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

A CHANGE nyelv/multiprocesszor

Legendi Tamás

Budapest, 1972.

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

Készült az Országos Müszaki Könyvtár és Dokumentációs Központ
házi sokszorositójában
F.v.: Janoch Gyula

B E V E Z E T É S

A dolgozat részletesen ismerteti a CHANGE nyelv elvi és gyakorlati céljait, felépitését, 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 kialakitásához vezettek, kiemelve az új illetve viszonylag új elemeket.

A hagyományos programozási nyelvek különböző előnyeik ellenére a feladatok jelentős részénél csak nehézkesen alkalmazhatóak (különösen célorientált nyelvek feldolgozására, adaptiv programok irására, szintaktikus elemzésre).

A nehézségek főbb okai:

1. A magasabb szintű nyelvek megtartják a gépi kód lineáris végrehajtási módját (az utasitások egymás után kerülnek végre hajtásra, kivéve, ha vezérlésátadó utasitás ir 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 módosithatják a programot tartalmazó rekeszek, az utasitásszámláló (k), valamint az utasitásszámláló-módositó (k) tartalmát.)

3. A magasszintű nyelvek szintakszisa és szemantikája kötött, a forditást és a futást vezérlő paraméterek többsége a forditó-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 utasitásszámláló felhasználásával. (A vezérlésátadó utasitások közvetlenül előirják az utasitásszámláló új értékét, az egyéb utasitások végrehajtása után 1-gyel nő az utasitásszámláló tartalma.) Ezt az eljárást általánositva, egy programot tetszőleges számú, önálló utasitásszámlálóval és utasitásszámláló-módositóval rendelkező processzor hajt végre. (Általános végrehajtási mód.) A processzorokat vezérlő (inditó, megállitó stb.) és az utasitásszámláló-módositó értékét előiró utasitások bevezetésével tetszőleges nem lineáris végrehajtási mód is (dinamikusan) programozható.

2. A végrehajtási módot vezérlő utasitások közvetlen hozzáférést biztositanak a végrehajtást befolyásoló legfontosabb értékekhez. A programok futás közbeni tetszőleges módositását az egyes utasitásokat a programon belül áthelyező utasitások végzik. Ezek az utasitások vagy változatlanul helyezik át az előirt utasitást,vagy pedig az áthelyezés előtt egyes kijelölt változókat aktuális értékükkel helyettesitenek is.
   (Mivel egy programon belül a teljes utasitáskészlet rendelkezésre állhat, igy a program bármely utasitásának helyére tetszőleges utasitás kerülhet.)

3. A kötöttségek feloldását elsősorban az új adattipusok definiálását, valamint automatikus feldolgozását lehetővé tevő kiterjesztő utasitások biztositják.
   Lényeges többlet a hagyományos kiterjeszthető nyelvekkel szemben, hogy a kiterjesztett nyelv (processzor) végrehajtási módja is előirható a kiterjesztő utasitások szemantikai részében.
   Ezenkivül a forditást és a futást vezérlő paraméterek (nyomkövetési szintek, optimalizálási szintek, mely utasitásokat kell csak egyszer végrehaj-

tani, a szubrutinmélység maximális és aktuális értékének előirása stb.) beállitására külön utasitások vannak, amelyek dinamikusan (futás alatt) is alkalmazhatóak.

Összefoglalva:

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

a. az adattipusok és az utasitások szintaktikus és szemantikus leirásán (a nyelv kiterjeszthető);
b. az utasitásokon (a programok futás alatt módosithatók programozott módon);
c. a végrehajtási módon (tetszőleges végrehajtási mód irható elő dinamikusan).

Az ezekkel a tulajdonságokkal rendelkező CHANGE nyelv előnyösen alkalmazható nem lineáris végrehajtási móddal rendelkező célorientált nyelvek definiálására és processzoraik automatikus elkészitésére, valamint adaptiv programok irására.
A nyelv esetleg alkalmazható univerzális programozási nyelvként is, de ebben az esetben a futás áttekinthetőségét zavaró eszközök használatát korlátozni kell és a nyomkövető utasitások fokozott mértékű alkalmazásával lehet elérni a futás megbizható ellenőrzését.

A. A nyelv célja

1. A programnyelvek linearitásának feloldása

2. A programok futás közben történő (ön) módositása programozott úton (újraforditás nélkül)

3. Programozási nyelveket (és egyben processzorokat) kaphassunk kiterjesztéssel egy hajlékony alapnyelvből

4. A programozó egy olyan magasszintű absztrakt számitógépet programozzon, amelyen mindenhez hozzáfér (tehát a nyelv minden lényeges alapelemén lehessen utasitásokkal műveleteket végezni)

5. A multiprocesszoros gondolkodásmód kialakitása

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 előiró utasitásoknak a nyelvbe való beépitésével.

A programok futás közbeni módositását, az utasitásokon műveleteket végző áthelyező utasitások biztositják.

A kiterjeszthetőséget az adattipusok és az utasitások szintaktikus és szemantikus leirásán műveleteket végző utasitások biztositják. A nem lineáris végrehajtási módot előiró utasitások felhasználása az új utasitások szemantikus leirásában lehetővé teszi tetszőleges végrehajtási móddal rendelkező kiterjesztett nyelv (processzor) létrehozását.

2. A célok következményei

a. Feltétlenül interpretálásra vagy inkrementális forditásra [2] van szükség a programot módositó (utasitásokat áthelyező) utasitások, a futás közben változtatható végrehajtási mód, és a nyomkövetési utasitások realizálásához.

b. Interpretálás vagy inkrementális forditás esetén a programok illetve egyes utasitásaik belső ábrázolása eleve megfelel az on-line feldolgozás követelményeinek, igy az utasitáskészlet minimális kiegészitése már lehetővé teszi a multiprocesszor on-line felhasználását is.

C. A nyelv összefoglaló jellemzése

1. Műveleteket lehet végezni egész, valós, logikai, karakter, szövegtipusu változótömbök illetve egyszerű listák elemein, utasitásokon, a CHANGE multiprocesszor végrehajtási módját szabályozó paramétereken, valamint az adattipusok és az utasitások szintaktikus és szemantikus leirásán.

2. A CHANGE nyelv alapnyelv jellegű.
A nyelv alkalmas általános algoritmusok leirására; alkalmazásának fő célja célorientált nyelvek definiálása, és egyben processzoraik automatikus elkészitése.

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 felépithetőek az elemi műveletek alkalmazásával.

4. Az alapnyelv kiterjeszthetősége

a. Új adattipusok és utasitások épithetők be a nyelvbe.

b. A párhuzamos végrehajtást előiró és az utasitásszámlálómódositó értékét kijelölő utasitások segitségével a programok végrehajtása tetszőleges, nem lineáris módon történhet (futás alatt dinamikusan változhat).

c. A CHANGE multiprocesszor végrehajtási módját előiró, a b. pontban emlitett utasitások szerepelhetnek az utasitáskiterjesztő utasitás szemantikai részében, igy speciális végrehajtási móddal rendelkező processzorok (nyelvek) definiálhatók. (Egyes speciális problémaosztályok számára természetesebb lehet (kialakult felfogásához, jelö-

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 utasitások.

D. A nyelv felépitése

1. Egy program felépitése

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

b. Az utasitások lehetnek egyszer végrehajtandó vagy állandó utasitások, ezek tetszőleges sorrendben állhatnak.
Az állandó utasitások mindannyiszor végrehajtásra kerülnek, amikor a vezérlés rájuk kerül. Az egyszer végrehajtandó utasitások csak akkor hajthatók végre, amikor először kerül rájuk a vezérlés, cimkéjüket és a programban elfoglalt helyüket megtartják ezután is. CHANGE utasitás(ok) is hivatkozhat(nak) ezekre az utasitásokra továbbra is. Ha a vezérlés ismételten egy egyszer végrehajtandó utasitásra kerül, akkor az adott végrehajtási lépés (ld. a D.2.b pontot) során az ezt az utasitást végrehajtó processzor nem végez semmilyen műveletet.

Az alapnyelv minden utasitása állandó utasitás.

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

d. Az utasitások pozitiv egész számokkal cimkézhetők.

e. Minden utasitás (belső) sorszámmal rendelkezik.

f. Az első utasitás sorszáma 1, a rákövetkező utasitások 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 utasitást hajt végre. Egy program végrehajtása tetszőleges számú, egymást követő végrehajtási lépésből áll.

A szinkron üzemmód egy speciális üzemmód.
Asszinkron processzorok helyettesítésére lehetőséget.

c. Minden processzor rendelkezik egy utasitásszámlálóval, amelynek tartalma pozitiv egész szám.

d. Minden processzor rendelkezik egy utasitásszámláló-módositóval, amelynek tartalma egész szám.

e. A végrehajtás egy adott lépése során egy processzor számára utasitásszámlálójának tartalma határozza meg a végrehajtandó utasitás sorszámát.

f. Az utasitásszámláló tartalma az utasitás végrehajtása után megváltozik.

megváltozhat! UTSzM 0 is lehet

g. Az UTSZ utasitás közvetlenül kijelöli az utasitásszámláló tartalmát.

h. A vezérlésátadó utasitások kijelölik a következő végrehajtandó utasitás cimkéjét, az ennek a cimkének megfelelő (belső) sorszám kerül az utasitásszámlálóba.

1. Ha nem UTSZ illetve vezérlésátadó utasitás került végrehajtásra,

akkor az utasitásszámláló tartalmához hozzáadódik az utasitásszámláló-módositó tartalma.

j. Az UTSZM utasitás meghatározza az utasitásszámláló-módositó tartalmát (és igy közvetve a következő végrehajtandó utasitás sorszámát).

k. Egy program végrehajtásának a kezdetekor egy processzor működik, utasitásszámlálójának tartalma +1, utasitásszámláló-módositójának tartalma +1. A processzorhoz az 1 sorszám van hozzárendelve.

l. A COPROCESSOR utasitással (a végrehajtó processzorhoz képest) mellérendelt, a SUBPROCESSOR utasitással (a végrehajtó processzorhoz képest) alárendelt új processzor működése inditható el. A processzorok mellérendeltsége (és alárendeltsége) reflexiv, szimmetrikus és tranzitiv reláció. Az egymással mellérendeltségi viszonyban levő processzorok ugyanazoknak a processzoroknak vannak alárendelve.

m. A COPROCESSOR illetve a SUBPROCESSOR utasitásban meg kell adni az új processzor által elsőnek végrehajtandó utasitás cimkéjét, és az elinditott processzorhoz sorszámot kell hozzárendelni (az új processzor a következő végrehajtási lépésben kezd működni, az inditó processzor is folytatja működését).

n. Egy processzor akárhány processzor működését elindithatja, a működő processzorokhoz rendelt sorszámok különbözőek kell hogy legyenek.

o. Bármely processzor számára a WAIT utasitás végrehajtása saját működésének felfüggesztését eredményezi.

p. A COWAIT utasitás előirja valamely, az utasitást végrehajtó processzorhoz képest alá- vagy mellérendelt processzor működésének felfüggesztését.

Az alá- vagy mellérendelt processzor állapotára kötött feltétel alapján vagy feltétlenül. A feltétel jó lehet.

q. A NOWAIT utasitás hatására valamely, az utasitást végrehajtó processzorhoz képest alá- vagy mellérendelt processzor folytatja működését.

r. Egy processzor működése a STOP utasitás hatására fejeződik be (ekkor sorszáma is megszűnik).

s. A SUBSTOP utasitás előirja valamely, az utasitást 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 utasitást végrehajtó processzornak alárendelt összes processzor működését megszünteti a GENERAL STOP utasitás végrehajtása.

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

4. A változóazonositók tetszőleges hosszúságú betűvel kezdődő alfanumerikus karaktersorozattal adhatók meg.
   A változóazonositók mindig egy változótömb azonositójául szolgálnak.

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

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

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

   b. Különböző tipusu tömbelemek különböző számú rekeszben kerülhetnek tárolásra.

   c. Az a. szerint tárolt tömbelemekre indexeik segitségével lehet hivatkozni mind a memóriába iró, mind a memóriából olvasó utasitásokban.

7. A szöveg tipusu változótömbök tárolása.

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

b. A szöveg tipusu tömbelemekre indexeik segitségével (csak) a memóriából olvasó utasitásokban 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 tipusu változótömbök elemei csak indexeik növekvő sorrendjében kaphatnak értéket, (a továbbiakban ismertetésre kerülő karakterszöveg tipusu értékadó utasitásokkal) 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 pozitiv egész szám, pl. a(1), b(5), alfa(20), vagy pozitiv értéket felvevő egész tipusu index nélküli változó, pl. a(j), b(k), alfa(beta).

c. Egy indexes változónak lehet kitevője
A kitevő nem negativ egész szám vagy nem negativ értékeket felvevő egész tipusu változó, pl. $a^{n}(j)$, b (5), alfa (beta).

A kitevő jelentése:

$a^{0}(j) = a^{0}(j) = j$ ha n = 0

$a^{1}(j) = a^{1}(j) = a(j)$ ha n = 1

$a^{3}(j) = a^{3}(j) = a(a(a(j)))$ ha n = 3

d. Index lehet indexes változó is
Pl. a(b(3)), a (b(3)), c(alfa (gamma)), c (beta (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 × jel állhat. (Csak a CHANGE utasitásokban van szerepe.)

9. Deklarációs utasitások

a. Tipusdeklarációs utasitások

A tipusdeklarációs utasitások 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 változóazonositó. A tipusdeklarációs utasitás a tömbnévhez hozzárendeli az utasitásban szereplő alapszónak megfelelő tipust, és helyet biztosit a tömb számára a memóriában a tömbnév után felsorolt (a tömb méretét megadó) konstansok illetve változóazonositók értékének megfelelően. (Ha a tömbdeklarátor csak tömbnévből áll, akkor egyelemű, egydimenziós tömb számára történik helyfoglalás.) Szöveg tipusu változók esetén a tömb első szabad eleme az 1 indexű elem. (Egydimenziósnak tekintve a tömböt.)
Ha egy tömb tipusa megváltozik az utasitás végrehajtásának hatására, akkor elemeinek értéke elvész (definiálatlan).

b. Tömbméret deklarációs utasitások

A DIMENSION alapszó után tetszőleges számú tömbdeklarátor állhat. Az egyes tömbdeklarátorokban szereplő változóazonositók, illetve konstansok száma adja meg az adott tömb dimenziószámát, (aktuális) értéküket határozza meg a tömb méretét (méreteit). Az utasitás végrehajtása helyet biztosit a memóriában az egyes tömbök számára méretüknek (és tipusuknak) megfelelően.
Ha egy tömb dimenziószáma megváltozik az utasitás végrehajtásának hatására, akkor elemeinek értéke elvész. Ha egy tömb dimenziószáma változatlan marad, akkor mindazon (régi) elemeinek értéke megőrződik, amelyek indexei az új indexhatárokon belül esnek.

mi adja meg a tipust? mi a különbség a, és b, között?

AUTODIM

Az AUTODIM utasitás végrehajtása után az egyes változótömbök mérete automatikusan megnő, ha a program a deklarált méretnél nagyobb méretű tömböt kisérel meg használni.

NO AUTODIM

Megszünteti az AUTODIM utasitás hatását.

10. Az értékadó utasitások -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 tipusa meg kell, hogy egyezzen a baloldali változó tipusával. A baloldalon álló változó felveszi a jobboldali kifejezés aktuális értékét.

a. Műveleti jelek: + , - , × , / , ××
(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 tipusu konstansok tizes számrendszerbeli számjegyekből állnak. (A számjegyek maximális száma az adott számitógéptől függ.)

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

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

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

A szöveg konstansok: kezdő és végzárójel közötti tetszőleges karaktersorozatok.

11. A vezérlésátadó utasitások

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

b. Feltétlen vezérlésátadás

GO TO P1

A vezérlés a P1 cimkéjű utasitásra adódik.

c. Feltételes vezérlésátadás

IF (V1) P1, P2, P3

Ha a V1 egész vagy valós tipusu változó aktuális értéke

negativ		P1
nulla	akkor a	P2 cimkéjű utasitásra kerül a vezérlés
pozitiv		P3

IF (V1.REL.V2) P1, P2

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

A megengedett relációk: EQ(=), NE(≠), GT(>), GE(≥), LT(<), LE(≤), IN(⊂), NI(⊄)

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

IF (L) P1, P2

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

d. Ciklusutasitás

DO P1 I1=I2,I3,I4

I1, I2, I3, I4 egész tipusu változó (konstans).
P1 (aktuális) értéke egy REPEAT (cikluszáró) utasitás cimkéjét kell, hogy megadja.

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

(P1) REPEAT

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

következő utasitásra, ellenkező esetben a REPEAT utasitás után (a végrehajtási mód szerint) következő utasitásra adódik a vezérlés.

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

e. Szubrutinhivás

SUBR P1

A SUBR P1 utasitás hatására a P1 cimkéjű utasitásra kerül a vezérlés, és tárolódik a SUBR utasitás után (a végrehajtási mód szerint) következő ("visszatérő") utasitás (belső) sorszáma.

EXIT

Az EXIT utasitás hatására feltétlen vezérlésátadás történik az EXIT utasitást végrehajtó processzor által utoljára végrehajtott szubrutinhivás által meghatározott visszatérő utasitásra.

A visszatérő utasitás sorszámának tárolása (egy-egy processzoron belül) az .SDI. azonositójú tömbben történik az alábbiaknak megfelelően. Az .SDI. tömbnek a fordító program az INTEGER.SDI.(50) utasitásnak megfelelően biztosit helyet a memóriában (automatikusan), az .SD. egész tipusu változónak a 0 értéket adja. Minden szubrutinhivásnál .SD. értéke 1-gyel nő először, majd .SDI.(.SD.)-be kerül beirásra a megfelelő visszatérő cim. Minden EXIT utasitás hatására .SDI.(.SD.) kerül az adott processzor utasitásszámlálójába, majd .SD. értéke csökken 1-gyel.

f. A szubrutinhivást és a visszatérést befolyásoló utasitások

.SDI.(I1)

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

AUTO .SDI.

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

NO AUTO .SDI.

Az utasitást végrehajtó processzorhoz tartozó .SDI. tömb (pillanatnyi aktuális) mérete rögzitett marad.

I1 = .SD.

Az I1 egész tipusu változó felveszi az utasitást végrehajtó processzorhoz tartozó .SD. értékét.

.SD. = I1

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

12. A CHANGE utasitások

a. CHANGE I1 = I2,I3 (utasitásmódositás)

I1, I2, I3 egész tipusu általános változó (konstans) lehet. Az utasitás végrehajtásakor I1, I2, I3 (aktuális) értéke pozitiv egész szám kell, hogy legyen, a programban szerepelnie kell I1, illetve I3 cimkéjű utasitásnak, I2 cimkéjű utasitás nem szerepelhet (kivéve ha I2 = I1, vagy I1 = I2 = I3).

Az utasitás hatására az I1 cimkéjű utasitás helyére az I3 cimkéjű utasitás aktuális értéke kerül, I2 cimkével. (Az I3 cimke, és az I3 cimkéjű utasitás változatlanul marad.)

Egy utasitás aktuális értékét úgy kapjuk, hogy azon változók helyébe, amelyek előtt × áll, beirjuk aktuális értéküket. (Egy utasitás aktuális értéke tehát szintén utasitás, egy olyan utasitás aktuális értéke, amelyben × jel nem szerepel, sajátmaga.)

Például az a(*I) = *S utasitás aktuális értéke
az a(2) = 5.1 utasitás, ha I aktuális értéke 2, aktuális értéke 5.1

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

Új utasitással akkor és csak akkor bővülhet a program, ha az I3 cimkéjű utasitás aktuális értéke nem egyezik meg az I3 cimkéjű utasitással.

Speciálisan fennállhat I1 = I2 (az I1 cimke nem változik) illetve

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

A programban leirt bármely utasitás (akárhányszor) áthelyezhető a program bármely utasitásának helyére. (Tehát a programban eredetileg egyszer szereplő utasitások több helyen is állhatnak - természetesen más-más cimkével -, egyes utasitások eltünhetnek a programból (helyükre valamely más utasitás kerül) speciális esetként elérhető a program utasitásainak átrendezése, illetve néhány (alap) utasitásból álló utasitáskészletből vett utasitásokból álló program futás alatt generálása (a generált program azonnal futhat, ha a generáló program ráadja a vezérlést).

b. CHANGE V1 = V2 (paramétermódositás)

csak V2 !?

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

13. A NULL utasitás

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

14. A végrehajtási módot előiró utasitások

A végrehajtási módot előiró utasitások részbeni ismertetését és jellemzését tartalmazza a D.2. pont, amely leirja, hogy a CHANGE multiprocesszor hogyan hajt végre egy programot, ezért ez a pont az utasitások tömör szintaktikus és szemantikus leirását tartalmazza.

a. UTSZ utasitások

I1 = UTSZ

Az I1 egész tipusu változó felveszi az (utasitást végrehajtó processzorhoz tartozó) utasitásszámláló értékét (az utasitásszámláló értéke az I1 = UTSZ utasitás belső sorszáma).

UTSZ = I1

Az (utasitást végrehajtó processzorhoz tartozó) utasitásszámláló felveszi az I1 egész tipusu változó (konstans) értékét. A következő végrehajtásra kerülő utasitás belső sorszáma I1 (aktuális) értéke.

b. UTSZM utasitások

I1 = UTSZM

Az I1 egész tipusu változó felveszi az (utasitást végrehajtó processzorhoz tartozó) utasitásszámláló-módositó értékét.

UTSZM = I1

Az (utasitást végrehajtó processzorhoz tartozó) utasitásszámláló-módositó felveszi az I1 egész tipusú változó (konstans) értékét.

Ha I1 (aktuális) értéke nem nulla, akkor a következő végrehajtandó utasitás (belső) sorszámát úgy kapjuk, hogy az utasitásszámlálóhoz (amelynek tartalma az UTSZM = I1 utasitás belső sorszáma) hozzáadjuk az utasitásszámláló-módositó új értékét.

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

c. A párhuzamos végrehajtást vezérlő utasitások

COPROCESSOR I1,I2 SUBPROCESSOR I1,I2

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

Az utasitás hatására a multiprocesszor működésének következő lépése során az I1 cimkéjű utasitás végrehajtásával elkezdi működését az I2 sorszámú processzor.

Az utasitást végrehajtó (indító) processzor is folytatja működését.

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

A COPROCESSOR utasitással elinditott processzor mellérendeltségi viszonyba kerül az indító processzorral, és az azzal mellérendelt processzorokkal.

A SUBPROCESSOR utasitással elinditott processzor alá van rendelve az indító processzornak.

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

WAIT

A WAIT utasitást végrehajtó processzor felfüggeszti működését.

COWAIT I1

I1 - egész tipusú változó (konstans)

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

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

NOWAIT I1

I1 - egész tipusú változó (konstans)

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

DCONTROL I1,I2

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

lépés során, amelyben nem független utasitások szerepelnek. Az I2 egész tipusú tömb tartalmazza az utoljára végrehajtott, nem független utasitások belső sorszámait.

NO DCONTROL

Megszünteti a DCONTROL utasitás hatását.

A végrehajtás a párhuzamosan végrehajtott utasitások függetlenségének vizsgálata nélkül történik. Az utasitások hatása követhető annak a konvenciónak a figyelembevételével, amely azt irja elő, hogy az alacsonyabb sorszámú processzor által előirt utasitást kell előbb végrehajtani, egy végrehajtási lépés során (egy processzorral rendelkező számitógépen való végrehajtás esetén).

Az utasitások 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 utasitások

STOP

A STOP utasitást végrehajtó processzor befejezi működését (sorszáma megszűnik).

SUBSTOP I1

I1 - egész tipusú változó (konstans)

I1 aktuális értéke valamely, az utasitást végrehajtó processzornak alárendelt processzor sorszáma kell, hogy legyen. (Sorszáma 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) utasitások

Az I/O utasitások 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 (pozitiv egész számok) lehet hivatkozni, a file-definiciós utasitásokban (előzőleg) meg kell adni, hogy az egyes file-ok hol találhatóak.

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

Minden file-hoz tartozik egy programozható (utasitással módositható) lokációs cim, amely megadja, hogy az adott file-ra vonatkozó adatátviteli műveletet hol kell elkezdeni.
(A cimzés egysége 1 memóriaelem.)

File definiáló utasitások

FILE F IS ARRAY T

F egész tipusú változó(constans).
T szöveg tipusú változó(constans).

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

FILE F IS DEVICE I

F,I egész tipusú változó(constans).

Az F sorszámú file az I sorszámú input/output egység.
(Az egyes számitógépeken az I/O egységeket egyértelműen sorszámozni kell, igy a konkrét multiprocesszorokban ez az utasitás tényleges hozzárendelést valósit meg.)

FILE F IS ON SECONDARY STORAGE

F egész tipusú változó(constans).

Az F sorszámu file a háttérmemóriába kerül.
A file-definiáló utasitások végrehajtásának hatására az adott filehoz tartozó lokációs cim értéke 1 lesz.

A lokációs címet meghatározó utasitás:

LOCATE F TO I

F,I egész tipusú változó(constans).

Az F file-hoz tartozó cim aktuális értéke I lesz az utasitás végrehajtásának hatására.

Az adatátvitelt előiró utasitás:

FROM F1 TO F2

F1, F2- egész tipusú változó(constans) vagy tömbelem (neve) vagy tömbnév.

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

Az adatátvitel az F1-hez illetve az F2-höz tartozó lokációs cimtől kezdődően történik. Az utasitás végrehajtásának hatására mindkét lokációs cimhez hozzáadódik az átvitt adat méretének megfelelő memóriaelemek száma.
F1 és F2 közül (legfeljebb) az egyik lehet tömbelem (neve) vagy tömbnév, ebben az esetben az adatátvitelben a megfelelő tömbelem illetve tömb vesz részt és ennek mérete adja meg az átviendő adat méretét. Ha F1 és F2 egyike sem tömbnév illetve tömbelem(neve) akkor az F2 file-ra át kell vinni az F1 file-nak (a hozzátartozó lokációs cimtől kezdődően) a teljes tartalmát.

17. A kiterjesztő utasitások

A kiterjesztő utasitások viszonylag egyszerű szintaktikával és tetszőlegesen bonyolult szemantikával rendelkező új utasitások definiálását teszik lehetővé.

A hagyományos kiterjeszthető nyelveknél nagyobb fokú hajlékonyságot biztosit egyrészt az, hogy a szintaktikus leirás változókat is tartalmazhat és ezek aktuális értékét a szemantikai leirást képviselő programrész felhasználhatja, másrészt pedig ez a szemantikai leirás két különböző szinten is módosithatja saját magát.
(tartalmazhat CHANGE utasitásokat, amelyek a program végrehajtása során aktivizálódnak, és tartalmazhat egy olyan programrészt, amely a programba való bekerülés pillanatában - egyes fordító-értelmező programoknál ez történhet a forditási fázis során - módositja a szemantikai leirást az aktuális paraméterek, a program egyes aktuális utasitásai függvényében).

a.) Az új utasitásokat definiáló utasitás(ok)

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

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

T1: vagy LABELED - az új utasitást mindig cimkével kell ellátni
vagy UNLABELED - az új utasitást tilos cimkével ellátni

Ha T1 elmarad, akkor az új utasitás állhat cimkével is, cimke nélkül is.

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

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

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

Az EXTERNAL LIBRARY utasitás 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 utasitás hatására a multiprocesszor megvizsgálja, hogy a CH azonositójú (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 utasitások között (19.pont) ismertetésre kerül egy hatékonyabb és egyben bonyolultabb könyvtárszervezésre szolgáló utasitáscsoport. Ezek alkalmazása lehetővé teszi, hogy a multiprocesszor egyes processzorai különböző CHANGE (nyelvű) könyvtárakkal rendelkezzenek.

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

vagy CLOSED - az új utasitás végrehajtásakor zárt szubrutinhivás aktivizálja az új utasitás szemantikáját leiró programot

T3 = OPEN csak akkor lehet, ha T2 = CHANGE

T4: vagy PERMANENT - az új utasitás állandó utasitás

vagy NONPERMANENT - az új utasitás egyszer végrehajtandó utasitás

P1, P2 - pozitiv értékeket felvevő egész tipusú változó (constans) lehet, (aktuális) értékük azokat a cimkéket adja meg, ahol az új utasitás szintakszisa illetve szemantikája a programban található.

A SYNTAX AT és a SEMANTICS AT alapszavak elhagyhatók.

Az új utasitás szintaktikus leirása kétféle módon történhet:
a) <P1> SYNTAX T1,T2,....,Tn

T1, T2, . . . Tn szövegtipusú változó (constans) lehet.
Az egyes T i-k (i=1,2,...n) aktuális értéke lehet

i) alapszó (pl. SYNTAX,EXTEND,DO,stb)
ii) adattipust megadó azonositó (pl. INT,REAL,LOG,CHAR,TEXT az alapnyelv adattipusai esetén, vagy valamely tipuskiterjesztő utasitásban szerepelt T2)
iii) *jel után álló egész tipusu változó, név vagy constans (szövegformában)
iv) kezdő vagy végzárójel

T1, T2,...Tn megadja, hogy balról-jobbra haladva egymás után milyen rögzitett alapszavak illetve milyen tipusú változók szerepelhetnek az új utasitásban. Az alapszavak és adattipusok egyszerű felsorolásán kivül lehetséges az esetleges ismétlődő részek tömör megadása a következő módon:

Ha Ti, Ti+1,....Tk az ismétlődő rész, akkor ezt zárójelek közé helyezzük (Ti-1-ben kezdő, Tk+1-ben záró zárójelet helyezünk el). Ha az ismétlődések száma előre rögzitett, akkor Tk+2 - ben egy olyan szöveget helyezünk el, amelynek első karaktere *, az ezután következő számjegyek irják elő az ismétlődések számát.
Ha az ismétlődések száma tetszőleges, akkor Tk+2 - ben egy olyan szöveget helyezünk el, amelynek első karaktere *, az ezután következő karakterek egy egész tipusú változó nevét adják meg. Az új utasitás szintaktikus elemzése során ennek az egész tipusú változónak az értéke az ismétlődések tényleges száma lesz. A továbbiakban ismertetésre kerülő szemantikai utasitás felhasználhatja ezt az értéket, ilyen módon más-más jelentés tulajdonitható a különböző ismétlésszámú (de egyébként azonos strukturájú) utasitásoknak.
Több ismétlődő rész is lehet, az ismétlődő részek (teljes egészükben) egymásba lehetnek ágyazva (tetszőleges mélység megengedett).
Az utasitás irható

<P1> SYNTAX T ARRAY

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

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

P3,P4 - pozitiv egész értékeket felvevő egész tipusú változó (constans) lehet, (aktuális) értékük két, a programban szereplő cimkét ad meg.

A P3 cimkéjű utasitással kezdődő és a P4 cimkéjű utasitással végződő programrész a szintaktikus analizist végző rutinhoz kerül kiegészitésként. Ha ez a rutin valamely utasitást nem tud feldolgozni, akkor átadja ezt az utasitást a K karakter tipusú tömbön a kiegészitő rutin(ok)nak, amely(ek) eredményként az a) pontnak megfelelelő szintaktikus leirást kell hogy adjon (adjanak) a T szöveg tipusú tömbön. Ha a kiegészitő rutin(ok) sem tudja (tudják) értelmezni az utasitást, akkor hibajelzést ad a multiprocesszor.

SYNTAX P1==SYNTAX P2

P1,P2 - pozitiv egész értékeket felvevő egész tipusú változó (constans) lehet (aktuális) értékük két, a programban szereplő SYNTAX utasitás cimkéjét kell hogy megadja.

Az utasitás hatására a P1 cimkéjű SYNTAX utasitással és a P2 cimkéjű SYNTAX utasitással leirt strukturájú utasitás (csoport)hoz ugyanaz a szemantikus leirás fog tartozni. (EXTEND utasitással előzőleg vagy csak a P1 cimkéjű SYNTAX utasitáshoz vagy csak a P2 cimkéjű SYNTAX utasitáshoz hozzá kellett rendelni szemantikust leirást. Ellenkező esetben - egyik utasitáshoz sem, ill. mindkettőhöz történt hozzárendelés - az utasitás hatástalan és hibajelzés történik).

Az új utasitás szemantikus leirása a következőképpen történik:

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

Pi (i=1,2,...6) pozitiv egész értékeket felvevő egész tipusú változó (constans) lehet, (aktuális) értékük a programban szereplő cimkéket kell hogy megadjon.

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

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

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

a)szimbólikus változók("formális paraméterek") amelyek helyére a végrehajtandó (új) utasitás tényleges változói ("aktuális paraméterei") kerülnek az alább ismertetett szabályok szerint.A szimbólikus változók tipusdeklarációs utasitásban nem szerepelhetnek.
b)ideiglenes változók, ("lokális változók") amelyekre csak addig van szükség, amig az (új) utasitás végre nem hajtódik. (Ezeknek szerepelniük kell a deklarációs utasitásokban)

A P3 cimkéjű utasitással kezdődő és a P4 cimkéjű utasitással végződő programrész irja elő (a szintaktikus leirással együtt) a szimbólikus - tényleges változócserét ("formális - aktuális paramétercsere")
Ebben a programrészben

V == t y p e n a m e (I) utasitások biztositják a változócserét.
V - szimbólikus változó
t y p e n a m e helyére tetszőleges adattipust megadó azonositó (INT, REAL,LOG,CHAR,TEXT vagy valamely tipuskiterjesztő utasitás T2 -je) irható.
I - egész tipusú változó (constans)

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

A jobb oldalon álló adattipus azonositó az utána zárójelben álló sorszámmal (I) együtt a végrehajtandó (új) utasitás valamely változóját adja meg. A sorszám határozza meg, hogy a végrehajtandó utasitásban balról-jobbra haladva az adattipus - azonositó által előirt tipusú változók közül hányadikat kell figyelembe venni.

A P3-P4 programrészben szerepelhetnek egyéb utasitások is (pl. ciklusutasitás) de az előforduló változóknak (ebben a programrészben szereplő) deklarációs utasitásokban szerepelniük kell.
Ezen kivül a vezérlés nem adható át a P3-P4 programrészen kivülre ennek

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

A külső nyelvű programok által leirt szemantika esetén az alapnyelv nem irja elő a paraméterátadást illetve a módositást szabályozó utasitásokat.

A P5 utasitással kezdődő és a P6 utasitással befejeződő zárt programrész irja elő a P1-P2 illetve a P3-P4 programrész módositását. Ez a módositás függhet a végrehajtandó (új) utasitás

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

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

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

Ha nincsen módositás, akkor a SEMANTICS utasitásból elhagyható a P4 utáni vesszővel kezdődő rész.

RELEASE P

P pozitiv egész értékeket felvevő egész tipusú változó (constans) lehet, (aktuális) értéke valamely, a programban szereplő SYNTAX vagy PROGRAM FOR SYNTAX utasitás cimkéjét kell hogy megadja.

Az utasitás hatására a végrehajtó processzor által használt könyvtárból (EXTERNAL,CHANGE) törlődik a P cimkéjű utasitás által leirt szintaktikájú utasitás.

b.) Az adattipus - kiterjesztő utasitás

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

T1, T2 szöveg tipusú változó(constans)
I1, P1,2,3 egész tipusú változó(constans)

T1 adja meg az uj adattipushoz tartozó tipust előiró új utasitás alapszavát. / Az új tipust előiró utasitás, az alapnyelv tipusdeklarációs utasitásaihoz hasonlóan a tipust előiró alapszóból, majd vesszőkkel elválasztott tetszőleges számú tömbdeklarátorból áll./

T2 irja elő, hogy az utasitáskiterjesztő utasitás(ok)ban milyen névvel kell hivatkozni az uj adattipusra.

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

P1 cimkéjű SYNTAX utasitás az uj adattipusnak megfelelő constans szintaktikus leirását 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 - cimke

A P4 - P5 zárt programrész irja elő, hogy a P1 cimkéjű SYNTAX utasitás által előirt külső adatábrázolási módnak milyen belső (memóriabeli) adatábrázolás felel meg. A SEMANTICS utasitással azonos módon történik a szimbólikus-tényleges változócsere. (a P6 - P7 programrész irja elő.)

Ezen kivül kitüntetett szerepe van a .NEWTYPE. változótömbnek. Ez nem szerepelhet tipusdeklarációs utasitásban, a multiprocesszor (automatikusan) hozzárendeli a megfelelő (új) tipust, tömbméret deklarációs utasitásban 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 - cimke

A .NEWTYPE. teljes tömb (deklarálása azonos korlátozásokkal történik, mint az INPUT utasitás esetében) tartalmazza az új tipusú adatot. A P8-P9 programrész irja elő a .NEWTYPE. tömbnek a P1 cimkéjű SYNTAX utasitás szerinti kivitelét (kiirását). A szimbólikus változócserét t y p e n a m e (I) = = V utasitások biztositják (a P10 - P11 zárt programrészben).

V - szimbólikus változó

t y p e n a m e (I) a SEMANTICS utasitásnál ismertetett módon a SYNTAX utasitásban 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 kiirásra.

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

Az EXTEND TYPE utasitás végrehajtása lehetővé teszi az új adattipust deklaráló utasitás használatát, a programokban az új adattipus változóként és constansként való alkalmazását, valamint az új tipusú adatok és constansok adatátviteli műveletekben való használatát.

Az új adattipussal végzendő műveleteket utasitás-kiterjesztő EXTEND utasi*ás(ok) segitségével lehet leirni.

És hogy hangzanak ez új utasitások?

18. A nyomkövető (TRACE) utasitások.

A nyomkövető utasitások a nyelv szerves részét képezik a "mindenhez hozzáférés" elvének megfelelően. Fontos feladatuk, hogy a nyelv "veszélyes" elemeinek (különösen a programot módosító utasitásoknak) a hatását ellensúlyozzák, biztositsák, hogy a programozó ne veszitse el a futó program feletti ellenőrzést. A programok elkészitése (belövése) alatt nyújtott hathatós segitségen kivül a nyomkövető utasitások lényeges információkat szolgáltathatnak a kész (hibátlan) programokról is. (a memória kihasználtságáról, az egyes programágak használatának gyakoriságáról, stb.) Ezen kivül jól felhasználhatók olyan célnyelvek definiálásánál, amelyeknél gyakoriak az "utasitásközi műveletek", azaz természetes igény, hogy (bizonyos tipusú) utasitások között (külön ki nem irt) standard rutinok kerüljenek végrehajtásra.
A nyomkövető utasitások eredménye folyamatosan kiiratható, vagy file-okon tárolható és külön utasitásokkal iratható ki. Megszüntethető külön-külön az egyes nyomkövető utasitások hatása, megszüntethető (egy utasitással) az összes előzőleg végrehajtott nyomkövető utasitás hatása, törölhető a programból az összes nyomkövető utasitás.

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

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

S T A C K I1

I1 pozitiv egész értékeket felvevő egész tipusú változó (vagy constans).

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

Ez az I1 utasitás és eredményeik ("a stack") kiiródnak, hiba miatti leállás esetén automatikusan, vagy kiirathatók az alábbi utasitásokkal.

W R I T E S T A C K

Az utasitás végrehajtásának hatására kiiródik a stack.

WRITE STACK m o d e

m o d e: AT P1 (amikor a vezérlés a P1 cimkéjű utasitásra kerül, kiiródik a stack)

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

Többféle m o d e is érvényben lehet egyidejüleg.

LABEL TRACE

A multiprocesszor által végrehajtott utasitások közül a cimkével ellátott utasitásokat figyeli, hatására kiiródnak az érintett utasitások cimkéi. Ha csak egy programrészen kivánjuk elérni ezt a hatást, akkor a

LABEL TRACE P1 - P2

utasitást kell alkalmazni. (hatása a P1 cimkéjű utasitással kezdődő és a P2 cimkéjű utasitással végződő programrészre terjed ki)

GENERAL TRACE

Minden végrehajtott utasitás és eredménye kiirásra kerül.Ha csak egy programrészen kivánjuk elérni ezt a hatást, akkor vagy a

TRACE P1 - P2

utasitást kell használni, (hatása a P1 cimkéjű utasitással kezdődő és a P2 cimkéjű utasitással végződő programrészre terjed ki) vagy a

TRACE P1

utasitást kell használni (hatása a P1 cimkéjű utasitásra terjed ki)

ARRAY T1 PROTECTED AGAINST WRITING

T1-szövegtipusu változó (vagy constans), valamely tömb nevét tartalmazza.
Ha a T1 aktuális értékének megfelelő tömb valamely elemébe beirási kisérlet történik, hibajelzés kerül kiirásra (a beirás nem történik meg, a program fut tovább)
Hasonlóan kiolvasás ellen is védhető valamely T1 tömb, az

ARRAY T1 PROTECTED AGAINST READING

utasitás alkalmazásával.

Programrészt tartalmazó memóriaterület is védhető irás/olvasás ellen a

PROGRAM P1-P2 PROTECTED AGAINST WRITING
PROGRAM P1-P2 PROTECTED AGAINST READING

utasitások alkalmazásával.
Védeni lehet egy tömb elemeit az előzetes értékadás nélkül történő felhasználástól az

ARRAY T1 PROTECTED AGAINST READING WITHOUT WRITING

utasitással.
Figyelhető egyes tömbök illetve programterületek aktuális értékének változása a

TRACE ARRAY T1 illetve a
TRACE PROGRAM P1-P2 utasitás alkalmazásával

T1-szöveg tipusu változó(constans) P1,P2-cimke.Kiirásra kerül a változást okozó utasitás és eredménye.(a program fut tovább)

Az UNUSED MEMORY ill. az UNUSED PROGRAM

utasitás hatására a multiprocesszor figyeli (tárolja) a felhasználásra nem kerülő tömbelemet illetve utasitásokat és kiirja ezeket a

WRITE UNUSED MEMORY illetve a

WRITE UNUSED PROGRAM
utasitás hatására.
COUNTING

Az utasitás hatására a multiprocesszor számlálja az egyes tömbök és az egyes utasitások használatát ill.v végrehajtását.

MEMORY MAP

Az utasitás hatására kiirásra kerülnek a használatban levő tömbök ne-

vei, tipusuk és aktuális méreteik (valamint használatuk gyakorisága, ha előzőleg COUNTING utasitás került végrehajtásra)

LIST

Az utasitás hatására kiirásra kerülnek a program aktuális utasitásai, a bennük szereplő változók aktuális értékei(valamint az utasitások végrehajtási gyakorisága, ha előzőleg COUNTING utasitás került végrehajtásra)

LIBRARY LIST

Az utasitás hatására kiirásra kerülnek a végrehajtó processzorhoz tartozó könyvtár(ak) utasitásai.

TRACE PROCESSZOR I1,P1 AT P2

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

Az utasitás hatására a P2 cimkéjű utasitás végrehajtásával egyidejűleg megkezdi működését az I1 sorszámú processzor a P1 cimkéjű utasitás végrehajtásával.(I1 sorszámú működő processzor nem lehet a multiprocesszorban a P2 cimkéjű utasitás végrehajtásakor)Az új, I1 sorszámú processzor alá van rendelve a TRACE PROCESSOR utasitást végrehajtó processzornak.

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

Futási hibák (túlcsordulás, a programban nem szereplő cimkére történő vezérlésátadás, stb.) esetén a multiprocesszor hibajelzést ad és folytatja működését. (a hibás művelet eredménye definiálatlan) Az eddig ismertetett nyomkövetési utasitások között szerepelt indextúlcsordulás figyelését, memóriavédelem, stb. megvalósitását előiró utasitás. Ezek a nyomkövetési utasitások is azt irják elő, hogy hibajelzést adjon a multiprocesszor, majd folytassa működését.

Az IF ERROR utasitás segitségével más tipusu akciók is előirhatók futási hibák esetén. A c o m a n d rész irja elő, hogy mely utasitásokra vonatkozik az IF ERROR utasitás, az a c t i o n rész pedig a végre-

hajtandó akciót irja elő.

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

P1

T

T szöveg tipusú változótömb, amely valamely utasitás (csoport) szintaktikus leirását tartalmazza, a SYNTAX utasitásnak megfelelő módon, azzal a különbséggel, hogy elegendő a szintaktikus leirás elejét irni le. Az IF ERROR utasitás hatása kiterjed valamennyi olyan utasitásra, amelynek a szintaktikus leirása úgy kezdődik, mint a T - ben levő szintaktikus leirás.

a c t i o n STOP (A hibás utasitást végrehajtó processzor és az annak alárendelt processzorok befejezik működésüket, a hibás utasitás kiirődik).

TRACE PROCESSOR I1, P1, I2, I3

A hibás utasitás végrehajtását követő lépésben megkezdi működését az I1 sorszámú processzor a P1 cimkéjű utasitás végrehajtásával. I3 a hibás utasitás (belső) sorszáma, I2 az ezt végrehajtó processzor sorszáma. Az I1 sorszámú processzor mellérendeltségi viszonyba kerül az I2 sorszámú processzorral. (I1, P1, I2, I3 pozitiv egész értékeket felvevő egész tipusú változók (I1,P1 constans is lehet), I1 sorszámú működő processzor nem lehet a multiprocesszorban).

Ellentétes hatású IF ERROR utasitások közül mindig az utoljára végrehajtott utasitásban előirt akció az érvényes.
A kapcsoló jellegű STACK, UNUSED MEMORY, UNUSED PROGRAM, COUNTING utasitások kivételével az összes eddig ismertetett nyomkövető utasitás elé (az utasitáson belül) irható a

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

utasitásrész, amelynek hatására az adott nyomkövető uta-sitás eredményei nem kerülnek kiirásra, hanem az F filera kerülnek.

A nyomkövetési erdmények kiirathatók a

WRITE TRACE FILE F m o d e utasitással.

Az F nyomkövetési információt tartalmazó file kiirásra kerül a m o d e rész által előirt időben. (a kiirás után a file tartal-ma törlödik).

m o d e az utasitás végrehajtásakor.

AT P1 a P1 cimkéjű utasitás végrehajtásakor.

EVERY I1 STEP minden I1 - edik végrehajtásil lépésben.
(I1 pozitiv értékékeket felvevő egész tipusú változó (constans)).

A nyomkövetési információk file-ra küldése felhasználható a különböző tipusú információk szétválasztására (külön-külön történő kiiratására) valamint a karakterformában tárolt információk programmal történő elemzésére.
(a kiiratás mellett vagy helyett).

NO TRACE

Valamennyi TRACE utasitás hatása megszűnik.

NO TRACE P1

A P1 cimkéjű TRACE utasitás hatása megszűnik.

CLEAR TRACE

A programból törlődnek az összes nyomkövető utasitások.

19. Speciális utasitások

A speciális utasitások közé olyan utasitás-csoportok kerültek be, amelyek nem teljesek (egyes utasitásaik más csoportoknál vannak felsorolva, további fejlesztés alatt állnak), vagy amelyek több csoportba is bekerülhettek volna (esetleg zavarták volna azoknak a csoportoknak az egységességét). Szerepel itt néhány olyan érdekes utasitás is, amely nem tartozik feltétlenül a nyelv egységes koncepciójának megfelelően a nyelvhez, de hasznos kiegészitő szerepe van (pl. a speciális szubrutin illetve corutinhivó utasitások).

a. A programokhoz (szerkezetükhöz, belső ábrázolásukhoz) való teljesebb hozzáférést segitik elő a következő (a végrehajtási módot előiró, a szubrutinokból való visszatérést befolyásoló és a nyomkövető utasitásokat kiegészitő) speciális utasitások:

I1 IS LABEL OF INSTRUCTION I2

I1,I2 - pozitiv értékeket felvevő egész tipusú változó (konstans)

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

INSTRUCTION I1 HAS LABEL I2

I1,I2 - pozitiv értékeket felvevő egész tipusú változó (konstans)

Az I1 változó értéke az I2 cimkéjű utasitás belső sorszáma lesz (ha

nincsen I2 címkéjű utasitás a programban, akkor I1 értéke 0 lesz).

LOCATION OF ARRAY T1 TO I1

T1 - szöveg tipusú változó (konstans)
I1 - egész tipusú változó (konstans)

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

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

(I1,I2 - egész tipusu változó (konstans))
speciális értékadó utasitásban szerepelhetnek.

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

DESCRIPTON OF INSTRUCTION I1 TO I

I1 - egész tipusú változó (konstans)
I - egész tipusú változótömb

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

Mivel egy paramétert több cim is leirhat, az egyes paraméterek leirását egy-egy 0 értékű tömbelem választja el.

Egy paraméter leirása a paraméterben szereplő tömbök kezdőcimeiből illetve a konstansok cimeiből áll, a paraméterben való előfordulásuk (balról-jobbra haladva) sorrendjében.

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

Egy konstans cime a .MEMORY. tömbön belüli indexe.

A paraméteren belül szereplő kitevő leirásában előforduló cimek negativ előjelet kapnak.

CODE OF INSTRUCTION P1 TO I1

P1,I1 - egész tipusú változó (konstans)
P1 - valamely SYNTAX utasitás cimkéje

Az utasitás hatására az I1 változó értéke a P1 cimkéjű SYNTAX utasitás által leirt utasitás(csoport) kódja (belső sorszáma) lesz.

SUBPROCESSORS OF I1 TO I

I1 - egész tipusú változó (konstans)
I - egész tipusú változó(tömb)

Az utasitás 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 negativ előjellel van ellátva.

COPROCESSORS OF I1 TO I

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

PROCESSOR LIST TO I

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

NUMBER OF PROCESSOR TO I

I - egész tipusú változó

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

UTSZ OF PROCESSOR I1 TO I2

I1 - egész tipusú változó (konstans)
I2 - egész tipusú változó

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

UTSZM OF PROCESSOR I1 TO I2

Az utasitás hatása megegyezik az előző utasitás hatásával, az egyetlen különbség az, hogy az utasitásszámláló-módositó értéke kerül átadásra.

Az előző két utasitásban szereplő feltételek mellett lehetséges a forditott értékadás is, az

I2 TO UTSZ OF PROCESSOR I1 és az

I2 TO UTSZM OF PROCESSOR I1

utasitások alkalmazásával.

b. Szöveg és karakter tipusú változókat kezelő utasitások

CLEAR T

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

T1 TO T

A T szöveg tipusú változótömb első szabad eleme felveszi a T1 szöveg tipusú 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 tipusú változó felveszi a T1 szöveg tipusú változó sorszámát a T szöveg tipusú 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 tipusú változó értéke a T1 szöveg tipusú változó karaktereinek száma lesz az utasitás 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 tipusú változótömb 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 tipusú változó (szöveg) értékében szereplő karakterek száma. Az utasitás 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 elősegitő utasitások:

EXECUTE

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

SAVE

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

SAVE AND EXECUTE

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

DELETE s p e c i f i c a t i o n

Az utasitás végrehajtása után törlődnek a programból a specifikációs rész által előirt utasitások.

s p e c i f i c a t i o n	I1	az I1 belső sorszámú utasitás
	I1-I2	az I1 belső sorszámú utasitással kezdődő, és az I2 belső sorszámú utasitással végződő programrész

LABELED P1	a P1 címkéjű utasitás
LABELED P1-P2	a P1-P2 programrész
SYNTAX AT P1	a P1 címkéjű SYNTAX utasitás által meghatározott utasitás(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 utasitás után (közvetlenül) az INSERT utasitást követően a multiprocesszorba beérkező utasitások, a P1 címkéjű utasitással bezárólag. A specifikációs rész megegyezik a DELETE utasitásnál felsoroltakkal, csak a SYNTAX AT P1 forma nem megengedett.

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

PROGRAM TO FILE F

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

PROGRAM AND DATA TO FILE F

A program aktuális utasitásai, 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 utasitást végrehajtó program és változóterülete törlődik a memóriából.

Tömbök illetve a program szegmentálását szolgálja a következő két utasitás:

ARRAY T1 TO FILE, BUFFER I1,I2

T1 - szöveg tipusú változó (konstans)
aktuális értékét T-vel jelöljük
I1,I2 - egész tipusú változó (konstans)

Az utasitás 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 kivánt tömbelemeknek a fő memóriába juttatását egy I1+I2+1 tömbelem méretű puffer gyorsitja meg. A puffer tartalma az utasitás végrahajtásakor a tömb első I1+I2+1 eleme lesz (egy-indexesnek tekintve a tömböt). A továbbiakban a puffer tartalma mindig addig marad változatlan, amig a kivánt tömbelemek megtalálhatók benne. Ha a keresett K indexű tömbelem nem található a pufferban, akkor a T(K-I1),T(K-I1+1),... ...,T(K),...,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 utasitáshoz hasonlóan, a P1-P2 programrész átkerül a háttérmemóriába, a fő memóriában elfoglalt helye felszabadul, a továbbiakban is rákerülhet a vezérlés, illetve utasitásaira lehet hivatkozni. Aktivizálását egy I1+I2+1 utasitás méretű puffer gyorsitja meg, amelynek tartalma az utasitás végrehajtásakor a P1 cimkéjű utasitás és az utána következő I1+I2 darab utasitás. A továbbiakban a puffer tartalma mindaddig változatlan marad, amig a keresett utasitások megtalálhatók benne. Ha a K belső sorszámú utasitás nem található a pufferban, akkor a K-I1,K-I1+1,..., K,...,K+I2-1,K+I2 belső sorszámú utasitások kerülnek a pufferba.

e. A file-ok (rá)irással és (ki)olvasással szembeni védelmét a nyomkövető utasitásoknál ismertetett, tömbök védelmét szolgáló utasitásokkal azonos módon biztositja a

FILE F1 PROTECTED AGAINST WRITING, a
FILE F1 PROTECTED AGAINST READING és a
FILE F1 PROTECTED AGAINST READING WITHOUT WRITING

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

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

Állandó file-ok használatát biztositja a következő három utasitás:

T IS PERMANENT FILE l e n g t h

T - szöveg tipusú változó (konstans)

l e n g t h - üres (változó hosszúság)
- LENGTH I1 egész tipusú változó (konstans)

Az utasitás 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 tipusú változó (konstans)
T - szöveg tipusú változó (konstans)

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

CLEAR FILE T

T - szöveg tipusú változó (konstans)

Törlődik a T file az utasitás hatására.

f. A CHANGE nyelv alap utasitáskészlete állandó utasitásokból áll. Egyszer végrehajtandó utasitások a következőképpen keletkezhetnek:

a. Új utasitások definiálásakor az EXTEND utasitás A4 paraméterének értéke NONPERMANENT.

b. A NONPERMANENT s p e c i f i c a t i o n utasitás segitségével

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 utasitás, illetve a P1 cimkéjű SYNTAX utasitás által leirt (minden) utasitás egyszer végrehajtandó utasitás lesz.

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

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

SUBR P1 BACK AFTER I1 STEPS

P1,I1 - egész tipusú változó (konstans)

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

COPROCESSOR V1, V2 DURING I1 STEPS
SUBPROCESSOR V1, V2 DURING I1 STEPS

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

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

NOWAIT V1 DURING I1 STEPS

V1,I1 - egész tipusú változó (konstans)

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

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

I1,I2,...,I(N) - egész tipusú változó

Az utasitás hatására N darab új, a végrehajtó processzorhoz képest alárendelt processzor kezd működni a következő végrehajtási lépés során az I1,I2,...,I(N) cimkéjű utasitásokat hajtva végre. Az új processzorokhoz automatikusan sorszámot rendel hozzá a multiprocesszszor, 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 utasitás GO TO ARRAY I formában is irható, ekkor a teljes I tömb játssza I1,I2,...,I(N) szerepét.

COMMENT T

A COMMENT alapszó után az utasitáson belül tetszőleges szöveg vagy szöveg tipusú változó irható. Az utasitás hatása megegyezik a NULL utasitás hatásával.

Minden processor egyaránt képes megérteni az új utasítás mintarészlet és semantikáját, azaz hogy új utasítást melyik processor végrehajthat. Miért kell azon mesterségesen különbséget tenni processzorok közöt? H. egyetlen processor "kiszolgálhatná" az új utasításokat a sorból.

h. Az EXTEND utasitásoknál leirt módon történő könyvtárszervezés és kezelés merev, egyes bonyolultabb feladatok megoldására igen nehézkesen használható fel. A következő utasitáscsoport a különböző könyvtárak dinamikus (futás alatti) kezelését biztositja, 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 adattipusainak és utasitásainak szintaktikus és szemantikus leirása található. A multiprocesszorhoz tartozó alapkönyvtár nem módositható és nem bővithető. A multiprcesszor működésének a kezdetekor az alapkönyvtár egy másolata áll rendelkezésre, ez módositható, bővithető, tetszőleges file-ra vihető. Egy file-ra vitt (tetszőlegesen módositott) alapkönyvtár behivható a multiprocesszorba az éppen bent lévő alapkönyvtár helyett. Egy alapkönyvtárban minden adattipusnak és utasitásnak egyértelmű sorszáma van.

Az EXTEND utasitások előirhatják CHANGE könyvtár kibővitését (létrehozását). Ezeknek az utasitásoknak a hatására az EXTEND utasitást végrehajtó processzorhoz tartozó CHANGE nyelvű könyvtárba bekerül az új adattipus illetve utasitás szintaktikus és szemantikus leirása, és ezenkivül az alapkönyvtárba is bekerül a szintaktikus leirás, és ott egyértelmű sorszámot kap. (Ez a sorszám tárolásra kerül a CHANGE könyvtárban is.)

Egy CHANGE könyvtár file-ra vihető, file-ról behivható valamely processzorba. CHANGE könyvtár létesitésekor egy üres (egész tipusú C-

-kat tartalmazó) file-ről kell behivni egy "CHANGE könyvtárat", az ennek a könyvtárnak a "kibővitését" előiró első EXTEND utasitás hatására a könyvtár iniciálása is megtörténik.

Az adattipusok és utasitások sorszámozását az teszi szükségessé, hogy egy program futása előtt általában nem lehet tudni, hogy az egyes utasitásokat mely processzorok fogják végrehajtani. A programok egyértelmű leforditásához (olyan egyértelmű belső ábrázolásához), amely lehetővé teszi, hogy bármely processzor végrehajthasson (egyértelműen) bármely utasitást, az alap könyvtárban össze kell gyűjteni a multiprocesszorban érvényben levő összes utasitásnak legalábbis a szintaktikus leirását.

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 leirások ugyanazzal a sorszámmal szerepelnek-e az alapkönyvtárban (ha előfordulnak az alapkönyvtárban). Hiba esetén a program futása leáll.

Új alapkönyvtár aktivizálásakor az egész program újra forditása kerül végrehajtásra.

Ez az utasitáscsoport lehetővé teszi különböző számitógépek egy rendszeren belül való együttműködésének modellezését.

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

Ezek az utasitások kisérleti jellegűek, ha alkalmazásuk hasznosnak bizonyul, akkor a könyvtárakra vonatkozó szabályok a multiprocesszor általános leirásánál illetve az EXTEND utasitásoknál kerülnek ismertetésre.

EXTERNAL LIBRARY TO FILE F

F - egész tipusú 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 utasitás végrehajtásának a hatására. (A multiprocesszorban levő alapkönyvtár törlődik.)

LIBRARY TO FILE F

F - egész tipusú változó (konstans)

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

LIBRARY FROM FILE F

F - egész tipusú változó (konstans)

Az utasitást 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 tipusú változó (konstans)

Az utasitást végrehajtó processzor számára az I sorszámú processzor alárendelt kell hogy legyen. Az utasitás 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.)

i. TRANSLATE P1-P2 TO F

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

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

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

E. Mit és hogyan biztosit a nyelv felépitése

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

   b. EXTEND utasitás(ok)ban használva a COPROCESSOR, a SUBPROCESSOR és az UTSZM utasitást, a kiterjesztett multiprocesszor tetszőleges nem lineáris módon hajthat végre programokat. Az egyes speciális problémaosztályok számára kedvezőbb lehet egy speciális nem-lineáris végrehajtási mód, mint az univerzális lineáris mód.

2. A CHANGE utasitások biztositják a programok dinamikus (futás közben történő) programozott módositását, lehetővé teszik CHANGE nyelvű programok futás közbeni generálását (adaptiv programok, forditóprogramok, stb.).

3. Az EXTEND utasitások lehetővé teszik új adattipúsok és utasitások definiálását, és a nyelvbe való automatikus beépitését.

4. A programozó igen szabad kezet kapott a forditás és a futás vezérlésére. A "mindenhez való hozzáférés" elvének realizálására rendelkezésre állnak a végrehajtási módot előiró (UTSZM és párhuzamos feldolgozást vezérlő) utasitások, a kiterjesztő (EXTEND) utasitások, a programok önmódositását lehetővé tevő CHANGE utasitások (ezek használata különösen a változódefinició által megengedett egyszerű listák segitségével lehet előnyös), a szubrutinmélységet szabályozó ISD utasitások, valamint a nyomkövető (TRACE) utasitások (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 beépitésüket).

5. A COPROCESSOR, a SUBPROCESSOR, a COWAIT és a NOWAIT utasitások segit-

$5_{,7} = 1/a_{,}$ ségével multiprocesszoros programok irhatók és futtathatók.

F. A processzor realizálása

1. Szűk subset interpretativ forditóprogramja a MINSZK-22 gépen [5].

2. USASI FORTRAN IV alapú processzor a CDC-3300-as gépen.

   a. A processzor interpretativ módon működik.

   b. Előfordító program (a program külső formájáról belső listastruktu-
   rában történő ábrázolásra való áttérés).

   c. Értelmező és végrehajtó program
   (A belső listastruktúrában ábrázolt program utasitásainak értelme-
   zését és végrehajtását biztositja.)

   d. Az előforditás alatt a szintaktika analizátor részekre bontja a le-
   forditandó utasitást, az egyes részeket külön-külön elemzi, és lefor-
   ditja egy közbenső nyelvre (igy tulajdonképpen az egész utasitást
   leforditja erre a közbenső nyelvre).

   e. A szintaktika analizátor az utasitás közbenső nyelvű ábrázolása a-
   lapján meghatározza az utasitás kódját (vagy esetleg hibajelzést ad).

   f. A szintetizáló program beépiti a belső listastrukturába az utasitás
   kódját és az utasitás paramétereit leiró cimeket (egy-egy paramétert
   általában több cim ir le).

   g. Az értelmező és végrehajtó program belső hierarchiájában az első he-
   lyen a multiprocesszor program áll, amely az egyes processzorok in-
   ditását, felfüggesztését, megszüntetését és futás alatti vezérlését
   irányitja.

h. Egy processzor működését vezérli az EXEC 1 program (az utasitásszámláló tartalma által meghatározott belső sorszámú utasitást értelmezi és végrehajtja, az utasitásszámláló-módositó tartalmát hozzáadja az utasitásszámlálóhoz a következő végrehajtandó utasitás sorszámának meghatározására - ha ez szükséges -, valamint végrehajtja az előirt nyomkövetési akciókat).

i. Egy utasitás végrehajtása során először a CIMSZÁMITÓ program meghatározza az utasitás paramétereit leiró cimeknek megfelelő aktuális cimeket, majd az utasitáskód által meghatározott könyvtári szubrutin kerül végrehajtásra (ez a szubrutin lehet a FORTRAN-könyvtárban, vagy a CHANGE-könyvtárban).

3. A CDC 3300-as processzorban alkalmazott eljárásokról

a. Egyetlen közös szimbólumtábla-kezelő rutin kerül felhasználásra a szöveg tipusú változók kezelésére, az előforditásra, a közbenső nyelvre forditott utasitások kódjának felismerésére, az EXTEND utasitások végrehajtására. (A gazdaságosság illetve áttekinthetőség mellett a processzor esetleges bootstrappelt felépitésére is lehetőséget nyújt ez a megoldás.) Miért?

b. A belső listastruktúra rendkivül tömör

Ha egy utasitás vagy egy paraméter többször is ismétlődik a programban, leirása csak egyszer kerül tárolásra. Ez a megoldás igen hajlékony, a programot módositó CHANGE utasitások végrehajtását megkönnyiti, és általában memórianyereséget tesz lehetővé, mivel az utasitások, illetve a paraméterek leirása a rájuk való hivatkozásoknál átlagosan 5-10-szer több helyet igényel. Ez az ábrázolási mód megkönnyitheti az automatikus szubrutinkészitést. Miért

4. CHANGE processzor kisszámitógépekhez

a. Kisszámitógép esetén nem jelent különösebb hátrányt az interpretativ végrehajtás (abban az értelemben, hogy a közepes vagy magasszintű nyelvű programok végrehajtása általában interpretativ módon történik a szűk utasitáskészlet és a viszonylag kis memóriaméret miatt).

b. A leforditott CHANGE program igen tömör ábrázolása viszonylag nagyobb méretű programok futtatását teszi lehetővé.

c. Ha a kisszámitógép szatellitként (is) működik, akkor a CHANGE processzor feladatai előnyösen feloszthatók a központi számitógép és a szatellit között.

On-line használat esetén a munkamegosztás a következő lehet:
A kisgép egyszerű előfordító-programja a szintaktikusan hibás utasitásokat visszautasitja, a szintaktikusan helyes utasitásokat tömör előforditott formában továbbitja a nagygép felé, ahol a forditás folytatódik, illetve a forditás befejezése után megtörténik a program végrehajtása.

Kész programok használata esetén célszerűnek látszik, hogy a nagygépen egy optimalizáló forditóprogram készitsen el egy viszonylag kis helyigényű leforditott programot a kisszámitógépen való futtatáshoz.

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

5. A 10010 számitógép CHANGE végrehajtó programja

(12)
-11-

a. Az előző (4.) pont a.,b.,c. részében felsorolt tények a 10010 esetén is fennállnak.

b. Rendelkezésre állnak az ALGOL forditóprogram interpretativ végrehajtáshoz készitett rutinjai, amelyek szervezése olyan, hogy viszonylag egyszerűen beépithetők a CHANGE végrehajtó program könyvtár részébe.

c. Igy reális célnak látszik első lépésként végrehajtó program elkészitése. A forditás a CDC 3300-on történhet.

G. A CHANGE nyelv általános jellegű alkalmazása

1. A nyelv kiterjeszthetősége lehetőséget nyújt arra, hogy nem túl bonyolult nyelveket (és egyben processzoraikat is) egyszerűen lehessen definiálni. (Nincs szükség az egyes nyelvekhez szintaktika analizátorok irására, a programok belövéséhez a nyomkövető utasitások segitséget nyújtanak.)
A definiálni kivánt nyelvek "erősen célorientáltak" lehetnek, azaz felépitésük igen közel maradhat a nyelv felhasználási területén szokásos irásmódhoz, mivel a CHANGE processzor végrehajtási módja előirható, és a programok futás alatt módosithatóak.

2. Diszkrét rendszerek szimulálása

A rendszer változó tartománnyal ábrázolt alapelemei a tipuskiterjesztő utasitásokban jellemezhetőek, a rendszer programmal leirt elemeit pedig a kiterjesztő utasitások segitségével egy-egy utasitás ábrázolhatja. Ez az ábrázolás több szinten is történhet, ahogy a rendszer egyre magasabb szintű blokkjai felépülnek.
A blokkok közti kapcsolatot a változótartományon keresztül, esetleg programok segitségével lehet megvalósitani.
A rendszer elemeinek egyidejű működése a párhuzamos műveletvégrehajtást előiró utasitások segitségével történhet.

Például: A rendszer legmagasabb szintű blokkjait egy-egy kiterjesztett utasitás képviseli. A blokkok közti kapcsolat a változótartományon keresztül történik, a blokkok működési sebessége megegyezik. A blokkok működése egyszerre kell, hogy elkezdődjön. Egy külső (fő) program hatására fejezhetik be működésüket.
Az igy megadott feladat realizálható pl. a következőképpen:
A blokkokat képviselő utasitásokat cimkével látjuk el, és egymás után irjuk azokat. A főprogram gondoskodik arról, hogy az egyes utasitásokat ugyanabban az időpillanatban különböző processzorok kezdjék végrehajtani

úgy, hogy utasitásszámláló-módositójuk tartalma 0 legyen, azaz mindig a blokkot képviselő utasitást hajtsák végre (ez az utasitás rendszerint nem vezérlésátadó utasitás). Mivel egyszerre tulajdonképpen csak egy processzor működése indítható el, a tényleges egyszerre való indítás a már elindított processzorok késleltetésével, pl. NULL utasitások végrehajtásával történhet. Az a fontos, hogy a blokkokat realizáló utasitásokra való rálépés történjen egyszerre. (A rálépés előtti utasitás 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 utasitás is. Az IDO változó értékét figyelheti a főprogram, vagy egy további processzor által állandóan végrehajtott feltételes vezérlésátadó utasitás:

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

3. Adaptiv programok (programot készitő ("iró") adaptiv programok)

Egy adott (az adaptiv programban leirt) alap-utasitáskészletből a programot módositó utasitások segitségével 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 leirt működésű időmérést és megszakitást szimuláló, két párhuzamosan futó processzor segitségével).
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 adaptiv program feladata, hogy az ellenőrző programrész által adott eredmény alapján módositsa a generált programot, amig csak szükséges.

4. Makroprocesszorok, listakezelő és szimbólummanipulációs nyelvek

A szövegkezelő (tulajdonképpen szimbólumtábla-kezelő) utasitások és a kiterjesztő utasitások természetes eszközöket adnak makroprocesszorok irására. Az általános változódefinició és a kiterjeszthetőség segitségével a listakezelő utasitások, a karakter és szöveg tipusú változók segitségével a szimbólumkezelő utasitások közelithetőek meg.

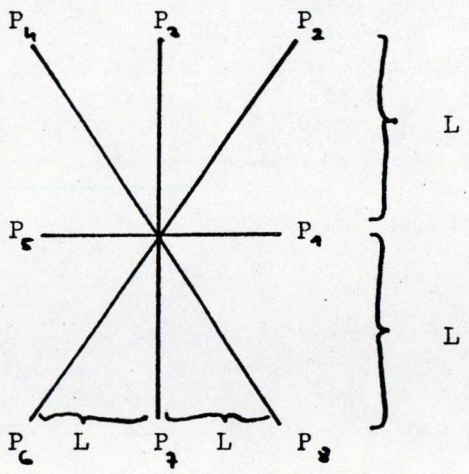
H. A CHANGE nyelv konkrét alkalmazásai

1. NC nyelvek (geometriai/technológiai leiró nyelvek)

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

b. ADMAP postprocesszor (optimalizálással) a CII 10010 számitógépre

Az ADMAP PP feladatát, funkcionális felépitését [6] irja le részletesen. A PP feladata, az ADMAP nyomtatott áramköri lapokat készitő berendezés vezérlő lyukszalagjának elkészitése. A berendezést működtető egyes utasitásoknak 1-1 karakter felel meg a lyukszalagon. Az utasitásrendszer egyszerű, a rajz (fúró stb.) fejet mozgató utasitásokból, toll-fel, toll-le, nagylépésváltás, kislépésváltás és fúrás utasitásokból áll. Mivel a berendezés rajz (fúró stb.) fejét két léptetőmotor hajtja, és ezek egyszerre is működtethetők, a fej egy adott P helyzetből az $U_0, U_1, \ldots, U_8$ utasitások hatására a $P_1, P_2, \ldots, P_8$ pontokba juthat.



$$L = \begin{cases} 0{,}25 \text{ mm ha az utol-} \\ \text{jára végrehajtott} \\ \text{váltóutasitás} & \text{kislépésváltó} \\ 2{,}5 \text{ mm ha az utol-} \\ \text{jára végrehajtott} \\ \text{váltóutasitás} & \text{nagylépésváltó} \end{cases} \quad \text{volt.}$$

A berendezést egy (0,0) kezdőpontból inditva a fej az (n.0,25 mm, m.0,25 mm) koordinátájú pontokba vezérelhető. A toll-fel utasitás hatására a rajzfej a lap (vagy papir) fölé emelkedik (ezután következő mozgató utasitások hatására a levegőben mozog), a toll-le utasitás hatására a rajzfej a lapra (papirra) ereszkedik (ezután következő mozgató utasitások hatására a lapon (papiron) mozog).

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

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

az anyag cime olvasható formában (legible punch),

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

majd az ADMAP által végrehajtandó utasitások.

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

A rajzleirás a lapon található vonalak felsorolásából áll. Az egyes vonalak megadása pontjaik felsorolásával történik, ezen kivül meg kell adni, hogy a vonal hányadik rétegben található (ha többrétegű a lap). Az egyes pontokat koordinátáikkal és technológiai utasitással kell megadni. Ezenkivül célszerű, hogy a rajzleirás tartalmazza a lap azonositóját és méretét.

A technológiai utasitás 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 utasitás(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 rajzleirásból álló adat felfogható, mint a rajzgép programozására szolgáló közepes szintű programozási nyelven irt program.)

Kiterjesztéssel keletkező új utasitások a PP utasitás, a technológiai műveletek, a műveletek vége, a CIM, a KEPMERET utasitás, a technológiai utasitások, valamint a VEGE utasitás.

Minden program a PP utasitással kell hogy kezdődjön. A PP utasitás előirja a végrehajtási módot (ennek lehetséges változatait tartalmazza a leirás további része), és a program szintaktikus szerkezetét (technológiai műveletek, műveletek vége, cim, képméret, technológiai utasitások, rajz vége). Ezután a CIM utasitásra kerül a vezérlés. (Ennek hatására olvasható formában lyukasztásra kerül a program neve.) Ezután a sorrendben következő KEPMERET utasitás végrehajtása következik. (Ennek hatására összehasonlitásra kerül a képméret utasitásban megadott maximális lapméret a technológiai utasitásokban megadott pontok koordinátáival.
- Hiba esetén az összehasonlitás befejezése után a program futása befejeződik - ha nincs hiba, akkor vezérlésátadás történik az első technológiai műveletre.)

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

Az egyes technológiai műveletek végrehajtása a művelet nevének olvasható lelyukasztásával kezdődik. Ezután a művelet tipusa (pont, vonal) szerint az adott művelet során figyelembeveendő pontok illetve vonalak listázása következik. (A technológiai utasitások definiciója tartalmazza, hogy mely technológiai műveletek során kell végrehajtani azokat, és milyen ADMAP uta-

sitássorozat felel meg az egyes technológiai műveleteknek.) A listázás történhet a rajzleirás-rész egyszeri végrehajtásával. Ezután az optimalizáló program [7] (csatolható FORTRAN rutinok) átalakitja a listát az optimális sorrendnek megfelelően. Most következhet a listában felsorolt utasitás(csoport)ok végrehajtása, amelynek során az abszolút értékükkel adott koordináták közti útnak, ADMAP elmozdulás utasitásokat feleltet meg a TRACE utasitásként csatolt ELMOZD FORTRAN rutin (azért TRACE utasitásként, mivel az elmozdulás számitása utasitásközi műveletként értelmezhető), és a technológiai utasitások helyébe ADMAP utasitás(csoport)ok kerülnek.

Az egyes technológiai műveletek végrehajtása során a rajzleirás változatlan maradt, a lista tartalmazta a rajzleirásban szereplő egyes utasitás(csoport)ok végrehajtásának sorrendjét, a lista elemeinek értékét közvetlenül is felveheti az utasitásszámláló (az UTSZ utasitás segitségével). Vonaltechnológiai tipusú műveletek végrehajtásakor előfordulhat, hogy az optimalizálás egy vonalnak megfelelő utasitáscsoport forditott sorrendben való végrehajtását irja elő. (Az UTSZM=-1 utasitás végrehajtásával elérhető a "visszafelé" való végrehajtás.)

Ha nem ragaszkodunk ahhoz, hogy a rajzleirás változatlan maradjon (például a háttérmemóriában tároljuk a rajzleirást és minden technológiai művelet elején újra beirjuk a memóriába), akkor a leválogatás és az optimalizálás az egyes utasitások fizikai átrendezésével is történhet. (Ezt CHANGE utasitások segitségével realizálhatjuk.)

A PP fenti módon történő realizálása viszonylag egyszerű, felhasználhatók a FORTRAN PP egyes rutinjai, az új utasitások definiciója mintegy 100 utasitásból álló programot igényel (külön szintaktika analizátor nem szükséges), egységes közös nyelven kerül leirásra a műveleti lista és a rajzleirás, valamint az ADMAP utasitások is.

2. Dialógus-gépek (dialógus-nyelvek)

A számitógépek interaktiv 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 leirni.

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

A dialógus minden lehetséges lefolyásának leirása 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 leirására szolgáló valamely dialógus-nyelven irni azokat.

Egy dialógus-nyelv fordító (és végrehajtó) programja (proces-szora) programozható közvetlenül valamely programnyelven, vagy valamely általános dialógus-nyelven ("dialógus-processzor összeállitó nyelv"). ez egyálalán nem valósinü, hogy dialógus-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 számitógépen USASI FORTRAN nyelven. A dialógus-programok jelenleg egy ún. direkt kódban (gépi

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

CHANGE nyelven (kiterjesztéssel) egyszerűen felépitheto magasabb szintű bemenő nyelv. Az általános változók használata lehetővé teszi pszeudodinamikus és korlátozott mértékben dinamikus dialógus-programok irását is, a programot módósitó (CHANGE) utasitások lehetővé teszik általános dinamikus dialógus-programok készitését is.

a. A DISTAR-B rendszer

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

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

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

b. Az AIR dialógus-nyelv

Az AIR dialógus-nyelv APT tipusú nyelveken [3] irt programok (párbeszédes) összeállitására készült, általánosabb célokra is használható (alfanumerikus display használatát feltételezi).

Az AIR nyelvű dialógus-programokban le kell irni 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 rögzitett 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 irni, hogy melyik csoportba kerüljön az új szöveg (egy-egy adott dialógus-állapotban).

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

o. Az AIR dialógus-nyelv realizálása CHANGE nyelven

Az AIR processzor központi utasitásai CHANGE nyelven mintegy 10 új kiterjesztett utasitás (kb. 80-90 soros program) segitségével előállithatók (a szöveg tipusú változók és a kezelésükre rendelkezésre álló utasitások, valamint az EXTEND utasitások segitenek ehhez a tömörséghez).

Az AIR/CH nyelven az egyes dialógus-állapotok leirása egy olyan programrésszel történik, amelynek első utasitása egy cimkével ellátott szövegkonstans, vagy szöveg tipusú változó. A cimke felel meg a dialógus-állapot sorszámának, a szöveg a kiadandó kérdésnek.

Ezután annyi utasitás következik, ahány válasz lehetséges (ezen az utasitáscsoporton belül bizonyos esetekben össze lehet vonni egyes utasitásokat).

Menü-rendszerű kérdés esetén a válasz vagy sorcim (annak a sornak a sorszáma, 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 tipusú változó (vagy konstans) határozza meg a választ, az ezután következő cimke jelöli ki az erre a válaszra következő dialógus-állapotot.

Egyéb (nem menü-rendszerű) kérdés esetén az egyes utasitások a válasznak megfelelő szövegkonstansból vagy szöveg tipusú változóból és egy cimkéből állnak (válasz-utasitás).
A cimke adja meg az adott válasz hatására bekövetkező új dialógus-állapotot.
A szöveg tipusu 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 válaszutasitásban a szöveg tipusú változó elé egy NEW csoportazonositót irunk (NEW POINT - új pont, NEW LINE - új vonal,...), ennek az utasitásnak a hatására a dialógus során az ebben a dialógus-állapotban beérkezett válasz lesz a szöveg tipusú változó értéke. Az AIR/CH processzor megvizsgálja, hogy a NEW csoportazonositónak 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 leiró utasitások összevonása lehetséges, ha változó indexes

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

Az egyes dialógus-állapotokat leiró programrészek végére irható egy ERROR utasitás, amelyben meg kell adni azt a cimkét (dialógus-állapotot), ahol a dialógus-program folytatja munkáját hibás válasz esetén. (Ha a programozó nem ir ERROR utasitást a dialógus-állapotot leiró programrész végére, akkor hibás válasz esetén az AIR/CH processzor leállitja 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 számitógép CHANGE/1 fordító-programja
Számitástechnika '71 konferencia

[6] Legendi T.: Az ADMAP nyomtatott áramköri lapokat gyártó berendezés post-processzor programja
Mérés és Automatika 1972.2

[7] Legendi T.: Keresési algoritmus egy speciális metrikus térben
Mérés és Automatika 1971.6

[8] Forgács T. - Krammer G. Distar-B általános dialógusrendszer
AKI Tanulmány 1972.

[9] Pikler Gy.: Minicomputer-based conversational program writing system PROLAMAT '73 IFAC konferencia

[10] Jay Earley - Howard Sturgis: A formalism for translator interactions 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, J. E. L. Peck, C. H. A. Koster: Report on the Algorithmic Language ALGOL 68 Numerishe Mathematik Band 14. Heft 2, 1969. 1969

T A R T A L O M