zápisu adresy (resp. spuštěním programu) lze příkaz ještě zrušit použitím CAPS SHIFT 5. Povšimněte si, že "J" ničí obsah registrů Z80 před výkonem programu; proto by program ve strojovém kódu neměl mít žádné vstupní podmínky, týkající se počátečních hodnot registrů. Chcete-li spustit program s registry nastavenými na určité hodnoty, použijte příkaz ssK:

"SYMBOL SHIFT K"

Pokračuje v provádění programu od běžné adresy v čítači instrukcí (PC). Tento příkaz bude pravděpodobně nejužívanějším ve spojitosti s příkazem "W". Příklad by měl vše objasnit:

Předpokládejme, že krokujete v programu uvedeném dále a dosáhli jste adresy #8920. Nemáte zájem o krokování podprogramem na adrese #9000, ale chcete vidět, jak jsou nastaveny indikátory po návratu z podprogramu na adrese #8800 (volaného instrukcí CALL #8000).

```
891E   3EEF        LD A,-1
8920   CD0090      CALL #9000
8923   2A0080      LD HL, (#8000)
8926   7E          LD A, (HL)
8927   111488      LD DE, #8814
892A   CD0088      CALL #8800
892D   2003        JR NZ,lab1
892F   320280      LD (#8002),A
8932   211488 lab1 LD HL, #8814
```

Postupujte takto:

Příkazem "W" nastavte bod přerušení na adresu #892D (nezapomeňte nejdřív použít příkaz "M" k nastavení střelky kompasu) a zadejte příkaz "ssK". Provádění programu pokračuje od adresy uložené v čítači instrukcí (PC), který má v tomto případě hodnotu #8920. Program probíhá až k adrese, na které je zadán bod přerušení (#892D). V tomto okamžiku se aktualizuje obsah kompasu - tak můžete zkoumat stavy indikátorů atd. po návratu z podprogramu na adrese #8800. Chcete-li, v krokování programem můžete pokračovat. Příkaz "ssK" je tedy užitečný pro provádění programu bez vynulování zásobníkové paměti a zničení obsahu registrů, jak to dělá příkaz "J". Nezapomeňte, že před zadáním příkazu "ssK" musí být obsah PC shodný s adresou střelky (viz krokování).

"L" (List)

Výpis bloku paměti od adresy střelky kompasu.

Příkaz "L" vymaže kompas a zobrazí prvních 80 bajtů od adresy střelky. Výpis je hexadecimální, připojeny jsou ekvivalenty ASCII. Adresy budou vyjádřeny buď hexadecimálně nebo dekadicky v závislosti na módu příkazu ss3 (viz výše). Zobrazení

se skládá z 20 řádek; každá obsahuje 4 bajty, ekvivalenty ASCII jsou na konci každé z nich. Pro účely zobrazení jsou všechny hodnoty nad 127 zmenšeny o 128 a hodnoty v rozmezí 0-31 včetně se zobrazí jako ".". Na konci stránky výpisu se buď můžete vrátit ke zobrazení kompasu (příkazem CAPS SHIFT 5), nebo pokračovat další stranou výpisu (opět 80 bajtů) stiskem jiné libovolné klávesy (kromě CAPS SHIFT 1 neboli EDIT).

"M" (Memory pointer - paměťový ukazatel)

Umístí střelku na danou adresu.

Ke znaku ":" vložte hexadecimální adresu (viz kapitola 1), Obsah střelky je pak aktualizován touto adresou. Zobrazení paměti na kompasu se odpovídajícím způsobem změní.

Příkaz "M" je účelný pro přímé vkládání strojového kódu, výpis paměti atd.

"N" (Next - další, příští)

Hledá další výskyt hexadecimálního řetězce naposledy zadaného příkazem "G" (viz výše). "G" vám umožní definovat řetězec a hledat jeho první výskyt v paměti. Chcete-li znát další místo výskytu, použijte příkaz "N". "N" začne hledat od adresy v paměťové střelce. Při nalezení dalšího řetězce je střelka aktualizována.

"O"

Přechod na místo určení relativní adresy.

Příkaz čte bajt adresovaný střelkou. Zachází s ním jako s relativní adresou a odpovídajícím způsobem aktualizuje zobrazení paměti.

Příklad:

Střelka je nastavena na adresu #6800, obsah adres #67FF a #6800 je #20 a #16. To může být interpretováno jako instrukce JR NZ, $+24 ($ je aktuální obsah čítače adres PC). Abychom zjistili, kam tato větev povede při nenulové podmínce, zadáme příkaz "O", když střelka ukazuje na bajt #16. Zobrazení kompasu bude aktualizováno a soustředěno kolem adresy #6817, což je hledané místo určení této větve.

Nezapomeňte, že s hodnotou relativního skoku větší než #7F (127) mikroprocesor zachází jako s hodnotou zápornou

(dvojkový doplněk). Příkaz "O" to respektuje. Ve spojení s "O" viz též příkaz "U".

"P"

Naplní paměť v daném intervalu určeným bajtem.

Příkaz si vyžádá počáteční ("od:") a koncovou ("do:") adresu bloku, který chcete zaplnit. Dále žádá ("znak:") hodnotu, kterou mají obsahovat všechny adresy zadaného bloku.

Příklad:

```
P
od:7000 ENTER
do:77FF ENTER
znak:55 ENTER
```

Zaplní oblast od #7000 do #77FF (včetně) bajtem #55 (znakem "U"). Je-li počáteční adresa větší než koncová, příkaz se neprovede.

"Q"

Změna sady registrů.

Po každém spuštění MONS3 paměťový kompas zobrazuje standardní sadu registrů (AF, HL, DE, BC). Příkaz "Q" přepne na zobrazení sady registrů alternativních (AF´,HL´,DE´,BC´), která je od standardní sady odlišena apostrofem za názvem registrů.

Příkaz účinkuje i opačně (přepíná).

SYMBOL SHIFT T

Nastaví bod přerušení za běžnou instrukci (počínající ve střelce) a provede běh programu až k bodu přerušení.

Příklad:

```
9000 B7     OR A
9001 C20098 CALL NZ,#9800
9004 010000 LD BC,0
9800 21FFFF LD HL,-1
```

Krokováním ve výše uvedeném programu dojdete na adresu #9001 s nenulovou hodnotou A, takže nulový indikátor bude po instrukci OR A ve stavu NZ (Z=0). Jestliže pro další krokování teď použijete příkaz "ssZ", program bude pokračovat na adrese podprogramu, tj. #9800. Nepřejete-li si krokovat tímto programem, pak při dosažení adresy #9001 zadejte příkaz "ssT" -

instrukce CALL se provede automaticky a program se zastaví na adrese #9004, kde můžete v krokování pokračovat. Pamatujte si, že "ssT" nastavuje bod přerušení za instrukci, uloženou od aktuální adresy ve střelce, a pak vlastně provede příkaz "ssK".

Viz i příkaz "ssZ" v rozšířeném příkladu krokování.

"T"

Dekompilace bloku programu s možností výstupu na tiskárnu.

Nejprve vás program vyzve ke vložení počáteční ("od:") a koncové ("do:") adresy programu, který si přejete dekompilovat (disassemblovat) - vložte ji hexadecimálně (podrobně vysvětleno v kapitole Z). Je-li počáteční adresa větší než koncová, příkaz se neprovede. Po vložení adres budete dotázáni "tlač Y/N?" (tiskárna?). Odpověď "Y" nasměruje výpis dekompilovaného programu na vaši tiskárnu - jakákoli jiná hodnota jej odešle na obrazovku.

Poté jste vyzváni slovem "zdroj.text Y/N" ke vložení (hexadecimální) počáteční adresy zdrojového textu, který má dekompilátor vyhotovit. Nechcete-li, aby byl generován textový soubor, neuvádějte žádnou adresu, ale stiskněte jen ENTER. Udáte-li adresu, pak bude vyhotoven dekompilovaný textový soubor ve formě použitelné pro GENS3 - v něm jej můžete použít pochopitelně za toho aktuálního předpokladu, že máte GENS3 v paměti počítače a byl již aspoň jednou volán. (viz i funkční příklad v manuálu).

Když chcete textový soubor použít v GENS3, pak jej musíte buď v něm generovat, nebo do něj převést. První adresu udá příkaz "X" edičního programu assembleru GENS3 - to je adresa počátku textu, kterou GENS3 očekává. Assembleru GENS3 musíte rovněž sdělit, kde textový soubor končí. To provedete tak, že adresu uvedenou u hlášení dekompilátoru "koniec:" (viz dále) vložíte do místa TEXTEND V GENS3 (viz manuál GENS3, kapitola 6.2). Pokud užijete neupraveného programu GENS3, musíte pak do něj vstoupit horkým startem, aby text zůstal zachován. U našeho upraveného se o to nemusíte starat.

Když při generování textového souboru začne text přepisovat MONS3, dekompilace se přeruší - stiskem libovolné klávesy se vrátíte ke zobrazení kompasu.

Jestliže jste zadali adresu textového souboru, budete po-

žádáni o adresu pracovní oblasti ("adr.prac.obl.:") - to by měla být adresa oblasti volné paměti, která se využívá jako primitivní tabulka symbolů pro veškerá návěští generovaná disassemblerem. Požadovaný objem paměti je 2 bajty pro každé generované návěští. Zvolíte-li standardní hodnotu tím, že stisknete jen ENTER, pak systém automaticky zvolí adresu #6000 (použijete-li MONS3 bez modulu DOKTOR). Standardní hodnotu můžeme určit řídícím modulem Dr.MG - běžně je nastavena na adresu #FFDC (65000).

Poté budete opakovaně žádáni o první ("od:") a poslední ("do:") - včetně - adresu jakýchkoli dalších datových oblastí v rámci bloku, který chcete dekompilovat. Datové oblasti jsou takové, jejichž obsah nemá být interpretován jako instrukce Z80 (např. text hlášení apod.). Datové oblasti vyvolají generování assemblerových pseudoinstrukcí DEFB. Je-li hodnota datového bajtu mezi 32 a 127 (#20-#7F) včetně, pak proběhne interpretace ASCII kódu tohoto bajtu. Např. #41 se změní na "A".

Po ukončení specifikace datových oblastí (resp. nechcete-li specifikovat žádnou) stiskněte v odpověď na požadovanou adresu samotný ENTER.

Příkaz "T" využívá paměťovou oblast na konci MONS3 k ukládání adres datových oblastí. Můžete určit jen tolik oblastí, kolik paměť obsáhne. Každá datová oblast vyžaduje 4 bajty paměti. Mějte na zřeteli, že použitím příkazu "T" zrušíte veškeré body přerušení, které jste před tím případně nastavili (viz příkaz "V")!

Nyní se vymaže obsah obrazovky. Jestliže jste si vyžádali tvorbu textového souboru, bude následovat krátká pauza (závisí na velikosti výseku paměti, který chcete dekompilovat). Během ní se sestavuje tabulka symbolů. Jakmile je její stavba ukončena, objeví se výpis dekompilace na obrazovce nebo na tiskárně. Výpis se po stisknutí tlačítka ENTER nebo SPACE zastaví na konci vypisovaného řádku. Pokračovat ve výpisu můžete stiskem libovolné klávesy (kromě cs1). Při zastaveném výpisu se lze stiskem ss5 vrátit ke zobrazení kompasu.

Neplatný operační kód je dekompilován jako NOP a indikován znakem "x" za výpisem operačního kódu.

Na konci dekompilace se zobrazení zastaví. Pokud jste zadali vytvoření textového souboru, zobrazí se zpráva

"koniec:xxxxx". xxxxx je adresa (hexadecimální nebo dekadická), která má být zapsána (např. pomocí příkazu POKE) na pozici TEXTEND v GENS3, aby se assembler při horkém startu v dekompilovaném textovém souboru orientoval (viz manuál GENS3). Když je dekompilace ukončena, stiskněte libovolnou klávesu pro návrat ke zobrazení kompasu - kromě příkazu cs!, kterým se provádí návrat do BASICu.

Návěští se generují ve formátu Lxxxx, kde L je zkratka angl.slova Label (přímý překlad je obálka, naše terminologie pro ni používá slovo návěští) a xxxx je absolutní (hexadecimální) adresa návěští. Adresa musí být v rozmezí dekompilace. Je-li adresa mimo tento rozsah, návěští se negeneruje a vypíše se pouze adresa (hexadecimální nebo dekadická).

Příklad:

Při zpracování oblasti #7000-#8000 bude instrukce C30078 dekompilována jako JP L7800. Při dekompilaci téže instrukce v oblasti adres #8000-#9000 bude disassemblována jen ve tvaru JP #7800 (nebo JP 30720, máme-li určen dekadický výpis). Jestliže je v instrukci odkaz na určitou adresu v rámci dekompilace, její návěští se vypíše v poli návěští (před mnemonikou) dekompilované instrukce na této adrese - ale jen když je výpis směrován do textového souboru (tedy je vložena první i poslední adresa textového souboru pro příkaz "T").

Příklad:

```
T
od:8B ENTER
do: 9E ENTER
tlač Y/N? Y
zdroj.text Y/N?: ENTER
od: 95 ENTER
do: 9E ENTER
od: ENTER
do: ENTER
008B    FE16    CP #16
008D    3801    JR C,L0090  (L, angl.label,značí návěští)
008F    23      INC HL
0090    37      SCF
0091    225D5C  LD (#5C5D),HL
0094    C9      RET
```

```
0095    BF524E  DEFB  # BF, "R", "N"
0098    C4494E  DEFB  # C4, "I", "N"
009B    4B4559  DEFB  "K", "E", "Y"
009E    A4      DEFB  # A4
```

"U"

Používá se ve spojení s příkazem "O".

Vzpomeňte si, že příkaz "O" aktualizuje zobrazení paměti ve vztahu k posuvu (displacement) instrukcí relativního adresování. Provede tedy skok instrukcí JR nebo DJNZ na určitou adresu. Příkaz "U" nastaví a zobrazí střelku zpět v místě, v němž byl zadán poslední příkaz "O".

Příklad:

7200	47	71F3	77
7201	20	71F4	C9
>7202	F2<	>71F5	F5<
7203	06	71F6	C5
Kompas 1		Kompas 2	

Máte zobrazen kompas 1 a přejete si vědět, kam se větví relativní skok 20F2. Stiskněte "O". Objeví se kompas 2. Teď můžete prošetřit instrukce, následující za #71F5. Chcete-li se vrátit k instrukci, která následuje za původním relativním skokem - abyste mohli zjistit, co se stane, když je nastaven nulový indikátor (Z=0) - stiskněte "U". Tak se dostanete opět na kompas 1.

Uvědomte si, že příkazem "U" se můžete vrátit pouze k místu posledního zadání příkazu "O". Všechna předešlá jsou ztracena.

"V"

Používá se ve spojení s příkazem "X".

Příkaz "V" je podobný příkazu "U" co do účinku, kromě toho, že aktualizuje zobrazení paměti v místě, kde byl zadán poslední příkaz "X".

Příklad:

8702	AF	842D	18
8703	CD	842E	A2
>8704	2F<	>842F	E5<
8705	84	8420	21
Kompas 1		Kompas 2	

Máte zobrazení kompasu 1 a chcete se podívat na podprogram volaný instrukcí CALL #842F. Do střelky umístěte bajt #2F dané instrukce (jak uvedeno) a stiskněte "X". Ukáže se výpis paměti kolem adresy #842F (kompas 2). Chcete-li se vrátit k instrukci původního volání tohoto podprogramu, stiskněte "V". Tak se dostanete zpátky na kompas 1. Stejně jako u příkazu "U" se můžete vrátit pouze na adresu, kde byl zadán poslední příkaz "X". Všechny předchozí zadané příkazy "X" jsou ztraceny.

"W"

Nastaví bod přerušení do střelky.

Bod přerušení je interní instrukce volání (CALL) v programu MONS3, která programátorovi umožní zastavit běh prošetřovaného programu a na zobrazeném kompasu zkoumat obsah registrů, indikátorů Z80, jakož i příslušné oblasti paměti.

Příklad:

Chcete-li zastavit běh programu třeba na adrese #9876, použijte nejdřív příkaz "M" k umístění adresy #9876 do střelky a příkaz "W" pro nastavení bodu přerušení na této adrese. Tři bajty od adresy #9876 výše se přemístí do "úschovny" v MONS3 a jsou nahrazeny instrukcí CALL, která zastavuje běh programu. Při programovém dosažení této instrukce se původní tři bajty opět vrátí na své místo a zobrazí se kompas se všemi registry a indikátory ve stavu těsně před bodem přerušení. Dále můžete využít všech schopností MONS3 obvyklým způsobem.

Poznámky:

MONS3 využívá pro uložení informace o bodech přerušení oblast paměti na svém konci. Můžete nastavit tolik bodů přerušení, kolik tato část paměti obsáhne. Každý bod přerušení vyžaduje 5 bajtů. Po vykonání všech přerušení obnoví MONS3 obsah paměti do stavu před nastavením bodu přerušení. Ale pozor - všechny body přerušení se zruší, když použijete příkaz "T"! Ten pro svou funkci využívá tutéž paměťovou oblast.

Protože bod přerušení je tvořen tříbajtovou instrukcí CALL, je třeba věnovat trochu pozornosti určitým zvláštním případům:

8000	3E	8008	00
8001	01	8009	00
8002	18	800A	06
8003	06	800B	02

```
>8004  AF        800C  18 <
 8005  0E        800D  F7
 8006  FF        800E  06
 8007  01        800F  44
```

Když nastavíte bod přerušení na #8004 a spustíte program od #8000, nejdříve se do registru A vloží hodnota 1. Program přejde na #800A a do registru B se vloží hodnota 2. Program pokročí na adresu #8005. Ale ta byla přepsána spodním bajtem instrukce přerušení!!! Tím se instrukce zkomolila a přinese nechtěné výsledky. Tato situace je spíše neobvyklá, ale mějte na paměti možnost jejího vzniku. V tomto případě vyhoví krokování (viz příkaz ssZ dále).

"X"

Aktualizuje střelku hodnotou absolutní adresy místa určení instrukce CALL nebo JP.

"X" sestavuje 16tibitovou adresu tak, že její nižší část tvoří obsah adresy střelky, vyšší část si odebere z adresy o 1 vyšší (za střelkou). Zobrazení kompasu je pak soustředěno kolem této adresy. Pamatujte, že v instrukcích Z80 je nižší polovina adresy dána prvním bajtem a vyšší polovina adresy druhým bajtem - formát INTEL.

Příklad:

Chceme se podívat na podprogram volaný instrukcí CALL #6305 (CD0563). Nastavíme střelku tak, aby adresovala bajt 05 instrukce CALL, a stiskneme "X". Paměťový kompas se soustředí kolem adresy #6305.

Viz též příkaz "V" v souvislosti s příkazem "X".

"Y"

Vkládá ASCII kódy do paměti od adresy střelky.

Příkaz "Y" vygeneruje řádek, na který můžete vkládat znaky ASCII přímo z klávesnice. Tyto znaky jsou zpětně kontrolovány a jejich hexadecimální hodnoty se ukládají do paměti od běžné adresy střelky. Řetězec znaků se ukončí stiskem cs5 (na to pozor - basicově-inputový návyk nás svádí ke stisku ENTERu). DELETE lze použít k vymazání znaku z řetězce. Po vložení znaku a stisknutí cs5 se zobrazení aktualizuje tak, že střelka adresuje místo o 1 vyšší než je adresa posledního znaku právě vlo-

ženého řetězce.

SYMBOL SHIFT Z

Krokování

Před použitím ssZ (nebo ssT) musíte nastavit jak programový čítač (PC), tak i střelku na stejnou adresu instrukce, od níž chcete krokovat programem. Příkaz ssZ provede stávající instrukci a zaktualizuje kompas tak, že odpovídá změnám, které instrukce vyvolala. Pamětí počítače můžete krokovat kdekoli v RAM i ROM. Protože se krokované instrukce zároveň provádějí, musíte zajistit, aby nebyla aktivována přerušení (EI).

Dále je uveden rozsáhlý příklad, který by měl objasnit použití mnoha odlaďovacích příkazů obsažených v MONS3. Doporučujeme vám pečlivě jej prostudovat a vyzkoušet.

Předpokládejme, že v počítači máme tři části programu ve strojovém kódu. První část tvoří hlavní program, který plní registry HL a DE čísly pro násobení, a pak volá podprogram (2.část), který čísla vynásobí a výsledek uloží do HL. Nakonec dvakrát volá podprogram, který vyšle výsledek na obrazovku (3.část).

```
7080  2A0072        LD   HL,(#7200)        ;CAST 1
7083  ED5B0272      LD   DE,(#7202)
7087  CD0071        CALL Mult
708A  7C            LD   A,H
708B  CD1D71        CALL Aout
708E  7D            LD   A,L
708F  CD1D71        CALL Aout
7092  210000        LD   HL,0
.
.
.
7100  AF     Mult   XOR  A                 ;CAST 2
7101  ED52         SCB  HL,DE
7013  19           ADD  HL,DE
7014  3001         JR   NC,Mu1
7016  EB           EX   DE,HL
7017  B2     Mu1   OR   D
7018  37           SCF
7019  C0           RET  NZ
```

```
701A   B3          OR   E
701B   5A          LD   E,D
701C   2007        JR   NZ,Mu4
701E   EB          EX   DE,HL
701F   C9          RET
7110   EB    Mu2   EX   DE,HL
7111   19          ADD  HL,DE
7112   EB          EX   DE,HL
7113   29    Mu3   ADD  HL,HL
7114   D8          RET  C
7115   1F    Mu4   RRA
7116   30FB        JR   NC,Mu3
7118   B7          OR   A
7119   20F5        JR   NZ,Mu2
711B   19          ADD  HL,DE
711C   C9          RET

711D   F5    Aout  PUSH AF              ;CAST 3
711E   0F          RRCA
711F   0F          RRCA
7120   0F          RRCA
7121   0F          RRCA
7122   CD2678      CALL Nibble
7125   F1          POP  AF
7126   E60F  Nibble AND 1111
7128   C690        ADD  A, #90
712A   27          DAA
712B   CE40        ADC  A, #40
712D   27          DAA
712E   FD213A5C    LD   IY, #5C3A
7132   D7          RST   #10
7133   C9          RET
.
.
7200   0B2A        DEFW 10779
7202   0300        DEFW 3
```

Prošetříme výše uvedený program, abychom zjistili, jak pracuje, nebo zda vůbec pracuje. Provedeme to následující řadou příkazů. Upozorňujeme, že je to pouze jeden z více možných způsobů krokování programem a nemusí být nezbytně nejúčinnější.

Spíše slouží jako vhodná ukázka postupného krokování:

```
M:7080 ENTER   Nastaví paměťovou střelku na #7080
7080           nastaví čítač instrukcí na #7080
ssZ            krok
ssZ            krok
ssZ            sledujeme volání (CALL)
M:7115 ENTER   vynecháme přípravu čísel
W              nastaví bod přerušení (BP)
ssK            pokračuje provádění od #7100 do BP
ssZ            krok
ssZ            krok
ssZ            krok
ssZ            krok
ssZ            krok
ssZ            krok
ssZ            krok
ssZ            návrat z podprogramu pro násobení
ssZ            krok
ssZ            sledujeme volání (CALL)
M:7128 ENTER   nastaví paměťovou střelku na zajímavou část
W              nastaví bod přerušení (BP)
ssK            pokračuje v provádění od #711D do BP
ssZ            krok
ssZ            krok
ssZ            krok
ssZ            krok
;              podíváme se na adresu návratu
W              nastavíme na ni bod přerušení
ssK            a pokračujeme
ssZ            krok
;              návrat z podprogramu Aout
W
ssK
ssZ            krok
ssT            provedení CALL na Aout
```

Projděte si uvedený příklad. Nejdříve zapište instrukce programu podle návodu v odstavci "Změny obsahu paměti" (nebo použijte GENS3). Pak proveďte uvedené příkazy. Zjistíte, že příklad je neobyčejně cenný pro pochopení postupu práce s MONS3

při zjišťování funkce jakéhokoli programu ve strojovém kódu.

SYMBOL SHIFT P

Příkaz je stejný jako příkaz "L", kromě toho, že výstup směřuje na tiskárnu, nikoli na obrazovku. Nezapomeňte na konci stránky buď stisknout cs5 pro návrat ke zobrazení kompasu, nebo si jinou libovolnou klávesou vyžádejte tisk další stránky.

Změny obsahu paměti

Obsah adresy, udané paměťovou střelkou, můžeme změnit vložením hexadecimálního čísla, za kterým následuje zakončovací znak (viz kapitola 1) - je dobré si zvyknout na používání jednoho, třeba ENTERu.

Dvě hexadecimální číslice nejnižšího řádu (pokud byla vložena pouze jedna, automaticky se zleva doplní nulou) se vloží na místo právě adresované paměťovou střelkou. Příkaz se provede, pokud byl použit zakončovací znak. Není-li zakončovací znak platným příkazem, je ignorován.

Příklady (povšimněte si, že u 3. a 4. příkladu se provedou oba příkazy vložené do příkazové řádky - na to někdy pozor!):

F2 ENTER	Vloží se #F2 a paměťová střelka postoupí o 1
123 CAPS SHIFT 8	Vloží se #23 a paměťová střelka postoupí o 8
EM:E000(mezera)	Vloží se #0E na pozici paměťové střelky a pak je paměťová střelka aktualizována na #E00. Všimněte si, že mezera zde byla použita pro ukončení příkazu "M".
8C0	Vloží se #8C, a pak je paměťová střelka aktualizována (protože byl použit příkaz "0") na hodnotu relativní adresy #8C, tj. na svou běžnou hodnotu - 115.
2A5D(mezera)	Vloží se #5D - paměťová střelka se nezmění (ukončovací znak je mezera, nikoli příkaz).

Změny obsahu registrů

Zapíšeme-li přímo k systémovému znaku ">" hexadecimální číslo a ukončíme je znakem ".", pak dané číslo bude vloženo do registru Z80, právě adresovaného kurzorem "-> ". Při vstupu do MONS3 ukazuje kurzor -> na čítač instrukcí. Proto užití "." jako ukončovacího znaku hexadecimálního čísla změní obsah

čítače instrukcí. Pokud budete chtít změnit obsah jiného registru, použijte napřed samotný znak "." - uvidíte, jak se kurzor -- cyklicky posunuje od registru PC k registru AF. Povšimněte si, že není možné adresovat (a tedy ani změnit) zásobníkovou paměť (SP) ani registry IR.

Příklad (začínáme s kurzorem v pozici PC):

```
.       adresujeme IY
.       adresujeme IX
0.      nastavíme IX na hodnotu 0
.       adresujeme HL
123.    nastavíme HL na hodnotu  #123
.       adresujeme DE
.       adresujeme BC
E2A7.   nastavíme BC na hodnotu  #E2A7
.       adresujeme AF
FF00.   nastavíme A  na hodnotu  #FF a vynulujeme indikátory
.       adresujeme PC
8000.   nastavíme PC na hodnotu  #8000
```

Nezapomeňte, že "." lze rovněž použít ke změně obsahu alternativních registrů (pokud jsou zobrazeny). Pro změnu zobrazení souboru registrů použijte příkaz "Q".

Příklad zobrazení kompasu

```
710C  2007      JR   NZ, #7115
-> PC 710C   20 07 EB C9 EB 19 EB
   SP D0AF   8A 70 06 03 0A 03 0D
   IY CF6A   0D 11 0C 0F 09 18 18
   IX D09F   04 03 04 00 00 00 1B
   HL 2A1B   0F FE 29 28 02 CF 02
   DE 0000   F3 AF 11 FF FF C3 CB
   BC 0004   FF C3 CB 11 2A 5D 5C
   AF 0304      V
   IR 3F7C

7100  AF   7108  37   7110  EB
7101  ED   7109  C0   7111  19
7102  52   710A  B3   7112  EB
7103  19   710B  5A   7113  29
7104  30  >710C  22<  7114  D8
7105  08   710D  27   7115  1F
```

```
7106  EB   710E  EB   7116  30
7107  B2   710F  C9   7117  FB
>
```

Zde je uvedeno poměrně typické zobrazení kompasu - toto zobrazení získáte při krokování podprogramu Mult (viz rozšířený příklad užití příkazu "SYMBOL SHIFT Z").

Prvních devět řádků zobrazuje registry Z80. Nejprve je uveden název registru, pak okamžitá hodnota registrů a konečně obsah sedmi paměťových míst, počínaje adresou uloženou v registru. Registr indikátorů je dekódován, aby ukázal okamžité nastavení indikátorů v pořadí, v jakém jsou uspořádány v registru F - má-li registr indikátorů hodnotu #FF, pak zobrazení bude vypadat takto:

```
OOFF   SZ HVNC
```

což znamená, že jsou nastaveny indikátory SIGN, ZERO, HALF, CARRY, PARITY/OWERFLOW, ADD/SUBTRACT a CARRY. Kurzor --> označuje běžně adresovaný registr (viz "Změny obsahu registrů").

Pod zobrazením registru následuje zobrazení 24 bajtů paměti a každá z osmi řádek je uspořádána takto:

adresa	obsah	adresa	obsah	adresa	obsah
2 bajty (4 znaky)	1 bajt (2 znaky)	2 bajty	1 bajt	2 bajty	1 bajt

Zobrazení je soustředěno kolem pozice střelky, značené "><". Příkaz se zapisuje na spodní řádek obrazovky za systémový znak ">". Zobrazení se aktualizuje po provedení každého příkazu.

KAPITOLA 4 SPUŠTĚNÍ

GENS 3 je assemblérový generátor strojového kódu instrukčního souboru mikroprocesoru Z80 (zkráceně se označuje jako assembler). Na rozdíl od některých jiných assemblerů pro mikropočítače je GENS3 rozsáhlým profesionálním softwarem. Proto doporučujeme prostudovat následující kapitoly spolu s příklady velmi pozorně. Jste-li assemblerový začátečník, začněte kapitolou 6.3.

GENS3 je zhruba 7K dlouhý. Používá svou vlastní zásobníkovou paměť (machine stack), takže je zcela autonomní. Obsahuje vlastní řádkový editor, který umísťuje textový soubor hned za GENS3. Za tento soubor se zapisuje tabulka symbolů assembleru. Umístění GENS3 do paměti musíte proto volit tak, aby zbyl dostatek místa jak pro vlastní assembler, tak tabulku symbolů i text, který budete při práci potřebovat. Častěji bývá vhodnější umístit GENS3 do spodní části paměti.

Budete-li chtít použít GENS3 bez modulu DOKTOR, načtěte jej do počítače následovně:

LOAD "GENS3" CODE xxxxx - kde xxxxx je první adresa, od které se GENS3 začne do paměti počítače umísťovat. Začátečníky tak upozorňujeme na to, že jakýkoli CODE (nejen GENS3 a MONS 3) lze do počítače umístit od libovolné adresy xxxxx v rozsahu paměti RAM (samozřejmě v závislosti na délce programu).

Po načtení GENS3 z pásku do počítače vstoupíte do assembleru příkazem RANDOMIZE USR xxxxx, kde xxxxx je táž adresa, jakou jste zapsali do výše uvedeného příkazu LOAD. Narazíte-li někdy na neupravenou verzi GENS3, pak vězte, že když při svých experimentech z GENS3 vystoupíte, můžete se do něj vrátit startem z jedné ze dvou adres:

a) xxxxx + 56 pro studený start
(dojde k vymazání všeho, co jste dosud zapsali)

b) xxxxx + 61 pro horký start
(vytvořený text zůstane zachován)

Přímo adresou xxxxx se neupravený GENS3 spouští jen poprvé (rozmísťuje se při něm část jeho strojového kódu).

Díky své autonomii funguje GENS3 i MONS3 bez vady i na Microdrivu a Wafadrivu bez jakýchkoli úprav.

Když GENS3 spustíte poprvé, na obrazovce se objeví nápis "dlžka bufferu:". Počítač čeká na vložení čísla od 0 do 9 včetně. Stiskem samotného ENTERu je automaticky vložena standardní hodnota 4. Vložené číslo určuje velikost paměti pro vkládání v blocích po 256 bajtech (viz kapitola 5.8 - vkládání). Chcete-li minimalizovat prostor zabraný GENS3, nebo nepotřebujete-li možnosti vkládání využít, vložte 0 - tak bude pro buffer vyhrazen nejmenší možný prostor 64 bajtů.

Po stanovení rozsahu bufferu se objeví znak ">". Znamená, že editor je připraven přijmout vaše povely a zápisy. V kapitole 6 najdete JAK, v kapitole 5 CO do editoru zapisovat.

Důležité upozornění - z důvodu vyšší rychlosti a lepší vnitřní komunikace programu je přerušení blokováno (DI). To znamená, že každý program, který přerušení používá, musí být vybaven instrukcí pro jeho uvolnění (EI) před tím, než bude požadováno. Rovněž je třeba dát pozor na to, že GENS3 interně používá registr IY!!!

KAPITOLA 5 PODROBNOSTI GENS3

5.0 Jak pracuje GENS3

GENS3 je rychlý assembler se dvěma průběhy. Lze jej přizpůsobit většině systémů osazených mikroprocesorem Z80. Zahrnuje kompilaci všech standardních instrukcí Z80. Skýtá možnost využití např. podmíněné kompilace, pseudoinstrukcí, binárně větvené tabulky symbolů atd.

Spustíte-li assembler (příkazem "A" - viz kap. 6), jste nejprve dotázáni na velikost tabulky symbolů "velkosť tab:" (dekadicky). Je to prostor paměti, který bude tabulce vyhražen v průběhu vaší další práce s editorem. Pouhým stiskem tlačítka ENTER GENS3 automaticky určí standardní velikost, kterou považuje za přiměřenou velikosti textu (zpravidla úplně postačí). Uvědomte si ovšem, že při vkládání (INCLUDE - viz dále) je pro tabulku symbolů nutno zvolit větší prostor než standardní; assembler nemůže předvídat objem souboru, který se rozhodnete vložit.

Po výzvě "velkosť tab:" se vás program dotáže, jaký "typ činnosti:" (resp. více typů činnosti) požadujete. V případě,

že zadáte více než jeden typ činnosti, vložte dekadický součet jejich indexů podle následující tabulky:

Činnost 1	Tvorba seznamu použitých symbolů na konci druhého průběhu assembleru
Činnost 2	Zabránění generace strojového kódu
Činnost 4	Zabránění výpisu výsledku kompilace
Činnost 8	Nasměrování výpisu kompilace na tiskárnu
Činnost 16	Umístění strojového kódu za tabulku symbolů. Čítač adresy je stále aktualizován pomocí ORG, takže strojový kód může být umístěn v jedné části paměti, ale pracovat může v jiné
Činnost 32	Vypnutí kontroly umístění strojového kódu - užitečné pro zrychlení kompilace

Příklad:

Činnost 36 (tj. 32 + 4) provede rychlou kompilaci - bez výpisu a bez kontroly, kam bude strojový kód umístěn.

Při použití činnosti 16 nelze použít příkaz assembleru ENT (viz dále). Můžete však zjistit umístění strojového kódu použitím příkazu "X", který nalezne konec textu (druhé zobrazené číslo), a k němu přičíst velikost tabulky symbolů +2.

Kompilace probíhá ve dvou průbězích!!!

Během 1. průběhu (chodu) GENS3 hledá chyby a kompiluje tabulku symbolů.

Při 2. průběhu se vytvoří strojový kód (pokud není zvolena činnost 2).

Při 1. průběhu se na obrazovce (resp. ani na tiskárně) nic neobjeví, pokud není zjištěna chyba - v tom případě se zobrazí chybný řádek a pod ním počet chyb (viz příloha 1). Kompilace je přerušena - stiskem "E" se provede návrat do editoru; stiskem jakéhokoli jiného tlačítka kompilace pokračuje od následujícího řádku. Na konci prvního průběhu se objeví informace:

chod 1 chyby: nn

(nn je počet chyb)

Vyskytne-li se chyba, assembler se zastaví a neprovede 2. průběh. Je-li užit symbol v poli operandu, který není definován v tabulce, objeví se zpráva "POZOR! návěští neexistuje" pro každou chybějící definici. Na místě slova se ve zprávě

zobrazí (resp.vytiskne) přímo jeho název.

Ve 2. průběhu se generuje strojový kód (pokud jeho tvorbě není zamezeno činností 2). V tomto průběhu se rovněž vytváří výpis výsledků kompilace (pokud ovšem nebyla zvolena činnost 4 nebo příkaz assembleru ≡L-).

Výpis má tvar:

```
C000 210100        25 skok1
                      LD    HL,1
1    6       15       21    26  ........(č.sloupce)
```

Na prvním místě je hodnota čítače adres (zde #C000), která udává začátek zpracování tohoto řádku (pokud nejsou použity pseudoinstrukce jako ORG, EQU nebo ENT - viz kapitola 5.6). Zpravidla je udána hexadecimálně, ale může být zobrazena i dekadicky užitím příkazu ≡D+ (viz kapitola 5.8).

Na druhém místě, od sloupce 6, je max. 8 znaků (tedy až 4 bajty) instrukce s operandem - ale povšimněte si funkce příkazu ≡C uvedeného níže.

Následuje číslo řádku - celé číslo v rozmezí 1 až 32767 včetně.

Sloupce 21-26 prvního řádku obsahují prvních šest znaků návěští definovaného na tomto řádku (zde skok1, ale může to samozřejmě být jakékoli jiné, nerezervované slovo).

Další dva odstavce se týkají pouze systémů s "úzkou" obrazovkou. Systémy s délkou řádky větší než 40 znaků mají každou celou řádku assembleru uvedenu na jedné řádce obrazovky.

Po každém návěští následuje nová řádka - na něm je uvedena instrukce ve sloupcích 21-24. Následuje pole operandu od sloupce 26 a nakonec komentář, který je vkládán na konec řádky a generuje novou řádku, pokud je to nutné.

Uvedený formát usnadňuje čitelnost u systémů s "úzkou" obrazovkou, jako má SPECTRUM. GENS3 nemění počet znaků na obrazovce, neboť by to vyžadovalo rozšíření GENS3 a nebylo by možno použít standardních rutin paměti ROM.

Příkaz assembleru ≡C- můžeme použít ke zkrácení řádky. Způsobí vynechání devíti znaků (8 strojového kódu plus 1 mezeru) řádky a umožní přehlednější zobrazení více řádek na obrazovce (viz kapitola 5.8 níže).

Formát výpisu řádky lze modifikovat pomocí příkazu POKE

ve třech adresách programu GENS3. Podrobnosti jsou uvedeny dále. Rozlišujme mezi řádkou assembleru a řádkou obrazovky. Řádka assembleru se nemusí nutně objevit celá na řádce obrazovky.

1. "Start GENS3 + 139 (#8B)"
   určuje, na které pozici (sloupec-5) skončí první řádka výstupu assembleru. Změnou tohoto bajtu na 0 se vytvoří nepřerušená řádka (užitečné u tiskáren s dlouhou řádkou). Hodnota menší než 256 ukončí řádku na dané pozici sloupce.
2. "Start GENS3 + 140 (#8C)"
   udává sloupec (počítáno od 1), na němž má začít následující řádka na obrazovce.
3. "Start GENS3 + 141 (#8D)"
   udává, kolik znaků (zbylých z řádky) má být zobrazeno na obrazovce po první obrazovkové řádce.

Pozn.: u neupraveného programu GENS3 místo 139 bude 51 atd.

Pro příklad předpokládejme, že požadujete, aby první obrazovková řádka měla 20 znaků (t.j. bez pole návěští). Každá následující obrazovková řádka má začínat na pozici 1 a vyplnit celou řádku. GENS3 jste uložili od adresy #C350 (50000 dekadicky). K vyvolání žádaných změn proveďte následující:

```
POKE 50139,20
POKE 50140,1     na začátku následující obrazovkové
POKE 50141,31    řádky musí být alespoň jedna mezera
```

Výše uvedené změny lze provést, jen když nebyl použit příkaz ≡C! Tento příkaz způsobí (v případě potřeby) "scrollování" textu.

Výpis assembleru lze přerušit na konci každé řádky stiskem CAPS SHIFT a SPACE - následujícím stiskem "E" se vrátíme do editoru; stiskem jiného libovolného tlačítka pokračuje výpis.

Jediné chyby, ke kterým může dojít při druhém průběhu, jsou: "chyba 10" - do bajtu ukládáme číslo větší než 255 a "chyba ORG" - hrozí přepsání GENS3 (ev. i MONS3 a modulu DOKTOR) naším programem. Vyhodnocení "chyba ORG" může být vypnuto činností 32. "Chyba 10" není podstatná, v kompilaci můžete pokračovat. Zatímco chyby jako "chyba ORG" podstatné jsou a ihned vracejí řízení zpět do editoru (obdoba pudu sebezáchovy).

Informace na konci druhého průběhu:

chod 2 chyby: nn

se zobrazí nejdříve; po ní následuje případné upozornění na chybějící návěští (viz výše). Následující informace:

využita tab: xxxxx z yyyyy

říká, jaká část tabulky symbolů xxxxx byla využita ve srovnání s rezervovaným prostorem yyyyy (rezervace viz výše).

Na tomto místě, byl-li správně použit příkaz ENT, se objeví zpráva "start pgm (R):nnnnn". Udává počáteční adresu strojového kódu, který můžete spustit příkazem "R". Pozor na použití tohoto příkazu před úspěšným ukončením celé kompilace, která je zřejmá až ze zprávy "start pgm (R):nnnnn"!

Byla-li zvolena činnost 1, vytvoří se abecední seznam použitých návěští s příslušnými hodnotami. Počet vstupů zobrazených na jedné řádce může být změněn pomocí POKE "Start GENS3 + 138" s příslušnou hodnotou. Standardní hodnota je 2.

Nyní se řízení vrátí do editoru.

5.1 Formát assembleru

Každá generovaná řádka textu má následující formát, v němž jsou uvedena příslušná pole:

Návěští	Mnemonika	Operandy	Komentář
Start	LD	HL,návěští	;uložení "návěští" do HL

Mezery a znaky pro tabelaci (vložené editorem) jsou obecně ignorovány.

Řádka je zpracována následujícím způsobem:
je vyhodnocen první znak řádky, následující činnost je závislá na významu tohoto znaku:

";" celá řádka je považována za poznámku (komentář), t.j. je ignorována

"*" GENS3 očekává, že další znak (znaky) bude tvořit příkaz assembleru (viz kapitola 5.8); všechny znaky následující za příkazem považuje za komentář

" " (znak konce řádku) řádka je ignorována

" " (mezera nebo znak tabelace) je-li první znak mezera nebo znak tabelace, pak GENS3 předpokládá, že první znak, který není mezera nebo tab, bude začátek mnemoniky Z80.

Je-li první znak jiný než uvedeno, pak jej assembler po-

važuje za symbol - viz kap. 5.2.

Po zpracování platného návěští, nebo je-li prvním znakem mezera/tab, assembler hledá další znak, který není mezera/tab. Přitom očekává, že to bude buď konec řádky nebo začátek mnemoniky Z80 (viz příloha 2), která má až 4 znaky a je ukončena mezerou/tab nebo znakem konce řádky. Je-li mnemonika platná a vyžaduje jeden nebo více operandů, pak jsou mezery/tab vynechány a je přikročeno ke zpracování pole operandu.

Návěští mohou tvořit i samostatnou řádku, což je užitečné pro zvýšení přehlednosti.

Komentáře mohou být umístěny kdekoli za polem operandu. Nemá-li mnemonika argument, pak za polem mnemoniky.

5.2 Návěští

Návěští je symbol představující až 16tibitovou informaci. Návěští může být použito:

a) k označení adresy určité instrukce nebo datové oblasti

b) jako konstanta pomocí příkazu EQU (viz kap. 5.6).

Je-li návěští přiřazeno více než 8 bitů (hodnota přes 255), a pak je použito jako osmibitová konstanta, ohlásí assembler chybu. Tak např.:

```
naves EQU  #1234
      LD   A,naves
```

vyvolá hlášení "chyba 10" při druhém průběhu.

Návěští může obsahovat libovolný počet znaků (viz níže). Ale pouze prvních 6 znaků je platných (GENS3 jich víc "nevnímá). Těchto prvních 6 znaků musí být jednoznačně určeno, jinak návěští nemůže být rozpoznáno ("chyba 4"). Návěští nesmí být tvořeno rezervovaným slovem (viz příloha 2), které ale může být jeho součástí.

Znaky, které mohou tvořit návěští, jsou:

0-9, $, A-z

Kódy A-z zahrnují všechna velká i malá písmena spolu se znaky ^, `, _, [, \,].

Návěští musejí začínat písmenem!!!

Několik příkladů návěští:

```
LOOP
loop
```

```
dlouhénávěští
L[1]
259let      - nelze, začíná číslem!
a
dvě<5
LDIR        - LDIR nemůže být návěští!! (rezervované slovo)
```

5.3 Čítač adres

Assembler pracuje s čítačem adres tak, že symbolům v poli návěští přiřadí adresu a symbol vloží do tabulky symbolů. Čítač adres může být nastaven na libovolnou hodnotu příkazem ORG (viz kap. 5.6).

Symbol $ může být použit ve vztahu k běžné (momentální) hodnotě čítače adres: LD HL, $+5 nastaví registrový pár HL na hodnotu o 5 větší než je běžná hodnota čítače adres.

5.4 Tabulka symbolů

Při prvním použití je symbol vložen do tabulky se dvěma ukazateli, které později pomohou určit jeho abecední pořadí mezi ostatními v tabulce. Při prvním výskytu symbolu v poli návěští je hodnota (udaná čítačem adres nebo daná příkazem EQU) vložena do tabulky. Jinak je hodnota vložena ihned, jakmile je symbol v poli nalezen.

Tento typ tabulky symbolů se nazývá binárně větvený. Jeho struktura umožňuje symboly vkládat i vybírat velmi rychle, což je důležité u rozsáhlých programů. Délka položky v tabulce se pohybuje od 8 do 13 bajtů v závislosti na délce symbolů.

Je-li při prvním průběhu symbol definován více než jednou, objeví se chybové hlášení ("chyba 4"), protože assembler neví, kterou z hodnot symbolu přiřadit.

Není-li symbolu přiřazena hodnota, objeví se na konci překladu zpráva "POZOR! symbol neexistuje". Chybějící definice symbolu ale kompilaci nepřeruší.

Do tabulky symbolů se vkládá pouze prvních 6 znaků symbolu, aby tabulka nebyla příliš rozsáhlá.

Na konci kompilace se objeví zpráva oznamující, kolik paměti bylo tabulkou užito. Rozsah paměti tabulce vymezený můžete upravit na počátku odpovědi na dotaz "velkosť tab:" (viz kapitola 5.0).

5.5 Výrazy

Výrazem je pole operandu, které se skládá buď z jednoduché konstanty, nebo z několika konstant oddělených operátory.

Definice konstanty a operátoru je následující:

KONSTANTA	dekadická konstanta, např.	1029
	hexadecimální konstanta, např.	#405
	binární konstanta, např.	%10001000
	znaková konstanta, např.	"a"
	návěští, např.	L1029
	rovněž "$" může být použit jako konstanta k určení hodnoty čísla adres	
OPERÁTOR	"+" součet	
	"-" rozdíl	
	"&" logická funkce AND	
	"@" logická funkce OR	
	"!" logická funkce XOR	
	"*" součin celých čísel	
	"/" podíl celých čísel	
	"?" funkce MOD (a?b=a-(a/b)*b)	

Poznámka: "#" označuje začátek hexadecimálního čísla, '%' binárního čísla a '"' znakovou konstantu. Při čtení čísel (dekadických, hexadecimálních nebo binárních) GENS3 uvažuje 16 bitů od nejnižší hodnoty (tj. MOD 65536) např.: 70016 má hodnotu 4480 a #5A2C4 má hodnotu #A2C4.

Široký sortiment operátorů však neumožňuje závorkování za účelem volby priority výpočtů - výrazy jsou prováděny vždy ZLEVA DOPRAVA!. Operátory "*", "/" a "?" jsou pro zjednodušení prováděny samostatně, nikoli jako součást celého výrazu, což by zvětšilo rozsah GENS3.

Je-li výraz vložen v závorkách, pak představuje nepřímé adresování. Např. v instrukci LD HL, (loc+5) se naplní registrový pár HL 16tibitovou hodnotou zapsanou na adrese "loc+5".

Určité instrukce Z80 (JR a DJNZ) předpokládají osmibitovou hodnotu - jedná se o instrukce relativního adresování. Při jeho použití GENS3 automaticky odečte hodnotu čítače adres, nastaveného na následující instrukci, od hodnoty dané polem operandu uvažované instrukce. Tak získá relativní adresu pro danou instrukci. Rozsah přípustných hodnot pro relativní adresu

je -126 až +127.

Když budete chtít vložit přímou relativní vzdálenost od čítače adres, musíte použít znaku "$" následovaného požadovanou hodnotou. Nyní se relativní adresa vztahuje přímo k hodnotě čítače adres. Proto se hodnota relativní adresy pohybuje v rozmezí -126 až +129 včetně.

Příklady platných výrazů:

#5000-naves	
%1001101 ! %1011	výsledek %1000110
#3456 ? #1000	" #456
4+5*3-8	" 19
$-naves+8	
2345/7-1	" 334
"A"+128	
"y"-";"+ý	
(5*naves-#1000 & %1111)	
17@ %1000	" 25

Mezery mohou být vloženy mezi konstanty a operátory, nikoli však uvnitř konstanty.

Je-li výsledkem násobení absolutní hodnota větší než 32767, objeví se hlášení "chyba 15", zatímco při dělení nulou získáme hlášení "chyba 14"; přeplnění je ignorováno. Veškerá aritmetika používá dvojkový doplněk (two's complement), kde všechna čísla větší než 32767 jsou považována za záporná,

např.: 60000= -5536 (60000-65536).

5.6 Řídící povely assembleru

GENS3 rozeznává určité pseudoinstrukce. Tyto řídící povely assembleru, už podle jejich názvu, nemají vliv na mikroprocesor Z80, tj. nejsou překládány do strojového kódu. Avšak výrazně přispívají k řízení assembleru při kompilaci a značně usnadňují práci programátora.

Pseudoinstrukce jsou sestavovány přesně stejně jako výkonné instrukce. Může jim předcházet návěští (nutné pro EQU) a mohou být následovány komentářem.

PSEUDOINSTRUKCE

ORG výraz

nastaví čítač adres na hodnotu "výraz". Není-li zvolena činnost 2 nebo 16 a ORG by způsobil přepsání GENS3, textu nebo tabulky symbolů, objeví se zpráva "Bad ORG", kompilace se přeruší. Jak činnosti 2 a 16 ovlivňují použití ORG, je uvedeno v kapitole 5.0.

EQU výraz

musí mu předcházet návěští. Přiřadí návěští hodnotu "výraz". Výraz nemůže obsahovat symbol, kterému dosud nebyla přiřazena hodnota ("chyba 13").

DEFB výraz, výraz,... Definovaný bajt (Byte)

každý výraz je vyhodnocen do 8 bitů. Bajt na adrese, kterí je v čítači adres, je nastaven na hodnotu "výraz" a čítač se posune o 1. Opakuje se pro každý výraz. Např.:

```
ORG 60000
DEFB 1,2,#A,#EF
```

uloží na adresy 60000-60003 postupně čísla 1,2,#A,#EF.

DEFW výraz, výraz,... Definované slovo (Word)

nastaví slovo (2 bajty) na adrese, která je v čítači adres, na hodnotu "výraz" a posune čítač o 2. Jako první se umístí nižší bajt následovaný vyšším. Opakuje se pro každý výraz. Např.:

```
ORG 60000
DEFW 9,255,#FFFF
```

uloží na adresy: 60000...9, 60001...0
60002...255, 60003...0
60004...255, 60005...255

DEFS výraz Definovaný prostor (Space)

zvýší hodnotu čítače adres o hodnotu "výraz", čili rezervuje blok paměti o velikosti dané hodnotou "výraz". Původní obsah rezervované paměti se tímto příkazem nezmění.

DEFM "g" Definovaná zpráva (Message)

definuje obsah n bajtů v paměti - ty pak obsahují ASCII kód řetězce "g", kde n je délka řetězce a může být teoreticky dlouhá 1 - 255. Prakticky je však délka řetězce omezena délkou řádku editoru. První znak pole operandu - uvozovky - je vyhodnocen jako začátek řetězce. Řetězec tvoří znaky mezi oběma uvozovkami. Znak konce řádku má rovněž význam konce řetězce. Např.:

```
ORG 60000
DEFM "TEXT"
```

uloží postupně na adresy 60000-60003 ASCII kódy slova TEXT.

ENT výraz

nastaví startovací adresu strojového kódu na hodnotu "výraz" - používá se ve spojení s příkazem editoru "R" (viz kapitola 6). Pro startovací adresu neexistuje standardní (předdefinovaná) hodnota (default). Když kompilace proběhne bezchybně, zobrazí se informace "start pgm (R):xxxxx", kde xxxxx je dekadická startovací adresa vašeho programu, od níž jej můžete spustit příkazem "R".

5.7 Podmíněné pseudoinstrukce

Podmíněné pseudoinstrukce IF, ELSE, END umožňují programátorovi vložit nebo vynechat určité části zdrojového textu v rámci kompilace.

IF výraz

vyhodnotí "výraz". Je-li výsledkem nula, pak se neprovede překlad následujících řádků, dokud assembler nenarazí na "ELSE" nebo "END". Je-li hodnota "výraz" nenulová, pak kompilace pokračuje normálně.

ELSE

Tato pseudoinstrukce spouští nebo ukončuje kompilaci. Byla-li kompilace spuštěna před "ELSE", ukončí ji, a naopak.

END

END jednoduše spouští kompilaci (blokovanou pseudoinstrukcí IF).

Poznámka: Podmíněné pseudoinstrukce se nesmějí používat ve vícenásobných cyklech; neprovádí se kontrola, proto takové použití vede k nejistým výsledkům.

5.8 Příkazy assembleru

Příkazy assembleru, stejně jako řídící povely, neovlivňují CPU Z80, protože se nepřekládají do strojového kódu. Příkazy assembleru upravují formát.

Příkaz assembleru je textový řádek, který začíná hvězdičkou "*". Písmeno následující za hvězdičkou určuje druh příkazu a musí být uvedeno velkým písmenem! Zbytek řádku může tvořit libovolný text s výjimkou příkazů "L" a "D", které očekávají "+" nebo "-" za příkazem.

PŘÍKAZY

*E (Eject) vytvoří tři prázdné řádky na obrazovce nebo na tiskárně - zvyšuje přehlednost a orientaci oddělením bloků textu.

*Hβ řetězec β se bere jako záhlaví a vytiskne se za každým příkazem *E. Slouží opět zvýšení orientace v textu. Samotné *H má stejnou funkci jako *E.

*S zastaví listování na běžícím řádku obrazovky. Listování pokračuje stisknutím libovolného tlačítka. Užitečné pro čtení střední části výpisu. Poznámka: *S se vyhodnocuje i po *L-. *S ale nezastaví výpis na tiskárně.

*L- zastaví listování i tisk, počínaje běžícím řádkem. Generování strojového kódu však pokračuje normálně.

*L+ obnoví listování i tisk, počínaje běžícím řádkem.

*D+ hexadecimální zobrazení adres se změní na dekadické (jen na začátku řádek, nikoli v instrukcích).

*D- vrací zpět k užití hexadecimálních čísel (opak *D+).

*C- zkrátí řádek assembleru počínaje následujícím řádkem. Ke zkrácení dojde vypuštěním výpisu strojového kódu, čímž se ušetří 9 znaků. Tak se zkrácený řádek assembleru vejde na jeden řádek obrazovky se 32 znaky a zvýší se přehlednost.

*C+ vrátí se k vypisování úplných řádků, jak je popsáno v kapitole 5.0 (opak *C-).

*F (název textu)

To je velice výkonný příkaz, který dovoluje načíst z pásky blok textu do vyrovnávací paměti (angl. buffer) a poté z ní provést kompilaci. To umožňuje vytvářet rozsáhlé programy ve strojovém kódu, protože překládaný text nezabírá mnoho paměti (ve zdrojovém textu se neobjeví).

Jméno textu (až 10 znaků), který chcete vložit do bufferu pro jeho kompilaci, může být zapsáno po "*F" a musí být odděleno mezerou. Není-li jméno uvedeno, vloží se první text, který bude na pásce nalezen. Příkaz "*F" vložíme na řádku, od níž se má včlenit výsledek překladu načítaného bloku.

Jakýkoli text, který chcete vkládat tímto způsobem, musí být předem uložen na pásku příkazem editoru "T" - NIKOLI "P"! To je nezbytné, neboť text pro vkládání musí být uložen v blocích s dostatečnými mezerami mezi bloky, které umožní překlad jednoho bloku před vložením dalšího z pásku. Velikost bloků užitých tímto příkazem (a příkazem editoru T) určujete na začátku při vstupu do GENS3, odpovědí na otázku "dlžka bufferu?" (viz kapitola 4). Vložené číslo (0-9) určuje velikost vyrovnávací paměti v jednotkách po 256 bajtech se standardní hodnotou 4*256 bajtů (po odpovědi přímým stiskem ENTERu). Možnost volit velikost paměti optimalizuje poměr velikost/rychlost při vkládání textu z pásky; např. nepředpokládáte-li použití příkazu "F", je vhodné zvolit velikost paměti 0 (jen 64 bajtů), a tak minimalizovat prostor, který GENS3 celkově zabere.

Uvědomte si, že velikost bufferu specifikovaná v době, kdy tvoříme soubor pro vložení, se musí rovnat velikosti bufferu v době, kdy chcete text vkládat.

Kdykoli assembler při kompilaci vyhodnotí příkaz "*F", objeví se "start MGF". Stane se tak při 1. i 2. průběhu, neboť vkládaný text musí být čten při obou průbězích. Na pásce je hledán buď soubor požadovaný, nebo jakýkoli první. Je-li nalezen soubor pro vložení, jehož jméno nesouhlasí s požadovaným, zobrazí se zpráva "????:název" a hledání pokračuje. V případě úspěšného nálezu se zobrazí "prog: název", soubor se blok po bloku nahrává do bufferu a odtud rozmístí v paměti. Viz kapitola 6.3, kde je uveden příklad použití tohoto příkazu. Jiné příkazy assembleru než "*F"

jsou rozpoznány pouze při druhém průběhu. Protože pro načtení tímto příkazem jsou potřeba 2 průběhy, je vhodné si na pásek nahrát dotyčný blok dvakrát za sebou, abyste nemuseli převíjet magnetofon.

Je-li kompilace vypnuta jednou z podmíněných pseudoinstrukcí, je vypnut i vliv všech příkazů assembleru.

KAPITOLA 6 EDITOR

6.1 Úvod do editoru

Editor GENS3 je jednoduchý řádkový editor pracující se všemi systémy osazenými mikroprocesorem Z80. Má celkem jednoduchou obsluhu a schopnost rychlého, efektivního vkládání programu z pásku i klávesnice.

Za účelem zmenšení velikosti textového souboru provádí editor potlačení určitého počtu mezer. Dosahuje se toho následujícím způsobem: po napsání řádky z klávesnice je tato, znak po znaku, vložena do vnitřní vyrovnávací paměti assembleru. Po ukončení řádky stiskem ENTERu se řádka z této paměti přesune do textového souboru. Během tohoto přesunu jsou určité mezery potlačeny. Řádka je vyhodnocována od prvního znaku. Jde-li o mezeru, nahradí se znakem pro tabelaci a všechny následující mezery se vynechají. Není-li prvním znakem mezera, pak se znaky přesunují do textového souboru, dokud není mezera objevena. Pak probíhá stejný postup, jako by mezera byla na prvním místě. To se dále opakuje, takže znaky tabelace jsou umístěny na začátek řádky, mezi návěští a mnemoniku, mezi mnemoniku a operandy, případně mezi operandy a komentář. Je-li vyhodnocen příkaz pro návrat (ENTER), překlad se ihned ukončí a program se vrátí do editoru.

Uživatel může jednoduše použít znaky pro tabelaci (CI - viz níže) pro vytvoření tabelovaného textového souboru, který se současně úsporně ukládá do paměti.

Povšimněte si, že mezery se v komentáři nepotlačují, ale uvnitř návěští, pole mnemoniky a operandu být nesmějí.

V průběhu této kapitoly se používají určité zkratky k označení příslušných řídících kódů:

ENTER	ENTER
CC	EDIT (pro přerušení vstupu)
CH	DELETE (CAPS SHIFT a 0) (mazání kurzorem zpět)
CI	Kurzor vpravo (posun na další tabelační pozici)
CX	Kurzor vlevo (vymazání vkládané řádky)

Spuštěním GENS3 automaticky vstoupíme do editoru. Na obrazovce se objeví copyrightový nápis a kurzor editoru '>'.

Za kurzor můžete vložit příkazový řádek následujícího formátu:

C N1,N2,S1,S2 ENTER

C je příkaz k provedení (viz kapitola 6.2 níže)
N1 je číslo 1-32767 včetně
N2 je číslo 1-32767 včetně
S1 je řetězec znaků délky max. 20
S2 je řetězec znaků délky max. 20

Čárka je použita k oddělení jednotlivých argumentů (může se také změnit - viz příkaz "S"). Mezery se ignorují s výjimkou řetězců. Žádný z příkazů není povinný, ale některé příkazy (např. "D") se neprovedou bez určení N1 a N2. Editor si pamatuje vložená čísla a řetězce - v případě, že je některá hodnota vynechaná, pak N1 a N2 nabydou hodnoty 10, řetězce zůstávají prázdné. V případě vložení neplatného příkazu jako F-1,100,HELLO je takový příkaz ignorován a zobrazí se otázka "Pardon?". Řádku je třeba přepsat správně, tj. F1,100,HELLO. Tato chybová zpráva se objeví rovněž v případě, že řetězec S2 překročí 20 znaků; při překročení délky u S1 se další znaky ignorují.

Příkazy můžete vkládat velkými nebo malými písmeny.

Při vkládání příkazové řádky mohou být použity řídící funkce uvedené výše, jako např. CX k vymazání začátku řádky, CI k posunutí kurzoru na následující tabelační pozici a pod.

Následující kapitola podrobně popisuje příkazy editoru.

V případě, že argument příkazu je vložen do symbolu '<>', pak je nutné argument použít, aby příkaz byl proveden.

6.2 Příkazy editoru

6.2.1 Vkládání textu

Text může být vložen do textového souboru:
a) buď napsáním čísla řádky, mezery a textu
b) nebo užitím příkazu "I".

Vepíšete-li číslo řádky bez textu, pak, pokud předtím existovala, je tato řádka vymazána (podobně jako v Basicu). Při vkládání textu lze použít řídící funkce CX, CI a CC. DELETE (CH) způsobí posun kurzoru zpět (nikoli však na začátku řádky). Text se vkládá do vnitřní paměti GENS3. Je-li tato paměť plná, je vkládání dalších znaků blokováno - musíte použít CH nebo CX pro její uvolnění.

Když editor zjistí, že se během vkládání konec textu blíží k RAMTOP, pak zobrazí zprávu "prepl.paměti". Znamená to, že není možné vkládat další text a vložený textový soubor nebo alespoň jeho část se musí nahrát na pásku pro pozdější použití.

Příkaz: I n,m

Tento příkaz umožňuje automatické řádkování (číslování řádek):

n - je číslo první vkládané řádky

m - je odstup mezi čísly řádky

Text se vkládá po zobrazení čísla řádky. Podle potřeby lze použít různé řídící kódy. Text se ukončí stiskem ENTERu. K ukončení činnosti užijte řídící funkci CC.

Vložíte-li řádku s číslem, které již v textu existuje, pak je stará řádka vymazána a nahražena novou po stisku ENTERu. Jestliže při automatickém číslování řádky přesáhne její hodnota 32767, vkládání se ukončí.

Když při psaní textu dosáhnete konce řádky na obrazovce, aniž jste napsali 64 znaků (velikost řádkové paměti), pak se obraz posune o řádku a můžete pokračovat na další - v textu se automaticky označí, že toto číslo řádky je oddělené od textu.

Zrušit režim automatického řádkování můžete stiskem CC.

6.2.2 Listování textem

Text můžeme prohlížet pomocí příkazu "L". Počet současně zobrazených řádek při tomto příkazu je předdefinován, ale může být změněn příkazem "K".

Příkaz: L n,m (List)

Zobrazí text na obrazovce od řádky n do řádky m. Standardní hodnota pro n je vždy 1 a pro m je 32767, tj. standardní hodnota je nezávislá na dříve užitých argumentech. K listování celým textovým souborem jednoduše použijte "L" bez argumentů. Levý okraj obrazovky je uspořádán tak, že číslování řádek je zřejmé. Tabelace řádky se provádí automaticky, jednotlivá pole jsou oddělena. Počet zobrazených řádek může být regulován příkazem "K" - po zobrazení příslušného počtu řádek se listování zastaví (pokud nebylo dosaženo řádky m). Stiskem CC lze provést návrat do editoru, stiskem jiného tlačítka pokračujete v listování.

Souhrnem:

L	prolistuje	celý zdrojový text
L n,m	"	zdrojový text od řádky n do m
L n	"	" " od řádky n do konce
L m	"	" " od začátku do řádky m

Příkaz: K n

"K" nastaví počet na obrazovce se zobrazujících řádek při listování textem, jak bylo popsáno výše u "L". Je vypočtena hodnota (n MOD 256) a uložena. Užitím K5 při následném "L" se zobrazí 5 řádek současně. Standardní hodnota n=15.

6.2.3 Úprava textu

Po vytvoření určitého textu může nastat potřeba vložení dalších řádků. Dále uvedené příkazy umožňují změnu, vypuštění, přesun a přečíslování řádek:

Příkaz: D n,m (DELETE)

Všechny řádky od n do m budou vypuštěny ze souboru. Je-li n větší než m, nebo je použito méně než dvou argumentů, příkaz se neprovede. To pomáhá zabránit chybám z nepozornosti. Jednu řádku lze vypustit při m=n; téhož lze dosáhnout napsáním čísla řádky s následným stiskem ENTERu.

Příkaz: M n,m (MOVE)

Přemístí text z řádky n na řádku m, kde nahradí starý text (pokud tam je). Řádka n se zruší. Tento příkaz umožní přesunout řádku textu na jinou pozici v rámci textového souboru. Pokud řádka s číslem n neexistuje, příkaz se neprovede.

Příkaz: N n,m (NUMBERING)

Přečísluje textový soubor s první řádkou n. Číslování dalších řádek má krok m. Musí být uvedeno n i m. V případě, že by číslování překročilo hodnotu 32767, zůstane původní číslování nezměněno.

Příkaz: F n,m,f,s (FIND)

f je hledaný, předem definovatelný řetězec. Rovněž definovatelný řetězec s nahrazuje nalezený řetězec f (substituce).

Příkaz hledá v rozmezí řádek n a m výskyt řetězce f. Je-li nalezen, pak se příslušná řádka zobrazí a vstoupí se do režimu EDIT - viz níže. Nyní lze použít příkazu editoru k vyhledání dalšího výskytu hledaného řetězce f v udaném rozmezí, nebo jej můžeme nahradit řetězcem s. Podrobnosti uvedeny níže. Rozmezí řádek i oba řetězce mohou být definovány již dříve jiným příkazem - pak stačí stisknout "F" pro vyhledání. Viz příklad v kapitole 6.4.

Příkaz: E n (EDIT)

Umožní úpravu řádky s číslem n. Pokud n neexistuje, příkaz se neprovede; jinak se řádka přepíše do vyrovnávací paměti a zobrazí i s číslem řádky. Její číslo se zobrazí ještě jednou na následující řádce obrazovky. Tak lze provádět úpravy nebo změny. Změny se provádějí ve vyrovnávací paměti a nikoli v samotném textu; lze tedy kdykoli obnovit původní obsah řádky.

Při této činnosti se zobrazí kurzor, který se pohybuje podél řádky (od prvního znaku). Přitom je možné použít různých pomocných příkazů pro úpravy nebo změny v řádce.

Pomocné příkazy jsou:

' ' (mezera) - posune kurzor o jedno místo vpřed. Nelze krokovat přes konec řádky.

CH (DELETE) - posune kurzor o jedno místo zpět. Nelze krokovat přes začátek řádky.

CI (řídící funkce) - posune kurzor na následující tabelační pozici.

ENTER - konec úpravy (všechny změny provedeny)

Q (QUIT) - přeruší úpravu a ignoruje provedené změny v řádce.

R (RELOAD) - obnoví obsah vyrovnávací paměti podle původního obsahu textového souboru (vrácení do původního tvaru).

L (LIST) - vypíše zbytek řádky, která je upravována, tj. zbytek řádky za pozicí kurzoru. Druh činnosti zůstává zachován s kurzorem na začátku řádky.

K (KILL) - vypustí znak na pozici kurzoru.

Z - vypustí všechny znaky od začátku řádky až ke kurzoru včetně.

F (FIND) - hledá další výskyt řetězce dříve definovaného

příkazovou řádkou (viz příkaz "F" výše). Tento pomocný příkaz automaticky ukončí úpravu dané řádky (zachová změny), nenalezne-li další výskyt požadovaného řetězce v této řádce. Vyskytne-li se řetězec v řádkách, které následují (v předem stanoveném rozmezí), přejde se k úpravě té řádky, kde byl řetězec znovu nalezen. Všimněte si, že kurzor se nastaví vždy na počátek nalezeného řetězce.

S (SUBSTITUTE) - nahradí předem definovaným řetězcem "s" nalezený řetězec "f" a provede příkaz "F". Tj. vyhledá následující výskyt řetězce "f". Ten se, spolu s příkazem "F" uvedeným výše, používá ke krokování textovým souborem, přičemž se řetězce "f" nahrazují řetězci "s" (viz příklad v kapitole 6.4).

I (INSERT) - vloží znaky na pozici kurzoru. V této činnosti setrvá, dokud nestisknete ENTER - ten vrátí činnost do hlavní smyčky s ukazatelem nastaveným za poslední vložený znak. Při této činnosti příkaz CH (DELETE) vymaže znaky vlevo od kurzoru, zatímco CI (řídící funkce) posune kurzor na další tabelační pozici a vloží mezery.

X - posune kurzor na konec řádky a automaticky přejde do činnosti vkládání (INSERT) popsané výše.

C (CHANGE) - změní znak. Tento příkaz umožňuje přepsat znak na pozici kurzoru a posunout kurzor o jeden krok dopředu. CHANGE působí, dokud nestisknete ENTER, kterým se vrátíte do hlavní smyčky programu; kurzor bude za posledním změněným znakem. CH (DELETE) v tomto případě pouze posune kurzor o jeden krok zpět, zatímco CI nemá žádný účinek.

6.2.4 Příkazy pro práci s magnetofonem

Text může být nahrán na pásek nebo načten z pásku použitím příkazu "P", "G" a "T".

Příkaz: P n,m,s

Oblast řádek od n do m uloží na pásek se jménem určeným

řetězcem "s". Pamatujte, že tyto argumenty mohou být definovány dříve vloženým příkazem. Před vložením tohoto příkazu se přesvědčte, že magnetofon je připraven k nahrávání. Příkaz se aktivuje ihned po stisku ENTERu. Nedefinujeme-li řetězec, nahraje se textový soubor bez názvu. Nepoužívejte tento příkaz, budete-li si později přát tuto oblast vložit "INCLUDE" do jiného programu - textu. V takovém případě použijte příkaz "T".

Příkaz: G,,s (Get)

Hledá na pásku soubor s názvem "s". Je-li nalezen, uloží se za konec stávajícího textu. Není-li vloženo jméno, z pásku se nahraje nejbližší první textový soubor.

Po vložení příkazu se objeví nápis "start MGF" - spusťte přehrávání. Nyní se hledá soubor s uvedeným jménem, nebo v případě, že jméno uvedeno nebylo, je sejmut první textový soubor. Při shodnosti názvů se zobrazí zpráva "prog:název", jinak "????:název", a hledání na pásce pokračuje.

Pamatujte, že je-li jakýkoli textový soubor již umístěn v paměti, pak nově nahraný soubor z pásky bude připojen ke stávajícímu souboru a celý soubor bude přečíslován počínaje řádkem 1 s krokem 1!!

Příkaz: T n,m,s

Nahraje na pásek blok textu definovaný intervalem řádek od n do m včetně ve tvaru vhodném k pozdějšímu vložení příkazem assembleru "*F" - viz kapitola 6.8. Soubor je nahrán s názvem s. Příkaz se aktivuje okamžitě po stisku ENTERu, proto se přesvědčte, zda je magnetofon připraven a spuštěn ještě před vložením této příkazové řádky. Protože příkaz assembleru *F vyžaduje dvě načtení (pracuje se dvěma průběhy), z důvodu jednodušší manipulace si nahrajte soubor dvakrát za sebou.

6.2.5 Kompilace a spuštění

Příkaz: A

Provede kompilaci od první řádky textového souboru, tedy generuje strojový kód (s výjimkou činnosti 2). Podrobněji viz kapitola 5.

Příkaz: R

Byla-li kompilace provedena bez chyby a příkazem ENT určena adresa spuštění, lze použít příkazu "R" ke spuštění programu ve strojovém kódu. V něm můžeme použít instrukci RET (#C9) k návratu do editoru, pokud ukazatel zásobníku (stack pointer) bude po provedení tohoto příkazu ve stejné pozici jako byl na začátku. Uvědomte si, že ENT se neprovede v případě, kdy byla pro kompilaci zvolena činnost 16.

6.2.6 Ostatní příkazy

Příkaz: B

Předává řízení modulu DOKTOR.

Příkaz: C

Tento příkaz uvádíme jen pro informaci o jeho existenci. Je určen majitelům starší verze GENS1. Umožňuje převést textové soubory vytvořené pomocí GENS1 na formát GENS3. Nahraje se textový soubor GENS1 příkazem "G" a potom se příkazem "C" provede konverze souboru. Dále se používá již jako soubor GENS3 - příkazem "P" se nahraje na pásek. "C" nemá argument a může chvíli trvat, než se konverze provede.

Příkaz: S,,d

Tento příkaz umožňuje změnit separátor, který odděluje argumenty v příkazové řádce. Při vstupu do editoru je čárka "," brána jako separátor. To můžeme změnit příkazem "S" na první znak specifikovaného řetězce d. Pamatujte, že nově definovaný separátor musí být používán, dokud tímto příkazem není specifikován jiný.

Jako separátor nesmí být použita mezera.

Příkaz: V

Zobrazí okamžitou hodnotu N1, N2, S1 a S2, tedy standardní hodnoty čísel, řádek a řetězců. Používá se před vkládáním příkazů používajících standardní hodnoty ke kontrole, zda jsou tyto hodnoty správné. Jinými slovy - zobrazí aktuální hodnoty n, m, f, s, jak jsou uvedeny u příkazu F n,m,f,s.

Příkaz: W n,m

Oblast s čísly řádek n až m včetně se vytiskne na tiskárně. Pokud n a m mají standardní hodnotu, pak se vytiskne celý textový soubor. Tisk se přeruší po počtu řádek, určeném příkazem "K", a pokračuje po stisku libovolného tlačítka.

Příkaz: X

Zobrazí počáteční a koncovou adresu uložení textového souboru v dekadickém tvaru. Užitečné pro uložení textu z Basicu (nebo modulu DOKTOR) nebo chcete-li zjistit, kolik paměti vám zbývá za textovým souborem. GENS3 očekává, že text začíná na první adrese dané příkazem "X". Koncovou adresu textu má uloženu na adrese TEXTEND, která je rovna součtu "Start of GENS3 + + 142" (u neupraveného GENS3 se přičte 54). Takže chcete-li (při práci bez modulu DOKTOR) vypsat textový soubor (např. vytvořený pomocí MONS3), musíte jej převést na adresu určenou první adresou zobrazenou příkazem "X", změnit TEXTEND tak, aby obsahoval koncovou adresu souboru, a nakonec vstoupit do GENS3 (je-li neupraven, pak horkým startem).

Příklad toho, co byste museli vykonat, nemít modul DOKTOR:

Vytvořili jste textový soubor, který je správně umístěný, a končí na adrese #9A02 (adresa za posledním znakem konce řádky). Dále předpokládejme, že jste načetli GENS3 od adresy 24064 - v Basicu proveďte příkazy POKE 24064+142,2 (#02) a POKE 24064+143,154 (#9A) a spusťte GENS3 pomocí RANDOMIZE USR 24125. Nyní můžete pracovat s textovým souborem přímo z editoru.

6.3 Příklad použití editoru

Předpokládejme, že jste psali do následujícího programu (pomocí příkazu I10,10). V programu jsou záměrné chyby, které budeme postupně odstraňovat:

```
10 #h    16 BIT RANDOM NUMBERS  (16tibitová náhodná čísla)
20
30 ;INPUT:  HL obsahuje predesle nahodne cislo nebo seed
40 ;OUTPUT: HL obsahuje nove nahodne cisro
50
60 Random PUSH AF      ;ulozeni obsahu registru
```

```
 70 PUSH BC
 80         PUSH HL
 90         ADD  HL,HL  ;=2
100         ADD  HL,HL  ;=4
110         ADD  HL,HL  ;=8
120         ADD  HL,HL  ;=16
130         ADD  HL,HL  ;=32
140         ADD  HL,HL  ;=64
150         PIP  BC     ;stare nahodne cislo
160         ADD  HL,DE
170         LD   DE,41
180         ADD  HL,DE
190         POP  BC     ;zaplneni registru ze zasobniku
200         POP  AF
210         REY
```

V programu jsou následující chyby:

řádek 10: v příkazu bylo použito h místo H
řádek 40: cisro místo cislo
řádek 70: PUSH BC začíná v poli návěští
řádek 150: PIP místo POP
řádek 160: chybí komentář (stylizační chyba)
řádek 210: REY místo RET

Rovněž dvě další řádky ADD HL,HL by měly být vloženy mezi řádky 140 a 150 a všechny odkazy na registrový pár DE na řádkách 160 až 180 by měly být na registrový pár BC.

Opravy provedeme takto:

```
E10 ENTER                pak-(mezera) C následuje H ENTER ENTER
F40,40,cisro,cislo ENTER  pak pomocný příkaz "S"
E70 ENTER                pak I--------(7 mezer) ENTER ENTER
I142,2 ENTER             142--------ADD HL,HL     ;=128
                         144--------ADD HL,HL     ;=256 a 'CC'
F150,150,PIP,POP ENTER   pak pomocný příkaz "S"
E160 ENTER               pak X---;=257 + 41 ENTER ENTER
F160,180,DE,BC ENTER     pak opakovaně použít "S"
E210 ENTER               pak CI CI ----C T ENTER ENTER
N10,10 ENTER             přečísluje text
```

Důrazně doporučujeme projít celý výše uvedený příklad přímo na počítači.

6.4 Funkční příklad

Následuje příklad typického použití GENS3 - jste-li začátečník programování v assembleru nebo jste-li trochu nejistý v používání editoru a assembleru, pak si tento příklad pozorně projděte. Slovo ENTER znamená, že máte stisknout tlačítko ENTER.

Cíl cvičení:

Napsat a vyzkoušet rychlou rutinu pro násobení celých čísel a nahrát její text na pásek pomocí příkazu editoru "T" tak, aby mohla být lehce použitelná v jiných programech.

Postup cvičení:

1. Napsat rutinu násobení jako podprogram a nahrát ji na pásek pomocí příkazu editoru "P", aby mohla být lehce znovu načtena do počítače pro odstranění možných chyb.
2. Odstranit chyby v podprogramu, upravit jej, bude-li třeba.
3. Bezchybný podprogram nahrát na pásku použitím příkazu editoru "T" tak, aby mohl být "vložen" do jiných programů.

Vstupte do GENS3, v odpověď na dotaz "dlžka bufferu:" vložte číslo 1 a stiskněte ENTER. Tak se vytvoří "vkládací" paměť velikosti 1*256 bajtů. Na obrazovce se objeví kurzor; editor očekává vaše příkazy pro tvorbu programu.

Část 1 - napsání rutiny pro násobení.

Použijeme příkaz editoru "I" k vložení textu a pomocí CI (znak tabelace) získáme tabelované zobrazení. I když CI nepoužijeme, vždy získáme výpis tabelovaný. Není označeno, kde je použit CI, ale předpokládáme, že před mnemonikou a mezi mnemonikou a operandem. Uvědomte si, že adresy uvedené ve výpisu assembleru, který následuje, nemusejí odpovídat adresám na vašem počítači; jsou uvedeny pouze pro ilustraci.

```
110,10 ENTER
  10 ;Rychla rutina nasobeni ENTER
  20 ;celych cisel. Nasobi HL ENTER
  30 ;s DE. Vysledek je pak ENTER
  40 ;v HL. C flag nastaven ENTER
  50 ;na overflow. ENTER
  60 ENTER
  70        ORG   #7F00 ENTER
  80 ENTER
```

```
 90 Mult   OR   A ENTER
100        SBC  HL,DE ;HL vetsi nez DE ENTER
110        ADD  HL,DE ENTER
120        JR   NC,Mu1 ENTER
130        EX   DE,HL ENTER
140 Mu1    OR   D ENTER
150        SCF ;overflow kdyz ENTER
160        RET  NZ ;DE vetsi nez 255 ENTER
170        OR   E ;krát 0? ENTER
180        LD   E,D ENTER
190        JR   NZ,MU4 ;ne ENTER
200        EX   DE,HL  ;0 ENTER
210        RET ENTER
220 ENTER
230 ;Hlavni rutina. ENTER
240 ENTER
250 Mu2    EX   DE,HL ENTER
260        ADD  HL,DE ENTER
270        EX   DE,HL ENTER
280 Mu3    ADD  HL,DE ENTER
290        RET  C ;overflow ENTER
300 Mu4    RRA ENTER
310        JR   NC,Mu3 ENTER
320        OR   A ENTER
330        JR   NZ,Mu2 ENTER
340        ADD  HL,DE ENTER
350        RET ENTER
360 CC
>P10,350,Mult ENTER
```

Vytvořený text se nahraje na pásek. Pamatujte, že magnetofon musí být spuštěn před vložením příkazu "P".

Část 2 - odstranění chyb v programu

Nejprve se přesvědčíme, zda kompilace proběhla správně. Použijeme činnost 6, takže se neprovede výpis a nevytvoří se strojový kód.

```
>A ENTER
velkosť tab: ENTER (standardní hodnota velikosti tabulky symb.)
typ činnosti: 6 ENTER
```

```
chod 1 chyby: 00     (to je zobrazení zpráv z obou průběhů
                      kompilace)
chod 2 chyby: 00
POZOR! MU4 neexistuje
využitá tab: 74  z  161
>
```

Z této kompilace vidíme, že jsme udělali chybu v řádce 190 a vložili MU4 namísto Mu4, které je použitým návěštím. Upravíme tedy řádek 190:

```
>F190,190,MU4,Mu4 ENTER
> 190        JR   NZ    (nyní užijte povel "S")
```

Teď provedeme kompilaci ještě jednou. Ta by už měla být bez chyby. Musíme vložit kód pro odzkoušení rutiny:

```
>N300,10 ENTER       (přečíslování pro vložení dalšího textu)
>110,10 ENTER
  10 ;kod pro zkouseni ENTER
  20 ;rutiny Mult. ENTER
  30 ENTER
  40        LD    HL,50 ENTER
  50        LD    DE,20 ENTER
  60        CALL  MULT ;Nasobeni ENTER
  70        LD    A,H ;Vysledek ENTER
  80        CALL Aout ENTER
  90        LD    A,L ENTER
 100        CALL Aout ENTER
 110        RET ;Navrat do editoru ENTER
 120 ENTER
 130 ;Routina to o/p A in hex ENTER
 140 ENTER
 150 Aout   PUSH AF ENTER
 160        RRCA ENTER
 170        RRCA ENTER
 180        RRCA ENTER
 190        RECA ENTER
 200        CALL Nibble ENTER
 210        POP  AF ENTER
 220 Nibble AND  %1111 ENTER
 230        ADD  A, #90 ENTER
```

```
  240         DAA ENTER
  250         ADC  A, #40 ENTER
  260         DAA ENTER
  270         LD   IY, #5C3A ;pro ROM ENTER
  280         RST   #10 ;Volání ROMky ENTER
  290         RET ENTER
  300 CC
>
```

Nyní zkompilujte zkušební rutinu spolu s rutinou pro násobení.

```
>A ENTER
velkosť tab: ENTER
typ činnosti: 6 ENTER

7EAC  190         RECA
chyby 02          (stiskněte jakékoli tlačítko pro pokračování)

chod 1 chyby: 01
využitá tab:  88  z  210
```

Chyba - místo RECA má být RRCA (řádka 190). Tedy:

```
>E190
   190            RECA
   190            C R ENTER ENTER
```

Nyní proveďte kompilaci znovu pomocí činnosti 4 (bez výpisu) a text bude kompilován bezchybně. Už můžeme vyzkoušet činnost celého programu. Ještě je třeba editoru sdělit, odkud se má spustit program. To provedeme pseudoinstrukcí ENT:

```
>35          ENT   $ ENTER
```

Provedeme kompilaci ještě jednou. Měla by být ukončena zprávou:

```
využitá tab:  88  z  211
start pgm (R): 32416
>
```

nebo podobnou. Nyní můžeme spustit program příkazem editoru "R". Má proběhnout násobení 50 krát 20, což je 1000 nebo #3E8 hex.

```
>R ENTER
0032>
```

Nepracuje! Proč? Vypište řádky 380 až 500 (L380,500). Na řádce 430 vidíte instrukci OR D následovanou RET NZ. To je lo-

gická operace OR mezi registrem D a akumulátorem A. Návrat s chybou (stavový registr C nastaven) v případě nenulového výsledku. Podstatou věci je zjistit, jestli je DE menší než 256 a zda násobení nezpůsobí přeplnění - to se provede kontrolou, zda registr D je nulový... Ale OR pracuje správně jen tehdy, když akumulátor A je na začátku nulový. To zde není zajištěno. Musíme zařídit, aby registr A byl nulový před provedením instrukce OR D. Jinak získáme nepředpověditelné přeplnění s návratem většího čísla, než jaké by měl obsahovat výsledek. Při prohlídce kódu vidíme, že OR A na řádce 380 můžeme nahradit instrukcí XOR A, a tím nastavit stavové registry pro SBC HL,DE a současně i nulovat akumulátor A. Takže:

```
>F380 ENTER
  380 Mult      OR   A
  380           I X ENTER ENTER
>
```

Provedeme opět kompilaci (činnost 4) a spustíme program příkazem "R". Odpověď by teď měla být správná - #3E8. Můžeme provést další kontrolu změnou řádek 40 a 50 (vložit jiná čísla pro násobení), pak kompilovat a opět spustit - zjistíme, že rutina pracuje perfektně.

Po prověření můžeme rutinu nahrát na pásek ve "vkládacím" formátu:

```
>T300,999,Mult ENTER
```

Nezapomeňte spustit magnetofon před stiskem tlačítka ENTER. Nahraná rutina může být vložena do programu takto:

```
500        RET
510
520 ;Vlozit rutinu Mult zde.
530
540 =F Mult
550
560 ;Dalsi rutina
```

Při kompilaci tohoto textu se při obou průbězích objeví nápis "start MGF", když kompilace dojde na řádek 540. Musíte přehrát rutinu Mult z pásku v obou případech. To znamená převinout pásek po prvním průběhu. Abyste se převíjení vyhnuli, můžete nahrát Mult dvakrát za sebou a pro druhý průběh použít

druhou nahrávku.

Prostudujte tento příklad pečlivě a sami jej proveďte.

DODATEK 1 PŘÍKLAD KOMBINOVANÉ PRÁCE S MONS3 A GENS3

Vstupte do GENS3, stiskněte ENTER a vložte níže uvedený program. V programu jsou opět záměrné chyby, které budeme následně odstraňovat.

```
 10 ≡C-; zkraceceni radky
 20 ;=======================
 30 ≡D+; dekadicky vypis
 40 ;=======================
 50         ORG 50000
 60 ;=======================
 70         LD      A,2WER
 80         CALL    #1601
 90 AZAZAZAZAZAZAZAZAZAZAZA
100         LD      BC, #0E1B
110         CALL    #0DD9
120 ;=======================
130 LD      HL,ZNAK
140         LD      B,20
150 ZAC     PUH     BC
160         LD      A,(HL(
170         INC     HL
180         PUSH    BL
190         RST       10
200         PEP     HL
210         PEP     BC
220         RET
230         DJNZ
240 ;=======================
250 ≡S; STOP listingu
260 ≡E-; tri radky pauza
270 ;+++++++++++++++++++++++
280 ZNAK    DEFM "MIKROBAZE VAS
     ZDRAVI"
290 ;+++++++++++++++++++++++
300   L-
310 ;dalsi listing
320 ;se nezobrazi
330 ;+++++++++++++++++++++++
```

```
340         IF 0
350 ;++++++++++++++++++++++++
360 ;pipani
370         LD      A,2
380         CALL    #2D28
390         CALL    #1F3A
400         LD      DE,#0001
410         LD      HL,#0100
420         CALL    #03B5
430 ;++++++++++++++++++++++++
440         END
450 ;++++++++++++++++++++++++
```

Program má 2 části. První končí řádkem 290 a vypíše vložený text na zvolené pozici PRINT AT na obrazovce. Druhá část - po pozdějším vložení do 1. části - imituje zvuk psacího stroje. Protože ji budeme (po odstranění chyb) vsouvat do 1.části příkazem "T" a ≡F, není vybavena instrukcí RET. Na řádce 340 je pseudoinstrukce IF 0, která zajistí, že se při generování strojového kódu všechny instrukce stojící za ní nepřeloží. Aby tato pseudoinstrukce neblokovala výpis překladu na obrazovku, je na řádku 440 pseudoinstrukce END, která výpis uvolní.

Chyby v 1.části textu opravíme následovně (napřed je uveden chybný tvar, pak sled operací pro provedení oprav a nakonec řádka po provedení oprav):

```
    10 ;zkraceceni radky
E10 ENTER SPACE (10krát) K DELETE K DELETE ENTER
    10 ;zkraceni radky

    70          LD      A,2WER
E70 ENTER SPACE (15krát) Z ENTER
    70          LD      A,2

    90 ÁZAZAZAZAZAZAZAZAZAZAZAZA
90  ENTER (vymaže celý řádek)

   130 LD       HL,ZNAK
E130 ENTER I ENTER ENTER
   130          LD HL,ZNAK

   150          PUH BC
E150 ENTER SPACE SPACE I S ENTER ENTER
```

```
   150          PUSH BS
   160          LD      A,(HL(
E160 ENTER X DELETE ) ENTER ENTER
   160          LD      A,(HL)
   180          PUSH BL
E180 ENTER C H ENTER ENTER
   180          PUSH HL
   200          PEP     HL
F 10,300,PEP,POP ENTER S S ENTER
   200          POP     HL
   210          POP     BC
   220          RET
   230          DJNZ
M 220,230 ENTER 220 ENTER N 10,10 ENTER
   220          DJNZ
   230          RET
```

Náš program je nyní bez chyb, i funkčně by měl být bez vady. Při první kompilaci příkazem "A" použijte činnost 2, která zabrání vytvoření strojového kódu. Když oba průběhy kompilace budou bez chyby, zadejte znova příkaz "A" s činností 1, která kód vygeneruje a uloží do paměti od adresy 50000 (#C350).

Příkazem "B" přestupte z GENS3 do řídícího modulu Dr.MG, a program odstartujte pomocí RANDOMIZE USR 50000. Poté z modulu vstupte do MONS3 a vložte:

```
M:C350 ENTER
ss4
```

Zjistíte, že instrukce RET je uložena na adrese #C369 a od adresy #C36A začíná definovaný text. Použijte příkaz:

```
T
C350 ENTER
C37D ENTER
ENTER ENTER
C36A ENTER
C37D ENTER
ENTER ENTER
```

Na obrazovce se objeví dekompilovaný výpis vašeho programu. Pro lepší kontrolu použití příkazů assembleru "T" a ≠F vy-

mažte strojový kód z paměti tímto postupem:

P
C350 ENTER
C37D ENTER
O ENTER

a příkazem "L" překontrolujte účinek příkazu "P". Strojový kód programu mezi uvedenými adresami už nebude. Stiskem EDITu a poté tlačítka G vstoupíte opět do GENS3. Připravte si magnetofon na nahrávání. Vložte:

T 370,420,název souboru

Spusťte magnetofon a nahrajte soubor dvakrát za sebou.

Na řádku 195 (za instrukci RST #10) umístěte příkaz ≠F. Řádky 290 až 450 vymažte příkazem:

D 290,450 ENTER

Zadejte příkaz "A" a činnost 1. Přetočte magnetofon před začátek první nahrávky a zapněte přehrávání. Program se při obou přechodech zastaví a soubor se načte. Po úspěšném assemblování opět přejděte do MONS3. Zadejte příkaz "L"; uvidíte, že v dekompilovaném výpisu je zařazena "vsuvka", doplněná příkazem ≠F, od adresy #C365 do #C373. Novým vstupem do GENS3 se přesvědčíte, že i když je strojový kód v paměti kompletní, ve zdrojovém textu změny nezjistíte. Program spusťte příkazem RANDOMIZE USR 50000 a uvidíte výsledek. Avšak pozor - nestartujte program z GENS3 pomocí pseudoinstrukce ENT 50000 a příkazu "R"!! Pokud jste pozorně četli manuál, víte proč.

Ze cvičných důvodů pokračujme dál. Vstupte do GENS3 a příkazem

D 10,280 ENTER

vymažte zdrojový text. Pak přestupte přes modul DOKTOR do MONS3, zkontrolujte, zda střelka ukazuje na adresu #C350, a příkazem "L" si zjistěte adresu konce programu a adresu začátku definovaného textu (bude to #C38E a #C37B). Tyto adresy potřebujete k tomu, abyste z vašeho strojového kódu mohli pro GENS3 vygenerovat zdrojový text a potom jej nahrát na pásek. Použijte následovný postup:

```
                T
od:             C350 ENTER
do:             C38E ENTER
tlac Y/N:       N
```

```
zdroj.text Y/N:     Y
adr.prac.obl.:      ENTER
od:                 C37B
do:                 C38E
od:                 ENTER
do:                 ENTER
```

Na obrazovce se objeví kompletně dekompilovaný výpis. Na jeho konci je adresa prvního volného bajtu za koncem uložení zdrojového textu v GENS3.

Vystupte z MONS3 a přivolejte GENS3. Uvidíte, že zdrojový text je v pořádku. Návěští zdrojového textu jsou ve formátu Lxxxx. Nyní jen zbývá zdrojový text přečíslovat, např. pomocí:

```
N 10,10
```

a nastavit čítač adres pseudoinstrukcí "ORG", např.:

```
ORG 50000
```

Zdrojový text tak již můžete assemblovat, a poté jej příkazem P n,m uložit na pásek. Pokud jste nikde neudělali chybu, vše bude fungovat, jak náleží.

Důkladně si uvedený příklad prostudujte. Poskytne vám cenné informace o tom, jak při tvorbě programů využívat předností GENS3 i MONS3 současně. V žádném případě se ve studiu všech ovládacích funkcí programů nezastavte. I když se už i s jen částečnou znalostí ovládání dá leccos vytvořit či monitorovat, neměli byste se s tím spokojit. Zbytečně byste se tak připravili o mnohé výhodné služby Dr.MG a ve výsledku by se vaše práce při tvorbě vlastních programů vyznačovala nízkou efektivitou.

DODATEK 2 KOMUNIKACE S TISKÁRNAMI

Bez jakýchkoli úprav můžete programové výpisy z MONS3 i GENS3 pořizovat na ZX Printeru, Alphacomu a tiskárnách, které jsou s počítačem propojeny přes interfejs s implikovanými tiskovými příkazy ZX Spectra (např. ZX LPRINT III).

Tisk prostřednictvím paralelních interfejsů, které takovou implikaci ve svém operačním systému nemají, vyžaduje znalost jejich tiskové rutiny (označuje se také jako driver). Zkušenější programátoři ji snadno objeví v některých programech, které disponují volbou tisku při komunikaci s různými interfejsy. Ti

nejzkušenější ji najdou přímo v jeho systému.

Tiskovou rutinu interfejsu umístíme na adresu, kterou určíme následovně:

Zjistíme vektorovou adresu kanálových dat na adrese systémových proměnných CHANS:

PRINT PEEK 23631+256*PEEK 23632

K výsledku připočteme 15. Tím obdržíme adresu, na níž je uložen další vektor, určující startovací adresu tiskové rutiny. Když budete do paměti ukládat rutinu vašeho interfejsu např. od adresy #F000, musíte změnit uvedený vektor tak, že na jeho nižší adresu uložíte bajt #00, na vyšší #F0. Tisk pak bude po zadání některého z příkazů k tisku z GENS3 či MONS3 probíhat zcela automaticky. Tuto úpravu si můžete natrvalo umístit do pomocného Basicu. Nezapomeňte však, že tiskárně musíte (prostřednictvím tiskové rutiny) předem zadat potřebné řídící kódy pro průběh tisku (především pro jeho formát), nejspíše však i vytvořit vlastní subrutinu pro průběžné ovládání tisku.

Pokud jde o tisk se sériovým přenosem, u něj se vše řídí obvykle implementovanými basicovými příkazy. Např. interfejs ZX1 má tyto příkazy implementovány (viz jeho manuál). Pochopitelně i zde můžete sestavit vlastní tiskovou rutinu ve strojovém kódu. K tomu je však bezpodmínečně nutný přístup k patřičné literatuře (přinejmenším k výpisu ROMky sériového interfejsu).

DOSLOV PÁR SLOV K ZAČÁTEČNÍKŮM

Jste-li začátečník v užívání programů typu generátor/monitor, věnujte svou pozornost následujícím dobře míněným slovům.

Nejen ze své zkušenosti víme, že brilantní ovládání obou těchto programů přichází až s jejich delším používáním. Lidé, kteří velmi dobře ovládají programovací jazyk assembler, ale nemají moc času na praktické využívání všech fines obou programů, často zůstávají u používání samotného monitoru s tím, že si napřed svůj prográmek na papíru přepíší z assembleru do strojového kódu, který pak s pomocí monitoru ukládají přímo do paměti počítače. Monitor má dostatek schopností pro dodatečné úpravy. Je pravda, že zápis do generátoru je mnohem elegantnější a dává vám nepoměrně větší šanci v tom, že se do svých rutin nezamotáte. Ale ovládnout perfektně oba programy je pro běžného amatérského programátora záležitostí dlouhodobějšího charakteru.

Proto bychom těm z vás, kteří nemáte ani moc zkušeností, ani času, chtěli doporučit, abyste se napřed zabývali monitorem, s nímž je zpočátku větší zábava. Během toho ale přece jen tu a tam vstupte do generátoru a pokoušejte se mu přijít na kloub. Vaším cílem se musí stát zvládnutí obou programů tak, abyste s nimi mohli pracovat v kombinaci přinejmenším tak, jak je uvedeno v "Příkladu kombinované práce s MONS3 a GENS3". Jinak bude vaše programování značně neefektivní. Co nejdříve se naučte využívat buffer GENS3 pro ukládání hotových rutin v kombinaci s částmi zdrojového textu, které budete dále rozvíjet či upravovat. Při tvorbě rozsáhlejších programů se bez využití této dispozice GENS3 neobejdete.

Úspěšné používání obou systémových programů předpokládá velmi dobrou znalost assembleru, strojového kódu i mnohého z toho, jak to v počítači všechno "funguje". Čím víc budete znát, tím vám programování půjde lépe od ruky. To se týká i analýzy programů. Dobrou vlastností monitoru je, že může být značně nápomocen i při samotné výuce programování, hlavně díky četným možnostem krokování rutinami.

Jako začátečníka vás ještě upozorníme na jednu věc, kterou je třeba mít stále na (své) paměti. Kdykoli vytvoříte nějaký program ve strojovém kódu, nikdy jej nespouštějte před tím, než jej zaznamenáte na magnetofonový pásek! I velmi schop-

ným assemblerovým programátorům nepracují jejich programy na první spuštění. Jako adepti assemblerové magie počítejte s takřka stoprocentní jistotou, že vaše čerstvě naprogramované rutiny po svém spuštění nenávratně zmizí spolu se vším, co jste v paměti počítače měli. A můžete si být jisti, že chyba nebyla nikde jinde než ve vašem programu. Generátor ani monitor za nic nemohou. Stejného překvapení se můžete dočkat i při krokování, kterým budete svůj program vylaďovat. Stačí malá chybička při zápisu vašeho kódu, sebemenší zmatek v paměťovém zásobníku a vše je neodvolatelně pryč. Proto je neustálé zaznamenávání kopií po každém úspěšném odladění jakékoli chyby naprostou nezbytností. Touto nepříjemnou vlastností se strojový kód liší od vyšších programovacích jazyků, které jsou do značné míry systémově kontrolovány a nekolabují tak často.

Mikrobázi nabízený program mikROMkód (lze objednat) vám poskytne případně chybějící znalosti v ovládání ZX Spectra i výbornou příležitost ke hrátkám se strojovým kódem za použití monitoru i generátoru. Měňte obsahy jednotlivých adres a sledujte, co to ve výsledku přinese. Přitom se ale snažte pochopit, proč se stalo to, co se stalo. Pokud něčemu z Dr.MG či mikROMkódu nebudete schopni ani za nic porozumět, napište Mikrobázi. Vaše dotazy zodpovíme v jejím zpravodaji.

Aby vaše námaha spojená se studiem programování mikroprocesoru Z80 byla vykoupena radostí z perfektních vlastních programů

vám přejí

autoři Dr.MG a Mikrobáze

Vydala 602. ZO Svazarmu jako součást dodávek programového vybavení v rámci služeb Mikrobáze. Samostatně neprodejné! Adresa vydavatele a výrobce: 602 ZO Svazarmu, Wintrova 8, 160 41, tel. 32 85 63.

PŘÍLOHA 1 CHYBOVÁ HLÁŠENÍ A JEJICH VÝZNAM

chyba 1	Chyba v kontextu dané řádky
chyba 2	Neznámá mnemonika
chyba 3	Chybně formulovaná příkazová řádka
chyba 4	Vícenásobně definovaný symbol
chyba 5	Řádka obsahuje nedovolený znak, tj. znak, který je v daném kontextu neplatný
chyba 6	Nedovolený operand
chyba 7	Použitý symbol je rezervované slovo
chyba 8	Neshoda registrů (mismatch)
chyba 9	Mnoho registrů v příkazu
chyba 10	Výraz použitý jako osmibitový má větší hodnotu než 255
chyba 11	Instrukce JP (IX+n) a JP (IY+n) nejsou dovoleny
chyba 12	Chyba v uspořádání assemblerového povelu
chyba 13	Nedovolená předběžná reference, tj. použití EQU pro dosud nedefinovaný symbol
chyba 14	Dělení nulou
chyba 15	Přeplnění při násobení
chyba ORG	Byl proveden ORG pro adresu, která zasahuje do GENS3, textového souboru nebo tabulky symbolů. Řízení se vrací do editoru.
zväčšiť tab!	Objeví se při prvním průběhu, pokud tabulka symbolů nemá dostatečnou velikost. Řízení se okamžitě vrací do editoru.
prepl.pamäti	Zobrazí se, když není prostor pro další text, tj. konec textu se blíží k RAMTOP. Stávající textový soubor nebo jeho část by se měla uložit na pásek.

PŘÍLOHA 2 REZERVOVANÁ SLOVA, MNEMONIKA ATD.

Zde je uveden seznam rezervovaných slov pro GENS3. Tyto symboly nesmějí být použity jako návěští, ale mohou tvořit část návěští. Všimněte si, že rezervovaná slova jsou tvořena velkými písmeny.

A	B	C	D	E	H	L	I	R	S
AF		AF'		BC		DE		HL	IX
IY		SP		NC		Z		NZ	M
P		PE		PO					

Následuje seznam platné mnemoniky Z80, příkazů a povelů assembleru. Všechna jejich písmena musejí být vkládána velká.

ADC	ADD	AND	BIT	CALL	CCF	CP	CPD	CPDR
CPI	CPIR	CPL	DAA	DEC	DI	DJNZ	EI	EX
EXX	HALT	IM	IN	INC	IND	INDR	INI	INIR
JP	JR	LD	LDD	LDDR	LDI	LDIR	NEG	NOP
OR	OTDR	OTIR	OUT	OUTD	OUTI	POP	PUSH	RES
RET	RETI	RETN	RL	RLA	RLC	RLCA	RLD	RR
PRA	RRC	RRCA	RRD	RST	SBC	SCF	SET	SLA
SRA	SRL	SUB	XOR					

DEFB	DEFM	DEFS	DEFW	ELSE	END	ENT	EQU	IF
ORG								

≡D	≡E	≡H	≡L	≡S	≡C	≡F

PŘÍLOHA 3 PŘEHLED PŘÍKAZŮ MONS3

Použité zkratky: adr - adresa, na niž ukazuje střelka
rr - registrový pár
st - střelka
cs - CAPS SHIFT
ss - SYMBOL SHIFT

Příkaz	funkce		Komentář
Manipulace se střelkou			
M NNNN	LD st,#NNNN		nastavení střelky z klávesnice
ENTER kur.nahoru kur.vpravo kur.vlevo	adr+1 adr-1 adr+8		skok střelky o 1 nebo 8 adres nahoru nebo dolů
X	JP (adr)	Návrat V	přímý skok střelky na adresu danou obsahy adr a adr+1
0	JR (adr)	Návrat U	relat.skok střelky, obsah adr je pojímán jako dvojk.doplněk
.	LD st,(SP)		přemístění střelky na adresu z konce zásobníku
Manipulace s registry			
.	Volba registrů		Posun registrového kurzoru
NNNN.	LD rr,#NNNN		Registry, na něž ukazuje kurzor, budou obsahovat #NNNN
Q	Přepnutí registrových bank		Výměna reg.párů nečárkovaných za čárkované a naopak
Modifikace paměti			
NN ENTER	LD (st), #NN		Vložení #NN či #NNNN na adr, resp. adr a adr+1
NNNN ENTER	LD (st), #NNNN		
P A1 A2 B	Bajt B do adres A1 až A2		Všechny adresy v rozsahu A1-A2 budou obsahovat bajt B
Y	Uložení ASCII	Návrat cs5	Umožnění zápisu znaků ASCII přímo na adresy od adr výše
I A1 A2 A3	Přenos bloku A1-A2 na A3		Obsah bloku adres A1-A2 se přenese do bloku s 1.adresou A3

Ladění programů

W	Bod přerušení		Bod se nastavuje na pozici st
ssK	Spuštění progr.		od adresy v PC po bod přerušení
J XXXX	-dtto-	Návrat cs5	Od adr #XXXX po bod přerušení
ssZ	Krokování progr.		PC a adr musejí být shodné!
ssT	Bod přer. za instr. ve st	Návrat cs5	Jako ssK, ale progr. se zastaví za instr.,kterou adresuje st; vhodné pro rychlé proved.subr.

Výpis a prohledávání paměti, generování zdrojového textu

L	Výpis od adr	Návrat cs5	Hexadek.a ASCII reprezentace obsahu 80ti sousedících adres
ssP	-dtto-	-dtto-	Totéž, ale výstup na tiskárnu
G xB ENTER	Hledání zadaného řetězce od adr		Řetězec složený z x bajtů B (každý potvrzen stiskem ENTERu) je v paměti hledán od adr
N	-dtto-		Hledání dalšího výskytu řetězce
ss4	Disassembler jednoduchý	Návrat ss4	Dekompilace bez generace návěští, 1 strana obsahuje blok 20ti adres, probíhá od adr
T xparametry viz manuál	Disassembler hlavní	Návrat cs5	Dekompilace s generací návěští, zdroj.textu pro GENS3 a možností výstupu na tiskárnu

Ostatní

H N ENTER	Čísel.konverze		dek.čísla N<65535 na hexadek.
ss3	Přep.č. repr.adr.	Návrat ss3	Mění výpis adres z hexadek. tvaru na dekad. a zpět
EDIT	Návrat z MONS3		do modulu DOKTOR, resp.Basicu

PŘÍLOHA 4 PŘEHLED PŘÍKAZŮ GENS3 - EDITOR

Příkaz	Funkce	Komentář

Funkce TAB

ENTER	CR (CGR$ 13)	Nová řádka
DELETE	Kurzor doleva	Posun kurzoru vlevo,výmaz řád.
cs8	Posun TAB	Skok kurzoru na další poz.TAB
cs5	Výmaz řádky	Vymazání vkládané řádky
SPACE	Mezera	Vložení mezery

Základní příkazy editoru

I n,m	Automat. řádkování	Návrat EDIT	Automaticky řádkuje řádky n, v intervalech m
L	Listování	Návrat EDIT	Výpis textu v intervalu řádek n až m, samotné L vypíše celý
K n	Počet ř.výpisu		Počet řádek 1 strany výpisu

Úprava zdrojového textu

D n,m	Výmaz řádek v intervalu n až m		
M n,m	Přesun řádek " "		
N n,m	Přečíslování řádek od řádky n s intervalem m		
Fn,m,$1,$2	Hledání v textu	Návrat ENTER	Je hledán řetězec $1 v interv. řádek n až m
F	-dtto-	-dtto-	Je hledán další výskyt řetězce $1 (viz Fn,m,$1,$2)
S	Výměna řetězců	-dtto-	Všechny řetězce s obsahem $1 jsou nahrazovány řetězcem $2

Úprava řádky zdrojového textu
(Návrat z těchto příkazů se provede stiskem ENTERu)

E n	Editace řádky n	V editační zóně se zobrazí řádka n
SPACE	Kurzor doprava	Posun kurzoru doprava po znaku
cs8	Na TAB	Posun kurz.na další pozici TAB
Q	Zrušení úpravy	Ignoruje na řádce provedené změny
R	Původní stav	Obnovení původního obsahu řádky z vyrovnávací paměti
L	Výpis	Vypsání zbytku řádky
K	Vymazání znaku	na pozici kurzoru
Z	Vymazání znaků	od poz.kurzoru do konce řádky
C	Změna znaku	na pozici kurzoru
I	Vsunutí znaku	" "
X	Kurz.na konec ř.	s aktivací funkce I

Práce s magnetofonem

G,,název	LOAD	Návrat BREAK	Text,zapsaný na pásek příkazem P, se uloží do paměti
Pn,m,název	SAVE	-dtto-	Zápis textu (řádek n až m) pro LOAD příkazem G
Tn,m,název	SAVE	-dtto-	dtto, ale pro LOAD příkazem ≡F
	- Příkaz ≡F (LOAD) viz assembler -		

Ostatní

W n,m	Tisk	Návrat BREAK	Výtisk textu (řádek n až m) na tiskárně
V	Výpis parametrů n,m,ß1,ß2 (viz Fn,m,ß1,ß2)		
X	Rozsah textu		Výpis 1.a posl.adresy uložení
S,,znak	Separátor		Určení nového separátoru
C	Přeformátování GENS1 na GENS3		
B	Návrat z GENS3		do modulu DOKTOR, resp.Basicu

PŘILOHA 5 PŘEHLED PŘIKAZŮ GENS3 - ASSEMBLER

Přímé příkazy

A ENTER	Kompilace	S následnou volbou činnosti
R ENTER	Start programu	Viz pseudoinstrukce ENT

Činnosti

Činnost 1	Kompilace s výpisem symbolů na konci 2.průběhu
" 2	Kompilace bez generování strojového kódu
" 4	Kompilace bez výpisu jejího výsledku
" 8	Kompilace s výstupem na tiskárnu
" 16	Kompilace s gener.str.kódu za tabulku symbolů
" 32	Kompilace bez kontroly umístění strojového kódu

Pseudoinstrukce nepodmíněné (v značí výraz)

ORG v	Aktualizace čítače adres	Nastavení čítače na hodnotu "v"
EQU v	Přiřazení hodnoty návěští	Návěští přijme hodnotu "v"
DEFB v,v..	Určení obsahu bajtu/-ů	Bajt/-y přijme/-ou hodnotu/-y "v"
DEFW v,v..	Určení obsahu slova	Uložení dvoubajt.slov na adresy (ve formátu Z80)
DEFS v	Rezervace oblasti paměti	Počet rezervovaných adres je dán hodnotou "v"
DEFM "ß"	Definice textu	Od dané adresy se uloží ASCII kody řetězce ß

Pseudoinstrukce podmíněné

ENT v	Start,adr.progr.	Hodnota "v" je start.adr.pro spuštění programu příkazem R
IF v	Když v=0, kompilace je blokována Když v má nenulovou hodnotu, kompilace proběhne	
ELSE	Přepínač aktivace kompilace	Byla-li kompilace spuštěna před ELSE, ukončí ji, a naopak
END	Aktivace kompilace	Zrušení blokády kompilace způsobené příkazem IF v, když v=0

Ostatní řádkové příkazy

≡E	Vložení 3 mezer		Pro přehlednost výpisu
≡Hβ	Vložení záhlaví		Vypíše záhlaví (obsah řetězce β) na řádkách s ≡H i s ≡E
≡S	Zastavení rolování výpisu		Pro snadnější "odchycení" části výpisu (neplatí pro tiskárnu)
≡L-	Blokáda výpisu	Návrat ≡L+	Přeruší výpis kompilace, ale generování str.kodu pokračuje
≡D+	Dek.výpis adres	Návrat ≡D-	Volba čís.reprezentace výpisu adres (dekad.nebo hexadec.)
≡C-	Zkrácení výp.řád.	Návrat ≡C+	Vynechá výpis 9 znaků (8 str. kodu a 1 mezeru); kratší řádka
≡F	LOAD s využitím bufferu pro souběžný překlad načítaného textu do strojového kodu; proběhne ve dvou průbězích, bez uložení textu		