MTA Számitástechnikai és Automatizálási Kutató Intézete, **Közlemények** 19/1978.

EGY MÓDSZER ASSEMBLEREK ELŐÁLLITÁSÁRA

Csörnyei Zoltán

Az assemblerek hasonlósága szükségessé és lehetővé tette egy olyan rendszer kidolgozását, amellyel ezek a fordítóprogramok egyszerűen és gyorsan előállithatók. A létrehozott fordítóprogramok két részből állnak: egy törzsből, amelyik közös mindegyik fordítóprogramban, és egy leiró táblából, amely az adott fordítóprogramtól függ. A törzs határozza meg a fordítóprogram jellegét, például, hogy hány passzból áll, milyen a szimbólumkezelése, input-outputja, mig a leiró tábla definiálja a fordítóprogram direktiváit és utasitásait.

A fordítóprogramok gyors előállitásának alapgondolata az, hogy a fordítóprogramok a fordítás folyamán sok hasonló, vagy azonos funkciót végeznek, igy ezek a különböző fordítóprogramokban azonosak is lehetnek. A következőkben ezeket a közös funkciókat határozzuk meg.

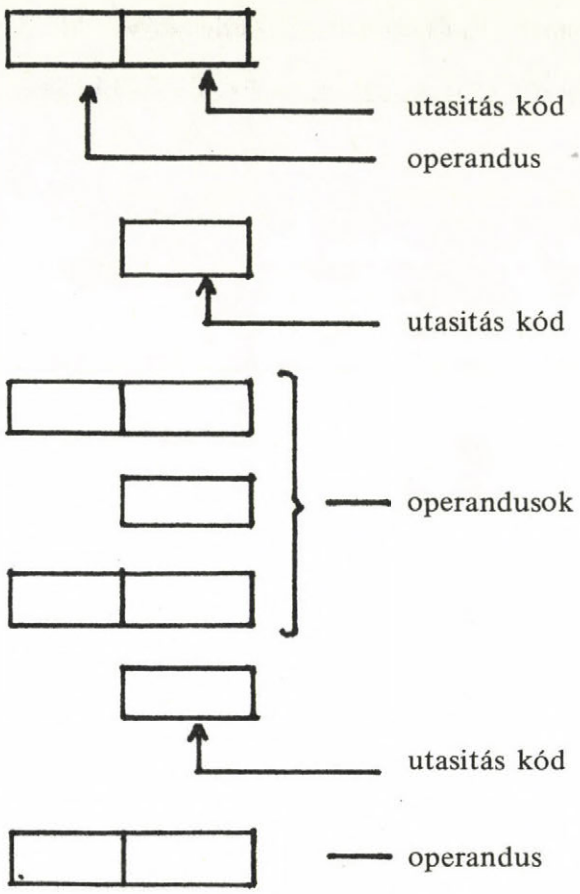
A fordítóprogramnak az a feladata, hogy a szimbólikus nyelven irt programból (forrásprogramból) létrehozzon egy azonnal, vagy csak szerkesztés után futtatható programot (tárgyprogramot). A tárgyprogram lehet gépi kódú, vagy valamilyen interpreter által végrehajtandó program. A forrásprogramok közös tulajdonságait határozzuk meg először.

A forrásprogramok közös tulajdonságai

A forrásprogram sorokból áll, minden egyes sor egy utasitás szimbólikus nyelvű leirása. Előfordulhat, hogy egy sor az adathordozón nem fér el fizikailag egy sorban, például, ha egy utasitásnak nagyon sok operandusa van; ilyenkor biztositani kell a sor tördelhetőségét, egy speciális jel (pl. <) szolgálhat a törés jelzésére.

A sor karakterekből álló karakterlánc, a sor végét egy, vagy két speciális karakter (kocsivissza, soremelés, új-sor, stb.) jelzi. Az egy ilyen sorból leforditott bináris információ két részből áll: az utasitás kódjából és az operandusokból. Az utasitáskód terjedelme két-három bittől két byte-ig terjed, az operandus az utasitástól függően lehet csak egy-két bit, vagy akár három-négy, vagy még több byte is.

A forrásprogram sorai zónákra tagolhatók, egy sor négy zónából áll: cimke, utasitás, operandus és komment zóna.



A cimkezóna lehet üres, vagy egy nevet (szimbólumot, cimkét) tartalmaz, amely az utasitássort azonositja. Általában ez a név az utasitás kód memóriarekeszének szimbolikus neve. A cimkezónába irható szimbólumok betüvel kezdődő, betüket és számokat tartalmazható karaktersorozatok, a karaktersorozatok hosszát általában maximálják. A szimbólumokban speciális jelek is előfordulhatnak (pl. : vagy @), ezeket célszerű a programozónak a szimbólum tipusjelzésére használni, feltéve, hogy a forditóprogram ezt nem követeli meg.

Az utasitászónában helyezkedik el az utasitás szimbólikus kódja (mnemonik). A tárgyprogram szempontjából a forrásprogram utasitásait három csoportra oszthatjuk. Az első csoportba azok az utasitások tartoznak, amelyek a sor lefordításakor a tárgyprogramba információt adnak. Ezek a gépi kódú utasitások és az adat (konstans) megadások. A második csoport utasitásai közvetlenül nem adnak információt, ilyen pl. egy szimbólum helyének, vagy értékének definiálása. A harmadik csoportba a forditóprogramnak szóló utasitások, a direktivák tartoznak.

Az utasitászónában lévő információtól függ, hogy a többi zónának kell, lehet, vagy nem lehet információt tartalmaznia.

Az operandus mezőbe kerülnek az operandusok. Több tagból is állhatnak az operandus tagjait legalább egy karakterrel (pl. vessző, vagy szóköz) kell elválasztani. Az itt lévő információból készülnek az utasitáskód utáni bináris értékek. A forrásprogram operandus tagjai lehetnek szimbólumok, számok, kifejezések. A kifejezés müveleti jelekkel összekapcsolt szimbólumok és számok sorozata, hossza álatlában nincs korlátozva. Az operandus tagjainak számát az utasitászónában lévő utasitás határozza meg.

A komment zóna kezdőkaraktere általában egy speciális karakter (pl. * vagy ;). A komment zóna a sor elején is kezdődhet, az ilyen sort komment sornak nevezzük. A komment a tárgyprogramba információt nem ad.

A zónák kötött, vagy szabad formában követhetik egymást. A kötött formában a zónák mindig egy előirt pozición kezdődnek, mig a szabad formában a zónák között egy terminátor karakter (általában szóköz, vagy tabulátor) helyezkedik el.

A forditóprogramok közös tulajdonságai

A forditóprogramok azonos tulajdonságait két szempontból vizsgáljuk. Az első a forditóprogram szolgáltatása, amelyet egy dialógussal lehet definiálni. A forditóprogramok háromféle eredményet adnak:

- bináris információ (tárgyprogram)
- lista
- cross-referencia lista.

Az utóbbi kettővel általában együtt jár a tárgyprogram készitése. A felhasználó a dialógus folyamán kijelöli, hogy milyen szolgáltatást kér; ez a forditóprogramban flagek állitását jelenti. A forditóprogramok általában mindig előállitják mindhárom output-ot, információhordozón való megjelenésüket tiltják, vagy engedélyezik a flagek. A dialógus programjára csak a forditás elején van szükség, ez a terület a forditás alatt még felhasználható.

A másik szempont a forditóprogramok müködése. A forditóprogramok kétpasszosak, bár néha lehet látni egypasszos forditókat is. Minden forditóprogram tartalmaz egy szervező részt, amely többnyire más funkcióval egybeépitve (pl. szimbólumtábla kezelés) a passzok szervezését végzi. A passz sorszám is egy flagben tárolódik, igy a feldolgozó programok mindkét passzban azonosak. A passztól függő funkciót a flag vizsgálatával választják ki.

A forditóprogramok a forrásprogramot soronként dolgozzák fel. Az első teendő a forrásprogram (következő) sorának beolvasása a háttértárolóról. Már a beolvasáskor átalakithatja a sort, és előkészitheti a listázásra. Kiszűri a felesleges karaktereket (szóközök, javitások), tabulál, stb. Az input feladata a sor hosszának megállapitása is.

Ha a beolvasott sor komment sor, azonnal indulhat a listázása, további munka az ilyen sorral nincs.

A nem-komment soroknál az első feladat az utasitás zónában lévő karaktersorozat azonositása. A lehetséges utasitásokat és a hozzájukrendelt kódokat általában egy táblázat tartalmazza. Ez a táblázat tartalmaz arra is utalást, hogy a többi zónában milyen információt kell keresni, azaz hogyan kell azokat feldolgozni.

Az utasitás azonositása után a cimke feldolgozása következik. Két tipusú cimke használata az általános: az egyik az utasitáskód memória rekeszének neve, a másik az értékadás. A két tipusú cimkefeldolgozó programnak használnia kell a szimbólumtáblákat is, hiszen a cimkét oda fel kell irni, vagy a multidefiniáltságot meg kell határozni. A szimbólumtáblát kezelő programnak a következő belépései a szokásosak:

– egy adott szimbólum felirása,

– egy adott szimbólum keresése,

– egy adott szimbólum törlése.

Az első mindig tartalmazza a második funkciót is. Az első a cimke, a második az operandus feldolgozásakor szerepel. Mivel kétpasszos forditóprogramnál két szimbólumtábla (predefinit és postdefinit cimkéknek) használatos, a felirás és keresés is többféle lehet. A harmadik a program, vagy a program egy részének végekor müködik.

Az operandus feldolgozása általában a legbonyolultabb. Ha az utasitástól függően nincs más korlátozás, az operandus tagjai kifejezések. Egy kifejezés feldolgozásakor a szimbólumtábla-kezelő, számátalakitó és természetesen a kifejezésben lévő müveleteket végző programokat kell használni. Mennél több kényelmet biztosit a forditóprogram a felhasználónak, annál bonyolultabb a kifejezésfeldolgozás a forditóprogramban.

Az operandus feldolgozásával a beolvasott sor feldolgozása is befejeződött. Az első passz csak a szimbolumtáblát és különböző pointereket kezel, a második passzban pedig a sor feldolgozása után a kijelölt output-okat kell végrehajtani. A tárgyprogrammal különösebb teendő nincs, ez egy háttértárolón tárolható. A lista többféle lehet. Általában négy részből áll a lista egy sora:

– az utasitássor sorszáma a forrásprogramban,

– memória cim,

– a memória cim tartalma,

– a forrásnyelvi utasitássor.

Az utasitássorban lévő utasitástól függ, hogy a négy rész közül a listában melyik szerepel.

A fordítóprogramok hibajelzései kétfélék: az egyik tipus a sor feldolgozása folyamán detektált hibákat jelzi, a másik a fordítás befejezésekor ismeretlen szimbólumokat irja. Mivel a hibajelző programot a fordítóprogram majdnem minden részéből hivhatja, a hibajelzést egy vagy több szubrutin végzi.

A fordítás befejezése után több szolgáltatás biztositja a felhasználó kényelmét, a fordítóprogramok kiirhatják a tárgyprogram hosszát, központi memóriában való elhelyezhetőségét, a szegmensek listáját, stb.

A fordítóprogram törzse

A fenti részben nem beszéltünk arról, hogy a fordítóprogram a forrásprogramot milyen tárgyprogrammá forditja, azaz az adott fordítóprogram milyen utasításkészletet ismer fel, milyen operandusai lehetnek, mi az operandusok sorrendje, milyen utasításoknak nem lehet cimkéje. Ez csak az adott forrásnyelv utasításainak definiciójától függ.

Még arról sem volt szó, hogy milyen a szimbólumábrázolás, milyen a forrásprogram struktúrája. Ez is a forrásnyelv definiciójától függ, de előfordulhat, hogy ez már több forrásnyelv definiciójában azonos.

A fordítóprogramok fenti jellemzéséből, valamint ezekből az észrevételekből kitünik, hogy a fordítóprogramok alapvetően két részre oszthatók. Az egyik rész forrásnyelv-specifikus a másik fordítóprogram-specifikus. A fordítóprogram-specifikus rész több fordítóprogramban lehet azonos. Ezt a részt nevezzük a fordítóprogram törzsének. A másik rész a forrásnyelv utasításait irja le, ez a leiró tábla. Ha a törzs tartalmaz egy interpretert, a leiró rész interpretativ leirású lehet, amivel lehetővé válik a forrásprogram utasításainak kis memória-igényü leirása.

Ha a törzs jól szegmentált, feladat-orientált felépitésü, akkor az egymástól lényegesen különböző forrásprogramok fordítóprogramjai is könnyen elkészithetők, feltéve, hogy ezek a fordítóprogramok is rendelkeznek a korábban leirt közös tulajdonságokkal. Egy törzsből a megfelelő módositások elvégzésével, de a törzs szerkezetének megtartásával egy egészen új tipusú fordítóprogram hozható létre.

A fordítóprogram törzse tartalmazza azokat a funkciókat megvalósító programokat, amelyeket a közös tulajdonságukról szóló részben irtunk le. Ezeket a programokat célszerü úgy irni, hogy az interpretativ leirás moduljai legyenek, azaz a leiró táblában egy kóddal lehessen rájuk hivatkozni. Az interpreter a leiró tábla kódjait értelmezi és inditja a kódhoz tartozó programot.

A leiró tábla

A fordítóprogramnak ez a része tartalmazza a fordítás szabályait, ez a rész az utasítások leirásából áll.

Itt vannak a mnemonikok és a hozzájuk tartozó szintaktikus szabályok. A leirás interpretativ, egy egy-byte-os interpretert választva a leirás rövid, kisebb mint a törzs egynegyede.

A forrásprogramban a felhasznált különböző utasitások darabszámának eloszlása nem egyenletes, igy lehet egy utasitás szimbólikus nevének keresése lineáris, ha a tábla elejére a leggyakrabban használt, végére a legritkábban használt utasitások kerülnek. Ha a keresést a tábla elején kezdjük, az utasitás felismeréséhez szükséges összehasonlitások száma kevesebb, mint rendezetlen tábla esetén. Nem-lineáris keresési algoritmussal lehet az összehasonlitások számát csökkenteni, de legtöbbször ez a program nagyságát növeli. Igy ha az utasitások száma nem nagy, a lineáris keresés a fordítás idejét lényegesen nem növeli.

A leiró táblában az utasitások leirása nem egyforma hosszúságú, ezért a kereséshez a táblában a láncolási cimeket el kell helyezni, a láncolási cim lehet pl. a következő utasitás leirásától való távolság is.

A forrásprogram utasitásainak egy jellemzője, hogy sok utasitás és szintaktikájuk csak a mnemonikban és az utasitáskódban különbözik egymástól. Ezeket az utasitásokat természetesen célszerü a leiró táblában úgy leirni, hogy csak a mnemonik felismerésének és a bináris kód megadásának leirása különbözzön, és a leirás többi része már csak egy helyen szerepeljen. Ehhez egy olyan interpretativ utasitás kell, amelyik a leiró táblában ugrásokat hajt végre.

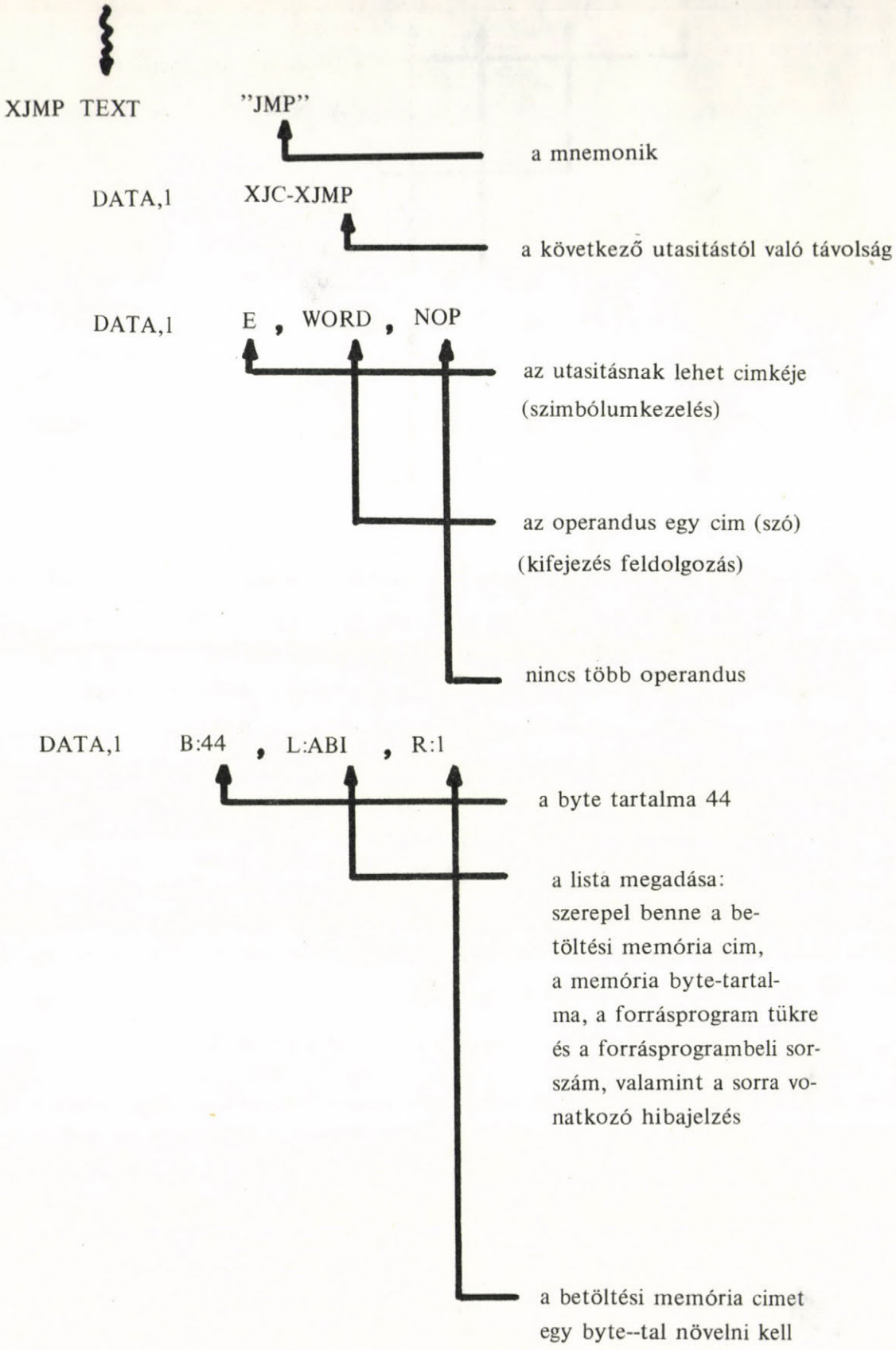
A forditóprogramok

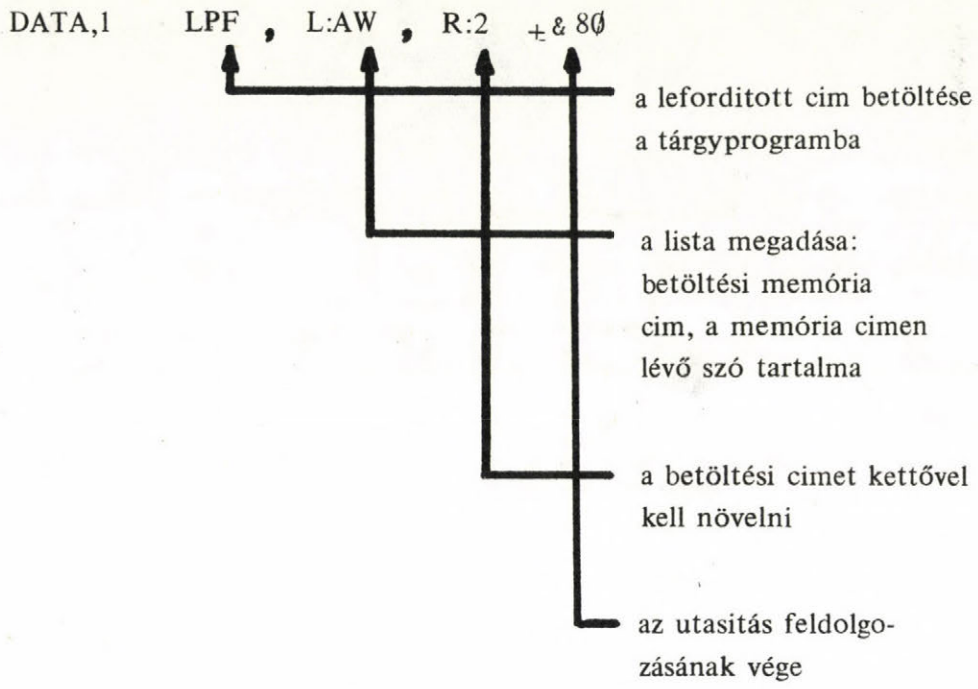
Az előállitott forditóprogramok memória mérete és a fordítás ideje lehet, hogy nem optimális. Mégis ami a módszer mellett szól, az a könnyen áttekinthető szerkezet és a gyors ellőállithatóság. A forditóprogram módositása, bővitése, szükitése sem okoz problémát, akár az utasitások leirását, akár a program törzsét kell megváltoztatni, elsősorban a tagolt, áttekinthető felépités miatt.

Ezzel a módszerrel már három forditóprogram készült el. Mindhárom R-10 számitógépen müködik. Az egyik egy R-10-en futó program fordítására, a másik kettő az R-10-en szimulált számitógépek tárgyprogramjainak előállitására szolgál.

Az első program a PROCESS-2/8K autókodja. A második az MTA SzTAKI-ban kifejlesztett MFB (Mikroprocesszoros Folyamatirányitó Berendezés) assemblere, a harmadik az MFB autokódja. Az utóbbi kettő program törzse is azonos, csak a leiró táblák különböznek. Az első és a másik kettő forditóprogram törzseinek felépitése hasonló, a törzs egyes moduljaiban van csak eltérés. Más pl. a szimbólumok formátuma, a lebegőpontos számábrázolás, igy az ezek feldolgozását végző modulok különbözőek.

Egy példával szemléltetjük az ilyen tipusú forditóprogramok leirását. A példában az MFB assembler leiró táblájának egy része szerepel, a JMP utasitás leirása. Az utasitás három byte-os, az első byte az utasitáskód, értéke 44, a második és harmadik byte egy memóriacim.


XJMP TEXT
"JMP"
a mnemonik
DATA,1
XJC-XJMP
a következő utasitástól való távolság
DATA,1
E , WORD , NOP
az utasitásnak lehet cimkéje
(szimbólumkezelés)
az operandus egy cim (szó)
(kifejezés feldolgozás)
nincs több operandus
DATA,1
B:44 , L:ABI , R:1
a byte tartalma 44
a lista megadása:
szerepel benne a betöltési memória cim,
a memória byte-tartalma, a forrásprogram tükre
és a forrásprogrambeli sorszám, valamint a sorra vonatkozó hibajelzés
a betöltési memória cimet
egy byte--tal növelni kell


DATA,1 LPF , L:AW , R:2 +& 8Ø
a leforditott cim betöltése a tárgyprogramba
a lista megadása: betöltési memória cim, a memória cimen lévő szó tartalma
a betöltési cimet kettővel kell növelni
az utasitás feldolgozásának vége


XJC TEXT "JC"
a következő utasitás

Irodalom

[1] Donovan, J.J.: Systems Programming, McGraw-Hill Book Company, New York, 1972.

[2] Gries, D.: Compliler Construction for Digital Computers, John Wiley and Sons, Inc. New York, 1971.

[3] Lee, J.A.N.: The Anatomy of a Compiler, Reinhold Publishing Corp., New York, 1967.

[4] Varga László: Rendszerprogramozás, ELTE TTK, Tankönyvkiadó, Budapest, 1975.

Summary

A method for the generation of assemblers

Zoltán Csörnyei

Similarities to be found in assemblers made the development of a system, intended to be used for the simple and rapid generation of these assembler programs necessary and possible. Assemblers produced this way consist of two parts: the body, which is common in every single assembler, and the description table, which depends on the nature of the given assembler. The main features of the assembler are determined by the body, for example the number of passes needed, the handling of symbols, its inputs and outputs etc., while the directives and the assembler's repertory of instructions are defined by the description table.

Р Е З Ю М Е

Один метод для разработки ассамблеров

Золтан Чернеи

Сходство ассамблеров привело к необходимости и возможности разработки такой системы, спомощью которой эти транслирующие программы разрабатываются просто и без больших временных затрат. Выполненные таким образом трасляторы состоят из двух частей: из общей части для всех трансляторов и из таблицы описания характеризующей данную транслирующую программу. Общая часть определяет характер транслятора, например, число этапов трансляций, обращение символами, вход и выход, а таблицей описания определяются директивы и команды ассамблера.