M A S M - makroassembler

1.- úvod

MASM je dvouprůchodový asembler generující cílové moduly kompatibilní s cílovými moduly vytvořenými kompilátory vyšších programovacích jazyků. Tím je dána možnost sestavovat úplné programy na základě kombinování cílových modulů (relokovatelné kódy) MASMu s cílovými moduly vygenerovanými jazykovými kompilátory. Relokovatelný cílový kód je soubor, v kterém je každé instrukci či jinému příkazu přiřazen offset od jeho segmentové báze. Sestavený kód je možné potom spojit pomocí programu LINK a vytvořit tak kód programu, který lze již zavést do paměti. Relokovatelný kód navíc znamená, že je možné programy vytvářet ve formě modulů, z nichž každý lze sestavit a odladit samostatně.

V prvním průchodu asembler:
vyhodnocuje příkazy a rozvinuje příkazy volání makroinstrukcí vypočítává velikost kódu, který vygeneruje zakládá tabulku symbolů, v níž se symbolům, proměnným, návěštím a makroinstrukcím přiřazují hodnoty

V druhém průchodu asembler:
Dosazuje za symboly, proměnné, návěští a výrazy hodnoty z tabulky symbolů rozvinuje příkazy volání makroinstrukcí zapisuje relokovatelný cílový kód do souboru, jehož jméno má standardní doplněk .OBJ. Další vlastností makroasembleru je možnost zapsat kódy bloků programu zkráceným povelem (makroinstrukce) - např. při častém opakování těchto bloků programu.

2.- Prvky asembleru

V asembleru lze použít tyto číselné základy:
Dekadické číslo (číslice 0 až 9, volitelně může následovat písmeno D) binární číslo (číslice 0 nebo 1, následuje písmeno B) oktalové číslo (číslice 0 až 7, následuje písmeno O nebo Q) hexadecimální číslo (číslice 0 až 9 nebo písmena A až F, následuje písmeno H řada musí začínat vždy číslicí 0 až 9). Standardní nebo v daném okamžiku platný základ je možné změnit direktivou .RADIX. Implicitně je číselná soustava dekadická.

Lze použít tyto rozsahy celých čísel:

Tvar	Bitů	Rozsah	Počet platných číslic
bajt bez znaménka	8	0 až 255	2 až 3
bajt se znaménkem	8	-128 až 127	2 až 3
slovo bez znaménka	16	0 až 65535	4 až 5
slovo se znaménkem	16	-32768 až 32767	4 až 5
dvouslovo bez znaménka	32	0 až 4294967295	9 až 10
dvouslovo se znaménkem	32	-2147483647 až 2147483648	9 až 10
čtyřslovo bez znaménka	64	0 až 1.84E19	19 až 20
čtyřslovo se znaménkem	64	-9e18 až 9e18	18

znaménko se uloží do nejvyššího bitu čísla (např. FFFFh = -1)

Reálná čísla:

Tvar	Bitů	Rozsah	Přesnost mantisy
jedn. přesn. Microsoft	32	3.0E-39 až 1.7E38	6 až 7 des. míst
dvoj. přesn. Microsoft	64	3.0E-39 až 1.7E38	6 až 17 des. míst
jedn. přesn. IEEE	32	1.175E-38 až 1.80E308	7 až 8 des. míst
dvoj. přesnost IEEE	64	1.175E-38 až 1.80E308	15 až 16 de. míst

Reálná čísla lze vyjadřovat též v kódovém zápisu. Kódový zápis tvoří řada hexadecimálních číslic ukončená písmenem R (řada musí začínat číslicí 0úažú9). Hodnoty exponentu, znaménka atd. jsou uvnitř čísla vyjádřena jako bitová pole (např. 3F800000R znamená 1.0 při deklaraci DD). Kódový zápis je možné definovat direktivami DD, DQ nebo DT.

Posledním typem čísel jsou desítková čísla ve zhuštěném formátu. Tato čísla se ukládají do 10 bajtů podle pravidel binárního kódu. řádově nejvyšší bit představuje znaménko (1=záporné) (např. DT -1234567890 odpovídá binárnímu kódu 8000000001234567890H).
Znaková konstanta se skládá z jediného znaku ASCII. řetězcová konstanta se skládá z jednoho nebo více znaků kódu ASCII. Tyto konstanty se uzavírají mezi znaky apostrofu ,. Je-li součástí řetězcové konstanty apostrof, pak se zapíše dvakrát (např. ‚a‘, „a“, „“nenalezen‘).
Identifikátor je kombinací písmen, číslic a speciálních znaků. Malá písmena se automaticky přeřazují na velká (pokud není nastaven parametr /ML nebo /MX).

Vyhrazená jména jsou jména, která již mají předem definovaný význam.
Kromě mnemotechnických zápisů instrukcí patří mezi vyhrazená jména:
.186 CH ELSE .ERRNDEF IFE MASK RECORD THIS .286c CL END .ERRNZ IFIDN MOD REPT TITLE .286p COMMENT ENDIF TYPE ES IFNB NAME .SALL .287 CREF ENDM EVEN IFNDEF NE SEG .TYPE .8086 CS ENDP EXITM INCLUDE NEAR SEGMENT WIDTH .8087 CX ENDS EXTRN IRP NOT .SFCOND WORD = DB EQ FAR IRPC OFFSET SHL .XALL AH DD EQU GE LABEL OR SHORT .XREF AL DH .ERR GROUP .LALL ORG SHR .XLIST AND DI .ERR1 GT LE %OUT SI XOR ASSUME DL .ERR2 HIGH LENGTH PAGE SIZE AX DQ .ERRB IF .LFCONDł PROC SP BH DS .ERRDEFł IF1 .LIST PTR SS BL DT .ERRDIFł IF2 LOCAL PUBLIC STRUC BP DW .ERRE IFB LOW PURGE SUBTTL BX DWORD .ERRIDNł IFDEF LT QWORD TBYTE BYTE DX .ERRNB IFDIF MACRO .RADIX .TFCONDł

Způsob zápisu řádku v makroasembleru:
Každý příkaz, instrukce či direktiva je umístěna na samostatném řádku. Každý řádek může začínat návěštím označujícím místo v programu nebo jméno proměnné. Na konci řádku může být umístěna za oddělovačem „“ poznámka.

3.- Operandy, operátory a výrazy
Operand je konstanta, návěští, proměnná nebo jiný symbol, který se používá v instrukci nebo direktivě pro označení hodnoty nebo paměťového místa, s nimiž se má pracovat.

Konstantové_operandy:
- číslo - neproměnná hodnota ve tvaru konstanty

- řetězec - obsahuje jeden nebo více znaků ASCII v apostrofech
- identifikátor - konstanta definovaná pomocí direktivy EQU nebo „=“
- výraz - přípustný výraz, který vyhodnocuje číslo nebo řetězcovou konst.
-

Přímé_paměťové_operandy:
reprezentují absolutní paměťové adresy jednoho nebo více bajtů paměti.

Syntaxe: segment:offset kde segment = CS, DS, SS nebo ES offset = celé číslo, identifikátor konstanty, výraz nebo symbol (hodnota 0 až 65535)

Relokovatelné_operandy:
reprezentují paměťové adresy (segment a offset) instrukcí nebo dat.
Syntaxe: symbol kde symbol = návěští, jméno proměnné, segmentu nebo skupiny

Operand_lokálního_čítače:
reprezentuje okamžitou pozici v okamžitém segmentu
Syntaxe: $ lokální čítač má atributy návěští NEAR

Registrové_operandy:
adresují příslušné registry procesoru AX,BX,...

Operandy_s_bází:
představují paměťové adresy závislé relativně na některém bázovém registru.

- Syntaxe 1: [posunutí]registr_nepřímé_báze
- Syntaxe 2: [bázový_registr + posunutí]
- Syntaxe 3: registr_nepřímé_báze.posunutí
- Syntaxe 4: registr_nepřímé_báze + posunutí

kde posunutí = libovolný přímý operand bázový_registr = BP (rel. v SS) nebo BX (rel. v DS)

Operandy_s_indexem:
představují paměťové adresy, které jsou relativní vůči index. registru

- Syntaxe 1: [posunutí]registr_nepřímého_indexu
- Syntaxe 2: [indexový_registr + posunutí]
- Syntaxe 3: registr_nepřímého_indexu.posunutí
- Syntaxe 4: registr_nepřímého_indexu + posunutí

kde posunutí = libovolný přímý operand indexový_registr = SI (v DS) nebo DI (v DS nebo ES)

Operandy_s_bází_a_indexem:
reprezentují paměťovou adresu, která je relativní vůči kombinaci bázového a indexového registru

- Syntaxe 1: [posunutí]nepřímý_bázový_registr registr_nepřímé_báze
- Syntaxe 2: [bázový_registr + indexový_registr + posunutí]
- Syntaxe 3: [bázový_registr + indexový_registr]
- Syntaxe 4: ukazatel_nepřímé_báze + registr_nepřímého_indexu + posunutí

Operandy_struktury:
reprezentují paměťovou adresu jednoho člena struktury
Syntaxe:
proměnná.pole kde proměnná = jméno operandu struktury
pole = jméno pole uvnitř struktury

Operandy_věty:
operandy věty se vztahují k hodnotě typu věty
Syntaxe:
jméno_věty<[hodnota][,[hodnota]]...>
kde jméno_věty = jméno qdefinováno direktivou RECORD
hodnota = hodnota, která se má umístit do pole věty

Operandy_polí_věty:
určují polohu pole v jeho odpovídající větě
Syntaxe: jméno_pole_věty
Pozn.: Výsledkem vyhodnocení operandu je bitová pozice řádově nejnižšího bitu v poli, kterou je možné použít jako konstantový operand.

Aritmetické_operátory:

operátor	význam
*	násobení
/	celočíselné dělení
MOD	modulo - zbytek po celočíselném dělení
SHR	posuv doprava (následuje počet posuvů)
SHL	posuv doleva (následuje počet posuvů)
- (unární)	změna znaménka
+	sčítání (jeden z operandů nemusí být konstanta)
	odečítání (levý operand nemusí být konstanta, nebo oba operandy nemusí být konstanty

Relační_operátory:

operátor	význam
EQ	rovnost - vrací TRUE, jsou-li si operandy rovny
NE	nerovnost - vrací TRUE, jsou-li si operandy nerovny
LT	menší než - vrací TRUE, je-li levý operand < než pravý
LE	menší nebo rovno - vrací TRUE, je-li levý oper. <= pravý
GT	větší než - vrací TRUE, je-li levý oper. > pravý
GE	větší nebo rovno - vrací TRUE, je-li levý oper. >= pravý

Pozn.: TRUE=0ffffh (=-1), ELSE=0

Logické_operátory:

operátor	význam
NOT	logické NOT (unární) - vytváří dvojkový doplněk
AND	logické AND - provádí bitově operaci AND
OR	logické OR - provádí bitově operaci OR
XOR	logické XOR - provádí bitově operaci XOR

Pozn.: Logické operace se provádějí s jednotlivými bity operandů, výsledkem operace je absolutní adresa. Nejjednodušším případem jsou logické stavy TRUE (hodnota -1) a FALSE (hodnota 0).

Atributové_operátory:
přepisné operátory: používají se pro přepis (dočasnou změnu) segmentu, offsetu, typu nebo vzdálenosti proměnných nebo návěští
Přepis segmentu:

Syntaxe 1:	segmentový_registr:adresový_výraz kde segmentový_registr = registr CS, DS, ES, SS
Syntaxe 2:	jméno_segmentu:adresový_výraz kde jméno_segmentu = jméno defin. direkt. SEGMENT
Syntaxe 3:	jméno_skupiny:adresový_výraz kde jméno_skupiny = jméno defin. direktivou GROUP

MASM předpokládá, že návěští jsou adresovatelná prostřednictvím okamžité hodnoty registru CS. Proměnné jsou standardně adresovatelné prostřednictvím okamžité hodnoty v registru DS, popř. ES (cílové řetězce) nebo SS (bázový registr BP). Je-li operand v jiném segmentu s nebyla-li použita direktiva ASSUME, je možné operátor pro přepis segmentu použít.

typ paměťové proměnné	standardní segm. reg.	alternativní	offset segm. reg.
vyvolání instrukce	CS	-	IP
operace se zásobníkem	SS	-	SP
proměnná	DS	CS, ES, SS	EA
řetězec (zdroj)	DS	CS, ES, SS	SI
řetězec (cíl)	ES	-	DI
bázový registr BP	SS	CS, DS, ES	EA

Pozn.: EA znamená „efektivní adresa“

PTR - přepis typu paměťové proměnné nebo vzdálenosti návěští:
Syntaxe: atribut PTR výraz

atribut	význam
BYTE	paměťový_výraz dostává atribut BYTE
WORD	paměťový_výraz dostává atribut WORD
DWORD	paměťový_výraz dostává atribut DWORD
QWORD	paměťový_výraz dostává atribut QWORD
TBYTE	paměťový_výraz dostává atribut TBYTE
NEAR	návěšťový_výraz dostává atribut NEAR
FAR	návěšťový_výraz dostává atribut FAR

výraz	význam
paměťový_výraz	platný výraz (i jednoduchá proměnná), jenž se vyhodnocuje na paměťovou adresu
návěšťový_výraz	platný výraz (i jednoduchá proměnná), jenž se vyhodnocuje na adresu místa, na něž je možné odskočit nebo které lze volat
registrový_výraz	platný výraz obsahující referenci na registr, který je možné vyhodnotit až při běhu programu
konstantový_výraz	platný výraz, který se vyhodnocuje na celočíselný konstantní offset segm. reg.

Operátor PTR slouží především pro dopřednou referenci typu proměnné nebo návěští a pro přístup k datům podle jiného typu než který je uveden v definici proměnné.

SHORT - přepis atributu návěští na vzdálenost NEAR:
Syntaxe: SHORT návěští
Přepisový operátor SHORT přepisuje atribut vzdálenosti NEAR návěští jako místa skoku instrukce JMP. Prostřednictvím SHORT se MASM dozví, že vzdálenost mezi příkazem JMP a návěštím uvedeným jako jeho operand není větší než 127 bajtů v obou směrech. Pokud tato dopředná reference není použita, vyhradí se pro instrukci JMP při 1. průchodu 3 bajty a pokud je vzdálenost místa skoku menší než 127, použije se automaticky při 2. průchodu instrukce JMP SHORT doplněna instrukcí NOP.

THIS - vytvoření operandu určitého typu nebo vzdálenosti:

Syntaxe 1:	THIS	vzdálenost	kde vzdálenost = buf NEAR nebo FAR vytváří operand s uvedeným atributem vzdálenosti a offsetem rovným okamžité hodnotě lok. čítače
Syntaxe 2:	THIS	typ	kde typ = BYTE, WORD, DWORD, QWORD nebo TBYTE vytváří operand s uvedeným atributem typu a offsetem rovným okamžité hodnotě lok. Čítače

Příklady: TAG EQU THIS BYTE (= TAG LABEL BYTE)

KONTROLA = THIS NEAR (= KONTROLA LABEL NEAR)

HIGH, LOW - operátory výběru bajtu:

Syntaxe 1:	HIGH výraz	vybere a oddělí horních 8 bitů slova nebo adresy
Syntaxe 2:	LOW výraz	vybere a oddělí dolních 8 bitů slova nebo adresy

Operátory vracející hodnotu:
tyto operátory vrací hodnoty atributů operandů, které jsou v zápisu uvedeny za nimi, ale nepřepisují je. Argumenty operátorů jsou návěští nebo proměnné.

SEG - vrací hodnotu segmentu návěští nebo proměnné:
Syntaxe 1: SEG návěští
Syntaxe 2: SEG proměnná

OFFSET - vrací hodnotu offsetu návěští nebo proměnné v segmentu nebo skupině, kde je tato proměnná nebo návěští definováno
Syntaxe 1: OFFSET návěští
Syntaxe 2: OFFSET proměnná

TYPE - vrací počet bajtů typu proměnné nebo návěští:

Syntaxe 1:	TYPE	návěští vrací operátor NEAR (FFFFh) nebo FAR (FFFFh)
Syntaxe 2:	TYPE	proměnná vrací následující hodnoty BYTE -> 1; WORD -> 2 ; DWORD -> 4 ; QWORD -> 8 TBYTE -> 10 ; STRUC -> počet bajtů deklarovaných pomocí STRUC

.TYPE - vrací bajt popisující mód proměnné a příznak, zda je externí:
Syntaxe: .TYPE proměnná navrácený bajt obsahuje tyto informace: bit 0 a 1:
0 = mód je absolutní
1 = mód je závislý na programu
2 = mód je závislý na datech
bit 5:
0 = výraz není definován nebo je externí
1 = výraz je definován
bit 7:
0 = výraz je lokální nebo obecný (public)
1 = výraz obsahuje externí referenci
Pozn.: jsou-li nastaveny bity 5 i 7, je výraz nesprávný

LENGTH - vrací počet jednotek typu proměnné:
Syntaxe: LENGTH proměnná
Je-li proměnná definována jako výraz DUP, potom LENGTH vrací počet jednotek typu, tj. číslo, které je uvedeno před prvním DUP ve výrazu. Není-li proměnná definována pomocí DUP, navrací se hodnota 1.

SIZE - vrací celkový počet bajtů přiřazených proměnné:
Syntaxe: SIZE proměnná
SIZE je součinem hodnoty LENGTH a hodnoty TYPE.

Větné operátory:
Větné operátory se používají pro oddělení polí ve větě.
Větné operátory se definují pomocí direktivy RECORD.
Věta může být dlouhá až 16 bitů, definuje se pomocí polí o délce 1 až 16 bitů.
operátor význam
jméno_pole_věty počet bitů od dolního konce věty k dolnímu konci pole (počet bitů, o něž se musí věta posunout vpravo k nejnižším bitům věty)

WIDTH pole , věta vrací počet bitů pole nebo věty

MASK jméno_pole hodnota věty v případě, že pole obsahuje svou maximální hodnotu a všechna ostatní pole jsou nulová (všechny bity v poli obsahují 1, všechny ostatní bity obsahují 0)

Priorita_operátorů:
Operátory LENGTH, SIZE, WIDTH, MASK, (), [], <> nejvyšší priorita
. (operátor jména-pole struktury)
: (segment)
PTR, OFFSET, SEG, TYPE, THIS
HIGH, LOW
+, - (unární)
*, /, MOD, SHL, SHR
+, - (binární)
EQ, NE, LT, LE, GT, GE
NOT
AND
OR, XOR

SHORT, .TYPE nejnižší priorita

Pozn.: Operace se stejnou prioritou se uskutečĺují zleva doprava.
Pořadí priority lze změnit použitím závorek.

4.- Direktivy pro soubor instrukcí a paměťové direktivy

Zdrojový soubor asembleru musí mít standardní formát ASCII (tj. bez řídicích znaků, každý řádek textu je ukončen dvojicí znaků CR-LF). Příkazy je možné psát pomocí velkých i malých písmen.
Pokud nejsou použity přepínače /R nebo /E, jsou všechna reálná čísla zobrazována ve formátu Microsoft (tj. tak, jak jej používá GWBASIC). Parametr /R vyžaduje použití koprocesoru, parametr /E vyžaduje spojení s knihovnou MATH.LIB.

Direktivy_pro_soubor_instrukcí
.8086, .8087
Direktiva aktivizuje sestavení instrukcí pro mikroprocesory 8086 a 8088. Současně zabraĺuje sestavení instrukcí, které přísluší procesorům 80186 a 80286.
Podobně direktiva .8087 aktivizuje sestavení instrukcí pro koprocesor 8087 pro práci s pohyblivou řádovou tečkou a zabraĺuje sestavení instrukcí, které přísluší koprocesoru 80287.
.186 Direktiva aktivizuje instrukce 8086 a kromě toho i instrukce pro mikroprocesor 80186.

.286 {CP}
Direktiva .286C aktivizuje sestavení instrukcí 8086 a nechráněných instrukcí 80286 (shodné s instrukcemi 80186). Direktiva .286P aktivizuje chráněné instrukce 80286 a instrukce 8086 a nechráněné instrukce 80286.

.287
Direktiva aktivizuje sestavení instrukcí pro koprocesor 80287 s pohyblivou řádovou tečkou.

Paměťové_direktivy

ASSUME
ASSUME informuje asembler o tom, že symboly v segmentu nebo skupině jsou přístupné prostřednictvím uvedeného segmentového registru.

Syntaxe 1: ASSUME segmentový_registr:jméno_segmentu[,.....]
Kde segmentový_registr = CS, DS, ES, SS
jméno_segmentu = jméno segmentu nebo NOTHING
(NOTHING=je nutné každý odkaz opatřit segm.)

Syntaxe 2: ASSUME NOTHING - ruší všechna přiřazení registrů DB, DW, DD, DQ, DT, (DEFINE)

Direktivy DEFINE se používají pro definování proměnných nebo pro inicializaci částí pamětí (tj. uložení hodnot).

Syntaxe: [jméno_proměnné] {DB|DW|DD|DQ|DT} výraz[,výraz]....
Je-li uvedeno volitelné jméno_proměnné, definují direktivy DEFINE jméno jako proměnnou. Má-li jméno_proměnné dvojtečku, stává se místo proměnnou návěštím typu NEAR.

DB - přiděluje jeden bajt (8 bitů)
DW - přiděluje jedno slovo (16 bitů = 2 bajty)
DD - přiděluje dvě slova (32 bitů = 4 bajty)
DQ - přiděluje čtyři slova (64 bitů = 8 bajtů)
DT - přiděluje deset bajtů
Výrazem může být:
konstantní výraz
znak ? pro neurčitou hodnotu (dosadí se 0)
adresový výraz (pouze pro DW a DD)
řetězec ASCII (pouze pro DB)
výraz DUP(?) - pokud je tento typ výrazu jediným argumentem definiční direktivy, vytvoří se neinicializovaný blok dat sloužící pouze k rezervaci místa (tím se zmenší velikost cílového souboru) výraz DUP(výraz[,výraz]...) - vytvoří blok dat, ovšem inicializovaný

END
Syntaxe:
END startovací_adresa
Příkaz END určuje konec programu. Za příkazem END lze uvést startovací adresu programu (při spojování modulů může tuto adresu obsahovat pouze jeden modul).

EQU
Syntaxe:
jméno EQU výraz
Příkaz EQU přiřazuje hodnotu výrazu jménu. Je-li výraz externí symbol nebo má-li jméno již hodotu, vznikne chyba. Jestliže je třeba v programu jméno znovu definovat, použije se místo EQU znak rovnítka (=). V mnoha případech se EQU používá jako náhrada jednoduchého textu jako je např. makroinstrukce. Jako výraz může být použit symbol („ALIAS“), jméno instrukce („OPCODE“), platný výraz (vyhodnocení na hodnotu 0 až 65535) („TEXT“) nebo jakýkoliv jiný prvek včetně textu („TEXT“).

= (rovnítko)
Syntaxe:
jméno = výraz
Rovnítko „=“ se používá k nastavení a novému předefinování symbolu. Výrazem musí být platný výraz (vyhodnocení na hodnotu 0 až 65535). Symbol je možné znovu definovat.

EVEN
Syntaxe: EVEN
Direktivou EVEN se lokační čítač přednastaví na sudou hranici, tj. na hranici, kde začíná slovo. Pokud není lokační čítač na sudé hranici, vygeneruje se instrukce NOP. Jestliže se EVEN použije pro segment zarovnaný na hranici bajtu, vznikne chyba. Tato direktiva se používá především pro zrychlení operací s pamětí, neboť k přenosu slova postačí pouze jeden přístup k paměti.

EXTRN
Syntaxe:
EXTRN jméno:typ [,jméno:typ]...
Direktiva označuje proceduru nebo funkci, která je součástí jiného v paměti uloženého modulu. „jméno“ je symbol, který je definován v jiném modulu jako PUBLIC. „typ“ může být jakýkoliv prvek následujícího seznamu a musí být pro jméno platným typem: BYTE, WORD, DWORD, QWORD, TBYTE NEAR nebo FAR

ABS pro čistá čísla (implicitní velikost WORD, ale i BYTE)

Je-li direktiva uvedena se segmentem, předpokládá se, že je symbol umístěn uvnitř segmentu. Není-li segment znám, potom se direktiva zapíše vně všech segmentů a pak se použije buď „ASSUMEúsegmentový_registr:SEGújméno“ nebo explicitní segmentový prefix.

GROUP
Syntaxe:
jméno GROUP jméno_segmentu[,jméno_segmentu]...
Direktiva GROUP sdružuje segmenty uvedené za GROUP do skupiny se stejným jménem. Používá se k tomu, aby program LINK zjistil, které segmenty se mají do paměti uložit současně. Pořadí segmentů pro zavádění do paměti je určeno pomocí CLASS v direktivě SEGMENT anebo pořadím, v němž se cílové moduly uvedou v odpovědi na výzvu programu LINK uvést seznam spojovaných modulů. Všechny segmenty v GROUP se musí vejít do 64K paměti (kontrolu provádí LINK). Jako „jméno_segmentu“ může byt:
jméno segmentu přiřazené direktivou SEGMENT
výraz: buď SEG proměnná nebo SEG návěští

LABEL
Syntaxe:
jméno LABEL typ
Definováním jména pomocí LABEL dostává asembler informaci o tom, že má okamžitému offsetu přiřadit jméno uvedené v LABEL. Jako „typ“ lze použít NEAR, FAR, BYTE, WORD, DWORD, QWORD, TWORD, jméno_struktury nebo jméno_věty (přiřadí jménu velikost struktury nebo věty). Tato direktiva se používá např. k vícenásobné definici prvku (např. lze definovat slovo jako bajt a odkazovat se tak na něj bez nutnosti používání PTR).

ORG
Syntaxe: ORG výraz
Touto direktivou se nastaví lokační čítač na hodnotu danou výrazem a assembler uloží vygenerovaný kód od této hodnoty. Hodnota výrazu musí být absolutní.

PROC
Syntaxe: jméno_procedury PROC [NEAR|FAR] ..... jméno_procedury ENDP
Direktiva PROC slouží jako prostředek strukturalizace programu pro zvýšení jeho přehlednosti, označuje část programu jako proceduru. Není-li uveden typ, použije se

NEAR. Direktiva informuje, zda se budou příkazy CALL a RET uvnitř procedury překládat jako typ FAR nebo NEAR. Procedury mohou být vzájemně vřazeny.

PUBLIC

Syntaxe:

PUBLIC symbol[,symbol]....

Direktiva se uvádí v každém modulu, který obsahuje symboly používané v ostatních modulech. Takto definované symboly jsou k dispozici ostatním modulům.

RECORD

Syntaxe:

jméno_věty RECORD jméno_pole:délka[=výraz][,jméno_pole: ....

Věta je určité rozvržení bitů v paměti, které se používá pro formátování bajtů a slov na bitové rovní.

COMMENT

Syntaxe:

COMMENT omezovač

Direktiva COMMENT slouží k oddělení následující části textu o libovolném počtu řádků (až po následující znak omezovače) jako poznámky. Jako omezovač lze použít libovolný přípustný znak kromě mezery nebo tabelátoru. Následující text je považován za poznámku až do chvíle, kdy je opět nalezen znak omezovače. Tímto způsobem lze „vypnout“ části programu.