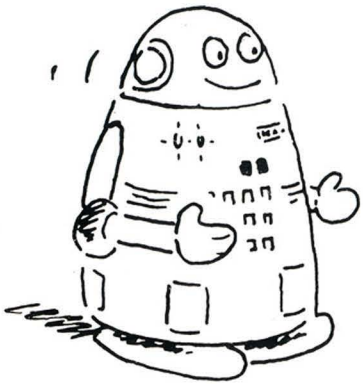

INTEGROVANÝ SYSTÉM PRO PROGRAMOVÁNÍ
NA ÚROVNI JAZYKA SYMBOLICKÝCH ADRES
PRO MIKROPOČÍTAČ

ONDRA = EDTASM '87

- Uživatelská příručka -

EDTASM je systém pro programování na úrovni jazyka symbolických adres - assembleru. EDTASM se skládá z těchto tří vzájemně spolupracujících částí:

1. EDITOR
2. ASSEMBLER
3. Z-BUG

EDITOR je řádkově orientovaný textový editor sloužící pro přípravu zdrojového textu pro ASSEMBLER.

ASSEMBLER je překladač jazyka symbolických adres v mnemonice mikroprocesoru Z-8Ø. Množství pseudooperací včetně podmíněného překladu a MACRO usnadňuje a zefektivňuje práci programátora.

Z-BUG je symbolický ladicí prostředek obsahující symbolický zpětný překladač (disassembler). Umožňuje krokování programu nastavení bodů zastavení, změnu obsahu registrů, výpis programu v různých formátech, obsahuje celočíselnou aritmetiku a mnoho dalších potřebných funkcí.

Popis funkcí editoru

V následujícím textu znamená klávesa **NEW LINE** klávesu se zalomenou šipkou dolů a doleva (odeslání řádky). Klávesa **ESCAPE** znamená stejnou klávesu, ale stisknutou současně s přeřaďovačem (SHIFT).

Po spuštění EDTASMu se vypíše úvodní text a spustí se EDITOR. EDITOR se přihlásí znakem * na začátku řádku a je připraven plnit příkazy. Příkazy jsou tvořeny jedním písmenem nebo znakem, za kterým mohou následovat parametry. Příkaz se provede až po stisku klávesy NEW LINE.

V parametrech příkazů je možno pro určení řádku nebo skupiny řádků použít těchto prvků:

číslo řádku	čísla řádků mohou být v rozsahu 1..65ØØ
:	označení skupiny řádek. Před dvojtečkou je uveden počáteční řádek, za dvojtečkou koncový řádek.
#	znak # znamená první řádek textu.

*	znak * znamená poslední řádek textu.
.	tečka znamená aktuální (naposledy vypsaný) řádek.
+	dopředný offset - označuje řádek, který se nachází daný počet řádků za uvedeným řádkem. Před plusem je uveden řádek, od kterého se odpočítá několik řádků, jejichž počet je uveden za plusem.
-	zpětný offset - označuje řádek, který se nachází daný počet řádek před uvedeným řádkem. Formát je stejný jako u dopředného offsetu.
!	označuje skupinu daného počtu řádků, počínaje uvedeným řádkem. Před vykřičníkem je uveden první řádek skupiny, za vykřičníkem počet řádků.
%	procento označuje celý text.

Význam a použití těchto prvků nejlépe osvětlí několik příkladů:

100	číslo řádku
200:500	skupina řádků od řádku 200 do řádku 500
#	první řádek textu
*	poslední řádek textu
#:930	skupina řádků od prvního řádku textu do řádku 930
.	aktuální řádek
.-5	pátý řádek před naposledy vypsaným řádkem
500+3	třetí řádek za řádkem 500
!200-2	druhý řádek před řádkem 1200
#+4	čtvrtý řádek za prvním řádkem textu (celkově pátý)
*-1	předposlední řádek textu
%	celý text (#:*)
500!5	5 řádků od řádku 500
#!3	první tři řádky textu
-9:	posledních deset řádků textu
100-10!10	deset řádků před řádkem 100
.-16!16	předchozí stránka

V dalším textu následuje popis jednotlivých příkazů EDITORu. Za písmenem označujícím příkaz jsou uvedeny parametry, které za tímto příkazem následují. Nepovinné parametry (nemusí být za příkazem uvedeny) jsou uzavřeny do složených závorek.

A ASSEMBLER

Příkazem A se spustí ASSEMBLER, který bude překládat text připravený v paměti pomocí EDITORu. Popis parametrů je v dalším textu u popisu ASSEMBLERu.

B BYE

Ukončení práce EDTASMu a návrat do operačního systému. Pokud text v paměti ve stávající podobě nebyl uložen do souboru, vypíše se "File isn't SAVED! Are you sure (Y/N)?". Skok do operačního systému se provede pouze při potvrzení klávesou Y. Tato funkce je pomůcka pro programátora, aby před ukončením práce nezapomněl zapsat aktuální verzi textu do souboru.

Opětovné spuštění EDTASMu lze docílit podržením šipky vlevo a současným stisknutím tlačítka RESET (RESTART programu).

C kam,co{,krok} COPY

Provede zkopírování části textu "co" na místo "kam". Na původním místě zůstane text zachován. "krok" je krokem číslování vsouvaných řádků na novém místě. Většinou je potřeba specifikovat krok 1. Pokud by při číslování vsouvaných řádků vzniklo číslo větší než je číslo následujícího řádku, vypíše se "No room between lines" a příkaz se neprovede. V takovém případě přečíslujte celý text pomocí příkazu N s větším krokem.

př. C 200,100:130,1

- za řádek 200 zkopíruje řádky 100 až 130 s krokem číslování 1.

D {co} DELETE

Smaže část textu v rozsahu určeném parametrem "co". Bez uvedení parametru smaže pouze aktuální řádek. Jeden řádek je možno zrušit i pouhým napsáním jeho čísla místo příkazu. Zrušené řádky nelze obnovit, je proto vhodné parametry příkazu pečlivě zkontrolovat před odesláním příkazu klávesou NEW LINE.

př. D 300
- smaže řádek 300
D 300:400
- smaže řádky od řádku 300 do řádku 400 (včetně)
D 1230:*
- smaže text od řádku 1230 až do konce
510
- smaže řádek 510 (pokud existoval)
D
- smaže aktuální řádek

E {co} EDIT

Umožňuje opravy řádků v rozsahu "co". Bez uvedení parametru se bude opravovat aktuální řádek.

př.	E100	oprava řádku 100
	E*	oprava všech řádků
	E	oprava aktuálního řádku

Po zadání příkazu E se vypíše číslo opravovaného řádku a EDITOR očekává speciální editační povely, které jsou tvořeny jedním písmenem nebo znakem. Před povelem lze uvést počet opakování daného povelu. To se provede stisknutím příslušných kláves s čísly. Počet opakování se zadává dekadicky, pokud počet neuvedeme, provede se povel jednou. Možnost opakování povelu je

naznačena písmenem n před písmenem povelu. Při opravě řádku je tedy možné použít těchto povelů:

nSPÁCE (mezerník) vypsání jednoho nebo více znaků řádku.
nBACKSPACE návrat o jeden nebo více znaků zpět.
NEW_LINE výpis zbytku řádku a ukončení opravy řádku, opravený řádek se uloží do textu v nové podobě.
ESCAPE ukončení módu INSERT (viz. dále).
A AGAIN - všechny změny provedené v textu řádku se zapomenou, vrátí se původní tvar řádku a editaci lze provést znovu. Tento povel je vhodný při chybně provedené opravě řádku.
nC CHANGE - změna uvedeného počtu znaků za jiné. Po vyvolání povelu písmenem C napíšeme příslušný počet nových znaků, které v textu řádku nahradí staré.
nD DELETE - vymazání znaku z textu řádku.
E END - ukončení opravy řádku, zbytek řádku se nevypisuje.
H HACK - vymazání zbytku řádku a přepnutí do módu INSERT - viz. další povel.
I INSERT - přepnutí do módu INSERT. Psané znaky se vkládají na dané místo řádku. Ukončení módu INSERT se provede klávesou ESCAPE.
nKa KILL - výmaz části řádku od aktuální pozice ke znaku "a" (znak napsaný po povelu K). Tento znak se již nemaže.
L LIST - vypíše aktuální stav řádku a nastaví pozici pro opravování řádku na začátek.
Q QUIT - ukončení editace, provedené změny se ignorují, řádek zůstává nezměněn.
nSa SEARCH - vyhledání znaku "a" (znak napsaný po vyvolání povelu S) a nastavení pozice pro opravování na něj.
X EXTEND - vypsání zbytku řádku a přepnutí do módu INSERT. Tento povel je vhodný pro doplnění textu na konec řádku.

F {kde}{$co} FIND
==========

Umožňuje hledání textu "co" v rozsahu řádků "kde". Oddělovačem hledaného textu je klávesa ESCAPE (v tomto textu je místo

ní znak $). Vypíše všechny řádky, které obsahují hledaný text. Poud není žádný výskyt textu nalezen, vypíše se "String not found". Pokud neuvedeme hledaný text, hledá se naposledy zadaný text. Bez uvedení rozsahu pro hledání parametrem "kde" se hledá od aktuálního řádku první výskyt textu. Součástí hledaného textu může být i tabelátor (TAB).

Maximální délka hledaného textu je 16 znaků.

př.	F%$CALL	hledá v celém textu slovo CALL
	F$LD	hledá od aktuálního řádku slovo LD
	F	hledá od aktuální řádku další výskyt slova

H {co} HARDCOPY

Vytiskne oblast textu "co" na tiskárně včetně čísel řádků. Bez uvedení rozsahu textu parametrem "co" se vytiskne celý text. Po vypsání "Ready printer" potvrďte připravenost tiskárny klávesou NEW LINE. Tisk je možno přerušit klávesou CTRL C.

Podprogram výstupu na tiskárnu není součástí EDTASMu ani operačního systému a je nutné ho předem nahrát do paměti a inicializovat příslušný vektor.

I {kam}{,krok} INSERT

Umožňuje vsouvání řádků do textu. Psané řádky jsou automaticky číslovány s krokem "krok". Je to základní způsob psaní textu. Řádky lze psát i způsobem jako u jazyka BASIC, tzn. místo příkazu napsat číslo řádku a za ním text. Vkládání příkazem I se ukončí klávesou BREAK.

př.	I 300,2	vsouvání řádků od čísla 300 s krokem 2
	I	vsouvání za aktuální řádek
	150 text	vsunutí nebo oprava řádku 150
	100	zrušení řádku 100

Při vsouvání více řádků může dojít k tomu, že by čísla vsouvaných řádků byla vyšší než číslo následujícího řádku. V

takovém případě se všechny vsouvané řádky číslují stejným číslem, které je o jedničku nižší než číslo následujícího řádku. Po ukončení vkládání se celý text automaticky přečíslujte od čísla 100 s krokem 20.

L [název] LOAD

Provede se načtení souboru s názvem "název". Pokud název neuvedeme, použije se naposledy použitý název (lze ho zjistit příkazem ?). Před názvem souboru musí být mezera. K názvu se automaticky připojuje typ .ASM.

Pokud při vyvolání příkazu L je v paměti nějaký text, vypíše se "Text in buffer. are you concatenating (Y/N)?". Po stisku klávesy Y se soubor připojí na konec textu v paměti, po N se napřed text v paměti smaže.

M kam,cof,krok] MOVE

Přesun oblasti textu určené parametrem "co" na místo "kam". Řádky se číslují s krokem "krok". Na původním místě je text zrušen.

Hlášení "No room between lines" znamená, že při číslování vkládaných řádků došlo k překročení čísla následujícího řádku. V tomto případě použijte menší krok nebo celý text přečíslujte příkazem N. Text v tomto případě zůstane beze změny.

Přenos větší části textu může trvat dosti dlouho, neztrácejte tedy trpělivost a vyčkejte dokončení příkazu.

M 500,100:350,1 přesune řádky 100 až 350 za řádek 500

N [od][,krok] NUMBER

Přečísluje všechny řádky s krokem "krok", první řádek má číslo "od". Protože při INSERTu se přečíslovává automaticky, není příkaz N většinou potřeba (pouze u MOVE a COPY). Pokud by při číslování došlo k překročení rozsahu čísel řádků (příliš

veliký krok číslování), sníží se krok na polovinu a funkce se zopakuje.

Po načtení souboru do paměti se automaticky provede příkaz N 100,20

př. N 100,10 přečísluje řádky od 100 s krokem 10

O ORG

Vypíše se rozsah volné paměti v následující formě:

```
FIRST=4321              {začátek volné paměti}
LAST=BCDE               {konec volné paměti}
USRORG=BCDC             {aktuální rozdělení paměti}
USRORG=
```

Nyní lze zadat adresu pro rozdělení paměti (šestnáctkové číslo). Oblast nad touto adresou bude chráněna, EDTASM ji nebude využívat. Po spuštění EDTASMu je tento předěl nastaven na konec volné paměti.

P {co} PRINT

Vypíše na displej část textu určenou parametrem "co". Bez tohoto parametru se vypíše jedna stránka textu od aktuálního řádku.

```
př.  P 100:230      vypíše řádky 100 až 230
     P%             vypíše celý text
     P              vypíše následující stránku
```

Q{A}{Z} QUASH

Tento příkaz umožňuje zrušit Z-BUG (příkaz QZ) nebo Z-BUG i ASSEMBLER (příkaz QA) pro získání většího místa v paměti. Po

příkazu se vypíše "Quash?" a ke zrušení dojde pouze po potvrzení klávesou NEW LINE. Obnovení Z-BUGu a ASSEMBLERu je možné pouze novým zavedením celého systému EDTASM z kazety.

př.	QZ	zruší Z-BUG
	QA	zruší Z-BUG i ASSEMBLER

R {co}{,krok} REPLACE

Zruší řádek "co" a vyvolá příkaz INSERT na jeho místě s krokem "krok".

př.	R 100,2	zruší řádek 100 a vyvolá příkaz I 100,2
	R 100	zruší řádek 100 a provede I 100
	R	zruší aktuální řádek a provede I

S {kde}{$který}{$čím} SUBSTITUTE

Provede v rozsahu řádků "kde" nahrazení textu "který" textem "čím". Vypíše všechny řádky, ve kterých dojde k nahrazení. Oddělovačem je opět klávesa ESCAPE. Bez uvedení textů se provádí nahrazování textů naposledy uvedených. Příkaz S bez parametrů provede nahrazení jednoho nalezeného textu, hledat se začíná od řádku následujícího po aktuálním řádku.

Maximální délka hledaného i nahrazujícího textu je 16 znaků.

př. S 300:400$původní text$nahrazující text
- na uvedených řádcích provede požadované nahrazení

S%LDMOVE
- v celém textu nahradí LD slovem MOVE

S.aaabbb
- provede nahražení pouze v aktuálním řádku

S300:400xxxyyy
- nahrazení se provede na řádcích 300 až 400 (včetně)

S

- od následujícího řádku hledá zadaný text a provede nahrazení pouze na tomto řádku

T {co} TYPE

Provede tisk na tiskárnu jako příkaz HARDCOPY, ale bez čísel řádků.

W {název souboru} WRITE

Zapíše celý text do souboru s názvem "název souboru". Pokud název neuvedeme, použije se předchozí (např. od příkazu LOAD). K názvu se automaticky připojuje typ .KÓD.

př. W ALS8Ø zapíše text do souboru ALS8Ø.KÓD
W zapíše text pod původním jménem

Z Z-BUG

Ukončí práci EDITORu a předá řízení Z-BUGu. Ze Z-BUGu je samozřejmě možné se vrátit zpět do EDITORu.

? QUESTION

Vypíše délku textu, zbývající volné místo a aktuální název souboru v následující formě:

```
Documenth length: ØØØØØ
Characters free:  31479
Current file name: FILE - saved
```

Slovo "saved" za aktuálním názvem souboru znamená, že text byl v této podobě zapsán do souboru. Pokud byl text opraven a

nebyl zapsán do souboru, vypíše se "updated". V tomto případě je také nutno potvrzovat příkaz BYE.

Povely editoru

EDITOR kromě příkazů zná ještě několik povelů. Povely jsou jednoznakové příkazy, které se provedou ihned po stisknutí příslušné klávesy. Povely je nutno psát ihned za *, jinak jsou ignorovány.

"šipka nahoru"	vypíše předchozí řádek
"šipka dolů"	vypíše následující řádek
#	vypíše první stránku
/	vypíše poslední řádek
@	vypíše následující stránku
:	vypíše předchozí stránku
CTRL "dolů"	smaže obrazovku
SHIFT "vlevo"	smaže vstupní řádek
SHIFT "nahoru"	vyvolá obrazovkový editor

Po vyvolání obrazovkového editoru lze s jeho pomocí připravit jeden řádek, který se odešle po stisknutí klávesy NEW LINE. Po odeslání řádku se EDITOR vrátí opět do řádkového módu. Délka řádky smí být maximálně 128 znaků. Při pokusu odeslat delší řádek se kurzor nastaví na první neplatný znak (129.) a počítač pípne. Řádek je nyní potřeba zkrátit.

Seznam příkazů editoru

A název /sw	ASSEMBLER
B	BYE
C kam,co,krok	COPY
D co	DELETE
E co	EDIT
F kde$co	FIND
H co	HARDCOPY
I kam	INSERT

L název	LOAD
M kam,co,krok	MOVE
N od,krok	NUMBER
O	ORG
P co	PRINT
Q co	QUASH
R co	REPLACE
S kdecočím	SUBSTITE
T co	TYPE
W název	WRITE
Z	Z-BUG
?	QUESTION

Spuštění assembleru

ASSEMBLER se spouští z EDITORu příkazem A, který má následující formát:

A {číslo řádky} {název souboru} {/sw/sw/sw...}

Překlad začne na řádku "číslo řádky", pokud není řádek uveden, začne se překládat od první řádky.

Parametry "/sw" mají tento význam:

/NL	NO LIST
/ST	SYMBOL TABLE
/NO	NO OBJECT
/IM	INTO MEMORY
/AO	ABSOLUTE ORG
/LP	LINE PRINTER
/NC	NO CHECK

NL - protokol o překladu se nebude vypisovat na displej. Protokol obsahuje absolutní adresy, strojové kódy instrukcí a text řádku včetně čísla řádku. Na konci protokolu se vypíše celkový počet chyb nalezených při překladu. Potlačení výpisu protokolu o překladu lze provést také pseudoinstrukcí .XLIST (viz. dále).

ST - po překladu se vypíše tabulka symbolů setříděná podle abecedy, u každého symbolu se vypíše jeho hodnota (adresa) a případné další specifikace:

M	symbol je název MACRO
U	symbol není definován
D	symbol je definován pomocí DEFL
R	symbol je definován vícekrát
F	MACRO je voláno před svou definicí

NO - negeneruje se kód. Bez tohoto parametru se automaticky generuje kód do souboru. Potvrzování je obdobné jako u příkazu WRITE v EDITORu. Pokud chceme kód zapsat do jiného souboru,

můžeme v příkazu A před parametry napsat nový název oddělený mezerou.

př. A PREKLAD /NL zapíše kód do souboru PREKLAD, protokol se nevypisuje

IM - způsobí ukládání kódu do paměti. Automaticky způsobí /NO. Adresy ukládání jsou kontrolovány, takže nemůže dojít k přepsání EDTASMu ani zdrojového textu.

AO - ve spojení s /IM způsobí ukládání kódu do paměti na adresy uvedené v programu pomocí pseudoinstrukce ORG. Bez /AO se kód ukládá do paměti hned za text.

LP - přepne výstup protokolu a tabulky symbolů na tiskárnu.

NC - při překladu do paměti (/IM) se kontroluje adresa pro ukládání kódu tak, aby nedošlo k přepsání EDTASMu nebo textu v paměti. Dolní hranice je dána koncem zdrojového textu a horní začátkem tabulky symbolů. Přepínačem /NC je možno tuto kontrolu potlačit.

Obecné poznámky

Překlad je možno kdykoli přerušit klávesou CTRL C, která způsobí návrat do EDITORu.

Pokud je při překladu nalezena chyba, chybný řádek se vypíše, počítač pípne a překlad se pozastaví. Pokračování překladu dosáhneme stiskem libovolné klávesy. Stisknutí klávesy C způsobí, že na dalších chybách se již nečeká.

ASSEMBLER překládá programy ve standardní mnemonice Z-8Ø, zde proto popíšeme pouze odlišnosti a rozšíření.

Názvy návěští mohou obsahovat písmena, číslice a znaky $, ., @, ? a podtržítko. Návěští nesmí tvořit žádné klíčové slovo assembleru (Z, NC, LD...). Návěští nesmí začínat číslicí.

Délka návěští není omezena, významných je ale pouze prvních šest znaků. Za návěštím nemusí být dvojtečka, ale z hlediska přehlednosti, snadného vyhledávání a slučitelnosti s jinými assemblery je vhodné dvojtečku za návěští psát. Za návěštím nemusí být žádný text. Návěští musí být hned na začátku řádku.

Formální parametry v definici MACRO musí začínat znakem #. Maximální délka skutečného parametru je 16. V definici MACRO není možné volat jiné MACRO, tzn. MACRO nelze hnízdit.

Překlad se provádí na dva nebo tři průchody. Při prvém průchodu se tvoří tabulka symbolů, při druhém se tiskne protokol, případně se ukládá kód do paměti. Třetí průchod se provádí pouze při generování kódu do souboru.

Pro generování různých návěští při opakovaném volání MACRO slouží symbol #$YM, který je při každém rozvoji MACRO nahražen jiným číslem (čtyřmi desítkovými číslicemi).

př.

```
MM      MACRO           ;definic MACRO s názvem MM
L#$YM:  CALL    NECO    ;použití symbolu #$YM
        DJNZ    L#$YM
        ENDM

        MM              ;první volání MM
LØØØØ:  CALL    NECO    ;#$YM je nahraženo textem ØØØØ
        DJNZ    LØØØØ

        MM              ;druhé volání MM
LØØØ1:  CALL    NECO    ;#$YM je nahraženo textem ØØØ1
        DJNZ    LØØØ1
```

Výrazy

Ve výrazech můžeme používat těchto prvků:

1, 1Ø, 1ØT, 4Ø96	desítkové konstanty
1ØH, 1ØØH, ØFFFFH	šestnáctkové konstanty
1Ø1ØØ1Ø1B, 1ØB, 1101X	dvojkové konstanty
'A', 'ab'	znakové konstanty

LABEL	návěští
+ - * / MOD DIV	aritmetické operátory
SHL, SHR	aritmetický posun vlevo, vpravo
EQ, NE, LT, LE, GT, GE	aritmetické porovnání
LOW, HIGH	nižší, vyšší bajt slabiky
AND, OR, XOR, NOT, NEQ	logické operátory
!, &	OR, AND
NUL	test na prázdný parametr
(,)	závorky pro změnu priority
$	aktuální PC

Číselné konstanty musí začínat číslicí 0-9. Pokud není uvedeno jinak pseudooperací .RADIX, čísla bez postfixu (znaku na konci) se berou jako desítková. Jsou použitelné tyto postfixy:

B,Y	dvojkový (2)
O,Q	oktalový (8)
D,T	desítkový (10)
H	šestnáctkový (16)

Pseudooperace

Pseudooperace jsou instrukce, které negenerují žádný kód, ale pomáhají programátorovi efektivněji pracovat. Popisovaný ASSEMBLER zná tyto pseudooperace:

DEFS, DS	definice volného místa
DEFM, DM	definice textu
DEFB, DB	definice slabiky
DEFW, DW	definice slova
DEFC, DC	definice textu
ORG	nastavení PC
END	konec textu, definice startovací adresy
EQU	definice hodnoty labelu
DEFL	definice hodnoty labelu, kterou je možno měnit
MACRO	definice bloku MACRO
ENDM	konec bloku MACRO

COND, IF, IFT	začátek bloku podmíněného překladu
IFF	podmíněný překlad s negovanou podmínkou
ELSE	změna stavu podmíněného překladu
ENDC	konec bloku podmíněného překladu
.LIST	povolení tisku protokolu
.XLIST	zákaz tisku protokolu
.RADIX	změna báze číselných konstant
.PRINTX	tisk hlášení na displej
$8Ø8Ø	zákaz instrukcí Z8Ø, které nezná 8Ø8Ø
$Z8Ø	povolení rozšířených instrukcí Z-8Ø

příklady:

```
        DB      1,40H, 'Text', LABEL-1,-1
        DEFM    'DEFM je totéž jako DEFB'
        DW      1234, 10011000101101018, 56*31 MOD 600
        DC      'Text s nastaveným bitem 7 na konci','Konec'
        END     START           ;definice startovací adresy
        END                     ;konec textu, bez startovací adresy
CR      EQU     0DH
LLL     DEFL    LLL+1

AAA     MACRO   #TEXT,#ADRESA          ;parametry musí začínat #
        DC      '#TEXT'
        DW      #ADRESA
        ENDM

        AAA     'LOAD',LOAD            ;volání MACRO

        COND    CPM                    ;podmíněný překlad
        .PRINTX 'Verze pro zavaděč CP/M'
        ELSE
        .PRINTX 'Verze pro operační systém'
        ENDC

        .RADIX  16                     ;šestnáctkově
```

Chybová hlášení při překladu

BAD LABEL	špatný název návěští
BRANCH OUT OF RANGE	relativní skok mimo rozsah
BAD ADDRESSING MODE	špatný adresní mód instrukce
BAD OPCODE	neznámá mnemonika
NO END STATEMENT	chybí END na konci textu
FIELD OVERFLOW	slabika větší než 255
SYNTAX ERROR	chyba při definici MACRO
NESTED MACROS	MACRO nelze hnízdit
ENDM WITHOUT MACRO	ENDM bez předchozího MACRO
ENDC WITHOUT COND	ENDC bez předchozího COND
ELSE WITHOUT COND	ELSE bez předchozího COND
BAD ADDRESS	adresa pro ukládání mimo rozsah
BAD MEMORY	špatná paměť
MISSING INFORMATION	na řádku chybí další text
MULTIPLE DEFINITION	opakovaná definice návěští
MULTIPLY DEFINED SYMBOL	opakovaně definované návěští
UNDEFINED SYMBOL	nedefinované návěští
BAD EXPRESSION	chyba ve výrazu
SYMBOL TABLE OVERFLOW	přeplněná tabulka symbolů
STACK OVERFLOW	podtekl zásobník
DIVISION BY Ø	dělení nulou
MACRO FWD REF	MACRO je voláno před definicí
Z-8Ø INSTRUCTION	rozšířená instrukce Z-8Ø

Spuštění Z-BUGu

Z-BUG se spouští z EDITORu příkazem Z. Po spuštění se vypíše "Z-Bug" a znak #, který oznamuje, že Z-Bug je připraven plnit příkazy nebo povely.

Povely jsou jednoznakové příkazy, které se provedou ihned po stisknutí příslušné klávesy. Příkazy jsou jednopísmenné a píší se po klávese ESCAPE (v dalším textu budeme místo klávesy ESCAPE psát $). Příkazy se provedou ihned po stisku písmene funkce, není třeba je potvrzovat klávesou NEW LINE.

Před příkazy i povely mohou být parametry, které od sebe oddělujeme čárkou.

V parametrech příkazů je možno použít stejné aritmetické operace a číselné nebo znakové konstanty jako v ASSEMBLERu, navíc lze použít těchto prvků:

A,B,C,D,E,H,L,F	- hodnota jednoduchého registru
BC,DE,HL,SP	- hodnota dvojitého registru
Z,NZ,CY,NC,P,M,PO,PE	- test nastavení nebo vynulování příznaku (Zero, Carry, Sign, Parity)

Příkazy Z-BUGu

$A ASCII

Přepne výstup do znakového režimu. Obsah paměti bude nadále vypisován ve formě znaků. Místo kódů menších než 32 se vypisuje tečka.

$B BYTE

Přepne výstup do bajtového režimu. Obsah paměti bude nadále vypisován ve formě čísel v nastaveném výstupním radixu (číselné soustavě - viz. příkaz O).

{kolik} $C CONTINUE

Pokračování běhu laděného programu po přerušení běhu na bodu zastavení nebo po krokování. Parametr "kolik" určuje počet průchodů daným bodem zastavení, než dojde k přerušení běhu. Bez uvedení parametru se bere 1.

Parametr "kolik" se čte při radixu 1Ø.

$D DISPLAY BREAKPOINTS

Vypíše aktuální body zastavení (číslo, adresa a podmínka).

Př.

```
Ø @ CYKL+4
2 @ INIT    NZ
3 @ SUBR+15 (HL=Ø AND CY)
```

$E EDITOR

Návrat ze Z-BUGu zpět do EDITORu.

{adr} $G GO

Spuštění programu od adresy "adr", při neuvedení parametru se pokračuje od aktuální hodnoty PC.

radix $I INPUT RADIX

Nastavení vstupního radixu - číselné soustavy na "radix". Radix musí být v mezích 2..16. Parametr se čte při radixu 1Ø.

{název} $L LOAD/TAPE

Načte program ze souboru do paměti. Startovací adresa se připraví do registru PC. K názvu souboru se automaticky připojí typ .KÓD.

$M MNEMONIC

Přepne výstup do mnemonického režimu. Obsah paměti je nadále vypisován ve formě mnemoniky Z-8Ø (ruší nastavení příkazy $A, $B a $W).

$N NUMERIC

Vypne symbolický výstup - viz. příkaz $S.

radix $O OUTPUT RADIX

Nastavení výstupního číselného radixu na "radix". Parametr se čte při radixu 1Ø. Výstupní radix smí být 8, 1Ø nebo 16. Při výstupu se za čísly píše tento postfix:

O radix=8 (oktal)
T radix=1Ø (decimal)
radix=16 (hexadecimal) - žádný postfix

od,do,start{,název} PRINT/TAPE

Zapíše blok paměti od adresy "od" do adresy "do" včetně do souboru. "start" je startovací adresa, od které je program po načtení spuštěn. Pokud je rovna Ø, program se po načtení nespustí. Pokud není uveden název, vezme se naposledy udaný název (příkazy editoru L, W a A). K názvu se automaticky připojí typ .KÓD.

$R REGISTERS

Vypíše aktuální hodnoty všech registrů.

$S SYMBOLIC

Zapnutí symbolického výstupu. Při výstupu adresy se prohledá tabulka symbolů a pokud hodnota vypisované adresy se liší od hodnoty některého návěští o méně než 32, vypíše se místo čísla návěští a offset.

```
Př.   LD   BC,LABEL+4
      LD   HL,(CITAC+17)
      LD   (FLAG),A
```

Příkaz $N potlačí hledání v tabulce symbolů, adresy se tedy vypisují jako číselné konstanty. Výpis adresy zobrazovaného místa paměti se vždy vypisuje v symbolické formě.

od, do $T TYPE

Výpis bloku paměti od adresy "od" k adrese "do" v nastavené výstupní formě.

$W WORD

Nastavení výstupu do režimu slovo. Obsah paměti je nadále vypisován ve formě slov - 16ti bitové hodnoty.

{adr} {,podmínka} $X SET BREAKPOINT

Nastavení bodu zastavení na adresu "adr", pokud není uvedena, tak na aktuální adresu. Při uvedení podmínky se zastavení na dané adrese provede pouze při splnění uvedené podmínky.

Najednou může být nastaveno až 8 bodů zastavení. Číslo bodu zastavení se neudává, Z-BUG si sám najde volné číslo.

Body zastavení smí být umístěny pouze v paměti RAM!

Př. CYKL $X — nastaví bod zastavení na CYKL,
VYSTUP+3,NZ $X — k zastavení na adrese VYSTUP+3 dojde při nenastavené příznaku Zero

{číslo} $Y — RESET BREAKPOINT

Zrušení bodu zastavení s číslem "číslo". Pokud není uveden parametr, zruší se všechny body zastavení. Čísla bodů zastavení se zjistí příkazem $D.

Při špatném parametru se vypíše "Zerr" a příkaz je ignorován.

Povely Z-BUGu

{adr} / — OPEN LOCATION

Nastaví ukazatel na adresu "adr" a vypíše obsah této adresy v nastavené formě. Při neuvedení "adr" se bere hodnota PC.

Tímto příkazem můžeme také měnit hodnotu registrů. Místo "adr" napíšeme název registru. Vypíše se aktuální hodnota tohoto registru, nyní napíšeme novou hodnotu a stiskneme NEW LINE.

{výraz} = — HOW MUCH IS IT ?

Vypočte se hodnota výrazu "výraz" a výsledek se vypíše v číselném tvaru. Bez uvedení parametru se vypíše hodnota bajtu na aktuální adrese v číselné formě.

: — FLAGS

Vypíše hodnotu bajtu na aktuální adrese ve formě příznaků (jako registr F).

{odkud} @ SINGLE-STEP

Provedení instrukce na adrese "odkud". Při dalším stisku klávesy @ se vždy provede instrukce na adrese PC a vypíše se následující instrukce. Krokovat lze pouze program v paměti RAM!

Po instrukci CALL se pokračuje na adrese podprogramu.

{odkud} ' CALL-STEP

Totéž jako povel @, ale instrukce CALL se bere jako jeden krok - podprogram se zavolá a krokování pokračuje instrukcí následující za instrukcí CALL. Znak ' je tzv. obrácený apostrof.

"šipka nahoru" UP

Nastavení aktuální adresy na předchozí instrukci nebo bajt - podle nastaveného výstupního režimu.

{hodnota} "šipka dolů" DOWN

Na aktuální adresu se uloží "hodnota" a přejde se na následující instrukci, bajt nebo slovo. Bez uvedení parametru se obsah aktuální adresy nemění.

{hodnota} NEW LINE UPDATE

Na aktuální adresu se uloží "hodnota" a aktuální adresa se uzavře.

; NUMERIC

Vypíše obsah aktuální adresy v numerickém režimu ($N).

Při špatném parametru se vypíše "Zerr". Při změně obsahu paměti se obsah zkontroluje a při chybě se vypíše "Bad memory at " a adresa, kde došlo k chybě při zápisu.

	$A	ASCII
	$B	BYTE
počet	$C	CONTINUE
	$D	DISPLAY BREAKPOINTS
	$E	EDITOR
adresa	$G	GO
číslo	$I	INPUT RADIX
název	$L	LOAD
	$M	MNEMONIC
	$N	NUMERIC
číslo	$O	OUTPUT RADIX
od,do,start,název	$P	PRINT
	$R	REGISTERS
	$S	SYMBOLIC
od,do	$T	TYPE
	$W	WORD
adr,podm	$X	SET BREAKPOINT
číslo	$Y	YOUNG BREAKPOINT
adr	/	OPEN LOCATION
výraz	=	HOW MUCH IS IT ?
	:	FLAGS
adresa	@	SINGLE-STEP
adresa	'	CALL-STEP
	"šipka nahoru"	UP
	"šipka dolů"	DOWN
hodnota	NEW_LINE	CLOSE LOCATION
	;	NUMERIC