MAGYAR TUDOMÁNYOS AKADÉMIA
Számítástechnikai Központja

SZELEZSÁN JÁNOS

ELEKTRONIKUS SZÁMOLÓGÉPEK PROGRAMOZÁSA

BUDAPEST, 1963

MAGYAR TUDOMÁNYOS AKADÉMIA
Számitástechnikai Központja

SZELEZSÁN JÁNOS

ELEKTRONIKUS SZÁMOLÓGÉPEK PROGRAMOZÁSA

BUDAPEST, 1963

Központunkat sokan keresik fel azzal a kéréssel, hogy adjunk magyar irodalmat, amelyből a számológépek programozása iránt érdeklődő különböző területen dolgozó szakemberek megismerkedhetnek azokkal az alapelemekkel, amelyek elegendőek ahhoz, hogy e gépek alkalmazási lehetőségeit ezen ismeretek révén áttekintsék. Sajnos könyv ezideig nem jelent meg e szükséglet kielégitésére. Ezt a hiányt igyekszik enyhiteni ez a jegyzet.

A jegyzet első fejezetében az elektronikus számológépek általános alapelveit és a programozás fogalmát, fogásait foglaljuk röviden össze. A második fejezet a hazánkban üzemelő gépek utasitásrendszerét ismerteti és egy egyszerü példán bemutatja az egyes gépeken a program felirásának módját.

A harmadik fejezet a formális gépi nyelvek közül az ELLIOTT-80 gép autókódját, valamint a FORTRAN-nyelvet, ezt követően pedig az ALGOL-60 formális gépi nyelvet ismerteti.

Megemlitjük, hogy a jegyzet teljességre nem tart igényt. Célul csak azt tüztük ki, hogy áttekinthető ismereteket tartalmazzon; a gépek müszaki specifikációival és általában azzal, hogy a gépek hogyan hajtják végre a müveleteket /hogy végzik pl. el két szám összeadását stb./ nem foglalkozik a jegyzet. Erre a programozónak általában nincs szüksége.

Ugy gondoljuk, hogy a jegyzet használható lesz majd tanfolyamok egyetemi előadások segédleteként is.

- Kéziratként -

A kiadásért felel: Frey Tamás mb. igazgató

Alak: A/4 Ivszám: 29,0
Megjelent 1963. julius hóban, 500 példányban

Készült:
az ÉM Épitésügyi Dokumentációs Iroda
Sokszorositó üzemében
F.v.: Szabó György

BEVEZETÉS

A fizikai munka gépesitésén, majd automatizálásán kivül az ember nagy erőfeszitéseket tett a szellemi tevékenység gépesitésére is.

A szellemi munkák közül a történelem folyamán lényegében először a számolást gépesitették.

Ez a folyamat a rendelkezésre álló adatok szerint Napier 1617-ben megszerkesztett mechanikus számolóberendezésével indul meg. Pascal számológépe 1642-ben épül meg, majd 1671-ben Leibnitz létrehozza a mai asztali számológépek ősét. A mai mechanikus (tekerős) és elektromechanikus asztali számológépek lényegében az Odhner által módositott Leibnitz-elv alapján müködnek.

A számológépekkel kapcsolatban meg kell emliteni Jacquard (1801) találmányát is: Jacquard vetette meg alapjait a szövőgépek lyukkártyák segitségével történő vezérlésének.

A modern elektronikus számológépek alapelvei lényegében Charles Babbage-től származnak,aki 1834-54 között megkisérelt egy programvezérelt számolóberendezést létrehozni. A technika akkori eredményei azonban nem tették lehetővé ilyen gép megépitését.

A századfordulón létrejöttek a főleg rendezési és számlálási feladatokat végző lyukkártyás Hollerith-gépek. A század közepén az ilyen tipusu gépekből álló gépparkok segitségével már nemcsak népszámlálási, hanem más statisztikai adat-

feldolgozási, ügyviteli feladatokat végeznek el.

A szorosabb értelemben vett programvezérelt digitális számjegyes számológépek első relékből felépitett modelljei 1940 körül készülnek el; majd 1942-45-ben elektroncsövekből megépitik az ENIAC nevü elektronikus számológépet az USA-ban. A mai számológépek funkcionális alapelve nem olyan mint az ENIAC-é mégis azt lehet mondani, hogy ez a gép tekinthető az első elektronikus számológépnek.

1946-52 között készült el az USA-ban (Cambridge University) az EDSAC nevü gép, amelyben már megtalálhatók mindazok a lényeges vonások, amelyek a programvezérelt digitális számológépeket jellemzik.

A mult évtizedben (1950-60) ugrásszerü változás ment végbe a számológépek gyártása és alkalmazása területén. Az évtized végén már többezer elektronikus számológép üzemelt a világon; több tipust sorozatban gyártottak.

Az 1960-as évektől kezdve az elektroncsöveket mindinkább kiszoritják a félvezetők, a modern mágneses elemek (tranzisztorok stb.). Ez a körülmény egyfelől a gépek stabilitását hibamentes üzemelés növelte meg rendkivüli mértékben, másfelől a gépek méreteit csökkentette le.

Ma a kisérletek a hibamentesség fokozására és a müködési sebesség növelésére irányulnak.

1962-ben már létrehoztak olyan elektronikus számológépeket, amelyek másodpercenként egy millió müveletet végeznek el (Strech; Atlas). Lehet olvasni olyan tervekről is, amelyek több millió müvelet/sec sebességü gépek létrehozására vonatkoznak.

Hazánkban is mindinkább nő az elektronikus számológépek száma.

Az első, szovjet dokumentációk alapján az M.T.A. Számitástechnikai Központjában épitett M-3 nevü gép 1960 óta üzemel. Annak ellenére, hogy kisteljesitményü gép és lényegében teljesen tapasztalatok nélkül épült, sok fontos és uttörő jellegü feladatot oldott meg. Jelenleg a gép sebessége 500 - 1000 müv./sec.

A KFKI-ben, és a TÁKI-ban egy-egy URAL-1 tipusu szovjet gyártmányu gép üzemel. Ezek müveleti sebessége cca 50 müv/sec.

A Nehézipari Minisztériumban 1962 eleje óta egy ELLIOTT-803 tipusu angol gyártmányu gépet üzemeltetnek. Ezidőszerint ez a legkorszerübb gép hazánkban. Müveleti sebessége 1000 - 1500 müv/sec.

A MÁV kezelésében egy Bull-gamma tipusu gépet müködtetnek. Ez a gép főleg adatfeldolgozási feladatok megoldására alkalmas.

Az 1963-as évben előreláthatólag 3 db. URAL-2 tipusu közepes teljesitményü (5000 müv/sec) szovjet gyártmányu gép és egy ujabb ELLIOTT-803 tipusu gép érkezik az országba.

AZ ELEKTRONIKUS SZÁMOLÓGÉPEK ALKALMAZÁSI TERÜLETEI

Kezdetben az elektronikus számológépeket főleg matematikai feladatok megoldására alkalmazták. Az eltelt évtized alatt azonban mindinkább benyomultak a mindennapi életbe. A tudományban, a technikában, a gazdasági életben szinte nélkülözhetetlenné válnak. A szellemi tevékenység széles területe átruházható az elektronikus számológépekre.

Az ürrakéták pályájának kiszámitása elképzelhetetlen elektronikus számológépek nélkül, de hasznos eszközei ezek a gépek az atomfizikának, a meteorológiának, az orvostudománynak, a közgazdasági tudományoknak, sőt a jogtudománynak is. A gépi forditás terén elért eredmények széles távlatokat nyitnak meg; irattak már e gépekkel költeményeket és zeneszerzésre is "megtanithatók". Több egyszerü játékban néha verhetetlen partnerei lesznek az embernek; és pl. jó stratégiát képesek kialakitani a sakkjátékban is.

Segitségükkel létrehozhatók automatikus gyárak is: az ember "vezérlési" funkciója átruházható az elektronikus számológépekre.

Az ügyvitel, az igazgatási feladatok nagyrészénél ugyancsak nagy sikerrel alkalmazzák korunk e nagyszerü vivmányait.

A bérszámfejtés, a könyvelés, a raktárnyilvántartás stb. munkája mind átruházható e gépekre. A gazdasági élet irányitásában optimális terveket készitenek elektromos számológépekkel.

Létrehozhatók már olyan elektronikus berendezések is,amelyeket tanitógépeknek hivnak. Ezek a tanitás munkájában segitik és bizonyos értelemben helyettesitik az embert. Más berendezésekkel a tanulás folyamatát "modellálják".

A felsorolt példák alapján nem teljesen alapnélküli dolog az elektronikus számológépekről mint "gondolkodó gépekről" beszélni.

A matematikának a szellemi tevékenység gépesitésénél centrális szerepe van. A matematika eszközeivel ugyanis nemcsak a mennyiségi kapcsolatok, összefüggések foghatók meg,hanem a minőségiek is. A matematika tudománya mindinkább benyomul a "humán" tudományokba is.

A matematika centrális szerepét ezenkivül az biztositja, a gondolkodási folyamatok gépesitésénél, hogy a gondolati, szellemi tevékenység módszerei,eljárásai, általában ugyancsak leirhatók a matematika nyelvén; az eljárások, módszerek matematikai formába öntött leirásai: az un. algoritmusok viszont már általában jól kezelhetők gépekkel.

A matematika nyelvezete is szélesedik. A gépek megjelenése a gépek és az emberek által egyaránt "érthető" nyelvvel gazdagitja a matematikát. E bővült nyelvvel már szinte minden algoritmus (pl. az egyik nyelvről a másikra való fordítás müvelete, a sakkozás szabályai, matematikai képletek, a könyvelés szabályai, a zeneszerzés szabályai stb.) leirható.

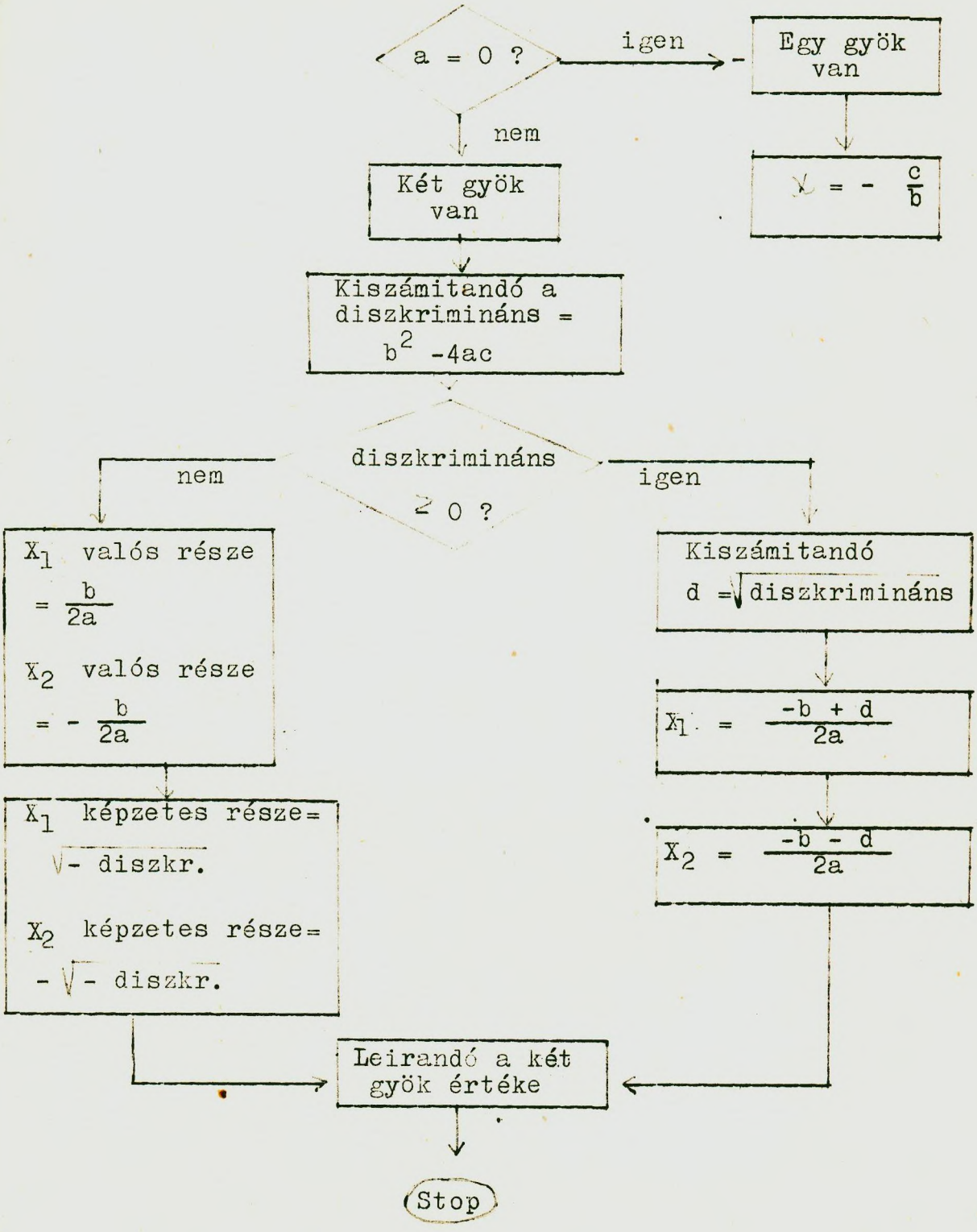
Azokra a gépekre tehát, amelyek a bővült matematikát gépesitik a szellemi tevékenység nagy területei átruházhatók.

ELJÁRÁSOK ALGORITMIKUS FELIRÁSA

Egy számitási eljárás végrehajtása általában elemi lépések egymásutánjából áll. Az elemi lépések kétféle tipusba sorolhatók: az egyik tipusba az un. értékadó (aritmetikai vagy függvényképző müveletek), a másikba a döntési müveletek, (más szóval logikai müveletek) tartoznak.

Az értékadó müveletekkel mennyiségekből mennyiségeket hozunk létre; a döntési müveletekkel valamilyen feltétel teljesülését, vagy az ellenkezőjét állapitjuk meg. A döntési müveletek az eljárásban un. elágazási pontokat jelentenek: attól függően hogy egy-egy ilyen ponton az adott feltétel teljesül-e vagy nem, az eljárást ilyen vagy olyan irányban kell folytatni. Állapodjunk meg abban, hogy az értékadó müveleteket (illetve ezek csoportját) téglalappal; a döntési müveleteket $\diamond$ jellel jelöljük; az eljárás "irányát" pedig nyillal jelezzük. E jelek segitségével szemléletesen irható fel valamely eljárás.

Példaként tekintsük az $ax^2 + bx + c = 0$ másodfoku egyenlet megoldási eljárását az ismert gyökképlet segitségével.


a = 0 ?
igen
Egy gyök van
nem
Két gyök van
x = - c/b
Kiszámitandó a diszkrimináns = b^2 -4ac
diszkrimináns ≥ 0 ?
nem
igen
X1 valós része = b/2a
X2 valós része = - b/2a
Kiszámitandó d = √diszkrimináns
X1 = (-b + d)/2a
X2 = (-b - d)/2a
X1 képzetes része= √- diszkr.
X2 képzetes része= - √- diszkr.
Leirandó a két gyök értéke
Stop

A fentihez hasonló eljárás - leírásokat menetrendnek, vagy blokkdiagrammnak hivjuk.

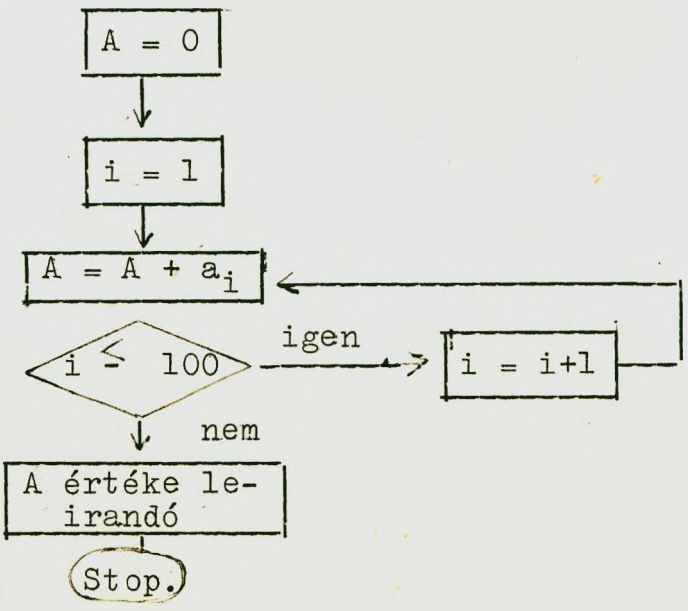
A blokkdiagrammban itt két elágazási pont van: egyik annak a feltételnek a megvizsgálását jelenti, hogy az x^2 együtthatója 0-e vagy nem, a másik pedig a diszkrimináns előjelét vizsgálja meg. A többi lépés értékadó müveletekből áll.

Ciklikus algoritmusok

Az eljárások nagyobb része nem olyan egyszerü, mint a fent leirt. A bonyolultabb eljárások ugyanis olyan részekből állnak, amelyeket többször meg kell ismételni mégpedig ugy, hogy az egyes ismétlések ugyanolyan lépések szerint, de más mennyiségekkel történnek. A legegyszerübb ilyen eljárásra példa az

$$S = \sum_{i=1}^{100} a_i \quad \text{kifejezés képzése.}$$

Az előbb bevezetett jelölésekkel ezen eljárás blokkdiagramja is felirható, a következőképpen:

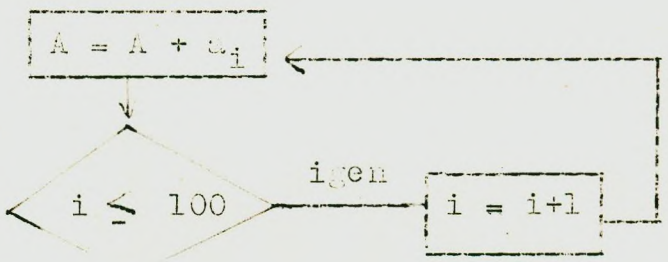


Elemezzük ki a fenti blokkdiagrammot.

Az első értékadó lépés A-nak 0 értéket; a második pedig az i indexnek 1 értéket ad. A harmadik értékadó lépés az A értékhez hozzáadja az i indexű a_i értéket, ezáltal az első végrehajtáskor az A értéke $0 + a_1$ lesz. A következő pont egy elágazási pont megvizsgálja, hogy az index elérte-e már a megadott felső határt, 100-at vagy nem.

Ha a feltétel teljesül, azaz $i \leq 100$ akkor i értéke eggyel módosul (azaz az első végrehajtáskor 2 lesz) és az $A = A + a_i$ eljárás megismétlődik az $i = 2$ értékre, majd így tovább egészen addig, amíg az i értéke el nem éri a 100-at, amikor is az eredmény leírása után az eljárásnak vége van.

A fenti eljárás



részét <u>ciklusnak</u> hívjuk.

A ciklus miatt látjuk egy <u>ciklusszervezőtől</u> (i) függő, amely meghatározza, hogy a <u>ciklus magját</u> (a "belső részt") hányszor kell megismételni. A vizsgálat mindig valamilyen döntési művelettel történik.

A ciklusok természetesen magukban is tartalmazhatnak ciklusokat. Többszörös ciklusra példa az

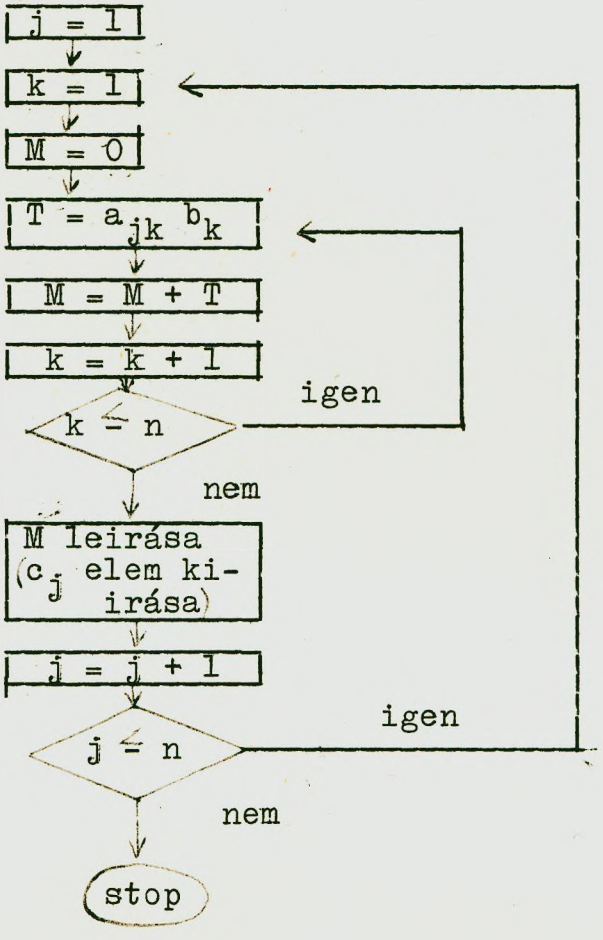
$$\underline{c} = \underline{\underline{A}}\,\underline{b}$$

szorzás elvégzése ahol

$$\underline{\underline{A}} = \begin{bmatrix} a_{11} & a_{12} \cdots\cdots & a_{1n} \\ \vdots & & \\ a_{n1} & a_{n2} \cdots\cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{jk} \end{bmatrix}$$

$$\underline{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Az algoritmus blokkdiagram segitségével a következő módon irható fel:



A fenti blokkdiagram két ciklust tartalmaz, mégpedig ugy, hogy az egyik /külső/ "burkolja" a másikat /belső/.

A blokkdiagram elemzését az előző példa alapján az olvasóra bizzuk.

Az elektronikus számológépek programozása lényegében nem jelent mást, mint az előzőekhez hasonló módon felépitett algoritmusoknak a gép nyelvén /a gép utasitásai segitségével/ történő felirását.

Ahhoz, hogy a blokkdiagramok alapján valamely gép szavai, a gép "nyelvezete" felhasználásával un. programokat irhassunk fel, ismerkedjünk meg azokkal a legfontosabb fogalmakkal, amelyek ehhez szükségesek.

I. fejezet

1. AZ ELEKTRONIKUS SZÁMOLÓGÉPEK MÜKÖDÉSI ELVE

1.1 Az információ ábrázolásának módja az elektronikus számológépekben

Az elektronikus számológépek nagy többsége diszkrét müködésü, azaz az információt (számok, betükből álló szavak stb.) egymástól egyértelmüen elkülönithető jelek kombinációi hordozzák; minthogy ezek a jelek egyuttal számjegyek is, ezért a szóbanforgó berendezéseket digitális elektronikus számológépeknek is hivják. (Digit angol szó, jelentése: számjegy.)

Az elektronikus számológépek általában kettes számrendszerben dolgoznak; ez azt jelenti, hogy az információt két jellel ábrázoljuk.

A "két jelből álló jelrendszer" kifejezés helyett azért használjuk a kettes számrendszer kifejezést, mert az elektronikus számológépekben ábrázolt minden információ száminformáció.

Mint ismeretes kettes számrendszerben egy A számot:

$$A = \sum_{i=-n}^{+m} \alpha_i \, 2^i \quad \text{alakban}$$

állithatunk elő, ahol $\alpha_i = 1$, vagy 0 (a 0-t és 1-et bit-nek hivjuk "binary digit"-ből).

Példaként felirjuk az első tiz tizesszámrendszerbeli számot a számjegyeket kettes számrendszerben:

10-es számrendszer	2-es számrendszer
1	$1 = 1.2^0$
2	$10 = 1.2^1 + 0.2^0$
3	$11 = 1.2^1 + 1.2^0$
4	$100 = 1.2^2 + 0.2^1 + 0.2^0$
5	$101 = 1.2^2 + 0.2^1 + 1.2^0$
6	$110 = 1.2^2 + 1.2^1 + 0.2^0$
7	$111 = 1.2^2 + 1.2^1 + 1.2^0$
8	$1000 = 1.2^3 + 0.2^2 + 0.2^1 + 0.2^0$
9	$1001 = 1.2^3 + 0.2^2 + 0.2^1 + 1.2^0$

1.2 Müveletek a kettes számrendszerben

A müveletek a kettes számrendszerben is számjegyeken végzett müveletekből állnak. A számjegyeken végzett müveleti szabályok azonban igen egyszerüek.

Szorzótábla:

1 . 1 = 1
1 . 0 = 0
0 . 1 = 0
0 . 0 = 0

Összeadó tábla:

1 + 1 = 0 és maradt 1 átvitel, azaz 1 + 1 = 10 (a kettőn azaz tizen felüli számot leirjuk és egyet átviszünk a következő helyértékre)
1 + 0 = 1
0 + 1 = 1
0 + 0 = 0

A számokon a fentiek alapján végezhető müveleteket példákon mutatjuk be.

a/ Összeadás

Eljárás: jobbról balra haladva: egy plusz egy egyenlő kettővel (10-zel); leirjuk a nullát és marad egy átvitel; egy + + egy = 10 (kettő), ehhez hozzáadva a maradék egyest: 11; leirjuk az egyet (a "tizen" felüli számot) és marad egy átvitel stb.

```
  110011111
+  1001111
  111011110
     10
```

Ugyanigy tört esetén:

```
   11011,1101
+  10011,0111
  101111,0100
```

b/ Kivonás

Eljárás: jobbról balra haladva: egyből kivonva egyet, az eredmény 0 (hiszen 1+0=1). A következő lépésben 0-ból 1-et ugy vonunk le, hogy a magasabb helyértékről egy egységgel a 0-t 10-re (kettőre) egészitjük ki, igy tizből (kettőből) egyet kivonva az eredmény: 1 (1+1 = 10) és marad egy kivonandó átvitel, amit a kivonandó következő helyértéken lévő jegyéhez hozzáadunk és az igy nyert számot vonjuk ki a kisebbitendő soron következő jegyéből stb.

```
  11011001
-  1101111
  01101010
```

Tört szám esetén hasonló módon járunk el:

```
1100101,11101
  11011,01111
1001010,01110
```

Szorzás

```
 11011101 x 1100l
 11011101
  11011101
   00000000
    00000000
     11011101
1010110010101
```

Eljárás:

ha a szorzó adott helyértékén 1-es áll, akkor a szorzandót változatlanul leirjuk (hiszen: 1x1=1, 1x0=0), ha 0 áll, akkor 0-át irunk. Természetesen a "helyértékeltolásokat" ugyanugy, mint a tizes számrendszerben, végrehajtjuk és a részeredményeket összeadjuk. A 0 végeredmények kiirására nincs szükség, de a részeredmények eltolásánál figyelembe kell venni.

Ugyanigy járunk el a törtszámok esetében is; a "tizedesjegyek" száma, akárcsak tizes számrendszerbeli számoknál, a tényezőkben szereplő "tizedesjegyek" számának összegével egyenlő.

Osztás

Az osztás müvelete a kettes számrendszerben is kivonási müveletek egymásutánja:

```
11010111101:101 = 101011001
- 101
  00110
  - 101
    00111
    - 101
      0101
      - 101
        000101
        - 101
          000
```

Eljárás:

leválasztjuk az osztandó egy olyan részét, amelyből az osztó már kivonható (amellyel az osztó osztható). Kivonjuk az osztót, a maradékot leirjuk, s a hányadosban egy egyest irunk fel. A maradékhoz hozzávesszük az osztandó soron következő helyértékén lévő jegyét. Ha ebben az osztó megvan (a kivonás elvégezhető), akkor az előbbi módon járunk el, ha nincs, akkor a hányados soron következő helyértékére 0-t irunk, s leir-

juk a következő számot stb. Tört esetén hasonló módon járunk el.

1.3 Tizes számrendszerből kettes számrendszerbe való átirás

1.31 Egész számok átalakitása

A legegyszerübb eljárás a következő: a számot elosztjuk a legmagasabb, a számban még meglévő kettő-hatvánnyal, hányadosként 1-et irunk, majd a maradékot megkiséreljük elosztani az eggyel alacsonyabb kettő-hatvánnyal. Ha az osztás nem végezhető el, akkor a soron következő kettő-hatványt vesszük, és az előbb nyert 1 mellé 0-t irunk; ha elvégezhető, akkor azt 1-sel jelöljük stb.

$$\text{Pl.} \quad 149_{(10)} = 10010101_{(2)},$$

mert

$$\begin{aligned} 149 &: 128 = 1 \\ 21 &: 64 = 0 \\ 21 &: 32 = 0 \\ 21 &: 16 = 1 \\ 5 &: 8 = 0 \\ 5 &: 4 = 1 \\ 1 &: 2 = 0 \\ 1 &: 1 = 1 \end{aligned}$$

Fenti eljárás egy kis módositásával a következő séma alapján végezhetjük el az átalakitást:

a/ Elosztjuk a számot 2-vel; a maradékot leirjuk (1 vagy 0). A hányadost ujra elosztjuk kettővel, a maradékot leirjuk stb.

b/ A nyert maradéksorozat az utolsó hányadossal együtt, (fordított arányban olvasva az utolsó jegy lesz az első) adja a 10-es számrendszerbeli szám kettes számrendszerbeli megfelelőjét.

Pl 135 : 2

① 67 : 2

① 33 : 2

① 16 : 2

⓪ 8 : 2

⓪ 4 : 2

⓪ 2 : 2

⓪ 1

Tehát: $135_{(10)} = 10000111_{(2)}$

1.32 Törtszámok átírása

Adva: $A = 0, a_1\, a_2 \ldots a_n$ (10-es számrendszerben),

akkor $0, \alpha_1\, \alpha_2 \ldots \alpha_n$ számjegyeit az

$$\alpha_k = \left[2b_{k-1}\right] \; ; \; \left\{b_k = 2b_{k-1}\right\} ; \quad b_0 = A$$

eljárással kapjuk, ahol $[\;]$ jellel a szám egész részét, $\{\;\}$ jellel törtrészét jelöljük.

Pl. $\underset{(10)}{\quad} = \quad$

$0{,}8125 = \; ?$

Fenti eljárásunk szerint:

0,	8125 x 2
1	6250 x 2
1	2500 x 2
0	5000 x 2
1	0000

Megszorozzuk a számot 2-vel; a nyert eredmény egész része (1) lesz az első bináris számjegy. Az eredmény törtrészét ujra megszorozzuk 2-vel stb. Tehát:

$0{,}8125_{(10)} \qquad 0{,}1101_{(2)}$

(Nyilvánvaló, hogy ha az átalakitandó szám nem kettőhatvány, akkor az eljárás nem véges; a kivánt pontosságig folytatjuk).

1.4 Kettes számrendszerből tizes számrendszerbe való átirás

Legegyszerübb mód az, hogy $A = \pm \sum_{k=n}^{-m} a_k \, 2^k$ felirás szerint elvégezzük a müveleteket. Pl.:

$$1101{,}101 = 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-3} =$$

$$= 8 + 4 + 1 + \frac{1}{2} + \frac{1}{8} = 13 + 0{,}5 + 0{,}125 = 13{,}625$$

1.5 Tetszőleges információ ábrázolása a kettes számrendszer segitségével

A számitástechnikai berendezések nemcsak számitási, hanem szinte tetszőleges információ feldolgozási feladatok elvégzésére alkalmasak (gépi forditás, egy gyár teljes adminisztrációja, komplex rendszerek vezérlése, sakkozás stb.) A kettes számrendszer segitségével ugyanis - mint ezen gépek "ABC"-jén - tetszőleges információ felirható. Egyrészt a betük - ezáltal a szavak, s ezen keresztül tetszőleges fogalmak kifejezhetők, másszóval kódolhatók a kettes számrendszerrel (két jel segitségével!). A betük például egyszerüen megszámozhatók, s a számok kettes számrendszerben felirhatók.

Pl:

a = 1	azaz	a = 1
b = 2		b = 10
c = 3		c = 11
d = 4		d = 100
e = 5		e = 101 stb.

Megállapodás szerint tetszőleges jelkombináció is hozzárendelhető egy-egy betühöz (vagy szóhoz). Pl.: a = 11001 is lehet, vagy asztal = 101001110.

A kettes számrendszer (két jel) segitségével másfelől logikai összefüggések is kifejezhetők, hiszen ismeretes, hogy állitás - tagadás, igen - nem (1-0 kombinációkkal, illetve ezen kombinációkon végzett müveletekkel a logikai kapcsolatok leirhatók. Tudjuk, hogy a logikai függvények kétértékü függvények, a logikai müveletek a kettes számrendszer müveleteivel összefüggésbe hozhatók.

Ezzel magyarázható, hogy az információfeldolgozás technikájában általában a kettes számrendszert (két jel segitségével történő kódolás) használják.

1.6 A kettes számrendszerben ábrázolt információ papiron történő felirása

Az elektronikus számológépek kettes számrendszerben dolgoznak; ezért az információt kettes számrendszerben kellene felirni. Papiron azonban igen nagy munkát jelent a számok kettes számrendszerben történő felirása. Ezért ugy járnak el, hogy az információt nyolcas számrendszerben irják fel. A nyolcas számrendszerben felirt információt könnyü átalakitani kettes számrendszerbe: minden nyolcas számrendszerbeli számjegy 3 bitnek felel meg. Ezt az átalakitást gépi feldolgozáskor már a gép bemenő egysége végzi.

Pl. a

111 010 101 110 011

bináris szám nyolcas számrendszerbeli alakja

7 2 5 6 3

annak megfelelően, hogy

$$111 = 1.2^2 + 1.2^1 + 1.2^0 = 7$$
$$010 = 0.2^2 + 1.2^1 + 0.2^0 = 2$$
$$101 = 1.2^2 + 0.2^1 + 1.2^0 = 5$$
$$110 = 1.2^2 + 1.2^1 + 0.2^0 = 6$$
$$011 = 0.2^2 + 1.2^1 + 0.2^0 = 3$$

Az ismertetendő gépek nagy részénél az információt 8-as számrendszerben kell felirni; ezen gépek utasitásai nyolcas számrendszerben felirt számok.

1.7 A kettes számrendszer "instrumentálása" az elektronikus számológépekben

A kettes számrendszer két jelét az elektronikus számológépekben két állapotu elemekkel, illetve jelenségekkel realizálják. A mai gépek elektronikus jelzője arra utal, hogy a szóbanforgó két állapotu elemek elektronikusak. A gépek nagy részében feszültségimpulzusok jelenléte vagy hiánya hordozza az 1-es ill. 0 jelet. Az elektronikus számológép lényegében nem más mint egy bonyolult áramkör. Ennek az áramkörnek az a feladata, hogy a feszültségimpulzusokat ugy "terelje" egyik pontról a másikra, hogy a gép elvégezhesse azokat a müveleteket, amelyeket logikai strukturája meghatároz. A logikai áramkör elektroncsövekből (illetve tranzisztorokból) valamint más elektromos elemből (kondenzátor, ellenállás stb) áll.

A tárolási, "emlékezési" funkciónál bizonyos mágneses tulajdonságokat használnak fel (remanens mágnesség). Egy elem (pl. kis speciális alaku és összetételü "vasmag") vagy valamilyen felület egy pontjának mágnesezett volta 1-est jelent; ha nincs mágnesezve akkor 0-t.

A kettes számrendszerben végzett müveletek a gépben lényegében impulzusokkal végzett müveletek.

2. AZ ELEKTRONIKUS SZÁMOLÓGÉPEK STRUKTURÁLIS FELÉPITÉSE

Az elektronikus számológépek általában az alábbi részekből állnak.

1. Memória egység
2. Aritmetikai egység
3. Bemenő egység
4. Kimenő egység
5. Vezérlő egység

2.1 Memória egység

A memória a feladat megoldásával kapcsolatos összes információ tárolására szolgál. Müszaki szempontból a memóriák általában a mágneses permanencia elve alapján müködnek.

Az országban jelenleg müködő elektronikus számológépek memóriái az alábbi tipusuak:

a/ Ferrit memória

E memóriánál a biteket a kettes számrendszerbeli jegyeket kisméretü ferritgyürük hordozzák; ezeket a gyürüket megfelelően "huzalhálózatba" épitik be.

A ferritmemóriák igen gyorsak.

b/ Dobmemória

A dobmemória lényegében egy mágneses réteggel bevont henger. Az információt mágneses jelekkel a henger palástjára ir-

ják fel. A henger fordulatszáma szabja meg az "emlékezés" gyorsaságát, az un. hozzáférési időt. Pl. az M-3 dobja 3000 ford/perc sebességgel forog.

c/ Szalag memória

A szalag memóriák lassu, de nagy kapacitásu memóriák. Szerkezeti elvük hasonló a magnetofonéhoz.

2.2 Aritmetikai egység

Az aritmetikai egység a müveleteket végzi el a kettes számrendszer müveleti szabályai alapján. Általában un. regiszterekből áll. Az M-3-ban pl. 4 regiszter van. (A, B, C, D) Az URAL gépekben és az ELLIOTT 803 gépben az egyik regiszternek kitüntetett szerepe van; ezt a regisztert akkumulátornak hivják.

2.3 Bemenő egység

Az elektronikus számológépekbe az információt általában lyukszalag segitségével lehet bevinni. Az információt perforátorral lyukszalagra lyukasztják. A bemenőberendezés pl. fotócellák segitségével letapogatja a lyukkombinációkat és ezeket impulzuskombinációkká alakitva a memóriába viszi át. Némely gépnél bevitel közben a gép a bevitt információról kontrollösszeget képez. A bevitel jóságának ellenőrzése ugy történik, hogy a szalagot kétszer visszük be. Ha jó a bevitel, a kontrollszámnak egyeznie kell.

A gépek nagy részének, igy például az M-3 és az ELLIOTT 803 gépeknek a beviteli egysége alfanumerikus. Ezen azt értjük, hogy ezeknél a gépeknél a számokat és betüket is be lehet vinni. (Más szóval: a betüknek megfelelő lyukkombinációkat /kódokat/ a gép megkülönbözteti a számjegyek kódjától.)

2.4 Kimenő egység

A kimenő egység a digitális számológépeknél valamilyen nyomtatóberendezés. Azokat a kiiró berendezéseket, amelyek számokat és betüket is kiirnak, alfa-numerikusoknak hivjuk. (Az M-3 gép, az ELLIOTT 803 kimenőegysége alfa-numerikus.)

2.5 A vezérlő egység

A vezérlő egység a gép automatikus vezérlését végzi. A programozás szempontjából részletesebben nem kell ismerni müködését.

3. A PROGRAMOZÁS FOGALMA

Az előzőkben röviden szóltunk az elektronikus számológépek felépitéséről.

A következőkben tekintsük át a programozásukat.

3.1 A címezhetőség elve

Mint emlitettük, minden információ a gép memóriájában tárolódik: ha a memóriát "megtöltöttük" megfelelő információkkal, akkor a feladatot a gép a továbbiakban automatikusan oldja meg, mégpedig ugy, hogy az egyes müveletek elvégzésére vonatkozó parancsokat, az un. utasitásokat, valamint a müveletek operandusait, az adatokat a memóriából viszi át a vezérlőegység bizonyos regisztereibe, ill. az aritmetikai egységbe és a müveletek eredményeit ugyancsak a memóriába irja vissza.

3.2 A szó fogalma

A memóriában az információt rögzitett nagyságu "kvantumokban", egységekben tároljuk; más szóval azt mondjuk, hogy

az információ szavakra bontva található a memóriában. Pl. az M-3-ban egy szó 31 kettes számrendszerbeli helyértékből (bitből) áll. A szó hosszán a szó bitjeinek számát értjük: eszerint az M-3 szóhossza 31 bit.

3.3 A rekesz fogalma

A szavakra bontott információt szavanként külön-külön tároljuk; egy szó helyét a memóriában rekesznek nevezzük. Egy olyan szó tárolása, amelynek hossza 31 bit, olyan rekeszben történik, amely 31 bit reprezentálására képes. A rekeszek tartalma egy szó.

3.4 A cim fogalma

A memória tehát rekeszekre van osztva. A rekeszeket egymástól számozással különitjük el. Minden rekesz kap egy egyszersmindenkorra rögzitett sorszámot: ezt a sorszámot címnek hivjuk. Egy-egy rekeszre tehát cimével lehet hivatkozni. A rekeszek sorszámozása 0-val kezdődik. Beszélhetünk tehát a memória 0-adik; tizedik, száztizenhetedik, ezredik stb. cimü rekeszéről.

A memóriában elhelyezett minden szóhoz tehát egy cim tartozik; annak a rekesznek a cime, amelyben az adott szót tároljuk.

A memória kapacitásán a benne tárolható szavak maximális számát értjük. Az M-3 gép kapacitása pl. 2048 szó, az ELLIOTT-803 gépé pedig 8192 szó.

4. A PROGRAM

Az elektronikus számológépek automatikusak: más szóval programvezérlésüek. Ez azt jelenti, hogy előre meg lehet adni a gép számára a megoldandó feladat algoritmusát, azaz a gép

nyelvén le lehet irni a megoldási eljárást. Egy feladat gépi programja utasitásokból áll.

4.1 Az utasitás fogalma

Egy gépi utasitás az alábbiakról tartalmaz információt:

a/ A memória, mely rekeszeiben találhatók azok a számok, amelyekkel az adott müveletet el kell végezni; pontosabban szólva az utasitás tartalmazza az operandusok cimét. Külön kihangsulyozzuk, hogy az utasitások nem az operandusokat magukat, hanem azon rekeszek cimeit tartalmazzák, amelyekben az operandusok elhelyezkednek. Egy-egy utasitás több cimre hivatkozhat egyidejüleg: eszerint lehet a gép egycimü, két cimü stb. Az M-3 gép kétcimü, az URAL gépek és az ELLIOTT-803 egycimüek.

b/ Jelölni kell az utasitásban azt is, hogy milyen müveleteket kell elvégezni a gépnek az utasitásban szereplő cimek tartalmaival: ezt az utasitás müveleti kódja jelöli.

A müveleti kód jelenthet valamilyen aritmetikai müveletet, de jelenthet más tipusu müveletet is: pl. olyant, hogy egy rekesz tartalmát át kell vinni egy másik rekeszbe.

Egy két cimü utasitás általános alakja a mondottaknak megfelelően:

m	a	b

ahol m a müveleti kód, a az egyik cim (első cim) b a másik cim (második cim). A konkrét utasitások felirásánál az m, a, b, betük helyére természetesen számokat kell irni. Pl.

32 0613 0714

azt jelenti, hogy a gép ezen számkombináció hatására a 613-as cimü (sorszámu) rekesz és a 714-es cimü rekesz tartalmával

elvégzi azt a müveletet, amit a 32-es müveleti kód a gép számára előir.

4.2 Utasitás-tipusok

Az utasitásokat az elvégzendő müveletek alapján csoportokba sorolhatjuk.

a/ Aritmetikai utasitások

Ezen utasitások hatására a gép az utasitásokban szereplő cimek tartalmain valamilyen aritmetikai müveletet végez (összeadás, kivonás, szorzás, osztás).

b/ Átviteli utasitások

Az átviteli utasitások arra használhatók, hogy a memória valamely rekeszének a tartalmát egy másik rekeszbe vigyük át.

c/ Logikai utasitások

Ezen utasitások az utasitásban szereplő cimekhez tartozó rekeszek tartalmain valamilyen logikai müveletet végeznek. (Pl. a binárisan felirt két szám bitjeit rendre az 1x1=1, 1x0=0, 0x0=0 szabály alapján összeszorozza.)

d/ Vezérlési utasitások

Ha az utasitásokat egymásután következő rekeszekbe irjuk be a memóriába és a gépet elinditjuk, akkor a gép az utasitásokat egymásután, rekeszről-rekeszre haladva hajtja végre. (Kiemeljük, hogy az utasitások, mint számok a memória meghatározott rekeszeiben vannak elhelyezve. Minden gép utasitásai között vannak azonban olyan utasitások is, amelyek feladata valamilyen feltételtől függően vagy függetlenül a "vezérlést" átadják; azaz ha a gép egy vezérlési utasitást talál, akkor a következő utasitást nem a soronkövetkező rekeszből veszi,

hanem egy olyan rekeszből, amelynek cime a vezérlési utasitásban szerepel.

e/ Beviteli utasitások

Az információ bevitele a gépbe történhet utasitással is. A beviteli utasitások hatására a gép a beviteli berendezésbe elhelyezett szalagról beolvas egy szót, vagy egy jelet a memória azon rekeszébe, amelynek cime a beviteli utasitásban szerepel.

f/ Nyomtatási utasitás

A nyomtatási utasitás hatására a gép a kiiró berendezés segitségével kiirja annak a rekesznek a tartalmát, amely a nyomtató utasitásban előfordul.

g/ Mágnesszalag utasitások

Ezen utasitásokkal a mágnesszalag memóriából lehet számokat átvinni a dob vagy ferrit memóriába és forditva.

h/ Megállási utasitás

Ezen utasitás hatására a gép abbahagyja a számolást, megáll.

Egy gép utasitásainak összességét utasitásrendszernek nevezzük. Egy-egy gépi utasitásrendszer általában 64 különböző utasitásból áll. ($64 = 2^6$; azaz 6 kettes számrendszerbeli jegyet szoktak az utasitás kódja részére fenntartani.)

5. AZ UTASITÁSOK ÉS SZÁMADATOK KÓDOLÁSÁNAK MÓDJA

5.1 Számok ábrázolása

a/ Fixpontos ábrázolás

Az elektronikus számológépek a számok ábrázolásának módjától függően kétfélék lehetnek, un. fixpontos gépek és lebe-

gőpontos gépek.

A fixpontos gépekben a tizedes "vesszőt" rögzitik. Általában ugy járnak el, hogy a gépben egy $a = \alpha_1 \alpha_2 \ldots \alpha_n$ jelkombinációt

$$a = \pm(\alpha_1 2^{-1} + \alpha_2 2^{-2} + \ldots + \alpha_n 2^{-n})$$

alaku számként tekintik, azaz a tizedes "vesszőt" közvetlenül az előjel után teszik ki, ami tehát azt jelenti, hogy ilyen esetben csak egynél abszolut értékben kisebb számok ábrázolhatók. Ha egy gép szóhossza pl. 31 bit, és az előjel részére - balról számitva - az első bitet tartjuk fenn, akkor egy szó alakja az alábbi lesz:

+ −	2^{-1}	2^{-2}	2^{-3}	.....	2^{-30}

előjel a szám számjegyei

Eszerint ha pl. a memória valamely rekeszében

előjel számjegyek

0101101 101

alaku információt tárolunk, akkor - amennyiben az számot ábrázol - ez az

$$+ 1.2^{-1} + 0.2^{-2} + 1.2^{-3} + 1.2^{-4} + 0.2^{-5} + 1.2^{-6} \ldots\ldots$$
$$\ldots\ldots 1.2^{-28} + 0.2^{-29} + 1.2^{-30}$$

értéket jelenti. (A 0 jel a +, az 1- jel a - jelet jelenti az előjel helyértéken.)

Megjegyzések

a/ Fixpontos ábrázolás esetén a megoldandó feladatot ugy

kell transzformálni, hogy minden bemenő adat, részeredmény és eredmény egynél abszolutértékben kisebb legyen (ez a lépték megfelelő választásával sokszor megoldható).

b/ Ha valamely müvelet (összeadás, osztás) elvégzése közben egynél nagyobb eredményt kapunk, akkor ezt a jelenséget balról történő tulcsodulásnak nevezzük. (Az átvitel balra csuszik ki).

c/ Ha a szorzat 2^{-30}-nál kisebb, akkor jobboldali tulcsordulásról beszélünk.

d/ Könnyü észrevenni, hogy a fixpontosan ábrázolt számok az összeadás és a kivonás müveletére nézve egészeknek is tekinthetők. Bármely, pl. 30 bitből álló szám, olyan egészként kezelhető, amelynél az egység 2^{-30}. Ha tehát pl. 233-at kell összeadni 321-el, akkor ezt ugy kell tekinteni, mint a $233.2^{-30} + 321.2^{-30} = 554.2^{-30}$ müvelet eredményét. A programok felirásainál ez kiemelendő körülmény, mert az utasitásokat lényegében egész számoknak tekintjük: az utasitásokon végzett müveletek, egész számokon végzett müveleteket jelentenek.

5.2 Lebegőpontos ábrázolás

Egy A számot bináris lebegőpontosan ábrázolunk, ha

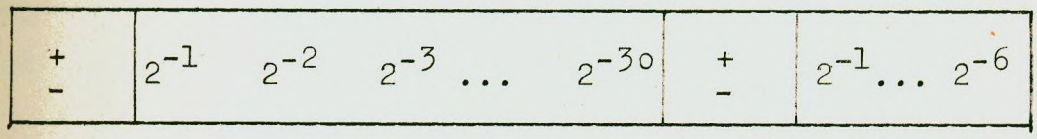
$A = q\ 2^p$ alakban állitjuk elő. Itt a q mantissza és a p kitevő is bináris számok.

Ha $\frac{1}{2} \leq q < 1$, akkor a számot normalizáltnak hivjuk, ekkor mantissza első, az előjel után következő jegye egyes.

Amikor a lebegőpontosan ábrázolt számot gépben reprezentálják a következőképpen járnak el.

A szót két részre osztják. Az első részben tárolják (fixpontosan) az előjellel ellátott mantisszát, a második

részben pedig az előjeles kitevőt. Ha például feltesszük, hogy gépünk szóhossza 38 bit és 31 bitet a mantisszára, 7 bitet a kitevőre hagyunk, akkor egy lebegőpontos szó alakja a következő lesz:



mantissza előjele | a mantissza számjegyei | kitevő előjele | a kitevő jegyei

A gép az igy ábrázolt számokkal az aritmetikai szabályok szerint végzi el a müveleteket: pl. két lebegőpontosan adott szám szorzásánál összeszorozza a mantisszákat és összeadja a kitevőket. (A gép a szó mantissza részével és kitevő részével külön-külön végzi el a müveleteket.)

A lebegőpontos gépek aritmetikai egységét általában ugy szerkesztik meg, hogy a müveletek elvégzése után a gép az eredményt normalizálja is, azaz olyan alakra hozza, hogy a mantissza első jegye 1 legyen. (A mantisszát balra tolja és ahány bittel balra tolta, annyival módositja a kitevőt.)

A lebegőpontos gépek strukturája általában bonyolultabb. Az ábrázolható számok tartománya természetesen itt is korlátozott: ha a kitevő előjel nélkül p bitből áll, akkor az ábrázolható legnagyobb szám 2^p-nél nem lehet nagyobb.

6. EGY FIKTIV GÉP UTASITÁSRENDSZERE

A következőkben abból a célból, hogy az utasitásrendszerek alkalmazását, a programozás fogásait bemutathassuk, felirunk egy fiktiv gépet (nem konkrét gép) illetve egy fiktiv utasitásrendszert. Nevezzük a gépet F gépnek.

6.1 Az F-gép utasitásrendszere

a/ Összeadás

00 a b

ez az utasitás azt jelenti, hogy összeadja a gép az a cimü és a b cimü rekeszek tartalmát és az eredményt beirja a b rekeszbe.

b/ Kivonás

01 a b

Jelentése: vonja ki a gép a b cimü rekesz tartalmából az a rekesz tartalmát és az eredményt irja be a b cimü rekeszbe.

c/ Szorzás

02 a b

Jelentése: szorozza össze a gép az a rekesz tartalmát a b rekesz tartalmával és az eredményt irja be a b rekeszbe.

d/ Osztás

03 a b

Jelentése: ossza el a gép a b rekesz tartalmát az a rekesz tartalmával és az eredményt irja be a b rekeszbe.

e/ Átvitel

04 a b

Jelentése: vigye át az a rekesz tartalmát a b rekes be (az a rekesz tartalma nem változik).

f/ Feltételes ugrás

05 a b

Jelentése: a vezérlést adja át a gép az a cimü utasitásra, ha az előző müvelet eredménye negativ és a b cimre, ha ez az eredmény pozitiv.

g/ Nyomtatás

06 0 b

Jelentése: a b rekesz tartalmát nyomtassa ki a gép a kinyomtatóberendezéssel.

h/ Bevitel

07 0 a

Jelentése: a lyukszalagon lévő számot vigye be a gép a memória a cimü rekeszébe.

i/ Abszolut értékképzés

10 0 a

Jelentése: képezi az a rekesz tartalmának abszolut értékét.

j/ Megállás

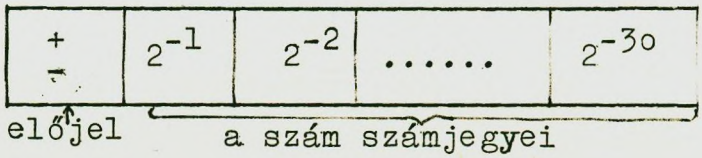
11 0 0

Jelentése: a gép álljon meg (a számitást fejezze be).

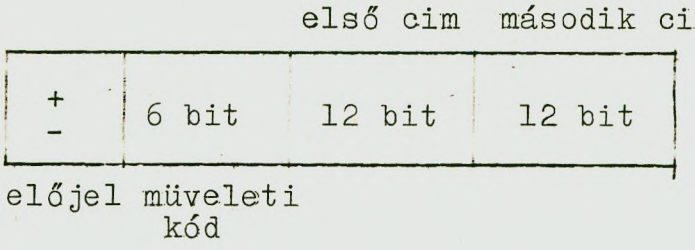
6.2 Az információ ábrázolása az F gépben

Tegyük fel, hogy gépünk fixpontos és szóhossza 31 bit; szószerkezete legyen a következő:

Szám ábrázolása:



Utasitás ábrázolása



7. FELADATOK PROGRAMOZÁSA AZ F UTASITÁSRENDSZER SEGITSÉGÉVEL

Készitsük el a fenti utasitásrendszer segitségével az alábbi képlet kiszámitásának programját.

$$f(x,y,a,b) = \frac{0{,}5\,x^3 + y\,(a^2 - b)}{0{,}7\,a^3 - xy^2}$$

(Az egyszerüség kedvéért feltesszük, hogy a kiinduló adatok, a részeredmények és a végeredmény is abszolutértékben egynél kisebbek.)

Mielőtt a programot felirnánk, néhány programozási elnevezést ismertetünk.

7.1 Rekesz tartalma

A megállapodás szerint jelöljük egy rekesz tartalmát a ( ) jellel. A (0020) = 0,5 jelölés tehát azt jelenti, hogy a 0020-as rekesz tartalma 0,5.

7.2 Rekeszelosztás

A programnak és az adatoknak a memóriában történő elhelyezése a rekeszek előzetes elosztása alapján történik: ez azt jelenti, hogy előre eldöntjük, hogy a memória mely rekeszeibe kerülnek az adatok, melyekbe a program utasitásai.

Állapodjunk meg például abban, hogy a fenti feladatnál az alábbi lesz a rekeszelosztás:

(0100) = a (0101) = b

(0102) = x (0103) = y

(0104) = 0,5 (0105) = 0,7

Tegyük fel, hogy ezek az adatok a memóriába nyolcas számrendszerben felirva valamilyen módon már bekerültek.

Helyezzük el a programot a 0200-as cimü rekesztől kezdve.

7.3 Munkarekeszek

A program felirása során szükség van olyan rekeszekre, amelyekben a részeredményeket és az eredményeket tároljuk. Ezeket munkarekeszeknek hivjuk.

Legyenek példánk esetén a munkarekeszek az alábbiak:

0106
0107
0110
0111

Programunk az alábbi lesz:

0200	04	0102	0106	x
0201	02	0102	0106	x^2
0202	04	0106	0107	x^2
0203	02	0102	0106	x^3
0204	02	0104	0106	$0,5\ x^3$
0205	04	0100	0110	a
0206	02	0100	0110	a^2
0207	04	0110	0111	a^2

0210	01	0101	0110	a^2 - b
0211	02	0103	0110	y (a^2 - b)
0212	00	0110	0106	számláló
0213	02	0100	0111	a^3
0214	02	0105	0111	0, $7a^3$
0215	02	0103	0107	x^2 . y
0216	01	0107	0111	nevező
0217	03	0111	0106	f (x,y,a,b)
0220	06	0000	0106	
0221	11	0000	0000	

Elemezzük ki a fenti programot. A program a 200-as utasitással indul (azaz. azzal az utasitással, amely a 200-as cimü rekeszben van).

Ez az utasitás átviszi (04) a 102-es rekesz tartalmát (x-et) a 106-os munkarekeszbe. A 201-es utasitás hatására a gép a 102-es rekesz tartalmát (x-et) megszorozza a 106-os rekesz tartalmával (x-el) és az eredményt beirja a 106-os rekeszbe (ezáltal /0106/ = x^2 . A 202-es utasitás átviszi x^2-et a 107-es rekeszbe, majd a 203-as utasitás a 102-es rekesz és a 106-os rekeszek tartalmait összeszorozza, azaz képezi $x^2 . x = x^3$-t. A 204-es utasitás hatására a 106-os rekesz tartalmát a gép megszorozza a 104-es rekesz tartalmával; ezáltal a 106-os rekesz tartalma 0.5 $(^3$ lesz.

A 205-ös utasitás átviszi a -t a 110-es munkarekeszbe, majd a 206-os képezi a^2-et ami a 207-es utasitás révén a 111-es rekeszbe kerül át. A 210-es utasitás kivonást végez: képezi az a^2 - b különbséget, amit - mint a 110-es rekesz tartalmát a gép megszorozza a 211-es utasitás hatására y-al.

A 212-es utasitás összeadja a 110-es és a 106-os rekesze

tartalmát: azaz képezi a $0,5\ x^3 + y\ (a^2 - b)$ összeget és ezt beirja 106-ba.

A 213-216 utasitások a nevezőt hozzák létre a 0111-es rekeszben. A 217-es utasitás elosztja a számlálót a nevezővel és az eredményt a 106-os rekeszbe teszi. Ennek a rekesznek a tartalmát (vagyis az eredményt) nyomtatja ki a 220-as utasitás, majd a 221-es utasitás hatására a gép a számitást befejezi.

7.4 Elágazásos programok

Mint már emlitettük, az algoritmusok nagy része az értékadó müveleteken kivül döntési müveleteket is tartalmaz. A döntési müveletek az algoritmusban elágazásokat jelölnek ki.

Az elágazási pontban valamilyen feltétel teljesülését kell megvizsgálni. Egy-egy ilyen pont logikai szerkezete a következő:

Ha F teljesül, akkor: A

Ha F nem teljesül, akkor: B

A programban ennek az felel meg, hogy összehasonlitjuk két rekesz tartalmát: pl. kivonjuk egymásból a két számot s ha az eredmény negativ, akkor az egyik, ha nem negativ, akkor a másik reláció teljesül.

Az F feltétel általában valamilyen összehasonlitás; valamilyen logikai relációmüvelet: $=$; $<$, $>$, $\leq$ stb .

Példaként készitsük el az alábbi feladat programját:

$$f(x, y, z) = \begin{cases} x^2 + y & \text{ha } x \geq y \\ y - 0,5\ y^2 & \text{ha } x < y \end{cases}$$

Rekeszelosztás:

Kiinduló adatok:

0100 = x

0101 = y

0102 = 0,5

Munkarekeszek:

0200 eredményrekesz

0201

Program:

1000	04	0100	0200	
1001	01	0101	0200	x - y képzése
1002	05	1007	1003	$\Rightarrow$ elágazási pont
1003	04	0100	0200	
1004	02	0200	0200	
1005	00	0101	0200	$x^2 + y$
1006	05	1017	1017	átadja a vezérlést 1017
1007	04	0100	0200	
1010	02	0101	0200	xy
1011	04	0101	0201	
1012	02	0201	0201	
1013	02	0102	0201	
1016	01	0201	0200	$xy - 0,5\,xy^2$
1017	10	0000	0000	

A program elemzését a megjegyzések segitségével az olvasóra bizzuk.

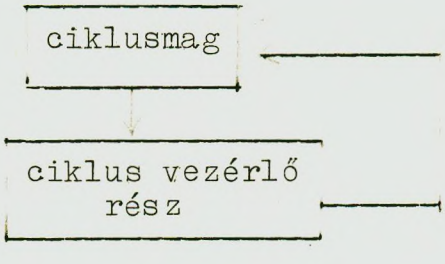
7.5 Ciklikus programok

Mint az eddigiekben láttuk, a programok olyan utasitásokból állnak, amilyen müveleteket a feladat végrehajtása

céljából el kell végezni. Nyilvánvaló, hogyha nem lenne alkalmazható valamilyen "fogás" aminek segitségével elérhető, hogy ne kelljen előre leirni minden elvégzendő müveletet, akkor az elektronikus számológép alkalmazása alig járna valamilyen előnnyel, hiszen például egymilliárd utasitást leirni gyakorlatilag annyi, mint ugyanezt el is végezni, elektromos asztali számológéppel.

A matematikai algoritmusok azonban általában olyan tipusuak, hogy ciklikusan egy bizonyos müveletsort kell elvégezni egymásután sokszor; az ilyen eljárásokat ciklikus eljárásoknak hivjuk. A ciklikus eljárásokat ciklikus programmal valósithatjuk meg.

A ciklusok valamilyen paramétertől függnek: ez a paraméter határozza meg, hogy a ciklus "belsejét", a ciklus "magját" hányszor, vagy milyen feltétel teljesüléséig kell ismételni. A ciklusos program ennek megfelelően két részből áll: a ciklusmagból és a ciklusvezérlő részből.



Változó utasitások

A ciklusok felépitésénél ugy járunk el, hogy a ciklus egyes utasitásait lépésenként (ciklislépés) változtatjuk: azaz a magban szereplő utasitások egy része un. változó utasitás lesz.

A mondottakat példával világitjuk meg.

Legyen a megoldandó feladat az

$$S = \sum_{i=1}^{64} a_i$$ összeg kiszámitása.

Ciklus nélkül a feladatot ugy lehet megoldani, hogy leirunk száz összeadó utasitást.

Tegyük fel, hogy az a_i számokat az alábbi módon helyeztük el:

$(1000) = a_1$

$(1001) = a_2$

.

.

.

$(1000+77) = a_{100}$ (77 nyolcas számrendszerben értendő: tizes számrendszerben ez: 63)

$(0000) = 0$

Munkarekeszek: 0010

Ekkor ciklus nélkül az alábbi program épithető fel:

				Megjegyzés
0100	04	0000	0010	$0 \Rightarrow$ 0010
0101	00	1000	0010	$a_1 + 0 \Rightarrow$ 0010
0102	00	1001	0010	$a_i + a_2 \Rightarrow$ 0010
0103	00	1002	0010	$a_1+a_2+a_s \Rightarrow$ 0010
0104	00	1003	0010	$a_1+a_2+a_3+a_4 \Rightarrow$ 0010
0105	00	1004	0010	
0106	.			
0107	.	s.i.t.		

Mint látjuk: a 0010-es rekeszbe beirtunk 0-t és ebben a rekeszben gyüjtöttük össze az összeget. Észrevehetjük, hogy a 0101, 0102, ... utasitásokban csak az első cim változik, mégpedig mindig eggyel lép tovább.

Ugy fogható fel a dolog, hogy a 00 1000 0010 utasitás változik középen eggyel minden lépésnél. Ezt az utasitásmódositást egy másik utasitással is el lehet végezni a következő módon:

0100	04	0000	0010	
0101	00	1000	0010	változó utasitás
0102	00	0004	0101	

Itt (0004 = 00 0001 0000 középen egy egyes a lépésköz).

A 0100-as utasitásnál elindulva, a 0101-es utasitást először végrehajtva, ott

00 1000 0010 áll;

azaz hozzáadja a 0010-es rekesz tartalmához az 1000-es rekesz tartalmát (képezi $a_1 + 0$ -t es beirja 0010-be).

A 0102-es utasitás viszont a 0101-es utasitást megnöveli középen 1-el (hozzáadja a 00 0001 0000 konstanst) igy ebből

00 1001 0010 lesz.

Ha most a vezérlést visszaadjuk a 0101-es utasitásra, akkor ez most a

00 1001 0010 müveletet hajtja végre (azaz képezi $a_1 + a_2$ -t és beirja 0010-be.)

A 0102-es utasitás ujra hozzáad a 0101-hez egy egyest miáltal ebből

00 1002 0010 lesz.

Ha ujra visszaugrunk erre a megváltozott utasitásra, akkor most már a gép ezt

00 1002 0010 alakban hajtja végre.

s.i.t.

Természetesen ezzel még a program nincs kész, mert nincs ami számlálja a szükséges visszaugrások számát. Az előbbi ciklusmagot tehát el kell látni vezérlőrésszel is.

Ebből a célból tegyük fel, hogy a 0011-es rekeszbe elhelyeztük az ismétlések számát meghatározó nyolcas számrendszerben felirt 77-es számot (tizesben 63), mégpedig a következő alakban:

(0011) = 00 0000 0077

Helyezzük el ezenkivül az 1-es számot (1.2^{-30}-at) a 0003-as rekeszbe, azaz legyen

(0003) = 00 0000 0001

Épitsük fel ezzel a két rekesszel a 0103-as utasitást az alábbi módon

0103 01 0003 0011

Ez az utasitás a 77-es számból kivon 1-et, igy első végrehajtása után

(0011) = 00 0000 0076

áll elő.

Ahányszor ezt az utasitást végrehajtjuk, annyival csökken a 0011-es rekesz tartalma.

Legyen a 0104-es utasitás a következő:

0104 05 0105 0101

Ez a vezérlést a 0101-es utasitásra adja át, ha az előző müvelet eredménye pozitiv: azaz mindaddig, amig a 0011-es rekesz tartalma nem csökken le -1-ig; tehát pontosan 100-szor (nyolcasban).

Programunk tehát a következő lesz:

```
0100    04    0000    0010
0101   (00    1000    0010)
0102    00    0004    0101
0103    01    0003    0011
0104    05    0105    0101
0105 . . . . .
```

A felirt program még nem teljes. Ha egyszeri végrehajtás után meg akarjuk ismételni (vagy a végrehajtás közben pl. géphiba miatt félbeszakad), akkor a 0101, és a 0011 rekeszek tartalmát vissza kell állitani. Általában a változó utasitásokat, illetve paramétereket ciklikus programoknál valamilyen módon vissza kell állitani.

A visszaállitásnál célszerübb az előre történő beállitás. Esetünkben ezt ugy oldjuk meg, hogy programot a következő két utasitással kezdjük:

```
0076    04    0001    0011
0077    04    0002    0101
```

ahol a

```
(0001) = 00    0000    0077
(0002) = 00    1000    0010
```

konstansok un. programkonstansok.

Ez a két utasitás a 0001-es és a 0002-es rekeszek tartalma segitségével (ezeket a tartalmakat a programmal együtt visszük be) előre beállitja a 0011 és a 0101 rekeszbe a szükséges információt mindannyiszor, ahányszor a programot itt kezdjük.

A teljes program tehát a következő lesz:

0076	04	0001	0011	beállitó rész
0077	04	0002	0101	
0100	04	0000	0010	
0101	00	1000	0010	ciklusmag
0102	00	0004	0101	
0103	01	0003	0011	vezérlőrész
0104	05	0105	0101	
0105	11	0000	0000	

Programkonstansok:

(0000)	=	00	0000	0000
(0001)	=	00	0000	0077
(0002)	=	00	1000	0001
(0003)	=	00	0000	0001
(0004)	=	00	0001	0000

Munkarekeszek:

0010 ; 0011 ;

A fenti példa alapján lerögzithetjük: a ciklikus eljárásokat végrehajtó ciklikus program tartalmaz:

a/ beállitórészt
b/ ciklusmagot
c/ vezérlőrészt

A ciklus lépéseinek számlálása a ciklusparaméter segitségével történik.

Az elektromos számológépek nagy részében a ciklus sokkal könnyebben végrehajtható, mert ezek a gépek rendelkeznek un. ciklus-utasitással is.

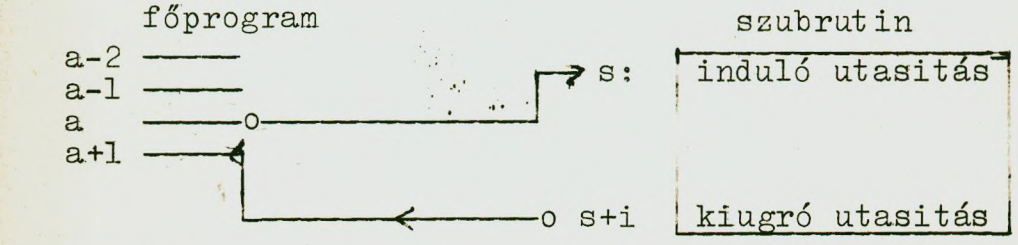
(Lásd a konkrét gépek utasitásrendszerét.)

7.6 A szubrutinok fogalma

A feladatok megoldása közben gyakran előfordul, hogy pl. trigonometrikus függvények, az ln függvény, vagy más elemi függvény értékére, vagy pl. egyenlet megoldó eljárásra van szükség. Ezért egy sor eljárásra célszerü mindenkor felhasználható programokat készíteni és ezeket "könyvtárba foglalni". Egy gyakran használt és csak paraméterektől függő, "alprogramot" szubrutinnak hivunk. Szubrutin lehet például az egyenletrendszert eliminációval megoldó program, amennyiben ugy müködik, hogy ha bizonyos rekeszekbe beirjuk az egyenletrendszer együtthatóit, valamilyen erre a célra kijelölt rekeszbe a rendszámot, akkor bizonyos előre kijelölt rekeszben megkapjuk a megoldás-vektort.

A szubrutin tehát lényegében egy standard program, amit a főprogramban "be lehet hivni". A behivás módja az alábbi:

Minden szubrutinnak van un. bemenete. Ez nem más mint munkarekeszek csoportja, ahova a szubrutin argumentumát (paramétereket) kell beirni. (Pl. sin x szubrutinnál az x argumentum rekesze . A szubrutin kimenetén értjük azt a rekeszt (vagy rekeszeket), ahol a szubrutin végrehajtása után az eredmény megjelenik. (Pl. az a rekesz, ahol sin x értéke adódik). A szubrutin induló utasitása az az utasitás, amelynek végrehajtásával kell kezdeni a szubrutint. A kiugró utasitás helye az a rekesz, ahová a szubrutin utolsó végrehajtott utasitás után a vezérlést átadjuk. Az egy argumentumu szubrutin behivás az alábbi ábra szerint történik:



Munkarekeszek: e = bemenet (argumentum helye)
e+1 = kimenet (eredmény helye)

Az a-2, a-1 cimü utasitással gondoskodunk arról, hogy az argumentum bekerüljön az e rekeszbe és arról, hogy az s+i rekeszbe pedig egy olyan utasitás kerüljön, amely visszaadja a vezérlést az a+1 cimü utasitásra. Miután ezekről gondoskodtunk, az a cimü utasitással a vezérlést a szubrutin s induló utasitására adjuk át.

A szubrutin (mert ugy épitettük fel) kiszámitja pl. az adott elemi függvényt és az eredményt elhelyezi az e+1 rekeszbe, majd a vezérlést visszaadja az a+1 -es utasitásra.

Példaként készitsünk el egy négyzetgyökvonó szubrutint. Az $0 \leq a < 1$ szám gyökét az alábbi (Newton féle) iterációs eljárással számitsuk ki.

$$x_n = \frac{1}{2}\left(\frac{a}{x_{n-1}} + x_{n-1}\right)$$

Feltesszük, hogy tulcsordulás nem lép fel.
A programot nem konkrét, hanem un. betücimekkel épitjük fel, pl. (Minden rekeszt valamilyen betüvel, ill. egy betüalaphoz viszonyitott számmal jelölünk, pl. a+20; b+3 stb.)

Rekeszelosztás:

Konstansok:

(c) = 0,5 (c+1) = 00 0000 0001 (2^{-30})

Munkarekeszek:

(p) = x_0 (az első közelités; pl. 1)
(p+1) = a (a szubrutin bemenete)
(p+2) = $\sqrt{a}$ (a szubrutin kimenete)

k	04	p	p+2	
k+1	04	p+1	m	
k+2	03	p+2	m	$\frac{a}{x_{n-1}}$

k+3	00	p+2	m	
k+4	02	c	m	$\frac{1}{2}\left(\frac{a}{x_{n-1}} + x_{n-1}\right)$
k+5	01	m	p+2	$\left.\begin{array}{l}\\\\\\\end{array}\right\}$
k+6	10	0000	p+2	$\left(x_n - x_{n-1}\right) \leq \varepsilon$
k+7	01	c+1	p+2	ahol $\varepsilon = 2^{-30}$
k+10	05	k+13	k+11	
k+11	04	m	p+2	
k+12	05	k+1	k+1	
k+13	05	m	p+2	
k+14	(kiugró utasitás)			

Ez a program szubrutinként müködhet, mert ha a p+1 rekeszbe elhelyezzük az a számot és gondoskodunk arról, hogy a k+14 utasitás kiugró utasitás legyen, akkor a k cimü utasitásra adva át a vezérlést, megkapjuk a p+2 rekeszben $\sqrt{a}$ - t.

Készitsük el példaként a szubrutin behivására a következő feladat programját az előbbiekben felirt fiktiv utasitásrendszer segitségével:

$$f(x, a, b) = 0{,}5\, x^2 + \sqrt{a^2 + 2a + b^2}$$

Az egyszerüség kedvéért feltesszük, hogy minden adat és részeredmény egynél abszolutértékben kisebb.

Rekeszelosztás:

Kiinduló adatok:

(0100) = x
(0101) = a
(0102) = b
(0103) = 0,5

Munkarekeszek:

0200
0201
0202

Gyökvonó szubrutin:
k - k + 14-ig
bemenet: p+1 eredmény: p+2

Program konstans:

(0104) = 05 1017 1017

A program:

1000	04	0100	0200	x
1001	02	0200	0200	x^2
1002	02	0103	0200	$0,5\ x^2$
1003	04	0101	0201	a
1004	02	0201	0201	a^2
1005	04	0101	0202	a
1006	02	0100	0202	ax
1007	03	0103	0202	$\frac{ax}{0,5} = 2ax$
1010	00	0202	0201	$a^2 + 2ax$
1011	04	0102	0202	b
1012	02	0202	0202	b^2
1013	00	0202	0201	$a^2 + 2ax + b^2$
1014	04	0201	p+1	(gökszubrutin be-
1015	04	0104	k+14	hivása)
1016	05	k	k	$\sqrt{a^2 + 2ax + b^2}$ ké
1017	00	p+2	0200	$f(x, a, b)$
1018	01	0000	0000	

A program elemzését az olvasóra bizzuk.

Az első fejezetben a teljességre egyáltalán nem tartva igényt, röviden összefoglaltuk azokat a tudnivalókat, amelyek szükségesek ahhoz, hogy a konkrét gép utasitásrendszerének ismeretében programot irjunk fel. A gépek nagy részének utasitásrendszere flexibilisebb, mint az itt leirt F - utasitásrendszer, ezért a bemutatott programok a konkrét gépeken esetleg gazdaságosabban épithetők fel.

A következő fejezetben konkrét gépeket ismertetünk. Javasoljuk az olvasónak, hogy az e fejezetben felirt programokat irja majd át a konkrét gépek utasitásrendszerével.

II. fejezet

AZ M-3, ELLIOTT-803, URAL-1, URAL-2 ELEKTRONIKUS SZÁMOLÓGÉPEK RÖVID ISMERTETÉSE

Az előző részben egy "fiktiv" kétcimü gép utasitásrendszere segitségével ismerkedtünk meg a programozás alapelemeivel. A következőkben néhány (hazánkban is előforduló) gép konkrét utasitásrendszerét ismertetjük meg az olvasóval.

Mielőtt azonban erre rátérnénk, néhány általános megjegyzést teszünk a konkrét programok felirásával, gépre való előkészitésével és a gépen történő futtatásával kapcsolatban.

A programok felirásának módja

Miután a programozandó algoritmust a programozáshoz teljesen előkészitettük, a következő feladat a program felirása. Általában a programokat betücimekkel épitjük fel. Ezekkel a program könnyebben áttekinthető s az átcimzés (beszurás, törlés, változtatás esetén) sokkal könnyebben megvalósitható. Amikor a betücimes program végleges alakot nyer (többszöri átnézés után), a programot kódoljuk: azaz a betükhöz konkrét cimeket rendelünk hozzá.

A kódolás a rekeszelosztással kezdődik: megszabjuk a program utasitásainak, a programkonstansoknak, a munkarekeszeknek és a kiinduló adatoknak a helyét a memóriában.

A konkrét rekeszek cimezése általában nyolcas számrendszerben történik (az ELLIOTT-803 gépnél tizes számrendszer-

ben): az igy felirt információ ugyanis lényegében minden átalakitás (átkódolás) nélkül kerülhet be a memóriába abból eredően, hogy az ilyen rendszerben történő felirás voltaképpen kettes számrendszerbeli felirást jelent azzal a különbséggel, hogy a bit-eket hármasával összefogjuk.

Ha gépünkben egy adott utasitás valamelyik cimrésze 12 bitből áll, akkor ez 4 nyolcas számrendszerbeli számjegyet jelent: e cimre tehát 0000-tól 7777-ig terjedő nyolcas számrendszerben felirt számok irhatók. (Az értéktelen nullákat is mindig ki kell irni; a 67-es sorszámu rekeszre tehát a fenti esetben a 0067 számmal kell hivatkozni.) Mint ismeretes, a nyolcas számrendszerben a "kilencesnek", a 7-es szám felel meg: erre a cimek sorszámozásánál ügyelni kell (kezdetben nehéz megszokni). Pl. 0000; 0001, 0002, 0003, 0004, 0005 0006, 0007, 0010, 0011, 0012, ..., 0017, 0020, 0021, ..., 0076, 0077, 0100, 0101, 0102, ..., 0776, 0777, 1000, 1001, ...

Nemcsak az utasitásokban szereplő cimek kódjai, hanem a müveletek kódjai is nyolcas számrendszerben felirt számok. Ha a müveleti kód 6 bit, akkor ez két nyolcas számrendszerbeli számjegy. Egy két cimü (cimenként 12-bites) utasitás alakja tehát mint már ismeretes, pl.:

23	0617	0002
müveleti kód	első cim	második cim

Ha ezt az utasitást a memória 1312-es számu rekeszébe helyezzük el, akkor az utasitásról szóló teljes információt a következő alakban irjuk fel:

utasitás cime	müveleti kód	első cim	második cim
1312	23	0617	0002

A programot lyukasztás és a gépbe való bevitel céljára általában ilyen számok sorozataként kell felirni.

Számadatok felirása

Az algoritmus kiinduló adatai általában tizes számrendszerbeli számok. Ezeket nyolcas, ill. kettes számrendszerbe kell átirni. Ez megtörténhet a gépbe történő bevitel előtt is: "kézzel" átkonvertáljuk az adott számot nyolcas számrendszerbe. A konvertálás a kettes számrendszerbe konvertáláshoz hasonlóan egynél kisebb számokra az alábbi rekurziv eljárással történik:

$$\alpha_k = \lfloor 8\, b_k \rfloor, \qquad b_k = \{8\, b_{k-1}\}, \quad b_0 = a$$

Pl.: $\overset{10}{a} = 0{,}358298$

0	3582987 x 8
2	8663896 x 8
6	9311168 x 8
7	4489344 x 8
3	5914752 x 8
4	7318016 x 8
5	8544128
.	
.	
.	

$\overset{8}{a} = 0{,}267345...$

A kézzel történő konvertálást ritkán, esetleg néhány szám esetén szokás alkalmazni.

Általában a konvertálást magával a géppel végeztetik el.

E célra un, input szubrutinokat készitenek.

A gépbe az adatokat ugyanis egy kapcsoló állástól függően vagy nyolcas számrendszerben (bináris rendszerben), vag bináris kódolt decimális rendszerben lehet bevinni. A decimálisan lyukasztott számokat a beviteli berendezés átkódolja, bináris-decimálissá alakitja át: igy kerülnek be a gépbe (a memóriába), ez utóbbit programmal alakitják át bináris rendszerüvé.

Kiirásnál ugyanez történik forditott irányban.

Az információ lyukasztása, bevitele

Az információt általában lyukszalagra perforálják: a perforátor mindenegyes lenyomott billentyühöz a megfelelő lyukkombinációt viszi rá a szalagra. (Az M-3 gépnél pl. telex-perforátorral telex-kódban történik a lyukasztás: egy jelet (számjegyet, betüt) öt lyukhelynek megfelelő lyukkombináció reprezentál: a bevitel "ötcsatornás" lyukszalaggal történik. A beviteli berendezések az öt bittel ábrázolt jelet 4 bitessé kódolják át, ha tizes számrendszerbeli számokat vittünk be: a hét számjegy mindegyike ugyanis négy bittel is ábrázolható, de a telex-kódban a számjegyek nem bináris kódban, hanem pl. Baudot-kódban vannak ábrázolva, ahol pl.:

6 = y = 10101 23 = b = 10011

Némely gépnél lyukkártyával is történhet a bevitel.

A tizes számrendszerben felirt számokat (adatokat) a kétféle bevitel miatt általában külön szalagra perforálják. (Külön készülnek az adatszalagok és a programszalagok.) A nyolcas számrendszerben felirt számok természetesen rákerülhetnek a programszalagra (pl. programkonstansok stb.).

A perforáló berendezések nagy része alfanumerikus: ez

azt jelenti, hogy rajtuk betük is perforálhatók: egy betüváltó billentyü különbözteti meg a kétféle irás módját. A beviteli berendezések általában alkalmasak arra is, hogy betükódokat is bevigyenek: ilyenkor a gépbe az adott betü kódja átalakitás nélkül 5 bittel kerül be a memóriába.

A gépek nagy részénél a vezérlőpultról is lehet a memória adott rekeszeibe információt beirni. A cimet a vezérlőpult cim-regiszterén kell beállitani, a beviendő számot pedig egy, erre a célra szolgáló regiszteren: az "irás" gombbal (kapcsolóval) megtörténhet a beirás. Az ilyen tipusu beirást természetesen csak "javitásra", módositásra használjuk.

Kiirás

A kiirás általában nyomtató berendezések segitségével történik. A kiiró berendezések egyidejüleg lyukszalagra perforálhatják is a kiirt információt.

A kiiróberendezés is lehet alfanumerikus, vagy csak szám kiirására képes. Előző esetben tetszőleges táblázatok készithetők (szöveg kiirásával is), utóbbi esetben meghatározott számu számjegyből álló számok nyomtathatók csak ki (pl. 1-2-3 oszlopos kiirás).

Számok ábrázolása

Az ismertetendő gépek fixpontos gépek; illetve az ELLIOTT-803 és az URAL-2 gép rendelkezik un. "bekötött lebegőponttal" is. (Ez azt jelenti, hogy a lebegőpontos müveletek egy permanens memóriában rögzitett program segitségével hajthatók végre.)

A fixpontos számokat az

$$a = \pm \alpha_1\alpha_2\alpha_3\ldots\alpha_n = \pm \alpha_1 2^{-1} + \alpha_2 2^{-2} + \alpha_3 2^{-3} + \ldots + \alpha_n 2^{-n}$$

alakban előállitottaknak tekintjük, ahol $\alpha_i = 0$, vagy 1. Szám-

lálás céljára általában a szó végén elhelyezkedő számokat használjuk: egy m egész számláló konstansot, tehát $m.2^{-n}$ alakunak tekintünk, ha a gép legkisebb helyértéke 2^{-n} -nek felel meg. Ha fixpontos müveletek végzése közben olyan eredmény jönne létre, amely a legnagyobb az előbbi módon ábrázolható számnál nagyobb, akkor azt mondjuk, hogy tulcsordulás lép fel.

Lebegőpontos számolás fixpontos gépekkel is végrehajtható. Ilyenkor a számokat az $x = p\,2^q$ alaknak megfelelően ábrázoljuk, mégpedig vagy egy szóban a mantisszát és a kitevőt is, vagy külön rekeszbe irjuk a mantisszát és az utána következő rekeszbe a kitevőt. A lebegőpontos müveleteket külön, un. lebegőpontos szubrutinnal hajtjuk végre. Természetesen ez általában lényegesen megnöveli egy-egy művelet végrehajtási idejét.

Az elektronikus számológépek vezérlőpultja

Az elektronikus számológépek kivülről a vezérlőpult segitségével vezérelhetők. A pulton nyomógombok és kapcsolók segitségével pl. az M-3 gépen lehetőség van valamely cim tartalmának a pult egy regiszterébe történő kiolvasására, vagy valamely információnak a memória egy adott rekeszébe való beirására. A vezérlőpulton kell beállitani általában az első végrehajtandó utasitás cimét is, itt helyezkedik el az inditó kapcsoló, amellyel a program végrehajtását el lehet inditani stb.

Regiszterek

A programozás szempontjából a gép néhány regiszterének fontos szerepe van. E regisztereket a gépek ismertetésénél mindig megemlitjük. Az ELLIOTT-803, az URAL-1 és az URAL-2 gépben kitüntetett szerepe van az un. akkumulátornak: mivel ezen gépek egycimüek, ezért utasitásaik általában az akkumulátorra és egy megadott cimre vonatkoznak.

Jelölések

A következőkben ismertetendő gépek utasitásrendszerével kapcsolatban az alábbi jelöléseket alkalmazzuk:

Az 'a jelentése: az a cimü rekesz tartalma

Az 'A jelentése: az A regiszter pl. az akkumulátor tartalma

A => jel jelentése: a jel baloldalára irt érték átmegy a jobboldalon szereplő rekeszbe, ill. regiszterbe

(Pl. 'a + 'b => B felirás azt jelenti, hogy az a cimü rekesz tartalma és a b cimü rekesz tartalma összeadódik és az eredmény bekerül a B regiszterbe.)

'a => a jelentése: a tartalma nem változik

Amikor külön ki akarjuk hangsulyozni, hogy az adott rekesznek, ill. regiszternek a müvelet végrehajtása előtti tartalmáról van szó, akkor ezt indexbe irt e betüvel jelöljük; pl. $'a_e$. Az egyes utasitások elnevezése után zárójelben odairjuk az adott utasitás rövid jelölését is (pl. Kiirás /K/).

A programok felirásánál az utasitások végrehajtása eredményének jelölésére is alkalmazzuk majd a fenti jelöléseket; de a => jel baloldalára magukat a mennyiségeket és nem azok cimét irjuk. Pl. $\frac{3x^2}{b}$ => A azt jelenti, hogy az A akkumulátor tartalma az adott utasitás végrehajtása után $\frac{3x^2}{b}$ lett.

Konkrét rekeszek tartalmát zárójellel jelöljük: pl. (0312) = 0,5 azt jelenti, hogy a 0312-es cimü rekesz tartalma 0,5.

A vezérlés átadásra a $\rightarrow$ jelet vezetjük be: $\vec{a}$ azt jelenti, hogy a vezérlés az a cimü utasitásra adódik át.

1. Az M-3 gép

Müszaki jellemzői: kisteljesitményü, elektroncsöves áramkörökkel. Szinkron, párhuzamos müködésü; bináris számrendszerü. Fogyasztása: 10 kW.

Bevitel: lyukszalaggal (5 csatornás távirókód), alfanumerikusan.

Kiirás: teletype irógéppel, alfanumerikusan is. Mind a bevitel, mind a kiirás történhet 8-as számrendszerben (direkt módon) és 10-es számrendszerben (programmal, konvertálással) is.

Memória: 1. dob memória (kb. 1280 szó kapacitással)
2. ferrit-memória (1024 szó kapacitással)
3. mágnesszalag-memória; haladási sebesség 0,7 m/sec; kapacitása: 80000 szó.

Átlagos müveleti sebesség: dobbal: 30 müv/sec.
ferritmemóriával: 1000 müv/sec.

Fontosabb regiszterek

a/ Aritmetikai egység: 4 regiszterből áll (A,B,C,D)

b/ Utasitásszámláló regiszter (UR): e regiszterbe állitjuk be az első végrehajtandó utasitás cimét (a vezérlőpultról); tartalma a mindenkori aktuális utasitás cime.

c/ Szelekciós regiszter: e regiszter választja ki a memóriából való olvasásnál, ill. irásnál az utasitás által kijelölt cimeket. A vezérlőpultról is beállitható a regisz-

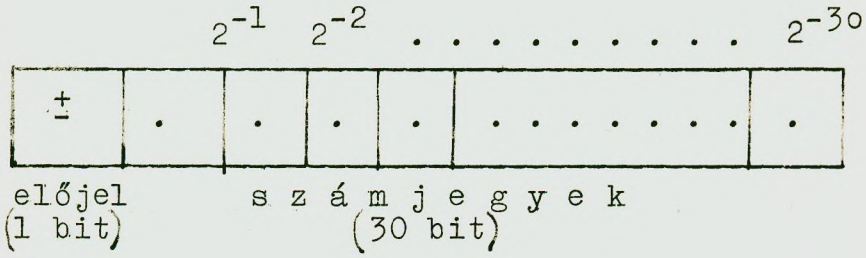
terbe egy adott cim: segitségével a pultról is kiolvasható valamely rekesz tartalma.

Szószerkezet

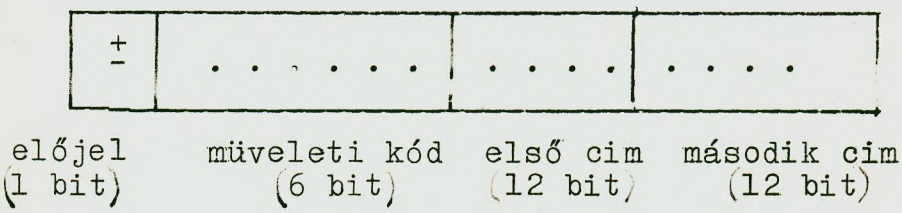
Kétcimü, fixpontos.

Szóhossz: 31 bit (ebből 1 bit az előjel).

Szám ábrázolása:



Utasitás ábrázolása:



Egy utasitás általános alakja tehát:

m a b

ahol m = ij kétjegyü nyolcas számrendszerben felirt szám (pl. 02 0316 0527)

Megjegyzés: a 0000-tól 1777-ig terjedő cimek ferrit-memória-cimek (gyors-memória) a 2000-től 4400-ig terjedőek pedig dob-cimek.

Utasitásrendszer

A gép kétcimü: az utasitások által realizált müveletek

a memória két rekeszében elhelyezett számra vonatkoznak. Egy-egy aritmetikai utasitás végrehajtása általában a következőképpen megy végbe: az első cim tartalma bekerül az aritmetikai egység egyik regiszterébe, a második cim tartalma pedig a másik regiszterbe; a két regiszterben tárolt számokkal a gép elvégzi a müveleti kód szerinti müveletet és az eredmény vagy a két regiszter egyikében, vagy a harmadik regiszterben jelenik meg és általában beiródik a memóriába az utasitás második cimén szereplő rekeszbe.

Az M-3 gépben -0 is van: a nulla előjelét a második cim tartalmának az előjele, ill. a B regiszter előjele határozza meg.

A gép aritmetikai egységének regiszterei közül kitüntetett szerepe van a B regiszternek: minden utasitás végrehajtása után az adott művelet eredménye a B regiszterben megmarad. Vannak olyan utasitások, amelyek az eredményt csak a B regiszterbe irják be (a memóriába nem); más utasitások viszont a művelet egyik komponensét nem a memóriából, hanem a B regiszterből veszik. Az első tipusu utasitásokat, vesszővel jelöljük (pl. +,) és vesszős utasitásnak hivjuk, a második tipusuakat nyillal jelöljük (pl. ↓+ és "nyilas" utasitásoknak hivjuk. Vannak olyan utasitások is, amelyek nyilasak és vesszősek is egyidejüleg.

Aritmetikai utasitások

1. Összeadás (m = i0)

+	tipus	00 a b	jelentése	'a + 'b => b
+,	"	10 a b	"	'a + b' => B
↓+	"	20 a b	"	'B + 'a => b
↓+,	"	30 a b	"	'B + 'a => B

+n tipus 40 a b jelentése 'a + 'b => b és az eredményt kinyomtatja

|+|, " 50 a b " 'a + 'b => B

↓+n " 60 a b " 'a + B => b és az eredményt kinyomtatja

↓|+|, " 70 a b " 'B + 'a => B

2. Kivonás (m = i 1)

- tipus 01 a b jelentése 'b - 'a => b

-, " 11 a b " 'b - 'a => B

↓- " 21 a b " 'B - 'a => b

↓-, " 31 a 0 " 'B - 'a => B

-n " 41 a b " 'b - 'a => b és az eredményt kinyomtatja

|-|, " 51 a b " 'b - 'a => B

↓-n " 61 a b " 'B - 'a => b és az eredményt kinyomtatja

↓|-|, " 71 a 0 " 'B - 'a => B

3. Osztás (m = i2)

: tipus 02 a b jelentése 'b : 'a => b

:, " 12 a b " 'b : 'a => B

↓: " 22 a b " 'B : 'a => b

↓:, " 32 a 0 " 'B : 'a => B

:n " 42 a b " 'b : 'a => b és az eredmény kinyomtatása

|:|, " 52 a b " 'b : 'a => B

↓:n " 62 a b " 'B : 'a => b és az eredmény kinyomtatása

|:|, tipus 72 a b jelentése 'B : 'a => b

4. Szorzás (m = i3)

x	tipus	03 a b	jelentése 'a . 'b => b
x,	"	13 a b	" 'a . 'b => B
↓x	"	23 a b	" 'B . 'a => b
↓x,	"	33 a b	" 'B . 'a => B
xn	"	43 a b	" 'a . 'b => b és az eredmény kinyomtatása
|x|,	"	53 a b	" 'a . 'b => B
xn	"	63 a b	" 'B . 'a => b és az eredmény kinyomtatása
↓|x|,	"	73 a 0	" 'B . 'a => B

5. Logikai szorzás (m = i6).

A logikai szorzás bitenkénti szorzást jelent: az egyik bináris szám minden bitje logikailag (1 x 1 = 1; 1 x 0 = 0 0 x 0 = 0) összeszorzódik a másik szám megfelelő bitjével (az előjelek nem; az eredmény előjele mindig a második cim, illetve a B regiszter előjele).

Λ	tipus	06 a b	jelentése: 'a Λ 'b => b
Λ,	"	16 a b	" 'a Λ 'b => B
↓Λ	"	26 a b	" 'B Λ 'a => b
↓Λ,	"	36 a b	" 'B Λ 'a => B
Λ n	"	46 a b	" 'a Λ 'b => b és az eredmény kinyomtatása
|Λ|,	"	56 a b	" 'a Λ 'b => B

$\downarrow\wedge$n tipus 66 a b jelentése: 'B $\wedge$ 'b $\Rightarrow$ B és az eredmény kinyomtatása

$\downarrow|\wedge|$ " 76 a 0 " 'B $\wedge$ 'a $\Rightarrow$ B

Nem aritmetikai utasitások

1/ Beviteli utasitás

07 0 b jelentése: a perforált szalagon lévő szám (egy szó) az utasitás hatására beiródik a b rekeszbe.

2/ Átviteli utasitás

05 a b jelentése: 'a $\Rightarrow$ b

3/ Ciklus utasitás

a/ $\oplus$ tipus: 57 a b; jelentése: képezi 'a + 'b -t és az eredményt mint uj utasitást végre is hajtja.

b/ $\downarrow\oplus$ tipus: 77 a b; jelentése: képezi 'a + 'B -t és az eredményt, mint uj utasitást végre is hajtja.

4/ Vezérlési utasitások

a/ Feltétlen ugrások

U1 tipus: 24 a b; jelentése: $\vec{a}$ és ezenkivül 'B $\Rightarrow$ b

U2 " 74 a b; " $\vec{b}$ és ezenkivül |'B| $\Rightarrow$ B

U3 " 44 a b; " $\vec{a}$ és ezenkivül 'b $\Rightarrow$ B

b/ Feltételes ugrás

FU tipus: 34 a b; jelentése: $\vec{a}$ ha 'B <0

$\vec{b}$ " 'B ≥ 0

c/ Megállási utasitás

04 a b jelentése: a gép megáll; az utasitásszámláló ban megjelenik 'b, a szelekciós regiszterben pedig 'a.

d/ Mágnesszalag utasitások

1. 17 0000 b; jelentése: a gép megkeresit a b cim zónát a szalagon (a keresés meginditása után a gép tovább számol; a zónakeresés közben folyik).

2, 25 a b; jelentése: a gép átolvassa a dobról a szalagra az előzőleg 17-es kódjelü utasitással megkeresett zónába a b cimtől az a-2 cimig tartó rekeszek tartalmát

3. 35 a b; jelentése: a gép a szalag előzőleg megadott zónájából átviszi a memória b-től a-ig tartó rekeszeibe a szalagon lévő információt.

Példa:

Készitsük el az utasitásrendszer alkalmazásának bemutatására (konkrét cimekkel) az M-3 gépre az alábbi feladat programját:

Számitsuk ki az

$$s = \begin{cases} \sum_{i=1}^{n} \left(\frac{x_i^3 + a}{2x_i + b} \right)^2 & \text{ha } a^2 < b \\ ab + \frac{x_1 + x_n}{2} & \text{ha } a^2 \geq b \end{cases}$$

kifejezés értékét.

Tegyük fel, hogy a szereplő kiinduló adatok, a részeredmények és a végeredmény is egynél abszolutértékben kisebbek. Tegyük fel ezenkivül, hogy a képletben szereplő adatokat már bevittük a memória adott rekeszeibe.

Legyen a rekeszelosztás a következő:

Adatok:

(0100) = x_1 (0000) = 0

(0101) = x_2 (0001) = a

(0002) = b

(0100+n-1) = x_n (0003) = $\frac{1}{2}$

Ciklus paraméter:

(0004) = 00 n 0000

= $n.2^{-18}$

Munkarekeszek: 0005, 0006, 0007, 0010, 0011.

Program:

Az utasitás címe	Müveleti jel	Müveleti kód	Első cim	Második cim	Megjegyzés
0020	A	05	0000	0010	gyüjtőrekesz
0021	A	05	0000	0005	számlálórekesz
0022	x,	13	0001	0001	a^2
0023	↓-,	31	0002	0000	$a^2 - b$
0024	FU	34	0025	0041	$a^2 - b \gtrless 0$
0025	⊕	57	0005	0051	$x_i \Rightarrow 0006$
0026	x,	13	0006	0006	x_i^2
0027	↓x,	33	0006	0000	x_i
0030	↓+	20	0001	0007	$x_i^3 + a \Rightarrow 0007$
0031	:,	12	0003	0006	$2x_i$
0032	↓+	20	0002	0011	$2x_i + b \Rightarrow 0011$
0033	:	02	0011	0007	$\frac{x_i^3 + a}{2x_i + b} = r_i \Rightarrow 0007$
0034	↓x,	33	0007	0000	r_i^2
0035	↓+	20	0010	0010	$(0010) \Rightarrow r_i^2$

Az utasitás cime	Müveleti jel	Müveleti kód	Első cim	Második cim	Megjegyzés
0036	+	00	0053	0005	ciklus müvelet
0037	↓|-| ,	71	0004	0000	" "
0040	FU	34	0025	0047	" "
0041	-,	11	0053	0004	
0042	U1	24	0043	0005	
0043	⊕	57	0005	0052	$x_1 + x_n$
0044	↓x	23	0003	0007	$\frac{x_1 + x_n}{2} \Rightarrow$ 000
0045	x,	13	0001	0002	a b
0046	↓+	20	0007	0010	0010 = a b + x_1
0047	Ma	04	0000	0000	megállás.

Utasitás-konstansok:

(0051) = 05 0100 0006
(0052) = 10 0100 0100
(0053) = 00 0001 0000

A program elemzését a megjegyzési rovat segitségével az olvasóra bizzuk.

2. Az ELLIOTT-803 (E-803) gép

Müszaki jellemzői: kis- közepes teljesitményü; tranzisztoros áramkörökkel. Bináris számrendszerü. Fogyasztása: 3,5 kW.

Bevitel: alfanumerikus lyukszalagolvasó berendezés, 500 jel/sec olvasási sebességgel.

Kiirás: szalaglyukasztó (csak perforál) berendezés: 100 jel/sec sebességgel.

Memória: 1. Ferritmemória, 8192 szó kapacitással.
2. Mágnesfilm segédmemória (2 db), egyenként 262 144 szó kapacitással.

Főbb regiszterei: Akkumulátor (A)
Segédregiszter (S.R)

Megjegyzés:

A lebegőpontos számoknál d 2^p előállitásban a gépben a p kitevőt p + 256 alakban ábrázoljuk: azaz a 0 kitevőnek 256 felel meg; tehát $-256 \leq p \leq 255$. A gép egycimü, fixpontos, behuzalozott lebegőponttal.

Átlagos müveleti sebességek

1/ fixponttal

összeadás, kivonás:	1700 müv/sec
szorzás:	1000 müv/sec
osztás:	80 müv/sec

2/ lebegőponttal

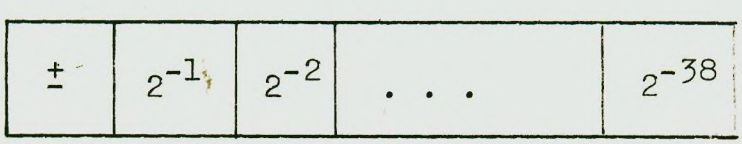
összeadás, kivonás:	1000 müv/sec
szorzás:	200 müv/sec
osztás:	100 müv/sec

Szószerkezet:

Szóhossz: 39 bit

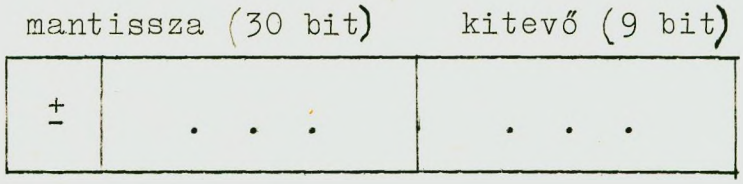
Számok ábrázolása:

fixponttal:



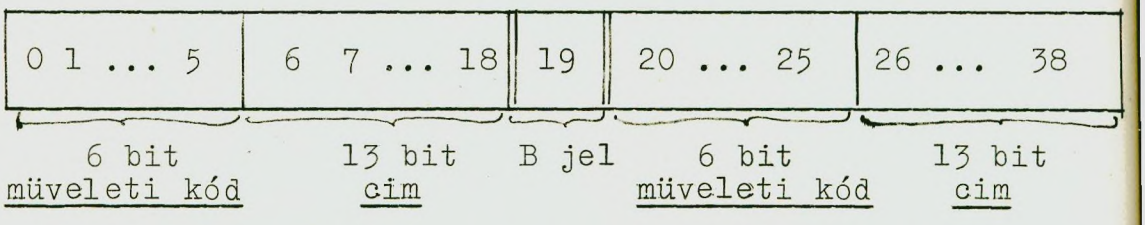
előjel 1 bit — számrész 38 bit

lebegőponttal:



Utasitások ábrázolása:

Egy szóban két utasitás helyezhető el a következő módon



Egy utasitás általános alakja tehát:

m a

ahol m = ij, 6 bit (2 nyolcas számrendszerbeli számjegy),
a pedig 13 bit ($a \leq 8191$) (Pl. 22 6392)

Az utasitások konkrét felirásában az a cimet tizes számrendszerben irjuk fel, a müveleti kódot pedig nyolcas számrendszerben. Az E-utasitásokban a cimrészben az értéktelen nullákat nem kell kiirni, 02 0028 helyett tehát 02 28 irható. A B jel 0 értékét ":"-vel, az 1-es értékét /jellel jelöljük. (Pl. 03 028/05 3619; 17 229: 16 2891)

A B jel utasitás-módositásra szolgál: B = 1 esetén a szó első utasitásának végrehajtása után, a második végrehajtása előtt, az első utasitásban szereplő cim tartalmának utolsó 19 bitje hozzáadódik a második utasitáshoz és az első utasitás után a második ilymódon módositva kerül végrehajtásra (a módositáshoz lényegében külön idő nem szükséges). Ha B = 0, akkor módositás nem történik.

A helyes müködés kontrollja paritás vizsgálattal törté-

nik. (Hiba esetén a program automatikusan leáll és a vezérlőpulton kigyullad a hibát jelző lámpa.)

A gép utasitásrendszere

Fixpontos müveletek

A/ Az alábbi ij a tipusu utasitásokban j a müveleti kód második nyolcas számrendszerbeli jegye (3 bit) a müveleti jel, i az első jegy (első 3 bit) pedig a művelet eredményének helyét, ill. az akkumulátornak és az a cimnek a művelet végrehajtása utáni tartalmát határozza meg.

Az i különböző értékeinél és ($0 \leq j \leq 7$ esetén) a következő történik:

a/ 0 j a tipusu utasitások (azaz i = 0) esetén a művelet eredménye az akkumulátorban marad és az a cimü rekesz tartalma nem változik, azaz:

$E = {}'A\ O\ {}'a \Rightarrow A$ és ${}'a_e \Rightarrow a$

O itt valamilyen művelet; E pedig a művelet eredménye.

b/ 1 j a esetén (i = 1) a művelet eredménye az akkumulátorban marad, előtte azonban az akkumulátor eredeti tartalma az a cimre megy át, azaz

$E = {}'A\ \Theta\ {}'a \Rightarrow A;\ {}'A \Rightarrow a$

c/ 2 j a esetén (i = 2) a művelet eredménye az a cimre kerül és az akkumulátor eredeti (a művelet végrehajtása előtti) tartalma változatlan marad, azaz

$E = {}'A\ O\ {}'a \Rightarrow a;\ {}'A_e \Rightarrow A$

d/ 3 j a esetén a művelet eredménye az a cimre kerül és az akkumulátor tartalma az a cim eredeti tartalma lesz, azaz

$E = {}'A\ O\ {}'a \Rightarrow a;\ {}'a_e \Rightarrow A.$

Az alábbi utasitások egészen a B pontig a fentiek alapján végzik el a megfelelő müveleteket.

1/ Átküldő művelet (Á): i o a

Ez a művelet lényegében átviteli művelet, amellyel az akkumulátor és egy adott rekesz tartalma "kicserélhető".

Jelentése - az előzőkben mondottaknak megfelelően:

i = 0 esetén 00 a: $'A_e \Rightarrow A$; $'a_e \Rightarrow a$

i = 1 " 10 a: $'a \Rightarrow A$; $'A_e \Rightarrow a$

i = 2 " 20 a: $'A_e \Rightarrow a$; $'A_e \Rightarrow A$

i = 3 " 30 a: $'a_e \Rightarrow a$; $'a_e \Rightarrow A$

(Itt az Á művelet az akkumulátor ill. az a rekesz tartalmának az olvasása.)

2/ Előjel-változtatás: (EV): i 1 a

Jelentése:

i = 0 esetén 01 a: $-'A_e \Rightarrow A$ $'a_e \Rightarrow a$

i = 1 " 11 a: $-'a \Rightarrow A$ $'A_e \Rightarrow a$

i = 2 " 21 a: $-'A_e \Rightarrow a$ $\bar{}A_e \Rightarrow A$

i = 3 " 31 a: $\bar{}'a_e \Rightarrow a$ $'a_e \Rightarrow A$

3/ Törlés (T): i 6 a

Jelentése:

i = 0 esetén 06 a: $0 \Rightarrow A$ $'a_e \Rightarrow a$

i = 1 " 16 a: $0 \Rightarrow A$ $'A_e \Rightarrow a$

i = 2 " 26 a: $0 \Rightarrow a$ $'A_e \Rightarrow A$

i = 3 " 36 a: $0 \Rightarrow a$ $'a_e \Rightarrow A$

4/ Leválasztó: (logikai szorzás: LE) i 3 a.

A müvelet eredménye: E = 'A$\wedge$'a (a megfelelő bitek logikailag összeszorzódnak: 1 x 1 = 1, 1 x 0 = 0).

A mondottaknak megfelelően:

i = 0 esetén: E = 'A$\wedge$'a $\Rightarrow$ A $'a_e \Rightarrow$ a

i = 1 " E = 'A$\wedge$'a $\Rightarrow$ A $'A_e \Rightarrow$ a

i = 2 " E = 'A$\wedge$'a $\Rightarrow$ a $'A \Rightarrow$ A

i = 3 " E = 'A$\wedge$'a $\Rightarrow$ a $'a_e \Rightarrow$ A

5/ Összeadás:

a/ Cimnövelő összeadás: (C.Ö; j = 2) i 2 a

A müvelet eredménye: E = 'a + 1; ('a + 1 jelentése: $'a + 1.2^{-38}$). Az előzőkben mondottaknak megfelelően az i = 0, 1, 2, 3 értékekre az utasitás a következőképpen működik:

02 a esetén 'a + 1 $\Rightarrow$ A; $'a_e \Rightarrow$ a

12 a " 'a + 1 $\Rightarrow$ A; $'A_e \Rightarrow$ a

22 a " 'a + 1 $\Rightarrow$ a; $'A_e \Rightarrow$ A

32 a " 'a + 1 $\Rightarrow$ a; $'a_e \Rightarrow$ A

b/ Összeadás: (Ö; j = 4) i 4 a

A müvelet eredménye: E = 'A + 'a

6/ Kivonás

a/ Egyenes kivonás (EK; j = 5): i 5 a

A müvelet eredménye: E = 'A - 'a

b/ Forditott kivonás (FK; j = 7) i 7 a

A müvelet eredménye: E = 'a - 'A

B/ Szorzás:

a/ Hosszu szorzás (HSz) (Ennél az utasitásnál az akkumulátor tartalmaz az S.R. segédregiszterrel 77 bitre bővithető (39 + 38 bit))

52 a jelentése: 'A x 'a ⇒ A + S.R. (Az eredmény 77 bit.)

b/ Rövid szorzás: (RSz) 53 a Jelentése:

'A x 'a ⇒ A és 0 ⇒ S.R.

c/ Osztás:

56 a Jelentése: az akkumulátor és a segédregiszter együttes tartalma elosztódik 'a -val és az eredmény az akkumulátorba kerül, azaz

'(A + S.R.) : 'a ⇒ A és 0 ⇒ S.R.

D/ Eltolási müveletek:

1/ 50 m (Hosszuszavas jobbratolás:) H JT

Jelentése: '(A + SR)-t n helyértékkel jobbra tolja ; az előjel nem változik (n ≤ 77 és tizes számrendszerben kell megadni).

2/ 51 n (Rövidszavas jobbratolás:) RJT

Jelentése: 'A eltolása n helyértékkel jobbra. A negativ előjel elvész. (n ≤ 39; 0 ⇒ SR; n-t tizes számrendszerben kell megadni.)

3/ 54 n (Hosszuszavas balra tolás HBT)

Jelentése: '(A + SR) eltolása balra n hellyel (n ≤ 77; n-t tizes számrendszerben kell megadni.)

4/ 55 n (Rövidszavas balratolás RBT)

Jelentése: 'A balra tolása n hellyel. (0 ⇒ SR; n ≤ 39; n-t tizes számrendszerben kell megadni.)

5/ 57 n A segédregiszter átvitele az akkumulátorba (SR => A)

Jelentése: A segédregiszter tartalmát áttolja az akkumulátor 1-38 bitjei helyére és az előjel bitje törlődik (SR tartalma marad).

Lebegőpontos müveletek

Az alábbiakban 'A = $x_1 2^{y_1}$; 'a = $x_2 2^{y_2}$. A müveletek eredményei normalizáltak.

a/ Lebegőpontos összeadás (LÖ): 60 a

Jelentése: 'A + 'a => A

b/ Lebegőpontos egyenes kivonás (LEK): 61 a

Jelentése: 'A - 'a => A

c/ Lebegőpontos forditott kivonás (LFK): 62 a

Jelentése: 'a - 'A => A

d/ Lebegőpontos szorzás (LSz): 63 a

Jelentése: 'A x 'a => A

e/ Lebegőpontos osztás (LO): 64 a

Jelentése: 'A : 'a => A

f/ Fixpontosan ábrázolt szám átalakitása lebegőpontos alakba: (F => L utasitás): 65 4096

Jelentése: az akkumulátor fixpontos tartalmát átalakitja lebegőpontos alakra és az akkumulátorban hagyja.

Megjegyzés: a lebegőpontos müveletek mind "rövidek"; végrehajtásuk közben SR tartalma nem változik.

Vezérlési utasitások

a/ Feltétlen ugrás (FNU)

1. FNU1: 40 a

Jelentése: az a cimü rekeszben az első félszóban szereplő utasitásra adja át a vezérlést. ($\vec{a}$)

2. FNU2: 44 a

Jelentése: az a cimü rekeszben a második félszóban lévő utasitásra adja át a vezérlést ($\vec{a}$)

b/ Feltételes ugrások

1. FEU1: 41 a

Jelentése: az a cimü rekesz első félszavára adja át a vezérlést, ha 'A<0; különben a soronkövetkező utasitásra.

2. FEU2: 45 a

Jelentése: az a cimü rekesz második félszavára adja át a vezérlést, ha 'A<0; különben a soronkövetkező utasitásra.

3. FEU3: 42 a

Jelentése: az a cimü rekesz első fél szavára adja át a vezérlést, ha 'A = 0, ellenkező esetben a soronkövetkezőre.

4. FEU4: 46 a

Jelentése: az a cimü rekesz második félszavára adja át a vezérlést, ha 'A = 0; ellenkező esetben a soronkövetkezőre.

5. FEUCS1 (tulcsordulásos ugrás): 43 a

Jelentése: az a cimü rekesz első félszavára adja át a vezérlést, ha a "tulcsordulás-trigger" lebillen.

6. FEUCS2 (tulcsordulásos ugrás): 47 a

Jelentése: az a cimü rekesz második félszavára adja át a vezérlést, ha a "tulcsordulás-trigger" bebillen.

Input és output utasitások

Az E-803 gépben a memória 0, 1, 2, 3 cimü rekeszébe egy standard program van irva, amely a bevitel céljára szolgál. (Ezekből a rekeszekből az akkumulátor nullát olvas ki.)

Pult-müveletek

1. Bevitel a vezérlőpultról: 70 0

Jelentése: a vezérlőpult beállitó-regiszterén beállitott szó bevitele az akkumulátorba.

2. Utasitás rekesz-cimének beirása: 73 a

Jelentése: ezen utasitás rekeszcimét beviszi az a rekesz második felébe.

Lyukszalag olvasási müveletek

1. Bevitel-1: 71 0:

Jelentése: egy lyukszalag jel leolvasása az 1.sz. olvasón és bevitele az akkumulátorba.

2. Bevitel-2: 71 2048.

Jelentése: Egy lyukszalag jel leolvasása a 2.sz. olvasón és bevitele az akkumulátorba.

Ezekkel az utasitásokkal csak egy jelet (5 bit) lehet bevinni. Az utasitások és a számok szalagról történő automatikus bevitele az erre a célra készitett input-programmal történik.

Lyukszalag kiirások

1. Kiirás-1: 74 a

Jelentése: az a rekesz utolsó 5 bitjét az 1.sz. lyukasztón a szalagra perforálja.

2. Kiirás-2: 74 2048+a

Jelentése: az a rekesz utolsó 5 bitjét a 2.sz. lyukasztón a szalagra perforálja.

3. Kiirás-3: 74 4096+a

Jelentése: az a rekesz utolsó 5 bitjének kiirása a nyomtató berendezéssel.

Mágnesszalag utasitások

Az E-803 mágnesszalagja egy mágnesréteggel ellátott filmszalag. A szalag nem "végtelenitett": két irányban mozog; mind a két irányban képes irni és olvasni.

Az E mágnesszalag memóriája 4 tárolóból áll (egyszerre 4 szalag kezelésére képes), jelöljük ezeket T1, T2, T3, T4-el.

A szalag 9 csatornás: ebből 7 csatorna tárolja a ráirt szavakat; egy (az ötödik) a szinkron csatorna és egy (a kilencedik), a blokkcimeket tárolja.

A szalagon az információt sorszámozott blokkokban irjuk fel. Egy blokk hossza: 64 szó (8 cm). Az egyik irányban a 0-tól 2047-ig terjedő sorszámu, a forditott irányban való mozgásnál pedig a 2048-tól 4096-ig terjedő sorszámu blokkokkal dolgozik a gép. Az i és a 4095-i sorszámu blokkok összetartoznak.

Egy szó a szalagon 6 sorban helyezkedik el; az előjel a 6. sorban van. Minden szó mellett a kontroll céljára párossági bitek is szerepelnek.

Szalag-kontrollszó. A szalagmüveletek kontrollja egy 5 bitből álló kontrollszóval történik; ez a szó az akkumulátor utolsó 5 bitje helyére kerül. Az 5 bit jelentése a következő:

5. bit: azt vizsgálja, hogy be van-e kapcsolva a szalag. (Ha értéke 0, akkor igen, ellenkező esetben nincs bekapcsolva.)

4. bit: a blokkcimben volt-e párossági hiba az előzőkben. (Ha volt, akkor értéke: 1.)

3. bit: arra ad választ, hogy szabad-e beirni a blokkba. (Ha értéke 1, akkor igen. Olvasni mindig lehet!)

2. bit: jelentése: a tároló éppen keresés alatt áll, ha értéke: 1.

1. bit: jelentése: az adatcimek között volt-e tárolási hiba az előzőekben.

Utasitások:

1/ 75 1027

Jelentése: a legutóbb leolvasott vagy kiirt blokk cimének leolvasása.

2/ 76 1024
76 1032
76 1040
76 1048

Jelentésük: a mágnesfilm kontroll-szavának beolvasása az akkumulátorba és a T1, T2, T3, T4 tárolón való olvasás előkészitése. (A 76 1024 a T1, a 76 1032 a T2 s.i.t. tárolóra vonatkozik.)

3/ 76 1025
76 1033
76 1041
76 1049

Jelentésük: a mágnesfilm kontroll szavának beolvasása az akkumulátorba és a T1, T2, T3, T4 tárolóra való kiirás előkészitése.

4/ 76 1026
76 1034
76 1042
76 1050

Jelentésük: a mágnesfilm kontrollszavának beolvasása az akkumulátorba és a T1, T2, T3, T4 tárolón való keresés előkészitése.

5/ 77 a. Olvasás, kiirás, vagy keresés a megelőző 76-os utasitásoknak megfelelően.

A szalag-beirás (olvasás és irás esetén egyaránt) két utasitással történik: az első valamilyen 76-os utasitás, a második pedig 77-es, Pl.

76 1033: 77 a.

Példaként az utasitásrendszer alkalmazására készitsük el e gépre is az

$$S = \begin{cases} \sum_{i=1}^{n} \left(\dfrac{x_i^3 + a}{2x_i + b} \right)^2 & \text{ha } a^2 < b \\ a\,b + \dfrac{x_1 + x_n}{2} & \text{ha } a^2 \geq b \end{cases}$$

kifejezés fixpontos programját (tegyük fel, hogy a kiinduló adatok, a részeredmények és az eredmények egynél abszolutértékben kisebbek).

Legyen a rekeszelosztás a következő (és tegyük fel, hogy a megfelelő kiinduló adatok bent vannak a gépben előzetes bevitel folytán).

Adatok:

(100)	$= x_1$	(10)	= 0
(101)	$= x_2$	(11)	= a
		(12)	= b
		(13)	= 0,5
(100+n-1)	$= x_n$		

Munkarekeszek: 50; 51; 52; 53

Paraméter: (14) $= n.2^{-38}$

Utasitás cime	Első utasitás	Második utasitás		Megjegyzések
20	30 11	: 53 11	ág vizsgálat	$a \Rightarrow A$; $a^2 \Rightarrow A$
21	05 12	: 41 27		$a^2 - b \Rightarrow A$;
22	00 14	/ 30 99	$a^2-b \geq 0$ ág	$x_n \Rightarrow A$
23	04 100	: 53 13		$x_1+x_n \Rightarrow A$; $(x_1+x_n)0,5 \Rightarrow A$
24	20 50	: 30 11		$\frac{x_1+x_n}{2} \Rightarrow 50$; $a \Rightarrow A$
25	53 12	: 24 50		$ab \Rightarrow A$; $\frac{x_1+x_n}{2}+ab \Rightarrow 50$
26	45 36	: 0		--
27	26 50	: 26 51		$0 \Rightarrow 50$; $0 \Rightarrow 51$
28	22 51	/ 30 99		$x_1 \Rightarrow A$
29	20 52	: 55 1	$a^2-b<0$ ág	$x_i \Rightarrow 52$; $2x_i \Rightarrow A$
30	04 12	: 20 53		$2x_i+b \Rightarrow A$; $2x_i+b \Rightarrow 53$
31	30 52	: 53 52		$x_i \Rightarrow A$; $x_i^2 \Rightarrow A$
32	53 52	: 04 11		$x_i^3 \Rightarrow A$; $x_i^3 + a \Rightarrow A$

Utasitás cime	Első utasitás	Második utasitás	Megjegyzések
33	56 53	: 20 53	$r_i = \frac{x_i^3 + a}{2x_i + b} \Rightarrow A$; A => 53
34	53 53	: 24 50	$r_i^2 \Rightarrow A$ [$r_i^2 \Rightarrow 50$
35	30 51	: 05 14	ciklusvizsgálat
36	42 28	: . . .	

3. Az URAL-1 gép

Müszaki jellemzői: kisteljesitményü, elektroncsöves áramkörökkel. Fogyasztása: 10 kW.

Bevitel: lyukasztott film, amelynek maximális hossza 250 m (sebessége: 77 szó/sec.)

Kiirás: speciális nyomtató (100 szó/perc) és kimenő perforátor (160 szó/perc).

Memória:

1/ dob-memória 100 szó/ sec. elérési idővel, 2024 szó kapacitással.

2/ Mágnesszalag-memória 80 000 szó kapacitással; sebessége 77 szó/sec. A szalag maximális hossza: 250 m.

Átlagos müveleti sebesség: 100 utasitás/sec.

Fontosabb regiszterek:

Akkumulátor (A)
Aritmetikai egység regisztere (AR)
Előjel regiszter (ER)

Ciklus számláló (CR)
Utasitás regiszter (UR)

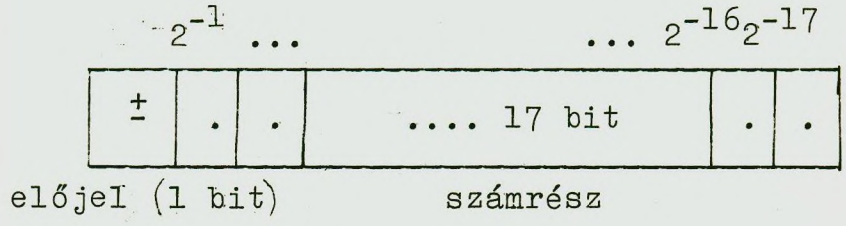
Szószerkezet: egycimü, fixpontos.

Szóhossz: a/ rövid szó 18 bit (ebből 1 bit az előjel)

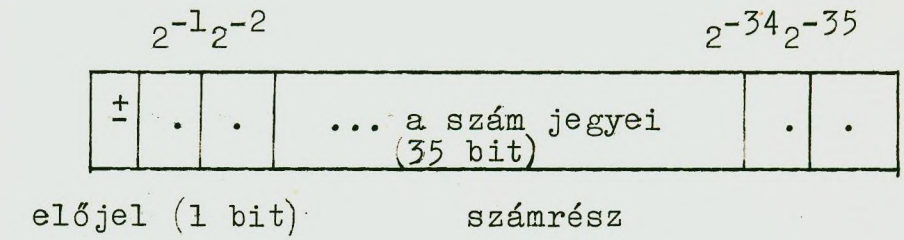
b/ hosszu szó: 36 bit (1 bit előjel)

Szám ábrázolása:

a/ rövid szóban

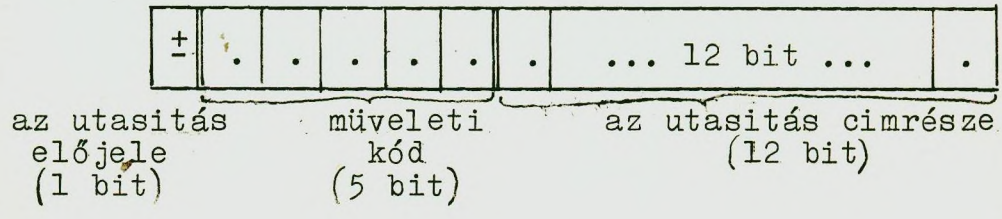


b/ Hosszu szóban



Megjegyzés: minden hosszu szót két rövid szó egyesitésének kell tekinteni. Egy hosszu rekesz cimét, ha azt az a és a+1 rekesz egyesitéséből kaptuk a+4000-el jelöljük, kikötve, hogy a csak páros lehet. Pl. a 4010 rekesz cim a 10-es és a 11-es rekesz egyesitéséből létrejött hosszu rekeszt jelöli.

Utasitás ábrázolása



Egy utasitás általános alakja tehát

m a ahol m = ij, egy kétjegyü

a = egy négyjegyü

nyolcas számrendszerben felirt szám.

A szó elejéről számitva a 7. bit azt jelöli, hogy az utasitásban szereplő cim hosszu vagy rövid cimet jelent-e: ha itt egyes áll, akkor az - a biteket hármasával összefogva - nyolcas számrendszerben éppen 4-et (azaz 4000-et) jelent, s igy az adott cim hosszu szóra vonatkozik, ellenkező esetben rövid szóra.

A gép utasitásrendszere

Mivel a gép egycimü, a müveletek általában egy adott cimre és az akkumulátorra vonatkoznak; néhány utasitás az R regisztert használja fel.

A különböző aritmetikai és logikai müveletek végrehajtásakor a gép aritmetikai egysége létrehoz két speciális jelet: az un. ε és az ω jelet, amelyet a vezérlőegység vesz át, ill. amelyek a vezérlési utasitásokban vesznek részt.

A ε jel tulcsordulás esetén jön létre: azt mondjuk,hogy ilyenkor értéke: 1, ellenkező esetben nulla.

A ε jelhez a vezérlőpulton egy "Megállás ε-re" kapcsoló tartozik: ha ez bekapcsolt állapotban van, akkor ε = 1 esetén a gép megáll; e kapcsoló kikapcsolt helyzete esetén viszont kihagyja az adott utasitás után következő utasitást.

Minden egyes utasitásnál külön megadjuk az ω jelről szóló információt. Vezérlési utasitásoknál a logikai feltételt mindig az előző utasitás végrehajtása folyamán keletkező ω jel értéke adja meg.

Aritmetikai müveletek

1/ Összeadás (Ö): 01 a

Jelentése: 'A + 'a ⇒ A és $\omega = 1$, ha 'A ≤ -0

$\varepsilon = 1$, ha $|'A| \geq 1$

2/ Kivonás (K): 03 a

Jelentése: 'A- 'a ⇒ A és $\omega = 1$, ha 'A ≤ -0

$\varepsilon = 1$, ha 'A ≥ 1

3/ Abszolutértékes kivonás (|K|): 04 a

Jelentése: $|'A| - |'a| \Rightarrow A$ és $\omega = 1$, ha 'A ≤ 0

$\varepsilon = 0$.

4/ Szorzás (Sz): 06 a

Jelentése: 'A . 'a ⇒ A és $\omega = 1$, ha 'A ≤ -0

$\varepsilon = 0$

5/ Összeadásos szorzás (ÖSZ): 05 a

Jelentése: 'A + 'R 'a ⇒ A és $\omega = 1$, ha 'A ≤ 0

$\varepsilon = 1$, ha $|'A| \geq 1$.

Megjegyzés: nyilvánvaló, hogy a müvelet végrehajtása előtt az R regiszterbe az egyik tényezőt be kell vinni.

6/ Összegező (ÖG): 26 a

Jelentése: ugyanaz, mint a 01 a utasitásé, azzal a különbséggel, hogy a ε jel blokkolódik (kontrollösszegek képzésére alkalmas).

7/ Utasitásmódositó összeadás (UM): 30 a

Jelentése: az utasitás 'a -t hozzáadja az utána következő utasitáshoz; és a módositott utasitást beirja az utasitásregiszterbe (nem a memóriába!), a soronkövetkező utasitást 'a -val módositva hajtja végre a gép. Az a cim csak rövid szó cime lehet.

8/ Osztás (O): 07 a

Jelentése: 'A : 'a => A és $\omega = 1$, ha 'A ≤ 0

$\varphi = 1$, ha 'A ≥ 1

"Adminisztrativ" utasitások

1/ Előjelképzés (E): 10 a

Jelentése: 'A.sign'a => A. (azaz az akkumulátor tartalmának előjele az a rekesz tartalmának az előjele lesz).

2/ Eltolás (ET): 11 0

Jelentése: az R regiszter tartalma az akkumulátor 19-24-ig terjedő helyértékein lévő szám által meghatározott mértékben balra vagy jobbra tolódik el, aszerint, hogy az akkumulátor tartalmának előjele pozitiv, vagy negativ. (A müvelet az előjelet is eltolja.)

3/ Logikai szorzás ($\wedge$): 12 a

Jelentése: 'A $\wedge$ 'a => A és $\omega = 1$, ha 'A = +0

$\varphi = 0$

(Az a rekesz és az akkumulátor tartalmának megfelelő bitjei logikailag összeszorzódnak: az $1 \wedge 1 = 1$; $1 \wedge 0 = 0$ szabálynak megfelelően.)

4/ Logikai összeadás (V): 13 a

Jelentése: 'A $\vee$ 'a => A és $\omega = 1$, ha 'A = +0;

$\varphi = 0$

(Az a rekesz és az akkumulátor tartalmának megfelelő bitjel logikailag összeadódnak: $1 \vee 1 = 1$;
$1 \vee 0 = 1$; $0 \vee 1 = 1$; $0 \vee 0 = 0$ alapján.

5/ Összehasonlitás (ÖH): 14 a

Jelentése: az a rekesz és az akkumulátor tartalmát bitenként összehasonlitja és az összehasonlitás eredményét

az akkumulátorban rögziti: ha két azonos helyértéken levő bit megegyezik, akkor az eredmény az azonos helyértékre irt 0; ha különbözik, akkor: 1, azaz

1 ÖH 1 = 0

1 ÖH 0 = 1

0 ÖH 1 = 1

0 ÖH 0 = 0

Megjegyzés a $\wedge$, $\vee$, ÖH utasitásokhoz: ha ezeket a müveleteket rövid szóra alkalmazzuk, akkor e szó cimének párosnak kell lennie. (Ugyanis ezek a müveletek hosszu szóra vonatkoznak: viszont a páros rövid szó egy hosszu szó első része; a páratlan rövid szó pedig a második része.)

6/ Normalizálás (N): 15 a

Jelentése: az utasitás az 'a = x fixpontos számot $x = p\,2^q$ alakban viszi át, mégpedig ugy, hogy $\frac{1}{2} \leq p < 1$ teljesüljön (normalizálja). Az eredmény p mantisszája az a rekeszbe kerül, a q kitevő pedig módositott kódban az akkumulátor 14-19-ig terjedő helyértékeire (valamint az R regiszter ugyanez helyértékeire). Itt: $\omega = 1$, ha 'a = $\pm$ 0; $\varphi = 0$, és az előjel-trigger tartalmazza a léptetések "irányát", azaz a kitevő előjelét.

7/ Átvitel az akkumulátorba (ÁO): 02 a

Jelentése: 'a => A

Itt $\omega = 1$, ha 'A $\leq$ 0; $\varphi = 0$.

8/ Átvitel egy rekeszbe (ÁI): 16 a

Jelentése: 'A => a. (Az akkumulátor tartalmát átviszi az a cimü rekeszbe.) Az akkumulátor tartalma megmarad.

Megjegyzés: rövid rekeszbe beirható az akkumulátor tartalmának második fele is, ha az adott rekesz cime páratlan:

ilyenkor a rövid rekesz cimét '4000-el meg kell növelni.

Itt $\omega = 1$, ha $'a \leq 0$; $\varphi = 0$.

9/ Átvitel az R regiszterbe (Á2): 17 a

Jelentése: $'a \Rightarrow R$. (Az a rekesz tartalmát átviszi az R regiszterbe.) Az akkumulátor tartalmát nem változtatja, de az előjel-trigger tartalma változhat, ezért az Á2 utasitás után az akkumulátor tartalmát csak az ÖSZ és ET utasitásokban szabad felhasználni.

Itt: $\omega = 1$, ha $'a = \pm 0$; $\varphi = 0$.

10/ Beirás az akkumulátorba (BA): 20 k

Jelentése: $k \Rightarrow A$. (Az utasitás cim részén szereplő szám bekerül az akkumulátor bal felébe: azaz $'A = k.2^{-17}$; $|k| \leq 3777$. A szám előjelét az utasitás 12. bitjére kell irni: $k < 0$ esetén 1-et, $k \geq 0$ esetén 0-t.)

Itt: $\omega = 1$, ha $k < 0$; $\varphi = 0$.

Vezérlési utasitások

1/ Feltételes ugrás (FU): 21 a

Jelentése: a vezérlést a soronkövetkező utasitásra adja át, ha $\omega = 0$, és az a cimü rekeszre, ha $\omega = 1$. (Azaz (b) = 21 a esetén $\overrightarrow{b+1}$ ha $\omega =$ és $\vec{a}$, ha $\omega = 1$./

Itt ω értéke nem változik meg; $\varphi = 0$.

2/ Feltételes ciklus-ugrás (FCU): 24 a

Jelentése: a vezérlést az ugró utasitás után soronkövetkező utasitásnak adja át, ha a ciklusszámláló tartalma 0; és az a cimü rekeszre, az ellenkező esetben. Egyidejüleg a ciklusszámláló regiszterhez 1, ill. 2 hozzáadódik, attól függően, hogy a regiszterben a szóhossz-jel helyen 0, vagy 1 áll-e (rövid, vagy hosszu szóra vonatkozik-e a ciklus).

Itt: $\omega = 0$; $\varphi = 0$.

3/ Feltételes beállitott ugrás (FBU): 23 k

Jelentése: a gép vezérlőpultján ezen utasitáshoz a k = 0001, 0002, 0003, 0004, 0005, 0006, 0007 értékeknek megfelelő 7 kapcsoló van. Ha a k sorszámu kapcsoló bekapcsolt állapotban van, akkor a 23 k utasitás után soronkövetkező utasitás végrehajtása elmarad. Ha az adott kapcsoló nincs bekapcsolva, akkor a program végrehajtásának menete nem változik.

Itt ω nem változik; $\varphi = 0$.

4/ Feltétlen ugrás (U): 22 a

Jelentése: a vezérlés átadódik az a cimü rekeszre, ahol a egy rövid szó cime (azaz: $\overrightarrow{a}$).

Itt ω nem változik; $\varphi = 0$.

5/ Ciklus kezdő (CK): 25 n

Jelentése: a ciklusszámlálóba (regiszterbe) átmegy - n.

Itt: $\omega = 0$; $\varphi = 0$.

Ezt az utasitást ciklusok felépitésénél a ciklus számlálására, illetve az utasitások módositására lehet felhasználni. A ciklus számláló által módositandó utasitásokat negativ előjellel kell ellátni.

Input-output utasitások

1. Bevitel. Az URAL-1 gépen az információ bevitele a gépbe lyukasztott filmszalagon történik, amelyen az információt zónákra osztva helyezzük el. A beviteli utasitással egyetlen különálló adatot nem lehet bevinni, hacsak a teljes zóna nem egyetlen adatból áll.

A bevitel csoportos utasitással történik, ugyanugy,mint más zónás beosztásu memóriaberendezésnél (pl. a mágnesszalag memóriák általában zónás beosztásuak). A zónákat zóna-sor-

számmal jelöljük meg. A bemenő berendezés egy input-utasitás hatására egy zónát visz be a gép memóriájába.

Az input utasitás három rövid szóból áll:

31 a_1

01 N

00 a_n

Jelentése: az utasitás hatására a lyukszalag N sorszámu zónájának tartalma beiródik a memória a_1-től a_n-ig terjedő rekeszeibe. Ha a bevitt információ program, vagy nyolcas számrendszerben felirt számok, akkor az a_1, ill. a_n cimek csak rövid szó cimei lehetnek; tizes számrendszerbeli számok esetén pedig hosszu szók cimei.

2. Kiirás (Ny): 32 000

Jelentése: az akkumulátor tartalmát kinyomtatja, illetve lyukszalagra perforálja. (A vezérlőpulton lévő kapcsoló állásától függően a számot nyolcas vagy tizes számrendszerben ábrázolt számként irja ki.)

3. Soremelés (SE): 34 0000

Jelentése: a kiiró berendezés ezen utasitás hatására egy sort emel ("üres" sor.)

Mágnesszalag müveletek

1. Olvasás a mágnesszalagról

Az utasitás három egymásután következő rövid szóból áll:

31 a_1

02 N

00 a_n

Jelentése: a mágnesszalag N sorszámu zónájának tartal-

mát átolvassa a memória a_1-től a_n-ig terjedő rekeszeibe. A memória rekeszei csak hosszu szók rekeszei lehetnek.

2. Irás a mágnesszalagra

Az utasitás itt is három egymásután következő rövid szóból áll:

. 31 a_1
03 N
00 a_n

Jelentése: a memória a_1-től a_n-ig terjedő rekeszeinek tartalmát átirja a mágnesszalag N sorszámu zónájába. A memória rekeszei csak hosszu szók rekeszei lehetnek.

Megállási utasitás (Ma): 37 a

Jelentése: a gép megáll (a számitást abbahagyja) és !a $\Rightarrow$ A.

Példaként az utasitásrendszer alkalmazására készitsük el itt is a már ismert feladat fixpontos programját:

Számitsuk ki az

$$S = \begin{cases} \sum_{i=1}^{n} \left(\frac{x_i^3 + a}{2x_i + b} \right)^2 & \text{ha } a^2 < b \\ ab + \frac{x_1 + x_n}{2} & \text{ha } a^2 \geq b \end{cases}$$

Tegyük fel itt is, hogy a szereplő kiinduló adatok, a részeredmények és a végeredmény is abszolutértékben egynél kisebbek.

Legyen a rekeszelosztás a következő:

Adatok (hosszu szóban elhelyezve)

(4200) = a

(4202) = b

(4204) = 0,5

(4300) = x_1

(4302) = x_2

⋮

(4300 + 2 (n-1) = x_n

Munkarekeszek: 4210, 4212, 4214

Utasitás paraméter: (0150) = 2 (n-1).2^{-17}

Utasitások cime	Müveleti kód	Cim		Megjegyzés
0100	02	4200		a => A
0101	06	4200		a^2 => A
0102	16	4210		
0103	02	4202		
0104	03	4210		
0105	21	0117		Vezérlés átadás
0106	02	4200	$a^2<b$ ág	a => A
0107	06	4202		ab => A
0110	17	4204		0,5 => R
0111	05	4300		ab+0,5x_1 => A.
0112	17	4204		0,5 => R
0113	30	0150		
0114	05	4300		(ab+0,5x_1)+0,5x_n => A
0115	16	4212		(4212) = S

Utasitások cime	Müveleti kód	Cim	Megjegyzések
0116	22	0142	$a^2 < b$ ág
0117	20	0000	
0120	16	4212	
0121	30	0150	
0122	25	4000	$-2(n-1)2^{-17} \Rightarrow$ CR
0123	-02	4300	$x_1 \Rightarrow$ A
0124	16	4210	$x_i \Rightarrow 4210$
0125	07	4204	$2x_i \Rightarrow$ A
0126	01	4202	$2x_i + b \Rightarrow$ A
0127	16	4214	$(4214)\ 2\ 2x_i + b$
0130	02	4010	$x_i \Rightarrow$ A
0131	06	4010	$x_i^2 \Rightarrow$ A
0132	06	4010	$x_i^3 \Rightarrow$ A
0133	01	4200	$x_i^3 + a \Rightarrow$ A
0134	07	4214	$r_i = \frac{x_i^3 + a}{2x_i + b} \Rightarrow$ A
0135	16	4214	
0136	06	4214	$r_i^2 \Rightarrow$ A
0137	01	4212	
0140	16	4212	$(4212) = \sum^{i} \frac{x_i^3 + a^2}{2x_i + b}$
0141	24	0123	
0142	37	4212	Megállás

A program elemzését az olvasóra bizzuk; a ciklus megvalósitásával kapcsolatban azonban néhány magyarázó megjegyzést teszünk.

A 121-es utasitás kiolvassa a 0150-es cimü rekeszből a ciklusparamétert: ez a ciklusok számának kétszerese, mert a ciklus hosszu szavakra vonatkozik. A 0122-es utasitás a ciklusparamétert negativ előjellel átviszi a ciklusszámláló regiszterbe. (Ebben az utasitásban a 4000-es cim azt jelöli, hogy a ciklusban hosszu szavak szerepelnek.)

A 0123-as utasitáshoz, végrehajtás előtt hozzáadódik a ciklusszámláló tartalma. Az első ciklus lépésben, igy a

-02	4300
-00	2(n-1)
-02	4300+2(n-1)

utasitást hajtja végre a gép, azaz az x_n értéket olvassa be a memóriából az akkumulátorba. A 0137-es utasitás a ciklus végét vizsgálja: az első ciklus lépés után, ha $n>1$ a ciklusszámláló regiszter tartalma nem 0, s igy ez az utasitás hozzáad a ciklusszámláló regiszterhez 2-öt (ennek tartalma ezáltal: $-2(n-1)+2$ lesz), és a vezérlést a 0123-as utasitásra adja át. A második ciklus lépésben, igy a

-02	4300
-00	-2n+4
-02	4300-2n+4

utasitást hajtja végre a gép, amely az x_{n-1} értéket olvassa be az akkumulátorba, s.i.t.

4. Az URAL-2 gép

Az URAL-2 gép az URAL-1 módositott és "fejlesztett" változata. Utasitásrendszere az URAL-1 utasitásrendszeréhez képest strukturában nem változott lényegesen, de valamelyest

bővült: a gép pl. lebegőpontos utasitásokkal is rendelkezik.

Müszaki jellemzői:

Közepes teljesitményü. Áramkörei elektroncsövekből állnak. Fogyasztása: 25 kW. Bináris számrendszerü.

Bevitel: Lyukszalag olvasó berendezés, amelynek olvasási sebessége: 150 szó/sec. A perforált szalag zónás beosztásu, a bevitel csoportos müvelettel valósitható meg. (Lásd: Csoportos bevitel.)

A géphez párhuzamosan 12 bemeneti berendezés kapcsolható (lyukkártyaolvasó, véletlenszámgeneráló, stb.).

Kiirás: speciális nyomtató (20 sor/sec.) és perforátor (160 szó/perc) segitségével, papirra nyomtatással, illetve szalagra lyukasztással történik.

A géphez párhuzamosan 12 kimenet (különféle kiiró berendezés; puffer-memória stb.) kapcsolható.

Memória:

1/ Ferritmagos gyorsmemória, amelynek kapacitása 2048 hosszu szó (40 bit) ill. 4096 rövid szó (20 bit).

2/ Mágnesdob. Kapacitása: 8192 hosszu szó.

Elérési sebesség: 3000 szó/sec. A dobról a ferritmemóriába csak csoportos utasitással lehet átvinni az információt.

3/ Mágnesszalag. Kapacitása: 10x100 000 hosszu szó. Olvasási sebesség: 1000 szó/sec.

Müveleti sebességek

Összeadás és kivonás

a/ fixponttal ≈ 12000 müv/sec.

b/ lebegőponttal ≈ 8000 müv/sec.

Szorzás: ≈ 2000 müv/sec.
Osztás: ≈ 1200 müv/sec.

Átlagos müveleti sebesség 5000 müv/sec.

(A fenti müveletek hosszuszavas müveletek!)

Az aritmetikai egység fontosabb regiszterei

R1 regiszter. Két részből áll:

MR1 mantissza regiszter (40 bit)

KR1 kitevő regiszter (7 bit)

R2 regiszter. Szintén két részből áll:

MR2 mantissza regiszter (40 bit)

MR2 kitevő regiszter (7 bit)

Akkumulátor: (A)

MA: mantissza akkumulátor (40 bit)

KA: kitevő akkumulátor (7 bit)

Szószerkezet: egycimü; fix- és lebegőpontos.

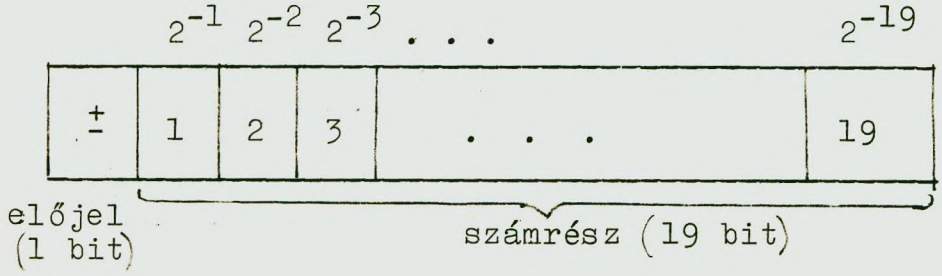
a/ hosszu szó: 40 bit

b/ rövid szó: 20 bit

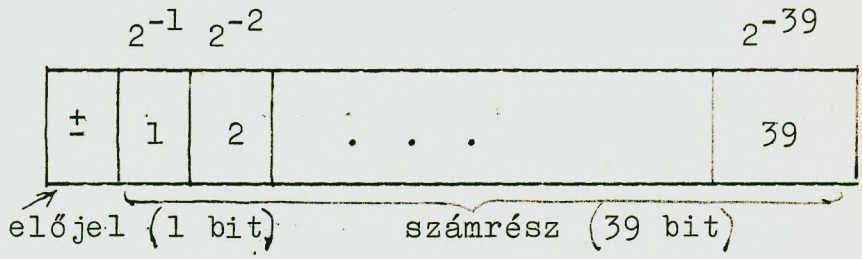
Számok ábrázolása:

Fixpontosan:

a/ rövid szóban

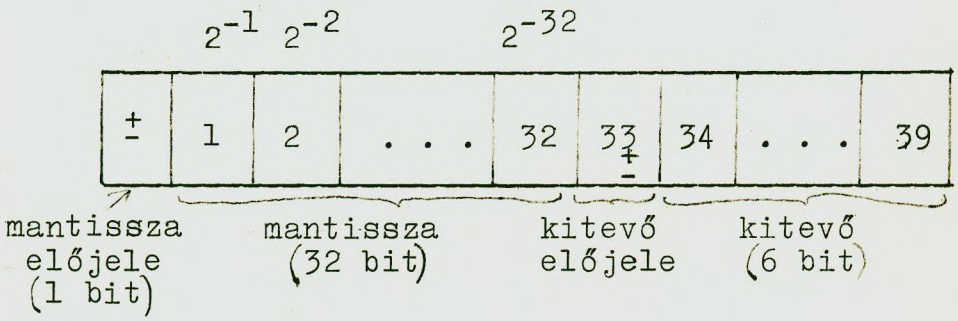


b/ hosszu szóban

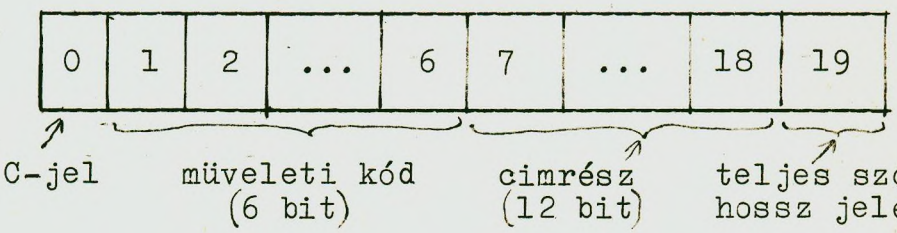


Hosszu szó az URAL-1-hez hasonlóan két egymás melletti rövid szóból képezhető.

Lebegőpontos hosszu szó



Utasitás ábrázolása:



Ha a huszadik helyértéken 1-es áll, akkor a megfelelő cim hosszu szóra vonatkozik; 0 esetén pedig rövid szóra. (A hosszu szó cime csak páros lehet.) Lyukasztásnál a teljes szóhossz jele ábrázolására nyolcas számrendszerben 4-et irunk. Egy hosszu szavas utasitás alakja tehát

01 7642 4

A C-jel utasitásmódositásra szolgál. (Ha itt 1-es áll, akkor az utasitás negativ előjelü, ami azt jelenti, hogy módositott utasitás lesz.)

Az utasitásban a cimrész ferritmemória cimet jelent: 0000-tól 7777-ig terjedhet (nyolcas számrendszerben).

Utasitásrendszer

Az alábbiakban a teljesség kedvéért felsoroljuk azokat az utasitásokat is, amelyek az URAL-1-hez képest változatlanok maradtak. Az ω és φ jelek szerepe nem változott.

Aritmetikai utasitások

Az aritmetikai müveleteknél $\omega = 1$, ha az akkumulátor tartalma (az eredmény) negativ, és $\omega = 0$, az ellenkező esetben. (Kivéve a 26-os és 30-as kódszámu összeadásokat.) A φ jel értéke itt is akkor 1, ha tulcsordulás lép fel: fixpontos számok esetén akkor, ha az eredmény 1-nél abszolutértékben nagyobb, lebegőpontos számok esetén pedig akkor, ha az eredmény kitevője 63-nál nagyobb.

a/ Fixpontos müveletek

1/ Összeadás: 01 a

Jelentése: 'A + 'a => A.

2/ Kivonás: 03 a

Jelentése: 'A - 'a => A.

3/ Szorzás: 06 a

Jelentése: 'A . 'a => A

4/ Osztás: 07 a

Jelentése: 'A : 'a => A

7/ Összegező: 26 a

Jelentése: az akkumulátor számrészéhez hozzáadódik az 'a rövid szó. Az összegezés az akkumulátor 20-39 helyértékein megy végbe; ha az eredmény "tulcsordul" azaz az utolsó átvitel a 19 helyértékre kerül, akkor az összeg csak hosszu szóban fér el, ezért az ilyen összeadás eredményét hosszu szóba vigyük át az akkumulátorból.

Itt $\omega = 0$; $\varphi = 0$

8/ Utasitásmódositó összeadás: 30 a

Jelentése: 'a-t hozzáadja az utána következő utasításhoz és ezt végrehajtja. (A módositott utasitás memóriabeli alakja természetesen nem változik.)

Lebegőpontos müveletek

Az alábbiakban mindig:

$$'a = x_1 \; 2^{p_1}$$

$$'A = x_2 \; 2^{p_2}$$

és az $E = x2^p$ eredmény mindig normalizált, azaz $\frac{1}{2} \leq x > 1$

1. Lebegőpontos összeadás: 41 a

Jelentése: 'a + 'A $\Rightarrow$ A

Ez az utasitás felhasználható egy normalizált alakban adott szám egész részének képzésére: a számhoz egyszerüen lebegőpontosan hozzá kell adni a 0.2^{39} számot.

2. Lebegőpontos kivonás: 43 a

Jelentése: 'A - 'a $\Rightarrow$ A

3. Lebegőpontos abszolutértékes kivonás: 44 a

Jelentése: $|'A| - |'a| \Rightarrow A$. Itt $\varphi = 0$.

4. Lebegőpontos szorzás: 46 a

Jelentése: 'A . 'a => A

5. Lebegőpontos osztás: 47 a

Jelentése: 'A : 'a => A. Ha 'a = 0, akkor a gép megáll és 'A változatlan marad.

Logikai utasitások

A logikai müveleteknél $\mathcal{C} = 0$.

1. Előjelképzés: 10 a

Jelentése: sign 'a . 'A => A.

Itt $\omega = 1$, ha az eredmény negativ.

2. Eltolás. Két alakja van:

a/ 11 k 0 (logikai eltolás)

Jelentése: '(MA)-nak (az akkumulátor mantissza része) mind a 40 bitje eltolódik k helyértékkel jobbra, vagy balra, attól függően, hogy a fenti utasitás 12. helyértéken 1-es vagy 0 áll-e. A k számot a 13-18 helyértékre irjuk. A "szótartományból" kitolt jegyek törlődnek. A KA a művelet végrehajtásával törlődik.

Itt $\omega = 1$, ha a nulladik helyértéken (az előjel helyén) 1-es áll.

b/ 11 k 4 (aritmetikai eltolás)

Jelentése: ugyanaz, mint az előbbinek, azzal a különbséggel, hogy az előjel helyén marad. Tulcsordulás esetén $\mathcal{C} = 1$.

Itt is $\omega = 1$, ha az eredmény negativ.

3. Logikai szorzás: 12 a

Jelentése: '(MA) $\wedge$ 'a => A. (Jegyenkénti logikai szorzás az 1 x 1 = 1, 1 x 0 = 0 szabályoknak megfelelően.) Ha az

eredmény 0, akkor $\omega = 1$. Itt $\varphi = 0$.

4. Logikai összeadás: 13 a

Jelentése: '(MA) V'a => A. (Jegyenkénti logikai összeadás az 1 + 1 = 0, 1 + 0 = 1 szabályoknak megfelelően.) Ha az eredmény 0, akkor $\omega = 1$. Itt $\varphi = 0$.

5. Összehasonlitás: 14 a

Jelentése: az MA és az a jegyeit bitenként összehasonlitja: ha két azonos helyértéken a jegyek megegyeznek, akkor az eredmény 0, ellenkező esetben 1. (Bitenkénti moduló 2 szerinti összeadás.) Ha nincs egyetlen egy egybevágó helyérték sem, akkor $\omega = 1$.

Átviteli utasitások

1. Fixpontos átvitel az akkumulátorba: 02 a

Jelentése: 'a => MA; 0 => KA

Ha 'a $<$ 0 akkor $\omega = 1$; $\varphi = 0$.

2. Fixpontos átvitel a ferritmemóriára: 16 a

Jelentése: '(MA) => a

Ha '(MA) $<$ 0, akkor $\omega = 1$; $\varphi = 0$.

3. Lebegőpontos átvitel az akkumulátorba: 42 a

Jelentése: 'a => A; mégpedig ugy, hogy 'a első 33 bitje MA-ba megy át az utolsó 7 bit pedig KA-ba. Ha az átvitt szám nem normalizált, akkor normalizálódik.

Itt $\omega = 1$, ha 'A $<$ 0 és $\varphi = 0$.

4. Lebegőpontos átvitel ferrit-memóriarekeszbe: 56 a

Jelentése: 'A => a, mégpedig ugy, hogy az akkumulátor szám része normalizálva és kerekitve átmegy az a rekesz első 33 bitje helyére, a kitevő rész pedig előjellel együtt az utolsó 7 bit helyére.

Itt ω = 1, ha az átvitt szám negativ; φ = 1, ha a kitevő nagyobb, mint 63.

5. Átvitel az akkumulátorba bevivő berendezésről: 40 m

Jelentése: mint mondottuk, az "URAL-2"-höz 12 különféle beviteli berendezés kapcsolható. Ez az utasitás az m jelzésü berendezésről átviszi az ott soronkövetkező számokat az akkumulátor mantissza részébe (MA).

Az m értékei a következők lehetnek: 0001, 0002, 0004, 0010, 0020 ... 4000.

6. Ciklusszámláló tartalmának átvitele ferritmemória cimre: 27 a

Jelentése: a ciklusszámláló tartalmát (egy n számot és a teljes szóhossz jelét) átviszi az a rekesz 7-19-ig terjedő bitjei helyére és az 1-6-ig terjedő bitek helyére beirja a 25-ös kódot.

Itt ω nem változik; φ = 0.

Vezérlési utasitások

1. Feltételes beállitott ugrás: 23 k

Jelentése: ha a pult k-adik "ugró kapcsolója" (ilyen 7 db van) bekapcsolt állapotban van, akkor a 23 k után következő utasitás végrehajtása elmarad, ellenkező esetben nem.

2. Feltételes ugrás: 21 a i, ahol i = 0, vagy 4.

Ennek az utasitásnak két változata van

a/ 21 a 0 (azaz i = 0)

Jelentése: az utasitás a soronkövetkező utasitásra adja át a vezérlést, ha ω = 0, és az a cimü utasitásra, ha φ = 1.

b/ 21 a 4 (azaz i = 4)

Jelentése: az előző utasitás ellentétje.

3. Feltétlen ugrás: 22 a i, ahol i = 0, vagy 4.

a/ 22 a 0 (azaz i = 0)

Jelentése: ez az utasitás az a cimü utasitásra adja át a vezérlést (a)

b/ 22 a 4 (azaz i = 4)

Jelentése: a vezérlést az a + 1 cimü utasitásnak adja át és ugyanakkor az a cimü rekeszbe beviszi a 22 b+10 utasitást, ahol b annak a rekesznek a cime, amelyben a 22 a 4 utasitás van. Ezt az utasitást szubrutinok behivásánál alkalmazzuk.

4. Ciklus vezérlés

A ciklus itt is két utasitással valósitható meg.

a/ Ciklus beállitó: 25 n

Ezt az utasitást a ciklus elejére helyezzük el.

Jelentése: a ciklusszámláló regiszterbe bemegy n. Az n számot az utasitásban a 7-18-ig terjedő bitek helyére irjuk. A 19. helyérték viszont a következő módon vesz részt a ciklus vezérlésében: ha ott 0 áll, akkor a gép a ciklust n + 1-szer hajtja végre (azaz rövid szavakkal), ha pedig 1, akkor $\frac{n}{2}$ + 1-szer, azaz hosszu szavakkal. (Ebben az esetben az n számnak párosnak kell lennie.)

b/ Ciklusugrás: 24 a

Jelentése: a vezérlést az a cimü utasitásra adja át, ha a ciklusszámláló tartalma nem 0, és a 24 a után következő utasitásra ellenkező esetben.

Kiirási utasitások

1. Kiirás tizes számrendszerben: 32 m

Jelentése: az akkumulátor tartalmát kiirja az m jelzésü kimeneti berendezésen tizes számrendszerben.

2. Kiirás nyolcas számrendszerben: 33 m

Jelentése: u.az mint az előbbi, azzal a különbséggel, hogy itt a kiirás 8-as számrendszerben történik.

3. Soremelő: 34 m

Jelentése: az m jelzésü kimeneti berendezés egy sort emel.

Megállási utasitás: 37 a

Jelentése: A gép befejezi a számitást.

Csoportos utasitások

A külső memóriákhoz csoportos utasitásokkal lehet fordulni. Mindegyik csoportos utasitás három rövid szóból áll.

Az első szó müveleti kód részébe (első hat bit) kerül a csoportos utasitás müveleti kódja. Az első és a harmadik szó cim része a ferritmemória valamely cimét jelenti. A második szó a külső memóriákról szóló információt tartalmazza: a mágnesszalag zóna számát, illetve a mágnesdob kezdő cimét. A dob cimeinek sorszámozása 0-val kezdődik: egy dob esetén a lehetséges cimek 00 0000-tól 03 7777-ig terjednek. Több dob összekapcsolása esetén a sorszámozás folyamatos: pl. a 8 dob esetén a cimek 00 0000-tól 37 7777-ig terjedhetnek.

A csoportos utasitások, kivéve a perforált szalag utasitásokat, csak rövid szavakra vonatkozhatnak.

A csoportos utasitások végrehajtása közben a gép felhasználja a ciklusszámlálót. Ezért, ha ilyen utasitások va-

lamely ciklus közben fordulnak elő, a ciklusszámláló tartalmát egy segédrekeszben tárolni kell (27-es utasitással). Ha csoportos utasitásokra alkalmazunk utasitás módositó müveletet (pl. 34-es utasitás), ugy az csak a csoport első tagjához nyul hozzá.

A csoportos utasitások előtt az akkumulátor törlődik és a müvelet végrehajtása után benne megjelenik az átvitt számok kontrollösszege. (Ellenőrzési célra felhasználható.)

Itt: ω nem változik; $\varepsilon = 0$.

1. Olvasás lyukszalagról:

50 a_k

00 c

00 a_v

Jelentése: a perforált szalag c sorszámu zónájáról a gép az információt beolvassa a ferritmemória a_k -tól a_v-ig terjedő rekeszeibe. A lyukszalagon tárolt információ lehet rövid vagy hosszu szavakból álló binárisan kódolt decimális információ. Hosszu szó esetén az a_k és a_v cimeknek párosaknak kell lenniök és a szokott 4-essel jelölni kell a hosszu szót. Binárisan kódolt decimális számok esetén a zónaszámot 1000-el meg kell növelni.

2. Olvasás a mágnesdobról

Két változata van

a/ 51 a_k 0

b_k 0

00 a_v 0

Jelentése: a mágnesdob b_k rekeszétől kezdve beolvassa

a dobon lévő információt a ferritmemória a_k-tól a_v-ig terjedő rekeszeibe.

b/ 51 a_k 4

b_k 0

00 a_v 0

Jelentése: az utasitás összegezi a mágnesdob b_k-tól $b_k + (a_k - a_v)$ -ig terjedő rekeszeinek tartalmát anélkül, hogy a ferritmemóriába beirná. (Az olvasási kontroll céljára szolgál.)

3. Irás a mágnesdobra:

52 a_k 0

b_k 0

00 a_v 0

Jelentése: a ferritmemória a_k-tól a_v-ig terjedő rekeszeinek tartalma átmegy a dob b_k cimtől kezdődő rekeszeibe.

4. Olvasás a mágnesszalagról

a/ 53 a_k 0

00 c 0

00 a_v 0

Jelentése: a mágnesszalag c sorszámu zónáján lévő információt átviszi a ferritmemória a_k-tól a_v-ig terjedő rekeszeibe.

b/ 53 a_k 4

00 c 0

00 a_v 0

Jelentése: kiolvassa, összegezi a mágnesszalag c sorszámu zónájának $1 + (a_k - a_v)$ számu szavát, de a ferritmemóriába nem irja be. A mágnesszalag kontrollja céljára használatos.

5. Irás a mágnesszalagra

54 a_k 0

00 c 0

00 a_v 0

Jelentése: a ferritmemória a_k-tól a_v-ig terjedő rekeszeiben lévő információt átviszi a mágnesszalag c sorszámu zónájába.

Az URAL-2 gépre program-példát nem irunk fel; a program strukturája azonos az URAL-1 gépre irt programokkal.

III. fejezet

FORMÁLIS GÉPI NYELVEK, AUTOMATIKUS PROGRAMOZÁSA

Az előző fejezetben az elektronikus számológépek "kódos" programozásának elveivel ismerkedtünk meg: megmutattuk, hogy egy adott gép utasitásrendszerével hogyan lehet felirni valamely feladat gépi programját.

A programozó matematikusok kezdettől fogva célul tüzték ki a programozási munkák automatizálását is.

E feladat egyik oldala, olyan nyelv megkonstruálása, amelyen az algoritmusok elég pontosan leirhatók ahhoz, hogy az e nyelven felirt programot egy adott gép utasitásrendszerével értelmezni, ill. az adott gép nyelvére a gép által leforditani lehessen. A feladat másik része olyan konkrét programok elkészitése az egyes gépekre, amelyek a formális nyelven felirt programot lefordit ják adott gépi nyelvre.

A programozás automatizálása különböző foku lehet.

Legegyszerübb - de mégis jelentősen csökkenti a programozás közben fellépő hibalehetőségeket - egy olyan program, amely egy un. relativ cimekkel felirt programot értelmez, illetve ir át konkrét cimekre. Ebben az esetben a programot nem kell a gépbe történő bevitel előtt átirni konkrét cimekre; egyszerüen lelyukasztjuk betükkel; az átcimező program feladata a beolvasott betü-cimeket tartalmazó utasitásoknak konkrét cimü utasitásokra való átirása. Ez a program minden betühöz egy "cim-alapot" rendel hozzá, és ehhez a cimalaphoz viszonyitva képezi a megfelelő cimeket. Ha például az átcimezendő utasitás:

× b + 10 c + 316

alaku; akkor az átcimező program az × jelhez hozzárendeli a 03 szorzási kódot; (ha a konkrét gépen ez a szorzás kódja), az első cim b betüjéhez hozzárendeli (a pl. előre megadott) mondjuk 2200 ; a c betühöz a pl: 3300 alapértékeket; s igy ezt az utasitást

03 . 2210 3 616 alakura irja át.

Az alapcimeket (a rekesz-elosztás alapján) általában kintről mint paramétert adjuk meg az átcimező programnak; de lehet olyan átcimező programot is készitени, amely maga végzi el a rekeszelosztást is.

Leginkább megkönnyiti azonban a programozási munkát az un. szimbolikus programozási nyelv ill. módszer.

Ezek közül ismertetjük az ELLIOTT-803 gép autókódját. A végső cél természetesen olyan formális algoritmikus nyelv létrehozása, amely univerzális nyelv lehet: olyan t.i. amely egyfelől a matematikai nyelvekhez igen közel áll (azaz: az algoritmusok közlésének emberi nyelve is lehet), másfelől elegendően preciz ahhoz, hogy e nyelven felirt programokat egy gép egyértelmüen végrehajthasson; végeredményben tehát az emberek és gépek közös nyelve legyen.

Várható, hogy az ALGOL nevü nyelv - amelyet a következőkben szintén ismertetünk, alapja lesz egy ilyen nyelvnek.

Megjegyezzük, hogy kisérletek történnek olyan gépek épitésére is, amelyek formula nyelven programozhatók anélkül, hogy olyan programmal kellene ellátni őket, amelyek a formális nyelven felirt programot leforditják. Az ilyen gépek "utasitásai" formula - müveletek lennének. E tekintetben jelentősek Kalmár László akadémikus eredményei.

1. AZ ELLIOTT-803 GÉP AUTÓKÓDJA

A különféle autokódnak közös jellemzőjük, hogy a gépi utasitásokhoz közelálló (- betükből valamint a megszokott müveleti jelekből felépitett -) formulákat tartalmaznak; az algoritmust ilyen formulák sorozatával kell felirni. Nagy előnyük, hogy a segitségükkel felirt programok jól áttekinthetők; az autokódos programozási módszerek könnyen elsajátithatók.

A következőkben az ELLIOTT-803 autokódját (a továbbiakban E-autokód) ismertetjük részletesen; főleg azért, mert a jelek szerint hazánkban ELLIOTT-803 gép a közeljövőben több példányban is üzemel.

1.1 Az E-autokód alapelemei

Az E-autokód programok az ABC nagybetüiből és a +, -, , (,) , =;@ ; ∅, %, :*, jelekből, továbbá a 0,1,2,. . ., 9 számjegyekből, illetve a felsorolt jelekből meghatározott szabályok révén alkotott kifejezésekből állnak. Kifejezésen a fenti jelek valamilyen - szabályok által megengedett - kombinációját értjük.

1.2 Változók

Változónak egy - esetleg valamilyen indexszel ellátott - nagybetüt nevezünk. Index lehet egy konstans, egy másik változó, illetve a változók valamilyen kifejezése. Az index jelölésére a (kezdő és a) végzárójelet használjuk fel. A müveleti jelekkel összekapcsolt változókat formuláknak hivjuk.

Amikor az E-autokódot valamilyen feladat programjának felirására használják fel, akkor egy változó lényegében a gép egy meghatározott cimét jelenti. Nevezzük a változó által reprezentált cim tartalmát a változó aktuális értékének. Az "aktuális" jelző azt fejezi ki, hogy a változónak "munkarekesz" jelentése van, tartalma, értéke dinamikusan változik a program végrehajtása során.

Annak megfelelően, hogy egy cim által jelölt rekesz tartalma lebegőpontosan, illetve fixpontosan ábrázolt szám lehet, kétféle változótipust különböztetünk meg.

Állapodjunk meg abban, hogy az E-autokód könnyebb leirása céljából az alábbi jelöléseket használjuk a következőkben:

A, B, C, C mindig lebegőpontos,

I, J, K, L mindig fixpontos változót,

l, m, n pozitiv egész konstansokat,

p, q, r tetszőleges egész konstansokat,

x, y, z lebegőpontos konstansokat

jelentenek. (A konstansokat a programokban numerikusan, számokként irjuk fel.)

Ezekkel a jelölésekkel a lebegőpontos ill. fixpontos változók az E-autokódban az alábbi alakuak lehetnek:

Lebegőpontos változók:

A (I+n) vagy A (n+I)

A (I-n)

A (n-I)

A (I $\pm$ J)

A (mI $\pm$ J)

A (mI)

A $(mI \pm n)$

A $(m \pm nI)$

Fixpontos változók

K $(I+n)$ vagy K $(n+I)$

K $(I-n)$

K $(n-I)$

K $(I \pm J)$

K $(I \pm nJ)$

K $(mI \pm J)$

K (mI)

K $(mI \pm n)$

K $(m \pm nI)$

Megjegyezzük, hogy a

1. K (0) változó jelével ekvivalens a K ill. KO írásmód;

2. és ehhez hasonlóan A (0) helyett irhatunk A-t, ill. AO-t is.

3. Ha a változó indexe konstans vagy egyetlen változó, a zárójelet elhagyjuk.

A változók fenti felirásánál kiköt jük, hogy az indexek nem lehetnek negativok.

1.3 Konstansok

A konstansok ugyancsak lehetnek lebegőpontosak és fixpontosak: ha egy konstans olyan kifejezésben szerepel, amelyben a változók lebegőpontosak, akkor az lebegőpontos, ha pedig fixpontos változók mellett áll, akkor fixpontos konstans.

Fixpontos konstansok. A fixpontos konstansokat a megszokott módon, mint egész számokat irjuk fel. Pl: 6; 6318; - 636; (pontot nem szabad az egészek után irni: azaz pl. 6318.0 nem megengedett irásmód). A legnagyobb ábrázolható egész szám 12 jegyet tartalmazhat és kisebbnek kell lennie 274 877 906 944 -nél.

Lebegőpontos konstansok. A lebegőpontos konstansokat többféle alakban is beirhatjuk. Ha egészeket használunk, mint lebegőpontos konstansokat, akkor ezeket a szokásos módon egész számként is irhatjuk: pl: 5;- 316; de irhatók ezek 5.0; - 316.0 alakban is. A tizedestörtekben a tizedespontos jelölést alkalmazzuk: pl. 36.723; -51.002; -.00072 (a tizedespont előtti értéktelen nullákat nem irjuk ki). Az ábrázolható lebegőpontos számok intervalluma a $4.3.10^{-78}$ $5.8\ 10^{78}$ intervallum. Az eredmények illetve a kiinduló adatok kitevős alakban is irhatók: pl. 365.72-t 3.6572/2 alakban irhatjuk, ami megfelel $3,6572.10^{2}$ -nek. Az adatszalagon a pozitiv előjelet nem kell kiirni.

1.4 Müveletek az E-autokódban

A fent bevezetett változók különféle müveletek argumentumai lehetnek. Megjegyezzük, hogy az E-autokóddal leirt algoritmusok gép által történő végrehajtása során a müveletek a változók által reprezentált cimek tartalmain végzett müveleteket jelentik.

a/ Átküldési vagy "értékadó" müvelet. A müvelet jele: =; pl. A = B azt jelenti, hogy a két változó értéke a müvelet végrehajtása után azonossá válik, azaz a két változó által jelölt két rekesz tartalma ugyanaz lesz; pontosabban az A változó uj értéke a B változó értéke lesz. Az E-autokódban az = jel jobb oldalán egy müveleti jellel összekö-

tött két változóból álló kifejezés is állhat, baloldalán pedig egy változó;pl.A = B⊖C, ahol ⊖ valamilyen müvelet. Állapodjunk meg abban, hogy értékadó formulának hivunk egy olyan kifejezést, amelyben átküldési müvelet is van.

Megjegyzés: az = jobboldalán konstans is szerepelhet: A = 2.5 is megengedett.

b/ Aritmetikai müveletek

Lebegőpontos müveletek

1. Összeadás: (jele: +)

A = B + C jelentése: az A változó értéke a B változó értékének és a C változó értékének összege lesz.

2. Kivonás (jele: -)

A = B - C jelentése: az A változó értéke a B és C változó értékének különbsége lesz.

3. Szorzás (jele: *)

A = B * C jelentése: az A változó értéke B és C értékének szorzata lesz.

4. Osztás (jele: /)

A = B/C jelentése: az A változó értéke a B változó és C változó értékének hányadosával lesz egyenlő.

Megjegyzés:

a/ Mind a négy müveletben az egyenlőség jobboldalán álló változók helyett konstansok is szerepelnek. Pl.: A = 2.5 + B; vagy A = B/2.5

b/ Az = jel után közvetlenül a változó előtt állhat negativ előjel is (-). Pl: A = -B+0; A = -4.5*C stb.

Példák:

A (2J) = C (I + 3J) - D5

B37 = -C (I + 6) / 6.5

D (2I - J) = C (K - 2J)** A (3 - I)

Fixpontos müveletek

A fixpontos müveleteket a lebegőpontos müveletekhez hasonlóan irjuk fel:

Összeadás: J = I + K

Kivonás: J = I - K

Szorzás: J = I * K

Osztás: nincs értelmezve.

Ezeknél a müveleteknél is szabad a müveletek argumentumait konstansokkal (természetesen fixpontos konstansokkal) felcserélni.

Pl: J = K + 1

J = J - 3

c/ Függvénymüveletek

A függvénymüveletek az E-autokódban valamely elemi függvény kiszámitását elvégző szubrutin behivását jelenti.

Az E-autokódban az alábbi függvénymüveletek szerepelnek:

1. A = SIN B

Jelentése: A értéke π B értékének sinusa lesz. B értékét, azaz az argumentumot π egységekben adjuk meg.

2. A = COS B

Jelentése: A értéke π B értékének cosinusa lesz; azaz A = cosπB

3. A = TAN B

Jelentése: A értéke π B értékének tangense lesz; azaz A = tgπB

4. A = ARCTAN B

Jelentése: A értéke B értéke arcustangensének $\frac{1}{\pi}$ -szerese lesz; azaz A=$\frac{1}{\pi}$ arctg B

5. A = LOG B

Jelentése: A értéke B értékének természetes logaritmusa lesz, azaz A = ln B

6. A = EXP B

Jelentése: A = e^B

7. A = SQRT B

Jelentése: A = $\sqrt{B}$

8. I = INT B

Jelentése: A = [B] (képezi B egész részét)

9. A = FRAC B

Jelentése: A = {B} (törtrészképzés)

10. A = MOD B

Jelentése: A = |B|. (Abszolutértékképzés)

11. A = STAND I

Jelentése: A értéke az I fixpontos változó értékének lebegőpontos alakja lesz.

d/ Vezérlés átadó müveletek

A referencia szám fogalma. Egy autokódban felirt algoritmus lépéseinek végrehajtási sorrendje általában a formulák természetes sorrendje. Az E-autokódban azonban éppugy

mint a gépi utasitásrendszerekben is - szerepelnek vezérlés átadó müveletek is. Ezek alkalmazásához valamilyen módon meg kell jelölni a formulákat. Egyik módja ennek az lenne, hogy minden formulához egy számot, mondjuk a felirásban meghatározott helyük sorszámát irnánk fel. Az E-autokódban nem ezt a módszert alkalmazzák; csak azokhoz a formulákhoz rendelnek hozzá valamilyen számot, amelyekre vezérlésátadó utasitás adja át a vezérlést. Az hogy egy formulát milyen számmal jelölünk meg, tetszőleges; természetesen az egyértelmüséget biztositani kell. A formulák jelzőszámait referenciaszámoknak hivjuk. A referenciaszám jele egy egész szám és a) jel. Pl.:

A 20 = BJ + D10

2)BK = C + D (J + 10)

C = =.5 * D10

⋮

Itt a 2) BK = C + D (J + 10) egy referenciaszámmal ellátott formula.

1. Ugró müveletek

a/ Feltétlen ugrás

Jele: JUMP @ n

Jelentése: az algoritmus következő lépése az n referenciaszámmal ellátott formula.

b/ Feltételes ugrás. Ennek a müveletnek az argumentuma egy logikai formula. (Olyan formula, amelyhez két értéket rendelünk hozzá: az egyik érték azt jelöli, hogy a formula teljesül, a másik azt, hogy nem.) A feltételes ugrás itt is - éspedig argumentumának két lehetséges értékétől függően - két formulára adhatja át a vezérlést.

Az E-autokódban kétféle feltételes ugrás van.

1/ A JUMP IF F_1 ⊕ F_2 @ n (JUMP = ugrás, IF = ha) ugrómüvelet jelentése a következő: az algoritmust az n referenciaszámmal rendelkező formulánál kell folytatni, ha az F_1 ⊕ F_2 logikai formula által kijelölt feltétel teljesül, és a soronkövetkező formulával az ellenkező esetben. F_1 itt mindig valamilyen változót, F_2 valamilyen formulát jelent; a ⊕ jel pedig az =, %, ≯, jelek valamelyike (a < jel aperforáló klaviaturáján nem szerepel, ezért ez a ≯ jellel, a > jelet pedig a % jellel helyettesitik.)

Pl.

JUMP IF A = B + C @ 6

JUMP IF A %B/2.6 @ 13

JUMP IF A (I + 5J) ≯ - CI - B (2J + 5) @ 13

Itt pl. a második sor jelentése a következő: a vezérlést a 13-as referenciaszámmal ellátott formulára adjuk át, ha $A > \frac{B}{2.6}$ és a soronkövetkezőre az ellenkező esetben.

2/ JUMP UNLESS F_1 ⊕ F_2 @ n

Ez a müvelet az előző müvelet tagadása. Jelentése: az n referenciaszámmal ellátott formulánál kell folytatni az algoritmust, - kivéve ha teljesül a F_1 ⊕ F_2 formula által kifejezett feltétel, mert akkor a soronkövetkező formulára kell rátérni.

A F_1 és F_2, valamint a ⊕ jelek jelentése ugyanaz mint az előbb.

Megjegyzés: az ugró müveletekben az n referenciaszám helyett fixpontos változót is irhatunk; ilymódon arra a referenciaszámra adhatjuk át a vezérlést, amely az emlitett változó aktuális értékével egyenlő. Pl. JUMP IF A % B @ I.

e/ Ciklusmüveletek

A ciklikus eljárások felirására az E-autokódban kétféle ciklusmüveletet vezettek be.

A/ CYCLE - tipusu ciklusos müveletek

a/ CYCLE-1 müvelet. Ezt a müveletet két kifejezés segitségével irjuk le. Az egyik kifejezést a ciklus magját alkotó M program előtt az alábbi alakban irjuk fel:

CYCLE I = J:K:L.

Ez a kifejezés arról tartalmaz információt, hogy a végrehajtandó ciklikus eljárásban mely változót tekintjük a ciklus paraméterének, mi a ciklus lépésköze és kijelöli a ciklus lépéseinek számát.

A ciklusmüvelet másik kifejezését az M program után irjuk és ennek alakja

REPEAT I

Egy ciklikus eljárás alakja e két kifejezés segitségével a következő:

CYCLE I = J:K:L

M program amelynek valamely változója, vagy változói függnek az I indextől.

REPEAT I.

Ebben az esetben a ciklus M magját a gép végrehajtja az I = J, J+K; J + 2K, . . . , L értékekre; azaz $\frac{L - J}{K}$ + 1-szer. Lényeges, hogy $\frac{L - J}{K} \geq 0$, és feltétlenül egész legyen.

A J, K, L változók negativ előjelüek is lehetnek, azaz van értelme a

```
CYCLE I = - J :-K :-L
```

felirásnak is; ezenkivül mindhárom változó helyettesithető konstansokkal is; pl:

```
CYCLE I = 1:1:32.
```

Ez utóbbi jelentése a fentiek alapján: végrehajtandó a ciklus I-nek: 1; 1+1=2; 1+1+1=3, ..., 32 értékeire.

Példa a CYCLE-l utasitásra:

```
CYCLE I = 1:1 : 37
```

```
M30 = 2.5*  A.
 :
A 20 = BK + CI
 :
B I+20  = C - DI
 :
```

A ciklus magja.

```
REPEAT I
```

A fenti programot a ciklus első lépésében az I szerint változó utasitásokra az

```
A20 = BK + C1
B21 = C - D1
```

a második lépésben az

```
A 20 = BK + C2
B22 = C - D2  s.i.t.
```

alakokkal hajtja végre a gép. A fenti ciklusmag I szerint változó részének az utolsó végrehajtott lépésben

$$A20 = BK + C37$$

$$B57 = C - D37$$

lesz az alakja.

Megjegyzés: az I ciklus-változó az egész változókra megengedett feltételeknek eleget tevő tetszőleges alakú változó lehet (Pl. I/3J-6/). A J,K,L változóknak azonban csak index nélküli alakja megengedett.

Pl:

CYCLE K (6I-2) = J:L:20 megengedett de

CYCLE K = L (20) : J (K+2): 30 nem.

b/ CYCLE-2. Ez a müvelet az előbbihez hasonlit; a különbség az, hogy a ciklus változói nem egészek, hanem lebegőpontosak. Alakja:

CYCLE a = B : C : D

a ciklus magja

REPEAT A.

Jelentése: a ciklus A-tól függő magját végrehajtja az A = B; B + C; ...; D értékekre. ($\frac{D-B}{C} \geq 0$ feltétel teljesülése esetén.) Ez a müvelet a ciklust a $\frac{D-B}{C}$ + 1-hez közelebb eső egész számszor hajtja végre.

Pl: CYCLE A = 1:1:5.2 esetén a gép a ciklusmagot az A = 1; 2; 3; 4; 5.2 értékekre;

CYCLE A = 1 : 1 : 5.8 esetén az

A = 1; 2; 3; 4; 5; 5.8 értékekre hajtja végre.

Itt is, mint az előbbi CYCLE-1 müveletben B,C,D változók elé negativ előjel is irható, azaz az értelmezett a

CYCLE A = -B : -C : -D

alak is;
valamint a B,C,D változók lebegőpontos konstansokkal is helyettesithetők: pl:

CYCLE A = 3,6 : 2 : 76

Megjegyzés: az A változó a fenti felirásban bármilyen - a megengedett - indexekkel ellátott változó lehet; B, C, D azonban csak index nélküli.

c/ CYCLE-3. A müvelet alakja:

CYCLE I = p, q, r, ...

a ciklus magja

REPEAT I

Jelentése: a ciklus I-től függő magját végrehajtja az I = p, q, r, ... értékekre. (A p, q, r konstansok száma nincs korlátozva; legfeljebb a memória kapacitás által.)

Az I változó tetszőleges indexes változó lehet.

d/ CYCLE-4. A müvelet alakja:

CYCLE A = x, y, z, ...

a ciklus magja

REPEAT A.

Jelentése: a ciklus magját végrehajtja az A = x, y, z, ... értékekre.

Az A változó tetszőleges indexes változó lehet.

B/ VARY - tipusu ciklusos utasitások

A másik tipusu ciklus-utasitásokat VARY-utasitásoknak hivják. Ennek is több fajtája van.

a/ VARY-1.

A müvelet alakja:

VARY I = J : K : L

| a ciklus magja |

REPEAT I.

Jelentése: a ciklus I-től függő magját végrehajtja az I = J, J+K, J+ 2K; ... ; J + (L-1) K értékekre, azaz összesen L-szer. (Az L változó értéke adja meg a ciklusok számát.)

A J és K változók negativ előjelet is kaphatnak; az L azonban szükséges, hogy pozitiv legyen. A J, K, L változók helyett fixpontos konstansok is irhatók, azaz értelmezve van a

VARY I = 2 : 5 : 39

felirás is.
Az I tetszőleges indexes változó lehet; J, K, L azonban nem kaphatnak indexet.

b/ VARY-2.

A müvelet alakja:

VARY A = B : C : L

a ciklus magja

REPEAT A

Jelentése: a ciklus A-tól függő magját végrehajtja az A = B, B + C, B + 2C, ... , B + (L-1) C értékekre, azaz L-szer (az L változó tartalma adja meg a ciklusok számát).

A B és C változók elé negativ előjel is irható, L elé azonban nem; ezenkivül B és C helyébe lebegőpontos konstans, L helyébe pedig fixpontos konstans irható.

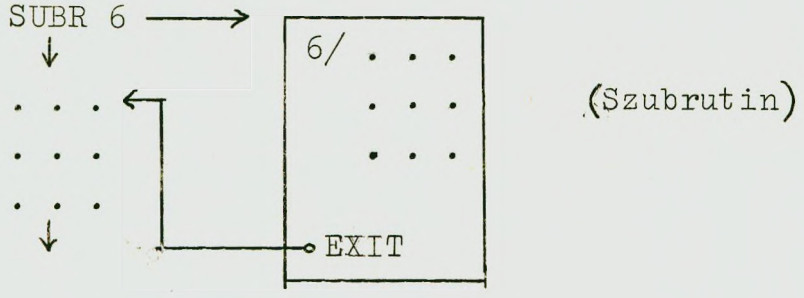
Az A tetszőleges indexes változó lehet, B, C, L azonban csak index nélküli.

f/ Szubrutin-müvelet az E-autokódban

Szubrutinok az E-autokódban egyszerüen hivhatók be. (A szubrutin itt is egy olyan programrészt jelent, amit a főprogram többször felhasználhat: ehhez egyszerüen csak a szubrutin argumentum helyeit ("bemenet") kell megfelelően előkésziteni, és a vezérlést a szubrutinra át kell adni.)

A szubrutin behivásának jele: SUBR n, ahol n a behivandó szubrutin első végrehajtandó formulájának referenciaszáma (a szóbanforgó formulának a referenciaszámát szokás a szubrutin referenciaszámának is hivni; természetesen ezen a referenciaszámon kivül a szubrutinban más referenciaszámok is szerepelhetnek.) A fenti behivó müvelet nem tartalmaz információt arra vonatkozóan, hogy a szubrutin vég-

rehajtása után melyik formulánál kell folytatni az algoritmust. Megállapodás szerint minden szubrutint el kell látni egy EXIT kifejezéssel; ez jelzi a szubrutin végét; (innen ugrik vissza a vezérlés a főprogramra), ennek hatására a vezérlés a SUBR n kifejezést - azaz a szubrutin behivásának helyét - természetes sorrendben követő formulára adódik át:



Megjegyzés: szubrutinba behivható egy másik szubrutin. (Az E-autokódban legfeljebb 6 egymásba skatulyázott szubrutin szerepelhet.)

g/ Beviteli müveletek

Minthogy az E-autokód gépi nyelv, ezért müveletei között szerepelnek beviteli müveletek is.

a/ Az egyik beviteli művelet jele: READ. (read=olvasni)

READ A azt jelenti, hogy az A változó értéke a lyukszalagról ezzel a művelettel beolvasott érték, szám lesz. Ugyanezt jelenti a READ I is (az előbbinél lebegőpontos változóról, itt fixpontosról van szó!)

b/ A fenti beviteli müveleten kivül az E-autokódban szerepel egy INPUT jelü beviteli művelet is. Az INPUT I utasitás hatására a gép egy jelet (5 bit; a lyukszalagon egy sor) olvas be.

h/ Kiirási müveletek

Az E-autokódban a következő kiirási müveletek vannak.

a/ PRINT-művelet

Ennek a müveletnek többféle tipusa van.

1. A PRINT A,m:n tipusu kiirási müvelettel a gép az A változó értékét ugy irja ki, hogy m jegyet ir ki a tizedespont előtt, és n jegyet a tizedespont után. Pl. ha A = 27.564235 és a kiirási művelet alakja PRINT A, 2:3, akkor a gép a 27.564 számot irja ki. Ezzel a művelettel elérhető, hogy a kiirásnál a tizedespont mindig ugyanazon a helyen maradjon. Pl:

27.564
7.323
.216
.016 stb.

Az értékes jegyek előtt lévő nullákat a gép nem irja ki, de a helyüket meghagyja.

2. PRINT A, m. E művelettel a gép kiirja az A változó értékének m első jegyét (tizedesponttal ellátva).

3. PRINT A, m/. E művelettel a gép kiirja az A normálalakban felirt értékéből a mantissza m jegyét, és a kitevőt. Pl.: PRINT A, 5/ esetén, ha A = $.5643215.10^4$, akkor .56432/4-et ir ki.

4. PRINT A, művelettel a gép kiirja az A változó normálalakban tekintett értékét teljes mantisszával és kitevővel, Pl:

.564376764/-3.

5. PRINT I, m esetén a gép az I fixpontos változó m első jegyét irja ki.

6. A PRINT I müvelettel a gép kiirja az I fixpontos változó teljes értékét.

b/ TITLE-müvelet

A TITLE müveleti jel hatására a gép kiirja az e jel után következő szöveget.

Pl.: "TITLE az A mátrix inverze a következő lesz." esetén a gép kiirja a fenti szöveget. A TITLE müvelet bármely formula után és előtt szerepelhet. (A szöveg végére lyukasztáskor néhány szinkron jelet kell perforálni.)

c/ LINE-müvelet. E jelre az E-autokódban a kiiróberendezés egy soremelést hajt végre. Ha több sort akarunk emelni, akkor "LINES n" jelet kell irni: e jel hatására a kiiróberendezés n sort emel. Ezen kivül "LINES I" is értelmezve van: a soremelések száma az I változó aktuális értéke.

d/ SPACES n-müvelet (vagy SPACES I). E jel hatására a kiiróberendezés n térközt hagy ki. (SPACES I esetén a kihagyott térközök száma az I változó értékével egyenlő.)

e/ OUTPUT n- müvelet

Jelentése: kiirja az n-edik (konkrét rekesz) utolsó 5 bitjét. Ugyanezt teszi az I változóval az OUTPUT I müvelet is.

f/ CHECK-müvelet

A CHECK A (ill. CHECK I) jel hatására a gép kiirja A -t, ha a vezérlőpulton egy adott kapcsoló a megfelelő helyzetben van. (Program probálásnál a közbülső eredmények kinyomtatásánál igen hasznos.)

i/ Megállási müveletek

Az E-autokódban kétféle megállás van:

a/ A STOP jel az algoritmus végét jelzi.

b/ A WAIT jel hatására a gép megáll; "vár". A vezérlőpulton lévő inditógomb megnyomása után a gép továbbfolytatja az algoritmust.

j/ Deklaráló-formulák

Ha egy algoritmust E-autokódban irunk fel és azt a konkrét gép utasitásrendszerére fordittatjuk le (az adott géppel), akkor néhány kiegészitő információval kell ellátni az autokódban felirt programot. Rögziteni kell például azt, hogy a programban szereplő változók közül melyek lebegőpontosak és melyek fixpontosak. Ilyen célokra az E-autokódban az alábbi"deklaráló formulákat" vezették be.

1. A SETS kifejezés után a program elején fel kell sorolni az összes fixpontos változókat, mégpedig ugy, hogy minden fixpontos változóként használt betü után zárójelben be kell irni az adott betü mellett a programban előforduló legmagasabb index értékét. Pl. SETS E (20) F(82) G(38). Ez a felirás azt jelenti, hogy a programban legfeljebb az E1, E2, ... E 20; F1, F2, ..., F82; G1, G2, ..., G 38 változók szerepelhetnek. (Nem szükséges, hogy mind feltétlenül elő is forduljon a programban; a lényeg csupán az, hogy a valamely betü mellé a fenti felsorolásban irt indexszámnál nagyobb indexü betü ne álljon sehol a programban.)

2. A SETV kifejezés után az előbbihez hasonló módon a programban előforduló lebegőpontos változókat soroljuk fel. Pl. SETV A(20) B C (31) D(16).

3. A SETF-kifejezés után a programban felhasznált elemi függvényeket kell felsorolni.

Pl: SETF LOG SIN SQRT. A MOD, STAND függvényeket nem szükséges beirni. A SIN, COS függvények helyett egyszerüen TRIG-et is irhatunk.

4. A SETR -kifejezés után a programban előforduló legnagyobb referenciaszám értékét kell beirni.

Pl: SETR 20 azt jelenti, hogy a programban 20-nál nagyobb referenciaszám nem szerepel. (Nem szükséges, hogy minden 20-nál kisebb elő is forduljon a programban.)

5. A START n kifejezés - amelyet mindig az algoritmus legutolsó leirt formulája után irunk - az algoritmus elinditásához szükséges: az n szám itt annak a formulának a referenciaszáma, amelynek végrehajtásával a programot elkezdjük.

k/ Hibajelző jelek az autokódban

1. Lebegőpontos tulcsordulás

Ez a hiba aritmetikai müveleteknél, valamint a READ, TAN, EXP függvényeknél fordulhat elő. Ebben az esetben a gép kiirja az @ jelet, és a pulton "kigyullad" a "lebegőpontos tulcsordulás" lámpa.

2. Trigonometrikus-argumentum tulcsordulása

Ha a trigonometrikus függvényeknél az argumentum 2^{28}-nál nagyobb, akkor a gép 1-esek sorozatát (folyamatosan) irja ki.

3. LOG-hiba

Ha a logaritmus argumentuma negativ, vagy 0, a gép 2-esek sorozatát irja ki.

4. e^x-hiba

Ha e^x értéke tulcsordul, akkor a lebegőpontos tulcsordulás mellett ezt 3-asok sorozatával jelöli.

5. INT-hiba

Ha az egészrész képzés közben az argumentum tul nagy, ezt 4-esek sorozatával jelzi.

6. SQRT-hiba

Ha negativ számból kellene gyököt vonni, a gép azt 5-ösök sorozatával jelzi.

7. SUBR-hiba

Ha 6-nál több szubrutin van egymásba skatulyázva, akkor a gép $ jelsorozatot ir ki.

8. READ-hiba

A kiiró berendezés távköz-sorozatot lyukaszt ki, ha a szalagon valamilyen hibás jel van.

1/ Gépi kód beiktatása az autokód programba

Az E-autokód program tartalmazhat gépi kódban felirt blokkokat is. Az ilyen blokkot a szokásos módon felirt referenciaszámmal látjuk el (pl. 5/ és a referenciaszám után @ jelet irunk vagy csak az @ jellel jelöljük. (A referenciaszámot a blokk első szava elé kell irni.) A blokkot) jellel zárjuk.

A gépi kód programban az 12, 13, 14 cimü rekeszeket használhatjuk fel munkarekeszekként.

Gépi kódos blokkra vagy egy autokódos JUMP müvelettel lehet "ráugorni", amely a blokk referenciaszámát tartalmazza, vagy egyszerüen beirjuk @ jellel ellátva autokódos utasitások közé egyszerü folytatásként. (Ilyenkor referenciaszám természetesen nem kell.)

A gépi kód blokkból vagy a gép ugróutasitásaival lehet "kiugorni" (ezek cimrészébe azt a referenciaszámot kell beirni, amelyre a vezérlést át akarjuk adni), vagy egyszerüen ugy épitjük be, hogy a blokk) jele után az autokód soronkövetkező utasitására térjen rá a gép.

A gépi kód blokkot kezelhetjük szubrutinként is, az-az behivhatjuk a SUBR n müvelettel is (itt n a blokkhoz az előbbi módon hozzárendelt referenciaszám). Ebben az esetben a blokkot természetesen EXIT-tel kell bezárni.

A gépi-kód blokk utasitásai tartalmazhatnak abszolut cimeket, a blokk első utasitásrekészére vonatkoztatott relativ cimeket, valamint konstans indexekkel ellátott autokód változókat is.

Összefoglalás

Az alábbiakban táblázatosan összefoglaljuk az E-autokód utasitásait:

Aritmetikai utasitások

A = B;	A = - B;	I = J;	I = -J;
A = B + C;	A = -B + C;	I = J + K;	I = -J + K;
A = B - C;	A = -B -C;	I = j - K;	I = -J - K;
A = B * C;	A = -B * C;	I = j * K;	I = -J * K;
A = B/C;	A = -B/C;		

Függvény utasitások

A = SIN B;	A = LOG B;	A = FRAC B;	-
A = COS B;	A = EXP B;	A = INT B;	I = INT A;
A = TAN B;	A = SQRT B;	A = STAND I;	--
A = ARCTAN B;		A = MOD B;	I = MOD J;

Ugró utasitások

JUMP @ K

JUMP IF A = B@K.

JUMP IF I = J@K

JUMP UNLESS A = B@K.

JUMP UNLESS I = J@K.

(K-nak nem lehet indexe)

A = B vagy I = J -t bármilyen megengedett aritmetikai vagy függvény-utasitás helyettesitheti, továbbá> (telekódban %) vagy < (telekódban ∅) állhat az = jel helyett.

Egyéb vezérlő utasitások

SUBR n ; EXIT ; STOP ; WAIT ;

VARY ÉS CYCLE utasitások

VARY A = B : C : L

CYCLE A = B : C : D

CYCLE A = x, y, z, ...

REPEAT A

VARY I = J : K : L

CYCLE I = J : K : L

CYCLE I = p, q, r, ...

INPUT utasitások

READ A READ I INPUT I

OUTPUT utasitások

PRINT A, n:m PRINT A,n PRINT A,n/ PRINT A

PRINT I,n PRINT I OUTPUT I

(OUTPUT I-ben I-nek csak számindexe lehet).

LINE LINES I SPACES I TITLE

CHECK A CHECK I

Deklaráló és start utasitások

SETS (egész változók)

SETV (lebegőpontos változók)

SETF (függvények)

Az autokódban felirt feladatok futtatása az ELLIOTT-803 gépen

Az autokódban felirt programokat - ugyanugy mint a "direkt" kódban felirtakat - lyukszalagra viszik rá. Ha a programban READ utasitás is szerepel, akkor egy adatszalagot is kell készíteni.

Az autokód programnak gépi kódba való forditása kétféleképpen történhet az E-803 gépen. Az A-2-es fordító program a leforditott programot mindjárt ki is lyukasztja, igy az autokódban felirt algoritmus végrehajtása a kilyukasztott program ujbóli bevitelével bármikor megtörténhet. Az A-3 fordító program viszont a leforditott programot nem nyomtatja ki, hanem a gép memóriájában elhelyezi. Ha a "rekeszelosztás" folyamán a fordító program nem talál valamely programhoz elegendő memóriarekeszt, akkor ezt jelzi és megáll.

Mint emlitettük, az eredményeket az E-803 gépen általában gyorsperforátorral lyukasztják ki; az igy nyert lyukszalagot azután kiiróberendezéssel iratják ki.

Az E-autokód szemléltetése céljából tekintsünk egy példát.

Példa:

Adott a,b paraméterértékek mellett számitsuk ki és nyomtassuk ki az

$$F/x,a,b/ = \frac{\sqrt{x+b} \cdot \ln\left(x^2 - \frac{a}{4}\right)}{\left(\sqrt{x - 0{,}62b}\right)^3}$$

függvény értékeit az $x_o \leq x \leq x_v$ intervallumban adott Δx lépésközzel.

Helyezzük el az adatszalagon az x_o, x_v, Δx,a,b értékeket a felirt sorrendben.

Programunk az alábbi lesz:

```
      SETV K VLAB M/2/
      SETF SQRT LOG EXP
1)    READ K
      READ V
      READ L
      READ A
      READ B
      CYCLE X = K : L : V
      M = X + B
      M = SQRT M
      M1 = X * X
      M2 = A/4
      M1 = M1 - M2
      M1 = LOG M1
      M2 = . 62 * B
```

az x_o, x_v, Δx, a, b adatok beolvasása

$x + b \Rightarrow M$

$\sqrt{x + b} \Rightarrow M$

$x^2 \Rightarrow M1$

$\frac{a}{4} \Rightarrow M2$

$x^2 - \frac{a}{4} \Rightarrow M1$

$\ln/x^2 - \frac{a}{4}/ \Rightarrow M1$

$0{,}62\ b \Rightarrow M2$

```
M2 = X . M2                    x - 0,62 b  ⇒M2
M2 = LOG M2                    ln /x-0,62b/⇒M2
M2 = 1.5 * M2                  1.5 ln /x-0,62b/⇒M2
M2 = EXP M2
M  = M * M1                    √(x-0,62 b)^3 ⇒M2
M = M/M2                       f/a,b,x/ ⇒ M.

TITLE  F (
PRINT A, 3:5
TITLE,
PRINT B, 3:5
TITLE,
PRINT X; 4:6
TITLE ) =
PRINT M
LINES 2
REPEAT X
START 1
```

A program elemzését az alábbi megjegyzésekkel az olvasóra bizzuk.

A beolvasás ciklussal is megvalósitható, ha az öt kiinduló értékhez egymásutáni rekeszeket rendelünk hozzá.

A $\sqrt{(x-0,62\ b)}^3$ kifejezést az

$$x^y = e^{y \ln x}$$

összefüggés alapján számitottuk ki.

A három TITLE-utasitással az alábbi tipusu lesz a kinyomtatott táblázat:

$$F/a,\ b,\ x/ = f_o$$

pl:

$$F/2.8,\ 3.2,\ 6.5/ = 862.467801$$

2. A FORTRAN-NYELV

A FORTRAN (Formula Translator) algoritmikus nyelvet az 1954-1956 években készitették el az IBM - 704 gépre. Röviden - inkább áttekintőleg - ismertetjük e nyelvet is.

A FORTRAN-ban egy adott algoritmust szintén a nyelv "szavainak" sorozatával kell felirni. A szavak itt is az abc nagybetüiből, számjegyekből, a szokványos ill. néhány speciális müveleti jelből állhatnak.

2.1 Konstansok

A fixpontos konstansok előjellel ellátott ötjegyü tizes számrendszerbeli (és abszolutértékben 32768-nál kisebb) számok lehetnek. Irásmódjuk pl: +6; -28789; stb.

A lebegőpontos konstansok előjellel a tizedesponttal ellátott tizes számrendszerbeli számok lehetnek. Pl: 17. 6.823 - .0005 stb. Van lehetőség normál alakban történő irásra is: pl. a 3.4×10^3 számot 3.4 E+3 alakban irjuk fel; a 6.47×10^{-5} számot pedig 6.47 E-5 alakban. (Az ábrázolható lebegőpontos konstansok tartománya a $(10^{-38}; 10^{38})$ intervallum.)

2.2 Változók

Fixpontos változók

Fixpontos változó lehet bármilyen maximálisan 6 betüből és számból álló jelsorozat, azzal a kikötéssel, hogy az első jelnek feltétlenül az I, J, K, L, M, N betük valamelyikének kell lennie.

Pl.: KB2; I20, LAJOS.

A fixpontos változók aktuális értéke tetszőleges, az előzőkben definiált fixpontos konstans lehet.

Lebegőpontos változók

Lebegőpontos változót jelöl minden olyan maximálisan 6 betüből és számból álló jelsorozat, amelynek első jele nem az I, J, K, L, M, N betük valamelyike.

Pl: C; B6; ALFA.

A lebegőpontos változók értéke tetszőleges az előzőekben definiált lebegőpontos konstans lehet.

2.3 Index és indexes változók

Indexek

Jelöljünk u-val egy tetszőleges fixpontos változót, k-val és K-val előjel nélküli tetszőleges fixpontos konstansokat. E jelöléseket alkalmazva indexek lehetnek az alábbi tipusu kifejezések: u; k; u + k; u - k; k * u; K * u + k; K * u - k. (A * jel a szorzás jele.)

Pl. K+137; MOB-17; 6+j+7; 7 * IMRE

Indexes változók

Bármilyen változó ellátható maximálisan 3 fenti tipusu, a változó után zárójelbe irt indexszel.

Pl: B(K); L(3) LAJOS (3*J, L+3,I20+5); ALFA (5*K+6),

2.4 Függvények jele

Függvény jele minden olyan 4-7 betüből és számból álló jelsorozat lehet, amelyben az utolsó betü F. A jelsorozat után zárójelbe irt kifejezés a függvény argumentuma.

Pl.: SIN F (C + D)
DELTA F (COS F(B))
BESSEL F (3*Y)

2.5 Kifejezések

Kifejezés lehet egy konstansokból, indexes vagy index nélküli változókból, függvényekből álló szósorozat, amelyben müveleti jelek, vesszők, zárójelek szerepelnek ugy,hogy az egy matematikai formulának megfeleljen.

2.6 Müveletek és müveleti jelek

Összeadás:	+	Pl. A + B
Kivonás:	-	A - B
Szorzás:	*	A * B
Osztás:	/	A / B
Hatványozás:	* *	A * * B = A^B

Példa kifejezésre:

A(I)**D-B(I+30)+2.697*A(10)+DELTAF(A+B(K))/3.26*SULY(5*K-

Ha egy kifejezésben a zárójelek nem definiálják egyértelmüen a müveletek sorrendjét, akkor a müveletek sorrendje az alábbi:

Hatványozás

szorzás és osztás

összeadás és kivonás.

Pl. az A + B/C + D** * E * F - G kifejezésnek eszerint az A + B/C +/D^E*F/ - G végrehajtási mód felel meg.

2.7 Aritmetikai formulák

Egy aritmetikai formula

a = b

alaku, ahol a valamilyen indexes vagy index nélküli vált-

zó, b pedig valamilyen kifejezés. A " = " jel jelentése: a aktuális értéke b.

Pl.: B(K+20) = (A(3*J+12) /COS F(P+Q(k)))**R

2.8 Függvényoperátorok

Ha egy program valamelyik aritmetikai formulájában egy függvény jele szerepel, akkor ezt a függvényt valamilyen módon definiálni kell. Pl.: a DELTA F(A+B(K)) kifejezés egy formulában csak azt jelenti, hogy a valamilyen módon definiált konkrét függvényt kell tekinteni az argumentumnak az A + B/K/ kifejezés által meghatározott értékeinél.

Egy függvényt az

a = b

alakkal lehet definiálni, ahol a valamilyen függvény jele, amely után zárójelbe irjuk és vesszővel választjuk el a függvény argumentumainak jeleit (ezek páronként különböző index nélküli változók); b egy olyan kifejezés, amelyben indexes változók nem szerepelnek.

Pl.: DELTA F (X) = A* X+B.

SZIGMA F/X,Y/ = (A*Y + Y* *2) /X-3.5 * Y

Hl F/I;A/ = (5.6* A**I) - A/3.5

Ha egy aritmetikai kifejezésben pl. DELTA F(3*C+D) szerepel, akkor ez azt jelenti, hogy a fent definiált DELTA függvényt az X argumentumnak 3*C+D értékére kell kiszámitani, azaz

DELTA (3*C+D) = A * (3*C + D) + B,

ahol természetesen az A, C, D, B változók aktuális értékét a függvény "behivása" előtt elő kell állitani.

2.9 Vezérlő operátorok

A FORTRAN-ban 15 vezérlő operátor van. Ezek közül néhányat ismertetünk.

Itt is, mint az E-autokódban azokat a formulákat, amelyekre vezérlőoperátor adja át a vezérlést, megszámozzuk. A számozás tetszőleges módon történhet; az ilyen számokat operátor-sorszámnak fogjuk nevezni.

Feltétlen ugrás: GO TO n

Jelentése: átadja a vezérlést az n számmal jelölt operátorra.

Pl.: GO TO 3

Feltétlen ugrás: IF /a/ n_1, n_2, n_3

Jelentése: Itt a egy kifejezés, n_1, n_2, n_3 pedig operátorsorszámok. Ez az operátor az n_1, n_2, n_3 számmal jelölt operátorra adja át a vezérlést attól függően, hogy az a értéke kisebb nullánál, egyenlő nullával, vagy nagyobb nullánál. Pl.: IF (C(K,L)-A) 10, 20, 30,

Diszkrimináló ugrás: GO TO (n_1, n_2, ..., n_m) I

Jelentése: ha az operátor végrehajtásának pillanatában az I változó értéke j-vel egyenlő, akkor a vezérlést az n_j sorszámu operátorra adja át.

Pl.: GO TO /30, 40, 50, 60/ I esetén, ha I értéke 3, akkor a vezérlést az 50 sorszámu operátorra adja át.

Megállás: STOP

Jelentése: az algoritmus végrehajtása befejeződött.

CONTINUE operátor

Egy egy "fiktiv" operátor. Általában a ciklusoperátor magjának végére teszik abból a célból, hogy a ciklusoperátor

"n"-je helyére a CONTINUE operátor sorszámát irhassák. (Lényegében egyszerüen az adott ciklus végét jelzi.)

Ciklus-operátor: DOni = m_1, m_2 vagy DOni = m_1, m_2, m_3 ahol n valamilyen operátor sorszáma, i index nélküli fixpontos változó; m_1, m_2, m_3 pedig vagy előjel nélküli egész, vagy index nélküli fixpontos változó.

Jelentése: az operátor után következő operátorokat (ciklusmag) egészen az n sorszámu operátorig bezárólag hajtsa végre a gép ciklikusan; ugy, hogy a ciklus első lépésben i értéke m_1 legyen, majd minden egyes lépésnél növelje i értékét m_3-mal, és i ezen értékeire hajtsa végre a ciklusmagot egészen addig, amig i értéke m_2 értékénél nem lesz nagyobb; ha i értéke nagyobb m_2-nél, akkor a vezérlést az n sorszámu operátor után következő operátorra adja át.

Pl.: DO 20 I = 1,10

DO 40 L = 1,J,2

(Ha m_3-at nem irunk az operátorban, akkor értéke 1-nek tekintendő.)

2.10 Input-output operátorok

Az input-output operátorok közül kettőt emlitünk meg.

Az input-output operátorok általában két részből állnak a FORTRAN-ban. Az operátor egyik része egy segédinformáció, a bevitel, vagy kiirás módjáról. Ennek az információnak a jele: FORMAT (...). A zárójelbe különféle tipusu kifejezések irhatók. Pl. A FORMAT (I2, E12, .4, F10.4) Az I, E, F jeleknek "szabvány-jelentésük" van.

Az I jel azt jelenti, hogy tizes számrendszerbeli egész számról, az E azt, hogy tizes számrendszerbeli lebe-

gőpontos számról, az F pedig, hogy fixpontos tizes számrendszerbeli (nem egész) számról van szó. A 2,12,10 számok azt jelentik, hogy az adott számok 2; 12; 10 jegynek megfelelő helyet foglalnak el (beleértve a két szám közötti térközök számát is), a 4-es szám pedig a tizedesvessző utáni kiirandó jegyek számát jelenti. Ha az előbbi FORMAT operátor pl. kiirási információ, akkor egy kiiró utasitás pl. a

2 7 - 0.9321 E 02 - 0.0076

számokat irja ki.

Ha az I, E, F betük elé egy n konstanst irunk, akkor n db egymásutáni adott tipusu információt fejez ki.

Pl.: FORMAT /I2, 3EI2.4/ jelentése egy kiiró utasitásnál

27 - 0.9321 E 02 - 0,7580 E-02 0.5536 E 00.

a/ READ-operátor

Alakja: READ n felsorolás

Itt n az ezen operátorhoz kapcsolódó FORMAT operátor sorszáma; a felsorolás pedig a beolvasandó változók felsorolását jelenti.

Pl.:

READ 20, A, B(3), C(I),D(I,2),I = 1,10 ,

jelentése: a 20-as sorszámu FORMAT operátorban megadott információ alapján beviendő a beviteli berendezésen soronkövetkező lyukkártyák tartalma az

A, B(3), C(1), D(1,2) C(2), D(2,2), ..., C(10); D(10,2)

változóknak megfelelő rekeszekbe.

b/ PRINT-operátor

Alakja: PRINT n [felsorolás]

Itt n és a felsorolás is ugyanazt jelenti, mint az előbb.

Pl.:

PRINT 30, A, B(3); B(10); E(I); F(I), I = 1,30

A fenti két "olvasó-iró" operátoron kivül még egy sor más információ beiró, kiiró, átvivő operátor van (mágnesszalag olvasás stb).

Specifikáló operátorok

A program elején természetesen a FORTRAN-nyelv alkalmazása esetén is valamilyen módon közölni kell a változókról szóló információt. Ezt a következő kifejezésekkel irhatjuk le.

1. DIMENSION u, u, u, ...

Itt az u betü valamilyen indexes változó, amelyben az indexek előjel nélküli számok.

Jelentése: megadja, hogy az egyes változókhoz hány rekeszt kell hozzárendelni (milyen tömb tartozik a változókhoz). Pl. DIMENSION A(20)B(4,12) C(3,4,5). Ebben a példában pl. a B változó két indexes; az általa reprezentált tömb 4.12 = 48 rekeszből áll.

2. EQUIVALENCE (a,b,c, ...), (d,e,f, ...)

Itt: az a, b, c, d, e, f, betük változókat jelentenek, amelyek előjel nélküli fixpontos-konstans indexet tartalmazhatnak.

Jelentése: Az operátor alkalmazásával lehetőség nyilik a programozó részére, hogy a rekeszelosztást irányitsa:

ezzel az operátorral egy adott rekeszbe a program folyamán több különböző érték helyezhető el; azaz azokat a munkarekeszeket, amelyeknek tartalma egy uj programrészben már nem lényeges, a program uj változók részére foglal le. Ezzel az operátorral arra is lehetőség van, hogy egy és ugyanazon mennyiséget több, különféle elnevezéssel lássunk el.

Az előzőekben bevezetett operátorokon kivül a FORTRAN-ban szerepel még egy sor más operátor is, különféle feltételes, a pulttal, az egyes regiszterek tartalmával összefüggő vezérlési operátorok, a mágnesszalaggal kapcsolatos operátorok stb. Ezeket itt nem ismertetjük.

A FORTRAN-nyelv szemléltetése céljából irjunk fel két egyszerübb programot.

1. Irjuk fel FORTRAN nyelven az alábbi feladat programját.

Mint ismeretes sorbakapcsolt ellenállásból, kondenzátorból és önindukciós tekercsből álló körön az átfolyó áram értékét amperban az alábbi képlet adja:

$$i = \frac{E}{\sqrt{R^2 + 2\pi fL - \frac{1}{2\pi fC}}},$$

ahol E a feszültség (voltokban) R az ellenállás (ohmokban), L az önindukciós együttható (Henryban) C a kapacitás (faradokban) f a frekvencia (herzekben).

Készitsünk táblázatot, amely az 1; 1,5; 2; 2,5; 3; feszültségértékek és a kapacitás C_0 -tól C_1 -ig terjedő 10^{-8} lépésközökben vett értékei esetén (adott ellenállás, frekvencia és önindukciós tényező mellett) megadja az i értékét.

A program az alábbi lesz:

```
10      READ 1, OHM, FREQ, HENRY
11      READ 1, FRD 1, FRDMX
12      VOLT = 1.0
13      FARAD = FRD 1
14      PUNCH 1, VOLT
16      AMP = VOLT/SQRTF (OHM**2+(6.2832*FREQ*
        HENRY -1. /(6.2832*FREQ*FARAD)) **2)
17      PUNCH 1, FARAD, AMP
18      IF (FARAD-FRDMX) 19, 21, 21
19      FARAD = FARAD + .00000001
20      GO TO 16
21      IF (VOLT-3.0) 22,22,24
22      VOLT = VOLT+0.5
23      GO TO 15
24      STOP
```

Elemezzük a programot.

A 10-es számu operátor beolvassa a lyukkártyáról és az OHM, FREQ, HENRY változókhoz "rendeli" az ellenállás, frekvencia, önindukció megfelelő értékeit.

A 11-es sz. operátor viszont egy másik lyukkártyáról leolvassa a kapacitás alsó és felső határát: C_0, és C_1-et; ezek az FRDL, és az FRDMX változónak lesznek az értékei.

A 12-es operátor "beállitja" a VOLT változó első értékét, azaz az első feszültségértéket. Ezzel az operátorral kezdődik el a külső ciklus. A 14-es operátor kinyomatja a "beállitott" feszültségértékeket.

A 16-os operátor egy aritmetikai formula; ez a megadott képlet szerint kiszámitja i értékét és ez lesz az AMP változó értéke. Ezzel kezdődik a belső ciklus.

A 17-es operátor kinyomtatja a megfelelő kapacitás és áramerősség értékét.

A 18-as operátor vezérlésátadó operátor: attól függően, hogy a kapacitásértékek átfutották-e már a (C_0, C_1) intervallumot vagy nem, a vezérlést a 19-es vagy a 21-es operátornak adja át.

A 19-es operátor növeli 10^{-8}-al a kapacitásértéket, majd a 20-as utasitás (a belső ciklus záróutasitása) átadja a vezérlést a 16-os utasitásnak.

A 21-es operátor a külső ciklus logikai feltételét vizsgálja meg: ha a feszültségérték elérte a megadott intervallum jobbvégpontját (3-at), akkor az algoritmus befejeződik; ellenkező esetben a 22-es operátor növeli a feszültségparaméter értékét, és a 23-as operátor a vezérlést a 15-ös operátornak adja át.

2. Matrix szorzó program

Az alábbi program két huszadrendü négyzetes matrix összeszorzását végzi el.

```
      DIMENSION A (20, 20,) B (20, 20)
      READ 1, A, B, M, L, N
7     DO 4 J=1,  N
1     DO 4 1=1,  M
6     SUM=0.0
2     DO 3 K=1, L
3     SUM=SUM+A (I,K) * B (K,J)
```

```
      5     PUNCH 1, SUM
      4     CONTINUE
      8     STOP
```

A program elemzését az olvasóra bizzuk.

3. AZ ALGOL - 60

Az elektronikus számológépek programozása lényegében algoritmusoknak a "gép nyelvén" történő felirását jelenti.

Ez a felirás, mint már mondottuk, erősen különbözik attól, ahogyan például az adott eljárást emberek számára közöljük. Az algoritmusok "közlésének" szokásos matematikai nyelve nemcsak azért nem alkalmas a gépek esetén, mert betüket és különféle jeleket tartalmaz, hanem azért sem, mert nem elegendően preciz a gép számára. A gépi kód viszont az emberek számára "áttekinthetetlen". A matematikusok szinte az első elektronikus számológépek megjelenésétől kezdve igyekeztek megoldani e kettős nehézséget: olyan formulanyelv megkonstruálását tüzték ki célul, amely elég "preciz" ahhoz is, hogy a gépek számára irjunk fel algoritmusokat (könnyen és főleg egyértelmüen értelmezhető legyen egy konkrét gép utasitásrendszere által), másfelől ennek ellenére elég közel áll a megszokott matematikai formulanyelvhez.

A kisérletek a különféle "autokód" tipusu, az előbbi fejezetben ismertetett nyelveken keresztül vezettek el az ALGOL (Algorithmic Language) formulanyelv megteremtéséig.

Az ALGOL-formulanyelvet 1958-ban egy Zürichben tartott konferencián kezdeményezték, ill. dolgozták ki; és 1959-ben közölték az első valamelyest kiforrott változatot. Ezután több konferencián e célból létrehozott bizottságok vizsgálták és egészitették ki ezt a változatot, a Communications of the ACM folyóirat által meginditott vitához beérkező javaslatokat figyelembe véve. A most ismertetendő ALGOL-60 formulanyelvet 1960 januárjában Párizsban fogadták el azzal,

hogy egy ideig nem is változtatnak rajta, abból kiindulva, hogy az állandó változtatás lehetetlenné tenné a nyelv nemzetközi alkalmazását, pl. algoritmusok cseréje céljából.

Az ALGOL-nyelvnek három változatát, "szintjét" különböztetjük meg:

a/ Hivatkozási nyelv (Etalon-nyelv)

b/ Publikációs nyelv

c/ Gépi reprezentánsok

A formulanyelv ezen szintjei nem a strukturában, hanem csupán a használt jelek tipusában különböznek egymástól.

A hivatkozási nyelv: a bizottság "munkanyelve", a nyelv etalonja, a definiáló nyelv. Ez a nyelv az alap és vezérfonal a fordítóprogramok készitésénél, valamint minden gépi reprezentáns számára. Az ALGOL - alapján készült főbb közlemények ezen a nyelven jelennek meg.

A publikációs nyelv a kézirásnak és nyomdatechnikának megfelelően a hivatkozási nyelvhez képest változtatásokat enged meg (pl. indexeket, közöket, exponenseket, görög betüket).

Ez a nyelv számitási eljárások leirására és közlésére használható.

A publikációs nyelvben használatos jelzések országonként különbözhetnek egymástól, de biztositani kell, hogy e jelölések kölcsönösen egyértelmü megfelelkezésben legyenek a hivatkozási nyelvvel.

A gépi reprezentánsok a hivatkozási nyelvnek olyan változatai, amelyekben a hivatkozási nyelv egyes jeleit a gép bemeneti berendezésén szereplő jelekkel cserélünk fel. Az adott gépre kidolgozott fordítóprogram ezen a nyelven felirt programok forditására képes.

A következőkben a hivatkozási nyelvet ismertetjük.

Az ALGOL-nyelv szerkezete

Az ALGOL-nyelv - mint minden más nyelv - megadott jelek, szimbólumok kombinációjából alkotott "szavakból" áll. A megengedett jelhalmaz: a betük, a számjegyek, különféle müveleti jelek, zárójelek és néhány egyéb jel pl., :, ↑, =, →, stb. A betüjelekből alkotott néhány szó a felsoroltakkal együtt az alapjeleket alkotja. Azokat az alapjeleket, amelyek egyuttal angol szavak is, nyomtatásban vastagon szedjük, kéziráskor, gépiráskor pedig aláhuzzuk. Jegyzetjüket [illegible] aláhuzást alkalmazunk. (Pl. real)

Az alapjelek segitségével kifejezések irhatók fel; ezek közül pl. a számitási algoritmusok leirására szolgáló aritmetikai kifejezések a megszokott, a matematikában alkalmazott aritmetikai kifejezésekhez hasonlóan, számokat, változókat, függvényeket tartalmaznak.

Az aritmetikai kifejezésekből bizonyos szabályok alkalmazásával formulákat, un. elemi vagy értékadó-utasitásokat hozhatunk létre.

Az értékadó-utasitások bizonyos "adminisztrativ", nem aritmetikai utasitásokkal összekapcsolva egy nagyobb egységbe, un. összetett-utasitásokká vonhatók össze. Egy összetett utasitás értékadó-utasitásokon kivül elágazási (döntési), ismétlő (ciklikus) utasitásokat is tartalmaz. Az értékadó utasitásokat "utasitászárójelek" segitségével fogjuk össze.

Az utasitásokban szereplő jelkombinációkat valamilyen módon specifikálni kell, azaz információkat kell adni arra nézve, hogy mit jelentenek. Közölni kell, hogy egy előforduló változó milyen tipusu értékeket vehet fel (egész, "lebegőpontos" stb); meg kell adni az összetett utasitásban szereplő "tömbök" (vektor, matrix) dimenzióját, stb. Minden

egyes összetett utasitást ilyen közleménnyel, un. deklarációval látunk el; a deklarációval rendelkező összetett utasitást, mint - ebben az értelemben - zárt egységet, blokknak nevezzük.

Az olyan összetett utasitást, amely önálló, azaz nem része más utasitásnak és nem használ fel olyan utasitásokat, amelyek nem szerepelnek benne, programnak hivjuk.

Az ALGOL-nyelv leirásának módja

Bármely nyelvet kétféleképpen lehet leirni. A szintaktikus leirás lényegében a nyelvben előforduló összes szó felsorolása. A szemantikus leirás viszont tartalmi; a szavak egymás közötti kapcsolatából származó belső összefüggéseket, a lényeget, az értelmet fogja meg; pontosabban szólva az egyes szavak vagy kifejezések jelentését fejezi ki. Az ALGOL-nyelv szintaktikus leirása "axiomatikusan" preciz; forditóprogramok készitésénél feltétlenül ebből kell kiindulni. Megérteni, ill. szemléletesen leirni a nyelvet viszont, bár esetleg kevésbé pontosan szemantikusan könnyebben lehet.

Mi a nyelv ismertetését ugy oldjuk meg, hogy általában az egyes szavak szemantikai jelentésére utalunk, ugyanakkor meghagyjuk az eredeti, szintaktikus leirás csoportositását. Függelékként közöljük a nyelv rövid szintaktikus leirását is.

Az ALGOL-nyelv alapjelei

1, Betük

Betükön a latin abc nagy és kisbetüit értjük.

2. Számjegyek

Az ALGOL-nyelvben a tizes számrendszer tiz ismert (arab) számjegyét használjuk (0, 1, 2, ..., 9).

3. Logikai értékek jelei

Mint ismeretes, a két értékü (szokásos) logikában bármely logikai változóhoz két értéket rendelünk hozzá: egyik érték az igaz érték, a másik a hamis. Az igaz ill. hamis jelet szokás 1-gyel és 0-val is jelölni. Az ALGOL-nyelvben ezen két értéket a

true

false

alapjelekkel jelöljük.

4. Elhatároló jelek

Elhatároló jeleknek az olyan jeleket nevezzük,amelyek ALGOL-kifejezések képzésében az előbb emlitett jeleken kivül felhasználhatók (pl. müveleti jelek, zárójelek stb).

4.1 Müveleti jelek

Az ALGOL-nyelvben is az eljárások, az algoritmusok, lényegében megadott sorrendben elvégzendő müveletekből állnak. Müveleten valamilyen elemi eljárást értünk (pl. összeadás, összehasonlitás, vezérlésátadás stb.).

4.1.1 Aritmetikai müveletek jelei

Ebbe a csoportba tartoznak a megszokott alapmüveleti jelek, valamint a hatványozás és az aritmetikai osztás jele:

Összeadás: +

Kivonás: -

Szorzás: x

Osztás: /

Hatványozás: ↑

Aritmetikai osztás: ÷ (egész osztás, lásd később)

4.1.2 Logikai müveletek jelei

E csoport jelei is a megszokott jelek:

ekvivalencia jele: ≡

implikáció jele: ⊃

konjunkció " : ∧

diszjunkció " : ∨

tagadás
negáció " : ¬

4.1.3 Reláció-jelek

E csoportba a megszokott relációjelek (összehasonlitójelek) tartoznak:

egyenlő: = ; nem egyenlő: ≠

kisebb: < ; nagyobb: >

kisebb vagy
egyenlő: ≤ ; nagyobb vagy egyenlő: ≥

4.1.4 Zárójelek

Az ALGOL-ban a következő tipusu "zárójeleket" vezették be:

kezdőzárójel: (; végzárójel:) ;

szögletes " : [; szögletes végzárójel:] ;

kezdő vessző: ‘ ; záró vessző: ’ ;

speciális kezdőjel: begin; speciális zárójel: end.

4.1.5 Elválasztójelek

vessző: , ; kettőspont egyenlőség: := ;

pont . ; térköz jel: ␣ ;

kettőspont: : ;

pontosvessző: ; ; kitevő- exponens jele: 10

"lépés" jel: step

"ig"-jel: until

"megjegyzés" jel: comment

"amig-csak"-jel: while

Utóbbi jelek tartalmi jelentéséből is következtethetünk arra, hogy az ALGOL-nyelven felirt algoritmusokban mi a szerepük:

A step jelet ciklikus eljárások felirásánál használjuk: step 1 pl. azt jelenti, hogy egy egységgel kell megnövelni a ciklus adott paraméterét.

Az until jel ugyancsak a ciklikus eljárásoknál nyer alkalmazást: until 89 pl. azt jelenti, hogy az adott ciklusparaméter 89-ig megy el.

A comment-jel egy "adminisztrativ" jel: azt jelenti, hogy az utána következő jelek (szavak) sorozata nem más, mint "kommentáló" szöveg, ami egy adott "ALGOL-programnak" gép által történő forditása esetén a gép számára nem "mond" semmit, csakis a program olvasói, alkalmazói részére. (Pl. az eljárás rövid, szavakkal történő leirása.)

A while -jel ciklusos algoritmusok felirásánál szerepel.

4.1.6 Deklarátor-jelek

Ezen jelcsoport elemei az utasitások deklarációjában, bizonyos ALGOL "objektumok" tulajdonságainak leirásában vesznek részt. Mint emlitettük az utasitásokban szereplő

jelek tulajdonságait (változók jelei, függvények jelei stb.) valamilyen módon előre specifikálni, "deklarálni" kell.

A deklarátor jelek az alábbiak:

own
===

Boolean
=======

integer
=======

real
====

array
=====

switch
======

procedure
=========

Ezen jelek jelentését a deklarációk leirásánál ismertetjük majd.

4.1.7 Vezérlő jelek

A vezérlő jeleket az algoritmus felirásában "vezérlésátadó" operátorok felirásánál használjuk. Ezek az alábbiak:

go
==

if
==

then
====

else
====

for
===

do
==

A fenti alapjelek tartalmi jelentésére a jelekhez hozzárendelt angol szó (ill. magyar forditása) is utal; pontos szerepüket a megfelelő helyen leirjuk majd.

4.1.8 Specifikáló jelek

A specifikáló alapjelek az alábbiak:

string

label

value

Az ALGOL-nyelv jelkészletét, az előzőekben felsorolt alapjelek halmaza képezi. Egy algoritmus ALGOL-nyelven a fenti alapjelekből alkotott szavakból (az alapjelek meghatározott elrendezése) áll. Az ALGOL-nyelv bővebb, mint a hétköznapi nyelv, amelyben csak betük kombinációjából előálló szavak és számjegyekből álló számok szerepelnek; bizonyos értelemben azonban szükebb, mint a matematika nyelve (nem szerepel pl $\int$ -jel, $\pm$ ∞ -jel, Σ jel stb.). Szerepelnek viszont az ALGOL-ban olyan alapjelek, amelyek a matematikai nyelvben nem fordulnak elő (pl. go, to, until, stb.) mint alapjelek.

A betükből álló és az angol nyelvben szemantikus jelentéssel is biró alapjelek (angol szavak) magyar forditása a következő:

array	tömb (mátrix, vektor, többdimenziós táblázat)
begin	kezdet
Boolean	Boole-féle (logikai)
comment	magyarázat
do	végezd el
else	máskülönben
end	vég

false	hamis
for	-ra, -re
go to ...	menj ...-hez
if	ha
integer	egész
label	cimke
own	saját
procedure	eljárás
real	valós
step	lépj, lépés
string	lánc-füzér
switch	kapcsoló
then	akkor
true	igaz
until	-ig
value	érték
while	amig csak

Az ALGOL-nyelv alapfogalmai

Mint emlitettük, bármely ALGOL nyelven felirt algoritmus a felsorolt jelek valamilyen kombinációjából áll. Az algoritmus maga jelekből álló szavakra bomlik; ezen szavak bizonyos önálló egységet képeznek. A következőkben az "ALGOL szavak" tipusait soroljuk fel; pontosabban szólva azt ismertetjük, hogy egy ALGOL-nyelven felirt algoritmus-

ban milyen önálló, kisebb és nagyobb egységek szerepelhetnek.

a/ Számok

A számok az ALGOL-ban is helyértékek alapján egymásmellé rendelt számjegyekből állnak. A felirásnál a tizes számrendszert használjuk. Az egész számokat integer-tipusuaknak (fixpontos), a tizedes számokat real-tipusuaknak nevezzük ("lebegőpontos").

Irásmódjuk a szokásos: lehetnek előjelesek, vagy (pozitiv szám esetén) előjel nélküliek. A "tizedes-vesszőt" ponttal jelöljük.

Ha a számokat normál-alakban (mantissza kitevő) irjuk fel, akkor a kitevő-alapjelet (10) alkalmazzuk. Pl.: a $5.76.10^6$ számot $5.76_{10}6$ alakban irjuk.

Példák: 0 ; 532; . 6826; + 0.3700 ;
-652.048 ; $08.34_{10}5$; -10^6 ; 10^{-5} ;
$3.48_{10}+12$; $-3_{10}-4$; -10^{-5} ;

b/ Azonositók

Az ALGOL-ban betük és számjegyek valamilyen kombinációját, amellyel az algoritmus részeit, ágait; az algoritmusban szereplő mennyiségeket vagy ezek csoportját (a tömböket) megnevezzük: azonositóknak hivjuk. Az azonositó tehát egy olyan ALGOL-fogalom, amely valamilyen ALGOL-beli objektum neve.

Pl. Hőmérséklet
Nyomás
H2O
ELÁGAZÁS 2.

c/ Változók

A változó nem más, mint egy érték jele, jelölése. (Értékben a szokásos fogalmat értjük.) A változó által reprezentált értéket bárhol felhasználhatjuk és un. értékadó utasitások segitségével a változó értékét megváltoztathatjuk. (Gépi nyelven szólva: a változó egy gépi rekesz címének jelek egy bizonyos sorozatával kifejtett szimbóluma.

Minden változó egy azonositóval jelölhető (betük és számjegyek kombinációja), Pl.

beta;

kapacitás,

A 32, Omega, Epszilon.

A változók tipusai

A változó lehet egész, valós, vagy logikai érték jele. Ennek megfelelően a változók lehetnek:

real
====

integer
=======

Boolean
=======

tipusuak. E megjelöléssel a tekintett változó szóbajövő értékkészletét határoljuk be.

Az algoritmusok deklarációs blokkjában (lásd később), amikor a változókat felsoroljuk, a fenti jelekkel irjuk le a tipusokat.

Egy változó értékén mindig ugynevezett aktuális értéket értünk: ez nem más, mint az az érték, amit abban a pillanatban, amikor erre hivatkozunk, a változó felvesz - ill. a gépi reprezentálásra gondolva, annak a rekesznek a pilla-

nyi tartalma, amelynek a címét a változó azonosítója jelöli ki.

Egy real tipusu változó értéke mindig valamilyen módon (pl. a gép számjegykapacitása által) korlátozott nagyságu, és számjegyü valós szám lehet.

Egy integer tipusu változó értéke pedig egy ugyancsak valamilyen módon korlátozott nagyságu és számjegyü egész szám.

Egy Boolean tipusu változónak két értéke lehet: a true alapjellel jelölt igaz érték (1) és a false alapjellel jelölt hamis érték (0).

Egy változó jelölhet olyan értéket is, amely egy,vagy többdimenziós sokaságnak, un, tömbnek az eleme. Pl. vektorkomponens; mátrix elem stb.) Ebben az esetben a változókat indexszel látjuk el.

Indexek

Az index fogalma az ALGOL-ban megegyezik a matematikában megszokott indexfogalommal; az indexes változót azonban nem a megszokott módon jelöljük, hanem az egy adott változóhoz tartozó indexeket a változó után szögletes "indexzárójelbe" irjuk. Az index nélküli változót skalár változónak, az indexes változót pedig tömbnek hivjuk az ALGOL-ban.

Az indexek lényegében integer-tipusu változók, azaz olyan változók, amelyek egész értékeket reprezentálnak. Az index maga is lehet indexes változó.

Példák:

Itt pl. a [i, j, k] jelentése .... a_{ijk}. (háromdimenziós tömb)

d/ A kifejezés fogalma

A kifejezés fogalmát a megszokott értelemben értjük és beszélni fogunk különféle fajtájáról: aritmetikai kifejezésről, logikai kifejezésről, helymegjelölő kifejezésről, elsődleges kifejezésről, feltételeket tartalmazó (feltételes) kifejezésről, függvénykifejezésről stb.

d.1. Függvénykifejezések

A függvény fogalma az ALGOL-ban megegyezik a megszokott függvényfogalommal: egy függvénykifejezés olyan eljárás jele, amely bizonyos adott változók értékeihez egy (vagy több) értéket rendel hozzá. A függvénykifejezést ("a függvény jelét") un. eljárás deklarációval (lásd később) "definiálni" kell: azaz a programban valahol meg kell adni, hogy egy függvénykifejezés milyen "szabályok", összefüggések alapján rendeli hozzá a "bemenő" változó (vagy változók) értékéhez a függvény értékét. A függvénykifejezés csak hivatkozik egy olyan értékre, amely a függvény-eljárás végrehajtása révén létrejön. A függvény kifejezés ennek az értéknek a jele.

A függvénykifejezések függvény jelből és aktuális paraméterérték-részből állnak.

Függvény jel az alapjelek (főleg betük) valamilyen kombinációja lehet.

A függvények az ALGOL-ban is egy vagy többváltozósak lehetnek: ezt itt ugy fejezzük ki, hogy a paraméterek száma a függvényben egy vagy több. A paraméterek a függvény "bemenetei". Egy függvénykifejezés paraméterei, "bemenetei" helyére nemcsak egy-egy betü (pl.: sinx; Epszilon /u,v,/) ir-

ható, hanem egy tetszőleges kifejezés is. Ezt úgy kell érteni, hogy ha egy ALGOL-nyelven irt algoritmusban valamely - az algoritmusban egyébként definiált - F /x/ függvénynek az x = G/u, v, ..., z/ argumentumnál vett értékére van szükség, akkor arra a programban

$F(G(u,v,z))$ alakkal hivatkozhatunk.

Ha például az F(x) függvény definiciója

$$F(x) = 3x^2 + \sqrt{x}$$

és ezt az x = (5a + 2b) c értéknél tekintjük, akkor az algoritmusnak azon a pontján, amelyen a fenti értékre szükségünk van, erre egyszerüen az

$$F((5a + 2b)c)$$

alakkal hivatkozunk, aminek a jelentése a fentiek szerint

$$3((5a + 2b)c)^2 + \sqrt{(5a + 2b)c}$$

Kiemeljük, hogy itt most a függvénykifejezések irásmódjáról van szó; arról, hogy hogyan hivatkozhatunk egy függvény értékére. Azt, hogy egy függvényt (amelyet éppen egy függvénykifejezés "aktivizál") hogyan definiáljunk, később irjuk le (lásd eljárásdeklarációk).

Példák:

1. J(v+s,n). Ennek a függvénykifejezésnek a következő a jelentése: a - valamilyen eljárás - deklarációval adott - J(x,y) függvény értékét kell venni az x = v+s,y = n helyen, ahol v, s, n valamilyen változók.

2. S(s-5) Hőmérséklet: (T) Nyomás: (P)

3. Sin (a + b/c)

Szabványos (standard) függvények

A szabványos függvények jelölésére megállapodásszerüen az ALGOL-ban a következő jelöléseket tartjuk meg:

abs (K)	Jelentése:	a K kifejezés abszolutértéke
sign (K)	"	a K kifejezés előjele $\text{sign}(K) = \begin{cases} 1 \text{ ha } K > 0 \\ -1 \text{ ha } K < 0 \\ 0 \text{ ha } K = 0 \end{cases}$
sqrt (K)	"	a K kifejezés négyzetgyöke
sin (K)	"	a K kifejezés szinusza
cos (K)	"	a K kifejezés koszinusza
arctan (K)	"	a K kifejezés arkusztangensének főértéke
ln (K)	"	a K kifejezés természetes logaritmusa
exp (K)	"	a K kifejezés exponenciális függvénye (e^K)
entier (K)	"	a K kifejezés egész értéke (A K kifejezés értékéhez azt a legnagyobb egészszámot rendeli hozzá, amely nem nagyobb mint K értéke.)

Állapodjunk meg abban, hogy ezeket a függvényeket nem kell külön deklarálni, és a K változó értéke lehet integer tipusu is, real tipusu is, a függvény értéke azonban csak real tipusu (kivéve a sign(K) és az entier( ) függvényeket.

d.2. Feltételes kifejezés fogalma

Konkrét algoritmusok felirásánál, pl. gépi programok szerkesztésénél igen gyakran fordulnak elő az algoritmus egyes helyein elágazási pontok. Ezekben a pontokban az algoritmus "irányát" megadott logikai feltételek határozzák meg. Ezen feltételek strukturája általában a következő:

ha F akkor A; különben B

ahol F valamilyen (tetszőleges alakban megadott) feltétel; A és B pedig az algoritmus két iránya.

Az ALGOL-ban feltételes kifejezésen (azaz feltételeket is tartalmazó kifejezésen) az if, then, else alapjelek segitségével létrehozott alábbi strukturáju kifejezést értjük:

if F_1 then A_1 else if F_2 then A_2 else .... else A_n ahol F_1, F_2, ..., F_{n-1} valamilyen logikai feltételt fejez ki (pl. logikai kifejezés; lásd később). A_1, A_2, ... A_n pedig valamilyen kifejezés. Nevezzük el a feltételes kifejezés

if F then

részét feltételnek.

A feltételes kifejezés legegyszerübb alakja kettős elágazásu:

if F then A else B

azaz: ha F teljesül, akkor: A

ha F nem teljesül, akkor: B

A feltételes kifejezések több célra felhasználhatók:

szerepelhetnek aritmetikai kifejezésekben, logikai kifejezésekben, feltételes utasitásokban (l. később).

d.3. Aritmetikai kifejezések

Aritmetikai kifejezésnek hivjuk az ALGOL-program olyan egységét, amely megadja valamilyen numerikus érték kiszámitásának szabályát. Az aritmetikai kifejezés fogalmának tartalmi jelentése lényegében ugyanaz, mint a matematikában megszokott aritmetikai kifejezésé, különbség csak a "formában", az irásmódban van.

Aritmetikai kifejezés szerkesztésének szabályai

Az aritmetikai kifejezések "alapelemei" a müveletek által összekapcsolt változók. Egy aritmetikai kifejezés értékét ugy keressük meg, hogy a változók aktuális értékein elvégezzük a kijelölt müveleteket. (Aktuális értéknek nevezzük,mint mondottuk, a változónak azt az értékét, amit az algoritmus végrehajtásának folyamán az adott pillanatban felvesz.)

A +, -, × jelek jelentése ugyanaz, mint a közönséges aritmetikában. Ezen müveletek eredménye integer- tipusu (egész) lesz, ha mindkét operandus integer-tipusu; minden más esetben real-tipusu (valós).

Az / osztási jel operandusai az integer és real tipusu változók mind a négy lehetséges kombinációja lehet; az eredmény mindig real-tipusu.

Az : jel az aritmetikai osztás jele ("egész-osztás", maradéknélküli osztás). Ez csak integer-tipusu változókra van értelmezve, és az eredmény is integer-tipusu, mégpedig az alábbi módon definiálva:

$$a \div b = \text{sign}\,(a/b) \times \text{entier}\,(\text{abs}\,(a/b))$$

Az ↑ jel a hatványozás jele. A hatványozás folyamán az alap és a kitevő tipusától függően az eredmény tipusát az alábbiak szerint kapjuk:

Jelöljünk i-vel egy egész-tipusu, r-rel egy valós tipusu és a a-val egy akár egész-, akár valós-tipusu számot.

1/ a ↑ i. Ha $i > 0$: $a \times a \times a \times \ldots \times a$ (i-szer); az eredmény a tipusával egyező tipusu.
Ha $i = 0$ és ha $a \neq 0$: 1; a-val egyező tipusu.
Ha $i = 0$ és ha $a = 0$: nincs értelmezve
Ha i 0 és ha $a \neq 0$: $1/a \times a \times \ldots \times a$/, (A nevezőben i számu tényező van); az eredmény real-tipusu.
Ha i 0 és ha $a = 0$: nincs értelmezve.

2/ a ↑ r. Ha $a > 0$: exp (r × ln (a)) ; az eredmény real-tipusu.
Ha $a = 0$ és ha $r > 0$: 0.0; az eredmény real-tipusu.
Ha $a = 0$ és ha $r \leq 0$: nincs értelmezve.
Ha $a < 0$: semmilyen kitevővel nincs értelmezve.

A müveletek végrehajtásának sorrendje

A müveletek végrehajtási sorrendjét az ALGOL-aritmetikai kifejezésekben is zárójelekkel jelöljük ki. A müveletek végrehajtása általában balról jobbra és zárójelek hiányában az alábbi elsőbbségi szabályok betartásával történik:

első a hatványozás (↑)

második a multiplikativ müvelet ×, /, ÷/

harmadik az additiv müvelet /+, -/

Az aritmetikai kifejezéseknél használni fogjuk még az alábbi - voltaképpen a közönséges algebrából vett megnevezéseket.

Elsődleges (vagy egytagu) kifejezés. Elsődleges kifejezésnek nevezünk egy előjel nélküli számot, vagy egy változót, vagy egy függvényt.

Pl. $7.395_{10}-8$

Összeg

w [i + 2]

cos (y + z × 3).

Tényező

Tényezőn is lényegében az aritmetikai értelemben vett tényezőt értjük. (Egy szorzatkifejezés egyik szorzandója.)

Példák:

Omega a ↑ cos (y + z × 3)

$7.395_{10}-8$ ↑ w[1+2] ↑ (a-3/y + vu ↑ 8).

Tag

Tagnak nevezzük valamely összegkifejezés egyik összeadandóját.

Példa:

w [i+2] ↑ (a-3/y + vu - 8) .

Feltétlen aritmetikai kifejezés

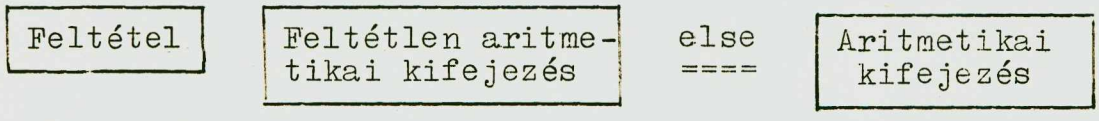
Feltétlen (feltétel nélküli) aritmetikai kifejezésen olyan aritmetikai kifejezést értünk, amely nem tartalmaz semmilyen feltételt.

Példa:

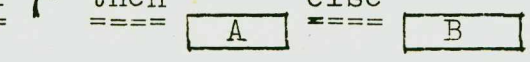
U-Yu + omega , összeg ↑ cos(y+[illegible]3) 7.394_{10}-8

↑w[i+2,8]↑(a-3/y+vu-8)

Egy <u>aritmetikai kifejezés</u> szerkezete a következő lehet:



vagyis:



; azaz

ha F akkor A különben B , ahol A és B aritmetikai kifejezések.

Példa:

1/ <u>if</u> q > 0 <u>then</u> S + 3×Q/A <u>else</u> 2×S+3×q.

E példa jelentése az alábbi:

Ha q > 0, akkor a kifejezés: $3Q/_A$ + S alaku, ellenkező esetben 2 S + 3 q alaku; azaz ha ez a kifejezés valamilyen függvényt definiál, akkor ez a következő kétágu függvény lesz

$$F = \begin{cases} 3Q/_A + S & \text{ha} \quad q > 0 \\ 2S+3q & \text{ha} \quad q \leq 0. \end{cases}$$

Az <u>else</u> alapjel után ujra következhet aritmetikai kifejezés, azaz ez a definició "láncszerü": a definiált fogalom a definicióban is szerepel. (Természetesen következhet az <u>else</u> után feltétlen aritmetikai kifejezés is, mint a fenti példában láttuk.)

További példák aritmetikai kifejezésekre:

$w \times u + Q\,(S + Cu) \uparrow 2$

if a < 0 then U + V else if a × b > 17 then U/V else

if k ≠ y then V/U else 0

a × sin (omega × t)

$0.57_{10}12 \times a\,[N \times (N-1)\ /2,0]$

$(A \times \arctan(y) + z) \uparrow (7 + Q)$

if q then n-1 else n

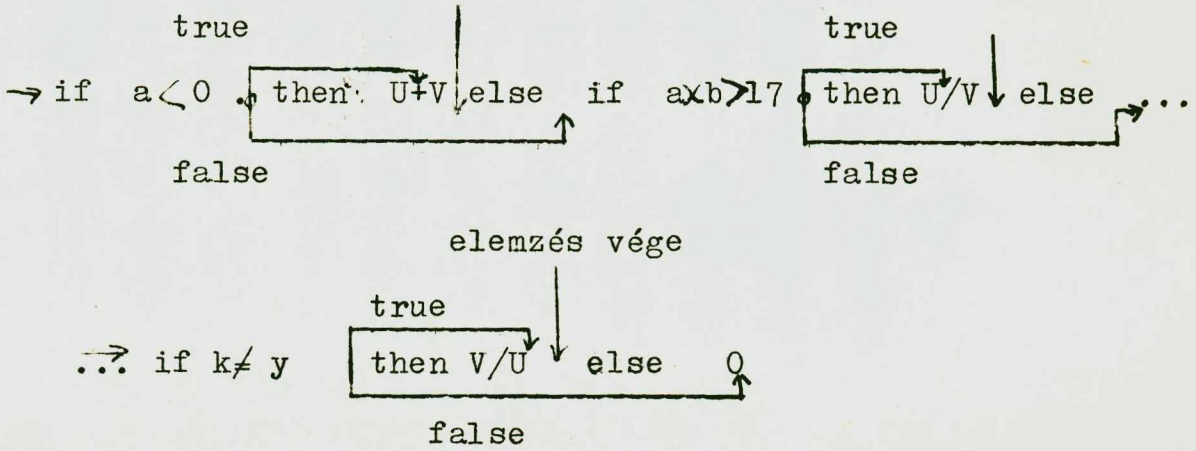
if a < 0 then A/B else if b = 0 then B/A else z

Elemezzük ki ezek közül - szemléltetés kedvéért - a másodikat, azaz az

if a < 0 then U + V else if a × b 17 then U/V else if k ≠
≠ y then V/U else 0 kifejezést.

Nevezzük el ezt a kifejezést K-nak.

A K kifejezés a következő ábrával szemléltethető:



Tegyük fel, hogy minden, a kifejezésben szereplő változó aktuális értéke a K aritmetikai kifejezés "realizálásánál" adott. (Előzőleg előállitottuk.)

A kifejezés "végrehajtása" az első logikai feltétellel kezdődik: az a<0 reláció vizsgálatával. Ha ennek a logikai kifejezésnek az értéke igaz-érték, azaz true-érték, akkor az U+V feltétlen aritmetikai kifejezés "kerül végrehajtásra", azaz a K kifejezés értéke U+V értékével lesz egyenlő.

Ha az a<0 reláció nem teljesül, azaz ennek a logikai kifejezésnek az értéke hamis, tehát - alapjellel kifejezve - false -érték, akkor a K aritmetikai kifejezés "értékelése" tovább folytatódik: az else alapjel után álló, jelenleg szintén aritmetikai kifejezés vizsgálata következik.

Ennek a kifejezésnek a logikai feltétele az a×.a<0 logikai kifejezés. Ha ennek értéke true, akkor a K kifejezés értéke előállt: U/V-vel egyenlő. Ha a logikai kifejezés értéke false-érték, akkor az else utáni rész elemzése következik.

Ez a rész ujra logikai feltétellel kezdődik: ha a k ≠ y logikai kifejezés értéke true-érték (azaz teljesül), akkor a K kifejezés értéke V/U lesz. A logikai kifejezés false-értéke esetén (azaz ha nem teljesül) a következő else alapjel utáni rész elemzése jön sorra. Ez itt egy feltétlen kifejezés (elsődleges kifejezés, vagy "egytag"); ez lesz a K kifejezés értéke, azaz 0.

A fenti aritmetikai kifejezés a megszokott függvénytani jelölésekkel az alábbi alakban irható fel:

$$K = \begin{cases} U+V & \text{ha } a \neq 0 \\ U/V & \text{ha } a \times b \neq 17 \\ V/U & \text{ha } k \neq y \\ 0 & \text{ha sem } a \neq 0 \\ & \quad \text{sem } a \times b \neq 17 \\ & \quad \text{sem } k \neq y \text{ nem teljesül.} \end{cases}$$

Néhány kiegészitő megjegyzés az aritmetikai kifejezésekkel kapcsolatban:

1/ A tag/tényező (ill. tag + tényező) szerkezetü kifejezések osztást jelentenek: másszóval a tagnak a tényező reciprokával való szorzatát, az elsőbbségi szabályok figyelembevételével.

Pl:

a / b × 7/ (p-q) × v/s

jelentése

$$(((a \times (b^{-1})) \times 7) \times ((p-q)^{-1})) \times v) \times (s^{-1})$$

2/ A valós tipusu (real) értékeket, számokat, és változókat (illetve ezek értékeit) korlátozott pontosságunaknak tekintjük (ugy, ahogy azt a numerikus matematikában tesszük általában a "papir", illetve a gépek számjegy-kapacitása miatt.) A különböző gépek, különböző pontossággal számitják ki egy és ugyanazon kifejezés értékét. A pontatlanság mértékét és következményeit a numerikus matematika módszereivel kell becsülni, számontartani.

d.4. Logikai kifejezések

A logikai kifejezések aritmetikai kifejezésekből, logikai müveleti jelekből, logikai változókból, az if, then,

else alapjelekből ill. ezek egy csoportjából épülnek fel. A logikai kifejezések egy logikai érték meghatározására szolgáló szabályt jelentenek.

A logikai változók kétértékűek: értékük vagy true, (igaz, azaz 1) vagy false (hamis, azaz 0). Ugyanigy kétértékűek a logikai függvény-kifejezések is.

Ahhoz hasonlóan, ahogy a számértékekkel biró változók tipusát alapjellel jelöltük meg abból a célból, hogy a deklarációban történő leirásuk könnyebb legyen (real; integer), a logikai változók, ill. függvény kifejezések tipusát is jelöljük: Boolean alapjellel.

A logikai müveletek (mint már azt az alapjeleknél felsoroltuk) a következők:

a/ reláció müveletek: $<$, $\leq$, $=$, $\geq$, $>$, $\neq$,

b/ predikatum müveletek: $\neg$ (nem)
$\wedge$ (és)
$\supset$ (implikáció)
$\equiv$ (ekvivalencia)

Ezen jelek jelentését a következő táblázat tartalmazza:

L 1	false	false	true	true
L 2	false	true	false	true
$\neg$ L 1	true	true	false	false
L 1 $\wedge$ L 2	false	false	false	true
L 1 $\vee$ L 2	false	true	true	true
L 1 $\supset$ L 2	true	true	false	true
L 1 $\equiv$ L 2	true	false	false	true

(A táblázatot fentről lefelé kell olvasni: az első két sorban helyeztük el az adott müveletekhez rendelhető két (L 1, L 2) változó lehetséges értékkombinációit, a további sorokban pedig az értékkombinációkhoz tartozó és a sorok baloldalan jelölt müveletek által létrejövő értékeket. Ha például L 1 értéke true, L 2 értéke false (3.oszlop), akkor L 1⊃L 2 értéke false (3 oszlop, 4. sor).

A müveletek elsőbbségi szabályai

Egy logikai kifejezésben szereplő müveletek végrehajtása általában balról jobbra haladva történik, figyelembe kell azonban venni az alábbi kiegészitő szabályokat:

elsődlegesek: az aritmetikai kifejezések

másodlagosak: $<$, $\leq$, $=$, $\geq$, $>$, $\neq$

harmadlagosak: $\neg$

negyedlegesek: $\wedge$

ötödlegesek: $\vee$

hatodlagosak: $\supset$

hetedlegesek: $\equiv$

A logikai kifejezések lehetnek feltétel nélküliek (feltétlen logikai kifejezés), vagy feltételt tartalmazók; ezeket egyszerüen logikai kifejezéseknek hivjuk.

A logikai kifejezés strukturája az alábbi:

Feltétel		Feltétlen logikai kifejezés	else	logikai kifejezés

vagyis

if	⊃	then	FL	else	L.

ahol ⊛ valamilyen logikai feltétel (reláció) FL feltétlen logikai kifejezés, L logikai kifejezés.

Példák:

$$Y > V \vee z < Q$$

$$a + b > -5 \wedge z - d > q \uparrow 2$$

$$p \wedge q \vee x \neq y$$

$$g \equiv \neg\, a \wedge b \wedge \neg\, c \vee d \vee e \supset \neg\, f$$

$$\underline{\underline{\text{if}}}\ k < l \quad \underline{\underline{\text{then}}}\ s > w \quad \underline{\underline{\text{else}}}\ h \leqq c$$

$$\underline{\underline{\text{if}}}\ \underline{\underline{\text{if}}}\ \underline{\underline{\text{if}}}\ a\ \underline{\underline{\text{then}}}\ b\ \underline{\underline{\text{else}}}\ c\ \underline{\underline{\text{then}}}\ d\ \underline{\underline{\text{else}}}\ f\ \underline{\underline{\text{then}}}\ g\ \underline{\underline{\text{else}}}\ h < k$$

d.5. Helymegjelölő kifejezések

A gépi kódokkal, vagy autokódokkal (E-autokód, FORTRAN) felirt algoritmusoknál is szükség van arra, hogy az algoritmus részeit valamilyen módon megjelöljük abból a célból, hogy hivatkozás ("behivás") esetén egyszerüen ezt a hivatkozási jelet használhassuk.

Az ALGOL-ban is szerepelnek olyan utasitások (pl. vezérlés-átadás az algoritmus valamely utasitására), amelynek "argumentuma" éppen valamilyen hivatkozási jel, amelyet az algoritmus valamely részéhez rendelünk hozzá: ezeket szokás "helymegjelölők"-nek is hivni.

Az utasitások "hivatkozási jelét", helymegjelölőjét, cimkének nevezzük. A cimke lehet egy természetes szám, vagy betü, és szám alapjelekből álló szó, azaz azonositó, pl.: V17a, LAJOS

Az ALGOL-ban nemcsak az értékadó utasitásokat "cimkézhetjük" meg; mód van arra is, hogy azokat a pontokat megnevezzük az algoritmusban, amelyekben elágazás van; az ilyen "pontokat" kapcsolóknak nevezzük. A kapcsolóknak nevet ad-

hatunk: egy kapcsoló neve tetszőleges azonositható lehet.

Pl. kapcsoló 5

elágazás 23.

Ha a kapcsolónév után szögletes ([ ]) zárójelbe valamilyen kifejezést irunk, azaz egy indexes kapcsolónevet képezünk, akkor azt mondjuk, hogy kapcsolókifejezés jött létre.

Pl.: válassz [n - 1]

város [if y < 0 then N else N-1]

Egy helymegjelölő kifejezés lényegében nem más, mint egy olyan szabály leirása, amely valamely utasitás cimkéjét vagy egy kapcsoló nevét hozza létre; másszóval amig az aritmetikai vagy logikai kifejezések értékeket (szám, illetve logikai értékeket) adó szabályokat irtak le, addig helymegjelölő kifejezés feladata egy cimke létrehozása.

A helymegjelölő kifejezés strukturája ugyanaz, mint az aritmetikai kifejezésé, azaz

if ⊛ then B else C

ahol ⊛ valamilyen feltétel, B valamilyen cimke, vagy kapcsolókifejezés, C pedig valamilyen helymegjelölő kifejezés.

Példák:

17

p 9

válassz [n - 1];

elágazás [N - 1]

kimenet [if y > 0 then K else G]

e/ Utasitások

Az ALGOL-ban - mint a bevezető részben már emlitettük - az algoritmus operativ részeit, egységeit utasitásoknak hivjuk. Az utasitásokat általában a felirás sorrendjében kell végrehajtani, hacsak nem átirányitó vagy feltételes utasitásokról van szó: ez utóbbik szerepe éppen az emlitett sorrend megváltoztatása, a vezérlés átadása bizonyos feltételektől függően vagy függetlenül az algoritmus "ágai" közül egy ágra vagy ágak egy csoportjára.

Mint már a helymegjelölő kifejezések fogalmának ismertetésénél elmondottuk, abból a célból, hogy az utasitásokra, vagy ezek valamilyen csoportjára - pl. valamely feltételes, vagy átirányitó utasitásban - hivatkozhassunk ezeket cimkékkel láthatjuk el.

A cimkéket az utasitásokhoz "szabványosan" rendeljük hozzá; egy cimkézett utasitás alakja a következő:

cimke neve (valamilyen azonosító)	kettőspont	utasitás

azaz pl.:

kezdet 26:

utasitás

vagy

QZOR:

utasitás

e.1. Utasitástipusok

Rendeltetésük szerint az utasitások különféle tipusuak lehetnek.

1. Értékadó utasitások

Az értékadó utasitások a bennük szereplő un. baloldali változókhoz értéket rendelnek hozzá.

Az "értékadás" jele a ":=" jel (pont - egyenlőség jel). E jel jelentése lényegében a következő:

Kiszámitandó a jel jobboldalán álló kifejezés értéke, mégpedig ugy, hogy az e kifejezésben szereplő változók aktuális értékeivel elvégzendők a kijelölt müveletek és a nyert eredmény legyen a := jel baloldalán szereplő változó értéke. (A := jel voltaképpen az aritmetikai =(egyenlőség) jelnek felel meg.)

Pl. a

$$B := C+D[i+3] + h \times q[k,j] + alfa[20]$$

értékadó utasitás a B változónak az

$$5 + 7 + 3 \times 6 + 9{,}8 = 39{,}8$$

értéket adja, ha feltesszük, hogy

C értéke	5
D[i+3] értéke	7
h értéke	3
q [k,j] értéke	6
α 20 értéke	9,8

azaz itt: B: = 39,8.

További példák:

S: = p [0]: = n : = n+1+s

S [v,k + 2] := 3 - arctan s×zeta

Ha egymásután több : = jel szerepel, akkor az alábbi módon járunk el:

a/ Rendre kiszámitjuk a : = jelek baloldalán szereplő változók indexeit.

b/ Kiszámitjuk az utolsó : = jel jobboldalán álló kifejezés értékét.

c/ A b/ pontban kiszámitott értéket veszi fel mindegyik; az : = jelek baloldalán álló változó.

Aż

S:= p [o] : = n : = n+1+S utasitás tehát azt jelenti, hogy

n:= n + 1 + S

p[o] : n + 1 + S

S: = n + 1 + S

2. Átirányitó utasitások

Az átirányitó utasitások gépi kód nyelveken szólva, feltétlen ugrások. Ezen ugró utasitások, mint ismeretes, két részből állnak: az "ugorj" müveleti kódból és egy olyan cimből, amelyre a vezérlést át akarjuk adni.

Az ALGOL-ban az "ugorj" müveletet a go to alapjellel fejezzük ki. Maga az átirányitó utasitás két részből áll: a go to alapjelből és egy helymegjelölő kifejezésből.

Példák:

go to S

go to kijárat [n + 1]

go to Város [if y < 0 then N else N + 1]

go to if AB < c then 17 else q [if w < 0 then 2 else n]

3. Feltételes utasitások

A feltételes utasitások szerepe, tartalmi jelentése itt is ugyanaz, mint a gépi kódoknál, ill. az autokódoknál: a vezérlést egy adott feltételtől függően az algoritmus különböző ágaira adják át.

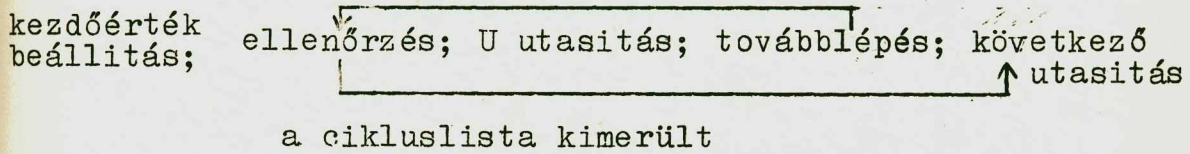
A feltételes utasitások strukturáját az alábbi példák világitják meg:

```
if x>0  then n := n + 1

if v>u  then V : q : = n + m  else go to R

if s<0∨P≤Q then AA:  begin if q<v  then a:= V/S
        else y: = 2×a end else if v>s then a := v - q
        else if v>s - 1 then go to S
```

4. Ciklusutasitások

A ciklusos algoritmusok leirásának strukturája is az ALGOL-ban lényegében olyan mint a gépi kódban, illetve az autokódban: arról tartalmaz információt, hogy mi az, amit ciklikusan ismételve kell végrehajtani (a ciklus magja) és arról, hogy a végrehajtások száma mitől és hogyan függ (ciklusparaméter).

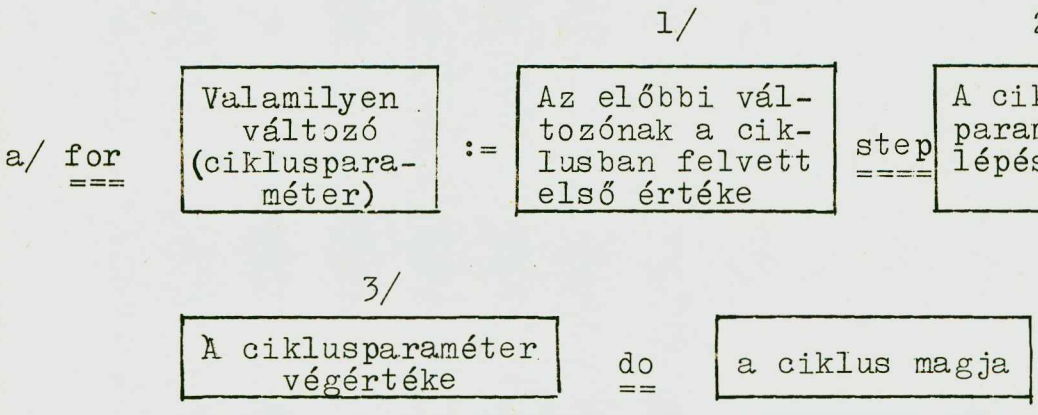
Egy ciklus általános szerkezete az alábbi:



Egy ciklus felirásánál az alábbi alapjeleket alkalmazzuk:

```
for,  step,  until,  while,  do.
```

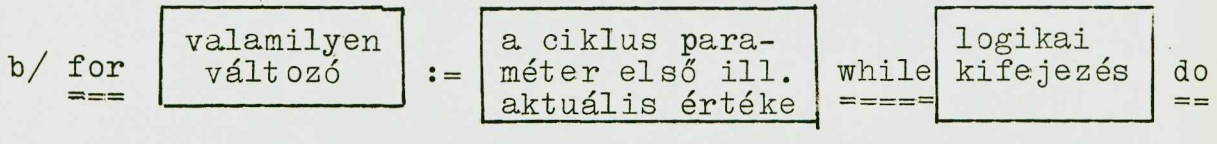
Egy egyparaméteres ALGOL-ciklus strukturája kétféle lehet:



Itt 1/ 2/ 3/ lehetnek számok, változók vagy aritmetikai kifejezések.

Pl.: for k : = 1 step 1 until h do ....

E példa jelentése: a ciklus magja végrehajtandó a k paraméter függvényében (for k); ennek 1 értékéből kiindulva (:=1) 1-es lépésközzel (step 1) egészen addig, amig k értéke el nem éri h értékét (until h).



Pl.:

for j:= 1 while k ≠ j do ....

E példa jelentése: a ciklus magja végrehajtandó a j paraméter függvényében (for j) mégpedig ennek 1 értékére (:=1) amig csak k ≠ j teljesül (while k≠j). Természetesen itt a k, valamint 1 értékét a ciklus végrehajtásakor előállitottnak tekintjük.

A többparaméteres ciklusok is egy for alapjellel kezdhetők; az egy-egy paraméterre vonatkozó információt vesszővel választjuk el.

Pl:

```
for k : = step 1, until h, p step 1 until n do ....
```

További példák ciklusutasitásokra:

```
for  q: = 1 step s until n do A [q] : = B [q]

for  k: = 1, V 1 x 2 while V 1< N do

     for j : = I + G,L,1 step 1 until N,C + D do

          A [k,j] : = B [k,j]
```

5. Eljárás-utasitások

Az eljárásutasitások lényegében a gépi kódban és az autokódban megismert szubrutin-behivó utasitásoknak felelnek meg.

Az ALGOL-programban ugyanis a "szubrutinokhoz" hasonlóan un. eljárásokat (procedure) lehet felirni: egy eljárás a "bemenetekre", az eljárás paramétereire alkalmazott valamilyen algoritmust jelent.

Az eljárásokat eljárás névvel jelöljük meg (pl: mátrix-inverzio, Absmax stb.) Egy eljárásutasitásnak tehát mindenekelőtt tartalmaznia kell a "behivandó" eljárás nevét.

Az eljárás "bemenetei" kétfélék lehetnek.

Érték-bemeneteknek vagy érték-paraméternek hivjuk az olyan bemeneteket, amelyeket olyan változó jelöl, amelynek értékét az eljárás behivása előtt elő kell állitani. Azt, hogy melyek az ilyen tipusu bemenetek, az eljárásban külön deklaráljuk (lásd eljárásdeklaráció).

Az eljárások bemeneteinek másik részét formális paraméternek hivjuk. A formális paramétereket az eljárás deklarációban szintén külön deklaráljuk. A formális paramétereknél nem kell azoknak tényleges értékét előállitani, mielőtt behivnók az adott eljárást, elegendő, ha megmondjuk, hogy a formális paramétereket milyen aktuális paraméter értékekre kell tekinteni. Az ilyen behivást "névvel történő behivásnak" hivjuk. Szemléltessük a mondottakat egy példával:

Legyen adva egy eljárásunk (nevezzük SUBR-nak), amelynek három bemenete van: B1, B2, B3. Tegyük fel, hogy a SUBR eljárást SUBR (B1, B2, B3) kifejezéssel hivhatjuk be.

Ha mind a három bemenet értékbemenet, akkor mint mondottuk - behivás előtt ezeket az értékeket elő is kell állitani, azaz programunk szerkezete ilyenkor az alábbi lesz:

```
    .
    .
    .
B1:= □  valamilyen kifejezés
        (értékadó utasitások)
    .
    .
    .
B2:= □
    .
    .
    .
B3:= □
    .
    .
    .
SUBR (B1, B2, B3).
```

Ha azonban pl. a B2 bemenet formális paraméter, akkor a behivás szerkezete az alábbi lehet:

```
    .
    .
    .
B1:= □  (valamilyen kifejezés)
    .
    .
    .
B3:= □
    .
    .
    .
SUBR (B1, C+A/B, B3)
```

Ilyenkor behivásnál tehát a formális paraméter helyére egyszerüen beirjuk azt a kifejezést, amelynek értékére akarjuk tekinteni az adott eljárást.

Az eljárás utasitások az eljárás nevét és az aktuális paraméterrészt tartalmazzák. Szerkezetük:

ELJÁRÁSNÉV (paraméterlista)

A paraméterlista kifejezésekből, tömbnevekből, kapcsolónevekből, eljárásnevekből áll; ezeket egymástól vagy ","-vel (vesszővel) választjuk el, vagy pedig a) szó: (elrendezésü jelkombinációval, ahol a szó egy tetszőleges betükombinációt jelent.

Példák:

Nyom (A) Rendszám: (7) Eredmény: (V)

Transzponálás (W, v+1)

Absmax (A, N, M, Yy, I, K)

Skalárszorzat (A [t, P, U,] B [P,] 10, P, y)

Elemezzük a második példát:

Tegyük fel, hogy a programban "deklaráltunk" (azaz részletesen felirtunk) egy TRANSZPONÁLÁS (Q, k) nevü eljárást, amelynek két bemenete van. (pl. a mátrix mint tömb és a mátrix rendszáma). Ha a programban valahol felirjuk a

TRANSZPONÁLÁS (W, v + 1)

utasitást, akkor ez azt jelenti, hogy a TRANSZPONÁLÁS-eljárást alkalmaztuk a W tömbre, amelynek rendszáma v + 1-el egyenlő. Ha a program más részén a

TRANSZPONÁLÁS (A + 2B, k + 1)

utasitást irjuk fel, akkor ez nyilvánvalóan azt jelenti, hogy az adott eljárást az A + 2B mátrixra kell alkalmazni, amelynek rendszáma k + 1.

Ha a szóbanforgó eljárásban a Q és k nem névszerinti paraméterek (ez az eljárás felirásától függ), akkor az előbbi tipusu, azaz névszerinti behivás nem lehetséges. Ebben az esetben a Q mátrixot az eljárás behivása előtt elő kell állitani és a rendszám értékét is előzőleg hozzá kell rendelni a k változóhoz; azaz ilyenkor csak

k: = □

⋮

Q: = □

⋮

TRANSZPONÁLÁS (Q,k) tipusu behivás lehetséges.

e.2. Utasitás fajták

Az utasitás fogalma "lépcsős"-fogalom: ezen azt értjük, hogy a "legszélesebb" utasitás-tipus, a blokk, összetett utasitásokat tartalmaz; utóbbi viszont egyszerü utasitásokból, vagy más néven alaputasitásokból épül fel.

a/ Alaputasitások

Alaputasitásoknak tekintjük az olyan utasitásokat, amelyek valamilyen müveletet, vagy müveletsort kijelölnek ugyan, de nem képezik egy adott eljárás "zárt", önálló részét, hanem csak részei egy önállóbb résznek: egy összetett utasitásnak. Cimkével alaputasitás is rendelkezhet.

Pl.

```
a := p+ q ;
     go to  ELÁGAZÁS 2 ;
ÁG5: t:= 7.865 + a x(i+j);
```

Az alaputasitásokat az algoritmusban pontosvesszővel választjuk el egymástól (nem kell minden alaputasitásnak uj sorban kezdődnie.) Pl.:

```
j:= 0; n [u]: = a+b [i];  go to T [beta];
a ↑ b + i [k];
```

b/ Összetett utasitások

Egy összetett utasitás strukturája az alábbi:

```
begin
=====
        ┌──────────┐
        │ alaputa- │
        │ sitások  │
        └──────────┘
end
===
```

Az összetett utasitások tehát a begin és end alapjelek közé zárt alaputasitásokból állnak. Az összetett utasitások lehetnek cimkézettek is. Jelöljünk egy tetszőleges cimkét C-vel, egy utasitást U-val; igy egy összetett utasitás szerkezete az alábbi módon irható fel:

```
C: begin U;U;,,,; U end.
   =====         ===
```

pl.:

```
P: begin h: = 0; for j: = 1 step 1 until
   =====         ===       ====   =====
   min (n-j,j) do;
       ==
   h := H+ (-1)↑ e 1× a [j-e1, m-1]×a;
   a [j,m] = X + epszilon × r [p] end.
                                  ====
```

c/ Blokkutasitás

A blokk lényegében egy önállósitott "zárt" összetett

utasitás: e két tulajdonsága azt jelenti, hogy a blokk el van látva a benne szereplő azonositókról szóló információval is: azaz tartalmaz deklarációkat is (lásd később). A blokk is begin alapjellel kezdődik és end alapjellel végződik. A blokk is rendelkezhet cimkével.

Jelöljük itt is egy tetszőleges cimkét C-vel, egy deklarációt D-vel, egy tetszőleges utasitást U-val, akkor a blokk strukturája az alábbi módon irható fel:

C: begin D;D; ... ; D; U; U ....; U end

Pl.:

```
Q: begin integer i, k; real w;                    1/

   for i:= step 1 until n do
   for k:= i + 1 step 1 until m do
   begin w: = A [i, k]; A [i, k]: = A[k, i]:      2/
         A [k, i] : = w end i-re és k-ra

   end  blokk Q
```

A Q azonositó itt egy cimke, mégpedig a blokk cimkéje. Az 1/-el jelölt bekeretezett rész a blokkhoz tartozó deklaráció, amit mindig a begin alapjel után irunk. A 2/-essel jelölt bekeretezett rész az utasitásokat tartalmazza. (Az első két sorban felirt utasitás ciklus utasitás, a harmadik és negyedik sorban felirtak pedig értékadó utasitások.) Az "end-blokk Q" kifejezés lezárja a blokkot.

A blokkban szereplő azonositók közül azokat, amelyeknek tulajdonságait a blokk deklarációja irja le, lokálisaknak nevezzük az adott blokkra nézve. Szerepelhetnek a blokkban olyan azonositók is, amelyeket a blokkon kivül deklarál-

tunk: ezek a blokkra nézve nem lokálisak. A fenti példában az i, k, w változók lokálisak voltak, mig az A [i, k] tömb nem lokális a Q blokkra nézve; a blokkon kivül természetesen valamilyen módon deklarálni kell.

f/ Deklarációk

Mint többször emlitettük már, az ALGOL-programot,ill. blokkokat deklarációval látjuk el: a deklaráció az adott blokkban szereplő azonositók tulajdonságairól szolgáltat információt, illetve az eljárás deklaráció a programban behivott eljárások leirására szolgál.

A deklarációt a blokk elejére irjuk a begin alapjel után. Ha egy, a deklarációhoz képest külső deklarációban valamely azonositó már szerepelt, és a blokk deklarációjában ujból fellép, akkor az uj deklarációtól kezdve elveszti eredeti jelentését. Azok az azonositók viszont, amelyeket a szóbanforgó blokkban nem deklarálunk, de blokkban felhasználunk, megtartják előzőleg definiált jelentésüket.

Ha a blokkból kilépünk (az end alapjelen keresztül, vagy egy átirányitó utasitással), akkor az összes a blokkra nézve deklarált azonositók elvesztik jelentésüket, hacsak a deklarációban nem szerepel egy own deklarátor alapjel. E jel hatására ugyanis az adott blokkban való minden ujabb belépés alkalmával az own jellel megjelölt mennyiségek ujból felveszik azt az értéket, amellyel a blokkból való utolsó kilépéskor rendelkeztek.

A deklarációk deklarátor-alapjelekből és azonositókból állnak. Egy és ugyanazon blokkfejben egyetlen azonositót sem szabad egynél többször deklarálni.

A deklarációban az egyes deklarációk

deklarációs-alapjel	felsorolás (vesszővel elválasztott azonositók)

szerkezetüek.

Pl.:

integer x , a, beta;

A deklarációk fajtái

Egy deklaráció lehet:

a/ tipusdeklaráció

b/ tömbdeklaráció

c/ kapcsoló deklaráció

d/ eljárásdeklaráció

1. Tipusdeklaráció

A tipusdeklarációval a blokkban előforduló változókat, illetve ezek értékeit jellemezzük; azaz az értékek azonositóiról megmondjuk, hogy azok real, integer, vagy Boolean tipusuak-e.

Szerkezetük: a fenti három alapjel után egyszerüen felsoroljuk azokat az azonositókat, amelyeket adott tipusunak akarunk deklarálni. A felsorolás után pontos-vesszőt irunk.

Példák:

```
integer p, q, s,
real X, Y, GX, rp;
Boolean A, ROOT, epsilon;
```

2. Tömbdeklaráció

A tömbdeklarációval az indexes változókat, illetve az általuk képviselt tömböt deklaráljuk. A tömbdeklarációban megadjuk a tömbök dimenziószámát, az indexek korlátait, és a változók tipusát.

Az indexek korlátait két, egy kettősponttal elválasztott aritmetikai kifejezéssel, un. korlátpárral adjuk meg; a kettőspont baloldalán álló jelenti az adott index alsó, a jobboldalon álló pedig a felső korlátját.

Pl.:

7:n

1:n+1

A tömbök dimenziószámát ugy jelöljük, hogy annyi előbbi tipusu, és vesszővel elválasztott korlátpárt irunk le, ahány dimenziósnak tekintjük az adott tömböt.

Pl.: 7:n, 1:n+1, 5:38 egy háromdimenziós tömböt jelöl.

A dimenziószám és a korlátpár segitségével egy tömbdeklarációt a következőképpen szerkesztünk meg:

a/ leirjuk a deklarálni kivánt tömb tipusát,

b/ leirjuk az array alapjelet

c/ utána irjuk a tömb-azonositót

d/ a korlátpárokat vesszővel elválasztva az azonositó után szögletes zárójelbe irjuk.

Ha több tömböt deklarálunk egyszerre, akkor az egyes tömbökre vonatkozó deklarációkat vesszővel választjuk el. Az array alapjel utáni részt tömblistának nevezzük. Ha tipusjelzés nincsen, akkor az adott tömböt real-tipusunak tekintjük.

Példák:

```
array a, b, c [7:n, 2:m,] s [-2:10]
own integer array A [if c<0 then 2 else 1:20]
real array q [-7:-1]
```

3. Kapcsolódeklarációk

A kapcsolódeklaráció arról tartalmaz információt, hogy az egyes a blokkban szereplő kapcsolók milyen "ágakat" jelölhetnek ki az algoritmusban. Az ágakat valamilyen kapcsoló képviseli.

Szerkezete:

```
switch  [kapcsolónév]  :=  [kapcsolólista]
```

Példák:

```
switch S:= S1, S2, Q [m], if v< -5 then S3
                  else S4
switch Q: = p1, w
```

4. Eljárásdeklarációk

Az eljárásdeklaráció a procedure alapjel után álló eljárásnévvel megjelölt eljárás leirását adja meg.

Az eljárás fogalmát már emlitettük az eljárásutasitás értelmezésénél: ez lényegében nem más, mint egy olyan zárt egység, amely bizonyos bemenetekre realizál valamilyen algoritmust.

Egy eljárás-deklaráció szerkezete az alábbi:

```
procedure [eljárás fej] [eljárás törzs] end [eljárás név]
```

Eljárás fej

Az eljárás fej az eljárás nevével kezdődik. Az eljárás neve után leirjuk a formális paraméter listát. A formális paraméter lista egymástól paraméter elválasztó jelekkel elválasztott paraméterekből áll. A paraméterek azonositók. A paraméter elválasztó jel, vagy "," (vessző), vagy) szó : (szerkezetü jelkombináció. (A szó egy tetszőleges betükombináció.) A formális paraméterlistát ;-vel zárjuk le.

Példák:

1. Bessel függvény (N, FX, LX, Z) Eredmény: (J,y);
2. Inverzió (A) rendsz. :(n) Inverz: (A 1);
3. Q (x);
4. Komplex osztás (a,b,c,d) Eredmény: (e, f);
5. RK (x,y,n, FGV, epsz, eta, XV, YV, fi);

Az aláhuzott rész a példákban az eljárás neve; a többi a formális paraméterlista.

A formális paraméterlista után az eljárásfejben az un. értékrész következik. Ennek szerkezete:

value	azonositó-lista (vesszővel elválasztott azonositók)	;

Az azonositólistán azokat a paramétereket soroljuk fel, amelyeknek behivása értékszerint történik egy eljárás utasitásban.

Pl.: value n, xo, yo, epszilon;

Az értékrész után a specifikációs részt irjuk fel az eljárás-fejben. A specifikációs rész az eljárás paramétereinek tipusáról, valamint az eljárás más objektumairól tartalmaz információt, ezeket deklarálja. (tipusdeklaráció stb.)

A specifikációs rész a következőkből áll:

a/ Tipusdeklarációs rész: itt a real, integer, Boolean, alapjelek után vesszővel elválasztva leirjuk azokat a változókat, amelyeket adott tipusunak deklarálunk; az egyes tipusdeklarációkat pontosvesszővel választjuk el.

Pl.: real a, b, omega; integer i, n, kappa; Boolean fi, lamda, u, v;

b/ Tömbdeklarációs rész: itt a

[tipus] array formában felirt

alapjelek után felsoroljuk a tömböket (ha tipust nem irunk, akkor az real tipust jelent).

Pl.: 1/ array x, y;

2/ integer array p, q, epszilon;

3/ real array delta, r, zeta, gyök;

c/ Procedure-specifikátor

Szerkezete:

tipus	procedure	felsorolás eljárás nevek (azonositók)

E részben felsoroljuk azokat a procedurákat, amelyeket maga a most deklarált eljárás behiv.

Pl.: real procedure Bessel;

procedure gamma

procedure RK L

(Ha tipust nem irtunk, akkor real tipust értünk.)

d/ Kapcsoló deklaráció

Szerkezete:

switch ⟨kapcsolónév⟩ : = ⟨kapcsolók felsorolása (vesszővel elválasztva)⟩

Pl.: switch U : = U1, U2;

e/ Cimke deklaráció

Szerkezete:

label ⟨felsorolás⟩

Az eljárás fejben comment alapjellel inditva szöveggel az eljárás lényege, vagy a fontosabb tudnivalók leirhatók.

Az eljárástörzs blokkokból áll.

Az eljárás deklarációt

end ⟨eljárás neve⟩

szerkezetü résszel zárjuk le.

Példa eljárás deklarációra:

```
procedure RES (n, c, alfa, mu, re, im, rt);
value, n, c, alfa; integer n, alfa; integer array mu;
array c, re, im, rt; switch F:= Ki, Be; procedure DET;
comment Az eljárás Rezultáns módszerrel megadja egy n-ed-
foku polinom gyökeit.
        ⟨Eljárástörzs⟩
end RES
```

Az alábbi procedure-példánkban konkrét eljárásokat irunk le.

1. Példa:

```
procedure Transzponálás: (a) Rendszám: (n); value n
comment  az eljárás egy mátrix transponáltját képezi;
array a; integer n;
begin real w; integer i, k;
for i:= 1 step 1 until n do
          for k:= i+1 step 1 until n do
          begin w: = a [i,k] ;
                a [i,k] : = a [k, i] ;
                a [k,i] : = w end
                end  Transzponálás
```

2. példa:

```
procedure Absmax (a) méretek: (n,m) Eredmény:(y)
Indexek: (i,k);
comment y az m-szer n-es  a  mátrix legnagyobb abszolutértékü elemének értékével lesz egyenlő; az  i  és  k  változók pedig a megfelelő indexpár változói lesznek.
array a; integer n, m, i, k; real y;
begin integer p, q;
y : = 0;
for p: = 1 step 1 until n do for q : = 1 step 1 until
m do
if abs (a [p,q])>y then begin y: = abs (a [p,q]);
```

i:= p; k: = q end

end Absmax

Elemezzük a második példát.

A procedure alapjel után az első pontosvesszőig tartó rész az eljárás feje. Az eljárás fejben Absmax az eljárás neve; a többi pedig a formális paraméterlista. Eszerint ennek az eljárásnak a paraméterei, bemenetei a következők:

a a mátrix neve "helye"

méretek: (n,m); a mátrix rendszáma

Eredmény: (y); a mátrixnak az ép most definiálandó eljárás által kiválasztott legnagyobb elemét megnevező változó (a legnagyobb elemet ehhez rendeljük hozzá).

Indexek: (i,k); a legnagyobb elem sor- és oszlopindexének értékét az i, és k változókhoz rendeljük hozzá.

A comment alapjel után a pontosvesszőig az eljárás rövid leirása szerepel.

Az

array a; integer n, m, i, k; real y;

rész az eljárás fej deklarációs része. Eszerint a egy egydimenziós tömb (sorfolytonosan felirt mátrix); az n,m,i, k változók integer tipusuak; az y változó pedig real tipusu.

A következő sorban begin alapjellel egy blokk kezdődik; a blokk-deklaráció:

integer p, q ;

vagyis a blokk lokális azonositói p, és q. A blokk az első end alapjelig tart.

Az y : = 0 értékadó utasitás az y változóhoz a 0

értéket rendeli hozzá (a továbbiakban az y változó lesz az addig kiválasztott legnagyobb elem változója, azonosítója).

A következő utasitás kettős ciklus-utasitás. Az egyik ciklus paraméter (a külső) p, a másik q. A külső ciklus 1-től 1-es lépésközzel n-ig megy (a sorok száma); a belső ciklus ugyancsak 1-től 1-es lépéssel megy egészen m-ig (az oszlopok száma), a következő utasitás a ciklus magját képezi. A mag az

if abs (a [p, q])>y then

feltétellel kezdődik, amely a p,q indexü elem értékét hasonlitja össze az y változó értékével. Ha a feltétel teljesül, akkor a vezérlést a

begin y: = abs (a [b,q]) ; i: = p; k:=q end

összetett utasitásra adja át (begin-nel kezdődik, end-del végződik és a két alapjel között alaputasitások vannak). Ez az összetett utasitás három értékadó utasitásból áll. Az első

y:= abs (a [p , q]) ;

az y változóhoz az a [p,q] változó értékét, mig a második ill. harmadik az i változóhoz a p változó értékét, a k változóhoz pedig a q változó értékét rendeli hozzá (azaz az eddig megtalált legnagyobb elemet és ennek indexeit rögziti). Az end Absmax az eljárás végét jelzi.

Ha a feltétel nem teljesül, akkor a ciklust folytatja.

Összefoglalás

Az alábbiakban röviden összefoglaljuk az ALGOL-nyelv legfontosabb alapfogalmait és az egyes objektumok szerkezetét.

1. Változó

Jele: valamilyen azonositó

Indexes változó:

Index jele: [változók vesszővel elválasztva]

Indexes változó jele: | Azonositó | | Index |

2. Kifejezés

Fajtái: a/ aritmetikai
b/ logikai
c/ helymegjelölő

Tipusai:
a/ Feltétel nélküli
b/ Feltételes (feltételt tartalmazó)

3. Utasitások

Fajtái:
a/ alaputasitás
b/ összetett utasitás

Szerkezete:

| cimke | : begin

| alaputasitások (vesszővel elválasztva) |

end

c/ blokk

Szerkezete:

| cimke | : begin

| deklaráció |

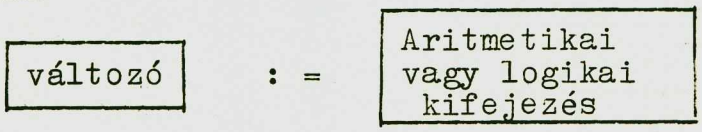
| összetett utasitások |

end blokk

Utasitás tipusok

a/ Értékadó utasitás

Szerkezete:

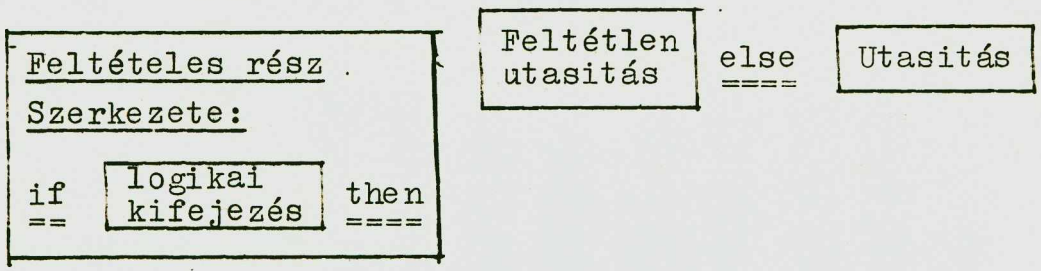


b/ Átirányitó utasitás

Szerkezete:

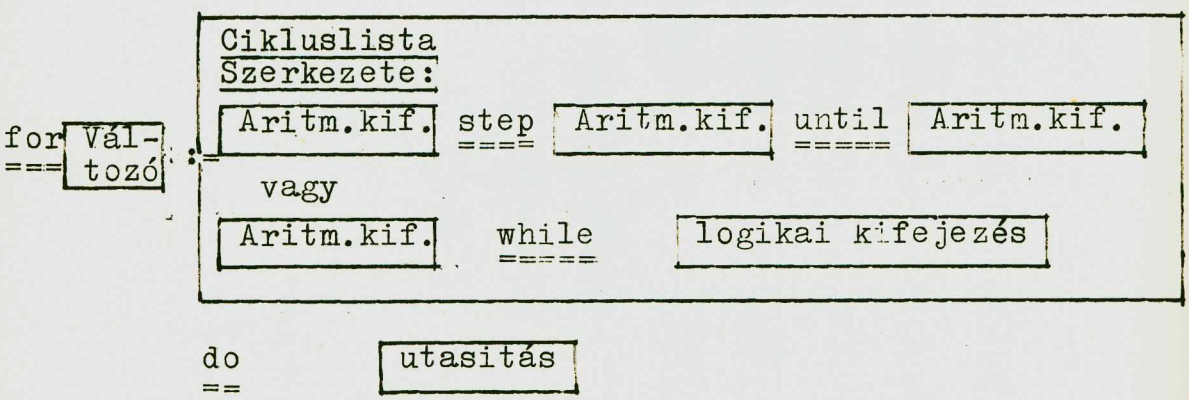
c/ Feltételes utasitás

Szerkezete:



d/ Ciklusutasitás

Szerkezete:



e/ Eljárás utasitás

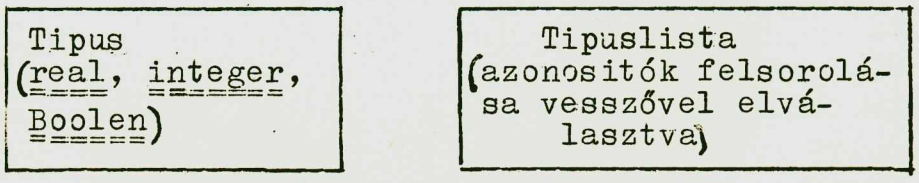
Szerkezete:

4. Deklaráció

Fajtái:

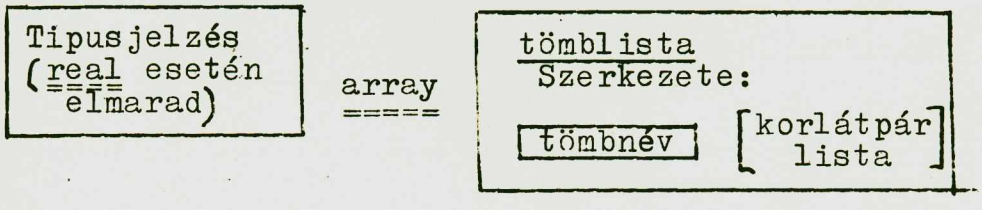
a/ Tipusdeklaráció

Szerkezete:

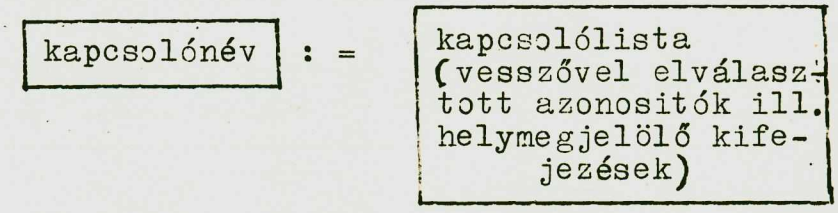


b/ Tömbdeklaráció

Szerkezete:



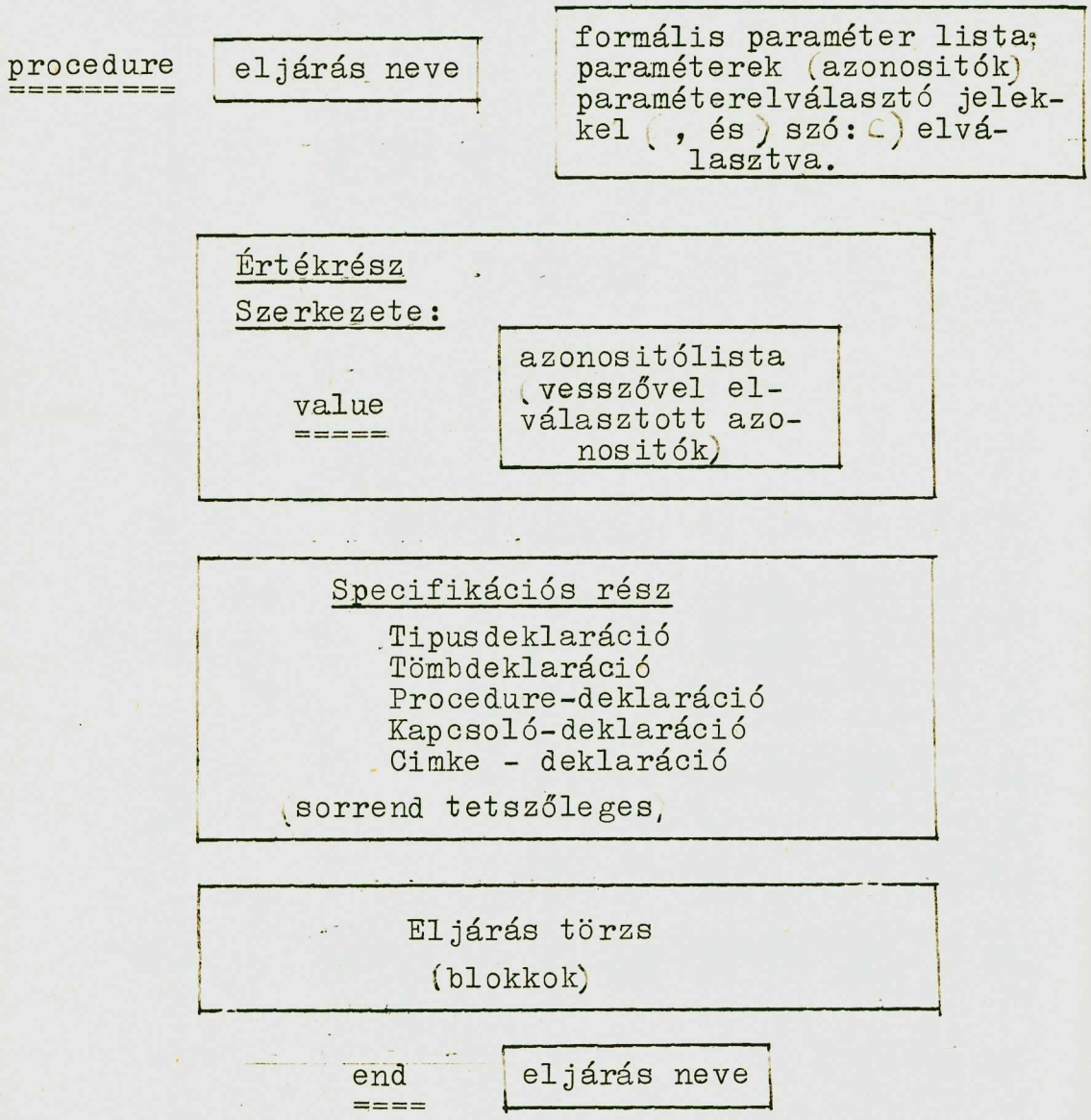
c/ Kapcsolódeklaráció



d/ Eljárásdeklaráció

Minden ALGOL-beli program egy szabványos módon deklarált eljárás. Bármely eljárás része lehet egy eljárásnak; bármely eljárás deklaráció maga is tartalmazhat eljárásdeklarációkat.

Szerkezete:



Példák ALGOL-nyelven felirt algoritmusokra

A következőkben az ALGOL-nyelv gyakorlása céljából felirunk néhány algoritmust.

1. Interpoláció Aitken módszerrel. (lásd Communications of the ACM 11. szám 1961. nov. 497. old.)

Az Aitken módszer Lagrange-féle polinommal közeliti meg egy n+1 pontban adott függvény tetszőleges x pontban felvett értékét, mégpedig az alábbi módon:

Tegyük fel, hogy a függvény az $x_0, x_1, \ldots, x_n$ pontokban vett $f(x_0) = f_0$, $f(x_1) = f_1, \ldots, f(x_n) = f_n$ értéke ismert. Képezzük a következő sémát:

$$
\begin{array}{l}
x_0 - x;\ (f_0); \\
x_1 - x;\ (f_1);\ (f_0, f_1); \\
x_2 - x;\ (f_2);\ (f_0, f_2);\ (f_0, f_1, f_2); \\
\ldots\ldots\ldots\ldots\ldots\ldots \\
x_n - x;\ (f_n);\ (f_0, f_n);\ (f_0, f_1, f_n) \ldots\ (f_0, f_1, \ldots f_n)
\end{array}
$$

A sémában $(f_0, f_1, \ldots, f_k)$ -val az $x_0, x_1, \ldots, x_k$ pontokra támaszkodó interpolációs polinomnak az adott x pontban felvett értékét jelöltük. Ezeket az értékeket oszloponként rekurzive számithatjuk ki az alábbi összefüggések alapján:

Az első oszlop k + 1-edik elemét, az

$$(f_k) = f(x_k) \qquad (k = 0,1,2,\ldots n)$$

összefüggés alapján, a második oszlop elemeit az

$$(f_0, f_k) = \frac{(x-x_0)\, f_k - (x-x_k)\, f_0}{x_k - x_0} \qquad (k=1,2,\ldots, n)$$

összefüggés alapján; a harmadik oszlop elemeit

$$(f_o,f_1,f_k) = \frac{(x-x_1)(f_o,f_k) - (x-x_k)(f_o,f_1)}{x_k - x_1} \quad (k = 2,3,\ldots n)$$

összefüggés alapján, s.i.t. a j-edik oszlop elemeit a

$$(f_o,f_1,..,f_{j-2},f_k) = \frac{(x-x_{j-2})(f_o,f_1,\ldots,f_{j-3},f_k)-(x-x_k)(f_o,f_1,..f_{j-2})}{x_k - x_{j-2}}$$

összefüggés alapján kapjuk.

ALGOL-nyelven az eljárás a következőképpen épithető fel:

a/ Eljárás fej

```
procedure  AITKEN-interpoláció  x,f,n,X,F ;
=========
real, array x, f; integer  n;  real  X, F;
====  =====       =======      ====
```

Itt:

1/ AITKEN-interpoláció az eljárás neve.

2/ (x,f,n, X,F) a formális paraméterlista: az eljárásban 5 paraméter szerepel:

x : az x_o, x_1,..., x_n argumentum értékek tömbje

f : az f_o, f_1,..., f_n függvényértékek tömbje

n : a pontok száma,

X : a megadott hely változója, ahol függvényértéket keressük,

F : a megtalált függvényértéket ehhez a változóhoz rendeljük hozzá.

3/ A

```
real array  x, f; integer n: real  X, F;
==== =====      =======     ====
```

rész az eljáráshoz tartozó deklaráció.

Eszerint: x és f egy-egy valós tömb

n egy egész tipusu változó

X, F real-tipusu változó

b/ Az algoritmus rövid leirása

Comment. Az algoritmus egy f(x) függvény $x_0, x_1, \ldots,$ x_n pontokban adott értékei segitségével kiszámitja a függvény tetszőleges x helyen felvett értékét AITKEN interpoláció segitségével. Minden paraméter névszerinti; itt f az algoritmus végrehajtása folyamán törlődik, mert a részeredményeket, az f tömbhöz rendeljük hozzá;

c/ Eljárás törzs

Az eljárás törzs kettős ciklusból áll. A belső ciklus a felirt séma egy-egy oszlopát hozza létre; ("sor-ciklus"). a külső pedig az oszlopok "léptetését" végzi (oszlop-ciklus).

Legyen a belső ciklus paramétere i. Jelöljük $f_{[i]}$-vel a j-edik oszlop i-edik elemét, vagyis az előbbi jelöléssel

$$f_{[i]} = (f_0, f_1, \ldots, f_i)$$

azaz az első oszlop képzésekor

$$f_{[1]} = (f_0, f_1)$$

$$f_{[2]} = [f_0, f_2]$$

$$\vdots$$

$$f_{[n]} = [f_0, f_n]$$

a második oszlop képzésekor

$$f[2] = (f_0, f_1, f_2)$$
$$f[3] = (f_0, f_1, f_3)$$
$$\vdots$$
$$f[n] = (f_0, f_1, f_n)$$

a harmadik oszlop képzésekor

$$f[3] = (f_0, f_1, f_3)$$
$$f[4] = (f_0, f_1, f_2, f_4)$$
$$\vdots$$
$$f[n] = (f_0, f_1, f_2, f_n) \text{ s.i.t.}$$

Ezekkel a jelölésekkel a j-edik oszlop i-edik eleme az

A (alaputasitás)

$$\{ f[i] := (X - x_j) \times f_i - (X - x_i) \times f_j / (x_i - x_j) \}$$

értékadó utasitással képezhető. Ez, mint a Comment-ban mondottuk az előző oszlop létrehozott értékeit "törli".

A sor ciklus lépéseinek száma az oszlopciklus sorszámának függvényében csökken: a sorok száma a nulladik oszlopban n+1, az első oszlopban n, a másodikban n-1, a harmadikban n-2, s.i.t. azaz a sorindex mindig j+1-el kezdődik. Ezért a belső ciklus a következőképpen alakul:

B belső ciklus: (összetett utasitás)

```
{ begin  t := j + 1
=====
        for i := t step 1 until n do.
        ===          ====   =====   ==
            A alaputasitás          end. }
                                    ====
```

A külső ciklusban a j paraméter 0-tól n-1-ig megy, ezért a kettős ciklus a következő lesz:

C (összetett utasitás)

```
for j := 0 step 1 until n-1 do
        .
    B belső ciklus
F := f [n]
```

(Az F := f [n] nem tartozik a ciklushoz.)

Ha a C összetett utasitást deklarációval is ellátjuk, akkor a

D (blokk)

```
begin integer i, j, t.
        C összetett
          utasitás
end
```

eljárástörzshöz jutunk.

A teljes eljárás:

```
procedure AITKEN interpoláció (x,f,n,X,F);
    real array x, f; integer n; real X, F;
comment: Az algoritmus egy f (x) függvény
```

$x_0, x_1, \ldots, x_n$ pontokban adott értékei segitségével kiszámitja a függvény tetszőleges x helyen felvett értékét Aitken interpoláció segitségével. Minden paraméter névszerinti; az f tömb az algoritmus végrehajtása folyamán törlődik, mert a részeredményeket az f tömbhöz rendeljük hozzá;

```
begin integer i,j,t;
    for j := 0 step 1 until n-1 do
      begin t := j+1
        for i := t step 1 until n do
```

```
f [i] := ((X - x [j])× f [i] - (X-x [i])× f [j])
                (x [i] - x [j])                    end
                                                   ===
        F := f [n]

end
====
```

2.Példa

Legyen adva az

$$y' = \left(\frac{f/5x-36/ \sin/5x^2-0.6/+\sqrt{y^2+f^2/x/}}{y^3 - 6e^{x+2}x^2 + \Omega^2} \right)^{\frac{3}{2}} \ln /x^5-2/$$

differenciálegyenlet, ahol f/x/ egy 500 pontban táblázatosan megadott függvény, Ω pedig egy paraméter. Keressük meg a differenciálegyenlet megoldását az $x_1 = x_i + h$ pontokban és a megoldás minimumát az /a,b,/ intervallumban.

Tegyük fel, hogy az f/x/ függvény az a,b intervallum t_1, t_2, t_3, t_4 ... t_{500} pontjain adott. Ahhoz, hogy tetszőleges x pontban kiszámitsuk az f/x/ értékét,szükség van az előző példában felirt Aitken interpolációs eljárásra.

A differenciálegyenlet integrálását Runge-Kutta módszerrel végezzük el; szükségünk van tehát egy Runge-Kutta eljárásra is.

A differenciálegyenlet jobboldalának kiszámitása céljából egy függvényeljárást kell felirni.

Az integrálás a következő programmal valósitható meg:

```
procedure Differenciálegyenlet -l integrálása
==========
(a,b,h, omega, y kezdeti, Minimum, Megoldásérték);
value, a,b,h, y kezdeti; real array Megoldásérték;
=====                    ====  =====
real Minimum, omega.
====
```

comment az eljárás a megadott differenciálegyenletet integrálja az /a,b/ intervallumban, Runge Kutta módszerrel. A megoldás $x_i = x_{i-1} + h$ pontban nyert értékeit a Megoldásérték-vektorhoz rendeli hozzá, a minimumot pedig a Minimum változóhoz.

Az eljárás konkrét behivásakor gondoskodni kell az a,b,h,omega, y kezdeti, bemenő paraméterek előállitásáról, valamint biztositani kell az f/x/ függvény-táblázatnak a megfelelő argumentumokkal együtt a t függvénytömbhöz való hozzárendelést.

```
procedure Aitken-interpoláció (x,f, m, X, F);
    value f; real array x,f; integer m; real X, F;
comment az f paraméter értékszerinti behivásával azt biztositjuk, hogy az eljárás végrehajtása után a táblázat még rendelkezésre álljon. Feltesszük, hogy interpolálandó függvényünk 500 pontban adva van.
begin integer i, j, t;
    for j: = j + 1
       for i : = t step 1 until n do
       f [i]: = (X - x[j])× f [i] - (X - x [i])× f [j])
                (x [i] - x [j])   end
       F : = f [n]
end
procedure Jobboldal  x,y, n, z  value x, y; integer n;
    real z ;
comment az eljárás a differenciálegyenlet jobboldalát számitja ki, és z-hez rendeli hozzá
```

```
begin real F1; F2, real array t [1:500] függv. [1:500];
      Aitken interpoláció (t, függv, 500, 5*36, F1);
      Aitken interpoláció (t, függv, 500 , x, F2 );
z := (F1 X sin (5x↑2 - 0.6) + sqrt (y↑2 + F2↑2)
     (y↑3 - 6 exp (x+2) + omega↑2) ↑ 1.5 × ln/x↑5-2/
end
end procedure Jobboldal
procedure RK (x,y,n,h,FGV, XV, YV)
        value x, y; integer n real x, XV, h
        array y, yv
comment
```

A RK procedure által realizált Runge Kutta módszer a következő:

Ha adva van a

$$\frac{dy_i}{dx} = f_i\ x, y_1, y_2, \ldots y_n \qquad i = 1,2,\ldots, n$$

lineáris differenciálegyenletrendszer és a megoldások értékeit ismerjük egy adott x_o pontban, akkor az $x_1=x_o+h$ pontban az $y_i\ /x_1/$ értékeket az

$$y_i/x_1/ = y_i/x_o/ + \frac{1}{6}\ /k_{1i} + 2k_{2i}, 2k_{3i} + k_{4i}\ /$$

összeg adja, ahol

$$k_{1i} = h\ f_i\ (x_o, y_1(x_o), y_2(x_o), \ldots, y_n(x_o))$$

$$k_{2i} = h\ f_i(x_o + \frac{1}{2}\ y_i(x_o) + \frac{1}{2}\ k_{11}, y_2(x_o) + \frac{1}{2}\ k_{12}, \ldots$$

$$\ldots, y_n(x_o) + \frac{1}{2}\ k_{1n})$$

$$k_{3i} = h\, f_i\left(x_0+\frac{1}{2}h, y_1(x_0)+\frac{1}{2}k_{21}, y_2(x_0)+\frac{1}{2}k_{22}, \ldots\right.$$

$$\left.\ldots, y_n(x_0)+\frac{1}{2}k_{2n}\right)$$

$$k_{4i} = h\, f_i\left(x_0+h; y_1(x_0)+k_{31}, y_2(x_0)+k_{32}, \ldots, y_n(x_0)+k_{3n}\right)$$

A fenti eljárás a h integrálási lépésközt nem keresi meg automatikusan. Kiegészithető azonban egy olyan eljárással, amely előre adott hibahatárhoz automatikusan választja meg az integrálási lépésközt.

```
begin array  W [1:n], a [1:5]
      integer  k, j.
      a [1]: = a[2]: = a[5]: = h/2; a[3] : = a[4]: = h:xv:= X;
      for k:= 1 step 1 until n do yv[k] := w[k] := y[k];
      for j : = 1 step 1 until 4 do
begin
      FGV /xv, w, n, z/ ;
      xv := x + a [j];
      for k := 1 step 1 until n do
      begin w [k]: = y [k] + a [j] × z [k];
            yv[k]: = yv[k] + a [j + 1] × z [k]3
      end k
end j
end RK;
      comment a program itt indul
      E:
        begin integer i; array Megoldásérték
```

```
            x := a + h; y := y kezdeti; Minimum:= y kezdeti
end.
Ismétlés:
      begin real XUJ, YUJ;
            RK (x,y,l,h, Jobboldal, XUJ, YUJ)
      if YUJ <Minimum then Minimum := YUJ
      else begin
            x := X + h; for i = step 1 while x ≤ b
      do Megoldásérték [i]:= YUJ
      go to Ismétlés; end
end
end blokk E
```

comment az E blokk végrehajtásával megkaptuk a megadott Ω-hoz tartozó /a,b/ intervallumbeli megoldásértékeket és ezek minimumát. Ezek a Megoldásérték tömbből és a Minimum "változóból" vagy kiirathatók, vagy az eljárást folytatva, valamely célra felhasználhatók. Ha pl. az Ω paraméter vagy az y kezdeti paraméter változó értékei mellett a fenti algoritmust többször végre kell hajtani, akkor gondoskodni kell megfelelő változtatásukról és a vezérlést az E cimkére kell átadni. Példánk olyan eljárás, amely maga is tartalmaz és behiv más eljárásokat.

A program végrehajtása az E cimkénél kezdődik, az ezelőtt lévő rész a deklarációs rész: itt deklaráltuk az összes, a főeljárásban szereplő procedurákat; ezeket eljárásutasitással hivtuk be.

Gyakorlatok:

1/ Ellenőrizzük, hogy az alábbi eljárás valóban egy

komplex szám logaritmusát számitja-e ki, azaz a c + di = ln (a + bi) algoritmust valósitja-e meg. (Mint ismeretes: $\ln (a + bi) = \ln\sqrt{a^2 + b^2} + (\operatorname{artg} \frac{b}{a})\, i$, ha a főágat tekintjük.)

Keressük meg a c és d változók végértékét, ha

a = 2 b = 5.

```
procedure LOG C /a, b, c, d/; value a, b;
    real a, b, c, d;
comment komplex szám logaritmusa;
    begin c: = sqrt (a×a + b×b)
          d: = arctan (b/a)
          c: = log (c)
end LOG C;
```

2/ Deklaráljuk, az $ax^2 + bx + c = o$ másodfoku egyenlet gyökeit az

$$x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$ megoldó

képlet alapján kiszámitó, alábbi blokkokból felépitett eljárást. (procedure)

```
    begin
          Diszkrimináns: = b ↑ 2 - 4 a c;
          if Diszkrimináns > 0 then
    Re x 1:=(- b + sqrt Diszkrimináns) / 2a
    Im x 1 := 0;
    Re x 2 := (- b - sqrt Diszkrimináns) / 2a ;
    Im x 2:= 0;
```

```
      else begin
           Re x 1 := - b
           Im X 1 := sqrt abs (Diszkriminánsz)/ 2 a ;
           Re x 2 := - b
           Im x 2 := - sqrt abs (Diszkriminánsz)/ 2 a; end.
end.
```

3/ Keressük meg a változók végértékeit az alábbi programban:

```
      real rl, ra, rb;
      integer n, i, j;
      n : = 5;
      rl := n (n + 15);
      rb := n + 6 / (6×rl + 0.5);
      i := n := n - 2;
      j := rb - i ;
      ra := (j - i) ×rl × (rb - 4);
      rl := ra + rb + n + i + j + 8×rl;
      rb := (rl - rb × n + j -ra) ↑ (rb - j) + ra;
      j := n := 1 + n ÷(j - 2);
      i := r + ra;
```

(Megoldás: rl=23; ra = 2; rb = 10; n = 2; i = 4; j = 2).

4/ Állapitsuk meg a változók végértékeit az alábbi programban:

```
begin integer i, j; integer array A [1:3, 1:2],
                                C [0:2];
      i := j := 1;
```

```
C [j-1] := A [j,i] := j := i + 2 × j + 2 ;
A [2 × i, [C j-2-3 × i] - 3] := j - 2 × 1;
C [A [2, 2×j - 8] -3 := i := i + j ;
A [C [j-i+1] /2, 4×A [1,1] - 3×i] := A [1, 2× (i-j)] :=
      A [2,2] - A [1,1] ;
i := -A [3,2] ;
j :=  i-j ;
A [i, - j -2] := C [i-1] := 7 ;
A [A [2,2], C [1] - C [0] := C [i] := 2×i
end.
```

Megoldás:

i = 2; j = -3; A [1,1] = 5; A [1,2] =-2, A [2,1] =7, A [2,2] = 3; A [3,1] = 4; A [3,2] = -2; C [o] = 6; C [1] = ; C [2] = 4).

5/ Az alábbi utasitások bizonyos értékeket adnak a SUM változónak. Keressük meg az első négy ilyen értéket.

```
real p, q, SUM;
integer n;
n : = 1;
p :  0,5 ;
SUM : = 0;
q : = 1 ;
ciklus : SUM := SUM + q/n;
q := q x p ;
n := n + 1 ;
go to ciklus ;
```

(Megoldás: SUM= 0; 1; 1,25; 1,333333.)

6/ Az alábbi programban keressük meg a STOP cimke elérésekor értelmezett változók végértékeit:

```
begin real  W, S, B, C ;
W := 8; S := 3 ;
B := 2xW - S ; C := B - W ;
begin real P, W ;
        W := B - 2× C ; P := C↑2 - B ;
A A :   W := P - 2× W;
        C := C + 1; if W>1 then go to AA;
        S := W - P + S
end;    W := W - C + S ;
STOP: end;
```

(Megoldás: W = -8; S = -9; B = 13; C = 7)

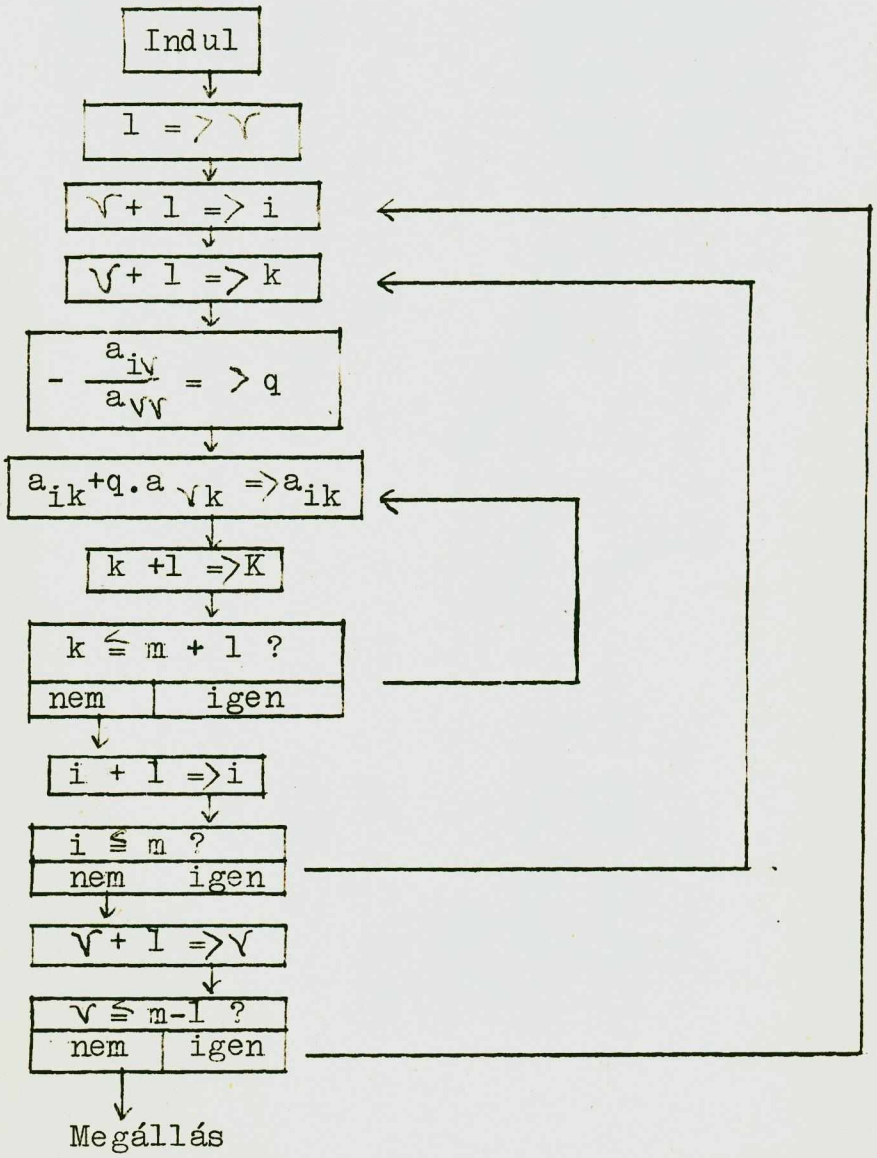
7/ Keressük meg az alábbi programban a változók végértékeit:

```
        real u, W ;
        Boolen B ;
        u := 3 ;
        B := true;
ism:    W := u - 2
        if u↑2 - 1/u > 0∧W > -2 then u := 1/u
        else if B then go to Z
        else go to  vége;
```

```
Z :     B : = false ;
        u : = W + 2 x U ;
        go to ism. ;
vége:   B : u ≥ W ;
```

(Megoldás: B = true; u = 13/15; W = -17/15)

8/ Irjuk fel az alábbi blokkdiagrammal megadott eljárást ALGOL-nyelven (Gauss-elimináció egyenletrendszerek megoldásánál).



AZ ALGOL-NYELV SZINTAKSZISA

A szemantikus leirás alapján ugyan könnyebb áttekinteni az ALGOL-nyelv strukturáját; pontosan egyértelmüen azonban a szintaktikus leirással lehet a nyelvet ismertetni.

Az alábbiakban megadjuk az ALGOL-nyelv lényegesebb fogalmainak szintaktikus leirását is.

Ahhoz, hogy magát a nyelvet leirjuk, egy másik nyelvre, egy "metanyelvre" van szükség. Az ALGOL-metanyelv alapszimbolumai a következők:

a/ "Metazárójel" : < >

Ennek jelentése a következő: a metazárójelbe irt szimbolumok a metanyelv változói; ezeknek a változóknak az "értékei" lesznek az ALGOL-nyelv "fogalmai".

b/ "Vagy"-jel: / . Ez a jel a vagy kötőszót jelöli.

c/ "Négypont-egyenlő" jel: ::= definició jel .
Jelentése: a jel baloldalán álló metanyelvbeli szimbolum értéke a jobboldalon álló szimbolumok lehetnek.

Példák:

a/ Metanyelvi változók:

⟨reláció jele⟩

⟨egészszám⟩

⟨logikai tag⟩

b/ Vagy jel:

⟨természetes szám⟩ ⟨valódi tizedestört⟩

Olvasása: természetes szám vagy valódi tizedestört.

c/ Definíciójel:

⟨számjegy⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Olvasása: számjegy lehet: 0, vagy 1, vagy 2, vagy 3, ..., vagy 9.

⟨zárójel⟩ :: = (|) | [|] | ' | ' | begin | end. |

Olvasása: zárójel lehet (vagy) vagy [, vagy] ...., s.i.t.

Az ALGOL-nyelv metanyelven történő leirásánál használjuk a láncképzés müveletet is: ha előzőleg definiált metaváltozókat egymásmellé irunk, ez azt jelenti, hogy egy uj változót definiáltunk, amely az előző két változó adott sorrendben történő felirása által jön létre.

Pl.: ⟨y⟩ :: = ⟨a⟩⟨b⟩

Olvasása: az y változó nem más, mint az ⟨a⟩ változó és utána ⟨b⟩.

Pl.: ⟨szó⟩ ::= ⟨betü⟩ | ⟨szó⟩ ⟨betü⟩.

Azaz: szó lehet egy betü, vagy szó és betü.

A.1. Alapjelek

A hivatkozási nyelv a következő alapjelekből épül fel:

⟨alapjel⟩ ::= ⟨betü⟩ | ⟨számjegy⟩ | ⟨logikai érték⟩ | ⟨elhatároló jel⟩

1.1. Betük

⟨betü⟩ ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Ehhez az ábécéhez hozzávehetünk még tetszőleges más jeleket vagy el is hagyhatunk belőle jeleket, feltéve, hogy (az első esetben) a hozzávett jelek nem azonosak, valamely más alapjellel (számjeggyel, logikai értékkel vagy elhatároló jellel).

A betüknek nincsen önálló jelentésük, ALGOL objektumok képzésére használjuk őket.

1.2 Számjegyek

⟨számjegy⟩ ::= 0|1|2|3|4|5|6|7|8|9|

A számjegyeket számok, azonositók képzésére használjuk.

1.3 Logikai értékek

⟨logikai érték⟩ ::= ⟨true⟩|⟨false⟩

1.4 Elhatároló jelek

⟨Elhatároló jel⟩ ::= ⟨müvelet jele⟩|⟨elválasztójel⟩|⟨záróje⟩|⟨deklarátor jel⟩|⟨specifikáló alapjel⟩

⟨müvelet jele⟩ ::= ⟨aritmetikai müvelet jele⟩|⟨reláció jele⟩|⟨logikai müvelet jele⟩|⟨vezérlőjel⟩

⟨aritmetikai müvelet jele⟩ ::= +|-|×|/|÷|↑

⟨reláció jele⟩ ::= <|≤|=|≥|>|≠

⟨logikai müvelet jele⟩ ::= ≡|⊃|∨|∧|¬

⟨vezérlőjel⟩ ::= go to|if|then|else|for|do

⟨elválasztójel⟩ ::= , | . | 10 | : | ; | :=|␣|step | until |
while | comment

⟨zárójel⟩ ::= (|)|[|]|‘|’| begin | end

⟨deklarátor jel⟩ ::= own | Boolean | integer | real |
array | switch | procedure

⟨specifikáló jel⟩ ::= string | label | value

1.5 Azonositók

⟨azonosító⟩ ::= ⟨betü⟩ | ⟨azonosító⟩ ⟨betü⟩ |
⟨azonosító⟩ ⟨számjegy⟩

1.6 Számok

⟨természetes szám⟩ ::= ⟨számjegy⟩ | ⟨természetes szám⟩
⟨számjegy⟩

⟨egész szám⟩ ::= ⟨természetes szám⟩ | +⟨természetes
szám⟩ | -⟨természetes szám⟩

⟨valódi tizedestört⟩ ::= .⟨természetes szám⟩

⟨kitevőrész⟩ ::= 10⟨egész szám⟩

⟨tizedesszám⟩ ::= ⟨természetes szám⟩ | ⟨valódi tizedes-
tört⟩ | ⟨természetes szám⟩⟨valódi ti-
zedestört⟩

⟨előjel nélküli szám⟩ ::= ⟨tizedesszám⟩ | ⟨kitevőrész⟩ |
⟨tizedesszám⟩⟨kitevőrész⟩

⟨szám⟩ ::= ⟨előjel nélküli szám⟩ | +⟨előjel nélküli szám⟩ |
-⟨előjelnélküli szám⟩

B. Kifejezések

⟨kifejezés⟩ ::= ⟨aritmetikai kifejezés⟩ | ⟨logikai kifejezés⟩ | ⟨helymegjelölő kifejezés⟩

1. Változók

⟨változónév⟩ ::= ⟨azonosító⟩

⟨skaláris változó⟩ ::= ⟨változónév⟩

⟨indexkifejezés⟩ ::= ⟨aritmetikai kifejezés⟩

⟨indexlista⟩ ::= ⟨indexkifejezés⟩ , ⟨indexlista⟩ | ⟨indexkifejezés⟩

⟨tömbnév⟩ ::= ⟨azonosító⟩

⟨indexes változó⟩ ::= ⟨tömbnév⟩ [⟨indexlista⟩]

⟨változó⟩ ::= ⟨skaláris változó⟩ | ⟨indexes változó⟩

2. Függvénykifejezések

⟨eljárásnév⟩ ::= ⟨azonosító⟩

⟨aktuális paraméter⟩ ::= ⟨idézet⟩ | ⟨kifejezés⟩ | ⟨tömbnév⟩ | ⟨kapcsolónév⟩ | ⟨eljárásnév⟩

⟨szó⟩ ::= ⟨betü⟩ | ⟨szó⟩ ⟨betü⟩

⟨paraméterelválasztó jel⟩ ::= , |) ⟨szó⟩ : (

⟨aktuális paraméter-lista⟩ ::= ⟨aktuális paraméter⟩ | ⟨aktuális paraméter⟩ ⟨paraméterelválasztó jel⟩ ⟨aktuális paraméter⟩

⟨aktuális paraméter-rész⟩ ::= ⟨üres⟩ | (⟨aktuális paraméterlista⟩)

⟨függvénykifejezés⟩ ::= ⟨eljárásnév⟩ ⟨aktuális paraméterrész⟩

3. Aritmetikai kifejezések

⟨additiv művelet jele⟩ ::= + | -

⟨multiplikativ művelet jele⟩ ::= X | / | +

⟨elsődleges kifejezés⟩ ::= ⟨előjel nélküli szám⟩ | ⟨változó⟩ | ⟨függvénykifejezés⟩ | (⟨aritmetikai kifejezés⟩)

⟨tényező⟩ ::= ⟨elsődleges kifejezés⟩ | ⟨tényező⟩↑⟨elsődleges kifejezés⟩

⟨tag⟩ ::= ⟨tényező⟩ | ⟨tag⟩ ⟨multiplikativ művelet jele⟩ ⟨tényező⟩

⟨feltétlen aritmetikai kifejezés⟩ ::= ⟨tag⟩ | ⟨additiv művelet jele⟩⟨tag⟩ | ⟨feltétlen aritmetikai kifejezés⟩ ⟨additiv művelet jele⟩ ⟨tag⟩

⟨feltétel-rész⟩ ::= if ⟨logikai kifejezés⟩ then

⟨Aritmetikai kifejezés⟩ ::= ⟨feltétlen aritmetikai kifejezés⟩ | ⟨feltétel-rész⟩ | ⟨feltétlen aritmetikai kifejezés⟩ else ⟨aritmetikai kifejezés⟩

4. Logikai kifejezések

⟨reláció jele⟩ ::= < | ≤ | = | ≥ | > | ≠

⟨reláció⟩ ::= ⟨aritmetikai kifejezés⟩ ⟨reláció jele⟩ ⟨aritmetikai kifejezés⟩

⟨elsődleges logikai kifejezés⟩ ::= ⟨logikai érték⟩ | ⟨változó⟩ | ⟨függvénykifejezés⟩ | ⟨reláció⟩ | (⟨logikai kifejezés⟩)

⟨logikai tényező⟩ ::= ⟨elsődleges logikai kifejezés⟩ | ¬⟨elsődleges logikai kifejezés⟩

⟨logikai tag⟩ ::= ⟨logikai tényező⟩ | ⟨logikai tag⟩ ∧ ⟨logikai tényező⟩

⟨diszjunkciós kifejezés⟩ ::= ⟨logikai tag⟩|
⟨diszjunkciós kifejezés⟩ ∨ ⟨logikai tag⟩

⟨implikációs kifejezés⟩ ::= ⟨diszjunkciós kifejezés⟩
⟨implikációs kifejezés⟩ ⊃ ⟨diszjunkciós kifejezés⟩

⟨feltétlen logikai kifejezés⟩ ::= ⟨implikációs kifejezés⟩|
⟨feltétlen logikai kifejezés⟩ ≡ ⟨implikációs kifejezés⟩

⟨logikai kifejezés⟩ :: = ⟨feltétlen logikai kifejezés⟩|
⟨feltétel-rész⟩ ⟨feltétlen logikai kifejezés⟩
else ⟨logikai kifejezés⟩

5. Helymegjelölő kifejezések

⟨cimke⟩ ::= ⟨azonositó⟩| ⟨természetes szám⟩

⟨kapcsolónév⟩ ::= ⟨azonositó⟩

⟨kapcsolókifejezés⟩ ::= ⟨kapcsolónév⟩[⟨indexkifejezés⟩]

⟨feltétlen helymegjelölő kifejezés⟩ ::= ⟨cimke⟩|
⟨kapcsolókifejezés⟩|(⟨helymegjelölő kifejezés⟩)

⟨helymegjelölő kifejezés⟩ ::= ⟨feltétlen helymegjelölő kifejezés⟩|⟨feltételrész⟩
⟨feltétlen helymegjelölő kifejezés⟩ else
⟨helymegjelölő kifejezés⟩

C. Utasitások

1. Összetett utasitások és blokkok

⟨cimkétlen alaputasitás⟩ ::= ⟨értékadó utasitás⟩
⟨átirányitó utasitás⟩|⟨üres utasitás⟩
⟨eljárásutasitás⟩

⟨alaputasitás⟩ ::= ⟨cimkétlen alaputasitás⟩|⟨cimke⟩ :
⟨alaputasitás⟩

⟨feltétlen utasitás⟩ ::= ⟨alaputasitás⟩|⟨ciklusutasitás⟩|
⟨összetett utasitás⟩ | ⟨blokk⟩

⟨utasitás⟩ ::= ⟨feltétlen utasitás⟩|⟨feltételes utasitás⟩

⟨összetett utasitás vége⟩::= ⟨utasitás⟩ end | ⟨utasitás⟩;
⟨összetett utasitás vége⟩

⟨blokkfej⟩ ::= begin ⟨deklaráció⟩|⟨blokkfej⟩;
⟨deklaráció⟩

⟨cimkétlen összetett utasitás⟩ ::= begin
⟨összetett utasitás vége⟩

⟨cimkétlen blokk⟩ ::= ⟨blokk fej⟩ ; ⟨összetett utasitás
vége⟩

⟨összetett utasitás⟩ ::= ⟨cimkétlen összetett utasitás⟩|
⟨cimke⟩ ; ⟨összetett utasitás⟩

⟨blokk⟩ ::= ⟨cimkétlen blokk⟩ | ⟨cimke⟩ : ⟨blokk⟩

2. Értékadó utasitások

⟨baloldal⟩ ::= ⟨változó⟩ :=

⟨baloldali változólista⟩ ::= ⟨baloldal⟩|
⟨baloldali változólista⟩ ⟨baloldal⟩

⟨értékadó utasitás⟩ ::= ⟨baloldali változólista⟩
⟨aritmetikai kifejezés⟩|⟨baloldali változólista⟩
⟨logikai kifejezés⟩

3. Átirányitó utasitások

⟨átirányitó utasitás⟩ ::= go to ⟨helymegjelölő kifejezés⟩

4. Feltételes utasitások

⟨feltétel-rész⟩ ::= if ⟨logikai kifejezés⟩ then

⟨feltétlen utasitás⟩ ::= ⟨alaputasitás⟩|⟨ciklusutasitás⟩|
⟨összetett utasitás⟩|⟨blokk⟩

⟨feltételesen végrehajtandó utasitás⟩ ::= ⟨feltétel-
rész⟩ ⟨feltétlen utasitás⟩|⟨cimke⟩ :
⟨feltételesen végrehajtandó utasitás⟩

⟨feltételes utasitás⟩ ::= ⟨feltételesen végrehajtandó
utasitás⟩|⟨feltételesen végrehajtandó utasi-
tás⟩ else ⟨utasitás⟩

5. Ciklusutasitások

⟨ciklusiista-elem⟩ ::= ⟨aritmetikai kifejezés⟩|
⟨aritmetikai kifejezés⟩ step ⟨aritmetikai ki-
fejezés⟩ until ⟨aritmetikai kifejezés⟩|
⟨aritmetikai kifejezés⟩ while ⟨logikai ki-
fejezés⟩

⟨ciklusiista⟩ ::= ⟨ciklusiista-elem⟩|⟨ciklusiista⟩,
⟨ciklusiista elem⟩

⟨cikluskezdet⟩ ::= for ⟨változó⟩ : =⟨ciklusiista⟩ do

⟨ciklusutasitás⟩ ::= ⟨cikluskezdet⟩⟨utasitás⟩|⟨cimke⟩ :
⟨ciklusutasitás⟩

6. Eljárásutasitások

⟨aktuális paraméter⟩ ::=⟨idézet⟩|⟨kifejezés⟩|⟨tömbnév⟩|
⟨kapcsolónév⟩|⟨eljárásnév⟩

⟨szó⟩ ::= ⟨betü⟩|⟨szó⟩⟨betü⟩

⟨paraméterelválasztó jel⟩ ::= ,|)⟨szó⟩:(

⟨aktuális paraméter-lista⟩ ::= ⟨aktuális paraméter⟩|

⟨akutális paraméterlista⟩ ⟨paraméterelválasztó jel⟩
⟨aktuális paraméter⟩

⟨aktuális paraméter-rész⟩ ::= ⟨üres⟩ |
(⟨aktuális paraméter-lista⟩)

⟨eljárásutasitás⟩ ::= ⟨eljárásnév⟩
⟨aktuális paraméter-rész⟩

D. Deklarációk

⟨deklaráció⟩ ::= ⟨tipusdeklaráció⟩ | ⟨tömbdeklaráció⟩ |
⟨kapcsolódeklaráció⟩ | ⟨eljárásdeklaráció⟩

1. Tipusdeklaráció

⟨tipuslista⟩ ::= ⟨sklarális változó⟩ | ⟨skaláris változó⟩,
⟨tipuslista⟩

⟨tipus⟩ ::= real | integer | Boolean

⟨tipusjelzés⟩ ::= ⟨tipus⟩ | own ⟨tipus⟩

⟨tipusdeklaráció⟩ ::= ⟨tipusjelzés⟩ ⟨tipuslista⟩

2. Tömbdeklarációk

⟨alsó korlát⟩ ::= ⟨aritmetikai kifejezés⟩

⟨felső korlát⟩ ::= ⟨aritmetikai kifejezés⟩

⟨korlátpár⟩ ::= ⟨alsókorlát⟩ : ⟨felső korlát⟩

⟨korlátpár-lista⟩ ::= ⟨korlátpár⟩ | ⟨korlátpár-lista⟩,
⟨korlátpár⟩

⟨tömbszelet⟩ ::= ⟨tömbnév⟩ [⟨korlátpár-lista⟩] |
⟨tömbnév⟩ , ⟨tömbszelet⟩

⟨tömblista⟩ ::= ⟨tömbszelet⟩ | ⟨tömblista⟩,
⟨tömbszelet⟩

⟨tömbdeklaráció⟩ ::= array ⟨tömblista⟩ | ⟨tipusjelzés⟩
array ⟨tömblista⟩

3. Kapcsolódeklarációk

⟨kapcsolólista⟩ ::= ⟨helymegjelölő kifejezés⟩|
⟨kapcsolólista⟩ , ⟨helymegjelölő kifejezés⟩

⟨kapcsolódeklaráció⟩ ::= switch ⟨kapcsolónév⟩ :=
⟨kapcsolólista⟩

4. Eljárásdeklarációk

⟨formális paraméter⟩ ::= ⟨azonositó⟩

⟨formális paraméter-lista⟩ ::= ⟨formális paraméter⟩|
⟨formális paraméterlista⟩ ⟨paraméterelválasztó
jel⟩ ⟨formális paraméter⟩

⟨formális paraméter-rész⟩ ::= ⟨üres⟩|
(⟨formális paraméter-lista⟩)

⟨azonositólista⟩ ::= ⟨azonositó⟩ | ⟨azonositólista⟩ ,
⟨azonositó⟩

⟨értékrész⟩ ::= value ⟨azonositólista⟩ ;| ⟨üres⟩

⟨specifikátor⟩ ::= string | ⟨tipus⟩ | array | ⟨tipus⟩
array | label | switch | procedure | ⟨tipus⟩
procedure

⟨specifikációs rész⟩ ::= ⟨üres⟩ | ⟨specifikátor⟩
⟨azonositólista⟩ ; | ⟨specifikációs rész⟩
⟨specifikátor⟩ ⟨azonositólista⟩ ;

⟨eljárás feje⟩ ::= ⟨eljárásnév⟩ ⟨formális paraméter-rész⟩;
⟨érték-rész⟩ ⟨specifikációs rész⟩

⟨eljárás törzs⟩ ::= ⟨utasitás⟩|⟨kód⟩

⟨eljárásdeklaráció⟩ ::= procedure ⟨eljárás feje⟩
⟨eljárástörzs⟩|⟨tipus⟩ procedure ⟨eljárás feje⟩
⟨eljárástörzs⟩

TARTALOMJEGYZÉK

I. f e j e z e t

II. f e j e z e t

III. f e j e z e t