ZX SPECTRUM & KOMP.

PROMETHEUS

ASSEMBLER
MONITOR
} Z 80

INSTRUCTION MANUAL

L a d í c í s y s t é m P R O M E T H E U S

I. Trocha sebechvály aneb odkud to přišlo

Při psaní programů v assembleru na Spectru člověk brzy narazí na problém velikosti paměti - do paměti se musí vejít assembler, zdrojový text, přeložený kód a pokud možno nějaký prostředek pro ladění. Nabízejí se dvě řešení, buď si koupit něco většího než je Spectrum - toto řešení má malý zádrhel, chce to peníze - nebo získat programy, které s tím malým množstvím paměti, které je k dispozici, zacházejí úsporně. Nápady jako jsou kompilace přímo z pásku jsou silně kostrbaté a pro lid nevlastnící ani disketovou jednotku ani microdrive prakticky nepoužitelné.

Rozhlédne-li se uživatel kolem sebe na nabídku assemblerů a ladících prostředků zjistí následující fakta:

GENS - nejstarší a nejrozšířenější assembler, zdrojový text je uložen jako textový soubor, poměr mezi délkou zdrojového textu a překladu je asi 8:1, snad nejhorší editor, který si lze vymyslet, assembler je dlouhý přibližně 8,5 KB, překládá poměrně rychle, silně obtěžující je neustálá otázka 'Table size:', kde implicitní hodnota obvykle nestačí a po hlášení 'No table space' je nutno kompilaci opakovat s větší tabulkou, také dotaz 'Option:' při každé kompilaci práci zbytečně neurychluje, program je relokovatelný, což jeho hodnotu velmi zvyšuje, k programu existuje monitor MONS - bohužel práce s ním není pohodlná pro uživatele nepoužívající hexa soustavu, ani služby, které monitor poskytuje už nejsou na úrovni doby, monitor je dlouhý 5 KB a je relokovatelný

Rozšířené verze GENSu

- GENSJE (Lamač) - vylepšaná verze programu GENS, obsahuje Full Screen Editor s automatickou tabelací a 42 znaky na řádek, program je 12 KB dlouhý a zachoval si relokovatelnost

- Adámkovo rozšíření - podobně jako GENSJE má Full Screen Editor s automatickou tabelací, na řádku je 64 znaků (horší čitelnost), program už není relokovatelný

- MASM (MACROS, ASSEMBLER 90) - u tohoto programu by se dalo polemizovat, jestli je nebo není od programu GENS odvozen, má s ním však velké množství shodných prvků (bohužel i špatných), délka programu je 11 KB, není relokovatelný, má Full Screen Editor s automatickou tabelací, zná 'neznámé instrukce', má křížové reference, celkově lze říci, že to není žádná revoluce, zdrojový text je uložen stejně neefektivně jako u programu GENS

OCP assembler - program je nerelokovatelný, zdrojový text zabírá více místa než stejný u programu GENS, před kompilací je nutno zadávat delší posloupnost parametrů, má Full Screen Editor, chyby na rozdíl od programu GENS (i modifikací) nevypisuje čísly (tzn mít po ruce tabulku) ale přímo slovy (práce je pohodlnější)

Laser Genius Assembler - program je relokovatelný, má délku kolem 20 KB (s lopatou na komára), pro svou délku je těžko použitelný bez disketové jednotky nebo microdrivů, existuje k němu Laser Genius Monitor, oba programy se však do paměti

najednou téměř nevejdou

Memory Resident System - první assembler, který se snaží uložit efektivně zdrojový text, poměr je okolo 4:1 (tedy při stejné délce pojme dvakrát tak dlouhý zdrojový text), program je dlouhý 11 KB a není relokovatelný (existují 3 verze), má Full Screen Editor a kontroluje instrukce už při vkládání (odstraní se mnoho chyb), ve snaze zkrátit zdrojový text byla použita některá ne příliš šťastná omezení - maximálně 255 návěští, ve výrazu se smí použít maximálně dva operandy a jeden operátor, nelze použít binární zápis čísel (zvláště první omezení doslova zabíjí dobrý nápad), rozporný je také monitor, dokáže při trasování hlídat assembler a zdrojový text před náhodným přepsáním, bohužel všechny výpisy jsou v šestnáctkové soustavě, v monitoru se lze odkazovat na existující návěští v tabulce symbolů, při psaní MRS byl zřejmě požadavek na krátkost přeceněn (program je dlouhý 12 KB) ale na úkor komfortu obsluhy i možností

PIKASM - program od T.R.C, je 11 KB dlouhý, není relokovatelný, má Full Screen Editor, zdrojový text ukládá v komprimované formě (poměr 5-6:1), na rozdíl od předchozích editorů má příjemnou vlastnost - po kompilaci zůstává kurzor na tom místě, kde byl před kompilací (při vstupu do editoru neskáče ani na konec ani na začátek), assembler také umí při kompilaci před každou instrukci řídící běh programu (call, jp, ret) vložit volání testu klávesy BREAK - užitečné při ladění bez monitoru (lze zastavit program, který se zacyklil (tzv. kousnul)

MON, MON2 - monitory, dlouhé 7,5 a 9 KB, oba jsou relokovatelné, umí krokovat instrukce - ručně i automaticky (s vypisováním po každé instrukci i bez něj - velmi rychlé), vypisovat paměť, měnit obsah paměti, plnit a přesouvat bloky, lze také nadefinovat oblast obrazovky, kam budou mířit výpisy (souvislý úsek řádků na obrazovce)

VAST (PIKOMON, MILIMON, MIKROMON) - monitor dlouhý 4,6 KB, umístěný v obrazovkové paměti - původně byl určen pro hledání nekonečných životů, lze ho použít i při ladění programů v závěrečné fázi kdy je v paměti málo místa. Existuje nová verze nazvaná DEVAST, vylepšená o další služby, tento program však na rozdíl od předchozích nemá na svědomí T.R.C nýbrž moje maličkost, existuje i verze pracující na zvoleném místě v paměti. Pro zajímavost: program DEVAST je dlouhý asi 4,6 KB a jeho zdrojový text v assembleru PROMETHEUS má délku kolem 16 KB

Uvedený přehled ukazuje autorovi známý stav a jeho názory na dostupné programy. Z uvedených programů jsou často používané kombinace:

GENS + MONS
GENS + MON (MON2)
OCP + MONS
Memory Resident System
Laser Genius

Při posouzení dostupných programů vzniká myšlenka vyrobit program, který by spojoval výhody všech předchozích programů a neobsahoval jejich nevýhody.

Nejdůležitější vlastnosti, které by takový program měl mít jsou dvě: co nejefektivnější uložení zdrojového textu a spojení assembleru a monitoru do jednoho celku, tak, aby mohl monitor

používal některé podprogramy z assembleru a používal existující tabulku symbolů při výpisech. Proto vznikl tento program.

Assembler a monitor, ladící systém PROMETHEUS vznikal za pomoci programů GENS3E, GENS 3.1, MON, MON2 a VAST. Program GENS byl používán do té doby, než se podařilo odladit první provozuschopnou verzi assembleru PROMETHEUS, další vylepšování se provádělo vždy na starší verzi - assembler byl překládán sám sebou, monitor vznikl úplně od začátku na assembleru PROMETHEUS. Pro srovnání - zdrojový text v GENS byl dlouhý kolem 40 KB a byl tedy nutně rozdělen na dva, překládal se celkem 40 sekund, po přetažení zdrojového textu do PROMETHEA se zdrojový text zkrátil na polovinu (20 KB), odpadla potřeba místa na tabulku symbolů - u obou částí asi po 3 KB, text se tedy vešel do paměti najednou (asi 5 KB kódu assembleru jsou tabulky a pracovní místa, která nebyla ve zdrojovém textu). Celý zdrojový text byl nyní přeložen za 3 sekundy (tedy 2 KB/s).

Nejdůležitější vlastnosti ladícího systému PROMETHEUS

- délka: 11 KB assembler, 5 KB monitor (celkem 16 KB, monitor lze odpojit a neužívat)
- celý program je relokovatelný, a není závislý na systému Spectra
- poměr mezi zdrojovým textem a přeloženým kódem je 3-4:1 (GENS 8:1)
- tabulka symbolů je organickou částí zdrojového textu, vytváří se hned při psaní textu, vždy je jihned přístupná - tzn. že pro kompilaci už není třeba ani byte navíc
- rychlost kompilace je asi 2-3 KB/s - 10x víc než GENS
- editor je řádkově orientovaný, má automatickou tabelaci - skrolování a stránkování je velmi rychlé, umožňuje mazání a kopírování bloků, vyhledávání a nahrazování řetězců, každý řádek je při odeslání syntakticky kontrolován
- obvyklé kazetové operace - SAVE, VERIFY, LOAD, možnost vyhrát i přihrát (na libovolné místo) kus zdrojového textu
- příkaz GENS umožňuje nahrát zdrojový text ve ve formátu assembleru GENS (MASM)
- monitor chránící zdrojový text a ladící systém
- několik trasovacích režimů
- možnost definovat okna se zákazem zápisu, čtení nebo běhu
- možnost zcela ovlivnit složení, tvar a polohu čelního panelu, počet a způsob výpisu všech položek
- při krokování umí monitor zjistit časovou náročnost krokovaného programu - počítá cykly procesoru (tzv Těčka)

Detailně jsou vlastnosti a možnosti tohoto programu popsány v dalším textu.

II. Instalace ladicího systému

Na kazetě je program uložen jako blok CODE dlouhý asi 18500 bytů - obsahuje 5000 bytů monitoru, 11000 bytů assembleru a 2500 bytů instalačního programu a tabulky relokačních adres.

Program lze nahrát na libovolnou adresu v rozmezí adres 24000 až 47000 a od této adresy spustit:

```
CLEAR adr-1 : LOAD ""CODE adr : RANDOMIZE USR adr
```

Po spuštění se objeví instalační obrazovka obsahující následující informace:

Instalation address:xxxxx_ - kde xxxxx je adresa kam byl program nahrán - tuto adresu je možno přepsat pomocí kláves 0 až 9 a DELETE (CAPS SHIFT+0) na libovolnou adresu v rozsahu adres 23296 až 49500 (23296 až 54500 bez monitoru). Na tuto adresu bude PROMETHEUS umístěn a od ní ho bude možno z BASICu kdykoliv opět spustit. Umístíte-li program od adresy 23296, nelze se již vrátit do BASICu - program přepíše systémové proměnné. Budete-li používat instalaci na adresu pod nastavený RAMTOP, nelze se již vrátit do BASICu bezpečně. Umístíte-li ladicí systém tak, že přepíše systémové proměnné (tzn. na adresu nižší než 23755), nesmí Vaše programy používat služby BASICu (tisk znaků atd.) a povolovat přerušení v módu IM 1. Obvykle je nejlépe použít pro instalaci adresu 25000.

Monitor:Yes - instalovat je možno buď celý ladicí systém nebo jen assembler, změnu volby docílíte stisknutím klávesy M. Instalovat jen samotný monitor není možné protože používá mnoho podprogramů assembleru.

Dále lze nastavit barvy, které bude assembler používat. Je to barvy papíru - klávesa P, barva inkoustu - klávesa I a jas těchto dvou barev - klávesa B. Při editaci je jeden řádek vyznačen jinou barvou - nastavení je obdobné, je nutné stisknout současně navíc CAPS SHIFT.

PROMETHEUS umí psát dvěma druhy písma - obvyklým a dvojnásobně tlustým (může být lépe čitelné) - přepínání mezi oběma typy písma klávesou D.

Při editaci vypisuje assembler (i disassembler) instrukce malými písmeny a návěští velkými písmeny, je-li však někdo zvyklý používat jen velká (jako GENS) nebo malá písmena (MRS), může toto nastavení změnit klávesou C - výsledek je hned vidět na instalační obrazovce.

Poslední vlastnost ladicího systému, kterou je možno ovlivnit při instalaci je klávesnicové echo - mění se klávesou X jedním směrem a klávesami CAPS SHIFT + X druhým směrem - současné nastavení je slyšet při stisku libovolné klávesy.

Po nastavení požadovaných parametrů stiskněte klávesu ENTER, program se přesune na zvolenou adresu, přepíší se absolutní adresy (relokace) a program se spustí.

Tuto instalaci můžete provádět při každém použití ladicího systému (obvykle stačí jen stisknout ENTER) nebo si požadovanou konfiguraci vyhrát na pracovní kazetu - nejlépe následovně:

Připište krátký program v Basicu:

```
    1 RANDOMIZE USR adr:STOP
 9999 CLEAR adr-1:LOAD ""CODE:RUN
```

Kam místo 'adr' a 'adr-1' dosadíte čísla (adresa kde je program) a na kazetu uložíte pomocí:

```
SAVE "PROMETHEUS" LINE 9999:
SAVE "prometheus" CODE adr,16000 (11000 bez monitoru)
```

Kde opět místo 'adr' bude adresa. Tuto konfiguraci nahrajete pomocí obyčejného LOAD "" a z Basicu budete spouštět jednoduše příkazem RUN. První řádek z tohoto programu je dobré vložit vždy - PROMETHEUS se bude spouštět příkazem RUN a nehrozí chyba, která by mohla vzniknout při psaní 'RANDOMIZE USR adr' , o tom, že funkce USR použitá se špatným argumentem může značně narušit duševní pohodu uživatele snad není nutno nijak zdůrazňovat.

III. assembler PROMETHEUS

1. editor

Assembler obsahuje pohodlný řádkový editor, který se hodně podobá editoru BASICu. V editoru se zadávají také všechny příkazy - odpadá tedy nutnost neustálého přecházení z editoru do systému a naopak spojená se ztrátou pozice v editoru jako je tomu u MRSu a rozšířených verzí GENSu. Editor umožňuje blokové operace - mazání a přesouvání - a vyhledávání a nahrazování řetězce v textu.

Označení:

přístupový (přístupný) řádek - barevně odlišený řádek ve výpisu

editační řádek - úplně spodní řádek, je na něm kurzor

stavový řádek obsahuje toto (zleva doprava):
- sdělení nebo chybové hlášení
- informaci o nastaveném režimu (INSERT/OVERWRITE)
- adresu posledního bytu zdrojového textu
- adresu posledního bytu oblasti pro zdrojový text, jakási obdoba RAMTOP v BASICu

Vkládání příkazů se provádí stisknutím klávesy SYMBOL SHIFT a odpovídajícího písmene při prázdném editačním řádku, tedy kurzor je úplně vlevo a za ním není žádný znak (ani mezera). Nejsou-li tyto podmínky splněny, napíše se znak na dané klávese se SYMBOL SHIFTem, jsou-li splněny, vypíše se celý příkaz (jako klíčové slovo) nebo se ihned provede (některé příkazy - skok na konec zdrojového textu ald.).

Zápis instrukce - na řádek lze zapsat maximálně 31 znaků, prvních 9 pozic je vyhrazeno pro návěští (maximálně 8 znaková), dalších 5 pozic je vyhrazeno pro mnemoniku a zbývající místa jsou vyhrazena pro operandy. Poznámky je možno psát pouze na zvláštní řádek a to hned od začátku za středník - nelze psát komentáře za instrukce (ony by se tam stejně nevešly). Pokud se někomu zdá 31 znaků málo, nezbývá mu než si buď zvyknout nebo používat jiný assembler. K tomuto omezení jsem dospěl na základě zkušeností svých i svých přátel, téměř nikdo, koho znám nepíše komentáře do zdrojového textu (maximálně na papír s programem), je to práce navíc a zdrojový text to neúměrně prodlužuje - a zpomaluje to překlad i nahrávání (to zvláště). Omezení má také své světlé stránky, zdrojový text je dobře čitelný a přehledný.

Mezerník automaticky tabeluje, instrukce je možno psát libovolně, malými nebo velkými písmeny, program rozliší návěští a zkontroluje vkládaný řádek, najde-li chybu, ohlásí ji a řádek nezařadí do výpisu.

Editační klávesy:

CAPS SHIFT + 1 (EDIT) - přenese přístupový řádek do editačního řádku

CAPS SHIFT + 2 (CAPS LOCK) - zapíná / vypíná CAPS LOCK, ten pracuje trochu jinak než obvykle, původně píše Spectrum při zapnutém CAPS LOCKu pouze velká písmena, nyní funguje spíše jako jakýsi invertor CAPS SHIFTu, když je zapnut CAPS LOCK a není stisknut CAPS SHIFT píše Spectrum velká písmena, a je-li stisknut CAPS SHIFT, píše Spectrum malá písmena

CAPS SHIFT + 3 (TRUE VIDEO) - přechod ve výpisu na následující stránku

CAPS SHIFT + 4 (INV. VIDEO) - přechod ve výpisu na předcházející stránku

CAPS SHIFT + 5 (šipka doleva) - posun kurzoru doleva o jeden znak v editačním řádku

CAPS SHIFT + 6 (šipka dolu) - posun přístupného řádku na předchozí řádek ve výpisu

CAPS SHIFT + 7 (šipka nahorů) - posun přístupného řádku na následující řádek ve výpisu

CAPS SHIFT + 8 (šipka doprava) - posun kurzoru vpravo o jeden znak v editačním řádku

CAPS SHIFT + 9 (GRAPH) - vymaž přístupový řádek posuň se na předchozí řádek

CAPS SHIFT + 0 (DELETE) - vymaž znak vlevo před kurzorem a posuň kurzor doleva

SYMBOL SHIFT + V - přepínání režimů INSERT a OVERWRITE (písmeno O nebo I ve stavovém řádku), při stisku EDIT se přepne režim OVERWRITE a po odeslání se vrátí zpět režim INSERT , chcete-li zkopírovat jeden řádek, vyeditujte ho, přepněte na INSERT a vložte na požadované místo, chcete-li úplně přepsat řádek, napište text, přepněte na OVERWRITE a odešlete

SYMBOL SHIFT + D - vymazání editačního řádku, ne při prázdném řádku

CAPS SHIFT + SYMBOL SHIFT (EXTEND MODE) - nastaví okraj bloku - příslušnost řádku k bloku je signalizována černým čtvercem na deváté pozici, mezi návěštím a instrukcí - označení je vidět pouze ve výpisu, nikoliv při hledání nebo tisku, není rozlišen začátek a konec, program si pamatuje vždy dva okraje a při zadání nového odstraní ten, co byl zadán dříve a přidá nový okraj, chcete-li nastavit blok, nastavte jeden okraj do přístupového řádku, stiskněte CAPS SHIFT + SYMBOL SHIFT , nastavte druhý okraj bloku do přístupového řádku a stiskněte CAPS SHIFT + SYMBOL SHIFT

ENTER - odeslání instrukce nebo příkazu, odeslání instrukce má za následek její syntaktickou kontrolu a zařazení do výpisu - podle režimu (I/O) se buď zařadí za instrukci v přístupném řádku nebo ji přepíše

SYMBOL SHIFT + E - jdi na konec zdrojového textu, jde o příkaz, který se jinned provede, je ho tedy nutno zadat při prázdném editačním řádku

SYMBOL SHIFT + K - jdi na začátek zdrojového

textu, jde o příkaz, který se ihned provede, je ho tedy nutno zadat při prázdném editačním řádku

SYMBOL SHIFT + H - přepínání výpisu čísel mezi desítkovou a šestnáctkovou soustavou, ovlivní výpisy ve stavovém řádku a v tabulce symbolů

Příkazy pro blokové operace:

COPY (SYMBOL SHIFT + C) - přenese nastavený blok před přístupový řádek - nelze kopírovat blok do sebe

DELETE (SYMBOL SHIFT + D) - vymaže nastavený blok zdrojového textu

Příkazy pro vyhledávání a nahrazování řetězců:

FIND (SYMBOL SHIFT + F) - vyhledávání (viz dále), nerozlišují se malá a velká písmena, není-li řetězec nalezen, zůstane výpis na místě

REPLACE (SYMBOL SHIFT + Z) - zaměňování řetězců

Vyhledávání řetězců:

FIND s:xxxxx - vyhledání řetězce 'xxxxx' od začátku zdrojového textu

FIND s - vyhledání naposledy zadaného řetězce od začátku zdrojového textu

FIND b:xxxxx - vyhledání řetězce 'xxxxx' v bloku od jeho od začátku bloku

FIND b - vyhledání naposledy zadaného řetězce v bloku od začátku bloku

FIND :xxxxx - vyhledání řetězce 'xxxxx' od řádku, který je právě přístupný buď do konce zdrojového textu nebo do konce bloku - toto v případě, že posledně byl prohledáván blok

FIND - vyhledání naposledy zadaného řetězce od řádku, který je právě přístupný buď do konce zdrojového textu nebo do konce bloku, opět to závisí na posledním způsobu prohledávání

FIND b: - tímto příkazem se lze dostat okamžitě na začátek bloku (za dvojtečku nutno zapsat jednu mezeru!)

Příklady:

- chci vyhledat všechny výskyty návěští 'KAREL', zadám 'FIND s:karel' (nerozlišují se malá a velká), po nalezení prvního výskytu zadám pokaždé jen 'FIND' pro nalezení dalšího, jakmile se po zadání 'FIND' jen obnoví výpis na stejném místě, jedná se o poslední výskyt řetězce 'KAREL'

- chci se opět dostat na první výskyt řetězce 'KAREL',

zadám jen 'FIND s' a dostanu se na první výskyt

- chci vyhledat výskyt návěští 'KAREL' pouze v poli návěští (toto je nejrychlejší způsob jak se dostat ve výpisu na požadované místo), zadám 'FIND s:KAREL ' - za návěští vložím mezeru!

- chci vyhledat první instrukci ld a,(hl) od začátku zdrojového textu, zadám 'FIND s:ld a,(hl)', mezi mnemoniku a operandy musím vložit tolik mezer, aby jejich počet plus počet písmen mnemoniky dal součet 5

Nahrazování řetězců:

REPLACE :xxxxx - nahradí všechny výskyty řetězce 'yyyy' vyhledaného příkazem 'FIND :yyyy' v daném řádku řetězcem 'xxxxx' a vyhledá další výskyt řetězce 'yyyy'

REPLACE - nahradí všechny výskyty řetězce, který byl zadán příkazem FIND, řetězcem, který byl zadán příkazem REPLACE

Příklad:

- chci najít a vyměnit některé výskyty řetězce 'KAREL' řetězcem 'JOSEF', zadám příkaz 'FIND s:karel', po nalezení výskytu řetězce 'KAREL' udělám:
  - buď zadám 'REPLACE :JOSEF' (podruhé a dále už jen 'REPLACE') když budu chtít výskyt nahradit
  - nebo zadám 'FIND' když budu chtít tento výskyt ponechat beze změny

  což opakuji tak dlouho dokud neprojdu všechny výskyty řetězce 'KAREL'

Mazání zdrojového textu a nastavení prostoru pro něj:

CLEAR (SYMBOL SHIFT + X) - příkaz vymaže zdrojový text a vyčistí tabulku symbolů (ponechá v ní jen uzamčené symboly - viz kapitolu 4 Tabulka symbolů), protože se jedná o příkaz, jehož chybné použití může značně zvýšit hladinu adrenalinu v krvi uživatele, je nutno jej zadávat ve tvaru 'CLEAR y'

U-TOP (SYMBOL SHIFT + U) - uživatelská zarážka, toto je poslední adresa, kam smí být uložen zdrojový text a kam smí být provedena kompilace (kap. 3 Překlad). Za příkazem U-TOP musíte uvést číslo nebo matematický výraz (bude vyhodnocen zleva doprava, může obsahovat čísla i existující definovaná návěští (jinak chyba), tento příkaz lze použít i jako kalkulačku, výsledek je trvale vidět vpravo nahoře, k nastavení zarážky na adresu 65535 používejte příkaz 'U-TOP -1', pomocí uživatelské zarážky získáte chráněnou oblast paměti, kam můžete uložit třeba tiskové podprogramy a jiné, pro assembler i monitor je oblast od zarážky až do konce paměti nepřístupná

Chyby při editování:

Bad mnemonic chyba v zápisu mnemoniky, nebo návěští v poli mnemoniky

Bad operand	v instrukci použit chybný operand
Big number	použité číslo je větší než 65535
Syntax horror	syntaktická chyba, některá položka je delší než může být, dvě operace ve výrazu vedle sebe, ...
Bad string	špatně zapsaný řetězec, úvozovka se píše do řetězce dvojitě, řetězec nesmí být prázdný
Bad instruction	instrukce zapsána formálně správně, použitá instrukce však nemá možnost pracovat s touto kombinací operandů
Memory full	paměť přidělená pro zdrojový text je vyčerpána, můžete se pokusit zkrátit zdrojový text (viz kapitolu 6. Formát zdrojového textu)
XXXX unknown	chyba při použití návěští, které v tabulce symbolů není nebo tam je, ale nemá definovanou hodnotu - tato chyba je hlášena u příkazu U-TOP

2. magnetofonové operace

Assembler PROMETHEUS umožňuje (kdyby ne, bylo by to na považenou) ukládat, kontrolovat a číst zdrojové texty (i jejich části) z magnetofonu.

SAVE (SYMBOL SHIFT + S)

Příkaz určený pro uložení zdrojového textu na kazetu.

SAVE :karel - uloží celý zdrojový text (zdrojový text a tabulku symbolů) na kazetu, blok bude uložen jako BASICovské CODE pod jménem "karel"

SAVE - uloží celý zdrojový text na kazetu, jako jméno bude vzato naposledy použité jméno - toto se hodí jestliže při verifikaci došlo k chybě a je nutno opakovat opět ukládání

SAVE b:karel - uloží na kazetu část zdrojového textu (nastavený blok) a celou tabulku symbolů - vybírání návěští, která jsou ve zvoleném bloku použita, by bylo časově a paměťově náročné - proto chcete-li na kazetu uložit opravdu jen část zdrojového textu, musíte vymazat všechno ostatní a potom vyčistit tabulku symbolů (viz kapitola 4 tabulka symbolů)

SAVE b - uloží zvolenou část zdrojového textu se jménem naposledy použitým v příkazu SAVE nebo LOAD

LOAD (SYMBOL SHIFT + L)

Příkaz sloužící k přinášení zdrojového textu z kazety. Chcete-li nový zdrojový text pouze nahrát, musíte ten starý nejprve vymazat (příkazem "CLEAR y" - SS + X).

LOAD : - nahraje první nalezený zdrojový text z kazety do počítače

LOAD :karel - nahraje do počítače zdrojový text se

jménem 'karel'

LOAD - nahraje do počítače zdrojový text se jménem, které bylo naposledy použito v příkazech LOAD nebo SAVE

Použijí-li se místo jména samé mezery, bude to bráno jako kdyby nebylo zadáno žádné jméno a načte se první nalezený blok CODE.

Nahrávání probíhá následovně:

1. prohledává se kazeta dokud se nenajde blok CODE se správným jménem

2. po nalezení takového bloku se zjistí, zda se blok vejde do volné paměti, pokud ne, je hlášena chyba Memory full

3. blok se nahraje na konec paměti určené pro zdrojový text (adresa posledního bytu je napsána vpravo ve stavovém řádku)

4. zjistí-li se, že se nejedná o zdrojový text tohoto assembleru, příkaz je přerušen

5. začne se vkládání nahraného zdrojového textu do zdrojového textu, který je v paměti od minula, při tělo operaci se každý nový řádek převede na textový tvar a vloží do původního textu - vkládá se stejně jako při vkládání z klávesnice - za přístupný řádek, bude-li při vkládání nalezena chyba, objeví se chybové hlášení a je možno chybu opravit, po opravě program pokračuje ve vkládání, při přihrání bloku se do paměti musí vejít současně dvě tabulky symbolů, stará (ta se zvětšuje o nová návěští) a nová, ze které se berou návěští při převodu řádků na textový tvar - tyto podrobnosti Vás nemusí zajímat, pokud se překládá za zdrojový text (nejčastější situace), tak problémy s místem nastanou jen těžko

6. stav vkládání lze zobrazit stiskem libovolné klávesy

7. stiskne-li se při vkládání klávesa SPACE, vkládání se ukončí a v editačním řádku bude právě vkládaná instrukce

8. pokud dojde při vkládání k chybě a v editačním řádku je chybná instrukce, lze vkládání ukončit vložením nějakého příkazu nebo povelu (třeba přepnutím mezi desítkovou a šestnáctkovou soustavou)

9. vkládání bude chvíli trvat - po dobu vkládání je na obrazovce nápis 'Wait please'

VERIFY (SYMBOL SHIFT + V)

Příkaz sloužící ke kontrole na pásce uloženého textu, je ho nutné provést ihned po příkazu SAVE, nepotřebuje žádné parametry, použije jméno z příkazu SAVE a testuje blok pouze s tímto jménem

GENS (SYMBOL SHIFT + G)

Příkaz slouží k přihrání zdrojového textu ve formátu assembleru GENS (MACROS), syntaxe je úplně stejná jako u příkazu LOAD, zdrojový text nahraje a vkládá stejně jako u příkazu LOAD, před vkládáním zdrojového textu z GENSu je výhodné text nejprve upravit - zkrátit řádky

tak, aby odpovídaly formátu PROMETHEA, přemístil možné komentáře na zvláštní řádky ald. - je s tím méně práce než opravovat řádky až při vkládání

Je-li konec prostoru pro zdrojový text nastaven na adresu 65535 (#FFFF), nebude rozeznán konec souboru a vkládání se zastaví na chybě - dojde k pokusu vzít jako řádek zdrojového textu několik prvních bytů z ROM, přeruště vkládání vložením nějakého povelu - viz bod 8. Chcete-li se tomu vyhnout, nastavte konec paměti pro zdrojový text třeba na adresu 65500 (příkaz U-TOP).

Pokud dojde při kazetových operacích ke stisku klávesy BREAK (SPACE), v editačním řádku zůstane naposledy zadaný povel. Pokud dojde v příkazu LOAD, CENS nebo VERIFY k chybě, vypíše se chybové hlášení 'Tape error' a opět je v editačním řádku posledně zadaný příkaz.

Pokud budete chtít do assembleru vložit zdrojový text z jiného assembleru, musíte ho převést na formát assembleru CENS, každý řádek začíná dvěma byty - číslo řádku (pro PROMETHEUS není podstatné), potom text řádku (stačí vždy jedna mezera místo několika) a na konci řádku je kód 13, takto upravený zdrojový text vyhrajte z paměti jako CODE a můžete ho příkazem CENS dostat do PROMETHEA. Další použitelný způsob (možná jednodušší) je popsán v kapitole 5. Ostatní (viz CENSOR).

3. překlad

ASSEMBLY (SYMBOL SHIFT + A) - příkaz provede kompilaci zdrojového textu. Za příkazem ASSEMBLY je možno uvést parametr 'D' (nebo 'd') - v tomto případě bude kompilován pouze nastavený blok zdrojového textu.

Při kompilaci neexistuje kompilační výpis - tato možnost byla pro jednoduchost a zvýšení rychlosti odstraněna, upřímně řečeno autorovi (a nejen jemu) se zdála zbytečná. Pokud se chcete dozvědět, na jaké adrese je která instrukce (to byla jediná námitka proti odstranění kompilačního výpisu), napište tam nějaké návěští (pokud už tam není) a po skončení kompilace se podívejte do tabulky symbolů na jeho hodnotu (viz kapitola 4. tabulka symbolů), lze také samozřejmě použít disassembler monitoru. Potřebujete-li z nějakých důvodů (?) protokol o překladu, přeneste zdrojový text do nějakého jiného assembleru (CENS pomocí programu CENSOR - viz kapitolu 5. ostatní - CENSOR) a s jeho pomocí ho získáte.

Kompilace je prováděna rychlostí přibližně 2 KB strojového kódu za sekundu - tedy ani dlouhý zdrojový text není překládán déle než 3-4 sekundy, u krátkých zdrojových textů je kompilace takřka okamžitá.

Kompilace je dvouprůchodová; při prvním průchodu se pouze zjišťují hodnoty návěští v poli návěští (nikoliv u příkazu EQU), při druhém průchodu se již generuje strojový kód, nyní musí být známa všechna návěští. Ukládání strojového kódu do paměti je kontrolováno tak, aby nebyl přepsán ladící systém, zdrojový text a oblast paměti nad uživatelskou zarážkou.

Možná chybová hlášení:

Bad PUT (ORG) - při druhém průchodu kompilace došlo k pokusu o přepsání ladícího systému, zdrojového textu nebo oblasti nad uživatelskou zarážkou strojovým kódem

XXXXX unknown - použité návěští 'XXXXX' nemá dosud přiřazenu žádnou hodnotu

Already defined - pokus o druhé definování hodnoty návěští

Big number - pokus o zapsání velkého čísla do 8 bitů, u příkazů relativních skoků to znamená, že vzdálenost je příliš velká, u instrukcí typu (ix+d) smí být d v rozsahu -128 až 127, příkazy typu ld a,N smí být N v rozsahu -256 až 255 (podrobněji dále)

Dojde-li k chybě při kompilaci, překlad se zastaví a obnoví se výpis tak, že řádek, na kterém došlo k chybě je nyní přístupný (jedná-li se o chybu Already defined tak ono podruhé definované návěští je samozřejmě v poli návěští, nikoliv v poli operandů). Pokud kompilace skončí úspěšně, vypíše se hlášení 'Assembly complete'.

RUN (SYMBOL SHIFT + R) - příkaz slouží ke spuštění programu od místa zadaného příkazem ENT (Entry point - viz dále), před spuštěním programu se vždy provede kompilace - příkaz zahrnuje příkaz ASSEMBLY (překlad je rychlý a není třeba zadávat dva příkazy), pokud nebude nalezen právě jeden příkaz ENT, vypíše se chybové hlášení 'ENT ?'. Také tento příkaz může obsahovat parametr 'b' - před spuštěním se přeloží pouze zvolený blok zdrojového textu. Váš program bude spuštěn se zakázaným přerušením a proto jej musíte případně povolit!

Instrukce:

PROMETHEUS překládá všechny základní instrukce (známé) a také většinu tzv. 'neznámých' instrukcí - jediné 'neznámé' instrukce, které assembler nezná jsou všechny rotace na adrese (ix+d) s následným přenosem do zvoleného registru, pokud budete chtít tyto instrukce použít, musíte je zapsat pomocí pseudoinstrukce DEFB (instrukce nebyly zařazeny proto že se v praxi nepoužívají a zvětšily by délku assembleru). Následuje seznam těch 'neznámých' instrukcí, které assembler používá:

Instrukce, které používají poloviny indexregistrů - IX,IY
Dolní polovina registru IX je značena LX, horní polovina bude označena HX, obdobně pro registr IY je LY a HY.

inc	inc	hx	inc	lx	inc	hy	inc	ly
dec	dec	hx	dec	lx	dec	hy	dec	ly
ld	ld	hx,N	ld	lx,N	ld	hy,N	ld	ly,N
	ld	b,hx	ld	b,lx	ld	b,hy	ld	b,ly
	ld	c,hx	ld	c,lx	ld	c,hy	ld	c,ly
	ld	d,hx	ld	d,lx	ld	d,hy	ld	d,ly
	ld	e,hx	ld	e,lx	ld	e,hy	ld	e,ly
	ld	hx,hx	ld	hx,lx	ld	hy,hy	ld	hy,ly
	ld	lx,hx	ld	lx,lx	ld	ly,hy	ld	ly,ly
	ld	a,hx	ld	a,lx	ld	a,hy	ld	a,ly
add	add	a,hx	add	a,lx	add	a,hy	add	a,ly
adc	adc	a,hx	adc	a,lx	adc	a,hy	adc	a,ly
sub	sub	hx	sub	lx	sub	hy	sub	ly
sbc	sbc	a,hx	sbc	a,lx	sbc	a,hy	sbc	a,ly
and	and	hx	and	lx	and	hy	and	ly
xor	xor	hx	xor	lx	xor	hy	xor	ly
or	or	hx	or	lx	or	hy	or	ly
cp	cp	hx	cp	lx	cp	hy	cp	ly

Rotace SLIA (shift left inverted arithmetic - invertovaný aritmetický posun doleva) - posune doleva zvolený registr, vystupující bit je uložen do CARRY, vstupující bit je jednička.

slia	slia b	slia c	slia d	slia e
	slia h	slia l	slia (hl)	slia a

U instrukcí relativních skoků a instrukce DJNZ se jako u ostatních assemblerů píše absolutní adresa. Při kompilaci se vypočte relativní adresa a uloží se, pokud je relativní adresa mimo rozsah -128 až 127, ohlásí se chyba 'Big number'.

Assembler používá některé pseudoinstrukce - slouží buď k zápisu čísel a textů do paměti nebo k ovládání ukazatele kompilace, umístění strojového kódu a k nastavování startu pro příkaz RUN a hodnot návěští - následuje seznam:

ORG výraz - nastaví čítač adres a ukazatel uložení do paměti na hodnotu výrazu. Ukazatel uložení do paměti určuje, kam se bude ukládat strojový kód, čítač adres určuje kde bude přeložený program pracovat - hodnota tohoto čítače se používá při dosazování hodnot návěští při prvním průchodu. Ukazatel uložení do paměti i čítač instrukcí se vždy zvětšují o délku právě přeložené instrukce.

PUT výraz - nastaví ukazatel uložení do paměti na hodnotu výrazu. Tento příkaz se používá v případě, že je potřeba uložit strojový kód jinam než bude spouštěn. Chcete třeba přeložit program tak, aby pracoval v obrazovkové paměti lady, od adresy 16384, protože překládat přímo do obrazovky není možné, vložte za příkaz ORG 16384 ještě příkaz PUT 60000 - program se bude překládat tak, aby pracoval od adresy 16384 ale ukládat se bude od adresy 60000. Nelze přehodit pořadí, ve kterém jsou obě pseudoinstrukce uvedeny, protože příkaz ORG nastavuje také ukazatel uložení do paměti! Příkaz PUT byl zaveden jako alternativa k možnosti ukládat přeložený program za tabulku symbolů bez ohledu na pseudoinstrukci ORG, kterou poskytuje assembler GENS (MASM), toto řešení je však podstatně přizpůsobivější, programátor se nemusí starat, kde tabulka symbolů končí a může si sám nastavit, kam se bude překlad ukládat. Chcete-li vyrobit program, který se po spuštění sám rozmístí po paměti, můžete to vyrobit třeba takto:

```
          org 60000

START     ld   hl,START1    ;adresa, kde je 1. blok uložen
          ld   de,30000     ;adresa, kde má 1. blok pracovat
          ld   bc,BLOK1LEN  ;délka 1. bloku
          ldir              ;přesuň 1. blok
          ld   de,35000     ;adresa, kde má 2. blok pracovat
          ld   bc,BLOK2LEN  ;délka 2. bloku
          ldir              ;přesuň 2. bloku
          jp   30000        ;skok na začátek 1. bloku
END1
          org  30000        ;první blok bude na adrese 30000
          put  END1         ;jeho kód se ukládá stále dále
START1    ???               ;zde je text prvního bloku
          ???
BLOK1LEN  equ  $-START1     ;BLOK1LEN bude obsahovat délku
END2
          org  35000        ;druhý blok bude na adrese 35000
          put  END2         ;kód se ukládá stále dále
START2    ???               ;zde je text druhého bloku
          ???
BLOK2LEN  equ  $-START      ;BLOK2LEN bude obsahovat délku
```

návěští EQU výraz - přiřadí použitému návěští hodnotu výrazu. V okamžiku, kdy dochází při druhém průchodu ke zpracování této instrukce už musí být známé hodnoty všech

návěští, která jsou použita ve výrazu, jinak dojde k chybě.

```
LABEL1   equ  LABEL2+1   ;takhle to bude fungovat, protože
LABEL2                   ;návěští LABEL2 bylo definováno
                         ;už při prvním průchodu překladu

LABEL1   equ  LABEL2+1   ;takhle to bohužel fungovat nebude
LABEL2   equ  32000      ;protože návěští LABEL2 ještě nemá
                         ;definovanou hodnotu
```

ENT výraz - tento příkaz slouží k nastavení adresy, od které bude spuštěn strojový kód v případě, že bude použit příkaz RUN.

```
         ent  START      ;program bude spuštěn od místa, na
                         ;němž se nachází návěští START
                         ;v poli návěští

         ent  $          ;program bude spuštěn od místa, na
                         ;kterém se vyskytuje tento příkaz
```

DEFB výraz,výraz,...,výraz - uloží od adresy, na kterou ukazuje ukazatel uložení do paměti, výsledky výrazů jako osmibitové hodnoty - musí ležet v rozmezí -256 až 255, jinak je hlášena chyba 'Big number'. Potom jsou zvýšeny hodnoty čítače adres a ukazatele uložení do paměti o počet bytů, které byly příkazem DEFB naplněny - počet výrazů.

DEFW výraz,výraz,...,výraz - podobné jako příkaz DEFB, ukládají se šestnáctibitové hodnoty (napřed nižší byte a potom vyšší byte). Ukazatel uložení do paměti a čítač adres se zvětší o počet bytů, které byly naplněny - počet výrazů krát dva.

DEFM řetězec - uložení řetězce do paměti, tento příkaz byl u assembleru rozšířen:

```
         defm 'karel'    ;uloží do paměti postupně ASCII
                         ;kódy jednotlivých písmen

         defm 'karel'    ;provede téměř totéž, poslední
                         ;znak bude mít ASCII hodnotu větší
                         ;o 128 - invertovaný znak, pokud
                         ;natušíte, k čemu je to dobré,
                         ;podívejte se na příklady programů
```

DEFS výraz - vyhodnotí se výraz a o jeho hodnotu se zvětší ukazatel uložení do paměti a čítač adres. Tato pseudoinstrukce se používá k vynechání místa o požadované délce.

```
BUFFER   defs 100        ;vynechání 100 bytů
BUFFER2  defs 1000       ;vynechání 1000 bytů
```

Výrazy:

Assembler umožňuje vložit na místo, kde je v instrukci číselná hodnota, matematický výraz. Matematický výraz se skládá z jednoho nebo několika návěští či konstant oddělených od sebe operátory. Všechna čísla jsou převedena na rozsah 0-65535 (záporná čísla jsou přičtena k číslu 65536 a tím jsou převedena na rozsah 0-65535), v tomto tvaru jsou provedeny operace a výsledek je použit podle potřeb instrukce. Výpočet je prováděn zleva doprava, není zachovávána priorita operátorů. Pokud není možno výraz zapsat tak, aby ho bylo možno provádět zleva doprava, musíte ho rozepsat na dva řádky a mezivýsledek přiřadit nějakému návěští a to použít na dalším řádku, autor se s touto eventualitou ještě nesetkal - nejčastější výrazy používají pouze jeden operátor.

Konstanty:

Jsou povoleny tyto konstanty:

desítkové:	1234	47	-32
šestnáctkové:	#ABCD	-#30	#a3
dvojkové:	%10111	%111	-%011010
znakové:	'a'	'aB'	''''

úvozovka se musí vložit dvojitě

Je možnost odkázat se na aktuální hodnotu čítače adres:

$ - na toto místo bude dosazena hodnota čítače adres

```
START   ????        ;zde je část textu, jejíž
        ????        ;délku potřebujeme zjistit
        ????        ;a vložit do návěští LENGTH
LENGTH  equ $-START ;toto je na konci oné části
```

Návěští:

Návěští je posloupnost čísel, písmen a znaku '_', která začíná písmenem. Znaků lze použít maximálně 8 a pro identifikaci jsou důležité všechny.

Operátory:
---

sčítání	+
odčítání	-
násobení	*
dělení	/
zbytek po dělení	?

Chcete-li získat dolní byte návěští TAB, vložte TAB?256, horní byte získáte pomocí TAB/256.

Důležité upozornění: při dělení nulou není hlášena chyba a výsledek je 65535.

4. Tabulka symbolů

Možnosti, které poskytuje PROMETHEUS s tabulkou symbolů jsou jedním ze znaků, které ho výrazně odlišují od ostatních assemblerů.

Tabulka symbolů existuje neustále a je vždy možné prohlédnout si abecedně seřazený výpis návěští a jejich hodnot, pokud jim byly nějaké přiřazeny. Neustálá existence tabulky symbolů se může zdát nevýhodná - je to však vlastnost, která umožňuje výrazná zkrácení zdrojového textu a také zrychlení kompilace.

Příkazy pro práci s tabulkou:

TABLE (SYMBOL SHIFT + T) - tento příkaz vypíše tabulku symbolů na obrazovku, zobrazí se jedna stránka (40 návěští) a očekává se stisk nějaké klávesy, pokud to bude klávesa SPACE, vypisování tabulky se skončí, pokud to klávesa SPACE nebude, vypíše se další stránka. Návěští jsou vypsána ve dvou sloupcích, před návěštím je vypsána buď mezera nebo hvězdička (její význam je popsán dále), za návěštím je vypsána hodnota, která byla

návěští přiřazena při posledním překladu, pokud mu žádná hodnota přiřazena nebyla, je vypsáno pět teček. Hodnota návěští je vypsána v nastavené číselné soustavě (desítkové či šestnáctkové viz kapitolu 1. Editor). Možnost přečíst si tabulku symbolů lze použít, zajímají-li Vás adresy některých částí programu nebo když si nejste jisti (u dlouhých textů se to občas stane), která návěští jste už použili a která ne.

TABLE p - provádí výpis tabulky symbolů na obrazovku a také na tiskárnu, pro podrobnosti se podívejte na příkaz PRINT, na každý řádek se vytiskne jedno návěští a jeho hodnota, jedna stránka obrazovky se tedy vypíše na 40 řádků papíru.

TABLE c - vyčištění tabulky symbolů, z tabulky symbolů se odstraní ta návěští, která se nevyskytují ve zdrojovém textu a nejsou uzamčena. Při vymazání řádku nebo bloku zdrojového textu se z tabulky symbolů neodstraní ta návěští, která se vyskytovala pouze ve vymazané části zdrojového textu - tato akce může být časově náročnější a nevyplatí se ji provádět neustále protože by mohla zdržovat, také několik bytů za to nestojí. Zbytečná návěští jsou vidět když si po provedení kompilace vypíšete tabulku symbolů - nemají definovanou hodnotu - pokud jejich počet vzroste, tak použijte příkaz pro vyčištění tabulky symbolů. Pokud nemáte problémy s volnou pamětí (pak je každý byte dobrý) můžete na tento příkaz zapomenout, při každém přečtení zdrojového textu z kazety (tuto akci je nutno provádět obvykle několikrát za den - při ladění programů ve strojovém kódu není nouze o stavy, kdy se zkoušený program odmítá vrátit zpět, případně se rovnou po spuštění vymaže) se ve zdrojovém textu zachovají jen použitá návěští.

Oddělená kompilace - kompilace po částech:

TABLE] - uzamkne všechna návěští v tabulce symbolů, tato návěští se při kompilaci chovají tak, jako by byla stále definována, příkaz TABLE c tato návěští neodstraní. Popsání smyslu tohoto příkazu následuje.

TABLE u - odemkne všechna zamčená návěští v tabulce symbolů, opak příkazu TABLE].

Dva předchozí příkazy jsou určeny pro umožnění oddělené kompilace - důvodem může být velký rozsah zdrojového textu. Zdrojový text je nutno rozdělit na jakýsi základ (pomocné podprogramy - tiskové rutiny, práce s obrazovkou, data,..) a na hlavní část (toto rozdělení může být i několikavrstevné), důležité je, aby se základ (nižší úroveň) neodkazovala na návěští v hlavní části (vyšší úroveň). Nyní se odladí základ, zamkne tabulka symbolů a vymaže zdrojový text. Po vymazání zdrojového textu zůstanou v paměti všechna návěští, která obsahoval a je možno se na ně odvolávat v dalším zdrojovém textu. Tuto možnost Vám snad lépe osvětlí následující příklad:

V příkladu je program rozdělen na tři vrstvy, v první vrstvě jsou uloženy texty, druhá část obsahuje program, který vytiskne text, jehož adresa mu bude zadána a poslední (nejvyšší) vrstva bude volat tisk textu.

```
            org  65000
TEXT1       defm 'Zkusebni '     ;toto je prvni cast programu,
            defm 'text c.1'      ;neni v ni zadny odkaz na
TEXT2       defm 'Pokusny '      ;navesti ve vyssi vrstve
            defm 'text c.2'
```

Tuto část programu zkompilujte (ASSEMBLY), zamkněte tabulku symbolů (TABLE]) a vymažte zdrojový text (CLEAR y).

```
            org  64000
PRINT       ld   a,(hl)          ;toto je druha cast programu,
            and  127             ;odkaz na nizsi uroven v ni
```

```
        rst  16              ;sice není, mohl by však být,
        ld   a,(hl)          ;odkaz do vyšší úrovně se
        inc  hl              ;zde vyskytovat nesmí
        and  128
        jr   z,PRINT
        ret
```

Nyní opět zkompilujte zdrojový text (ASSEMBLY), uzamkněte opět tabulku symbolů (TABLE), nyní jsou v ní již tři návěští, a opět vymažte zdrojový text (CLEAR y).

```
        org 63000
MAIN    ld   a,2             ;poslední a nejvyšší úroveň
        call #1601           ;programu, může se odkazovat
        ld   hl,TEXT1        ;i na návěští definovaná v
        call PRINT           ;obou nižších úrovních
        ret
```

Po vložení poslední části zdrojového textu jsou v tabulce symbolů návěští TEXT1, TEXT2, PRINT a MAIN. První tři návěští jsou uzamčena.

Je zřejmé, že v uvedeném příkladě nemá rozdělení tohoto krátkého zdrojového textu na kusy smysl, ilustruje však způsob, jakým lze zpracovávat i dlouhé zdrojové texty.

Tento způsob práce byl použit při ladění monitoru, z kompilace zdrojového textu byla zachována tabulka symbolů a přeložený kód, do paměti byl vkládán zdrojový text monitoru a laděn - tímto způsobem bylo možno odladit rozumným způsobem program dlouhý 11 KB, odpovídající zdrojový text by byl dlouhý celkem asi 40 KB a nevešel by se za žádných okolností spolu s assemblerem do paměti, o potřebě místa pro přeložený kód ani nemluvě.

Při kompilaci jednotlivých vrstev je vhodné nižší vrstvy chránit tím, že se uživatelská zarážka nastaví pod ně - nehrozí nebezpečí, že již hotové části budou přepsány zdrojovým textem nebo generovaným strojovým kódem.

Při každém zahájení práce je nutno opět zkompilovat všechny vrstvy, aby se do tabulky symbolů dostala potřebná návěští, a vyplatí se tedy spíše vyhrát na kazetu celou paměť, tedy ladící systém i se zdrojovým textem a již přeloženými vrstvami programu. Může se to zdát, že se nahrává zbytečně velký rozsah paměti, ale z časových důvodů je to lepší, než při zapnutí počítače provádět některé akce vždy znovu, navíc tento postup umožňuje mít v paměti i další programy a data a nemuset je pokaždé nahrávat z deseti různých kazet po dlouhém hledání a převíjení kazet z jednoho konce na druhý, takto popsaný způsob jsem si nevycucal z palce ale sám jsem ho používal při práci na monitoru k tomuto ladícímu systému - assembler už byl plně funkční v této době. Můžete použít následující modifikaci tohoto způsobu:

1. odladíte si základní vrstvu

2. pro ladění další vrstvy si vyrobíte jakousi instalaci, ta bude obsahovat krátký zaváděcí BASIC program a dva bloky CODE - první bude tvořit ladící systém spolu s tabulkou symbolů vzniklou odladěním základní vrstvy, druhý pak přeložený kód a případně nějaká data nebo cokoliv jiného - rozdělení je proto, že mezi ladícím systémem a přeloženým kódem je obvykle velká mezera a nemá smysl nahrávat několik KB nul, to ovšem neplatí když překládáte před ladící systém a ne za něj

   Zaváděcí BASIC může vypadat třeba takto:

```
    1 RANDOMIZE USR 24e3:STOP
 9999 CLEAR 23999:LOAD ""CODE:LOAD ""CODE:RUN
```

   Odpovídající bloky dostanete na kazetu pomocí příkazů:

```
SAVE "Instalace" LINE 9999:
SAVE "Code 1" CODE 24e3,konec-23999:
SAVE "Code 2" CODE začátek,délka
```

'konec' je konec zdrojového textu (první číslo ve slavové řádce, začátek a délka jsou čísla určující polohu a rozsah přeložené nižší vrstvy a případně dat, v tomto příkladu je assembler umístěn na adrese 24000.

3. v instalaci si můžete také nastavit monitor podle svých požadavků a nemusíte jej pak pokaždé nastavovat znovu

4. po odladění další vrstvy si vyrobíte opět novou další instalaci - a takto pořád dokola dokud není program celý hotov

Můžete samozřejmě také základní vrstvu přeložit a z dalších vrstev se odkazovat už přímo na adresy, to má nejméně dvě nevýhody - při používání monitoru se nebudete moci odkazovat na návěští v již hotové vrstvě (nebudou v tabulce) a v případě, že se rozhodnete pro změnu některé části v nižší vrstvě, budete muset přepsat každý odkaz na adresy ve vyšší vrstvě protože se tyto adresy posunou, použijete-li výše popsaný postup, stačí vyrobit si novou instalaci s novými hodnotami v tabulce symbolů a vyšší vrstvu nemusíte měnit (pokud ovšem nezměníte nějak více základní vrstvu - jiná návěští, jiný smysl).

Nutnost použití oddělené kompilace se však díky značnému zkrácení zdrojového textu (proti GENS je poloviční) odsouvá až na hranici kolem 5-7 KB generovaného strojového kódu, kratší strojový kód lze ladit najednou - jaké to má výhody snad není potřeba moc zdůrazňovat.

5. Ostatní příkazy a možnosti:

Doposud nebyly uvedeny některé příkazy, které lze v assembleru používat, zde je jejich přehled.

PRINT (SYMBOL SHIFT + P) - tento příkaz slouží k vytištění zdrojového textu na tiskárnu nebo plotter. Program bere jednotlivé řádky a posílá je znak za znakem do kanálu systému Spectra #3 - kanál pro tiskárnu. Každá řádek je ukončen kódem 13 (ENTER) a nejsou v něm použity žádné tabelátory - mezery jsou opravdu tištěny všechny. Pokud v tomto příkazu použijete parametr 'b', bude vytištěn pouze nastavený blok. Chcete-li si vytisknout tabulku symbolů, použijte příkaz 'TABLE p'. Provádění lze zastavit stiskem klávesy SPACE. Jakým způsobem si připojíte svou tiskárnu na kanál #3 je vaše věc - stačí, aby byl k dispozici program, který 'umí' LPRINT.

MONITOR (SYMBOL SHIFT + M) - příkaz slouží k přechodu do monitoru (viz dále). Použijete-li parametr 'a', provede se před spuštěním monitoru kompilace zdrojového textu - je to výhodnější než zadávat příkazy ASSEMBLY a MONITOR postupně. Pokud jste si při instalaci monitor odpojili, spuštění se neprovede.

BASIC (SYMBOL SHIFT + B) - vrátí řízení do systému Spectra, do BASICu. Pokud se Vám však podařilo přepsat některé systémové proměnné Spectra (program pracující s obrazovkou Vám 'utekl' a prošel přes tuto oblast), použijte pro návrat do BASIC raději příkaz NEW. Pro nové spuštění PROMETHEA použijte adresu, na které je instalován - při návratu je assembler ve stejném stavu v němž jste ho opustili (pokud jste ho ovšem z BASIC nějak 'nevylepšili'). Při návratu se nastaví potřebné hodnoty do registrů IY a SP, zapne přerušení v módu IM 1 a provede se skok do hlavní provádějící smyčky (bude se interpretovat další příkaz), při tomto návratu není potřeba nastavovat registr HL'!

NEW (SYMBOL SHIFT + N) - příkaz provede stejnou akci jako v BASIC příkaz NEW - vymaže celou paměť až po nastavený RAMTOP. Do assembleru byl zařazen proto že při trasování nebo při běhu se může podařit přepsat systémové proměnné a hrozí, že se při návratu pomocí příkazu BASIC systém zhroutí. Pokud nevíte, zda se nepřepsala také systémová proměnná RAMTOP, podívejte se na její hodnotu (je to hodnota na adresách 23730 a 23731) a případně ji opravte předtím, než použijete příkaz NEW (toto lze provést buď v monitoru nebo tak, že vyrobíte krátký program, který to provede a ten spustíte příkazem RUN:

```
org  23296
ent  $
ld   hl,23999
ld   (23730),hl
ret
```

Program přeložíte a spustíte příkazem 'RUN b' - předtím ho nastavíte jako blok - potom můžete bez obav provést příkaz NEW. Pokud používáte rutiny z ROM Spectra, které přepisují systémové proměnné, je někdy lepší používat pro návrat do systému vždy příkaz NEW - po skončení Vašeho programu totiž nemusí být v tom stavu, v jakém by být měly.

QUIT (SYMBOL SHIFT + Q) - pokud bude tento příkaz použit tak zaručeně jako poslední, k použití jiných už nebudete mít možnost - provádí RESET (RANDOMIZE USR 0). Tímto příkazem pouze elegantně ukončíte práci a zcela vyčistíte počítač. Protože tento příkaz je mimořádně nebezpečný a jeho nechtěné použití by mohlo vyvolat u uživatele silnou depresi (už se stalo), je nutno za něj přidat parametr 'y', pouze takto se provede.

CENSOR:

Způsob, jak dostat zdrojový text z programu GENS do tohoto assembleru byl již zmíněn, může nastat také opačná potřeba, k tomuto účelu byl vyroben krátký program CENSOR.

Program CENSOR se skládá z krátké části v BASICu a přibližně z 200 bytů strojového kódu. Ke své činnosti potřebuje, aby byla oblast paměti od adresy 55000 až ke konci prázdná - sem se ukládá zdrojový text ve formátu assembleru GENS (MASM). Nahrajte do paměti PROMETHEUS (můžete odpojit monitor aby bylo více místa) a instalujte ho třeba na adresu 25000. Nahrajte do assembleru zdrojový text - pokud by přesahoval adresu 55000, budete ho muset rozdělit. Jestliže je adresa konce zdrojového textu menší než 55000 můžete nahrát program CENSOR příkazem LOAD "CENSOR". Po nahrání přepíšete v příkazu RANDOMIZE USR adr, proměnnou 'adr' skutečným startem assembleru (nyní 25e3). Nyní můžete zadat příkaz GO TO 0 (ne RUN!), provede se inicializace a program CENSOR se připojí na tiskový kanál #3, potom se provede spuštění assembleru. Nastavte začátek bloku na začátek zdrojového textu a konec dejte o maximálně 30 stránek dál, nyní zadejte příkaz 'PRINT b' - zadaný blok půjde místo na tiskárnu do programu CENSOR, který ho ukládá za sebe - omezení na 30 stránek je dáno velikostí paměti, do které se může zdrojový text ukládat (kdyby byly jednotlivé řádky zdrojového textu delší než je obvyklé - data nebo texty vyplňující celý řádek - vložte raději konec bloku dříve než po 30 stránkách). Zadejte příkaz BASIC a nahrajte právě vytvořenou část zdrojového textu na kazetu, po návratu do assembleru nastavte další část zdrojového textu a opakujte dokud celý zdrojový text není nahrán na kazetě.

Tento způsob přenesení zdrojového textu do programu GENS můžete použít i u jiných assemblerů, máte-li zdrojový text ve formátu pro GENS, můžete ho snadno přenést do PROMETHEA.

6. formát zdrojového textu a další podrobnosti o assembleru:

Tato kapitola je určena všem, kteří by se rádi dozvěděli jak vypadá zdrojový text, jakým způsobem je možno ho ještě zkrátit (v mezích možností samozřejmě) a proč vlastně dosahuje assembler v některých ohledech takové výkony (rychlost a délka či spíše úspornost uložení zdrojového textu) proti jiným assemblerům.

Na počátku byly tyto myšlenky:

- k tomu, aby se dala instrukce zcela jednoznačně rozlišit stačí její operační kód, informace o prefixech a tvaru případného číselného operandu, tedy informace, které lze zapsat do dvou bytů, toto řešení výrazně urychlí překlad, bude stačit jen brát operační kódy, přidávat prefixy, nebo vyhodnocovat výrazy a dosazovat jejich hodnoty do místa pro číselné operandy - nebude potřeba zjišlovat operační kódy a hlídat syntaxi příkazů při kompilaci - tyto akce se provádějí už při editování (v této době čeká počítač na člověka a má dost času aby tyto akce provedl) - vedlejším efektem pak je, že už při psaní zdrojového textu dojde k vychytání spousty chyb a překlepů

- návěští ve zdrojovém textu mohou být uložena jen jako odkazy do tabulky symbolů, tyto odkazy zaberou jen dva byty, u návěští dlouhých alespoň tři znaky vzniká úspora - tabulka symbolů se bude vytvářet již při psaní zdrojového textu, nebude tedy potřeba vytvářet ji až při překladu, také každý odkaz na návěští bude nasměrován při editaci a nebude potřeba při překladu tabulku prohledávat

Nyní šlo o to, jak to provést s nejmenším možným nárokem na čas - třeba jedna stránka výpisu vlastně znamená jakési disassemblování 20 instrukcí - o rychlosti, s jakou se to provádí se můžete přesvědčit sami (provádí se po každém odeslání instrukce).

Každý řádek zdrojového textu má tvar:

- první byte je operační kód instrukce, jsou zde ale také pseudoinstrukce a další 'neinstrukce' jako je prázdný řádek a komentář, ty mají také přiřazeny své kódy, aby se odlišily od obyčejných instrukcí, mají v druhém bytu řádku (informační byte) takovou kombinaci prefixů, kterou nemá žádná instrukce, je kombinace prefixů #DD a #FD, a v posledních třech bitech je uloženo číslo 7 (mimo prázdný řádek, ten tu má uloženu 0), hodnota informačního bytu závisí také na tom, jestli je nebo není před instrukcí či neinstrukcí návěští nebo ne - pro signalizaci této skutečnosti je určen 3 bit informačního bytu, když je návěští použito, je zde zapsána 1, odpovídající hodnota je zapsána v závorce, následuje přehled kódů 'neinstrukcí:

neinstrukce	operační kód	informační byte
prázdný řádek	0	48 (56)
komentář	1	55
ent	2	55 (63)
equ	3	(63)
org	4	55 (63)
put	5	55 (63)
defb	6	55 (63)
defm	7	55 (63)

defs	8	55 (63)
defw	9	55 (63)

- druhý byte je informační byte, obsahuje tyto údaje:

  7 bit - #CB prefix (203), 0 ne, 1 ano
  6 bit - #ED prefix (237)
  5 bit - #DD prefix (221)
  4 bit - #FD prefix (253)
  3 bit - v poli operandů je návěští
  2-0 bit - typ číselného výrazu - podle tohoto se dosazuje výsledek výrazu, význam hodnoty je následující:

    0 - instrukce nemá číselný operand
    1 - jeden byte, rozsah -256 až 255
    2 - dva byty, mezi -65536 až 65535
    3 - jeden byte, rozsah -128 až 128
    4 - jeden byte, typu (ix+d)
    5 - typ (ix+d),n
    6 - rst p (p se vloží do op. kódu)
    7 - neinstrukce

  uvedené rozsahy je možno používat v textu

- další byty jsou použity jen v případě, že je 3 bit informačního bytu nastaven na 1 nebo když poslední tři bity informačního bytu neobsahují nuly, tvar je následující:

  1. bit 3 je 1, bity 0 až 2 jsou všechny 0

     potom v dalších dvou bytech je zapsáno pořadí daného návěští v tabulce a na konci je číslo 192+2 (délka)

  2. bit 3 je 0, bity 0 až 2 nejsou jenom 0

     v dalších bytech je zapsán celý výraz návěští jsou nahrazena svým pořadím v tabulce symbolů, ostatní znaky jsou zapsány obvyklým způsobem, na konci je číslo 192+délka výrazu

     například výraz 2*LABEL+#23

     je zapsán takto:

     '2','*',128+H,L,'+','#','2','3',192+8

     kde H a L jsou horní a dolní byte pořadí návěští LABEL v tabulce symbolů

     výraz (2*LABEL+#23) bude uložen také takto, jestli se jedná nebo nejedná o adresu se zjistí z operačního kódu

  3. bit 3 je 1, bity 0 až 2 nejsou jenom 0

     uložení bude stejné jako v předchozím případě, před výrazem budou dva byty obsahovat pořadí návěští a délka bude o dva byty větší

Možnosti zkrácení zdrojového textu:

Dostane-li se programátor do situace, že potřebuje několik desítek bytů paměti a tyto nejsou k dispozici, pokusí se zkrátit zdrojový text tak, aby byl při stejném obsahu kratší. Několik způsobů, které vedou k výsledku:

1. Vyčistěte tabulku symbolů od zbytečných návěští příkazem 'TABLE c'.

2. Odstraňte komentáře a prázdné řádky

3. Pokud se ve zdrojovém textu vyskytují příkazy DEFB nebo DEFW za sebou, snažte se dostat co nejvíce výrazů do jednoho příkazu (bude méně operačních kódů a informačních bytů) - např:

```
            defb 255
přepisem    defb 0      na  .defb 255,0,32   ušetříte 4 byty
            defb 32
```

4. Zvykněte si pokud používáte texty s posledním invertovaným znakem používat rozšíření příkazu DEFM - sekvenci

```
            defm "Invertovaný text"   ;v CENSu to jinak
            defb "t"+128              ;nelze udělat
```

lze zapsat také takto: defm 'Invertovaný text', je to přehlednější a kratší (zvyk bývá železná košile).

5. U konstant použijte desítkovou soustavu, stejné číslo lze zapsat nejkratším způsobem, je to sice méně přehledné, ale dá se s takto ušetřit mnoho - hlavně u binárního zápisu, počet znaků potřebný na zápis stejného čísla je různý, třeba kód mezery je možno zapsat:

```
'32'        v desítkové soustavě
'#20'       v šestnáctkové soustavě
' '         jako znak
'%100000'   ve dvojkové soustavě
```

zápis v desítkové soustavě je viditelně ze všech nejkratší. Také je vhodné místo 2+3 psát rovnou 5. Někdy je vhodné psát čísla v záporném tvaru, třeba číslo 65535 lze zapsat jako -1, totéž platí pro číslo 255, pouze však je-li použito v instrukci s osmibitovou konstantou.

6. Zkraťte návěští, o kolik znaků zkrátíte návěští, o tolik se zkrátí tabulka symbolů a tedy i celý zdrojový text, vyplatí se ovšem zkracovat větší množství návěští najednou, používáte-li třeba návěští LOOP s číslem, pak je vhodné nahradit stejnou část kratší sekvencí - třeba LP - zde si však dejte dobrý pozor abyste prohozením nezpůsobili, že dříve různá návěští budou nyní stejná, po prohození návěští samozřejmě musíte vyčistit tabulku symbolů.

7. Vyskytuje-li se ve zdrojovém textu některá konstanta velmi často a je-li dlouhá alespoň 3 znaky, je vhodné její hodnotu přiřadit některému návěští a nahradit ji všude tímto návěštím - tento postup není dobrý jen pro zkrácení ale i pro zpřehlednění a budete-li chtít konstantu změnit, stačí ji změnit na jednom místě a ne všude! Je-li tato konstanta použita v příkazu DEFB (DEFW) a použijete-li jednopísmenné návěští, může se Vám vejít na jeden řádek vícekrát než původní konstanta.

Uvedené způsoby nejsou doporučením jak psát zdrojové texty, vedou totiž až na malé výjimky k horší přehlednosti. Je to způsob, jak se vyhnout nutnosti použít oddělenou kompilaci. Pokud se Vám podaří zjistit, že se zdrojový text nevejde do paměti už v polovině práci, nezbyde Vám než ho rozdělit, začnou-li problémy s pamětí až na konci, stojí za pokus zkrátit zdrojový text - obvykle se to podaří - stále je to lepší než se zabývat dělením zdrojového textu a používat oddělenou kompilaci.

Tabulka symbolů:

Tabulka je uložena za zdrojovým textem a má následující tvar:

čítač	dva byty	počet návěští v tabulce
tabulka vektorů	dva byty .... dva byty	odkazy do vlastní tabulky, počet odkazů je roven počtu návěští, v horním bytu odkazu je v bitech č. 6 a 7 uložena informace o definování nebo uzamčení daného návěští
vlastní tabulka	dva byty, jméno .... dva byty, jméno	dva byty obsahují hodnotu daného návěští, poslední znak jména je invertován, identifikátory jsou abecedně seřazeny, jejich počet je roven počtu návěští

Podrobnosti editoru:

Na tomto místě se dozvíte o způsobu, kterým probíhá převádění řádku z textového tvaru do tvaru, ve kterém je uložen. Nejdříve se oddělí návěští v poli návěští (pokud tam nějaké je), potom se rozdělí instrukce na mnemoniku a dva operandy (operand může být i prázdný), zjistí se o jaký typ mnemoniky a operandů se jedná a prohledá se tabulka, jestli taková instrukce existuje. Při prohledávání operandů se nejprve hledají registry a když se nenajdou, snaží se assembler vyhodnotit operand jako výraz, tento způsob umožňuje, že je možno použít jako návěští zcela libovolnou kombinaci znaků, nejsou zde žádná rezervovaná slova - pouze u některých slov je nutno použít takový zápis, aby byl vyhodnocen jako výraz:

např. chcete použít jako návěští řetězec 'SP' - tento řetězec by se vyhodnotil jako registr SP, napíšete-li však SP+0 , bude vše v pořádku

```
        ld   hl,sp        ;takto zadaný řádek bude
                          ;vyhodnocen jako chyba

        ld   hl,sp+0      ;při tomto zadání už bude
                          ;přijat a slovo 'sp' bude
                          ;vyhodnoceno jako návěští

        ld   hl,+sp       ;pokud to zadáte takhle, dojde
                          ;také k přijetí, zmizí však
                          ;znaménko '+' a při vyeditování
                          ;a zpětném vložení bude opět
                          ;hlášena chyba (totéž při LOAD)
```

Poslední poznámka se týká používání závorek pro označení adresy - pokud je v závorce číselná hodnota (výraz), stačí zadat jen tu levou, pravá závorka bude doplněna. Pokud však totéž uděláte a v závorce má být registr (např. (hl)), bude slovo 'hl' vyhodnoceno jako návěští, nikoliv jako registr - zde je nutno závorku uzavřít. Použití závorek při uzávorkování výrazu není povoleno!

```
        ld   a,(hl))      ;toto se vezme jako   ld   a,(hl)

        ld   a,(hl        ;toto bude vyhodnoceno jako
                          ;ld   a,(HL), tedy HL je návěští!

        ld   a,(label     ;toto stačí k tomu aby to bylo
                          ;bráno jako   ld   a,(LABEL)
```

IV. Monitor PROMETHEUS

Monitor PROMETHEUS byl vytvořen tak, aby vhodně doplnil možnosti, které poskytuje assembler PROMETHEUS. Poskytuje mnoho funkcí, které usnadňují ladění programů. Při návrhu ovládání byla zachována shoda s programem VAST od T.R.C prakticky ve všech shodných funkcích - uživatelé tohoto programu jistě rádi tuto skutečnost ocení.

Po vstupu do monitoru se vymaže obrazovka a vypíše čelní panel, pokud nebyl čelní panel předefinován, bude vypadat takto:

```
00000    di                              ;výpis dvou řádků strojového
00001    xor  a                          ;kódu od aktuální adresy

A:000 00000000 .  DI  (SP):45043  ;výpisy registrů A,B,C,D,E,H,L
B:175 \                   65297   ;a dvojregistrů BC,DE,HL,IX,
C:255 e  BC:45055 SP:00000 50175  ;IY,SP a výpis stavu přerušení
D:001 .  DE:00257 IX:00000 04555  ;výpis zásobníku (5 adres),
E:001 .  HL:00000 IY:23610 23850  ;výpis registrů R a F - jsou
H:000 .                           ;vypsány platné podmínky, dále
L:000 .  R:000 NZ NC PO P T:00000 ;výpis počítadla T-cyklů
 UNIVERSUM Control ON  Call NON   ;stavový a editační řádek
```

Po vypsání čelního panelu je očekáván stisk nějaké klávesy případně více kláves, po stisknutí klávesy se zjistí, odpovídá-li stisknutá klávesa nějaké funkci, pokud ano, přejde řízení do této funkce, pokud ne, opět se čeká na stisk klávesy.

Přístup k paměti (aktuální adresa) a pomocné funkce

Návrat do systému assembleru - klávesa Q

Po stisku klávesy Q se provede návrat do assembleru, po návratu bude assembler ve stejném stavu, v jakém byl opuštěn, toto neplatí v případě, že byla prováděna zpětná kompilace.

Nastavení aktuální adresy - klávesa M

Po stisknutí klávesy M se ve spodním řádku objeví otázka Memory a kurzor, nyní je možno vložit matematický výraz (pravidla jsou stejná jako v assembler), ten bude vyhodnocen zleva doprava, v případě, že dojde při vyhodnocování výrazu k chybě, zazní zvukový signál a do editačního řádku se na několik sekund vypíše chybové hlášení, potom se opět objeví editační řádek s textem, který v něm byl při odeslání a je umožněna oprava chyby. Pokud chcete přerušit vkládání, stiskněte klávesu EDIT (CAPS SHIFT + 1), řízení se vrátí do hlavní smyčky. Hodnota výrazu bude přiřazena aktuální adrese.

Posun na další instrukci - klávesy CAPS SHIFT + 6

Při použití této volby se zjistí délka instrukce na aktuální adrese a aktuální adresa se zvětší o tuto délku. Po provedení se obnoví výpis čelního panelu.

Posun o jeden byte zpět - klávesy CAPS SHIFT + 7

Odečtení jedničky od aktuální adresy, po provedení se obnoví výpis čelního panelu.

Vnoření o jednu úroveň - klávesy CAPS SHIFT + 8

Při prohlížení programu ve strojovém kódu se často objeví nutnost podívat se do nějakého podprogramu a neztratit přitom současné místo - v prohlíženém podprogramu může dojít ke stejné potřebě. Proto existuje funkce vnořování - po zvolení této funkce se objeví dotaz 'Memory', současná hodnota aktuální adresy se uloží na zvláštní zásobník (kapacita 10 adres) a aktuální adresa je nastavena vloženou hodnotu. Pokud je zásobník plný, příkaz se neprovede. Obnoví se výpis čelního panelu.

Vynoření o jednu úroveň - klávesy CAPS SHIFT + 5

Vynoření je opak funkce vnoření. Funkce odebere ze zásobníku hodnotu a nastaví na ni aktuální adresu, pokud na zásobníku není žádná adresa, nic se neprovede. Obnoví se výpis čelního panelu.

Smazání obrazovky - klávesy CAPS SHIFT + 0

Tato funkce smaže obrazovku, nastaví barvy, které používá assembler (i monitor) a obnoví se výpis čelního panelu.

Smazání výpisového okna - klávesy CAPS SHIFT + 2

Smazání části obrazovky, která je určena pro výpisy - tato část obrazovky je volitelný počet řádků - viz Editor výpisů.

Volání podprogramů a použití BREAKPOINTu:

Volání podprogramu - klávesy SYMBOL SHIFT + H

Po stisku klávesy se objeví dotaz 'Call' na adresu, o níž má být volán podprogram. Po vložení adresy (zadávejte velmi pozorně, pokud se zmýlíte, může to mít neodstranitelné následky) se nastaví všechny registry na hodnoty uvedené v čelním panelu, podle stavu indikátoru se provede zavolání podprogramu buď s povoleným nebo zakázaným přerušením. Mód přerušení bude takový, jaký byl nastaven naposledy - jádící systém pracuje pod zakázaným přerušením a nemění nastavený mód, pouze při návratu do systému BASIC je nastaveno IM 1 (příkazy BASIC, NEW). Po návratu z podprogramu se uloží všechny registry, zjistí se jestli při návratu bylo nebo nebylo povoleno přerušení a obnoví se výpis čelního panelu. Při volání se správně ukládá také hodnota registru R. Při použití tohoto příkazu si ověřte, zda je registr SP nastaven na správné místo - nesmí být samozřejmě nastaven do oblasti paměti ROM (1-16385), takto se na zásobníku neuloží návratová adresa (ROM nelze přepsat) a program se nevrátí na správné místo!

Nastavení startu pro použití BREAKPOINTu - klávesa V

Tato funkce vezme hodnotu aktuální adresy a uschová ji pro potřebu běhu s BREAKPOINTem - viz následující funkce.

Běh s pomocí BREAKPOINT - klávesy SYMBOL SHIFT + U

Funkce vyzvedne tři byty na aktuální adrese, uschová je a na jejich místo vloží skok do monitoru. Potom se nastaví všechny registry na hodnoty uvedené v čelním panelu, povolí nebo zakáže se přerušení a provede skok na adresu zadanou stiskem klávesy V. Pokud program proběhne místem s BREAKPOINTem, vrátí se zpět do monitoru, budou uloženy všechny registry, zjištěn stav přerušení a na své místo se opět uloží odebrané tři byty. Z uvedeného je zřejmé, že tato funkce není použitelná v paměti ROM. Opět je zprávně zpracován registr R. U této instrukce není potřebné hlídat hodnotu SP, důležité je pouze to, aby procesor při provádění programu na BREAKPOINT narazil.

Příklad:

```
  49999     ...
  50000     ld   hl,START      ;na tomto místě stiskněte V
  50003     ld   bc,30000      ;počet průchodů
  LOOP      ld   a,(hl)
  50005     cpl
  50006     ld   (hl),a
  50007     inc  hl
  50008     dec  bc
  50009     ld   a,b
  50010     or   c
  50011     jr   nz,LOOP
  50013     call SUBROUT       ;na tomto místě stiskněte klávesy
  50016     ...                ;SYMBOL SHIFT + U
```

Označení 'na tomto místě' znamená, že v okamžiku stisknutí klávesy V bude aktuální adresa rovna 50000 a v okamžiku stisku kláves SYMBOL SHIFT + U bude aktuální adresa rovna 50013.

Varovný příklad:

Při vybírání místa pro vložení BREAKPOINTU kontrolujte důsledně, aby BREAKPOINT (3 byty) nepřepsal jinou část programu, kterou bude běh požadovat:

```
  50000     ld   hl,43210      ;na této instrukci stiskněte V
  50003     call PROCEDUR      ;
  50006     ret                ;na této instrukci má být konec

  PROCEDUR  ld   a,(hl)        ;kdyby byl použit BREAKPOINT
  50008     inc  hl            ;na adrese 50006, přepíší se
  50009     ret                ;také adresy 50007 a 50008 což
                               ;nelze potřebovat a je tedy
                               ;nutno zvolit jiný způsob
                               ;projdení této části programu
```

Přístup na magnetofon - LOAD a SAVE:

Uložení bloku na kazetu - klávesa S

Assembler umožňuje ukládat na kazetu jak bezhlavičkové bloky, tak bloky CODE (se standartní hlavičkou jako v BASICu). Postup je následující:

- stiskněte klávesu S

- na dotaz 'First' vložte adresu prvního bytu bloku, který má být nahrán na kazetu

- na dotaz 'Last' vložte poslední adresu zvoleného bloku

- na dotaz 'Leader' můžete odpovědět dvěma způsoby:

  1. vložením čísla - v tomto případě bude na kazetu uložen bezhlavičkový blok, jako leader (značkový byte) bude použito toto zadané číslo, nahrávání se provádí hned po odeslání, nečeká se na další stisk klávesy

  2. vložením dvojtečky a jména (použije se prvních 10 znaků, ostatní budou oddělený stejně jako v BASICu) - v tomto případě bude vytvořena hlavička, ve které bude uložen začátek bloku (First), jeho délka (Last-First+1) a zadané jméno, po stisku klávesy ENTER se počká na stisk nějaké klávesy a pak se začne ukládání bloku na kazetu (magnetofon musíte spustit sami)

Uložení bloku na kazetu - klávesy SYMBOL SHIFT + S

Tento příkaz pracuje stejně jako předcházející příkaz, pouze místo dotazu na 'Last' se ptá na 'Length', tedy délku ukládaného bloku - zadání parametrů bloku je takto úplně stejné jako v BASICu. Chcete-li například uložit přeložený program na kazetu, provedete to nejlépe takto:

- na začátek programu vložte návěští A0START - písmeno A a číslo 0 jsou tam proto, aby při výpisu tabulky symbolů nebylo třeba toto návěští hledat - bude hned na začátku

- za konec zdrojového textu programu vložte sekvenci

```
A0LENGTH equ $-A0START
```

  písmeno A a číslo 0 jsou tu ze stejných důvodů jako u návěští A0START

- proveďte kompilaci příkazem 'MONITOR a' a v monitoru zvolte nyní popisovanou funkci, na dotaz 'First' vložíte A0START a na dotaz 'Length' A0LENGTH

- návěští A0LENGTH Vám může poskytovat informaci jak dlouhý je vyrobený strojový kód, která není nikdy na škodu

Nahrání bloku z kazety do počítače - klávesa J

Funkce se zeptá na adresu, kam se má blok nahrávat - dotaz 'First', na poslední adresu nahrávaného bloku - dotaz 'Last', a na značkový byte nahrávaného bloku - dotaz 'Leader'. Potom se nahraje první blok, který bude nalezen, v případě, že nebude splňovat parametry, které byly zadány, nebo bude zjištěna chyba v paritě bude ohlášeno chybové hlášení 'READ/WRITE Error'. Tato chyba bude hlášena také v případě, že blok bude zasahovat ladicí systém nebo zdrojový text - další provádění se přeruší ještě před zadáním značkového bytu.

Nahrání bloku z kazety do počítače - klávesy SYMBOL SHIFT + J

Funkce pracuje jako funkce předešlá, liší se jen tím, že se místo dotazu 'Last' používá dotaz 'Length'.

Přečtení hlavičky nebo značkového bytu - klávesa Y

Provádí se čtení hlavičky z kazety - vypíše se typ bloku (0 je BASIC, 1 je číselné pole, 2 je znakové pole a 3 je CODE), potom jméno bloku, jeho počáteční adresu, jeho délku a další informaci, jejíž význam závisí na typu bloku. Po vypsání informací se očekává stisk klávesy, pokud se jedná o klávesu J, bude nahrán blok podle údajů zjištěných z hlavičky. Jinak se vrací řízení do hlavní smyčky.

Paměťové výpisy

Výpis disassembleru - klávesy SYMBOL SHIFT + 4

Po stisku klávesy se do výpisového okna vypisují disassemblované instrukce. Výpis lze přerušit stiskem klávesy EDIT (CAPS SHIFT + 1). Počáteční adresa výpisu je právě nastavená aktuální adresa.

Výpis disassembleru od zadané adresy - klávesa V

Příkaz se dotáže na adresu, od níž má být disassemblování prováděno, otázkou 'First'. Další akce jsou stejné jako v předchozím příkazu.

Způsob výpisu adres - klávesy SYMBOL SHIFT + C

Při vypisování hodnot dvojbytových číselných operandů může disassembler pracovat ve třech režimech, přepínání těchto režimů je cyklické:

1. vypíše se běžným způsobem číselná hodnota
2. pokud se hodnota, která má být vypsána, rovná hodnotě některého návěští v tabulce symbolů, vypíše se místo čísla toto návěští, stejně tak se návěští vypisuje místo adresy před instrukcí
3. vypisuje návěští stejně jako v režimu č. 2, navíc pokud se hodnota zmenšena o jedno rovná hodnotě některého návěští, bude vypsáno: 'návěští+1'

po instalaci ladícího systému je nastaven vypisovací režim č. 3

Povolení / zákaz výpisu adres - klávesa C

Monitor umožňuje zakázat výpis adres před instrukcemi, tento zákaz se netýká návěští, která mohou být vypisována před instrukcemi při zvolení režimů 2 a 3.

Změna číselné soustavy - klávesy SYMBOL SHIFT + 3

Změna číselné soustavy se týká všech čísel vypisovaných ve všech typech paměťových výpisů, netýká se ovšem výpisu registrů v čelním panelu - toto lze ovlivnit v editoru čelního panelu.

Výpis disassembleru na tiskárnu - klávesa D

Program umožňuje posílat výpis disassembleru na tiskárnu. Komunikace se provádí stejně jako v assembleru přes kanál č. 3 operačního systému Spectra. Při zvolení této funkce se program zeptá na první 'First' a na poslední 'Last' adresu úseku, který má být vypsán na tiskárnu - instrukce začínající na poslední 'Last' adrese už vypsána nebude. Vypisování lze přerušit stiskem kláves CAPS SHIFT + ENTER po každé instrukci, každý řádek je pro kontrolu také vypisován do výpisového okna.

Zpětný překlad do zdrojového textu - klávesy SYMBOL SHIFT + D

Funkce se podobá předchozí, výpis strojového kódu směřuje místo na tiskárnu do zdrojového textu v assembleru - vkládání se provádí za příslušný řádek (zcela jako při např. příkazu LOAD). Opět dotaz na 'First' a 'Last', instrukce počínající na adrese 'Last' se také do zdrojového textu nedostane. Pokud dojde k chybě, ohlásí se chybové hlášení a uživateli je dána možnost nalezenou chybu opravit, bude-li hlášena chyba 'Memory full', nezbyde než stiskem EDIT (CAPS SHIFT + 1) celou akci přerušit. Chcete-li převod předčasně ukončit, stiskněte současně klávesy CAPS SHIFT + ENTER. Při zpětném překladu se opět každý řádek vypisuje do výpisového okna. Více Vám poví příklad - viz přílohy.

Znakový výpis paměti - klávesa D

Protože při prohlížení obsahu paměti nepotřebujeme vždy vidět jen strojový kód, umožňuje monitor vypisování čísel v paměti jako znaků. Každý řádek výpisu obsahuje adresu a dvacet pět znaků od této adresy. Znaky s kódy v rozsahu 32 až 127 se vypisují obvyklým způsobem, kódy v rozsahu 0 až 31 jsou nahrazeny tečkou. Pokud je kód znaku větší než 127, odečte se číslo 128 a znak se vypíše stejně jako předtím, pouze bude prohozena barva papíru a barva inkoustu - inverzní výpis. Jako první adresa se vezme aktuální adresa. Výpis se provádí tak dlouho, dokud je stisknuta libovolná adresa. Přerušení výpisu a návrat do hlavní smyčky zajistí stisk klávesy EDIT.

Znakový výpis od zadané adresy - klávesy SYMBOL SHIFT + D

Naprosto totéž jako v předchozím případě, výpis se provádí od adresy zadané na dotaz 'First'.

Výpis čísel - klávesa L

Poslední výpis umožněný monitorem je číselný výpis od aktuální adresy. Na řádku bude vypsána adresa, pět jednobytových čísel a pět znaků odpovídajících těmto číslům. Všechno ostatní je shodné jako u znakového výpisu

Výpis čísel od zadané adresy - klávesy SYMBOL SHIFT + L

Výpis se bude provádět od adresy zadané na dotaz 'First'. Jinak se tato funkce neliší od předchozí.

Vyhledávání posloupností bytů v paměti

Zadání posloupnosti a vyhledání prvního výskytu - klávesa C

Příkaz umožňuje vyhledávat 5 bytů dlouhou posloupnost, některé byty mohou být pro porovnávání nevýznamné. Po zvolení se objeví dotaz '1. byte' až '5. byte', můžete vložit buď požadované číslo (výraz) nebo dvojtečku - takto se označují nevýznamné byty v hledané posloupnosti. Po odeslání pátého čísla se provede hledání, pokud bude posloupnost nalezena, nastaví se na adresu jejího výskytu aktuální adresa, potom se obnoví výpis čelního panelu. Hledání se provádí od adresy, která je větší o 1 než aktuální adresa až do konce paměti.

Příklady:

hledáte výskyt posloupnosti znaků 'ABcd', zadejte:

1. byte 'A'
2. byte 'B'
3. byte 'c'
4. byte 'd'
5. byte :

hledáte instrukci ld hl,33333 , zadejte:

1. byte #21
2. byte 333337256
3. byte 33333/256
4. byte :
5. byte :

Kód instrukce ld hl,N buď znáte, najdete ho v tabulce nebo do prázdného místa v paměti tuto instrukci vložíte (viz editace paměti) a podíváte se na její operační kód.

Vyhledání dalšího výskytu zadané posloupnosti - klávesa N

Aktuální adresa se zvětší o jedničku a od této adresy se hledá další výskyt posloupnosti zadané předchozím příkazem, pokud bude posloupnost nalezena, aktuální adresa se nastaví na její začátek.

Přenosy a plnění bloků

Přenos bloku - klávesa I

Program se zeptá na první 'First' adresu bloku, na poslední 'Last' adresu bloku a na místo, kam má být blok přenesen 'To'. Po zadání se provede kontrola jestli se místo kam má být blok přenesen nepřekrývá s ladícím systémem nebo zdrojovým textem, a provede se přesun. Dojde-li při testu ke

zjišťání kolize těchto dvou částí paměti, provádění se přeruší a vypíše se chybové hlášení 'READ/WRITE Error'. Místa odkud a místa kam se bude přenášet se mohou překrývat, blok bude vždy přesunut dobře.

Přenos bloku - klávesy SYMBOL SHIFT + I

Příkaz se od předchozího liší pouze způsobem zadání parametrů bloku, místo otázky 'Last' je nyní otázka 'Length' na délku přenášeného bloku.

Plnění bloku - klávesa F

Program se zeptá na začátek 'First' a na konec 'Last' bloku, provede se kontrola, zda blok nezasahuje do paměti obsazené ladícím systémem a zdrojovým textem - pokud ano, ohlásí se chyba 'READ/WRITE Error', pokud ne, zeptá se program ještě na to, čím 'With' chcete plnit zvolený blok, potom se blok vyplní.

Plnění bloku - klávesy SYMBOL SHIFT + F

Obdoba předchozí funkce - blok se tentokrát zadává parametry začátek 'First' a délka 'Length'.

Editace paměti

Výraznou vlastností monitoru PROMETHEUS je, že umožňuje zapisovat v monitoru do paměti přímo instrukce, není tedy nutno používat tabulky operačních kódů. Tato funkce využívá podprogramy pro kompilaci z assembleru.

Jednorázová editace paměti - klávesa SPACE

Po stisku klávesy SPACE se objeví v editačním řádku kurzor úplně vlevo, nyní je možno zadat instrukci strojového kódu a to zcela stejným způsobem jako při vkládání zdrojového textu, tedy instrukce začíná od desátého znaku na řádce - u tohoto příkazu funguje automatická tabelace, stačí stisknout ještě jednou mezerník. Po vložení instrukce se provedou oba průchody kompilace a instrukce bude uložena na aktuální adresu. Při ukládání instrukce se kontroluje, aby instrukce nebyla uložena do oblasti ladícího systému se zdrojovým textem ani za uživatelskou zarážku. Pokud chcete instrukci za uživatelskou zarážku přece jen vložit, musíte se vrátit do assembleru, přepsat její hodnotu, vrátit se do monitoru a vložit instrukci. Chcete-li do paměti přece jen zapsat číslo nebo text, použijte odpovídající způsob jako ve zdrojovém textu - pseudoinstrukce DEFB, DEFW, DEFM se stejným účinkem. Tento příkaz sice slouží k editování paměti, můžete ho použít i pro nadefinování hodnoty nějakého (i nového) návěští - použijte pseudoinstrukci EQU, více se dozvíte v příloze. Pokud bude ve vložené instrukci nalezena chyba, ohlásí se do editačního řádku na několik sekund chybové hlášení, bude-li to pokus o zápis do zakázané oblasti, vypíše se chybové hlášení 'Bad PUT (ORG)'. Ve fázi vkládání instrukce do editačního řádku lze funkce přerušit stiskem EDIT.

Opakovaná editace paměti - klávesa E

Chcete-li do paměti vložit více instrukcí, není jednorázová editace příliš pohodlná - je nutno neustále přeskakovat za právě vloženou instrukci - proto použijte tento příkaz. Po zvolení této funkce se v editačním řádku také objeví kurzor, stejně se funkce chová i dále, po zapsání instrukce do paměti se ukazatel posune za vloženou instrukci, obnoví se výpis čelního panelu (důležité jsou dva disassemblované řádky) a je možno vložit další instrukci. Při vkládání můžete používat před instrukcí také návěští, bude mu přiřazena hodnota ukazatele pro ukládání instrukce. Při psaní se lze odvolávat na všechna již existující definovaná návěští - tedy napřed návěští použít v poli návěští a pak se na něj lze odvolat v poli operandů, jinak bude hlášena chyba, odvolávat se samozřejmě lze i na návěští definovaná v průběhu poslední kompilace nebo definovaná dříve pomocí jednorázové editace pseudoinstrukcí EQU - viz příloha.

Krokování a trasování programů

Krokování a trasování jsou nejdůležitější funkce, které monitor poskytuje. Při trasování a krokování instrukcí se provádí kontroly jestli instrukce nepoužívá zakázané oblasti pro zápis, čtení a běh - takových oblastí je možno pro každou činnost definovat až 5, navíc oblast v níž se nachází ladící systém a zdrojový text se kontroluje na všechny činnosti. Dále se kontroluje, aby se nepoužila instrukce HALT při zakázaném přerušení. Provádění kontrol je možno vypnout a rychlost trasování se přibližně zdvojnásobí - kontrola na instrukci HALT se vypnout nedá, vypínají se kontroly na zakázané oblasti paměti včetně oblasti s ladícím systémem a zdrojovým textem. U instrukcí CALL a RST lze volit tři režimy - obvyklou simulaci, u vybraných adres (možno až 10 adres) se tyto instrukce provedou přímo (nebude se simulovat volání podprogramu ale celý podprogram se zavolá najednou) anebo se budou všechny instrukce CALL a RST provádět najednou. Poslední dva režimy zvyšují rychlost trasování, je tu však riziko ztráty kontroly nad programem - režim rychlého volání vybraných podprogramů lze výhodně použít na dokonale odladěných podprogramech pro zvýšení rychlosti trasování. V částech programu, které jsou přímo volány samozřejmě neexistují žádné kontroly. V editačním řádku jsou vypsány indikace stavu kontroly (Control ON/OFF) a způsobu, jakým se budou provádět instrukce CALL a RST (Call NONV/DEF/ALL) (žádné/difinované/všechny instrukce se budou přímo volat).

Nevypnete-li kontrolu a budete-li krokovat či trasovat své programy se zakázaným přerušením, nemůže se stát, že by monitor ztratil řízení programu. Budete-li používat povolené přerušení v módu IM 1 nesmíte měnit hodnotu registru IY (je používán systémem jako ukazatel do systémových proměnných), dávejte si také pozor, aby se registr SP (ukazatel na zásobník) nenastavil do oblasti paměti ROM. Budete-li povolovat přerušení v módu IM 2, musí nastaveno správně - přerušení není trasováno ani nijak kontrolováno!

Při krokování a trasování počítá monitor také časovou náročnost programu - počítadlo T-cyklů (T:00000). Počítání cyklů se hodí při práci s na programech, které musí trvat přesně zadanou dobu - programy LOAD a SAVE, hudební rutiny, barevné efekty na obrazovce a v borderu. Na začátku nastavte počítadlo na nulu a protrasujte tu část, jejíž časovou délku chcete zjistit. Při trasování se nesmí žádná instrukce CALL nebo RST provádět přímo.

Krokování - klávesy SYMBOL SHIFT + Z

Po stisknutí kláves se provede simulace jedné instrukce na

aktuální adrese, před provedením se obnoví hodnoty všech registrů (včetně R) a případně se povolí přerušení, po provedení simulace se hodnoty všech registrů opět uloží, zaznamená se stav přerušení a obnoví se výpis čelního panelu. Pokud je povolena kontrola, provedou se potřebné testy před vlastní simulací, když je zjištěna nějaká kolize, vypíše se chybové hlášení a čeká se na stisk klávesy. Simulace instrukcí CALL a RST se provádí podle zvoleného režimu. Aktuální adresa se nastaví podle toho, jak simulovaná instrukce změní obsah čítače instrukcí.

Pomalé trasování - klávesa T

Funkce provede simulaci instrukce na aktuální adrese, obnoví výpis čelního panelu, změní aktuální adresu, provede test stisknuté klávesy a pokud není stisknuta klávesa BREAK (CAPS SHIFT + SPACE) opakuje celý cyklus stále znovu. Stisknete-li při trasování současně klávesy CAPS SHIFT + ENTER, vynechá se po dobu, kdy budou klávesy stisknuty, obnovování čelního panelu, tato možnost je užitečná pro zrychlení trasování na kratší dobu - třeba pro urychlení některého cyklu. Simulace probíhá stejně jako při krokování.

Rychlé trasování - klávesy SYMBOL SHIFT + T

Při zvolení této možnosti se program zeptá na adresu instrukce na níž by se měl zastavit - 'Last', a provádí cyklicky tyto akce: simuluje instrukci na aktuální adrese, změní aktuální adresu, pokud jsou stisknuty klávesy CAPS SHIFT + ENTER obnoví čelní panel, testuje stisk klávesy BREAK (CAPS SHIFT + SPACE) a testuje, jestli se aktuální adresa nerovná zadané poslední adrese, pokud tomu tak není a není stisknut BREAK, skočí opět na začátek cyklu. V opačném případě se obnoví výpis čelního panelu a rychlé trasování se zastaví. Pokud nevíte, na které adrese by se měl program zastavit (nezáleží na tom - zastavíte ho BREAKem) vložte třeba 0.

Práce s registry

Změna stavu indikátoru přerušení - SYMBOL SHIFT + M

Přehodí hodnotu indikátoru stavu přerušení z EI na DI nebo naopak. Indikátor určuje v jakém stavu přerušení se bude simulovat každá instrukce, volat podprogram nebo provádět běh s pomocí BREAKPOINTu, po skončení každé takové akce se indikátor nastaví podle skutečnosti po provedení instrukce.

Přehození obsahů základních a alternativních registrů

klávesy SYMBOL SHIFT + B

Provede výměnu základní a alternativní sady registrů, ve strojovém kódu je to ekvivalentní instrukcím EXX a EX AF,AF'. Při trasování nebo krokování jsou základní vždy ty registry, které jsou viditelné na čelním panelu. Chcete-li se tedy při krokování podívat na hodnoty alternativních registrů, musíte před dalším krokováním registry vrátit.

Nastavení obsahu registrů - klávesy SYMBOL SHIFT + N

Po zvolení této možnosti se v editačním řádku objeví nápis 'ld ', vložte jméno registru, jehož hodnotu chcete změnit, čárku nebo mezeru a napište číslo (výraz), které chcete do tohoto registru zapsat, registry jsou tyto:

Jednobytové registry: A,B,C,D,E,H,L,HX,LX,HY,LY,I,R

Dvoubytové registry : AF,BC,DE,HL,IX,IY,SP

Počítadlo T-cyklů : T

Paměťové ukazatele : X,Y (viz editor čelního panelu)

Stavový registr : F

U stavového registru (F registr) je možno měnit jednotlivé bity následujícím způsobem:

ld f,c - změní hodnotu CARRY flagu z NC na C a naopak

ld f,s - změní hodnotu SIGN flagu z P na M a naopak

ld f,z - změní hodnotu ZERRO flagu z NZ na Z a naopak

ld f,p - změní hodnotu PARITY flagu z PO na PE a zpět

Nastavení zakázaných oblastí a jiných parametrů

Nastavení DEFB oblastí - klávesa 1

Monitor dovoluje nastavovat oblasti, které budou při výpisu disassemblerem vypsány jako jednobytová data, takových oblastí si můžete nadefinovat celkem 5. Každá oblast je zadána prvním a posledním bytem (včetně). Po stisku klávesy 1 se do výpisového okna vypíší současně nastavené oblasti, každý řádek začíná číslem oblasti (0 až 4), za ním je vypsána adresa prvního bytu oblasti a pokud je stejná jako hodnota nějakého návěští, je vypsáno také toto návěští, nakonec je vypsána adresa posledního bytu oblasti a případné návěští. Nyní můžete stisknou buď klávesu s číslem některé oblasti - tato oblast se vymaže a znovu se vypíše seznam nastavených oblastí (windows-okna), nebo klávesu 1, v tomto případě se program zeptá na první 'First' a na poslední 'Last' adresu oblasti, prověří, jestli platí, že 'First'<='Last', a přidá novou oblast za již definované oblasti, pokud je již všech pět oblastí definováno, klávesa 1 nebude reagovat. Stisknete-li jinou klávesu než 0,1,2,3,4,1, program se vrátí do hlavní smyčky. Za oblast s jednobytovými hodnotami je považován také prostor paměti obsazený ladicím systémem a zdrojovým textem.

Nastavení DEFW oblastí - klávesa 2

Další typ oblastí, které lze používat jsou oblasti s dvoubytovými hodnotami (tabulky adres), jejich nastavení je úplně stejné jako u oblastí DEFB.

Nastavení oblastí, z nichž se nesmí číst - klávesa 3

Při ladění programu je výhodné mít jistotu, že program

nečte data z míst v paměti, která k tomu nejsou určena, proto byla do programu dána možnost tato místa instrukcím zakázat. Každá instrukce, která nějakým způsobem čte data z paměti, např. přímo z adresy, z adresy v registru, pomocí indexregistrů nebo přes SP registr (tedy i RET, POP a EX (SP),HL), bude nejprve prověřena, zda tak nečiní z oblasti, kde to má zakázáno, pokud ano, její simulace se neprovede a bude ohlášeno chybové hlášení. Za oblast se zákazem čtení je navíc považován pamětový prostor obsazený ladicím systémem a zdrojovým textem. Vlastní definice těchto 'No read' oblastí je stejná jako u DEFB oblastí.

Nastavení oblastí, do nichž se nesmí zapisovat - klávesa 4

Důležitější, než zjišťovat, zda program nečte data z míst v paměti, z nichž by neměl, je mít možnost hlídat program, aby nezapisoval do jiných než povolených úseků paměti. U každé instrukce, která provádí zápis do paměti (včetně instrukcí CALL RST, PUSH, EX (SP),HL) se provede kontrola jestli se nepřepíší zakázané oblasti, pokud ano, simulace se neprovede a bude hlášena chyba 'READ/WRITE Error'. Nastavení je obdobné jako u DEFB oblastí. Oblast s ladicím systémem a zdrojovým textem je také považována za oblast se zákazem zápisu - No write.

Nastavení oblastí se zákazem běhu - klávesa 5

Posledním typem oblastí, kde je možnost něco zakázat, jsou oblasti se zákazem běhu. Jakmile by se po provedení nějaké instrukce octl program v oblasti se zakázaným během, instrukce se nevykoná a ohlásí se chyba 'RUN Error'. Zadávání je stejné jako u všech předchozích. Použití snad ani není potřeba nějak vysvětlovat - například když se program nevrací tam, kam by měl, zakažte vše, kromě vlastní oblasti s programem, jakmile se nějaká instrukce při trasování pokusí o skok mimo povolenou oblast, bude hlášena chyba - tyto problémy vyvolává nejčastěji chybná práce se zásobníkem návratových adres pomocí instrukcí PUSH a POP - obvykle něco chybí nebo přebývá.

Definování adres pro přímé volání - klávesa 6

Výhodný způsob, jak urychlit trasování programů, je provádět vybrané příkazy CALL a RST přímo (nesimulovat každou instrukci ve volaných podprogramech - volat celý podprogram najednou). K tomuto účelu si můžete zvolit celkem deset adres. Pokud nastavíte režim volání definovaných podprogramů (Call DEF) bude se každý příkaz CALL nebo RST, jehož adresní část je rovna některé adrese ze seznamu definovaného tímto příkazem, nahrazovat zavoláním podprogramu na této adrese. Při ladění programu se tato možnost výhodně používá na již odladěné a dobře fungující podprogramy (nehrozí tu již nebezpečí ztráty kontroly, podprogram se vrací zpět tam, odkud byl zavolán) - tisk znaku, test klávesnice pokud není používáno přerušení - a také na podprogramy, jejichž práci není možno zpomalit - nahrávání a zvukové podprogramy. Ukázky použití v příloze. Editace je podobná jako u definování oblastí, vymazávání je možno navíc provádět klávesami 0 až 9.

Změna režimu provádění instrukcí CALL a RST - klávesa X

Jak bylo dříve uvedeno, mohou se instrukce CALL a RST provádět třemi různými způsoby, které se liší hlavně rychlostí a bezpečností - bohužel nepřímo úměrně. Tyto režimy jsou indikovány v editačním řádku (NON/DEF/ALL) a jejich význam už

byl popsán dříve.

Zapnutí / vypnutí kontroly instrukcí - SYMBOL SHIFT + X

Tímto příkazem lze najednou vypnout provádění všech kontrol, nejméně dvojnásobně to zvýší rychlost provádění trasování. Tuto možnost používejte uváženě.

Editor čelního panelu

Přechod do editoru čelního panelu - klávesy SYMBOL SHIFT + W

Po přepnutí do editoru čelního panelu se vymaže obrazovka, obnoví výpis čelního panelu a rozsvítí se jedna z položek v čelním panelu. Nyní je možno pro každou položku definovat její umístění, způsob výpisu a další možnosti - viz dále.

Ovládání editoru čelního panelu

CAPS SHIFT + 1 (EDIT) - opuštění editoru čelního panelu

4 - přechod na následující položku v seznamu (viz dále)

3 - přechod na předchozí položku v seznamu

5 - posun položky o jednu pozici doleva

6 - posun položky o jednu pozici dolů

7 - posun položky o jednu pozici nahoru

8 - posun položky o jednu pozici doprava

A až Z - nastavení velikosti položky (A až Z), pro registry znamená, jestli se budou nebo nebudou vypisovat, pro pamětové výpisy se takto mění počet adres, které se vypisují

SYMBOL SHIFT + D - zapnutí nebo vypnutí výpisu také v desítkové soustavě

SYMBOL SHIFT + H - zapnutí nebo vypnutí výpisu také v šestnáctkové soustavě

SYMBOL SHIFT + B - zapnutí nebo vypnutí výpisu také ve dvojkové soustavě

SYMBOL SHIFT + C - zapnutí nebo vypnutí výpisu také ve znakové formě

SYMBOL SHIFT + T - změna typu výpisu u pamětových výpisů - vypisují se buď jednobyty nebo dvoubyty

SYMBOL SHIFT + S - změna směru výpisu u pamětových výpisů - obsahy jednotlivých adres jsou vypisovány buď vedle sebe nebo pod sebe

Seznam položek, které lze používat:

Editační řádek - lze ovlivnit pouze, který řádek bude používán pro tyto účely

Výpisové okno - lze ovlivnit počet řádků, které budou použity, lze také změnit polohu okna

Výpisové okno disassembleru - při obnovování čelního panelu se vypisuje několik řádků instrukcí strojového kódu od aktuální adresy, počet řádků a polohu okna lze nastavit

Stav přerušení - výpis EI nebo DI

Jednobytové registry - A,B,C,D,E,H,L,I,R,HX,LX,HY,LY a F - lze volit soustavy, ve kterých budou jejich hodnoty vypisovány - registry mohou být vypisovány ve všech najednou, u registru F je možno volit (SYMBOL SHIFT + B) mezi binárním výpisem a výpisem splněných podmínek

Dvojbytové registry - AF,BC,DE,HL,SP,IX,IY - podobá se jednobytovým registrům

Výpis počítadla T-cyklů - lze nastavit stejně jako dvojbytový registr

Adresové výpisy od zadané adresy (ukazatele X,Y) u těchto výpisů lze volit počet bytů, které budou vypisovány, lze měnit směr, kterým se jednotlivé hodnoty vypisují, lze měnit také jestli se budou vypisovat jednobytové nebo dvoubytové hodnoty

Adresové výpisy od adresy v dvojregistru - (BC),(DE),(HL),(SP),(IX),(IY) - nastavit lze stejné vlastnosti jako u předcházející položky

Nejjistější způsob, jak si osvojit způsob použití editoru čelního panelu, je ho zkusit použít - nemůžete nic zkazit.

V. Příklady práce s ladicím systémem

1. Příklad - zpětný překlad

Tento příklad Vám pomůže vyzkoušet a pochopit způsob jak lze převést strojový kód do zdrojového textu. Provádějte přesně popsané úkony a sledujte výsledky.

Nahrajte PROMETHEUS na adresu 24000 a instalujte ho včetně monitoru. V assembleru vložte příkaz MONITOR, objeví se výpis čelního panelu, nyní stiskněte klávesu SPACE, dolní řádek panelu zmizí a objeví se kurzor, napište 'ldbytes equ #556' a stiskněte ENTER. Nyní stiskněte klávesu M a na dotaz 'Memory:' vložte text 'ldbytes'. Po odeslání se obnoví výpis čelního panelu a uvidíte výpis obsahu paměti ROM od adresy 1366, místo čísla 1366 však bude vypsáno návěští LDBYTES.

Nyní stiskněte klávesy SYMBOL SHIFT (dále SS) a 4, objeví se několik řádků výpisu disassembleru, až se ve výpisu objeví instrukce 'call 1511', přerušte výpis klávesou EDIT. Stiskněte opět klávesu SPACE a vložte 'ldedge1 equ 1511'.

Opakujete-li znovu disassemblování naleznete další adresu v instrukci 'jr nc,1387'. Vložte 'ldbreak equ 1387' a vylistujte si několik instrukcí (SS+4), návěští LDBREAK se objevilo na všech místech, kde bylo dříve číslo 1387, podobně tak návěští LDEDGE1. Vypište si instrukce tak, aby návěští LDBREAK na začátku řádku bylo v horní části výpisu.

Další adresa je číslo 1396, vložte 'ldwait equ 1396'. Obnovte výpis.

Ve výpisu můžete vidět instrukci 'jr nc,LDBREAK+1', toto je skok na druhou instrukci od adresy LDBREAK, pokud by se mezi tyto dvě instrukce později už ve zdrojovém textu vložila další instrukce, nepracoval by program správně. Vložte tedy raději 'ldstart equ ldbreak+1', uvědomte si, že můžete používat již definovaná návěští místo adres, snížíte si tak pravděpodobnost omylu při vkládání.

Abychom se nezdržovali vložte postupně tato návěští:

```
ldedge2  equ 1507
ldleader equ 1408
ldsync   equ 1423
ldmarker equ 1480
ldflag   equ 1459
ldverify equ 1469
ldnext   equ 1474
lddec    equ 1476
ld8bits  equ 1482
ldloop   equ 1449
```

Prohlédněte si disassemblerem paměť, v hlavní části programu je každá adresa nahrazena návěštím. Nyní se podíváme na oba podprogramy: LDEDGE1 a LDEDGE2 nejprve LDEDGE2, nastavte aktuální adresu na LDEDGE2 (klávesa M) a při disassemblování naleznete adresy 1513 a 1517, vložte návěští LDDELAY a LDSAMPLE, pokud jste si dobře všimli, je návěští LDEDGE1 na třetí instrukci za návěštím LDEDGE2 a tedy společná část již neobsahuje adresy, které by nebyly nahrazeny návěštími.

Nyní provedeme zpětný překlad: Stiskněte SS a D, na dotaz 'First:' vložte 'ldbytes' a na dotaz 'Last:' vložte '1540+1'. Po odeslání uvidíte výpis disassemblované části paměti nyní bez adres před instrukcemi, pouze s návěštími. Po skončení stiskněte klávesu C - opět se zapne výpis adres před instrukcemi.

Vraťte se do assembleru stiskem klávesy 0.

Pokud jste všechno provedli přesně podle pokynů, měl by zdrojový text končit na adrese 40607. Stiskněte SS+K a dostanete se na začátek zdrojového textu. Najděte instrukci 'ld hl,1343', je to sedmá instrukce od začátku, nastavte do příslušného řádku následující instrukci 'push hl' a stiskněte dvakrát CAPS SHIFT+9 a vymažte tyto dva řádky.

Začátek programu bude po provedení změn následující:

```
LDBYTES  inc  d
         ex   af,af'
         dec  d
         di
         ld   a,15
         out  (254),a
         in   a,(254)
         ...
```

Zdrojový text bude končit na adrese 40598.

V tomto okamžiku je nejvyšší čas pro uložení zdrojového textu na kazetu. Stiskněte SS+S a vložte ':loader', spusťte volnou kazetu a stiskněte nějakou klávesu, po provedení stiskněte SS+V (VERIFY) a po odeslání vraťte kazetu a zkontrolujte nahrávku, pokud bude chybná, opakujte VERIFY a když bude neúspěšné, opakujte SAVE, nyní už můžete odeslat jen SAVE bez ničeho a jako jméno se vezme poslední zadané jméno.

Nyní připište před program instrukce:

```
         org  60000
         ent  $
START    ld   ix,16384
         ld   de,6912
         ld   a,255
         scf
         call LDBYTES
         ret
```

Vezměte kazetu s libovolnou obrazovkou a nastavte ji před blok dat (za hlavičku) a zadejte příkaz RUN (SS+R) a spusťte magnetofon, po skončení nahrávání bude program čekat na stisk libovolné klávesy, aby se mohl vrátit do editoru.

Vložte 'FIND s:or 2', změníte-li číslo 2 na nějaké číslo z rozsahu 0..7, změní se barva pruhů - zůstanou zachovány dvojice:

černá - bílá
modrá - žlutá
červená - světle modrá
fialová - zelená

Další instrukce, která mění barvy pruhů je instrukce CPL, zaměňte ji za instrukci xor %11111111, změníte-li jedničky na posledních třech místech za nuly (no všechny, různě zkoušejte), můžete získat další kombinace barev. Poslední instrukce měnící barvy je instrukce xor 3 za návěštím LDSYNC, která mění barvy zaváděcích pruhů na barvy při nahrávání.

Vložte příkaz 'MONITOR a' (SS+M), provede se překlad a skok do monitoru. Nastavte aktuální adresu na LDEDGE1, stiskněte SS+N a zadejte 't,0' - vynuluje se počítadlo T cyklů. Stiskněte SS+T a vložte 'ldsample-1' - trasování do adresy LDSAMPLE-1. Po skončení ukazuje počítadlo T cyklů hodnotu 354. Toto je počet cyklů procesoru potřebný na provedení tohoto programu:

```
LDEDGE1  ld   a,22
LDDELAY  dec  a
         jr   nz,LDDELAY
```

Tato část programu se provede dvakrát při nahrání každého bitu - tedy celkem 16 krát při nahrání 1 bytu. Stiskněte 0 a

vraťte se do assembleru.

Nahraďte uvedené tři řádky takto:

```
LDEDGE1  ld   a,7
LDDELAY  dec  a
         set  3,a
         out  (254),a
         res  3,a
         jr   nz,LDDELAY
```

Vložte 'MONITOR a', nastavte aktuální adresu na 'ldedge1', nastavte počítadlo T cyklů na 0 a trasujte program až do adresy 'ldsample-1'. Počítadlo T cyklů ukazuje hodnotu 346, tedy o 8 cyklů méně, než by mělo, přidejte dvě instrukce NOP za instrukci 'jr nz,LDDELAY', zopakujte výpočet T cyklů, nyní je hodnota 354 a tedy stejná jako dříve a loader bude pracovat opět bez chyb.

Spusťte program příkazem RUN a vyzkoušejte nahrávání, výsledek uvidíte sami.

Většina loaderů reaguje na stisk SPACE, pokud budete chtít tuto možnost vyloučit, nahraďte instrukci 'ret nc' za návěstím LDSAMPLE instrukcí 'nop'.

Další možnosti už musíte vymyslet a vyzkoušet sami.

2. Příklad - možnosti trasování

Vložte příkaz 'CLEAR y', napište tento krátký program:

```
         ent  $

START    xor  a
         out  (254),a
FILL     ld   hl,16384
         ld   bc,6912
FILL2    ld   a,r
         ld   (hl),a
         inc  hl
         dec  bc
         ld   a,b
         or   c
         jr   nz,FILL2
         ret
```

Proveďte RUN - výsledek nevypadá špatně, zkuste to znovu, je to jiné, také ne ošklivé (pokud nemáte barevnou televizi, tak máte smůlu).

To, že po každém spuštění vypadá obrázek jinak, má na svědomí registr R. Vložte tedy za instrukci 'xor a' instrukci 'ld r,a', tentokrát bude výsledek vždy stejný, registr R má vždy při spuštění hodnotu nastavenu na nulu.

Vložte 'MONITOR a' a přejdete do monitoru, nastavte aktuální adresu na návěstí START, zjistěte, na které adrese je instrukce 'ret', vymažte obrazovku (CS+0) a spusťte rychlé trasování (SS+T), na dotaz 'Last:' dejte adresu instrukce 'ret'. Po odeslání uvidíte značně zpomaleně všechny akce, které program provádí - nyní můžete:

a) nedělat nic - počkat až to samo skončí

b) stisknout CAPS SHIFT a ENTER a sledovat co se děje s jednotlivými registry, rychlost trasování se značně zpomalí

c) stisknout CAPS SHIFT a SPACE a trasování zastavit, případně opět spustit nebo krokovat jednotlivé instrukce

Chcete-li při trasování sledovat jednotlivé registry, použijte pomalé trasování (T), znovu můžete:

a) opět nedělat nic - tentokrát program poběží tak dlouho, dokud nenarazí na instrukci, která by prováděla nebezpečnou akci

b) stisknout CAPS SHIFT a ENTER, teď se program naopak zrychlí, přestanou se vypisovat hodnoty registrů

c) stisknout CAPS SHIFT a SPACE a zastavit trasování

Nastavte opět aktuální adresu na návěští START, vymažte obrazovku (SS+0) a vypněte provádění kontrol. Vyzkoušejte opět rychlé i pomalé trasování - u rychlého by mělo být patrné zvýšení rychlosti.

Až se do sytosti přesvědčíte o rozdílu v rychlostech, kontrolu opět zapněte (SS+X).

Zrychlení trasování vypínáním kontrol raději nepoužívejte, a pokud ano, tak si dobře rozmyslete zda program nebude páchat nějaké nepříjemnosti - při prvním spuštění vždy s kontrolami. Několik sekund takto ušetřených Vás může stát spoustu času ztraceného nahráváním programu znovu do počítače.

Vraťte se do assembleru a před instrukci 'ld hl,16384' vložte tyto řádky:

```
LOOP    push af
        call FILL
        pop  af
        inc  a
        cp   128
        jr   c,LOOP
        ret
```

a takto upravený program spusťte příkazem RUN.

Po skončení se přepněte do monitoru (SS+M) a přepněte režim provádění instrukcí 'call' a 'rst' na režim 'ALL' (X). Stiskněte klávesu 3 a nastavte okno se zakázaným čtením na rozsah 0-1 (po výpisu prázdného seznamu stiskněte 1 a zadejte obě čísla), potom nastavte registr SP na 0. Aktuální adresu nastavte na START a nyní krokujte (SS+Z), pomalu (T) nebo rychle (SS+T) trasujte daný program (pozor na SP - vždy na 0). Instrukce 'call FILL' se provede najednou i s celým podprogramem.

3. Příklad - trasování

Instalujte PROMETHEUS na adresu 25000, přejděte do monitoru, nastavte aktuální adresu na #12a2, registry IY na hodnotu 23610 a SP na 0, povolte přerušení (SS+M) a spusťte rychlé trasování, jako poslední adresu uveďte 0.

- vymaže se obrazovka (uvidíte zpomalené CLS)
- provede se automatický listing (pokud je co vypsat)
- objeví se kurzor v editační zóně

Stiskněte klávesu O, pomalu se vypíše příkaz PLOT, připište ještě '0,0' a stiskněte ENTER. Po chvíli se objeví lečka, stejně vložte příkaz 'DRAW 255,175', při psaní musíte klávesu držet poněkud déle než obvykle, při psaní '55' raději po napsání první číslice klávesu na chvíli pusťte a potom opět stiskněte. Klávesu můžete pustit v okamžiku kdy se z počítače ozve cvakání - zpomalené klávesové echo.

Nastavte režim 'DEF' (X) a vložte jako 'Call' adresy hodnoty #F2C, 16 a #556 (klávesa 6 a] - podobně jako okna).

Nastavte registry, aktuální adresu a přerušení jako předešle a zkoušejte vložit tyto příkazy:

```
PLOT 0,0: DRAW 255,175
CIRCLE 127,87,87 - trpělivost, chvíli počkejte
POKE 25000,0 - program se zastaví na instrukci
     'ld (bc),a', která realizuje příkaz POKE
LIST
LOAD ''SCREEN$ - a samozřejmě také pusťte kazetu
FOR i=22528 TO 23295:POKE i,PEEK 23672:NEXT i
CLS
BEEP .05,40
RANDOMIZE USR 25000 - program se zastaví na adrese
     25000 až bude chtít traspovat assembler
```

4. Příklad - zdrojový text na ukázku

Na kazetě je také hudební rutina ze hry SKATE CRAZY, vymažte zdrojový text (CLEAR y) a nahrajte ho do paměti, chvíli počkejte, než se uloží a spusťte jej příkazem RUN.

OBSAH :