

Halmá

Kézirat.

MAGYAR TUDOMÁNYOS AKADÉMIA
KIBERNETIKAI KUTATÓ CSOPORTJA

Az M-3 elektronikus számológép programozása.

Az MTA Kibernetikai Kutató Csoportja
1958-59 évben tartott előadássorozatának
anyaga.

Budapest
1959. november

Szerkesztették: Dömölki Bálint, Frey Tamás (felelős szerkesztő), Gergely József, Lőcs Gyula, Révész Pálné, Sándor Ferenc, Szelezsán János, Veidinger László.

T A R T A L O M J E G Y Z É K

| | |
|---|-------|
| 1. Fejezet. | oldal |
| BEVEZETÉS | 1 |
| (Írta: Sándor Ferenc) | |
| 1.1. Az M-3 gép jellemzői | 1 |
| 1.2. A kettes számrendszerőről | 8 |
| 1.3. A program szerepe..... | 10 |
| 1.4. A számológépek rövid történeti áttekintése | 13 |
| 1.5. Egy adott számítási probléma gépi megoldását elősegítő munkakülönböző lépései | 15 |
| 1.6. A számológépek alkalmazásai | 18 |
| | |
| 2. Fejezet. | |
| AZ M-3 ORGANIZÁCIÓJÁNAK ÉS UTASÍTÁSRENDSZERÉNEK RÉSZ- LETES LEÍRÁSA. | 21 |
| (Írta: Dömölki Bálint) | |
| 2.1. A gép fő egységei | 21 |
| 2.2. A műveletek végrehajtásának módja | 25 |
| 2.2.1. Összeadás | 27 |
| 2.2.2. Kivonás | 29 |
| 2.2.3. Szorzás | 32 |
| 2.2.4. Osztás | 36 |
| 2.2.5. Logikai szorzás | 40 |
| 2.3. Az adatok be- és kivitele | 41 |
| 2.4. A vezérlőegység és működése | 48 |
| 2.5. Vezérlés - átadó utasítások | 57 |
| 2.6. Az utasítások végrehajtásának időtartama | 61 |

| | |
|---|-------|
| 3. Fejezet. | oldal |
| A PROGRAMKÉSZÍTÉS GYAKORLATI MÓDSZEREI. (Írta: Révész Pálné) | 62 |
| 3.1. A direkt programozás bemutatása példákön. | 62 |
| 3.2. Direkt programozás feltételes ugró utasítások- kal. Példák..... | 70 |
| 3.3. Egyszerű programok, a ciklusutasítások válto- zatlanok..... | 77 |
| 4. Fejezet. | |
| CÍM ÉS MŰVELETI UTASÍTÁSOK MENET KÖZBENI MÓDOSÍTÁSA. (Írta: Sándor Ferenc) | 89 |
| 4.1. A címutasítások módosítása | 90 |
| 4.2. Műveleti utasítások módosítása | 95 |
| 5. Fejezet. | |
| CIKLIKUS PROGRAMOK VÁLTOZÓ UTASÍTÁSOKKAL. (Írta: Sándor Ferenc) | 106 |
| 5.1. Utasítások visszaállítása beültetéssel | 106 |
| 5.2. A program visszaállítása utókorrekcióval | 109 |
| 5.3. Gyakorlati példák | 111 |
| 5.4. Többszörös ciklusok | 119 |
| 6. Fejezet. | |
| A SZÁMOK FIXPONTOS ÁBRÁZOLÁSÁBÓL ADÓDÓ NUMERIKUS PROBLÉMÁK. (Írta: Szelecsán János) | 119 |
| 6.1. Fixpontos ábrázolásról | 119 |
| 6.2. Transzformációs módszerek | 121 |
| 6.2.1. Skálafaktor módszer | 122 |
| 6.3. A bináris pont középre helyezésének módszere... | 127 |
| 6.4. Egyéb módszerek | 130 |
| 6.4.1. Reciprok érték felhasználása | 130 |
| 6.4.2. Automatikus normálás módszere | 130 |

| | |
|--|-------|
| 7. Fejezet. | oldal |
| SZUBRUTINOK ALKALMAZÁSA. | 136 |
| (Írták: Révész Pálné és Veidinger László) | |
| 7.1. Nyílt és zárt szubrutinok | 136 |
| 7.2. Zárt szubrutinok behívása, visszaugrás a főprogramra | 144 |
| 7.3. Programparaméterek beültetése szubrutinokba ... | 151 |
| 7.4. Szabványos szubrutinok relatív címezéssel | 154 |
| 7.5. További példák | 157 |
| 7.5.1. Elemi függvények szubrutinjai | 157 |
| 7.5.2. Szubrutin $\int_a^b f(x) dx$ kvadraturájához | 172 |
| 8. Fejezet. | |
| AZ INPUT ÉS OUTPUT SZUBRUTINJA. | 176 |
| (Írta: Veidinger László) | |
| 8.1. Input decimálisan adott számok esetén | 176 |
| 8.2. Output decimális nyomtatáshoz | 179 |
| 8.3. Relatív címezésű szubrutinok bevitele | 183 |
| 9. Fejezet. | |
| ÉRTELMEZŐ ÉS KONVERZIÓS SZUBRUTINOK LOGIKAI (TECHNIKAI) LEHETŐSÉGEK PÓTLÁSÁRA | 189 |
| (Írta: Szelezsán János) | |
| 9.1. Lebegőpontos számolás az M-3 gépen | 190 |
| 9.2. A komplex aritmetika programozása | 197 |
| 9.3. Értelmező szubrutin - lebegőpontos programhoz.. | 201 |
| 9.4. Értelmező szubrutin - a komplex aritmetikához.. | 208 |
| 9.5. Konverziós szubrutinok | 210 |

| | |
|--|-------|
| 10. Fejezet. | Oldal |
| GYAKORLATI PÉLDÁK. | 220 |
| (Írták: Veidinger László és Szelezsán János) | |
| 10.1. Lineáris egyenletrendszerek | 220 |
| 10.2. Differenciálegyenletek numerikus integrálása | 227 |
| 10.2.1. A Runge-Kutte módszer elsőrendű egyen- leteknél | 227 |
| 10.2.2. A differenciamódszerek | 254 |
| 10.2.3. Elsőrendű közönséges differenciál- egyenletrendszerek | 240 |
| 10.3. Parciális differenciálegyenletek | 240 |

F. FÜGGELÉK.

(Írták: Dömölki Bálint és Szelezsán János)

| | |
|--|-----|
| F.1. Az M-3 gép kezelése | 241 |
| F.1.1. Az adatok lyukasztása és bevitele | 242 |
| F.1.2. Számok beállítása a C, SZR, PR regiszterek- be | 244 |
| F.1.3. A program végrehajtása | 244 |
| F.1.4. A program lépésenkénti végrehajtása | 245 |
| F.1.5. Beírás a memóriába | 245 |
| F.1.6. Kihívás a memóriából | 246 |
| F.1.7. Megállás egy adott címnél | 246 |
| F.2. Az M-3 utasításrendszere | 248 |

1. Fejezet.

B E V E Z E T É S

Ebben a munkában az M-3 jelű szovjet mintájú számológép programozásával foglalkozunk. Az M-3 gép az univerzális, automatikus vezérlésű, tárolt programú, digitális, elektronikus számológépek családjába tartozik.

1.1. Az M-3 jellemzői.

Vegyük sorra a géptípus egyes felsorolt tulajdonságait.

A fenti géptípus egyik legfőbb jellemzője, hogy digitális. Ez azt jelenti, hogy a számításokban résztvevő mennyiségeket a gép részére számjegyekből álló számok formájában adják meg és a gép a számjegyeken végzett műveletek által oldja meg a teljes számítási feladatot. Ebben térnek el a digitális számológépek az u.n. analóg gépektől, amelyek egy bizonyos adott számítási feladat fizikai analogonját valósítják meg és a kiinduló adatok, valamint a végeredmények is fizikai mennyiségek (pl. szögelfordulás, áramerősség, feszültség, stb.) formájában jelentkeznek.

Az elektronikus jelző megkülönbözteti a fenti típus gépeit a jól ismert mechanikus és elektromechanikus gépektől, amelyek mozgó alkatrészekkel végzik el a számításokat. Az elektronikus számológépekben a számítások elvégzésére, egyes kivételektől eltekintve, nincsenek mozgó alkatrészek (de természetesen a kiinduló adatok bevitele és az eredmények kiadása mechanikus, mozgó berendezések segítségével történik), az aktív elemek elektroncsövek, kristálydiódák, újabban csövek helyett tömör elemek: mágneses magok, tranzistorok.

Univerzális azt jelenti, hogy a gép elvileg bármilyen (akármilyen összetett) számítási feladat elvégzésére alkalmas. Ebben az értelemben univerzálisak a mechanikus vagy elektromechanikus asztali számológépek is, amelyeken egyetlen lépésben csak az aritmetikai alapműveletek (sokszor csak összeadás, kivonás és a számok helyértékeltołása) végezhető el. Ismeretes azonban, hogy összetett számítási feladatok (pl. algebrai-, transzcendens-, differenciálegyenletek stb.) megoldása u.n. numerikus módszerek segítségével visszavezethető aritmetikai alapműveletek meghatározott sorozatának az elvégzésére. Ilyen numerikus módszereket kidolgoztak a feladatok legáltalánosabb válfajaira. Asztali számológépek segítségével tehát elvileg megoldható minden számítási feladat, amely számára numerikus módszer létezik, azaz amelyik aritmetizálható.

Ugyanilyen univerzálisak az automatikus vezérlésű (vagy: programvezérelt) számológépek is. Ezek u.n. "aritmetikai egysége" szintén csak az aritmetikai alapműveleteket (s esetleg néhány elemi logikai műveletet) képes elvégezni, de ezeknek a műveleteknek bizonyos egymásutánjával minden számítási feladat megoldható, amelyhez numerikus módszer tartozik. Az automatikus vezérlésű gépek éppen abban térnek el a közönséges asztali számológépektől, hogy míg utóbbiak esetében minden egyes aritmetikai műveletet a kezelőnek kell beállítania, előbbieknél a feladat alkalmas formában való megadása után önműködően végzik el az elemi számítások szükséges sorozatát egészen az eredménynek kívánt formában való kiadásáig és üzem közben nem igényelnek emberi beavatkozást. A feladatnak a gép számára alkalmas formában való megadása a feladat gépi programozását jelenti, amely jelen előadás-sorozatunk tulajdonképpeni tárgya. Ez röviden abban áll, hogy az automatikus vezérlésű számológép részére a számítási feladat kiinduló számadatain kívül szükséges előre megadni az u.n. utasításokat is, amelyek tartalmazzák, hogy milyen műveleteket kell elvégeznie a gépnek a megadott számokon, illetve közbenső eredményeken az adott számítási feladat (meghatározott numerikus módszer szerinti) megoldása céljából.

Az utasításokat a számadatokhoz hasonló számjegy-kombinációk alakjában adjak meg a gépnek. A gép belsejében tárolja az utasításokat, csakúgy, mint a kiinduló mennyiségeket, és azokat a közbenső eredményeket, amelyekre a számítás további során szükség van.

Egy adott speciális probléma gépi megoldására szolgáló utasítások összességét programnak nevezik. Programozás pedig egy bizonyos program megszerkesztését, megírását jelenti, azzal a szükséges előkészítő munkával együtt, amely megelőzi az utasítások részletes felírását. A fenti jellemzésben tárolt programú azt jelzi, hogy a számológép a számítási programot (az azt alkotó utasításokat) a számadatokkal azonos módon tárolja.

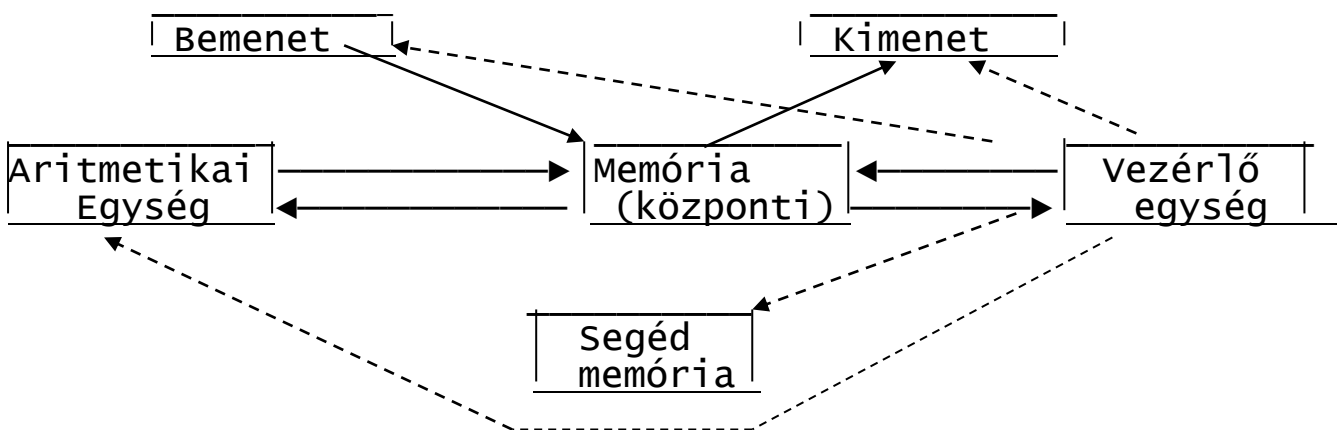
A jellemzett típusú gépek, amelyek közé az M-3 is tartozik, a modern számológép-technika legfejlettebb, legnagyobb teljesítményű gépei. Az elektronikus áramkörök és az automatikus vezérlés olyan működési sebességet tesz lehetővé, amely által ezek a gépek rendkívül összetett numerikus feladatokat a gyakorlat számára reális idő alatt tudnak megoldani.

A programozás maga úgy tekinthető, mint az emberi (matematikai) nyelvről az adott gép "nyelvére" való fordítási művelet. A numerikus módszer alapján megadott számítási feladatban matematikai jelölések és a közönséges nyelvben megfogalmazott döntések vannak. A számológép "nyelve" viszont egyszerű aritmetikai, logikai és választási műveletekből áll, melyeket bizonyos numerikus kódok ("kulcsok", számjegy-kombinációk) képviselnek. Ezek szerint a kódok szerint kell megadni az utasításokat, amelyek alapján a gép önműködően végigszámítja az adott numerikus eljárást.

Tulajdonképpen a programozásról csak univerzális, digitális és természetesen automatikus vezérlésű számológépeknél lehet szó. Analóg gépek vagy olyan digitális célgépek esetén, melyek csak egy speciális feladat (pl. adott algebrai vagy

differentiálegyenlet) megoldására szolgálnak, nincs szükség programozásra, ezeknek fix programjuk van.

Mielőtt magára az M-3 gépre és annak programozására rátérnénk, meg kell ismerkednünk a modern digitális számológépek általános (funkcionális) alapelveivel. Az 1. ábrán látható blokk-diagram mutatja be ezeknek a számológépeknek az elvi felépítését. A számológép öt főegységből áll, annak az öt fő funkciónak megfelelően, amelyet egy számítási feladat automatikus megoldásához el kell végezni. Az egységek a következők: bemenet, memória, aritmetikai egység, vezérlő egység, kimenet. Az ábrán egymáshoz való kapcsolatuk is fel van tüntetve, a folytonos vonalak az információk (szám adatok, utasítások vagy a gép által tárolható egyéb adatok), a szaggatott vonalak a vezérlő-jelek útján jelzik. Meg kell jegyezni, hogy ezek az elvi egységek fizikai megvalósításukban (mechanikus és elektronikus szerelvények) is általában egymástól elkülönítetten vannak felépítve és kábelekkel összekapcsolva, de ez a külön keretekre való tagozódás nem feltétlenül követi a blokk-diagram elvi tagozódását.



1. ábra

A gép bemeneti egysége olyan szerelvényekből áll, amelyek lyukkártyákról, lyukszalagról vagy mágneses szalagról felveszik az információt és elhelyezik azt a memóriában. Ezt a műveletet hívják olvasásnak. A bemeneti berendezések feladata tulajdonképpen átültetni az információt külső megjelenési formájáról, ahogyan pl. egy lyukkártyán szerepel, arra a formára, amelyben a memória tárolja. Az 1. ábrán látható nyíl, amely a bemenetet a memóriával összekapcsolja, azt jelzi, hogy az információ a bemenettől csak a memóriába mehet, és további műveleteknek kell azt a memóriából a gép más részeibe elvinni. Bár a továbbiakban is óvatosnak kell lennünk az analógiákkal, mert azok többnyire félrevezetőek, mégis összehasonlíthatjuk az automatikus számológépek billentyűzetével. Ez az analógia azért nem tökéletes, mert az asztali számológépeknek nincs igazi belső memóriájuk. A szükséges adatokat a kezelő emlékezetében vagy papíron tárolja.

Az automatikus vezérlésű számológépek a számokat, utasításokat és a számítás egyéb szükséges adatait a memóriában tárolják. Ezen minden információ keresztülmegy. Aritmetikai műveletek csak olyan számokon végezhetők el, amelyek benn vannak a memóriában és minden utasításnak, amely a gép működését irányítja, előbb benn kell lennie a memóriában, mielőtt a vezérlő egységbe kerülhet és hatását kifejtheti. A memóriával szembeni követelmény: legyen nagy és gyors. Ez azt jelenti, hogy sok (jelenlegi gépeknél 1000-30.000) szám, utasítás, vagy egyéb adat (közös néven: szó) tárolására legyen képes, és ezeket minimális késéssel továbbítsa a gép aritmetikai és vezérlő egységének. Azt az időt, amely valamilyen adatnak - szónak - a memóriába való bejegyzéséhez, (beírásához), illetve az onnét való kivételéhez (kiolvasásához) szükséges, a memória elérési (hozzáférési) idejének nevezik. Ez az idő a jelenlegi leggyorsabb számológépeknél 10 μ sec körül van. A tárolt programú gépekben általában nincs megkülönböztető jel arra, hogy egy szó számadatot,

utasítást vagy egyebet jelent-e. Ez nagyon fontos a programozás szempontjából, amint azt a későbbiekben látni fogjuk.

Műszaki és gazdaságossági okokból nem célszerű a nagysebességű memóriát olyan nagy kapacitásúvá építeni, hogy a számításához szükséges összes adat beleférjen. A megoldás az, hogy a memóriát két vagy több lépcsőben építik ki, és az információnak azt a részét, amelyre a számítás későbbi fázisában lesz szükség, egy nagykapacitású, de lassúbb segédmemóriában helyezik el. Amint az 1. ábrából látható, a segédmemória csak a központi memóriával van összeköttetésben, tehát az adatokat az utóbbiba kell hozni, mielőtt részt vehetnek a számításban.

Mint említettük: a számításban közvetlenül a központi memória vesz részt. Ez rekeszekre oszlik, amelyek mindegyike egy szó tárolására alkalmas. Annyi rekesz van, ahány szó tárolására képes a memória, azaz amekkora a memória kapacitása. A programkészítés céljára szükség van a memória rekeszeinek valamilyen azonosítására. A gépi utasításokban ugyanis meg kell adni, hogy egy bizonyos művelethez honnét kell venni a számadatokat, hová kell elhelyezni az eredményt és hol található a következő végrehajtandó utasítás. A megoldás az, hogy a memória rekeszeihez azonosítási számokat rendelnek hozzá, 0-tól $(n-1)$ -ig, ahol n a rekeszek száma. Ez a sorszám a rekesznek, illetve tartalmának címe. Közelfekvő analógia adódik a számológépek memóriája és a póstafiókok között. A póstafiókoknak azonosítási célokból névtábláik vannak, amelyek azonban semmit sem mondanak a fiók tartalmáról. A névtáblák funkciója: ha egy bizonyos levelet elhelyeznek az x feliratú fiókban, akkor ugyanez a levél a későbbiek folyamán megtalálható lesz, ha az x feliratú fiókban keresik. Ugyanez egy memória-rekesz címének a funkciója. Ha egy bizonyos számot elhelyezünk pl. az 1507-es rekeszben, később visszatérve az 1507-es rekeszhez, kiolvashatjuk belőle a belehelyezett

számot. Az 1507-es cím semmiképpen sem jelenti azt, hogy ebben a rekeszben az 1507-es szám van tárolva, kivéve a véletlen egyezés esetét. Bár mindez rendkívül kézenfekvőnek látszik, a tapasztalat szerint bizonyos nehézséget szokott jelenteni a kezdők számára.

A memóriának két fontos tulajdonsága van, amely nem vág a póstafiók hasonlathat. Először, egy memóriarekesz egyszerre csak egy szó tárolására képes és egy új szónak a beírása véglegesen kitörli a régi tartalmat. Ez egyrészt előnyös, mert egy szó tárolásakor nem kell törődni a rekesz előzetes kiürítésével, másrészt azonban a programban ügyelni kell arra, hogy a tároláskor ne vesszen el szükséges információ. Másodszor: a memóriából való kiolvasás változatlanul benne hagyja a kiolvasott szót a memóriában, mintegy "másolat" készül róla a gép más egységei számára.

A számológép aritmetikai egysége a gép tulajdonképpeni számoló szerve. Ez végzi el a négy alapműveletet (egyeknél csak hármat, osztás nélkül), ezenkívül általában képes számok helyérték eltolására és egyszerű logikai műveletek elvégzésére. Az aritmetikai egység megfelel az asztali számológépek számológerekeinek és tengelyeinek, amelyek a tulajdonképpeni számolást végzik.

A számológép vezérlő egysége a memóriában tárolt utasítások (számjegy-kombinációk) szerint vezérlőjeleket küld a gép többi egységeinek a megfelelő utasítás végrehajtása céljából. Az ábrán két folytonos vonal jelzi, hogy az utasítások oda-vissza közlekedhetnek a memória és a vezérlő egység között. A szaggatott vonalak a vezérlőjelek útját jelzik. A vezérlőegység összehasonlítható az asztali számológépek kiváltó billentyűivel, amelyeket a különböző aritmetikai műveletek elvégzése céljából kell lenyomni, bár ez a hasonlat eléggé tökéletlen.

A gép aritmetikai és vezérlő egységében fontos szerepük van az u.n. regisztereknek. Ezek egy-két szó, vagy szónál kisebb számjegycsoport ideiglenes tárolására szolgáló berendezések. Addig tárolják az információt, ameddig arra az utasítás megfelelő szakaszának végrehajtása során szükség van. Regiszterekben történik a számok helyértékeltoítása és egyszerű műveletek (pl. összeadás) elvégzése is. Az automatikus számológépek regisztereinek hasonló szerepük van, mint az asztali számológépek regisztereinek (számtárcsáinak).

A számológép kimeneti egysége a számítási feladatok megoldását és ezenfelül esetleg a memória egyéb tartalmát adja ki megfelelő formában. Az eredmények és egyéb adatok kiadása lyukkártyákon, lyukszalagon, mágnesszalagon, nyomtatott íveken, stb. történhet. Az 1. ábra szerint az információ csak a memóriából kerülhet a kimenetre. Ez alapján így van, bár a legtöbb gépnél az aritmetikai egység is résztvesz az adatok közvetítésében.

A bemeneti és kimeneti berendezések általában a gép többi részétől elkülönített kezelőasztalon vagy kereten vannak. Rendszerint külön-külön kerete van a központi memóriának és a segédmemóriának is. Az aritmetikai és a vezérlőegység szokványos elektronikus szerelvényekből áll és ezek rendszerint közös kereten vannak. Fentiekén kívül a számológépnek tápegysége is van, külön kereten, de erre, ebben a funkcionális ismertetésben nem térünk ki bővebben.

1.2. A kettes számrendszerről

A számológépen belül az adatok (szavak) feszültségimpulzus sorozatok formájában közlekednek. Minthogy műszakilag csak az impulzus jelenlétét vagy hiányát célszerű megkülönböztetni és az adatok tárolása olyan elemekkel történik, amelyeknek két lehetséges állapotuk van, ebből adódik

a bináris (kettes) számrendszer alkalmazásának előnye a számológépen belül.

Ismeretes, hogy bármely szám kifejezhető bármely számrendszerben, amelynek alapszáma egynél nagyobb természetes szám.

$$X = a_m r^m + a_{m-1} r^{m-1} + \dots + a_0 + a_{-1} r^{-1} + \dots + a_{-n} r^{-n} + \dots = \\ = (a_m a_{m-1} \dots a_0, a_{-1} \dots a_{-n} \dots)_r$$

ahol x az ábrázolandó szám,

r a számrendszer alapszáma,

az indexszel ellátott a -k x számjegyei az r alapú számrendszerben való előállításban (r -nél kisebb természetes számok, vagy 0-ok)

m a "tizedes" vesszőtől balra,

n a "tizedes" vesszőtől jobbra lévő számjegyek száma.

Így történik a számok jól ismert ábrázolása a decimális (tízes) számrendszerben (persze a szokványos ábrázolásnál nem kell megjelölni a számrendszer alapszámát). Számoknak bináris (kettes) számrendszerben történő ábrázolásánál fentiek szerint csak kétfajta számjegy szükséges: 1 és 0. Ezért alkalmas ez a számrendszer a digitális számológépben való felhasználásra: pl. egy impulzus jelenléte az 1-es, hiánya a 0 számjegyet jelenti. Ennek ellenére vannak decimális rendszer szerint működő gépek is, de ezekben minden decimális számjegyet bináris számjegyeknek (biteknek) egy csoportja képvisel.

1.3. A program szerepe.

Az utasítás és a program fogalmának jobb megvilágítása céljából ismét visszanyúlunk az asztali számológépek példájához. Egy összetett számítási feladatnak asztali gépen való megoldásához az adott numerikus módszer alapján részletes rutint kell kidolgozni a kezelő számára. Ez a rutin sorrendben tartalmazza, hogy mely mennyiségeken milyen műveleteket kell elvégezni. A számítás elemi lépések egymásutánjából tevődik össze, melyek mindegyike egy aritmetikai műveletet és általában új számadatot is tartalmaz. - Az automatikus számológépeknél a helyzet nagyjából hasonló. A megoldandó feladatot olyan lépések sorozatára kell felbontani, amelyek mindegyike valamilyen aritmetikai vagy a számítás szervezésével kapcsolatos műveletet és - a géptípustól függően - egy, két vagy több adatra való hivatkozást tartalmaz. E tekintetben a legfőbb különbség az asztali és az automatikus számológépek között a következő. Az asztali gépek esetében sok minden van rábízva a kezelőre, a számítás bizonyos pontjainál neki kell megválasztania a továbbiakban követendő lépések sorozatát. Ezzel szemben az automatikus számológépeknél a végrehajtandó lépéseket szigorúan definiálva kell megadni, miközben az adott probléma gépi számítása során számbajöhető valamennyi eshetőséget tekintetbe kell venni, és előre gondoskodni kell a gép által követendő programról bármelyik lehetőség bekövetkezése esetére.

Mint már említettük, az utasítások numerikus kód alakjában, a számadatokkal azonos formában szerepelnek a gép memóriájában. Az utasítások szerkezete különböző gépeknél különböző. Általában két fő része van az utasításoknak: a műveleti és a címrész. A műveleti rész jelzi a gépnek, milyen műveletet kell elvégeznie, a címrész pedig a memória azon rekeszeit jelöli meg, amelyek tartalmán kell a jelzett műveletet elvégezni. Aszerint, hogy egy adott gép utasításai-

nak címrésze a memória hány rekeszére hivatkozik, beszélünk egy, két, stb. című gépekről. Bár az utasítások műveleti része (műveleti jele) a memóriában mint numerikus kód szerepel, az utasítások leírásánál ehelyett bizonyos szimbólumokat (betűrövidítéseket vagy egyebet) használnak.

Minden gépnek van utasításrendszere, amely az illető gép által egy utasításban elvégezhető műveletek és a hozzá tartozó műveleti jelek (szimbólumok és kódok) listája. Egy adott gép számára a számítási programot az utasításrendszerében szereplő utasításokból kell összeállítani. Ez egyben mindig lehetséges is.

A számológépek többsége (ezek közé tartozik az M-3 is) a tárolási sorrendjében hajtja végre az utasításokat, és ezért egy program utasításai általában a memória egymást követő rekeszeiben vannak elhelyezve. Az utasítások végrehajtásának ez a sorrendje csak akkor változik meg, ha a program egy u.n. ugró-utasításhoz érkezik, amelyiknek (egyik) címe megadja a következő végrehajtandó utasítás helyét a memóriában. Az ugrás végrehajtása után a program ismét a tárolás sorrendjében követi az utasításokat, az új címtől kiindulva a következő ugró-utasításig. Az ugró-utasítások működését a vezérlésátadásának nevezik. Vannak feltétlen és feltételes ugró-utasítások. Előbbiek minden esetben, utóbbiak csak bizonyos feltételek teljesülése esetén (pl. adott regiszter tartalmának előjelétől függően) adják át a vezérlést, illetőleg a feltétel teljesülésétől függően adják azt át egyik vagy másik címre. A feltételes ugró-utasítások teszik lehetővé a különböző számítási utak (programváltozatok) közötti választást (ezek az u.n. elágazások) az addigi számítás folyamán kialakult kritériumok alapján, és ezek teszik lehetővé az u.n. ciklusok programozását is. Jelentőségükkel a későbbiek folyamán meg bőven fogunk foglalkozni.

Vannak u.n. megállító utasítások. Ha a program megállító utasításhoz érkezik, a gép megáll, azaz beszünteti az utasítások egymásutáni automatikus végrehajtását. Ezután a gép csak a kezelő beavatkozása esetén működik újra. A megállító utasítások általában úgy vannak elhelyezve a programban, hogy a gép a következő esetekben álljon meg:

1. Végetért a számítás, és az eredmények a kívánt formában vannak.
2. A számítás nem végezhető el (pl. negatív számból kell négyzetgyököt vonni, és a további számításhoz valós gyök szükséges).
3. A gép a számítás során hibát követ el.

A gép bizonyos esetekben megállító utasítások nélkül is megáll. Ezekről az esetekről az M-3 részletes tárgyalásánál lesz szó.

Egy utasítás végrehajtása nagyjából a következő lépésekben történik: 1. az utasítás a memóriából a vezérlő egységbe kerül. 2. A vezérlő egység a műveleti rész számjegykombinációjának (kódjának) megfelelő vezérlőjelet küld az aritmetikai egységnek (dekódolja, "értelmezi" az utasítás műveleti részét). 3. A címrész alapján a vezérlőegység jelzést ad a memóriának, amelyből a megfelelő címen (címeken) tárolt szavak az aritmetikai egységbe kerülnek. 4. Az aritmetikai egység a kapott számadatokkal végrehajtja a megfelelő műveletet. 5. Az aritmetikai egységben ideiglenesen tárolt eredmény a memóriának az utasításban megadott címére kerül. - Az utasítások végrehajtásának valóságos menete gépenként kisebb-nagyobb mértékben eltér a fenti sémától, előfordul, hogy egyes utasításon csak a fenti lépések egy részét tartalmazzák.

1.4. A számológépek rövid történeti áttekintése.

A jelenlegi digitális számológépek fentiekben vázolt jellemzői egy fejlődési folyamat eredményei. Hasznos, ha azok, akik számológép programozásával kívánnak foglalkozni, megismerkednek ennek a fejlődésnek főbb állomásaival.

Az első nagyobb számológépet egy angol matematikus, Charles Babbage kezdte építeni 1812-ben. Ez az u.n. differencia-gép (Difference Engine) természetesen teljesen mechanikai elvek szerint működött, és függvénytáblázatoknak differencia módszerrel való kiszámítására szolgált. 1833-ban Babbage egy második gépet tervezett, az u.n. analitikus gépet (Analytical Engine). Ez a gép a modern számológépek őseinek tekinthető, univerzális gép. A terv szerint a Jacquard-féle szövőszéknél használt lyukkártyák által lehetett megadni számára az elvégzendő műveletek sorozatát, a számokat mechanikus számkerekek tárolták, és az összes műveletet automatikusan hajtotta végre. Teljesen mechanikus elvek szerint működött volna, de egészében soha sem épült fel. Ennek oka részben az anyagi fedezet hiánya, főleg azonban az volt, hogy tisztán mechanikai úton nem lehetett megvalósítani az univerzális automatikus számológép funkcióit. Fejlett elektronikára és impulzustechnikára volt szükség a modern számológépek megvalósításához.

Az első modern számológépek, amely Babbage alapelveit követte, a 30-as években kezdték építeni a Harvard egyetemen Aiken vezetésével. Ez a Mark I. 1944-ben készült el, elektromágneses jelfogókat használt tisztán mechanikus elemek helyett, a számítási programot pedig lyukszalagon tárolta.

Az ENIAC (Electronic Numerical Integrator and Computer) jelentős fejlődést képviselt a számológép-technikában, mert

ez az első olyan számológép, amely belső működésében (tehát az adatok bevitelétől és kiadásától eltekintve) csak elektronikus elemeket használ. J.P. Eckert és J.W. Mauchly építették a pennsylvaniai egyetemen és 1946-ban készült el. A gép vezérlése (a műveletek sorozatának automatikus elvégzése) hüvelymezőbe dugaszolt vezetékek kombinációjával történt. Ez a gép csak a feladatok bizonyos körének elvégzésére volt alkalmas.

Mindezek a számológépek és mások is, amelyek abban az időben épültek, bizonyos külső eszközöket használtak a programvezérlésre: lyukkártyákat, lyukszalagot, dugaszolt vezetékeket. A memóriát csak a számok tárolására használták. 1945-ben Neumann János, a híres magyarszármazású matematikus, egy jelentésben megfogalmazta a tárolt programú gépek elvét, azt az elvet, hogy az utasításokat is a gép memóriájában tárolják. Ez az elgondolás azóta a modern számológépek alapelve. Ennek alapján az ENIAC-tól teljesen eltérő gépeket szerkesztettek, amelyeknél a tárolt program és a kettes számrendszer alkalmazása sokkal nagyobb teljesítményt tett lehetővé kisebb költségű technikai eszközök által. Az első gép, amely e szerint épült, az EDVAC volt, nemsokára azután készült az EDSAC a cambridgei egyetemen, 1949-ben. Azóta az ehhez a típushoz tartozó gépek igen nagy fejlődésen mentek keresztül a működési sebesség, megbízhatóság és a kezelés egyszerűsége tekintetében.

A UNIVAC (a Sperry Rand Co. gyártmánya) volt 1951-ben az első kereskedelmi forgalomban megjelent, tömeggyártásban készült gép. Azóta ennek a gépnek számos továbbfejlesztése, "új kiadása" van. 1953-ban az IBM elindította az u.n. 700-as sorozat gépeit. Ezek nagyteljesítményű, tömeggyártásban készült gépek. A sorozat egyes tagjai természettudományos, mérnöki számítások céljaira, mások üzleti, adatfeldolgozások céljaira készültek. Az alkalmazások két említett típusa

főleg abban különbözik, hogy előbbinél aránylag kevés adaton kell hosszú számítási programot elvégezni (szoros értelemben vett számítási feladatok), utóbbinál sok adaton rövid számítást kell elvégezni és az adatokat csoportosítani, szortírozni kell (adatfeldolgozási feladatok). A kétféle típusú alkalmazások céljaira szolgáló gépek többek között egyes egységeik (aritmetikai egység, memória) teljesítmény-arányában különböznek egymástól.

A legújabbban tervezés alatt álló gépek az előzetes adatok alapján nagyságrendekkel felülmúlják a jelenlegi gépek teljesítményét. Pl. Az IBM tervezett STRETCH nevű gépe 200-szor gyorsabb a jelenlegi univerzális gépeknél.

A Szovjetunióban is nagy eredménnyel folytatják a modern számológépek kifejlesztését és üzemeltetését. Nagy teljesítményű gépek: a BESZM, a SZTRELA. Közepes teljesítményű: az M-2. Kis teljesítményűek: az ESZM, az URAL, az M-3. Nagy teljesítményűnek nevezik azokat a gépeket, amelyek másodpercenként 2000-20.000 műveletet végeznek, közepes teljesítményűnek azokat, amelyek másodpercenként 500-2000 műveletet végeznek, kis teljesítményűnek pedig azokat, amelyek másodpercenként 10-500 műveletet végeznek el. Bár a legnagyobb teljesítményű számológépeket a Szovjetunióban, az Egyesült Államokban és Angliában gyártják és üzemeltetik, most már az európai országok többségének és sok nem európai országnak is van saját számológépe.

1.5. Egy adott számítási probléma gépi megoldását előkészítő munka különböző lépései.

Alábbiakban felsoroljuk azokat a különböző fázisokat, amelyekre egy probléma gépi megoldásának előkészítése feloszlik. Egyesekről, már volt szó az előző tárgyalás során.

a.) Numerikus analízis. Mint már említettük, egy problémának gépi megoldásához elengedhetetlen, hogy "aritmetizálható" legyen, azaz létezzék számára numerikus módszer, amely révén a probléma a gép által-elvégezhető elemi műveletek (aritmetikai alpműveletek) sorozatára bontható. Ennek a numerikus módszernek a konkrét feladatra való kidolgozása a numerikus analízis. Bár ennél bizonyos mértékig figyelembe kell venni az adott gép sajátosságait (pl. hogy a gép utasításrendszere tartalmaz-e osztást, stb.), ez a munka független a gép részletadataitól. A numerikus analízis fő módszere: a feladatnak azonosan vagy csekély, változással ismétlődő műveletcsoportokra való bontása. Ezeket az ismétlődő műveletcsoportokat a gépi programban utasítások csoportjai képviselik, amelyeket a gép egymásután többször hajt végre; a továbbiakban látni fogjuk, hogy lehetséges az ismétlődő csoportok utasításait kisebb mértékben változtatni az egyes ismétlődések közben. A szükséges ismétlődések száma egyes esetekben előre ismert, más esetekben nem ismert. Mindenképpen egy feltételes ugró-utasítás zárja le az utasítások ismétlődő csoportját, amely vagy visszaadja a vezérlést a csoport első utasítására, vagy - elegendő ismétlődés (iteráció) esetén - rátér a program következő részére.

Bár a numerikus analízis rendkívül fontos a problémák gépi feldolgozása szempontjából, előadássorozatunkban nem foglalkozunk vele. Ez a munka ugyanis, mint már említettük, nem tartozik speciálisan a modern számológépekhez, hanem bármilyen aritmetikai eszközzel való számolás (akár asztali számológépekkel, akár kézzel való számolás) esetén is szükséges. A következőkben feltételezzük, hogy a megoldandó feladatok numerikus módszere adott.

- b.) Programozás. A programozás tágabb értelmébe beleértik a numerikus analízis egy részét és az utasítások részletes felírását is. Szűkebb értelmében a programozás a numerikus analízis után következő és a számítógéppel szorosan kapcsolatos előkészítő munka. Ide tartozik elsősorban a gépi számítás logikai menetrendjének (blokk-diagramjának) elkészítése. A blokk-diagram a számítás menetének diagramszerű ábrázolása, amely feltünteti az ismétlődő utasításcsoportokat ("hurkokat") és a program elágazásait. A szűkebb értelemben vett programozáshoz tartozik még: megtervezni a számadatoknak és a program egyes részeinek a memóriában való elhelyezését és megbecsülni a számítás géppel való elvégzésének időszükségletét.
- c.) Kódolás. Ez a számítási program részletes gépi utasításainak megírása, számjegy-kombinációk (számjegykódok) vagy szimbólumok formájában. A kezdő rendszerint ezzel a munkával találkozik legelőször. Egyes problémáknál, amelyeknél kritikus követelmények vannak a memória kapacitásával vagy a program lefutási idejével kapcsolatban, a kódolásra igen nagy figyelmet kell fordítani. A kódolás legszorosabb kapcsolatban van az adott gép utasításrendszerével és egyéb jellemzőivel. Ezért bizonyos értelemben újra kell megtanulni egy másik gép kódolását, bár az újratanulás sokkal könnyebb, mint a kódolás eredeti megtanulása. Ezzel a munkával kapcsolatban követhető el a legtöbb hiba, és ez a körülmény, valamint az, hogy a kódolás a leghosszadalmasabb előkészítő munka, készíti a fejlesztőket a kódolás automatizálására, azaz magával a géppel való elvégeztetésére.
- d.) Ellenőrzés. Az eddig felsorolt előkészítő lépésekben annyi hibalehetőség rejlik, hogy a programok első felírásban többnyire hibásak. Ezek a hibák vagy olyan ter-

mészetűek, hogy a gép megáll, mielőtt bármilyen eredményt adna, vagy hatásukra rossz eredmények születnek. A programot ezért többszörösen ellenőrizni kell, úgy is, hogy a gép által kapott első eredményeket (amelyeknél ez gyakorlatilag lehetséges) kézi számítással ellenőrzik.

1.6. A számológépek alkalmazásai.

A számológép meglepően sok feladattípus megoldásánál használható. Ezekből az alábbi fő csoportokat emeljük ki:

a.) Szoros értelemben vett számítási feladatok:

1. Matematikai és mérnöki feladatok nagypontosságú megoldása. Kézi eszközökkel való számítások esetén a legtöbb problémát lényegesen le kell egyszerűsíteni, változókat, magasabb rendű tagokat el kell hanyagolni. A modern számológépek elég gyorsműködésűek ahhoz, hogy lényegesen kevesebb elhanyagolás váljék szükségessé.
2. Új konstrukciók (pl. gőzturbinák, aerodinamikai profilok) működését azok megépítése előtt a számológép által meg lehet vizsgálni. Ezzel nagy összegek és sok idő megtakarítható. A számológép pótolja a modell létesítését, mintegy "szimulálja" a megépítendő berendezést.
3. Új konstrukciók (pl. turbinák, erősáramú hálózatok, stb.) paramétereire számára adott szempontból az optimális kombináció megtalálása. Ez úgy történik, hogy valamennyi lehetséges paraméter-kombinációt végig kell számolni, ami a legegyszerűbb esetektől eltekintve csak gyorsműködésű számológépen oldható meg.
4. Üzemek egészére, vagy résztevékenységeire vonatkozó hatékonysági számítások. Ilyen számítások alapját

olyan matematikai modellek képezik, melyekben a gazdasági eredmény az azt lényegesen befolyásoló tényezők függvényeként kerül kifejezésre. A feladat többnyire a lehetséges eljárások közül a leggazdaságosabb kiválasztásában áll. Ilyen számítások iparági, sőt népgazdasági szinten is végezhetők.

b.) Adatfeldolgozási feladatok.

1. Mérési adatok feldolgozása. Fizikai kísérletsorozatok, új berendezések vagy modellek mérési adatainak (hőmérséklet, nyomás, sebesség, stb. adatok millióinak) a feldolgozása, átlagok, szórások stb. képzése olyan feladat, amely csak számológépeken oldható meg.
2. Üzemek gazdasági és műszaki ügyvitelének gépesítése. Ide tartozik a termeléssel, szállítással és értékesítéssel kapcsolatos nagytömegű adatok rendszeres feldolgozása, a gazdasági egység vezetéséhez, illetve az egész népgazdaság irányításához szükséges módon. Elsősorban a termelés, továbbá munka (bér) és anyagráfordítások elszámolása, a termelési program összeállítása és ütemezése és készletek állományváltozásának tág értelemben vett nyilvántartása sorolható ide. E folyamatok gépesítése révén a regisztráláson túlmenően lehetővé válik a nyilvántartások közvetlen operatív felhasználása. Pl. meghatározott készlet-szint fenntartásával kapcsolatos intézkedések.
3. Az államigazgatás nagytömegű adatnyilvántartásával és adatfeldolgozásával kapcsolatos folyamatainak gépesítése. Ide sorolhatók a bármiféle statisztikai munkák, társadalombiztosítási nyilvántartások és számítások, adónyilvántartás és kivetés, állami költségvetéssel és beruházásokkal kapcsolatos nyilvántartások és számítások.

Lehetséges, hogy a gazdasági alkalmazások a közeljövőben sokkal nagyobb számításai volument fognak képviselni, mint a természettudományos és műszaki alkalmazások.

A fenti felsorolt alkalmazásoknak többé-kevésbé közös jellemzője, hogy a feladat megoldásán belül bizonyos alapvető számítást kell igen sokszor elvégezni, különböző adatokon. Pl. a bérelszámolás esetében azonos típusú számítást kell elvégezni minden munkavállalóra és minden fizetési időszakra. Egy berendezés mérési adatainak feldolgozásánál sokezerszer kell elvégezni ugyanazt a fajta számítást. Valamilyen műszaki terv optimalizálása esetén a paraméterek minden lehetséges kombinációjára azonos számításokat kell elvégezni. Ezek a feladatok mind lehetővé teszik az ismétlődő műveletcsoportok, ismétlődő (iterált) utasításcsoportok szerinti programozást, az u.n. utasításciklusok programozását, melyekről már előbb említés esett. Gyakorlatilag nem érdemes olyan feladatot számológépre vinni, ahol a feladaton belül az alapvető számítás csak kevésszer ismétlődik. Ez a későbbiek folyamán nyilvánvalóvá lesz.

2. Fejezet.

AZ M-3 ORGANIZÁCIÓJÁNAK ÉS UTASÍTÁSRENDSZERÉNEK RÉSZLETES LEÍRÁSA.

2.1. A gép fő egységei.

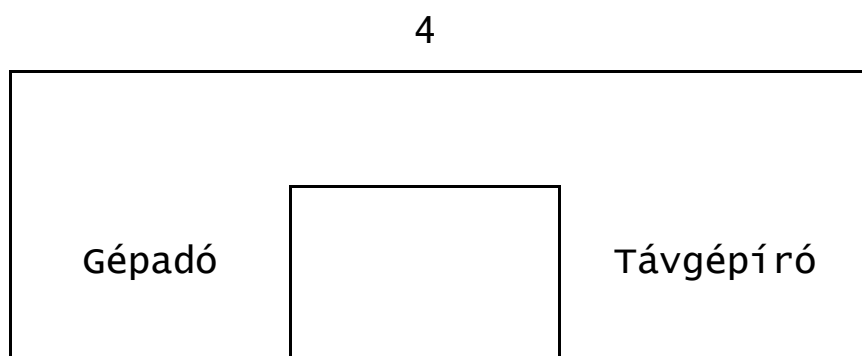
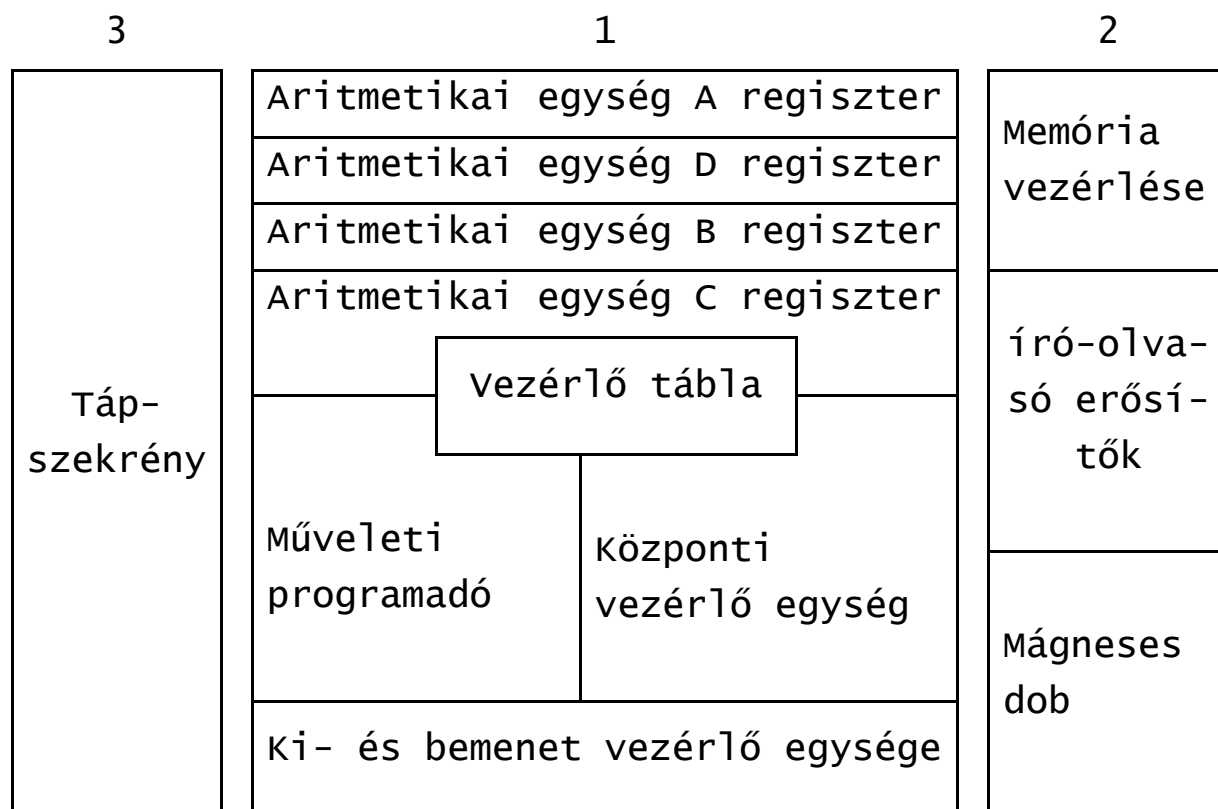
Az előző fejezetben megismert univerzális digitális, automatikus tárolt programú, elektronikus számológépek családjában a nálunk épülő M-3 gép a kisméretű, kisteljesítményű gépek közé tartozik.

külső felépítését tekintve a gép a következő részekből áll (1. a 2. ábrát).

1. Főszekrény, amely az aritmetikai és vezérlőegységeket tartalmazza.
2. Memória-szekrény.
3. Tápszekrény, amely a gép működéséhez szükséges stabil feszültségeket állítja elő.
4. Asztal a ki- és bemenő berendezések számára.

A gép összesen kb. 800 elektroncsövet tartalmaz, fogyasztása 100 kw.

1. A főszekrény felső részén az aritmetikai egység (AE) helyezkedik el, amely négy regiszterből áll. Az M-3 gép 31 bináris jegyet ("bit") tartalmazó szavakkal dolgozik. Számok ábrázolásánál ebből egy bit az előjelet jelöli, 30 bit pedig a szám abszolút értékét ábrázolja. Így az AE, amely csak ez utóbbiakkal végez műveleteket, 30 bináris jegy tárolására alkalmas regisztereket tartalmaz.



2. ábra.

AZ M-3 gép egységei.

A bináris számjegyek tárolása u.n. triggerekben történik: ezek olyan elektronikus áramkörök, amelyeknek két stabil, egymástól jól megkülönböztethető állapotuk van. Az egyik ilyen állapotot a "0", a másikat az "1" bináris számjegyek feleltetjük meg. Egy-egy regiszter tehát, amely 30 ilyen triggerből áll, képes egy M-3-ban használt szó tárolására.

A főszekrény alsó részén elhelyezkedő vezérlőegység három részből áll.

- a.) Központi vezérlő egység.
- b.) Műveleti programadó.
- c.) Ki- és bemenet vezérlése.

Ezekben találhatunk kisebb regisztereket egyes adatok (pl. az utasításokban szereplő címek) tárolására, ezenkívül számlálókat, valamint a vezérlő impulzusok képzését és továbbítását végző áramköröket.

A főszekrény középső részén találjuk a vezérlőtáblát, ahol a gép irányításához, valamint ellenőrzéséhez szükséges fontosabb kapcsolók és jelzőlámpák vannak összegyűjtve. Ennek ismertetésére még visszatérünk.

2. A M-3 gép memóriáját alkotó mágneses dob a memóriaszekrény alsó részében forog 3000 fordulat/perc sebességgel. Eredetileg 2048 szó befogadására van tervezve, ezt azonban egyelőre nem lehet teljes mértékben kihasználni.

Egy megadott címhez tartozó rekesz megkeresésének átlagos ideje, az u.n. elérési idő a dobnál 10 msec. Az M-3 gépnél ez az aránylag nagy elérési idő egyben a gép általános sebességét is korlátozza, mert - mint majd látni fogjuk - az egyes műveletek végrehajtási idői nagyjából elhanyagolhatók a memóriából való kiolvasás, ill. oda való beírás fenti idejéhez képest. Mivel a tapasztalatok azt mutatták, hogy egy utasítás végrehajtásához átlagosan

valamivel több, mint 3-szor kell a memóriához fordulni, azt mondhatjuk, hogy a gép kb. 30 utasítást tud másodpercenként végrehajtani, azaz kb. ennyi aritmetikai műveletet végez másodpercenként, ami 100.000 műv./óra sebességnek felel meg.

A memóriaszekrényben a dob felett helyezkednek el az u.n. író-olvasó erősítők, valamint a memória vezérlőberendezése.

Az M-3 felépítése lehetővé teszi a belső memóriának 4096 szó befogadóképességűre való bővítését a gép többi részének lényeges megváltoztatása nélkül. A gép kiegészíthető gyorsműködésű ferritmemóriával, melynek igen rövid elérési ideje lehetővé teszi az AE működési sebességének lényegesen jobb kihasználását és ezzel a gép átlagos sebességének kb. 1500-2000 művelet/másodpercre való növelését. Kiegészíthető a gép mágnesszalagos segédmemóriával is, ami nagymértékben megnöveli a gép adattároló képességét és ezáltal alkalmazásának lehetőségeit is.

3. A tápszekrény a programozással semmiféle kapcsolatba nem kerül, ezért részletes ismertetését mellőzzük.
4. Az M-3 gép bemenőberendezését egy szabványos Siemens gyártmányú gépadó, kimenőberendezését egy ugyanilyen távgépíró alkotja. A kiszámításra kerülő feladatok programját és számadatait egy perforátor-távgépíró segítségével szalagra kell lyukasztani, a gépadó a lyukakat letapogatja, és impulzusok formájában beviszi a gépbe. A bevétel jelenlegi sebessége 30 szó/perc. Ez megfelelő gyorsműködésű beviteli berendezés segítségével lényegesen (akár százszorosára is) növelhető.

A közbűlső és végeredmények, de a memória bármilyen rekeszének tartalma is a távgépíró segítségével 1, 2, 3, 4 vagy 5 oszlopos táblázatok formájában kiírathatók. A kiírandó mennyiségeket természetesen a programozás során kell előre kijelölni. Egy szó kiírása a jelenlegi berendezéssel kb. 2 másodpercet vesz igénybe, a kiírási sebességet a bemenethez hasonlóan lényegesen növelni lehet.

2.2. A műveletek végrehajtásának módja.

Rátérve a gép működésének ismertetésére, először azt nézzük meg, hogyan végzi el a gép aritmetikai egysége a műveleti programadó irányításával magukat az egyes alpműveleteket.

A gép a 30 bináris jegyből és előjelből álló számot egy 1-nél kisebb abszolútértékű törtszámnak tekinti, azaz a

| | | | | |
|---|----------------|----------------|--|-----------------|
| ± | a ₁ | a ₂ | | a ₃₀ |
|---|----------------|----------------|--|-----------------|

számot $\pm 0, a_1 a_2 \dots a_{30} = \pm (a_1 \cdot 2^{-1} + a_2 \cdot 2^{-2} + \dots + a_{30} \cdot 2^{-30})$

értékűnek veszi, ahol a_1, a_2, \dots, a_{30} természetesen csak 0 vagy 1 lehet.

A számok ábrázolásának ezt a módját fixpontos ábrázolásnak nevezzük, mert a tizedespont szerepét játszó "bináris pont"-ot egy fix helyre, - jelen esetben a szám elejére - képzeljük. Így a gép közvetlenül csak a -1 és +1 közé eső számokkal tud műveleteket végezni, sőt ha valamely művelet eredményeképpen ezen intervallumon kívüleső számot kapunk, a gép megáll. Ezért minden feladatnál a programozónak gondoskodnia kell arról, hogy mind a kiinduló adatok, mind pedig a közbűlső és végeredmények ebbe az intervallumba transzformálódjanak.

Ennek módszereiről, a későbbiekben lesz szó.

Az AE a gépben ábrázolható számokon a következő műveleteket tudja elvégezni:

- 1.) összeadás (+),
- 2.) kivonás (-),
- 3.) osztás (:),

amennyiben a művelet elvégezhető, azaz az eredmény abszolút értéke kisebb marad, mint egy.

- 4.) szorzás (x),

ez mindig elvégezhető, mert 1-nél kisebb számok szorzata mindig 1-nél kisebb,

5.) helyértékenkénti logikai szorzás (\wedge), ami azt jelenti, hogy az eredmény egyes helyértékeire akkor és csak akkor írunk 1-et, ha mindkét tényező ugyanazon helyértéke 1-es volt.

$$p1. (11010) \wedge (01101) = 01000$$

Az AE-nek, mint már említettük, négy regisztere van. Ezek feladata:

A. regiszter: Tárolja az összeadandót, kivonandót, szorzandót, ill. osztót.

B. regiszter: Tárolja az összeadandót, kisebbítendőt, szorzót, ill. osztandót. Itt keletkezik az összeg, különbség és szorzat.

C. regiszter: Ugyanazokat a mennyiségeket tárolja, mint a B regiszter. Itt keletkezik a hányados és a logikai szorzás eredménye. Összeköti a ki és bemenő berendezést, valamint a memóriát egymással és a gép többi egységeivel.

D. regiszter szerepét a következőkben ismertetjük.

Valamennyi művelet eredménye - függetlenül attól, hogy a B vagy C regiszterben keletkezett - átmegy a másik regiszterbe is, a művelet elvégzése után tehát a B és C regiszterekben egyaránt megtalálható.

2.2.1.) Összeadás.

A kettes számrendszerben - mint minden más számrendszerben is, két szám összeadása úgy történik, hogy a legkisebb helyértéktől kezdve rendre összeadjuk az azonos helyértékeken lévő számjegyeket, és ha az összeg a számrendszer alapszámát (jelen esetben kettőt) nem éri el - azaz az összeg egy számjeggyel leírható, akkor beírjuk, az eredmény megfelelő helyértékére. Ha az összeg meghaladja az alapszámot, akkor az eredmény megfelelő számjegye az a szám lesz, amely azt mutatja, hogy mennyivel haladja meg az összeg az alapszámot, és a tőle balra lévő (eggyel magasabb) helyérték összegéhez 1 adódik hozzá. A papíron történő összeadásnál ezeket az "átviteleket" általában fejben szoktuk megjegyezni, semmi akadálya nincs azonban annak, hogy ezeket külön helyen fel is jegyezzük. Ilyen módon az összeadást két lépésben végezhetjük el.

- a.) Minden helyértéknél képezzük és feljegyezzük a balra lévő helyértékre a keletkező átviteleket, természetesen ismét a legkisebb helyértéktől elindulva és mindenütt figyelembe véve az átfutás során addig már felírt átviteleket.
- b.) Elvégezzük, helyértékenként egyszerre, a két szám és az átvitelek összeadását (az alapszámot meghaladó összegből levonva az alapszámot, de újabb átviteleket természetesen már nem képezve).

P1:

| <u>tíz-es számrendszerben</u> | | <u>kettes számrendszerben</u> |
|-------------------------------|------------|-------------------------------|
| 1.) szám | 106 | 1101010 |
| 2.) szám | 295 | 100101111 |
| átvitelek | <u>011</u> | <u>1101110</u> |
| összeg | 401 | 110011001 |

Fentieket a kettes számrendszerre alkalmazva a következő gépies szabályokat állapíthatjuk meg:

- a.) ha valamely helyértéken a két szám megfelelő helyértéke, valamint az arra a helyértékre vitt átvitel közül legalább két 1-es van, akkor a tőle balra lévő helyérték átviteli helyére 1-est írunk, különben 0-át.
- b.) Ha valamely helyértéken a 3 említett számjegy közül páratlan számú 1-es van, az összeg megfelelő helyértékére 1-est, ha páros, akkor 0-át írunk.

Az M-3 gép aritmetikai egysége az összeadást a fenti szabályok alapján végzi el.

A két összeadandó az A és B regiszterben van elhelyezve, az átvitelek a D regiszterben lesznek feljegyezve. Az összeadás első lépése az, hogy a D regiszterben jobbról-balra "végigfut" az átvitel: a legkisebb helyértéktől kezdve D minden helyérték-triggerre az a.) szabály szerint megfelelő állapotba kerül. Ennek elvégzése után helyértékenként egyszerre történik meg az összeg jegyeinek képzése, a b.) szabály alapján, a B regiszterben. Az egész összeadást a gép 60 μ sec alatt végzi el.

Példa:

$$\begin{array}{r} 1.) \text{ A: } 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \\ \text{D: } 0 \ 1 \ 1 \ 1 \ 1 \ 1 \quad \text{az átvitel végigfutása után} \\ \text{B: } \underline{0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1} \\ 2.) \text{ A: } 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \\ \text{D: } \underline{\quad} \underline{\quad} \underline{\quad} \underline{\quad} \underline{\quad} \underline{\quad} \underline{\quad} \quad \text{összeg képzése után} \\ \text{B: } 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \end{array}$$

Az összeadás fenti módszerének nagy előnye az, hogy a művelet elvégezhetősége rendkívül egyszerűen dönthető el. Nyilvánvaló ugyanis, hogy két egynél kisebb szám összeadásánál akkor és csak akkor kaphatunk egynél nagyobb eredményt, ha a legmagasabb - azaz első - helyértékről is megy tovább átvitel a - nemlétező - "nulladik" helyérték felé, mert a bináris pontot az első helyérték elé képzeljük. Ha tehát a D regiszternek erre a nulladik helyre is teszünk egy D_0 -al jelölt - triggert, akkor ennek állapota megmutatja a művelet elvégezhetőségét: ha $D_0 = 0$, akkor az összeg kisebb mint 1, tehát a művelet elvégezhető, ha $D_0 = 1$, akkor az összeg nagyobb vagy egyenlő lesz mint 1, tehát a művelet nem végezhető el. Ez utóbbi esetben nem történik meg az összeg képzése és a gép megáll.

Példa:

$$\begin{array}{r} \text{A: } \quad 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \\ \text{D: } \quad 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \\ \text{B: } \underline{\quad \quad 1 \ 1 \ 1 \ 0 \ 1 \ 0} \\ \text{Az összeadás nem végezhető el!} \end{array}$$

2.2.2.) Kivonás.

A gép a kivonás műveletét az összeadásra vezeti vissza a következő módon:

Tekintsünk a gépben egy $a \geq 0$ számot, és változtassuk az ellenkezőjére minden egyes számjegyet (1-et 0-ra, 0-t 1-re), jelöljük az eredményt \bar{a} -al. Ekkor

$$\bar{a} + a = 0,111 \dots 1 = 1 - 2^{-3^0}$$

és így

$$\bar{a} + 2^{-3^0} = 1 - a$$

Az \bar{a} számot az a szám komplementének fogjuk nevezni.

A fentiek alapján, ha $b \geq 0$

$$\bar{a} + 2^{-3^0} + b = 1 - a + b = 1 + (b - a)$$

A gép tehát kivonás esetén mindenekeelőtt az A regiszterben elhelyezett a kivonandónak képezi a komplementét oly módon, hogy az A regiszter minden triggerének állapotát megváltoztatja, és 2^{-3^0} hozzáadása céljából a D regiszter utolsó - eddig feltétlenül 0 állapotban levő - triggerét állítja 1 állapotba, ami mint könnyen belátható, 2^{-3^0} hozzáadását eredményezi, az összeadás elvégzése során.

Ezután megpróbáljuk elvégezni az összeadást a B regiszterben levő számmal, azaz elvégezzük az átvitel végigfuttatását. Ha ennek hatására $D_0 = 1$ lesz, akkor az azt jelenti, hogy

$$\bar{a} + 2^{-3^0} + b = 1 + (b-a) \geq 1.$$

Tehát az összeget képezve nem annak valódi értékét fogjuk megkapni, hanem mivel a B regiszter 0-ik helyértékét nem vesszük figyelembe, éppen az összeg törtrészét, azaz a $(b-a) \geq 0$ különbséget kapjuk, a kivonás kívánt eredményét.

Kivonás esetén tehát $D_0 = 1$ nem állítja le a gépet, mint összeadásnál, hanem ellenkezőleg lehetővé teszi az összeg képzését.

Más a helyzet, ha $D_0 = 0$. Ekkor

$$\bar{a} + 2^{-3^0} + b = 1 + (b-a) < 1,$$

tehát az összeg képzése most már a valódi összeget fogja adni, nekünk azonban a negatív b-a értékre van szükségünk. Ezért felcseréljük a és b szerepét úgy, hogy képezzük mindkét regiszter (A és B) komplementjét az előbbi módon: így A-ban újra a lesz és B-ben B. Így feltétlenül $D_0 = 1$ lesz, mert most

$$a + B + 2^{-30} = 1 + (b-a) \geq 1$$

lesz, ha $1 + (b-a) < 1$ volt.

Így az összeg képzése már elvégezhető, és megadja a $(b-a) < 0$ különbség abszolút értékét. Ugyanakkor gondoskodik a gép arról, hogy az eredmény előjele negatívra változzék.

A kivonást a gép kb. 70 ill. 120 μsec alatt végzi el, attól függően, hogy a fenti két eset közül melyikkel kerül szembe.

| | | | | |
|----------|-----|---------------------|----------|---------------------|
| Példa: | 1.) | A: 0 0 1 0 1 | 2.) | A: 1 0 1 0 1 |
| | | D: - - - - - | | D: - - - - - |
| | | <u>B: 0 1 0 1 1</u> | | <u>B: 0 1 0 1 1</u> |
| A kompl. | | A: 1 1 0 1 | A kompl. | A: 0 1 0 1 0 |
| | | D: 1 1 0 1 1 1 | | D: 1 0 1 0 1 1 |
| | | <u>B: 0 1 0 1 1</u> | | <u>B: 0 1 0 1 1</u> |
| Összeg | | A: 1 1 0 1 0 | A és B | A: 0 1 0 1 0 |
| képzése | | D: - - - - - | kompl. | D: 1 0 1 0 1 1 |
| | | <u>B: 0 0 1 1 0</u> | | <u>B: 1 0 1 0 1</u> |
| | | | Összeg | A: 1 0 1 0 1 |
| | | | képzése | D: - - - - - |
| | | | | <u>B: 0 1 0 1 0</u> |

és a B-ben lévő különbség előjele megváltozik!

Az eddigiekben feltételeztük, hogy a és b pozitívek; negatív számokkal a gép úgy végzi el az összeadást és kivonást, hogy visszavezeti az abszolút értékekkel végzett megfelelő műveletre, így különböző előjelű számok összeadása helyett kivonást, kivonása helyett összeadást végez. Az eredmény előjelét mindig a b szám (a kisebbítendő) előjele határozza meg (a kivonás második eseténél tehát ezt kell megváltoztatni).

Végeredményben tehát a gép mindig összeadást vagy pozitív eredményű kivonást végez, és ezekből a műveleteknek és az eredmény előjelének megfelelő megválasztásával az előjeles számok összeadásának és kivonásának összes esete előállítható.

2.2.3.) Szorzás.

Az M-3 szorzási módszerét legkönnyebben a szokásos szorzási eljárás vizsgálatával érthetjük meg. Tekintsük pl. a következő szorzatot:

$$\begin{array}{r} 0,78125 \cdot 0,875 \\ \hline 390625 \\ 546875 \\ 625000 \\ \hline 0,68359375 \end{array}$$

Az itt alkalmazott szorzási módszert a következő módon is leírhatjuk:

Képezzük az első részletszorzatot, írjuk be egy "regiszter"-be, amely a tizedesvessző után annyi jegyet tartalmaz, mint a szorzandó:

$$3|90625|$$

majd "léptessük" egy helyértékkel jobbra minden számjegyet, írjuk alá a következő részletszorzatot, és adjuk össze a két számot:

$$\begin{array}{r} 3\ 9\ 0\ 6\ 2\ 5 \\ 5\ 4\ 6\ 8\ 7\ 5 \\ \hline 5\ 8\ 5\ 9\ 3\ 7\ 5 \end{array}$$

Az eredményt ismét léptessük jobbra egy helyértékkal, adjuk hozzá a következő részletsorozatot,

$$\begin{array}{r} 5\ 8\ 5\ 9\ 3\ 7\ 5 \\ 6\ 2\ 5\ 0\ 0\ 0 \\ \hline 6\ 8\ 3\ 5\ 9\ 3\ 7\ 5 \end{array}$$

majd léptessük az eredményt ismét jobbra egy helyértékkal:

$$68359\ 375$$

A kettes számrendszerben a helyzet annyiban egyszerűsödik, hogy a részletszorzatok képzése csak 1-el vagy 0-val való szorzást jelent, azaz az előzőleg keletkezett és már jobbra léptetett részletsorozat-összeghez vagy hozzáadjuk a szorzandót, vagy nem adunk hozzá semmit.

P1. a fenti szorzás:

$$\begin{array}{r} 0,11100 \cdot 0,11001 \\ \underline{11100} \\ 0 \\ 0 \\ 11100 \\ \underline{11100} \\ 0,101011100 \end{array}$$

illetve az ismertetett módszer szerint:

$$1.) \quad \begin{array}{r|rrrrrr|r} & 1 & 1 & 1 & 0 & 0 & & 0 \\ & 0 & 0 & 0 & 0 & 0 & & \\ \hline & & 1 & 1 & 1 & 0 & & 0 \end{array}$$

$$2.) \quad \begin{array}{r|rrrrrr|r} & & & 1 & 1 & 1 & & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & & & \\ \hline & & 1 & 1 & 1 & 0 & & & 0 \end{array}$$

A 2.) lépést (B és C léptetése jobbra) a gép oly módon végzi el, hogy összehasonlítja a szomszédos triggerek állapotait és nemegyezés esetén a jobboldaliakat megváltoztatja. Balról a B_0 helyértékre 0 kerül, B_0 eddigi tartalma B_1 -be megy át é.i.t., végül a B_{30} helyérték tartalma elvész; a C regiszterben ugyanez történik, azzal az eltéréssel, hogy itt - tekintve, hogy a gépen C_0 nincsen - C_1 -be kerül 0 minden léptetésnél. A léptetés eredményeképpen C_{30} -ba mindig a szorzandó soron következő számjegye kerül. Így az 1.) szabály valóban a soron következő részletszorzat hozzáadását jelenti az eddigi összeghez.

A B-regiszter tartalmának sorozatos jobbra-léptetése természetesen azt eredményezi, hogy a - hatvan számjegyből álló szorzat utolsó 30 jegye elveszik. A hiba, amit a gép ilyen módon elkövet, kisebb, mint 2^{-30} , azaz kisebb a gépen ábrázolható legkisebb számnál.

Egy szorzás a M-3-on kb. 1900 μ sec-ot vesz igénybe.

Példaképpen végezzük el az előbbi szorzást, az egyszerűség kedvéért 30 helyett csak 5 számjegyet írunk be.

Az átvitelek képzését (D regiszter kitöltése) a léptetésekkel együtt tüntettük fel, és csak azokban az esetekben végeztük el, amikor a következő lépésben összeadást kellett végezni.

| | | |
|----|------------------|----------------------------------|
| A: | 1 1 1 0 0 | B törlés |
| D: | 0 0 0 0 0 | |
| B: | 0 0 0 0 0 | |
| C: | <u>1 1 1 0 0</u> | |
| A: | 1 1 1 0 0 | Mivel $C_{30}=1$, elvégezzük az |
| D: | 0 0 0 0 0 | összeadást |
| C: | <u>1 1 1 0 0</u> | |

A: 1 1 1 0 0
B: 0 0 1 1 1 0
C: 1 1 1 0 0
A: 1 1 1 0 0
B: 0 0 1 1 1 0
C: 1 1 1 0 0
A: 1 1 1 0 0
D: 0 0 0 0 0 0
B: 0 0 0 0 1 1
C: 0 0 0 1 1
A: 1 1 1 0 0
B: 0 1 1 1 1 1
C: 0 0 0 1 1
A: 1 1 1 0 0
D: 1 1 1 0 0 0
B: 0 0 1 1 1 1
C: 0 0 0 0 1
A: 1 1 1 0 0
B: 1 0 1 0 1 1
C: 0 0 0 0 1
A: 1 1 1 0 0
B: 0 1 0 1 0 1
C: 0 0 0 0 0

B és C léptetése jobbra, összeadást nem végzünk, mert $C_{30}=0$

B és C második léptetése jobbra, összeadást nem végzünk, mert $C_{30}=0$

B és C harmadik léptetése jobbra; mivel $C_{30}=1$

Elvégezzük az összeadást.

B és C negyedik léptetése jobbra: mivel $C_{30}=1$

elvégezzük az összeadást.

B és C ötödik és egyben utolsó léptetése jobbra

A szorzatot (pontosabban annak első öt számjegyét) a B regiszterben kaptuk.

2.2.4.) Osztás.

Az M-3 gépen az osztás csak abban az esetben végezhető el, ha az osztandó kisebb, mint az osztó, hiszen különben az eredmény 1-nél nagyobb vagy 1-el egyenlő lenne. A művelet elvégezhetőségét a gép még a végrehajtás kezdete előtt megvizsgálja, és ha azt találja, hogy nem végezhető el, akkor megáll.

Az osztás műveletét papíron a következő módon szoktuk végrehajtani (feltesszük, hogy az osztandó kisebb, mint az osztó).

Kitésszük a tizedesvesszőt, és "lehozunk egy 0-át" (azaz az osztandót (b) balra léptetjük egy helyértékkel), és megnézzük, hogy mi az a legnagyobb k nemnegatív egészszám, amivel az osztót (a) szorozva még az osztandónál nem nagyobb számot kapunk. Ez feltétlenül kisebb lesz mint a számrendszer alapszáma, hiszen az osztandó kisebb volt az osztónál, és a balra léptetés, azaz a 0-hozzáírás éppen a számrendszer alapszámával való szorzást jelenti (ha $b < a$, és $k \cdot a \leq 10 \cdot b$, akkor $k < 10$ lesz, mert $k \geq 10$ és $a \geq b$ -ből $k \cdot a > 10 \cdot b$ következne).

Így a keresett szám egy számjegy lesz az adott számrendszerben. Ez a számjegy lesz a hányados első számjegye. Vonjuk le a fenti számjegynek az osztóval való szorzatát az osztandóból, és a maradékkal folytassuk az eljárást: léptessük balra, keressük meg azt a számjegyet, amivel az osztót szorozva, még kisebb számot kapunk, mint az így nyert szám, és így tovább.

$$\begin{array}{r} \text{Pl.: } 0,59375 : 0,75000 = 0,79166 \\ \quad 593750 \\ \quad - 525000 \\ \quad \hline \quad 687500 \\ \quad - 675000 \\ \quad \hline \quad 125000 \\ \quad - 75000 \\ \quad \hline \quad 500000 \\ \quad - 450000 \\ \quad \hline \quad 50000 \end{array}$$

Kettes számrendszerben a helyzet ismét annyiban egyszerűsödik, hogy csak 2 számjegyet kell tekinteni, csak azt kell megvizsgálni, hogy az osztó kisebb-e mint a balra léptetett osztandó (ill. maradék) vagy nem. Előző esetben 1-el, utóbbi esetben 0-val szorozva kell kivonnunk, azaz az előbbi esetben elvégezzük a kivonást, az utóbbi esetben nem.

A gépen az A regiszterben helyezük el az osztót (a) és a B regiszterben az osztandót (b). A hányados a C regiszterben fog keletkezni. Ezután elvégezzük egy $b-a$ kivonás előkészítését egészen az átvitel végigfuttatásáig. Ennek eredménye - mint ezt a kivonásnál már láttuk - megmutatja, hogy a $b > a$, ill. $b < a$ egyenlőtlenségek közül melyik áll fenn. Az előbbi esetben osztás nem végezhető el, és a gép megáll, míg az utóbbi esetben megkezdődik az osztás elvégzése. Ez azt jelenti, hogy 30-szor végrehajtjuk a következőket:

A B és C regiszter tartalmát balra léptetjük, ami éppen fordítottját jelenti a szorzásnál megismert jobbra léptetésnek. Itt a 30. helyértékekre fog 0 kerülni, és a B_0 , ill. C_1 tartalma fog elveszni (C tartalma kezdetben közömbös). Ezután végrehajtjuk az átvitel végigfuttatását, hogy megnézzük, elvégezhető-e az A regiszter tartalmának kivonása a B-regiszter balra léptetett tartalmából, vagy nem; azaz megnézzük, hogy a hányados soronlévő számjegye 1 lesz-e vagy 0? Ha $D_0 = 1$, az - mint a kivonásnál már láttuk - azt jelenti, hogy

$$(1 - a) + 2b \geq 1$$

azaz $2b \geq a$, tehát a kivonás elvégezhető (itt figyelembe kell venni egyrészt, hogy az A regiszter tartalmának már az előkészítő kivonás során komplementjét vettük és D_{30} -ba egyidejűleg "1"-et írtunk, másrészt a B regiszter tartalma a balra való léptetés során változott $2b$ -re). A kivonás azonban akkor is elvégezhető, ha $B_0=1$, hiszen ez azt jelenti, hogy a balra való léptetés által a B regiszter tartalma 1-nél nagyobbá lett, és így belőle az 1-nél kisebb osztó feltétlenül kivonható.

Így végeredményben, ha a balraléptetés és az átvitel végigfuttatása után B_0 és D_0 valamelyike is "1" állapotban van, akkor elvégezzük a kivonást és a hányados soronlévő számjegyét, - amely a C regiszter sorozatos balraléptetései

miatt mindig C_{30} -ba tehető - "1"-re állítjuk. Ellenkező esetben, ha tehát $B_0=0$ és $D_0=0$, kivonást nem végzünk, és C_{30} -at is "0" állapotban hagyjuk.

Ilyen módon az osztás is a léptetés és összeadás 30-szoros ismétlésével valósítható meg. Egy osztás végrehajtása a gépen kb. 2000 μ sec-ot vesz igénybe.

Példa: (öt számjegyre)

```
A:  1 1 0 0 0
B:  0 1 0 0 1 1
C:  - - - - -
-----
A:  0 0 1 1 1
D:  0 0 1 1 1 1
B:  0 1 0 0 1 1
C:  - - - - -
-----
A:  0 0 1 1 1
D:  0 0 1 1 1 1
B:  1 0 0 1 1 0
C:  - - - - - 0
-----
A:  0 0 1 1 1
B:  1 0 1 1 1 0
C:  - - - - - 0
-----
A:  0 0 1 1 1
D:  1 1 1 1 1 1
B:  0 1 1 1 0 0
C:  - - - - 1 0
-----
A:  0 0 1 1 1
B:  1 0 0 1 0 0
C:  - - - 1 0
-----
A:  0 0 1 1 1
D:  0 1 1 1 1 1
B:  0 0 1 0 0 0
C:  - - 1 1 0
```

Az osztó az A, az osztandó a B regiszterben.

Képezzük A komplementjét, D_{30} -ba 1-et írunk és elvégezzük az átvitel végigfuttatását. Mivel $D_0=0$, az osztás elvégezhető.

B és C első léptetése balra és az átvitel végigfuttatása. Az összeadást (amely értelem-szerűen tkp. kivonás) elvégezhetjük, mert $B_0=1$

Összeadás elvégzése és $C_{30}=1$ beállítása.

B és C második léptetése balra és az átvitel végigfuttatása. Az összeadást elvégezhetjük, mert $D_0=1$

összeadás elvégzése és $C_{30}=1$ beállítása.

B és C harmadik léptetése balra és az átvitel végigfuttatása. Összeadást nem végzünk, mert $D_0=B_0=0$

A: 0 0 1 1 1
D: 0 0 1 1 1 1
B: 0 1 0 0 0 0
C: - 1 1 0 0
A: 0 0 1 1 1
D: 0 0 1 1 1 1
B: 1 0 0 0 0 0
C: 1 1 0 0 0
A: 0 0 1 1 1
B: 1 0 1 0 0 0
C: 1 1 0 0 1

B és C negyedik léptetése balra és az átvitel végigfuttatása. Összeadást nem végzünk, mert $D_0 = B_0 = 0$
B és C ötödik léptetése balra és az átvitel végigfuttatása. Összeadást elvégezhetjük, mert $B_0 = 1$
összeadás elvégzése és $C_{30} = 1$ beállítása.

A C regiszterben megkaptuk a hányadost. (B regiszterben az osztásnál keletkezett maradék 2^{30} -szorososa található, azt azonban a gép nem tudja felhasználni.)

2.2.5.) Logikai szorzás.

A művelet elvégzése a már ismertett definíció alapján történik: a tényezők az A és C regiszterekben vannak, és azon helyértékeken, ahol az A regiszter 0-t tartalmaz a C regiszter tartalma is 0 lesz, a többi helyértéken változatlan marad.

Példa: (Öt számjegyre)

A: 0 0 1 1 1 a művelet előtt
C: 0 1 1 0 1
A: 0 0 1 1 1 a művelet után
C: 0 0 1 0 1

2.3. Az adatok be- és kivitele.

A számítás kezdete előtt a gépbe be kell vinni a programot és a megoldandó feladat kiinduló adatait. Ezeket számok formájában szalagra kell lyukasztanunk. A szalagon az adatok öt helyértékből álló sorokban helyezkednek el. A bevitelre kerülő adatok tízes vagy kettes számrendszerben lehetnek megadva. A számrendszert egy kapcsoló állítja be. Az előbbi esetben minden sorban egy számjegyet helyezünk el oly módon, hogy az első helyértékre lyukat teszünk (ez felel meg az 1-nek) a többi négy helyértékre pedig a decimális szám bináris megfelelőjét írjuk:

| | | | | | |
|---|---|---|---|---|---|
| o | | | | | 0 |
| o | | | | o | 1 |
| o | | | o | | 2 |
| o | | | o | o | 3 |
| o | o | | | | 4 |
| o | o | | | o | 5 |
| o | o | o | | | 6 |
| o | o | o | o | | 7 |
| o | o | | | | 8 |
| o | o | | | o | 9 |

A számok előjelét külön sorba lyukasztjuk a következő módon:

| | | | | | |
|---|---|---|---|---|---|
| o | o | o | o | | + |
| o | o | o | o | o | - |

Amennyiben kettes számrendszerben felírt adatot akarunk a gépbe bevinni (ilyenek pl. a program utasításai), akkor három bináris számjegyet ábrázolunk egy sorban oly módon, hogy az első helyértékre ismét 1-t írunk, a második helyérték feltétlenül üresen marad és a többi három helyértékre beírjuk a három ábrázolandó számjegyet. Ily módon a

0-7 decimális számjegyeknek megfelelő jelkombinációkat kapjuk.

Ezt úgy is mondhatjuk, hogy ezek a 8-as alapú számrendszer számjegyeinek felelnek meg. Valóban, ha az adott bináris számot ebben a számrendszerben írjuk fel, ugyanerre az eredményre jutunk, hiszen egy a 2-es alapú számrendszerben felírt számot a $8 = 2^3$ alapú számrendszerbe átírni nem jelent mást, mint 3-3 bináris számjegyet összefogva – rövidítésképpen – egy új számjeggyel jelölni. A tízes számrendszerben ennek az felel meg, amikor egy hosszú számot 3-3 számjegyből álló csoportokra bontva írunk le. Pl.:

609 753 438 214

Ha most minden egyes három decimális jegyből álló kombinációnak (tehát a 0-999-ig terjedő decimális számoknak) egy-egy jelet vezetnénk be, akkor azt mondhatnánk, hogy a megadott számot az $1000 = 10^3$ alapú számrendszerben írtuk fel.

A kettes számrendszerben ilyen "új jelek" gyanánt a 0-7-ig terjedő decimális számjegyeket használjuk a három bináris jegyből álló kombinációk jelölésére, és ezt úgy mondjuk, hogy a bináris számokat 8-as számrendszerben írjuk fel.

Pl. a gépben

|0|1|1|0|0|1|1|0|0|0|1|1|0|0|1|0|1|0|1|1|1|1|0|0|1|0|1|0|1|0|0|

alakban ábrázolt bináris számot röviden

+ 6 3 0 6 2 5 7 1 2 4

formában írhatjuk fel, és a szalagon ennek megfelelően a következő módon ábrázolhatjuk:

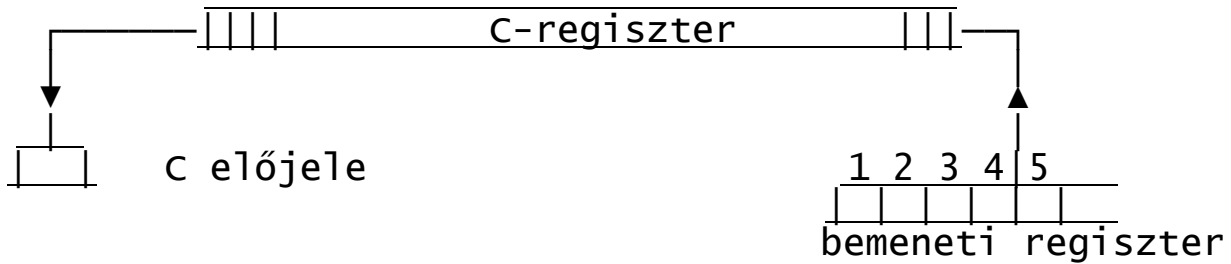
| | | | | |
|---|---|---|---|---|
| ○ | ○ | ○ | ○ | |
| ○ | | ○ | ○ | |
| ○ | | | ○ | ○ |
| ○ | | | | |
| ○ | | ○ | ○ | |
| ○ | | | ○ | |
| ○ | | ○ | | ○ |
| ○ | | ○ | ○ | ○ |
| ○ | | | | ○ |
| ○ | | | ○ | |
| ○ | | ○ | | |

Egy binárisan megadott szám felírása tehát 11 sort vesz igénybe.

Tekintettel arra, hogy a szabványos távgépírók a számokat nem az M-3 számára szükséges, hanem a nemzetközi távgépíró-szabvány szerinti jelkombinációk alapján lyukasztják, a gépbe speciális "átkódoló" van beépítve, amely a szalagra lyukasztott kombinációkat a fenti kombinációkká alakítja át.

A perforált szalag tartalmát egy szalagolvasó berendezés (gépadó) soronként elolvassa, majd minden sor tartalma - az átkódoló által átalakítva - a ki- és bemenet vezérlőegységének öt helyértékből álló bemeneti regiszterébe kerül. Innen - aszerint, hogy a számok tízes vagy nyolcas (tkp. kettes) számrendszerben vannak megadva - különböző módon fog bekerülni az AE C regiszterébe.

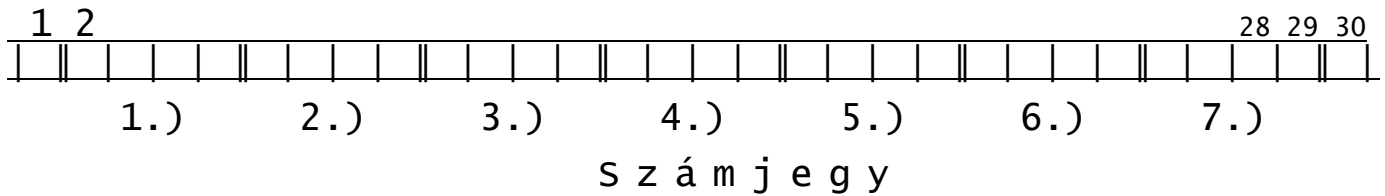
Tekintsük először a binárisan megadott számok esetét. Ekkor a bemeneti regiszter utolsó három helyértékén találjuk azokat a számjegyeket, amelyeket be kell vinni a C regiszterbe. Ezért a C regisztert és a bemeneti regisztert a következő módon "egyesítve" képzeljük el.



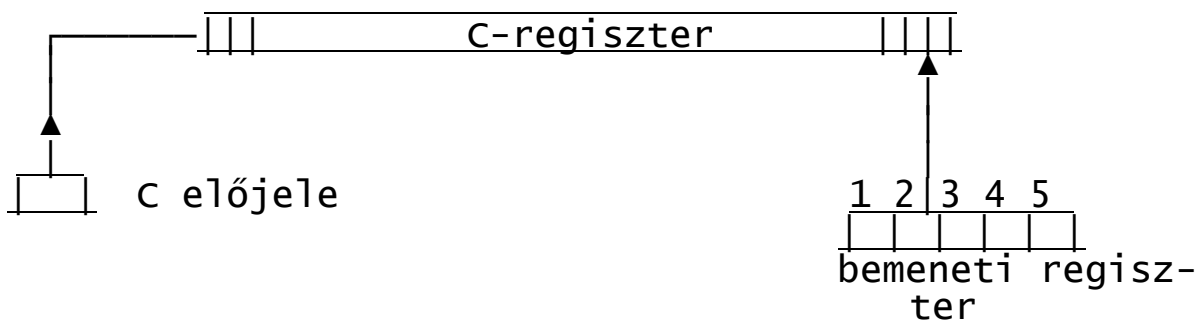
Így, ha balra való léptetést végzünk, a bemeneti regiszter ötödik helyértékének tartalma a negyedikre, a negyedik tartalma a harmadikra, a harmadik helyérték tartalma pedig C_{30} -ba megy át, a C regiszter tartalma balra lép, végül tartalma átmegy C előjelének helyére.

Így három ilyen léptetés eredményeképpen a bevételre kerülő három bináris jegy a C regiszter három utolsó helyértékére ($C_{28}C_{29}C_{30}$) kerül. A következő sor leolvasása utáni három léptetésnél ezek a C_{25} , C_{26} és C_{27} helyértékekre kerülnek, az utolsó három helyértékének helyet adva a most leolvasott három bitnek é.i.t. Az utolsó (tizenegyedik) sor leolvasása után az előszörre leolvasott - az előjelnek megfelelő - sor három bináris jegye (1, 1, előjel) a végrehajtott tízszer három léptetés eredményeképpen előrejut $C_1 C_2 C_3$ -ra. Utána, a C regiszter többi helyértékén a további 9 sornak megfelelő 27 bináris jegy található, míg a most leolvasott utolsó három jegy a bemeneti regiszterben van. A most következő utolsó három léptetés az első két helyértéken lévő (felesleges) 1-est "kitolja a regiszterből" a harmadik helyértéken lévő - az előjelt megadó - számjegy az előjel helyére kerül, a szám abszolút értékének 30 bináris jegye pedig a regiszterben helyezkedik el.

Más a helyzet tízes számrendszerben megadott számok bevételénél. Tekintettel arra, hogy egy decimális jegy bináris ábrázolása négy helyértéket vesz igénybe, egy 31 jegyű szóban az előjelen kívül hét decimális jegyet lehet ábrázolni, (az 1-28 helyértékekig) és a két utolsó helyérték üresen marad.



Ezért itt a bemeneti regiszter második helyértéke van összekötve a C regiszter 28. helyértékével, és egy-egy sor leolvasása után négy léptetés történik:



A nyolcadik sor leolvasása utáni léptetések befejezése után a szám a fenti ábrán feltüntetett módon a C regiszterben helyezkedik el. (Tekintettel arra, hogy a gép a műveleteket kettes számrendszerben felírt számokkal végzi, a fenti módon - u.n. binárisan kódolt decimális - alakban felírt számot át kell alakítani kettes számrendszerbeli megfelelőjévé.

Ezt az átalakítást a gép már nem végzi el automatikusan, erre külön program segítségével kell a gépet utasítani. Ilyen "számrendszerváltó" programokról egy későbbi fejezetben lesz szó).

A C regiszterbe bevitt számnak a gép memóriájába való beírásánál fontos szerepet játszik a központi vezérlő egységben helyet foglaló szelekciós regiszter (SZR). Ebben tároljuk mindig annak a memóriarekesznek a címét, amelyet beírás vagy kiolvasás céljából meg akarunk keresni. Amennyiben ugyanis a memória "írás" vagy "olvasás" utasítást kap, mindig az SZR-ben tárolt címre írja be a C-regiszter

tartalmát, illetve az itt tárolt cím tartalmát viszi be a C-regiszterbe.

Így mielőtt a szalagról valamely szót a C-regiszterbe bevinnénk, valamilyen módon be kell vinni a SZR-be annak a rekesznek a címét, ahová ezt a számot el akarjuk helyezni.

Aszerint, hogy ez hogyan történik, kétféle beviteli módot különböztetünk meg:

- a.) program segítségével történő bevitel
- b.) cím megadásával " "

Az a.) módszernél a bevitelre a gépben már bentlévő program egy utasítása ad parancsot a gépnek, az említett cím magában az utasításban van feltüntetve, és innen kerül - a későbbiekben ismerttetendő módon - az SZR-be. A bevitelre kerülő szó leolvasása után a gép a szalagon egy "szó vége" jelet talál, melynek hatására „írás” utasítást ad a memóriának.

A b.) módszernél a címet magán a szalagon adjuk meg: minden szó elé odaírjuk, hogy milyen címre akarjuk elhelyezni. Itt tehát a gép először a 12 helyértékből álló címnek megfelelő négy sort olvassa le a szalagról és helyezi el a C regiszter utolsó 12 helyértékén. Ezután a szalagon egy "cím vége" jel következik, amelynek hatására a C₁₉-C₃₀ helyértékek tartalma, tehát éppen a most beírt cím átmegy az SZR-be. Ezután a bevitelre kerülő számnak megfelelő 11 sor tartalma (31 bit) tölti meg lépésenként a C regisztert, majd itt is a "szó vége" jel következik.

Ennek a módszernek az az előnye, hogy nincs szükség előre bevitt programra a bevitel egy gombnyomás indítja. Kátránya, hogy csak binárison megadott anyag bevitelénél alkalmazható, mert a címek bevitele egyébként nem lehetséges.

Példák (az M-3 gépnek megfelelő, tehát átkódolást nem igénylő perforálást feltételezve).

1.) A decimálisan megadott - 0, 7968345 számot

| | | | | | |
|---|---|---|---|---|--------------|
| ○ | ○ | ○ | ○ | ○ | (-) |
| ○ | | ○ | ○ | ○ | (7) |
| ○ | ○ | | | ○ | (9) |
| ○ | | ○ | ○ | | (6) |
| ○ | ○ | | | | (8) |
| ○ | | | ○ | ○ | (3) |
| ○ | | ○ | | | (4) |
| ○ | | ○ | | ○ | (5) |
| | | ○ | ○ | | ("szó vége") |

alakban perforáljuk, a bevétel csak az a.) módszer szerint lehetséges.

2.) A binárisan megadott 7641053214 szót vigyük be a 412 címre (b.) módszer).

| | | | | | |
|---|---|---|---|---|--------------|
| ○ | | | | | (0) |
| ○ | | ○ | | | (4) |
| ○ | | | | ○ | (1) |
| ○ | | | ○ | | (2) |
| | | | | ○ | ("cím vége") |
| ○ | ○ | ○ | ○ | | (+) |
| ○ | | ○ | ○ | ○ | (7) |
| ○ | | ○ | ○ | | (6) |
| ○ | | ○ | | | (4) |
| ○ | | | | ○ | (1) |
| ○ | | | | | (0) |
| ○ | | ○ | | ○ | (5) |
| ○ | | | ○ | ○ | (3) |
| ○ | | | ○ | | (2) |
| ○ | | | | ○ | (1) |
| ○ | | ○ | | | (4) |
| | | ○ | ○ | | ("szó vége") |

A számok kiírása, ugyancsak a C-regiszteren keresztül történik, tízes vagy "nyolcas" számjegyként. A C-regiszter első négy, illetve három helyértéke van összeköttetésben a kimenő távgépíróval, és a C-regiszter tartalmának a fentiek szerinti balra történő léptetése biztosítja azt, hogy ide minden helyérték tartalma eljusson. Különleges áramkörök gondoskodnak a sorváltás elvégzéséről, soronként 1, 2, 3, 4 vagy 5 szó leírása után, előzetes beállítás szerint.

Amennyiben tízes számrendszerben akarjuk az eredményeket kiírni, egy "számrendszer-váltó" program segítségével gondoskodni kell arról, hogy a kiírandó szám a 45. oldalon ismertetett binárisan kódolt decimális formában kerüljön a C regiszterbe.

Mind a bevitelnél, mind a kiírásnál külön kapcsoló szabályozza, hogy a gép decimális vagy 8-as (tkp. bináris) rendszerben dolgozzon-e.

2.4. A vezérlőegység és működése.

Az eddigiekben megismertük az egyes műveleteknek, az úgynevezett gépi alpműveleteknek az elvégzését, feltételezve, hogy azok a számok, amelyekkel számolni akarunk, már a megfelelő regiszterekben vannak. A számoknak a regiszterekben való elhelyezését és általában egy gépi utasítás végrehajtásának menetét a központi vezérlő egység vezérli.

Ez három részből áll:

- 1.) impulzus elosztó
- 2.) szelekciós regiszter (SZR)
- 3.) indító regiszter (v. utasításszámláló).

Az M-3 gép utasításai a következő alakúak:

| | | | |
|-------|--------------|--------|---------|
| ± | Műveleti jel | I. cím | II. cím |
| 1 bit | 6 bit | 12 bit | 12 bit |

Az első hat helyértékben elhelyezkedő műveleti jel azt mutatja meg, hogy az adott utasítás milyen műveleti elvégzésére utasítja a gépet. A 7-18, illetve 19-30. helyértéken elhelyezkedő első, ill. második cím a memóriának azon rekeszeit adja meg, amelyeknek tartalmával végezzük el a műveleteket.

Az utasítást tartalmazó szó előjele az utasítás végrehajtását nem befolyásolja.

A központi vezérlő egység minden egyes utasítás végrehajtását nyolc ütemben végzi el. Az ütemek számlálására az impulzus elosztóban egy három triggert tartalmazó számláló van elhelyezve.

Az utasítások - mint tudjuk - a gép memóriájában vannak elhelyezve. A végrehajtásra soronkövetkező utasítás címét mindig a központi vezérlő egység 12 helyértékes indító regisztere tárolja.

Az utasítás végrehajtásának ütemei:

1. ütem: az indító regiszter tartalma átmegy a szelekciós regiszterbe (SZR) és "olvasás" utasítás megy a memória felé,

2. ütem: megtörténik az SZR-ben tárolt cím megkeresése és az olvasás; a kiolvasott szó (azaz a végrehajtandó utasítás) a C regiszterben helyezkedik el,

3. ütem: a C-regiszter első 6 helyértékének tartalma (azaz a műveleti jel) átmegy a műveleti programadó megfelelő regiszterébe, a műveleti regiszterbe, és "olvasás" parancs megy a memória felé. Ugyanakkor az indítóregiszter tartalmához 1 adódik hozzá, annak biztosítására, hogy a következő nyolcas ciklusban a következő rekeszben tárolt utasítás kerüljön végrehajtásra.

4. ütem: miután a memória az előző ütemben kiadott "olvasás" parancs hatására - megtalálta az SZR-ben lévő címnek (az utasítás első címének) megfelelő rekeszt, a C-regiszter 19.-30. helyértékei (tehát a második cím) mennek át az SZR-be. Ezután következik be az előzőleg megtalált első cím tartalmának beírása a C-regiszterbe, melynek eddigi tartalmára (az utasításra) már nincs szükség, hiszen annak már minden részét felhasználtuk, vagy tároljuk.

5. ütem; a C-regiszter tartalma (az első címről kiolvasott szám) átmegy az A-regiszterbe és "olvasás" parancs megy a memória felé.

6. ütem: az SZR-ben lévő cím (az utasítás második címe) tartalma az "olvasás" parancs hatására a memóriából a C-regiszterbe kerül.

7. ütem: a C-regiszter tartalma (a második címről kiolvasott szám) átmegy a B-regiszterbe.

0. (azaz nyolcadik) ütem: parancs megy a műveleti programadó felé a művelet elvégzésére.

A művelet elvégzése után "írás" parancs megy ki a memória felé.

A vezérlő egység aszinkron működésű, ami azt jelenti, hogy az egyes ütemeket az előző ütemekben kijelölt feladat elvégzését jelző "válasz" beérkezése indítja. Így a második, ütem csak akkor indul el, ha az első ütemben kiadott "olvasás" parancs hatására a memória már "megtalálta" az SZR-ben lévő címet, és ugyanez a helyzet a negyedik és hatodik ütemek indításával. Ugyanígy a következő utasítás végrehajtási ciklusának első ütemét a művelet befejezése után (a nulladik ütemben) kiadott "írás" utasításnak megfelelő válasz jel indítja el.

Minden egyes művelet eredménye a B vagy C regiszterben keletkezik, és a művelet befejezése után a másik regiszterbe is átmegy: így végeredményben a B és C regiszterekben egyaránt megtalálható. Ezért, amikor a művelet befejezése után "írás" parancs megy a memória fele, akkor ennek hatására a művelet eredménye íródik be az SZR-ben jelenleg lévő címre, azaz a végrehajtott utasítás második címére. Fenti típusú utasítás a gépnek mind az öt alpműveletére megadható. Ezeket a műveleti jel második három számjegye különbözteti meg egymástól. Itt az első három számjegy 0, ami azt jelzi, hogy az utasításnak a fent ismertetett alapváltozáról van szó:

| művelet szimbóluma | műveleti jel 6 jegye | röviden (8-as számrendszerben) |
|--------------------|----------------------|--------------------------------|
| + | 0 0 0 0 0 0 | 0 0 |
| - | 0 0 0 0 0 1 | 0 1 |
| : | 0 0 0 0 1 0 | 0 2 |
| x | 0 0 0 0 1 1 | 0 3 |
| Λ | 0 0 0 1 1 0 | 0 6 |

Tehát pl. a 8-as számrendszerben

0 0 0 0 1 2 0 0 3 1

alakban felírt utasítás végrehajtása azt jelenti, hogy a 012 és 0031 memória rekeszek tartalmainak összege beíródik a 0031 rekeszbe.

A fenti utasítások végrehajtása azzal a hátránnyal jár, hogy a második címre beíródó eredmény eltörli az ott tárolt számot (az adott műveletben résztvevő egyik számot). Sokszor fordul elő olyan eset, hogy a számítás során erre a számra még szükségünk van, ezért a művelet eredményét

Az utasítások végrehajtásának általános menete.

| ütem | Indító regiszter | Szelekciós regiszter | Olvasás parancs | C-regiszter | Műveleti regiszter | A-regiszter | 0-regiszter |
|------|---------------------------------|-----------------------------|-----------------|---------------------------|-----------------------------|----------------------|-------------------------|
| 0 | végrehajtandó utasítás címe | 0 | | (előző művelet eredménye) | (előző művelet jele) | 0 | előző művelet eredménye |
| 1 | ” | végrehajtandó utasítás címe | van | ” | ” | 0 | ” |
| 2 | ” | 0 | | végrehajtandó utasítás | ” | 0 | ” |
| 3 | végrehajtandó utasítás címe + 1 | Az utasítás első címe | | ” | az utasítás műveleti jele. | 0 | ” |
| 4 | ” | Az utasítás második címe | | az első cím tartalma | ” | 0 | ” |
| 5 | ” | ” | van | ” | ” | az első cím tartalma | ” |
| 6 | ” | ” | | az második cím tartalma | ” | ” | ” |
| 7 | ” | ” | | ” | ” | ” | A második cím tartalma |
| 0 | végrehajtandó utasítás címe + 1 | 0 | | a művelet eredménye | a végrehajtott művelet jele | 0 | művelet eredménye |

valahova máshova kell beírni.

A gépen minden egyes műveletnél lehetőség van a művelet végrehajtása után következő "írás" utasítás mellőzésére is.

Az ilyen utasítások végrehajtása után tehát a művelet befejezését jelző "válasz" azonnal indítja a következő ciklust. Ezeket az utasításokat a művelet szimbóluma után tett vesszővel (,) jelöljük, és a műveleti jel harmadik helyértékére írt 1-gyel különböztetjük meg az előzőktől.

| művelet szimbóluma | műveleti jel 6 jegye | röviden (8-as számrendszerben) |
|--------------------|----------------------|--------------------------------|
| +, | 0 0 1 0 0 0 | 10 |
| -, | 0 0 1 0 0 1 | 11 |
| :/, | 0 0 1 0 1 0 | 12 |
| x, | 0 0 1 0 1 1 | 13 |
| ^, | 0 0 1 1 1 0 | 16 |

Természetesen minden ilyen "vesszős" utasítás után gondoskodni kell arról, hogy a művelet eredménye, - amely mint tudjuk a B és C-regiszterekben maradt meg, - valamilyen módon felhasználható legyen. Ha t.i. a "vesszős" utasítás után az eddig ismert utasítások valamelyikét írják, akkor a művelet eredménye elveszne, mert a következő utasítás végrehajtása során a C regisztert maga az utasítás, majd az első és második cím tartalma töltene meg, és az utóbbi a B regiszterbe is átmenne. Ezért szükség van olyan utasításokra, amelyek a B regiszter tartalmát változatlanul hagyják, és így az előző művelet eredményével végeznek műveleteket. (A művelet eredményét a C regiszterben nem lehetne változatlanul hagyni, mert erre a regiszterre az utasítás és az első cím tartalmának kiolvasásánál szükség van!) Az ilyen utasítások végrehajtásánál tehát az ötödik ütemben a központi

vezérlő egység nem ad parancsot a második cím tartalmának kiolvasására, és a hetedik ütemben - ahelyett, hogy a C-regiszter tartalma menne át a B-be, - a B-regiszterben tárolt előző eredmény megy át a C-regiszterbe, és így az utasításban kijelölt műveletet a gép ezen szám és az utasítás első címének - az ötödik ütemben a regiszterbe vitt - tartalma között végzi el, és a művelet eredményét a nulladik ütem "írás" parancsa alapján beírja a második címre. Ezek az utasítások természetesen olyan utasítások után is használhatók, melyeknél a művelet eredménye beíródott a második címre, hiszen a B regiszterben az eredmény ebben az esetben is megmarad. Ilyen utasítás is minden gépi alapműveletnél található, ezeket a művelet jelzése elé tett nyíllal, illetve a műveleti jel második számjegyébe írt számmal, 1-el jelöljük:

| művelet szimbóluma | műveleti jel 6 jegye | (röviden 8-as számrendszerben) |
|--------------------|----------------------|--------------------------------|
| ↓+ | 0 1 0 0 0 0 | 2 0 |
| ↓- | 0 1 0 0 0 1 | 2 1 |
| ↓: | 0 1 0 0 1 0 | 2 2 |
| ↓x | 0 1 0 0 1 1 | 2 3 |
| ↓∧ | 0 1 0 1 1 0 | 2 6 |

Lehetőség van a fent említett két eset összekapcsolására is: A nyíllal és vesszővel ellátott utasításoknál a művelet az előző művelet eredménye és az első cím tartalma között végezzük el, és az eredmény nem íródik be a memóriába, csak a B regiszterben marad meg. Ezeknél az utasításoknál a második címre bármit írhatunk, hiszen az nem kerül felhasználásra.

| művelet szimbóluma | műveleti jel 6 jegye | (röviden 8-as számrendszerben) |
|--------------------|----------------------|--------------------------------|
| ↓+, | 0 1 1 0 0 0 | 3 0 |
| ↓-, | 0 1 1 0 0 1 | 3 1 |
| ↓:, | 0 1 1 0 1 0 | 3 2 |
| ↓x, | 0 1 1 0 1 1 | 3 3 |
| ↓∧, | 0 1 1 1 1 0 | 3 6 |

Eddig még nem használtuk fel a műveleti jel első számjegyet. Ennek szerepe a következő:

1. ha a műveleti jel első számjegye 1 és a harmadik számjegye 0, akkor a gép a művelet eredményét, - amellett, hogy a második címre beírja - ki is nyomtatja a távgép-író segítségével. Ezt a művelet jelzése utáni "n" betűvel jelöljük.
2. ha a műveleti jel első és harmadik számjegye 1, akkor a gép a műveletet a megfelelő számok helyett azok abszolút értékeivel végzi el. Ezt úgy jelöljük, hogy a művelet szimbólumát két függőleges vonal közé írjuk (pl. |+,).

Mindkét esetben természetesen a műveletet - a második helyértéken lévő 0, illetve 1 szerint - a két cím tartalma illetve az előző művelet eredménye és az első cím, tartalma között végezzük el.

Összefoglalva: minden egyes művelethez az alábbi nyolc utasítás tartozik (* az öt gépi "alpművelet" bármelyikét jelentheti).

| Művelet szimbóluma | Műveleti jel első 3 számjegye | A műveletben résztvevő számok | | az eredmény elhelyezése | |
|--------------------|-------------------------------|-------------------------------|--|-------------------------|---------------------------------------|
| * | 0 0 0 (0) | első cím | második tartalma | B, C regiszterbe | második címre |
| *, | 0 0 1 (1) | " | " | " | / |
| ↓* | 0 1 0 (2) | első cím tartalma | B-regiszter tartalma (előző műv. eredménye). | " | második címre |
| ↓*, | 0 1 1 (3) | " | " | " | / |
| *n | 1 0 0 (4) | első cím tartalma | második tartalma | " | második címre és kiírás a távgépíron. |
| * , | 1 0 1 (5) | első cím abszolút értéke | második tartalmának abszolút értéke | " | / |
| ↓*n | 1 1 0 (6) | első cím tartalma | B-regiszter tartalma (előző műv. eredménye). | " | második cím és kiírás a távgépíron |
| ↓ * , | 1 1 1 (7) | első cím abszolút értéke | B-regiszter tartalmának abszolút értéke | " | / |

A műveleti jel utolsó három bináris - azaz második 8-as számjegye a megadott műveletet jelöli. Láttuk, hogy

0 0 0 (0) = összeadás (+)
 0 0 1 (1) = kivonás (-)
 0 1 0 (2) = osztás (:)
 0 1 1 (3) = szorzás (x)
 1 1 0 (6) = logikai szorzás (∧)

A műveleti jel második nyolcas helyértékén álló 7 számjegy (111) a gép beviteli utasításait jelzi (lásd a bevitel a.) módszerét az előző előadáson). Ezeknél a gép a szokásos módon elvégzi a nyolcas ciklust, majd a 07 és 27 műveleti jelek esetén a nulladik ütemben - a kijelölt művelet végrehajtása gyanánt - beviszi a perforált szalagon a soron-

következő szót az utasítás második címére (a ciklus végrehajtása során, ugyanis ez kerül utoljára az SZR-be).

A beviteli utasítások szimbóluma "Be". A 07 és 27 műveleti jelek teljesen egyenértékűek.

Itt kell megemlítenünk azt is, hogy a gép jelenlegi állapotában a műveletek közül csak összeadás és kivonás után végezhető kiírás, tehát a 42, 43, 46, ill. 62, 63, 66 utasítások nem használhatók.

A 05, 15, 45 és 55 jelű utasítások a számoknak a memória egyik rekeszéből a másikba való átadását végzik. Ezek végrehajtásánál az ötödik ütem "olvasás" parancsa helyett egy "írás" parancs megy ki, ennek megfelelően a hatodik ütemben megtörténik a C regiszterben lévő számnak - azaz az első cím tartalmának, amely a negyedik ütemben C-be került - beírása az SZR-ben lévő második címre. Ezután a hetedik ütemben a C tartalmának B-be való szokásos átvitele biztosítja, hogy a "művelet eredménye", az a szám tehát, amelyet az utasítás áthelyezett - a B regiszterben is megmaradjon és így a következő utasítás során felhasználható legyen.

Közben természetesen - mint a gépen minden olvasásnál és átadásnál - az átvitt szám az eredeti helyén, jelen esetben az első címen is megmarad.

2.5. Vezérlés-átadó utasítások.

A gép a program utasításait általában címeik sorrendjében hajtja végre (ezért növeli 1-gyel minden ciklusban az indítóregiszter tartalmát). Néha szükség van azonban arra, hogy eltérjünk ettől a sorrendtől, és előírjuk, hogy valamely utasítás végrehajtása után a gép a számítást egy tetszőleges, megadott utasítás végrehajtásával folytassa. Azokat az utasításokat, amelyekkel ezt elvégezzük vezérlés-átadó vagy ugróutasításoknak nevezzük. Ezeknél az utasítás végrehajtásának alapciklusa úgy módosul, hogy elmarad az

első cím tartalmának kiolvasása, ehelyett az első cím a szelekciós regiszterből (ahova a harmadik ütemben átment) átmegy az indítóregiszterbe, és így a következő ciklusban az ezen címen tárolt utasítás kerül végrehajtásra.

Így történik minden olyan utasítás végrehajtása, ahol a műveleti jel második (nyolcas) számjegye 4. Ezek közül azokat használjuk, amelyeknél a műveleti jel második bináris számjegye 1-es (ezek a műveletnél a "nyilas" utasításoknak felelnek meg), tehát itt a második cím tartalmának kiolvasása is elmarad, és a hetedik ütemben B tartalma megy át a C-be és nem fordítva.

Ezek:

| | |
|----|----------|
| 24 | (010100) |
| 34 | (011100) |
| 64 | (110100) |
| 74 | (111100) |

A 24 és 64 jelű utasítások végrehajtásánál a ciklus fenti módosításain kívül a hetedik ütem végén, miután a B regiszter tartalma átmegy a C-be, egy "írás" parancs megy ki, és a 64 jelű utasításnál még egy kiírási parancs is a távgépíró felé. Ezek az utasítások tehát:

1. az indítóregiszterbe beírják az első címet azaz „átadják a vezérlést” az ezen a címen tárolt utasításnak;
2. az előző művelet eredményét beírják a második címre (64 még ki is íratja a távgépírón).

A 24 jelű utasítás szimbóluma U1, a 64-é U1n, jelezve, hogy az első címen tárolt utasításra való "ugrásról" van szó.

A 34 és 74 jelű utasítások végrehajtásánál a hetedik ütemben, "írás" parancs nem megy ki, ehelyett, amennyiben a B regiszter tartalmának előjele pozitív, az SZR tartalma átmegy az indítóregiszterbe. Ez azt jelenti, hogy ezeknél a vezérlésátadást B előjelétől függően vagy az első, vagy

a második címre végzi el a gép: ha B előjele negatív, akkor a negyedik ütemben bevitt első cím marad az indítóregiszterben és ez lesz a következő végrehajtandó utasítás címe, ha B előjele pozitív, akkor ezt a hetedik ütemben a második címre változtatja át. Ezenkívül tekintve, hogy ezeknél az utasításoknál semmiféle műveletet, vagy írást nem végzünk, ami a következő utasítást indítana, a nulladik ütemben a gép indítja a következő ciklus első ütemét.

A 74 jelű utasítás végrehajtásánál - tekintve, hogy az ismertetett eltérésektől eltekintve a nyolcas ciklust a gép úgy végzi el, mint más olyan utasításoknál, melynek első számjegye 7, ezek pedig a számok abszolút értékével végeznek műveleteket - a gép az A és B regiszterek előjelét pozitívrá változtatja, így B előjele mindenképpen pozitív lesz, tehát a másik cím kerül az indítóregiszterbe. Ezért ez az utasítás a 24 és 64 jelűekhez hasonlóan - feltétlen ugrást jelent, mégpedig ezektől eltérően a második címre. Szimbóluma: U2.

Ezzel szemben a 34-es utasításnál a gép feltételes ugrást végez az előző művelet eredményének előjele szerint, vagy az első, vagy pedig a második címen tárolt utasításra. Szimbóluma: FU.

Ha az előző művelet eredménye nulla, akkor a helyzet a következő: Ha valamely művelet eredménye nulla, ennek előjele lehet pozitív is, negatív is (pozitív, ill. negatív 0). Láttuk ugyanis az egyes műveleteknél, hogy a gép a számok abszolút értékét és előjeleit külön kezeli, pl. kivonásnál (ill. különböző előjelű számok összeadásánál) B előjelét csak akkor kellett megváltoztatni, ha a kivonandó abszolút értékben nagyobb volt a kisebbítendőnél. Így ha a két szám megegyezik, akkor B előjelében megmarad a második címen lévő szám - a kisebbítendő - előjele. Tehát:

$$\begin{aligned} (+a) - (+a) &= (+a) + (-a) = +0 \\ (-a) - (-a) &= (-a) + (+a) = -0 \end{aligned}$$

ahol az első helyen álló szám van a második címen (illetve "nyilas" műveleteknél ez az „előző művelet eredménye”), mert ez van a B regiszterben.

Szorzásnál és osztásnál a 0-eredmény előjelét is az ismert előjelszabály adja meg, míg a logikai szorzásnál, mely az előjelet nem változtatja, az eredmény előjele mindenképpen a B regiszterben elhelyezett tényező előjelével egyezik meg.

Abban az esetben tehát, ha a B regiszter tartalma 0, akkor a feltételes ugrást a gép a 0 szám fenti értelemben vett előjele szerint végzi el.

A 04, 14, 44 és 54 jelű utasítások nyolcas ciklusának végrehajtása is a 4-jelű utasítások fent ismertetett végrehajtása szerint történik, ezeknél azonban a következő utasítás végrehajtását semmi nem indítja el, ezek tehát a gép megállási utasításai. Ugyancsak megállást eredményeznek a 17, 37, 57 és 77 jelű utasítások is. A megállás után az egyes regiszterek tartalma a végrehajtott nyolcas ciklusnak megfelelően más és más lesz.

A megállási utasításokon kívül természetesen megállítható a gép a vezérlőtábla megfelelő kapcsolójával is. Ezenkívül van az M-3 gépen egy olyan megállítási lehetőség is, hogy beállítunk egy címet a vezérlőtábla megfelelő 12 helyértékes "beállítóregiszter"-ében és a gép megáll, mihamarabb az indítóregiszterben vagy a szelekciósregiszterben ez a cím megjelenik (azt, hogy melyik regiszter szerint áll meg, egy külön kapcsoló szabályozza).

Így tehát megállíthatjuk a gépet egy adott utasítás első végrehajtásánál, vagy egy adott rekesz első felhasználásánál. Ez a megállítási lehetőség bonyolult programok gépi ellenőrzésénél hasznos.

2.6. Az utasítások végrehajtásának időtartama.

Láttuk, hogy az egyes utasítások végrehajtásának időtartamát lényegében a memóriához való fordulások (írás, olvasás) száma határozza meg: minden ilyen memóriához fordulás átlagosan 10 msec-ot vesz igénybe.

A memóriához való fordulásokat az egyes utasításoknál az alábbi táblázat szemlélteti (x jelenti a 0, 1, 2, 3 és 6 számok valamelyikét).

| műveleti jel | memóriához való fordulások. | végrehajtás kb. ideje msec-ben. |
|--------------|--|---------------------------------|
| 0 x | utasítás első cím } kiolvasása második cím } eredmény beírása | 40 |
| 1 x 5 x | utasítás első cím } kiolvasása második cím } | 30 |
| 2 x | utasítás } kiolvasása első cím } eredmény beírása | 30 |
| 3 x 7 x | utasítás } kiolvasása első cím } | 20 |
| 24 | utasítás kiolvasása előző művelet eredményének beírása. | 20 |
| 34 74 | utasítás kiolvasása | 10 |
| 05 15 | utasítás } kiolvasása első cím } Az első cím tartalmának beírása | 30 |

Azoknak az utasításoknak a végrehajtási idejét, melyeknél számok bevitele vagy kiírása történik (07, 27, ill. 3x, 6x, 64, 45, 55) a bevétel, ill. kiírás sebessége határozza meg (kb. 2 sec).

3. Fejezet.

A PROGRAMKÉSZÍTÉS GYAKORLATI MÓDSZEREI.

3.1. A direkt programozás bemutatása példákon.

A korábbi fejezetekben megismertük az M-3 gép utasításrendszerét. Most olyan egészen egyszerű feladatok programozását mutatjuk be, amelyeknél a gép az utasításokat a tárolás sorrendjében hajtja végre, azaz nincsenek ugró utasítások. Ezt nevezzük direkt programozásnak.

Az itt közölt példák hosszabb programok kivágott részletei, ahol feltételezzük, hogy az utasítások, konstansok és a számadatok a memória ismert helyein vannak, az eredmények további felhasználás céljából szintén a memóriában maradnak és az indító regiszter éppen azt a címet tartalmazza, ahol a vizsgált részletprogram kezdődik. Az adatok és a program bevitele az M-3 gépen vagy cím megadásával történik (ezzel a módszerrel már megismerkedtünk), vagy külön program segítségével, amelyről majd később lesz szó. Az eredmények kinyomtatásáról a program folyamán külön kell gondoskodni. Ez az u.n. nyomtató utasítások segítségével történik. Felhívjuk a figyelmet arra, hogy a programot, tehát a címeket és az utasítások kódjeleit nyolcas számrendszerben kell megadni, hogy közvetlenül bevihetők legyenek a gépbe.

Tudjuk, hogy az M-3 gép csak 1-nél kisebb abszolút értékű számokkal tud dolgozni. Programozásnál ügyelni kell arra, hogy mind az eredmény, mind a részleteredmények a $(-1, 1)$ intervallumba kerüljenek. Ennek érdekében bizonyos numerikus átalakításokat kell végezni, amelyekről részletesen a későbbiekben lesz szó. Feltételezzük, hogy a most következő példákban ez a követelmény ki van elégítve.

Mielőtt a programozáshoz hozzákezdenénk, néhány jelölést kell bevezetni.

Jelölések:

1. Valamely x cím tartalmát (x) -szel jelöljük. Tehát $(x) = a$ azt jelenti, hogy az a mennyiséget az x címen találhatjuk a memóriában.
2. Valamely mennyiség címét $\langle a \rangle$ -val jelöljük. Tehát a fenti mennyiségek esetében $\langle a \rangle = x$.
3. Valamely mennyiségnek egy adott címre való elhelyezését a "=>" jellel jelöljük. Pl.: $a+b=>n$ azt jelenti, hogy az a és b mennyiség összegét tegyük az n címre.

Az áttekinthetőség kedvéért a program összeállításánál a következő táblázatot használjuk:

PR

| utasítás rekesze | a művelet szimbóluma | jelle | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|----------------------|-------|-----------------|--------|-----------------|--------|
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |

A program összeállítását az (1), (2), (4), (6) oszlopok kitöltésével kezdjük. Utána az (5), (7) oszlopokat töltjük ki attól függően, hogy a konstansokat és számadatokat hová tesszük a memóriában, végül a (3) oszlopot töltjük ki a gép utasításrendszere alapján. A gépbe csak az (1), (3), (5), (7) oszlopok kerülnek. Az utasítás a (3), (5), (7) oszlop tartalma, az indítóregiszter tartalma az (1) oszlopban álló szám, amikor a gép végrehajtja az illető utasítást. Cím megadásával történő bevétel esetén ezt a perforált szalagra kell lyukasztani; program segítségével történő bevételnél a program gondoskodik az utasítások megfelelő elhelyezéséről.

Most áttérünk egyszerű példák kidolgozására. Tegyük fel, hogy az a,b,c,d számok rendre a 0010, 0012, 0013 címeken vannak

Azaz

<a> = 0010 <c> = 0012
 = 0011 <d> = 0013

illetőleg

(0010) = a (0012) = c
(0011) = b (0013) = d

A későbbiekben inkább az utóbbi felírást használjuk. A számítások végeredményét a 0020-as rekeszbe helyezzük el. A programokat a 0100-as memória rekesztől kezdve tároljuk és feltételezzük, hogy az indítóregiszter tartalma 0100, azaz a gép éppen az ezen a címen tárolt utasítás végrehajtását kezdi el.

1. Példa.

$$x = a + b + c + d \implies 0020$$

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | +, | 10 | a | 0010 | b | 0011 |
| 0101 | ↓+, | 30 | c | 0012 | - | - |
| 0102 | ↓+ | 20 | d | 0013 | x | 0020 |

A 0100-as utasításban azért használunk "+,-"-t, mivel lehetséges, hogy a számítás folyamán a b számra még szükség lesz, tehát nem törölhetjük ki a 0011 rekeszből. Azonkívül az a+b részleteredményre a továbbiakban nem lesz szükségünk, tehát nem kell tárolni a memóriában. Továbbá, amint már láttuk, a "+,-" utasítás kevesebb időt vesz igénybe, mint a "+". A 0101-es utasításban a "↓+,-" utasítást használunk. Az előző lépés következtében ennek az utasításnak szükségképpen

”nyilas” utasításnak kell lennie, mivel az előző művelet eredményével számolunk tovább, amely az előző ”+,” utasítás miatt a memóriában nem található. Mivel pedig erre a részleteredményre sem lesz a továbbiakban szükségünk, ”↓+,” utasítást célszerű használni. Említettük már, hogy az ilyen típusú utasítás veszi a legkevesebb időt igénybe, tehát használata igen előnyös.

A 0102-es utasítás egyértelműen meg van határozva: nyilasnak kell lennie, mert a ”B” regiszter tartalmával dolgozik, de nem lehet vesszős, mert ez már az eredményt adja, amelyet tárolni kell a memóriában.

2. Példa.

$$x = a + bd - c \implies 0020$$

Itt célszerű a műveletek sorrendjét felcserélni és a számítást a következő sorrendben végrehajtani: $bd + a - c = x$.

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | x, | 13 | b | 0011 | d | 0013 |
| 0101 | ↓+, | 30 | a | 0010 | - | - |
| 0102 | ↓- | 21 | c | 0012 | x | 0020 |

Ennek a példának a felépítése teljesen hasonló az előzőhöz, azzal a különbséggel, hogy ott mindhárom utasításban összeadás szerepel, itt pedig az elsőben szorzás, a harmadikban kivonás.

3. Példa.

$$x = a + \overset{b+b}{2} b + \overset{c+c+c}{3} c \implies 0020$$

Mivel a 2 és 3 számokat a gépben nem tudjuk ábrázolni, ezeket a szorzásokat úgy végezzük el, hogy $\frac{1}{2}$ -del, illetve $\frac{1}{3}$ -dal osztunk. Legyen $(0014) = 1/2$ és $(0015) = 1/3$

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|---------------------|------------|------|--------------------|--------|--------------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | : | 12 | 1/2 | 0014 | b | 0011 |
| 0101 | U1 | 24 | - | 0102 | 2b | 0020 |
| 0102 | : | 12 | 1/3 | 0015 | c | 0012 |
| 0103 | ↓+, | 30 | 2b | 0020 | - | - |
| 0104 | ↓+ | 20 | a | 0010 | - | 0020 |

Ez a példa két szempontból is tanulságos volt. Először is láttuk, hogy 2 b kiszámítása után ezt a mennyiséget félre kellett tenni, míg 3c-t kiszámítjuk. Ebből a célból felhasználunk egy olyan memóriarekeszt, amelynek tartalmára más célból nincs szükség. Az ilyen memóriarekeszt "munkarekesz"-nek nevezzük.

Másodszor a 2b mennyiséget a 0101-es utasítással helyezzük el a 0020-as rekeszbe. Nézzük meg ezt az utasítást közelebbről. Ez egy feltétel nélküli ugró utasítás, amelynek első címében annak az utasításnak a címe áll, ahová ugrani kell, második címében pedig annak a memóriarekesznek a címe, ahova az előző művelet eredményét be kell írni. A mi esetünkben csak az utasítás második címére van szükség, ezért az első címre a programban soron következő utasítás rekeszének címét írjuk. Mindig így járunk el, ha egy olyan mennyiséget akarunk elhelyezni a memóriába, amelynek kiszámításánál az utolsó utasítás második címe egy olyan mennyiség rekeszének címe, amelyet nem szabad törölni. Ez esetben az utolsó utasítás vesszős utasítás és külön utasítás kell az eredmény elhelyezésére. Elvégezhetnék ezt úgy is, hogy a kiszámított mennyiséghez, amely a B regiszterben található, 0-t adunk hozzá egy olyan "↓+" utasítás segítségével, amelynek a második címe jelzi, hová akarjuk vinni ezt a mennyiséget. Ez a módszer több időt vesz igénybe,

mivel a 0-t elő kell venni a memóriából és az összeadást elő kell végezni. Az "U1" utasítás sokkal gyorsabban hajtható végre.

4. Példa.

$$x = ad - cd \implies 0020$$

a.) változat

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | x, | 13 | c | 0012 | d | 0013 |
| 0101 | U1 | 24 | - | 0102 | cd | 0020 |
| 0102 | x, | 13 | a | 0010 | d | 0013 |
| 0103 | ↓- | 21 | cd | 0020 | x | 0020 |

A műveletek sorrendjét itt is fel kell cserélni. A kivonandó szorzatot célszerű először kiszámítani, mivel a kivonás nem kommutatív művelet, és csak abban a sorrendben lehet nyílas műveletet használni.

b.) változat.

végezzük elő a következő átalakítást. Emeljük ki d-t:

$$ad - cd = (a-c) d.$$

végezzük elő így a programozást.

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | -, | 11 | c | 0012 | a | 0010 |
| 0101 | ↓x | 23 | d | 0013 | x | 0020 |

Látjuk, hogy az átalakítás segítségével az utasítások száma felére csökkent. Nagyon fontos szempont a programozásnál, hogy a feladatot mindig igyekezzünk olyan alakra hozni, hogy a program a lehető legrövidebb legyen, és ezért a legkisebb tároló kapacitást igényelje. (A másik szempont az időbeli rövidség. E két szempont a legtöbbször ellentmondó, amint a későbbiekben látni fogjuk.)

5. Példa.

$$x = (ab + c) (a - bc) \implies 0020$$

Ezt a feladatot is kétféleképpen oldjuk meg.

a.) változat.

0021 = munkarekesz.

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | x, | 13 | a | 0014 | b | 0011 |
| 0101 | ↓+ | 20 | c | 0102 | ab+c | 0020 |
| 0102 | x, | 13 | b | 0015 | c | 0012 |
| 0103 | U1 | 24 | - | 0020 | bc | 0021 |
| 0104 | -, | 11 | bc | 0020 | a | 0010 |
| 0105 | ↓x | 23 | ab+c | 0010 | x | 0020 |

b.) változat.

Átalakítjuk a feladatot.

$$(ab + c) (a - bc) = (ab + c) a - (ab + c) bc.$$

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | x, | 13 | a | 0010 | b | 0011 |
| 0101 | ↓+ | 20 | c | 0012 | ab+c | 0020 |
| 0102 | x, | 13 | b | 0011 | c | 0012 |
| 0103 | ↓x | 23 | ab+c | 0020 | (ab+c)bc | 0021 |
| 0104 | x, | 13 | a | 0010 | ab+c | 0020 |
| 0105 | ↓- | 21 | (ab+c)bc | 0021 | x | 0020 |

Látjuk, hogy ebben az esetben nincs sok különbség a két változat között. Az utasítások száma mindkét esetben megegyezik és mindkét esetben fel kell használni egy új munkarekeszt. Az a.) változatban az egyik utasítás "U1", tehát ezt a változatot a gép valamivel gyorsabban hajtja végre.

6. Példa.

$$x = \frac{abc+d}{4(a+b)} \implies 0020 \quad (0016) = 1/4$$

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | +, | 10 | a | 0010 | b | 0011 |
| 0101 | ↓: | 22 | 1/4 | 0016 | 4(a+b) | 0020 |
| 0102 | x, | 13 | a | 0010 | b | 0011 |
| 0103 | ↓x, | 33 | c | 0012 | - | - |
| 0104 | ↓x, | 30 | d | 0013 | - | - |
| 0105 | ↓: | 22 | 4(a+b) | 0020 | x | 0020 |

A 0103 és 0104 utasításokban nincs kihasználva, hogy a gép kétcímű, viszont ezek az utasítások hajthatók a leggyorsabban végre, mivel csak egyszer kell a memóriához fordulni.

7. Példa.

$$x = -a - b + |c| - |d| \implies 0020$$

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| (1) | (2) | (3) | (4) | (5) | (6) | (7) |
| 0100 | - , | 51 | d | 0013 | c | 0012 |
| 0101 | ↓-, | 31 | a | 0010 | - | - |
| 0102 | ↓- | 21 | b | 0012 | x | 0020 |

Itt felhasználjuk az abszolút értékkel végzett műveleteket.

Feladat:

Készítsük el a 3. példa programját úgy, hogy ne használjuk az 1/2 és 1/3 konstansokat.

3.2. Direkt programozás feltételes ugró utasításokkal.

Példák.

Eddig csak a szorosan vett direkt programozással foglalkoztunk. Most olyan programokat fogunk készíteni, amelyekben előfordulnak ugyan ugró utasítások, de u.n. ciklusok nem. (Ciklus a program olyan része, amely ugró utasítások segítségével egymásután többször kerül végrehajtásra.) Olyan programokat, amelyekben ciklusok is vannak, csak a következő pontban ismertetünk.

Bonyolultabb programoknál célszerű a számítás menetét u.n. menetrend formájában megtervezni. A menetrenddel a példák során ismerkedünk meg.

1. Példa.

Adva van három szám. Ha mindegyik különbözik nullától, akkor adjuk őket össze. Ellenkező esetben állítsuk meg a gépet.

Tervezzük meg először is a számítás menetrendjét. (L. a I. táblát.)

A menetrend a géptípustól független. Az utasításrendszer jellege alig befolyásolja. Az összetartozó utasításcsoportokat egy blokkba írjuk.

Ha a gép egy utasításcsoport eredményétől függően két lehetőség között választ, az utasításcsoportnak megfelelő blokkot egy csúcsra állított rombuszal ábrázoljuk. Az ilyen "választó" blokknak mindig két kimenete van. A választó blokkba beírjuk azt a logikai feltételt, amely szerint a gép választ, kimeneteinél pedig megjelöljük, hogy milyen döntésnél merre halad tovább a program végrehajtásában.

A megállás blokkját körrel jelöljük, ennek természetesen nincs kimenete.

Az M-3 gépen a feltételes ugró utasítás, az „FU” úgy választ két lehetőség között, hogy megvizsgálja a B regiszter tartalmának előjelét. A mi példánkban azonban nem előjel szerint kell választani, hanem aszerint, hogy a B regiszter tartalma 0 vagy sem. Azt, hogy a gép választani tudjon, úgy érhetjük el, hogy a B regiszter tartalmának abszolút értékét kivonjuk nullából. Az eredmény akkor és csak akkor lesz negatív, ha a B regiszter tartalma nem volt 0.

Újból felhívjuk a figyelmet az M-3 gép egy különös sajátosságára. Ha valamely művelet eredménye 0, akkor ez a nulla pozitív vagy negatív lehet a következők szerint:

- a.) szorzásnál és osztásnál az előjel-szabály érvényesül (egy művelet eredménye a gép szerint akkor is nullával egyenlő, ha abszolút értékben kisebb, mint 2^{-30});
- b.) összeadásnál, kivonásnál és logikai szorzásnál a nulla a második címen lévő szám előjelét kapja;
- c.) abszolút értékes műveleteknél mindig pozitív.

A mi esetünkben a nulla mindig pozitív. Készítsük el a programot a már ismertetett menetrend alapján.

Program.

(0010)=a; (0011)=b; (0012)=c; (0013)=0; $x=a+b+c \implies 0020$.
A programot a 0100 memóriarekesztől kezdve helyezzük el.

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| 0100 | - , | 51 | a | 0010 | 0 | 0013 |
| 0101 | FU | 34 | - | 0102 | - | 0106 |
| 0102 | - , | 51 | b | 0011 | 0 | 0013 |
| 0103 | FU | 34 | - | 0104 | - | 0106 |
| 0104 | - , | 51 | c | 0012 | 0 | 0013 |
| 0105 | FU | 34 | - | 0107 | - | 0106 |
| 0106 | megállás | | | | | |
| 0107 | +, | 10 | a | 0010 | b | 0011 |
| 0110 | ↓+ | 20 | c | 0012 | x | 0020 |

2. Példa

Számítsuk ki az $y = \frac{A+BX}{C+DX} + \frac{x^2}{2}$ függvény helyettesítési értékét valamely $x = x_0$ helyen. Ha $C + Dx_0 = 0$, akkor állítsuk meg a gépet. A menetrendet a II. táblán ábrázoljuk.

Program:

(0010) = x_0 ; (0013) = C;
 (0011) = A; (0014) = D;
 (0012) = B; (0015) = 2^{-30} ;
 (0016) = 1/2
 $y_0 \implies 0020$

Munkarekesz: 0020

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-------------------------|--------|-------------------------|--------|
| | szimbóluma | jele | | | | |
| 0100 | x, | 13 | D | 0014 | x_0 | 0010 |
| 0101 | ↓+ | 20 | C | 0013 | DX_0+C | 0020 |
| 0102 | ↓ - , | 71 | 2^{-30} | 0015 | - | - |
| 0103 | FU | 34 | - | 0104 | - | 0105 |
| 0104 | megállás | | | | | |
| 0105 | x, | 13 | B | 0012 | x_0 | 0010 |
| 0106 | ↓+, | 30 | A | 0011 | - | - |
| 0107 | ↓:, | 22 | $C+DX_0$ | 0020 | $\frac{A+BX_0}{C+DX_0}$ | 0020 |
| 0110 | x, | 13 | x_0 | 0010 | x_0 | 0010 |
| 0111 | ↓x, | 33 | $1/2$ | 0016 | - | - |
| 0112 | ↓+ | 20 | $\frac{A+BX_0}{C+DX_0}$ | 0020 | y_0 | 0020 |

Ebben a példában sincs baj a negatív nullával, hiszen abszolút-értékes művelet segítségével választunk.

3. Példa.

$\text{Max}(a,b,c) \implies 0020$, azaz az a,b,c , számok közül a legnagyobbat helyezzük el 0020-ban. Tervezzük meg itt is a számítás menetrendjét. (L. a XII. táblát.)

A menetrend szerint először megnézzük, hogy a és b közül melyik a nagyobb. A nagyobb számot jelöljük y -nal.

Azaz

$$y = \begin{cases} a & \text{ha } a > b \\ b & \text{ha } a < b \end{cases}$$

A következő lépésnél y -t összehasonlítjuk c -vel és 0020-ba az y -t vagy a c -t írjuk be, aszerint, hogy a kettő közül melyik a nagyobb. Azaz $\text{max}(y,c) \implies 0020$. A választást itt is az "FU" feltételes ugró utasítással hajtjuk végre.

Ezt ebben a feladatban közvetlenül alkalmazhatjuk, hiszen itt negatív és pozitív szám között kell választani. A negatív nullával most sem lehet baj; ha t.i. a két szám egyenlő, teljesen mindegy, hogy melyik számot "tekintjük nagyobbnak".

Program:

$$(0010) = a; (0011) = b; (0012) = c.$$

A programot a 0100 memóriarekesztől kezdve helyeztük el.

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------|--------|
| | szimbóluma | jele | | | | |
| 0100 | -, | 11 | b | 0011 | a | 0010 |
| 0101 | FU | 34 | - | 0104 | - | 0102 |
| 0102 | -, | 11 | c | 0012 | a | 0010 |
| 0103 | FU | 34 | - | 0112 | - | 0106 |
| 0104 | -, | 11 | c | 0012 | b | 0011 |
| 0105 | FU | 34 | - | 0212 | - | 0111 |
| 0106 | Á | 05 | a | 0010 | x | 0020 |
| 0107 | U2 | 74 | - | | - | 0113 |
| 0110 | Á | 05 | b | 0011 | x | 0020 |
| 0111 | U2 | 74 | - | | - | 0113 |
| 0112 | Á | 05 | c | 0012 | x | 0020 |
| 0113 | | | | 0020 | | |

4. Példa.

Oldjuk meg az

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

kétismeretlenes lineáris egyenletrendszer.

A két ismeretlenre: x_1 , illetve x_2 -re a következő kifejezések adódnak:

$$x_1 = \frac{b_1 a_{22} - b_2 a_{12}}{a_{11} a_{22} - a_{12} a_{21}} ; \quad x_2 = \frac{a_{11} b_2 - b_1 a_{21}}{a_{11} a_{22} - a_{12} a_{21}}$$

Arról, hogy ezek az értékek valóban kielégítik az egyenletrendszert, behelyettesítés segítségével könnyen meggyőződhetünk.

Jelöljük a törtek nevezőjét - amely mindkét kifejezésben azonos - D -vel, x_1 számlálóját D_1 -gyel, x_2 számlálóját D_2 -vel.

Tehát

$$x_1 = \frac{D_1}{D}; \quad x_2 = \frac{D_2}{D}.$$

Ha $D=0$, azaz az egyenletrendszer nem oldható meg egyértelműen, a gépet megállítjuk.

A számítás menetrendjét a IV. táblán láthatjuk.

Program:

$$(0010) = a_{11}; \quad (0011) = a_{12}; \quad (0012) = a_{21}; \quad (0013) = a_{22};$$

$$(0014) = b_1; \quad (0015) = b_2; \quad (0016) = 2^{-30};$$

$$x_1 \implies 0020; \quad x_2 \implies 0021.$$

Munkarekesz: 0020.

A program a 0100 memóriarekesztől indul.

| utasítás rekesze | A művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------------------|--------|
| | szimbóluma | jele | | | | |
| 0100 | x, | 13 | a_{12} | 0011 | a_{21} | 0012 |
| 0101 | U1 | 24 | - | 0102 | $a_{12}a_{21}$ | 0020 |
| 0102 | x, | 13 | a_{11} | 0010 | a_{22} | 0013 |
| 0103 | ↓- | 21 | $a_{12}a_{21}$ | 0020 | $a_{12}a_{22}-a_{12}a_{21}$ | 0022 |
| 0104 | ↓ - , | 71 | 2^{-30} | 0016 | - | - |
| 0105 | FU | 34 | - | 0121 | - | 0106 |
| 0106 | x, | 13 | b_2 | 0015 | a_{12} | 0011 |
| 0107 | U1 | 24 | - | 1100 | b_2a_{12} | 0020 |
| 0110 | x, | 13 | b_1 | 0014 | a_{22} | 0013 |
| 0111 | ↓-, | 31 | b_2a_{12} | 0020 | - | - |
| 0112 | ↓: | 22 | D | 0022 | x_1 | 0020 |
| 0113 | x, | 13 | a_{21} | 0012 | b_1 | 0014 |
| 0114 | U1 | 24 | - | 0015 | $a_{21}b_1$ | 0021 |
| 0115 | x, | 13 | a_{11} | 0010 | b_2 | 0015 |
| 0116 | ↓-, | 31 | $a_{21}b_1$ | 0021 | - | - |
| 0117 | ↓: | 22 | D | 0022 | x_2 | 0021 |
| 0120 | U2 | 74 | - | - | - | 0122 |
| 0121 | megállás | | | 0020 | | |

Feladatok:

1. Készítsünk programot a szabadesés útképletéhez:

$$y = v_0t + \frac{g}{2} t^2$$

Konstansok: (0010) = v_0 ; (0011) = t ; (0012) = g ;

$$(0013) = \frac{1}{2}$$

Munkarekesz: 0021.

$$y \implies 0020$$

A programot a 0100-as címnél kezdjük.

$$2. \quad y = \frac{a + bf}{c}$$

Megvizsgálandó, hogy az osztás elvégezhető-e, azaz ha $c = 0$, a gépet állítsuk meg.

$$3. \quad y = a + 10f + 100 f^2.$$

$$4. \quad y = \frac{a}{f} + \frac{b}{f^2} + \frac{c}{f^3}.$$

A 2, 3, 4 feladatokhoz szükséges konstansok: (0050) = a;
(0051) = b; (0052) = c; (0053) = f.

$$y \implies 0060$$

A programok a (0200)-as címnél kezdődjenek.

5. Számítsuk ki a következő kifejezést:

$$x = a_1b_1 + a_2b_2 + a_3b_3.$$

Legyen:

$$(0010) = a_1; \quad (0013) = b_1;$$

$$(0011) = a_2; \quad (0014) = b_2;$$

$$(0012) = a_3; \quad (0015) = b_3;$$

$$x \implies 0020$$

3.3. Egyszerű ciklikus programok; a ciklusutasítások változatlanok

Említettük az előzőkben, hogy ciklusnak a program olyan részét nevezzük, amely ugró utasítások segítségével egymásután többször kerül végrehajtásra. Ebben a pontban olyan ciklusokról lesz szó, amelyeknél a program egy részét minden változtatás nélkül egymásután többször végrehajtjuk.

Ilyen ciklus kétféle lehet:

1. Előre ismert az a szám, ahányszor a ciklust ismételni kell. Ilyenkor számláljuk az ismétléseket.
2. Az ismétlések száma nem ismert. Ekkor egy logikai feltétel teljesülése vagy nem teljesülése ugratja ki a gépet a ciklusból.

Az 1. pont szerinti ciklus fordul elő például akkor, ha valamely mennyiséget hatványozunk. A hatványozandó mennyiséget egymásután többször megszorozzuk önmagával, annyiszor t.i. amekkora, a kitevő.

Példa:

$$y = x^5 \implies 0032 \quad \langle x \rangle = 0010$$

A program összeállításánál a következő konstansokra lesz szükségünk:

$$(3 \cdot 2^{-30}) = 0011, \quad \langle 2^{-30} \rangle = 0012$$

Az utasításokat a 0050 memóriarekesztől kezdve helyezzük el.

Program:

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-------------------|--------|
| | szimbóluma | jele | | | | |
| 0050 | Á, | 05 | x | 0010 | x | 0032 |
| 0051 | x | 03 | x | 0010 | x^k | 0032 |
| 0052 | - | 01 | 2^{-30} | 0012 | $3 \cdot 2^{-30}$ | 0011 |
| 0053 | FU | 34 | - | 0054 | - | 0051 |

Ebben a programban a 0051-es utasítást hajtjuk többször végre. A 0052 utasítás eredményeképpen mindaddig pozitív számot kapunk, amíg a 0032 memóriarekesz tartalma x^5 nem lesz. Ha már megkaptuk ezt a végeredményt, a 0052 utasítás végrehajtása után negatív szám képződik, és a 0053 utasításban álló FU kiugratja a gépet a ciklusból.

Állítsuk össze ennek alapján a hatványozás programját általános esetben.

$$y = x^n \implies 0032 \quad \langle x \rangle = 0010; \quad \langle (n-2) \cdot 2^{-30} \rangle = 0011;$$

$$2^{-30} \quad 0012$$

A programot tároljuk most is a 0050-es memóriarekesztől kezdve.

Program:

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------|--------|-----------------------|--------|
| | szimbóluma | jele | | | | |
| 0050 | A, | 05 | x | 0010 | x | 0032 |
| 0051 | x | 03 | x | 0010 | x^k | 0032 |
| 0052 | - | 01 | 2^{-30} | 0012 | $(n-2) \cdot 2^{-30}$ | 0011 |
| 0053 | FU | 34 | - | 0054 | - | 0051 |

Vizsgáljuk meg a programot $n = 2$ esetére. Ekkor a 0051-es utasítással a 0032-es rekeszben képezi x^2 -et, majd $(n-2) \cdot 2^{-30}$ -ből, azaz $0 \cdot 2^{-30}$ -ből 2^{-30} -at kivon. Az eredmény -2^{-30} , vagyis negatív szám. Az FU utasítás kiugratja a gépet a ciklusból. Rajzoljuk meg ennek a programnak a menetrendjét. (L. az V. táblát.)

A menetrendről leolvasható az ilyen típusú ciklusok általános szerkezete.

Az alábbiakban olyan feladatokat ismertetünk, amelyeknél szintén célszerű ciklikusan visszatérni a program egy alkalmas pontjára, az ismétlések száma azonban nem ismert előre. Az iterációs eljárások pl. ilyen módon programozhatók. Legyen az iterációt értelmező formula

$$x_k = f(x_{k-1})$$

alakú az x_0 kezdeti érték pedig adott. Ezt vagy a program

számítja ki a ciklus előtt, vagy előre ismerjük, s a perforált szalagon visszük be a memóriába. A logikai feltétel, melynek teljesülése vagy nem teljesülése kiugratja a gépet a ciklusból, feladatonként változik.

Szerkesszük meg például a négyzetgyökvonás programját.

$$x = \sqrt{a} \implies 0007$$

Feltesszük, hogy $a > 0$, tehát a gyökvonás elvégezhető.

A számítás az

$$x_n = 0,5 \left(\frac{a}{x_{n-1}} + x_{n-1} \right)$$

formula alapján történik, ahol n a közelítő iterációs lépés sorszáma. Ez az u.n. Newton-féle módszer.

Mielőtt a programot megszerkesztjük, az iterációs képletet átalakítjuk a következő módon:

$$x_{n+1} = 0,5 \left(\frac{a}{x_n} + x_n \right) = \frac{|0,5 \left(\frac{a}{x_n} - x_n \right)|}{D} + x_n$$

A számítást akkor fejezzük be, ha a közelítés pontosságát nem lehet tovább fokozni, azaz ha az utolsó két lépésnél kapott számok különbsége - a képletben szereplő D mennyiség - 2^{-30} -nál kisebb. Megszerkesztjük a menetrendet.

(L. a VI. táblát.)

Program: Az utasításokat a 0000 rekesztől kezdve helyezzük el.
(0010) = a ; (0014) = x_0 ; (0011) = $1/2$; (0013) = 2^{-30}

| utasítás rekesze | a művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|------------|------|-----------------------------|--------|-----------------------------|--------|
| | szimbóluma | jele | | | | |
| 0000 | Á | 05 | x_0 | 0014 | x_0 | 0017 |
| 0001 | : | 12 | x_n | 0017 | a | 0010 |
| 0002 | ↓- | 31 | x_n | 0017 | - | - |
| 0003 | ↓x | 23 | $1/2$ | 0011 | $1/2 (\frac{a}{x_n} - x_n)$ | 0012 |
| 0004 | ↓+ | 20 | x_n | 0017 | x_{n+1} | 0017 |
| 0005 | - , | 51 | $1/2 (\frac{a}{x_n} - x_n)$ | 0012 | 2^{-30} | 0013 |
| 0006 | FU | 34 | - | 0001 | - | 0007 |

Szerkesszük meg ugyanilyen módszerrel az $x = \sqrt[n]{a}$ programját. Itt az iterációs képlet:

$$\sqrt[n]{x_{k+1}} = x_k - \frac{x_k^n - a}{n x_k^{n-1}}$$

ahol k a közelítő iterációs lépés sorszáma. Ennél a feladatnál a logikai feltétel, mely szerint abbahagyjuk az iterációt, az lesz, hogy a hibatag, azaz $z_k = \frac{x_k^n - a}{n x_k^{n-1}}$ kisebb legyen, mint 2^{-30} , - amit a gép, mint 0-t "érezkelne" amúgy is.

Feltesszük, hogy a művelet elvégezhető, vagyis ha n páros szám, akkor $a > 0$. Ezenkívül feltesszük még, hogy $n > 2$.

Legyen x_0 egy olyan kezdeti érték, amelynél az iteráció konvergál. Könnyen belátható, hogy a $\in (-1; 1)$ esetén (a feladatot 2 - esetleg 10 - alkalmas hatványának kiemelésével mindenképp ilyen alakúra kell redukálnunk, hiszen egyébként nem tudnók ábrázolni a gépben a -t) az $x_0 = 1 - 2^{-30}$ kezdőérték mellett n tetszőleges pozitív értékénél konvergens a kapott számsorozat.

Megszerkesztjük a menetrendet; (L. a VII. táblát). Az n gyökkitevő csak egész szám lehet. Ezt $n \cdot 2^{-30}$ alakban tesszük be a memóriába. Ezáltal elérjük, hogy n a szó végén

helyezkedik el. Ezt a számot, amely aszerint változik, hogy a programot hányadik gyök vonására használjuk fel, "program-paraméter"-nek nevezzük. Valahányszor a programot használjuk, ezt a számot mindig külön be kell írni a memóriába. Az olyan konstansokat, amelyek függnek ettől az n programparamétertől, a program segítségével kell előállítani.

Program:

$$x \implies 0050; \quad (0010) = a; \quad (0011) = n \cdot 2^{-30}; \quad (0012) = 3 \cdot 2^{-30}$$

$$(0013) = 2^{-30}; \quad (0014) = x_0$$

| utasítás rekesze | A művelet | | 1. cím tartalma | 1. cím | 2. cím tartalma | 2. cím |
|------------------|-------------|------|-----------------------|--------|-----------------------|--------|
| | szim-bóluma | jele | | | | |
| 0100 | : | 12 | $n \cdot 2^{-30}$ | 0011 | 2^{-30} | 0013 |
| 0101 | U1 | 24 | - | 0102 | $\frac{1}{n}$ | 0020 |
| 0102 | - | 11 | $3 \cdot 2^{-30}$ | 0010 | $n \cdot 2^{-30}$ | 0011 |
| 0103 | U1 | 24 | - | 0104 | $(n-3) \cdot 2^{-30}$ | 0021 |
| 0104 | Á | 05 | x_0 | 0014 | x_0 | 0050 |
| 0105 | Á | 05 | $(n-3) \cdot 2^{-30}$ | 0021 | $(n-3) \cdot 2^{-30}$ | 0023 |
| 0106 | Á | 05 | x_k | 0050 | x_k | 0022 |
| 0107 | x | 03 | x_k | 0050 | x_k^n | 0022 |
| 0110 | - | 01 | 2^{-30} | 0013 | $(n-3) \cdot 2^{-30}$ | 0023 |
| 0111 | FU | 34 | - | 0112 | - | 0107 |
| 0112 | x, | 13 | x_k^{n-1} | 0022 | x_k | 0050 |
| 0113 | ↓- | 31 | a | 0010 | - | - |
| 0114 | ↓: | 32 | x_k^{n+1} | 0022 | - | - |
| 0115 | ↓x, | 23 | $\frac{1}{n}$ | 0020 | z_k | 0023 |
| 0116 | - | 01 | x_k | 0023 | x_k | 0050 |
| 0117 | - , | 51 | 2^{-30} | 0013 | z_k | 0023 |
| 0120 | FU | 34 | - | 0121 | - | 0105 |

Munkarekeszek: 0020, 0021, 0022, 0023.

Néhány további példát adunk meg ismétlődő ciklusok programozására:

1. Számítsuk ki a következő függvény helyettesítési értékét az $x = x_0$ helyen:

$$f(x) = \frac{x_1}{2} + \frac{x_2}{4} + \dots + \frac{x_{10}}{2^{10}} = \sum_{i=1}^{10} \frac{x_i}{2^i}$$

ahol $x_{\underline{i}} = ax_{\underline{i-1}} + b;$ $x_{\underline{1}} = ax_0 + b;$

Szerkesszük meg a menetrendet (L. a VIII. tábla 1. ábráját).

Konstansok: (0010) = x_0 ; (0011) = a ; (0012) = b ;

(0013) = $1/2$; (0014) = 2^{-30} ; (0015) = $9 \cdot 2^{-30}$; (0016) = 0

Munkarekeszek: 0020, 0021, 0022

$$f(x) \implies 0050$$

Program:

| | | | | | | |
|------|-----|----|------------------------------------|------|--------------------------------|------|
| 0100 | Á | 05 | 0 | 0016 | 0 | 0050 |
| 0101 | Á | 05 | $9 \cdot 2^{-30}$ | 0015 | $9 \cdot 2^{-30}$ | 0023 |
| 0102 | Á | 05 | x_0 | 0010 | x_0 | 0020 |
| 0103 | Á | 05 | $\frac{1}{2}$ | 0013 | $\frac{1}{2}$ | 0021 |
| 0104 | x, | 13 | a | 0011 | x_{k-1} | 0020 |
| 0105 | ↓+ | 20 | b | 0012 | x_k | 0020 |
| 0106 | ↓x, | 33 | $\frac{1}{2^k}$ | 0021 | - | - |
| 0107 | ↓+ | 20 | $\sum_{k=1}^{i-1} \frac{x_k}{2^k}$ | 0050 | $\sum_{k=1}^i \frac{x_k}{2^k}$ | 0050 |

| | | | | | | |
|------|----|----|---------------|------|-----------------|------|
| 0110 | - | 01 | 2^{-30} | 0014 | $(9-i)2^{-30}$ | 0023 |
| 0111 | FU | 34 | - | 0014 | - | 0112 |
| 0112 | x | 03 | $\frac{1}{2}$ | 0013 | $\frac{1}{2^k}$ | 0021 |
| 0113 | U2 | 74 | - | - | - | 0104 |
| 0114 | . | . | . | . | . | . |

2. Számítsuk ki az

$$x = a^{20} + a^{19} b + a^{28}b^2 + \dots + a b^{19} + b^{20} \text{ \u00e9rt\u00e9ket:}$$

A menetrendet a VIII. t\u00e1bla m\u00e1sodik \u00e1br\u00e1ja mutatja.

Konstansok: (0010) = a; (0011) = b; (0012) = $18 \cdot 2^{-30}$; (0013) = $19 \cdot 2^{-30}$; (0014) = 2^{-30}

Munkarekeszek: 0030, 0031, 0032, 0033

Program:

| | | | | | | |
|------|--------------------|----|-----------------------------|------|---------------------------------|------|
| 0100 | \u00c1 | 05 | $18 \cdot 2^{-30}$ | 0016 | $18 \cdot 2^{-30}$ | 0031 |
| 0101 | \u00c1 | 05 | $19 \cdot 2^{-30}$ | 0015 | $19 \cdot 2^{-30}$ | 0032 |
| 0102 | \u00c1 | 05 | a | 0010 | a | 0033 |
| 0103 | x | 03 | a | 0013 | a^k | 0033 |
| 0104 | - | 01 | a | 0011 | $(18-k) \cdot 2^{-30}$ | 0031 |
| 0105 | FU | 34 | 2^{-30} | 0012 | - | 0103 |
| 0106 | :, | 12 | a | 0021 | b | 0011 |
| 0107 | U1 | 24 | - | 0050 | $a^{-1} b$ | 0030 |
| 0110 | \u00c1 | 05 | a^{20} | 0050 | - | 0040 |
| 0111 | x | 03 | $a^{-1} b$ | 0050 | $a^{20-i} b^i$ | 0033 |
| 0112 | \u2193+ | 20 | $\sum_{k=0}^i a^{20-k} b^k$ | 0050 | $\sum_{k=0}^{i+1} a^{20-k} b^k$ | 0040 |
| 0113 | - | 01 | 2^{-30} | 0050 | $(19-i) 2^{-30}$ | 0032 |
| 0114 | FU | 34 | - | 0050 | - | 0111 |
| 0115 | meg\u00e1ll\u00e1s | | | | | |

3. Számítsuk ki az $x = \frac{1}{n!}$ értékét.

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$$

A számítást a következőképpen végezzük el:

$$\frac{1}{3!} = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{2} \cdot \frac{1}{2+1}$$

$$\frac{1}{4!} = \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2+1} \cdot \frac{1}{3+1}$$

.....
.....

$$\frac{1}{n!} = \frac{1}{2} \cdot \frac{1}{3} \cdot \dots \cdot \frac{1}{n-1} \cdot \frac{1}{n} = \frac{1}{2} \cdot \frac{1}{2+1} \cdot \frac{1}{3+1} \cdot \dots \cdot \frac{1}{(n-1)+1} =$$

$$= \frac{1}{2} \cdot \frac{1}{2+1} \cdot \frac{1}{2+2} \cdot \dots \cdot \frac{1}{2+(n-2)}$$

Tehát a számítást a következő képlet alapján végezzük el:

$$\frac{1}{n!} = \prod_{i=0}^{n-2} \frac{1}{(2+i)}$$

A menetrendet a IX. táblán, az első ábrán találjuk.

Konstansok: (0010) = $\frac{1}{2}$; (0011) = 2^{-30} ; (0012) = $n \cdot 2^{-30}$;

$$(0015) = 2 \cdot 2^{-30}$$

Munkarekeszek: 0031, 0032

$$\frac{1}{n!} \implies 0030$$

Program:

| | | | | | | |
|------|-----|----|-----------|------|-------------------|------|
| 0100 | - , | 11 | 2^{-30} | 0011 | $n \cdot 2^{-30}$ | 0012 |
| 0101 | U1 | 24 | - | 0102 | - | 0031 |
| 0102 | Á | 05 | $1/2$ | 0010 | $\frac{1}{2}$ | 0030 |

| | | | | | | |
|------|----|----|--------------------|------|-------------------|------|
| 0103 | Á | 05 | $2 \cdot 2^{-30}$ | 0013 | $2 \cdot 2^{-30}$ | 0032 |
| 0104 | + | 00 | 2^{-30} | 0011 | $k \cdot 2^{-30}$ | 0032 |
| 0105 | :, | 12 | $k \cdot 2^{-30}$ | 0032 | 2^{-30} | 0011 |
| 0106 | ↓x | 23 | $\frac{1}{(k-1)!}$ | 0030 | $\frac{1}{k!}$ | 0030 |
| 0107 | -, | 11 | $k \cdot 2^{-30}$ | 0032 | $(n-1)2^{-30}$ | 0031 |
| 0110 | FU | 34 | - | 0111 | - | 0104 |
| 0111 | . | . | . | . | . | . |

4. Számítsuk ki az $y = \frac{1}{a-bx}$ függvény helyettesítési értékét az $x=x_0$ helyen hatványsora segítségével.

Hatványsorba fejtve:

$$\frac{1}{a-bx} = \frac{1}{a} + \frac{bx}{a^2} + \frac{b^2x^2}{a^3} + \dots \quad (|x| < \left| \frac{a}{b} \right|)$$

A számítást akkor fejezzük be, ha $\left| \frac{b^n x^n}{a^{n+1}} \right| < 2^{-30}$

A menetrendet a IX. tábla második ábráján találjuk.

Konstansok: (0010) = $\frac{1}{a}$; (0011) = b; (0012) = x_0 ;

(0013) = 2^{-30} ; (0014) = 0

Munkarekeszek: 0020, 0021

A program a 0120-as memóriarekesznél kezdődik.

$y \implies 0030$

Program:

| | | | | | | |
|------|----|----|---------------|------|------------------|------|
| 0120 | Á | 05 | $\frac{1}{a}$ | 0010 | $\frac{1}{a}$ | 0030 |
| 0121 | Á | 05 | $\frac{1}{a}$ | 0010 | $\frac{1}{a}$ | 0020 |
| 0122 | x, | 13 | b | 0000 | x_0 | 0012 |
| 0123 | ↓x | 23 | $\frac{1}{a}$ | 0010 | $\frac{bx_0}{a}$ | 0021 |

| | | | | | | |
|------|------|----|--|------|--|------|
| 0124 | x | 03 | $\frac{bx_0}{a}$ | 0021 | $\frac{b^{k-1} x_0^{k-1}}{a^k}$ | 0020 |
| 0125 | ↓+ | 20 | $\sum_{i=0}^{k-i} \frac{b^{i-1} x^{i-1}}{a^i}$ | 0030 | $\sum_{i=0}^k \frac{b^{i-1} x_0^{i-1}}{a^i}$ | 0030 |
| 0126 | - , | 51 | 2^{-30} | 0013 | $\frac{b^{i-1} x_0^{k-1}}{a^k}$ | 0020 |
| 0127 | FU | 34 | - | 0130 | - | 0124 |
| 0130 | . | . | . | . | . | . |

5. Számítsuk ki az $y = \ln \frac{1+x}{1-x}$ függvény helyettesítési értékét az $x = x_0$ helyen.

Hatványsorba fejtve:

$$\ln \frac{1+x}{1-x} = 2 \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots \right)$$

Ha $|x| < 1$, a sor konvergens.

$$\ln \frac{1+x_0}{1-x_0} \implies 0050$$

A számítást akkor fejezzük be, ha

$$\ln \left| \frac{x_0^{2k+1}}{2k+1} \right| < 2^{-30},$$

ha tehát az utolsó tag nem ad már javítást a kiszámított értékhez.

A menetrendet a x. táblán láthatjuk.

Konstansok: (0010) = x_0 ; (0011) = 2^{-30} ; (0013) = $\frac{1}{2}$;

$$(0012) = 2^{-30}$$

Munkarekeszek: 0030, 0031, a = 0032 , 0033

A program a 0100-as memóriarekesznél kezdődik.

$$\ln \frac{1+x_0}{1-x_0} \implies 0050$$

Program:

| | | | | | | |
|------|-----|----|---|------|---|------|
| 0100 | Á | 05 | x_0 | 0010 | x_0 | 0050 |
| 0101 | Á | 05 | x_0 | 0010 | x_0 | 0030 |
| 0102 | Á | 05 | 2^{-3^0} | 0011 | 2^{-3^0} | 0032 |
| 0103 | x, | 13 | x_0 | 0010 | x_0 | 0010 |
| 0104 | U1 | 24 | - | 0105 | x_0^2 | 0031 |
| 0105 | x | 03 | x_0^2 | 0031 | x_0^{2k+1} | 0030 |
| 0106 | + | 00 | $2 \cdot 2^{-3^0}$ | 0012 | $(2k+1)2^{-3^0}$ | 0032 |
| 0107 | :, | 12 | $(2k+1)2^{-3^0}$ | 0032 | 2^{-3^0} | 0011 |
| 0110 | ↓x | 23 | x_0^{2k+1} | 0030 | $\frac{x_0^{2i+1}}{2^{i+1}}$ | 0033 |
| 0111 | ↓+ | 20 | $\sum_{i=0}^{k-i} \frac{x_0^{2i+1}}{2^{i+1}}$ | 0050 | $\sum_{i=0}^k \frac{x_0^{2i+1}}{2^{i+1}}$ | 0050 |
| 0112 | - , | 51 | $\frac{x_0}{2^{i+1}}$ | 0033 | 2^{-3^0} | 0011 |
| 0113 | FU | 34 | - | 0105 | - | 0114 |
| 0114 | : | 02 | 1/2 | 0013 | $2 \sum_{i=0}^k \frac{x_0^{2i+1}}{2^{i+1}}$ | 0050 |
| 0115 | . | . | . | . | . | . |

4. Fejezet.

CÍM- és MŰVELETI UTASÍTÁSOK MENETKÖZBENI MÓDOSÍTÁSA.

Az előző fejezetekben megismerkedtünk az u.n. direkt programozással, a választási műveletek és a ciklusok programozásával. Mindeddig nem használtuk ki az M-3 gépnek azt a tulajdonságát (amely egyébként minden modern számológép közös tulajdonsága), hogy tárolt programú, azaz a programot alkotó utasítások a számadatokkal azonos formában szerepelnek a gép belső (operatív) memóriájában.

A direkt programozásnál az utasítások az előre megadott változatlan sorrendben és formában kerülnek végrehajtásra. Azoknál a választási műveleteknél és ciklusoknál, amelyekkel eddig foglalkoztunk, az utasítások végrehajtásának sorrendje ugyan eltér a tárolás sorrendjétől, de az utasítások változatlanok, végig olyanok, ahogyan kezdetben megadták őket.

Fenti részprogramok ugyanúgy végrehajtásra kerülhetnek; ha az utasítások valamilyen külső közegben (pl. lyukkártyán vagy lyukszalagon) lennének tárolva, változtathatatlan módon.

Összetett feladatoknak a megoldása viszont azt követeli meg, hogy a program maga változtathassa meg a saját utasításait, akár bizonyos feltételektől függően, akár egy ciklusban minden egyes ismétlődés alkalmával. A programok nagy többségében ugyanis - az előző fejezet példáitól eltérően - a ciklusokat változó utasításokkal kell végrehajtani. Mielőtt azonban ezeknél az általános ciklikus programoknak a tárgyalására térnénk, foglalkozni fogunk az utasításmódosítások módszereivel és avval, hogyan állít-

ja össze, szerkeszti meg egy adott számítási program egyes saját utasításait, amelyeket a későbbiek folyamán végrehajt. Mindezekre az utasításokkal való manipulációkra az ad lehetőséget, hogy a gép tárolt programú, és az utasításokkal ugyanazok a műveletek végezhetőek el, mint a számadatokkal. Azoknál a gépeknél, amelyeknél a program külsőleg van megadva, az utasításokkal nem végezhetőek műveletek, és azok nem változtathatók. A módszert néhány példa kapcsán mutatjuk be.

4.1. A címutasítások módosítása.

1. Adjuk össze az 1-től 40-ig terjedő természetes számokat és az összeget tegyük az 500-as rekeszbe. Mivel itt 40 összeadásról van szó, célszerű azokat nem külön, direkt módon programozni, hanem ciklikus programmal keresni a megoldást. A program menetrendjét a XI. táblán találjuk.

Mint látjuk, az 500-as rekesz nemcsak a végeredményt, hanem mindenkori részletösszeget is tartalmazza, az 501-es rekesz pedig a mindenkori összeadandó ideiglenes tárolására szolgál. Ugyanakkor ezzel az összeadandóval végezzük a ciklus-számlálást is. Ha t.i. az 501-es rekesz tartalma elérte a 40-et, a program befejeződött (nem szükséges mindig külön számlálórekeszt használni). A következőkben nem írjuk ki az utasítások címeinek tartalmát.

A program az 1000-es memóriarekesztől kezdődően:

| | | | | |
|------|----|----|------|------|
| 1000 | Á | 05 | <0> | 0500 |
| 1001 | Á | 05 | <0> | 0501 |
| 1002 | + | 00 | <1> | 0501 |
| 1003 | + | 00 | 0501 | 0500 |
| 1004 | -, | 11 | <40> | 0501 |
| 1005 | FU | 34 | 1002 | 1006 |
| 1006 | -- | -- | -- | -- |

Megjegyzések:

- a.) A felhasznált konstansok (0, 1, 40) elmeit szimbolikusan szögletes zárójellel jelöltük. Ez azt jelenti, hogy ezeket a konstansokat elvben bármelyik szabad memória-rekeszben elhelyezhetjük. Konkrét program szerkesztésekor a választott rekeszek címeit meg kell adni.
- b.) Az 1-től 40-ig terjedő természetes számokat (így tehát a programban szereplő 1 és 40 konstansokat is) 2^{-30} -al szorozva helyezzük el a gép memóriájában. Ez azt jelenti, hogy ezek a számok a gépi szavak "jobboldalán" helyezkednek el.

Mint látjuk, a fenti feladat programja olyan ciklikus program, amelynek utasításai változatlanok, olyan program tehát, amelynek típusával az előző fejezetben foglalkoztunk. Ha azonban a feladatot egy kissé módosítjuk, akkor már nem tudunk fix utasítású ciklussal programozni. Legyen az új feladat a következő: adjunk össze 40 számot, amelyek között nincs megadva semmilyen összefüggés. Ekkor a 40 összeadandó számot előzetesen a gép memóriájában tárolnunk kell, például a 101-től 150-ig terjedő rekeszekben (mint ismeretes, a címek 8-as számrendszerben vannak megadva!). Gyűjtsük az összeget ismét az 500-as rekeszben. A program kezdetén, mint az előző esetben, most is ki kell írítani az 500-as rekeszt ("0-t kell belevinni"). Ezután az 500-as rekesz tartalmához sorra hozzá kell adni a 101, 102, ..., 150 rekeszek tartalmát. A program kezdete a következő lesz:

| | | | | |
|------|---|----|------|------|
| 1000 | Á | 05 | <0> | 0500 |
| 1001 | + | 00 | 0101 | 0500 |

Ezzel bevittük az első összeadandót a gyűjtőrekeszbe. Ezután szükség lenne az 1001-es utasításnak még 39-szer történő megismétlésére úgy, hogy első címének helyén 0101 helyett rendre 0102, 0103, ..., 0150 legyen. Ez azt jelenti: a program egy szakaszát - jelen esetben egyetlen utasítást -

kell megismételni, minden ismétlés alkalmával módosított formában. Ez a módosítás úgy érhető el, hogy az 1001-es utasítás első címéhez - minden ismétlés előtt - 1-et adunk hozzá, és minden ismétléskor megvizsgáljuk, hogy ez a módosított cím elérte-e a 150-et. Utóbbi esetben a program véget ér, "kiugrik a ciklusból".

A címmódosítás céljára felhasználjuk a 2^{-18} konst. Az utasítások első címe ugyanis, mint ismert, a szó 7.-16. helyértékein helyezkedik el. Ezt a címet úgy lehet eggyel növelni, ha az utasítást képviselő szóhoz - mint számadathoz - 2^{-18} -at adunk hozzá. Legyen 2^{-18} az 501-es rekeszben, azaz: $(501) = 2^{-18}$ és $(502) = 00\ 0150\ 0500$

A fenti program folytatása:

| | | | | |
|------|----|----|---------------------------|------|
| 1002 | + | 00 | $\langle 2^{-18} \rangle$ | 1001 |
| 1003 | -, | 11 | 1001 | 0502 |
| 1004 | FU | 34 | 1005 | 1001 |
| 1005 | -- | -- | -- | -- |

Az 1002-es utasítás módosítja az 1001-es utasítást. Az 1003-as utasítás az 502-es rekeszben tárolt u.n. utasításkonstanssal való összehasonlítás által megvizsgálja: túlhaladta-e már az 1001-es utasítás első címe a 150-et. Ha nem, akkor az 1004-es feltételes ugró utasítás visszaadja a vezérlést az 1001-es utasításra, módosított formában megismélteti a ciklust. Ha igen, akkor ugyanez a feltételes ugró utasítás "kiugratja" a programot a ciklusból.

A fenti példán láttuk, hogyan történik az utasítások módosítása a ciklusokban. A módszer ugyanaz akkor is, ha egy ciklusban nem egy, hanem több utasítást kell megváltoztatni. A példa programja végrehajtja a kitűzött feladatot, de mégis van egy hiányossága, amiért ebben a formában nem lehetne alkalmazni egy valóságos program részeként. Ez a hiányosság: lefutása után más alakban marad, mint amelyben eredetileg volt (nevezetesen: az 1001-es

utasítás megváltozott, és nem lehet újra felhasználni). Az általános ciklusok tárgyalásánál részletesen fogunk foglalkozni azokkal a módszerekkel, amelyekkel ez a probléma megoldható.

2. Gyakran előfordul, hogy a gép memóriájában tárolt programnak különböző részeit kell végrehajtani egy - a konkrét számítás kezdetén megadott és a memória meghatározott rekeszébe bevitt paraméter értékétől függően. Tekintsük például azt a feladatot, amelyben a program egy bizonyos ponttól kezdve az 1500-as, 1510-es, 1520-as, ..., 1550-es utasításnál folytatódik attól függően, hogy az 500-as rekesz tartalma 0, 1, 2, ..., 5. Az 500-as rekesz tartalma a paraméter. A feladat megoldása: olyan feltétlen ugró utasítást kell szerkeszteni, amelynek ugrócíme (az a címe, amely a következő utasítást megadja):

$$1500 + 10 \cdot (0500)$$

(8-as számrendszerben, a "10"-es tehát 8-at jelent!) A paraméter 2^{-30} -cal szorozva szerepel az 500-as rekeszben, mert az ugró utasítás második címének képzéséről kell gondoskodnunk.

A program a következő:

| | | | | | |
|------|----|----|-------------------------------|------|---------------------------------|
| 1000 | : | 12 | $\langle \frac{1}{8} \rangle$ | 0500 | |
| 1001 | ↓+ | 20 | 0501 | 1002 | Konstans: (0501) = 74 0000 1500 |
| 1002 | - | - | - | - | |

Az 501-es rekeszben lévő konstans u.n. utasításkonstans, az 1002-es utasítás felépítésére szolgál. (Azért nem adtuk meg ennél az U2 műveleti szimbólumot, mert ez a konstans az 501-es rekeszben nem kerül utasításként végrehajtásra, hanem mint számadat szerepel.) Az 1000-es utasítás megszorozza a paramétert 8-cal (gépi "10"-zel), illetve, minthogy 8 nem helyezhető el a gépben, elosztja $\frac{1}{8}$ -dal. Az így kapott számot az

1001-es utasítás összeadja az 501-es rekeszben tárolt utasításkonstanssal, és az eredményt - a kész ugró-utasítást elhelyezi az 1002-es rekeszben. Ezután kerül sor az 1002-es rekeszben elhelyezett vezérlésátadás végrehajtására, amely által a program a megadott paraméternek megfelelő részprogramra ugrik át. Az 1002-es utasítás eredetileg nincs kitöltve, eredeti tartalma közömbös.

Itt tehát példát láttunk arra, hogy a program valamely utasítását - mielőtt végrehajtására sor kerülne - maga a program állítja össze.

A fenti típusú programrészeket, amelyek egy beállítható paramétertől függően különböző elágazásokat valósítanak meg, a blokkdiagramban úgynevezett állítható kapcsolóval jelzik, a XII. tábla első ábrája szerint.

A kapcsoló beállításának (ez példánkban a megfelelő paraméter elhelyezése az 500-as rekeszben) a szimbóluma pl.:

$$\boxed{1 \implies \alpha}$$

Ez azt jelenti, hogy az α kapcsoló az 1-es állásba került, a program az α_1 -el jelölt úton fog folytatódni.

Egy kapcsolóállítást és kapcsolót tartalmazó programrészlet diagramját ábráztuk a XII. tábla második részében.

3. Mátrix-elem elhelyezése a memóriában.

Tegyük fel, hogy egy $m \times n$ -es mátrix van a gép memóriájában, úgy, hogy az egyes elemeket az egymásra következő sorok sorrendjében, az 1500-as rekesztől kezdődő egymásutáni rekeszek tárolják. Feladatunk: egy adott számot, közelebbről az 500-as rekesz tartalmát, az a_{ij} elem számára fenntartott rekeszbe kell elhelyezni.

Az n , i , j számok 2^{-15} -el megszorozva - mint paraméterek szerepelnek a gép memóriájában. Az új elem címe:

$$1500 + n \cdot (i-1) + j-1.$$

Ez lesz az átviteli utasítás címe. A program:

| | | | | |
|------|------------------|----|-----------------------------------|---------------------------|
| 1000 | -, | 11 | $\langle 2^{-15} \rangle$ | $\langle 2^{-15} \rangle$ |
| 1001 | $\downarrow x$, | 33 | $\langle n \cdot 2^{-15} \rangle$ | |
| 1002 | $\downarrow :$, | 32 | $\langle 2^{-15} \rangle$ | |
| 1003 | $\downarrow +$, | 30 | $\langle j \cdot 2^{-15} \rangle$ | |
| 1004 | $\downarrow -$, | 31 | $\langle 2^{-15} \rangle$ | |
| 1005 | $\downarrow x$, | 33 | $\langle 2^{-15} \rangle$ | |
| 1006 | $\downarrow +$ | 20 | 0501 | 1007 |
| 1007 | | - | - | - |

(utasításkonstans (0501) = 15 0500 1500)

Az 1002-es és 1005-ös utasítások a részeredmények megfelelő helyérték-eltolását szolgálják. A kiszámított cím értékének 2^{-30} -cal kell szorozva lennie, mert az átviteli utasítás második címéről van szó. Az 1007-es átviteli utasítást maga a program állítja elő. A programban három paraméter (n , i , j) és egy konstans (2^{-15}) szerepel.

4.2. Műveleti utasítások módosítása.

Az utasításokkal való manipuláció nem korlátozódik az utasítás címeinek módosítására, vagy a címek összeállítására. Minthogy a műveleti jel is számjegy-kombináció, ezen ugyanúgy végezhetők aritmetikai műveletek, mint a címeken. Ilyen esetekben persze figyelembe kell venni a szereplő utasítások konkrét műveleti jeleit (kódjeleit). Pl. egy számítás folyamán valamely két adott mennyiségnek, az 501 és 502 rekeszek tartalmának, összegét vagy különbségét kell képezni,

attól függően, hogy egy paraméternek - az 500-as rekesz tartalmának - értéke 0 vagy 2^{-30} .

A programrészlet, amellyel ez a feladat megoldható:

| | | | | |
|-----------|----|----|---------------------------|------|
| 1000 | : | 12 | $\langle 2^{-24} \rangle$ | 0500 |
| 1001 | ↓x | 30 | 1002 | 1002 |
| 1002 | + | 00 | 0502 | 0501 |
| | | | | |

Az 500-as rekeszben tárolt paramétert 24 helyértékekkel el kell tolni balra (ezt valósítja meg az 1000-es utasítás a 2^{-24} -el való osztás által), hogy a műveleti jel megfelelő helyértékére, a szó 6. helyértékére kerüljön. Ha az így kapott értéket hozzáadjuk az 1002-es utasításhoz (ezt az 1001-es utasítás végzi), úgy az 1002-es utasítás a paramétertől függően változatlan marad, vagy kivonó utasítássá lesz.

Megjegyzések:

- a.) Ha az 500-as rekeszben tárolt paraméter-érték $2 \cdot 2^{-30}$, $3 \cdot 2^{-30}$ vagy $6 \cdot 2^{-30}$, úgy a fenti program az ismertetett módon az 501-es és 502-es rekeszekben tárolt számoknak hányadosát, szorzatát, vagy logikai szorzatát képezi.
- b.) Az 500-as rekeszben tárolt paraméter elhelyezhető a számítás kezdete előtt (ha pl. ugyanazt a programot többféle változatban kívánjuk felhasználni), de lehetséges, hogy magának az adott számítási programnak előző szakasza számítja ki és helyezi el a paramétert.

A blokkdiagramban a fenti típusú program-részletet az előzőekben megismert állítható kapcsolóval jelölik, annyi kapcsolóállással, ahány változatot a program ténylegesen megvalósít.

A továbbiakban gyakorló-példákat dolgozunk ki az utasításokkal való különböző manipulációkra.

1. Az 500-as rekeszben tárolt paraméter a következő 8 értéket veheti fel: $0, 2^{-30}, 2 \cdot 2^{-30}, \dots, 7 \cdot 2^{-30}$. A paraméter értékétől függően adjunk 2^{-18} -at az 510-es, 511-es, ... illetőleg az 517-es rekesz tartalmához. Ez gyakran fordul elő olyan esetekben, amikor egy kiszámított eredménytől függően valamely rekeszt meg kell jelölni.

A paramétertől függő utasítás a következő alakú:

+ 00 < 2^{-18} > 051..

A részprogram:

| | | | | |
|------|---|----|---------------|------|
| 1000 | + | 00 | 0500 | 1001 |
| 1001 | + | 00 | < 2^{-18} > | 0510 |

2. Ha az 500-as rekesz tartalma: $x < 2^{-3}$ adjunk 2^{-30} -at az 510-es rekesz tartalmához. Ha $2^{-3} \leq x < 2 \cdot 2^{-3}$, adjunk 2^{-30} -at az 511-es rekesz tartalmához sít., ha $7 \cdot 2^{-3} \leq x < 1$, adjunk 2^{-30} -at az 517-es rekesz tartalmához.

A program a következő:

| | | | | |
|-----------|----|----|-------------------|------|
| 1000 | x | 03 | < $\frac{1}{2}$ > | 0500 |
| 1001 | Á | 05 | <0> | 0501 |
| 1002 | + | 00 | < 2^{-4} > | 0501 |
| 1003 | + | 00 | < 2^{-30} > | 1006 |
| 1004 | -, | 11 | 0501 | 0500 |
| 1005 | FU | 34 | 1006 | 1002 |
| 1006 | + | 00 | < 2^{-30} > | 0507 |
| | | | | |

A program az 501-es rekeszben rendre elhelyezi 2^{-4} többszöröseit. Azért nem 2^{-3} többszöröseit, mert ez túlcsoorduláshoz vezetne. Ennek megfelelően az 1000-es utasítás egy helyértékkel jobbra tolja az x számot.

Az 1004-es utasítás végzi el az összehasonlítást $\frac{x}{2}$ és 2^{-4} többszöröse között. A program tulajdonképpen ismétlődő ciklus, melynek utasításai fixek, és amely minden ismétlődés-kor módosítja az 1006-os utasítást (ez kívül van a cikluson), úgy, hogy végül a kívánt rekesz tartalmához adódjék hozzá 2^{-30} .

Ha megengedhető az 500-as rekeszben tárolt x szám elvesztése (elrontása), akkor a program négy utasításból állítható össze:

| | | | | |
|-----------|----|----|---------------------------|------|
| 1000 | + | 00 | $\langle 2^{-30} \rangle$ | 0103 |
| 1001 | - | 01 | $\langle 2^{-3} \rangle$ | 0500 |
| 1002 | FU | 34 | 1003 | 1000 |
| 1003 | + | 00 | $\langle 2^{-30} \rangle$ | 0507 |
| | | | | |

Az 1001-es utasítás $\frac{1}{8}$ -os lépésekkel csökkenti x -et, amíg az negatívvá nem válik. Minden ismétléskor hozzáadódik 1 az 1003-as utasítás második címéhez.

3. Ha $(0500) = 0$, adjuk össze az 501-es és 502-es rekeszek tartalmát és helyezzük el az összeget az 510-es rekeszben. Ha $(0500) = 2^{-30}$, akkor végezzük el ugyanezt a műveletet a megfelelő mennyiségek abszolút értékén.

Itt nyilvánvalóan a műveleti jel módosításáról lesz szó.

A program:

| | | | | |
|-----------|----|----|---------------------------|------|
| 1000 | ∴, | 12 | $\langle 2^{-29} \rangle$ | 0500 |
| 1001 | ↓+ | 30 | 1002 | 0102 |
| 1002 | +, | 10 | 0501 | 0502 |
| 1003 | U1 | 24 | ---- | 0510 |
| | | | | |

Ha az 500-as rekesz tartalma 2^{-30} , úgy az 1002-es utasítás műveleti jeléhez hozzáadódik $2^{-1} = 4 \cdot 2^{-3}$. Az új műveleti jel 50 lesz, ami a $i+1$, utasításnak felel meg. Ha $(0500) = 0$, az 1002-os utasítás változatlan marad.

4. Oldjuk meg az $ax^2 + bx + c = 0$ másodfokú egyenletet és helyezzük el a 600-as rekeszbe az x_1 gyököt, ha $p = 0$, az x_2 gyököt, ha $p = 1$, p illetve $p \cdot 2^{-30} = (0500)$. Ismeretes, hogy

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

A program menetrendjét a XIII. táblán láthatjuk.

A program

| | | | | | |
|------|-----|----|-------------------------------|---------------------------|--------------------|
| | | | | (0505) = 040000 | 0000 |
| 1000 | ∴, | 12 | $\langle 2^{-24} \rangle$ | 0500 | |
| 1001 | ↓+ | 20 | 1017 | 1017 | |
| 1002 | x, | 13 | $\langle a \rangle$ | $\langle ac \rangle$ | ac |
| 1003 | ↓∴ | 22 | $\langle \frac{1}{4} \rangle$ | 0501 | 4ac |
| 1004 | x, | 13 | $\langle b \rangle$ | $\langle b \rangle$ | b^2 |
| 1005 | ↓- | 21 | 0501 | 0502 | $b^2 - 4ac$ |
| 1006 | FU | 34 | 0505 | 1007 | } Négyzetgyökvonás |
| 1007 | Á | 05 | $\langle 1 - 2^{-30} \rangle$ | 0503 | |
| 1010 | ∴, | 12 | 0503 | 0502 | |
| 1011 | ↓- | 31 | 0503 | - | |
| 1012 | ↓x | 23 | $\frac{1}{2}$ | 0504 | |
| 1013 | ↓+ | 20 | 0503 | 0503 | |
| 1014 | ↓- | 51 | 0504 | $\langle 2^{-28} \rangle$ | |
| 1015 | FO | 34 | 1010 | 1016 | |
| 1016 | -, | 11 | $\langle b \rangle$ | $\langle c \rangle$ | |
| 1017 | ↓+ | 30 | 0503 | - | |
| 1020 | ↓x, | 33 | $\langle \frac{1}{2} \rangle$ | - | |
| 1021 | ↓∴ | 22 | $\langle a \rangle$ | 0600 | |
| . | . | . | . | . | . |

Megjegyzések:

- a.) Az 1007-től 1015-ig terjedő rekeszekben az ismert négyzetgyökvonó ciklust helyeztük el.
- b.) A gyök alatti kifejezés (az egyenlet diszkriminánsa) az 502-es rekeszben keletkezik, a négyzetgyök az 503-as rekeszben. Az 504-es rekesz a gyökvonó ciklus munka-rekesze, 505 tartalma pedig egy megállító utasítás. Erre ugratja a gépet az 1006-os feltételes ugró utasítás, ha a diszkrimináns negatív. Az 1017-es utasítás az 500-as rekeszben lévő paramétertől függ: ha a paraméter értéke 0, ez az utasítás nem változik, ha a paraméter értéke 2^{-30} , akkor kivonás lesz belőle (műveleti jele 31 lesz). Így a paraméternek megfelelő gyök kerül a 600-as rekeszbe.

Bemutatunk most egy olyan példát, amelynél a program által összeállítandó, majd végrehajtandó művelet nem egy, hanem két paramétertől függ. A feladat a következő:

5. Az 500-as rekesz tartalma: $a \cdot 2^{-30}$, ahol az a paraméter 5 különböző értéket vehet fel: $a = 0, 1, \dots, 4$. Az 501-es rekesz tartalma: $b \cdot 2^{-30}$, ahol a b paraméter 8 különböző értéket vehet fel: $b = 0, 1, \dots, 7$. Adjunk hozzá a 600 + 2a című rekesz tartalmához $4b \cdot 2^{-30}$ -t. Itt tehát változik a hozzáadandó szám és a rekesz is, amelynek tartalmához hozzáadunk. Mint látjuk, 40 különböző eset lehetséges. $b=0$ esetén gyakorlatilag nem történik semmi.

A program a következő:

```

(0400) = + 00 0401 0600
1000  :,  12  < 1/2 > 0500 2a · 2-30
1001  ↓+  20  0400  1004
1002  :,  12  < 1/4 > 0501 4b · 2-30
1003  U1  24  1004  0401
1004  ( . . . . . )

```

Az 1004-es utasítást, amely a kitűzött feladatot végrehajtja, a program állítja össze a 400-as rekeszben tárolt utasításkonstansból és az a paraméterből. Az 1004-es utasítás elhelyezése ("beültetése") az 1001-es utasítás által történik, de első címének (a 401-es rekesznek) a tartalmát, $4b$ -t, csak az 1002-es és 1003-as utasítások képezik és helyezik el. Amikor az 1004-es utasítás végrehajtására kerül, már minden szükséges előkészület megtörtént. A 401-es rekesz a program munkarekesze, eredeti tartalma közömbös (csakúgy mint az 1004-es rekesz eredeti tartalma).

6. Módosítsuk az előző feladatunkat a következőképpen: $600 + 2a$ tartalmához adjuk hozzá $700 + 4b$ tartalmát! Itt tehát a b paraméter is egy címet határoz meg (úgy, mint az a paraméter), nem pedig közvetlenül egy számadatot. A módosított feladat programja:

| | | | | | | | | | |
|------|----|----|-------------------------------|--------------|------|----------------------|--|--|--|
| | | | | | | | | | |
| | | | | (0400) = +00 | 0700 | 0600 | | | |
| 1000 | : | 12 | $\langle \frac{1}{2} \rangle$ | 0500 | 2a | a második cím helyén | | | |
| 1001 | ↓+ | 20 | 0400 | 1004 | | | | | |
| 1002 | : | 12 | $\langle 2^{-14} \rangle$ | 0501 | 4b | az első cím helyén | | | |
| 1003 | ↓+ | 20 | 1004 | 1004 | | | | | |
| 1004 | | | (.) | | | | | | |

Megjegyzés: Itt kihasználjuk, hogy a b paramétert 4-el, 2^2 -tel kell szorozni. Az 1002-es utasítás ezt a szorzást az első cím helyére való eltolással, azaz 2^{12} -vel való szorzással egyszerre végzi el.

Előző példánknál a $4b$ mennyiséget egy bizonyos címre kellett elhelyezni (a 401-es munkarekeszbe), hogy mint összeadandó szerepelhessen (az utasítások nem magukkal a számadatokkal, hanem azok címével dolgoznak!) Ennél a példánál erre nem volt szükség, mert mindkét paraméter címet módosított.

7. Oldjuk meg az 5. ill. 6. feladatot úgy, hogy mindkét paramétert egy szóban, az 500-as rekeszben helyezzük el. $(0500) = b \cdot 2^{-18} + a \cdot 2^{-30}$, azaz b az első, a a második cím helyén lesz. Szükségünk lesz a két paraméter szétválasztására (szelektálására), amihez a logikai szorzás műveletét és két további állandót, u.n. szűrőszámokat fogunk felhasználni.

A program:

| | | | | | |
|------|--------|-------------------------------|--------------|------|------|
| | | | (0400) = +00 | 0700 | 0600 |
| | | | (0401) = +00 | 0777 | 0000 |
| | | | (0402) = +00 | 0000 | 7777 |
| 1000 | ∧, 16 | 0402 | 0500 | | |
| 1001 | ↓:, 32 | $\langle \frac{1}{2} \rangle$ | 0500 | | |
| 1002 | ↓+, 20 | 0400 | 0501 | | |
| 1003 | ∧, 16 | 0401 | 1004 | | |
| 1004 | ↓:, 32 | $\langle \frac{1}{2} \rangle$ | 1004 | | |
| 1005 | ↓+ 20 | 1006 | 1004 | | |
| 1006 | (-- | ---- | ---- | | |

A 401-es és 402-es rekeszekben olyan konstansok vannak, amelyek az 1. ill. 2. cím helyén egyeseket, a többi helyen 0-okat tartalmaznak. Az ilyen típusú konstansokat, amelyek rendeltetése, hogy - logikai szorzás művelete által - egy adott szó bizonyos részét szelektálják "kiszűrjék", szűrőszámoknak nevezzük.

8. Az előző módszerrel oldható meg a következő feladat: Az 1500-tól 1577-ig terjedő rekeszek egy bizonyos programot tartalmaznak. Az 1520-as utasítás műveleti jelét, az 1540-es utasítás első címét és az 1560-as utasítás második címét bizonyos paraméterértékek hozzáadásával, a megfelelő utasítások végrehajtása előtt, módosítani kell. Célszerű a három módosító paramétert egyetlen szóban, egy utasításkonstansban egyesíteni, pl. az 500-as rekeszben, úgy hogy az első a műveleti jel helyén, a második az első cím helyén, a harmadik a második cím helyén legyen.

A program:

| | | | | | | | |
|------|----|----|------|------|------------|------|------|
| | | | | | (0501)=+77 | 0000 | 0000 |
| 1000 | Λ, | 16 | 0501 | 0500 | (0502)=+00 | 7777 | 0000 |
| 1001 | ↓+ | 20 | 1520 | 1520 | (0503)=+00 | 0000 | 7777 |
| 1002 | Λ, | 16 | 0502 | 0500 | | | |
| 1003 | ↓+ | 20 | 1540 | 1540 | | | |
| 1004 | Λ, | 16 | 0503 | 0500 | | | |
| 1005 | ↓+ | 20 | 1560 | 1560 | | | |
| 1006 | U2 | 74 | | 1500 | | | |

Az 1006-os utasítás, az előkészítés után, átadja a vezérlést az 1500-as utasítással kezdődő programnak.

9. 64 szám van elhelyezve a 100-tól 177-ig terjedő című rekeszekben. Egyik szám címe az 500-as rekesz második címrészének a helyén van. Helyezzük el ezt a számot a 600-as rekeszbe, a rákövetkező rekeszben lévő a 601-be, a rákövetkező rekeszben lévő a 602-be. vizsgáljuk meg, hogy az 500-as rekeszben elhelyezett cím nem nagyobb-e 175-nél, és ha igen, állítsuk meg a gépet. Ekkor ugyanis nem hajtható végre a feladat.

A program:

| | | | | | | | |
|------|------|-----|---------------------|---------------------------|------------|------|------|
| 1000 | - , | 51 | 0500 | <175 · 2 ⁻³⁰ > | (0501)=+05 | 0000 | 0600 |
| 1001 | FU | 34 | 0502 | 1003 | (0502)=+04 | 0000 | 0000 |
| 1002 | ∴, | 12 | <2 ⁻¹² > | 0500 | (0503)=+00 | 0001 | 0001 |
| 1003 | ↓+ | 20 | 0501 | 1006 | | | |
| 1004 | ↓+ | 20 | 0503 | 1007 | | | |
| 1005 | ↓+ | 20 | 0503 | 1010 | | | |
| 1006 | | (-- | ---- | ---- | | | |
| 1007 | | (-- | ---- | ---- | | | |
| 1010 | | (-- | ---- | ---- | | | |

.

Az 501-es rekesz tartalmazza azt az utasításkonstanst, amelynek segítségével a program összeállítja az 1006-os, 1007-es és 1010-es utasításokat. 502 tartalma egy megállító utasítás, 503 tartalma pedig egy konstans, amelyvel az első változó utasításból előállítható a másik kettő.

10. Tekintsünk olyan példát, amelynél a program nemcsak egyes utasításait változtatja, hanem az u.n. utasításkonstanst is egy paramétertől függően állítja be. Emlékezzünk vissza egy előző példánkra: a 101-től 150-ig terjedő című rekeszekben lévő 40 szám összeadására.

Újból leírjuk a fenti programot

```

                                (0502)=+00  0500  0500
1000      Á   05  <0>      0500
1001      +   00  0101     0500
1002      +   00  <2-18>  1001
1003      -,  11   1001     0502
1004      FU  34   1005     1001
1005      . . . . .
```

Feladatunkat most úgy módosítjuk, hogy a 101-es rekesz tartalmától kezdve egy $p \geq 1$ paraméterrel megadott számú rekesz tartalmát adjuk össze. Mint könnyen belátható: fenti programban az 502-es rekeszben elhelyezett utasításkonstanst kell a paramétertől függően kialakítani, mert ettől a konstanstól függ, hány számot adunk össze (persze ebben az esetben ebből a mennyiségből nem konstans, hanem változó lesz.) Legyen a paraméter a 600-as rekeszben, az első cím helyén elhelyezve.

Az új program:

| | | | | (0502)= +00 | 0100 | 0500 |
|------|----|----|---------------------|-------------|------|------|
| 1000 | + | 00 | 0600 | | | 0502 |
| 1001 | Á | 05 | <0> | | | 0500 |
| 1002 | + | 00 | 0101 | | | 0500 |
| 1003 | + | 00 | <2 ⁻¹⁸ > | | | 1001 |
| 1004 | -, | 11 | 1001 | | | 0502 |
| 1005 | FU | 34 | 1006 | | | 1002 |
| 1006 | . | . | . | . | . | . |

5. Fejezet

CIKLIKUS PROGRAMOK VÁLTOZÓ UTASÍTÁSOKKAL.

5.1. Utasítások visszaállítása beültetéssel.

Miután megismerkedtünk az utasítások módosításának és összeállításának módszereivel és legfontosabb alkalmazásaival, rátérhetünk az általános (vagy változó utasítású) ciklusok tanulmányozására.

A ciklusok fontossága kiderül abból a megfontolásból, hogy egy direkt program, amely teljesen megtöltené a gép memóriáját, még a relatíve lassú M-3 gép esetén is kb. 1 perc alatt lefutna. Ezzel szemben nagyobb számítási feladatok programjai órákat, esetleg napokat is igénybevesznek.

Vegyük elő ismét a 4. fejezetben ismerttetett első példánkat, a 101-től 150-ig terjedő című rekeszekben tárolt 40 szám összeadását és az összeg elhelyezését az 500-as rekeszben.

| | | | | |
|------|----|----|---------------------|------------------------|
| | | | | (0502)= + 00 0150 0500 |
| 1000 | Á | 05 | <0> | 0500 |
| 1001 | + | 00 | 0101 | 0500 |
| 1002 | + | 00 | <2 ⁻¹⁸ > | 1001 |
| 1003 | -, | 11 | 1001 | 0502 |
| 1004 | FU | 34 | 1005 | 1001 |
| 1005 | . | . | . | . |

Megjegyeztük a fenti programmal kapcsolatban, hogy elvégezni ugyan a kitűzött feladatot, de újra nem használható fel, mert a változó utasítás, az 1001-es, más formában marad, mint amilyenben a program kezdetén volt. Ez a probléma azáltal oldható meg pl., hogy a változó utasítást kezdetben, tehát legelső formájában maga a program helyezi el.

A módosított program:

| | | | | | | | | |
|------|----|------|---------------------|------|---------------|----------|------|--|
| | | | | | (0502)= + 00 | 0150 | 0500 | |
| | | | | | (0503)= + 00 | 0101 | 0500 | |
| 1000 | Á | 05 | <0> | 0500 | } előkészítés | } ciklus | | |
| 1001 | Á | 05 | 0503 | 1002 | | | | |
| 1002 | (| -- | ---- | ---- |) számolás | | | |
| 1003 | + | 00 | <2 ⁻¹⁸ > | 1002 | módosítás | | | |
| 1004 | - | , 11 | 1002 | 0502 | } vizsgálat | | | |
| 1005 | FU | 34 | 1006 | 1002 | | | | |
| 1006 | - | - | - | - | | | | |

Ez a program egy utasítással hosszabb az előzőnél, és egy konstanssal (utasítás-konstanssal, "pszeudo-utasítással") többet tartalmaz. A változó utasítás kiinduló alakját az 503-as rekesz tárolja, változatlan állandóként (pszeudo-utasításként), és ezt az 1001-es utasítás helyezi el az 1002-es rekeszbe, a következő végrehajtandó utasítás helyére. Az utasításmódosítás az 1002-es helyen tárolt utasítást érinti, minden ismétlődés alkalmával, az ismétlődések az 1002-es utasítással kezdődnek. Ha ezt a részprogramot újra felhasználjuk, elejétől, az 1000-es utasítástól kezdve, úgy ismét megtörténik az 1002-os változó utasításnak eredeti formában való kitöltése. Ez azt jelenti, hogy a program ismételten használható, azaz tökéletes ciklikus programmal van dolgunk.

Ha ennek különböző részeit elemezzük, a következőket találjuk:

Az első két utasítás a tulajdonképpeni cikluson kívül áll, és azt a célt szolgálja, hogy bizonyos kezdeti értékeket helyezzen el. Ez az u.n. előkészítő rész. A többi utasítás mind a ciklushoz tartozik, azaz minden ismétlődés alkalmával végrehajtásra kerül. Az 1002-es változó utasítás a program tulajdonképpeni számoló része. A következő utasítás (1005-as) alkotja a módosítási részt, majd az utolsó két utasítás az u.n. vizsgáló részt. Ez utóbbi vizsgálja meg, szükség van-e még a ciklus megismétlésére, és ha igen, átadja a vezérlést a ciklus elejére.

A fenti négy rész minden változó utasítású ciklusos programban megtalálható. Természetesen mindegyik rész több utasításból is állhat, mint jelenlegi egyszerű példánkban. Lehetséges a cikluson belüli egyes részeket felcserélni.

P1. fenti példánk esetében:

| | | | | | | | | |
|------|----|----|---------------------|------|---------------|----------|------|---|
| | | | | | (0502)= + 00 | 0147 | 0500 | |
| | | | | | (0503)= + 00 | 0100 | 0500 | |
| 1000 | Á | 05 | <0> | 0500 | } előkészítés | | | |
| 1001 | Á | 05 | 0503 | 1005 | | | | |
| 1002 | -, | 11 | 1005 | 0502 | } vizsgálat | } ciklus | | |
| 1003 | FU | 34 | 1007 | 1004 | | | | |
| 1004 | + | 00 | <2 ⁻¹⁸ > | 1005 | módosítás | | | |
| 1005 | (| -- | ---- | ---- |) számolás | | | |
| 1006 | U2 | 74 | ---- | 1002 | | | | |
| 1007 | . | . | . | . | . | . | . | . |

Mint látjuk, ebben az esetben eggyel több utasításra van szükségünk, és az utasításkonstansokat is meg kellett változtatni, különben a program a 102-től 151-ig terjedő rekeszek tartalmát adná össze.

Általában nagyon kell ügyelni arra, hogy a vizsgálat céljára felhasznált számláló, vagy utasításkonstans helyesen van-e megállapítva. Programozási hiba esetén könnyen lehetséges, hogy a ciklus n-szeres ismétlődés helyett

n-1-szer, n+1-szer, 2n-szer, egyáltalán nem; vagy vég nélkül ismétlődik. Ezért célszerű mindig megvizsgálni, hogy mi történék adott esetben, ha a ciklus egyszeri ismétlődést programoznák be.

5.2. A program visszaállítása utókorrekcióval.

Van olyan programozási módszer is, amelynél a változó utasítások nem a program elején (az előkészítő részben) lesznek kitöltve, hanem: eredeti alakjukban már a helyükön vannak, a ciklusok folyamán változnak, majd a ciklust követő programrész - az u.n. helyreállító rész - eredeti alakjukban állítja vissza őket. - Ebben az esetben is lehetséges a program ismételt felhasználása.

Fenti példánk tehát így is átírható:

| | | | | | | | | |
|------|-----|----|---------------------|------|---------------|------|------|----------|
| | | | | | (0502)= + 00 | 0150 | 0500 | |
| | | | | | (0503)= + 00 | 0050 | 0000 | |
| 1000 | Á | 05 | <0> | 0500 | előkészítés | | | |
| 1001 | + | 00 | 0101 | 0500 | számlás | | | } ciklus |
| 1002 | + | 00 | <2 ⁻¹⁸ > | 1001 | módosítás | | | |
| 1003 | - , | 11 | 1001 | 0502 | vizsgálat | | | |
| 1004 | FU | 34 | 1005 | 1001 | | | | |
| 1005 | - | 01 | 0503 | 1001 | helyreállítás | | | |

Meg kell jegyezni, hogy az előző módszer az előnyösebb, amelynél a változó utasítások egy változatlan "őskép" szerint kerülnek kitöltésre a program elején. Ezzel kombinálható a változó utasítások bármilyen időközbeni "elromlása".

Mint látjuk, a ciklusos program előkészítő része csak egyszer kerül végrehajtásra, míg a tulajdonképpeni ciklus igen sokszor ismétlődhetik. Ezért esetenként célszerű az előkészítő részt tetemesen megnövelni, ha ezáltal a ciklus maga (esetleg csak kisebb mértékben)

lerövidíthető. Ez olyankor lehetséges, amikor egy bizonyos számítás több alternatív úton (több formula szerint) végezhető el. Minden esetben bizonyos kompromisszumra kell törekedni a program hossza (utasítások száma) és végrehajtásának időtartama között. A két követelmény ellentmondó. Maga a ciklusos programozás lényegesen megnöveli a feladat végrehajtásának idejét (esetleg tízszeresére is) a program rövidsége kedvéért. Ez példánkból is meglátszik, ahol a tulajdonképpeni számolást csak egyetlen utasítás végzi, a többi a ciklus érdekében van ott. Direkt programozás esetén 40 összeadó utasításra lenne szükség, de a végrehajtás ideje sokkal kisebb lenne. A program hossza és végrehajtásának ideje közötti kompromisszumot aszerint kell megállapítani, hogy adott esetben melyik szempont a kritikusabb.

Feladatok:

1. Szerkesszük meg a fenti programozási változatokat a következő feladatra:

$$(n) \cdot (n+1) - (n+2) \implies n \quad (n = 1201, 1202, \dots, 1300)$$

2. Szerkesszük meg a 40 szám összeadásának programját úgy, hogy a ciklus számoló része - az első végrehajtáskor - a következő két utasítás legyen:

| | | | |
|----|----|------|------|
| +, | 10 | 0101 | 0102 |
| ↓+ | 20 | 0500 | 0500 |

Ezáltal a ciklus maga egy utasítással hosszabb, de csak feleannyi (40 helyett 20) ismétlésre van szükség (minden ismétlésnél nem egy, hanem két számot adunk hozzá a részletösszeghez).

5.3. Gyakorlati példák.

Lássunk egy gyakorlatilag igen fontos példát, a lineáris interpoláció példáját. Tegyük fel, hogy az $y = f(x)$ függvénynek bizonyos táblázata benne van a gép memóriájában, a következőképpen:

$$\begin{array}{ll} (0501) = x_1 & (0601) = y_1 \\ (0502) = x_2 & (0602) = y_2 \\ \vdots & \vdots \\ (0600) = x_{100} & (0700) = y_{100} \end{array}$$

x és y indexeit, akárcsak a rekeszek címeit, 8-as számrendszerben ábrázoljuk. A táblázat tehát a függvénynek a független változó 64 különböző helyéhez tartozó értékeit tartalmazza. Feltételezzük, hogy az "x-ek" növekvő sorrendben elhelyezett nem negatív számok, az y -okról semmilyen megszorítást nem tételezünk fel.

Legyen a 300-as rekeszben x -nek egy ismert értéke, feladatunk az, hogy az adott táblázat alapján lineáris interpolációval meghatározzuk a hozzátartozó y értéket és elhelyezzük a 400-as rekeszbe. Mivel csak olyan x esetén lehet interpolálni, amely a táblázatban szereplő értelmezési tartományon belül van, a programnak meg kell vizsgálnia, hogy ez a feltétel teljesül-e és ha nem, meg kell állítania a gépet.

A lineáris interpoláció képlete:

$$y = y_n + \frac{y_{n+1} - y_n}{x_{n+1} - x_n} (x - x_n)$$

ahol x_{n+1} és x_n az adott x -et közrefogó táblázati független változó értékek, $x_n \leq x \leq x_{n+1}$, y_{n+1} és y_n az ezeknek megfelelő függvény értékek.

A számítás blokk-diagramját a XIV. tábla első ábrája mutatja.

A második blokk, x_n és x_{n+1} meghatározása ciklust képez.

A program:

| | |
|---------------|-----------|
| (0301) = + 04 | 0000 0300 |
| (0302) = + 11 | 0300 0501 |
| (0303) = + 00 | 0000 7777 |
| (0304) = + 05 | 0000 0351 |
| (0305) = + 00 | 0001 0001 |
| (0306) = + 00 | 1000 1002 |

| | | | | | |
|------|-----|----------|---------------------------|------|------------------------|
| 1000 | -, | 11 | 0501 | 0300 | |
| 1001 | FU | 34 | 0301 | 1002 | |
| 1002 | -, | 11 | 0300 | 0600 | |
| 1003 | FU | 34 | 0301 | 1004 | |
| 1004 | Á | 05 | 0302 | 1006 | |
| 1005 | + | 00 | $\langle 2^{-30} \rangle$ | 1006 | |
| 1006 | | (-- ---- | ---- | | |
| 1007 | FU | 34 | 1005 | 1010 | |
| 1010 | ∧, | 16 | 1006 | 0303 | |
| 1011 | ↓:, | 32 | $\langle 2^{-12} \rangle$ | | |
| 1012 | ↓+ | 20 | 0304 | 1010 | |
| 1013 | ↓- | 21 | 0305 | 1017 | |
| 1014 | ↓+ | 20 | 0306 | 1020 | |
| 1015 | ↓+ | 20 | 0305 | 1021 | |
| 1016 | | (-- ---- | ---- | | $x_{n+1} \implies 351$ |
| 1017 | | (-- ---- | ---- | | $x_{n+1} \implies 351$ |
| 1020 | | (-- ---- | ---- | | $x_{n+1} \implies 351$ |
| 1021 | | (-- ---- | ---- | | $x_{n+1} \implies 351$ |

| | | | | | |
|------|-----|----|------|------|---------------------------------------|
| 1022 | - | 01 | 0350 | 0351 | ← (x _{n+1} -x _n) |
| 1023 | - | 01 | 0350 | 0353 | ← (y _{n+1} -y _n) |
| 1024 | - | 01 | 0350 | 0300 | (x-x _n) |
| 1025 | ↓x, | 33 | 0353 | | |
| 1026 | ↓:, | 32 | 0351 | | |
| 1027 | ↓+ | 20 | 0352 | 0400 | |

.

A követhetőség kedvéért egyes utasítások mellé odaírtuk, mi történik az utasítások révén. Ha az illető utasítás, eredménye beíródik a 2. címre, ezt egy nyíllal jelöltük (az 1022-es és 1023-as utasításnál). Az első négy utasítás megvizsgálja, hogy x a kérdéses tartományban van-e. Ha nem, úgy a gép megáll a 301-es megállító utasítás miatt. Az 1004-től 1007-ig terjedő utasítások x_{n+1} megkeresésére szolgálnak (1004-es: előkészítő rész, 1005–1007-ig ciklus). Az 1010-től 1012-ig terjedő utasítások x_{n+1} címének leválasztásával képeznek egy átviteli utasítást, az 1013-tól 1015-ig terjedő utasítások további három átviteli utasítást képeznek. Ez a négy átviteli utasítás helyezi el az x_n, x_{n+1}, y_n, y_{n+1} értékeket. végül következik az interpolációs formula kiszámítása.

További példák:

1. A 100-tól 150-ig terjedő című rekeszek 41 számot tartalmaznak. Képezzük a szomszédos rekeszekben levő számok első differenciáit és helyezzük el azokat a 200-tól 247-ig terjedő rekeszekben.

A program:

| | | |
|--------|--------|-----------|
| (0500) | = + 11 | 0100 0101 |
| (0501) | = + 24 | 1004 0200 |
| (0502) | = + 24 | 1004 0247 |
| (0503) | = + 00 | 0001 0001 |

| | | | | | | | | |
|------|----|-----|---------------------|------|---|-------------|---|--------|
| 1000 | Á | 05 | 0500 | 1002 | } | előkészítés | } | ciklus |
| 1001 | Á | 05 | 0501 | 1003 | | | | |
| 1002 | | (-- | ---- | ---- | } | számolás | | |
| 1003 | | (-- | ---- | ---- | | | | |
| 1004 | + | 00 | 0503 | 1002 | } | módosítás | | |
| 1005 | + | 00 | <2 ⁻³⁰ > | 1003 | | | | |
| 1006 | -, | 11 | 1003 | 0502 | } | vizsgálat | | |
| 1007 | U2 | 74 | ---- | 1002 | | | | |
| 1010 | . | . | . | . | . | . | | |

2. Legyen két 16-dimenziós vektor a memóriában a következőképpen: az első vektor komponenseit a 101-től 120-ig terjedő rekeszek, a második vektor komponenseit a 121-től 140-ig terjedő rekeszek tárolják. Helyezzük el a két vektor skaláris szorzatát az 500-as rekeszben. Mint ismeretes, két $\underline{x} = (x_1, x_2, \dots, x_n)$ és $\underline{y} = (y_1, y_2, \dots, y_n)$ vektor skaláris szorzata az $x_1y_1 + x_2y_2 + \dots + x_ny_n$ mennyiség.

A program:

(0501) = + 13 0101 0121
 (0502) = + 00 0101 0101
 (0503) = + 00 0120 0140

| | | | | | | | | |
|------|----|-----|------|------|---|-------------|---|--------|
| 1000 | Á | 05 | <0> | 0500 | } | előkészítés | } | ciklus |
| 1001 | Á | 05 | 0501 | 1002 | | | | |
| 1002 | | (-- | ---- | ---- | } | számolás | | |
| 1003 | ↓+ | 20 | 0500 | 0500 | | | | |
| 1004 | + | 00 | 0502 | 1002 | } | módosítás | | |
| 1005 | -, | 11 | 1002 | 0503 | | | | |
| 1006 | FU | 34 | 1007 | 1002 | } | vizsgálat | | |
| 1007 | -- | -- | ---- | ---- | | | | |

Az 500-as rekeszben nemcsak a végeredményt, de a részletösszeget is elhelyezzük (az 500-as: u.n. gyűjtőrekesz, akkumulátor szerepét játssza.)

5.4. Többszörös ciklusok.

Következő példánk a többszörös ciklusokat fogja illusztrálni. Így nevezzük azokat a programokat, amelyeknél egy ciklusnak beépített része egy másik ciklus, annak esetleg ismét egy további ciklus a része sít. Egy háromszoros ciklus blokkdiagramja például a XIV. tábla második ábráján látható.

A műveleti idők becslése céljából megemlítjük, hogy ha e sémában az A ciklus a-szor, a B ciklus b-szer, a C ciklus c-szer ismétlődik, akkor a C ciklus B minden egyes ismétlődésekor c-szer. A minden ismétlődésekor b·c-szer, az egész programban tehát a·b·c-szer kerül végrehajtásra.

Konkrét példaként programozzuk az alábbi problémákat:

1. 16 db 16 dimenziós vektor a gép memóriájában a következőképpen helyezkedik el: az első vektor komponensei a 0-tól 17-ig, a második vektor komponensei a 20-tól 37-ig, ..., az utolsó vektor komponensei a 360-tól 377-ig terjedő című rekeszekben. Képezzük az első vektor skaláris szorzatát a kilencedikkel, tizedikkel, ..., tizenhatodikkal, majd a második vektor skaláris szorzatát a kilencedikkel, tizedikkel, ..., tizenhatodikkal, sít. végül a nyolcadik vektor skaláris szorzatát a kilencedikkel, tizedikkel-, tizenhatodikkal és helyezzük el a kapott eredményeket – összesen 64 számot – a 400-tól 477-ig terjedő rekeszekben.

Mint könnyen belátható, ez háromszoros hurok.

A program:

```
(0501) = + 12  7760  0200
(0502) = + 00  0001  0001
(0503) = + 05  0603  0400
```

| | | | | | |
|------|----|-----|------------------------|------|---|
| 1000 | Á | 05 | <0> | 0601 | |
| 1001 | Á | 05 | 0503 | 1015 | |
| 1002 | Á | 05 | <7·2 ⁻³⁰ > | 0604 | |
| 1003 | Á | 05 | <7·2 ⁻³⁰ > | 0600 | |
| 1004 | + | 20 | <16·2 ⁻¹⁸ > | 0601 | |
| 1005 | ↓+ | 20 | 0501 | 1010 | |
| 1006 | Á | 05 | <15·2 ⁻³⁰ > | 0602 | |
| 1007 | Á | 05 | <0> | 0603 | |
| 1010 | | (-- | ---- | ---- | |
| 1011 | ↓+ | 20 | 0603 | 0603 | |
| 1012 | FU | 34 | 0502 | 1010 | |
| 1013 | - | 01 | <2 ⁻³⁰ > | 0602 | |
| 1014 | FU | 34 | 1015 | 1010 | |
| 1015 | | (-- | ---- | ---- | |
| 1016 | + | 00 | <2 ⁻³⁰ > | 1015 | |
| 1017 | - | 01 | <16·2 ⁻¹⁸ > | 1010 | |
| 1020 | - | 01 | <2 ⁻³⁰ > | 0600 | |
| 1021 | FU | 34 | 1022 | 1006 | |
| 1022 | - | 01 | <2 ⁻³⁰ > | 0604 | |
| 1023 | FU | 34 | 1024 | 1003 | |
| 1024 | . | . | . | . | . |

Megfelelő betűkkel jeleztük a XIV. táblán lévő blokk-diagram szerinti egymásba skatulyázott ciklusokat. Az 1010-es utasítás a mindhárom ciklus szerint változó számoló utasítás. Első formája x, 13 0000 0200. A 601-es munkarekesz a változtató konstansok előállítására, a 600, 602, 604-es rekeszek a három ciklusszámlálásra szolgálnak.

Feladat: Elemezzük az egymásba skatulyázott A,B,C ciklusokat.

2. Az alábbi számsémát helyeztük el a gép memóriájában:

```

a11  a12.....a1n
a11  a12.....a1n
a11  a12.....a1n
    
```

Az első sor számértékeit a 100-tól 100+n-1-ig terjedő rekeszek, a második sor számértékeit a 200-tól 200+n-1-ig terjedő rekeszek, a harmadik sor számértékeit a 300-tól 300+n-1-ig terjedő rekeszek tárolják.

$$n \cdot 2^{-30} = (0500)$$

Feladatunk: $\frac{1}{4} (a_{1,i} + a_{2,i-1} + a_{2,i+1} + a_{3,i}) \implies a_{2,i}$

($i = 2, \dots, n-1$), azaz: helyettesítsük a számséma belső elemeit a körülöttük lévő négy szám számtani közepével.

A program:

```

(0501) = + 23  <1/4> 0200
(0502) = + 10  0300 0501
(0503) = + 00  0000 7777
(0504) = + 10  0000 0351
(0505) = + 13  1000 1002
    
```

```

1000  Á    05  0200    0600
1001  Á    05  0501    1011
1002  Á    05  0502    1005
1003  -,   11  <2·2-30> 0500
1004  U1   24  1012    0602 ← (n-2)·2-30
1005      (-- ----)
1006      (-- ----)
1007      (-- ----)
1010      (-- ----)
1011      (-- ----)
1012  +    00  <2-12> 1005
1013  ↓+   20  0503    1010
    
```

| | | | | |
|------|----|----|----------------------------|------|
| 1014 | ↓+ | 20 | 0505 | 1006 |
| 1015 | ↓- | 21 | 0505 | 1007 |
| 1016 | + | 00 | $\langle 2^{-3^0} \rangle$ | 1011 |
| 1017 | - | 01 | $\langle 2^{-3^0} \rangle$ | 0602 |
| 1020 | FU | 34 | 1021 | 1005 |
| 1021 | . | . | . | . |

Ennél a feladatnál ügyelni kell arra, hogy miután a második sor elemeit felcseréljük az újonnan kiszámított szám-tani közepekkel, a régi elemekre a következő lépés számítása céljából még szükség van. Ezért a régi elemeket ideiglenesen a 600-as rekesz tárolja. Az 1005-től 1011-ig terjedő változó utasításokat az 1012-től 1016-ig terjedő utasítások töltik ki, illetőleg változtatják meg minden ciklus-ismétlődés alkalmával. A program a ciklus első lefutásakor átugorja a még részben kitöltetlen 1005-1011 utasításokat. Az 501-505-ös rekeszekben lévő konstansok a változó utasítások össze-állítását szolgálják. A ciklusszámlálás a 602-es rekeszben tárolt számlálóval történik.

6. Fejezet

A SZÁMOK FIXPONTOS ÁBRÁZOLÁSBÓL ADÓDÓ NUMERIKUS PROBLÉMÁK.

6.1. A fixpontos ábrázolásról.

Mint ismeretes, az M-3 gépben csak olyan számok ábrázolhatók, amelyek abszolút értékben egynél kisebbek, pontosabban szólva, amelyek a $-(1-2^{-30}) \leq x \leq 1-1-2^{-30}$ intervallumba esnek. A bináris pontot ugyanis megállapodásszerűen a szó egy rögzített helyére, közvetlenül az előjel után helyeztük el, azaz az előjel után következő első bitet a 2^{-1} , a második bitet a 2^{-2} , s.í.t. a harmincadik bitet a 2^{-30} helyértéknek feleltettük meg, és így egy a gépben ábrázolt $x = \pm a_1, a_2 \dots a_{30}$ számnak az $x = \pm(a_1 2^{-1} + \dots + a_{30} 2^{-30})$ értéket tulajdonítjuk, ($a_i = 0$, vagy 1)

A bináris pont szimbolikusan a szó más helyén is elhelyezhető, de a gép mindig az előbbiek szerint "értelmezi" a számokat, t.í. a szorzás és osztás instrumentálása a számok fenti alakjának megfelelően történik. (Az összeadás és kivonás szempontjából a bináris pont bárhol lehet.)

Ha egy művelet eredménye abszolútértékben meghaladja az $1-2^{-30}$ -at, azt mondjuk, hogy "túlcsordulás" lép fel, ilyenkor - mint tudjuk - az M-3 gép megáll.

A bináris pontnak ez a rögzítettsége azt jelenti, hogy az M-3 fixpontos gép.

Vannak u.n. lebegőpontos gépek is. A lebegőpontos gépekben a számok ábrázolása a számoknak u.n. lebegőpontos normalizált alakjukban való felírásán alapszik.

Egy x szám lebegőpontos alakján kettes számrendszer esetén az

$$x = d, 2^p \text{ alakot értjük,}$$

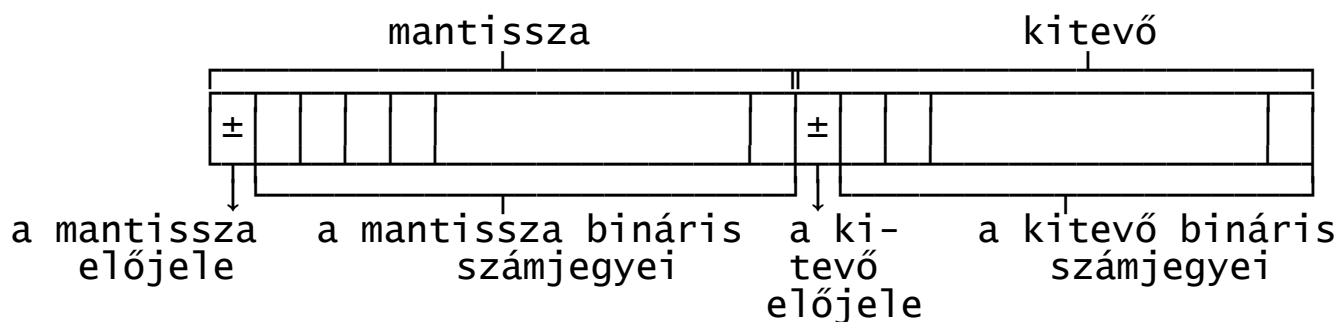
ahol $d = \pm (a_1 2^{-1} + a_2 2^{-2} + \dots + a_n 2^{-n})$ $a_i = 0, \text{ vagy } 1$
számot az illető szám mantisszájának; a

$$p = \pm (b_1 2^m + \dots + b_{m+1} 2^0) \quad b_i = 0, \text{ vagy } 1$$

számot pedig a szám kitevőjének nevezzük. Az adott x szám normalizált, ha

$$\frac{1}{2} \leq d < 1 \quad a_1 = 1$$

A lebegőpontos gépekben minden szó két részből áll, az egyik részben a mantisszát, a másik részben a kitevőt ábrázoljuk:



Az így ábrázolt számokkal a gép az aritmetikai szabályoknak megfelelően végzi az adott műveleteket külön a mantisszával, külön a kitevővel.

A két géptípus közti különbség nem fejlődési fokozat. A programozás szempontjából a lebegőpontos gépek kényelmesebbek, szerkezetileg viszont bonyolultabbak; a fixpontos gépekkel nagyobb gyorsaság érhető el.

Mint ahogy az M-3 gép csak $1-2^{-30}$ -nál abszolútértékben nem nagyobb számokkal tud számolni, s a gyakorlatban előforduló számításokban szereplő adatokra ez a feltétel általában nem teljesül, felvetődik az a probléma, hogyan lehet ezeket a feladatokat az M-3 gép segítségével megoldani.

A fixpontból adódó nehézséget lényegében kétféleképpen lehet megkerülni:

- a.) lebegőpontos programozás módszerével
- b.) transzformációs módszerekkel.

A lebegőpontos programozás lényegében a lebegőpontos gépek működésének egy program segítségével való utánozását jelenti. Ez esetben a számokat az előzőekben felírt lebegőpontos alakjukban ábrázoljuk, mégpedig vagy úgy, hogy egyetlen szóban ábrázoljuk a mantisszát és kitevőt is, vagy pedig külön-külön szóban ábrázoljuk a mantisszát és a kitevőt. A műveleteket a lebegőpontos alakban felírt számokra vonatkozó aritmetikai szabályoknak megfelelően kell programozni. (Pl. szorzásnál összeszorozzuk a mantisszákat, s összeadjuk a kitevőket stb.)

Ez a módszer mindig alkalmazható ugyan, de a végrehajtási idő és a memóriarekesz-szükséglet lényeges megnöveléséhez vezet, ezért - hacsak lehet - más módszert használunk. A későbbiekben ezt a módszert részletesen tárgyalni fogjuk.

6.2. Transzformációs módszerek.

A transzformációs módszereknek az a lényegük, hogy a programozásra kerülő feladatot (képletet) úgy alakítjuk át, hogy az abban szereplő adatok - eredmények, részeredmények - abszolút értéke egynél kisebb legyen. Szemléletesen arról van szó, hogy ha egy feladat számadatai egy, az egységtartománynál nagyobb tartományban helyezkednek el, akkor ezt a tartományt az egységtartomány belsejébe zsugorítjuk.

A szóbanforgó transzformáció többféle lehet; összefoglaljuk azt a néhány módszert, amelyek általában alkalmazhatók. Speciális transzformációk esetleg feladatonként választhatók. Megjegyezzük, hogy a feladatnak az előbbi értelemben vett transzformálása a programozási munka jelentős részét teszi ki, és nem mindig könnyen megoldható probléma.

6.2.1 skálafaktor-módszer.

A skálafaktor-módszer a feladatok transzformálásának legáltalánosabban használt módszere.

Tekintsünk egy x számot. Ha erre a számra $10^{q-1} \leq x < 10^q$ ($q=0, \pm 1, \pm 2, \dots$) akkor $\bar{x} = \frac{x}{10^q}$ nyilván olyan szám, amelyre $0,1 \leq \bar{x} < 1$. Az \bar{x} számot az x szám gépi - (vagy normált) alakjának, a 10^q mennyiséget pedig skálafaktornak (vagy normálótényezőnek) nevezzük. Egy a gépen ábrázolt \bar{x} szám tényleges x értéke tehát a hozzátartozó skálafaktortól függ az $x = 10^q \bar{x}$ egyenlőségnek megfelelően.

Természetesen skálafaktorként nemcsak 10 hatványait választhatjuk. Ha $1 < |x| < M$, akkor az $\bar{x} = \frac{x}{M}$ számra már teljesül $0 < |\bar{x}| < 1$. A skálafaktornak az utóbbi módon való megválasztása azért előnyös, mert nem "zsugorítja" felesleges mértékben az adott számot. (A zsugorítás általában jegyvesztéssel jár.)

A szám \bar{x} gépi alakjának előállítására viszont kényelmesebb 10 hatványok segítségével. Általában úgy járunk el, hogy a gépben tárolt kiinduló adatokat és a kinyomtató adatokat 10 hatványaival normáljuk, s a gépen belül végzett műveleteknél fellépő túlcsoordulást M -típusú skálafaktoral kerülnünk el. Többnyire célszerű $M = 2^p$ -t választani.

A feladatoknak skálafaktor módszerrel való transzformálását a következő lépésekben végezzük el.

1. Megállapítjuk a képletben szereplő adatok alsó és felső korlátját, s megválasztjuk a megfelelő skálafaktorokat.
2. Felírjuk az adott képletet a számok gépi alakja segítségével, s elvégezzük a lehetséges egyszerűsítéseket.
3. Nyilvánvaló, hogy ha a formulában szereplő kiinduló adatok nem is csordulnak túl, lehetséges, hogy az ezekkel végzett műveletek eredményei túlcordulnak. Meg kell tehát vizsgálni az eredményként létrejövő adatoknak a műveletben szereplő adatoktól függő alsó és felső korlátját, s újabb skálafaktorok bevezetésével el kell érni, hogy az adott művelet elvégezhető legyen. Azaz pl. osztásnál a nevező nagyobb legyen a számlálónál, összeadásnál és kivonásnál a skálafaktor közös legyen, stb.
4. Meg kell vizsgálni, hogy a formulában szereplő skálafaktorok az eredményt hogyan módosítják; fel kell jegyezni, hogy az eredményhez milyen skálafaktor tartozik. Lássunk egy példát:

Legyen a programozandó képlet a következő:

$$F(a, b, c, x, y) = \frac{\sqrt{ax} + c}{\frac{b}{d} + 121y} - \frac{d}{a}$$

$$1 < a < 8$$

$$125 < b < 705$$

$$5 < c < 10$$

$$1 < d < 5$$

$$50 < x < 100$$

$$0,01 < y < 0,9$$

- a.) Az adatok alsó és felső korlátját megadtuk; normált alakjuk a következő:

$$\bar{a} = \frac{a}{10} \quad \bar{c} = \frac{c}{10} \quad \bar{x} = \frac{x}{10^2}$$

$$\bar{b} = \frac{b}{10^3} \quad \bar{d} = \frac{d}{10} \quad \bar{y} = y$$

b.) Felírjuk a képletet a számok normált alakja segítségével:

$$F = \frac{\sqrt{10 \bar{a} 10^2 \bar{x}} + 10 \bar{c}}{\frac{\bar{b} \cdot 10^3}{\bar{d} \cdot 10} + 121 \bar{y}} - \frac{\bar{d} \cdot 10^3}{\bar{a} \cdot 10}$$

Elvégezzük a lehetséges egyszerűsítéseket

$$F = \frac{10 \sqrt{10 \bar{a} \bar{x}} + 10 \bar{c}}{\frac{\bar{b} \cdot 10^2}{\bar{d}} + 121 \bar{y}} - \frac{\bar{d}}{\bar{a}} = \frac{10 \sqrt{10 \bar{a} \bar{x}} + \bar{c}}{\frac{\bar{b} \cdot 10^2}{\bar{d}} + 12,1 \bar{y}} - \frac{\bar{d}}{\bar{a}}$$

c.) Vizsgáljuk meg a részeredmények túlcsoordulási lehetőségeit: számítsuk ki felső korlátjaikat, s válasszuk meg az újabb skálafaktorokat.

$$\max (\sqrt{10 \bar{a} \bar{x}} + \bar{c}) = \sqrt{10} \sqrt{\max \bar{a} \max \bar{x}} + \max \bar{c} < \sqrt{10} + 1 < 5$$

$$\begin{aligned} \max \left(\frac{10 \bar{b}}{\bar{d}} + 12,0 \bar{y} \right) &= \frac{10 \max \bar{b}}{\min \bar{d}} + 12,0 \max \bar{y} < \\ &< \frac{10 \cdot 0,705}{0,1} + 12,1 \cdot 0,9 < 100 \end{aligned}$$

$$\max \left| \frac{\bar{d}}{\bar{a}} \right| = \frac{\max |\bar{d}|}{\min |\bar{a}|} < \frac{0,5}{0,1} = 5.$$

Az első tört maximuma a nevező minimumától függ, ezt is meg kell tehát határozni.

$$\begin{aligned} \min \left(\frac{10 \bar{b}}{\bar{d}} + 12,1 \bar{y} \right) &= \frac{10 \min \bar{b}}{\max \bar{d}} + 12,1 \min \bar{y} = \\ &= \frac{10 \cdot 0,125}{0,5} + 12,1 \cdot 0,01 > 2. \end{aligned}$$

Így tehát:

$$A = \max \frac{\sqrt{10} \bar{a} \bar{x} + 10 \bar{c}}{\frac{10 \bar{b}}{\bar{d}} + 12,1 \bar{y}} = \frac{\max (\sqrt{10} \bar{a} \bar{x} + \bar{c})}{\min (\frac{10 \bar{b}}{\bar{d}} + 12,1 \bar{y})} < \frac{5}{2} = 2,5$$

$$\text{tehát: } \max |F| < |A| + \max \frac{\bar{d}}{\bar{a}} < 2,5 + 5 < 10$$

$$\text{s így } \frac{|F|}{10} < \frac{\left| \frac{1}{10} \sqrt{10} \bar{a} \bar{x} + \frac{\bar{c}}{10} \right|}{\left| \frac{10 \bar{b}}{\bar{d}} + 12,1 \bar{y} \right|} + \left| \frac{1}{10} \frac{\bar{d}}{\bar{a}} \right| < 1$$

$$\left| \frac{\sqrt{10}}{10} \sqrt{\bar{a} \bar{x}} + \frac{\bar{c}}{10} \right| < 1$$

Még az első tört nevezője túlcsordulásának elkerülésére vegyük a százezer részét, természetesen akkor a számlálót is el kell osztani 100-al.

$$\text{Így } \frac{F}{10} = \frac{\frac{\sqrt{10}}{10 \cdot 100} \sqrt{\bar{a} \bar{x}} + \frac{\bar{c}}{10 \cdot 100}}{\frac{10 \bar{b}}{\bar{d}} + 12,1 \bar{y}} - \frac{1}{10} \frac{\bar{d}}{\bar{a}} =$$

$$= \frac{\frac{\sqrt{10}}{1000} \sqrt{\bar{a} \bar{x}} + \frac{\bar{c}}{1000}}{\frac{10 \bar{b}}{\bar{d}} + 12,1 \bar{y}} - \frac{1}{10} \frac{\bar{d}}{\bar{a}}$$

s ebben a képletben nem lép fel túlcsordulás. Látjuk, hogy az eredmény tizedrésze kerül kinyomtatásra.

Írjuk fel a feladat programját:

Legyen a rekeszelosztás a következő:

$$(0100) = \frac{\sqrt{10}}{1000} \quad (0106) = \bar{c}$$

$$(0101) = \frac{1}{1000} \quad (0107) = \bar{d}$$

$$\begin{aligned}
 (0102) &= \frac{1}{10} & (0110) &= \bar{x} \\
 (0103) &= 0,121 & (0111) &= \bar{y} \\
 (0104) &= \bar{y} & (0120) &= y_0 \quad (\text{A gyökvonás kezdő-} \\
 (0105) &= \bar{b} & & \text{értéke})
 \end{aligned}$$

Munkarekeszek: 0112, 0113, 0114, 0115, 0116, 0117.

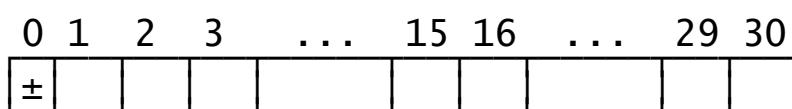
| | | | | | |
|------|-----|----|-------------------|------|---|
| 0150 | x, | 13 | 0102 | 0107 | $\frac{1}{10} \bar{d}$ |
| 0151 | ↓: | 22 | 0104 | 0112 | |
| 0152 | x, | 13 | 0102 | 0105 | |
| 0153 | ↓: | 22 | 0107 | 0113 | |
| 0154 | x, | 13 | 0103 | 0111 | |
| 0155 | ↓+ | 20 | 0113 | 0113 | $\frac{1}{10} - \frac{\bar{b}}{a} + 0,121 \bar{y}$ |
| 0156 | x, | 13 | 0101 | 0106 | |
| 0157 | U1 | 24 | 0160 | 0114 | $\frac{\bar{c}}{10^3}$ |
| 0160 | x, | 13 | 0104 | 0110 | |
| 0161 | U1 | 24 | 0162 | 0115 | |
| 0162 | Á | 05 | <y ₀ > | 0116 | Gyökvonás az $x_n = \frac{1}{2} \left(\frac{\bar{ax}}{y_{n-1}} - y_{n-1} \right) + y_{n-1}$ képlet alapján (A gyök $\approx y_n$) |
| 0163 | x, | 13 | 0116 | 0113 | |
| 0164 | ↓-, | 31 | 0116 | | |
| 0165 | ↓x | 23 | < $\frac{1}{2}$ > | 0117 | |
| 0166 | ↓+ | 20 | 0116 | 0116 | |
| 0167 | - , | 51 | 0117 | | |
| 0170 | FU | 34 | 0163 | 0171 | |
| 0171 | x, | 13 | 0116 | 0100 | |
| 0172 | ↓+, | 30 | 0114 | | |
| 0173 | ↓:, | 32 | 0113 | | |
| 0174 | ↓- | 21 | 0112 | 0112 | $\frac{F}{10} = \bar{F}$ |

A példából is látható, hogy a fixpontból adódó nehézségnek skálafaktor módszerrel történő megkerülése a programozási munkának egy jelentős része, - s ez vonatkozik minden transzformációs módszerre. Figyelmes és minden lehetséges esetre kiterjedő meggondolásokra van szükség:

voltaképpen lépésről–lépésre végig kell követni a számítás folyamatát. Külön figyelemmel kell azt is kísérni, hogy a jegyveszteség minimális legyen.

6.3. A bináris pont közére való helyezésének módszere.

Mint mondtuk, csupán megegyezés kérdése, hogy a bináris pontot hova képzeljük el. Elhelyezhetjük a bináris pontot a szó közepén is, a 16. bit után (az első bit az előjel helye!)



Ekkor tehát egy a gépben ábrázolt $x = \pm a_{14} a_{13} \dots a_0 \dots a_{-15}$ számhoz az

$$x = \pm a_{14} 2^{14} + \dots + a_0 2^0 + a_{-1} 2^{-1} + \dots + a_{-15} 2^{-15}$$

értéket rendeljük hozzá, ($a_i = 0$ vagy 1)

A gép értelmezési tartománya tehát ebben az esetben a $-2^{15} < x < 2^{15}$ intervallum, a számok ábrázolásának pontossága 2^{-15} . A skálafaktor fogalmát felhasználva, voltaképpen egyszerűen arról van szó, hogy az adott feladat minden egyes adatához a 2^{-15} közös skálafaktort rendeljük hozzá; azaz

$$\bar{x} = 2^{-15} x$$

Ezt a módszert olyankor alkalmazzuk, amikor a szereplő adatok felső korlátjaira nem tudunk pontos becslést adni, de valamilyen - az adatok természetéből adódó - okunk van feltenni, hogy 2^{15} -nél abszolút értékben kisebbek.

Ha a szereplő adatok nagyságrendje lényegesen különbözik, ez a módszer nem előnyös, mert igen nagy jegyveszteséggel jár.

A módszer alkalmazásánál állandóan ügyelni kell arra, hogy a bináris pont középen maradjon (pl. szorzás esetében!).

Lássunk egy példát.

Készítsük el az $y = \frac{a + bx}{c + dx} + \frac{x^2}{2}$ függvény programját azzal a feltételezéssel, hogy

$$2 < |c + dx| < 2^7; |a + bx| < 2^{15}; |bx| < 2^{15}; |dx| < 2^{15}; |x^2| < 2^{15};$$

Helyezzük el a bináris pontot középre, s legyen a rekeszelosztás (a címeket számok helyett betűkkel jelölve) a következő:

$$\begin{aligned} (p) &= x \cdot 2^{-15} & (p + 3) &= c \cdot 2^{-15} \\ (p + 1) &= a \cdot 2^{-15} & (p + 4) &= d \cdot 2^{-15} \\ & & (p + 2) &= b \cdot 2^{-15} \end{aligned}$$

Munkarekeszek: m, m+1

Program:

| | | | | |
|------|-----|----|---------------------|-----------------------------------|
| t | x, | 13 | p+2 | p |
| t+1 | ↓:, | 32 | <2 ⁻¹⁵ > | |
| t+2 | ↓+ | 20 | p+1 | m ← (a+bx)2 ⁻¹⁵ |
| t+3 | x, | 13 | p+4 | p |
| t+4 | ↓:, | 32 | <2 ⁻¹⁵ > | |
| t+5 | ↓+ | 20 | p+3 | m+1 ← (c+dx)2 ⁻¹⁵ |
| t+6 | : | 02 | <2 ⁻⁸ > | m+1 |
| t+7 | x, | 13 | <2 ⁻⁷ > | m |
| t+10 | ↓: | 22 | m+1 | m+1 ← $\frac{a+bx}{c+dx} 2^{-15}$ |
| t+11 | x, | 13 | p | p |
| t+12 | ↓:, | 32 | <2 ⁻¹⁴ > | |
| t+13 | + | 00 | m+1 | m+1 ← y2 ⁻¹⁵ |

A t , $t+3$ és a $t+11$ utasítások végrehajtása során a bináris-pont a szó végére került, a $t+2$, $t+4$, $t+12$ utasításokkal tehát "visszatoljuk" középre. A $t+6$, $t+7$ utasítások segítségével elértük, hogy az $\frac{a+bx}{c+dx}$ hányadosban is középre kerül a bináris pont. Az eredmény is $y \cdot 2^{-15}$ alakban adódik.

Vannak olyan feladatok is, amelyeknél nem a 2^{-15} , hanem egy más, a feladat adataiból adódó közös lépésfaktort választunk, azaz a bináris pontot nem középen, hanem a szó egy adott más helyértéke után helyezzük el.

Ha pl. két n -edrendű négyzetes mátrixot: $A = [a_{ij}]$
 $B = [b_{ij}]$ ($i, j = 1, 2, \dots, n$) szorzunk össze, amelyeknél az $|a_{ij}| < \frac{1}{\sqrt{n}}$ és $|b_{ij}| < \frac{1}{\sqrt{n}}$ feltételek teljesülnek, akkor a $C = AB$ eredmény-mátrix minden c_{ij} eleme, amelyet a mátrix-szorzás szabálya alapján a $c_{ij} = \sum_{v=1}^n a_{iv} b_{vj}$ képlet szerint nyerünk - egynél abszolútértékben kisebb; ugyanis

$$|c_{ij}| = \left| \sum_{v=1}^n a_{iv} b_{vj} \right| \leq \sum_{v=1}^n |a_{iv} b_{vj}| \sum_{v=1}^n \frac{1}{\sqrt{n}} \frac{1}{\sqrt{n}} =$$
$$= \sum_{v=1}^n \frac{1}{n} = n \frac{1}{n} = 1$$

Ha tehát $\max(a_{ij}) = M$ és $\max(b_{ij}) = N$ akkor a $\max(M, N) \sqrt{n} = Q$ közös lépésfaktor választásával (azaz minden elem helyett $\frac{a_{ij}}{Q}$, ill. $\frac{b_{ij}}{Q}$ -t véve; vagy ha $2^{q-1} \leq Q < 2^q$, akkor a bináris pontnak a q -edik bit után való elhelyezésével) elkerülhető a túlcscordulás.

Azokat az n egész számokat, amelyeket számlálásra, címmódosításra használunk, vagy amelyek valamilyen egészértékű függvény (pl. számelméleti függvények, stb.) adatai, célszerű $n \cdot 2^{-30}$ alakban ábrázolni; ez azt jelenti, hogy a bináris pontot a szó végére képzeljük el.

64. Egyéb módszerek.

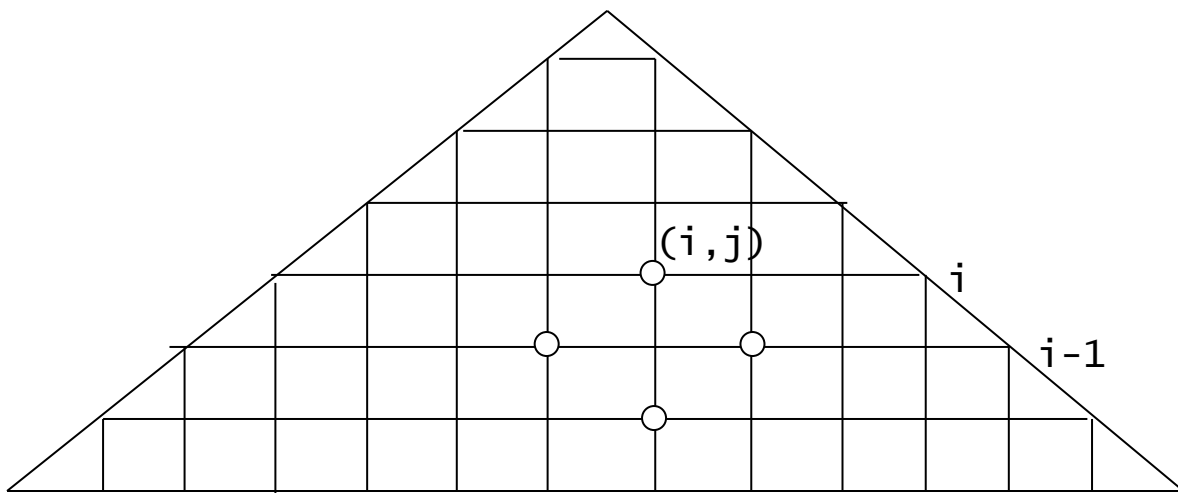
6.4.1. Reciprok érték felhasználása.

Egynél nagyobb számok (különösen számkonstansok) esetén néha célszerű reciprokjukat venni. Ha például tudjuk, hogy $|x|=|7a|<1$, akkor a számítást $x = \frac{a}{\frac{1}{7}}$ formában végezzük el.

Ugyanúgy járunk el, ha egynél csak nagyobb értékeket felvevő függvényekkel számolunk. Pl. e^x helyett ($x>0$) $\frac{1}{e^x} = e^{-x}$ -et számítjuk ki.

6.4.2. Automatikus normálás módszere.

Egyes feladatoknál az adatok normálása a számítás folyamán magával a géppel végezhető el. Pl. Legyen adva az 1. ábrán látható rácstartomány; s legyenek az első két sorában lévő rácspontértékek ismertek. Számítsuk és nyomtassuk ki egymásután a többi rácspontban is az értékeket, ha az i -edik sor j -edik elmét: a_{ij} -t az



$$a_{i,j} = a_{i-1,j-1} + a_{i-1,j+1} - a_{i-2,j} \text{ képlet adja.} \quad (\times)$$

Nyilvánvaló, hogy ha $|a_{i-1,j-1}| < \frac{1}{3}$; $|a_{i-1,j+1}| < \frac{1}{3}$;

$|a_{i-2,j}| < \frac{1}{3}$, akkor $|a_{i,j}| < |a_{i-1,j-1}| + |a_{i-1,j+1}| +$

$+ |a_{i-2,j}| < 1$;

tehát a következőképpen járhatunk el. Mielőtt egy új sor elemeit kezdenők számolni, előbb megvizsgáljuk, hogy az előtte lévő két sor minden eleme kisebb-e $\frac{1}{3}$ -nál. Ha kisebb, akkor a (*) képlet alapján elvégezzük a számítást. Ha nem, akkor előbb a szóbanforgó két sor minden elemét 3-mal osztjuk, de egyúttal megjegyezzük (pl. 2^{-30} kinyomtatásával), hogy az ezután következő a_{ij} értékek helyett azok egyharmadát számoltuk ki. Ha valamelyik következő sor kiszámításához újra normálni kell, akkor újabb 2^{-30} kinyomtatásával jelezzük, hogy az ez után következő értékek kilencszeresét kell venni, hogy a tényleges értéket megkapjuk, ... stb.

Másik példa:

Készítsük el az e^{-x} függvény ($x > 0$) kiszámításának programját az

$$e^{-x} = \frac{1}{(a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5)^4} \quad (**)$$

közelítőképlet alapján, ahol

$$a_0 = 1$$

$$a_1 = 0,25001095$$

$$a_2 = 0,03119805$$

$$a_3 = 0,00267325$$

$$a_4 = 0,00012799$$

$$a_5 = 0,00001487$$

(A (**)) közelítőképlet hibája = 0,000002.)

A számlálót és a nevezőt is 2-vel osztva:

$$e^{-x} = \left(\frac{\frac{1}{2}}{\frac{a_0}{2} + \frac{a_1}{2}x + \dots + \frac{a_5}{2}x^5} \right)^4 \quad (\times\times\times)$$

Ha $x > 0$, akkor $e^{-x} < 1$; tehát csak arra kell ügyelni, hogy a nevező ne csorduljon túl. Látható, hogy ha $x < 1$, akkor

$$\left(\frac{1}{2} + \frac{a_1}{2}x + \dots + \frac{a_5}{2}x^5\right) < 1$$

Tegyük fel, hogy

$$2^{n-1} \leq x < 2^n, \text{ s számítsuk ki } e^{-x} \text{ helyett } e^{-\bar{x}} = e^{-\frac{x}{2^n}}.$$

$$e^{-\frac{x}{2^n}} = e^{-\bar{x}} \left(\frac{\frac{1}{2}}{\frac{a_0}{2} + \frac{a_1}{2}\bar{x} + \frac{a_2}{2}\bar{x}^2 + \dots + \frac{a_5}{2}\bar{x}^5} \right)^4$$

s így a nevező sem csordul túl.

$$\text{Nyilván: } e^{-x} = (e^{-\bar{x}})^{2^n}$$

Készítsünk el egy olyan programot, amely a $0 < x < 16$ -ban levő x értékekre működik, mégpedig úgy, hogy mindig $\frac{x}{16}$ -ból indul ki, de ha pl. $x < 8$, akkor $2 \left(\frac{x}{16}\right) = \frac{x}{8}$, ha $x < 4$, akkor $4 \left(\frac{x}{16}\right) = \frac{x}{4}$ s.í.t. értékkel számol.

Rendezzük át a $(\times\times\times)$ képlet nevezőjét a Horner séma szerint. Így:

$$e^{-\bar{x}} = \left(\frac{\frac{1}{2}}{\left(\left(\left(\left(\left(\frac{a_5}{2}\bar{x} + \frac{a_4}{2}\right)\bar{x} + \frac{a_3}{2}\right)\bar{x} + \frac{a_2}{2}\right)\bar{x} + \frac{a_1}{2}\right)\bar{x} + \frac{a_0}{2}\right)} \right)^4$$

Legyen a rekeszelosztás a következő: (a címeket betűvel jelölve.)

Munkarekeszek: $m, m+1, m+2$.

Konstansok:

$$\begin{aligned}
 (k) &= \downarrow + 20 & k + 3 & m + 2 \\
 (k+1) &= \downarrow + 20 & k + 10 & m + 2 \\
 (k+2) &= \frac{a^5}{2} \\
 (k+3) &= \frac{a^4}{2} \\
 (k+4) &= \frac{a^3}{2} \\
 (k+5) &= \frac{a^2}{2} \\
 (k+6) &= \frac{a^1}{2} \\
 (k+7) &= \frac{a^0}{2}
 \end{aligned}$$

Legyen $(m) = \frac{x}{16} = \bar{x}$.

A program:

| | | | | |
|------|---------------------|----|-----------------------------------|------|
| t | Á | 05 | k | t+12 |
| t+1 | Á | 05 | $\langle 4 \cdot 2^{-30} \rangle$ | m+1 |
| t+2 | -, | 11 | $\langle \frac{1}{2} \rangle$ | m |
| t+3 | FU | 34 | t+4 | t+7 |
| t+4 | : | 02 | $\langle \frac{1}{2} \rangle$ | m |
| t+5 | - | 01 | $\langle 2^{-30} \rangle$ | m+1 |
| t+6 | FU | 34 | t+10 | t+2 |
| t+7 | + | 00 | $\langle 2^{-30} \rangle$ | m+1 |
| t+10 | Á | 05 | k+2 | m+2 |
| t+11 | x | 03 | m | m+2 |
| t+12 | (- | - | - | -) |
| t+13 | + | 00 | $\langle 2^{-18} \rangle$ | t+12 |
| t+14 | $\downarrow - ,$ | 71 | k+1 | |
| t+15 | FU | 34 | t+11 | t+16 |

| | | | | |
|------|----|----|---------------------------|-------------------------------|
| t+16 | : | 12 | m+2 | $\langle \frac{1}{2} \rangle$ |
| t+17 | U1 | 24 | t+21 | m+2 |
| t+20 | x | 03 | m+2 | m+2 |
| t+21 | - | 01 | $\langle 2^{-30} \rangle$ | m+1 |
| t+22 | FU | 34 | t+23 | t+20 |
| t+23 | . | . | . | . |

A felsorolt módszereken kívül egyes feladatoknál bizonyos speciális módszerek is használhatók. Periodikus függvényeknél pl. eltolási transzformáció alkalmazható. Így a $\sin x$, ill. $\cos x$ függvény esetén a $\sin x = \sin(x - 2k\pi)$, $\cos x = \cos(x - 2k\pi)$ egyenlőség alapján az $u = x - 2k\pi$ transzformációval elérhető, hogy

$$-\pi \leq u \leq \pi \text{ legyen.}$$

Mínt hogy $\sin(-u) = -\sin u$, és $\cos(-u) = \cos u$, azért elég u pozitív értékeire szorítkozni.

$$\text{Ha } |u| \leq \frac{\pi}{2}, \text{ akkor } \frac{u}{2} \leq \frac{\pi}{4} < 1$$

Ha $|u| > \frac{\pi}{2}$, akkor felhasználva $\sin(\pi - |u|) = \sin |u|$; $\cos(\pi - |u|) = -\cos |u|$ egyenlőségeket a $v = \pi - |u|$ transzformációval $0 \leq \frac{v}{2} \leq \frac{\pi}{4} < 1$

$$\text{Így a } \sin v = 2 \sin \frac{v}{2} \cos \frac{v}{2}; \cos |x| = \cos^2 \frac{v}{2} - \sin^2 \frac{v}{2}$$

képletek alapján végül:

$$\sin |x| = 2 \sin \frac{v}{2} \cdot \cos \frac{v}{2}; \cos |x| = \cos^2 \frac{v}{2} - \sin^2 \frac{v}{2}$$

Ezt a transzformációt természetesen a géppel végeztetjük el; feltesszük, hogy $x < 2^{15}$, és a bináris pontot közepre helyezzük.

A transzformációs módszerek segítségével nem lehet minden problémát megoldani. Nem mindig sikerül pl. a feladatban szereplő adatok számára még csak közelítő korlátokat sem adni: Egy differenciálegyenlet vagy lineáris egyenletrendszer megoldásának maximumáról például általában semmit sem tudunk mondani.

Ha a szereplő adatok nagyságrendje lényegesen különbözik, akkor előfordulhat, hogy a skálafaktorok bevezetésével a kis számok gépi 0-vá válnak, vagy legalábbis a jegyvesztés igen nagy lesz.

Ha a transzformációs módszerek a fenti okok miatt nem alkalmazhatók, akkor a már említett másik módszerhez, a lebegőpontos programozás módszeréhez folyamodunk. Erről majd a későbbiekben lesz szó.

7. Fejezet.

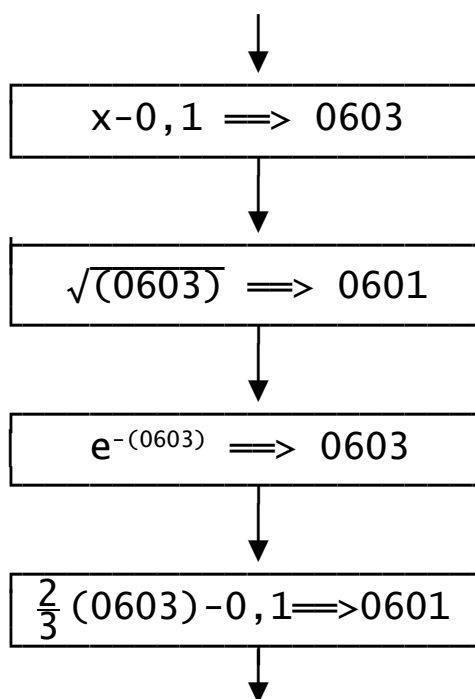
SZUBRUTINOK ALKALMAZÁSA

7.1. Nyílt és zárt szubrutinok.

Szubrutinomnak nevezzük általában az olyan részprogramokat, amelyek egy-egy meghatározott, önálló feladatot hajtanak végre. A szubrutinokat, mint "építőköveket" sok esetben felhasználhatjuk hosszabb programok összeállításánál. Lássunk ezzel kapcsolatban egy konkrét példát.

Készítsünk programot az $y = \frac{2}{3} e^{-\sqrt{x-0,1}} - 0,1$ függvény értékeinek kiszámításához. A program elkészítésénél tegyük fel, hogy a $0,1 \leq x < 1$ intervallumba eső x érték a 0600 című memóriarekeszben van elhelyezve, a hozzá tartozó y értéket pedig a 0601 című memóriarekeszbe kell bevinni. A programot magát a 0100 című rekesztől kezdve, a program konstansait a 0-500 című rekesztől kezdve helyezzük el, munkarekeszeknek a 0602-0603 rekeszeket használjuk.

A számítás menetrendje igen egyszerű:



Az első blokknak megfelelő utasításcsoportot azonnal felírhatjuk:

| | | | |
|------|-------|------|------|
| 0100 | -, 11 | 0500 | 0600 |
| 0101 | u1 24 | 0102 | 0603 |

ahol (0500) = 0,1 (konstans).

A második blokk programozásánál emlékeztetünk arra, hogy a négyzetgyökvonás programját már elkészítettük. Ez a program a (véletlenül választott) 0011 című rekeszben tárolt szám négyzetgyökét képezi az $y_0 = (0015)$ kezdőértékből kiindulva a 0012 című rekeszben. A program a következő konstansokból, illetve utasításokból áll: (0014) = 0; (0016) = $\frac{1}{2}$; (0017) = $= 2^{-30}$. Maga a program:

| | | | |
|------|--------|------|------|
| 0000 | Á 05 | 0014 | 0013 |
| 0001 | Á 05 | 0015 | 0012 |
| 0002 | + 00 | 0013 | 0012 |
| 0003 | :, 12 | 0012 | 0011 |
| 0004 | ↓-, 31 | 0012 | - |
| 0005 | ↓x 23 | 0016 | 0013 |

| | | | |
|------|--------|------|------|
| 0006 | ↓+, 30 | 0017 | - |
| 0007 | FU 34 | 0002 | 0010 |
| 0010 | - - | - | - |

A felírt programot könnyen "beépíthetjük" a tekintett példa programjába, ha figyelembe vesszük a következőket:

- a.) Az y_0 kezdőértéket numerikusan meg kell adni; jelen esetben célszerű az $y_0 = 1 - 2^{-30}$ ("gépi egyes") választás.
- b.) A négyzetgyökvonás programját most nem a 0000-0007, hanem a 0102-0111 rekeszekben kell elhelyezni.

A négyzetgyökvonáshoz szükséges konstansok és paraméterek tárolására, illetve munkarekeszekként továbbra is felhasználhatnók a 0011-0017 rekeszeket (feltéve, hogy ezek a rekeszek nincsenek valami más célra lefoglalva), azonban az egyöntetűség kedvéért, és szükségtelen átviteli utasítások megtakarítása végett mégis célszerűbb a négyzetgyökvonó program konstansait a 0501-0502 és 0515 rekeszekben, az x számot a 0605 rekeszekben, az $y_0 = 1 - 2^{-30}$ értéket a 0505 rekeszben, a négyzetgyökvonás eredményét a 0601 rekeszben tárolni, munkarekesznek pedig a 0602 rekeszt használni. Ennek megfelelően természetesen a négyzetgyökvonás programját teljesen "át kell címezni": minden mennyiség régi címe helyébe az illető mennyiség új címét kell beírni. Az átcímezésnél jól felhasználhatjuk az alábbi "átcímező táblázatot":

| Megnevezés | Régi cím | Új cím | |
|------------------------------|---------------|--------|------|
| x | 0011 | 0603 | |
| $y = \sqrt{x}$ | 0012 | 0601 | |
| y_0 | 0015 | 0503 | |
| a négyzetgyökvonás programja | 0000 - | 0102 - | |
| konstansok { | $\frac{1}{2}$ | 0016 | 0501 |
| | + 0 | 0014 | 0502 |
| | 2^{-30} | 0017 | 0515 |
| munkarekesz | 0013 | 0602 | |

A második blokknak végeredményben a következő utasításcsoport felel meg:

| | | | | |
|------|-----|----|------|------|
| 0102 | Á | 05 | 0502 | 0602 |
| 0103 | Á | 05 | 0503 | 0601 |
| 0104 | + | 00 | 0602 | 0601 |
| 0105 | ∴, | 12 | 0601 | 0603 |
| 0106 | ↓-, | 31 | 0601 | - |
| 0107 | ↓x | 23 | 0501 | 0602 |
| 0110 | ↓+, | 30 | 0515 | - |
| 0111 | FU | 34 | 0104 | 0112 |

A harmadik blokk programozásánál hasonló módon felhasználhatjuk az $y = e^{-x}$ függvény értékeinek kiszámítására készített programot. Ebben a programban nem tüntettük fel a szereplő mennyiségek valódi (abszolút) címeit, hanem relatív címeket (betű-címeket, illetve szögletes zárójellel kijelölt címeket) alkalmaztunk. A program az m című rekeszben tárolt $\bar{x} = \frac{x}{16}$ értékből az e^{-x} értéket az $m+2$ című rekeszben állítja elő, és a következő konstansokból, illetve utasításokból áll:

$$\begin{aligned}
 (k) &= \downarrow+20 \quad k+3 \quad m+2; & (k+1) &= +20 \quad k+10 \quad m+2; \\
 (k+2) &= +0,00000744; & (k+3) &= +0,00006399; \\
 (k+4) &= +0,00133662; & (k+5) &= +0,01559902; \\
 (k+6) &= +0,12500546; & (k+7) &= \frac{1}{2};
 \end{aligned}$$

$m, m+1, m+2$ munkarekeszek. A program maga:

| | | | | |
|-------|----|----|-----------------------------------|--------|
| t | Á | 05 | k | $t+12$ |
| $t+1$ | Á | 05 | $\langle 4 \cdot 2^{-30} \rangle$ | $m+1$ |
| $t+2$ | -, | 11 | $\langle \frac{1}{2} \rangle$ | m |
| $t+3$ | FU | 34 | $t+4$ | $T+7$ |
| $t+4$ | ∴ | 02 | $\langle \frac{1}{2} \rangle$ | m |
| $t+5$ | - | 01 | $\langle 2^{-30} \rangle$ | $m+1$ |
| $t+6$ | FU | 34 | $t+10$ | $t+2$ |

| | | | | |
|------|-------|----|---------------------------|-------------------------------|
| t+7 | + | 00 | $\langle 2^{-30} \rangle$ | m+1 |
| t+10 | Á | 05 | k+2 | m+2 |
| t+11 | x | 03 | t+11 | m+2 |
| t+12 | (↓+ | 20 | k+3 | m+2) |
| t+13 | + | 00 | $\langle 2^{-18} \rangle$ | t+12 |
| t+14 | ↓ - , | 71 | k+1 | - |
| t+15 | FU | 34 | t+11 | t+16 |
| t+16 | ∴, | 12 | m+2 | $\langle \frac{1}{2} \rangle$ |
| t+17 | U1 | 24 | t+21 | m+2 |
| t+20 | x | 03 | m+2 | m+2 |
| t+21 | - | 01 | $\langle 2^{-30} \rangle$ | m+1 |
| t+22 | FU | 34 | t+23 | t+20 |
| t+23 | . | . | . | . |

Ezt a programot is könnyen beépíthetjük, ha a relatív címek helyébe megfelelő abszolút címeket írunk be. A jelen esetben a t, k és m betű-címeknek célszerű a $t = 0113$, $k = 0504$, $m = 0601$ értékeket adni. Legyen továbbá $(\frac{1}{2}) = 0513$, $(2^{-18}) = 0514$, $(2^{-30}) = 0515$, $(4 \cdot 2^{-30}) = 0516$.

A helyettesítéseket elvégezve azonnal felírhatjuk a harmadik blokknak megfelelő utasításcsoportot:

| | | | | |
|------|-----|----|------|-------|
| 0112 | x | 03 | 0520 | 0601 |
| 0113 | Á | 05 | 0504 | 0125 |
| 0114 | Á | 05 | 0516 | 0602 |
| 0115 | -, | 11 | 0513 | 0601 |
| 0116 | FU | 34 | 0117 | 0122 |
| 0117 | ∴ | 02 | 0513 | 0601 |
| 0120 | - | 01 | 0515 | 0602 |
| 0121 | FU | 34 | 0123 | 0115 |
| 0122 | + | 00 | 0515 | 0602 |
| 0123 | Á | 05 | 0506 | 0603 |
| 0124 | x | 03 | 0601 | 0603 |
| 0125 | (↓+ | 20 | 0507 | 0603) |
| 0126 | + | 00 | 0514 | 0125 |

| | | | | |
|------|-------|----|------|------|
| 0127 | ↓ - , | 71 | 0505 | - |
| 0130 | FU | 34 | 0124 | 0131 |
| 0131 | : | 12 | 0603 | 0513 |
| 0132 | U1 | 24 | 0134 | 0603 |
| 0133 | x | 03 | 0603 | 0603 |
| 0134 | - | 01 | 0515 | 0604 |
| 0135 | FU | 34 | 0136 | 0133 |

A negyedik blokknak megfelelő utasításcsoport:

| | | | | |
|------|----|----|------|------|
| 0136 | x, | 13 | 0517 | 0603 |
| 0137 | ↓- | 21 | 0500 | 0601 |
| 0140 | . | . | . | . |

ahol $(0517) = \frac{2}{3}$ és $(0520) = 2^{-4}$.

Az adott programozási feladat ismertetett megoldásában két előre elkészített szubrutint úgy használnánk fel, hogy a kész szubrutinokat mintegy "beleépítettük" az összeállítandó programba, vagyis a szubrutinok utasításait, konstansait, stb. megfelelő módon átcímezve az összeállítandó program többi részének - az u.n. főprogramnak - utasításai, konstansai közé helyeztük. A gép mindegyik szubrutin végrehajtását a memóriában közvetlenül a szubrutin utasításai előtt álló utasítás végrehajtása után kezdi meg, és a szubrutin végrehajtása után a memóriában közvetlenül a szubrutin utasításai után következő utasítás végrehajtására tér rá. Ilyen esetben azt mondjuk, hogy az előre elkészített szubrutinokat - példánkban az $y = \sqrt{x}$ és az $y = e^{-x}$ függvény szubrutinját - mint nyílt szubrutinokat használtuk fel az összeállítandó programban. A nyílt szubrutinok tehát a program szerves részévé válnak.

Az adott programozási feladatot azonban másképpen is megoldhatjuk. A két szubrutint ennél a megoldásnál nem építjük bele a programba, hanem a főprogramtól és egymástól teljesen függetlenül, a memória két tetszőleges helyére viszük be.

Feltesszük, hogy a 0000-0100 rekeszek nincsenek másra felhasználva. A négyzetgyökvonó szubrutint ebben az esetben eredeti helyén hagyhatjuk, az $y = e^{-x}$ függvény szubrutinját pedig elhelyezhetjük például a 0020 című rekesztől kezdve; a szubrutin konstansainak tárolására és munkarekeszekként felhasználhatjuk például a 0050-0064, illetve a 0070-0072 rekeszeket (vagyis legyen $t = 0020$, $k = 0050$, $\langle \frac{1}{2} \rangle = 0061$, $\langle 2^{-18} \rangle = 0062$, $\langle 2^{-30} \rangle = 0063$, $\langle 4 \cdot 2^{-30} \rangle = 0064$, $m = 0070$).

A főprogram a független változó kiszámított értékét a 0011 (illetve a 0070) rekeszbe viszi be, és egy-egy feltétlen ugró utasítás segítségével átadja a vezérlést az $y = \sqrt{x}$ ($y = e^{-x}$) függvény értékeit kiszámító szubrutinnak ("behívja" a megfelelő szubrutint). Természetesen mindkét esetben gondoskodni kell arról, hogy a szubrutin végrehajtása után a gép ismét visszatérjen a főprogramra. Ezt a legegyszerűbben úgy érhetjük el, hogy mindegyik szubrutin behívása előtt egy-egy feltétlen ugró utasítást vitetünk be a főprogrammal a 0010 (illetve a 0043) című rekeszbe. A főprogram a következő konstansokból illetve utasításokból áll:

| | | | | | |
|------|----|----|------|------|-----------------------|
| 0100 | Á | 05 | 0500 | 0010 | (0500)= 74 --0103 |
| 0101 | -, | 11 | 0501 | 0600 | (0501)= 0,1 |
| 0102 | U1 | 24 | 0000 | 0011 | (0502)= 74 --0106 |
| 0103 | Á | 05 | 0502 | 0043 | (0503)= 2^{-4} |
| 0104 | x, | 13 | 0012 | 0503 | (0504)= $\frac{2}{3}$ |
| 0105 | U1 | 24 | 0020 | 0070 | |
| 0106 | x, | 13 | 0504 | 0072 | |
| 0107 | ↓- | 21 | 0501 | 0601 | |

Ez a megoldás tipikus példa volt arra, hogy egy program összeállításánál előre elkészített szubrutinokat úgy is fel lehet használni, hogy a szubrutinokat egymástól és a főprogramtól teljesen függetlenül helyezzük el a memóriában. Ebben az esetben a főprogram egy feltétlen ugró utasítás segítségével adja át a vezérlést a megfelelő szubrutin első utasításának, és a szubrutin végrehajtása után egy fel-

tétlen ugró utasítás adja vissza a vezérlést a főprogramnak. Ilyenkor azt mondjuk, hogy az előre elkészített szubrutinokat, mint zárt szubrutinokat használtuk fel az összeállítandó programban.

A kidolgozott példából látható, hogy zárt szubrutinok alkalmazása nyílt szubrutinok helyett bizonyos (minimális) veszteséget jelent az elfoglalt memóriarekeszek számában, és a program végrehajtásához szükséges gépi időben egyaránt. Ezzel szemben észrevehető az is, hogy a zárt szubrutinok alkalmazása jelentősen megkönnyíti a programozó munkáját, hiszen zárt szubrutinok alkalmazása esetén a programozónak a teljes program helyett lényegében csak a sokkal rövidebb főprogrammal kell dolgoznia.

A zárt szubrutinoknak a nyílt szubrutinokkal szemben megvan az az előnyük is, hogy ugyanazt a zárt szubrutint többször, a főprogram különböző helyein be lehet hívni. Zárt szubrutin alkalmazásával könnyen megoldhatjuk például az alábbi programozási feladatot:

készítsünk programot az

$$y = \sqrt{0,9 - x} - \sqrt{x - 0,1}$$

függvény értékeinek kiszámításához. A $0,1 \leq x \leq 0,9$ intervallumba eső x érték a 0600 című rekeszben van elhelyezve, az y értéket a 0601 című rekeszbe kell bevinni. A négyzetgyökvonó szubrutin utasításai a 0000-0007 című rekeszekben vannak tárolva.

A főprogram:

(0500) = + 0,9; (0501) = + 0,1; (0503) = + 74 - 0103;
(0504) = + 74 - 0107

| | | | | |
|------|----|----|------|------|
| 0100 | Á | 05 | 0503 | 0010 |
| 0101 | -, | 11 | 0600 | 0500 |
| 0102 | U1 | 24 | 0000 | 0011 |
| 0103 | Á | 05 | 0012 | 0601 |
| 0104 | Á | 05 | 0504 | 0010 |
| 0105 | -, | 11 | 0501 | 0600 |
| 0106 | U1 | 24 | 0000 | 0011 |
| 0107 | - | 01 | 0012 | 0601 |

Gyakorlásként készítsünk programot ugyanilyen feltételek mellett az $y = \sqrt[4]{0,9-x} - \sqrt[4]{x-0,1}$ függvény értékeinek kiszámításához; a negyedik gyökvonást kétszeres négyzetgyökvonással végeztessük el.

7.2. Zárt szubrutinok behívása; visszaugrás a főprogramra.

Az eddig ismertetett példákban a zárt szubrutinokat mindig úgy hívtuk be, hogy először egy átviteli utasítással egy feltétlen ugró utasítást vitettünk be a szubrutint követő rekeszbe (ennek eredeti tartalma irreleváns), majd egy feltétlen ugró utasítással átadattuk a vezérlést a szubrutin kezdőutasításának. Ha ezt a behívási sémát alkalmazzuk, akkor minden szubrutin behívása a főprogramban általában két-két utasításba és egy-egy konstansba "kerül".

Sok esetben előnyösebb a zárt szubrutinokat a következő két utasítás segítségével behívni:

| | | | | |
|-------|----|----|---|-------|
| m + 1 | +, | 10 | k | m + 1 |
| m + 2 | U1 | 24 | b | b' |

ahol m a szubrutin behívását közvetlenül megelőző utasítás rekesze, b a szubrutin kezdő utasításának rekesze, a k rekesz a + 64 0000 0002 konstansot tartalmazza, és b' a szubrutin után következő rekesz. A két utasítás közül az első önmagához hozzáadja a + 64 0000 0002 konstansot, és így a B-regiszterben egy + 74 k a + 3 alakú utasítást hoz létre, a második

ezt az utasítást a b' rekeszben helyezi el, és átadja a vezérlést a szubrutin kezdőutasításának. A + 64 0000 0002 konstansot célszerű "szabványos" konstansként mindig egy meghatározott memóriarekeszben, például a 0001 című rekeszben tárolni (szabványos konstansnak érdemes kijelölni a programozás gyakorlatában leggyakrabban előforduló konstansokat, például 2^{-30} -at, $\frac{1}{2}$ -et, $\frac{1}{4}$ -et, stb.) Minden zárt szubrutin behívása ezzel a módszerrel a főprogramban két-két utasításba és összesen egy konstansba kerül.

A továbbiakban újabb példákat dolgozunk ki a zárt szubrutinok alkalmazásával kapcsolatban:

1. Példa: Készítsünk programot az

$$y = \sqrt{a + bx} + a \sin bx + b \log (a - e^{-cx})$$

függvény értékeinek kiszámítására, ahol a, b és c konstansok, $\alpha \leq x \leq \beta$. Feltesszük, hogy az a, b, c konstansok és ($\alpha \leq x \leq \beta$ esetén) az $a + bx$, $a - e^{-cx}$, $\log (a - e^{-cx})$, $\sqrt{a + bx} + a \sin bx$ kifejezések és az y függvény valósak és abszolút értékben 1-nél kisebbek. Feltesszük továbbá, hogy rendelkezésünkre állnak a következő szubrutinok.

a.) $y = \sqrt{x}$ szubrutinja. Ez a szubrutin a 0021 rekeszben elhelyezett x szám ($0 \leq x < 1$) négyzetgyökét állítja elő a 0025 rekeszben, a szubrutin utasításai a 0010-0017 rekeszekben vannak elhelyezve.

b.) $y = \sin \frac{\pi}{2} x$ szubrutinja. Ez a szubrutin a 0150 rekeszben elhelyezett x értékhez ($-1 < x < 1$) tartozó y értékét a 0151 című rekeszben állítja elő, a szubrutin utasításai a 0100-0111 rekeszekben vannak elhelyezve.

c.) $y = e^{-x}$ szubrutinja. Ez a szubrutin a 0250 rekeszben elhelyezett $x = \frac{x}{16}$ értékből az e^{-x} értéket a 0251 rekeszben állítja elő, a szubrutin utasításai a 0200-0222 rekeszekben vannak elhelyezve.

d.) $y = \log x$ szubrutinja. Ez a szubrutin a 0550 rekeszben

elhelyezett x értékhez ($e^{-1} < x < 1$) tartozó y értéket a 0351 rekeszben állítja elő, a szubrutin utasításai a 0300-0332 rekeszekben vannak elhelyezve.

A főprogram utasításait az 1000 című rekesztől kezdve, konstansait az 1100 című rekesztől kezdve helyezzük el. Az x független változó értékét az 1120 rekeszbe visszük be, a megfelelő y értéket az 1121 rekeszben kapjuk meg. A főprogram a következő:

- (1100) = a
- (1101) = b
- (1102) = c
- (1103) = $\frac{2}{\pi}$
- (1104) = $\frac{1}{16}$
- (0001) = +64 0000 0002
(szabványos konstans!)

| | | | | |
|------|----|----|------|------|
| 1000 | x, | 13 | 1101 | 1120 |
| 1001 | ↓+ | 20 | 1100 | 0021 |
| 1002 | + | 10 | 0001 | 1002 |
| 1003 | U1 | 24 | 0010 | 0020 |
| 1004 | x, | 13 | 1101 | 1120 |
| 1005 | ↓x | 23 | 1103 | 0150 |
| 1006 | + | 10 | 0001 | 1006 |
| 1007 | U1 | 24 | 0100 | 0112 |
| 1010 | x, | 13 | 1100 | 0151 |
| 1011 | ↓+ | 20 | 0025 | 1121 |
| 1012 | x, | 13 | 1102 | 1120 |
| 1013 | ↓x | 23 | 1104 | 0250 |
| 1014 | + | 10 | 0001 | 1014 |
| 1015 | U1 | 24 | 0200 | 0223 |
| 1016 | -, | 11 | 0251 | 1100 |
| 1017 | U1 | 24 | 1020 | 0350 |
| 1020 | + | 10 | 0001 | 1020 |
| 1021 | U1 | 24 | 0300 | 0333 |
| 1022 | x, | 13 | 1101 | 0351 |
| 1023 | ↓+ | 20 | 1121 | 1121 |
| 1024 | . | . | . | . |

2. Példa: Az M-3 gép perforált szalagjára binárisan kódolt decimális alakban felvittünk 200 számot (lásd a bevitelről szóló részt). Készítsünk olyan programot, amely a 200 számot összegezi, az eredményt decimális alakban kinyomtatja, végül pedig megállítja a gépet.

A program elkészítésénél feltesszük, hogy rendelkezésünkre állnak a következő szubrutinok:

- a.) Input (bemeneti) szubrutin. Ez a szubrutin egy számot leolvas a perforált szalagról (közben a perforált szalagot megfelelően tovább mozditja), és a számot bináris alakra konvertálva a 0451 rekeszben helyezi el. Az input szubrutin utasításai a 0400-0411 rekeszekben vannak tárolva.
- b.) Output (kimeneti) szubrutin. Ez a szubrutin a 0550 rekeszben bináris alakban elhelyezett számot binárisan kódolt decimális alakba konvertálja, és decimális alakban kinyomtatja. Az output szubrutin utasításai a 0500-0511 rekeszekben vannak elhelyezve.

Feltesszük továbbá, hogy az összeadandók úgy vannak normálva, hogy az összegezés közben nem lép fel túlcserdülés.

A főprogram utasításait az 1000 című rekesztől, konstansait az 1100 című rekesztől kezdve helyezzük el. Az 1200 és 1201 című rekeszeket munkarekeszeknek használjuk. A főprogram a következő:

$$(1100) = 0$$

$$(1101) = 2^{-30}$$

$$(1102) = 200 \cdot 2^{-30}$$

$$(0001) = +64 \ 0000 \ 0002$$

(szabványos konstans!)

| | | | | |
|------|-----|-----|------|------|
| 1000 | Á | 05 | 1101 | 1200 |
| 1001 | Á | 05 | 1100 | 1201 |
| 1002 | +, | 10 | 0001 | 1002 |
| 1003 | U1 | 24 | 0400 | 0412 |
| 1004 | + | 00 | 0451 | 1200 |
| 1005 | + | 00 | 1101 | 1201 |
| 1006 | ↓-, | 31 | 1102 | - |
| 1007 | U1 | 24 | 1002 | 1010 |
| 1010 | Á | 05 | 1200 | 0550 |
| 1011 | + | 00 | 0001 | 1011 |
| 1012 | U1 | 24 | 0500 | 0512 |
| 1013 | | +77 | 0000 | 0000 |

Az utolsó utasítás hatására a gép megáll.

Az a program az összes eddigitől abban különbözik, hogy nem egy hosszabb program kiragadott része, hanem önállóan lefuttatható, teljes program; a szükséges adatokat a perforált szalagról (binárisan kódolt) decimális alakban olvassa le, az eredményt decimális alakban kinyomtatja, és végül megállítja a gépet.

3. Példa: A perforált szalagra binárisan kódolt decimális alakban felvitettünk 30 mérési adatot. Kiszámítandó a mérési adatok \bar{x} számtani közepe, és D "szórása"; az utóbbi a

$$D = \sqrt{\frac{\sum_{i=1}^{50} (\bar{x} - x_i)^2}{50}}$$

képlet alapján, ahol x_i -vel jelöltük az i -edik mérési adatot. Az \bar{x} és D értéket a gép (decimális alakban) nyomtassa ki. A főprogram a két eredmény kinyomtatása után állítsa meg a gépet.

A program elkészítésénél feltesszük, hogy rendelkezésünkre állnak a következő szubrutinok:

- a.) Az input és az output szubrutin. A két szubrutin a memóriában ugyanott van elhelyezve, mint a 2. példában.
- b.) A négyzetgyökvonó szubrutin. A szubrutin ugyanott van elhelyezve, mint az 1. példában.

Az \bar{x} illetve a D képzésénél a túlcsoordulás veszélyét úgy kerüljük el, hogy rögtön a $\sum_{i=1}^{50} \frac{x_i}{50}$, illetve a

$\sum_{i=1}^{50} \frac{(\bar{x}-x_i)^2}{50}$ összeget számítjuk ki.

A főprogram utasításait az 1000 című rekesztől, konstansait az 1100 című rekesztől kezdve helyezzük el. Az 1200 és 1201 című rekeszeket munkarekeszeknek használjuk. Az x_i mérési adatot a főprogram a $0700 + i - 1$ című rekeszbe viszi be. A főprogram a következő:

(1100) = 0
 (1101) = 2^{-30}
 (1102) = $\frac{1}{50}$
 (1103) = +05 0451 1200
 (1104) = 2^{-18}
 (1105) = +11 0762 1200
 (0001) = +65 0000 0002
 (szabványos konstans)

| | | | | | |
|------|-----|----|------|-------|---|
| 1000 | Á | 05 | 1100 | 1200 | } A mérési adatok bevitele és \bar{x} kiszámítása |
| 1001 | +, | 10 | 0001 | 1001 | |
| 1002 | U1 | 24 | 0400 | 0412 | |
| 1003 | (Á | 05 | 0451 | 0700) | |
| 1004 | ↓x, | 33 | 1102 | - | |
| 1005 | ↓+ | 20 | 1200 | 1200 | |
| 1006 | + | 00 | 1101 | 1003 | |
| 1007 | ↓-, | 31 | 1103 | - | |
| 1010 | FU | 34 | 1001 | 1011 | |

| | | | | | | |
|------|-----|----|------|-------|---|-----------------------------|
| 1011 | Á | 05 | 1100 | 0021 | } | D kiszámítása |
| 1012 | (-, | 24 | 0700 | 1200) | | |
| 1013 | U1 | 24 | 1014 | 1201 | | |
| 1014 | ↓x, | 33 | 1201 | - | | |
| 1015 | ↓x, | 33 | 1102 | - | | |
| 1016 | ↓+ | 20 | 0021 | 0021 | | |
| 1017 | + | 00 | 1104 | 1012 | | |
| 1020 | ↓-, | 31 | 1105 | - | | |
| 1021 | FU | 34 | 1012 | 1022 | | |
| 1022 | +, | 10 | 0001 | 1022 | | |
| 1023 | U1 | 24 | 0010 | 0020 | } | \bar{x} és D kinyomtatása |
| 1024 | Á | 05 | 0025 | 0550 | | |
| 1025 | +, | 10 | 0001 | 1025 | | |
| 1026 | U1 | 24 | 0500 | 0512 | | |
| 1027 | Á | 05 | 1200 | 0550 | | |
| 1030 | +, | 10 | 0001 | 1030 | | |
| 1031 | U1 | 24 | 0500 | 0512 | | |
| 1032 | + | 00 | 0000 | 0000 | | megállás |

Az 1003 és 1012 című rekeszek tartalmát szándékosan nem állítottuk helyre; teljes programok változó utasításait általában nem szoktuk helyreállítani.

Gyakorlásként készítsünk egy olyan teljes programot, amely az $y = \sin \frac{\pi}{2} x$ függvény értéktáblázatát az $a \leq x \leq b$ intervallumban h lépésközzel kinyomtatja (a , b és h abszolút értékben egynél kisebb számok, $\frac{a-b}{h}$ egész szám). A program elkészítésénél tegyük fel, hogy a , b és h (ebben a sorrendben) binárisan kódolt decimális alakban fel vannak lyukasztva az M-3 perforált szalagjára. A $y = \sin \frac{\pi}{2} x$ függvény szubrutinja, az input és az output szubrutin a memóriában ugyanott vannak elhelyezve, mint az 1., illetve 2. példában. A főprogram utasításait az 1000, konstansait az 1100 című memóriarekesztől kezdve helyezzük el.

7.3. Programparaméterek beültetése szubrutinokba

Az egyik előző fejezetben egy hosszabb program részeként elkészítettük két 16 dimenziós vektor skaláris szorzásának programját. A két vektor a 0000–0017, illetve a 0200–0217 rekeszokban volt elhelyezve, a skaláris szorzatot a 0603 rekeszben kaptuk meg.

A program a következő volt:

(0502) = 00 00010001

| | | | | |
|------|----|----|------------------------|-------|
| 1006 | Á | 05 | <15·2 ⁻³⁰ > | 0602 |
| 1007 | Á | 05 | <0> | 0603 |
| 1010 | (x | 13 | 0000 | 0000) |
| 1011 | ↓+ | 20 | 0603 | 0603 |
| 1012 | + | 00 | 0502 | 1010 |
| 1013 | - | 13 | <2 ⁻³⁰ > | 0602 |
| 1014 | FU | 34 | 1015 | 1010 |
| 1015 | . | . | . | . |

Ezt a programot csak 16 dimenziós vektorok skaláris szorzásánál lehetne szubrutinként felhasználni, és használata ebben az esetben is kényelmetlen lenne, mert a két vektort a szubrutin behívása előtt mindig át kellene vinni a 0000–0017, illetve a 0200–0217 rekeszbe. Készíthetünk azonban olyan általános szubrutint is, amely össze tudja szorozni skalárisan azt az n dimenziós a vektort, amelynek elemei az $a, \dots, a+n-1$ rekeszben vannak elhelyezve, azzal a szintén n dimenziós b vektorral, amelynek elemei a $b, \dots, b+n-1$ című rekeszben vannak tárolva (a, b és n a szubrutin u.n. paraméterei).

A szubrutin összeállításánál először tegyük fel azt, hogy az a, b és n paraméter-értékek 2^{-18} -nál megszorozva rendre a $p, p+1, p+2$ című rekeszben vannak elhelyezve.

Az a b skaláris szorzat értékét a szubrutin a p+3 című rekeszben képezze. A szubrutin ennek megfelelően a következő:

(k) = 2^{-12}
 (k+1) = +13 0000 0000
 (k+2) = 0
 (k+3) = +00 0001 0001
m: munkarekesz

| | | | | |
|------|-----|----|-----|------|
| t | x, | 13 | k | p+1 |
| t+1 | ↓+, | 30 | p | - |
| t+2 | ↓+ | 20 | k+1 | t+5 |
| t+3 | ↓+ | 20 | p+2 | m |
| t+4 | Á | 05 | k+2 | p+3 |
| t+5 | (| | |) |
| t+6 | ↓+ | 20 | p+3 | p+3 |
| t+7 | + | 00 | k+3 | t+5 |
| t+10 | -, | 11 | m | t+5 |
| t+11 | FU | 34 | t+5 | t+12 |
| t+12 | . | . | . | . |

A szubrutin használata ebben az alakjában elég nehézkes: behívása előtt a p, p+1, p+2 rekeszben mindig el kell helyezni az $a \cdot 2^{-18}$, $b \cdot 2^{-18}$, $n \cdot 2^{-18}$, értékeket. Ez a főprogramban a szubrutin minden egyes behívásánál általában három utasításba és három konstansba kerül. Ilyen esetekben sokszor célszerű egy memóriarekeszben több paramétert elhelyezni. Így pl., ha $n \leq 63$, az n paraméter értékét elhelyezhetjük a p című rekeszben a műveleti jel helyén; az a és b paramétereket pedig (a és b memóriarekeszek címeit jelentik, és így 2048-nál kisebbek) ugyanebben a rekeszben az első, illetve a második cím helyén: $(p) = n \cdot 2^{-6} + a \cdot 2^{-18} + b \cdot 2^{-30}$. Az a b skaláris szorzat értékét a szubrutin ebben az esetben a p+1 című rekeszben tárolhatja; a p+2 és p+3 című paraméter-rekeszekre nincs szükség. Ennek megfelelően természetesen az előbb felírt szubrutint át kell alakítani. Az új szubrutin:

(k) = +00 7777 7777
 (k+1) = +13 0000 0000
 (k+2) = +77 0000 0000
 (k+3) = 2^{-12}
 (k+4) = +0
 (k+5) = +00 0001 0001
m: munkarekesz

| | | | | |
|------|-----|----|-----|------|
| t | Λ, | 16 | k | p |
| t+1 | ↓+, | 30 | k+1 | t+0 |
| t+2 | Λ, | 16 | k+2 | p |
| t+3 | ↓x, | 33 | k+3 | - |
| t+4 | ↓+ | 20 | t+6 | m |
| t+5 | Á | 05 | k+4 | p+1 |
| t+6 | (| | |) |
| t+7 | ↓+ | 20 | p+1 | p+1 |
| t+10 | + | 00 | k+5 | t+6 |
| t+11 | -, | 11 | m | t+6 |
| t+12 | FU | 34 | t+6 | t+13 |
| t+13 | . | . | . | . |

A skaláris szorzás szubrutinját jól fel lehet használni az alábbi programozási feladat megoldásánál.

Szoroztassuk össze az M-3 géppel azt a 35-ödrendű A mátrixot, amelynek elemei "sorfolytonosan" a 0100-2410 rekeszekben vannak tárolva (vagyis az A mátrix a_{ij} eleme ($i, j = 1, \dots, 35$) a $0100+(i-1)43^+ + (j-1)$ rekeszben van elhelyezve) azzal a b vektorral, amelynek elemei a 2500-2542 rekeszben vannak tárolva. Az A b vektor elemeit nyomtassuk ki, az utolsó elem kinyomtatása után a gép álljon meg.

Feltesszük, hogy a 0005-0100 rekeszek szabadok; ebben az esetben a skaláris szorzás szubrutinjának utasításait elhelyezhetjük például a 0005 rekesztől kezdve (vagyis $t = 0005$); legyen továbbá $p = 0021$, $k = 0023$, $m = 0031$.

⁺ 35 nyolcas rendszerben 43

Az output szubrutin utasításait a 0032 rekesztől kezdve, a főprogram utasításait a 0050, konstansait a 0070 rekesztől kezdve helyezzük el. A főprogram a következő:

(0070) = -43 0100 2500
(0071) = +00 0043 0000
(0072) = +43 2410 0000
(0001) = +64 0000 0002
(szabványos konstans)

| | | | | |
|------|-----|-----|------|------|
| 0050 | Á | 05 | 0070 | 0021 |
| 0051 | +, | 10 | 0001 | 0051 |
| 0052 | U1 | 24 | 0005 | 0020 |
| 0053 | Á | 05 | 0022 | 0045 |
| 0054 | +, | 10 | 0001 | 0054 |
| 0055 | U1 | 24 | 0032 | 0043 |
| 0056 | + | 00 | 0071 | 0070 |
| 0057 | ↓-, | 31 | 0072 | - |
| 0060 | FU | 34 | 0050 | 0061 |
| 0061 | | +77 | 0000 | 0000 |

Ha a skaláris szorzás szubrutinjának első változatát alkalmaztuk volna (vagyis a skaláris szorzás három paraméterét három memóriarekeszben helyeztük volna el), akkor a főprogram két utasítással és egy konstanssal tartalmazna többet.

7.4. Szabványos szubrutinok relatív címzéssel.

A szubrutinok alkalmazásával kapcsolatban kidolgozott példákból jól láthatjuk, hogy ugyanazt a szubrutint (különböző főprogramok mellett) gyakran különböző helyeken kell tárolni a memóriában. A szubrutinokat mindig ugyanazon a helyen tárolni már csak azért sem lehetséges, mert az M-3 gépre eddig elkészített szubrutinok együttesen többszörösen kitöltenék a gép memóriakapacitását. Ezért a szubrutin-

nok utasításainak, konstansainak, paramétereinek és munka-rekeszeinek célszerű relatív címeket (betű címeket) adni. A relatív címeket valóságos (abszolút) címekké átalakítani általában könnyebb, mint az abszolút címekkel elkészített programot átcímezni. Az eddig kidolgozott szubrutinok közül relatív címmel készítettük el skaláris szorzás és az $y = e^{-x}$ függvény szubrutinját.

A relatív címezés egymagában persze meg nem oldja meg a szubrutinok racionális felhasználásánál: problémáját. A relatív címek átalakítása abszolút címekké teljesen gépies, de hosszabb szubrutinoknál elég hosszadalmas és fáradságos munka. Ezenkívül a szubrutint minden esetben újra kell lyukasztani, ami szintén elég sok időt és munkát vesz igénybe. Gyökeres megoldást csak az jelent, ha egy olyan programot készítünk az M-3 gépre, amelynek segítségével a perforált szalagra relatív címmel fellyukasztott szubrutinokat a gép automatikusan a memória kijelölt helyeire viszi be, és közben a relatív címeket átalakítja megfelelő abszolút címekké. Az ilyen programot beindító rutinnak, a beindító rutin segítségével a memóriában akárhol elhelyezhető, relatív címmel szubrutinokat szabványos vagy könyvtári szubrutinoknak, a perforált szalagra fellyukasztott szabványos szubrutinok összességét szubrutinkönyvtárnak nevezzük.

Az M-3 beindító rutinjára vonatkozóan számos terv született, ezek közül itt azt a tervet fogjuk részletesebben ismertetni, amely lényegében az angol EDSAC gép beindító rutinjának elvét viszi az M-3-ra.

Ez a beindító rutin minden olyan relatív címmel szubrutin bevitelére alkalmas, amelyben a relatív címzés alapjainak száma (a szereplő "betűk" száma) legfeljebb 15. A relatív címezés alapjait célszerű az a, b, c, ..., betűkkel jelölni. A szabványos szubrutinoknak az a, b, c, ...0

alapú relatív címeken kívül szerepelhetnek abszolút címek is (pl. a szabványos konstansok elmei). A szabványos szubrutinok konstansait a memóriában általában közvetlenül a szubrutin utasításai után helyezük el.

Egy szabványos szubrutin utasításait és konstansait ennek megfelelően egy szalagra visszük fel, és pedig először az utasításokat, utánuk a konstansokat. Egy-egy utasítás, illetve konstans felvitele a perforált szalagra a következő módon történik: minden egyes utasítást (konstanst) először a relatív címek alapjainak feltüntetése nélkül lyukasztunk fel (tehát például a $+05\ b + 7\ c + 11$ konstansnál először $+05\ 0007\ 0011$ -et lyukasztunk). Az utasítás (konstans) végét jelző "írás" jel után még öt sort perforálunk a következő szabály szerint: az első két sorba 00-t perforálunk, ha az első cím abszolút, 01-et, ha az első cím relatív, és alapja $a, \dots, 17$ -et, ha az első cím relatív, és alapja 0. A következő két sorba 00-t lyukasztunk, ha a második cím abszolút, 01-et, ha a második cím relatív, és alapja $a, \dots, 17$ -et, ha a második cím relatív, és alapja 0. Az ötödik sorba egy "írás" jelet lyukasztunk.

Az M-3 gép szubrutinkönyvtára a terv szerint az ilyen módon perforált szabványos szubrutinokból fog állni. A szabványos szubrutinok bevitelét, és magát a beindító rutint egy következő fejezetben ismertetjük.

A beindító rutinnak a szubrutinok relatív címezéssel való bevitelén kívül még egy jelentős alkalmazása van. A szubrutinokban azokon a paramétereken kívül, amelyeknek értékét a főprogram helyezi el a szubrutin paraméter-rekeszeiben, gyakran szerepelnek olyan paraméterek is, amelyeknek értékét minden egyes alkalommal a programozó írja be a szubrutin megfelelő helyeire. Az előbbieket program-paraméternek, az utóbbiakat előre beállított (preset) paraméternek nevezzük. Így például a skaláris szorzás fent ismertetett szubrutinjában a , b és n program-paraméterek.

Ha a, b és n előre beállított paraméterek lennének, akkor a szubrutin egyszerűen a következő lenne (a b ==> p1):

| | | | | | |
|-------|---|----|----|------|------|
| (k) | = | x, | 13 | a | b |
| (k+1) | = | x, | 13 | a+n | b |
| (k+2) | = | + | 00 | 0000 | 0000 |
| (k+3) | = | - | 00 | 0001 | 0001 |

| | | | | |
|-----|-----|----|-----|-----|
| t | Á | 05 | k | t+2 |
| t+1 | Á | 05 | k+2 | p |
| t+2 | (| | |) |
| t+3 | ↓+ | 20 | p | p |
| t+4 | + | 00 | k+3 | t+2 |
| t+5 | ↓-, | 30 | k+1 | - |
| t+6 | FU | 34 | t+2 | t+7 |
| t+7 | . | . | . | . |

Ebben az alakban a szubrutint természetesen csak akkor lehet használni, ha a, b és n értéke a főprogram végrehajtása során állandó; ezeket az állandó értékeket a programozónak kellene minden alkalommal külön beírni. A beindító rutin segítségével a szabványos szubrutin bevitelénél a gép elvégzi a relatív címeknek abszolút címekké való átalakításán kívül az előre beállított paraméterek értékeinek behelyettesítését is.

7.5. További példák.

7.5.1. Elemi függvények szubrutinjai.

Az alábbiakban néhány fontos elemi függvény szubrutinját mutatjuk be.

1. Az $y = \ln x$ függvény szubrutinba.

Ennek a függvénynek a kiszámítására két különböző szubrutint fogunk vizsgálni.

a.) Tudjuk, hogy az x argumentum 0 -tól $+\infty$ -ig változhat. Ezért a fix bináris pontot a 15 -ik helyérték után helyezzük el. Így $2^{-15} \leq x < 2^{15}$. (Ez az intervallum elég nagy ahhoz, hogy a gyakorlatilag előforduló értékek logaritmusát ki tudjuk számítani.) Ekkor $|\ln x| < 2^4$ lesz. Így a gép kapacitásának teljes kihasználása céljából az eredménynél a bináris pontot a negyedik helyérték után helyezzük el. Helyettesítsünk $x = 1 + y$ -t, és fejtsük az $\ln x = \ln(1+y)$ függvényt Taylor sorba.

$$\ln(1+y) = y - \frac{y^2}{2} + \frac{y^3}{3} - \dots + (-1)^{n-1} \frac{y^n}{n} + \dots$$

Ez a sorfejtés a $-1 < y \leq 1$ intervallumban érvényes.

A sor akkor konvergál jól, ha y közel van a nullához, azaz ha a tulajdonképpeni argumentum, $x = 1+y$ közel van 1 -hez. Ezért a $[2^{-15}, 2^{15})$ intervallumot leképezzük az $[1, \frac{6}{5})$ intervallumra. Ezt a következő lépésekben végezzük el:

I. Az x számot $x = d \cdot 2^q$; $\frac{1}{2} \leq d < 1$; $-15 \leq q < 15$ alakba írjuk. Tudjuk, hogy ez az x számnak az ún. lebegőpontos alakja. Mivel a gépben $\bar{x} = x \cdot 2^{-15}$ van elhelyezve, ezt hozzuk $\bar{x} = d \cdot 2^{-15} = d \cdot 2^k$ alakra, ahol $-30 \leq k < 0$. Ekkor

$$\log x = \log d + q \cdot \ln 2.$$

Tehát a feladat $\log d$ kiszámítására redukálódott, ahol $\frac{1}{2} \leq d < 1$.

II. A $t = (\frac{6}{5})^i d$ leképezéssel, ahol $i = 1, 2, 3, 4$ a szerint, hogy d az $[\frac{5}{6}, 1)$, $[(\frac{5}{6})^2, \frac{5}{6})$, $[(\frac{5}{6})^3, (\frac{5}{6})^2)$, $[\frac{1}{2}, (\frac{5}{6})^3)$ intervallumok közül melyikben van, $1 \leq t < \frac{6}{5}$ adódik.

III. A $t = 1 + \frac{z+1}{10}$ leképezéssel $-1 \leq z < 1$ adódik.

Így

$$\ln t = \ln \left(1 + \frac{z+1}{10} \right) = \frac{z+1}{10} - \frac{1}{2} \left(\frac{z+1}{10} \right)^2 + \dots - \frac{1}{10} \left(\frac{z+1}{10} \right)^{10} + R_{10}$$

ahol az R_{10} maradéktagra az alábbi becslés adható:

$$|R_{10}| = \left| \frac{1}{11} \cdot \left(\frac{\frac{z+1}{10}}{1+\xi} \right)^{11} \right| \leq \frac{1}{11} \left(\frac{1}{5} \right)^{11} < 0,2 \cdot 10^{-8}$$

mert

$$0 \leq \xi \leq \frac{z+1}{10} \leq \frac{1}{5}$$

A hatványozásokat elvégezve:

$$\ln t = c'_0 + c'_1 z + c'_2 z^2 + \dots + c'_8 z^8 + c'_9 z^9 + c'_{10} z^{10} + R_{10}$$

Az együtthatókat kiszámítva azt kapjuk, hogy

$$|R'| = |c'_8 z^8 + c'_9 z^9 + c'_{10} z^{10}| < 0,08 \cdot 10^{-8}$$

tehát ez a három tag a gépi számításnál elhagyható. Ezáltal a közelítő polinom hetedfokúra redukálódott. Ennek a fokszáma még kettővel csökkenthető a Csebisev-polinomok segítségével; z^6 t.i. egy negyedfokú, z^7 pedig egy ötöd-fokú polinommal pótolható a tekintett intervallumban, és pedig úgy, hogy a hiba R' nagyságrendjével összemérhető. Az n -ed fokú Csebisev polinom definíciója a következő:

$$T_n(z) = \frac{1}{2^{n-1}} \cdot \cos(n \arccos z)$$

Látható, hogy $|T_n(z)| \leq \frac{1}{2^{n-1}}$

Ennek alapján könnyen felírhatjuk a hatod és hetedfokú Csebisev polinomokat:

$$T_7(z) = z^7 - \frac{7}{4}z^5 + \frac{7}{8}z^3 - \frac{7}{64}z \leq \frac{1}{64}$$

$$T_6(z) = z^6 - \frac{3}{2}z^4 + \frac{9}{16}z^2 - \frac{1}{32} \leq \frac{1}{32}$$

Így

$$c'_6 z^6 + c'_7 z^7 = c'_6 \left(\frac{3}{2} z^4 - \frac{9}{16} z^2 + \frac{1}{32} \right) + c'_7 \left(\frac{7}{4} z^5 - \frac{7}{8} z^3 + \frac{7}{64} z \right) + R''$$

$$R'' = c'_6 T_6(z) + c'_7 T_7(z) \text{ és így } R'' \leq 0,032 \cdot 10^{-8}$$

Végeredményben tehát:

$$\ln t = c_0 + c_1 z + c_2 z^2 + c_3 z^3 + c_4 z^4 + c_5 z^5 + R$$

ahol

$$|R| \leq |R_{10}| + |R'| + |R''| < 0,6 \cdot 10^{-8}$$

és

$$c_0 = 0,095310177$$

$$c_1 = 0,090909092$$

$$c_2 = -0,004132178$$

$$c_3 = 0,000250432$$

$$c_4 = -0,00017217$$

$$c_5 = 0,000001254$$

$\ln x \approx q \ln 2 + \ln d = q \ln 2 - i \ln \frac{6}{5} + c_6 + \dots + c_5 z^5$,
ahol az aláhúzás 2^{-4} -nel való szorzást jelent.

Program:

| | | | | | |
|------|----|------|----------|------|--|
| 0001 | 11 | -, | 0101 | 0100 | } $x \leq 0$ -ra megállás |
| 0002 | 34 | FU | 0003 | 0004 | |
| 0003 | | | megállás | | |
| 0004 | 12 | ∴, | 0102 | 0103 | } $16 \ln 2 \implies 0116$ |
| 0005 | 24 | UI | 0006 | 0116 | |
| 0006 | 05 | Á | 0100 | 0117 | } $d = 2^k x \implies 0017$ |
| 0007 | 01 | - | 0103 | 0116 | |
| 1010 | 51 | - , | 0104 | 0117 | } $(0116) =$ $= (15+k) \ln 2 = q \ln 2$ |
| 0011 | 34 | FU | 0012 | 0014 | |
| 0012 | 02 | : | 0104 | 0117 | |
| 0013 | 24 | UI | 0007 | 0117 | |

| | | | | | |
|------|---------------------------|-----|------|-------|--|
| 0014 | 03 | x | 0104 | 0117 |] $\frac{d}{2} \implies 0117$ |
| 0015 | 01 | - | 0105 | 0116 | |
| 0016 | 02 | : | 0021 | 0021 |] $(0116) = q \ln 2 -$] $- i \ln \frac{6}{5}$ |
| 0017 | 21 | ↓- | 0104 | 0120 | |
| 0020 | 34 | FU | 0015 | 0021 | |
| 0021 | 02 | : | 0104 | 0120 | |
| 0022 | 11 | -, | 0107 | 0120 |] $z = \frac{(\frac{6}{5})^i d-1-0,1}{0,1} \implies$] $\implies 0120$ |
| 0023 | 22 | ↓: | 0107 | 0120 | |
| 0024 | 05 | Á | 0110 | 0117 |] $\ln t$] polinom kiszámítása] Horner elrendezés-] sel] $\longrightarrow \ln x$ |
| 0025 | 13 | x, | 0117 | 0120 | |
| 0026 | (20 | ↓+ | 0111 | 0117) | |
| 0027 | 00 | + | 0035 | 0026 | |
| 0030 | 51 | - , | 0026 | 0036 | |
| 0031 | 34 | FU | 0032 | 0025 | |
| 0032 | 01 | - | 0037 | 0026 | |
| 0033 | 00 | + | 0116 | 0117 | |
| 0034 | (kiugrató utasítás helye) | | | |] Pseudó utasítások |
| 0035 | 00 | + | 0001 | 0000 | |
| 0036 | 20 | ↓+ | 0115 | 0117 | |
| 0037 | 00 | + | 0005 | 0000 | |

Konstansok:

- (0000) = 0
- (0100) = x
- (0101) = 2^{-30}
- (0102) = 2^{-4}
- (0103) = $\ln 2$
- (0104) = $\frac{1}{2}$
- (0105) = $\frac{\ln \frac{6}{5}}{5}$
- (0106) = $\frac{5}{6}$
- (0107) = 0,1
- (0110) = c_5
- (0111) = c_4
- (0112) = c_3

$$(0113) = c_2$$

$$(0114) = c_1$$

$$(0113) = c_0$$

b.) Abban az esetben, ha $0 \leq x \leq 1$, az $\ln(1+x)$ függvény egy nyolcadfokú polinommal jól közelíthető.

$$\ln(1+x) = a_1x + a_2x^2 + \dots + a_8x^8$$

ahol

$$a_1 = 0,9999964239$$

$$a_5 = 0,1676540711$$

$$a_2 = -0,4998741238$$

$$a_6 = -0,0953293897$$

$$a_3 = 0,3317990258$$

$$a_7 = 0,0360884937$$

$$a_4 = -0,2407338084$$

$$a_8 = 0,0064535442$$

és $|R'| < 4 \cdot 10^{-8}$

(A képlet megtalálható O. Hastings: "Approximations for Digital Computers" című könyvében.)

A fenti polinom az $\ln(1+x)$ függvényt a $0 \leq x \leq 1$ intervallumban közelíti jól. Ha $y = 1 + x$ az $1 \leq y \leq 2$ szakaszon kívül esik, akkor az $\bar{y} = \frac{y}{2^k}$ transzformáció segítségével ebbe az intervallumba hozzuk. Ebben az esetben a keresett logaritmusérték

$$\ln y = \ln \bar{y} + k \ln 2$$

alakban nyerhető, és a Hastings-féle polinomot az $x = \bar{y} - 1$ helyen kell kiszámítani. Mindezek a transzformációk természetesen a főprogramban végzendők el.

Így a program egy nyolcadfokú polinom kiszámítására korlátozódott. Ezt a nyolcadfokú polinomot a Horner-elrendezés segítségével számítjuk ki.

Program:

| | | | | |
|------|----|-----|------|-------------------------|
| 0001 | 05 | Á | 0012 | 0003 |
| 0002 | 05 | Á | 0000 | 0015 |
| 0003 | | | |) |
| 0004 | 23 | ↓x | 0013 | 0015 |
| 0005 | 00 | + | 0014 | 0003 |
| 0006 | 51 | - , | 0003 | 0011 |
| 0007 | 34 | FU | 0010 | 0003 |
| 0010 | | | | kiugrató utasítás helye |

Konstansok:

| | | | | |
|--------|---|-----------|------|------|
| (0011) | = | +00 | 0027 | 0015 |
| (0012) | = | +00 | 0020 | 0015 |
| (0013) | = | x | | |
| (0014) | = | 2^{-18} | | |
| (0000) | = | 0 | | |
| (0020) | = | a_8 | | |
| (0021) | = | a_7 | | |
| (0022) | = | a_6 | | |
| (0023) | = | a_5 | | |
| (0024) | = | a_4 | | |
| (0025) | = | a_3 | | |
| (0026) | = | a_2 | | |
| (0027) | = | a_1 | | |

Látjuk, hogy ez az utóbbi program jóval rövidebb és egyszerűbb az előbbinél. Mégis vannak hátrányai az előbbivel szemben. Pl.: az első program a $2^{-15} \leq x < 2^{15}$ intervallumba eső argumentum értékekre használható, a transzformáció a szubrutinban történik. Az utóbbi viszont csak akkor alkalmazó, ha $0 \leq x \leq 1$, minden más esetben a főprogramban végre kell hajtani azt a transzformációt, amely x-et ebbe az intervallumba hozza.

2. Trigonometrikus függvények szubrutinjai.

A trigonometrikus függvények kiszámításánál felhasználjuk a Hastings-féle közelítő polinomokat.

$\sin \frac{\pi}{2} x$ kiszámítása pl. a következő közelítő formula alapján történik:

$$\sin \frac{\pi}{2} x \approx c_1 x + c_3 x^3 + c_5 x^5 + c_7 x^7 + c_9 x^9$$

ahol

$$-1 \leq x \leq 1$$

és

$$c_1 = 1,57079631847$$

$$c_3 = -0,64596371106$$

$$c_5 = 0,07968967928$$

$$c_7 = -0,00467376557$$

$$c_9 = 0,00015148419$$

Ekkor a számítás relatív hibája kisebb lesz, mint $5 \cdot 10^{-9}$. Az M-3 gépen pedig az eredmény maximális pontossága csak 7 tizedesjegy. Mivel a trigonometrikus függvények argumentuma $-\infty$ -től $+\infty$ -ig változhat, a számítás megkezdése előtt az argumentumot úgy kell redukálni, hogy abszolút értékben ne legyen nagyobb $\frac{\pi}{2}$ -nél.

Ezt a redukciót több lépésben hajtjuk végre. Az eredeti argumentumot y -nal jelöljük.

I. Az argumentum abszolút értékét vesszük. Ezt megtehetjük, mivel

$$\sin(-y) = -\sin y$$

$$\cos(-y) = \cos y$$

Tehát $\sin |y|$ kiszámítása után, ha az eredeti argumentum negatív volt, a kiszámított függvényértéket ellenkező előjellel kell venni.

II. $|y|$ -ből levonjuk 2π egész számú többszörösét úgy, hogy $-\pi \leq |y| - 2k\pi = \underline{y} < \pi$ legyen.

III. $\frac{y}{2} = \dot{y}$ -t képezzük. Ez az érték már a megadott intervallumba esik.

IV. Az eredeti argumentum sinusát és cosinusát a következő képletek alapján számítjuk ki:

$$\sin x = 2 \sin \frac{x}{2} \cos \frac{x}{2}$$

$$\cos x = \cos^2 \frac{x}{2} - \sin^2 \frac{x}{2}$$

A Hastings-féle közelítő polinom $\sin \frac{\pi}{2} x$ kiszámítására alkalmas, ahol x a $[-1, 1]$ intervallumba esik. Ezért a redukált \dot{y} argumentumot $\frac{2}{\pi}$ -vel meg kell szorozni, és erre a $\frac{2}{\pi} \dot{y} = x$ -re kell alkalmazni a képletet. Így $\sin \dot{y}$ -ot fogjuk kapni eredményül.

$\cos \dot{y}$ -ot a következő összefüggés segítségével kapjuk meg:

$$\cos \dot{y} = \sin \left(\frac{\pi}{2} - |\dot{y}| \right)$$

Tehát képezzük a $z = \frac{\pi}{2} - |\dot{y}|$ értéket, ennek a $\frac{2}{\pi}$ -szeresét, tehát $1 - \frac{2}{\pi} |\dot{y}| = 1 - |x|$ értéket, és majd erre alkalmazzuk a Hastings-féle közelítést.

A trigonometrikus függvények kiszámítására két-féle szubrutint ismertetünk:

a.) Szubrutin, amely vagy $\sin \dot{y}$ vagy $\cos \dot{y}$ ($-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$) kiszámítására alkalmas, aszerint, hogy hol kezd el a gép a program végrehajtását.

Mivel a Hastings-féle közelítő polinomnak van 1-nél nagyobb együtthatója is, ezeknek az együtthatóknak a felét helyezzük el a gépben, ezekkel számolunk, majd a kapott eredményt 2-vel megszorozzuk.

A szubrutin során felhasználásra kerülő konstansok a következők:

$$\begin{aligned}(0040) &= \frac{C_1}{2} \\(0041) &= \frac{C_3}{2} \\(0042) &= \frac{C_5}{2} \\(0043) &= \frac{C_7}{2} \\(0044) &= \frac{C_9}{2} \\(0045) &= + 20 0043 0057 \\(0046) &= + 20 0040 0057 \\(0050) &= + 77 7777 7777 \\(0051) &= + 00 0001 0000 \\(0052) &= \frac{1}{2}\end{aligned}$$

Az x argumentumot a 0060-as rekeszben helyezzük el. Az eredmény mindig a 0057-es rekeszben képződik.

Munkarekesznek felhasználjuk a 0056-os rekeszt.

$\sin x$ számításánál a bemenet: 0001
 $\cos x$ " " : 0003

Program:

| | | | | |
|------|------|----|------|------|
| 0001 | Á | 05 | 0060 | 0056 |
| 0002 | U2 | 74 | 0000 | 0005 |
| 0003 | - , | 51 | 0060 | 0050 |
| 0004 | U1 | 24 | 0005 | 0056 |
| 0005 | Á | 05 | 0045 | 0011 |
| 0006 | Á | 05 | 0044 | 0057 |
| 0007 | X, | 13 | 0056 | 0056 |
| 0010 | ↓X, | 33 | 0057 | - |

| | | | | |
|------|-------------|----|------|-----------------------|
| 0011 | (.) | | | |
| 0012 | - | 01 | 0051 | 0011 |
| 0013 | ↓ - , | 71 | 0046 | - |
| 0014 | FU | 34 | 0017 | 0007 |
| 0015 | X | 03 | 0056 | 0057 |
| 0016 | ↓: | 22 | 0052 | 0057 |
| 0017 | | | | kiugró utasítás helye |

b.) Szubrutin, amely kiszámítja $\sin x$ -et, $\cos x$ -et és $\operatorname{tg} x$ és $\operatorname{cotg} x$ közül az egyiket aszerint, hogy $|x|$ kisebb vagy nagyobb $\frac{\pi}{4}$ -nél.

Ugyanis, ha $|x| < \frac{\pi}{4}$, akkor $\operatorname{tg} x < 1$ és $\operatorname{cotg} x > 1$
 ha $|x| > \frac{\pi}{4}$, akkor $\operatorname{cotg} x < 1$ és $\operatorname{tg} x > 1$

$\operatorname{tg} x$ -et és $\operatorname{cotg} x$ -et a következő összefüggések felhasználásával számítjuk:

$$\operatorname{tg} x = \frac{\sin x}{\cos x}; \operatorname{cotg} x = \frac{\cos x}{\sin x}$$

A szubrutin ugyanazokat a konstansokat használja, mint az előbbi, ezeken kívül azonban még további hármat:

$$\begin{aligned} (0047) &= \frac{2}{\pi} \\ (0053) &= 2 \cdot x^{-30} = + 00\ 0000\ 0001 \\ (0054) &= 0 = + 00\ 0000\ 0000 \end{aligned}$$

Az x argumentumot ($-1 < x < 1$) a 0060-as rekeszben helyezzük el. Az eredmények a következő rekeszbe kerülnek:

$$\begin{aligned} \sin x &\implies 0061 \\ \cos x &\implies 0062 \\ \operatorname{tg} x &\implies 0063 \\ \operatorname{cotg} x &\implies 0064 \end{aligned}$$

$\operatorname{tg} x$ vagy $\operatorname{cotg} x$ értéke egynél nagyobb lesz: ennek a helyére a gép 0-t ír, ami itt nem eredményt jelez.

Program:

| | | | | | |
|------|-----------|----|------|-------|-------------------------|
| 0000 | Á | 05 | 0053 | 0055 | |
| 0001 | x, | 13 | 0047 | 0060 | |
| 0002 | U1 | 24 | 0005 | 0056 | |
| 0003 | - , | 51 | 0056 | 0050 | |
| 0004 | U1 | 24 | 0005 | 0056 | |
| 0005 | Á | 05 | 0045 | 0011 | |
| 0006 | Á | 05 | 0044 | 0057 | |
| 0007 | x, | 13 | 0056 | 0056 | |
| 0010 | ↓x, | 33 | 0057 | - | |
| 0011 | (.. .. .) | .. | |) | |
| 0012 | - | 01 | 0051 | 0011 | |
| 0013 | ↓ - , | 71 | 0046 | - | |
| 0014 | FU | 34 | 0015 | 0007 | |
| 0015 | x | 03 | 0056 | 0057 | |
| 0016 | ↓: | 22 | 0052 | 0057 | |
| 0017 | - | 01 | 0053 | 0055 | |
| 0020 | FU | 34 | 0023 | 0020 | ← sin x |
| 0021 | Á | 05 | 0057 | 0061 | |
| 0022 | U2 | 74 | - | 0003 | ← cos x |
| 0023 | Á | 05 | 0057 | 0062 | |
| 0024 | - , | 51 | 0061 | 0062 | |
| 0025 | FU | 34 | 0031 | 0026 | |
| 0026 | Á | 05 | 0054 | 0064 | |
| 0027 | :, | 12 | 0062 | 0061 | |
| 0030 | U1 | 24 | 0034 | 0063 | ← tg x |
| 0031 | Á | 05 | 0054 | 0063 | |
| 0032 | :, | 12 | 0061 | 0062 | |
| 0033 | U1 | 24 | 0034 | 0064 | ← cotg x |
| 0034 | | | | | kiugrató utasítás helye |

3. A trigonometrikus függvények inverzeinek szubrutinjai:

3.a.) Arc sin x és arc cos x szubrutinja:

Felhasználjuk az

$$\text{arc sin } x = \frac{\pi}{4} \left\{ \text{sign } y_1 + \frac{1}{2} \text{sign } (y_1 y_2) + \dots + \frac{1}{2^{k-1}} \text{sign } (y_1 \dots y_k) + \dots \right\}$$

sorfejtést, ahol $y_1 = x$

$$y_{k+1} = 2 y_k^2 - 1$$

Ez a sorfejtés arc sin x-et lépcsősfüggvények limeszeként állítja elő. Szorítkozzunk $|x| \leq 1 - 2^{-30}$ -ra. Ez nem jelent lényeges megszorítást, mert $|x| = |\sin y| \leq 1$.

A gép kapacitását figyelembe véve, elegendő a sorfejtésben annyi tagig elmenni, ameddig

$$\left| \frac{\pi}{4} \frac{1}{2^{k-1}} \right| \geq 2^{-30}$$

még érvényes. Az ennél kisebb értékű tagokat a gép már nem tudja figyelembe venni, s az ezen tag utáni maradékösszeget sem, mivel az összegezés szukcesszíve történik. Így tehát $k \leq 30$ adódik. A $k \geq 30$ indexű tagok maradékösszege t.i. szintén kisebb, mint 2^{-29} , hiszen

$$|\text{arc sin } x - S_k| = \left| \frac{\pi}{4} \left(\frac{1}{2^k} \text{sign } (y_1 \dots y_{k+1}) + \dots \right) \right| \leq$$

$$\leq \frac{\pi}{4} \left(\frac{1}{2^k} + \frac{1}{2^{k+1}} + \dots \right) = \frac{\pi}{4} \frac{1}{2^{k-1}}$$

A $-\frac{\pi}{2} \leq \text{arc sin } x \leq \frac{\pi}{2}$ közön arc sin x egynél nagyobb értékeket is felvehet. Vegyünk ezért arc sin x helyett 2^{-1}

arc sin x-et, azaz a képletben $\frac{\pi}{4}$ helyett $2^{-1} \frac{\pi}{4}$ -et. Mivel $\text{arc sin } (-x) = - \text{arc sin } x$, tehát akkor sincs baj, ha az argumentum negatív, csak ezt a végeredménynél figyelembe kell venni.

Emellett $\text{arc cos } x = \frac{\pi}{2} - \text{arc sin } x$, és így arc sin x kiszámított értékéből könnyen megkapjuk arc cos x értékét is. Az y_{k+1} -et meghatározó rekurziós formulában y_{k+1} helyett $\frac{1}{2} y_{k+1}$ -gyel számolunk. Ezt megtehetjük, mivel ennek csak az előjelére van szükség. Tehát

$$\frac{y_{k+1}}{4} = 2 \frac{y_k^2}{4} - \frac{1}{2}$$

A felhasználásra kerülő konstansok a következők:

| | | | | |
|--------|---|------------------------------|-------------------------------------|------|
| (0000) | = | 0 | | |
| (0010) | = | x | $2^{-1} \text{arc sin } x \implies$ | 0036 |
| (0011) | = | $\frac{\pi}{2} \cdot 2^{-1}$ | $2^{-1} \text{arc cos } x \implies$ | 0037 |
| (0013) | = | $\frac{1}{2}$ | | |
| (0014) | = | $\frac{1}{4}$ | | |

A szubrutint a 0015-ös rekesztől kezdve helyezzük el.

Program:

| | | | | | |
|------|----|----|------|------|---|
| 0015 | x | 03 | 0014 | 0010 | |
| 0016 | Á | 05 | 0011 | 0040 | |
| 0017 | Á | 05 | 0000 | 0036 | |
| 0020 | x | 03 | 0013 | 0040 | |
| 0021 | : | 02 | 0013 | 0010 | |
| 0022 | FU | 34 | 0023 | 0025 | |
| 0023 | -, | 11 | 0040 | 0000 | |
| 0024 | U1 | 24 | 0025 | 0040 | |
| 0025 | + | 00 | 0040 | 0036 | $\longleftarrow 2^{-1} \text{arc sin } x$ |
| 0026 | x, | 13 | 0010 | 0010 | |

| | | | | |
|------|-----|----|-------------------------|---------------------------|
| 0027 | ↓:, | 32 | 0013 | - |
| 0030 | ↓- | 21 | 0014 | 0010 |
| 0031 | - , | 51 | 0040 | 0000 |
| 0032 | FU | 34 | 0033 | 0020 |
| 0033 | -, | 11 | 0036 | 0011 |
| 0034 | U1 | 24 | 0035 | 0037 ← 2^{-1} arc cos x |
| 0035 | | | kiugrató utasítás helye | |

3.b.) Arc tg x és arc cotg x szubrutinja.

Felhasználjuk az

$$\text{arc tg } x = \text{arc sin } \frac{x}{\sqrt{1+x^2}}$$

$$\text{arc cotg } x = \text{arc cos } \frac{x}{\sqrt{1+x^2}} \text{ összefüggéseket.}$$

Mind az arc tg x, mind az arc cotg x értelmezési tartománya $-\infty \leq x \leq \infty$, a gép kapacitása miatt szorítkozunk $|x| \leq 2^{15}$ -re. Tekintsük tehát az x argumentum helyett az $x \cdot 2^{-15}$ -öt:

$$\text{arc tg } x = \text{arc sin } \frac{x \cdot 2^{-15}}{\sqrt{2^{-30} + x^2 \cdot 2^{-30}}}, \text{ illetve}$$

$$\text{arc cotg } x = \text{arc cos } \frac{x \cdot 2^{-15}}{\sqrt{2^{-30} + x^2 \cdot 2^{-30}}}$$

Mivel mind arc tg x, mind pedig arc cotg x értékészlete túllépi az egységet, ezért 2^{-1} arc tg x-et, illetve 2^{-1} arc cotg x-et számítjuk ki.

Legyen a négyzetgyökvonó szubrutin az α -tól $\alpha+k-1$ -ig terjedő rekeszekben elhelyezve, és legyen az argumentum helye a γ , az eredmény helye a β rekesz.

A felhasznált rekeszek a következők;

$$(0046) = x \cdot 2^{-15}$$

$$(0044) = + 74 \dots 0054$$

$$(0045) = 2^{-30}$$

$$2^{-1} \text{ arc tg } x \implies 0034; 2^{-1} \text{ arc cotg } x \implies 0037$$

Program:

| | | | | | |
|------|-----|----|----------|------------|--|
| 0050 | Á | 05 | 0044 | $\alpha+k$ | |
| 0051 | x, | 13 | 0046 | 0046 | |
| 0052 | ↓+, | 30 | 0045 | - | |
| 0053 | U1 | 24 | α | γ | |
| 0054 | : | 02 | β | 0046 | |
| 0055 | U1 | 24 | 0015 | 0010 | áttérés arc sin x és arc cos x kiszámítására, amelyek eredményeképp a keresett függvényértékeket kapjuk. |

7.5.2. Szubrutin az $\int_a^b f(x) dx$ kvadratúrájához.

A numerikus kvadratúra szubrutinja valamely függvény határozott integráljának közelítő értékét számolja a Simpson-formula alapján; így

$$\int_a^b f(x) dx = \frac{h}{3} [f(x_0) + 4f(x_1) + 2f(x_2) + \dots + 4f(x_{n-1}) + f(x_n)] + H$$

ahol a H a közelítés hibája és

$$h = \frac{b-a}{n}; \quad x_0 = a; \quad x_i = x_0 + ih \quad \text{és } n \text{ páros.}$$

Jelöljük $\max(|a|, |b|)$ értékét $\frac{L}{2}$ -vel, és legyen $|f(x)| < F$ az (a, b) intervallumban. Ekkor $|\frac{h}{3}| = |\frac{b-a}{3n}| \leq \frac{L}{3n}$, és a határozott integrálra a Simpson-formula alapján a következő becslést kapjuk:

$$\left| \int_a^b f(x) dx \right| \leq \frac{h}{3} [|f(x_0)| + 4 |f(x_1)| + 2 |f(x_2)| + \dots + 4 |f(x_{n-1})| + |f(x_n)|] + |H| < \frac{L}{3n} F (1+4+2+\dots+4+1) + |H| = \frac{L}{3n} F \cdot 3n + |H| = L F + |H|$$

Legyen az a és b skálafaktora L , $f(x)$ skálafaktora F .

Legyen tehát $\bar{a} = \frac{a}{L}$; $\bar{b} = \frac{b}{L}$; $\bar{f}(\bar{x}) = \frac{f(x)}{F}$; $\bar{x} = \frac{x}{L}$.

Ekkor az integrál gépi alakjára a következő becslést kapjuk:

$$\int_{\bar{a}}^{\bar{b}} \bar{f}(\bar{x}) d\bar{x} = \frac{1}{L} \int_a^b \frac{1}{F} f(x) dx = \frac{1}{LF} \int_a^b f(x) dx < \frac{1}{LF} (LF + |H|) = 1 + \frac{|H|}{LF}.$$

Az n növelésével elérhető, hogy $|H|$ tetszőleges kis szám legyen, L és F pedig megválasztható úgy, hogy az

$$\left| \int_{\bar{a}}^{\bar{b}} \bar{f}(\bar{x}) d\bar{x} \right| < 1$$

egyenlőtlenség is teljesüljön.

A fentiekből következik, hogy sem az integrál értéke, sem a közbenső eredmények nem csordulnak túl.

A szubrutin először $n=2m$ -re számolja a közelítő értéket, majd $n=4m$ -re. (Az m érték valamilyen egész szám, pl. $m=2$.) Ezután összehasonlítja a két közelítést, és ha ezek eltérése nagyobb valamilyen ε küszöbszámnál, akkor megkét-szerezi az osztópontok számát és az így kiszámított közelí-

tést összehasonlítja az előzővel. Ezt az eljárást addig folytatja, míg a két egymásutáni közelítés különbsége ε -nál kisebb nem lesz. Nulladik közelítésnek a gépi 1-est $(1-2^{-30})$ -at vesszük. ε -t a kívánt pontosságnak megfelelően kell megválasztani.

Legyen a szubrutin kezdőrekeszének címe: i . Az \bar{a} , illetve \bar{b} értékeket a főprogram viszi be az $i+43$ és $i+44$ rekeszekbe. Az $\frac{1}{m}$ és ε mennyiségeket az $i+47$ és $i+46$ -os rekeszekben kell elhelyezni. Példánkban legyen $\frac{1}{m}=\frac{1}{2}$ és $\varepsilon=4\cdot 2^{-30}$. A határozott integrál értéke az $i+45$ -ös rekeszben képződik. A visszaugrató utasítást az $i+34$ -es rekeszben kell elhelyezni.

A kvadratura szubrutin behívja az $f(x)$ függvényt előállító szubrutint. Legyen \bar{x} címe: $r+1$, $\overline{f(x)}$ címe: $r+2$. A szubrutin bemenete: $r+3$, a visszaugrató utasítás helye az r rekesz.

Program:

| | | | | |
|--------|----|----|--------|--------|
| i | -, | 11 | $i+43$ | $i+44$ |
| $i+1$ | ↓x | 23 | $i+47$ | $i+50$ |
| $i+2$ | ↓x | 23 | $i+36$ | $i+51$ |
| $i+3$ | Á | 05 | $i+37$ | $i+45$ |
| $i+4$ | Á | 05 | $i+43$ | $r+1$ |
| $i+5$ | Á | 05 | $i+45$ | $i+52$ |
| $i+6$ | Á | 05 | $i+53$ | $i+45$ |
| $i+7$ | x | 03 | $i+35$ | $i+50$ |
| $i+10$ | x | 03 | $i+35$ | $i+51$ |
| $i+11$ | +, | 10 | $i+40$ | $i+11$ |
| $i+12$ | U1 | 24 | $r+3$ | r |
| $i+13$ | x | 03 | $i+51$ | $r+2$ |
| $i+14$ | U2 | 74 | - | $i+21$ |
| $i+15$ | + | 00 | $i+41$ | $i+14$ |
| $i+16$ | U2 | 74 | - | $i+20$ |

| | | | | |
|------|----------|----|-------------------------|------|
| i+17 | : | 02 | i+35 | r+2 |
| i+20 | : | 02 | i+35 | r+2 |
| i+21 | + | 00 | r+2 | i+45 |
| i+22 | - | 01 | i+42 | i+14 |
| i+23 | + | 00 | i+50 | r+1 |
| i+24 | - , | 51 | r+1 | i+44 |
| i+25 | FU | 34 | i+26 | i+11 |
| i+26 | x | 03 | i+35 | r+2 |
| i+27 | - | 01 | r+1 | i+45 |
| i+30 | + | 00 | i+42 | i+14 |
| i+31 | - , | 51 | i+45 | i+52 |
| i+32 | ↓ - , | 71 | i+46 | - |
| i+33 | FU | 34 | i+34 | i+4 |
| i+34 | | | kiugrató utasítás helye | |

Konstansok:

| | | | | | | |
|------|---|----|------|------|---|-------------------|
| 1+53 | = | 00 | 0000 | 0000 | = | 0 |
| 1+55 | = | 40 | 0000 | 0000 | = | $\frac{1}{2}$ |
| i+36 | = | 25 | 2525 | 2525 | = | $\frac{1}{3}$ |
| 1+37 | = | 77 | 7777 | 7777 | = | $1-2^{-30}$ |
| i+40 | = | 64 | 0000 | 0002 | | |
| i+41 | = | 00 | 0000 | 0004 | = | $4 \cdot 2^{-30}$ |
| i+42 | = | 00 | 0000 | 0002 | = | $2 \cdot 2^{-30}$ |

i+45-től i+ 52-ig terjedő rekeszek a paraméterek helyei és a munkarekeszek.

Megjegyzés: Az ebben a fejezetben ismertetett szubrutinok nagy része nem a közölt formában található a szubrutin-könyvtárunkban. Az elemi függvények kiszámítására az itt ismertetett módszereknél általában jobbak is találhatóak, és a szubrutin-könyvtárban az általunk ismert legjobb módszer van beprogramozva relatív címekkel, hogy minden esetben, amikor szükséges, felhasználható legyen.

8. Fejezet.

AZ INPUT ÉS OUTPUT SZUBRUTINJA.

8.1. Input decimálisan adott számok esetén.

Az M-3 gép input szubrutinjai (amelyek közül az egyiket az előző fejezetekben ismertetett programokban többször behívtuk) a perforált szalagról binárisan kódolt decimális alakban leolvasnak egy vagy több számadatot (e-közben a szalagot megfelelően tovább mozgatják), és a leolvasott számadatokat bináris alakra konvertálva bizonyos előre meghatározott memóriarekeszekben helyezik el.

Tekintsük először azt - a gyakorlatban legtöbbször előforduló - esetet, amikor abszolút értékben egynél kisebb számadatokat fixpontos bináris alakban akarunk a gépben ábrázolni, és a fix bináris pontot megállapodás szerint a szó második bitje előtt helyezzük el (az első bit az előjel helye!).

A $t = \pm 0, \beta_1 \dots \beta_7$ hétjegyű tizedes tört a gépbe a következő alakban kerül be (lásd a bevitelről szóló részt, ill. a XV. tábla első ábráját!): minden egyes tizedes jegy négy bináris jeggyel van kódolva, és a szó utolsó két bitjében 0 áll.

A kettes számrendszerbe való átalakítást a megfelelő input a szubrutin a következő képlet alapján végzi el:

$$t = \pm \sum_{k=1}^7 \frac{\beta_k}{10^k} = \sum_{k=1}^7 \frac{a_k}{\left(\frac{10}{16}\right)^k}$$

$$= (((((((a_7 \frac{16}{10} + a_6) \frac{16}{10} + a_5) \frac{16}{10} + a_4) \frac{16}{10} + a_3) \frac{16}{10} + a_2) \frac{16}{10} + a_1) \frac{16}{10}$$

ahol

$$a_k = \pm \frac{\beta_k}{16^k};$$

az a_k számot a gépben a XV. tábla második ábráján látható szó ábrázolja.

Az alábbi szubrutin egyetlen számadatot olvas le, és azt binárisra konvertálva az $m+2$ című rekeszben helyezi el:

| | | | | | |
|-------|---|---|----|------|--------------------------|
| (k) | = | + | 00 | 0000 | 0074; |
| (k+1) | = | + | 50 | 0000 | 0000 = $\frac{10}{16}$; |
| (k+2) | = | + | 04 | 0000 | 0000 = 2^{-4} ; |
| (k+3) | = | + | 00 | 0000 | 0000 |

$m, m+$ munkarekeszek

| | | | | |
|------|-----|----|-----|------|
| a | Á | 05 | k+3 | m+2 |
| a+1 | Á | 05 | k | m+1 |
| a+2 | Be | 07 | - | m |
| a+3 | U2 | 74 | - | a+5 |
| a+4 | : | 02 | k+2 | m+1 |
| a+5 | Λ, | 16 | m+1 | m |
| a+6 | ↓+, | 30 | m+2 | - |
| a+7 | ↓: | 22 | k+1 | m+2 |
| a+10 | - , | 51 | k+2 | m+1 |
| a+11 | FU | 34 | a+4 | a+12 |
| a+12 | .. | .. | ... | |

Az ismertetett input szubrutinon kívül néha szükség lehet egy olyan input szubrutinra is, amelynek segítségével az eredetileg tízes számrendszerben megadott $t' = \pm \beta'_6 \dots \beta'_0$ hétjegyű egész számot fixpontos bináris alakra konvertálva

úgy helyezhetjük el a gépben, hogy a fix bináris pont a szó harmincegyedik, utolsó bitje után álljon, illetve - precízebben megfogalmazva - bináris alakban elhelyezhetjük a gépben a $t^* = t \cdot 2^{-30}$ számot. (Az M-3 gépen a fix bináris pont természetesen mindig a szó második bitje előtt van!)

A t szám a gépbe ugyanolyan alakban kerül be, mint az előző példában a t tizedes tört. A szubrutin a kettes számrendszerbe való konvertálást az alábbi identitás alapján végzi el:

$$\bar{t} = \pm \sum_{k=0}^6 \bar{\beta}_k 10^k = \sum_{k=0}^6 \bar{a}_k \left(\frac{10}{16}\right)^k =$$

$$= ((((((\bar{a}_6 \frac{10}{16} + \bar{a}_5) \frac{10}{16} + \bar{a}_4) \frac{10}{16} + \bar{a}_3) \frac{10}{16} + \bar{a}_2) \frac{10}{16} + \bar{a}_1) \frac{10}{16} + \bar{a}_0$$

ahol

$$\bar{t} = t \cdot 2^{-28}; \quad \bar{\beta}_k = \beta'_k \cdot 2^{-28}; \quad \bar{a}_k = \pm \beta_k \cdot 16^k;$$

a \bar{t} szám bináris alakjából $\frac{1}{4}$ -del való szorzással rögtön megkapjuk a t^* szám bináris alakját.

Az alábbi szubrutin egyetlen egész számadatot olvas le, és annak 2^{-30} -szorosát bináris alakra konvertálva az $m+2$ című rekeszben helyezi el. (A relatív címezés alapjait itt, és az ezután következő szubrutinoknál is ugyanazokkal a betűkkel jelöljük, mint az első input szubrutinnál.)

| | | | | | | |
|-------|---|---|----|------|------|-------------------|
| (k) | = | + | 74 | 0000 | 0074 | |
| (k+1) | = | + | 50 | 0000 | 0000 | = $\frac{10}{16}$ |
| (k+2) | = | + | 04 | 0000 | 0000 | = 2^{-4} |
| (k+3) | = | + | 00 | 0000 | 0000 | |
| (k+4) | = | + | 00 | 0000 | 0000 | = 2^{-28} |
| (k+5) | = | + | 20 | 0000 | 0000 | = $\frac{1}{4}$ |

$m, m+1$ munkarekeszek.

| | | | | |
|------|-------|----|------|-----|
| a | Á | 05 | k+3 | m+2 |
| a+1 | Á | 05 | k | m+1 |
| a+2 | Be | 07 | - | m |
| a+3 | x | 03 | k+1 | m+2 |
| a+4 | Λ, | 16 | m+1 | m |
| a+5 | ↓+ | 20 | m+2 | m+1 |
| a+6 | x | 03 | k+2 | m+2 |
| a+7 | ↓ - , | 71 | k+4 | - |
| a+10 | FU | 34 | a+11 | a+3 |
| a+11 | x | 03 | k+5 | m+2 |

A két ismertetett input szubrutin felhasználásával könnyen készíthetünk olyan input szubrutinokat, amelyek tízes rendszerben felírt számadatokat bináris alakra konvertálva úgy helyeznek el a gépben, hogy a fix bináris pont az i -edik bit mögött legyen ($1 \leq i \leq 30$). Könnyen össze lehet állítani olyan input szubrutinokat is, amelyek egymásután n számadatot visznek be binárisra konvertálva az $r, \dots, r+n-1$ című rekeszekbe; a két programparamétert, r -et és n -et, célszerűen egy $+ 00 n r$ alakú szóval adjuk meg (lásd a 7. fejezetben készített programokat). Ezeket a szubrutinokat itt most nem ismertetjük.

8.2. Output decimális szubrutinokhoz.

Az M-3 gép output szubrutinjai (amelyek közül az egyiket szintén többször behívtuk az előző fejezetekben ismertetett programokban) egy vagy több, az M-3 gépben bináris alakban tárolt számot decimális alakra konvertálva a távgépírón kinyomtatnak.

A gyakorlatban itt is az az eset a legfontosabb, amikor a számítás kinyomtatandó végeredménye vagy rész-eredménye fixpontos bináris alakban keletkezik, és a

fix bináris pont a természetes helyen, vagyis a szó második bitje előtt áll. A tízes számrendszerbe való konverzió ebben az esetben azon a könnyen igazolható tényen alapszik, hogy ha egy kettes számrendszerben felírt, abszolút értékben egynél kisebb számot megszorozunk $\frac{10}{16}$ -dal, a szorzatnak a bináris pont után következő négy bitjében binárisan kódolva megkapjuk a tizedespont után álló első decimális jegyet; az első négy bit helyére nullákat írva, és ismét $\frac{10}{16}$ -dal megszorozva, a szorzat második négy bitjében megkapjuk a tizedespont után álló második decimális jegyet, stb.

Az alábbi szubrutin egyetlen, az $m+2$ rekeszben bináris alakban tárolt számot nyomtat ki hét tizedes jegyre:

| | | | | | | |
|-------|---|---|----|------|------|-------------------|
| (k) | = | + | 74 | 0000 | 0000 | |
| (k+1) | = | + | 50 | 0000 | 0000 | = $\frac{10}{16}$ |
| (k+2) | = | + | 04 | 0000 | 0000 | = 2^{-4} |
| (k+3) | = | + | 00 | 0000 | 0000 | |
| (k+4) | = | + | 00 | 0000 | 0074 | |

m, m+1, m+3 munkarekeszek.

| | | | | |
|------|-------|----|------|-----|
| a | Á | 05 | k+3 | m+3 |
| a+1 | Á | 05 | k | m+1 |
| a+2 | x | 03 | k+1 | m+2 |
| a+3 | ↓∧ | 26 | m+1 | m |
| a+4 | ↓+ | 20 | m+3 | m+3 |
| a+5 | - | 01 | m | m+2 |
| a+6 | x | 03 | k+2 | m+2 |
| a+7 | ↓ - , | 71 | k+4 | - |
| a+10 | FU | 34 | a+11 | a+3 |
| a+11 | +n | 40 | k+3 | m+3 |

A konvertált szám (binárisan kódolt decimális alakban) megmarad az $m+3$ című rekeszben.

Bizonyos esetekben előfordulhat, hogy egy számítás végeredményei vagy részeredményei a gépben $t \cdot 2^{-30}$ alakúak, ahol t egy 10^7 -nél kisebb egész szám, és a gépnek a $t = \pm \beta'_6 \dots \beta'_0$ számot kell tízes számrendszerben kinyomtatni. Az ilyenkor használt konverziós eljárás alapelve a következő: ha

$$t^* = t \cdot 2^{-30}; \beta^*_k = \beta'_k \cdot 2^{-30}; a^*_k = \pm \beta^*_k 16^k,$$

akkor nyilván

$$\begin{aligned} t^* &= \pm \sum_{k=0}^6 \beta^*_k 10^k = \sum_{k=0}^6 a^*_k \left(\frac{10}{16}\right)^k = \\ &= a^*_6 \left(\frac{10}{16}\right)^6 + \dots + a^*_1 \left(\frac{10}{16}\right)^1 + a^*_0 \end{aligned}$$

és

$$t^* \frac{16}{10} = a^*_6 \left(\frac{10}{16}\right)^5 + \dots + a^*_1 + \frac{a^*_0}{10} \cdot 16$$

A jobboldalon álló első hat tag összege (jelöljük ezt az összeget t^{**} -gal) 2^{-26} -nál osztható, a hetedik tag pedig 2^{-26} -nál kisebb; a t^{**} számot tehát a $t^* \frac{16}{10}$ szorzatból egyszerűen úgy nyerjük, hogy az utolsó négy bit helyére nullákat írunk. A t^{**} ismeretében viszont a^*_0 -t azonnal megkaphatjuk:

$$a^*_0 = t^* - \frac{10}{16} t^{**}.$$

Az eljárást ugyanilyen módon tovább folytatva rendre előállíthatjuk az a^*_1, \dots, a^*_6 számokat; ezeknek összegét $\frac{1}{4}$ -del elosztva megkapjuk a szó első huszonnyolc bitjében a t szám binárisan kódolt decimális alakját.

Az alábbi output szubrutinban $t^* = (m+2)$; a t szám binárisan kódolt decimális alakban megmarad az $m+3$ című rekeszben:

| | | | | | | |
|-------|---|---|----|------|------|-------------------|
| (k) | = | + | 74 | 0000 | 0074 | |
| (k+1) | = | + | 50 | 0000 | 0000 | = $\frac{10}{16}$ |
| (k+2) | = | + | 04 | 0000 | 0000 | = 2^{-4} |
| (k+3) | = | + | 00 | 0000 | 0000 | |
| (k+4) | = | + | 00 | 0000 | 0000 | = 2^{-30} |
| (k+5) | = | + | 20 | 0000 | 0000 | = $\frac{1}{4}$ |
| (k+6) | = | + | 03 | 7777 | 7777 | |

m, m+1 munkarekeszek.

| | | | | |
|------|-------|----|------|-----|
| a | Á | 05 | k+3 | m+3 |
| a+1 | Á | 05 | k+6 | m |
| a+2 | : | 02 | k+2 | m |
| a+3 | :, | 12 | k+1 | m+2 |
| a+4 | ↓∧ | 26 | m | m+1 |
| a+5 | ↓x | 33 | k+1 | - |
| a+6 | ↓- | 21 | m+2 | m+2 |
| a+7 | - | 01 | m+2 | m+3 |
| a+10 | - | 01 | k | m |
| a+11 | Á | 05 | m+1 | m+2 |
| a+12 | ↓ - , | 71 | k+4 | - |
| a+13 | FU | 34 | a+14 | a+2 |
| a+14 | :n | 42 | k+5 | m+3 |

A két ismertetett output szubrutin felhasználásával könnyen összeállíthatunk olyan output szubrutinokat, amelyek abban az esetben alkalmazhatók, ha a fix bináris pontot gondolatban az i -edik bit mögött helyezzük el ($1 \leq i \leq 30$); továbbá olyan output szubrutinokat is, amelyek egymás után konvertálják és kinyomtatják az $r, r+1, \dots, r+n-1$ rekeszekben bináris alakban tárolt számokat (r és n programparaméterek).

8.3. Relatív címezésű szubrutinok bevitele.

A 7. fejezetben röviden foglalkoztunk a relatív címezésű szubrutinok bevitelének kérdésével: megemlítettük, hogy a szabványos alakban elkészített relatív címezésű szubrutinokat (az u.n. szabványos vagy könyvtári szubrutinokat) speciális programok, az u.n. beindítórutinok segítségével lehet a gépbe bevinni, és ismertettük az M-3 gép szubrutinkönyvtárának tervét. Ebben a paragrafusban az M-3 gép két különböző alapelvek szerint működő beindítórutinját, és a szabványos szubrutinoknak a beindítórutinok segítségével történő bevitelét fogjuk részletesebben tárgyalni.

Az első beindítórutin (amely lényegében az angol EDSAC gép beindítórutinjának elvét követi) a már ismertett módon perforált szabványos szubrutinok bevitelére alkalmas: a beviendő szubrutin utasításait és konstansait a memória megadott című rekeszeiben helyezi el, és közben a relatív címeket megfelelő módon átalakítja abszolút címekké.

A beindítórutin használata esetén minden egyes szabványos szubrutin bevitele előtt "közölni kell a géppel" a relatív címezés alapjainak számértékeit. Ez a következő módon történik meg: egy külön szalagra felvisszünk először egy

```
+ 77 7777 7777
+ 20 0115 0041
```

alakú u.n. vezérlő kombinációt; ennek hatására a beindítórutin a szalagról leolvasott további szavakat a 0041, 0042, ... rekeszekben fogja elhelyezni. A szalagra ezután ábécé sorrendben felvisszük a relatív címezés alapjának számértékeit: minden egyes értéket külön szóban helyezünk el a szó 7-18 bitjében (vagyis az első cím helyén), a szó

többi bitje zérus lesz. A szó végét jelző "írás" jel után négy zérust, majd ismét egy "írás" jelet perforálunk.

A relatív címezés alapjainak számértékei után a szalagra egy

+ 77 7777 7777

+ 20 0115 m

alakú újabb "vezérlőkombinációt" viszünk fel, ahol m a szabványos szubrutin első utasításának címe. Ennek hatására a beindítórutin a szabványos szubrutin utasításait és konstansait az m című rekesztől kezdve fogja elhelyezni. Ezt követi egy

+ 77 7777 7777

- 77 0000 0000

alakú vezérlőkombináció, amelynek leolvasása után a beindítórutin megállítja a gépet. A szabványos szubrutin perforált szalagjának behelyezése után a gépet újraindítjuk. Ekkor a beindítórutin sorra leolvassa a szalagról a szabványos szubrutin utasításait (konstansait); minden egyes utasításban (konstansban) a relatív címeket átalakítja abszolút címekké, majd az utasítást (konstansot) elhelyezi az $m+r$ című rekeszben, ahol r az illető utasítás (konstans) relatív címét jelöli, a szabványos szubrutinok végére szintén egy

+ 77 7777 7777

- 77 0000 0000

alakú vezérlőkombinációt kell rávinni, ennek hatására a beindítórutin a szubrutin bevitele után megállítja a gépet.

A beindítórutinál nem lehet a gépbe vinni a $\pm 77 7777 7777$ konstansokat (ezeket egyébként is célszerű szabványos konstansokként állandóan a gépben tárolni).

A beindítórutin a következő:

| | | | | | |
|--------|---|---|----|------|------|
| (0040) | = | + | 00 | 0000 | 0000 |
| (0104) | = | + | 00 | 0000 | 0077 |
| (0105) | = | + | 00 | 0000 | 7700 |
| (0106) | = | + | 13 | 0112 | 0040 |
| (0107) | = | + | 30 | 0040 | 0000 |
| (0111) | = | + | 77 | 7777 | 7777 |
| (0112) | = | + | 00 | 0100 | 0000 |
| (0113) | = | + | 01 | 0000 | 0000 |
| (0114) | = | + | 00 | 0000 | 0001 |

0115 munkarekesz

| | | | | |
|------|------|----|------|------|
| 0062 | Be | 07 | -- | 0115 |
| 0063 | Be | 07 | -- | 0071 |
| 0064 | - , | 51 | 0111 | 0115 |
| 0065 | FU | 34 | 0072 | 0066 |
| 0066 | Á | 05 | 0071 | 0071 |
| 0067 | FU | 34 | 0071 | 0070 |
| 0070 | FU | 34 | 0062 | 0101 |
| 0071 | (| | |) |
| 0072 | Λ, | 16 | 0104 | 0071 |
| 0073 | ↓+ | 20 | 0106 | 0077 |
| 0074 | Λ, | 16 | 0105 | 0071 |
| 0075 | ↓: | 32 | 0113 | -- |
| 0076 | ↓+ | 20 | 0107 | 0100 |
| 0077 | (| | |) |
| 0100 | (| | |) |
| 0101 | (| | |) |
| 0102 | + | 00 | 0114 | 0101 |
| 0103 | U2 | 74 | -- | 0062 |

A felírt beindítórutin elvben, minden relatív címezésű szubrutin bevitelére alkalmas; a relatív címezés alapjainak számát csupán memória-takarékossági okokból maximál-

tuk 15-re. Hátránya azonban, hogy minden egyes címnél külön fel kell tüntetni a relatív címezés alapját, ami megnehezíti a perforálást, és meghosszabbítja a bevétel idejét. A továbbiakban ismertetni fogunk egy másik beindítórutint, amely csak olyan címezésű szubrutinok bevételére alkalmas, ahol a relatív címeknek egy közös alapjuk van: a szubrutin első utasításának címe. Ez a rutin azonban az abszolút címekkel való bevételhez képest nem hosszabbítja meg lényegesen a bevétel idejét. Az alábbiakban közölt beindítórutin, amelyet némileg egyszerűsített formában fogunk felírni, Pekingben készült az ottani M-3 gépre.

A beindítórutin azon a tényen alapszik, hogy míg a jelenlegi M-3 gépen egy memóriarekesz címe maximálisan 4000 lehet (nyolcas rendszerben), a szavak első, illetve második címrészének hosszúsága 12-12 bit, tehát minden utasítást ábrázoló szóban címrészenként egy-egy bit (t.i. a hetedik, illetve tizenkilencedik bit) feltétlenül zérus.

A relatív címeket az abszolút címektől úgy különböztetjük meg, hogy ha egy utasítás első címe relatív, akkor a hetedik bit, ha a második cím relatív, akkor a tizenkilencedik bit helyére 1-et írunk, vagyis minden relatív címet 4000-el növelünk. Ha a beindítórutin egy utasításban 4000-nél nagyobb címét talál, akkor "tudni fogja", hogy az illető cím relatív, és 4000-m levonásával (m-mel jelöljük annak a memóriarekesznek a címét, amelybe a szubrutin első utasítását akarjuk bevinni) kialakítja a megfelelő abszolút címet. Mindez természetesen csak az utasításokra és az utasításjellegű konstansokra vonatkozik, a számkonstansokat, amelyeket a beindítórutin használata esetén mindig az utasítások és utasításjellegű konstansok címei után következő címekre viszünk be, a beindítórutin minden változtatás nélkül elhelyezi a megfelelő munkarekeszekben.

A szabványos szubrutinok szalagjának elejére mindig egy + 00 p d alakú szót kell lyukasztani, ahol d a szabványos szubrutin utasításainak és konstansainak együttes számát, p az utasítások és utasítás-jellegű konstansok együttes számát jelöli.

A beindítórutin használata igen egyszerű: a 0101 című rekeszbe beírunk egy + 00 0000 m alakú szót, ahol m annak a rekesznek a címe, amelybe a szubrutin első utasítását akarjuk bevinni, a gépadóban elhelyezzük a szubrutin szalagját, majd a 0040 című utasítással elkezdjük a beindítórutin végrehajtását.

A beindítórutin a következő:

(0070) = + 00 0000 7777; (0071) = + 20 0102 0000;
(0072) = + 00 7777 0000; (0073) = + 11 0102 0000;
(0074) = + 00 0100 0000 = 2^{-12} ; (0075) = + 00 0000 0001;
(0076) = + 00 0000 4000 = 2^{-19} ; (0077) = + 00 4000 0000;
0100, 0102, 0103 munkarekeszek.

| | | | | |
|------|-----|----|------|------|
| 0040 | Be | 07 | - | 0100 |
| 0041 | + | 10 | 0071 | 0101 |
| 0042 | U1 | 24 | 0043 | 0057 |
| 0043 | ^ , | 56 | 0070 | 0100 |
| 0044 | ↓+, | 30 | 0057 | -- |
| 0045 | ↓- | 21 | 0073 | 0103 |
| 0046 | ^ , | 56 | 0072 | 0100 |
| 0047 | ↓x, | 33 | 0074 | -- |
| 0050 | ↓+ | 20 | 0057 | 0100 |
| 0051 | ↓- | 21 | 0073 | 0063 |
| 0052 | - , | 51 | 0076 | 0101 |
| 0053 | ↓: | 22 | 0076 | 0101 |
| 0054 | Be | 07 | - | 0102 |
| 0055 | ^, | 16 | 0077 | 0102 |

| | | | | |
|------|-----|----|------|------|
| 0056 | ↓x, | 33 | 0101 | -- |
| 0057 | (| | |) |
| 0060 | + | 00 | 0075 | 0057 |
| 0061 | ↓-, | 31 | 0100 | -- |
| 0062 | FU | 34 | 0054 | 0063 |
| 0063 | (| | |) |
| 0064 | + | 00 | 0075 | 0063 |
| 0065 | ↓-, | 31 | 0103 | -- |
| 0066 | | | | 0057 |
| 0067 | | | 0000 | 0000 |

9. Fejezet.

ÉRTELMEZŐ ÉS KONVERZIÓS SZUBRUTINOK LOGIKAI (TECHNIKAI) LEHETŐSÉGEK PÓTLÁSÁRA.

Az eddigiekben nem adtunk teljes választ arra, hogyan járhatunk el akkor, ha a transzformációs módszer nem alkalmazható a fixponttal kapcsolatos nehézség megkerülésére. Nem beszéltünk eddig arról sem, hogyan lehet az M-3 gépen komplex számokkal számolni.

Említettük azonban, hogy a transzformációs módszereken kívül alkalmazhatjuk még a lebegőpontos programozás módszerét, s arról is szó volt, hogy ez lényegében nem más, mint egy lebegőpontos gép működésének egy program segítségével történő utánozása. Ugyanis a komplex számokkal való műveletek elvégzésére is elkészíthetünk egy programot, mondhatnók úgy is, hogy beprogramozzuk egy olyan feltételezett gép működését, amely komplex számokkal is számol.

Ezeket a programokat célszerű szubrutinokként használni, segítségükkel elérhető, hogy a fixpontos M-3 géppel elvégezhetőek legyenek olyan műveletek is, amelyeket a szerkezetileg bonyolultabb lebegőpontos (illetve a feltételezett komplex) számokkal dolgozó gépek végeznek el. Ebben az értelemben tehát logikai (technikai) lehetőségek programok segítségével történő pótlásáról van szó.

9.1. Lebegőpontos számolás az M-3 képen.

Ha egy lebegőpontos gép működését akarjuk utánozni, lényegében a gép két egységének, az aritmetikai egységnek és a vezérlőegységnek a működését kell programozni. Nevezük az aritmetikai egységnek megfelelő programot lebegőpontos szubrutinnak, a vezérlő egységnek megfelelőt pedig értelmező (interpretáló) szubrutinnak. A lebegőpontos szubrutin az alapműveleteket végzi el, lebegőpontos normalizált számokkal.

Emlékeztetünk arra, hogy egy x szán kettes számrendszerbeli lebegőpontos (féllogaritmikus) normalizált alakján az

$$x = d \cdot 2^p \text{ alakot értjük,}$$

ahol

$$d = \pm (\alpha_1 2^{-1} + \dots + \alpha_{30} 2^{-30}) \text{ a mantissza}$$

$$p \cdot 2^{-30} = \pm (\beta_1 2^{-1} + \dots + \beta_{30} 2^{-30}), \text{ a mantissza } 2^{-30} \text{ szorosa}$$

és

$$\alpha_1 = 1, \text{ azaz } \frac{1}{2} \leq d < 1. \text{ a kitevő } 2^{-30} \text{ szorosa}$$

$$\text{Ha } \alpha_1 = 0, \text{ akkor } x = 0, \text{ (gépi 0)}$$

Tároljuk az így felírt számot a memória két rekeszében; megállapodásszerűen a mantisszát egy páros, a kitevőt (2^{-30} -as léptékfaktorral megszorozva) a rákövetkező páratlan című rekeszben, s jelöljük ezt az $(a; a+1) = x$ alakkal, ami tehát azt jelenti, hogy $(a) = d$, $(a+1) = p \cdot 2^{-30}$, ha $x = d \cdot 2^p$.

Jelöljük ki a memóriában a P , $P+1$, Q , $Q+1$ rekeszeket. Ezek töltsék be a regiszterek szerepét, mégpedig úgy, hogy

Q; Q+1 feleljen meg az A regiszternek, P; P+1 pedig a B regiszternek (azaz minden művelet eredménye maradjon meg (P; P+1-ben)).

A lebegőpontos szubrutin a következő részekből áll:

- a.) normalizáló szubrutin,
- b.) lebegőpontos szorzás szubrutinja,
- c.) " " osztás " " ,
- d.) „ „ összeadás, kivonás szubrutinja.

Minden művelet után átadjuk a vezérlést a normalizáló szubrutinnak.

a.) Normalizáló szubrutin

Ha két normalizált számmal végzünk valamilyen aritmetikai műveletet, az eredmény általában nem lesz normalizált. El kell készíteni tehát egy normalizáló szubrutint, azaz egy olyan szubrutint, amely egy $x = d \cdot 2^p$; $d < \frac{1}{2}$ számot olyan alakra hoz, hogy $\frac{1}{2} \leq d < 1$ legyen.

A normalizáló szubrutint úgy készítjük el, hogy mindig a "B regiszter" tartalmát, tehát (P; P+1)-t normalizálja.

A szubrutin blokkdiagramját a XVI. táblán láthatjuk.

Program:

| | | | | |
|-----|------|----|-------------------------------|-----|
| t | - , | 51 | $\langle 2^{-30} \rangle$ | P |
| t+1 | FU | 34 | t+7 | t+2 |
| t+2 | - , | 51 | $\langle \frac{1}{2} \rangle$ | P |
| t+3 | FU | 34 | t+4 | t+7 |
| t+4 | - | 01 | $\langle 2^{-30} \rangle$ | P+1 |
| t+5 | : | 02 | $\langle \frac{1}{2} \rangle$ | P |
| t+6 | U2 | 74 | - | t+2 |
| t+7 | | | (kiugrató utasítás helye) | |

b.) A lebegőpontos szorzás ($z = xy$).

Ha $(P; P+1) = x = a \cdot 2^p$, és $(Q; Q+1) = y = b \cdot 2^q$, akkor
 $z = xy = a \cdot 2^p \cdot b \cdot 2^q = ab \cdot 2^{p+q} = c \cdot 2^r \longrightarrow (P; P+1)$
 ahol $c = ab$, $r = p+q$.

Tehát a szorzást úgy végezzük el, hogy a mantisszákat összeszorozzuk, s a kitevőket összeadjuk:

$$\begin{array}{l} (P) \cdot (Q) \longrightarrow P \\ (P+1) + (Q+1) \longrightarrow P+1 \end{array}$$

Program:

| | | | | | |
|------|----|----|-----|-----|--------------|
| t+10 | + | 00 | Q+1 | P+1 | ← p+q |
| t+11 | x | 03 | Q | P | ← ab |
| t+12 | U2 | 74 | - | t | normalizálás |

c.) A lebegőpontos osztás ($z = \frac{x}{y}$)

Ha $(P; P+1) = x = a \cdot 2^p$ és $(Q; Q+1) = y = b \cdot 2^q$, akkor
 $z = \frac{x}{y} = \frac{a \cdot 2^p}{b \cdot 2^q} = \frac{a}{b} \cdot 2^{p-q} = c \cdot 2^r \longrightarrow (P; P+1)$
 ahol $c = \frac{a}{b}$, $r = p-q$

Mint hogy $\frac{1}{2} \leq a < 1$, és $\frac{1}{2} \leq b < 1$, ezért előfordulhat, hogy

$$\left| \frac{a}{b} \right| > 1$$

Ha viszont $x = a \cdot 2^{p-t}$ $x = \frac{a}{2} \cdot 2^{p+1} = a' \cdot 2^{p+1}$ alakra hozzuk, akkor nyilván

$$\left| \frac{a'}{b} \right| < 1$$

Tehát az osztást

$$z = \frac{x}{y} = \frac{a'}{b} \cdot 2^{p+1-q} \longrightarrow (P; P+1)$$

alapján végezzük el, azaz

$$\frac{\frac{1}{2} (P)}{(Q)} \longrightarrow P; \quad (P+1) + 1 - (Q+1) \longrightarrow P+1$$

Program:

| | | | | |
|------|----|----|---------------------|----------------|
| t+13 | -, | 11 | Q+1 | P+1 |
| t+14 | ↓+ | 20 | <2 ⁻³⁰ > | P+1 |
| t+15 | X, | 13 | < $\frac{1}{2}$ > | P |
| t+16 | ↓: | 22 | Q | P |
| t+17 | U2 | 74 | - | t normalizálás |

d.) Lebegőpontos összeadás és kivonás.

Legyen $(P; P+1) = x = a \cdot 2^p$ és $(Q; Q+1) = y = b \cdot 2^q$. Ekkor az összeadást (illetve a kivonást) a

$$z = x \pm y = a \cdot 2^p \pm b \cdot 2^q = c \cdot 2^r \longrightarrow (P; P+1)$$

képlet alapján célszerű elvégezni, ahol $r = \max(p, q) + 1$ és

$$c = \begin{cases} \frac{a}{2} \pm \frac{b'}{2} & \text{ha } p > q, \text{ ahol } b' = b \cdot 2^{-(p-q)} \\ \frac{a'}{2} \pm \frac{b}{2} & \text{ha } p < q, \text{ ahol } a' = a \cdot 2^{-(q-p)} \end{cases}$$

A mantisszák felezésére, s ennek megfelelően a közös kitevő eggyel való növelésére a túlcsondulás elkerülése végett van szükség, hiszen, $\frac{1}{2} \leq a < 1$ és $\frac{1}{2} \leq b < 1$ miatt $|a + b| > 1$ is előfordulhat.

Tehát az összeadást (illetve a kivonást) úgy végezzük el, hogy a két számot az $r = \max(q;p) + 1$ közös kitevőre hozzuk, majd a felezett mantisszákat összeadjuk, (illetve kivonjuk) s az eredmény kitevője a közös kitevő lesz.

Ha valamelyik összeadandó 0 (azaz mantisszája gépi 0), akkor az összeadást nem végezzük el, az eredmény a

0-tól különböző összeadandó.

Ha $p-q > 30$, akkor a nagyobb kitevőjű számot tekintjük eredménynek.

A kivonást úgy végezzük el, hogy a kivonandó negatívját adjuk hozzá a kisebbítendőhöz.

A blokkdiagram a XVII. táblán látható.

Program:

| | | | | | |
|------|------|----|------------------------------------|------|-----------------|
| t+20 | x | 03 | $\langle 2^{-30}-1 \rangle$ | Q | |
| t+21 | - , | 51 | $\langle 2^{-30} \rangle$ | Q | |
| t+22 | FU | 34 | t+7 | t+23 | |
| t+23 | - , | 51 | $\langle 2^{-30} \rangle$ | P | |
| t+24 | FU | 34 | t+25 | t+30 | |
| t+25 | Á | 05 | Q | P | |
| t+26 | Á | 05 | Q+1 | P+1 | |
| t+27 | U2 | 74 | - | t+7 | |
| t+30 | - | 01 | P+1 | Q+1 | |
| t+31 | FU | 34 | t+36 | t+32 | |
| t+32 | ↓+ | 20 | P+1 | P+1 | |
| t+33 | Á | 05 | P | m | (m munkarekesz) |
| t+34 | Á | 05 | Q | P | |
| t+35 | Á | 05 | m | Q | |
| t+36 | - , | 51 | $\langle 30 \cdot 2^{-30} \rangle$ | Q+1 | |
| t+37 | FU | 34 | t+40 | t+7 | |
| t+40 | ↓+ | 20 | $\langle 30 \cdot 2^{-30} \rangle$ | Q+1 | |
| t+41 | x | 03 | $\langle \frac{1}{2} \rangle$ | Q | |
| t+42 | - | 01 | $\langle 2^{-30} \rangle$ | Q+1 | |

| | | | | |
|------|----|----|-------------------------------|------|
| t+43 | FU | 34 | t+44 | t+41 |
| t+44 | + | 00 | $\langle 2^{-30} \rangle$ | P+1 |
| t+45 | x | 03 | $\langle \frac{1}{2} \rangle$ | P |
| t+46 | ↓+ | 20 | Q | P |
| t+47 | U2 | 74 | - | t |

Egy lebegőpontos művelet ezekután a következőképpen végezhető el:

A műveletben szereplő két számot elhelyezzük a két kettős "regiszterben"; a normalizáló szubrutin kiugrató utasításainak a helyére beírjuk a visszaugrató utasítást, majd átadjuk a vezérlést a lebegőpontos szubrutin megfelelő címére. Ha az eredményt meg kell őriznünk, akkor (P; P+1)-et átviszük abba a két rekeszbe, amelyben az eredményt tárolni akarjuk. Ezek a lépések fixpontos utasításokkal programozhatók.

Az előbbieknek megfelelően programozott lebegőpontos műveletekből álló programot lebegőpontos programnak nevezzük.

Lássunk egy példát: kiszámítandó az

$$y = \frac{\frac{a}{b} + cx}{dx + a}$$

mennyiség. Legyen a rekeszelosztás a következő:

| | |
|--------------|------------------|
| (r; r+1) | = a |
| (r+2; r+3) | = b |
| (r+4; r+5) | = c |
| (r+6; r+7) | = x |
| (r+10; r+11) | = d |
| (r+02) | = + 64 0000 0002 |

Legyen a normalizáló szubrutin n-től n+7-ig elhelyezve. Kezdődjön az összeadás programja az ö utasítással, a kivonásé k-val, az osztásé o-val, a szorzásé sz-szel.

Program:

| | | | | | | |
|------|----|----|------|------|---|---|
| t | x, | 13 | r+6 | r+10 | } | Elvégezzük a dx szorzást és normalizálunk |
| t+1 | U1 | 24 | t+2 | P | | |
| t+2 | +, | 10 | r+7 | r+11 | | |
| t+3 | U1 | 24 | t+4 | P | | |
| t+4 | +, | 10 | r+12 | t+4 | | |
| t+5 | U1 | 24 | n | n+7 | } | dx + a képzése |
| t+6 | Á | 05 | r | Q | | |
| t+7 | Á | 05 | r+1 | Q+1 | | |
| t+10 | +, | 10 | r+12 | t+10 | | |
| t+11 | U1 | 24 | ö | n+7 | | |
| t+12 | Á | 05 | P | m | } | átviszi dx+a-t az m, m+1 re- keszekbe |
| t+13 | Á | 05 | P+1 | m+1 | | |
| t+14 | Á | 05 | r | P | } | $\frac{a}{b}$ képzése |
| t+15 | Á | 05 | r+1 | P+1 | | |
| t+16 | Á | 05 | r+2 | Q | | |
| t+17 | Á | 05 | r+3 | Q+1 | | |
| t+20 | +, | 10 | r+12 | t+20 | | |
| t+21 | U1 | 24 | o | n+7 | } | $\frac{a}{b} \implies m+2, m+3$ |
| t+22 | Á | 05 | P | m+2 | | |
| t+23 | Á | 05 | P+1 | m+3 | | |
| t+24 | x, | 13 | r+4 | r+6 | } | cx képzése |
| t+25 | U1 | 24 | t+26 | P | | |
| t+26 | +, | 10 | r+5 | r+7 | | |
| t+27 | U1 | 24 | t+30 | P+1 | | |
| t+30 | +, | 10 | r+12 | t+30 | | |
| t+31 | U1 | 24 | n | n+7 | } | cx + $\frac{a}{b}$ képzése |
| t+32 | Á | 05 | m+2 | Q | | |
| t+33 | Á | 05 | m+3 | Q+1 | | |
| t+34 | +, | 10 | r+12 | t+34 | | |
| t+35 | U1 | 24 | ö | n+7 | | |

| | | | | | |
|------|-------------|----|------|------|---|
| t+36 | Á | 05 | m | Q | } $\frac{ax + \frac{c}{b}}{dx + a}$ képzése |
| t+37 | Á | 05 | m+1 | Q+1 | |
| t+40 | +, | 10 | r+12 | t+40 | |
| t+41 | U1 | 24 | o | n+7 | |
| t+42 | (megállás) | | | | |

Látható, hogy a lebegőpontos alakban megadott számokkal végzett aritmetikai műveletek végrehajtási ideje lényegesen nagyobb a fixpontos műveletekénél. A kiinduló adatok tárolására szolgáló rekeszek, és a munkarekeszek száma kétszer akkora, mint a fixpontos számításnál, ehhez jönnek még azok a rekeszek, amelyeket a műveleti előírásoknak megfelelő lebegőpontos szubrutinok kötnek le.

9.2. A komplex aritmetika programozása.

Az alábbiakban olyan szubrutint készítünk, amely komplex számokon végez aritmetikai műveleteket.

A komplex számok tárolására a lebegőpontos számok esetéhez hasonlóan két rekeszre van szükség. Megállapodás-szerűen a komplex számok valós részét egy páros, képzetes részét a rákövetkező páratlan című rekeszben tároljuk.

Jelöljük ki itt is a (P; P+1), (Q; Q+1) rekeszeket, B és A regiszterként.

Természetesen a túlcsordulásra továbbra is ügyelni kell; a következőkben feltesszük, hogy a szereplő adatok abszolút értékben egynél kisebbek, (Itt is alkalmazható természetesen mind a transzformációs módszer, mind a lebegőpontos programozás.)

Beprogramozzuk komplex számok

- a) összeadását,
- b) kivonását,
- c) szorzását,
- d) osztását.

a) Összeadás.

Legyen $(P; P+1) = z_1 = a_1 + b_1i$

$(Q; Q+1) = z_2 = a_2 + b_2i$

és

$$z_1 + z_2 = (a_1 + a_2) + (b_1 + b_2)i \longrightarrow P; P+1$$

Ha

$$|a_1| < \frac{1}{2}; |a_2| < \frac{1}{2}; |b_1| < \frac{1}{2}; |b_2| < \frac{1}{2}, \text{ akkor}$$

$$|a_1 + a_2| < 1, \quad |b_1 + b_2| < 1.$$

Program:

| | | | | |
|-----|----|----|-----|------|
| t | + | 00 | Q | P |
| t+1 | + | 00 | q+1 | P+1 |
| t+2 | U1 | 74 | - | t+33 |

b) Kivonás.

Legyen $(P; P+1) = z_1 = a_1 + b_1i$

$(Q; Q+1) = z_2 = a_2 + b_2i$

és

$$z_1 - z_2 = (a_1 - a_2) + (b_1 - b_2)i \longrightarrow P; P+1$$

Tegyük fel, hogy a valós és képzetes részekre az összeadásnál tett kikötések teljesülnek.

Program:

| | | | | |
|-----|----|----|-----|------|
| t+3 | - | 01 | Q | P |
| t+4 | - | 00 | q+1 | P+1 |
| t+2 | U1 | 74 | - | t+33 |

c) Szorzás.

$$\text{Legyen } (P; P+1) = z_1 = a_1 + b_1 i$$

$$(Q; Q+1) = z_2 = a_2 + b_2 i$$

és

$$\begin{aligned} z_1 z_2 &= (a_1 + b_1 i) \cdot (a_2 + b_2 i) + (a_1 \cdot b_2 + a_2 \cdot b_1) i = \\ &= (a_1 \cdot a_2 - b_1 \cdot b_2) + (a_1 \cdot b_2 + a_2 \cdot b_1) i = \\ &= a + b i \longrightarrow (P; P+1) \end{aligned}$$

Ha $|z_1| < 1$ és $|z_2| < 1$, akkor

$$|a_1 \cdot a_2 - b_1 \cdot b_2| < 1 \text{ és } |a_1 \cdot b_2 + a_2 \cdot b_1| < 1.$$

Ugyanis (a számok trigonometrikus alakjával számolva)

$$z_1 z_2 = |z_1| |z_2| (\cos (\varphi_1 + \varphi_2) + i \sin (\varphi_1 + \varphi_2))$$

tehát

$$|a_1 a_2 - b_1 b_2| = ||z_1| |z_2| \cos (\varphi_1 + \varphi_2)| < 1,$$

$$|a_1 b_2 + a_2 b_1| = ||z_1| |z_2| \sin (\varphi_1 + \varphi_2)| < 1.$$

Program:

| | | | | | |
|------|----|----|------|------|-----------------|
| t+6 | x, | 13 | P+1 | Q+1 | |
| t+7 | U1 | 24 | t+10 | m | (m munkarekesz) |
| t+10 | x, | 13 | P | Q | |
| t+11 | ↓- | 21 | m | m | |
| t+12 | x | 03 | Q+1 | P | |
| t+13 | x, | 13 | Q | P+1 | |
| t+14 | ↓+ | 20 | P | P+1 | |
| t+15 | Á | 05 | m | P | |
| t+16 | U2 | 74 | - | t+35 | |

d) Osztás.

$$\text{Legyen } (P; P+1) = z_1 = a_1 + b_1 i$$

$$(Q; Q+1) = z_2 = a_2 + b_2 i$$

és

$$\begin{aligned} \frac{z_1}{z_2} &= \frac{a_1 + b_1 i}{a_2 + b_2 i} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + \frac{a_2 b_1 + a_1 b_2}{a_2^2 + b_2^2} \cdot i = \\ &= a + b i \longrightarrow (P; P+1) \end{aligned}$$

Ha $|z_1| < |z_2| < 1$, akkor nyilván $|a| < 1$, $|b| < 1$.

Ugyanis (a számok trigonometrikus alakjával számolva)

$$\frac{z_1}{z_2} = \frac{|z_1|}{|z_2|} (\cos(\varphi_1 - \varphi_2) + i \sin(\varphi_1 - \varphi_2))$$

tehát

$$\left| \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} \right| = \left| \frac{|z_1|}{|z_2|} \cdot \cos(\varphi_1 - \varphi_2) \right| < 1$$

$$\left| \frac{a_2 b_1 + a_1 b_2}{a_2^2 + b_2^2} \right| = \left| \frac{|z_1|}{|z_2|} \cdot \sin(\varphi_1 - \varphi_2) \right| < 1$$

Program:

| | | | | |
|------|----|----|-----|-----|
| t+17 | Á | 05 | P | m |
| t+20 | ↓x | 23 | Q | P |
| t+21 | x, | 13 | P+1 | Q+1 |
| t+22 | ↓+ | 20 | P | P |
| t+23 | x | 03 | Q+1 | m |
| t+24 | x, | 13 | P+1 | Q |
| t+25 | ↓- | 21 | m | P+1 |
| t+26 | x | 03 | Q | Q |
| t+27 | x, | 13 | Q+1 | Q+1 |

| | | | | |
|------|---------------------------|----|-----|-----|
| t+30 | ↓+ | 20 | Q | Q+1 |
| t+31 | : | 02 | Q+1 | P |
| t+32 | : | 02 | Q+1 | P+1 |
| t+33 | (kiugrató utasítás helye) | | | |

Ha komplex számokkal végzünk el valamilyen műveletet, akkor ugyanúgy kell eljárni, mint lebegőpontos számolás esetén: a két számot el kell helyezni a két kettős regiszterben, s a visszaugrató utasítás elhelyezése után átadni a vezérlést a megfelelő műveletet végző szubrutinra. Természetesen abszolútértékkel történő műveleteknél az adott komplex számok abszolút értékeit külön programmal képezni kell. Valós számok esetén a képzetes rész helyére 0-t kell írni.

9.3. Értelmező szubrutin – lebegőpontos programhoz.

Az egyes problémák megoldását reprezentáló program egészen más felépítésű, ha a feladatban szereplő adatokat lebegőpontos alakban ábrázoljuk, mint a fixpontos ábrázolásnak megfelelő program. Az előbbi mindig lényegesen több utasításból áll. Nagyon célszerűnek látszik tehát egy olyan szubrutin szerkesztése, amely egy "fixpontos" program utasításait megfelelően transzformálja, és így lebegőpontos programmá alakítja. (Ugyanez a feladat komplex számokkal kapcsolatban is felmerülhet.)

Egy olyan szubrutinra van tehát szükség, amely az eredeti program utasításait "értelmezi", azaz a számokat, amelyekkel a műveletet el kell végeznünk, a lebegőpontos szubrutin két kettős rekeszébe helyezi el alkalmasan transzformálva őket, majd behívja az értelmezett műveleti utasításnak megfelelő lebegőpontos művelet szubrutinját, végül elhelyezi az eredményt. A szubrutinnak tehát egy lebegőpontos gép vezérlőegységét kell imitálnia.

Ismeretes, hogy egy adott utasítás lényegében kódolt algoritmus (elemi műveletek sorozata) alakjában táplál be információt a gépbe; ennek alapján hajtják végre a szerkezeti elemek a szükséges műveleteket a már ismertetett 8-as taktusokban. Egy utasításhoz, mint információhoz hozzárendelhető azonban egy másképp programozott algoritmus is. Az értelmező szubrutinok valamely utasítás alakú kódolt információhoz egy meghatározott algoritmust (műveletsort) rendelnek hozzá, s ezt az algoritmust hajtják végre.

Szerkesszünk meg egy olyan értelmező szubrutint, amely valamely fixpontos utasításhoz a megfelelő - ekvivalens - lebegőpontos algoritmust rendeli hozzá, s azt végrehajtja.

Egy adott fixpontos program utasításai a következő három csoportba sorolhatók:

1. Vannak olyan utasítások, amelyek adatokon végeznek műveleteket; nevezzük ezeket adat-utasításoknak.
2. Nevezzük cím-utasításoknak azokat, amelyek utasításokkal végeznek műveleteket (pl.: címmódosítás, utasításbeültetés, stb.) és ezeken kívül a megállási és kinyomtató utasításokat.
3. végül vannak vezérlés- és adatátadó utasítások is.

Lássuk el - megállapodásszerűen - a címutasításokat negatív előjellel!

Fentiek szerint egy adott program utasításai a következőképpen különböztethetők meg:

1. Ha egy utasítás negatív előjelű, akkor címutasításról van szó, tehát változatlanul - fixpontosan - hajtandó végre. Állapodjunk meg abban, hogy a logikai szorzást csak címutasításként használjuk.

2. Ha az utasítás pozitív előjelű, és műveleti kódjának második jegye 0,1,2,3, akkor adat-utasítás.
3. Ha az utasítás pozitív előjelű, és műveleti kódjának második jegye 4 vagy 5, akkor vezérlés- vagy adatátadásról van szó.

Megállapodunk abban is, hogy az egyes fixpontos adat-utasításokba a műveletekben szereplő lebegőpontos alakban adott számok mantisszáinak címét kell beírni. Egy-egy ilyen fixpontos utasítás tehát: $m a' b'$ alakú, ahol m a műveleti kód, a' az egyik, b' a másik mantissza címe.

Ha egy eredetileg fixpontos alakban használt programot akarunk értelmezni, akkor az értelmezés előtt át kell ezt úgy alakítani, hogy az eredetileg fixpontos alakban tárolt adatok címei helyett a nekik megfelelő lebegőpontos címek (mégpedig a mantisszák címei) szerepeljenek a programban. Erre a célra külön gépi program készíthető; erről a későbbiekben lesz szó.

Maga az értelmezés jelen esetben abban áll, hogy minden pozitív előjelű utasításhoz hozzárendeljük a benne szereplő kódjelnek megfelelő művelet lebegőpontos programját. Adatutasításnál ez a program abból áll, hogy a szóbanforgó utasítás első elmének tartalmát (a hozzátartozó kitévővel együtt) átvisszük a lebegőpontos "A regiszterbe": $Q; Q+1$ -be; a második cím tartalmát pedig a lebegőpontos "B regiszterbe": $P; P+1$ -be. (Az $m a' b'$ utasítás ugyancsak a mantisszák rekeszeinek címeit tartalmazza; a hozzájuk tartozó kitévőkről viszont tudjuk, hogy az $a'+1; b'+1$ rekeszekben vannak.) Miután a "regiszterekbe" beírtuk a műveletben szereplő két számot, a vezérlést átadjuk a lebegőpontos szubrutin megfelelő részére (végrehajtjuk a megfelelő műveletet), s ha a művelet "vesszőtlen", akkor az eredményt beírjuk a második címre.

Vezérlés-, illetve adatátadó utasításnál átadjuk a vezérlést, illetve átvisszük az adatokat a megfelelő címekre. (Itt az átvitel kettős művelet, hiszen a mantisszát és a kitevőt egyaránt át kell vinni.)

A negatív előjelű utasításokat a gép közvetlenül végrehajtja.

Szerkesszük meg ezek után az értelmező szubrutint a XVIII. táblán látható blokkdiagram alapján.

Konstansok:

| | | | | | | | | |
|--------|---|----|-----------|------|-------|--------|---|-------------------|
| (k) | = | U2 | 74 | - | a+46 | (k+14) | = | 2^{-24} |
| (k+1) | = | | 07 | 0000 | 0000 | (k+15) | = | 2^{-27} |
| (k+2) | = | | 00 | 0000 | 0004 | (k+16) | = | $5 \cdot 2^{-30}$ |
| (k+3) | = | U2 | 74 | 0000 | a+110 | (k+17) | = | 2^{-12} |
| (k+4) | = | U2 | 74 | - | a+24 | (k+20) | = | 2^{-26} |
| (k+5) | = | | 00 | 0000 | 7777 | | | |
| (k+6) | = | Á | 05 | 0000 | P | | | |
| (k+7) | = | | 00 | 0001 | 0001 | | | |
| (k+10) | = | | 00 | 7777 | 0000 | | | |
| (k+11) | = | Á | 05 | 0000 | Q | | | |
| (k+12) | = | Á | 05 | P | 0000 | | | |
| (k+13) | = | | 2^{-18} | | | | | |

Paraméter:

| | | | | | |
|-------|---|---|----|---|---|
| (a+2) | = | Á | 05 | u | a |
|-------|---|---|----|---|---|

Program:

| | | |
|-----|---------------|--|
| a | (..) | (A program az a+2-nél kezdődik!) |
| a+1 | + 00 k+13 a+2 | n+i a normalizáló szubrutin kiugró utasításának helye; u az értelmezendő program első utasításának címe! |
| a+2 | (Á 05 u a) | |
| a+3 | FU 34 a a+4 | |
| a+4 | Á 05 k n+i | |
| a+5 | Λ, 16 k+1 a | |

| | | | | |
|------|-----|----|------|------------------------------|
| a+6 | ↓x, | 33 | k+13 | - |
| a+7 | ↓-, | 31 | k+2 | - |
| a+10 | FU | 34 | a+11 | a+54 |
| a+11 | ↓+ | 20 | k+3 | a+45 |
| a+12 | x, | 13 | k+15 | a |
| a+13 | ↓- | 21 | k+14 | a+14 |
| a+14 | (U2 | 74 | - | a+24-i) i = 0, 1, 2, 3, 5, 7 |
| a+15 | - | 01 | k+16 | a+37 |
| a+16 | U2 | 74 | - | a+21 |
| a+17 | - | 01 | k+16 | a+37 |
| a+20 | U2 | 74 | - | a+23 |
| a+21 | + | 00 | k+16 | n+i |
| a+22 | U2 | 74 | - | a+32 |
| a+23 | + | 00 | k+16 | n+i |
| a+24 | ∧, | 16 | k+5 | a |
| a+25 | ↓:, | 32 | k+17 | - |
| a+26 | ↓+ | 20 | k+6 | a+30 |
| a+27 | ↓+ | 20 | k+7 | a+31 |
| a+30 | (| | |) |
| a+31 | (| | |) |
| a+32 | ∧, | 16 | | a |
| a+33 | ↓+ | 20 | | a+35 |
| a+34 | ↓+ | 20 | | a+36 |
| a+35 | (| | |) |
| a+36 | (| | |) |
| a+37 | U2 | 74 | - | a+45 |
| a+40 | + | 00 | k+16 | a+37 |
| a+41 | + , | 51 | <0> | P |
| a+42 | U1 | 24 | a+43 | P |
| a+43 | + , | 51 | <0> | Q |
| a+44 | U1 | 24 | a+45 | Q |
| a+45 | (| | |) |
| a+46 | ∧, | 16 | k+5 | a |
| a+47 | ↓+ | 20 | k+12 | a+51 |

| | | | | |
|-------|-----|----|------|-------|
| a+50 | ↓+ | 20 | k+7 | a+52 |
| a+51 | (| | |) |
| a+52 | (| | |) |
| a+53 | U2 | 74 | - | a+1 |
| a+54 | x, | 13 | k+20 | a |
| a+55 | ↓+ | 20 | a+64 | a+56 |
| a+56 | (| | |) |
| a+57 | Á | 05 | a | a+61 |
| a+60 | ↓+ | 20 | k+7 | a+73 |
| a+61 | (| | |) |
| a+62 | U1 | 24 | a+73 | P |
| a+63 | U2 | 74 | - | a+101 |
| a+64 | U2 | 74 | - | a+57 |
| a+65 | Á | 05 | P | P |
| a+66 | FU | 34 | a+67 | a+75 |
| a+67 | Λ, | 16 | k+10 | a |
| a+70 | ↓+ | 20 | a+72 | a+2 |
| a+71 | U2 | 74 | - | a+2 |
| a+72 | Á | 05 | 0000 | a |
| a+73 | (| | |) |
| a+74 | U1 | 24 | a+1 | P+1 |
| a+75 | Λ, | 16 | k+5 | a |
| a+76 | ↓:, | 32 | k+17 | |
| a+77 | ↓+ | 20 | a+72 | a+2 |
| a+100 | U2 | 74 | - | a+2 |
| a+101 | Λ, | 16 | k+10 | a |
| a+102 | ↓+ | 20 | a+72 | a+2 |
| a+103 | U2 | 74 | - | a+46 |
| a+104 | U2 | 74 | - | ö |
| a+105 | U2 | 74 | - | k |
| a+106 | U2 | 74 | - | o |
| a+107 | U2 | 74 | - | sz |

Az utolsó 4 utasításban ö az összeadó, k a kivonó, o az osztó, sz a szorzó lebegőpontos szubrutin első utasításának címe.

Az értelmező szubrutinokból való kiugrás negatív előjelű vezérlésátadó utasítás segítségével történik. (A pozitív előjelű vezérlésátadó utasításokat ugyanis a szubrutin értelmezné, ezek a vezérlést további értelmezendő utasításokra adnák át.)

Az értelmezendő fixpontos programok elkészítésénél ügyelni kell arra, hogy a negatív utasítások címeit úgy módosítsuk, hogy közben szemelőtt tartsuk: a gép ezeket az utasításokat negatív számokként kezeli.

Látható, hogy egy utasítás értelmezése és lebegőponttal való végrehajtása átlagosan 60-70 utasítás végrehajtását teszi szükségessé. Természetes tehát az a törekvés, hogy lehetőleg inkább a transzformációs módszereket használjuk a fixponttal kapcsolatos nehézség megkerülésére, jól lehet lebegőpontos programozásnál nincsen szükség a túlcsordulás vizsgálatára és a transzformációkra.

Az értelmező szubrutinokkal kapcsolatban felhívjuk a figyelmet arra, hogy ezek "magasabb rendű" programoknak tekinthetők, amelyeknél a kiinduló információ maga is program (az értelmezendő program).

Kézenfekvő ezzel kapcsolatban felvetni azt a gondolatot, hogy elkészíthetők bizonyos u.n. szimbolikus programok is: a gép utasításrendszerétől függetlenül megállapodás-szerűen kódolt algoritmusok. Ezek értelmező szubrutin segítségével a gép utasításrendszerének megfelelően végrehajthatók. El lehet készíteni tehát több gép számára közös szimbolikus programokat is, s külön értelmező szubrutinokkal desiffrálni és végrehajtani azokat.

9.4. Értelmező szubrutin - a komplex aritmetikához.

A komplex számok aritmetikája éppen úgy kétparaméteres műveletekből épül fel, mint a lebegőpontos számok aritmetikája. Ha tehát egy fixpontos programhoz egy komplex számokkal működő algoritmust rendelünk hozzá, azaz ha az adott műveletsorozatot komplex számokon kell végrehajtani, akkor a lebegőpontos számítás esetéhez hasonlóan el lehet készíteni egy értelmező szubrutint. Ez a szubrutin nagyrészt megegyező a lebegőpontos szubrutinnal.

A nem abszolút-értékes műveleteknél csak annyi a különbség, hogy az a+104, a+105, a+106, a+107 rekeszekbe olyan utasításokat helyezünk el, amelyek a komplex aritmetikai szubrutin megfelelő részére adják át a vezérlést.

Tehát

| | | |
|-------------------|-------------------|----------------|
| (a+104) = U2 74 - | komplex összeadás | szubrutinjának |
| (a+105) = U2 74 - | " kivonás | " " kezdő címe |
| (a+106) = U2 74 - | " osztás | " " |
| (a+107) = U2 74 - | " szorzás | " " |

Állapodjunk meg abban, hogy ha egy nem abszolút-értékes művelet után következik feltételes ugrás, akkor ez a váls rész előjelétől függően adja át a vezérlést.

Mint ahogy komplex számok esetén az abszolút-érték képzése nem úgy történik, mint lebegőpontos számok esetén, ezért az abszolútérték-képző blokkot ki kell cserélnünk.

A komplex értelmező szubrutin tehát (csak a lebegőpontos szubrutintól különböző részt írjuk le, az egyező részt pusztán jelöljük) ilyen alakú:

| | | | | | |
|-------|-----------------------------|----|------|---------|-------------------------|
| a | [Megegyezik a lebegőpontos | | | | |
| . | értelmező szubrutinnal | | | | |
| . | | | | | |
| a+40 |] | | | | |
| a+41 | U2 | 74 | - | a+110 | |
| a+42 | | 64 | 0000 | 0002 | } konstansok |
| a+43 | | 00 | Q | P | |
| a+44 | | | | | } szabad rekesz |
| a+45 | [Megegyezik | | | | |
| . | | | | | |
| . | | | | | |
| a+103 |] | | | | |
| a+104 | U2 | 74 | - | komplex | összeadás |
| a+105 | U2 | 74 | - | " | kivonás |
| a+106 | U2 | 74 | - | " | Szorzás |
| a+107 | U2 | 74 | - | " | osztás |
| a+110 | x | 03 | Q | Q | |
| a+111 | x, | 13 | Q+1 | Q+1 | |
| a+112 | ↓+ | 20 | Q | g | gy a gyökvonó szubrutin |
| a+113 | +, | 10 | a+42 | a+113 | kezdő utasítása g a be- |
| a+114 | U1 | 24 | gy | gy+i | menete gy+i a kiugrató |
| a+115 | Á | 05 | e | Q | utasítás helye (e)=√ā |
| a+116 | x | 03 | P | P | |
| a+117 | x, | 13 | P+1 | P+1 | |
| a+120 | ↓+ | 20 | P | g | |
| a+121 | +, | 10 | a+42 | a+121 | |
| a+122 | U1 | 24 | gy | gy+i | |
| a+123 | Á | 05 | e | P | |
| a+124 | Λ, | 16 | k+1 | a | |
| a+125 | ↓+ | 20 | a+43 | a+126 | |
| a+126 | (| | |) | |
| a+127 | Á | 05 | <0> | P+1 | |
| a+130 | U2 | 74 | | a+1 | |

Ha abszolút-értékes művelet után következik feltételes ugrás, akkor e program attól függően adja át a vezérlést az első vagy a második címre, hogy az előző - valós számokkal (t.i. a megfelelő komplex számok valós részével) végzett művelet eredménye negatív, vagy pozitív-e.

9.5. Konverziós szubrutinok.

Az előzőekben megjegyeztük, hogy az értelmező szubrutinok a szó bizonyos értelmében "magasabb rendű" programok, hiszen egy elég bonyolult információt - t.i. egy programot - dolgoznak fel. A feldolgozás abban áll, hogy egy program formájában kódolt információt adott szempontok szerint értelmeznek, s egyben ezeket a szempontokat figyelembevevő, a gép által végrehajtható részprogramokat dolgoznak ki.

Azokat a programokat, amelyek valamilyen információból egy teljes programot dolgoznak ki, programozó programoknak nevezzük. Az értelmező szubrutinok tehát speciális programozó programoknak is tekinthetők, noha itt a részprogramokból (egy szimbolikus utasításnak megfelelő valóságos utasításcsoportok) nem állítunk össze egy teljes programot, hanem a részprogramot mindjárt végre is hajtjuk.

Elkészíthető azonban olyan program is, amely miután értelmez egy utasítást, nem hajtja azt végre, hanem tárolja a kialakított utasításcsoportot, s ezekből az utasításcsoportokból egy teljes programot állít össze. Azokat a programokat, amelyek egy bizonyos programból valamilyen más programot állítanak elő, konverziós szubrutinoknak hívjuk.

A konverziós szubrutinok tehát lényegében program-transzformációk: egy adott programhoz egy másik programot rendelnek hozzá. Egy program transzformálása utasításainak transzformálásából áll.

A legegyszerűbb konverziós szubrutin egy átcímező szubrutin, amely egy adott program minden utasításának első, illetve második címéhez hozzáad egy s számot, ha ezek a címek valamilyen y számnál nagyobbak. Az értelmező szubrutin tárgyalásánál megemlítettük, hogy mielőtt egy adott programot értelmeznénk, át kell alakítani a programot úgy, hogy abban a lebegőpontos adatok címei szerepeljenek. Az ezt végrehajtó program a következőképpen működik.

Tegyük fel, hogy a kiinduló adatokat és a részeredményeket fixponttal történő számolásnál az u -tól $u+i-1$ -ig terjedő rekeszekben tároljuk (a program után legyenek elhelyezve); ugyanezeket lebegőponttal való számolás esetén kétszerannyi rekeszben kell elhelyezni! Legyenek ezek: a v -tól $v+k-1$ -ig terjedő rekeszek (itt $k=2i$).

A címutasításokat (negatív előjelűek) nem kell módosítani, hiszen azok adatkímeket nem tartalmaznak.

A pozitív előjelű utasítások közül (ezek nem címutasítások) át kell címezni minden adatutasítást, ezeken kívül még az $U1$ 24 vezérlésátadó utasítás második címét is (ez ugyanis adatkím).

Program:

| | | | | |
|------|----|----|---------------------------|------|
| t | (| |) | |
| t+1 | + | 00 | $\langle 2^{-18} \rangle$ | t+2 |
| t+2 | Á | 05 | w | t |
| t+3 | FU | 34 | t+1 | t+4 |
| t+4 | Λ, | 16 | t+30 | t |
| t+5 | ↓- | 21 | t+36 | m |
| t+6 | FU | 34 | t+7 | t+20 |
| t+7 | -, | 11 | t+30 | t |
| t+10 | FU | 34 | t+11 | t+1 |
| t+11 | -, | 11 | t+36 | t |
| t+12 | ↓: | 32 | | - |

A t+2 utasítással kezdjük a program végrehajtását

| | | | | |
|------|-----|----|-------------------------------|------|
| t+13 | ↓+, | 30 | t+37 | - |
| t+14 | ↓∧ | 26 | t+32 | m |
| t+15 | ∧, | 16 | t+33 | t |
| t+16 | ↓+ | 20 | m | m |
| t+17 | U2 | 74 | - | t+22 |
| t+20 | ↓:, | 32 | $\langle \frac{1}{2} \rangle$ | - |
| t+21 | ↓+ | 20 | t+37 | m |
| t+22 | ∧, | 16 | t+33 | t+2 |
| t+23 | ↓x, | 33 | $\langle 2^{-12} \rangle$ | - |
| t+24 | ↓+ | 20 | t+34 | t+27 |
| t+25 | ∧, | 16 | t+35 | t |
| t+26 | ↓+, | 30 | m | - |
| t+27 | (| | |) |

Konstansok:

| | | | | |
|--------|-----|----|------|------|
| (t+30) | = + | 00 | 7777 | 7777 |
| (t+31) | = + | 34 | 0000 | 0000 |
| (t+32) | = + | 00 | 7777 | 7777 |
| (t+33) | = + | 00 | 7777 | 0000 |
| (t+34) | = + | 24 | t+1 | 0000 |
| (t+35) | = + | 77 | 0000 | 0000 |

Paraméterek:

| | | | | |
|--------|-----|----|---|---|
| (t+36) | = + | 00 | u | u |
| (t+37) | = + | 00 | v | v |
| (t+2) | = + | 05 | Á | w t, ahol w az értelmezendő program első utasításának címe. |

Munkarekesz: m.

Megszerkeszthető egy olyan konverziós szubrutin is, amely egy skalárműveletekkel programozott algoritmushoz hozzárendeli a megfelelő vektorműveletekből álló programot.

A következőkben vázlatosan ismertetünk egy olyan konverziós szubrutint, amely egy fixpontos programot lebegő-

pontos programmá transzformál át. Nevezzük ezt a szubrutint a továbbiakban K szubrutinnak.

A K szubrutin két fő blokkból áll:

Az első blokk utasításonként végigfutja a konvertálandó fixpontos programot, s megvizsgálja, hogy egy kiválasztott utasítás adatutasítás, vezérlésátadó utasítás, vagy címmódosító utasítás-e? Ha a tekintett program-elem nem adatutasítás, lássuk el negatív előjellel. A szubrutin kialakítja az adatutasítások végleges formáját, s beülteti a konvertált program soronkövetkező helyére. Kialakítja ezenkívül a címmódosító utasításoknak megfelelő utasításcsoportokat is, s ezeket is végleges formában helyükre teszi. A vezérlésátadó, utasítások megfelelő képeit is előállítja, de nem végleges formában, hanem úgy, hogy azokban a fixpontos program címei szerepeljenek. (Ezeket ugyanis - ha a vezérlést a program következő részére adják át - nem is tudnók kialakítani, hiszen nem tudjuk, hogy a vezérlést a konvertált program melyik utasítására kell átadni.) A kialakított, de fixpontos címet tartalmazó utasításcsoportokat szintén elhelyezzük a konvertált program megfelelő helyére, de a negatív előjelet meghagyjuk.

A későbbiek folyamán szükség van arra az információra, hogy egy adott fixpontos utasításnak megfelelő utasításcsoport a konvertált programban hol helyezkedik el. Ezért az egyes utasítások transzformálása után, annak eredeti helyére beírjuk a $00\ 00rs\ w$ alakú információt, ahol r egy bizonyos segédinformáció, s a transzformált utasításcsoport hossza, w pedig a szóbanforgó utasításcsoport első utasításának a címe a konvertált programban.

Az első blokk eredményeként tehát lényegében létrejön a konvertált program, de a vezérlésátadó utasítások még a fixpontos program elmeit tartalmazzák. A második blokk fel-

adata ezek helyett a címek helyett beírni a konvertált program megfelelő címeit, felhasználva a megtartott 00 00rs w információt.

Vizsgáljuk meg részletesebben a K szubrutinnak azt a blokkját, amely az adatutasítások képét szerkeszti meg. Ez a szubrutin önállóan is működik, olyan programok konvertálására alkalmas, amelyekben nincsenek vezérlésátadó és átcímező utasítások.

Jelöljük ki vizsgáló rekesznek a t rekeszt, utasításszámlálónak a q rekeszt.

Ismeretes, hogy ha egy m a b alakú fixpontos utasítást lebegőpontosan akarunk végrehajtani, akkor helyette a következő utasításcsoportot kell felírni. (Nevezzük ezt M -blokknak.)

| | | | | |
|-----|----|----|--------------|------|
| w | Á | 05 | b' | P |
| w+1 | Á | 05 | b'+1 | P+1 |
| w+2 | Á | 05 | a' | Q |
| w+3 | Á | 05 | a'+1 | Q+1 |
| w+4 | +, | 10 | <6400000002> | w+4 |
| w+5 | U1 | 24 | m'+1 | n+1 |
| w+6 | Á | 05 | P | b' |
| w+7 | Á | 05 | P+1 | b'+1 |

ahol m' az m -nek megfelelő lebegőpontos szubrutin első utasításának címe, $n+1$ a normalizáló szubrutin kiugrató utasításának a helye, a' , $a'+1$, b' , $b'+1$ a műveletben szereplő számok lebegőpontos alakjának rekeszeit jelentik.

Ha a művelet "vesszős" (első kódja 1-es), akkor a $w+6$, $w+7$, ha nyílas (első kódja 2-es), akkor a w $w+1$, ha nyílvesszős (első kódja 3-as), akkor w , $w+1$, $w+6$, $w+7$ utasításokra nincs szükség.

Tehát a K program a következőképpen jár el:

kialakítja az M blokkot, s ha a művelet vessző és nyíl nélküli, akkor a w-től w+7-ig terjedő, ha vesszős, akkor a w-től w+5-ig terjedő, ha nyílas, akkor a w+2-től w+7-ig terjedő, ha nyílvesszős, akkor a w+2-től w+5-ig terjedő utasításokat viszi át a konvertált program megfelelő helyére, s legyen az első és második esetben az \underline{r} segédinformáció: $r=4$, a harmadik és negyedik esetben $r=2$.

Természetesen ügyelni kell arra, hogy a w+4 utasításban a második címen éppen az a cím szerepeljen, ahová ez az utasítás kerülni fog.

A program:

Konstansok:

| | | | | | |
|--------|---|----|---------------|-------|------|
| (k) | = | | 00 | 7777 | 7777 |
| (k+2) | = | | 07 | 0000 | 0000 |
| (k+3) | = | U1 | 24 | h | n+1 |
| (k+4) | = | | 00 | 0004 | 0000 |
| (k+5) | = | FU | 34 | t+16 | t+17 |
| (k+6) | = | | $\frac{1}{2}$ | | |
| (k+7) | = | | 10 | 0000 | 0000 |
| (k+10) | = | | 00 | 0000 | 0007 |
| (k+11) | = | | 00 | 0000 | 0004 |
| (k+12) | = | Á | 05 | t+120 | 0000 |
| (k+13) | = | | 00 | 0000 | 0005 |
| (k+14) | = | | 00 | 0000 | 0002 |
| (k+15) | = | Á | 05 | t+122 | 0000 |
| (k+16) | = | | 00 | 0000 | 0003 |
| (k+17) | = | | 05 | 0000 | 0000 |
| (k+20) | = | | 00 | 0000 | 7777 |
| (k+21) | = | | 00 | 7777 | 0000 |
| (k+22) | = | Á | 05 | 0000 | P |
| (k+23) | = | | 00 | 0001 | 0001 |

| | | | | | |
|--------|---|----|----|-----------|------|
| (k+24) | = | Á | 05 | 0000 | Q |
| (k+25) | = | Á | 05 | P | 0000 |
| (k+26) | = | +, | 10 | k+34 | 0000 |
| (k+27) | = | ↓+ | 20 | m+1 | 0000 |
| (k+30) | = | | | 2^{-12} | |
| (k+31) | = | | | 2^{-30} | |
| (k+32) | = | | | 2^{-15} | |
| (k+33) | = | | | 2^{-18} | |
| (k+34) | = | | 64 | 0000 | 0000 |

Munkarekeszek: m, m+1, s, r, q.

| | | | | | | |
|--------------|-------|---|----|------|---|---|
| Paraméterek: | (p) | = | 00 | a | a | <u>a</u> a fixpontos program kiinduló adatainak első rekesze. |
| | (p+1) | = | 00 | A | A | <u>A</u> az <u>a</u> -nak megfelelő rekesz a lebegőpontos programban. |
| | (p+2) | = | 00 | 0000 | w | <u>w</u> a lebegőpontos program első utasításának a helye. |
| | (t+3) | = | 05 | u-1 | t | <u>u</u> a fixpontos program első utasításának a helye. |

Legyen:

| | | | | | |
|-----|----|----|---|------------|-----------|
| h-4 | U2 | 74 | - | absz. ért. | összeadás |
| h-3 | U2 | 74 | - | " " | kivonás |
| h-2 | U2 | 74 | - | " " | osztás |
| h-1 | U2 | 74 | - | " " | szorzás |
| h | U2 | 74 | - | | összeadás |
| h+1 | U2 | 74 | - | | kivonás |
| h+2 | U2 | 74 | - | | osztás |
| h+3 | U2 | 74 | - | | szorzás |

| | | | | | |
|------|---------------------|----|----------|-------|--|
| t | (| | |) | |
| t+1 | Á | 05 | p+2 | q | A program végrehajtását a t+1 utasítással kezdjük. |
| t+2 | + | 00 | k+33 | t+3 | |
| t+3 | Á | 05 | u-1 | t | |
| t+4 | FU | 34 | megállás | t+5 | |
| t+5 | ∧, | 16 | k | t | |
| t+6 | ↓-, | 31 | p | - | |
| t+7 | ↓:, | 32 | k+6 | - | |
| t+10 | ↓+ | 20 | p+1 | m | |
| t+11 | ∧, | 16 | k+2 | t | |
| t+12 | ↓x | 23 | k+30 | m+1 | |
| t+13 | ↓+ | 20 | k+3 | t+125 | |
| t+14 | -, | 11 | k+4 | m+1 | |
| t+15 | FU | 34 | t+16 | t+111 | |
| t+16 | Á | 05 | k+5 | t+21 | |
| t+17 | + | 00 | k+4 | t+21 | |
| t+20 | - | 01 | k+7 | t | |
| t+21 | (| | |) | |
| t+22 | Á | 05 | k+10 | s | |
| t+23 | Á | 05 | k+11 | r | |
| t+24 | +, | 10 | k+12 | q | |
| t+25 | U1 | 24 | t+60 | t+76 | |
| t+26 | Á | 05 | k+13 | s | |
| t+27 | U2 | 74 | - | t+23 | |
| t+30 | [szabad rekeszek] | | | | |
| t+31 | [| | | | |
| t+32 | Á | 05 | k+13 | s | |
| t+33 | Á | 05 | k+14 | r | |
| t+34 | +, | 10 | k+15 | q | |
| t+35 | U1 | 24 | t+64 | t+76 | |
| t+36 | Á | 05 | k+16 | s | |
| t+37 | U2 | 74 | - | t+33 | |
| t+40 | [| | | | |
| · | [szabad rekeszek] | | | | |
| · | [| | | | |
| · | [| | | | |
| t+45 | [| | | | |

| | | | | |
|-------|---------------------|----|-------|-------|
| t+46 | - | 01 | k+4 | t+125 |
| t+47 | U2 | 74 | - | t+26 |
| t+50 | [Szabad rekeszek] | | | |
| . | | | | |
| . | | | | |
| . | | | | |
| t+55 | | | | |
| t+56 | - | 01 | k+4 | t+125 |
| t+57 | U2 | 74 | - | t+536 |
| t+60 | ∧, | 16 | k+20 | m |
| t+61 | ↓:, | 32 | k+30 | - |
| t+62 | ↓+ | 20 | k+22 | t+120 |
| t+63 | ↓+ | 20 | k+23 | t+121 |
| t+64 | ∧, | 16 | k+21 | m |
| t+65 | ↓+ | 20 | k+24 | t+122 |
| t+66 | ↓+ | 20 | k+23 | t+123 |
| t+67 | ∧, | 16 | k+20 | m |
| t+70 | ↓+ | 20 | k+25 | t+126 |
| t+71 | ↓+ | 20 | k+23 | t+127 |
| t+72 | +, | 10 | r | q |
| t+73 | ↓+ | 20 | k+26 | t+124 |
| t+74 | :, | 12 | k+30 | s |
| t+75 | ↓+ | 20 | q | m+1 |
| t+76 | (| | |) |
| t+77 | + | 00 | k+23 | t+76 |
| t+100 | - | 01 | k+31 | s |
| T+101 | FU | 34 | t+102 | t+76 |
| t+102 | ∧, | 16 | k+21 | t+3 |
| t+103 | ↓x | 23 | k+30 | - |
| t+104 | ↓+ | 20 | k+27 | t+106 |
| t+105 | :, | 12 | k+32 | r |
| t+106 | (| | |) |
| t+107 | ∧, | 16 | k+20 | t+76 |
| t+110 | U1 | 24 | t+2 | q |

| | | | | | |
|-------|----|----|-------|-------|--------------------|
| t+111 | +, | 10 | k+17 | m | |
| t+112 | U1 | 24 | t+113 | t+120 | |
| t+113 | ↓+ | 20 | k+23 | t+121 | |
| t+114 | Á | 05 | k+31 | s | |
| t+115 | Á | 05 | k+14 | r | |
| t+116 | +, | 10 | k+12 | q | |
| t+117 | U1 | 24 | t+74 | t+76 | |
| t+120 | (| | |) | } M-blokk helye |
| t+121 | (| | |) | |
| t+122 | (| | |) | |
| t+123 | (| | |) | |
| t+124 | (| | |) | |
| t+125 | (| | |) | |
| t+126 | (| | |) | |
| t+127 | (| | |) | |
| t+130 | (| | |) | |

Az itt tárgyalt konverziós szubrutinok csak azt akarták bemutatni, hogy a programozás munkájának egy részét át lehet hárítani a gépre; csak a gép logikai lehetőségeitől és a mi gyakorlatunktól függ, hogy a programkészítés munkájának milyen hányadát végeztetjük magával a géppel.

10. Fejezet.

GYAKORLATI PÉLDÁK.

Az alábbiakban néhány - a gyakorlatban sokszor előforduló - probléma részletes numerikus analízisét és erre támaszkodva a megoldás teljes programját mutatjuk be. Meg kell azonban említenünk, hogy a parciális differenciálegyenletekkel csak futólag foglalkozunk, jóllehet ilyen jellegű feladatok gyakran előfordulnak, numerikus szempontból pedig épp ezek megoldása jelenti a fő nehézséget. Mégsem célszerű ezekkel e helyütt részletesen foglalkozni, mert általánosságban semmit sem tudunk mondani még az alkalmazandó - legcélszerűbb - numerikus módszerekről sem. A parciális differenciálegyenletekkel kapcsolatos problémák ugyanis egyediek, egyedi tehát numerikus kezelésük, ill. az ennek megfelelő gépi program is.

10.1. Lineáris egyenletrendszerek.

Az M-3 gép egyik fontos alkalmazási területe a nagy ismeretlen-számú lineáris egyenletrendszerek megoldása. Igen sok gyakorlati számítási feladat vezethető vissza lineáris egyenletrendszerek megoldására, részint többé-kevésbé közvetlenül (például egyes statikai és szilárdságtani számítások, a geodéziai kiegyenlítő számítások, gazdasági téren különböző tervezési és árrendezési problémák stb.), részint közvetve, (például az áramlási profilok számításai, rövid időre szóló időjárás-prognózisok készítése, stb. ahol

tulajdonképpen, lineáris közönséges vagy parciális differenciálegyenletekre vonatkozó peremérték-feladatok, illetve lineáris integrálegyenletek megoldásáról van szó).

A lineáris egyenletrendszerek gépi megoldási módszereinek ismertetésénél mindenekelőtt meg kell jegyezni, hogy az u.n. Cramer-szabály - amelynek értelmében az

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &= b_1 \\ \cdot & \\ \cdot & \\ \cdot & \\ a_{n1}x_1 + \dots + a_{nn}x_n &= b_n \end{aligned} \tag{1}$$

n ismeretlenes lineáris egyenletrendszer x_i megoldása ($i=1, \dots, n$) az

$$x_i = \frac{D_i}{D}$$

alakban írható fel, ahol D az egyenletrendszer determinánsa (itt és a továbbiakban is feltesszük, hogy $D \neq 0$), és

$$D_i = \begin{vmatrix} a_{11} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1n} \\ \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot & & \cdot \\ a_{n1} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{nn} \end{vmatrix}$$

- általában nem alkalmazható a lineáris egyenletrendszerek (kézi vagy gépi) numerikus megoldására. Ennek oka egyszerűen az, hogy a numerikus analízis nem ismer olyan eljárást, amelynek segítségével egy általános n -edrendű determináns értékét n^3 -nél nagyságrendben kevesebb algebrai művelettel ki lehetne számítani. A Cramer-szabály alkalmazásánál $n+1$ determináns értékét kell meghatározni, összesen tehát n^4 -nél nagyságrendben nem kevesebb műveletre volna szükség;

ugyanekkor viszont a továbbiakban ismertetett Gauss-féle eliminációs eljárással egy n ismeretlenes lineáris egyenlet rendszert kb. n^3 számú művelettel meg lehet oldani.

Az u.n. direkt eljárások közül (amelyek véges számú lépés után elvben pontosan megadják az ismeretlenek értékeit) az M-3 gépre való alkalmazás szempontjából a Gauss-féle eliminációs eljárás látszik a legelőnyösebbnek: főként azért, mert ezt az eljárást legkönnyebb fixpontos számolással, indokolatlan túlcsoordulás veszélye nélkül keresztülvinni. A Gauss-féle eliminációs eljárás az M-3 gépre alkalmazott formájában két különálló részből áll; az első részben a gép $n-1$ eliminációs lépést hajt végre, és ezeknek eredményeképpen az eredetileg adott (1) egyenletrendszerből egy vele ekvivalens

$$\begin{aligned} \alpha_{11}x_1 + \alpha_{12}x_2 + \dots + \alpha_{1n}x_n &= \beta_1 \\ \alpha_{22}x_2 + \dots + \alpha_{2n}x_n &= \beta_2 \\ &\cdot \\ &\cdot \\ &\cdot \\ &+ \alpha_{nn}x_n = \beta_n \end{aligned} \tag{2}$$

”trianguláris” egyenletrendszert állít elő, a második részben a gép a (2) egyenletrendszert oldja meg. Az eljárás megkezdése előtt a következő módon visszük be a gép memóriájába az a_{ij} ($i, j = 1, \dots, n$) és b_k ($k = 1, \dots, n$) számokat: az $r, \dots, r+n-1$ című rekeszekben elhelyezzük az a_{11}, \dots, a_{1n} együtthatókat, az $r+n$ című rekeszben a b_1 állandót, az $r+n+1, \dots, r+2n$ rekeszekben az a_{21}, \dots, a_{2n} együtthatókat, az $r+2n+1$ rekeszben a b_2 állandót, stb.

Az eljárás végén az $s, \dots, s+n-1$ című rekeszekben rendre megkapjuk az x_1, \dots, x_n ismeretleneket.

Az első eliminációs lépésben a gép a következő műveleteket hajtja végre:

a) Megnézi, hogy az a_{ij} és b_k számok között van-e abszolút értékben $\frac{1}{2}$ -nél nagyobb; ha igen mindegyik egyenletet végigszorozza $\frac{1}{2}$ -del.

b) Kiválasztja az x_1 együtthatói közül a legnagyobb abszolút értékűt; ha az a k -adik egyenletben van, akkor a_{k1} lesz az u.n. első főelem.

c) Az első egyenletet felcseréli a k -adik egyenlettel (az a_{11} elemet az r című rekeszből az $r+(k-1) \cdot (n+1)$ című rekeszbe, az a_{k1} elemet az $r+(k-1) \cdot (n+1)$ című rekesztől az r című rekeszbe, viszi át, stb.)

d) Az így kapott egyenletrendszer második egyenletéhez hozzáadja az első egyenlet $-\frac{a_{21}}{a_{11}}$ -szeresét (a felcseréléssel kapott egyenletrendszer együtthatóit az egyszerűség kedvéért ugyanúgy jelöljük, mint az eredeti egyenletrendszer együtthatóit), harmadik egyenletéhez az első egyenlet $-\frac{a_{31}}{a_{11}}$ -szeresét, ..., n -edik egyenletéhez az első egyenlet $-\frac{a_{n1}}{a_{11}}$ -szeresét.

e) Megváltoztatja az elimináció paramétereit, megnézi, hogy vége van-e az eliminációs résznek, és (ha $n > 2$) rátér a második eliminációs lépésre.

Az a) blokkra azért van Cramern szükség, hogy a d) blokkban végrehajtott összeadásoknál ne léphessen fel túlcsondulás. A főelem kiválasztásának és a megfelelő egyenletcserének kettős célja van: egyrészt a) blokkban a hányadosok képzésénél ($i=2, \dots, n$) elkerüljük a túlcsondulás veszélyét; másrészt, csökkentjük a szorzásoknál és osztásoknál fellépő kerekítési hibák befolyását (ez utóbbi állításunk bizonyítására itt most nem térünk ki).

Az első eliminációs lépéssel az x_1 ismeretlent kiküszöböltük, a második, harmadik, ..., n -edik egyenletből; az első egyenlet változatlan maradt.

Az i -edik eliminációs lépésben ($i=2, \dots, n-1$) a gép először megnézi, hogy az $i-1$ -edik lépéssel kapott egyenletrendszer együtthatói és az egyenletrendszer jobb oldalán álló állandók között van-e abszolút értékben $\frac{1}{2}$ -nél nagyobb; ha igen, az egész egyenletrendszert végigszorozza $\frac{1}{2}$ -del. Ezután megállapítja, hogy az i -edik, $i-1$ -edik, ..., n -edik egyenlet közül melyikben legnagyobb az x_i ismeretlen együtthatójának abszolút értéke (a legnagyobb abszolút értékű együttható az un. i -edik főelem), ezt az egyenletet felcseréli az i -edik egyenlettel, majd az első eliminációs lépés d) blokkjához hasonlóan az x_i ismeretlent kiküszöböli az $i+1$ -edik, $i+2$ -edik, ..., n -edik egyenletből. végül ismét megváltoztatja az elimináció paramétereit, és, ha még nincs vége az eliminációnak (vagyis, ha $i < n-1$), rátér az $i+1$ -edik eliminációs lépésre. Az i -edik lépésben az első, második, ..., i -edik egyenlet nem változik.

Az eljárás második része, a (2) egyenletrendszer megoldása igen egyszerű: az n -edik egyenletből rögtön megkapjuk az x_n ismeretlen értékét, ezt az értéket az $n-1$ -edik egyenletbe visszahelyettesítve megkapjuk x_{n-1} -et; x_n -et és x_{n-1} -et az $n-2$ -edik egyenletbe visszahelyettesítve, megkapjuk x_{n-2} -t stb. Az ebben a részben végrehajtott összeadásoknál és osztásoknál túlcsoordulás léphet fel. Ha a túlcsoordulás az x_i ismeretlen értékének kiszámítása közben történt, akkor a gép újraindítása után egy külön program segítségével megszorozzuk $\frac{1}{2}$ -del a már meghatározott x_{i+1}, \dots, x_n ismeretleneket, a β_1, \dots, β_i állandókat és ismét megpróbáljuk kiszámítani az x_i ismeretlent; ha megint, túlcsoordulás lép fel, akkor újból szorzunk $\frac{1}{2}$ -del, stb. Végeredményben az x_1, \dots, x_n ismeretleneket egy 2^{-m} alakú lépésfaktorral szorozva kapjuk meg, ahol m a fellepett túlcsoordulások száma.

A Gauss-féle eliminációs eljárás teljes programja elég hosszú, ezért itt most csak az eljárás egyszerűsített változatát fogjuk programozni: az eliminációs lépéseknél a), b) és c) blokkokat elhagyjuk, csak a d) blokkot (a tulajdonképpeni eliminációt) tartjuk meg. Ez az egyszerűsített program a gyakorlatban abban az esetben használható, ha az (1) egyenletrendszer együtthatói közül az a_{ii} együtthatók. ($i=1, \dots, n$) dominálnak (nagyságrendben nagyobbak az összes többi együtthatónál).

Az eliminációs rész egyszerűsített programja a következő:

(k) = (+13 k+1 r+n+1); (k+1) = - 77 7777 7777;
 (k+2) = (+20 r+n+2 r+n+2); (k+3) = + 00 0001 0001;
 (k+4) = + 00 0001 0001; (k+5) = (x, 13 p r+n+1);
 (k+6) = (+00 0000 n); (k+7) = (+ 00 0000 0001);
 (k+10) = + 00 0000 n+1; (k+11) = + 13 k+1 r+n(n+1);
 (k+12) = + 00 0000 n+2; (k+13) = + 00 n+2 n+2;
 (k+14) = 00 n+2 0000; p: munkarekesz.

| | | | | |
|------|-------|----|------|-------|
| a | Á | 05 | k | a+2 |
| a+1 | Á | 05 | k+2 | a+5 |
| a+2 | (| | |) |
| a+3 | (↓: | 22 | r | p) |
| a+4 | (x, | 13 | p | r+1) |
| a+5 | (| | |) |
| a+6 | + | 00 | k+3 | a+5 |
| a+7 | + | 00 | k+4 | a+4 |
| a+10 | ↓ - , | 71 | k+5 | - |
| a+11 | FU | 34 | a+4 | a+12 |
| a+12 | - | 01 | k+6 | a+4 |
| a+13 | + | 00 | k+7 | a+5 |
| a+14 | + | 00 | k+10 | a+2 |
| a+15 | ↓ - , | 71 | k+11 | - |
| a+16 | FU | 34 | a+2 | a+17 |

| | | | | |
|------|-------|----|------|------|
| a+17 | + | 00 | k+13 | k+2 |
| a+20 | + | 00 | k+14 | a+3 |
| a+21 | + | 00 | k+12 | a+4 |
| a+22 | + | 00 | k+10 | k+5 |
| a+23 | - | 01 | k+4 | k+6 |
| a+24 | + | 00 | k+3 | k+7 |
| a+25 | + | 00 | k+3 | k |
| a+26 | ↓ - , | 71 | k+11 | - |
| a+27 | FU | 34 | a | a+30 |

A trianguláris egyenletrendszer megoldásának programja:

$$\begin{aligned}
 (k+15) &= (+ 13 a+n r+n(n+1)-1); & (k+16) &= + 00 n+1 0000; \\
 (k+17) &= + 00 0001 n+2; & (k+20) &= + 24 a+43 a; \\
 (k+21) &= (+ 13 a+n r+n(n+1)-1);
 \end{aligned}$$

| | | | | |
|------|-------|----|------------|------------|
| a+30 | Á | 05 | k+21 | a+34 |
| a+31 | (Á | 05 | r+n(n+1)-1 | m) |
| a+32 | - , | 51 | k+15 | a+34 |
| a+33 | FU | 34 | a+34 | a+41 |
| a+34 | (x, | 13 | s+n | r+n(n+1)-1 |
| a+35 | ↓x, | 33 | k+1 | - |
| a+36 | ↓+ | 20 | p | p |
| a+37 | + | 00 | k+3 | a+34 |
| a+40 | U2 | 74 | - | a+40 |
| a+41 | (:, | 12 | r+n(n+1)-2 | p) |
| a+42 | (U1 | 24 | a+43 | s+n-1) |
| a+43 | - | 01 | k+16 | a+31 |
| a+44 | - | 01 | k+17 | a+21 |
| a+45 | - | 01 | k+14 | a+41 |
| a+46 | - | 01 | k+10 | a+15 |
| a+47 | - | 01 | k+4 | a+42 |
| a+50 | ↓ - , | 71 | k+20 | - |
| a+51 | FU | 34 | a+52 | a+30 |
| a+52 | .. | .. | | |

Meg kell említenünk, hogy az itt leírt Gauss-féle eliminációs módszer komoly hátránya: nem veszi figyelembe, hogy az egyenletrendszer mátrixa néha igen egyszerű felépítésű (az elemek között igen sok zérus értékű lehet), olyannyira, hogy valamelyik iterációs eljárás lényegesen gyorsabban eredményhez vezetne. Ezt tehát mindig meg kell vizsgálnunk!

10.2. Differenciálegyenletek numerikus integrálása.

Közönséges differenciálegyenletek numerikus integrálására - ill. az ezekkel kapcsolatos kezdeti-, perem- és vegyes-problémák numerikus megoldására - számos módszert feldolgozott az irodalom. A probléma jellege és a pontossági igények alapján kell eldöntenünk, hogy melyiket célszerű adott esetben használni.

Az alábbiakban példaképp bemutatjuk, hogy hogyan lehet elsőrendű egyenletek esetében a Runge-Kutta, ill. a differencia-módszert fixpontosan programozni.

10.2.1 . A Runge-Kutta módszer elsőrendű egyenleteknél.

A Runge-Kutta módszer az egyik legjobban elterjedt numerikus integrálási módszer. Előnye, hogy eléggé pontos, s elég nagy lépésközzel lehet használni. Kézi számításnál nagy hátránya viszont, hogy sok számolást igényel. Gépi számításnál különösen fontos az az előnye, hogy nem kell kezdőértékeket számolni, mint más módszereknél.

Az $y(x_0) = y_0$ kezdeti feltétellel adott $y' = f(x, y)$ differenciálegyenlet megoldásának az $(x_i = x_{i-1} + h = x_0 + i \cdot h)$ pontban vett y_i közelítőértékét a Runge-Kutta módszer szerint az

$$y_{i+1} = y_i + k; \quad (i = 0, 1, 2, \dots, n-1; \quad x_0 = a; \quad x_n = b)$$

képlet alapján számíthatjuk ki, ahol a, ill. b azon intervallum alsó, ill. felső határpontja, ahol a megoldást keressük, és

$$k = \frac{1}{6} (k_1 + k_2 + k_3 + k_4)$$

$$k_1 = h f(x_i, y_i)$$

$$k_2 = h f\left(x_i + \frac{h}{2}; y_i + \frac{k_1}{2}\right)$$

$$k_3 = h f\left(x_i + \frac{h}{2}; y_i + \frac{k_2}{2}\right)$$

$$k_4 = h f(x_i + h; y_i + k_3)$$

Ezt a képletsorozatot a következő sémába foglalhatjuk:

| x | y | k=h · f(x,y) |
|-----------------------|-------------------------|---|
| x_i | y_i | k_1 |
| $x_i + \frac{1}{2} h$ | $y_i + \frac{1}{2} k_1$ | k_2 |
| $x_i + \frac{1}{2} h$ | $y_i + \frac{1}{2} k_2$ | k_3 |
| $x_i + h$ | $y_i + k_3$ | k_4 |
| $x_{i+1} = x_i + h$ | $y_{i+1} = y_i + k$ | $k = \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4)$ |

A módszer részletes leírása megtalálható L. Collatz: Numerische Behandlung von Differentialgleichungen (Springer-Verlag, Berlin) könyvében az 59-67 oldalon.

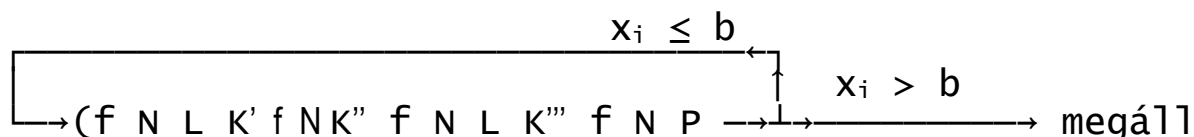
Írjuk fel a feladat blokkdiagramját. Minthogy a programozással kapcsolatos irodalomban gyakran találkozhatunk a blokkdiagramoknak az eddig ismert módnál némileg eltérő felírásával, írjuk fel most a blokkdiagramot a következőképpen:

Nevezzük az egyes blokkokat operátoroknak, és jelöljük nagybetűkkel.

A Runge-Kutta módszer a következő operátorokból áll:

- L - operátor: az x_i argumentumérték növelése $\frac{h}{2}$ -vel,
- K' operátor: y_i növelése $\frac{k_1}{2}$ -vel,
- K'' operátor: y_i növelése $\frac{k_2}{2}$ -vel,
- K''' operátor: y_i növelése k_3 -mal
- K operátor: k kiszámítása
- f operátor: az f függvény kiszámítása az adott helyen.
- N operátor: $k_j = h f(x,y)$ kiszámítása (j=1, 2, 3, 4)
- P operátor: megvizsgálja, hogy $x_i \leq b$ teljesül-e?

Így a számítás menete az



blokkdiagram alapján programozható. Elkészítünk egy fix-pontos programot; a túlcsordulásra természetesen ügyelni kell. Tegyük fel, hogy $|y'| = |f(x,y)| \leq K_0 < 1$, $|y| \leq K_0 < 1$ és $|x| < 1$ teljesül. K_0 , K_1 -et úgy választjuk meg, hogy a közbülső eredményekben se lépjen fel túlcsordulás.

A közbűlső eredményekre a következő (durva) becslés adható:

$$|k_j| \leq |h f(x,y)| < h \quad (j=1, 2, 3, 4)$$

$$\text{és } |y_i + k_j| \leq |y_i| + |k_j| + h < 1 - 2^{-30}$$

ha

$$|y_i| \leq 1 - 2^{-30} - h.$$

A k -ra viszont

$$|k| = \left| \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) \right| \leq \frac{1}{6} 6h = h \quad (h \leq \frac{1}{3}, \text{ mert } n \geq 3)$$

Ha tehát $K_0 = \frac{1}{2}$; $K_1 = \frac{1}{2}$, akkor biztosan nem lép fel túlcsordulás.

A túlcsordulás elkerülhető a következő feltételek teljesülése esetén is:

$$|y'| < 1, \quad |y_0| \leq 1 - \delta \text{ és } |x| \leq \delta < 1.$$

Ugyanis $|k| < h$ lévén

$|y_i - y_{i-1}| < h$; tehát a megoldás lépésenként legfeljebb h -val növekszik.

Az i -edik pontban vett közelítő értékre tehát:

$$|y_i| \leq |y_0| + ih.$$

így, ha

$$|y_0| + ih < 1,$$

azaz

$$|y_0| < 1 - ih,$$

akkor

$$|y_i| < 1$$

Ha tehát $|y_0| < 1 - \delta \leq 1 - ik$ ($nh \leq \delta$), akkor $\max |y_i| < 1$
(1)

Ha a differenciálegyenlet nem teljesíti az előbbi feltételeket, s nem lehet úgy transzformálni, hogy teljesítse, akkor lebegőpontos programra van szükség.

Rekeszelosztás.

Az $f(x,y)$ függvény megfelelő értékeit külön segéd-szubrutin felhasználásával számítjuk. Legyen ennek a kezdete q ; a visszaugrató utasítás helye; $q + w$ és

$$(\alpha) = x$$

$$(\delta) = f(x,y).$$

Legyen a kinyomtató szubrutin az r -től $r + j$ -ig terjedő rekeszben elhelyezve, és a kinyomtatandó adat helye legyen s .

Legyenek δ_1 ; δ_2 ; δ_3 ; a ; γ ; m ; $m+1$; munkarekeszek.

Program:

| | | | | | |
|------|----|----|-------------------|----------|----------------------------|
| t | Á | 05 | <x ₀ > | α | |
| t+1 | Á | 05 | <y ₀ > | β | |
| t+2 | x, | 13 | < $\frac{1}{2}$ > | <h> | |
| t+3 | U1 | 24 | t+4 | m | $\leftarrow \frac{h}{2}$ |
| t+4 | Á | 05 | <0> | m+1 | |
| t+5 | Á | 05 | β | γ | |
| t+6 | +, | 10 | t+44 | t+6 | |
| t+7 | U1 | 24 | q | q+w | |
| t+10 | x | 03 | m | δ | |
| t+11 | ↓+ | 20 | m+1 | m+1 | $\leftarrow \frac{k_1}{2}$ |

| | | | | | |
|------|--------------------|----|-----------------------------------|---------------------|---|
| t+12 | + | 00 | m | α | |
| t+13 | + | 00 | $\langle 6 \cdot 2^{-30} \rangle$ | q+w | |
| t+14 | +, | 10 | δ | γ | |
| t+15 | U1 | 24 | q | β | |
| t+16 | x | 03 | $\langle b \rangle$ | δ | |
| t+17 | ↓+ | 20 | m+1 | m+1 | $\longleftarrow \frac{k_1}{2} + k_2$ |
| t+20 | + | 00 | $\langle 6 \cdot 2^{-30} \rangle$ | q+w | |
| t+21 | x, | 13 | $\langle \frac{1}{2} \rangle$ | δ | |
| t+22 | ↓+, | 30 | γ | - | |
| t+23 | U1 | 24 | q | β | |
| t+24 | x | 03 | $\langle b \rangle$ | δ | |
| t+25 | ↓+ | 20 | m+1 | m+1 | $\longleftarrow \frac{k_1}{2} + k_2 + k_3$ |
| t+26 | + | 00 | m | α | |
| t+27 | + | 00 | $\langle 6 \cdot 2^{-30} \rangle$ | q+w | |
| t+30 | +, | 10 | δ | γ | |
| t+31 | U1 | 24 | q | β | |
| t+32 | x, | 13 | m | δ | |
| t+33 | ↓+, | 30 | m+1 | | |
| t+34 | ↓x, | 33 | $\langle \frac{1}{3} \rangle$ | | |
| t+35 | ↓+ | 20 | γ | β | $\left. \begin{array}{l} \longleftarrow y_i = y_{i-1} + k \\ y_i \text{ kinyomtatása} \end{array} \right\}$ |
| t+36 | Á | 05 | β | δ | |
| t+37 | +, | 10 | t+44 | t+37 | |
| t+40 | U1 | 24 | r | r+j | |
| t+41 | - , | 51 | α | $\langle b \rangle$ | |
| t+42 | FU | 34 | t+43 | t+4 | |
| t+43 | megállási utasítás | | | | |
| t+44 | | 64 | 0000 | 0002 | |

Ez a program a számítás kezdetén megválasztott h lépésközzel működik. Elkészíthető olyan program is, amely automatikusan választja a lépésközt. A lépésköz automatikus változtatása kétféleképpen történhet.

1. Az egyik lehetőség: minden lépésnél megvizsgáljuk a

$$\left| \frac{k_2 - k_3}{k_1 - k_2} \right|$$

hányadost.

Ha

$$\left| \frac{k_2 - k_3}{k_1 - k_2} \right| < \varepsilon \quad (\varepsilon=0,01),$$

akkor a szereplő h lépésköz jónak tekinthető. Ellenkező esetben felezzük a lépésközt, s ezzel a felezett lépésközzel számolunk tovább.

2. A lépésköz automatikus megválasztásának másik lehetősége:

Adott h -val kiszámítjuk az y_i értéket (ezt jelöljük $y_i^{(h)}$ -val), majd felezzük a lépésközt, s kiszámítjuk a megoldás közelítő értékét az $x_{i-1} + \frac{h}{2}$ helyen; ennek segítségével mint intervallum-kezdőértékkel ismét kiszámítjuk újra y_i -t; jelöljük ezt $y_i^{\frac{h}{2}}$ -lel.

Összehasonlítjuk a két lépésben számított $y_i^{\frac{h}{2}}$ értéket az egy lépésben nyert y_i értékkel, pontosabban képezzük különbségüket.

Ha

$$\left| y_i^{(h)} - y_i^{(\frac{h}{2})} \right| \leq \varepsilon, \text{ ahol } \varepsilon \text{ a közelítés pontossá-}$$

ga ($\varepsilon = 15 \frac{\Delta}{N}$, ahol Δ a pontos értékből való

eltérés: előre adott, N a lépések száma,) akkor $y_i^{\frac{h}{2}}$ -t

helyes értéknek fogadjuk el, s a számítást h lépés-

közzel folytatjuk tovább. Ha $|y_i^{(h)} - y_i^{\frac{h}{2}}| > \varepsilon$, akkor $\frac{h}{2}$ -det

vesszük egész lépésköznek, s az előző eljárást ezzel

ismételjük meg.

10.2.2. A differenciamódszerek.

Az $y' = f(x,y)$ differenciálegyenlet numerikus integrálására szolgáló differenciamódszerek a Runge-Kutta módszerhez hasonlóan szintén pontonként s az előző pontokban már ismert deriváltak segítségével állítják elő a megoldás közelítő értékeit, mégpedig a szóbanforgó pont körüli pontokban képezett differenciák segítségével. Közülük Milne módszerét ismertetjük részletesen.

Az $y(x_0) = y_0$ kezdeti feltétellel adott $y' = f(x,y)$ differenciálegyenletnek Milne módszer szerinti numerikus integrálása a következő módon történik:

A kezdeti feltétel mellett meg kell adni (valamilyen más módszerrel ki kell számítani) az y_1, y_2, y_3 értékeket.

Az $y_{i-3}; y_{i-2}; y_{i-1}; y_i$ értékek segítségével az

$$y_{i+1}^{(1)} = y_{i-3}; + \left(\frac{8h}{3} f_i - \frac{4h}{3} f_{i-1} + \frac{8h}{3} f_{i-2} \right) \quad (1)$$

képlet alapján kiszámítjuk y_{i+1} első közelítését.

Ezzel az első közelítéssel kiszámítjuk f_{i+1} -et; s y_{i+1} második, harmadik, stb., v -edik közelítését az

$$y_{i+1}^{(v)} = y_{i-1} + h \left(\frac{1}{3} f_{i+1}^{(v-1)} + \frac{4}{3} f_i + \frac{1}{3} f_{i-1} \right) \quad (2)$$

képlettel számítjuk ki. Az iterációs eljárást a (2) képlet alapján addig folytatjuk, amíg

$$\left| y_{i+1}^{(v)} - y_{i+1}^{(v-1)} \right| < \varepsilon,$$

adódik, ahol ε előre adott. (Lásd: Békéssy András: "Közönséges differenciálegyenletek numerikus integrálása" című jegyzet 27-28. oldal, Mérnöki Továbbképző Intézet 2706).

Elkészítünk egy fixpontos programot a Milne módszerre:

Tegyük fel, hogy $|y| \leq K_0 < 1$, $|y'| < 1$, $|x| < 1$. Ha $K_0 = \frac{1}{2}$ -et választunk, akkor a közbülső eredmények sem csorbulnak túl.

Ugyanis:

$$\left| \frac{8}{3} h f_i - \frac{4}{3} h f_{i-1} + \frac{8}{3} h f_{i-2} \right| \leq \frac{20}{3} h < 7h$$

$$\left| y_{i+1}^{(1)} \right| \leq y_{i+3} + 7h;$$

ha tehát $|y_{i+3}| \leq \frac{1}{2} \leq 1 - 7$ (ha $h \leq \frac{1}{14}$; $n \geq 14$), akkor

$$\left| y_{i+1}^{(1)} \right| < 1. \text{ Ugyanígy:}$$

$$\begin{aligned} \left| y_{i+1}^{(v)} \right| &\leq \left| y_{i+1} \right| + \left| \frac{h}{3} f_{i+1}^{(v-1)} \right| + \frac{5h}{3} \leq \frac{6h}{3} + |y_{i+1}| < \\ &< 2h + |y_{i-1}| \leq 1, \end{aligned}$$

ha

$$|y_{i-1}| < 1 - 2h$$

Rekeszelosztás.

Legyen a $hf(x,y)$ -t kiszámító segédszubrutin az f -től $f+k$ -ig terjedő rekeszekben elhelyezve, s legyen ebben a szubrutinban:

x címe α
 y címe β
és $hf(x,y)$ címe δ .

Legyenek $m_1; m_1+1; m_1+2; m_1+3; m, m+1, m+2, M, K$ munkarekeszek.

Írjuk fel a számítás menetét.

A számításban előforduló operátorok a következők:

A₀ operátor: $y_0; y_1; y_2; y_3; x_0$ átvitele az $m_1; m_1+1; m_1+2; m_1+3; \alpha$ rekeszekbe.

F₀ operátor: $h f(x_0 + h; y_1) \Rightarrow m; h f(x_0 + 2h; y_1); \Rightarrow m+1;$
 $h f(x_0 + 3h; y_3) \Rightarrow m+2; x_0 + 3h \Rightarrow \alpha$

A₁ operátor: $y_{i+1}^{(1)}$ képzése:

azaz

$$\frac{8}{3} (m) + \frac{8}{3} (m+2) - \frac{4}{3} (m+1) + (m_1) \Rightarrow \beta$$

A₂ operátor: $y_{i-1} + \frac{4}{3} h f_i + \frac{1}{3} h f_{i-1}$ képzése,

azaz

$$\frac{1}{3} (m+1) + \frac{4}{3} (m+2) + (m_1+2) \Rightarrow K$$

H₀ operátor: $(\alpha) + h \implies \alpha$ (azaz: az x argumentum növe-
lése h -val).

F₁ operátor: $f((\alpha), (\beta)) \implies \delta$

nem
P₀ igen operátor: $|(M) - (\beta)| \leq \varepsilon$ teljesül-e?

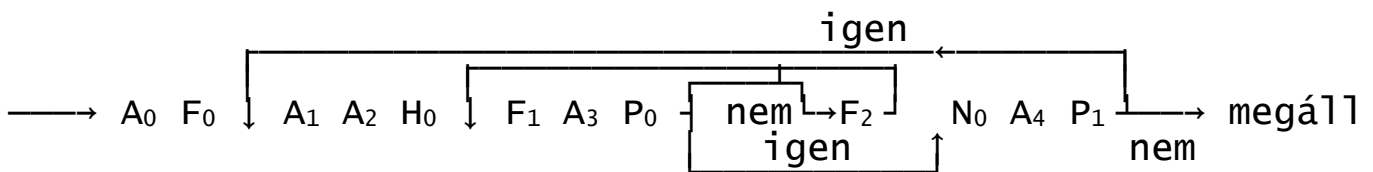
F₂ operátor: M átvitele β -ba

A₄ operátor: $(m_1+1) \implies m_1$ -be
 $(m_1+2) \implies m_1-1$ -be
 $(m_1+3) \implies m_1-2$ -be
 $(M) \implies m_1-3$ -ba
 $(m+1) \implies m$ -be
 $(m+2) \implies m+1$ -be
 $(\delta) \implies m+2$ -be

N₀ operátor: M kinyomtatása

igen
P₁ nem operátor: $|(\alpha)| \leq b$ teljesül-e?

A számítás menete:



A program a következő:

| | | | | | | |
|------|-----|----|-------------------------|-------------------|---|-------------------------|
| t | Á | 05 | <y ₀ > | m ₁ | } | A ₀ operátor |
| t+1 | Á | 05 | <y ₁ > | m ₁ +1 | | |
| t+2 | Á | 05 | <y ₂ > | m ₁ +2 | | |
| t+3 | Á | 05 | <y ₃ > | m ₁ +3 | | |
| t+4 | Á | 05 | <x ₀ > | α | | |
| t+5 | Á | 05 | m ₁ +1 | β | | |
| t+6 | Á | 05 | t+20 | f+k | | |
| t+7 | + | 00 | <h> | α | } | F ₀ operátor |
| t+10 | U2 | 74 | - | f | | |
| t+11 | Á | 05 | δ | m | | |
| t+12 | Á | 05 | m ₁ +2 | β | | |
| t+13 | + | 00 | <4 · 2 ⁻³⁰ > | f+k | | |
| t+14 | U2 | 74 | - | t+7 | | |
| t+15 | Á | 05 | δ | m+1 | | |
| t+16 | Á | 05 | m ₁ +3 | β | | |
| t+17 | U2 | 74 | - | t+13 | | |
| t+20 | U2 | 74 | - | t+11 | | |
| t+21 | Á | 05 | δ | m+2 | } | A ₁ operátor |
| t+22 | +, | 10 | m+2 | m | | |
| t+23 | ↓: | 22 | < $\frac{3}{8}$ > | K | | |
| t+24 | :, | 12 | < - $\frac{3}{4}$ > | m+1 | | |
| t+25 | ↓+, | 30 | K | β | } | A ₂ operátor |
| t+26 | ↓+ | 20 | m ₁ | | | |
| t+27 | x, | 13 | < $\frac{1}{3}$ > | m+1 | } | H ₀ operátor |
| t+30 | ↓+ | 20 | m ₁ +2 | K | | |
| t+31 | :, | 12 | < $\frac{3}{4}$ > | m+2 | | |
| t+32 | ↓+ | 20 | K | K | } | H ₀ operátor |
| t+33 | + | 00 | <h> | α | | |

| | | | | | |
|------|-------|----|-------------------|-------------------|------------------------------------|
| t+34 | +, | 10 | <64 0000 0002> | t+34 | } F ₁ operátor |
| t+35 | U1 | 24 | f | f+k | |
| t+36 | x, | 13 | < $\frac{1}{3}$ > | δ | } A ₃ operátor |
| t+37 | ↓+ | 20 | K | M | |
| t+40 | ↓ - , | 71 | β | | } P ₀ operátor |
| t+41 | ↓ - , | 71 | <ε> | | |
| t+42 | FU | 34 | t+45 | t+43 | |
| t+43 | Á | 05 | M | β | |
| t+44 | U2 | 74 | - | f | } F ₂ operátor |
| t+45 | Á | 05 | M | S | |
| t+46 | +, | 10 | <64 0000 0002> | t+46 | } kinyomtatás (N ₀) |
| t+47 | U1 | 24 | r | r+1 | |
| t+50 | Á | 05 | m ₁ +1 | m ₁ | } A ₄ operátor |
| t+51 | Á | 05 | m ₁ +2 | m ₁ +1 | |
| t+52 | Á | 05 | m ₁ +3 | m ₁ +2 | |
| t+53 | Á | 05 | M | m ₁ +3 | |
| t+54 | Á | 05 | m+1 | m | |
| t+55 | Á | 05 | m+2 | m+1 | |
| t+56 | -, | 11 | α | | } P ₁ operátor |
| t+57 | FU | 34 | t+60 | t+21 | |
| t+60 | | | | | |

A differenciamódszereknek nagy hátrányuk, hogy a számítás elkezdéséhez m+1 kezdő értékre van szükség (m a legmagasabb felhasznált differencia fokszáma), ezeket valamilyen más módszerrel előzőleg meg kell határozni.

10.2.3. Elsőrendű közönséges differenciálegyenlet-rendszerek.

Az elsőrendű közönséges differenciálegyenlet-rendszerek numerikus integrálása a közönséges elsőrendű differenciálegyenletek numerikus integrálására vonatkozó ismert módszerek bármelyikével elvégezhető, mégpedig úgy, hogy az adott módszert a rendszer minden egyes egyenletére egyszerre alkalmazzuk. Legalkalmasabbnak látszik a Runge-Kutta módszer.

10.3. Parciális differenciálegyenletek.

Az itt adódó problémák egyedi bánásmódot igényelnek. Az alkalmazandó numerikus módszer kiválasztása után a programozás feladatát az ismertetett eljárások alapján oldjuk meg.

F. FÜGGELÉK.

F. 1. Az M-3 gép kezelése.

Az alábbiakban röviden összefoglaljuk; azokat a kezelési utasításokat, amelyek segítségével a programozó által elkészített - papíron számok formájában felírt - programot a géppel végrehajtatjuk. Ehhez mindenekelőtt meg kell ismerkednünk a gép vezérlő pultjával (lásd a XIX. Táblán)

A vezérlőpulton egyrészt nyomógombok és kapcsolók, másrészt jelzőlámpák vannak elhelyezve. Az előbbiek segítségével lehetőségünk van kívülről beavatkozni a gép megfelelő regiszterébe, az utóbbiakon pedig az egyes regiszterek mindenkori tartalma látható.

A vezérlő pult felső részén a C, SZR, PR regiszterek lámpái és nyomógombjai felett lévő számok jelzik az ábrázolt számok "súlyát" a nyolcas számrendszerben. Pl.: (*-vel jelölve az égő lámpát - illetve a benyomott gombot, o-vel a nem égő lámpát, illetve a nem benyomott gombot). - A

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 |
| o | * | * | * | o | * | * | * | o | o | o | * |

állapot a 3561 számnak felel meg.

A vezérlő pult nyomógombjai és kapcsolói szerepét a gép működése közben a XIX. táblán látható elnevezések szerint ismertetjük.

F.1.1. Az adatok lyukasztása és bevitele.

Említettük, hogy az M-3 gépen az adatok (utasítások, kiinduló számadatok) bevitele lyukszalag segítségével történik. Az elkészített program realizálásának első lépése tehát az adatoknak lyukszalagra történő perforálása. Ez egy bármilyen perforátorral ellátott normál távgépíróval elvégezhető (Siemens). A (') billentyű a "cím vége", a (=) billentyű a "szó vége" jelet perforálja.

A számok bevitele történhetik cím megadásával vagy program segítségével. Az előző esetben az adatokat a következőképpen lyukasztjuk:

1. Beütjük annak a rekesznek a címét, ahova a szóbanforgó adatot el akarjuk helyezni.
2. Beütjük a "cím vége" billentyűt.
3. Beütjük az adatot (utasítás vagy nyolcas számrendszerben beírt szám).
4. Beütjük a "szó vége" jelet (=).

A perforátor a billentyűknek megfelelő jeleket a szalagra lyukasztja, és egyúttal, mint írógép, ki is írja a beütött jeleket. Ezáltal ellenőrizhető a beütés helyessége. A fenti négy lépés után pl. a 0006'+ 20 0017 0022 = kifejezés jelenik meg a lapon, amely tehát azt jelenti, hogy a gép a + 20 0017 0022 utasítást (vagy számot) a 0006 rekeszbe fogja beírni. Ha szó vége jelet nem írunk, akkor a szóbanforgó szám nem íródik a memóriába. Ilyen módon érvényteleníthetjük az esetleg tévesen beütött számokat.

Ezután a bevitelet a következő módon hajtjuk végre:

1. A "Folyamatos-Lépésenként" kapcsolót „Lépésenként” helyzetbe állítjuk.

2. A "Bevitel 8-10" kapcsolót "8" helyzetbe állítjuk (minthogy a cím megadásával való bevitel csak nyolcas számrendszerben lehetséges).

3. Attól függően, hogy a bevitelt lépésenként vagy folyamatosan akarjuk végrehajtani, a "Bevitel Lépésenként-Folyamatos" kapcsolóját a megfelelő állásba hozzuk, (Lépésenkénti bevitelnél egy-egy szó bevitele után a gép megáll.)

4. Beindítjuk a transzmitter motorját a megfelelő kapcsoló segítségével.

5. A bevitel "Be" gombjának benyomása után a transzmitter elkezd a szalagon levő adatok beolvasását (bevitel jelzőlámpája kigyullad).

Azokat az adatokat, amelyeket valamely program végrehajtása során viszünk be (a Be 07 utasítással) cím és cím vége jel nélkül perforáljuk. Pl.: A +0,01079 tízes számrendszerben felírt számot a + 0107900 alaknak megfelelően perforáljuk (a megfelelő billentyűk leütésével), és a szám után leütjük a szó vége jelet.

Ha olyan programot hajtunk végre, amelynek során bevitelt kell végeznünk (a programban szerepel a 07 utasítás), akkor a program elindítása előtt a következőket kell tenni.

1. Bekapcsoljuk a transzmitter motorját.
2. A "Bevitel 8-10" kapcsolót a megfelelő helyzetbe hozzuk.
3. A "Bevitel Folyamatos-Lépésenként" kapcsolót "Lépésenként" helyzetbe állítjuk.

Ily módon, amikor a gépben már bentlévő programban a végrehajtás során egy beviteli utasítás következik, (Be 07) akkor a gép beolvassa a szalagról a megfelelő adatot, és beírja az utasítás második címének megfelelő rekeszébe.

F.1.2. Számok beállítása a C, SZR, PR regiszterekbe.

A vezérlő pultra kivezetett regiszterekbe (C; SZR; PR.) a számokat nyomógombok segítségével is beállíthatjuk. Ez úgy történik, hogy az illető regiszter "Beállítás-Munka" kapcsolóját "Beállítás" helyzetbe hozzuk, a regiszter nyomógombjain benyomjuk a megfelelő számot. Az "Impulzus" gomb megnyomásával az adott szám beíródik a megfelelő regiszterbe, és leolvasható az illető regiszter jelzőlámpáin.

F.1.5. A program végrehajtása.

Tegyük fel, hogy a programot és a szükséges adatokat már bevittük. Akkor ahhoz, hogy a gép megkezdje a program végrehajtását, a következőket kell a vezérlő pulton elvégezni.

1. PR-be beállítjuk az első utasítás címét.
2. A SZR, PR, C kapcsolókat Munka helyzetbe hozzuk.
3. Az "RI törlés" gombbal töröljük a taktus számlálót.
4. Ha a programban nyomtatás is szerepel, akkor előkészítjük a gépet kivitelre.
 - a) A megfelelő kapcsolóval bekapcsoljuk a távgépíró motorját.
 - b) A "kivitel 8-10" kapcsolót a megfelelő helyzetbe állítjuk.
 - c) A "kivitel" kapcsolót munkahelyzetbe állítjuk.
5. A "Folyamatos-Lépésenként" kapcsolót "Folyamatos" helyzetbe állítjuk.
6. Ha ezek után az "RI indítás" gombot benyomjuk, a gép elkezdí a program végrehajtását.

F.1.4. A program lépésenkénti végrehajtása.

(A "Folyamatos-Lépésenkénti" kapcsoló "Lépésenként" helyzetében).

Mikor a program végrehajtását elkezdjük, célszerű az első néhány utasítást lépésenként végrehajtani. A gép működésének helyességét is ellenőrizhetjük az egyes utasítások lépésenkénti végrehajtásával. Egy adott utasítást három lépésben lehet végrehajtani:

1. Az "RI Indítás" gomb megnyomásával a gép végrehajtja az adott utasítás első és második taktusát, és utána leáll. A C regiszterben megjelenik az adott utasítás, az SZR-ben az utasítás címe.

2. Újra benyomva az "RI Indítás" gombot a gép végrehajtja a harmadik, negyedik, ötödik és hatodik taktust, majd megáll. A C regiszterben megjelenik a második cím tartalma és az SZR-ben maga a cím. A PR-ben megjelenik a soronkövetkező utasítás címe.

3. Ismét benyomva az "RI Indítás" gombot a gép végrehajtja az utasítást. A B és C regiszterben megjelenik a művelet eredménye.

F.1.5. Beírás a memóriába.

A vezérlő pulton lévő nyomógombok segítségével lehetőség van arra is, hogy a memóriába adatokat vigyünk be. Ez a következőképpen történik:

1. Beállítjuk a C regiszterbe a beviendő számot.

2. Beállítjuk az SZR regiszterbe azt a címet, ahova a számot írni akarjuk.

3. Az "Írás" gomb megnyomásával megtörténik a memóriába való bevétel.

F.1.6. Kihívás a memóriából.

A memóriába való beíráson kívül a memória egy adott rekesze tartalmának a kiolvasására is van lehetőség. Ez a következőképpen történik:

1. A "Folyamatos-Lépésenként" kapcsolót "Lépésenként" helyzetbe állítjuk.
2. Az SZR-ben beállítjuk a kiolvasandó rekesz címét.
3. Az "Olvasás" gomb megnyomásával a C regiszterben megjelenik az SZR-ben beállított cím tartalma.

F.1.7. Megállítás egy adott címnél.

A vezérlő pulton lévő kapcsolók segítségével lehetséges a program végrehajtása során a gépnek egy adott cím szerinti megállítása is. Ez a cím lehet egy utasítás vagy egy adat címe. A cím szerinti megállítással a gép helyes működését ellenőrizhetjük. A cím szerinti megállás beállítása a következőképpen történik:

1. A "Folyamatos-Megállás" kapcsolót "Megállás" helyzetbe hozzuk.
2. A vezérlő pult jobboldalán lévő kapcsolókon beállítjuk az adott címet.
3. Ha egy utasítás címe szerint akarjuk leállítani a gépet, akkor a "PR-SZR" kapcsolót PR helyzetbe hozzuk. Ha a program végrehajtása során PR-ben megjelenik az adott utasítás címe, akkor a gép leáll.
4. Ha egy adat címe szerinti megállást akarunk, akkor a "PR-SZR" kapcsolót SZR helyzetbe hozzuk. A gép megáll, ha a szóbanforgó adat címe (egy utasítás első vagy második címe) az SZR-ben megjelenik.

A vezérlőpulton kívül külön kapcsolótáblát találunk a memóriaegységen, a tápegységen és külön kapcsolókat az egyes kereteken is. Ezek azonban a program végrehajtása szempontjából közvetlenül nem játszanak szerepet.

Ha a gép működés közben megáll, a megállás oka a regiszterek jelzőlámpáin leolvasható; mégpedig

a) ha a B'0-ban megállási utasítás van, akkor a megállás ennek következménye,

b) ha összeadás és osztás esetén a D regiszter nulladik helyértékén 1-es áll, akkor a megállás túlcsoordulás miatt következik be,

c) minden más esetben a megállás program vagy gép-hiba következménye. Ilyen esetben célszerű az indítóregiszterben lévő cím előtti címen található utasítást megvizsgálni.

F.2. Az M-3 utasításrendszere.

| Kód: | Jel: | Magyarázat: |
|------|-------|---|
| 00 | + | Az első cím tartalmához hozzáadódik a második cím tartalma és beíródik a memóriába a második címre. |
| 10 | +, | Az első cím tartalmához hozzáadódik a második cím tartalma, de a memóriába nem íródik be. |
| 20 | ↓+ | Az előző művelet eredményéhez hozzáadódik az első cím tartalma, és az eredmény beíródik a memóriába a második címre. |
| 30 | ↓+, | Az előző művelet eredményéhez hozzáadódik az első cím tartalma, de a memóriába nem íródik be. |
| 40 | + n | Az első cím tartalmához hozzáadódik a második cím tartalma. Az eredmény beíródik a memóriába a második címre, és kinyomtatódik. |
| 50 | + , | Az első cím tartalmának abszolút értékéhez hozzáadódik a második cím tartalmának abszolút értéke, de a memóriába nem íródik be. |
| 60 | ↓+ n | Az előző művelet eredményéhez hozzáadódik az első cím tartalma. Az eredmény beíródik a memóriába a második címre, és kinyomtatódik. |
| 70 | ↓ + , | Az előző művelet eredményének abszolút értékéhez hozzáadódik az első cím tartalmának abszolút értéke, de a memóriába nem íródik be. |
| 01 | - | A második cím tartalmából kivonódik az első cím tartalma, és az eredmény beíródik a memóriába a második címre. |

| Kód: | Jel: | Magyarázat: |
|------|-------|---|
| 11 | -, | A második cím tartalmából kivonódik az első cím tartalma, de a memóriába nem íródik be - az eredmény megmarad az aritmetikai egységben. |
| 21 | ↓- | Az előző művelet eredményéből kivonódik az első cím tartalma, az eredmény beíródik a memóriába a második címre. |
| 31 | ↓-, | Az előző művelet eredményéből kivonódik az első cím tartalma, az eredmény nem íródik be. |
| 41 | - n | A második cím tartalmából kivonódik az első cím tartalma, az eredmény beíródik a második címre és kinyomtatódik. |
| 51 | - , | A második cím tartalmának abszolút értékéből kivonódik az első cím tartalmának abszolút értéke, az eredmény nem íródik le. |
| 61 | ↓- n | Az előző művelet eredményéből kivonódik az első cím tartalma. Az eredmény beíródik a második címre és kinyomtatódik. |
| 71 | ↓ - , | Az előző művelet eredményének abszolút értékéből kivonódik az első cím tartalmának abszolút értéke, de a memóriába nem íródik be. |
| 02 | : | A második cím tartalma elosztódik az első cím tartalmával. Az eredmény beíródik a memóriába a második címre. |
| 12 | :, | A második cím tartalma elosztódik az első cím tartalmával. Az eredmény a memóriába nem íródik be. |
| 22 | ↓: | Az előző művelet eredménye elosztódik az első cím tartalmával, és az eredmény beíródik a memóriába a második címre. |
| 32 | ↓:, | Az előző művelet eredménye elosztódik az első cím tartalmával. Az eredmény a memóriába nem íródik be. |

| Kód: | Jel: | Magyarázat: |
|------|-------|---|
| 42 | : n | A második cím tartalma elosztódik az első cím tartalmával. Az eredmény beíródik a memóriába a második címre és kinyomtatódik. |
| 52 | : | A második. cím tartalmának abszolút értéke elosztódik az első cím tartalmának abszolút értékével, de a memóriába nem íródik be. |
| 62 | ↓: n | Az előző művelet eredménye elosztódik az előző cím tartalmával. Az eredmény beíródik a memóriába a második címre és kinyomtatódik. |
| 72 | ↓ : , | Az előző művelet eredményének abszolút értéke elosztódik az első cím tartalmának abszolút értékével, de a memóriába nem íródik be. |
| 03 | x | A második cím tartalma összeszorozódik az első cím tartalmával. Az eredmény beíródik a memóriába a második címre. |
| 13 | x, | A második cím tartalma összeszorozódik az első cím tartalmával. Az eredmény a memóriába nem íródik be. |
| 23 | ↓x, | Az előző művelet eredménye összeszorozódik az első cím tartalmával. Az eredmény beíródik a memóriába a második címre. |
| 33 | ↓x | Az előző, művelet eredménye összeszorozódik az első cím tartalmával. Az eredmény a memóriába nem íródik be. |
| 43 | x n | A második cím tartalma összeszorozódik az első cím tartalmával. Az eredmény beíródik a memóriába a második címre és kinyomtatódik. |
| 53 | x , | A második cím tartalmának abszolút értéke összeszorozódik az első cím tartalmának abszolút értékével, de a memóriába nem íródik be. |

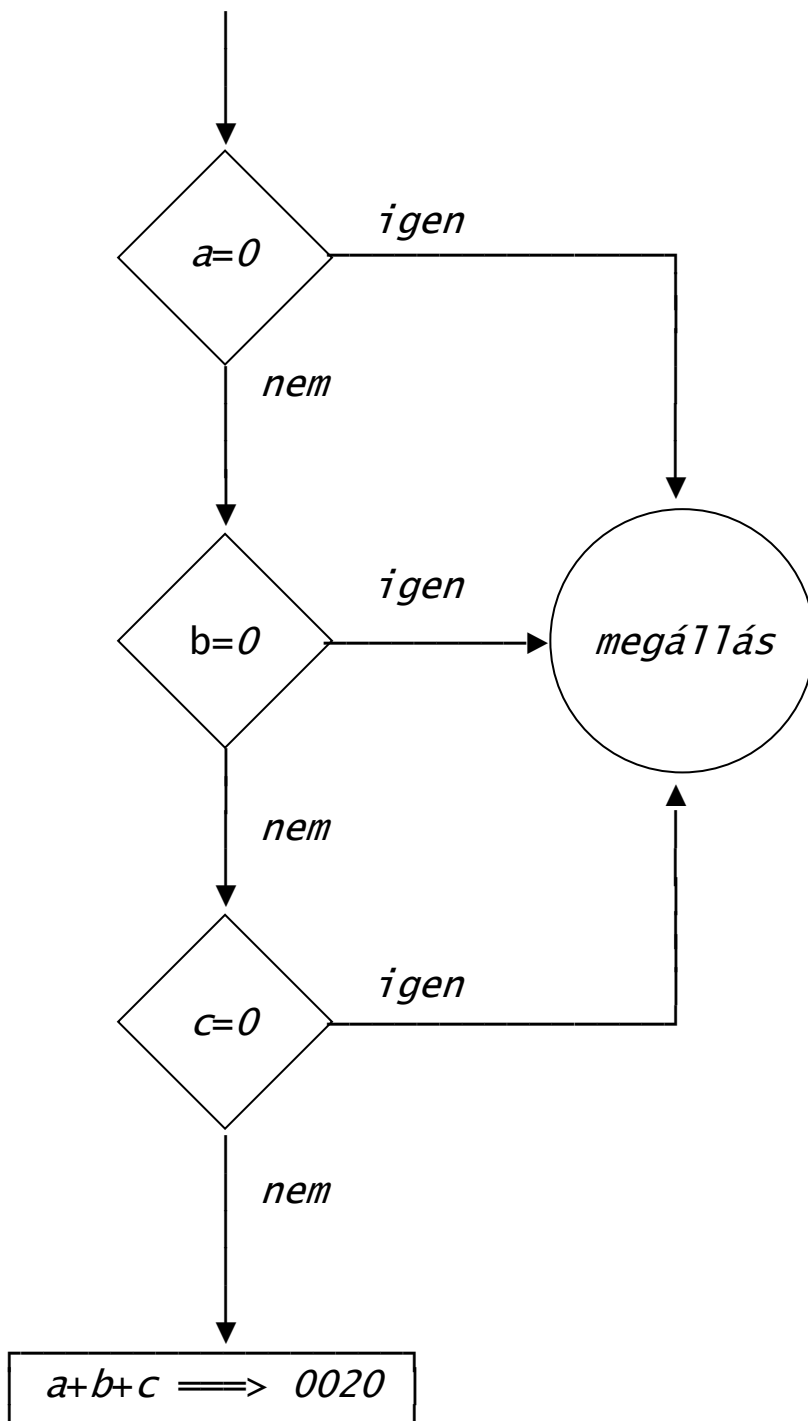
| Kód: | Jel: | Magyarázat: |
|------|------------------------|--|
| 65 | $\downarrow x n$ | Az előző művelete eredménye összeszorzódik az első cím tartalmával. Az eredmény beíródik a második címre és kinyomtatódik. |
| 73 | $\downarrow x ,$ | Az előző művelet eredményének abszolút értéke összeszorzódik az első cím tartalmának abszolút értékével, de a memóriába nem íródik be. |
| 06 | \wedge | A második cím tartalma logikailag összeszorzódik az első cím tartalmával, és az eredmény beíródik a memóriába a második címre. |
| 16 | $\wedge,$ | A második cím tartalma logikailag összeszorzódik az első cím tartalmával. Az eredmény megmarad az aritmetikai egységben. |
| 26 | $\downarrow \wedge$ | Az előző művelet eredménye logikailag összeszorzódik az első cím tartalmával és az eredmény beíródik a memóriába a második címre. |
| 36 | $\downarrow \wedge,$ | Az előző művelet eredménye logikailag összeszorzódik az első cím tartalmával. Az eredmény a memóriába nem íródik be. |
| 46 | $\wedge n$ | A második cím tartalma logikailag összeszorzódik az első cím tartalmával. Az eredmény beíródik a második címre és kinyomtatódik. |
| 56 | $ \wedge ,$ | A második cím tartalmának abszolút értéke logikailag összeszorzódik az első cím tartalmának abszolút értékével. Az eredmény a memóriába nem íródik be. |
| 66 | $\downarrow \wedge n$ | Az előző művelet eredménye logikailag összeszorzódik az első cím tartalmával. Az eredmény beíródik a memóriába a második címre és kinyomtatódik. |
| 76 | $\downarrow \wedge ,$ | Az előző művelet eredményének abszolút értéke logikailag összeszorzódik az első cím tartalmának abszolút értékével, de a memóriába nem íródik be. |

| Kód: | Jel: | Magyarázat: |
|--|----------|--|
| 07; 27 | Be | Szám bevitele. A szám a perforált szalagról beíródik a második címre. A regiszterben nem marad meg. |
| 05; 15 | Á | Az első cím tartalma átmegy a második címre és megmarad az aritmetikai egység kettes számú regiszterében. |
| 45; 55 | Án | Az első cím tartalma átmegy a második címre és kinyomtatódik, de az aritmetikai egységben nem marad meg. |
| 24 | U1 | Vezérlésátadás. A következő utasítást a gép az első címről veszi. Az előző művelet eredménye beíródik a második címre és megmarad az aritmetikai egységben. |
| 64 | U1n | Ugyanaz, mint a 24-es utasítás, de a szám még ki is nyomtatódik. |
| 74 | U2 | A következő utasítást a gép a második címről veszi. Az aritmetikai egységben megmarad az előző művelet eredményének abszolút értéke. |
| 34 | FU | Feltételes ugrás. A következő utasítást a gép a második címről veszi, ha az előző művelet eredménye pozitív, és az első címről, ha az előző művelet eredménye negatív. |
| 04 14 44 54 17 37 57 77 | megállás | A megállási utasítások az aritmetikai egység regisztereinek, valamint a kiválasztó és beindító regiszterek tartalmában különböznek egymástól. |

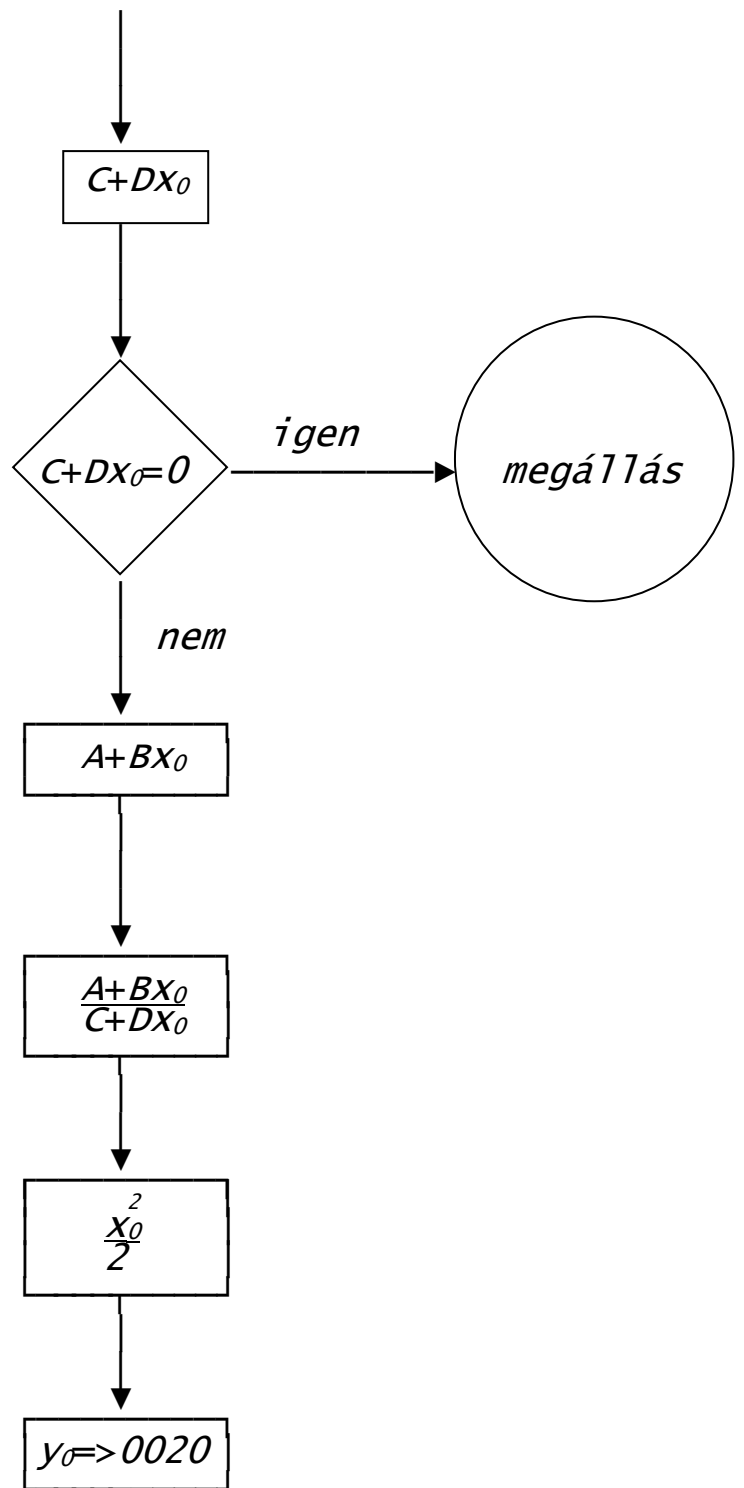
282/D/59.november 9.

V.Iné.

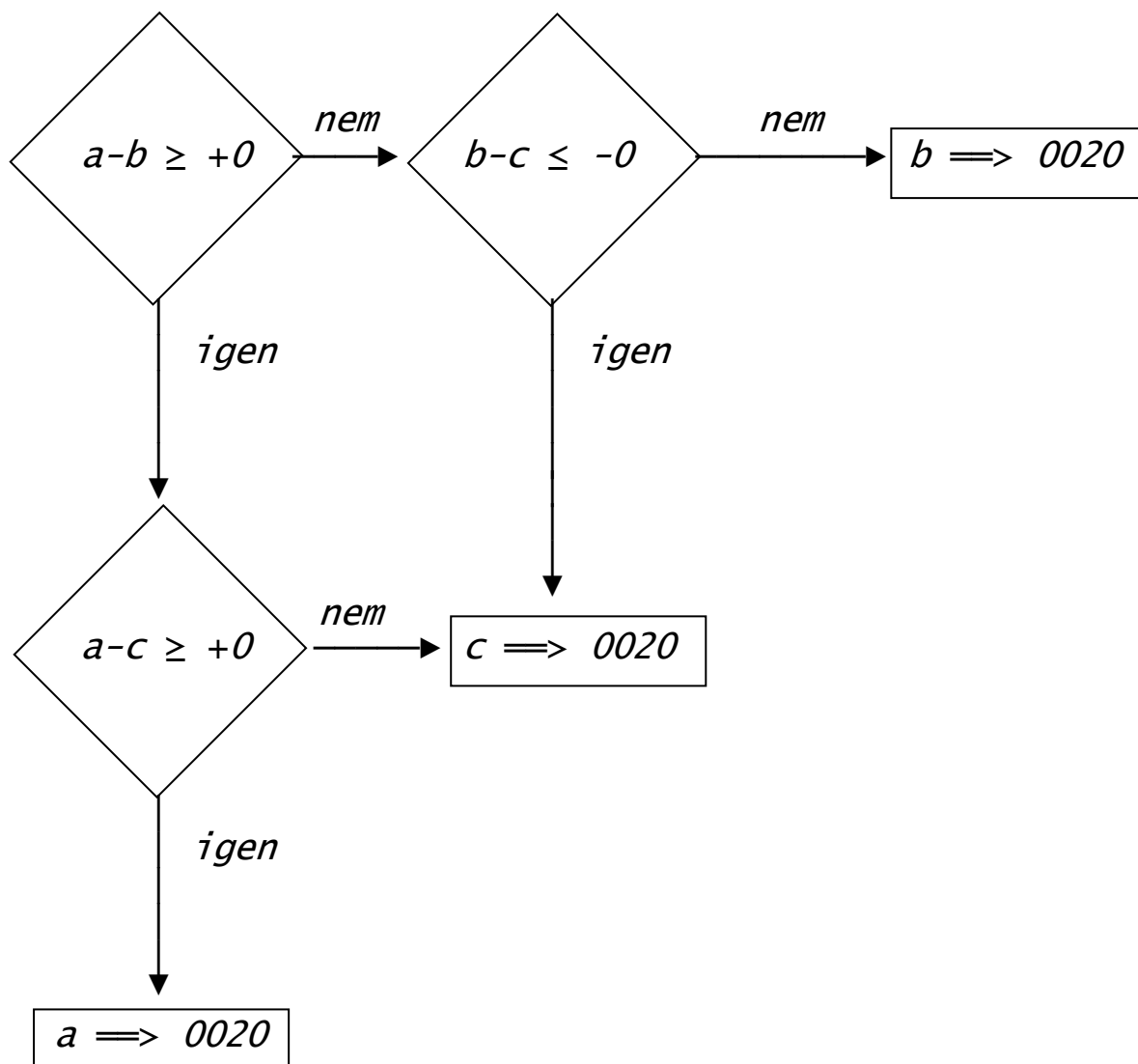
Készült: 150 pld-ban.



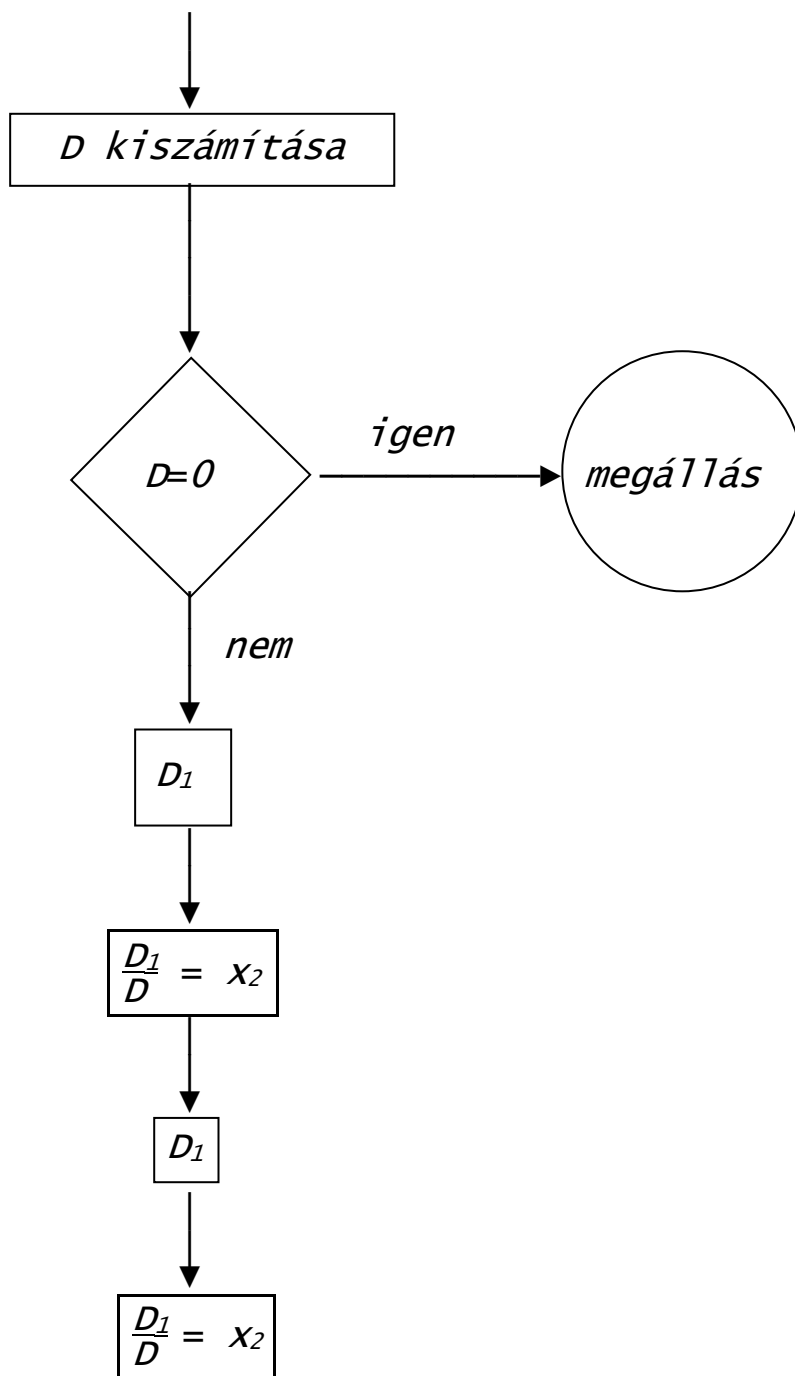
I. tábla



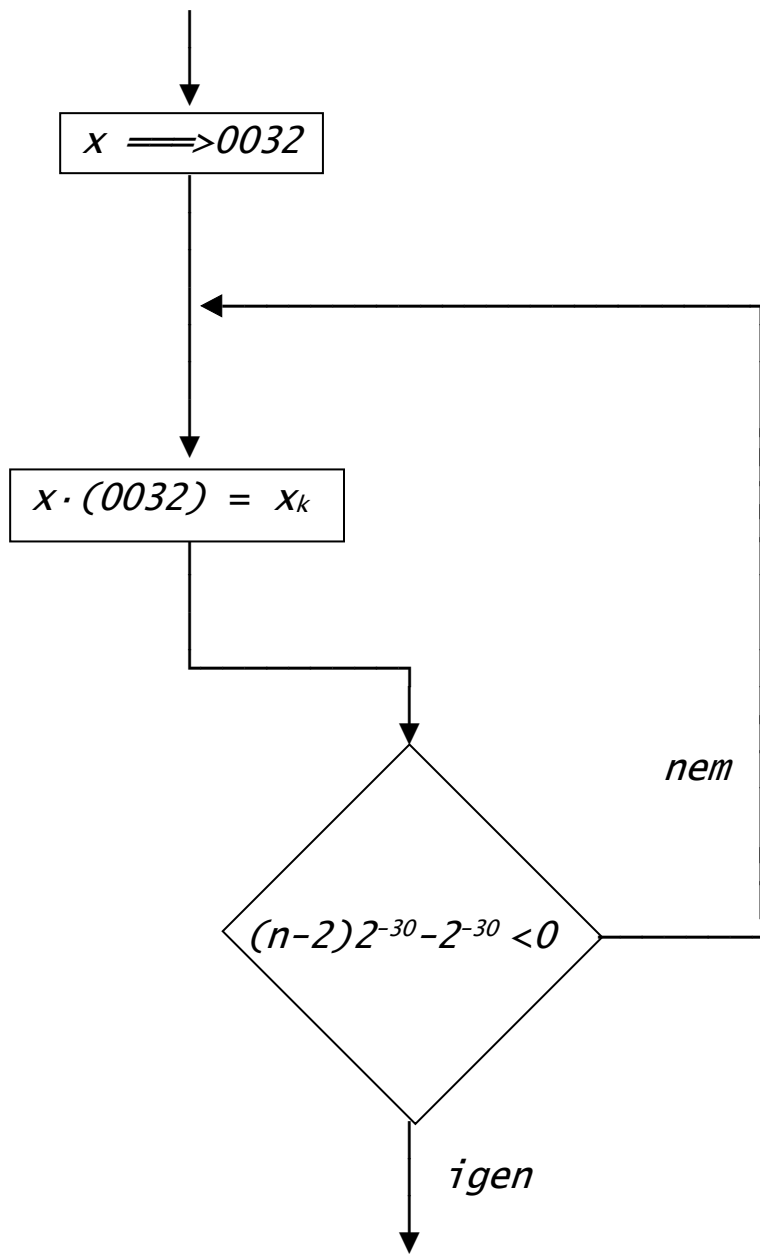
II. tábla



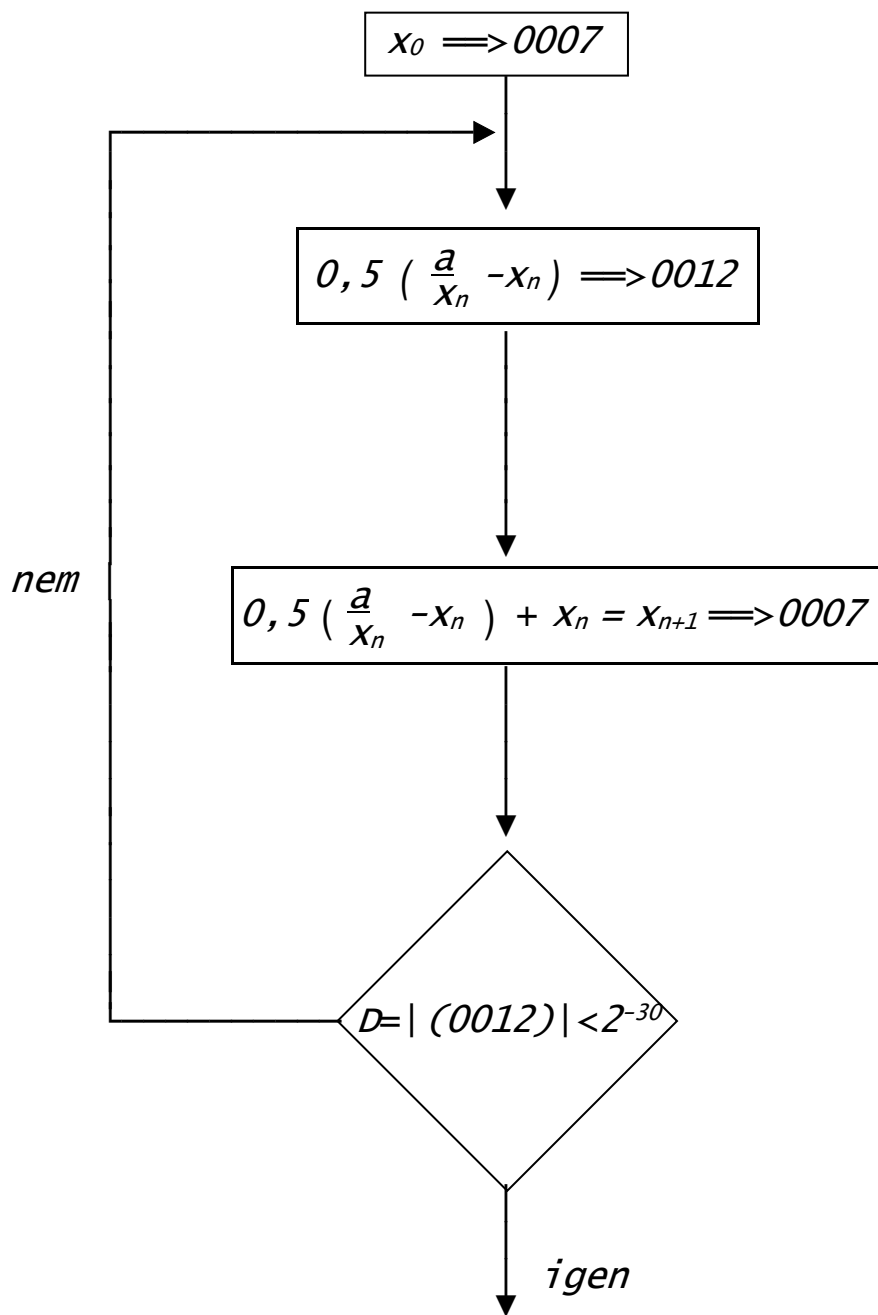
III. tábla



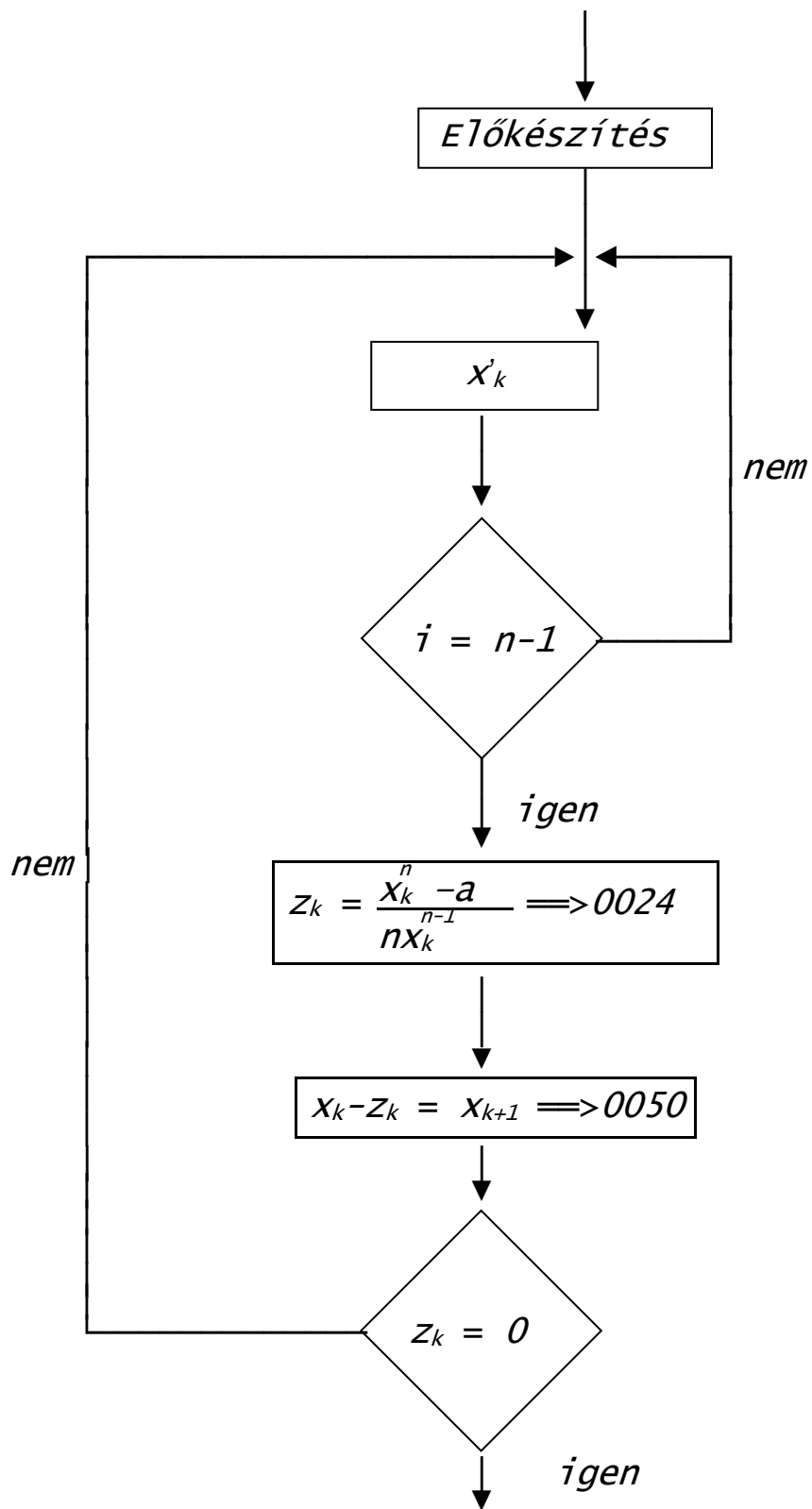
IV. tábla



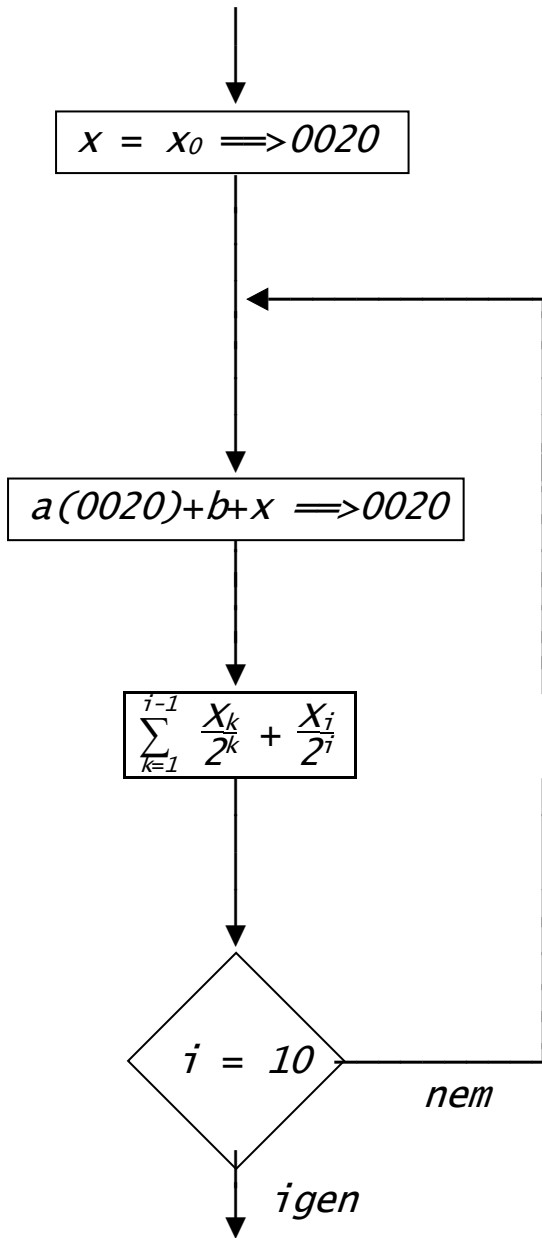
v. tábla



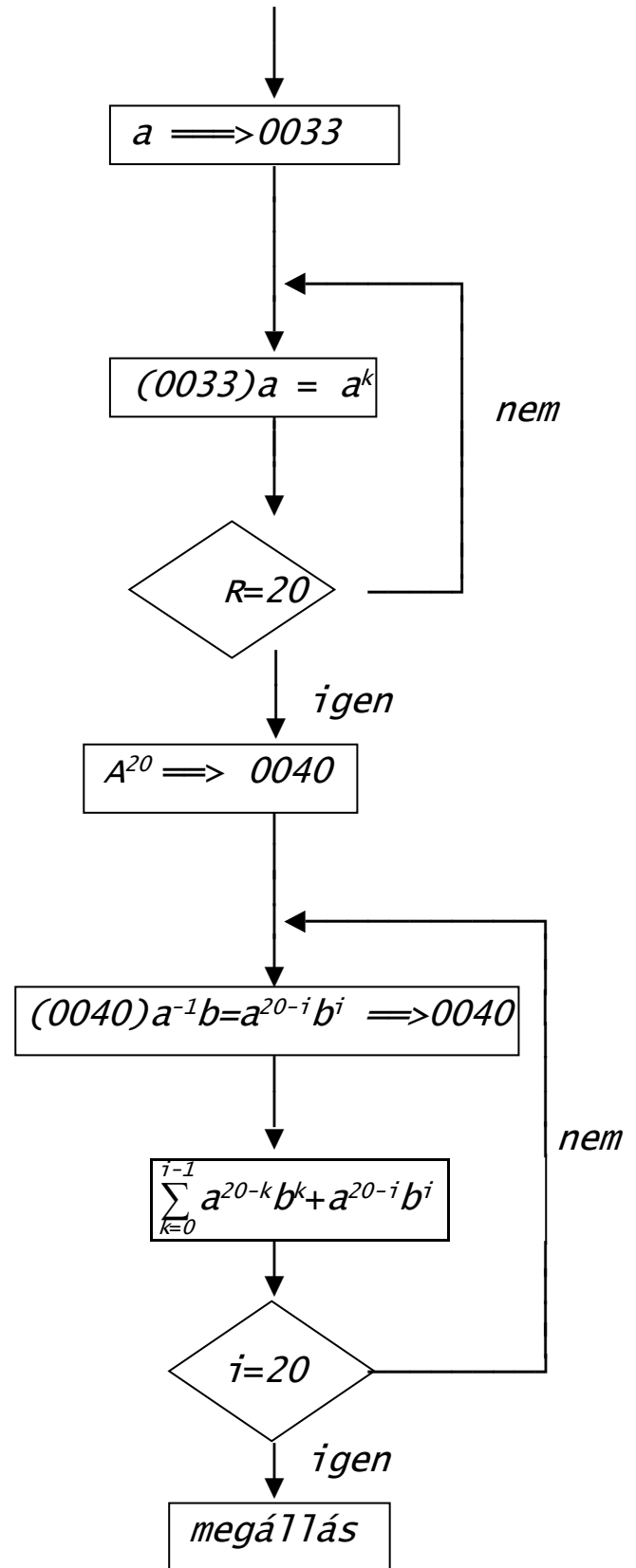
VI. tábla



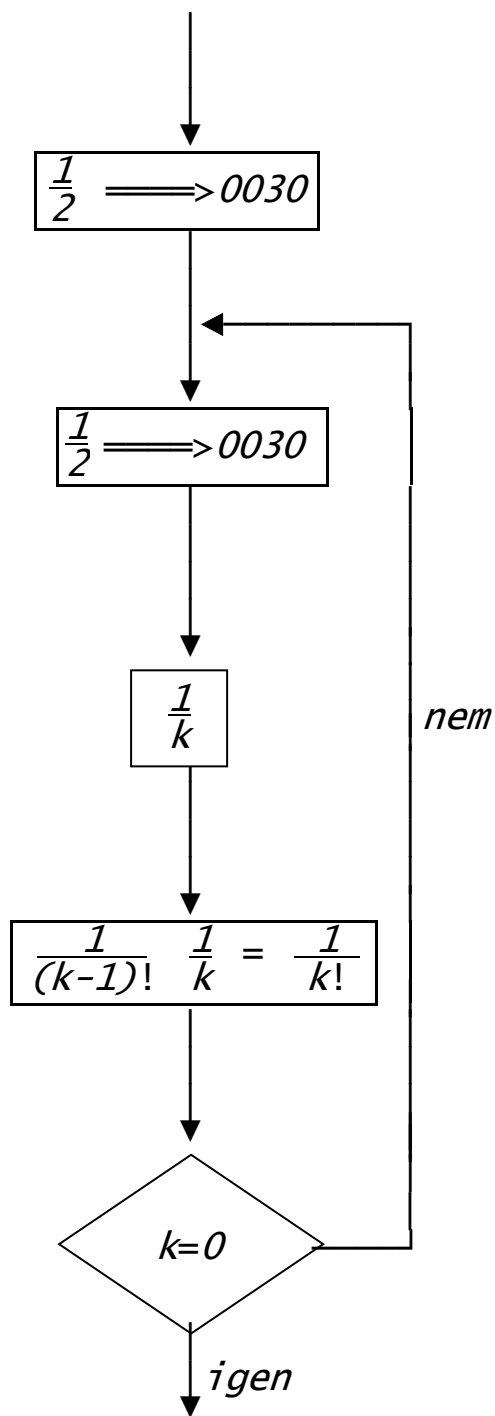
VII. tábla



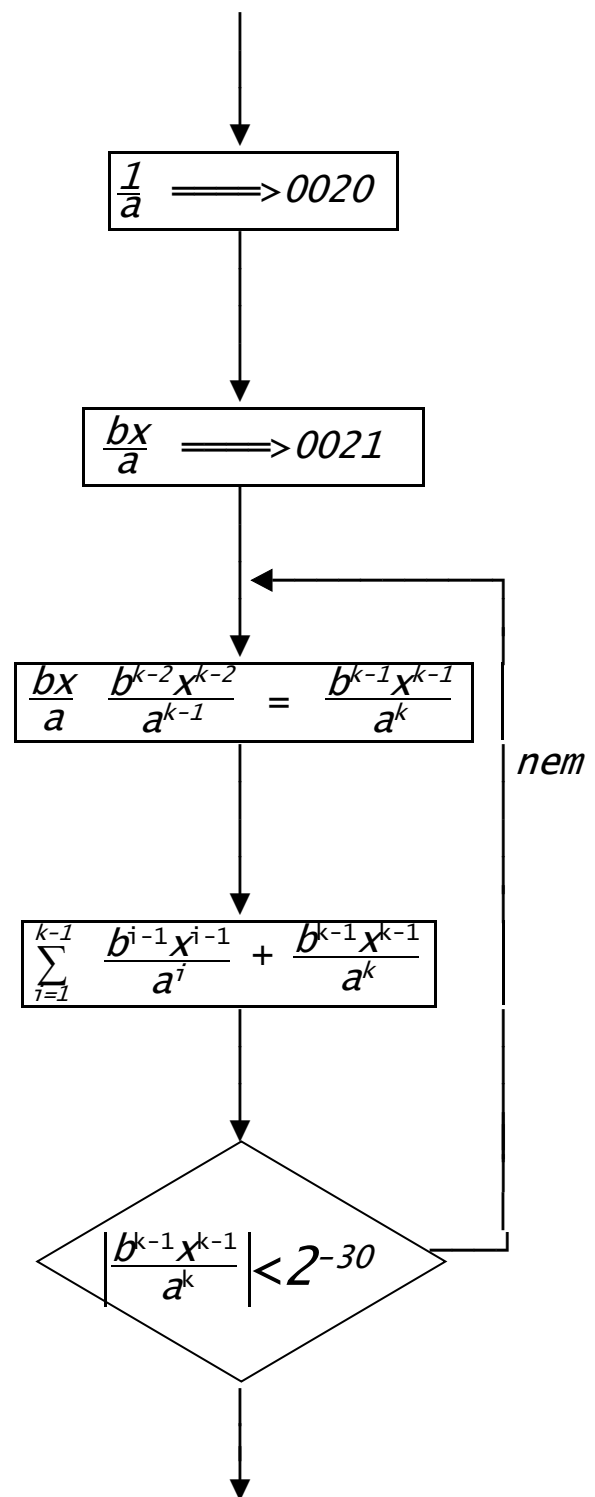
1. ábra



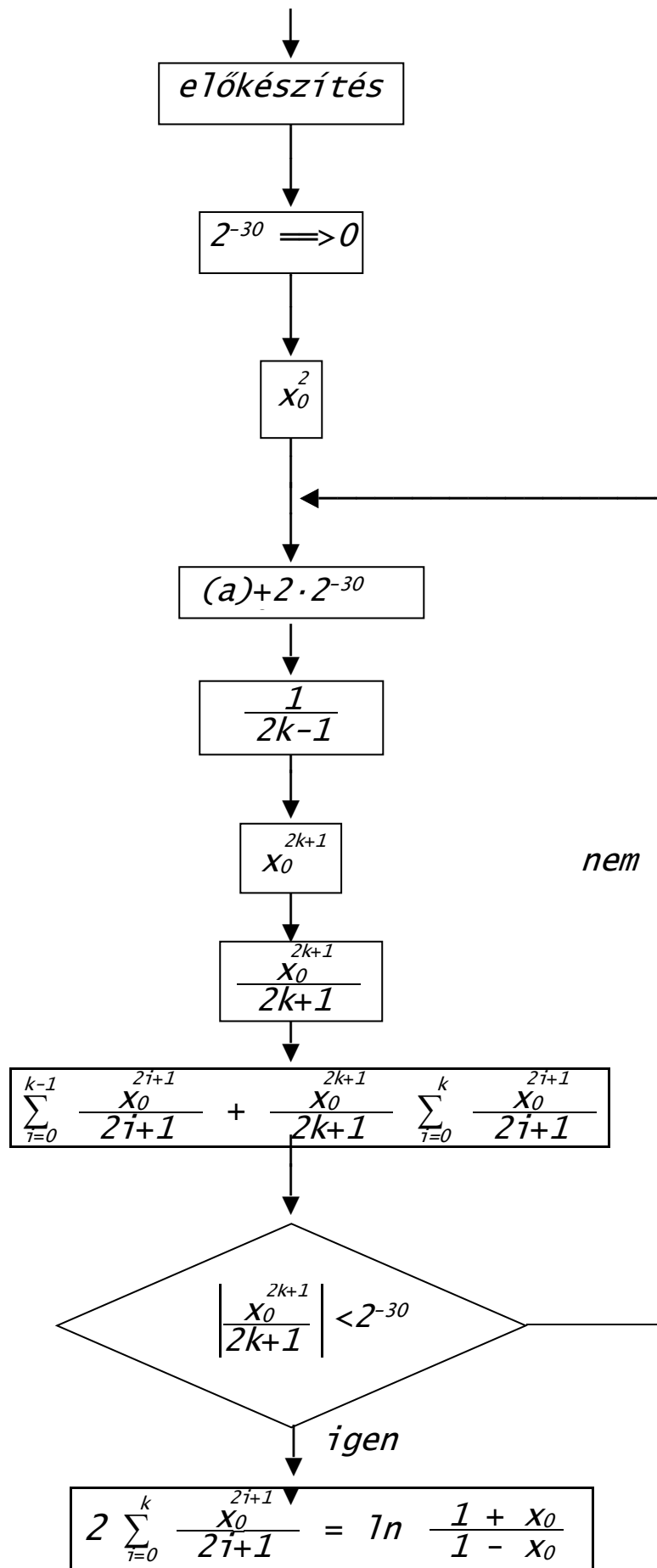
2. ábra



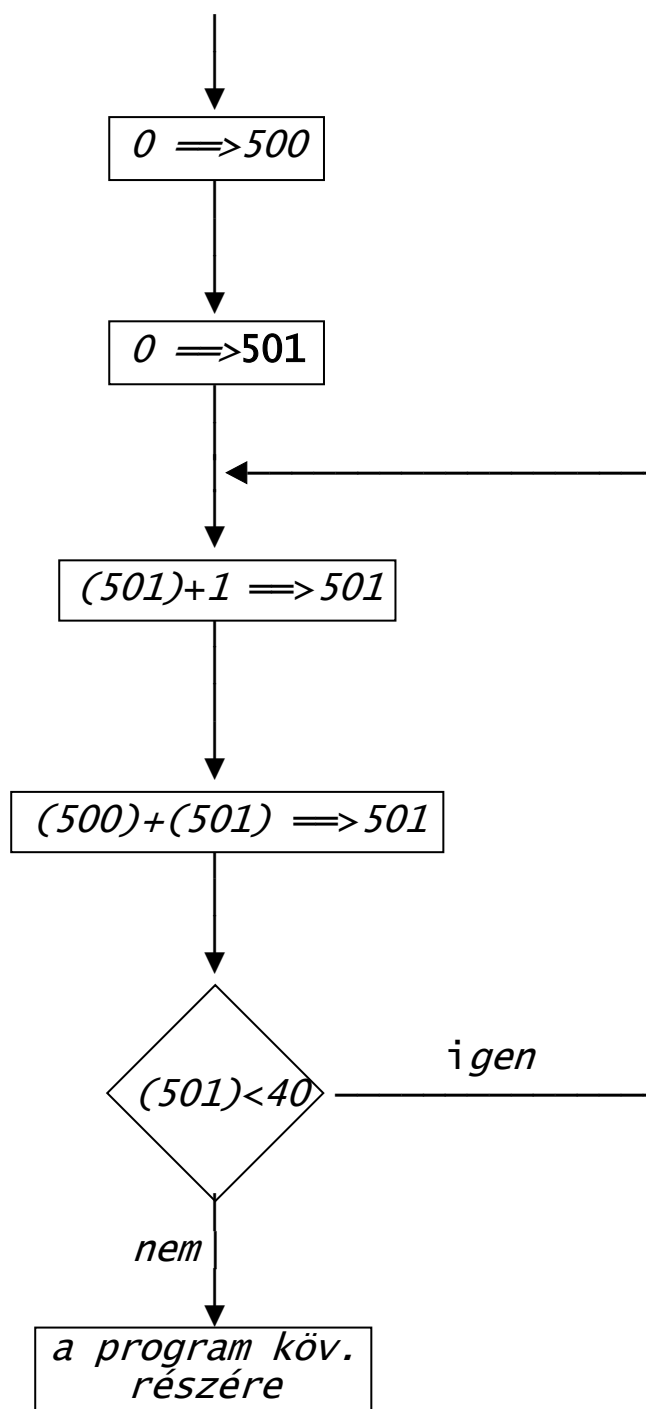
1. ábra



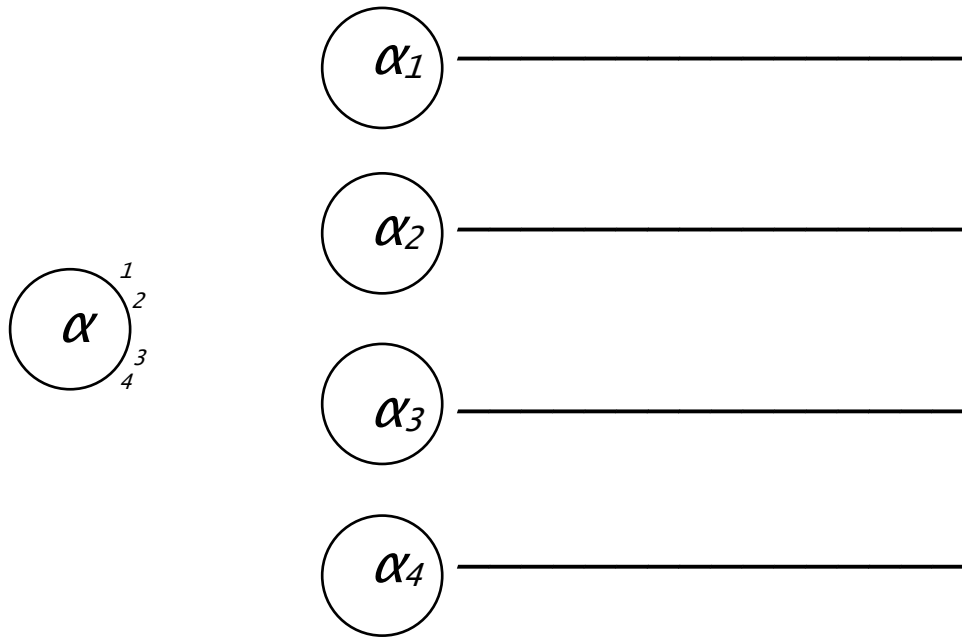
2. ábra



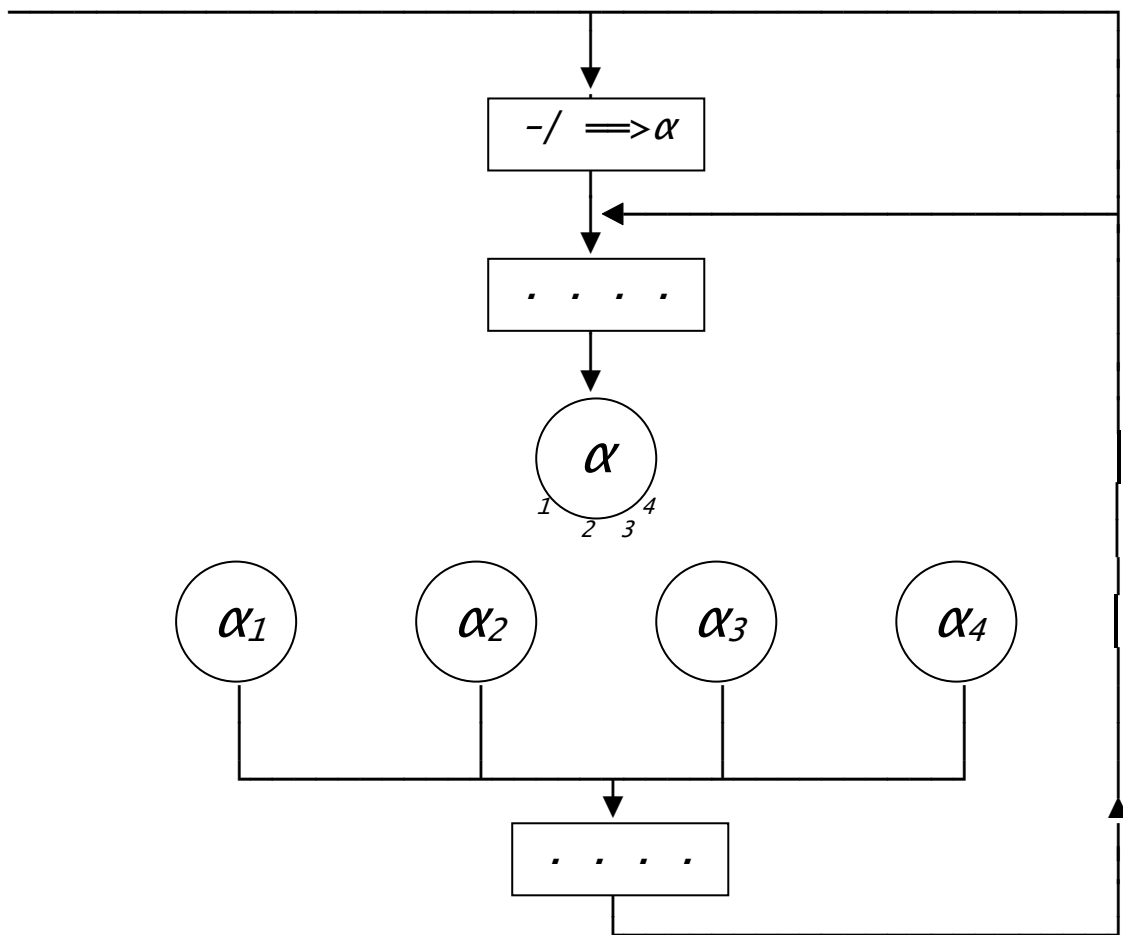
x. tábla



XI. tábla

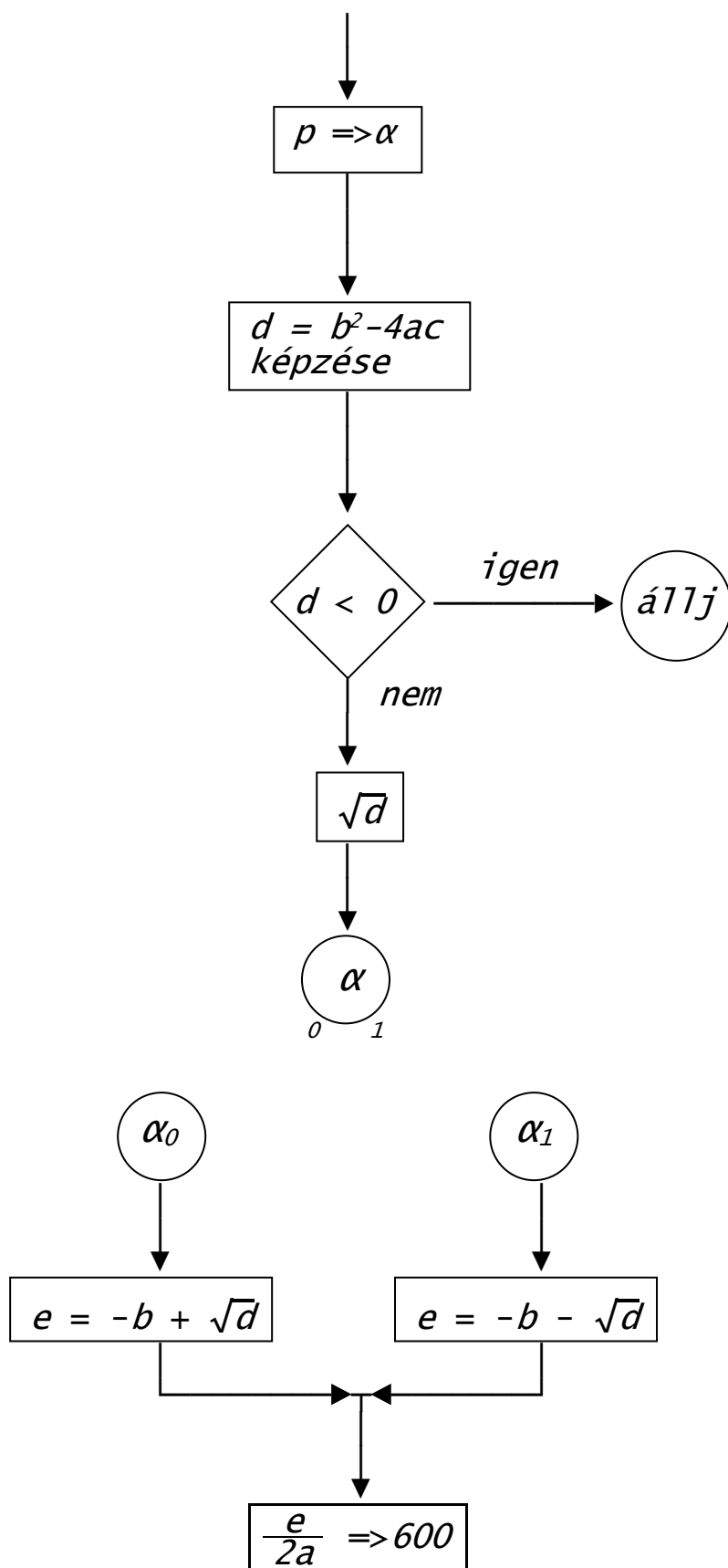


1. ábra

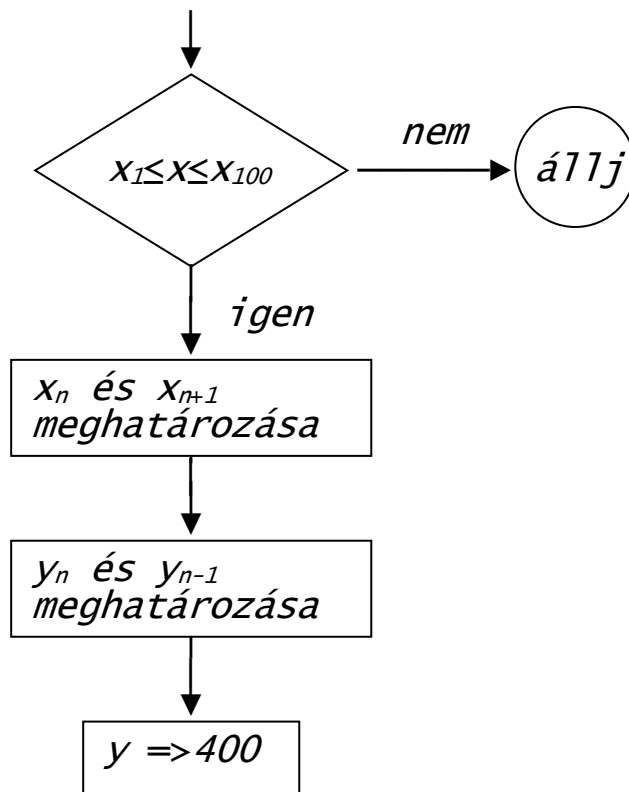


2. ábra

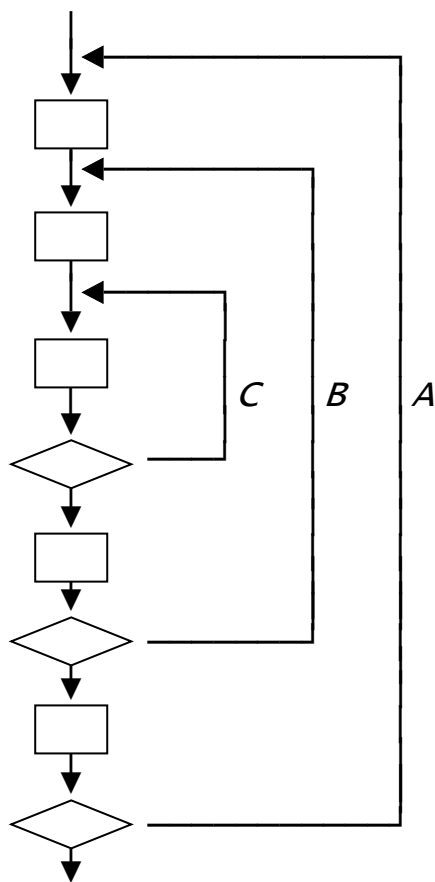
XII. tábla



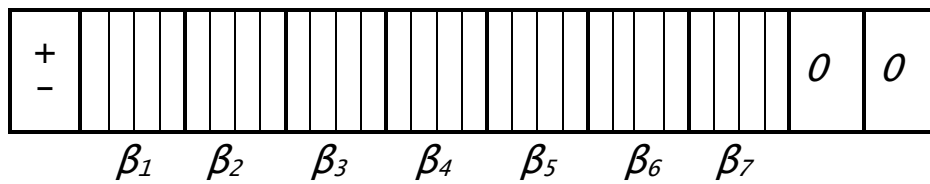
XIII. tábla



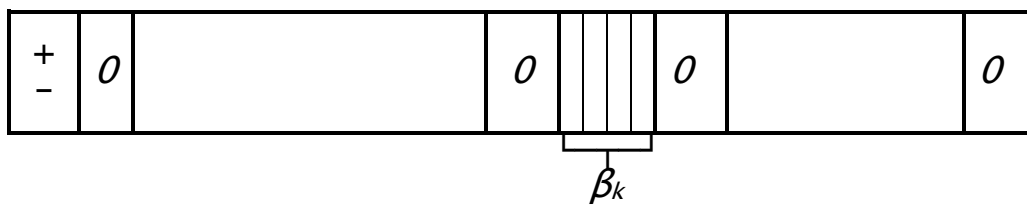
1. ábra



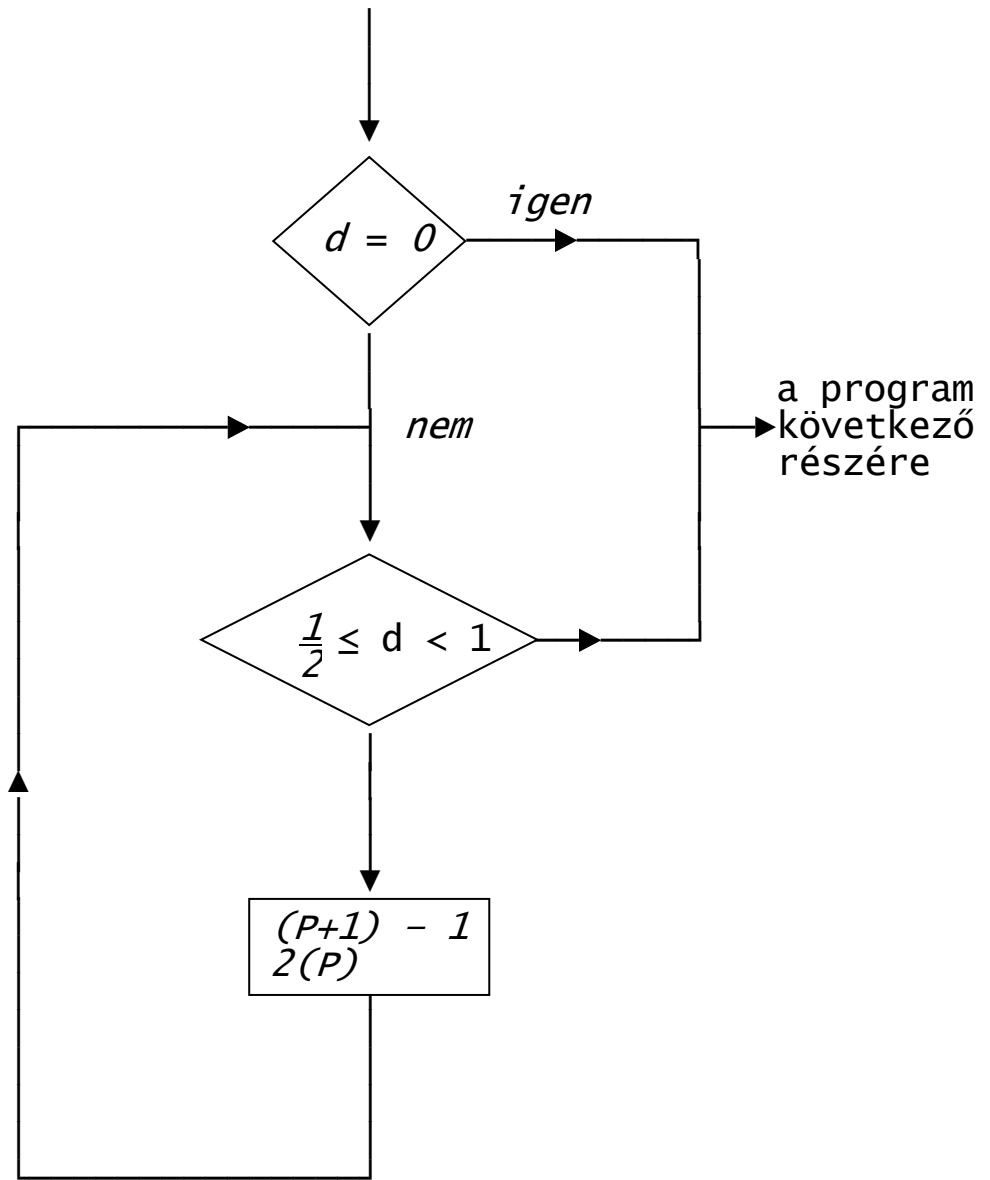
2. ábra



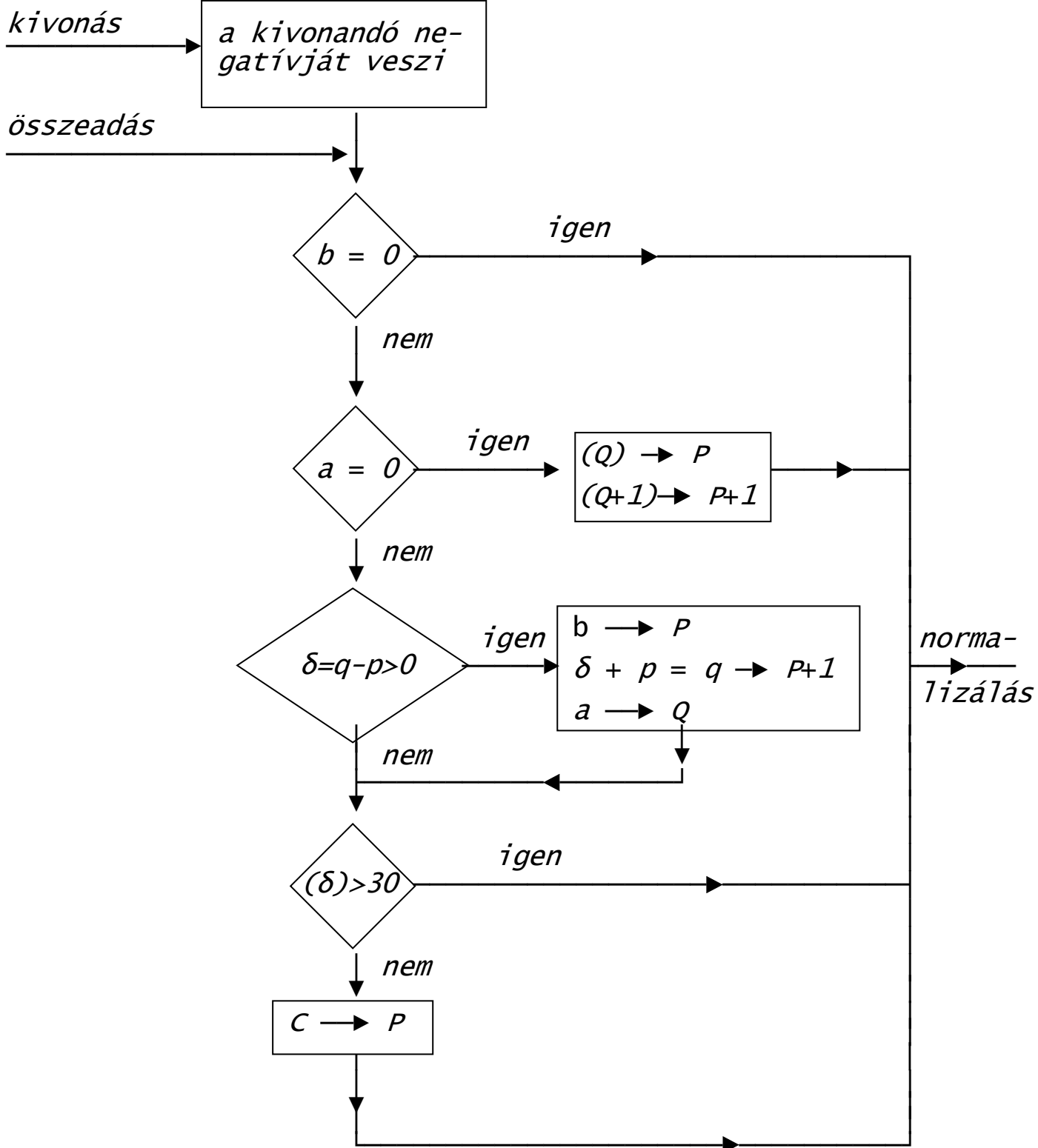
1. ábra



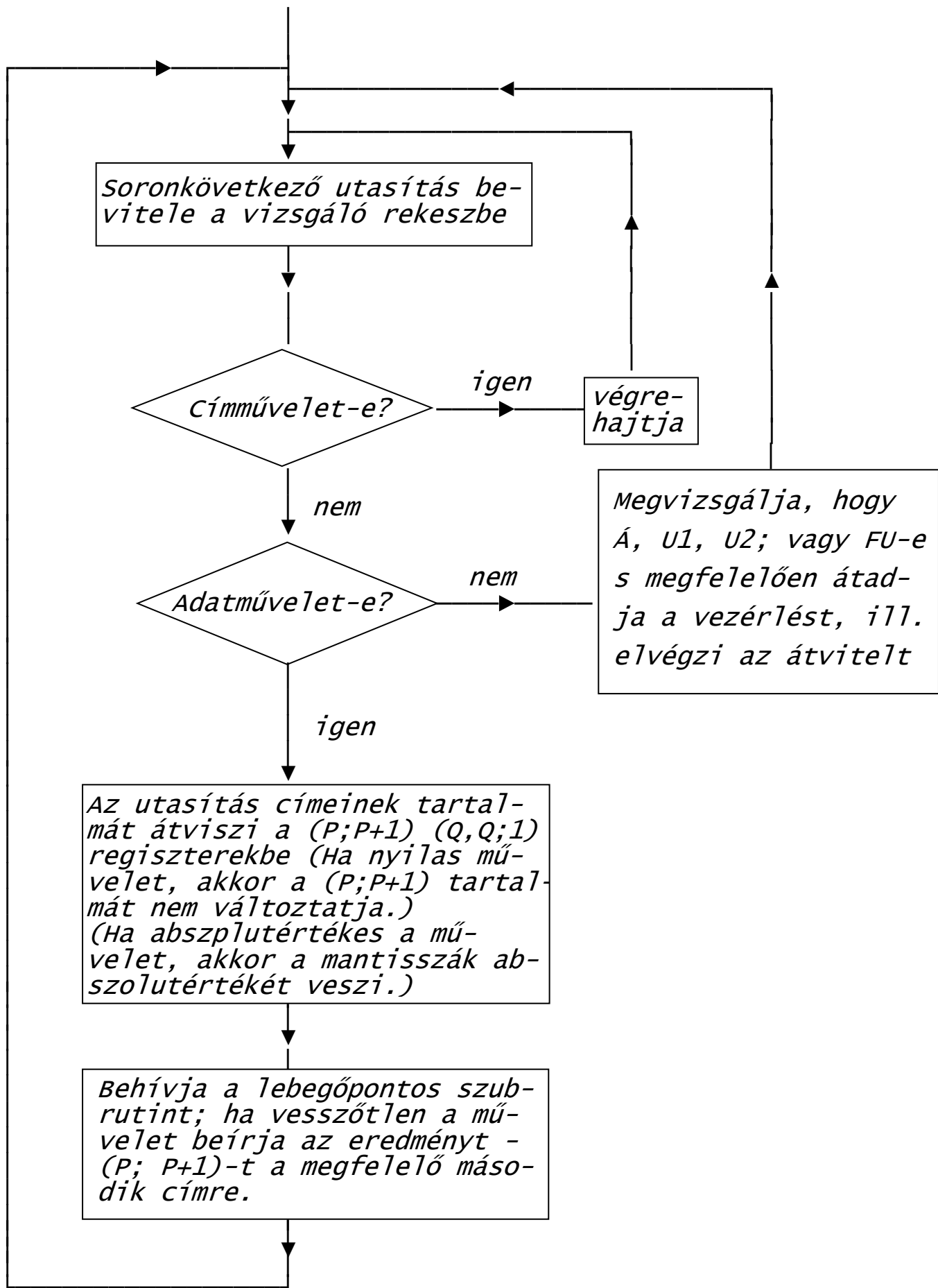
2. ábra



XVI. tábla



XVII. tábla



XVIII. tábla

EX LIBRIS

KALMÁR
LÁSZLÓ

Kalmár László