

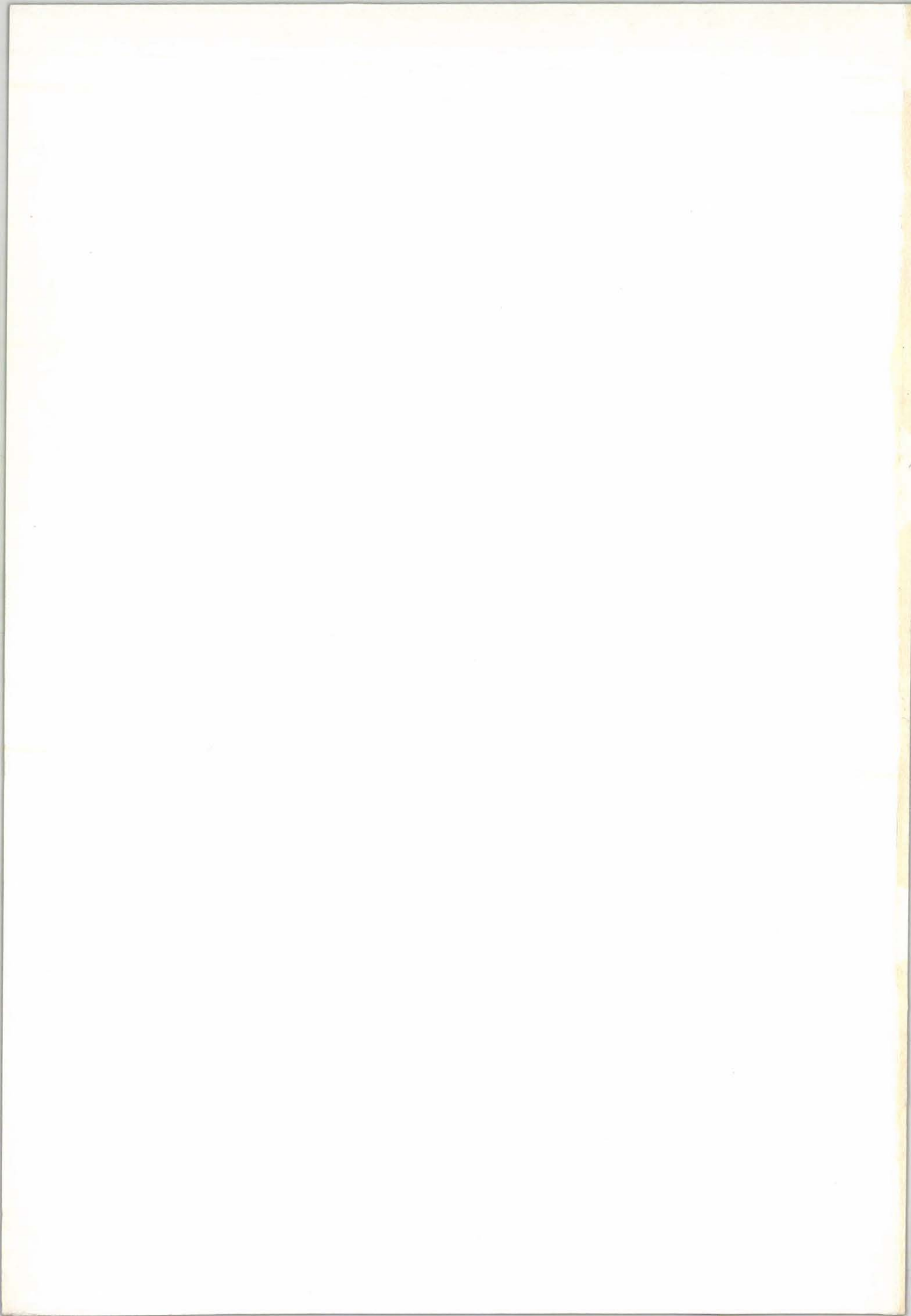
317.057

tanulmányok

7/1973

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest





Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutató Intézete
Computer and Automation Institute, Hungarian Academy of Sciences
Institut für Rechen-technik und Automatisierung der Ungarischen Akademie der Wissenschaften
Исследовательский Институт Вычислительной Техники и Автоматизации Венгерской Академии Наук

A CHANGE nyelv/multiprocesszor

Legendi Témás

Budapest, 1972.

A kiadásért felelős
Dr. Vámos Tibor
az
MTA Számítástechnikai és Automatizálási
Kutató Intézetének
igazgatója

MAGYAR
TUDOMÁNYOS AKADEMIÁ
KÖNYVTÁRA

Készült az Országos Műszaki Könyvtár és Dokumentációs Központ
házi sokszorosítójában
F.v.: Janoch Gyula

B E V E Z E T É S

A dolgozat részletesen ismerteti a CHANGE nyelv elvi és gyakorlati céljait, felépítését, általános és konkrét alkalmazásait, fordító-, értelmező- és végrehajtó programjainak szerkezetét.

A bevezetés röviden ismerteti azokat a problémákat és a megoldásukra irányuló javaslatokat, amelyek a nyelv kialakításához vezettek, kiemelve az új illetve viszonylag új elemeket.

A hagyományos programozási nyelvek különböző előnyeik ellenére a feladatok jelentős részénél csak nehézkesen alkalmazhatók (különösen célorientált nyelvek feldolgozására, adaptív programok írására, szintaktikus elemzésre).

A nehézségek főbb okai:

1. A magasabb szintű nyelvek megtartják a gépi kód lineáris végrehajtási módját (az utasítások egymás után kerülnek végre hajtásra, kivéve, ha vezérlésátadó utasítás ír elő más sorrendet). Az algoritmusok jelentős részénél, a célorientált nyelvek túlnyomó többségénél a természetes végrehajtási mód nem lineáris, és ezeknek a feladatoknak lineáris programmá való átkódolása többletmunkát okoz a programozónak (az elkészült program kevéssé tükrözi az eredeti feladatot).
2. Magasabb szintű nyelveknél elvész a gépi kódban elérhető csaknem teljes közvetlen hozzáférés a program végrehajtását befolyásoló értékekhez. (A gépi kódú programok közvetlenül módosíthatják a programot tartalmazó rekeszek, az utasításslámláló (k), valamint az utasításslámláló-módosító (k) tartalmát.)
3. A magas szintű nyelvek szintakszisa és szemantikája kötött, a fordítást és a futást vezérlő paraméterek többsége a fordító-végrehajtó program kizárólagos ellenőrzése alatt áll, dinamikusan nem változtatható.

A javasolt megoldás:

1. A végrehajtási mód és vezérlése.

A lineáris végrehajtási módot az jellemzi, hogy a programokat 1 processzor hajtja végre, 1 utasításszámláló felhasználásával. (A vezérlésátdó utasítások közvetlenül előírják az utasításszámláló új értékét, az egyéb utasítások végrehajtása után 1-gyel nő az utasításszámláló tartalma.) Ezt az eljárást általánosítva, egy programot tetszőleges számú, önálló utasításszámlálóval és utasításszámláló-módosítóval rendelkező processzor hajt végre. (Általános végrehajtási mód.) A processzorokat vezérlő (indító, megállító stb.) és az utasításszámláló-módosító értékét előíró utasítások bevezetésével tetszőleges nem lineáris végrehajtási mód is (dinamikusan) programozható.

2. A végrehajtási módot vezérlő utasítások közvetlen hozzáférést biztosítanak a végrehajtást befolyásoló legfontosabb értékekhez. A programok futás közbeni tetszőleges módosítását az egyes utasításokat a programon belül áthelyező utasítások végzik. Ezek az utasítások vagy változatlanul helyezik át az előírt utasítást, vagy pedig az áthelyezés előtt egyes kijelölt változókat aktuális értékükkel helyettesítenek is.

(Mivel egy programon belül a teljes utasításkészlet rendelkezésre állhat, így a program bármely utasításának helyére tetszőleges utasítás kerülhet.)

3. A kötöttségek feloldását elsősorban az új adattípusok definiálását, valamint automatikus feldolgozását lehetővé tevő kiterjesztő utasítások biztosítják.

Lényeges többlet a hagyományos kiterjeszthető nyelvekkel szemben, hogy a kiterjesztett nyelv (processzor) végrehajtási módja is előírható a kiterjesztő utasítások szemantikai részében.

Ezenkívül a fordítást és a futást vezérlő paraméterek (nyomkövetési szintek, optimalizálási szintek, mely utasításokat kell csak egyszer végrehaj-

tani, a szubrutinmélység maximális és aktuális értékének előírása stb.) beállítására külön utasítások vannak, amelyek dinamikusan (futás alatt) is alkalmazhatóak.

Összefoglalva:

A nyelv nagyfoku hajlékonysággal rendelkezik, minden lényeges alapelemén lehet műveleteket végezni, így többek között:

- a. az adattípusok és az utasítások szintaktikus és szemantikus leírásán (a nyelv kiterjeszhető);
- b. az utasításokon (a programok futás alatt módosíthatók programozott módon);
- c. a végrehajtási módon (tetszőleges végrehajtási mód írható elő dinamikusan).

Az ezekkel a tulajdonságokkal rendelkező CHANGE nyelv előnyösen alkalmazható nem lineáris végrehajtási móddal rendelkező célorientált nyelvek definiálására és processzoraik automatikus elkészítésére, valamint adaptív programok írására.

A nyelv esetleg alkalmazható univerzális programozási nyelvként is, de ebben az esetben a futás áttekinthetőségét zavaró eszközök használatát korlátozni kell és a nyomkövető utasítások fokozott mértékű alkalmazásával lehet elérni a futás megbízható ellenőrzését.

A. A nyelv célja

1. A programnyelvek linearitásának feloldása
2. A programok futás közben történő (ön) módosítása programozott úton (újrarendelés nélkül)
3. Programozási nyelveket (és egyben processzorokat) kaphassunk kiterjesztéssel egy hajlékony alapnyelvből
4. A programozó egy olyan magasszintű absztrakt számítógépet programozzon, amelyen mindenhez hozzáfér (tehát a nyelv minden lényeges alapelemén lehessen utasításokkal műveleteket végezni)
5. A multiprocesszoros gondolkodásmód kialakítása

B. A célok elemzése

1. A célok közötti összefüggések

A bevezetésben szereplő gondolatmenet világosan mutatja, hogy a negyedik cél a legáltalánosabb.

A linearitás feloldása elérhető a végrehajtási mód fogalmának bevezetésével és a végrehajtási módot előíró utasításoknak a nyelvbe való beépítésével.

A programok futás közbeni módosítását, az utasításokon műveleteket végző áthelyező utasítások biztosítják.

A kiterjeszthetőséget az adattípusok és az utasítások szintaktikus és szemantikus leírásán műveleteket végző utasítások biztosítják. A nem lineáris végrehajtási módot előíró utasítások felhasználása az új utasítások szemantikus leírásában lehetővé teszi tetszőleges végrehajtási móddal rendelkező kiterjesztett nyelv (processzor) létrehozását.

2. A célok következményei

a. Feltétlenül interpretálásra vagy inkrementális fordításra [2] van szükség a programot módosító (utasításokat áthelyező) utasítások, a futás közben változtatható végrehajtási mód, és a nyomkövetési utasítások realizálásához.

b. Interpretálás vagy inkrementális fordítás esetén a programok illetve egyes utasításaik belső ábrázolása eleve megfelel az on-line feldolgozás követelményeinek, így az utasításkészlet minimális kiegészítése már lehetővé teszi a multiprocesszor on-line felhasználását is.

C. A nyelv összefoglaló jellemzése

1. Műveleteket lehet végezni egész, valós, logikai, karakter, szövegtípusú változótömbök illetve egyszerű listák elemein, utasításokon, a CHANGE multiprocesszor végrehajtási módját szabályozó paramétereken, valamint az adattípusok és az utasítások szintaktikus és szemantikus leírásán.

2. A CHANGE nyelv alapnyelv jellegű.

A nyelv alkalmas általános algoritmusok leírására; alkalmazásának fő célja célorientált nyelvek definiálása, és egyben processzorok automatikus elkészítése.

3. Az alapnyelv jellegnek megfelelően az 1. pontban felsorolt objektumokon nagyszámú elemi művelet végezhető. A műveletek rendszere minimális redundanciát tartalmaz, az általában használt összetett műveletek felépíthetők az elemi műveletek alkalmazásával.

4. Az alapnyelv kiterjeszhetősége

a. Új adattípusok és utasítások építhetők be a nyelvbe.

b. A párhuzamos végrehajtást előíró és az utasításszámlálómódosító értékét kijelölő utasítások segítségével a programok végrehajtása tetszőleges, nem lineáris módon történhet (futás alatt dinamikusan változhat).

c. A CHANGE multiprocesszor végrehajtási módját előíró, a b. pontban említett utasítások szerepelhetnek az utasításkiterjesztő utasítás szemantikai részében, így speciális végrehajtási móddal rendelkező processzorok (nyelvek) definiálhatók. (Egyes speciális problémaosztályok számára természetesebb lehet (kialakult felfogásához, jelöl-

lésmódjához, illetve nyelvéhez közelebb állhat) valamely nem lineáris végrehajtási mód, mint az univerzális lineáris végrehajtási mód.)

- d. A végrehajtási módot dinamikusan (futás közben) befolyásolhatják a nyomkövető utasítások.

D. A nyelv felépítése

1. Egy program felépítése

a. Egy program tetszőleges számú, egymást követő utasításokból áll.

b. Az utasítások lehetnek egyszer végrehajtandó vagy állandó utasítások, ezek tetszőleges sorrendben állhatnak.

Az állandó utasítások mindannyiszor végrehajtásra kerülnek, amikor a vezérlés rájuk kerül. Az egyszer végrehajtandó utasítások csak akkor hajthatók végre, amikor először kerül rájuk a vezérlés, címkéjüket és a programban elfoglalt helyüket megtartják ezután is. CHANGE utasítás(ok) is hivatkozhat(nak) ezekre az utasításokra továbbra is. Ha a vezérlés ismételten egy egyszer végrehajtandó utasításra kerül, akkor az adott végrehajtási lépés (ld. a D.2.b pontot) során az ezt az utasítást végrehajtó processzor nem végez semmilyen műveletet.

Az alapnyelv minden utasítása állandó utasítás.

c. Minden program utolsó utasítása a FINIS utasítás, amely a program végét jelzi a fordító-értelmező program számára. A vezérlés soha nem kerülhet rá.

d. Az utasítások pozitív egész számokkal címkézhetők.

e. Minden utasítás (belső) sorszámmal rendelkezik.

f. Az első utasítás sorszáma 1, a rákövetkező utasítások sorszáma mindig eggyel nő.

2. Egy program végrehajtása (a multiprocesszor)

- a. Egy CHANGE programot tetszőleges számú, egyidejűleg működő processzor hajthat végre. A processzorok között alá-, (fölé-) illetve mellérendeltségi viszony áll fenn, ez szabja meg, hogy az egyes processzorok hogyan vezérelhetik egymás működését.
- b. Egy program végrehajtásának egy lépése során minden működő processzor 1-1 utasítást hajt végre. Egy program végrehajtása tetszőleges számú, egymást követő végrehajtási lépésből áll.
- c. Minden processzor rendelkezik egy utasításszámlálóval, amelynek tartalma pozitív egész szám.
- d. Minden processzor rendelkezik egy utasításszámláló-módosítóval, amelynek tartalma egész szám.
- e. A végrehajtás egy adott lépése során egy processzor számára utasításszámlálójának tartalma határozza meg a végrehajtandó utasítás sorszámát.
- f. Az utasításszámláló tartalma az utasítás végrehajtása után megváltozik.
- g. Az UTSZ utasítás közvetlenül kijelöli az utasításszámláló tartalmát.
- h. A vezérlésátadó utasítások kijelölik a következő végrehajtandó utasítás címkéjét, az ennek a címkének megfelelő (belső) sorszám kerül az utasításszámlálóba.
- i. Ha nem UTSZ illetve vezérlésátadó utasítás került végrehajtásra,

akkor az utasításslámláló tartalmához hozzáadódik az utasításslámláló-módosító tartalma.

- j. Az UTSZM utasítás meghatározza az utasításslámláló-módosító tartalmát (és így közvetve a következő végrehajtandó utasítás sorszámát).
- k. Egy program végrehajtásának a kezdetekor egy processzor működik, utasításslámlálójának tartalma +1, utasításslámláló-módosítójának tartalma +1. A processzorhoz az 1 sorszám van hozzárendelve.
- l. A COPROCESSOR utasítással (a végrehajtó processzorhoz képest) mellérendelt, a SUBPROCESSOR utasítással (a végrehajtó processzorhoz képest) alárendelt új processzor működése indítható el. A processzorok mellérendeltsége (és alárendeltsége) reflexív, szimmetrikus és tranzitív reláció. Az egymással mellérendeltségi viszonyban levő processzorok ugyanazoknak a processzoroknak vannak alárendelve.
- m. A COPROCESSOR illetve a SUBPROCESSOR utasításban meg kell adni az új processzor által elsőnek végrehajtandó utasítás címkéjét, és az elindított processzorhoz sorszámot kell hozzárendelni (az új processzor a következő végrehajtási lépésben kezd működni, az indító processzor is folytatja működését).
- n. Egy processzor akárhány processzor működését elindíthatja, a működő processzorokhoz rendelt sorszámok különbözőek kell hogy legyenek.
- o. Bármely processzor számára a WAIT utasítás végrehajtása saját működésének felfüggesztését eredményezi.
- p. A COWAIT utasítás előírja valamely, az utasítást végrehajtó processzorhoz képest alá- vagy mellérendelt processzor működésének felfüggesztését.

- q. A NOWAIT utasítás hatására valamely, az utasítást végrehajtó processzorhoz képest alá- vagy mellérendelt processzor folytatja működését.
- r. Egy processzor működése a STOP utasítás hatására fejeződik be (ekkor sorszáma is megszűnik).
- s. A SUBSTOP utasítás előírja valamely, az utasítást végrehajtó processzorhoz képest alárendelt processzor működésének befejezését.
- t. Egy teljes program végrehajtása akkor fejeződik be, amikor egyetlen processzor sem működik (ebből a szempontból a felfüggesztett működésű processzort nem működőnek kell tekinteni). Az utasítást végrehajtó processzornak alárendelt összes processzor működését megszünteti a GENERAL STOP utasítás végrehajtása.

3. Az alapnyelvben megengedett változó típusok: egész, valós, logikai, karakter, szöveg.

4. A változóazonosítók tetszőleges hosszúságú betűvel kezdődő alfanumerikus karaktersorozattal adhatók meg.

A változóazonosítók mindig egy változótömb azonosítójaként szolgálnak.

5. Az egyes változótömböknek a számítógépben 1-1 összefüggő memóriaterület felel meg, amelyen folyamatosan helyezkednek el a tömb elemei, (növekvő) indexeik sorrendjének megfelelően.

6. Az egész, valós, logikai és karakter típusú tömbök tárolása.

a. Azonos típusú tömbök egyes elemei azonos számú memóriarekeszben kerülnek tárolásra.

b. Különböző típusú tömbelemek különböző számú rekeszben kerülhetnek tárolásra.

c. Az a. szerint tárolt tömbelemekre indexeik segítségével lehet hivatkozni mind a memóriába író, mind a memóriából olvasó utasításokban.

7. A szöveg típusú változótömbök tárolása.

a. A szöveg típusú változótömbök egyes elemei különböző számú memóriarekeszt foglalhatnak el, azaz a szöveg típusú változók értéke tetszőleges hosszúságú karaktersorozat lehet.

- b. A szöveg típusu tömbelemekre indexeik segítségével (csak) a memóriából olvasó utasításokban lehet hivatkozni (azaz a tömb elemeinek értékét megkaphatjuk a megfelelő indexre való hivatkozással, a tömb valamely elemének nem adhatunk értéket indexére való hivatkozással).
- c. A szöveg típusu változótömbök elemei csak indexeik növekvő sorrendjében kaphatnak értéket, (a továbbiakban ismertetésre kerülő karakter-szöveg típusu értékadó utasításokkal) ez az értékadás bármikor kezdődhet újra a tömb első eleménél, de ekkor a tömbelemek előző értéke elvész.

8. Az általános változók szerkezete

a. A változók lehetnek index nélküliek vagy indexesek.

(Az index nélküli változó mindig a tömb első elemét jelenti.)

b. Index lehet pozitív egész szám, pl. $a(1)$, $b(5)$, $\text{alfa}(20)$, vagy pozitív értéket felvevő egész típusu index nélküli változó, pl. $a(j)$, $b(k)$, $\text{alfa}(\text{beta})$.

c. Egy indexes változónak lehet kitevője

A kitevő nem negatív egész szám vagy nem negatív értékeket felvevő egész típusu változó, pl. $a(j)$, $b(5)$, $\text{alfa}(\text{beta})$.

A kitevő jelentése:

$a(j) = a(j) = j$	ha $n = 0$
$a(j) = a(j) = a(j)$	ha $n = 1$
$a(j) = a(j) = a(a(a(j)))$	ha $n = 3$

d. Index lehet indexes változó is

Pl. $a(b(3))$, $a(b(3))$, $c(\text{alfa}(\text{gamma}))$, $c(\text{beta}(\text{delta}(9)))$; stb.

e. A változók lehetnek többdimenziósak, (lehet több indexük, ezeket vesszővel kell elválasztani) a dimenziószám tetszőleges.

f. A legbelső zárójelben álló változó(k) vagy a zárójelet nem tartalmazó változó előtt \times jel állhat. (Csak a CHANGE utasításokban van szerepe.)

9. Deklarációs utasítások

a. Tipusdeklarációs utasítások

A típusdeklarációs utasítások az INTEGER (egész), REAL (valós), LOGICAL (logikai), CHARACTER (karakter), TEXT (szöveg) (v.ö.3.pont) alapszavak valamelyikével kezdődnek, majd tetszőleges számú, vesszővel elválasztott tömbdeklarátorral folytatódnak. Tömbdeklarátor lehet tömbnév, vagy tömbnév és utána (kerek zárójelben álló, vesszővel elválasztott) tetszőleges számú konstans vagy változóazonosító. A típusdeklarációs utasítás a tömbnévhez hozzárendeli az utasításban szereplő alapszónak megfelelő típust, és helyet biztosít a tömb számára a memóriában a tömbnév után felsorolt (a tömb méretét megadó) konstansok illetve változóazonosítók értékének megfelelően. (Ha a tömbdeklarátor csak tömbnévből áll, akkor egyelemű, egydimenziós tömb számára történik helyfoglalás.) Szöveg típusu változók esetén a tömb első szabad eleme az 1 indexű elem. (Egydimenziósnak tekintve a tömböt.) Ha egy tömb típusa megváltozik az utasítás végrehajtásának hatására, akkor elemeinek értéke elvész (definiálatlan).

b. Tömbméret deklarációs utasítások

A DIMENSION alapszó után tetszőleges számú tömbdeklarátor állhat. Az egyes tömbdeklarátorokban szereplő változóazonosítók, illetve konstansok száma adja meg az adott tömb dimenziószámát, (aktuális) értékük határozza meg a tömb méretét (méreteit). Az utasítás végrehajtása helyet biztosít a memóriában az egyes tömbök számára méretüknek (és típusuknak) megfelelően.

Ha egy tömb dimenziószáma megváltozik az utasítás végrehajtásának hatására, akkor elemeinek értéke elvész. Ha egy tömb dimenziószáma változatlan marad, akkor mindazon (rég)i elemeinek értéke megőrződik, amelyek indexei az új indexhatárokon belül esnek.

AUTODIM

Az AUTODIM utasítás végrehajtása után az egyes változó tömbök mérete automatikusan megnő, ha a program a deklarált méretnél nagyobb méretű tömböt kísérel meg használni.

NO AUTODIM

Megszünteti az AUTODIM utasítás hatását.

10. Az értékadó utasítások -ban az egyenlőségjel baloldalán mindig egy általános változó áll, a jobboldalon

vagy egy általános változó,

vagy műveleti jellel összekötött két általános változó,

vagy relációjellel összekötött két általános változó,

vagy egy belső függvénynév és egy változó neve állhat (a jobboldalon változó helyett bárhol állhat konstans).

A jobboldali kifejezés kiértékelésével kapott érték típusa meg kell, hogy egyezzen a baloldali változó típusával. A baloldalon álló változó felveszi a jobboldali kifejezés aktuális értékét.

a. Műveleti jelek: + , - , × , / , xx

(aritmetikai)

.NOT., .AND., .OR., .NAND., .NOR.

(logikai)

Megjegyzés: a .NOT. műveleti jel előtt nem állhat változó, utána egy változó állhat.

b. Belső függvények: sin, cos, atan, abs, alog, alog 10, exp, sqrt, max, min, float, ifix

c. Relációjelek: .EQ., .NE., .GT., .GE., .LT., .LE., .IN., .NI.

(jelentésükről ld. a 11.c. pontot)

d. Konstansok:

Az egész típusu konstansok tízes számrendszerbeli számjegyekből állnak. (A számjegyek maximális száma az adott számítógéptől függ.)

A valós konstansok szintén tízes számrendszerbeli számjegyekből állnak, valahol ki kell tenni a tizedespontot.

A logikai konstansok: `.TRUE.` (igaz) és
`.FALSE.` (hamis)

A karakter konstansok: az adott számítógépen elérhető teljes karakterkészlet, az egyes karaktereket két ferde zárójel közé kell zárni.

A szöveg konstansok: kezdő és végzárójel közötti tetszőleges karakter sorozatok.

11. A vezérlésátadó utasítások

a. Az összes vezérlésátadó utasításban a vezérlésátadás helyét (helyeit) megadó paraméter(ek) lehet(nek) pozitív egész szám(ok), vagy pozitív értéket felvevő egész típusu változó(k) (jelölésük P és egy utána álló egész szám).

b. Feltétlen vezérlésátadás

GO TO P1

A vezérlés a P1 címkéjű utasításra adódik.

c. Feltételes vezérlésátadás

IF (V1) P1, P2, P3

Ha a V1 egész vagy valós típusu változó aktuális értéke

negatív P1

nulla akkor a P2 címkéjű utasításra kerül a vezérlés

pozitív P3

IF (V1.REL.V2) P1, P2

Ha a V1 és V2 megegyező típusu általános változók (konstansok) között fennáll a REL reláció, akkor a P1 címkére, ellenkező esetben a P2 címkére adódik a vezérlés.

A megengedett relációk: EQ(=), NE(\neq), GT(>), GE(\geq), LT(<), LE(\leq),
IN(\subset), NI($\not\subset$)

Az .EQ. és a .NE. relációjel tetszőleges típusú változók között állhat, a .GT. a .GE. az .LT. és az .LE. relációjel egész, illetve valós típusú változók között állhat, az .IN. és az .NI. relációjel pedig szöveg típusú változók között állhat.

IF (L) P1, P2

Ha az L logikai változó aktuális értéke igaz (.TRUE.) akkor a P1 címkéjű utasításra, ellenkező esetben a P2 címkéjű utasításra kerül a vezérlés.

d. Ciklusutasítás

DO P1 I1=I2,I3,I4

I1, I2, I3, I4 egész típusú változó (konstans).

P1 (aktuális) értéke egy REPEAT (cikluszáró) utasítás címkéjét kell, hogy megadja.

Ha I2 értéke nem haladja meg I3 értékét (I3 a ciklus változó maximális értéke), akkor az utasítás hatására az I1 ciklusváltozó felveszi az I2 kezdőértéket. Ellenkező esetben a P1 címkéjű REPEAT utasítás után (a végrehajtási mód szerint) következő utasításra történő feltétlen vezérlésátadást ír elő az utasítás.

(P1) REPEAT

A REPEAT utasítás végrehajtásakor az I1 ciklusváltozó aktuális értékéhez hozzáadásra kerül I4. Ha az I1 ciklusváltozó értéke nem haladja meg I3 aktuális értékét, akkor a hozzátartozó DO utasítás után (a DO utasítás utolsó végrehajtásakor érvényes végrehajtási mód szerint)

következő utasításra, ellenkező esetben a REPEAT utasítás után (a végrehajtási mód szerint) következő utasításra adódik a vezérlés.

A ciklus futása alatt P1, I1, I2, I3, I4 értéke módosítható. Az összetartozó DO - REPEAT utasításpárt különböző processzorok is végrehajthatják.

e. Szubrutinhívás

SUBR P1

A SUBR P1 utasítás hatására a P1 című utasításra kerül a vezérlés, és tárolódik a SUBR utasítás után (a végrehajtási mód szerint) következő ("visszatérő") utasítás (belső) sorszáma.

EXIT

Az EXIT utasítás hatására feltétlen vezérlésátadás történik az EXIT utasítást végrehajtó processzor által utoljára végrehajtott szubrutinhívás által meghatározott visszatérő utasításra.

A visszatérő utasítás sorszámanak tárolása (egy-egy processzoron belül) az .SDI. azonosítójú tömbben történik az alábbiaknak megfelelően. Az .SDI. tömbnek a fordító program az INTEGER.SDI.(50) utasításnak megfelelően biztosít helyet a memóriában (automatikusan), az .SD. egész típusú változónak a 0 értéket adja. Minden szubrutinhívásnál .SD. értéke 1-gyel nő először, majd .SDI.(.SD.)-be kerül beírásra a megfelelő visszatérő cím. Minden EXIT utasítás hatására .SDI.(.SD.) kerül az adott processzor utasításslámlálójába, majd .SD. értéke csökken 1-gyel.

f. A szubrutinhívást és a visszatérést befolyásoló utasítások

.SDI.(I1)

Az utasítást végrehajtó processzorhoz tartozó .SDI. tömb mérete I1 (aktuális) értéke lesz. Az új indexhatáron belül eső régi tömbelemek értéke megőrződik.

AUTO .SDI.

Az utasítást végrehajtó processzorhoz tartozó .SDI. tömb mérete automatikusan nő, illetve csökken a végrehajtás során.

NO AUTO .SDI.

Az utasítást végrehajtó processzorhoz tartozó .SDI. tömb (pillanatnyi aktuális) mérete rögzített marad.

I1 = .SD.

Az I1 egész típusu változó felveszi az utasítást végrehajtó processzorhoz tartozó .SD. értékét.

.SD. = I1

Az utasítást végrehajtó processzorhoz tartozó .SD. felveszi az I1 változó értékét.

12. A CHANGE utasítások

a. CHANGE I1 = I2, I3 (utasításmódosítás)

I1, I2, I3 egész típusu általános változó (konstans) lehet. Az utasítás végrehajtásakor I1, I2, I3 (aktuális) értéke pozitív egész szám kell, hogy legyen, a programban szerepelnie kell I1, illetve I3 címkejű utasításnak, I2 címkejű utasítás nem szerepelhet (kivéve ha $I2 = I1$, vagy $I1 = I2 = I3$).

Az utasítás hatására az I1 címkejű utasítás helyére az I3 címkejű utasítás aktuális értéke kerül, I2 címkével. (Az I3 címke, és az I3 címkejű utasítás változatlanul marad.)

Egy utasítás aktuális értékét úgy kapjuk, hogy azon változók helyébe, amelyek előtt x áll, beírjuk aktuális értéküket. (Egy utasítás aktuális értéke tehát szintén utasítás, egy olyan utasítás aktuális értéke, amelyben x jel nem szerepel, saját maga.)

Például az $a(*I) = *S$ utasítás aktuális értéke

az $a(2) = 5.1$ utasítás, ha I aktuális értéke 2, aktuális értéke 5.1

(Az $a(*I) = *S$ utasítás általában azonos hatású, mint az $a(I) = S$ utasítás, csak ha CHANGE utasításban szerepel, akkor kell aktuális értékét venni figyelembe.)

Új utasítással akkor és csak akkor bővíthet a program, ha az I3 címkejű utasítás aktuális értéke nem egyezik meg az I3 címkejű utasítással.

Speciálisan fennállhat $I1 = I2$ (az I1 címke nem változik) illetve

I1 = I3 (az I1 címkeű utasítás nem változik, ha * jel nem szerepel benne). I1 = I2 = I3 esetén az utasítás végrehajtódik, de hatására semmilyen változás sem következik be (ha az I1 címkeű utasításban * jel nem szerepel).

A programban leírt bármely utasítás (akárhányszor) áthelyezhető a program bármely utasításának helyére. (Tehát a programban eredetileg egyszer szereplő utasítások több helyen is állhatnak - természetesen más-más címkével -, egyes utasítások eltűnhetnek a programból (helyükre valamely más utasítás kerül) speciális esetként elérhető a program utasításainak átrendezése, illetve néhány (alap) utasításból álló utasításkészletből vett utasításokból álló program futás alatt generálása (a generált program azonnal futhat, ha a generáló program ráadja a vezérlést).

b. CHANGE V1 = V2 (paramétermódosítás)

V1 és V2 azonos típusú általános változó (konstans). V1 helyére a programban mindenütt V2 kerül.

13. A NULL utasítás

A NULL utasítás üres utasítás, végrehajtása nem eredményez változást sem a programban, sem a változótartományban. Szerepe lehet a CHANGE utasítás végrehajtásakor - mintegy helyet biztosít a betoldandó utasításoknak, ezenkívül mint üres utasítás, eredményezheti egy-egy processzor késleltetését, üresjárását, ha pl. össze kell hangolni két vagy több processzor egyidejű működését.

14. A végrehajtási módot előíró utasítások

A végrehajtási módot előíró utasítások részbeni ismertetését és jellemzését tartalmazza a D.2. pont, amely leírja, hogy a CHANGE multiproceszor hogyan hajt végre egy programot, ezért ez a pont az utasítások tömör szintaktikus és szemantikus leírását tartalmazza.

a. UTSZ utasítások

I1 = UT SZ

Az I1 egész típusu változó felveszi az (utasítást végrehajtó processzorhoz tartozó) utasításszámláló értékét (az utasításszámláló értéke az I1 = UT SZ utasítás belső sorszáma).

UT SZ = I1

Az (utasítást végrehajtó processzorhoz tartozó) utasításszámláló felveszi az I1 egész típusu változó (konstans) értékét. A következő végrehajtásra kerülő utasítás belső sorszáma I1 (aktuális) értéke.

b. UTSZM utasítások

I1 = UTSZM

Az I1 egész típusu változó felveszi az (utasítást végrehajtó processzorhoz tartozó) utasításszámláló-módosító értékét.

UTSZM = I1

Az (utasítást végrehajtó processzorhoz tartozó) utasításszámláló-módosító felveszi az I1 egész típusú változó (konstans) értékét.

Ha I1 (aktuális) értéke nem nulla, akkor a következő végrehajtandó utasítás (belső) sorszámát úgy kapjuk, hogy az utasításszámlálóhoz (amelynek tartalma az UTSZM = I1 utasítás belső sorszáma) hozzáadjuk az utasításszámláló-módosító új értékét.

Ha I1 (aktuális) értéke nulla, akkor a következő végrehajtandó utasítás (belső) sorszámát úgy kapjuk, hogy az utasításszámláló tartalmához hozzáadjuk az utasításszámláló-módosító régi (az UTSZM = 0 utasítás végrehajtása előtti) értékét. Az utasításszámláló-módosító tartalma 0 lesz az utasítás végrehajtása után. (Tehát ha a következő végrehajtásra kerülő utasítás nem UTSZ, UTSZM vagy vezérlésátadó utasítás, akkor mindig ugyanezt az utasítást fogja végrehajtani az adott processzor. Ennek a lehetőségnek a felhasználásával a szimulációs alkalmazásokat tárgyaló G.2. pont foglalkozik.)

c. A párhuzamos végrehajtást vezérlő utasítások

COPROCESSOR I1,I2

SUBPROCESSOR I1,I2

I1,I2 - egész típusú változó (konstans) kell, hogy legyen. A programban szerepelnie kell I1 címkejű utasításnak. A multiprocesszorban nem lehet I2 sorszámú processzor (sem működő, sem felfüggesztett állapotban).

Az utasítás hatására a multiprocesszor működésének következő lépése során az I1 címkejű utasítás végrehajtásával elkezd működését az I2 sorszámú processzor.

Az utasítást végrehajtó (indító) processzor is folytatja működését.

Az új processzor alá van rendelve mindazon processzoroknak, amelyeknek az indító processzor alá van rendelve.

A COPROCESSOR utasítással elindított processzor mellérendeltségi viszonyba kerül az indító processzornal, és az azzal mellérendelt processzorokkal.

A SUBPROCESSOR utasítással elindított processzor alá van rendelve az indító processzornak.

Az elindított alárendelt processzor mellérendeltségi viszonyba kerül az indító processzor által előzőleg elindított (az annak közvetlenül alárendelt processzorokkal).

WAIT

A WAIT utasítást végrehajtó processzor felfüggeszti működését.

COWAIT I1

I1 - egész típusú változó (konstans)

I1 (aktuális) értéke valamely, az utasítást végrehajtó processzornak alá- vagy mellérendelt processzor sorszáma kell, hogy legyen.

Az utasítás hatására az I1 sorszámú processzor felfüggeszti működését. Egy processzor működésének felfüggesztése nincs hatással a processzorok mellé- illetve alárendeltségi viszonyára.

NOWAIT I1

I1 - egész típusú változó (konstans)

I1 aktuális értéke valamely alá- vagy mellérendelt (felfüggesztett működésű) processzor sorszáma kell, hogy legyen. Az utasítás hatására az I1 sorszámú processzor folytatja működését.

DCONTROL I1,I2

Az utasítás végrehajtása után a multiprocesszor ellenőrzi, hogy az egyes lépések során párhuzamosan végrehajtandó utasítások függetlenek-e egymástól. Az I1 egész típusú változó értéke 0 mindaddig, amíg a végrehajtott utasítások függetlenek, értéke 1-gyel nő minden olyan végrehajtási

lépés során, amelyben nem független utasítások szerepelnek. Az I2 egész típusú tömb tartalmazza az utoljára végrehajtott, nem független utasítások belső sorszámainak.

NO DCONTROL

Megszünteti a DCONTROL utasítás hatását.

A végrehajtás a párhuzamosan végrehajtott utasítások függetlenségének vizsgálata nélkül történik. Az utasítások hatása követhető annak a konvenciónak a figyelembevételével, amely azt írja elő, hogy az alacsonyabb sorszámú processzor által előírt utasítást kell előbb végrehajtani, egy végrehajtási lépés során (egy processzorral rendelkező számítógépen való végrehajtás esetén).

Az utasítások függetlensége azt jelenti, hogy azok (egy processzorral) bármely sorrendben való végrehajtása azonos eredményt ad. (Az adat és a programterületet is, valamint a végrehajtási mód paramétereit is beleértve.)

15. A STOP utasítások

STOP

A STOP utasítást végrehajtó processzor befejezi működését (sorszámuk megszűnik).

SUBSTOP I1

I1 - egész típusú változó (konstans)

I1 aktuális értéke valamely, az utasítást végrehajtó processzornak alárendelt processzor sorszámára kell, hogy legyen. (Sorszámuk megszűnik.)

GENERAL STOP

A végrehajtó processzor, és az annak alárendelt összes processzor befejezi működését (sorszámuk megszűnik).

Egy processzor működésének befejezése (sorszámának megszüntetése), befolyásolhatja a processzorok mellé- illetve (közvetlen) alárendeltségi viszonyát.

A megszünt processzornak (MP) közvetlenül alárendelt processzorok mellérendeltségi viszonyba kerülnek az MP - vel mellérendelt processzorokkal, és közvetlenül alá lesznek rendelve az MP-nek közvetlenül fölérendelt processzornak.

16. Az adatátviteli (input - output) utasítások

Az I/O utasítások file-ok között irnak elő adatátviteli műveleteket. File-nak tekintendők az I/O eszközök, a háttérmemória részei és a memória részei.

A file-okra sorszámukkal (pozitív egész számok) lehet hivatkozni, a file-definiációs utasításokban (előzőleg) meg kell adni, hogy az egyes file-ok hol találhatóak.

A file-ok mérete tetszőleges mértékben kiterjeszthető, ez az adatátviteli utasítások hatására automatikusan történik.

Minden file-hoz tartozik egy programozható (utasítással módosítható) lokációs cím, amely megadja, hogy az adott file-ra vonatkozó adatátviteli műveletet hol kell elkezdni.

(A címzés egysége 1 memóriaelem.)

File definiáló utasítások

FILE F IS ARRAY T

F egész típusú változó(constans).

T szöveg típusú változó(constans).

Az F sorszámú file a T aktuális értéke által adott nevű tömb.

FILE F IS DEVICE I

F,I egész típusú változó(constans).

Az F sorszámú file az I sorszámú input/output egység.

(Az egyes számítógépeken az I/O egységeket egyértelműen sorszámozni kell, így a konkrét multiprocesszorokban ez az utasítás tényleges hozzárendelést valósít meg.)

FILE F IS ON SECONDARY STORAGE

F egész típusú változó(constans).

Az F sorszámú file a háttérmemóriába kerül.

A file-definiáló utasítások végrehajtásának hatására az adott filehoz tartozó lokációs cím értéke 1 lesz.

A lokációs címet meghatározó utasítás:

LOCATE F TO I

F,I egész típusú változó(constans).

Az F file-hoz tartozó cím aktuális értéke I lesz az utasítás végrehajtásának hatására.

Az adatátvitelt előíró utasítás:

FROM F1 TO F2

F1, F2- egész típusú változó(constans) vagy többelem (neve) vagy tömbnév.

Az utasítás az F1 file-ról az F2 file-ra történő adatátvitelt ír elő.
(Az F1 file-on az adat változatlanul megmarad).

Az adatátvitel az F1-hez illetve az F2-höz tartozó lokációs címtől kezdődően történik. Az utasítás végrehajtásának hatására mindkét lokációs címhez hozzáadódik az átvitt adat méretének megfelelő memóriaelemek száma.

F1 és F2 közül (legfeljebb) az egyik lehet többelem (neve) vagy tömbnév, ebben az esetben az adatátvitelben a megfelelő többelem illetve tömb vesz részt és ennek mérete adja meg az átvendő adat méretét. Ha F1 és F2 egyike sem tömbnév illetve többelem(neve) akkor az F2 file-ra át kell vinni az F1 file-nak (a hozzátartozó lokációs címtől kezdődően) a teljes tartalmát.

17. A kiterjesztő utasítások

A kiterjesztő utasítások viszonylag egyszerű szintaktikával és tetszőlegesen bonyolult szemantikával rendelkező új utasítások definiálását teszi lehetővé.

A hagyományos kiterjeszhető nyelveknél nagyobb fokú hajlékonyságot biztosít egyrészt az, hogy a szintaktikus leírás változókat is tartalmazhat és ezek aktuális értékét a szemantikai leírást képviselő programrész felhasználhatja, másrészt pedig ez a szemantikai leírás két különböző szinten is módosíthatja saját magát.

(tartalmazhat CHANGE utasításokat, amelyek a program végrehajtása során aktivizálódnak, és tartalmazhat egy olyan programrészt, amely a programba való bekerülés pillanatában - egyes fordító-értelmező programoknál ez történhet a fordítási fázis során - módosítja a szemantikai leírást az aktuális paraméterek, a program egyes aktuális utasításai függvényében).

a.) Az új utasításokat definiáló utasítás(ok)

EXTEND T1, T2, T3, T4, SYNTAX AT P1, SEMANTICS AT P2

T1, T2, T3, T4 - az alábbiakban ismertetésre kerülő alapszavak, vagy ezeket tartalmazó szövegtípusú változók, T1 kivételével kötelező valamennyit kiírni.

T1: vagy LABELED - az új utasítást mindig címkével kell ellátni
vagy UNLABELED - az új utasítást tilos címkével ellátni

Ha T1 elmarad, akkor az új utasítás állhat címkével is, címke nélkül is.

T2: vagy CHANGE - az új utasítás szemantikáját megadó program CHANGE nyelvű és így a CHANGE könyvtárba kerül.

vagy EXTERNAL - az új utasítás szemantikáját megadó program nem CHANGE nyelvű, így az alap (külső nyelvű) könyvtárba kerül.

A CHANGE értelmező - fordító program tartalmaz egy alapkönyvtárat, ez nem változtatható. Az új utasításokat (adattípusokat) definiáló EXTEND vagy utasításokat törölő (RELEASE) utasítások hatása ideiglenes (csak az azokat tartalmazó program végrehajtása során érvényes) hacsak a program során előzőleg EXTERNAL LIBRARY illetve CHANGE LIBRARY utasítás nem került végrehajtásra. (Ebben az esetben a megfelelő könyvtár tartalma módosul).

Az EXTERNAL LIBRARY utasítás hatására a multiprocesszor megvizsgálja, hogy az EXT nevű file-on (ennek megnyitásáról előzőleg gondoskodni kell) van-e már külső nyelvű könyvtár. Ha van, akkor ennek alapján folytatódik a fordítás és a végrehajtás; ha nincs, akkor átmásolódik az alapkönyvtár az EXT file-ra és ennek alapján folytatódik a fordítás és a végrehajtás.

A CHANGE LIBRARY utasítás hatására a multiprocesszor megvizsgálja, hogy a CH azonosítójú (nyitott) file-on van-e CHANGE könyvtár. Ha van, akkor ennek tartalmát is figyelembevéve folytatódik a fordítás és a végrehajtás; ha nincs, akkor egy (egyenlőre üres) CHANGE nyelvű könyvtár kerül a CH file-ra.

A speciális utasítások között (19.pont) ismertetésre kerül egy hatékonyabb és egyben bonyolultabb könyvtárszervezésre szolgáló utasításcsoport. Ezek alkalmazása lehetővé teszi, hogy a multiprocesszor egyes processzorai különböző CHANGE (nyelvű) könyvtárakkal rendelkezzenek.

T3: vagy OPEN - az új utasítás helyére nyílt szubrutinként kerül az új utasítás szemantikáját meghatározó CHANGE program (a megfelelő paraméterátadással)

vagy CLOSED - az új utasítás végrehajtásakor zárt szubrutin hívás aktivizálja az új utasítás szemantikáját leíró programot

T3 = OPEN csak akkor lehet, ha T2 = CHANGE

T4: vagy PERMANENT - az új utasítás állandó utasítás

vagy NONPERMANENT - az új utasítás egyszer végrehajtandó utasítás

P1, P2 - pozitív értékeket felvevő egész típusú változó (constans) lehet, (aktuális) értékük azokat a címkeket adja meg, ahol az új utasítás szintakszisa illetve szemantikája a programban található.

A SYNTAX AT és a SEMANTICS AT alapszavak elhagyhatók.

Az új utasítás szintaktikus leírása kétféle módon történhet:

a) <P1> SYNTAX T₁,T₂,.....T_n

T₁, T₂, . . . T_n szövegtípusú változó (constans) lehet.

Az egyes T_i-k (i=1,2,...n) aktuális értéke lehet

- i) alapszó (pl. SYNTAX,EXTEND,DO,stb)
- ii) adattípust megadó azonosító (pl. INT,REAL,LOG,CHAR,TEXT az alapnyelv adattípusai esetén, vagy valamely típuskiterjesztő utasításban szerepelt T₂)
- iii) *jel után álló egész típusú változó, név vagy constans (szövegformában)
- iv) kezdő vagy végzőjel

T₁, T₂,...T_n megadja, hogy balról-jobbra haladva egymás után milyen rögzített alapszavak illetve milyen típusú változók szerepelhetnek az új utasításban. Az alapszavak és adattípusok egyszerű felsorolásán kívül lehetséges az esetleges ismétlődő részek tömör megadása a következő módon:

Ha T₁, T₁₊₁,.....T_k az ismétlődő rész, akkor ezt zárójelek közé helyezzük (T₁-ben kezdő, T_k-ben záró zárójelet helyezünk el). Ha az ismétlődések száma előre rögzített, akkor T_{k+2} - ben egy olyan szöveget helyezünk el, amelynek első karaktere *, az ezután következő számjegyek írják elő az ismétlődések számát.

Ha az ismétlődések száma tetszőleges, akkor T_{k+2} - ben egy olyan szöveget helyezünk el, amelynek első karaktere *, az ezután következő karakterek egy egész típusú változó nevét adják meg. Az új utasítás szintaktikus elemzése során ennek az egész típusú változónak az értéke az ismétlődések tényleges száma lesz. A továbbiakban ismertetésre kerülő szemantikai utasítás felhasználhatja ezt az értéket, ilyen módon más-más jelentés tulajdonítható a különböző ismétlésszámú (de egyébként azonos strukturájú) utasításoknak.

Több ismétlődő rész is lehet, az ismétlődő részek (teljes egészükben) egymásba lehetnek ágyazva (tetszőleges mélység megengedett).

Az utasítás írható

<P1> SYNTAX T ARRAY

formában is, ebben az esetben a teljes T szöveg típusú tömb játssza T₁, T₂,.....T_k szerepét.

b) <P1> PROGRAM FOR SYNTAX P3-P4. K.T

P3,P4 - pozitív egész értékeket felvevő egész típusú változó (constans) lehet, (aktuális) értékük két, a programban szereplő címkét ad meg.

A P3 címkéjű utasítással kezdődő és a P4 címkéjű utasítással végződő programrész a szintaktikus analízist végző rutinhoz kerül kiegészítésként. Ha ez a rutin valamely utasítást nem tud feldolgozni, akkor átadja ezt az utasítást a K karakter típusú tömbön a kiegészítő rutin(ok)-nak, amely(ek) eredményként az a) pontnak megfelelő szintaktikus leírást kell hogy adjon (adjanak) a T szöveg típusú tömbön. Ha a kiegészítő rutin(ok) sem tudja (tudják) értelmezni az utasítást, akkor hibajelzést ad a multiprocesszor.

SYNTAX P1==SYNTAX P2

P1,P2 - pozitív egész értékeket felvevő egész típusú változó (constans) lehet (aktuális) értékük két, a programban szereplő SYNTAX utasítás címkéjét kell hogy megadja.

Az utasítás hatására a P1 címkéjű SYNTAX utasítással és a P2 címkéjű SYNTAX utasítással leírt strukturájú utasítás (csoport)hoz ugyanaz a szemantikus leírás fog tartozni. (EXTEND utasítással előzőleg vagy csak a P1 címkéjű SYNTAX utasításhoz vagy csak a P2 címkéjű SYNTAX utasításhoz hozzá kellett rendelni szemantikust leírást. Ellenkező esetben - egyik utasításhoz sem, ill. mindkettőhöz történt hozzárendelés - az utasítás hatástalan és hibajelzés történik).

Az új utasítás szemantikus leírása a következőképpen történik:

>P> SEMANTICS BODY P1-P2. PARAMETERS P3-P4.
CONDITIONAL MODIFICATIONS P5-P6

P_i (i=1,2,...6) pozitív egész értékeket felvevő egész típusú változó (constans) lehet, (aktuális) értékük a programban szereplő címkéket kell hogy megadjon.

Az új utasítás hatására végrehajtandó programot ("szubrutint") a P1

cimkéjű utasítással kezdődő és a P2 címkejű utasítással végződő programrész adja meg. Ha az (új) utasítás jelentését leíró program nem CHANGE nyelvű, akkor a P1 címkejű utasítás EXTERNAL utasítás kell, hogy legyen, közvetlenül ez után következik a külső nyelvű program, majd a P4 címkejű END OF EXTERNAL utasítás jelzi a külső nyelvű program végét.

Ha a P1-P2 címkekkel határolt programrész CHANGE nyelvű, akkor az abban szereplő változók (tömbök) két csoportra oszthatók:

- a) szimbólikus változók ("formális paraméterek") amelyek helyére a végrehajtandó (új) utasítás tényleges változói ("aktuális paraméterei") kerülnek az alább ismertetett szabályok szerint. A szimbólikus változók típusdeklarációs utasításban nem szerepelhetnek.
- b) ideiglenes változók, ("lokális változók") amelyekre csak addig van szükség, amíg az (új) utasítás végre nem hajtódik. (Ezeknek szerepelniük kell a deklarációs utasításokban)

A P3 címkejű utasítással kezdődő és a P4 címkejű utasítással végződő programrész írja elő (a szintaktikus leírással együtt) a szimbólikus - tényleges változócsereét ("formális - aktuális paramétercsere")

Ebben a programrészben

V == t y p e n a m e (I) utasítások biztosítják a változócsereét.

V - szimbólikus változó

t y p e n a m e helyére tetszőleges adattípust megadó azonosító (INT, REAL, LOG, CHAR, TEXT vagy valamely típuskiterjesztő utasítás T2 -je) írható.

I - egész típusú változó (constans)

Az utasítás hatására a baloldalon szereplő szimbólikus változó helyére kerül a jobboldal által megadott tényleges változó.

A jobb oldalon álló adattípus azonosító az utána zárójelben álló sorszám (I) együtt a végrehajtandó (új) utasítás valamely változóját adja meg. A sorszám határozza meg, hogy a végrehajtandó utasításban balról-jobbra haladva az adattípus - azonosító által előírt típusú változók közül hányadikat kell figyelembe venni.

A P3-P4 programrészben szerepelhetnek egyéb utasítások is (pl. ciklusutasítás) de az előforduló változóknak (ebben a programrészben szereplő) deklarációs utasításokban szerepelniük kell.

Ezen kívül a vezérlés nem adható át a P3-P4 programrészen kívülre ennek

a programrésznek a végrehajtása során. (a programrész zárt).

A külső nyelvű programok által leírt szemantika esetén az alapnyelv nem írja elő a paraméterátadást illetve a módosítást szabályozó utasításokat.

A P5 utasítással kezdődő és a P6 utasítással befejeződő zárt programrész írja elő a P1-P2 illetve a P3-P4 programrész módosítását. Ez a módosítás függhet a végrehajtandó (új) utasítás

tényleges változóitól (pl. hogy vannak-e köztük azonosak, stb.)

környezetétől (milyen utasítások előtt ill. után áll, stb.)

a szintaktikus leírásban szereplő, ismétlődő részek tényleges számától vagy tetszőleges egyéb programozható feltételektől.

Ha nincsen módosítás, akkor a SEMANTICS utasításból elhagyható a P4 utáni vesszővel kezdődő rész.

RELEASE P

P pozitív egész értékeket felvevő egész típusú változó (constans) lehet, (aktuális) értéke valamely, a programban szereplő SYNTAX vagy PROGRAM FOR SYNTAX utasítás címkéjét kell hogy megadja.

Az utasítás hatására a végrehajtó processzor által használt könyvtárból (EXTERNAL,CHANGE) törlődik a P címkéjű utasítás által leírt szintaktikájú utasítás.

b.) Az adattípus - kiterjesztő utasítás

EXTEND TYPE T1, T2, I1, SYNTAX AT P1, INPUT AT P2, OUTPUT AT P3

T1, T2 szöveg típusú változó(constans)

I1, P1,2,3 egész típusú változó(constans)

T1 adja meg az új adattípushoz tartozó típust előíró új utasítás alapszavát. / Az új típust előíró utasítás, az alapnyelv típusdeklarációs utasításaihoz hasonlóan a típust előíró alapszóból, majd vesszőkkel elválasztott tetszőleges számú tömbdeklarátorból áll./

T2 írja elő, hogy az utasításkiterjesztő utasítás(ok)ban milyen névvel kell hivatkozni az új adattípusra.

I1 adja meg, hogy hány memóriaegységet kell biztosítani 1-1 új típusú adat számára. Ha ez az érték határozatlan (dinamikusan változhat) I1 értéke 0 kell hogy legyen.

P1 címkéjű SYNTAX utasítás az új adattípusnak megfelelő constans szintaktikus leírását adja meg. (Az új constans előfordulhat a programokban és az adatok között egyaránt.)

<P2> INPUT BODY P4 - P5. PARAMETERS P6 - P7

P4,5,6,7 - címke

A P4 - P5 zárt programrész írja elő, hogy a P1 címkéjű SYNTAX utasítás által előírt külső adatábrázolási módnak milyen belső (memóriabeli) adatábrázolás felel meg. A SEMANTICS utasítással azonos módon történik a szimbólikus-tényleges változócsere. (a P6 - P7 programrész írja elő.)

Ezen kívül kitüntetett szerepe van a .NEWTYPE. változótömbnek. Ez nem szerepelhet típusdeklarációs utasításban, a multiprocesszor (automatikusan) hozzárendeli a megfelelő (új) típust, tömbméret deklarációs utasításban azonban szerepelnie kell. A (teljes) .NEWTYPE. tömb kell, hogy felvegye az új adat (constans) belső ábrázolásának megfelelő értéket a P4 - P5 programrész végrehajtása után.

<P3> OUTPUT BODY P8 - P9. PARAMETERS P10 - P11

P8,9,10,11 - címke

A `.NEWTYPE.` teljes tömb (deklarálása azonos korlátozásokkal történik, mint az `INPUT` utasítás esetében) tartalmazza az új típusú adatot. A P8-P9 programrész írja elő a `.NEWTYPE.` tömbnek a P1 címkéjű `SYNTAX` utasítás szerinti kivitelét (kiírását). A szimbólikus változó-cserét `type name (I) = = V` utasítások biztosítják (a P10 - P11 zárt programrészben).

V - szimbólikus változó

`type name (I)` a `SEMANTICS` utasításnál ismertetett módon a `SYNTAX` utasításban szereplő valamely változót ad meg.

A P8 - P9 programrészben a szimbólikus változó(k) értéket kell, hogy kapjanak, és ez(ek) az érték(ek) kerül(nek) a multiprocesszor által automatikusan kiírásra.

Adatátviteli utasításokra nincs szükség sem a P4 - P5 (input body), sem a P8 - P9 (output body) programrészben.

Az `EXTEND TYPE` utasítás végrehajtása lehetővé teszi az új adattípust deklaráló utasítás használatát, a programokban az új adattípus változóként és konstansként való alkalmazását, valamint az új típusú adatok és konstansok adatátviteli műveletekben való használatát.

Az új adattípussal végzendő műveleteket utasítás-kiterjesztő `EXTEND` utasítás(ok) segítségével lehet leírni.

18. A nyomkövető (TRACE) utasítások.

A nyomkövető utasítások a nyelv szerves részét képezik a "mindenhez hozzáférés" elvének megfelelően. Fontos feladatuk, hogy a nyelv "veszélyes" elemeinek (különösen a programot módosító utasításoknak) a hatását ellensúlyozzák, biztosítsák, hogy a programozó ne veszítse el a futó program feletti ellenőrzést. A programok elkészítése (belövése) alatt nyújtott hathatós segítségen kívül a nyomkövető utasítások lényeges információkat szolgáltathatnak a kész (hibátlan) programokról is. (a memória kihasználtságáról, az egyes programágak használatának gyakoriságáról, stb.) Ezen kívül jól felhasználhatóak olyan célnyelvek definiálásánál, amelyeknél gyakoriak az "utasításközi műveletek", azaz természetes igény, hogy (bizonyos típusú) utasítások között (külön ki nem irt) standard rutinok kerüljenek végrehajtásra.

A nyomkövető utasítások eredménye folyamatosan kiiratható, vagy file-okon tárolható és külön utasításokkal iratható ki. Megszüntethető külön-külön az egyes nyomkövető utasítások hatása, megszüntethető (egy utasítással) az összes előzőleg végrehajtott nyomkövető utasítás hatása, törölhető a programból az összes nyomkövető utasítás.

I N D E X O V E R F L O W

A processzor külön utasítás nélkül nem tekinti hibának (nem figyeli) az indextúlsordulást (vagyis hogy valamely utasítás nem használ-e olyan tömb-elemet, amelynek indexe nagyobb a tömb aktuális méreténél). Az INDEX OVERFLOW utasítás hatására a processzor figyeli, hogy nem fordul-e elő indextúlsordulás, ha igen, akkor erről informatív hibajelzést ad. (a program fut tovább)

S T A C K I 1

I1 pozitív egész értékeket felvevő egész típusú változó (vagy constans).

Az utasítás hatására a multiprocesszor által végrehajtott utolsó I1 db utasítás eredményeivel együtt folyamatosan tárolásra kerül. Mindig az utoljára végrehajtott STACK I1 utasítás hatása érvényes, ha I1 (aktuális) értéke 0 vagy negatív, akkor a stack nem kerül tárolásra.

Ez az I1 utasítás és eredményeik ("a stack") kiiródnak, hiba miatti leállítás esetén automatikusan, vagy kiirathatók az alábbi utasításokkal.

W R I T E S T A C K

Az utasítás végrehajtásának hatására kiiródik a stack.

WRITE STACK m o d e

m o d e: AT P1 (amikor a vezérlés a P1 címkejű utasításra kerül,
kiíródik a stack)

EVERY I1 STEP (minden I1-edik végrehajtási lépésben kiíró-
dik a stack)

Többféle m o d e is érvényben lehet egyidejűleg.

LABEL TRACE

A multiprocesszor által végrehajtott utasítások közül a címkevel el-
látott utasításokat figyeli, hatására kiíródnak az érintett utasitá-
sok címkei. Ha csak egy programrészen kívánjuk elérni ezt a hatást,
akkor a

LABEL TRACE P1 - P2

utasítást kell alkalmazni. (hatása a P1 címkejű utasítással kezdődő
és a P2 címkejű utasítással végződő programrészre terjed ki)

GENERAL TRACE

Minden végrehajtott utasítás és eredménye kiírásra kerül. Ha csak egy
programrészen kívánjuk elérni ezt a hatást, akkor vagy a

TRACE P1 - P2

utasítást kell használni, (hatása a P1 címkejű utasítással kezdődő
és a P2 címkejű utasítással végződő programrészre terjed ki) vagy a

TRACE P1

utasítást kell használni (hatása a P1 címkejű utasításra terjed ki)

ARRAY T1 PROTECTED AGAINST WRITING

T1-szövegtípusú változó (vagy constans), valamely tömb nevét tartal-
mazza.

Ha a T1 aktuális értékének megfelelő tömb valamely elemébe beírási
kísérlet történik, hibajelzés kerül kiírásra (a beírás nem történik
meg, a program fut tovább)

Hasonlóan kiolvasás ellen is védhető valamely T1 tömb, az

ARRAY T1 PROTECTED AGAINST READING

utasítás alkalmazásával.

Programrészt tartalmazó memóriaterület is védhető irás/olvasás ellen a

PROGRAM P1-P2 PROTECTED AGAINST WRITING

PROGRAM P1-P2 PROTECTED AGAINST READING

utasítások alkalmazásával.

Védeni lehet egy tömb elemeit az előzetes értékadás nélkül történő felhasználástól az

ARRAY T1 PROTECTED AGAINST READING WITHOUT WRITING

utasítással.

Figyelhető egyes tömbök illetve programterületek aktuális értékének változása a

TRACE ARRAY T1 illetve a

TRACE PROGRAM P1-P2 utasítás alkalmazásával

T1-szöveg típusu változó(constans) P1,P2-címke. Kiírásra kerül a változást okozó utasítás és eredménye.(a program fut tovább)

Az UNUSED MEMORY ill. az UNUSED PROGRAM

utasítás hatására a multiprocesszor figyeli (tárolja) a felhasználásra nem kerülő tömbelemet illetve utasításokat és kiírja ezeket a

WRITE UNUSED MEMORY illetve a

WRITE UNUSED PROGRAM

utasítás hatására.

COUNTING

Az utasítás hatására a multiprocesszor számlálja az egyes tömbök és az egyes utasítások használatát ill.v végrehajtását.

MEMORY MAP

Az utasítás hatására kiírásra kerülnek a használatban levő tömbök ne-

vei, típusuk és aktuális méreteik (valamint használatuk gyakorisága, ha előzőleg COUNTING utasítás került végrehajtásra)

LIST

Az utasítás hatására kiírásra kerülnek a program aktuális utasításai, a bennük szereplő változók aktuális értékei (valamint az utasítások végrehajtási gyakorisága, ha előzőleg COUNTING utasítás került végrehajtásra)

LIBRARY LIST

Az utasítás hatására kiírásra kerülnek a végrehajtó processzorhoz tartozó könyvtár(ak) utasításai.

TRACE PROCESSZOR I1.P1 AT P2

I1 - egész típusu változó (constans)
P1, P2 - egész típusu változó (constans)

Az utasítás hatására a P2 címkéjű utasítás végrehajtásával egyidejűleg megkezdí működését az I1 sorszámú processzor a P1 címkéjű utasítás végrehajtásával. (I1 sorszámú működő processzor nem lehet a multiprocesszorban a P2 címkéjű utasítás végrehajtásakor) Az új, I1 sorszámú processzor alá van rendelve a TRACE PROCESSOR utasítást végrehajtó processzornak.

IF ERROR c o m a n d a c t i o n

Futási hibák (túlsordulás, a programban nem szereplő címke történő vezérlésátadás, stb.) esetén a multiprocesszor hibajelzést ad és folytatja működését. (a hibás művelet eredménye definiálatlan) Az eddig ismerttetett nyomonkövetési utasítások között szerepelt indextúlsordulás figyelését, memóriavédelem, stb. megvalósítását előíró utasítás. Ezek a nyomonkövetési utasítások is azt írják elő, hogy hibajelzést adjon a multiprocesszor, majd folytassa működését.

Az IF ERROR utasítás segítségével más típusu akciók is előírhatók futási hibák esetén. A c o m a n d rész írja elő, hogy mely utasításokra vonatkozik az IF ERROR utasítás, az a c t i o n rész pedig a végre-

hajtandó akciót írja elő.

c o m a n d : P1 - P2 (Az utasítás az egész programra, a P1 - P2
programrészre illetve a P1
P1 utasításra vonatkozik)

T

T szöveg típusú változótömb, amely valamely utasítás (csoport) szintaktikus leírását tartalmazza, a SYNTAX utasításnak megfelelő módon, azzal a különbséggel, hogy elegendő a szintaktikus leírás elejét írni le. Az IF ERROR utasítás hatása kiterjed valamennyi olyan utasításra, amelynek a szintaktikus leírása úgy kezdődik, mint a T - ben levő szintaktikus leírás.

a c t i o n STOP (A hibás utasítást végrehajtó processzor és az annak alárendelt processzorok befejezik működésüket, a hibás utasítás kiíródik).

TRACE PROCESSOR I1, P1, I2, I3

A hibás utasítás végrehajtását követő lépésben megkezdí működését az I1 sorszámú processzor a P1 címkejű utasítás végrehajtásával. I3 a hibás utasítás (belső) sorszáma, I2 az ezt végrehajtó processzor sorszáma. Az I1 sorszámú processzor mellérendeltségi viszonyba kerül az I2 sorszámú processzorral. (I1, P1, I2, I3 pozitív egész értékeket felvevő egész típusú változók (I1, P1 constans is lehet), I1 sorszámú működő processzor nem lehet a multiprocesszorban).

Ellentétes hatású IF ERROR utasítások közül mindig az utoljára végrehajtott utasításban előírt akció az érvényes.

A kapcsoló jellegű STACK, UNUSED MEMORY, UNUSED PROGRAM, COUNTING utasítások kivételével az összes eddig ismertetett nyomkövető utasítás elő (az utasításon belül) írható a

T O F : (F - file sorszámát megadó egész egész típusú változó vagy constans)

utasításrész, amelynek hatására az adott nyomkövető utasítás eredményei nem kerülnek kiírásra, hanem az F filera kerülnek.

A nyomkövetési eredmények kiirathatók a

WRITE TRACE FILE F m o d e utasítással.

Az F nyomkövetési információt tartalmazó file kiírásra kerül a m o d e rész által előírt időben. (a kiírás után a file tartalma törlődik).

m o d e az utasítás végrehajtásakor.

AT P1 a P1 címkéjű utasítás végrehajtásakor.

EVERY I1 STEP minden I1 - edik végrehajtás lépésben.

(I1 pozitív értékeket felvevő egész típusú változó (constans)).

A nyomkövetési információk file-ra küldése felhasználható a különböző típusú információk szétválasztására (külön-külön történő kiíratására) valamint a karakterformában tárolt információk programmal történő elemzésére.

(a kiíratás mellett vagy helyett).

NO TRACE

Valamennyi TRACE utasítás hatása megszűnik.

NO TRACE P1

A P1 címkéjű TRACE utasítás hatása megszűnik.

CLEAR TRACE

A programból törlődnek az összes nyomkövető utasítások.

19. Speciális utasítások

A speciális utasítások közé olyan utasítás-csoportok kerültek be, amelyek nem teljeselek (egyes utasításaik más csoportoknál vannak felsorolva, további fejlesztés alatt állnak), vagy amelyek több csoportba is bekerülhetnek volna (esetleg zavarták volna azoknak a csoportoknak az egységességét). Szerepel itt néhány olyan érdekes utasítás is, amely nem tartozik feltétlenül a nyelv egységes koncepciójának megfelelően a nyelvhez, de hasznos kiegészítő szerepe van (pl. a speciális szubrutin illetve corutinhívó utasítások).

- a. A programokhoz (szerkezetükhöz, belső ábrázolásukhoz) való teljesebb hozzáférést segítik elő a következő (a végrehajtási módot előíró, a szubrutinokból való visszatérést befolyásoló és a nyomkövető utasításokat kiegészítő) speciális utasítások:

I1 IS LABEL OF INSTRUCTION I2

I1, I2 - pozitív értékeket felvevő egész típusú változó
(konstans)

Az utasítás hatására az I1 változó értéke az I2 belső sorszámú utasítás címkéjének értéke lesz (ha az I2 utasítás nincs címkével ellátva, akkor I1 értéke 0 lesz, ha nincsen a programban I2 belső sorszámú utasítás, I1 értéke -1 lesz).

INSTRUCTION I1 HAS LABEL I2

I1, I2 - pozitív értékeket felvevő egész típusú változó
(konstans)

Az I1 változó értéke az I2 címkéjű utasítás belső sorszáma lesz (ha

nincsen I2 címkejű utasítás a programban, akkor I1 értéke 0 lesz).

LOCATION OF ARRAY T1 TO I1

T1 - szöveg típusú változó (konstans)

I1 - egész típusú változó (konstans)

A multiprocesszorban az aktuális tömbök a végrehajtó program által kezelt .MEMORY. tömbön helyezkednek el. A .MEMORY. tömb memórielemenként címezhető (indexelhető). A .MEMORY. tömb (elemei) csak a

.MEMORY.(I1) = .MEMORY.(I2)

(I1, I2 - egész típusú változó (konstans))

speciális értékadó utasításban szerepelhetnek.

A LOCATION OF... utasítás hatására az I1 változó értéke a T1 (aktuális értéke) által adott változótömbnek a .MEMORY. tömbön belüli kezdő címét (indexét) adja meg.

DESCRIPTION OF INSTRUCTION I1 TO I

I1 - egész típusú változó (konstans)

I - egész típusú változótömb

Az I1 (belső) sorszámú utasítás leírása az I tömbre kerül a következő formában: I(1) tartalmazza az utasítás kódját (belső sorszámát), I(2) tartalmazza az utasítás paramétereinek (általános változóinak) számát (n), az I(3), I(4), ..., I(m) tömbelemek tartalmazzák az utasítás paramétereinek a leírását. A paraméterek leírása az utasításban való előfordulásuk (balról-jobbra haladva) sorrendjében kerül felsorolásra.

Mivel egy paramétert több cím is leírhat, az egyes paraméterek leírását egy-egy 0 értékű tömbelem választja el.

Egy paraméter leírása a paraméterben szereplő tömbök kezdőcímeiből illetve a konstansok címeiből áll, a paraméterben való előfordulásuk (balról-jobbra haladva) sorrendjében.

Egy tömb kezdőcímét első elemének a .MEMORY. tömbön belüli sorszámából egyet levonva kaphatjuk meg.

Egy konstans címe a .MEMORY. tömbön belüli indexe.

A paraméteren belül szereplő kiegészítő leírásában előforduló címek negatív előjelet kapnak.

CODE OF INSTRUCTION P1 TO I1

P1,I1 - egész típusú változó (konstans)

P1 - valamely SYNTAX utasítás címkéje

Az utasítás hatására az I1 változó értéke a P1 címkéjű SYNTAX utasítás által leírt utasítás(csoport) kódja (belső sorszáma) lesz.

SUBPROCESSORS OF I1 TO I

I1 - egész típusú változó (konstans)

I - egész típusú változó(tömb)

Az utasítás hatására az I tömb első elemének értéke az I1 sorszámú processzornak alárendelt processzorok (darab) száma (n) lesz, az I(2),I(3), ...,I(n+1) elemek értéke az alárendelt processzorok sorszáma lesz. A felfüggesztett működésű processzorok sorszáma negatív előjellel van el látva.

COPROCESSORS OF I1 TO I

A SUBPROCESSORS utasítással azonos módon az I tömb fogja tartalmazni az I1 sorszámú processzorral mellérendelt viszonyban levő processzorok (darab)számát és sorszámait az utasítás végrehajtásának hatására.

PROCESSOR LIST TO I

A SUBPROCESSOR utasítással azonos módon az I tömb fogja tartalmazni az összes processzorok (darab)számát és sorszámait.

NUMBER OF PROCESSOR TO I

I - egész típusú változó

Az utasítás végrehajtásának hatására I értéke a végrehajtó processzor sorszáma lesz.

UTSZ OF PROCESSOR I1 TO I2

I1 - egész típusú változó (konstans)

I2 - egész típusú változó

Az I2 változó értéke az I1 sorszámú processzor utasításszámlálójának értéke lesz, ha az I1 sorszámú processzor alá van rendelve az utasítást végrehajtó processzornak (ha ez nem áll fenn, akkor az értékadás nem történik meg, és hibajelzés kerül kiírásra).

UTSZM OF PROCESSOR I1 TO I2

Az utasítás hatása megegyezik az előző utasítás hatásával, az egyetlen különbség az, hogy az utasításszámláló-módosító értéke kerül átadásra.

Az előző két utasításban szereplő feltételek mellett lehetséges a fordított értékadás is, az

I2 TO UTSZ OF PROCESSOR I1 és az
I2 TO UTSZM OF PROCESSOR I1

utasítások alkalmazásával.

b. Szöveg és karakter típusú változókat kezelő utasítások

CLEAR T

A T szöveg típusú változótömb elemeinek értéke definiálatlanná válik az utasítás hatására, a tömb első szabad eleme T(1) az utasítás végrehajtása után. (Egydimenziósnak tekintve a tömböt.)

T1 TO T

A T szöveg típusú változótömb első szabad eleme felveszi a T1 szöveg típusú változó értékét, és foglalt elemmé válik. (A T tömb első szabad eleme ezután az 1-gyel nagyobb indexű elem, egydimenziósnak tekintve a tömböt.)

T1 IS I1 IN T

Az I1 egész típusú változó felveszi a T1 szöveg típusú változó sorszámát a T szöveg típusú tömbben. (T1 sorszáma T-ben azt adja meg, hogy T1 értéke T hányadik elemének értékével egyezik meg, T-t egydimenziósnak tekintve.) Ha nincs a T tömbben olyan elem, amelynek értéke megegyezne T1 értékével, akkor I1 értéke 0 lesz, ha több ilyen elem is van, akkor a legkisebb sorszám lesz I1 értéke.

NUMBER OF CHAR IN T1 IS I1

Az I1 egész típusú változó értéke a T1 szöveg típusú változó karaktereinek száma lesz az utasítás végrehajtása után.

C(I1)-C(I2) TO T

Az I2 index (aktuális) értéke nagyobb vagy egyenlő kell hogy legyen,

mint az I1 index (aktuális) értéke. A T szöveg típusú változó tömb első szabad eleme felveszi a C(I1), C(I1+1), ..., C(I2) karakterekből álló szövegértéket (és foglalt elemmé válik).

T1 TO C(I1)-C(I2)

Az I2-I1+1 kifejezés aktuális értéke nagyobb vagy egyenlő kell hogy legyen, mint a T1 szöveg típusú változó (szöveg) értékében szereplő karakterek száma. Az utasítás hatására C(I1) értéke T1 első karaktere, C(I1+1) értéke T1 második karaktere, ..., C(I1+N-1) értéke T1 N-dik karaktere (ha T1 N darab karakterből áll).

c. A multiprocesszor on-line felhasználását elősegítő utasítások:

EXECUTE

Az utasítás végrehajtása után a multiprocesszorba (folyamatosan) beérkező utasítások azonnal végrehajtásra kerülnek, és nem őrződnek meg (újra végrehajtani azokat vagy hivatkozni rájuk nem lehet a továbbiakban).

SAVE

Az utasítás végrehajtása után a multiprocesszorba beérkező utasítások tárolásra kerülnek (folyamatosan, +1-gyel növekvő belső sorszámmal). A multiprocesszor működése a beérkező utasításoktól függetlenül folytatódik.

SAVE AND EXECUTE

Az utasítás végrehajtása után a multiprocesszorba beérkező utasítások tárolásra kerülnek (a SAVE utasítás által előírt módon), és azonnal végre is hajtódnak (az érvényben levő végrehajtási mód szerint).

DELETE specification

Az utasítás végrehajtása után törlődnek a programból a specifikációs rész által előírt utasítások.

s p e c i f i c a t i o n I1 az I1 belső sorszámu utasítás

I1-I2 az I1 belső sorszámu utasitással kezdődő, és az I2 belső sorszámu utasítással végződő programrész

LABELED P1	a P1 címkéjű utasítás
LABELED P1-P2	a P1-P2 programrész
SYNTAX AT P1	a P1 címkéjű SYNTAX utasítás által meghatározott utasítás(csoport)

INSERT s p e c i f i c a t i o n END AT P1

A programba bekerülnek a specifikációs részben megadott utasítás után (közvetlenül) az INSERT utasítást követően a multiprocesszorba beérkező utasítások, a P1 címkéjű utasítással bezárólag. A specifikációs rész megegyezik a DELETE utasításnál felsoroltakkal, csak a SYNTAX AT P1 forma nem megengedett.

- d. Teljes programoknak a háttérmemóriában való tárolását és onnan történő aktivizálását szolgálják a következő utasítások:

PROGRAM TO FILE F

A program aktuális utasításai és a végrehajtási mód jellemzői az F file-ra kerülnek, a program folytatja futását.

PROGRAM AND DATA TO FILE F

A program aktuális utasításai, a végrehajtási mód jellemzői és a változóterület az F file-ra kerül, a program folytatja futását.

PROGRAM FROM FILE F

Az F file-on tárolt program és változóterülete, ha az is az F file-on van, bekerül a memóriába, és folytatja futását a (szintén az F file-on tárolt) végrehajtási mód szerint. Az utasítást végrehajtó program és változóterülete törlődik a memóriából.

Tömbök illetve a program szegmentálását szolgálja a következő két utasítás:

ARRAY T1 TO FILE, BUFFER I1,I2

T1 - szöveg típusú változó (konstans)

aktuális értékét T-vel jelöljük

I1,I2 - egész típusú változó (konstans)

Az utasítás végrehajtása során a T tömb átkerül a multiprocesszor által kezelt (háttérmemória) file-ok valamelyikére, a fő memóriában elfoglalt helye felszabadul.

A program további futása során változatlanul lehet hivatkozni a T tömb elemeire. A kívánt tömbelemeknek a fő memóriába juttatását egy $I1+I2+1$ tömbelem méretű puffer gyorsítja meg. A puffer tartalma az utasítás végrehajtásakor a tömb első $I1+I2+1$ eleme lesz (egy-indexesnek tekintve a tömböt). A továbbiakban a puffer tartalma mindig addig marad változatlan, amíg a kívánt tömbelemek megtalálhatók benne. Ha a keresett K indexű tömbelem nem található a pufferban, akkor a $T(K-I1), T(K-I1+1), \dots, T(K), \dots, T(K+I2-1), T(K+I2)$ tömbelemek kerülnek a pufferba (egy-indexesnek tekintve a tömböt).

PROGRAM P1-P2 TO FILE, BUFFER I1,I2

Az előző utasításhoz hasonlóan, a P1-P2 programrész átkerül a háttérmemóriába, a fő memóriában elfoglalt helye felszabadul, a továbbiakban is rákerülhet a vezérlés, illetve utasításaira lehet hivatkozni.

Aktivizálását egy $I1+I2+1$ utasítás méretű puffer gyorsítja meg, amelynek tartalma az utasítás végrehajtásakor a P1 című utasítás és az utána következő $I1+I2$ darab utasítás. A továbbiakban a puffer tartalma mindaddig változatlan marad, amíg a keresett utasítások megtalálhatók benne. Ha a K belső sorszámú utasítás nem található a pufferban, akkor a $K-I1, K-I1+1, \dots, K, \dots, K+I2-1, K+I2$ belső sorszámú utasítások kerülnek a pufferba.

- e. A file-ok (rá)írással és (ki)olvasással szembeni védelmét a nyomkövető utasításoknál ismertetett, tömbök védelmét szolgáló utasításokkal azonos módon biztosítja a

FILE F1 PROTECTED AGAINST WRITING, a

FILE F1 PROTECTED AGAINST READING és a

FILE F1 PROTECTED AGAINST READING WITHOUT WRITING

utasítás, az egyetlen különbség az, hogy itt a file sorszámát az F1

egész típusú változó (konstans) adja meg. (A tömbökre vonatkozó utasításban a T1 szöveg típusú változó adta meg a megfelelő tömb nevét.) A védelem megszüntethető a NO TRACE P1 utasítás alkalmazásával.

Állandó file-ok használatát biztosítja a következő három utasítás:

T IS PERMANENT FILE l e n g t h

T - szöveg típusú változó (konstans)

l e n g t h - üres (változó hosszúság)

- LENGTH I1 egész típusú változó
(konstans)

Az utasítás végrehajtása után a háttérmemóriában rendelkezésre áll a T nevű I1 memóriaelem (vagy tetszőleges, azaz dinamikusan változtatható) méretű állandó file.

FILE F IS FILE T

F - egész típusú változó (konstans)

T - szöveg típusú változó (konstans)

Az előzőleg állandó file-nak deklarált T file-ra F file azonosítóval lehet hivatkozni az utasítás végrehajtása után.

CLEAR FILE T

T - szöveg típusú változó (konstans)

Törlődik a T file az utasítás hatására.

f. A CHANGE nyelv alap utasításkészlete állandó utasításokból áll. Egyszer végrehajtandó utasítások a következőképpen keletkezhetnek:

a. Új utasítások definiálásakor az EXTEND utasítás A4 paraméterének értéke NONPERMANENT.

b. A NONPERMANENT s p e c i f i c a t i o n utasítás segítségével

s p e c i f i c a t i o n P1-P2

P1

SYNTAX AT P1

A P1-P2 programrész, a P1 utasítás, illetve a P1 címkéjű SYNTAX utasítás által leírt (minden) utasítás egyszer végrehajtandó utasítás lesz.

A PERMANENT s p e c i f i c a t i o n utasítás hatására állandó utasítás keletkezik (a specifikációs rész ugyanaz lehet, mint a NONPERMANENT utasítás esetében).

- g. A szubrutin- illetve corutinhívás speciális változatai nemcsak EXIT (WAIT) utasítás végrehajtása esetén teszik lehetővé a rutinból való visszatérést, hanem bizonyos számú utasítás végrehajtása is lehet a visszatérés feltétele.

SUBR P1 BACK AFTER I1 STEPS

P1,I1 - egész típusú változó (konstans)

A vezérlés a P1 címkéjű utasításra kerül, I1 végrehajtási lépés letelte után a vezérlés a SUBR P1 BACK... utasítás után (a végrehajtási mód szerint) következő utasításra kerül (ha addig EXIT, STOP, stb. utasítás nem került végrehajtásra).

COPROCESSOR V1, V2 DURING I1 STEPS

SUBPROCESSOR V1, V2 DURING I1 STEPS

V1,V2,I1 - egész típusú változó (konstans)

A V1 sorszámú mellé- (alá-) rendelt processzor elkezdí működését a következő végrehajtási lépésben a V2 címkéjű utasítás végrehajtásával, I1 végrehajtási lépés letelte után a V1 sorszámú processzor felfüggesztett állapotba kerül (ha addig STOP, WAIT stb. utasítást nem hajt végre).

NOWAIT V1 DURING I1 STEPS

V1,I1 - egész típusú változó (konstans)

A V1 sorszámú processzor folytatja működését, legkésőbb I1 végrehajtási lépés múlva (újra) felfüggesztett állapotba kerül.

GO TO I1,I2,I3,...,I(N)

I1,I2,...,I(N) - egész típusú változó

Az utasítás hatására N darab új, a végrehajtó processzorhoz képest alárendelt processzor kezd működni a következő végrehajtási lépés során az I1,I2,...,I(N) címkejű utasításokat hajtva végre. Az új processzorokhoz automatikusan sorszámot rendel hozzá a multiprocesszor, és az I1,I2,...,I(N) változók értéke a megfelelő új processzor sorszáma lesz. A sorszámok mindig a multiprocesszorban rendelkezésre álló legkisebb értékű sorszámok lesznek.

Az utasítás GO TO ARRAY I formában is írható, ekkor a teljes I tömb játssza I1,I2,...,I(N) szerepét.

COMMENT T

A COMMENT alapszó után az utasításon belül tetszőleges szöveg vagy szöveg típusú változó írható. Az utasítás hatása megegyezik a NULL utasítás hatásával.

h. Az EXTEND utasításoknál leírt módon történő könyvtárszervezés és kezelés merev, egyes bonyolultabb feladatok megoldására igen nehézkesen használható fel. A következő utasításcsoport a különböző könyvtárak dinamikus (futás alatti) kezelését biztosítja, és lehetővé teszi több (az egyes processzorokhoz hozzárendelt) CHANGE nyelvű könyvtár egyidejű használatát.

A multiprocesszorban mindig egy (külső nyelvű) alapkönyvtár van (amely minden processzorra érvényes), az egyes processzorokhoz tartozhat (nem feltétlenül tartozik) egy-egy CHANGE nyelvű könyvtár. Több processzorhoz is tartozhat ugyanaz a CHANGE könyvtár.

Az alapkönyvtárban az alapnyelv adattípusainak és utasításainak szintaktikus és szemantikus leírása található. A multiprocesszorhoz tartozó alapkönyvtár nem módosítható és nem bővíthető. A multiprocesszor működésének a kezdetekor az alapkönyvtár egy másolata áll rendelkezésre, ez módosítható, bővíthető, tetszőleges file-ra vihető. Egy file-ra vitt (tetszőlegesen módosított) alapkönyvtár behívható a multiprocesszorba az éppen bent lévő alapkönyvtár helyett. Egy alapkönyvtárban minden adattípusnak és utasításnak egyértelmű sorszáma van.

Az EXTEND utasítások előírhatják CHANGE könyvtár kibővítését (létrehozását). Ezeknek az utasításoknak a hatására az EXTEND utasítást végrehajtó processzorhoz tartozó CHANGE nyelvű könyvtárba bekerül az új adattípus illetve utasítás szintaktikus és szemantikus leírása, és ezenkívül az alapkönyvtárba is bekerül a szintaktikus leírás, és ott egyértelmű sorszámot kap. (Ez a sorszám tárolásra kerül a CHANGE könyvtárban is.)

Egy CHANGE könyvtár file-ra vihető, file-ról behívható valamely processzorba. CHANGE könyvtár létesítésekor egy üres (egész típusú C-

-kat tartalmazó) file-ről kell behívni egy "CHANGE könyvtárat", az ennek a könyvtárnak a "kibővítését" előíró első EXTEND utasítás hatására a könyvtár inicialása is megtörténik.

Az adattípusok és utasítások sorszámozását az teszi szükségessé, hogy egy program futása előtt általában nem lehet tudni, hogy az egyes utasításokat mely processzorok fogják végrehajtani. A programok egyértelmű lefordításához (olyan egyértelmű belső ábrázolásához), amely lehetővé teszi, hogy bármely processzor végrehajthasson (egyértelműen) bármely utasítást, az alap könyvtárban össze kell gyűjteni a multiprocesszorban érvényben levő összes utasításnak legalábbis a szintaktikus leírását.

A programozó különböző futások alkalmával más-más CHANGE és alapkönyvtárakat hozhat létre, azokat file-okon tárolhatja, és azután bármikor aktivizálhatja.

A multiprocesszor ellenőrzi az egyes könyvtárak aktivizálásakor, hogy a CHANGE könyvtárakban előforduló szintaktikus leírások ugyanazzal a sorszámmal szerepelnek-e az alapkönyvtárban (ha előfordulnak az alapkönyvtárban). Hiba esetén a program futása leáll.

Új alapkönyvtár aktivizálásakor az egész program újra fordítása kerül végrehajtásra.

Ez az utasításcsoport lehetővé teszi különböző számítógépek egy rendszeren belül való együttműködésének modellezését.

Egy másik fontos alkalmazás az utasítások kiterjesztésének a lehetősége régi szintaktika teljes egészében történő megőrzésével. Az utasítás jelentése más és más lehet aszerint, hogy melyik processzor hajtja végre.

Ezek az utasítások kísérleti jellegűek, ha alkalmazásuk hasznosnak bizonyul, akkor a könyvtárakra vonatkozó szabályok a multiprocesszor általános leírásánál illetve az EXTEND utasításoknál kerülnek ismertetésre.

EXTERNAL LIBRARY TO FILE F

F - egész típusú változó (konstans)

A külső nyelvű (alap)könyvtár az F file-ra kerül.

(A multiprocesszorban is változatlanul megmarad.)

EXTERNAL LIBRARY FROM FILE F

Az F file-on levő alapkönyvtár lesz a multiprocesszor alapkönyvtára az utasítás végrehajtásának hatására. (A multiprocesszorban levő alapkönyvtár törlődik.)

LIBRARY TO FILE F

F - egész típusú változó (konstans)

Az utasítást végrehajtó processzor CHANGE nyelvű könyvtára az F file-ra kerül.

LIBRARY FROM FILE F

F - egész típusú változó (konstans)

Az utasítást végrehajtó processzor CHANGE nyelvű könyvtára az F file-on tárolt CHANGE könyvtár lesz. (Esetleges előző CHANGE könyvtára törlődik.)

LIBRARY FROM FILE F TO PROCESSOR I

I,F - egész típusú változó (konstans)

Az utasítást végrehajtó processzor számára az I sorszámú processzor alárendelt kell hogy legyen. Az utasítás hatására az I sorszámú processzor CHANGE nyelvű könyvtára az F file-on tárolt CHANGE könyvtár lesz.

(Az I sorszámú processzor esetleges előző CHANGE könyvtára törlődik.)

1. TRANSLATE P1-P2 TO F

P1,P2,F - egész típusú változó (konstans)

Az utasítás hatására a P1-P2 programrészt a multiprocesszor lefordítja az EXTERNAL könyvtárnak megfelelő külső nyelvre, és az így kapott program az F file-ra kerül.

A lefordított program általában lényegesen gyorsabban futtatható. (A gyorsabb futást az teszi lehetővé, ha a multiprocesszort realizáló interpretatív végrehajtó program részben vagy teljes egészében feleslegessé válik, ami akkor áll fenn, ha önmódosító, multiprocesszoros és nyomkövető utasítások nem szerepelnek a P1-P2 programrészben. Mivel ez igen erős megszorítás, az utasítás főleg a CHANGE programmal generált programok lefordítására használható előnyösen.)

E. Mit és hogyan biztosít a nyelv felépítése

1. a. A COPROCESSOR és a SUBPROCESSOR utasítás, illetve az UTSZM utasítás külön-külön is lehetővé teszi nem lineáris programok írását.

b. EXTEND utasítás(ok)ban használva a COPROCESSOR, a SUBPROCESSOR és az UTSZM utasítást, a kiterjesztett multiprocesszor tetszőlegesen nem lineáris módon hajthat végre programokat. Az egyes speciális problémaszintályok számára kedvezőbb lehet egy speciális nem-lineáris végrehajtási mód, mint az univerzális lineáris mód.
2. A CHANGE utasítások biztosítják a programok dinamikus (futás közben történő) programozott módosítását, lehetővé teszik CHANGE nyelvű programok futás közbeni generálását (adaptív programok, fordítóprogramok, stb.).
3. Az EXTEND utasítások lehetővé teszik új adattípusok és utasítások definiálását, és a nyelvbe való automatikus beépítését.
4. A programozó igen szabad kezet kapott a fordítás és a futás vezérlésére. A "mindenhez való hozzáférés" elvének realizálására rendelkezésre állnak a végrehajtási módot előíró (UTSZM és párhuzamos feldolgozást vezérlő) utasítások, a kiterjesztő (EXTEND) utasítások, a programok önmódosítását lehetővé tevő CHANGE utasítások (ezek használata különösen a változódefiníció által megengedett egyszerű listák segítségével lehet előnyös), a szubrutinmélységet szabályozó ISD utasítások, valamint a nyomkövető (TRACE) utasítások (amelyekre a "mindenhez való hozzáférés" elvének megfelelően közvetlenül is szükség van, de az elv realizálásához kiválasztott, az előzőekben felsorolt eszközök feletti ellenőrzés is szükségessé teszi a nyelvbe való beépítésüket).

5. A COPROCESSOR, a SUBPROCESSOR, a COWAIT és a NOWAIT utasítások segítségével multiprocesszoros programok írhatók és futtathatók.

F. A processzor realizálása

1. Szűk subset interpretatív fordítóprogramja a MINSZK-22 gépen [5].
2. USASI FORTRAN IV alapú processzor a CDC-3300-as gépen.
 - a. A processzor interpretatív módon működik.
 - b. Előfordító program (a program külső formájáról belső listastruktúrában történő ábrázolásra való áttérés).
 - c. Értelmező és végrehajtó program
(A belső listastruktúrában ábrázolt program utasításainak értelmezését és végrehajtását biztosítja.)
 - d. Az előfordítás alatt a szintaktika analízátor részekre bontja a lefordítandó utasítást, az egyes részeket külön-külön elemzi, és lefordítja egy közbelső nyelvre (így tulajdonképpen az egész utasítást lefordítja erre a közbelső nyelvre).
 - e. A szintaktika analízátor az utasítás közbelső nyelvű ábrázolása alapján meghatározza az utasítás kódját (vagy esetleg hibajelzést ad).
 - f. A szintetizáló program beépíti a belső listastruktúrába az utasítás kódját és az utasítás paramétereit leíró címeket (egy-egy paramétert általában több cím ír le).
 - g. Az értelmező és végrehajtó program belső hierarchiájában az első helyen a multiprocesszor program áll, amely az egyes processzorok indítását, felfüggesztését, megszüntetését és futás alatti vezérlését irányítja.

- h. Egy processzor működését vezérli az EXEC 1 program (az utasításszámláló tartalma által meghatározott belső sorszámú utasítást értelmezi és végrehajtja, az utasításszámláló-módosító tartalmát hozzáadja az utasításszámláléhoz a következő végrehajtandó utasítás sorszámának meghatározására - ha ez szükséges -, valamint végrehajtja az előírt nyomkövetési akciókat).
- i. Egy utasítás végrehajtása során először a CIMSZÁMITÓ program meghatározza az utasítás paramétereit leíró címeknek megfelelő aktuális címeket, majd az utasításkód által meghatározott könyvtári szubrutin kerül végrehajtásra (ez a szubrutin lehet a FORTTRAN-könyvtárban, vagy a CHANGE-könyvtárban).

3. A CDC 3300-as processzorban alkalmazott eljárásokról

- a. Egyetlen közös szimbólumtábla-kezelő rutin kerül felhasználásra a szöveg típusú változók kezelésére, az előfordításra, a közbenső nyelvre fordított utasítások kódjának felismerésére, az EXTEND utasítások végrehajtására. (A gazdaságosság illetve áttekinthetőség mellett a processzor esetleges bootstrappelt felépítésére is lehetőséget nyújt ez a megoldás.)
- b. A belső listastruktúra rendkívül tömör
Ha egy utasítás vagy egy paraméter többször is ismétlődik a programban, leírása csak egyszer kerül tárolásra. Ez a megoldás igen hajlékony, a programot módosító CHANGE utasítások végrehajtását megkönnyíti, és általában memórianyereséget tesz lehetővé, mivel az utasítások, illetve a paraméterek leírása a rájuk való hivatkozásoknál átlagosan 5-10-szer több helyet igényel. Ez az ábrázolási mód megkönnyítheti az automatikus szubrutinkészítést.

4. CHANGE processzor kissetőítőgépekhez

- a. Kissetőítőgép esetén nem jelent különösebb hátrányt az interpretatív végrehajtás (abban az értelemben, hogy a közepes vagy magasszintű nyelvű programok végrehajtása általában interpretatív módon történik a szűk utasításkészlet és a viszonylag kis memóriaméret miatt).
- b. A lefordított CHANGE program igen tömör ábrázolása viszonylag nagyobb méretű programok futtatását teszi lehetővé.
- c. Ha a kissetőítőgép szatellitként (is) működik, akkor a CHANGE processzor feladatai előnyösen feloszthatók a központi számítőgép és a szatellit között.

On-line használat esetén a munkamegosztás a következő lehet:

A kisgép egyszerű előfordító-programja a szintaktikusan hibás utasításokat visszautasítja, a szintaktikusan helyes utasításokat tömör előfordított formában továbbítja a nagygép felé, ahol a fordítás folytatódik, illetve a fordítás befejezése után megtörténik a program végrehajtása.

Kész programok használata esetén célszerűnek látszik, hogy a nagygépen egy optimalizáló fordítóprogram készítsen el egy viszonylag kis helyigényű lefordított programot a kissetőítőgépen való futtatáshoz.

- d. A címszámító és az EXEC 1 rutin fixmemóriában (kevesebb mint 500 byte) való elhelyezésével egy nagyságrenddel javítható a futási sebesség.

5. A 10010 számítógép CHANGE végrehajtó programja

- a. Az előző (4.) pont a.,b.,c. részében felsorolt tények a 10010 esetén is fennállnak.
- b. Rendelkezésre állnak az ALGOL fordítóprogram interpretatív végrehajtáshoz készített rutinjai, amelyek szervezése olyan, hogy viszonylag egyszerűen beépíthetők a CHANGE végrehajtó program könyvtár részébe.
- c. Így reális célnak látszik első lépésként végrehajtó program elkészítése. A fordítás a CDC 3300-on történhet.

G. A CHANGE nyelv általános jellegű alkalmazása

1. A nyelv kiterjeszhetősége lehetőséget nyújt arra, hogy nem túl bonyolult nyelveket (és egyben processzorait is) egyszerűen lehessen definiálni. (Nincs szükség az egyes nyelvekhez szintaktika analízátorok írására, a programok belövéséhez a nyomkövető utasítások segítséget nyújtanak.) A definiálni kívánt nyelvek "erősen célorientáltak" lehetnek, azaz felépítésük igen közel maradhat a nyelv felhasználási területén szokásos írásmódhoz, mivel a CHANGE processzor végrehajtási módja előírható, és a programok futás alatt módosíthatóak.

2. Diszkrét rendszerek szimulálása

A rendszer változó tartománnyal ábrázolt alapelemei a típuskiterjesztő utasításokban jellemezhetőek, a rendszer programmal leírt elemeit pedig a kiterjesztő utasítások segítségével egy-egy utasítás ábrázolhatja. Ez az ábrázolás több szinten is történhet, ahogy a rendszer egyre magasabb szintű blokkjai felépülnek.

A blokkok közti kapcsolatot a változó tartományon keresztül, esetleg programok segítségével lehet megvalósítani.

A rendszer elemeinek egyidejű működése a párhuzamos műveletvégrehajtást előíró utasítások segítségével történhet.

Például: A rendszer legmagasabb szintű blokkjait egy-egy kiterjesztett utasítás képviseli. A blokkok közti kapcsolat a változó tartományon keresztül történik, a blokkok működési sebessége megegyezik. A blokkok működése egyszerre kell, hogy elkezdődjön. Egy külső (fő) program hatására fejezhetik be működésüket.

Az így megadott feladat realizálható pl. a következőképpen:

A blokkokat képviselő utasításokat címkével látjuk el, és egymás után írjuk azokat. A főprogram gondoskodik arról, hogy az egyes utasításokat ugyanabban az időpillanatban különböző processzorok kezdjék végrehajtani

úgy, hogy utasításslámláló-módosítójuk tartalma 0 legyen, azaz mindig a blokkot képviselő utasítást hajtsák végre (ez az utasítás rendszerint nem vezérlésátadó utasítás). Mivel egyszerre tulajdonképpen csak egy processzor működése indítható el, a tényleges egyszerre való indítás a már elindított processzorok késleltetésével, pl. NULL utasítások végrehajtásával történhet. Az a fontos, hogy a blokkokat realizáló utasításokra való rá lépés történjen egyszerre. (A rá lépés előtti utasítás UTSZM=0 kell hogy legyen.)

A főprogram gondoskodhat az egyes processzorok felfüggesztéséről, illetve a szimuláció befejezéséről.

A szimuláció "idejének mérésére" szolgálhat egy további processzor által állandóan (UTSZM=0) végrehajtott $IDO=IDO+1$ utasítás is. Az IDO változó értékét figyelheti a főprogram, vagy egy további processzor által állandóan végrehajtott feltételes vezérlésátadó utasítás:

(P2) (IF(IDO.GT.MAXIDO) P1,P2), amely az "idő túllépése", azaz a feltétel bekövetkezése esetén elindítja valamely program működését (amelynek első utasítása valószínűleg megváltoztatja UTSZM értékét, a továbbiakban pedig befolyásolja a főprogram működését).

3. Adaptív programok (programot készítő ("író") adaptív programok)

Egy adott (az adaptív programban leírt) alap-utasításkészletből a programot módosító utasítások segítségével programok generálhatóak és futtathatóak a főprogram vezérlése alatt. Ez a vezérlés különböző formákban történhet, pl. a főprogram átadja a vezérlést a generált programnak, és az bizonyos idő múlva (vagy időnként) visszaadja a vezérlést a főprogramnak (esetleg az előző pont végén leírt működésű időmérést és megszakítást szimuláló, két párhuzamosan futó processzor segítségével). A főprogram folytathatja futását párhuzamosan a generált programmal, mintegy működés közben figyelve azt. A figyelés eszköze, akár folyamatosan, akár szakaszosan történjen, valamely ellenőrző programrész lehet,

amely vizsgálja, hogy a generált program megfelelően működik-e. Ez az ellenőrző blokk az esetek egy részében hasonló lehet a szimulációs programokhoz, illetve egy megfelelően paraméterezett általános szimulációs program lehet.

Az adaptív program feladata, hogy az ellenőrző programrész által adott eredmény alapján módosítsa a generált programot, amíg csak szükséges.

4. Makroprocesszorok, listakezelő és szimbólummanipulációs nyelvek

A szövegkezelő (tulajdonképpen szimbólumtábla-kezelő) utasítások és a kiterjesztő utasítások természetes eszközöket adnak makroprocesszorok írására. Az általános változódefiníció és a kiterjeszthetőség segítségével a listakezelő utasítások, a karakter és szöveg típusú változók segítségével a szimbólumkezelő utasítások közelíthetők meg.

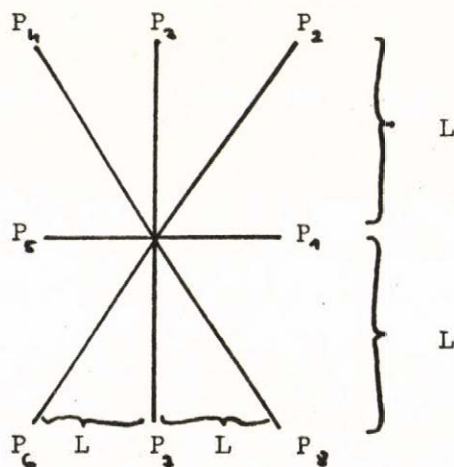
H. A CHANGE nyelv konkrét alkalmazásai

1. NC nyelvek (geometriai/technológiai leíró nyelvek)

a. Az NC nyelvekre általában jellemző a G.1. pontbeli általános leírás.

b. ADMAP postprocesszor (optimalizálással) a CII 1001C számítógépre

Az ADMAP PP feladatát, funkcionális felépítését [6] írja le részletesen. A PP feladata, az ADMAP nyomtatott áramköri lapokat készítő berendezés vezérlő lyukszalagjának elkészítése. A berendezést működtető egyes utasításoknak 1-1 karakter felel meg a lyukszalagon. Az utasításrendszer egyszerű, a rajz (fűrő stb.) fejet mozgó utasításokból, toll-fel, toll-le, nagylépésváltás, kislépésváltás és fűrész utasításokból áll. Mivel a berendezés rajz (fűrő stb.) fejét két léptetőmotor hajtja, és ezek egyszerre is működtethetők, a fej egy adott P helyzetből az U_1, U_2, \dots, U_7 utasítások hatására a P_1, P_2, \dots, P_8 pontokba juthat.



L =	{	0,25 mm ha az utol-		
		jára végrehajtott		
		váltóutasítás	kislépésváltó	
		2,5 mm ha az utol-		volt.
		jára végrehajtott		
		váltóutasítás	nagylépésváltó	

A berendezést egy (0,0) kezdőpontból indítva a fej az (n.0,25 mm, m.0,25 mm) koordinátájú pontokba vezérelhető. A toll-fel utasítás hatására a rajzfej a lap (vagy papír) fölé emelkedik (ezután következő mozgató utasítások hatására a levegőben mozog), a toll-le utasítás hatására a rajzfej a lapra (papírra) ereszkedik (ezután következő mozgató utasítások hatására a lapon (papíron) mozog).

A fúrás utasítás hatására a fúrófej leereszkedik a lapra, lyukat fúr, majd a lap fölé emelkedik (a fúrófej mindig a lap fölött mozog).

A PP által elkészített teljes eredményszalagnak a következő részekből kell állnia:

az anyag címe olvasható formában (legible punch),

az egyes technológiai műveletek olvasható neve
(ez tulajdonképpen a gépkezelőnek szóló utasítás),

majd az ADMAP által végrehajtandó utasítások.

A program adatszalgja két fő részből áll - az elvégzendő technológiai műveletek listájából és a (nyomtatott áramköri lapot jellemző) rajzleírásból.

A rajzleírás a lapon található vonalak felsorolásából áll. Az egyes vonalak megadása pontjaik felsorolásával történik, ezen kívül meg kell adni, hogy a vonal hányadik rétegben található (ha többretegű a lap). Az egyes pontokat koordinátáikkal és technológiai utasítással kell megadni. Ezenkívül célszerű, hogy a rajzleírás tartalmazza a lap azonosítóját és méretét.

A technológiai utasítás azt határozza meg, hogy az adott pontban a különböző technológiai műveletek során mit kell a rajzgépnek csinálnia, - azaz egyáltalán el kell-e jutnia a fejnek az adott koordinátájú pontba, és ha igen, milyen utasítás(sorozat) hajtandó végre ebben a pontban.

Példa ADMAP adatszalagra

x forrasztaasoldal

nagyfurat

x

vezeerloo lap 1971.10.1.

keepmeeret: 112,48,

t:19,43, n:21,44.5, (az első vonal)

n:22,44.5, t:24,46.5,

n:19,45, (a második vonal)

n:23,16.5, (a harmadik vonal)

n:17,45, (a negyedik vonal)

v

A PP jelenlegi adata kiterjesztett CHANGE nyelvű programként kerülhet végrehajtásra. (A végrehajtandó technológiai műveletek listájából és a rajzleírásból álló adat felfogható, mint a rajzgép programozására szolgáló közepes szintű programozási nyelven írt program.)

Kiterjesztéssel keletkező új utasítások a PP utasítás, a technológiai műveletek, a műveletek vége, a CIM, a KEPMERET utasítás, a technológiai utasítások, valamint a VEGE utasítás.

Minden program a PP utasítással kell hogy kezdődjön. A PP utasítás előírja a végrehajtási módot (ennek lehetséges változatait tartalmazza a leírás további része), és a program szintaktikus szerkezetét (technológiai műveletek, műveletek vége, cim, képméret, technológiai utasítások, rajz vége). Ezután a CIM utasításra kerül a vezérlés. (Ennek hatására olvasható formában lyukasztásra kerül a program neve.) Ezután a sorrendben következő KEPMERET utasítás végrehajtása következik. (Ennek hatására összehasonlításra kerül a képméret utasításban megadott maximális lapméret a technológiai utasításokban megadott pontok koordinátaival.

- Hiba esetén az összehasonlítás befejezése után a program futása befejeződik - ha nincs hiba, akkor vezérlésátadás történik az első technológiai műveletre.)

A KEPMERET utasítás után a technológiai műveletek kerülnek egymás után (vagy egymással párhuzamosan, ha a PP utasítás ezt írja elő) végrehajtásra. A technológiai műveletek végét jelző utasítás hatására befejeződik a program futása.

Az egyes technológiai műveletek végrehajtása a művelet nevének olvasható lyukasztásával kezdődik. Ezután a művelet típusa (pont, vonal) szerint az adott művelet során figyelembeveendő pontok illetve vonalak listázása következik. (A technológiai utasítások definíciója tartalmazza, hogy mely technológiai műveletek során kell végrehajtani azokat, és milyen ADMAP utasításokkal.)

sítássorozat felel meg az egyes technológiai műveleteknek.) A listázás történhet a rajzleírás-rész egyszeri végrehajtásával. Ezután az optimalizáló program [7] (csatolható FORTRAN rutinok) átalakítja a listát az optimális sorrendnek megfelelően. Most következhet a listában felsorolt utasítás(csoport)ok végrehajtása, amelynek során az abszolút értékükkel adott koordináták közti útnak, ADMAP elmozdulás utasításokat feleltet meg a TRACE utasításként csatolt ELMOZD FORTRAN rutin (azért TRACE utasításként, mivel az elmozdulás számítása utasításközi műveletként értelmezhető), és a technológiai utasítások helyébe ADMAP utasítás(csoport)ok kerülnek.

Az egyes technológiai műveletek végrehajtása során a rajzleírás változatlan maradt, a lista tartalmazta a rajzleírásban szereplő egyes utasítás(csoport)ok végrehajtásának sorrendjét, a lista elemeinek értékét közvetlenül is felveheti az utasításszámláló (az UTSZ utasítás segítségével). Vonaltechnológiai típusú műveletek végrehajtásakor előfordulhat, hogy az optimalizálás egy vonalnak megfelelő utasításcsoport fordított sorrendben való végrehajtását írja elő. (Az UTSZM=-1 utasítás végrehajtásával elérhető a "visszafelé" való végrehajtás.)

Ha nem ragaszkodunk ahhoz, hogy a rajzleírás változatlan maradjon (például a háttérmemóriában tároljuk a rajzleírást és minden technológiai művelet elején újra beírjuk a memóriába), akkor a leválogatás és az optimalizálás az egyes utasítások fizikai átrendezésével is történhet. (Ezt CHANGE utasítások segítségével realizálhatjuk.)

A PP fenti módon történő realizálása viszonylag egyszerű, felhasználhatók a FORTRAN PP egyes rutinjai, az új utasítások definíciója mintegy 100 utasításból álló programot igényel (külön szintaktika analízátor nem szükséges), egységes közös nyelven kerül leírásra a műveleti lista és a rajzleírás, valamint az ADMAP utasítások is.

2. Dialógus-gépek (dialógus-nyelvek)

A számítógépek interaktív használata során az ember-gép párbeszéd (dialógus) szervezése és lefolyása a különböző alkalmazásoknál sok közös elemet tartalmaz és többé-kevésbé független a megoldandó konkrét feladatoktól. A dialógusoknak ezeket az általános jellemzőit különböző szintű dialógus-nyelveken lehet leírni.

Egy dialógus elemi dialógus-lépésekből áll. Egy dialógus-lépés során az ember információt közöl a számítógéppel ("kérdés"), majd információ érkezik a számítógéptől ("válasz"). A számítógép által küldött információ függhet a dialógus előző részétől.

A dialógus minden lehetséges lefolyásának leírása a dialógus programja. Egy dialógus-program felfogható mint egy dialógus-gép, amelyen dialógusok futtathatók.

Egy dialógus-gép (dialógus-program) programozható közvetlenül valamely programnyelven, célszerű dialógus-programok leírására szolgáló valamely dialógus-nyelven írni azokat.

Egy dialógus-nyelv fordító (és végrehajtó) programja (proceszora) programozható közvetlenül valamely programnyelven, vagy valamely általános dialógus-nyelven ("dialógus-processzor összeállító nyelv").

Egy általános dialógus-gépen általános dialógus-programokat lehet futtatni, eredményül dialógus-nyelvek processzorait kapjuk.

A DISTAR-B [8] rendszer egy általános dialógus-gép részének tekinthető. A DISTAR-B rendszer felhasználásával készült az AIR [9] dialógus-nyelv. Mindkét rendszer realizálásra került a CDC 3300-as számítógépen USASI FORTRAN nyelven. A dialógus-programok jelenleg egy ún. direkt kódban (gépi

kód szintű) írhatók le. Tervezés alatt áll egy assembler szintű nyelv.

CHANGE nyelven (kiterjesztéssel) egyszerűen felépíthető magasabb szintű bemenő nyelv. Az általános változók használata lehetővé teszi pszeudodinamikus és korlátozott mértékben dinamikus dialógus-programok írását is, a programot módosító (CHANGE) utasítások lehetővé teszik általános dinamikus dialógus-programok készítését is.

a. A DISTAR-B rendszer

A DISTAR-B rendszer a dialógus-nyelvek processzorainak összeállításához felhasználható általános szubrutinokat tartalmaz.

A DISTAR-B dialógus-nyelveken leírható dialógus-gépek olyan absztrakt számítógépnek tekinthetők, amely véges sok különböző dialógus-állapotban lehet, az egyes állapotokban végrehajt valamely állapotkezdő akciót, kiad egy kérdést, választ fogad és elemez, majd átmeny valamely dialógus-állapotba.

A DISTAR-B rendszer CHANGE nyelven történő realizációja bonyolultabb, de hasonló elvek alapján történhet, mint amelyekben az AIR/CH nyelvnél ismertetésre kerülő eljárás alapszik (ld. előbb). A rendszer kiterjesztése általános dialógus-géppé viszonylag egyszerűen történhet.

b. Az AIR dialógus-nyelv

Az AIR dialógus-nyelv APT típusú nyelveken [3] írt programok (párbeszéd-
des) összeállítására készült, általánosabb célokra is használható (alfanumerikus display használatát feltételezi).

Az AIR nyelvű dialógus-programokban le kell írni az egyes dialógus-állapotokban kiadandó kérdéseket (ezek szemantikailag két csoportba oszt-

hatóak: olyan kérdések, amelyekre szöveg választ vár a dialógus-gép, és ún. "menük", amelyekre a menüben szereplő elemek közül történő választás a válasz), valamint, hogy a kapott válasz alapján mi a következő dialógus-állapot.

A menü-rendszerű kérdésekre a válasz csak a menüből való választás lehet (sorcim, vagy cursorcim megadásával), az egyéb kérdésekre adott szöveg válaszok lehetnek a dialógus-programban előre rögzített szövegek, vagy a felhasználó által a dialógus során adott (új) szövegek. Az új szövegválaszok különböző csoportokba tartoznak (pontok, vonalak, körök, stb. nevei).

A dialógus-programban elő kell írni, hogy melyik csoportba kerüljön az új szöveg (egy-egy adott dialógus-állapotban).

Tehát a dialógus-programban rögzíteni kell a dialógus-állapotok számát és a kérdéseket, valamint a válaszokra következő dialógus-állapotokat. A válaszok lehetnek rögzítettek vagy szimbólikusan jelöltek (ekkor az egyes dialógusok futása alatt kell hogy szöveg értékeket vegyenek fel).

c. Az AIR dialógus-nyelv realizálása CHANGE nyelven

Az AIR processzor központi utasításai CHANGE nyelven mintegy 10 új kiterjesztett utasítás (kb. 80-90 soros program) segítségével előállíthatók (a szöveg típusú változók és a kezelésükre rendelkezésre álló utasítások, valamint az EXTEND utasítások segítenek ehhez a tömörséghez).

Az AIR/CH nyelven az egyes dialógus-állapotok leírása egy olyan programrészlettel történik, amelynek első utasítása egy címkével ellátott szövegkonstans, vagy szöveg típusú változó. A címke felel meg a dialógus-állapot sorszámának, a szöveg a kiadandó kérdésnek.

Ezután annyi utasítás következik, ahány válasz lehetséges (ezen az utasításcsoporton belül bizonyos esetekben össze lehet vonni egyes utasításokat).

Menü-rendszerű kérdés esetén a válasz vagy sorcim (annak a sornak a sorszám, amelyben a kiválasztott válasz található), vagy cursorcim (amely megadja, hogy a képernyő melyik pontján kezdődik a kiválasztott válasz). Ennek megfelelően a LINE (sor) illetve a CURSOR alapszó és az utána álló egész típusú változó (vagy konstans) határozza meg a választ, az ezután következő címke jelöli ki az erre a válasza következő dialógus-állapotot.

Egyéb (nem menü-rendszerű) kérdés esetén az egyes utasítások a válasznak megfelelő szövegkonstansból vagy szöveg típusú változóból és egy címkéből állnak (válasz-utasítás).

A címke adja meg az adott válasz hatására bekövetkező új dialógus-állapotot.

A szöveg típusu változónak értéket kell adni a dialógus futása előtt, vagy a dialógus futása alatt (mielőtt az adott dialógus-állapotra sor kerül).

Ez az értékadás a dialógus futása alatt úgy történhet, hogy a válasz-utasításban a szöveg típusú változó elé egy NEW csoportazonosítót írunk (NEW POINT - új pont, NEW LINE - új vonal,...), ennek az utasításnak a hatására a dialógus során az ebben a dialógus-állapotban beérkezett válasz lesz a szöveg típusú változó értéke. Az AIR/CH processzor megvizsgálja, hogy a NEW csoportazonosítónak megfelelő csoportban nincsen-e már ezzel az értékkel megegyező szöveg (ha van, hibajelzést ad, ha nincs, akkor ez az érték is bekerül a csoportba és a dialógus további futása során előfordulhat a válaszok között).

Választ leíró utasítások összevonása lehetséges, ha változó indexes

változó jelöli ki a választ, és a rákövetkező dialógus-állapotot is. Ekkor az utasításon belül megadható, hogy mely indexpárokat figyelembe véve kell (többszörösen) végrehajtani az utasítást.

Az egyes dialógus-állapotokat leíró programrészek végére írható egy ERROR utasítás, amelyben meg kell adni azt a címkét (dialógus-állapotot), ahol a dialógus-program folytatja munkáját hibás válasz esetén. (Ha a programozó nem ír ERROR utasítást a dialógus-állapotot leíró programrész végére, akkor hibás válasz esetén az AIR/CH processzor leállítja a dialógus-program futását, és hibajelzést ad.)

I R O D A L O M

- [1] P. Wegner: Programming languages, information structures and machine organization
Mc Graw Hill 1968.
- [2] Wesley J. Rishel: Incremental compilers
Datamation 1970.1.p.129
- [3] F. E. Dress: APT Training Manual for Engineers
Manual 1, 2, 3
Sandia Co. 1964.
- [4] Harry Katzan Jr.: Programming and operating systems
"Advanced programming"
Van Nostrand Reinhold Company 1970.
- [5] Legendi T.: A MINSZK-22 számítógép CHANGE/1 fordító-
programja
Számítástechnika '71 konferencia
- [6] Legendi T.: Az ADMAP nyomtatott áramkörti lapokat
gyártó berendezés post-processor programja
Mérés és Automatika 1972.2
- [7] Legendi T.: Keresési algoritmus egy speciális metrikus
térben
Mérés és Automatika 1971.6
- [8] Forgács T. - Distar-B általános dialógusrendszer
Krammer G. AKI Tanulmány 1972.

- [9] Pikler Gy.: Minicomputer-based conversational
 program writing system
 PROLAMAT '73 IFAC konferencia
- [10] Jay Earley - A formalism for translator interactions
 Howard Sturgis: CACM 1970.10
- [11] Mc Intyre, D. E.: An introduction to Illiac IV computer
 Datamation 1970.4
- [12] Paul Berry: IBM APL/360 Primer Student Text
 IBM Technical Publications Department 1969.
- [13] C. T. Fike: PL/1 for Scientific Programmers
 Prentice - Hall, 1970.
- [14] A. Van Wijngaarden,
 B. J. Mailloux, Report on the
 J. E. L. Peck, Algorithmic Language
 C. H. A. Koster: ALGOL 68
 Numerische Mathematik Band 14. Heft 2, 1969.
 1969

T A R T A L O M

Bevezetés.....	3 - 5
A. A nyelv célja.....	A - 1
B. A célok elemzése.....	B - 1
C. A nyelv összefoglaló jellemzése.....	C - 1
D. A nyelv felépítése.....	D.1-1 - D.19.1-1
1. Egy program felépítése.....	D.1 - 1
2. Egy program végrehajtása (a multiprocesszor).....	D.2-1 - D.2-3
3 - 6. Változók és tárolásuk.....	D.3 - D.6-1
7. A szöveg típusú változótömbök tárolása.....	D.7 - 1
8. Az általános változók szerkezete.....	D.8 - 1
9. Deklarációs utasítások.....	D.9-1 - D.9-2
10. Az értékadó utasítások.....	D.10-1 - D.10-2
11. A vezérlésátadó utasítások.....	D.11-1 - D.11-4
12. A CHANGE utasítások.....	D.12-1 - D.12-2-D.13
13. A NULL utasítás.....	D.12-2 - D.13
14. A végrehajtási módot előíró utasítások.....	D.14-1 - D.14-5
15. A STOP utasítások.....	D.15-1
16. Az adatátviteli (input-output) utasítások...	D.16-1 - D.16-3
17. A kiterjesztő utasítások.....	D.17.a-1 - D.17.b-3
18. A nyomkövető (TRACE) utasítások.....	D.18-1 - D.18-9
19. Speciális utasítások.....	D.19.a-1 - D.19.1-1
E. Mit és hogyan biztosít a nyelv felépítése...	E-1 - E-2
F. A processzor realizálása.....	F-1 - F-4
G. A CHANGE nyelv általános jellegű alkalmazása	G-1 - G-3
H. A CHANGE nyelv konkrét alkalmazásai.....	H.1-1 - H.2-5

Irodalom



