PROFIMON

Bevezetés

A PROFIMON általános célú, gépi (assembler) nyelvű programok írására, fordítására és futtatására készült. A szokásos monitorokhoz képest többletszolgáltatása a beépített SZÖVEGSZERKESZTŐ (EDITOR), Z80 ASSEMBLER fordító és relokáló BETÖLTŐ (Loader) programok.

Jelen gépkönyvünk célja, hogy a PROFIMON lehetőségeit teljes részletességgel bemutassuk. Nem lehetett azonban a TVC Z80-as központi egységének gépi (assembler) rendszerét ismertetni helyhiány miatt. Ezért javasoljuk, hogy az assembler nyelven történő programozás mélyebb megértése céljából egy ezzel foglalkozó könyvet vásároljanak. (Javasoljuk Sztrókay Kálmán: A Z80 assembler c. művet, mert igen részletes ismertetést ad a programozáshoz, jó összefoglaló táblázatokkal nyújt segítséget, nem is említve, hogy eredeti Z80 (Zilog) elnevezésekkel dolgozik.)

Ugyancsak nem térhettünk ki a TVC hardware-software rendszerének ismertetésére, amely lényeges lenne esetleges programozási trükkök alkalmazására. Erre is tudunk azonban irodalmat ajánlani. Most jelenik meg a NOVOTRADE RT gondozásában Garai Géza: VIDEOTON TV COMPUTER operációs rendszere c. könyv.

Néhány fontos megjegyzés a PROFIMON felhasználói számára.
Szalagról való betöltés és indulás után a felhasználó számára a gép 64K RAM memóriája van kijelölve. Ennek felső 16K-s részében (U3) van a PROFIMON (C000H-F7FFH, és FF00H-FFFFH). A monitor belső programjai az 1B00H-BDFFH tartományt biztositják a felhasználó számára.

A TVC BASIC operációs rendszer ugyan a 19EFH címtől engedélyezi a RAM memória használatát, de itt van a PROFIMON újraindító rutinja. Ez esetleges program elakadás esetén egyszeri RESET gomb megnyomás után RUN utasítással tudja elindítani a monitort, ha ez az indítórész és a PROFIMON sértetlen. A PROFIMON a perifériák elérését a TVC BASIC operációs rendszeren keresztül biztosítja, így az ehhez szükséges alsó memóriatartományt nem szabad gépi kódú programozásra használni.

Ha gépi kódú programunkban használjuk a TVC operációs rendszer utasításait, akkor ügyeljünk arra, hogy töréspontot csak úgy helyezzünk el, hogy az operációs rendszert hívó RST 30 utasítást és a mögötte lévő adatot átugorjuk. (A PROFIMON ugyanis az operációs rendszerrel azonos címen van, és ezért nem tudja a beégetett EPROM memóriát vizsgálni.) Ügyeljünk a memórialapozásra! Az alsó (U0) 16K RAM-ot nem szabad kilapozni, mert ott van a rendszer stack és az RST 30 utasítás kiszolgáló rutinja is.

Az U2 és a VIDEO RAM memóriákat lapozhatjuk, ha éppen nem ezeken van a futó program! Ügyeljünk arra, hogy programjaink mindig az U2 és U3-t hagyják belapozva, mert a PROFIMON csak induláskor állítja be a lapszervezést (U0, U1, U2, U3-ra).
Csak igen gondos előtanulmányok után foglalkozzunk megszakításos (interrupt) programok írásával, mert ez önmagában is nehéz és a TVC hardware rendszerének kialakítása tovább bonyolítja.

A PROFIMON nem használja a TVC képernyő szerkesztő programját, ezért a botkormány nem használható. A monitor utasítások leírásánál ügyeljünk, mert azok nem módosíthatók, csak a . (pont) karakter leütése után újra írhatók. Rossz szám begépelése esetén (ha még határoló karaktert nem ütöttünk le) gépeljük tovább - most már helyesen - a számokat, mert a rendszer úgyis csak az utolsó 4 (ill. 2) számot veszi figyelembe.

A PROFIMON :MW és :MR rutinjai a mágnesszalagot vagy diszket blokkolt file formájában használják, mert csak így biztosítható nagyobb méretű forrásanyagok lefordítása. A SZÖVEGSZERKESZTŐ (EDITOR) kimenő adatait lehetőleg a kazetta elejére tegyük a könnyű visszaállás érdekében (a fordító program ugyanis egyes esetekben 2 menetes). Nem túl hosszú (<12kbyte) forrásnyelvi program esetén érdemes a memóriabázisú operációs rendszer (5. fejezet) lehetőségeit kihasználni (gyors, nem igényel közben magnetofont).

Ezek után lássuk, mit kínál kézikönyvünk!

Tartalomjegyzék 1

1. A PROFIMON a TVC gépi kódú (assembler) operációs rendszere

A PROFIMON operációs rendszer, mely a TVC gépi (assembler) programfejlesztő része, annak hatékony használatát biztosítja. Lehetőséget ad a memóriák, regiszterek tartalmának megjelenítésére, felülírására, kimeneti kapuk (portok) felülírására, bemeneti kapuk vizsgálatára, töréspont elhelyezésére. Alkalmas memóriaterület mágnesszalagról történő feltöltésére, ill. mágnesszalagra másolására, tetszőleges memóriacímre vezérlésátadásra.
A PROFIMON a 64k-s TVC memóriájának C000H-F7FFH címtartományában helyezkedik el. A PROFIMON 256 byte RAM-ot használ kisegítő és átmeneti tárolás céljára az FF00H-FFFFH tartományban.

1.1. Kisegítő tár (lásd 1 ábra)

A 256*8 bites kisegítő tár a PROFIMON változóinak átmeneti tárolására szolgál. Tartalmazza a CPU regisztereinek értékét vezérlésátadás előtt és megszakitás (break) pont elérésekor mint "tükör tár", valamint szabad területet a felhasználó által definiált mnemonikai kódok és végrehajtásukhoz szükséges rutinok számára.
A kisegítő tárban található a CPU regiszterek "tükör tár"-a. Ennek funkcióját a 2.ábra segítségével mutatjuk be. A Z80 CPU regisztereinek állapotát, miközben a felhasználói program tesztelését végezzük, a PROFIMON a kisegítő tárba másolja egy töréspont (break point) elérésekor.
A PROFIMON utasításaival kiírathatjuk, és/vagy felülírhatjuk ezen "tükör tár" memória tartalmát.
Ha a G utasítással a vezérlést visszaadjuk a felhasználói programnak, a CPU regisztereibe a PROFIMON visszairja a "tükör" regiszter táblát, és amennyiben értéküket módosítottuk, a CPU ezen új értékekkel folytatja a felhasználói program futtatását.

FFFF	CPU regiszterek "tükör tár"-a
FFE6	
	Felhasználó által definiált határ
	Felhasználó által definiált mnemonikai kódok
FF33	I/O meghajtók kezdőcímei a 6 I/O csatornához
FF27	PROFIMON flag és
FF00	mutatótára

1.ábra. Kisegítő tár

CÍM	REGISZTER	
FFFF	PC	MSB
FFFE		LSB
FFFD	A	
FFFC	F	
FFFB	I	
FFFA	IF	
FFF9	B	
FFF8	C	
FFF7	D	
FFF6	E	
FFF5	H	
FFF4	L	
FFF3	A'	
FFF2	F'	
FFF1	B'	
FFF0	C'	
FFEF	D'	
FFEE	E'	
FFED	H'	
FFEC	L'	
FFEB	IX	MSB
FFEA		LSB
FFE9	IY	MSB
FFE8		LSB
FFE7	SP	MSB
FFE6		LSB

2.ábra. CPU regiszterek "tükör tár"-a

A kisegítő tárban található a kimenő és bemenő logikai és fizikai eszközöket összerendelő tábla (lásd 3.ábra). Segítségével rendkívül rugalmas periféria vezérlést valósíthatunk meg.
A hat logikai csatornának megfelelő 6*2 byte-ba a meghajtó programok kezdőcímeit helyezhetjük, így a

rendszerhez a perifériák széles skálája csatlakoztatható. A felhasználó a rendszerhez egyedi perifériát is csatlakoztathat, amennyiben a megírt és a memóriában elhelyezett periféria meghajtó program kezdőcímét a tábla adott csatornájának megfelelő 2 memóriarekeszbe írja.

FF32	Forrás ki meghajtója	MSB	
FF31		LSB	:SO
FF30	Forrás be meghajtója	MSB	
FF2F		LSB	:SI
FF2E	Tárgy ki meghajtója	MSB	
FF2D		LSB	:OO
FF2C	Tárgy be meghajtója	MSB	
FF2B		LSB	:CI
FF2A	Consol ki meghajtója	MSB	
FF29		LSB	:CO
FF28	Consol be meghajtója	MSB	
FF27		LSB	:CI

3.ábra. I/O csatornák meghajtóinak kezdőcímtára

:CI, :CO- A Consol Input és Consol Output csatorna, a felhasználóval történő kommunikálásra szolgál (kezelői utasítások, hibaüzenetek, stb.).

:OI, :OO- Az Object Input és Object Output tárgy csatorna a gépi orientált mágnesszalagról történő memóriafeltöltésre, ill. memóriamező tartalmának kivitelére szolgál (természetesen mágnesszalag helyett floppy-disc is használható).

:SI, :SO- A Source Input és Source Output forrás csatornákat a Z80-ASSEMBLER és a SZÖVEG-SZERKESZTŐ (EDITOR) parancsok használják.

1.2. A PROFIMON aktivizálása

A TVC RESET gombját benyomva, a PROFIMON mágnesszalagját LOAD utasítással beolvassuk. A bejelentkező TVC 64K PROFIMON 1.2 szöveg után soremelés, kocsi vissza és %-jel kiküldése jelzi, hogy a rendszer kész az utasítások fogadására.
A funkció betű és az esetleges operandusok megadása után a rendszer az utasítást egy zárókarakter leütése után végrehajtja.
A továbbiakban a PROFIMON soremelés, kocsi vissza és %-jel kiküldésével jelzi, hogy az utasítást végrehajtotta, és kész a következő utasítás fogadására.

1.3. Utasításformátum

Funkció kód	Operandus (hiányozhat is)	Zárókarakter
Az angol nagy ABC betűi (A-Z)	- Hexadecimális (0...9, A-F) karakterek - Mnemonikai kódok (:CO :SI, stb.) - $	- kocsi vissza (return, <cr>) - ↑ - .

1.4. Operandusok aaaa+bbbb+cccc+.....+zzzz művelet is lehet

A szám előjelét pozitívnak veszi, ha előjel beírás külön nem történik.
Ha két számjegyes az operandus, az utolsó két számjegyet veszi figyelembe.

A műveletek tagjai lehetnek:

- hexadecimális számok, melyből az utolsó négyet veszi figyelembe és a kezdő nullák elhagyhatók,
- :KK mnemonikai kódok, melyek 4 számjegyes hexadecimális címet jelentenek (lásd 4. ábra, és az 1.8. fejezet),
- $, amely egyes funkcióknál a folytató címet jelenti,
- egyenlőségjel, amelynek leütése után kiírja a művelet kiszámított értékét.

Ha az utasítás egynél több operandust igényel, az egyes operandusokat vessző, vagy szóköz karakterrel kell elválasztani.

MNEMONIK	CÍM	ÉRTELMEZÉS (*=2 byte)
:PC	FFFE	felhasználói PC regiszter*
:A	FFFD	" A "
:F	FFFC	" F "
:I	FFFB	" I "
:IF	FFFA	" IF "
:B	FFF9	" B "
:C	FFF8	" C "
:D	FFF7	" D "
:E	FFF6	" E "
:H	FFF5	" H "
:L	FFF4	" L "
:A'	FFF3	" A' "
:F'	FFF2	" F' "
:B'	FFF1	" B' "
:C'	FFF0	" C' "
:D'	FFEF	" D' "
:E'	FFEE	" E' "
:H'	FFED	" H' "
:L'	FFEC	" L' "
:IX	FFEA	" IX " *
:IY	FFE8	" IY " *
:SP	FFE6	" SP " *
:CI	FF27	consol bemenet csatorna kezdőcíme*
:CO	FF29	" kimenet " *
:OI	FF2B	tárgy bemenet " *
:OO	FF2D	" kimenet " *
:SI	FF2F	forrás bemenet " *
:SO	FF31	" kimenet " *

4.ábra. A PROFIMON mnemonikai kódjai.

1.5. Zárókarakterek

Kocsi vissza (RETURN billentyű, továbbiakban <cr>) hatására a PROFIMON végrehajtja az utasítást. Megjelenítéskor (S és P utasítás) kiírja a soron következő memóriacím tartalmát, mnemonikai kód értékét, vagy a bemeneti kapuról beolvasott adatot.

A ↑ (SHIFT N): csak a P és S utasításokra érvényes. Az S utasítással módosított memóriarekesz értékét ismét kiírja.
Ha nincs változtatás, a ↑ zárókarakter hatására az előző memóriacím tartalmát, mnemonikai kód értékét, vagy a bemeneti kapuról beolvasott adatot írja ki.

A . (pont): törli az utasítást, soremelés, kocsi vissza, %-jel kiküldésével jelzi a PROFIMON, hogy kész az új utasítás fogadására.

1.6. Az utasítások részletes ismertetésénél használt jelölések

- aa 2 számjegyes hexadecimális értéket jelöl,
- aaaa 4 számjegyes hexadecimális értéket jelöl,
- t zárókaraktert jelöl.

1.6.1. Memória tartalom megjelenítése és felülírása

%S aaaat - aaaa a lekérdezett memóriarekesz hexadecimális címe.

Példák:

- Relatív ugrás utasítás betétele a 8100-as cimre és az argumentum számítása 8100-ról 8139-re.

```
%S 8100<cr>
8100 00 18<cr> JR utasításkódja
8001 00 8139-$=0037↑
8001 37 .
%
```

- Kimenet/bemenet csatornameghajtó program átrendelése.

```
%S :SI<cr>
:SI :MR :IB<cr> - a forrás bemenő csatorna a B memória-
                  terület lesz (lásd később),
:SO :MW :LP<cr> - forrás csatornakimenet a sornyomtató
                  lesz,
FF33 80 .       - azon mnemonikai tábla kezdete, ahová a
                  felhasználó definiálhat pótlólagos
                  mnemonikai kódokat.
%
```

- Pótlólagos mnemonikai kód definiálása (a kód neve :MN lesz, végrehajtási címe FF86).

```
%S FF33<cr>
FF33  80  4D<cr>   ASCII M
FF34  00  4E<cr>   ASCII N vagy N+80 ha 2 byte-os
FF35  00  86<cr>   cím alsó (LSB) byte
FF36  00  FF<cr>   cím felső (MSB) byte
FF37  00  80<cr>   mnemonikai tábla új vége
                   (80=zárókarakter)
FF38  00  .
%
```


1.6.2. Kimeneti kapun adat felülírása, bemeneti kapu kiolvasása

%P aat - aa a kimeneti kapu hexadecimális címe.

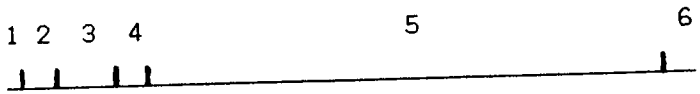
Példa:

```
- %P 03<cr>   03H kimeneti kapura C8H-t ír
  03 F2 C8 .  03H bemeneti kapun az adat F2
              volt
  %
```

1.6.3. Memóriamező tartalmáról INTEL HEX formátumú adatfile készítése (Write) az :00 periférián

(formátumot lásd 5.ábra.)

%W aaaa,bbbb<cr> - aaaa, bbbb a memóriamező határai.




```
:100000007A2F577B2F5F132100003E11E519D21282
:00000001FF
```

1. sor kezdőjel
2. adat darabszám a sorban
3. betöltés kezdőcíme
4. sor tipusa (00=nem utolsó sor, 01=utolsó sor)
5. adatok
6. ellenőrző összeg

5.ábra. A W utasítás által előállított "INTEL-HEX" formátum.

1.6.4. A W utasítással készített adatfile visszatöltése (Read) az :OI perifériáról

%R<cr> - checksum (ellenörző összeg) hiba esetén a hibás sort követő sor kezdőcímét kinyomtatja.

1.6.5. Vezérlésátadás (Go)

%G aaaa<cr> - vezérlésatadás az aaaa címre, azaz a felhasználói program végrehajtása az aaaa címtől kezdve.

%G<cr> - vezérlésátadás a "tükör" regisztertábla PC regisztere által meghatározott címre.

Visszatérés: egy előzőleg elhelyezett töréspontttal, vagy egy JP E000H utasítással (TVC memória lapszervezésre ügyelni!).

1.6.6. Hexadecimális aritmetika

%H +-aaaa+-bbbb.....+-yyyy=zzzz<cr>

1.6.7. Memóriablokk tartalmának áthelyezése (Copy)

%C aaaa,bbbb,cccc<cr> - aaaa - bbbb memóriablokkot cccc címre másolja.

1.6.8. Töréspont utasítás (Break point)

%B aaaa,b<cr> - aaaa töréspont hexadecimális címe.
Ha b=1, kiírja a CPU összes regiszterének tartalmát, a töréspont elérésekor, majd visszatér az operációs rendszerbe.
Ha b=0, vagy a töréspontcímet közvetlenül zárókarakter követi, a töréspont elérésekor kiírja a PC,A,F regiszterek tartalmát, majd visszatér az

operációs rendszerbe.
Ezután soremelés, kocsi vissza, %-jel kiküldésével jelzi a PROFIMON, hogy kész a következő utasítás fogadására. Egyidőben egyszerre, csak egyetlen töréspont definiálható.
Ha a CPU végtelen ciklusba kerül RESET-tel lehet visszatérni a BASIC operációs rendszerbe, újra indítva a PROFIMONT majd a töréspontot törölni kell. Mivel a töréspont definiálása nem más, mint a töréspont címre elhelyezett 3 byte-os - ugorj a PROFIMON "töréspont" programcímére - utasítás, törlésre két lehetőség van:
- az ugrás utasítás helyére S utasítással visszaírjuk az eredeti 3 byte-ot,
- vagy a %B<cr> utasítással töréspont törlést hajtunk végre.

1.6.9. CPU regisztertartalmak kinyomtatása (eXamine registers)

%X <cr> - CPU regisztereinek kinyomtatása fejléc nélkül
%X 1<cr> - CPU regisztereinek kinyomtatása fejléccel. IF jelentése a fejlécben: interrupt flip-flop.
Ha IF=0 interrupt letiltás állapot, ha IF=1 interrupt engedélyezve állapot volt a töréspont elérése pillanatában.

1.6.10. %A<cr>

utasítással léphetünk be az Assembler fordító programba (lásd 3. fejezet).

1.6.11. %D aaaa,bbbb<cr>

(Dump) utasítással a konzolra (TV) kiírathatjuk az aaaa-bbbb memóriaterület tartalmát hexadecimális formában. Ha az adott memóriarekesz tartalma egy megjeleníthető ASCII kódnak felel meg, akkor a megfelelő betű vagy jel képe megjelenik a tartalom alatt. Az aaaa értékét a program automatikusan aaa0-ra változtatja.
%D aaaa<cr> utasítással az aaa0 címtől kezdődően kb. 70

memóriacím tartalmát jelenithetjük meg.
%D $<cr> utasitással az előzőleg éppen befejezett D utasitást folytathatjuk újabb cím leirása nélkül. (A $ változó tárolja az utolsó megjelenitett címet.)

1.6.12. %E<cr>
utasitással indítjuk először a PROFIMON EDITOR (SZÖVEGSZERKESZTŐ) programját (lásd 2. fejezet). Ha a PROFIMON Memória bázisú operációs rendszerét (lásd 5. fejezet) használni akarjuk, akkor az utasitás előtt %M utasitással osztjuk fel a memóriát 3 részre. Hosszabb programok irása esetén viszont ne használjuk "szűz indítás" után a %M utasitást, mert igy a teljes memóriaterület "A" területként áll az EDITOR (SZÖVEGSZERKESZTŐ) rendelkezésére kb. 25 kbyte kapacitással.

1.6.13. %F aaaa,bbbb,cc<cr>
utasitással feltölthetjük (FILL) az aaaa-bbbb memóriatartomány valamennyi címét a cc byte-tal.

1.6.14. %I f,aaaa<cr>
utasitással (INPUT) beolvashatunk a mágnesszalagról egy programot vagy adattömböt az aaaa memóriacímtől kezdve. A program neve az f értéke, ami A, B, C, D, E, F lehet. A beolvasandó file-t %O (Output) utasitással vagy BASIC SAVE utasitással kellett előzőleg létrehozni (blokkolatlan file). Az utasitás végrehajtása után esetleges hibajelzés a $ változóban. Ez %H $= utasitással irható ki. Ha nincs hiba $=0, különben a TVC operációs rendszer hibakódja jelenik meg.

1.6.15. %K aaaa,bbbb,cccc<cr>
utasitással az aaaa-bbbb memóriatartomány tartalmát hasonlithatjuk össze a cccc címen kezdődő memóriatartománnyal (Komparálás). A különbségeket a képernyőn irja ki. Bármely billentyű leütésével az utasitás végrehajtása megállitható.

1.6.16. %L aaaa,bbbb<cr>
utasitással az aaaa-bbbb memóriatartományban levő programot listázhatjuk ki úgy, mintha az assembler fordító program listáját jelenitenénk meg. Az egyes sorok között a továbblépés a RETURN billentyűvel történik. (Visszafordítás vagy disassemblálás.)

1.6.17. %O f,aaaa,bbbb<cr>
utasitással az aaaa-bbbb memóriatartomány tartalmát vihetjük ki mágnesszalagra vagy diszkre (Output) blokkolatlan file formában. A file neve f, amely A, B, C, D, E, F lehet.

1.6.18. %Q aaaa,bbbb<cr>
utasitással az aaaa-bbbb memóriatartomány tartalmának aritmetikai összegét képezhetjük. (Ellenörző összeg képzés.)

1.6.19. %T aaaa,bbbb,c<cr>
a Trace utasitás segitségével az aaaa címen kezdődő programot bbbb=0 esetén egyes utasitásonként (különben bbbb lépésközzel) lépésenként hajthatjuk végre. Az egyes lépések között a Z80 központi egység állapota kijelzésre kerül a c paramétertől függő módon:

c=0 csak PC és AF,
c=1 teljes regiszter térkép (a %X utasitásnak megfelelően),
c=2 a visszafordított utasitás Z80 assembler mnemonikja valamint a PC és AF,
c=3 a visszafordított utasitás Z80 assembler mnemonikja és a teljes regiszter térkép.

1.7. A PROFIMON hívható szubrutinjai
(a kezdőcímeket lásd az 1. függelékben)

A fejezet a PROFIMON hívható szubrutinjainak rövid leírását tartalmazza. Ezen szubrutinokat a felhasználó saját programjaiba építheti.

1.7.1. RDCHR

Az E regiszterben kijelölt csatornáról egy ASCII karaktert beolvas.

Bemenő paraméterek:
- E regiszter - csatornakijelölő regiszter
  - E = 0,1 consol csatorna
  - E = 2,3 tárgy csatorna (object)
  - E = 4,5 forrás csatorna (source)

Kimenő adatok:
- D, A regiszter - az ASCII kódot tartalmazza (a paritás bitet törli).
- E regiszter - változatlan

1.7.2. WRCHR

Az E regiszterben kijelölt csatornára egy ASCII karaktert ír.

Bemenő paraméterek:
- E regiszter - csatornakijelölő regiszter.
  - E = 0,1 consol csatorna
  - E = 2,3 tárgy csatorna
  - E = 4,5 forrás csatorna
- D - regiszter - a kiírandó karaktert tartalmazza.

Kimenő adatok:
- D regiszter - tartalmazza a kiírt karaktert
- E regiszter - változatlan
- A regiszter - nulla

1.7.3. PACC

Az E regiszterben kijelölt csatornára az A regiszter felső és alsó 4 bitjének tartalmát ASCII karakterként írja ki. Pl:
A = 4EH, az ASCII 4 (34H) és E (45H) karaktereket írja ki.

Bemenő paraméterek:
- E regiszter - csatornakijelölő regiszter, ugyanúgy, mint WRCHR-nél
- A regiszter - a két nyomtatandó hexadecimális számjegy értékét tartalmazza bináris formában.

Kimenő adatok:
- A regiszter - nulla

Belső szubrutinhívás:
- PRVAL, WRCHR

1.7.4. PRVAL

Az A regiszter alsó 4 bitjének megfelelő ASCII karaktert előállítja.

Bemenő paraméterek:
- A regiszter - alsó négy bitje tartalmazza az átalakítandó hexadecimális számjegyet.

Kimenő adatok:
- A,D regiszter - tartalmazzák a megfelelő ASCII kódot.

1.7.5. ECHO

Az E regiszterben kijelölt csatornáról beolvasott karaktert ugyanarra a csatornára kiírja.

Bemenő paraméterek:
E regiszter - csatornakijelölő regiszter, ugyanúgy mint RDCHR, WRCHR-nél

Kimenő adatok:
D regiszter - a beolvasott és kiírt karaktert tartalmazza
A regiszter - nulla

Belső szubrutinhívás:
RDCHR, WRCHR

1.7.6. CRLF

Az E regiszterben kijelölt csatornára soremelés, kocsi vissza karaktereket küld ki.

Bemenő paraméterek:
E regiszter - csatornakijelölő regiszter, ugyanúgy mint WRCHR-nél

Kimenő adatok:
D regiszter - soremelés (0AH) karaktert tartalmaz
A regiszter - nulla

Belső szubrutinhívás:
WRCHR

1.7.7. SPACE

Az E regiszterben kijelölt csatornára szóköz (20H) karaktert küld ki.

Bemenő paraméterek:
E regiszter - csatornakijelölő regiszter, ugyanúgy mint WRCHR-nél

Kimenő adatok:
D regiszter - space (szóköz) (20H) karaktert tartalmaz
A regiszter - nulla

Belső szubrutinhívás:
WRCHR

1.7.8. PTXT

A HL regiszterpárban tárolt címtől kezdve amíg ETX(03H) karaktert talál, a címek tartalmát ASCII karakternek veszi, és az E regiszterben kijelölt csatornára kiírja.

Bemenő paraméterek:
E regiszter - csatornakijelölö regiszter, ugyanúgy mint WRCHR-nél
HL regiszterpár - a szövegmező kezdőcimét tartalmazza (H-ban a cím felső, L-ben alsó byte-ját)

Kimenő adatok:
D regiszter - az ETX(03H) karaktert tartalmazza
HL regiszterpár - ETX karakter címét tartalmazza
A regiszter - nulla

Belső szubrutinhívás:
WRCHR

1.7.9. ASBIN

Az A regiszterben lévő hexadecimális számjegynek megfelelő ASCII karaktert bináris megfelelőjévé alakítja.

Bemenő paraméterek:
A regiszter - tartalmazza az ASCII kódot

Kimenő adatok:
A regiszter - tartalmazza a megfelelő bináris értéket

1.8. I/O meghajtó programok

- Szubrutinként hívhatók a következő I/O meghajtók

:TV	- monitor képernyő
:BB	- billentyűzet
:MR	- magnetofon lejátszás (Read)
:MW	- magnetofon felvétel (Write)
:IA	- az A memóriaterület olvasása (input)
:IB	- a B memóriaterület olvasása (input)
:OB	- a B memóriaterület írása (output)
:LP	- sornyomtató (Line Printer).

Bemenő paraméterek:
D regiszter - adatkivitelnél adatregiszter

Kimenő adatok:

E regiszter	- nullázza a 3. és 7.bitjét
D regiszter	- a kiírt adatot kivitelnél tartalmazza
A regiszter	- nullázza adatkivitelnél
A,D regiszter	- adatbeolvasásnál a behozott adatot tartalmazza.

A bemenő és kimenő paraméterek figyelembevételével a felhasználó egyedi periféria meghajtó programot írhat. A megírt programot RAM-ban tárolhatja.

- Indulásnál vagy újraindításnál (JP E000) a:

:CI	csatornához	:BB	billentyűzet
:CO	"	:TV	televízió ernyő
:OI	"	:MR	magnetofon lejátszás
:OO	"	:MW	magnetofon felvétel
:SI	"	:MR	magnetofon lejátszás
:SO	"	:MW	magnetofon felvétel

funkciókat rendeli a rendszer.

1.9. Mágnesszalag ill. diszk műveletek

esetén, ha az utasítás külön file-név jelzést nem tartalmaz, akkor az olvasó jellegű műveletek az 1621H memóriacímen levő ASCII karaktert használják file névként. A rendszer indulásánál ott 41H van, ami az A betűnek felel meg.
Iró jellegű műveletek esetén az 1623H memóriarekesz címét használja a rendszer, ahol 44H található induláskor, ami a D betűnek felel meg.

2. A Text Editor, a PROFIMON SZÖVEGSZERKESZTŐ programja

A SZÖVEGSZERKESZTŐ- vel a RAM-ba töltött forrásprogramok szerkesztését, írását végezhetjük el.

2.1. Definíciók

- Forrás - forrásprogram, amit a felhasználó ír.
- Sor - a forrásprogram egy sora kocsi vissza karakterrel lezárva.
- Munkaterület - az a RAM terület, ahol a forrásprogramot tároljuk.
- Sormutató - a munkaterület egy sorának kezdetét jelenti. Az utasítások erre a sorra vonatkoznak.
- Aktuális sor - a sormutató által meghatározott sor.
- Beszúrás - az aktuális sor után új sor beírását jelenti.
- Törlés - az aktuális sor törlése.

2.2. Beugrás a SZERKESZTŐ programba

A PROFIMON %E<cr> utasításával.
A SZERKESZTŐ program a > karakter kiküldésével jelzi, hogy kész az utasítás fogadására.
Egy sorban több utasítás is adható, ha az egyes utasításokat betűközzel választjuk el.
Visszatéréskor a %U<cr> utasítást használjuk. Ez az utasítás beugráskor nem törli a munkaterületet. (Újra belépés a SZÖVEGSZERKESZTŐBE.)

2.3. Felhasznált I/O csatornák

- :CI utasítások és módosítások bevitelére
- :CO visszajelzések a SZERKESZTŐ programtól. V utasítás esetén munkaterület tartalmának listázására.
- :OO P utasítással a megszerkesztett modul kivitelére
- :SI R utasítással forrásmodul bevitelére a munkaterületre
- :SO munkaterület tartalmának listázására a sorszámokkal W utasítás esetén

2.4. Utasításkészlet

Az utasítások azonosítóból és egy n 0-9999D tartományba eső operandusból állnak. Ha az operandust elhagyjuk, a SZERKESZTŐ nullának tekinti.
A consol csatornán beadott karaktereket törölhetjük.
- egy karaktert, (a mindenkori utolsót) DEL karakterrel törölhetünk
- control-U karakterrel a teljes sort törölhetjük.

Az ESC karakter leütésekor visszatér a PROFIMON operációs rendszerbe. Ekkor lehetőség van az I/O meghajtók átállítására és %U paranccsal visszatérni ismét a SZÖVEGSZERKESZTŐBE az eredeti szöveg megtartásával.

2.4.1. R

a :SI csatornáról beolvassa (Read) a forrásprogramot. Addig olvas, amíg EOT(04H), vagy ETX(03H) karaktert talál, vagy amíg a munkaterület megtelik. A következő R utasítás az előzőleg betöltött modul után tölt. Minden file beolvasása után az utolsó sorban megjelenik a file végét jelző karakter. Ezt egy P vagy W utasítás előtt feltétlenül töröljük ki!

"ESC" = ESCAPE ASCII karakter (1BH)
"ETX" = END OF TEXT " (03H) (CTRL-C)
"STX" = START OF TEXT " (02H) (CTRL-B)
"EOT" = END OF TRANSMISSION karakter (04H) (CTRL-D)

2.4.2. Bn

a sormutatót n-el csökkenti (Back). Ha n=0 a sormutató az első sor kezdetére fog mutatni.

2.4.3. An

a sormutatót n-el növeli (Advance). Ha n=0 a sormutató az utolsó sor kezdetére fog mutatni.

2.4.4. Pn

az :00 csatornán az aktuális sortól kezdve n sort kiír (Print). Ha n=0 az aktuális sorral kezdve végig kiírja a munkaterületet (sorszám nélkül).
A kiírás előtt STX (CTRL-B) karaktert visz ki, a kiírás befejezése után ETX (CTRL-C) karaktert.

2.4.5. Sn "karaktersorozat"

az aktuális sortól kezdve a "karaktersorozat" előfordulását figyeli (Search), a sormutató értéke azon sor címe lesz, ahol n.-szer előfordult. Ha n=0 a sormutató értéke azon sor címe lesz, ahol először előfordult. Az utasítást és a karaktersorozatot vesszővel vagy betűközzel kell elválasztani. A karaktersorozat határoló karaktere tetszőleges a "karaktersorozat"-ban nem előforduló karakter lehet.

2.4.6. Cn "karakterek1"karakterek2"

az aktuális sortól kezdve ahol "karakterek1"-t talál "karakterek2"-re változtatja (Change). A "karakterek1" n-dik előfordulásáig módosít.
Ha n=0 a "karakterek1" első előfordulását "karakterek2"-re változtatja.

2.4.7. I

az aktuális sor után sort, vagy sorokat beszúr (Insert). Az egyes sorokat kocsi vissza karakterrel kell elválasztani, a soremelést a SZERKESZTŐ program generálja.
A < karakter kiküldésével jelzi, hogy kész a sor beszúrására. Ha a beszúrást befejeztük üres sorral jelezzük ezt a programnak (a < jel után rögtön CR). Ha BUFFER FULL üzenetet küldi a program, a nem befejezett sort control-U karakterrel töröljük, és üres sorral lezárjuk a beszúrásokat.
Üres sor után a SZERKESZTŐ program > karakter kiküldésével jelzi, hogy kész a következő utasítás fogadására.

2.4.8. T

a forrásprogram elejére szúrhatunk be sorokat. Szabályai ugyanazok, mint az I utasításnál.

2.4.9. Dn

az aktuális sortól kezdve n sort töröl (Delete). Ha n=0 az aktuális sort törli.

2.4.10. Wn

a :S0 csatornán az aktuális sorral kezdve n sort, sorszámmal együtt kiír (Write). Ha n=0 az aktuális sort írja ki.
A sormutató változatlan.

2.4.11. Vn

ugyanaz, mint Wn, csak a :C0 csatornán ír ki.

2.4.12. Ln

a sormutató értékadó utasítása, n nem lehet nulla (Line). Csak növekvő sorszámok irányába működik.

2.4.13. N

a munkaterület első és utolsó sorának számát és az aktuális sor számát írja ki a :CO csatornán (Numbers).

2.4.14. Mb

(b=1,2) a SZERKESZTŐ utasításaiból leírhatunk egy sort (az egyes utasítások között betüközzel elválasztva) amelyet a rendszer nem hajt végre, csak eltárol (utasítás MACRO).

2.4.15. Xb

(b=1,2) az előzőleg definiált M1 ill. M2 utasítás macro végrehajtása. Segítségével a többször ismételendő feladatokat csak egyszer kell leírni és több alkalommal lehet használni.
Megj.: Csak az eredeti forrás sorokat számozza a SZÖVEGSZERKESZTŐ, a beszúrt sorok sorszáma 0000 lesz, a törölt soroknak a sorszámát is törli.
Ha a SZERKESZTŐ program BUFFER FULL üzenetet küld, a rendelkezésre álló (max.80) üres byte lehetőséget biztosit a félbemaradt sor - amelyet éppen beszúrtunk - befejezésére.

Az STX (02H) karaktert a SZERKESZTŐ program az első sor karakterének tekinti. Igy célszerű az STX karakter után egy kommentet elhelyezni, amit kocsi vissza karakterrel lezárunk.
Igy nem fordulhat elő, hogy az STX karaktert, ha azt nem követi kocsi vissza karakter a forrásprogram első sora elé helyezi. (STX, ETX karakterek használatát lásd az ASSEMBLER fordító program 3.7. pontjában.)

3. Z80 ASM a TVC assembler fordító programja

A Z80 ASM program a TVC RAM memóriájában a C000H kezdőcímtől található. A SZÖVEGSZERKESZTŐ és SZERKESZTŐ-BETÖLTŐ programokkal együtt a Z80-as CPU-ra írt programok szerkesztését, assembler fordítását, összefűzését, betöltését végzi el.

A Z80 ASM a Z80 mnemonikai utasításait, az assembler rendszer pszeudó utasításait beolvassa, assembler listázást végez és előállitja a tárgyprogramot. A Z80 ASM a PROFIMON rendszerben működik, felhasználja annak:

- consol I/O csatornáját
- source I/O csatornáját
- object O csatornáját
- min. 4K RAM-ot
- PROFIMON OPERÁCIÓS RENDSZER-t,

valamint a SZÖVEGSZERKESZTŐ és SZERKESZTŐ-BETÖLTŐ szubrutinjait.

A Z80 ASM által lefordítható programok méretét a felhasználó által definiált cimketábla mérete határolja. A forrás kódban a sorhossz max. 7FH (127D) lehet!

3.1. Definiciók

3.1.1. Forrás modul

- a felhasználó forrásnyelvű programja. A Z80 ASM ezt tárgy modulra fordítja.
Megj.: - a forrásmodult END pszeudó utasítással vagy EOT(04H) karakterrel kell lezárni.
- a forrásmodult a PROFIMON :SI csatornáján olvassa be.

3.1.2. Tárgy modul

- a TVC-ASM kimeneti modulja, mely ASCII kódokban tartalmazza
  - a program gépi kódú fordítását,
  - információkat, címeket a SZERKESZTŐ-BETÖLTŐ program számára,
  - ellenőrző összeg információt.

3.1.3. Lista modul

- a program gépi kódú fordításának és az eredeti forrás modulnak kombinálásával kialakuló olvasható modul.

3.1.4. Lokális cimke

- a forrásprogram cimkéje, amely megjelenik a cimkemezőben.

3.1.5. Belső cimke

- a GLOBAL pszeudó utasítással definiálható, ugyanannak a forrásmodulnak a cimkemezejében megjelenő cimke. A SZERKESZTŐ-BETÖLTŐ program által összefűzött tárgymodulok mindegyike felismeri.

3.1.6. Külső cimke

- a GLOBAL pszeudó utasítással definiálható, de nem jelenik meg ezen modul cimkemezejében, viszont másik modulban mint belső cimke definiált.

3.1.7. Globális cimke

- több modulban található cimke. Minden modulban, ahol előfordul a GLOBAL pszeudó utasítással definiált. Azt,

hogy külső, vagy belső cimke az ASM dönti el.

3.1.8. Kezdőcím független

- az a modul, amely tetszőleges kezdőcímen elhelyezhető és ehhez a tárgy modulban nem igényel kiegészítő információt.

3.1.9. Áthelyezhető

- az a program, amely a SZERKESZTŐ-BETÖLTŐ program részére kiegészítő információt tartalmaz a tárgymodulban, így tetszőleges kezdőcímre helyezhető.

3.1.10. Abszolút

- az a program, amely nem tartalmaz kiegészítő információt a tárgymodulban, így csak egy, a program eredeti (ORG pszeudó utasítással definiált) kezdőcímére tölthető. (Csak INTEL HEX formátumú adatot tartalmaz.)

3.1.11. Összefűzhető

- az a program, mely a tárgymodulban tartalmazza a külső és belső cimkékre vonatkozó információkat, amelyek segítségével a SZERKESZTŐ-BETÖLTŐ program összefűzi az egyes modulokat.

3.2. Szintaxis:

A forrásmodul cimkéket, Z80 utasításokat, pszeudó utasításokat és megjegyzést (comment) tartalmaz.
Az egyes elemeket egy vagy több szóközzel, vesszővel, vagy CTRL/I (=tabulátor) karakterrel (ASCII 09H) kell elválasztani. A cimkét és a műveleti kódot kettősponttal is elválaszthatjuk.

3.2.1. Cimkék

A cimke egy, vagy több karakterből állhat. Ha több mint hat karaktert tartalmaz a TVC-ASM csak az első hat karaktert veszi figyelembe.

A cimke első karaktere nem lehet szám (0-9).
A cimke nem tartalmazhatja a következő karaktereket: -'()"+,-<>=./:; és szóköz.

A cimke tetszőleges helyen kezdődhet, ha azt kettőspont követi. Ha a lapszélen kezdődik, a kettőspont a cimke után elhagyható.

3.2.2. Utasítás kódok

Z80 utasításkészletének kódjai, amelyek megfelelnek az egyes mnemonikoknak.

3.2.3. Operandusok lehetnek:

- mnemonikai kódok, lásd 6.ábrát.
- cimkék,
- konstansok: - decimális - D karakter követi a számot (ez el is hagyható),
  - hexadecimális - H karakter követi a számot (0-9-ig terjedő valamilyen számmal kell kezdődnie),
  - oktális - Q vagy O karakter követi a számot,
  - bináris - B karakter követi a számot.
  - ASCII kód pl. 'A' = 41H.

Megjegyzés: - cimke definiálásakor csak olyan cimke lehet az operandusmezőben, amelyet előzőleg már definiáltunk.
- a $ karakter a "programszámláló pillanatnyi értéke+1" jelenti.

A - A regiszter
B - B regiszter
C - C regiszter
D - D regiszter
E - E regiszter
F - F regiszter (flag regiszter)
H - H regiszter
L - L regiszter
AF-AF regiszterpár
AF'-AF' regiszterpár
BC-BC regiszterpár
DE-DE regiszterpár
HL-HL regiszterpár
IX-IX indexregiszterpár
IY-IY indexregiszterpár
SP - zsákmemória címregisztere
$ - programszámláló aktuális értéke+1
I - interrupt regiszter
R - memóriafelfrissítés regisztere
NZ - nem nulla
Z - nulla
NC - carry nem
C - carry
PO - paritás páratlan/nincs túlcsordulás
PE - paritás páros/túlcsordulás
P - pozitív előjel
N - negatív előjel

6.ábra. Assembler operandusok mnemonikai kódjai

Műveletek az operandusmezőben.

Műveleti jelek: + - .

A művelet végén leütött . karakter a kifejezés értékét jobbra shifteli 8 bittel úgy, hogy a felső 8 bitbe 0-kat tölt, pl. 0BC93H. = 00BCH

Zárójel használata engedélyezett.

Ha a teljes kifejezés zárójelben van az memóriacímet jelent, és ezen memóriarekesz tartalma kerül az operandusmezőbe.

TVC-ASM hibaüzenetet küld, ha az operandus értéke túllépi a 0-0FFFFH tartományt, vagy relatív ugrás (JR) műveleti kód esetén a -126-+129D tartományt.

Egy kifejezésben max. 15 műveleti jel, konstans, cimke használható.

Relatív ugrás, DJNZ utasítás operandusa CIMKE-$ kifejezéssel számítható.

Külső cimke nem használható a műveletekben.

3.2.4. Kommentek (megjegyzések)

Pontosvessző után következhetnek.
Ha * karakter van a sor elején, a teljes sort megjegyzésnek tekinti.
A kommenteket a TVC-ASM nem fordítja, de assembler listázáskor kinyomtatja.

3.2.5. TVC-ASM pszeudó utasításai

A program írásakor az utasítás mezőbe (nem közvetlenül a sor elejére) írjuk a pszeudó utasításokat.

- ORG nn - program kezdőcíme.

- CIMKE EQU nn - cimke értékadó utasítása, mely bármely cimkére, de csak egyszer használható.

- CIMKE DEFL nn - cimke értékadó utasítása, ugyanazon cimkére többször is használható.

- DEFM 'aa' - a soron következő memóriarekeszeket az idézőjelbe tett karakterek ASCII kódjaival tölti fel. 'aa' maximum 62 karaktert tartalmazhat.

- DEFB n - a soron következő byte-ba n-t tölt. Az n lehet pl: 'A' alakú is, ami az A betű ASCII kódját jelenti (41H).

- DEFW nn - a soron következő byte-ba nn alacsonyabb az utána következő byte-ba nn magasabb helyiértékű byte-ját tölti.

- DEFS nn - nn byte-ot lefoglal a memóriában a soron következő byte-tól kezdve.

- END nn - a programot lezáró utasítás. Az END utasítással vagy az EOT (04H) karakterrel mindig le kell zárni a forrás modult. Az nn operandus választható. Jelentése a SZERKESZTŐ-BETÖLTŐ alkalmazása esetén, a betöltés után a G utasítás hatására a vezérlés az nn címre átadódik azonnal. Hiánya esetén a töltés befejezésével a PROFIMON ill. a SZERKESZTŐ-BETÖLTŐ program kapja meg a vezérlést.

- GLOBAL CIMKE - globális cimkék definiáló utasítása.

- NAME CIMKE - modul azonosítójának definiálására. A cimke megjelenik a tárgymodul elején és az assembler lista fejlécében.

- PSECT kkk - kkk=REL - tetszőleges kezdőcímre tölthető, áthelyezhető modul (Relocate).
  - kkk=ABS - csak az ORG pszeudó utasítással megadott kezdőcímre tölthető (Absolut).

3.2.6. Assembler listázás

A TVC-ASM a listázást a :SO csatornán végzi.

Az áthelyezhető modulok kezdőcím függő címeit, amelyeket a SZERKESZTŐ-BETÖLTŐ program módosítani fog " jellel jelöli.
A hibaüzeneteket a :CO csatornán nyomtatja.

Utasításai: pszeudó utasítások

- EJECT - laponkénti szervezésben listáz.
- TITLE kk - a kk karaktersorozat max. 32 karakter hosszúságú, a listázáskor minden lap előtt mint fejléc kerül nyomtatásra.
- LIST - listázás kérés.
- NLIST - listázás kérés törlés (No list).

3.3. Áthelyezhető programmodul szabályok

A program áthelyezhető formátumú fordítása készül, ha PSECT ABS pszeudó utasítás nem használt, vagy a PSECT REL pszeudó utasítással definiált.

Az assemblerben minden globális cimke címet jelöl. Ezen 16 bites kezdőcímfüggő címeket a SZERKESZTŐ-BETÖLTŐ program az assembler által készített tárgymodul alapján módosítja.

Az egyéb cimkék közül csak azon cimkéket módosítja a SZERKESZTŐ-BETÖLTŐ program, amelyek 16 bites címet jelölnek.

Cimkék, amelyek konstans értékű cimkével definiáltak konstansok, kezdőcímfüggő cimkével definiált cimkék kezdőcímfüggőek.

3.4. Globális cimke használatának szabályai

Mindkét pass-ban futtatni kell az assemblert, ha a program globális cimkét tartalmaz.

Szintaktikailag ugyanazok a szabályok vonatkoznak a globális cimkékre, mint a lokális cimkékre. Lásd 3.2.1.

Az assemblerben minden globális cimke kezdőcímfüggőnek számító 16 bites címet jelent, amelyet a SZERKESZTŐ-BETÖLTŐ program a betöltés kezdőcímének megfelelően módosít.

Ugyanaz a belső cimke nem definiálható kétszer a felhasznált modulokban.

Külső cimke nem használható műveleti tagként kifejezésekben, valamint EQU és DEFL operandusaként.

3.5. A TVC Z80-ASM használata

3.5.1. A forrásprogramot

el kell készíteni mágnesszalagon a PROFIMON SZÖVEGSZERKESZTŐVEL (%E) vagy a TVC SZÖVEGSZERKESZTŐVEL. Ez utóbbi esetén a sima filenév (nem kettősponttal) használandó az ESC-W utasításnál, és a szöveg elején egy CTRL-B karaktert, végén egy CTRL-C karaktert tartalmazó sort kell előállitani.

3.5.2. A PROFIMON operációs rendszerben

a TVC-ASM által használt csatornákhoz a periféria meghajtó programokat hozzá kell rendelni. Lásd: 1.6.1.

3.5.3. Beugrás az assemblerbe

%A<cr> utasítással.

3.5.4. Az assembler bejelentkezik

és OPTIONS? kérdéssel utasításra vár.

Lehetséges utasítások:

K - listázást nem kér a felhasználó (KILL LIST).
L - listázást kér a felhasználó a :SO csatornán (LIST).
N - tárgy modul kimenetet nem kér a felhasználó (NO OBJECT).
O - tárgy modul kimenetet kér a felhasználó az :OO csatornán (OBJECT).
P - csak 2. pass-t kér a felhasználó.
    Csak 2. pass akkor kérhető, ha
    - nincs globális cimke a programban,
    - nincs NAME pszeudó utasítás a programban,
    - nincs hivatkozás olyan cimkére, amely csak a hivatkozás után jelenik meg a cimkemezőben.
Q - visszatérés a PROFIMON operációs rendszerbe.
R - cimketábla törlése (pass 1 előtt automatikusan végbemegy).
S - cimketábla listázást kér a felhasználó az assembler listázás végén.
. - törli az utasításokat, új utasítások adhatók.

3.5.5. Kocsi vissza karakterrel

lezárva az OPTIONS utasításokat, ha még nem definiáltuk a cimketábla határait az 5.1. fejezetben leírt módon, a MEMORY DIVISION utasítással, az assembler bejelentkezik és kéri a cimketábla határait (SYMBOL TABLE LIMITS?). Válaszul két dolgot tehetünk:

a.) Ha a kocsi vissza karaktert leütjük, a cimketábla határainak kijelölése automatikusan 1E00H-2900H, amely 312 cimkét tartalmazhat.

b.) Két operandussal a felhasználó megválaszthatja a cimketábla határait, s utána kocsi vissza karakter (op1>1E00H op2>op1). Egy cimke 9 byte-ot igényel. Op1-nek és op2-nek nem kell kilenccel oszthatónak lenni.

3.5.6. TVC-ASM végrehajtja

a pass 1-et (a zárókarakterek leütése előtt ne felejtsük el a magnetofont elindítani, ha a forrás anyagot onnan küldjük); és bejelentkezik
READY PASS 2?

3.5.7. A kazetta visszatekercselése

után bármely karakter leütésére megkezdi a pass 2 végrehajtását. Ügyeljünk arra, hogy a kazettát a file elejére állítsuk.

3.5.8. Pass 2 végén

kiírja decimálisan a talált hibák számát ERRORS = nn formában, és visszatér a PROFIMON operációs rendszerbe.

3.6. Hibaüzenetek:

A hibás sor elején az assembler listán a :SO kimenő eszközön a következő jelölések lehetnek:

B - illegális művelet a kifejezésben.
D - az operandus mezőben egy szám illegális karaktert tartalmaz.
E - külső cimke hibás használata (műveleti tagként, EQU vagy DEFL pszeudó utasítások operandusaként).
I - illegális operandus.
L - illegális karakter a cimkében.
M - multidefiniált cimke.

N - EQU vagy DEFL pszeudó utasítást cimke nélkül használtunk.
O - illegális műveleti kód.
P - a PSECT pszeudó utasítás több mint egyszer szerepel.
Q - nem bezárt idézőjel használata.
R - operandus értékhatár túllépés (lásd 3.2.3.).
S - szintaktikai hiba.
T - egy sor a forrásprogramban max. 127 karakterből állhat. Ennek túllépését jelzi.
U - nem definiált cimke.
V - művelet értékének (aritmetikai operandus) számítása közben a Z80 CPU-ban túlcsordulás keletkezett.
X - cimketábla kijelölési hiba 1E00H < alsó-határ és alsó-határ<felsőhatár nem teljesül. Ez ugyanúgy, mint az F hibajelzás, megszakítja a fordítást.
F - cimketábla túlcsordulás. Ez a hiba megállítja a fordítást a hibaüzenetet kiírja a :CO csatornára és visszatér a PROFIMON operációs rendszerbe.

3.7. Több forrásprogram együttes fordítása

Az egyes forrásmoduloknak STX (02H) karakterrel kell kezdődnie és ETX (03H) karakterrel kell lezáródnia. A sorban utolsó modult END pszeudó utasítással, vagy EOT (04H) karakterrel kell lezárni.

Mindkét pass végrehajtásakor az egyes modulokat sorban kell beolvastatni.

Az egyes file nevek azonosan egyetlen betűből állnak, amelynek az ASCII kódja az 1621H memória címen található.

4. A PROFIMON Relocating Linking Loader, a TVC SZERKESZTŐ-BETÖLTŐ programja

A PROFIMON által készített tárgymodulok betöltését, összefűzését végzi. (A nem áthelyezhető, nem összefűzhető tárgymodulokat a TVC R utasításával betölthetjük, lásd 1.6.4.). A SZERKESZTŐ-BETÖLTŐ az egyenként megírt és ASSEMBLER-rel lefordított, egyenként tesztelt modulokat egy programmá szerkeszti. A program méretét a globális cimketábla és a RAM mező mérete - ahová a szerkesztés történik - korlátozza.

4.1. Beugrás a SZERKESZTŐ-BETÖLTŐ programba

%R aaaa,bbbb<cr>

- aaaa, bbbb-re ugyanazok a szintaktikai szabályok vonatkoznak, mint az egyéb operandusokra (lásd 1.4.).

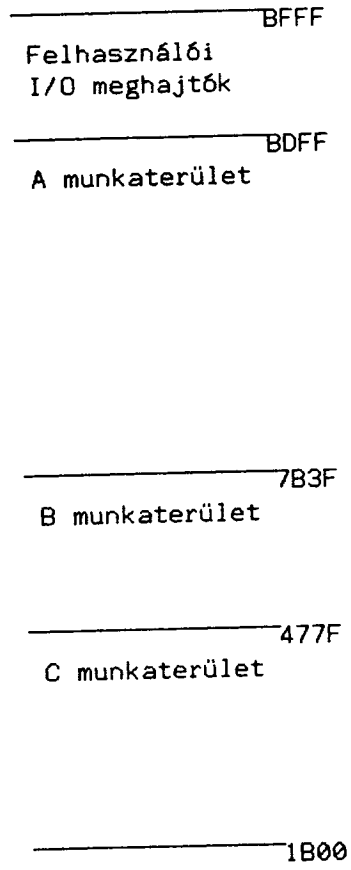
- aaaa - az a cím ahova a betöltést kezdjük,

- bbbb - globális cimketábla kezdete,

- ha mindkét operandus hiányzik a betöltést a PROFIMON R utasításának megfelelően végzi el, és ezután visszatér a PROFIMON operációs rendszerbe.

- bbbb - elhagyható, ilyenkor a globális cimketábla kezdőcímét a program határozza meg a következők szerint:

  a). Ha előzőleg MEMORY DIVISION (%M) utasítással automata memóriafelosztást (lásd 5.1, 5.2) definiáltunk, a cimketábla kezdete a C terület vége lesz. Ettől a címtől felfelé tölti a globális cimkéket. Egy cimke 11 byte-ot foglal le (7.ábra).



7.ábra Automatikus RAM felosztás

b.) Ha nem definiáltunk memóriafelosztást, a globális cimketábla kezdete az A terület végétől lesz. Ettől a címtől kezdve felfelé tölti a globális cimkéket.

Ha az aaaa operandust megadjuk, a betöltés kezdőcime

a.) áthelyezhető (REL) modulok betöltése esetén, aaaa + a modul kezdőcíme (ORG pszeudó utasítással definiált) lesz,

b.) nem áthelyezhető (ABS) modulok betöltése esetén a betöltés a modul eredeti (ORG pszeudó utasítással definiált) kezdőcímére történik.

Ezután a * karakter kiküldésével bejelentkezik a SZERKESZTŐ-BETÖLTŐ program.

4.2. SZERKESZTŐ-BETÖLTŐ program utasításai

***L aaaa<cr>** (Load, az :OI csatornán olvas be).

- nem áthelyezhető (ABS) modulok betöltése esetén a betöltés mindig a modul eredeti (ORG pszeudó utasítással definiált) kezdőcímére történik.

- áthelyezhető (REL) modulok betöltése esetén a betöltés az aaaa + a modul eredeti (ORG) kezdőcímére történik. Ha az aaaa operandust elhagyjuk, a modult az előző modul után az eredeti (ORG) címmel eltolva tölti. Az utasítás végrehajtása után mindig kiírja a modul kezdő és végcímét és a nem definiált cimkék számát decimálisan, majd a * karakterrel jelzi, hogy kész a következő utasítás végrehajtására.

***T<cr>** (Type)

Kinyomtatja a globális-cimke táblát. A nem definiált cimkéket **** karaktersorozattal jelöli.

*E<cr> (Execute)

A betöltés után az utasítás hatására az első modul END pszeudó utasításának nnnn operandusával meghatározott címen kezdődik a felhasználói program végrehajtása. Lásd 3.2.5.

Az egyes modulok betöltése után ha az nnnn operandus definiált, EXECUTE nnnn üzenetként kinyomtatja nnnn értékét. Ha kilépünk a SZERKESZTŐ-BETÖLTŐ programból, az nnnn címet nem tárolja a visszatérésig.

*.<cr>

Visszatérés a PROFIMON operációs rendszerbe.

4.3. A globális-cimketábla határainak módosítása

A PROFIMON operációs rendszerbe visszatérve S utasítással megjeleníthetjük és/vagy felülírhatjuk a "határbyte"-ok értékét. A kezdőcímet a kisegítő-tár FF04H és FF05H című rekesze, a végcímet FF0AH és FF0BH című rekesz tartalmazza. (Mivel a globális-cimke táblát felfelé tölti a program, a kezdőcím mindig nagyobb, mint a végcím.)

Visszatérni a SZERKESZTŐ-BETÖLTŐ programba %N<cr> (Next Read) utasítással lehet (Ha R utasítással térnénk vissza, az törölné a cimketáblát).

4.4. Hibajelzések (a :CO csatornán)

4.4.1. A hibák első csoportja

nem szakítja meg a betöltést, csak a betöltés végén hibaüzenetet nyomtat.

a.) Ellenörző összeg (checksum) hiba:
ERROR 1 üzenetet nyomtat ki, ha a hiba a betöltendő programot megelőző, az áthelyezhetőséghez és összefűzéshez szükséges kiegészítő információkat tartalmazó részben van.

aaaa ERROR 1 üzenetet nyomtat, ha a hiba a betöltendő programban van. aaaa értéke a hibás sort követő sor kezdőcíme lesz.

b.) CIMKE ERROR 2 üzenetet nyomtat ki, ha a programmodulban multidefiniált globális cimkét talál. A második definíciót a SZERKESZTŐ-BETÖLTŐ program nem veszi figyelembe.

4.4.2. A hibák második csoportja

megszakítja a betöltést és visszatér a PROFIMON operációs rendszerbe (ABORT ERRORS). A hibák a következők lehetnek:

a.) a modult a címketábla területére akarjuk tölteni,

b.) a modult nem a felhasználható memóriaterületre akarjuk tölteni (ERROR 4),

c.) globális-cimketábla túlcsordulás.

5. A PROFIMON memória-bázisú operációs rendszere

Az ASSEMBLER, a SZÖVEGSZERKESZTŐ és a SZERKESZTŐ-BETÖLTŐ programok használatakor, az egyes modulokat valamilyen külső hordozóról (pl: mágnesszalag) beolvassuk, s a végeredményt valamilyen külső hordozóra kivisszük. A következő lépésben ezt ismét beolvassuk, stb.

A program készítését jelentősen leegyszerűsíti, ha pl. a SZÖVEGSZERKESZTŐ-vel szerkesztett forrásmodult közvetlenül a memóriába fordíthatjuk, és innen egy harmadik területen a SZERKESZTŐ-BETÖLTŐ programmal közvetlenül összefűzhetjük a tárgymodulokat és a külső hordozóra csak ez a végleges összefűzött, letesztelt program kerül. Így a memória-bázisú operációs rendszerrel egy magas hatásfokú interaktív programfejlesztés valósítható meg. Ez akkor hatásos, ha a forrásprogramunk befér az A munkaterületre!

5.1. A memória automatikus felosztása

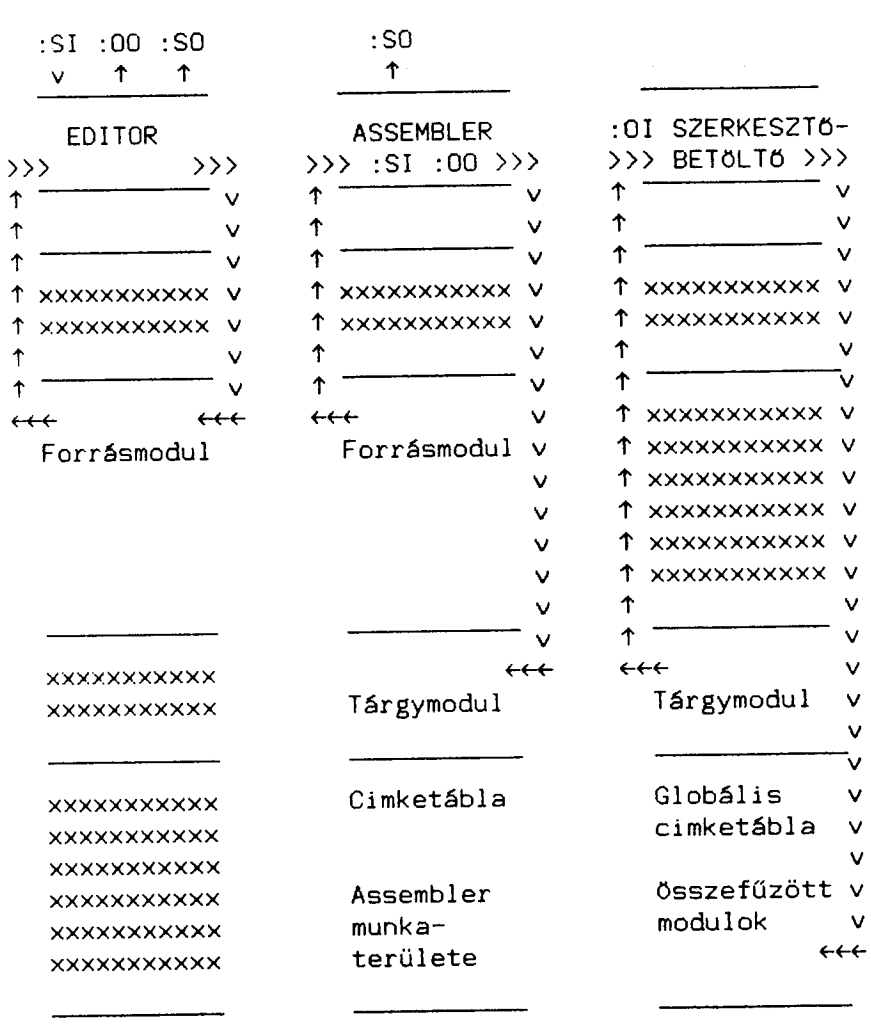
Egy szubrutin, a RAM területet négy funkcionális egységre osztja (7.-8.ábra).

5.1.1. "A" terület

Forrásmodul munkaterülete, az ASSEMBLER és a SZÖVEGSZERKESZTŐ program részére.

5.1.2. "B" terület

Az ASSEMBLER tárgymodul kimenetének tárolására.



8.ábra. A PROFIMON memória-bázisú operációs rendszer működési vázlata az EDITOR, ASSEMBLER és BETÖLTŐ programok esetén

5.1.3. "C" terület

1B00H-1E00H tartományt az ASSEMBLER használja átmeneti tárolásra.
1E00H-C vége tartományt az ASSEMBLER használja cimke-táblaként.
Ha a SZERKESZTŐ-BETÖLTŐ programot használjuk, C végétől lefelé a globális cimketábla helyezkedik el. Az 1B00H cimtől felfelé az összefűzött modulok tára található.

5.1.4. I/O meghajtó programok tára

Az A végétől a PROFIMON kezdetéig terjedő tartományt használhatjuk (512 byte) a felhasználó által írt periféria meghajtó programok tárolására.

5.2. A memóriafelosztás aktivizálása

A PROFIMON operációs rendszer %M<cr> utasításával végezhetjük el az automatikus felosztást. Erre a program:

- Behívja a MEMTOP szubrutint, mely kimenetként a kisegítő tár FF24-FF25H rekeszébe írja a címmező utolsó byte-jának címét (BFFFH).

- A kisegítő tár FF06-FF07H című rekeszében flag-et állít be, mellyel jelzi a rendszernek, hogy az automatikus memória felosztás definiált.

Kiszámítja a RAM terület A, B, C, munkaterületeinek határait.

C vége = 477FH
B vége = 7B3FH
A vége = BFFFH - 512

Ezen határok címét a kisegítő tárba írja.

A vége FF00-FF01H címre
B vége FF02-FF03H címre
C vége FF04-FF05H címre

- MEMORY DIVIDED! üzenet kiküldésével jelzi, hogy a határkijelölést végrehajtotta.

5.3. Munkaterület határainak módosítása

a kisegítő tárban található, a munkaterületek határcímeit tartalmazó byte-ok értékének módosításával lehetséges (lásd: 5.2).

5.4. Az automatikus határkijelölést kitörölhetjük,

ha a MEMMAP flag byte-jait nullázzuk.

5.5. PROFIMON "memória meghajtó" programok:

- IN A (:IA)
  :SI csatornához rendeljük, az A forrásterületről olvas.

- IN B (:IB)
  :OI csatornához rendeljük, a B tárgyterületről olvas.

- OUT B (:OB)
  :OO csatornához rendeljük, a B tárgyterületre ír.

1. Függelék

a.) A PROFIMON hívható belső szubrutinjai egy táblázaton keresztül érhetők el. A táblázat egyes elemei egy-egy ugró (JP) utasítást tartalmaznak az illető szubrutin kezdőcímére. A felhasználónak egy szubrutinhívással (CALL) kell a táblázat megfelelő elemét hívni.

Név (1.7 fej.)	**Cím**
RDCHR	0EF00H
WRCHR	0EF03H
PACC	0EF06H
PRVAL	0EF09H
INECHO	0EF0CH
CRLF	0EF0FH
SPACE	0EF12H
PTXT	0EF15H
ASBIN	0EF18H

b.) A SZÖVEGSZERKESZTŐ (EDITOR) használatánál sok beszúrás esetén jó szolgálatot tesz az átsorszámozás. Ez kétféleképpen hajtható végre.

1. Memória-bázisú operációs rendszer használata esetén a P utasítással az :OB rutinon keresztül a "B" területre írjuk ki a szöveget, majd onnan %E utasítás és :IB rutin használatával újra beolvassuk R utasítással az EDITOR munkaterületére.

2. P utasítással magnetofonra vagy diszkre kiírjuk, majd %E után R-el visszaolvassuk a szöveget.

2. Függelék

A PROFIMON parancsok összefoglalása.

%A	Belépés a Z80 Assembler fordítóprogramba.
%B nnnn,b	Megszakítási (Break) pont beépítése a gépi kódú program nnnn címére. Ha b=0 (vagy nincs), töréspont elérésekor PC és AF kiírás, ha b=1 töréspont elérésekor teljes regiszter térkép kiírás.
%B	Törli a meglévő megszakítási pontot. Ezt egy újabb %B nnnn parancs is megteszi, így mindig csak az utolsónak beadott parancs érvényesül. (Csak 1 megszakítási pont lehet!)
%C aaaa,bbbb,cccc	Az aaaa-bbbb memóriatartományt átmásolja (Copy) a cccc-vel kezdődő tartományra.
%D aaaa,bbbb	Az aaa0-bbbb memóriatartományról hexadecimális és ASCII listát ad (Dump). A $ tartalma az utoljára kiírt cím. Ha bbbb elmarad, akkor aaa0-tól kb. 70 memóriahely tartalma jelenik meg.
%E	Belépés a SZÖVEGSZERKESZTŐBE (EDITOR).
%F aaaa,bbbb,kk	Az aaaa-bbbb memóriatartományt tölti a kk 8 bites karakterrel (Fill).

%G nnnn

A gépi program indítása (Go) az nnnn memóriacímről. Ha nnnn elmarad a tükör regiszter PC értéke lesz az indító cím.

%H +-aaaa+-bbbb..=

Hexadecimális számok összegének és különbségének kiszámítása.

%I f,nnnn

Blokkolatlan file beolvasása (Input) mágnesszalagról az nnnn memóriacímtől kezdődő területre. f=file név (A, B, C, D, E, F). $=0, ha a beolvasás hibátlan, különben a hibakód.

%K aaaa,bbbb,cccc

Az aaaa-bbbb memóriatartomány összehasonlítása (komparálása) a cccc-n kezdődő tartománnyal.

%L aaaa,bbbb

Az aaaa-bbbb memóriatartomány listázása assembler kódra való visszafordított állapotban.

%M

Automata memóriafelosztás aktiválása.

%N

A SZERKESZTŐ-BETÖLTŐ programba ismételt belépés (Next Read).

%O f,aaaa,bbbb

Blokkolatlan file kiírása mágnesszalagra az aaaa-bbbb memóriatartomány adataival (Output). f=a file név (A, B, C, D, E, F). $=0 ha hibátlan volt a kivitel, különben a hibakód.

%P nn

Az nn nevű (0-FF) bemenő periféria (Port) olvasása, ill. az nn nevű kimenő periféria írása az olvasás után.

%Q aaaa,bbbb

Ellenőrző összeg készítés az aaaa-bbbb memóriatartomány adatainak egyszerű aritmetikai összegzésével.

%R aaaa,bbbb

A Z80 ASSEMBLER ill. a %W utasítás által előállított file memóriába hozása (Read) az :OI eszközről. Az aaaa az eltolási cím, bbbb=1 esetén ABSZOLUT (INTEL-HEX ill. W utasítás) file eltolt töltése, különben belépés a SZERKESZTŐ-BETÖLTŐ programba. Ott bbbb=a címketábla kezdete.

%S nnnn

Az nnnn címtől kezdve a memóriatartalom vizsgálata (Search) és felülírása. $=az utolsó címet követő cím.

%T aaaa,bbbb,cc

Gépi kódú program lépésenkénti futtatása az aaaa memóriacímtől (Trace) bbbb=0 egy utasításonként, különben bbbb lépésközzel. Az egyes lépések utáni kijelzés cc függvényében:
cc=0, PC és AF
cc=1, teljes regiszter térkép,
cc=2, az utasítás Z80 mnemonikja, PC és AF
cc=3, az utasítás Z80 mnemonikja és a teljes regiszter térkép.

%U

Újra belépés a SZÖVEGSZERKESZTŐBE a meglévő szöveg megtartásával.

%V aaaa,bbbb

Blokkolatlan file (magnetofonra írás) utáni ellenőrzés (Verify). Az aaaa és bbbb a memóriatartományt jelenti, ahonnan az %O utasítással a file-t előállítottuk.

%W aaaa,bbbb

INTEL-HEX formátumú kiírás (Write) az :00 eszközön az aaaa-bbbb memóriatartomány adataival.

%X cc

A CPU regisztereinek kiíratása (eXamine registers) a tükörtárból. cc=0 (vagy elmarad) fejléc nélkül, cc=1 fejléccel.

%Y aaaa,bbbb,cccc

Az aaaa-bbbb memóriatartomány tartalmát mintának véve, a cccc címtől kezdve megvizsgálja a memória tartalmát és azonos mintázatot keres. Ha talál, akkor kiírja a minta kezdőcímét. RETURN billentyű lenyomására tovább vizsgál a memória végéig.

%Z cc

Ha valamilyen hiba esetén nem történt meg a kimenő file lezárása, cc=0 a :00 és cc=1 esetén a :S0 periférián kényszerített file lezárás lesz.