FULL SCREEN EDITOR / ASSEMBLER

Autor : JAMES HUTCHBY

1. UVOD

EDITOR/ASSEMBLER (dale jen E/A) tvori spolecne s programem MACHINE CODE TEST TOOL (MCTT) kompletni vyvojovy system. E/A obsahuje editor pro tvorbu zdrojovych textu a assemblerovsky prekladac. MCTT je ladici prostredek pro programy ve strojovem kodu. (U verze OCP-D-100 je pouzit misto MCTT program MONS 3, ktery je lepsi).

2. LOADOVANI

E/A se naloaduje standartnim zpusobem (LOAD""). Program se sklada ze tri souboru (casti) a spusti se automaticky. (E/A pro 16K-verzi SPECTRA je rozdelen do dvou samostatnych casti - editor a assembler z duvodu omezene pametove kapacity.)

Kdyz chceme pracovat soucasne s E/A i s MCTT, musime naloadovat nejdrive MCTT a az potom E/A !

3. TEXTOVY EDITOR

Textovy editor je strankoveho typu, umoznuje tedy pohybovat kurzorem po cele plose obrazovky.

Funkce klavesnice je klasicka; v E/A pochopitelne neplati BASICovske klicove slova. Funkce preradovace (CAPS LOCK) se indikuje v pravem hornim rohu obrazovky inverznim pismenem C. Ke specialnim symbolum, ktere jsou na klavesnici SPECTRA vyznaceny cervenou barvou, mame pristup pomoci SYMBOL SHIFT. Tlacitka Q,W,E,Y,U,I,A,S,D,F,G pri soucasnem stisknuti SYMBOL SHIFTu davaji znak ("copyright").

Vseobecne plati, ze neni rozdilu mezi velkymi a malymi pismeny. Kde je mozne pouzit velke pismena, je mozne pouzit i male a naopak. Male a velke pismena je mozne dokonce michat. Z tohoto pravidla existuji vsak dve vyjimky:

(a) u prikazu C (Change) (EXT,COM,PROC, - viz dale) a
(b) u jmena souboru na magneticke pasce se respektuji velke a male pismena tak, jak jsme je zadali.

Oblast pameti, ktera obsahuje zdrojovy text, zpracovavany editorem, se jmenuje TEXT BUFFR. Obsah obrazovky je vlastne "okenkem" do text-buffru.

Kazdy radek zdrojoveho textu je rozdelen na tri casti (pole): pole navesti, pole instrukci a pole operandu. Textovy editor si sam zabezpeci tabulaci psaneho textu. Radky s poznamkami, zacinajici znakem ; (strednik) vsak nikdy nemeni upravu, zustavaji vzdy v takove forme, jak jsme je napsali. Poznamkovy radek muze mit nejvice 32 znaku (vcetne stredniku).

Kazdy radek v text-bufru je oznacen peticifernym cislem (00000 az 99999), ktere se objevuje pri levem okraji obrazovky. Je dulezite si uvedomit, ze tato cisla nemaji takovy vyznam, jako cisla radku v BASICu. Nemaji zadny vliv na poradi zpracovani radku a dokonce se v textu muze vyskytnout vice radku se stejnym cislem. Vyznam cislovani radku spociva vylucne v identifikaci chyb pri praci assemblerovskeho prekladace.

E/A pouziva specialni zpusob zobrazovani, ktery umoznuje psat do jednoho radku obrazovky 42 znaku. V zakladnim rezimu se zobrazuje bilym pismem na modrem pozadi, ale barvy je mozne zmenit z BASICu (viz prikaz BASIC [EXT,COM,PROC.]).

V E/A se pouziva kurzor ve tvaru podcarky (_). Pokud je videt na obrazovce blikajici kurzor - podcarku, jsme v rezimu NORMAL EDIT. Kurzorem je mozne pohybovat ve vsech smerech pomoci tlacitek oznacenych sipkami. Pri pohybu kurzorem nahoru nebo dolu muzeme narazit na horni nebo dolni okraj; tehdy zacne text scrollovat.

Kdyz chceme rychle "prebehnout" textem, muzeme pouzit strankovani (TRUE VIDEO smerem nahoru resp. INV. VIDEO smerem dolu).

VYMAZANI ZNAKU se provede nastavenim kurzoru na znak, ktery chceme odstranit a stlacenim tlacitka DELETE (CAPSSHIFT + 0). Kurzor zustane na puvodnim miste, ale cast textu napravo od vymazaneho znaku se posune smerem doleva (vyplni vzniklou mezeru). Tak se nad kurzor dostane novy znak a pokud stale drzime DELETE, mazani pokracuje.

VYMENA ZNAKU je velmi jednoducha. Staci umistit kurzor na urcene misto a jednoduse prepisovat stary text novym textem.

VSUNUTI ZNAKU provedeme tak, ze umistime kurzor na ten znak, pred ktery chceme vsunout novy text. Stlacime GRAPHICS (CAPS SHIFT + 9), cimz editor prejde do rezimu INSERT. Kurzor prestane blikat, pred vyznacenym znakem se vytvori mezera, do ktere muzeme vsunout znaky. Zbyvajici cast radku se automaticky posouva doprava. (Pritom se vsak muze ztratit znak, ktery "vybehne" z obrazovky!) V rezimu INSERT muzeme pouzivat sipky doleva a doprava a DELETE. Sipka doprava vsune mezery az po nasledujici zarazku tabulatoru (to plati jen pro rezim INSERT!). Vsouvani znaku zakoncime stlacenim ENTER. Normalni kurzor zacne opet blikat a to je znameni, ze jsme v rezimu NORMAL EDIT.

Jak jsme jiz uvedli, editor automaticky obstarava rozdeleni textu do prislusnych poli. Staci tedy pole oddelovat mezerou - editor se postara o ostatni. Je treba vsak vedet, ze v poli navesti mame k dispozici maximalne 6 znaku, v poli instrukci maximalne 4 a v poli operandu nejvice 20. To se samozrejme netyka radku s poznamkami, ktere zacinaji strednikem.

Pomoci tlacitka EDIT (CAPS SHIFT + 1) muzeme zrusit vsechny zmeny, ktere jsme v radku provedli a kurzor se vrati na prvni znak v radku. Dokud neopustime radek, mame vzdy moznost se vratit k puvodnimu stavu radku.

RADKOVE PRIKAZY (LINE COMMANDS)

Radkove prikazy provadeji editovaci operace s celymi radky, nebo bloky (vice radku). Do rezimu radkovych prikazu vstoupime tak, ze nastavime kurzor na libovolne misto zvoleneho radku a soucasne stlacime CAPS SHIFT a SYMBOL SHIFT. Klasicky kurzor zmizi a zvoleny radek se zvyrazni malym bilym ctvercem napravo od cisla radku. Nyni jsme v radkovoprikazovem rezimu (nebo-li v rozsirenem rezimu = EXTENDET modu, zkracene v E modu), ve kterem muzeme pouzit nasledujici prikazy:

Zruseni EXTENDET modu - (Abort) CAPS SHIFT + SYMBOL SHIFT

Timto prikazem zrusime rozsireny rezim a vratime se do rezimu NORMAL EDIT.

Vymazat radek - (Delete line) D

Oznaceny radek se odstrani, nasledujici radky scrolluji smerem nahoru a nastavi se NORMAL EDIT.

Vloz radek/radky - (Insert line(s)) I

Stisknutim I se bezprostredne po oznacenem radku vsune prazdny radek s cislem 00000. V jeho prvnim sloupci se objevi neblikajici kurzor. Nyni muzeme vpisovat text a pouzivat sipku doprava pro posun k nasledujici zarazce tabulatoru, sipku doleva pro posun o jeden znak doleva a DELETE na vymazavani znaku. Po stlaceni ENTER v tomto rezimu se cely proces zopakuje, t.j. znovu se nastavi novy prazdny radek a v jeho prvnim sloupci se objevi neblikajici kurzor. Po napsani posledniho radku, ktery jsme chteli vlozit, nestlacime ENTER, ale soucasne obe tlacitka SHIFT. Kurzor zacne blikat a jsme opet v rezimu NORMAL EDIT.

Kopie radku - (Copy line) C

Stiskem C se oznaci radek, ktery chceme okopirovat. Blikajici kurzor bude dale fungovat normalne, ale znacka radku zustane, dokud se neskonci kopirovani.

Presun radku - (Move line) M

Stiskem M se oznaci radek, ktery chceme premistit.

Presun/opis sem - (Move/copy to here) H

Kopirovani nebo premistovani nutno ukoncit nastavenim blikajiciho kurzoru na radek, za ktery chceme predtim urceny radek prekopirovat resp. premistit. Abychom nezapomneli, ze jsme otevreli a neuzavreli kopirovani nebo presun, pripomina nam to ctverecek v levem hornim rohu obrazovky. Zacatou operaci vsak muzeme zrusit stiskem ENTER v rezimu LINE COMMAND (viz dale).

Radek nelze prekopirovat nebo presunout na uplny zacatek text-bufru. Kdyz potrebujeme takovou operaci provest, musime radek umistit bezprostredne za prvni radek a potom puvodni prvni radek premistit az za nej. (Jednodusseji se to provede tak, ze si prvni radek nechavame vzdy prazdny pro podobne pripady - pozn. prekl.)

Nastav znacku bloku - (Set block marker) B

Podobne jako s jednotlivymi radky, muzeme manipulovat i s celymi bloky - t.j. s vice radky najednou. Blok se zvoli tak, ze pomoci B oznacime jeho prvni, nebo posledni radek. Kdyz v dalsim pouzijeme pri nekterem radku C,M nebo D, bude se tento prikaz tykat celeho bloku, ohraniceneho na jedne strane znackou (B) a na druhe strane radkem, kde jsme zadali C,M nebo D (vcetne). Operaci nutno opet ukoncit stiskem H na tom miste, kam chceme blok prekopirovat nebo premistit (netyka se to prikazu D). Toto misto musi byt pochopitelne mimo definovany blok, jinak dostaneme chybovou zpravu "INVALID REQUEST" (neplatny pozadavek).

Odstran znacku bloku - (Remove block markers) ENTER

Zapocatou operaci, jakoz i oznaceni bloku muzeme zrusit stlacenim ENTER v rezimu LINE COMMAND.

Zobraz zacatek - (Display from top) T

Stiskem T se na obrazovce objevi prvnich 24 radku text-bufru.

Zobraz konec - (Display end) E

Stisknutim E se zobrazi posledni radek text-bufru.

Opakuj hledani - (Repeat find) F

Stisknutim F se zopakuje posledni prikaz hledani (viz rezim EXTENDED COMMAND PROCESSOR).

Vyvolej rezim EXTENDED COMMAND PROCESSOR SPACE

REZIM ROZSIRENYCH PRIKAZU (EXTENDED COMMAND PROCESSOR)

Stiskem obou tlacitek SHIFT soucasne, s naslednym stiskem SPACE vstoupime do rezimu rozsirenych prikazu-povelu (oznacujeme jej zkratkou EXT.COM.PROC.) V tomto rezimu se v dolnim radku obrazovky objevi vyzva, abychom napsali prikaz. Prikaz EXT.COM.PROC. se sklada ze slova (vetsinou jednopismenoveho) a z argumentu. Prikaz se provede po stisku ENTER. Muzeme pouzit tyto prikazy:

Precislovani radku - (Renumber) N start,inc

Prikazem muzeme precislovat radky v text-bufru. Prvni radek bude mit v novem cislovani cislo 'start', druhy 'start+inc' atd. Kdyz vynechame nektery (nebo oba) z argumentu, pouziji se naposledy definovane hodnoty - na zacatku to jsou 10,10. I kdyz cisla radku nemaji vliv na editovaci proces, doporucuje se precislovat radky pred assemblovanim (prekladem) - kvuli jednodussimu vyhledavani chyb.

Skok na radek - (Goto line) G number

Zobrazi obsah text-bufru pocinajic radkem s cislem 'number'. Pozor! Text bufr se pritom neprohledava od zacatku, ale od radku, ktery je prave zobrazen jako horni. V pripade neuspesneho hledani se zobrazi chybove hlaseni "NOT FOUND" (nenalezeno).

Najdi retezec znaku - (Find character string) F string

Prohledava text-bufr na vyskyt definovaneho retezce, ale opet pocinajic hornim zobrazovanym radkem. Radek obsahujici hledany retezec se pri uspesnem hledani zobrazi v horni casti obrazovky, jinak se vypise chybove hlaseni "NOT FOUND". Nepouzivat zadne uvozovky! Vsechny znaky (maximalne 20) nasledujici po pismenu F jsou vyznamne! Tento prikaz muzete vyhodne kombinovat s radkovym prikazem F, ktery zacne prohledavat text-bufr od druheho radku na obrazovce s tim samym argumentem, jako posledne pouzity prikaz F (EXT.COM.PROC.).

Znovu je nutne upozornit, ze neni rozdilu mezi velkymi a malymi pismeny, proto napr. prikazem Fnop vyhledame vsechny vyskyty retezcu nop, Nop, NoP, nOP, atd.

Zamen retezec znaku - (Change character string) C string

Timto prikazem se vyhledavaji vsechny vyskyty retezce, definovaneho predtim prikazem F (EXT.COM.PROC.) a kazdy z nich se nahradi novym retezcem 'string'. Hledani opet zacina od prvniho zobrazovaneho (horniho) radku. Nejsou potrebne zadne uvozovky, vyznamnych je prvnich 20 znaku po pismenu C. Kdyz vlozime samotne pismeno C (bez argumentu), potom vsechny v F definovane retezce budou z text-bufru vymazany.

Zobraz volnou pamet - (Display free memory) MEM

Prikazem se zobrazi delka volne pameti (decimalne) a pocatecni adresa volne oblasti (hexadecimalne). Volna pamet zahrnuje i oblast, kterou momentalne zabira tabulka symbolu, nebot i tato je dostupna text-bufru. Do rezimu NORMAL EDIT se vratime stiskem libovolneho tlacitka.

Navrat do BASICu - (Return to BASIC) BASIC

Timto prikazem se muzeme vratit do zakladniho operacniho systemu. Restart se potom provede temito prikazy :

```
RANDOMIZE USR cold  . . . . studeny start
RANDOMIZE USR warm  . . . . teply start
```

Studeny start vycisti text-buffr, zatimco teply start uchova jeho puvodni obsah.

Vymazani bufru - (Clear buffer) CLEAR

Prikaz vycisti text-buffr a vlozi do nej prazdny radek s cislem 00000; je tedy ekvivalentni studenemu startu. Pouzitim tohoto prikazu se dale vsechny symboly v tabulce symbolu deklaruji jako chranene proti vymazani (viz prislusny prepinac).

Volani monitoru - (Call monitor) MCTT

Tento prikaz se tyka jen verze pro 48K. Vola program OCP MACHINE CODE TEST TOOL, pokud byl tento predem naloadovany. V jinem pripade oznami "INVALID REQUEST". V programu MCTT je mozne pouzit prikaz S na navrat do E/A. (Poznamka pro uzivatele MCTT: Namisto pouzivani breakpointu v MCTT [prikaz B] je mozne do zdrojoveho textu vlozit instrukci CALL 0F8C6H, ktera pri behu programu, spusteneho prikazem RUN (viz dale), vyvola preruşovaci rutinu z MCTT. Avsak pozor! MCTT musi byt predtim aktivovan aspon jednim explicitnim volanim pomoci prikazu MCTT z E/A.)

Volani uzivatelskeho programu - (Call user program) RUN

Prikaz provede program, ktery se nachazi v object-code-buffru. Prikaz nezahrnuje zadne testovani na platnost instrukci nachazejicich se v teto oblasti pameti, proto je pri jeho pouzivani nutna zvysena opatrnost. Do E/A se muzete vratit instrukci RET ve zdrojovem programu. Doporucuje se pouzivat tento prikaz jen bezprostredne po assemblovani "do pamati", protoze dalsi pripadne editovaci operace by mohly narusit obsah object-code-buffru.

Tisk - (Print) P number

Vytiskne 'number' radku text-buffru na tiskarne. Listing zacina hornim zobrazovanym radkem. Kdyz vynechame parametr 'number', udavajici pocet radku, vytiskne se cely obsah text-buffru. BREAKem (CAPS SHIFT + SPACE) je mozne se kdykoliv vratit do rezimu NORMAL EDIT.

Ulozeni na pasek - (SAVE to tape) S filename

Prikaz se pouziva na ulozeni obsahu text-buffru, pricemz jako jmeno souboru se pouzije parametr 'filename'. Zduraznujeme, ze ani v tomto pripade se nepouzivaji uvozovky a ze vyznamnych je 10 znaku nasledujucich za pismenem S. Pokud se neuvede jmeno souboru, pouzije se standartne jmeno "SourceCode". Saveovani probiha stejne jako z BASICu.

Natazeni z pasky - (LOAD from tape) L filename

Prikaz loaduje z pasky do text-buffru, pricemz prepisuje jeho predchazejici obsah. Pri absenci jmena souboru se nahrava prvni soubor, ktery nasleduje na pasce. Kdyz velikost nahravaneho souboru presahuje pamet, vyda se chybova zprava "TOO BIG" (prilis velke), loadovani neprobehne a zachova se puvodni obsah text-buffru. V tomto pripade se muze stlacenim libovolneho tlacitka vratit do rezimu NORMAL EDIT. Pokud se pri loadovani vyskytne chyba, vyda se zprava "TAPE ERROR" (chyba pasky). Libovolnym tlacitkem se vratime do rezimu NORMAL EDIT. Po uspesnem loadovani se na obrazovce objevi prvni stranka text-buffru a nastavi se rezim NORMAL EDIT.

Pridavani z pasky - (Append from tape) X filename

Prikaz je podobny prikazu L, ale nahravany text neprepisuje puvodni obsah text-buffru, jen se pridava na jeho konec. (Obdoba prikazu MERGE v BASICu, avsak v tomto pripade se nezohlednuji cisla radku! - pozn. prekl.)

Verifikace pasky - (VERIFY from tape) V filename

Prikazem je mozne zkontrolovat spravnost nahravky, a to bud obsahu text-buffru, nebo object-code-buffru (viz dale). Kdyz verifikace selze, objevi se chybova zprava "TAPE ERROR", ze ktere se stlacenim libovolneho tlacitka dostaneme do rezimu NORMAL EDIT.

Preklad text buffru - (Assemblace) A filename/s1/s2/s3...

Prikazem se provede preklad (assemblace) zdrojoveho textu ulozeneho v text-buffru, pricemz se pouziji prepinace (klice prekladace) s1, s2, s3... Kdyz se jako vysledek prekladu zhotovuje nahravka na pasku (object-code-tape), potom soubor ma nazev 'filename'. Pri absenci jmena souboru se pouzije standartni nazev "ObjectCode". Prikaz assemblace (prekladu zdrojoveho textu do strojoveho kodu) bude podrobneji popsan v nasledujici casti, kde uvedeme i mozne prepinace (klice).

4. ASSEMBLER

Jak jsme jiz uvedli, assembler se aktivizuje prikazem A (EXT,COM,PROC.). Prepinace jsou dvoupismenove prikazy, ktere blize specifikuji zpusob prekladu. Oddeluji se lomitkem (/) a muzete je v prikaze A uvadet v libovolnem poradi v libovolnem poctu.

Assembler pracuje na principu dvou pruchodu. Pri prvnim pruchodu se kontroluje syntakticka spravnost zdrojoveho textu a vytvari se tabulka symbolu. Behem prvniho pruchodu je obrazovka cista. Pri druhem pruchodu (prekladu) se mnemokody prekladaji do vysledneho tvaru strojoveho kodu a vycisluji se operandy. Tyto informace se zaroven zobrazuji. Pomoci BREAKu (CAPS SHIFT + SPACE) muzete proces prekladu kdykoliv zrusit . Krome toho, pri druhem pruchodu je mozne preklad (assemblaci) pozastavit libovolnym tlacitkem (krome BREAK) a podobne ho znovu nechat pokracovat.

ASSEMBLEROVSKY LISTING se sklada ze tri casti. Prava strana predstavuje zdrojovy text. Na leve strane jsou ve dvou sloupcich zobrazeny adresa a samotny cilovy kod (object-code). Obe informace jsou v hexadecimalnim tvaru. Chybove zpravy se vzdy tykaji radku, ktery jim predchazi. Nakonec se vyda zprava o celkovem poctu nalezenych chyb.

Assembler uklada cilovy kod, ktery produkuje, do oblasti pameti, kterou nazyvame object-code-buffr. Obycejne je to oblast pameti nasledujici bezprostredne za text-buffrem. To vsak zpravidla nebyva ta oblast, kterou programator predurcil pro beh ciloveho kodu. Kdyz vsak object-code-buffr nahrajeme na pasku, informace se ulozi v takove forme, ze po nahrati z BASICu (pomoci LOAD "" CODE) se ulozi na spravne predepsane adresy, kde je potom mozne program spustit. Object-code se zapisuje na pasku po druhem pruchodu. Objevi se bezna zprava "Start tape..." a nutno si pocinat jako pri beznem saveovani.

SYMBOLICKE NAZVY jsou retezce o maximalni delce 6 znaku. Prvnim znakem musi byt pismeno, potom mohou nasledovat pismena (A az Z), cislice (0 az 9), nebo nektery ze specialnich znaku (# . ? @ % $ _). Mnemonicke nazvy registru a registrovych paru neni mozne pouzivat jako symboly s jinym vyznamem. Opet neexistuje rozdil mezi velkymi a malymi pismeny. Na konci assemblerovskeho listingu se zobrazi tabulka symbolu. Pismenove znacky maji nasledujici vyznam :

L oznacuje symbol typu DEFL;
M oznacuje vicenasobne pouzity symbol (viz cast o chybach);
U oznacuje symbol s neznamou hodnotou.

ORGANIZACE PAMETI

	BASIC ROM
	SCREEN + ATRIBUTY
	SYSTEMOVE PROMENNE atd.
	BASIC PROGRAM
5EEDH	EDITOR/ASSEMBLER
89EDH	TEXT BUFFR
	OBJECT CODE BUFFR
	VOLNA OBLAST
	TABULKA SYMBOLU
F780H	MCTT (pokud se pouziva)

VSTUPNI ADRESY :

EDITOR/ASSEMBLER	warm = 24304	(teply start)
	cold = 24301	(studeny start)
SAMOSTATNY EDITOR	warm = 24204	(teply start)
	cold = 24201	(studeny start)
SAMOSTATNY ASSEMBLER	24201	

ASSEMBLEROVSKE PREPINACE (ASSEMBLER SWITCHES)

(Klice prekladace)

Nebude ukladat na pasek - (No Object tape switch) /NO

Klic potlaci tvorbu cilove pasky na konci prekladu. (Nenahraje strojni kod na pasek). Doporucuje se pouzivat ve fazi ladeni.

Zadna tabulka symbolu - (No Symbol table listing) /NS

Prepinac (klic) potlaci vypis (tisk) tabulky symbolu.

Zadny assemblerovsky listing - (No assembly listing) /NL

Klic potlaci listovani v prubehu prace assembleru, ale radky obsahujici chyby se i presto zobrazi. Kdyz soucasne pouzijeme prepinace /NS a /NL, na konci prekladu se zobrazi jen zprava o poctu chyb. To je nejrychlejsi zpusob assemblace (prekladu do strojniho kodu).

Vystup ass. listingu a tabulky symbolu na tiskarnu /LP

Prepinac umoznuje vytisknout protokol o prekladu.

Cekej pri chybach - (Wait on Errors switch) /WE

Pri pouziti tohoto klice se preklad prerusi vzdy, kdyz se narazi na chybu. Stlaceni libovolneho tlacitka (krome C a BREAK) vyvola pokracovani prekladu. Kdyz stlacime C, vypneme prepinac /WE a preklad bude dale pokracovat bez zastavek. BREAK rusi cely proces prekladu.

Preklad do pameti - (Assemble Into Memory switch) /IM

Vytvareny cilovy kod se pri pouziti tohoto klice muze spustit prikazem RUN (EXT.COM.PROC.). Pritom neni nutno pouzit direktivu ORG. Klic /IM implikuje klic /NO. O tom, kam byl cilovy kod ulozen, se muzete presvedcit pomoci prikazu MEM (EXT.COM.PROC.). Pri pouziti direktivy ORG se object-code ulozi primo na definovane adresy.

Preklad s absolutnimi adresami /AO

Pouziti tohoto klice vyvola tvorbu cilove pasky. V tomto pripade je treba pouzivat direktivy ORG. Cilovou pasku mozno nahrat z BASICu pouzitim prikazu LOAD "" CODE. Tento klic je standartni, t.j. prikaz pro preklad A/AO je ekvivalentni jednoduchemu A (vsechno EXT.COM.PROC.) Vseobecne vsak cilovy kod, generovany pri pouziti /AO, nelze spustit, kdyz je umistnen v object-code-buffru.

Preklad s pouzitim stare tabulky symbolu /OS

Tento klic souvisi s pouzitim prikazu CLEAR (EXT.COM.PROC.). Ma vyznam predevsim pro verzi 16K, ale uplatni se i ve verzi 48K hlavne pri delsich programech. Kdyz je tento klic neaktivni, preklad zacina s prazdnou tabulkou symbolu. Pri pouziti /OS se vsak vsechny symboly ve stare tabulce, ktere byly deklarovany jako chranene proti vymazani (toto provede prikaz CLEAR), prenesou do nove tabulky. Takto lze vyuzivat symboly, ktere byly definovany v jinych castech programu, ne v te, kterou prave prekladame. Pri prekladu dlouheho programu postupujeme takto:

Rozdelime program na mensi casti, kazdou prelozime samostatne a vytvarime cilove pasky. Poznamename si konecnou adresu jedne casti (nejjednodusseji to provedeme tak, ze direktive END predradime navesti) a preneseme ji do direktivy ORG nasledujici casti. Mezi jednotlivymi castmi pouzivame CLEAR a prekladame s klicem /OS. Nakonec naloadujeme vsechny casti z BASICu a nahrajeme je na jeden soubor pomoci prikazu SAVE "filename" CODE.

V tabulce symbolu se vsak zachovavaji jen ty symboly, ktere jsou chraneny proti vymazani. Bezne symboly - ktere doda prave prekladany program - se vymazou pred kazdym dalsim prekladem. Symboly prenasene z programu do programu si ponechavaji svuj statut - t.j. DEFL-symboly zustanou DEFL-symboly. Kdyz prave prekladany program obsahuje stejnojmenny symbol, hodnota stareho symbolu se rusi.

KONSTANTY

mohou byt techto typu :

a/ SYMBOLICKE NAVESTI

b/ POCITADLO - ($) - reprezentuje okamzitou hodnotu assemblerovskeho adresoveho pocitadla - t.j. adresu, do ktere se (logicky) ma umistit nasledujici byte object-codu.

c/ RETEZCE ZNAKU - pouzivame uzavrene mezi apostrofy ('). Vyznamne jsou pritom jen dva znaky, ktere se v retezci vyskytuji nejvice vpravo a tyto se prelozi jako 16-bitove slovo odpovidajici ASCII kodum. Samotny apostrof se vyjadri zdvojenim, napr. LD A,'' znamena LD A,27H , nebot v ASCII apostrofu (') odpovida 27H.

d/ NUMERICKE KONSTANTY lze psat ve tvaru binarnim, oktalovem, decimalnim, nebo hexadecimalnim. Ciselnou soustavu specifikuje suffix (dodatecne pismeno za cislem) :

B	- binarni konstanta
O nebo Q	- oktalova konstanta
D	- decimalni konstanta (suffix nepovinny)
H	- hexadecimalni konstanta.

Kdyz chybi suffix, predpoklada se, ze jde o decimalni konstantu. Pozornost venujte psani hexadecimalnich konstant, nebot tyto nesmi zacinat pismenem! Kdyby mel takovy pripad nastat, je treba pouzit na zacatku nulu - napr. namisto FFH musime psat 0FFH. Kdybychom napsali FFH, assembler by to pochopil jako symbol. Vsechny konstanty jsou vnitrne interpretovany jako 16-bitove. Assembler sice akceptuje i cisla vetsi nez 65535, ale pretecena cast se ztraci!

VYCISLOVANI VYRAZU

Assembler obsahuje vykonny evaluator vyrazu. Vyrazy mohou obsahovat nize uvedene operatory, pricemz se zohlednuje jejich priorita. Prioritu je mozne ovlivnit pouzitim zavorek.

OPERATOR	FUNKCE	PRIORITA	
=	rovna se	1	(nejnizsi)
<>	nerovna se	1	
>	vetsi nez	1	
<	mensi nez	1	
>=	vetsi nebo rovna se	1	
<=	mensi nebo rovna se	1	
+	secitani	2	
-	odecitani	2	
OR	log. soucet	3	
!	log. soucet	3	
XOR	vylucny (exkl.)log. soucet	3	
AND	log. soucin	4	
&	log. soucin	4	
*	nasobeni (16-ti bitove)	5	
/	deleni (celociselne)	5	
MOD	modulo (celocis. zbytek po deleni)	5	
NOT	negace (jednickovy doplnek)	6	
()	zavorky	7	(nejvyssi)

Poznamky k tabulce :

1. Relacni operatory (prvnich sest v tabulce) davaji vysledek -1, kdyz je vyrok pravdivy ; v opacnem pripade je vysledek 0.
2. Logicke operatory funguji jako bitove nad 16-bitovymi slovy.
3. Deleni je celociselne. Nasobeni je 16-bitove a preteceni se nezohlednuje.
4. Modulo znamena zbytek po celociselnem deleni.
5. Operator NOT je unarni a tyka se 16-bitoveho slova.
6. Zavorky je mozne pouzit do libovolne hloubky.
7. Prvnim operatorem ve vyrazu nemuze byt leva zavorka, napr. nespravne by bylo LD A, (4*8)+3. Musi se napsat LD A,3+(4*8)
8. Pokud mozno, pouzivejte mezi operatory mezery, aby nedoslo k chybam, jako napr. ve vyrazu 10101AND0111, kde 'A' ve slovu 'AND' by assembler chapal jako posledni znak konstanty 10101A. Proto je nutne napsat 1010 AND 0111.

PSEUDOINSTRUKCE (zkracene psi)

nejsou skutecne instrukce mikroprocesoru, jsou to direktivy, ktere poskytuji dulezite informace a ridici funkce pro praci samotneho prekladace. E/A pouziva tyto psi :

ORiGin - pocatecni adresa ORG

Nastavuje pocitadlo adres na zvolenou hodnotu. Kdyz ma ORG navesti, pak je tomuto navesti prirazena hodnota pocitadla pred zmenou. (ORG se uvadi na zacatku programu a i prelozeny strojovy kod bude potom nahran na pasku s pocatkem na cisle adresy, jakou ma ORG. Napr. napiseme-li ORG 50000, bude prelozeny strojni kod ulozen od adresy 50000. Jednoduchou zmenou prepsani hodnoty ORG napr. na 33000, bude po novem prekladu zacatek strojniho programu presunut na adresu 33000 atd.)

END - konec zdrojoveho textu END

Oznacuje konec zdrojoveho textu urceneho k prekladu. Jeji pouziti neni zavazne. Pokud se pouzije, zpusobi, ze vsechny instrukce za ni nasledujici, budou ignorovany.

DEFine Byte - definice bytu DEFB, nebo DB

Argumentem je 8-bitova konstanta, nebo vyraz, jehoz hodnota se ulozi na adresu urcenou assemblerovskym pocitadlem adres. Vice techto psi je mozno sloucit v jednu, pricemz jednotlive argumenty oddelujeme carkami. Kdyz je hodnota argumentu vetsi nez 255, nebo mensi nez -256, vyda se chybova zprava "FIELD OVERFLOW" (preteceni pole) a uplatni se jen 8 nejmene vyznamnych bitu.

DEFine Word - definice slova DEFW, nebo DW

Argumentem je 16-bitova hodnota, ktera se do pameti uklada podle ZILOGovske koncepce - mene vyznamny byte se ulozi pred vyznamnejsim byte. Jinak plati totez, co u DEFB.

DEFine Message - definice zpravy (textu) DEFM, nebo DM

Argumentem je retezec ASCII znaku, ktery se interpretuje a uklada do pameti jako posloupnost jejich ASCII kodu (CODE). Retezec musi byt uzavren mezi apostrofy ('), samotny apostrof se vyjadri jeho zdvojenim.

DEFine Storage - definice pametove oblasti DEFS, nebo DS

Rezervuje v pameti urcity pocet bytu, ktery udava jeji argument. V podstate zpusobi zvyseni assemblerovskeho pocitadla adres o hodnotu danou argumentem. Kdyz pouzijeme navesti, bude mu prirazena hodnota pocitadla pred zvysenim.

DEFine Label-definice navesti,hodnoty symbolu DEFL,nebo DL:EQU

Psi DEFL a EQU negeneruji zadny object-code, ale pouzivaji se na prirazeni hodnot symbolum. Vzdy musi mit navesti a hodnotu jako argument. Rozdil mezi EQU a DEFL je v tom, ze v programu lze pomoci DEFL priradit temuz symbolu postupne i vice hodnot, kdezto hodnota symbolu deklarovaneho pomoci EQU musi byt v celem

programu nemenna. V assemblerovskem listingu se u techto psi zobrazi v prvnim sloupci hodnota prirazena symbolu, ne adresa dana pocitadlem. Kdyz byl symbol deklarovan pomoci DEFL, lze mu v dalsi casti programu zmenit hodnotu pomoci dalsiho zadani DEFL, nebo EQU, avsak nelze jej jiz pouzit jako navesti (navesti musi mit nemennou hodnotu). V tabulce symbolu se zobrazi jen posledni hodnota prirazeni symbolu.

CHYBOVA HLASENI (zpravy)

Chyby pri prekladu lze vseobecne rozdelit podle zavaznosti na tri kategorie : katastroficke, fatalni a nefatalni.

CHYBY KATASTROFICKE spocivaji v preteceni object-code buffru a v preteceni tabulky symbolu. V obou pripadech se preklad okamzite prerusi a vypise se chybova zprava. Stiskem libovolneho tlacitka se dostaneme do rezimu NORMAL EDIT. Obe chyby spocivaji v nedostatecne kapacite pameti a musi se resit zkracenim zdrojoveho textu (napr. odstranenim poznamek, zkracenim nazvu symbolu), nebo rozdelenim programu na casti.

CHYBY FATALNI vedou k modifikaci object-codu vzhledem ke zdrojovemu textu. Pri fatalni chybe assembler nepochopil instrukci, resp. ztratila se klicova informace pro tvorbu ciloveho kodu. Nefatalni chyby nejsou natolik zavazne - cilovy kod se sice vygeneruje, ale nektere casti (napr. argumenty) se nastavi na nulu, resp. v pripade chyby "FIELD OVERFLOW" se zohledni jen nejmene vyznamne bity. Pokud je to mozne, chybne misto je indikovano malou sipkou. Kdyz radek obsahuje vice chyb, assembler upozorni jen na tu nejzavaznejsi z nich.

Prehled chybovych zprav a jejich pricin :

BAD LABEL - nefatalni chyba. Chybny nazev symbolu, nebo navesti. Viz zasady pro tvormu jmen.

MULTIPLE DEFINITION -nefatalni chyba. Totez navesti (symbol) bylo pouzito vicekrat s ruznymi hodnotami. Assembler pouzije hodnotu, ktera byla deklarovana jako prvni.

BAD OPCODE - fatalni chyba. Assembler nerozeznal mnemokod instrukce.

UNDEFINED SYMBOL - nefatalni chyba. Vyskytl se symbol, ktery nemel prirazenou hodnotu. Pouzit EQU, nebo DEFL.

MULTIPLE DEFINED SYMBOL - nefatalni chyba. Vyraz obsahuje vicenasobne definovany symbol.

DIVISION BY ZERO - nefatalni chyba. Nedovolene deleni nulou.

FIELD OVERFLOW - nefatalni chyba. Operand instrukce pretekl pres rozsah, ktery je pro danou instrukci stanoven. Napr. byl pouzit 16-bitovy operand misto 8-bitoveho.

BRANCH OUT OF RANGE - nefatalni chyba. Relativni adresa v instrukcich JR nebo DJNZ byla vetsi nez +127, nebo mensi nez -128. Nutno zkratit program, nebo pouzit instrukci skoku JP.

MISSING INFORMATION - fatalni chyba. Chybi operandy, nebo je radek nejak jinak nekompletni.

BAD ADRESSING MODE - fatalni chyba. Byl pouzit nespravny zpusob adresovani.Nutno venovat pozornost souboru instrukci

BAD EXPRESSION - fatalni, nebo nefatalni chyba. Operand nebo vyraz nema spravnou syntaxi.

OCP EDITOR/ASSEMBLER + MONS 3

(Prehled prikazu)

Umoznuje tvorbu a preklad programu v mnemonice Z-80. Je ulozen v pameti od adresy 24301 a zabira asi 11 kB. Pro studeny start se zada prikaz 'RANDOMIZE USR cold', pro teply 'RANDOMIZE USR warm'. Po nahrati se program nachazi v rezimu editoru.Z neho je mozne prejit do rezimu radkovych povelu a z nej zase do rezimu rozsirenych povelu.

NUMERICKE KONSTANTY mohou byt zadavany :

H - hexadecimalne
D - decimalne
B - binarne
O,Q - osmickove
$ - okamzita hodnota PC

OPERATORY:

=,<,>,<=,>=,<>,
+,-
OR,! - log OR
XOR - log XOR
AND,&- log AND
* - 16-bitove nasobeni
/ - celociselne deleni
MOD - celociselny zbytek po deleni
NOT - negace
() - zavorky, nesmi byt 1. operator ve vyrazu

PRIORITA:

1 - nejnizsi
2
3
3
4
5
5
6
6
7 - nejvyssi

PSEUDOINSTRUKCE :

ORG - nastavi citac adres PC
EQU - priradi hodnotu navesti
END - konec prekladu
DEFB,DB - definuje 1-bytove hodnoty, napr. DB 10,0C6H,'a'
DEFW,DW - definuje 2-bytove hodnoty, napr. DW 998,5C3AH
DEFM,DM - definuje retezec ASCII, napr. DM 'Hallo'
DEFS,DS - zvysi PC o danou hodnotu, rezervuje pamet
DEFL,DL - umoznuje vicekrat priradit hodnotu stejnemu navesti

PRIKAZY

TEXTOVY EDITOR:

ENTER- tabulator
CS+1 - zruseni oprav v radku
CS+2 - uzamknuti velkych pismen
CS+3 - strankovani nahoru
CS+4 - strankovani dolu
CS+5,6,7,8 - kurzor
CS+9 - vkladani znaku do radku (ENTER=navrat)
CS+0 - mazani znaku
CS+ENTER - novy radek
CS+SS - prechod do rezimu radkovych povelu

REZIM RADKOVYCH POVELU :

D - vypusteni radku, nebo vymazani bloku (viz prikaz B)
I - vlozeni radku, v tomto rezimu se pise program, CS+SS=navrat
B - nastaveni znacky 'zacatek bloku' pro prikazy D,C,M
C - nastaveni znacky 'konec bloku' pro kopirovani
M - nastaveni znacky 'konec bloku' pro presun
H - vykonani operace presunu, kopirovani od pozice kurzoru
T - zobrazeni prvni stranky, kurzor na prvni radek
E - kurzor na posledni radek
ENTER - navrat z rezimu presunu (rusi blokove znacky)
F - opakovani hledani retezce (viz povel 'F' v rezimu rozs.p.)
SPACE - prechod do rezimu rozsirenych povelu

REZIM ROZSIRENYCH POVELU :

N pocatek,prirustek - precislovani radku
G cislo radku - od 1. radku obrazovky hleda dany radek, na ktery potom nastavi kurzor
F retezec - vyhledavani retezce pocinaje prvnim radkem obrazovky
C retezec - zmena retezce po 'F' (od 1. radku obrazovky)
P pocet - od 1. radku obrazovky vytiskne na tiskarne dany pocet radku, kdyz pocet neni uveden, nebo je=0, vytiskne se cely text. Vypis se da prerusit BREAKem.
MEM - vypise pocet bytu volne pameti pro editor a hexa adresu prvni volne pam. bunky
CLEAR - vymazani textoveho buffru (tabulka symbolu se nemaze)
MCTT - volani ladiciho modulu (pokud je pritomen)
RUN - spusteni binarniho programu, pokud byl prelozen bez ORG a pomoci klice 'IM'. Navrat je instrukci RET.
S jmeno-ulozeni text. buffru na pasku s danym jmenem
L jmeno-nahraje text. buffr z pasky
V jmeno-verifikuje dany soubor
X jmeno-prihraje dalsi text za text. buffr
A jmeno/K1/K2/.../K8 - preklad text. buffru s danymi klici

KLICE PREKLADACE:

/NO - preklad bez ulozeni binarniho kodu na pasku
/NS - preklad bez vypisu tabulky symbolu
/NL - preklad bez vypisu protokolu o prekladu (krome chyb)
/AO - preklad od adresy urcene pseudoinstrukci ORG
/WE - zastaveni vypisu pri vyskytu chyby ('C' dezaktivuje klic)
/IM - ulozeni absolutniho kodu do pameti (aktivuje i klic /NO)
/OS - preklad s tabulkou symbolu z predchazejiciho programu
/LP - nasmerovani vypisu prekladu na tiskarnu

MONS 3

MONS 3 opetovne spustime od adresy o 29 vetsi, nez byla startovaci. Kdekoliv se pouziva klavesa ENTER k ukonceni hexadecimalniho cisla, lze pouzit libovolny nehexadecimalni znak. Je-li koncovym znakem znamenko minus, pak po vlozeni dostaneme zapornou hodnotu vlozeneho cisla.
MONS 3 blokuje preruseni.

PRIKAZY MONS 3

ENTER- pricita jednicku k pametovemu ukazateli
CS+7 - odecita jednicku od pametoveho ukazatele
CS+8 - pricita osm k pametovemu ukazateli
CS+5 - odecita osm od pametoveho ukazatele
M - nastavi pametovy ukazatel na zadanou adresu
. - nastavi ukazatel na nasledujici dvojici registru. Jestlize se predtim uvede hexa cislo, zmeni se dvojice registru
SS+Z - krokovani. Pred pouzitim prikazu musi byt jak citac adres tak i pametovy ukazatel nastaven na stejnou adresu.
SS+T - nastavi bod preruseni za instrukci a pokracuje v programu
W - nastavi bod preruseni do pametoveho ukazatele.Muze se zadat i vice bodu preruseni. Ty se zrusi po prikazu 'T'.
J - provadeni programu od dane adresy. Pred ukoncenim adresy lze prikaz zrusit CS+5. Nici obsah registru.
SS+K - provadi program od adresy v citaci PC. Uchovava registry.
SS+4 - dekompiluje stranku od adresy v pam. ukazateli.SS+4 vraci zobrazeni. Jinou klavesou si vyzadame dalsi stranku.
L - vypis 80 bytu s ASCII ekvivalenty od adresy v ukazateli. CS+5 vraci zobrazeni, jina klavesa - dalsi stranka.
SS+P - stejny prikaz jako 'L', jen vystup smeruje na tiskarnu
H - prevod dekadickeho cisla na jeho hexadecimalni ekvivalent
SS+3 - meni zaklad ciselne soustavy mezi 16 a 10
Y - vlozeni ASCII znaku do pameti od hodnoty ukazatele. Retezec se ukonci SS+5 a DELETE lze pouzit k vypusteni znaku.
G - vyhledani retezce v pameti ("G" a retezec)
N - hleda dalsi vyskyt hex. retezce zadaneho prikazem 'G'
, - vlozi do pam. ukazatele adresu z SP
X - prechod na misto urceni abs. adresy instrukci CALL, JP
V - zobrazi pamet na misto, kde byl zadan posledni prikaz 'X'
O - prechod na misto urceni relativni adresy
U - zobrazi pamet na misto, kde byl zadan posledni prikaz 'O'
I - kopie bloku v pameti z jedne pozice na jinou. Zkopiruje blok i na misto, kde prekryje svou puvodni polohu.
P - naplni pamet mezi zadanymi hranicemi urcenym bytem
Q - zmena sady registru
T - vlozi se pocatecni (First:) a koncova (Last:) adresa dekompilovaneho programu. Po stisku 'Y' na naznak 'Printer' se nasmeruje vypis na tiskarnu. Po stisku ENTER na naznak 'Text' se negeneruje zdrojovy text. Pote se uda pocatecni a koncova adresa datove oblasti.Vypis se zastavi stiskem ENTER nebo SPACE, pak se lze CS+5 vratit k celnimu panelu
Vlozime-li na naznak 'Text' hexa adresu (danou prikazem editoru 'X'), generuje se zdr. text pro GENS 3.Naznak 'Workspace' ocekava adresu volne pameti pro tabulku symbolu.Po dekompilaci vlozime adresu 'End of text' do mista TEXTEND v GENS 3.Pak vstoupime do GENS 3 teplym startem.

```
**************************************************
*                                                *
*  DODATKY K EDIT ASSEMBLERU verse OCP GD 100    *
*         (pouziva MONS 3, misto MCTT)           *
*                                                *
**************************************************
```

---

1. V teto versi jsou odstraneny chyby, ktere vznikaly pri prekladu instrukci vyuzivajicich IX a IY registry.
2. Bylo odstraneno zakmitavani klavesnice.
3. Tisk je predelan na tisk pres IF"Soldan" pres port A
4. Assembler je schopen zpracovat i texty napsane v GENS 3 a tedy i texty vytvoreny MONSem - to je vyborna moznost !

ROZSIRENE PRIKAZY PRO OCP GD 100:

GENS L - nahraje z pasky zdrojovy text vytvoreny v GENS2 a prelozi jej do tvaru OCP.
GENS X - prihraje z pasky text vytvoreny v GENS2 za text ulozeny jiz v pameti a prelozi jej do tvaru OCP.
GENS M - prelozi text v pameti vytvoreny pomoci MONS3 (ulozeny za bufferem OCP) do tvaru OCP. Pak je nutno precislovat radky.
CLEARX - zrusi prihravany text pri "TAPE ERROR".
CLEARB - navrat do Basicu.
MONS - vstup na MONS3

ROZSIRENE PRIKAZY PRO MONS :

True video - zrusi ochranu MONSu
Inv. video - nastavi ochranu MONSu
R - navrat z MONS3 do OCP

Pri vytvareni zdrojoveho textu "T" pri otazce TEXT: se musi odpovedet 1. Text bude ulozen za buffer OCP. Potom je nutno v OCP pouzit prikaz GENS M pro preklad textu do tvaru OCP a precislovat radky.

Natisteno pro klub vypocetni techniky pri ZO SVAZARMU Karolinka

---===K O N E C===---