Kalmár

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.

TARTALOMJEGYZÉK

MAGYAR TUDOMÁNYOS AKADÉMIA
Kibernetikai Kutató Csoportja

1. Fejezet.

B E V E Z E T É S

Ebben a munkában az M-3 jelü szovjet mintáju számológép programozásával foglalkozunk. Az M-3 gép az univerzális, automatikus vezérlésü, tárolt programu, digitális, elektronikus számológépek családjába tartozik.

1.1. Az M-3 jellemzői.

Vegyük sorra a géptipus egyes felsorolt tulajdonságait.

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

Az elektronikus jelző megkülönbözteti a fenti tipus gépeit a jólismert mechanikus és elektromechanikus gépektől, amelyek mozgó alkatrészekkel végzik el a számitásokat. Az elektronikus számológépekben a számitások elvégzésére, egyes kivételektől eltekintve, nincsenek mozgó alkatrészek /de természetesen a kiinduló adatok bevitele és az eredmények kiadása mechanikus, mozgó berendezések segitségével történik/, az aktiv elemek elektroncsövek, kristálydiodák, ujabban csövek helyett tömör elemek: mágneses magok, tranzisztorok.

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

Ugyanilyen univerzálisak az automatikus vezérlésü /vagy: programvezérelt/ számológépek is. Ezek u.n. "aritmetikai egysége" szintén csak az aritmetikai alapmüveleteket /s esetleg néhány elemi logikai müveletet/ képes elvégezni, de ezeknek a müveleteknek bizonyos egymásutánjával minden számitási feladat megoldható, amelyhez numerikus módszer tartozik. Az automatikus vezérlésü gépek éppen abban térnek el a közönséges asztali számológépektől, hogy mig utóbbiak esetében minden egyes aritmetikai müveletet a kezelőnek kell beállitania, előbbiek a feladat alkalmas formában való megadása után önmüködően végzik el az elemi számitások szükséges sorozatát egészen az eredménynek kivánt 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 előadássorozatunk tulajdonképeni tárgya. Ez röviden abban áll, hogy az automatikus vezérlésü számológép részére a számitási feladat kiindulószámadatain kivül szükséges előre megadni az u.n. utasitásokat is, amelyek tartalmazzák, hogy milyen müveleteket kell elvégeznie a gépnek a megadott számokon, illetve közbenső eredményeken az adott számitási feladat /meghatározott numerikus módszer szerinti/ megoldása céljából.

Az utasitásokat a számadatokhoz hasonló számjegykombinációk alakjában adják meg a gépnek. A gép belsejében tárolja az utasitásokat, csakugy mint a kiinduló mennyiségeket, és azokat a közbenső eredményeket, amelyekre a számitás további során szükség van.

Egy adott speciális probléma gépi megoldására szolgáló utasitások összességét programnak nevezik. Programozás pedig egy bizonyos program megszerkesztését, megirását jelenti, azzal a szükséges előkészitő munkával együtt, amely megelőzi az utasitások részletes felirását. A fenti jellemzésben tárolt programu azt jelzi, hogy a számológép a számitási programot /az azt alkotó utasitásokat/ a számadatokkal azonos módon tárolja.

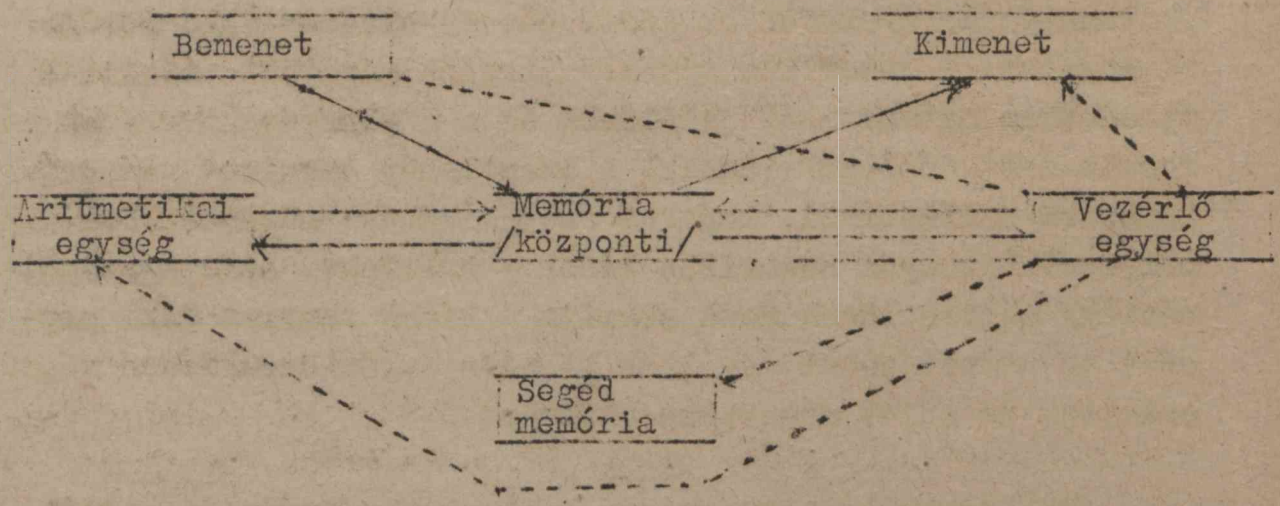
A jellemzett tipusu gépek, amelyek közé az M-3 is tartozik, a modern számológéptechnika legfejlettebb, legnagyobb teljesitményü gépei. Az elektronikus áramkörök és az automatikus vezérlés olyan müködési sebességet tesz lehetővé, amely által ezek a gépek rendkivül összetett numerikus feladatokat a gyakorlat számára reális idő alatt tudnak megoldani.

A programozás maga ugy tekinthető, mint az emberi /matematikai/ nyelvről az adott gép "nyelvére" való forditási müvelet. A numerikus módszer alapján megadott számitási feladatban matematikai jelölések és a közönséges nyelvben megfogalmazott döntések vannak. A számológép "nyelve" viszont egyszerü aritmetikai, logikai és választási müveletekből áll, melyeket bizonyos numerikus kódok /"kulcsok", számjegykombinációk/ képviselnek. Ezek szerint a kódok szerint kell megadni az utasitásokat, amelyek alapján a gép önmüködően végigszámitja az adott numerikus eljárást.

Tulajdonképeni programozásról csak univerzális, digitális és természetesen automatikus vezérlésü számológépeknél lehet szó. Analóg gépek vagy olyan digitális célgépek esetén, melyek csak egy speciális feladat /pl. adott algebrai vagy

differenciálegyenlet/ megoldására szolgálnak, nincs szükség programozásra, azeknek 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 felépitését. A számológép öt főegységből áll, annak az öt fő funkciónak, megfelelően, amelyet egy számitási 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 /számadatok, utasitások vagy a gép által tárolható egyéb adatok/, a szaggatott vonalak a vezérlő-jelek utján jelzik. Meg kell jegyezni, hogy ezek az elvi egységek fizikai megvalósitásukban /mechanikus és elektronikus szerelvények/ is általában egymástól elkülönitetten vannak felépitve 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 müveletet hivják olvasásnak. A bemeneti berendezések feladata tulajdonképen átültetni az információt külső megjelenési formájáról, ahogyan pl. egy lyukkártyán szerepel, arra a formára, amelyben a memória tárolja. Az 1. ábrán látható nyil, amely a bemenetet a memóriával összekapcsolja, azt jelzi, hogy az információ a bemenettől csak a memóriába mehet és további müveleteknek kell azt a memóriából a gép más részeibe elvinni. Bár a továbbiakban is óvatosnak kell lennünk az analógiákkal, mert azok többnyire félrevezetők, mégis összehasonlithatjuk az automatikus számológépek billentyüzetével. Ez az analógia azért nem tökéletes, mert az asztali számológépeknek nincs igazi belső memóriájuk. A szükséges adatokat a kezelő emlékezetében vagy papiron tárolja.

Az automatikus vezérlésü számológépek a számokat, utasitásokat és a számitás egyéb szükséges adatait a memóriában tárolják. Ezen minden információ keresztülmegy. Aritmetikai müveletek csak olyan számokon végzehetők el, amelyek benn vannak a memóriában és minden utasitásnak, amely a gép müködését irányitja, 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, utasitás, vagy egyéb adat /közös néven: szó/ tárolására legyen képes és ezeket minimális késéssel továbbitsa a gép aritmetikai és vezérlő egységének. Azt az időt, amely valamilyen adatnak - szónak - a memóriába való bejegyzéséhez, /beirásához/, illetve az onnét való kivételéhez /kiolvasásához/ szükséges, a memória elérési /hozzáférési/ idejének nevezik. Ez az idő a jelenlegi leggyorsabb számológépeknél 10 usec körül van. A tárolt programu gépekben általában nincs megkülbönöztető jel arra, hogy egy szó számadatot,

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

Müszaki és gazdaságossági okokból nem célszerü a nagysebességü memóriát olyan nagy kapacitásuvá épiteni, hogy a számitáshoz szükséges összes adat beleférjen. A megoldás az, hogy a memóriát két vagy több léposőben épitik ki és az információnak azt a részét, amelyre a számitás későbbi fázisában lesz szükség, egy nagykapacitásu, de lassubb segédmemóriában helyezik el. Amint az 1. ábrából látható, a segédmemória csak a központi memóriával van összeköttetésben, tehát az adatokat az utóbbiba kell hozni, mielőtt résztvehetnek a számitásban.

Mint emlitettük: a számitásban közvetlenül a központi memória vesz részt. Ez rekeszekre oszlik, amelyek mindegyike egy szó tárolására alkalmas. Annyi rekesz van, ahágy szó tárolására képes a memória, azaz amekkora a memória kapacitása. A programkészités céljára szükség van a memória rekeszeinek valamilyen azonositására. A gépi utasitásokban ugyanis meg kell adni, hogy egy bizonyos müvelethez honnét kell venni a számadatokat, hová kell elhelyezni az eredményt és hol található a következő végrehajtandó utasitás. A megoldás az, hogy a memória rekeszeihez azonossági számokat rendelnek hozzá, 0-tól /n-1/-ig, ahol n a rekeszek száma. Ez a sorszám a rekesznek, illetve tartalmának <u>cime</u>. Közelfekvő analógia adódik a számológépek memóriája és a póstafiókok között. A póstafiókoknak azonositási célokból névtáblái vannak, amelyek azonban semmit sem mondanak a fiók tartalmáról. A névtáblák funkciója: ha egy bizonyos levelet elhelyeznek az <u>x</u> feliratu fiókban, akkor ugyanez a levél a későbbiek folyamán megtalálható lesz, ha az <u>x</u> feliratu fiókban keresik. Ugyanez egy memória-rekesz cimének a funkciója. Ha egy bizonyos számot elhelyezünk pl. az 1507-es rekeszben, később visszatérve az 1507-es rekeszhez, kiolvashatjuk belőle a beléhelyezett

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

A memóriának két fontos tulajdonsága van, amely nem vág a póstafiók hasonlattal. Először, egy memóriarekesz egyszerre csak egy szó tárolására képes és egy uj szónak a beirása véglegesen kitörli a régi tartalmat. Ez egyrészt előnyös, mert egy szó tárolásakor nem kell törődni a rekesz előzetes kiüritésével, 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 tulajdonképeni számoló szerve. Ez végzi el a négy alapmüveletet /egyes gépeknél csak hármat, osztás nélkül/, ezenkivül általában képes számok helyérték eltolására és egyszerü logikai müveletek elvégzésére. Az aritmetikai egység megfelel az asztali számológépek számolókerekeinek és tengelyeinek, amelyek a tulajdonképeni számolást végzik.

A számológép vezérlő egysége a memóriában tárolt utasitások /számjegykombinációk/ szerint vezérlőjeleket küld a gép többi egységeinek a megfelelő utasitás végrehajtása céljából. Az ábrán két folytonos vonal jelzi, hogy az utasitások oda-vissza közlekedhetnek a memória és a vezérlő egység között. A szaggatott vonalak a vezérlőjelek utját jelzik. A vezérlőegység összehasonlitható az asztali számológépek kiváltó billentyüivel, amelyeket a különböző aritmetikai müveletek elvégeztetése céljából kell lenyomni, bár ez a hasonlat eléggé tökéletlen.

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

A számológép kimeneti egysége a számitási feladatok megoldását és ezenfelül esetleg a memória egyéb tartalmát adja ki megfelelő formában. Az eredmények és egyéb adatok kiadása lyukkártyákon, lyukszalagon, mágnesszalagon, nyomtatott iveken, stb. történhet. Az 1. ábra szerint az információ csak a memóriából kerülhet a kimenetre. Ez alapjában igy van, bár a legtöbb gépnél az aritmetikai egység is részt vesz az adatok közvetitésében.

A bemeneti és kimeneti berendezések általában a gép többi részétől elkülönitett 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. Fentieken kivül 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 müszakilag csak az impulzus jelenlétét vagy hiányát célszerü 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.

$$X = a_m r^m + a_{m-1} r^{m-1} + \ldots + a_0 + a_{-1} r^{-1} + \ldots + a_{-n} r^{-n} + \ldots =$$

$$= /a_m a_{m-1} \ldots a_0 , a_{-1} \ldots a_{-n} \ldots /_r$$

ahol X az ábrázolandó szám,
r a számrendszer alapszáma,
az indexszel ellátott a-k X számjegyei az r alapu számrendszerben való előállitásban /r-nél kisebb természetes számok, vagy 0-ok/
m a "tizedes" vesszőtől balra,
n a "tizedes" vesszőtől jobbra lévő számjegyek száma.

Igy történik a számok jólismert ábrázolása a decimális /tizes/ 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 müködő gépek is, de ezekben minden decimális számjegyet bináris számjegyeknek /biteknek/ egy csoportja képvisel.

1.3. A program szerepe.

Az utasitás és a program fogalmának jobb megvilágitása céljából ismét visszanyulunk az asztali számológépek példájához. Egy összetett számitási 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 müveleteket kell elvégezni. A számitás elemi lépések egymásutánjából tevődik össze, melyek mindegyike egy aritmetikai müveletet és általában uj számadatot is tartalmaz. - Az automatikus számológépeknél a helyzet nagyjából hasonló. A megoldandó feladatot olyan lépések sorozatára kell felbontani, amelyek mindegyike valamilyen aritmetikai vagy a számitás szervezésével kapcsolatos müveletet és - a géptipustól 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 rábizva a kezelőre, a számitás bizonyos pontjainál neki kell megválasztania a továbbiakban követendő lépések sorozatát. Ezzel szemben az automatikus számológépeknél a végrehajtandó lépéseket szigoruan definiálva kell megadni, miközben az adott probléma gépi számitása során számbajöhető valamennyi eshetőséget tekintetbe kell venni és előre gondoskodni kell a gép által követendő programról bármelyik lehetőség bekövetkezése esetére.

Mint már emlitettük, az utasitások numerikus kód alakjában, a számadatokkal azonos formában szerepelnek a gép memóriájában. Az utasitások szerkezete különböző gépeknél különböző. Általában két fő része van az utasitásoknak: a müveleti és a cimrész. A müveleti rész jelzi a gépnek, milyen müveletet kell elvégeznie, a cimrész pedig a memória azon rekeszeit jelöli meg, amelyek tartalmán kell a jelzett müveletet elvégezni. Aszerint, hogy egy adott gép utasitásai-

nak címrésze a memória hány rekeszére hivatkozik, beszélünk egy, két, stb. címü gépekről. Bár az utasitások müveleti része /müveleti jele/ a memóriában mint numerikus kód szerepel, az utasitások leirásánál ehelyett bizonyos szimbólumokat /betürövidítéseket vagy egyebet/ használnak.

Minden gépnek van utasitásrendszere, amely az illető gép által egy utasitásban elvégezhető müveletek és a hozzátartozó müveleti jelek /szimbólumok és kódok/ listája. Egy adott gép számára a számitási programot az utasitásrendszerében szereplő utasitásokból kell összeállitani. Ez egyben mindig lehetséges is.

A számológépek többsége /ezek közé tartozik az M-3 is/ a tárolási sorrendjében hajtja végre az utasitásokat és ezért egy program utasitásai általában a memória egymást követő rekeszeiben vannak elhelyezve. Az utasitások végrehajtásának ez a sorrendje csak akkor változik meg, ha a program egy u.n. ugró-utasitáshoz érkezik, amelyiknek /egyik/ címe megadja a következő végrehajtandó utasitás helyét a memóriában. Az ugrás végrehajtása után a program ismét a tárolás sorrendjében követi az utasitásokat, az uj címtől kiindulva a következő ugró-utasitásig. Az ugró-utasitások müködését a vezérlés átadásának nevezik. Vannak feltétlen és feltételes ugró-utasitások. 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-utasitások teszik lehetővé a különböző számitási utak /programváltozatok/ közötti választást /ezek az u.n. elágazások/ az addigi számitás folyamán kialakult kritériumok alapján és ezek teszik lehetővé az u.n. ciklusok programozását is. Jelentőségükkel a későbbiek folyamán még bőven fogunk foglalkozni.

Vannak u.n. megállitó utasitások. Ha a program megállitó utasitáshoz érkezik, a gép megáll, azaz beszünteti az utasitások egymásutáni automatikus végrehajtását. Ezután a gép csak a kezelő beavatkozása esetén működik ujra. A megállitó utasitások általában ugy vannak elhelyezve a programban, hogy a gép a következő esetekben álljon meg:

1. Végetért a számitás és az eredmények a kivánt formában vannak.

2. A számitás nem végezhető el /pl.negativ számból kell négyzetgyököt vonni és a további számitáshoz valós gyök szükséges/.

3. A gép a számitás során hibát követ el.

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

Egy utasitás végrehajtása nagyjából a következő lépésekben történik: 1. az utasitás a memóriából a vezérlő egységbe kerül. 2. A vezérlő egység a müveleti rész számjegykombinációjának /kódjának/ megfelelő vezérlőjelet küld az aritmetikai egységnek /dekódolja, "értelmezi" az utasitás müveleti részét/. 3. A cimrész alapján a vezérlőegység jelzést ad a memóriának, amelyből a megfelelő cimen /cimeken/ tárolt szavak az aritmetikai egységbe kerülnek. 4. Az aritmetikai egység a kapott számadatokkal végrehajtja a megfelelő müveletet. 5. Az aritmetikai egységben ideiglenesen tárolt eredmény a memóriának az utasitásban megadott cimére kerül. - Az utasitások végrehajtásának valóságos menete gépenként kisebb-nagyobb mértékben eltér a fenti sémától, előfordul, hogy egyes utasitások 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 kivánnak 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 müködött és függvénytáblázatoknak differencia módszerrel való kiszámitására szolgált. 1833-ban Babbage egy második gépet tervezett, az u.n. analitikus gépet /Analytical Engine/. Ez a gép a modern számológépek ősének tekinthető, univerzális gép. A terv szerint a Jacquard-féle szövőszéknél használt lyukkártyák által lehetett megadni számára az elvégzendő müveletek sorozatát, a számokat mechanikus számkerekek tárolták és az összes müveletet automatikusan hajtotta végre. Teljesen mechanikus elvek szerint müködött volna, de egészében soha sem épült fel. Ennek oka részben az anyagi fedezet hiánya, főleg azonban az volt, hogy tisztán mechanikai uton nem lehetett megvalósitani az univerzális automatikus számológép funkcióit. Fejlett elektronikára és impulzustechnikára volt szükség a modern számológépek megvalósitásához.

Az első modern számológépet, 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 számitási 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 elvégeztetése/ 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 magyarszármazásu matematikus, egy jelentésben megfogalmazta a tárolt programu gépek elvét, azt az elvet, hogy az utasitásokat 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 teljesitményt tett lehetővé kisebb költségü 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 tipushoz tartozó gépek igen nagy fejlődésen mentek keresztül a működési sebesség, megbizhatóság és a kezelés egyszerüsége 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, "uj kiadása" van. 1953-ban az IBM elindította az u.n. 700-as sorozat gépeit. Ezek nagyteljesitményü, tömeggyártásban készült gépek. A sorozat egyes tagjai természettudományos, mérnöki számitások céljaira, mások üzleti, adatfeldolgozások céljaira készültek. Az alkalmazások két emlitett tipusa

főleg abban különbözik, hogy előbbinél aránylag kevés adaton kell hosszu számitási programot elvégezni /szoros értelemben vett számitási feladatok/ utóbbinál sok adaton rövid számitást kell elvégezni és az adatokat csoportositani, szortirozni kell /adatfeldolgozási feladatok/. A kétféle tipusu alkalmazások céljaira szolgáló gépek többek között egyes egységeik /aritmetikai egység, memória/ teljesitmény-arányában különböznek egymástól.

A legujabban tervezés alatt álló gépek az előzetes adatok alapján nagyságrendekkel felülmulják a jelenlegi gépek teljesitményét. Pl. Az IBM tervezett STRETCH nevü gépe 2oo-szor gyorsabb a jelenlegi univerzális gépeknél.

A Szovjetunióban is nagy eredménnyel folytatják a modern számológépek kifejlesztését és üzemeltetését. Nagy teljesitményü gépek: a BESZM, a SZTRELA. Közepes teljesitményü: az M-2. Kis teljesitményüek: az ESZM, az URAL, az M-3. Nagy teljesitményünek nevezik azokat a gépeket, amelyek másodpercenként 2ooo-2o.ooo müveletet végeznek, közpes teljesitményünek azokat, amelyek másodpercenként 5oo-2ooo müveletet végeznek, kis teljesitményünek pedig azokat, amelyek másodpercenként lo-5oo müveletet végeznek el. Bár a legnagyobb teljesitményü 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 számitási probléma gépi megoldását előkészitő munka különböző lépései.

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

a./ Numerikus analizis. Mint már emlitettük, egy problémának gépi megoldásához elengedhetetlen, hogy "aritmetizálható" legyen, azaz létezzék számára numerikus módszer, amely révén a probléma a gép által elvégezhető elemi müveletek /aritmetikai alapmüveletek/ sorozatára bontható. Ennek a numerikus módszernek a konkrét feladatra való kidolgozása a numerikus analizis. Bár ennél bizonyos mértékig figyelembe kell venni az adott gép sajátságait /pl. hogy a gép utasitásrendszere tartalmaz-e osztást, stb./, ez a munka független a gép részletadataitól. A numerikus analizis fő módszere: a feladatnak azonosan vagy csekély változással ismétlődő müveletcsoportokra való bontása. Ezeket az ismétlődő müveletcsoportokat a gépi programban utasitások csoportjai képviselik, amelyeket a gép egymásután többször hajt végre; a továbbiakban látni fogjuk, hogy lehetséges az ismétlődő csoportok utasitásait kisebb mértékben változtatni az egyes ismétlődések közben. A szükséges ismétlődések száma egyes esetekben előre ismert, más esetekben nem ismert. Mindenképen egy feltételes ugró-utasitás zárja le az utasitások ismétlődő csoportját, amely vagy visszaadja a vezérlést a csoport első utasitására, vagy - elegendő ismétlődés /iteráció/ esetén - rátér a program következő részére.

Bár a numerikus analizis rendkivül fontos a problémák gépi feldolgozása szempontjából, előadássorozatunkban nem foglalkozunk vele. Ez a munka ugyanis, mint már emlitettük, 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 analizis egy részét és az utasitások részletes felirását is. Szükebb értelmében a programozás a numerikus analízis után következő és a számológéppel szorosan kapcsolatos előkészitő munka. Ide tartozik elsősorban a gépi számitás logikai menetrendjének /blokk-diagramjának/ elkészitése. A blokk-diagram a számitás menetét diagramszerü ábrázolása, amely feltünteti az ismétlődő utasitácsoportokat /"hurkokat"/ és a program elágazásait. A szükebb értelemben vett programozáshoz tartozik még: megtervezni a számadatoknak és a program egyes részeinek a memóriában való elhelyezését és megbecsülni a számitás géppel való elvégzésének időszükségletét.

c./ Kódolás. Ez a számitási program részletes gépi utasitásainak megirása, számjegykombinációk /számjegykódok/ vagy szimbólumok formájában. A kezdő rendszerint evvel 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 forditeni. A kódolás legszorosabb kapcsolatban van az adott gép utasitásrendszerével és egyéb jellemzőivel. Ezért bizonyos értelemben ujra kell megtanulni egy másik gép kódolását, bár az ujratanulás 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 előkészitő munka, készteti a fejlesztőket a kódolás automatizálására, azaz magával a géppel való elvégeztetésére.

d./ Ellenőrzés. Az eddig felsorolt előkészitő lépésekben annyi hibalehetőség rejlik, hogy a programok első felirásban többnyire hibásak. Ezek a hibák vagy olyan ter-

mészetüek, hogy a gép megáll, mielőtt bármilyen eredményt adna, vagy hatásukra rossz eredmények születnek. A programot ezért többszörösen ellenőrizni kell, ugy is, hogy a gép által kapott első eredményeket /amelyeknél ez gyakorlatilag lehetséges/ kézi számitással ellenőrzik.

1.6. A számológépek alkalmazásai.

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

a./ Szoros értelemben vett számitási feladatok:

1. Matematikai és mérnöki feladatok nagypontosságu megoldása. Kézi eszközökkel való számitások esetén a legtöbb problémát lényegesen le kell egyszerüsiteni, változókat, magasabbrendü tagokat el kell hanyagolni. A modern számológépek elég gyorsmüködésüek ahhoz, hogy lényegesen kevesebb elhanyagolás váljék szükségessé.

2. Uj konstrukciók /pl. gőzturbinák, aerodinamikai profilok/ müködését azok megépitése előtt a számológép által meg lehet vizsgálni. Ezzel nagy összegek és sok idő megtakaritható. A számológép pótolja a modell létesitését, mintegy "szimulálja" a megépitendő berendezést.

3. Uj konstrukciók /pl. turbinák, erősáramu hálózatok, stb./ paraméterei számára adott szempontból az optimális kombináció megtalálása. Ez ugy történik, hogy valamennyi lehetséges paraméter-kombinációt végig kell számolni, ami a legegyszerübb esetektől eltekintve csak gyorsmüködésü számológépen oldható meg.

4. Üzemek egészére, vagy résztevékenységeire vonatkozó hatékonysági számitások. Ilyen számitások 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 számitások iparági, sőt népgazdasági szinten is végezhetők.

b./ Adatfeldolgozási feladatok.

1. Mérési adatok feldolgozása. Fizikai kisérletsorozatok, uj 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 müszaki ügyvitelének gépesitése. Ide tartozik a termeléssel, szállitással és értékesitéssel kapcsolatos nagytömegü adatok rendszeres feldolgozása, a gazdasági egység vezetéséhez, illetve az egész népgazdaság irányitásához szükséges módon. Elsősorban a termelés, továbbá munka /bér/ és anyagráfordítások elszámolása, a termelési program összeállitása és ütemezése és készletek állományváltozásának tág értelemben vett nyilvántartása sorolható ide. E folyamatok gépesitése révén a regisztráláson tulmenően lehetővé válik a nyilvántartások közvetlen operativ felhasználása. Pl. meghatározott készletszint fenntartásával kapcsolatos intézkedések.

3. Az államigazgatás nagytömegü adatnyilvántartásával és adatfeldolgozásával kapcsolatos folyamatainak gépesitése. Ide sorolhatók a bármiféle statisztikai munkák, társadalombiztositási nyilvántartások és számitások, adónyilvántartás és kivetés, állami költségvetéssel és beruházásokkal kapcsolatos nyilvántartások és számitások.

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

A fenti felsorolt alkalmazásoknak többé-kevésbé közös jellemzője, hogy a feladat megoldásán belül bizonyos alapvető számitást kell igen sokszor elvégezni, különböző adatokon. Pl. a bérelszámolás esetében azonos tipusu számitást kell elvégezni minden munkavállalóra és minden fizetési időszakra. Egy berendezés mérési adatainak feldolgozásánál sok ezerszer kell elvégezni ugyanazt a fajta számitást. Valamilyen müszaki terv optimatizálása esetén a paraméterek minden lehetséges kombinációjára azonos számitásokat kell elvégezni. Ezek a feladatok mind lehetővé teszik az ismétlődő müveletcsoportok, ismétlődő /iterált/ utasitáscsoportok szerinti programozást, az u.n. utasitásciklusok programozását, melyekről már előbb emlités esett. Gyakorlatilag nem érdemes olyan feladatot számológépre vinni, ahol a feladaton belül az alapvető számitás csak kevésszer ismétlődik. Ez a későbbiek folyamán nyilvánvalóvá lesz.

2. Fejezet.

AZ M-3 ORGANIZÁCIÓJÁNAK ÉS UTASITÁSRENDSZERÉNEK RÉSZLETES LEIRÁSA.

2.1. A gép fő egységei.

Az előző fejezetben megismert univerzális digitális, automatikus tárolt programu, elektronikus számológépek családjában a nálunk épülő M-3 gép a kisméretü, kisteljesitményü gépek közé tartozik.

Külső felépitését tekintve a gép a következő részekből áll /1.a 2. ábrát/.

1. Főszekrény, amely az aritmetikai és vezérlőegységeket tartalmazza.

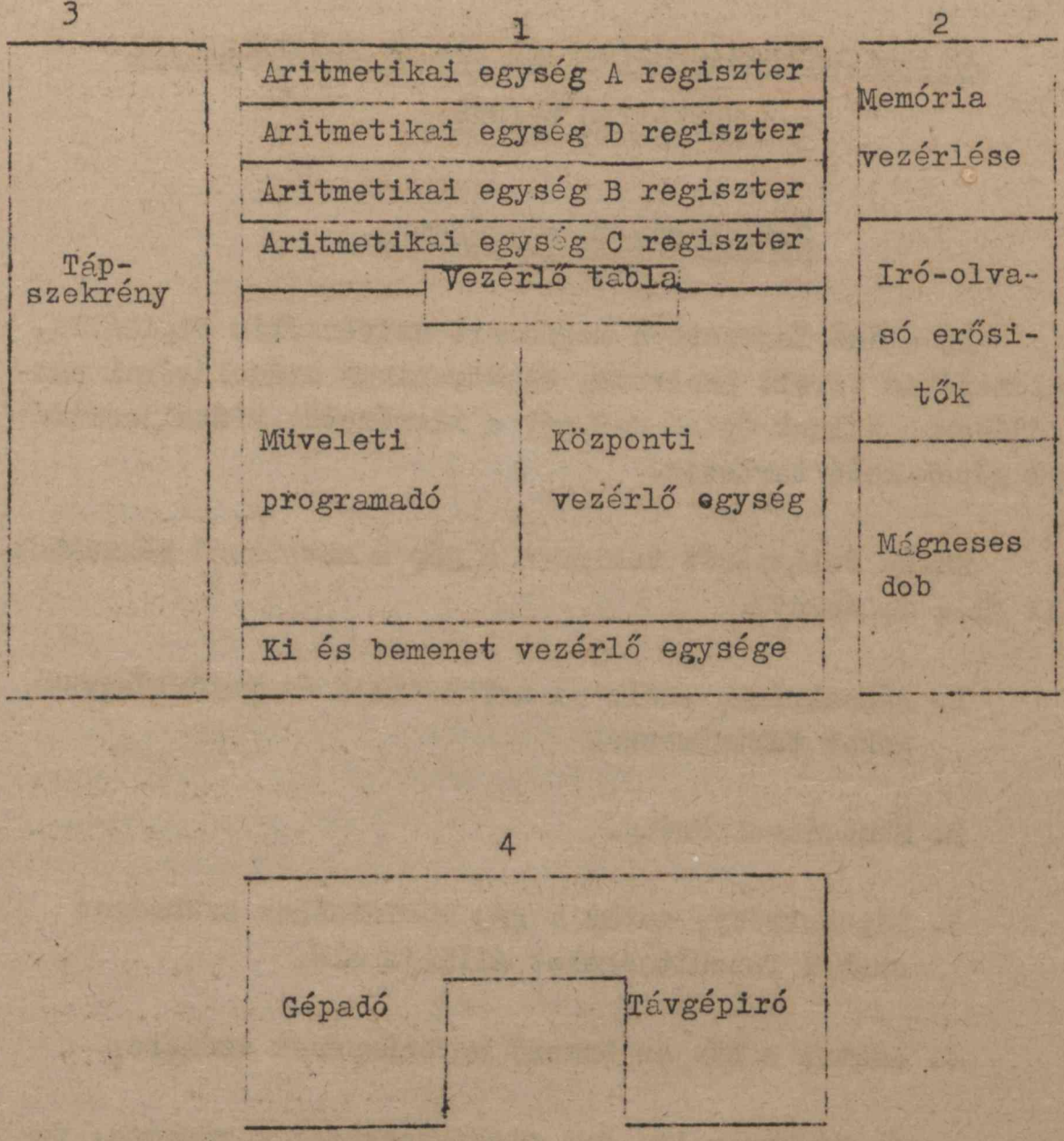
2. Memória-szekrény.

3. Tápszekrény, amely a gép müködéséhez szükséges stabil feszültségeket állitja 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 abszolut értékét ábrázolja. Igy az AE, amely csak ez utóbbiakkal végez müveleteket 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 számjegynek 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./ Müveleti programadó.
c./ Ki- és bemenet vezérlése.

Ezekben találhatunk kisebb regisztereket egyes adatok /pl.az utasitásokban szereplő cimek/ tárolására, ezenkivül számlálókat, valamint a vezérlő impulzusok képzését és továbbitását végző áramköröket.

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

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

Egy megadott cimhez tartozó rekesz megkeresésének átlagos ideje az u.n. elérési idő a dobnál 1o 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 müveletek végrehajtási idői nagyjából elhanyagolhatók a memóriából való kiolvasás ill. oda való beirás fenti idejéhez képest. Mivel a tapasztalatok azt mutatták, hogy egy utasitás végrehajtásához átlagosan,

valamivel több mint 3-szor kell a memóriához fordulni, azt mondhatjuk, hogy a gép kb. 3o utasitást tud másodpercenként végrehajtani, azaz kb. ennyi aritmetikai müveletet végez másodpercenként, ami 1oo.ooo müv./óra sebességnek felel meg.

A memóriaszekrényben a dob felett helyezkednek el az u.n. iró-olvasó erősitők, valamint a memória vezérlőberendezése.
Az M-3 felépitése lehetővé teszi a belső memóriának 4096 szó befogadóképességüre való bővitését a gép többi részének lényeges megváltoztatása nélkül. A gép kiegészithető gyorsmüködésü ferritmemóriával, melynek igen rövid elérési ideje lehetővé teszi az AE müködési sebességének lényegesen jobb kihasználását és ezzel a gép átlagos sebességének kb. 15oo-2ooo művelet/másodpercre való növelését. Kiegészithető 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 gyártmányu gépadó, kimenőberendezését egy ugyanilyen távgépiró alkotja. A kiszámitásra kerülő feladatok programját és számadatait egy perforátor-távgépiró segitségével szalagra kell lyukasztani, a gépadó a lyukakat letapogatja és impulzusok formájában beviszi a gépbe. A bevitel jelenlegi sebessége 3o szó/perc. Ez megfelelő gyorsmüködésü beviteli berendezés segitségével lényegesen /akár százszorosára is/ növelhető.

A közbülső és végeredmények, de a memória bármilyen rekeszének tartalma is a távgépirő segitségével 1, 2, 3, 4. vagy 5. oszlopos táblázatok formájában kiirathatók. A kiirandó mennyiségeket természetesen a programozás során kell előre kijelölni. Egy szó kiirása a jelenlegi berendezéssel kb. 2 másodpercet vesz igénybe, a kiirási sebességet a bemenethez hasonlóan lényegesen növelni lehet.

2.2. A müveletek végrehajtásának módja.

Rátérve a gép müködésének ismertetésére, először azt nézzük meg, hogyan végzi el a gép aritmetikai egysége a müveleti programadó irányitásával magukat az egyes alapmüveleteket.

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

±	a_1	a_2		a_{30}

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

értékünek veszi, ahol $a_1, a_2, \ldots a_{30}$ 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. Igy a gép közvetlenül csak a -1 és +1 közé eső számokkal tud müveleteket végezni, sőt ha valamely müvelet eredményeképpen ezen intervallumon kivüleső számot kapunk, a gép megáll. Ezért minden feladatnál a programozónak gondoskodnia kell arról, hogy mind a kiinduló adatok, mind pedig a közbülső és végeredmények ebbe az intervallumba transzformálódjanak.

Ennek módszereiről a későbbiekben lesz szó.

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

1./ összeadás /+/,
2./ kivonás /-/,
3./ osztás /:/,

amennyiben a müvelet elvégezhető, azaz az eredmény abszolut 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 /∧/,

ami azt jelenti, hogy az eredmény egyes helyértékeire akkor és csak akkor irunk 1-est, ha mindkét tényező ugyanazon helyértéke 1-es volt.

Pl. /11010/ ∧ /01101/ = 01000

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

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

B. regiszter: tárolja az összeadandót, kisebbitendőt, 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 ugy 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 leirható, akkor beirjuk, 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 papiron történő összeadásnál ezeket az "átviteleket" általában fejben szoktuk megjegyezni, semmi akadálya nincs azonban annak, hogy ezeket külön helyen fel is jegyezzük. Ilyenmódon 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 figyelembevéve az átfutás során addig már felirt átviteleket.

b./ Elvégezzük, helyértékenként egyszerre, a két szám és az átvitelek összeadását /az alapszámot meghaladó összegből levonva az alapszámot, de ujabb átviteleket természetesen már nem képezve/.

Pl.:

tizes számrendszerben		kettes számrendszerben
1./ szám	106	1101010
2./ szám	295	100101111
átvitelek	011	1101110
összeg	401	110011001

Fentieket a kettes számrendszerre alkalmazva a következő gépies szabályokat állapithatjuk 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 irunk, különben 0-át.

b./ Ha valamely helyértéken a 3 emlitett számjegy közül páratlan számu 1-es van, az összeg megfelelő helyértékére 1-est, ha páros, akkor 0-át irunk.

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-triggere 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 μsec alatt végzi el.

Példa:

1./ A: 1 0 0 1 0 1 1
D: 0 1 1 1 1 1 1 — az átvitel végigfutása után
B: 0 0 1 1 1 0 1

2./ A: 1 0 0 1 0 1 1
D: = = = = — — — összeg képzése után
B: 1 1 0 1 0 0 0

Az összeadás fenti módszerének nagy előnye az, hogy a művelet elvégezhetősége rendkivül egyszerüen 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: A: 1 0 0 1 0 1 1
D: 1 1 1 1 0 1 0 0
B: 1 1 1 0 1 0

Az összeadás nem végezhető el !

2.2.2./ Kivonás.

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

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

$$\bar{a} + a = 0,111 \ldots 1 = 1 - 2^{-30}$$

és igy

$$\bar{a} + 2^{-30} = 1 - a$$

A $\bar{a}$ számot az a szám komplemensének fogjuk nevezni.

A fentiek alapján, ha $b \geq 0$

$$\bar{a} + 2^{-30} + b = 1 - a + b = 1 + /b - a/$$

A gép tehát kivonás esetén mindenekelőtt az A regiszterben olhelyezett a kivonandónak képezi a komplenensét olymódon, hogy az A regiszter minden triggerének állapotát megváltoztatja és 2^{-30} hozzáadása céljából, a D regiszter utolsó - eddig feltétlenül 0 állapotban levő - triggerét állitja 1 állapotba, ami mint könnyen belátható, 2^{-30} 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

$$\bar{a} + 2^{-30} + b = 1 + /b-a/ \geq 1.$$

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

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

Más a helyzet, ha $D_0 = 0$, Ekkor

$$\bar{a} + 2^{-30} + b = 1 + /b-a/ < 1,$$

tehát az összeg képzése most már a valódi összeget fogja adni, nekünk azonban a negativ b-a értékre van szükségünk. Ezért felcseréljük a és b szerepét ugy, hogy képezzük mindkét regiszter /A és B komplemensét az előbbi módon: igy A-ban ujra a lesz és B-ben $\bar{b}$. Igy feltétlenül $D_o = 1$ lesz, mert most

$$a + \bar{b} + 2^{-30} = 1 + /a-b/ \geq 1$$

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

Igy az összeg képzése már elvégezhető és megadja a $/b-a/ < 0$ különbség abszolut értékét. Ugyanakkor gondoskodik a gép arról, hogy az oredmény előjele negativra változzék.

A kivonást a gép kb. 7o ill. 12o µsec alatt végzi el, attól függően, hogy a fenti két eset közül melyikkel kerül szembe.

Példa: 1./

	A:	0 0 1 0 1
	D:	- - - - -
	B:	0 1 0 1 1
A kompl.	A:	1 1 0 1 0
	D:	1 1 0 1 1 1
	B:	0 1 0 1 1
Összeg képzése	A:	1 1 0 1 0
	D:	- - - - -
	B:	0 0 1 1 0

2./

	A:	1 0 1 0 1
	D:	- - - - -
	B:	0 1 0 1 1
A kompl.	A:	0 1 0 1 0
	D:	0 1 0 1 1 1
	B:	0 1 0 1 1
A és B kompl.	A:	1 0 1 0 1
	D:	1 0 1 0 1 1
	B:	1 0 1 0 0
Összeg képzése	A:	1 0 1 0 1
	D:	- - - - -
	B:	0 1 0 1 0

és a B-ben lévő különbség előjele megváltozik !

Az eddigiekben feltételeztük, hogy a és b pozitivek; negativ számokkal a gép ugy végzi el az összeadást és kivonást, hogy visszavezeti az abszolut értékekkel végzett megfelelő müveletre. Igy különböző előjelü számok összeadása helyett kivonást, kivonása helyett összeadást végez. Az eredmény előjelét mindig a b szám /a kisebbitendő/ előjele határozza meg /a kivonás második eseténél tehát ezt kell megváltoztatni/.

Végeredményben tehát a gép mindig összeadást, vagy pozitiv eredményü kivonást végez és ezekből a müveleteknek és az eredmény előjelének megfelelő megválasztásával az előjeles számok összeadásának és kivonásának összes esete előállitható.

2.2.3./ Szorzás.

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

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

Az itt alkalmazott szorzási módszert a következő módon is leirhatjuk:

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

3|90625|

majd "léptessük" egy helyértékkel jobbra minden számjegyet, irjuk alá a következő részletszorzatot ésadjuk össze a két számot:

```
   3 9 0 6 2 5
 5 4 6 8 7 5
 ___________
 5 8 5 9 3 7 5
```

Az eredményt ismét léptessük jobbra egy helyértékkel, adjuk hozzá a következő részletszorzatot,

```
   5 8 5 9 3 7 5
 6 2 5 0 0 0
 _____________
 6 8 3 5 9 3 7 5
```

majd léptessük az eredményt ismét jobbra egy helyértékkel:

68359 375

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

Pl. a fenti szorzás:

```
    0,11100 . 0,11001
      11100
          0
         0
    11100
   11100
 __________
 0,1010111100
```

illetve az ismertetett módszer szerint:

```
1./    | 1 1 1 0 | 0
       |0 0 0 0 0|
     --+---------+---
       | 1 1 1 0 | 0


2./    |   1 1 1 | 0 0
       |0 0 0 0 0|
     --+---------+------
       |0 0 1 1 1|
```

3./

```
        1 1 1 0 0
  1 1 1 0 0
-------------------
  1 1 1 1 1 1 0 0
```

4./

```
      1 1 1 1 1 1 0 0
    1 1 1 0 0
-----------------------
  1 0 1 0 1 1 1 0 0
```

5./

```
  1 0 1 0 1 1 1 1 0 0
```

Mint látjuk mindig annyi léptetés szükséges, ahány számjegyből áll a szorzó.

A gépen a szorzandó az A a szorzó pedig a B és C regiszterekben helyezkedik el. A szorzás a B regiszter kiürítésével /"törlésével"/ kezdődik, ami azt jelenti, hogy minden helyérték triggerét 0 helyzetbe állitjuk. Ez azért szükséges, mert ebben a regiszterben fognak a részletszorzatok összeadódni. Ezután 3o-szor elvégezzük a következőket:

1./ ha a C regiszter 3o-ik helyértékén "1" áll /C_{3o}=1/ hozzáadjuk A tartalmát /szorzandó/ B tartalmához /a részlet szorzatok eddigi összege/, és az eredményt B-ben hagyjuk.

2./ B és C tartalmát jobbra léptetjük egy helyértékkel.

Az 1./ lépés közönséges összeadást jelent, evvel már foglalkoztunk. Meg kell még jegyezni azt, hogy itt előfordulhat az, hogy valamelyik részeredmény 1-nél nagyobb lesz, ez azonban nem okoz bajt, mert a következő jobbra léptetés már 1-nél kisebb számot eredményez. Igy szorzásnál az összeadás elvégzését nem D_o állapota, hanem kizárólag C_{3o} állapota dönti el. Ahhoz, hogy az 1-nél nagyobb eredményt ideiglenesen tárolni tudjuk, be kell vezetnünk a B-regiszterben is egy 0-ik helyértéket /B_o/.

A 2./ lépést /B és C léptetése jobbra/ a gép olymódon végzi el, hogy összehasonlitja 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 soronkövetkező számjegye kerül. Igy az 1./ szabály valóban a soronkövetkező 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 ilymódon elkövet, kisebb mint 2^{-30}, azaz kisebb a gépen ábrázolható legkisebb számnál.

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

Példaképpen végezzük el az előbbi szorzást, az egyszerüség kedvéért 30 helyett csak 5 számjegyet irunk 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:	1 1 0 0 1	
A:	1 1 1 0 0	Mivel C_{30}=1, elvégezzük az
B:	0 1 1 1 0 0	összeadást.
C:	1 1 0 0 1	

A: 1 1 1 0 0	B és C léptetése jobbra, össze-
B: 0 0 1 1 1 0	adást nem végeztünk, mert
C: 0 1 1 0 0	C_{30}= 0
A: 1 1 1 0 0	B és C második léptetése jobb-
B: 0 0 0 1 1 1	ra, összeadást nem végzünk,
C: 0 0 1 1 0	mért C_{30}=0
A: 1 1 1 0 0	B és C harmadik léptetése jobb-
D: 0 0 0 0 0 0	ra; mivel C_{30}=1
B: 0 0 0 0 1 1	
C: 0 0 0 1 1	
A: 1 1 1 0 0	Elvégezzük az összeadást.
B: 0 1 1 1 1 1	
C: 0 0 0 1 1	
A: 1 1 1 0 0	B és C negyedik léptetése
D: 1 1 1 0 0 0	jobbra: mivel C_{30}= 1
B: 0 0 1 1 1 1	
C: 0 0 0 0 1	
A: 1 1 1 0 0	elvégezzük az összeadást.
B: 1 0 1 0 1 1	
C: 0 0 0 0 1	
A: 1 1 1 0 0	B és C ötödik és egyben
B: 0 1 0 1 0 1	utolsó léptetése jobbra
C: 0 0 0 0 0	

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 müveletét papiron a következő módon szoktuk végrehajtani /feltesszük, hogy az osztandó kisebb mint az osztó/.

Kitesszük a tizedesvesszőt és "lehozunk egy 0-át" /azaz az osztandót /b/ balra léptetjük egy helyértékkel/ és megnézzük, hogy mi az a legnagyobb k nemnegativ 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-hozzáirás éppen a számrendszer alapszámával való szorzást jelenti /ha $b < a$, és k. $a < 10.b$, akkor $k < 10$ lesz, mert $k \geq 10$ és $a > b$-ből k. $a > 10$ b következnék/.

Igy 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 igy nyert szám, és igy tovább.

Pl.: 0,59375 : 0,75000 = 0,79166

```
  593750
- 525000
  687500
- 675000
  125000
-  75000
  500000
- 450000
   50000
```

Kettes számrendszerben a helyzet ismét annyiban egyszerüsödik, hogy csak 2 számjegyet kell tekintetni, csak azt kell megvizsgálni, hogy az osztó kisebb-e mint a balralépteteti 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 helyezzük el az osztót /a/ és a B regiszterben az osztandót /b/. A hányados a C regiszterben fog keletkezni. Ezután elvégezzük egy b-a kivonás előkészitését egészen az átvitel végigfuttatásáig. Ennek eredménye - mint ezt a kivonásnál már láttuk - megmutatja, hogy a $b \geq a$ ill. $b < a$ egyenlőtlenségek közül melyik áll fenn. Az előbbi esetben osztás nem végezhető el és a gép megáll, mig az utóbbi esetben megkezdődik az osztás elvégzése. Ez azt jelenti, hogy 3o-szor végrehajtjuk a következőket:

A B és C regiszter tartalmát balra léptetjük, ami éppen forditottját jelenti a szorzásnál megismert jobbra léptetésnek. Itt a 3o. 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 balraléptetett 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 előkészitő kivonás során komplenensét vettük és D_{30}-ba egyidejüleg "1"-et irtunk/, 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 igy belőle az 1-nél kisebb osztó feltétlenül kivonható.

Igy 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 soronlevő számjegyét, - amely a C regiszter sorozatos balraléptetései

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

Ilyen módon az osztás is a léptetés és összeadás 30-szoros ismétlésével valósitható meg. Egy osztás végrehajtása a gépen kb. 2ooo µsecot vesz igénybe.

Példa:/öt számjegyre/

A:	1 1 0 0 0	Az osztó az A, az osztandó a
B:	0 1 0 0 1 1	B regiszterben.
C:	- - - - - -	
A:	0 0 1 1 1	Képezzük A komplemensét, D_{30}-ba
D:	0 0 1 1 1 1	1-et irunk és elvégezzük az
B:	0 1 0 0 1 1	átvitel végigfuttatását. Mivel
C:	- - - - - -	$D_0 = 0$, az osztás elvégezhető.
A:	0 0 1 1 1	B és C első léptetése balra
D:	0 0 1 1 1 1	és az átvitel végigfuttatása.
B:	1 0 0 1 1 0	Az összeadást /amely értelem-
C:	- - - - - 0	szerüen tkp.kivonás/ elvégez-
		hetjük, mert $B_0 = 1$
A:	0 0 1 1 1	Összeadás elvégzése és $C_{30} = 1$
B:	1 0 1 1 1 0	beállitása.
C:	- - - - - 1	
A:	0 0 1 1 1	B és C második léptetése balra
D:	1 1 1 1 1 1	és az átvitel végigfuttatása.
B:	0 1 1 1 0 0	Az összeadást elvégezhetjük,
C:	- - - - 1 0	mert $D_0 = 1$
A:	0 0 1 1 1	összeadás elvégzése és $C_{30} = 1$
B:	1 0 0 1 0 0	beállitása.
C:	- - - 1 1	
A:	0 0 1 1 1	B és C harmadik léptetése bal-
D:	0 1 1 1 1 1	ra és az átvitel végigfuttatása.
B:	0 0 1 0 0 0	Összeadást nem végzünk, mert
C:	- - 1 1 0	$D_0 = B_0 = 0$

A:	0 0 1 1 1	B és C negyedik léptetése balra
D:	0 0 1 1 1 1	és az átvitel végigfuttatása.
B:	0 1 0 0 0 0	Összeadást nem végzünk, mert
C:	- 1 1 0 0	$D_0 = B_0 = 0$
A:	0 0 1 1 1	B és C ötödik léptetése balra
D:	0 0 1 1 1 1	és az átvitel végigfuttatása.
B:	1 0 0 0 0 0	Az összeadást elvégezhetjük,
C:	1 1 0 0 0	mert $B_0 = 1$
A:	0 0 1 1 1	összeadás elvégzése és $C_{30} = 1$
B:	1 0 1 0 0 0	beállitása.
C:	1 1 0 0 1	

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

2.2.5./ Logikai szorzás.

A müvelet elvégzése a már ismertetett definició 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 müvelet előtt
C:	0 1 1 0 1	
A:	0 0 1 1 1	a müvelet után
C:	0 0 1 0 1	

2.3. Az adatok be- és kivitele.

A számitás 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 tizes vagy kettes számrendszerben lehetnek megadva. A számrendszert egy kapcsoló állitja be. Az előbbi esetben minden sorban egy számjegyet helyezünk el olymódon, hogy az első helyértékre lyukat teszünk /ez felel meg az 1-nek/ a többi négy helyértékre pedig a decimális szám bináris megfelelőjelét irjuk:

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 felirt adatot akarunk a gépbe bevinni /ilyonek pl. a program utasitásai/, akkor három bináris számjegyet ábrázolunk egy sorban olymódon, hogy az első helyértékre ismét 1-et irunk, a második helyérték feltétlenül üresen marad és a többi három helyértékre beirjuk a három ábrázolandó számjegyet. Ilymódon a 0-7 decimális számjegyeknek megfelelő jelkombinációkat kapjuk.

Ezt ugy is mondhatjuk, hogy ezek a 8-as alapu számrendszer számjegyeinek felelnek meg. Valóban, ha az adott bináris számot ebben a számrendszerben irjuk fel, ugyanerre az eredményre jutunk, hiszen egy a 2-es alapu számrendszerben felirt számot a $8 = 2^3$ alapu számrendszerbe átirni nem jelent mást, mint 3-3 bináris számjegyet összefogva - röviditésképen - egy uj számjeggyel jelölni. A tizes számrendszerben ennek az felel meg, amikor egy hosszu számot 3-3 számjegyből álló csoportokra bontva irunk 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 uj jelet vezetnénk be, akkor azt mondhatnók, hogy a megadott számot az $1000 = 10^3$ alapu számrendszerben irtuk fel.

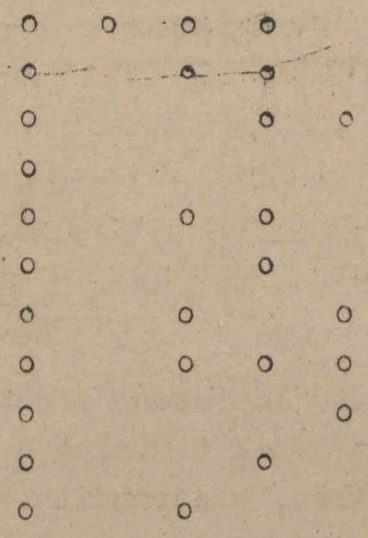
A kettes számrendszerben ilyen "uj jelek" gyanánt a 0-7ig terjedő decimális számjegyeket használjuk a három bináris jegyből álló kombinációk jelölésére, és ezt ugy mondjuk, hogy a bináris számokat 8-as számrendszerben irjuk fel. Pl. a gépben

0	1 1 0	0 1 1	0 0 0	1 1 0	0 1 0	1 0 1	1 1 0	0 1 0	1 0 0

alakban ábrázolt bináris számot röviden

+ 6 3 0 6 2 5 7 1 2 4

formában irhatjuk fel és a szalagon ennek megfelelően a következő módon ábrázolhatjuk:

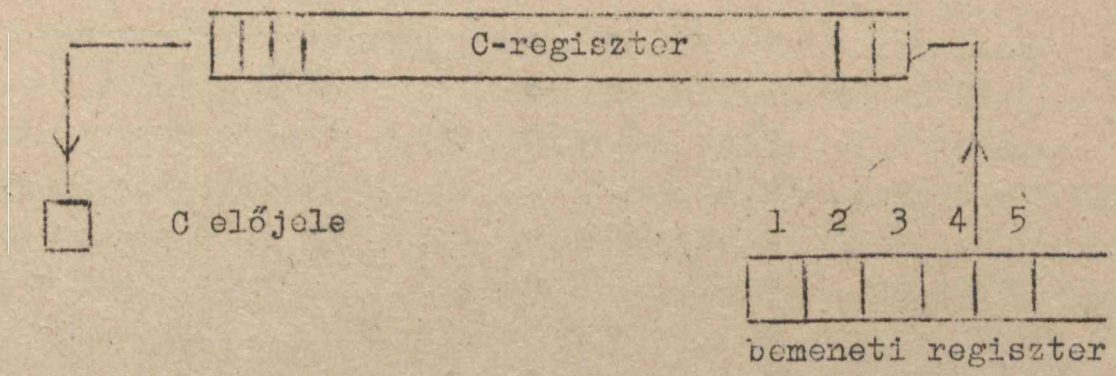


Egy binárisan megadott szám felirása tehát 11 sort vesz igénybe.

Tekintettel arra, hogy a szabványos távgépirók a számokat nem az M-3 számára szükséges, hanem a nemzetközi távgépiró-szabvány szerinti jelkombinációk alapján lyukasztják, a gépbe speciális "átkodoló" van beépitve, amely a szalagra lyukasztott kombinációkat a fenti kombinációkká alakitja át.

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

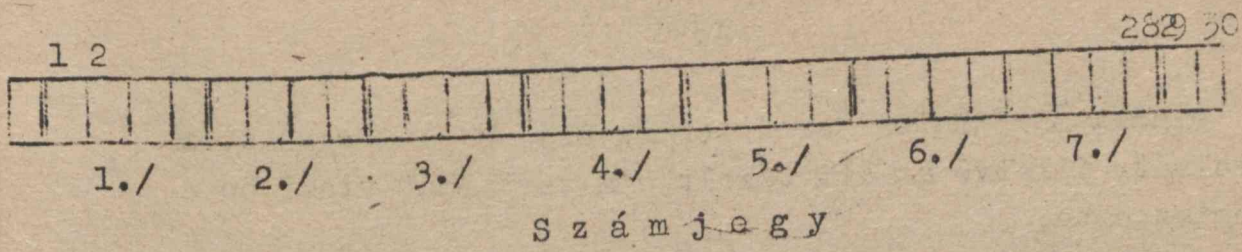
Tekintsük először a binárisan megadott számok esetét. Ekkor a bementi regiszter utolsó három helyértékén találjuk azokat a számjegyeket, amelyeket be kell vinni a C regiszterbe. Ezért a C regisztert és a bemeneti regisztert a következő módon "egyesitve" képzeljük el.



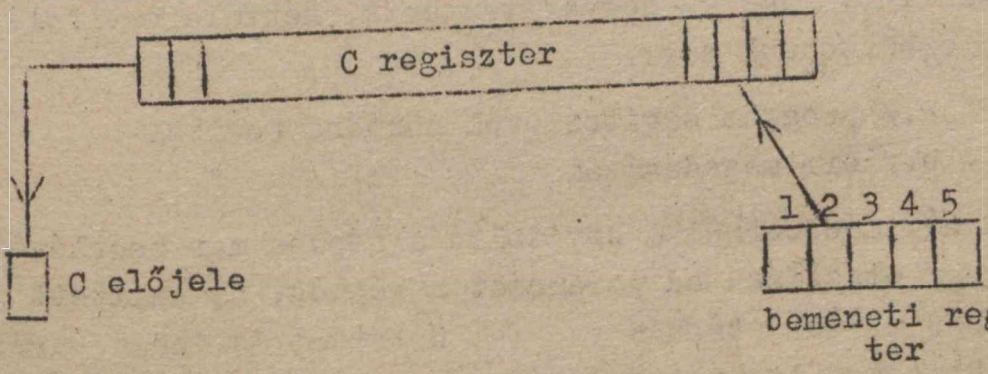
Igy, ha balra való léptetést végzünk, a bemeneti regiszter ötödik helyértékének tartalma a negyedikre, a negyedik tartalma a harmadikra, a harmadik helyérték tartalma pedig C_{30}-ba megy át, a C regiszter tartalma balra lép, végül C_1 tartalma átmegy C előjelének helyére.

Igy három ilyen léptetés eredményeképpen a bevitelre 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 először leolvasott - az előjelnek megfelelő - sor három bináris jegye /1, 1, előjel/ a végrehajtott tizszer három léptetés eredményeképen 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 "kitólja 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 tizes számrendszerben megadott számok bevitelénél. Tekintettel arra, hogy egy decimális jegy bináris ábrázolása négy helyértéket vesz igénybe, egy 31 jegyü szóban az előjelen kivül 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 müveleben teket kettes számrendszerben felirt számokkal végzi, a fenti módon - u.n. binárisan kódolt decimális - alakban felirt számot át kell alakitani kettes számrendszerbeli megfelelőjévé.

Ezt az átalakitást a gép már nem végzi el automatikusan, erre külön program segitségével kell a gépet utasitani. Ilyen "számrendszerváltó" programokról egy későbbi fejezetben lesz szó/.

A C regiszterbe bevitt számnak a gép memóriájába való beirásánál fontos szerepet játszik a központi vezérlő egységben helyetfoglaló szelekciós regiszter /SZR/. Ebben tároljuk mindig annak a memóriarekesznek a cimét, amelyet beirás vagy kiolvasás céljából meg akarunk keresni. Amennyiben ugyanis a memória "irás" vagy "olvasás" utasitást

kap, mindig az SZR-ben tárolt címre irja be a C-regiszter tartalmát, illetve az itt tárolt cim tartalmát viszi be a C-regiszterbe.

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

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

a./ program segitségével történő bevitel
b./ cim megadásával " "

Az a./ módszernél a bevitelre a gépben már bentlévő program egy utasitása ad parancsot a gépnek, az emlitett cim magában az utasitásban van feltüntetve, és innen kerül - a későbbiekben ismertetendő módon - az SZR-be. A bevitelre kerülő szó leolvasása után a gép a szalagon egy "szó vége" jelet talál, melynek hatására "irás" utasitást ad a memóriának.

A b./ módszernél a cimet magán a szalagon adjuk meg: minden szó elé odairjuk, hogy milyen cimre akarjuk elhelyezni. Itt tehát a gép először a 12 helyértékből álló cimnek megfelelő négy sort olvassa le a szalagról és helyezi el a C regiszter utolsó 12 helyértékén. Ezután a szalagon egy "cim vége" jel következik, amelynek hatására a C_{19}-C_{30} helyértékek tartalma tehát épen a most beirt cim 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 bevitelt egy gombnyomás inditja. Hátránya, hogy csak binárisan megadott anyag bevitelénél alkalmazható, mert a cimek bevitele egyébként nem lehetséges.

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

1./ A decimálisan megadott - 0,7968345 számot

```
o o o o o /-/
o   o o o /7/
o o     o /9/
o   o o   /6/
o o       /8/
o     o o /3/
o   o     /4/
o   o   o /5/
    o o   /"szó vége"/
```

alakban perforáljuk,a bevitel csak az a./ módszer szerint lehetséges.

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

```
o         /0/
o   o     /4/
o       o /1/
o     o   /2/
        o /"cim vége"/
o o o o   /+/
o   o o o /7/
o   o o   /6/
o   o     /4/
o       o /1/
o         /0/
o   o   o /5/
o     o o /3/
o     o   /2/
o       o /1/
o   o     /4/
    o o   /"szó vége"/
```

A számok kiirása, ugyancsak a C-regiszteren keresztül történik, tizes vagy "nyolcas" számjegyként. A C-regiszter első négy illetve három helyértéke van összeköttetésben a kimenő távgépiróval, és a C-regiszter tartalmának a fentiek szerinti balra történő léptetései biztositják azt, hogy ide minden helyérték tartalma eljusson. Különleges áramkörök gondoskodnak a sorváltás elvégzéséről, soronként 1,2,3,4, vagy 5 szó leirása után, előzetes beállitás szerint.

Amennyiben tizes számrendszerben akarjuk az eredményeket kiirni, egy "számrendszer-váltó" program segitségével gondoskodni kell arról, hogy a kiirandó szám a 45. oldalon ismertetett binárisan kódolt decimális formában kerüljön a C regiszterbe.

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

2.4. A vezérlőegység és müködése.

Az eddigiekben megismertük az egyes müveleteknek, az ugynevezett gépi alapmüveleteknek 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 utasitás végrehajtásának menetét a központi vezérlő egység vezérli.

Ez három részből áll:
1./ impulzus elosztó
2./ szelekciós regiszter /SzR/
3./ inditó regiszter /v.utasitásszámláló/.

Az M-3 gép utasitásai a következő alakuak:

±	Müveleti jel	I.cim	II.cim
1 bit	6 bit	12 bit	12 bit

Az első hat helyértéken elhelyezkedő müveleti jel azt mutatja, meg, hogy az adott utasitás milyen müveleti elvégzésére utasitja a gépet. A 7-18 illetve 19-30. helyértéken elhelyezkedő első ill. második cim a memóriának azon rekeszeit adja meg, amelyeknek tartalmával végezzük el a müveleteket.

Az utasitást tartalmazó szó előjele az utasitás végrehajtását nem befolyásolja.

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

Az utasitások - mint tudjuk - a gép memóriájában vannak elhelyezve. A végrehajtásra soronkövetkező utasitás cimét mindig a központi vezérlő egység 12 helyértékes inditó regisztere tárolja.

Az utasitás végrehajtásának ütemei: /

1. ütem: az inditó regiszter tartalma átmegy a szelekciós regiszterbe /SzR/ és "olvasás" utasitás megy a memória felé,

2. ütem: megtörténik az SzR-ben tárolt cim megkeresése és az olvasás; a kiolvasott szó /azaz a végrehajtandó utasitás/ a C regiszterben helyezkedik el,

3. ütem: a C-regiszter első 6 helyértékének tartalma /azaz a müveleti jel/ átmegy a müveleti programadó megfelelő regiszterébe a müveleti regiszterbe: a 7.-18. helyértékek tartalma /első cim/ pedig az SzR-be, és "olvasás" parancs megy a memória felé. Ugyanakkor az inditóregiszter tartalmához 1 adódik hozzá, annak biztositására, hogy a következő nyolcas ciklusban a következő rekeszben tárolt utasitás kerüljön végrehajtásra.

4. ütem: miután a memória az előző ütemben kiadott "olvasás" parancs hatására - megtalálta az SzR-ben lévő címnek /az utasitás első címének/ megfelelő rekeszt, a C-regiszter 19.-30. helyértékei /tehát a második cim/ mennek át az SzR-be. Ezután következik be az előzőleg megtalált első cim tartalmának beirása a C-regiszterbe, melynek eddigi tartalmára /az utasitásra/ 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 cim /az utasitás 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 müveleti programadó felé a művelet elvégzésére.

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

A vezérlő egység aszinkron működésü, ami azt jelenti, hogy az egyes ütemeket az előző ütemekben kijelölt feladat elvégzését jelző "válasz" beérkezése inditja. Igy 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 inditásával. Ugyanigy a következő utasitás végrehajtási ciklusának első ütemét a művelet befejezése után /a nulladik ütemben/ kiadott "irás" utasitásnak megfelelő válasz jel inditja el.

Minden egyes müvelet eredménye a B vagy C regiszterben keletkezik és a müvelet befejezése után a másik regiszterbe is átmegy: igy végeredményben a B és C regiszterekben egyaránt megtalálható. Ezért, amikor a müvelet befejezése után "irás" parancs megy a memória felé, akkor ennek hatására a müvelet eredménye iródik be az SzR-ben jelenleg lévő cimre, azaz a végrehajtott utasitás második cimére. Fenti tipusu utasitás a gépnek mind az öt alapmüveletére megadható. Ezeket a müveleti jel második három számjegye különbözteti meg egymástól. Itt az első három számjegy 0, ami azt jelzi, hogy az utasitásnak a fent ismertetett alapváltozatáról van szó:

müvelet szinboluma	müveleti 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
∧	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 felirt utasitás végrehajtása azt jelenti, hogy a 0012 és 0031 memória rekeszek tartalmainak összege beiródik a 0031 rekeszbe.

A fenti utasitások végrehajtása azzal a hátránnyal jár, hogy a második cimre beiródó eredmény eltörli az ott tárolt számot /az adott müveletben résztvevő egyik számot/. Sokszor fordul elő olyan eset, hogy a számitás során erre a számra még szükségünk van, ezért a müvelet eredményét

Az utasitások végrehajtásának általános menete.

Ütem	Inditó regiszter	Szelekciós regiszter	Olvasás parancs	C-regiszter	Müveleti regiszter	A-regiszter	O-regiszter
0	végrehajtandó utasitás cime	0		/előző müvelet eredménye/	/előző művelet jele/	0	előző művelet eredménye.
1	"	végrehajtandó utasitás cime	van	"	"	0	"
2	"	0		végrehajtandó utasitás	"	0	"
3	végrehajtandó utasitás cime + 1	az utasitás első cime	van	"	az utasitás müveleti jele.	0	"
4	"	az utasitás második cime		az első cim tartalma	"	0	"
5	"	"	van	"	"	Az első cim tartalma	"
6	"	"		a második cim tartalma	"	"	"
7	"	"		"	"	"	a második cim tartalma
0	a végrehajtott utasitás cime + 1	A müvelet elvégzése és az eredmény beirása. 0		a müvelet eredménye	a végrehajtott müvelet jele	0	müvelet eredménye

valahová máshová kell beirni.

A gépen minden egyes müveletnél lehetőség van a müvelet végrehajtása után következő "irás" utasitás mellőzésére is.

Az ilyen utasitások végrehajtása után tehát a müvelet befejezését jelző "válasz" azonnal inditja a következő ciklust. Ezeket az utasitásokat a müvelet szimboluma után tett vesszővel /,/ jelöljük, és a müveleti jel harmadik helyértékére irt 1-el különböztetjük meg az előzőktől.

müvelet szinboluma	müveleti jel 6 jegye	röviden /8-as számrendszerben/
+,	0 0 1 0 0 0	1 0
-,	0 0 1 0 0 1	1 1
:,	0 0 1 0 1 0	1 2
x,	0 0 1 0 1 1	1 3
^,	0 0 1 1 1 0	1 6

Természetesen minden ilyen "vesszős" utasitás után gondoskodni kell arról, hogy a müvelet eredménye, - amely mint tudjuk a B és C-regiszterekben maradt meg, - valamilyen módon felhasználható legyen, Ha t.i. a "vesszős" utasitás után az eddig ismert utasitások valamelyikét irnók, akkor a müvelet eredménye elveszne, mert a következő utasitás végrehajtása során a C regisztert maga az utasitás, majd az első és második cim tartalma töltené meg, és az utóbbi a B regiszterbe is átmenne. Ezért szükség van olyan utasitásokra, amelyek a B regiszter tartalmát változatlanul hagyják és igy az előző müvelet eredményével végeznek müveleteket. /A müvelet eredményét a C regiszterben nem lehetne változatlanul hagyni, mert erre a regiszterre az utasitás és az első cim tartalmának kiolvasásánál szükség van!/ Az ilyen utasitások végrehajtásánál tehát az ötödik ütemben a központi

vezérlő egység nem ad parancsot a második cim 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 igy az utasitásban kijelölt müveletet a gép ezen szám és az utasitás első ciménck - az ötödik ütemben a regiszterbe vitt - tartalma között végzi el, és a művelet eredményét a nulladik ütem "irás" parancsa alapján beirja a második cimre. Ezek az utasitások természetesen olyan utasitások után is használhatók, melyeknél a művelet eredménye beirodott a második cimre, hiszen a B regiszterben az eredmény ebben az esetben is megmarad. Ilyen utasitás is minden gépi alapmüveletnél található, ezeket a művelet jelzése elé tett nyillal, illetve a müveleti jel második számjegyébe irt számmal 1-el jelöljük:

művelet szinboluma	müveleti 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
↓A	0 1 0 1 1 0	2 6

Lehetőség van a fent emlitett két eset összekapcsolására is: A nyillal és vesszővel ellátott utasitásoknál a művelet az előző művelet eredménye és az első cim tartalma között végezzük el és az eredmény nem iródik be a memóriába, csak a B regiszterben marad meg. Ezeknél az utasitásoknál a második cimre bármit irhatunk, hiszen az nem kerül felhasználásra.

művelet szinbóluma	műveleti jel 6 számjegye	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 számjegyét. Ennek szerepe a következő:

1. ha a műveleti jel első számjegye 1 és a harmadik számjegy 0, akkor a gép a művelet eredményét, - amellett, hogy a második címre beirja - ki is nyomtatja a távgépiró segitségével. Ezt a művelet jelzése utáni "n" betüvel jelöljük.

2. ha a műveleti jel első és harmadik számjegye 1, akkor a gép a műveletet a megfelelő számok helyett azok abszolut értékeivel végzi el. Ezt ugy jelöljük, hogy a művelet szimbolumát két függőleges vonal közé irjuk /pl. |+|,/.

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

Összefoglalva: minden egyes művelethez az alábbi nyolc utasitás tartozik /ж az öt gépi "alapművelet" bármelyikét jelentheti/.

Müvelet szimboluma	müveleti jel első 3 számjegye	a müveletben résztvevő számok	az eredmény elhelyezése
ж	0 0 0 /0/	első második cim tartalma	B,C regiszterbe második cimre
ж,	0 0 1 /1/	" "	" /
↓ ж	0 1 0 /2/	első cim tartalma B-regiszt. tartalma /előző müv. eredménye/.	" második cimre
↓ ж,	0 1 1 /3/		" " /
ж n	1 0 0 /4/	első második cim tartalma	" második cimre és kiirás a távgépirón.
|ж|,	1 0 1 /5/	első második cim tartalmának abszolut értéke.	" /
↓ ж n	1 1 0 /6/	első cim tartalma B-regiszt. tartalma /előző müv. eredménye/.	" második cim és kiirás a távgépirón.
↓ |ж|,	1 1 1 /7/	első cim tartalmának abszolut értéke. B regiszt.	" /

A müveleti jel utolsó három bináris - azaz második 8-as számjegye a megadott müveletet 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 müveleti jel második nyolcas helyértékén álló 7 számjegy /111/ a gép beviteli utasitásait jelzi /lásd a bevitel a,/ módszerét az előző előadáson. Ezeknél a gép a szokásos módon elvégzi a nyolcas ciklust, majd a 07 és 27 müveleti jelek esetén a nulladik ütemben - a kijelölt müvelet végrehajtása gyanánt - beviszi a perforált szalagon a soron-

következő szót az utasitás második címére /a ciklus végrehajtása során ugyanis ez kerül utoljára az SzR-be/.

A beviteli utasitások szimboluma "Be". A 07 és 27 müveleti jelek teljesen egyenértékűek.

Itt kell megemlitenünk azt is, hogy a gép jelenlegi állapotában a müveletek közül csak összeadás és kivonás után végezhető kiirás, tehát a 42, 43, 46 ill. 62, 63, 66 utasitások nem használhatók.

A 05, 15, 45 és 55 jelü utasitások a számoknak a memória egyik rekeszéből a másikba való átadását végzik. Ezek végrehajtásánál az ötödik ütem "olvasás" parancsa helyett egy "irás" parancs megy ki, ennek megfelelően a hatodik ütemben megtörténik a C regiszterben lévő számnak - azaz az első cim tartalmának, amely a negyedik ütemben C-be került - beirása az SzR-ben lévő második címre. Ezután a hetedik ütemben a C tartalmának B-be való szokásos átvitele biztositja, hogy a "müvelet" eredménye" az a szám tehát, amelyet az utasitás áthelyezett - a B regiszterben is megmaradjon és igy a következő utasitás 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 utasitások.

A gép a program utasitáasit általában címeik sorrendjében hajtja végre /ezért növeli 1-gyel minden ciklusban az inditóregiszter tartalmát/. Néha szükség van azonban arra, hogy eltérjünk ettől a sorrendtől, és előirjuk, hogy valamely utasitás végrehajtása után a gép a számitást egy tetszőleges, megadott utasitás végrehajtásával folytassa. Azokat az utasitásokat, amelyekkel ezt elvégezzük vezérlés-átadó vagy ugróutasitásoknak nevezzük. Ezeknél az utasitás végrehajtásának alapciklusa ugy módosul, hogy elmarad az

első cim tartalmának kiolvasása, ehelyett az első cim a szelekciós regiszterből/ahova a harmadik ütemben átment/ átmegy az inditóregiszterbe, és igy a következő ciklusban az ezen cimen tárolt utasitás kerül végrehajtásra.

Igy történik minden olyan utasitás végrehajtása, ahol a müveleti jel második /nyolcas/ számjegye 4. Ezek közül azokat használjuk, amelyeknél a müveleti jel második bináris számjegye 1-es /ezek a müveletnél a "nyilas" utasitásoknak felelnek meg/, tehát itt a második cim tartalmának kiolvasása is elmarad és a hetedik ütemben B tartalma megy át a C-be és nem forditva.
Ezek:

24	/010100/
34	/011100/
64	/110100/
74	/111100/

A 24 és 64 jelü utasitások végrehajtásánál a ciklus fenti módositásain kivül a hetedik ütem végén, miután a B regiszter tartalma átmegy a C-be, egy "irás" parancs megy ki és a 64 jelü utasitásnál még egy kiirási parancs is a távgépiró felé. Ezek az utasitások tehát:

1. az inditóregiszterbe beirják az első cimet azaz "átadják a vezérlést" az ezen a cimen tárolt utasitásnak;
2. az előző müvelet eredményét beirják a második cimre /64 még ki is iratja a távgépirón/.

A 24 jelü utasitás szimboluma Ul, a 64-é Uln, jelezve, hogy az első cimen tárolt utasitásra való "ugrásról" van szó.

A 34 és 74 jelü utasitások végrehajtásánál a hetedik ütemben, "irás" parancs nem megy ki, ehelyett, amennyiben a B regiszter tartalmának előjele pozitiv, az SzR tartalma átmegy az inditóregiszterbe. Ez azt jelenti, hogy ezeknél a vezérlés-átadást B előjelétől függően vagy az első, vagy

a második címre végzi el a gép: ha B előjele negativ, akkor a negyedik ütemben bevitt első cim marad az inditóregiszterben és ez lesz a következő végrehajtandó utasitás címe, ha B előjele pozitiv, akkor ezt a hetedik ütemben a második címre változtatja át. Ezenkivül tekintve, hogy ezeknél az utasitásoknál semmiféle müveletet, vagy irást nem végzünk, ami a következő utasitást inditaná, a nulladik ütemben a gép inditja a következő ciklus első ütemét.

A 74 jelü utasitás végrehajtásánál - tekintve, hogy az ismertetett eltérésektől eltekintve a nyolcas ciklust a gép ugy végzi el, mint más olyan utasitásoknál, melynek első számjegye 7, ezek pedig a számok abszolut értékével végeznek müveleteket - a gép az A és B regiszterek előjelét pozitivra változtatja, igy B előjele mindenképen pozitiv lesz, tehát a másik cim kerül az inditóregiszterbe. Ezért ez az utasitás a 24 és 64 jelüekhez hasonlóan - feltétlen ugrást jelent, mégpedig ezektől eltérően a második címre. Szimbóluma: U2.

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

Ha az előző müvelet eredménye nulla, akkor a helyzet a következő: Ha valamely müvelet eredménye nulla, ennek előjele lehet pozitiv is, negativ is /pozitiv ill.negativ 0/. Láttuk ugyanis az egyes müveleteknél, hogy a gép a számok abszolut értékét és előjeleit külön kezeli, pl. kivonásnál /ill.különböző előjelü számok összeadásánál/ B előjelét csak akkor kellett megváltoztatni, ha a kivonandó abszolut értékben nagyobb volt a kisebbitendőnél. Igy ha a két szám megyegyezik, akkor B előjelében megmarad a második címen lévő szám - a kisebbitendő - előjele. Tehát:

$$/+a/ - /+a/ = /+a/ + /-a/ = +0$$
$$/-a/ - /-a/ = /-a/ + /+a/ = -0$$

ahol az első helyen álló szám van a második cimen /illetve "nyilas" müveleteknél ez az "előző müvelet eredménye"/, mert ez van a B regiszterben.

Szorzásnál és osztásnál a 0-eredmény előjelét is az ismert előjelszabály adja meg, mig a logikai szorzásnál, mely az előjelet nem változtatja, az eredmény előjele mindenképen 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 jelü utasitások nyolcas ciklusának végrehajtása is a 4-jelü utasitások fent ismertetett végrehajtása szerint történik, ezeknél azonban a következő utasitás végrehajtását semmi nem inditja el, ezek tehát a gép megállási utasitásai. Ugyancsak megállást eredményeznek a 17, 37, 57 és 77 jelü utasitások is. A megállás után az egyes regiszterek tartalma a végrehajtott nyolcas ciklusnak megfelelően más és más lesz.

A megállási utasitásokon kivül természetesen megállitható a gép a vezérlőtábla megfelelő kapcsolójával is. Ezenkivül van az M-3 gépen egy olyan megállitási lehetőség is, hogy beállitunk egy cimet a vezérlőtábla megfelelő 12 helyértékes "beállitóregiszter"-ében és a gép megáll, mihelyt az inditóregiszterben vagy a szelekciósregiszterben ez a cim megjelenik /azt, hogy melyik regiszter szerint áll meg, egy külön kapcsoló szabályozza/.

Igy tehát megállithatjuk a gépet egy adott utasitás első végrehajtásánál, vagy egy adott rekesz első felhasználásánál. Ez a megállitási lehetőség bonyolult programok gépi ellenőrzésénél hasznos.

2.6. Az utasitások végrehajtásának időtartama.

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

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

müveleti jel	memóriához való fordulások.	végrehajtás kb. ideje msec-ben.
0 x	utasitás első cim második cim eredmény beirása } kiolvasása	40
1 x 5 x	utasitás első cim második cim } kiolvasása	30
2 x	utasitás első cim } kiolvasása eredmény beirás	30
3 x 7 x	utasitás első cim } kiolvasása	20
24	utasitás kiolvasása előző müvelet eredményének beirása.	20
34 74	utasitás kiolvasása	10
05 15	utasitás első cim } kiolvasása az első cim tartalmának beirása.	30

Azoknak az utasitásoknak a végrehajtási idejét, molyeknél számok bevitele, vagy kiirása történik /07, 27, ill. 3x, 6x, 64, 45, 55/ a bevitel ill. kiirás sebessége határozza meg /kb. 2 sec/.

3. Fejezet.

A PROGRAMKÉSZITÉS GYAKORLATI MÓDSZEREI.

3.1. A direkt programozás bemutatása példákon.

A korábbi fejezetekben megismertük az M-3 gép utasitásrendszerét. Most olyan egészen egyszerü feladatok programozását mutatjuk be, amelyeknél a gép az utasitásokat a tárolás sorrendjében hajtja végre, azaz nincsenek ugró utasitások. Ezt nevezzük direkt programozásnak.

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

Tudjuk, hogy az M-3 gép csak 1-nél kisebb abszolut értékü számokkal tud dolgozni. Programozásnál ügyelni kell arra, hogy mind az eredmény, mind a részleteredmények a /-1, 1/ intervallumba kerüljenek. Ennek érdekében bizonyos numerikus átalakitásokat kell végezni, amelyekről részletesen a későbbiekben lesz szó. Feltételezzük, hogy a most következő példákban ez a követelmény ki van elégitve.

Mielőtt a programozáshoz hozzákezdenénk, néhány jelölést kell bevezetni.

Jelölések:

1. Valamely x cim tartalmát /x/-szel jelöljük. Tehát /x/ = a azt jelenti, hogy az a mennyiséget az x cimen találhatjuk a memóriában.

2. Valamely mennyiség cimét ⟨a⟩ -val jelöljük. Tehát a fenti mennyiségek esetében ⟨a⟩ = x.

3. Valamely mennyiségnek egy adott cimre való elhelyezését a "⇒" jellel jelöljük. Pl.: a+b⇒n azt jelenti, hogy az a és b mennyiség összegét tegyük az n cimre.

Az áttekinthetőség kedvéért a program összeállitásánál a következő táblázatot használjuk:

PR

utasitás rekesze	a müvelet szimboluma	jele	1.cim tartalma	1.cim	2.cim tartalma	2 cim
/1/	/2/	/3/	/4/	/5/	/6/	/7/

A program összeállitását 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 utasitásrendszere alapján. A gépbe csak az /1/, /3/, /5/, /7/ oszlopok kerülnek. Az utasitás a /3/, /5/, /7/ oszlop tartalma, az inditóregiszter tartalma az /1/ oszlopban álló szám, amikor a gép végrehajtja az illető utasitást. Cim megadásával történő bevitel esetén ezt a perforált szalagra kell lyukasztani; program segitségével történő bevitelnél a program gondoskodik az utasitások megfelelő elhelyezéséről.

Most áttérünk egyszerü példák kidolgozására. Tegyük fel, hogy az a,b,c,d számok rendre a 0010, 0011, 0012, 0013 cimeken vannak.

Azaz

$\langle a \rangle = 0010$ $\quad\quad$ $\langle c \rangle = 0012$

$\langle b \rangle = 0011$ $\quad\quad$ $\langle d \rangle = 0013$

illetőleg

/0010/ = a $\quad\quad$ /0012/ = c

/0011/ = b $\quad\quad$ /0013/ = d

A későbbiekben inkább az utóbbi felirást használjuk. A számitások végeredményét a 0020-as rekeszbe helyezzük el. A programokat a 0100-as memória rekesztől kezdve tároljuk és feltételezzük, hogy az inditóregiszter tartalma 0100, azaz a gép éppen az ezen a cimen tárolt utasitás végrehajtását kezdi el.

1. Példa.

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

utasitás rekesze	a művelet szimbóluma	a művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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 utasitásban azért használunk "+,"-t, mivel lehetséges, hogy a számitás folyamán a b számra még szükség lesz, tehát nem törölhetjük ki a 0011 rekeszből. Azonkivül az a+b részleteredményre a továbbiakban nem lesz szükségünk, tehát nem kell tárolni a memóriában. Továbbá, amint már láttuk, a "+," utasitás kevesebb időt vesz igénybe, mint a"+". Az 0101-es utasitásban a "↓+," utasitást használunk. Az előző lépés következtében ennek az utasitásnak szükségképen

"nyilas" utasitásnak kell lennie, mivel az előző művelet eredményével számolunk tovább, amely az előző "+," utasitás miatt a memóriában nem található. Mivel pedig erre a részleteredményre sem lesz a továbbiakban szükségünk, "↓+," utasitást célszerü használni. Emlitettük már, hogy az ilyen tipusu utasitás veszi a legkevesebb időt igénybe, tehát használata igen előnyös.

A 0102-es utasitás egyértelmüen 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.

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

Itt célszerü a müveletek sorrendjét felcserélni és a számitást a következő sorrendben végrehajtani: bd + a-c=x.

utasitás rekesze	a művelet szimboluma	a művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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 felépitése teljesen hasonló az előzőhöz, azzal a különbséggel, hogy ott mindhárom utasitásban összeadás szerepel, itt pedig az elsőben szorzás, a harmadikban kivonás.

3. Példa. b+b c+c+c

$$x = a + 2b + 3c \Rightarrow 0020$$

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

Utasitás rekesze	a müvelet szimbóluma	a müvelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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 kiszámitása után ezt a mennyiséget félre kellett tenni, mig 3c-t kiszámitjuk. 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 utasitással helyezzük el a 0020-as rekeszbe. Nézzük meg ezt az utasitást közelebbről. Ez egy feltétel nélküli ugró utasitás, amelynek első cimében annak az utasitásnak a cime áll, ahová ugrani kell, második cimében pedig annak a memóriarekesznek a cime, ahova az előző müvelet eredményét be kell irni. A mi esetünkben csak az utasitás második cimére van szükség, ezért az első cimre a programban soron következő utasitás rekeszének cimét irjuk. Mindig igy járunk el, ha egy olyan mennyiséget akarunk elhelyezni a memóriába, amelynek kiszámitásánál az utolsó utasitás második cime egy olyan mennyiség rekeszének cime, amelyet nem szabad törölni. Ez esetben az utolsó utasitás vesszős utasitás és külön utasitás kell az eredmény elhelyezésére. Elvégezhetnők ezt ugy is, hogy a kiszámitott mennyiséghez, amely a B regiszterben található, 0-t adunk hozzá egy olyan "↓+" utasitás segitségével, amelynek a második cime 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 el kell végezni. Az "U1" utasitás sokkal gyorsabban hajtható végre.

4. Példa.

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

a./ változat

utasitás rekesze	a művelet szimboluma	a művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2 cim
/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 müveletek sorrendjét itt is fel kell cserélni. A kivonandó szorzatot célszerü először kiszámitani, mivel a kivonás nem kommutativ müvelet és csak abben a sorrendben lehet nyilas müveletet használni.

b./ változat.

Végezzük el a következő átalakitást. Emeljük ki d-t:

$$ad - cd = /a-c/\ d.$$

Végezzük el igy a programozást.

utasitás rekesze	a művelet szimboluma	a művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/1/	/2/	/3/	/4/	/5/	/6/	/7/
0100	-,	11	c	0012	a	0010
0101	↓x	23	d	0013	x	0020

Látjuk, hogy az átalakitás segitségével az utasitások 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.

$$x = /ab + c/ \quad /a - bc/ \Longrightarrow 0020$$

Ezt a feladatot is kétféleképen oldjuk meg.

a./ változat.

0021 = munkarekesz.

Utasitás rekesze	a művelet szimbóluma	jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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	U1	24	-	0104	bc	0021
0104	-,	11	bc	0021	a	0010
0105	↓x	23	ab+c	0020	x	0020

b./ változat.

Átalakitjuk a feladatot.

/ab + c/ /a -bc/ = /ab + c/ a- /ab + c/ bc.

utasitás rekesze	a művelet szimbóluma	jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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 utasitások száma mindkét esetben megegyezik és mindkét esetben fel kell használni egy uj munkarekeszt. Az a./ változatban az egyik utasitás "Ul", tehát ezt a változatot a gép valamivel gyorsabban hajtja végre.

6. Példa.

$$x = \frac{abc+d}{4/a+b/} \Longrightarrow 0020 \qquad /0016/ = 1/4$$

utasitás rekesze	a müvelet szimboluma	a müvelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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	↓+,	30	d	0013	-	-
0105	↓:	22	4/a+b/	0020	x	0020

A 0103 és 0104 utasitásokban nincs kihasználva, hogy a gép kétcimü, viszont ezek az utasitások hajthatók a leggyorsabban végre, mivel csak egyszer kell a memóriához fordulni.

7. Példa.

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

utasitás rekesze	a müvelet szimboluma	a müvelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
/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 abszolut értékkel végzett müveleteket.

Feladat:

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

3.2. Direkt programozás feltételes ugró utasitásokkal.

Példák.

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

Bonyolultabb programoknál célszerü a számitás 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 állitsuk meg a gépet.

Tervezzük meg először is a számitás menetrendjét. /L.a I. táblát./

A menetrend a géptipustól független. Az utasitásrendszer jellege alig befolyásolja. Az összetartozó utasitáscsoportokat egy blokkba irjuk.

Ha a gép egy utasitáscsoport eredményétől függően két lehetőség között választ, az utasitáscsoportnak megfelelő blokkot egy oldalra állitott rombusszal ábrázoljuk. Az ilyen "választó" blokknak mindig két kimenete van. A választó blokkba beirjuk 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 utasitás, az "FU" ugy 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, ugy érhetjük el, hogy a B regiszter tartalmának abszolut értékét kivonjuk nullából. Az eredmény akkor és csak akkor lesz negativ, ha a B regiszter tartalma nem volt 0.

Ujból felhivjuk a figyelmet az M-3 gép egy különös sajátosságára. Ha valamely művelet eredménye 0, akkor ez a nulla pozitiv vagy negativ lehet a következők szerint:

a./ szorzásnál és osztásnál az előjel-szabály érvényesül /egy művelet eredménye a gép szerint akkor is nullával egyenlő, ha abszolut értékben kisebb mint 2^{-30}/;

b./ összeadásnál, kivonásnál és logikai szorzásnál a nulla a második cimen lévő szám előjelét kapja;

c./ abszolut értékes müveleteknél mindig pozitiv.

A mi esetünkben a nulla mindig pozitiv. Készitsük el a programot a már ismertetett menetrend alapján.

Program.

/0010/ =a; /0011/=b; /0012/=c; /0013/=0; x=a+b+c ===> 0020.

A programot a 0100 memóriarekesztől kezdve helyezzük el.

utasitás rekesze	művelet szimboluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
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

Számitsuk ki az $y = \frac{A+Bx}{C+Dx} + \frac{x^2}{2}$ függvény helyettesitési értékét valamely $x = x_0$ helyen. Ha $C + Dx_0 = 0$, akkor állitsuk meg a gépet. A menetrendet a II. táblán ábrázoltuk.

Program:

/0010/ = x_0; /0013/ = C;

/0011/ = A; /0014/ = D;

/0012/ = B; /0015/ = 2^{-30};

/0016/ = 1/2

y_0 ⟹ 0020

Munkarekesz: 0020

utasitás rekesze	művelet szimboluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
0100	x,	13	D	0014	x_o	0010
0101	↓+	20	C	0013	Dx_o+C	0020
0102	↓|-| ,	71	2^{-30}	0015	-	-
0103	FU	34	-	0104	-	0105
0104	megállás					
0105	x,	13	B	0012	x_o	0010
0106	↓+,	30	A	0011	-	-
0107	↓:	22	$C+Dx_o$	0020	$\frac{A+Bx_o}{C+Dx_o}$	0020
0110	x,	13	x_o	0010	x_o	0010
0111	↓x,	33	1/2	0016	-	-
0012	↓+	20	$\frac{A+Bx_o}{C+Dx_o}$	0020	y_o	0020

Ebben a példában sincs baj a negativ nullával, hiszen abszolut-értékes művelet segitségével választunk.

3. Példa.

Max /a,b,c,/ $\Longrightarrow$ 0020, azaz az a,b,c, számok közül a legnagyobbat helyezzük el 0020-ban. Tervezzük meg itt is a számitás menetrendjét. /L. a III. táblát./

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

Azaz

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

A következő lépésnél y-t összehasonlitjuk c-vel és 0020-ba az y-t vagy a c-t irjuk be, aszerint, hogy a kettő közül melyik a nagyobb. Azaz max /y,C/ $\Longrightarrow$ 0020. A választást itt is az "FU" feltételes ugró utasitással hajtjuk végre.

Ezt ebben a feladatban közvetlenül alkalmazhatjuk, hiszen itt negativ és pozitiv szám között kell választani. A negativ 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.

utasitás rekesze	művelet szimboluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
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	-	0112	-	0110
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						

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 egyenletrendszert.

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

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

Arról, hogy ezek az értékek valóban kielégitik az egyenletrendszert, behelyettesitès segitségével 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

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

Ha D=0, azaz az egyenletrendszer nem oldható meg egyértelmüen, a gépet megállitjuk.

A számitás menetrendjét a IV. táblán láthatjuk.

Program:

/0010/ = a_{11}; /0011/ = a_{12}; /0012/ = a_{21}; /0013/ = a_{22};

/0014/ = b_1; /0015/ = b_2; /0016/ = 2^{-30};

$x_1 \Longrightarrow 0020$; $x_2 \Longrightarrow 0021$.

Munkarekesz: 0020.

A program a 0100 memóriarekesztől indul.

utasitás rekesze	művelet szimboluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
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	$a_{11}a_{22}-a_{12}a_{21}$	0022
0104	↓|-|,	71	2^{-30}	0016	-	-
0105	FU	34	-	0121	-	0106
0106	x,	13	b_2	0015	a_{12}	0011
0107	U1	24	-	0110	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					

Feladatok:

1. Készitsünk programot a szabadesés utképletéhez:

$$y = v_0 t + \frac{g}{2} t^2 .$$

Konstansok: /0010/ = v_0; /0011/ = t; /0012 = g ;

/0013/ = $\frac{1}{2}$.

Munkarekesz: 0021.

$y \Longrightarrow 0020$

A programot a 0100-as cimnél kezdjük.

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

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

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

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

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

$y \Rightarrow$ 0060

A programok a /0200/-as címnél kezdődjenek.

5. Számitsuk ki a következő kifejezést:

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

Legyen:

/0010/ = a_1; /0013/ = b_1;
/0011/ = a_2; /0014/ = b_2;
/0012/ = a_3; /0015/ = b_3;

$x \Rightarrow$ 0020.

3.3. Egyszerü ciklikus programok; a ciklusutasitások változatlanok.

Emlitettük az előzőkben, hogy ciklusnak a program olyan részét nevezzük, amely ugró utasitások segitségével 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:

$$y = x^5 \Longrightarrow 0032 \qquad \langle x \rangle = 0010$$

A program összeállitásánál a következő konstansokra lesz szükségünk:

$$/3.2^{-30}/ = 0011, \quad \langle 2^{-30} \rangle = 0012$$

Az utasitásokat a 0050 memóriarekesztől kezdve helyezzük el.

Program:

utasitás rekesze	művelet szimbóluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
0050	Á	05	x	0010	x	0032
0051	x	03	x	0010	x^k	0032
0052	-	01	2^{-30}	0012	3.2^{-30}	0011
0053	FU	34	-	0054	-	0051

Ebben a programban a 0051-es utasitást hajtjuk többször végre. A 0052 utasitás eredményeképen mindaddig pozitiv számot kapunk, amig a 0032 memóriarekesz tartalma x^5 nem lesz. Ha már megkaptuk ezt a végeredményt, a 0052 utasitás végrehajtása után negativ szám képződik és a 0053 utasitásban álló FU kiugratja a gépet a ciklusból.

Állitsuk össze ennek alapján a hatványozás programját általános esetben.

$$y = x^n \Rightarrow \underset{2^{-30}}{0032} \quad \langle x \rangle = \underset{0012}{0010}; \quad \langle /n-2/.2^{-30} \rangle = 0011;$$

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

Program:

utasitás rekesze	művelet szimboluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
0050	A	05	x	0010	x	0032
0051	x	03	x	0010	x^k	0032
0052	-	01	2^{-30}	0012	$/n-2/2^{-30}$	0011
0053	FU	34	-	0054	-	0051

Vizsgáljuk meg a programot n = 2 esetére. Ekkor a 0051-es utasitással a 0032-es rekeszben képezi x^2-et, majd $/n-2/.2^{-30}$-ből, azaz 0.2^{-30}-ból 2^{-30}-at kivon. Az eredmény -2^{-30}, vagyis negativ szám. Az FU utasitás kiugratja a gépet a ciklusból. Rajzoljuk meg ennek a programnak a menetrendjét./L. az V. táblát./

A menetrendről leolvasható az ilyen tipusu ciklusok általános szerkezete.

Az alábbiakban olyan feladatokat ismertetünk, amelyeknél szintén célszerü 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

$$x_k = f\,/x_{k-1}/$$

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

számitja 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.

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

Feltesszük, hogy a > 0, tehát a gyökvonás elvégezhető.
A számitás az

$$x_n = 0{,}5 / \frac{a}{x_{n-1}} + x_{n-1} /$$

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

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

$$x_{n+1} = 0{,}5 / \frac{a}{x_n} + x_n / = \underbrace{| 0{,}5 / \frac{a}{x_n} - x_n / |}_{D} + x_n$$

A számitást akkor fejezzük be, ha a közelités 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 utasitásokat a 0000 rekesztől kezdve helyezzük el. /0010/= a; /0014/ = x_0; /0011/ = 1/2; /0013/ = 2^{-30}

utasitás rekesze	művelet szimbóluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
0000	Á	05	x_o	0014	x_o	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:

$$\downarrow x_{k+1} = x_k - \frac{x_k^n - a}{n\, x_k^{n-1}}$$

ahol k a közelitő 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 "érzékelne" amugy is.

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

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

Megszerkesztjük a menetrendet: /L.a VII. táblát/.

Az n gyökkitevő csak egész szám lehet. Ezt $n.2^{-30}$ 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 irni a memóriába. Az olyan konstansokat, amelyek függnek ettől az n programparamétertől, a program segitségével kell előállitani.

Program:

$x \Longrightarrow$ 0050; /0010/ = a; /0011/ = $n \cdot 2^{-30}$; /0012/ = 3.2^{-30}

/0013/ = 2^{-30}; /0014/ = x_0

utasitás rekesze	művelet szimboluma	művelet jele	1.cim tartalma	1.cim	2.cim tartalma	2.cim
0100	:,	12	$n.2^{-30}$	0011	2^{-30}	0013
0101	U1	24	-	0102	$\frac{1}{n}$	0020
0102	-,	11	3.2^{-30}	0012	$n.2^{-30}$	0011
0103	U1	24	-	0104	$/n-3/.2^{-30}$	0021
0104	Á	05	x_0	0014	x_0	0050
0105	Á	05	$/n-3/.2^{-30}$	0021	$/n-3/2^{-30}$	0023
0106	Á	05	x_k	0050	x_k	0022
0107	x	03	x_k	0050	x_k^m	0022
0110	-	01	2^{-30}	0013	$/n-3/.2^{-30}$	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	z_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. Számitsuk ki a következő függvény helyettesitési értékét az $x = x_0$ helyen:

$$f\,/x/ = \frac{x_1}{2} + \frac{x_2}{4} + \ldots + \frac{x_{10}}{2^{10}} = \sum_{i=1}^{10} \frac{x_i}{2^i}$$

ahol $x_i = a\,x_{i-1} + b$; $x_1 = ax_0 + b$

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/ = 9.2^{-30}; /0016/ = 0

Munkarekeszek: 0020, 0021, 0022

$f\,/x/ \Longrightarrow$ 0050

Program:

0100	Á	05	0	0016	0	0050
0101	Á	05	9.2^{-30}	0015	9.2^{-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	-	0114	-	0112
0112	x	03	$\frac{1}{2}$	0013	$\frac{1}{2^k}$	0021
0113	U2	74	-	-	-	0104
0114	. . . . .					

2. Számitsuk ki az

$x = a^{20} + a^{19} b + a^{18} b^2 + ... + a b^{19} + b^{20}$ értéket.

A menetrendet a VIII. tábla második ábrája mutatja.

Konstansok: /0010/ = a; /0011/ = b; /0012/ = 18. 2^{-30};

/0013/ = 19.2^{-30}; /0014/ = 2^{-30}

Munkarekeszek: 0030, 0031, 0032, 0033

Program:

0100	Á	05	18.2^{-30}	0012	18.2^{-30}	0031
0101	Á	05	19.2^{-30}	0013	19.2^{-30}	0032
0102	Á	05	a	0010	a	0033
0103	x	03	a	0010	a^k	0033
0104	-	01	2^{-30}	0014	/18-k/.2^{-30}	0031
0105	FU	34	-	0106	-	0103
0106	:,	12	a	0010	b	0011
0107	U1	24	-	0110	a^{-1} b	0030
0110	Á	05	a^{20}	0033	-	0040
0111	x	03	$a^{-1}b$	0030	a^{20-i} b^i	0033
0112	↓+	20	$\sum_{k=0}^{i} a^{20-k} b^k$	0040	$\sum_{k=0}^{i+1} a^{20-k} b^k$	0040
0113	-	01	2^{-30}	0014	/19-i/ 2^{-30}	0032
0114	FU	34	-	0115	-	0111
0115		megállás				

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

$$n! = 1.2.3. \ldots /n-1/ \cdot n$$

A számitást a következőképen végezzük el:

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

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

. . . . . .

. . . . . .

$$\frac{1}{n!} = \frac{1}{2} \cdot \frac{1}{3} \cdot \ldots \cdot \frac{1}{n-1} \cdot \frac{1}{n} = \frac{1}{2} \cdot \frac{1}{2+1} \cdot \frac{1}{3+1} \cdot \ldots \cdot \frac{1}{/n-1/+1} =$$

$$= \frac{1}{2} \cdot \frac{1}{2+1} \cdot \frac{1}{2+2} \cdot \ldots \cdot \frac{1}{2+/n-2/}$$

Tehát a számitást 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: $/0010/ = \frac{1}{2}$; $/0011/ = 2^{-30}$; $/0012/ = n.2^{-30}$;

$/0013/ = 2.2^{-30}$

Munkarekeszek: 0031, 0032

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

Program:

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

0103	Á	05	2.2^{-30}	0013	2.2^{-30}	0032
0104	+	00	2^{-30}	0011	$k.2^{-30}$	0032
0105	:,	12	$k.2^{-30}$	0032	2^{-30}	0011
0106	↓x	23	$\frac{1}{/k-1/!}$	0030	$\frac{1}{k!}$	0030
0107	-,	11	$k.2^{-30}$	0032	$/n-1/2^{-30}$	0031
0110	FU	34	-	0111	-	0104
0111	. . . . .					

4. Számitsuk ki az $y = \frac{1}{a-bx}$ függvény helyettesitési értékét az $x=x_o$ helyen hatványsora segitségével.

Hatványsorba fejtve:

$$\frac{1}{a-bx} = \frac{1}{a} + \frac{bx}{a^2} + \frac{b^2x^2}{a^3} + \ldots \quad / \; |x| < \left|\frac{a}{b}\right| \; /$$

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

A menetrendet a IX. tábla második ábráján találjuk.

<u>Konstansok:</u> /0010/ = $\frac{1}{a}$; /0011/ = b; /0012/ = x_o;

/0013/ = 2^{-30}; /0014/ = o

<u>Munkarekeszek:</u> 0020, 0021

A program a 0120-as memóriarekesznél kezdődik.

$y \Rightarrow 0030$

<u>Program:</u>

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	0011	x_o	0012
0123	↓x	23	$\frac{1}{a}$	0010	$\frac{bx_o}{a}$	0021

0124 x 03 $\frac{bx_o}{a}$ 0021 $\frac{b^{k-1}x_o^{k-1}}{a^k}$ 0020

0125 + 20 $\sum_{i=1}^{k-1} \frac{b^{i-1}x_o^{i-1}}{a^i}$ 0030 $\sum_{i=1}^{k} \frac{b^{i-1}x_o^{i-1}}{a^i}$ 0030

0126 |-|, 51 2^{-30} 0013 $\frac{b^{i-1}x_o^{k-1}}{a^k}$ 0020

0127 FU 34 - 0130 - 0124

0130 . . . . . .

5. Számitsuk ki az . $y = \ln \frac{1+x}{1-x}$ függvény helyettesitési értékét az $x = x_o$ helyen.

Hatványsorba fejtve:

$$\ln \frac{1+x}{1-x} = 2 /x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots /$$

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

$$\ln \frac{1+x_o}{1-x_o} \Longrightarrow 0050$$

A számitást akkor fejezzük be, ha

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

ha tehát az utolsó tag nem ad már javitást a kiszámitott értékhez.

A menetrendet a X. táblán láthatjuk.

Konstansok: /0010/ = x_o; /0011/ = 2^{-30}; /0013/ = $\frac{1}{2}$;

/0012/ = 2.2^{-30}

Munkarekeszek: 0030, 0031, a = 0032, 0033

A program a 0100-as memóriarekesznél kezdődik.

$$\ln \frac{1+x_o}{1-x_o} \Longrightarrow 0050$$

Program:

0100	Á	05	x_o	0010	x_o	0050
0101	Á	05	x_o	0010	x_o	0030
0102	Á	05	2^{-30}	0011	2^{-30}	0032
0103	x,	13	x_o	0010	x_o	0010
0104	U1	24	-	0105	x_o^2	0031
0105	x	03	x_o^2	0031	x_o^{2k+1}	0030
0106	+	00	2.2^{-30}	0012	$/2k+1/\ 2^{-30}$	0032
0107	:,	12	$/2k+1/2^{-30}$	0032	2^{-30}	0011
0110	↓x	23	x_o^{2k+1}	0030	$\frac{x_o^{2k+1}}{2k+1}$	0033
0111	↓+	20	$\sum_{i=o}^{k-i} \frac{x_o^{2i+1}}{2i+1}$	0050	$\sum_{i=o}^{k} \frac{x_o^{2i+1}}{2i+1}$	0050
0112	/-/,	51	$\frac{x_o^{2k+1}}{2k+1}$	0033	2^{-30}	0011
0113	FU	34	-	0105	-	0114
0114	:	02	1/2	0013	$2 \sum_{i=c}^{k} \frac{x_o^{2i+1}}{2i+1}$	0050
0115	. . . . .					

4. Fejezet.

CIM- és MÜVELETI UTASITÁSOK MENETKÖZBENI MÓDOSITÁSA.

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

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

Fenti részprogramok ugyanugy végrehajtásra kerülhetnének; ha az utasitások 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 utasitásait, akár bizonyos feltételektől függően, akár egy ciklusban minden egyes ismétlődés alkalmával. A programok nagy többségében ugyanis - az előző fejezet példáitól eltérően - a ciklusokat változó utasitásokkal kell végrehajtani. Mielőtt azonban ezeknél az általános ciklikus programoknak a tárgyalására térnénk, foglalkozni fogunk az utasitásmódositások módszereivel és avval, hogyan állit-

ja össze, szerkeszti meg egy adott számitási program egyes saját utasitásait, amelyeket a későbbiek folyamán végrehajt. Mindezekre az utasitásokkal való manipulációkra az ad lehetőséget, hogy a gép tárolt programu és az utasitásokkal ugyanazok a müveletek végezhetők el, mint a számadatokkal. Azoknál a gépeknél, amelyeknél a program külsőleg van megadva, az utasitásokkal nem végezhetők müveletek és azok nem változtathatók. A módszert néhány példa kapcsán mutatjuk be.

4.1. A cimutasitások módositása.

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, célszerü azokat nem külön, direkt módon programozni, hanem ciklikus programmal keresni a megolást. 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 evvel 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 irjuk ki az utasitások cimeinek 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/ cimeit szimbolikusan szögletes zárójellel jelöltük. Ez azt jelenti, hogy ezeket a konstansokat elvben bármelyik szabad memória-rekeszben elhelyezhetjük. Konkrét program szerkesztésekor a választott rekeszek cimeit meg kell adni.

b./ Az 1-től 40-ig terjedő természetes számokat /igy 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 utasitásai változatlanok, olyan program tehát, amelynek tipusával az előző fejezetben foglalkoztunk. Ha azonban a feladatot egy kissé módositjuk, akkor már nem tudunk fix utasitásu ciklussal programozni. Legyen az uj 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 cimek 8-as számrendszerben vannak megadva!/. Gyüjtsük az összeget ismét az 500-as rekeszben. A program kezdetén, mint az előző esetben, most is ki kell üriteni 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 gyüjtőrekeszbe. Ezután szükség lenne az 1001-es utasitásnak még 39-szer történő megismétlésére ugy, hogy első cimének helyén 0101 helyett rendre 0102, 0103, ..., 0150 legyen. Ez azt jelenti: a program egy szakaszát - jelen esetben egyetlen utasitást -

kell megismételni, minden ismétlés alkalmával módositott formában. Ez a módositás ugy érhető el, hogy az 1001-es utasitás első ciméhez - minden ismétlés előtt - 1-et adunk hozzá és minden ismétléskor megvizsgáljuk, hogy ez a módositott cim elérte-e a 150-et. Utóbbi esetben a program véget ér, "kiugrik a ciklusból".

A cimmódositás céljára felhasználjuk a 2^{-18} konstanst. Az utasitások első cime ugyanis, mint ismert, a szó 7.-18. helyértékein helyezkedik el. Ezt a cimet ugy lehet eggyel növelni, ha az utasitást 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 utasitás módositja az 1001-es utasitást. Az 1003-as utasitás az 502-es rekeszben tárolt u.n. utasitáskonstanssal való összehasonlitás által megvizsgálja: tulhaladta-e már az 1001-es utasitás első cime a 150-et. Ha nem, akkor az 1004-es feltételes ugró utasitás visszaadja a vezérlést az 1001-es utasitásra módositott formában megismételteti a ciklust. Ha igen, akkor ugyanez a feltételes ugró utasitás "kiugratja" a programot a ciklusból.

A fenti példán láttuk, hogyan történik az utasitások módositása a ciklusokban. A módszer ugyanaz akkor is, ha egy ciklusban nem egy, hanem több utasitást kell megváltoztatni. A példa programja végrehajtja a kitüzött 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

utasitás megváltozott és nem lehet ujra 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 számitás 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 utasitásnál 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 utasitást kell szerkeszteni, amelynek ugrócime /az a cime, amely a következő utasitást megadja/:

1500 + 10. /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 utasitás második cimének képzéséről kell gondoskodnunk.

A program a következő:

1000	:,	12	$\langle\frac{1}{8}\rangle$	0500
1001	↓+	20	0501	1002
1002		--	--	--

Konstans: /0501/ = 74 0000 1500

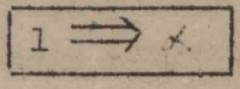
Az 501-es rekeszben lévő konstans u.n. utasitáskonstans, az 1002-es utasitás felépitésére szolgál. /Azért nem adtuk meg ennél az U2 müveleti szimbolumot, mert ez a konstans az 501-es rekeszben nem kerül utasitásként végrehajtásra, hanem mint számadat szerepel. Az 1000-es utasitás megszorozza a paramétert 8-cal /gépi "10"-zel/, illetve, minthogy 8 nem helyezhető el a gépben, elosztja $\frac{1}{8}$-dal. Az igy kapott számot az

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

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

A fenti tipusu programrészeket, amelyek egy beállitható paramétertől függően különböző elágazásokat valósitanak meg, a blokkdiagramban ugynevezett állitható kapcsolóval jelzik, a XII. tábla első ábrája szerint.

A kapcsoló beállitásának /ez példánkban a megfelelő paraméter elhelyezése az 500-as rekeszben/ a szimboluma pl.:



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

Egy kapcsolóállitást és kapcsolót tartalmazó programrészlet diagramját ábrázoltuk a XII.tábla második részében.

3. Mátrix-elem elhelyezése a meóriában.

Tegyük fel, hogy egy m x n-es mátrix van a gép memóriájában, ugy, hogy az egyes elemeket az egymásra következő sorok sorrendjében, az 1500-as reksztől 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 uj elem cime:

$$1500 + n./i-1/ + j-1.$$

Ez lesz az átviteli utasitás cime. A program:

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

/utasitáskonstans /0501/ = 15 0500 1500/

Az 1002-es és 1005-ös utasitások a részeredmények megfelelő helyérték-eltolását szolgálják. A kiszámitott cim értékének 2^{-30}-al kell szorozva lennie, mert az átvételi utasitás második ciméről van szó. Az 1007-es átviteli utasitást maga a program állitja elő. A programban három paraméter /n, i, j/ és egy konstans /2^{-15}/ szerepel.

4.2. Müveleti utasitások módositása.

Az utasitásokkal való manipuláció nem korlátozódik az utasitás cimeinek módositására, vagy a cimek összeállitására. Minthogy a müveleti jel is számjegykombináció, ezen ugyanugy végezhetők aritmetikai müveletek, mint a cimeken. Ilyen esetekben persze figyelembe kell venni a szereplő utasitások konkrét müveleti jeleit /kódjeleit/. Pl.egy számitás 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	↓+	30	1002	1002
1002	+	00	0502	0501
. . . . .				

Az 500-as rekeszben tárolt paramétert 24 helyértékkel el kell tolni balra /ezt valósitja meg az 1000-es utasitás a 2^{-24}-el való osztás által/, hogy a müveleti jel megfelelő helyértékére, a szó 6. helyértékére kerüljön. Ha az igy kapott értéket hozzáadjuk az 1002-es utasitáshoz /ezt az 1001-es utasitás végzi/, ugy az 1002-es utasitás a paramétertől függően változatlan marad, vagy kivonó utasitássá lesz.

Megjegyzések:

a./ Ha az 500-as rekeszben tárolt paraméter-érték 2.2^{-30}, 3.2^{-30} vagy 6.2^{-30}, ugy 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 számitás kezdete előtt /ha pl. ugyanazt a programot többféle változatban kivánjuk felhasználni/, de lehetséges, hogy magának az adott számitási programnak előző szakasza számitja ki és helyezi el a paramétert.

A blokkdiagramban a fenti tipusu program-részletet az előzőkben megismert állitható kapcsolóval jelölik, annyi kapcsolóállással, ahány változatot a program ténylegesen megvalósit.

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

1. Az 500-as rekeszben tárolt paraméter a következő 8 értéket veheti fel: 0, 2^{-30}; 2.2^{-30},... 7.2^{-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 kiszámitott eredménytől függően valamely rekeszt meg kell jelölni.

A paramétertől függő utasitás a következő alaku:

+ 00 $\langle 2^{-18} \rangle$ 051..

A részprogram:

1000	+	00	0500	1001
1001	+	00	$\langle 2^{-18} \rangle$	0510

2. Ha az 500-as rekesz tartalma : $x < 2^{-3}$, adjunk 2^{-30}-at az 510-es rekesz tartalmához. Ha $2^{-3} \leq x < 2.2^{-3}$, adjunk 2^{-30}-at az 511-es rekesz tartalmához sit., ha $7.2^{-3} \leq x < 1$, adjunk 2^{-30}-at az 517-es rekesz tartalmához.

A program a következő:

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

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

Az 1004-es utasitás végzi el az összehasonlitást $\frac{x}{2}$ és 2^{-4} többszörösei között. A program tulajdonképen ismétlődő ciklus, melynek utasitásai fixek és amely minden ismétlődéskor módositja az 1006-os utasitást /ez kivül van a ciklusson/, ugy, hogy végül a kivánt rekesz tartalmához adódjék hozzá 2^{-30}.

Ha megengedhető az 500-as rekszben tárolt x szám elvesztése /elrontása/, akkor a program négy utasitásból állitható össze:

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

Az 1001-es utasitás $\frac{1}{8}$ -os lépésekkel csökkenti x-et, amig az negativvá nem válik. Minden ismétléskor hozzáadódik 1 az 1003-as utasitás 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 müveletet a megfelelő mennyiségek abszolut értékén.

Itt nyilvánvalóan a müveleti jel módositásáról lesz szó.

A program:

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

Ha az 500-as rekesz tartalma 2^{-30}, ugy az 1002-es utasitás müveleti jeléhez hozzáadódik $2^{-1} = 4.2^{-3}$. Az uj müveleti jel 50 lesz, ami a |+|, utasitásnak felel meg. Ha /0500/ = 0, az 1002-es utasitás változatlan marad.

4. Oldjuk meg az $ax^2 + bx + c = 0$ másodfoku 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 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	⟨a⟩	⟨c⟩	ac
1003	↓:	22	$\langle \frac{1}{4} \rangle$	0501	4ac
1004	x,	13	⟨b⟩	⟨b⟩	b^2
1005	↓-	21	0501	0502	b^2-4ac
1006	FU	34	0505	1007	
1007	Á	05	$\langle 1-2^{-30} \rangle$	0503	
1010	:,	12	0503	0502	
1011	↓-,	31	0503	-	Négyzetgyökvonás
1012	↓x	23	$\frac{1}{2}$	0504	
1013	↓+	20	0503	0503	
1014	↓-,	51	0504	$\langle 2^{-28} \rangle$	
1015	FU	34	1010	1016	
1016	-,	11	⟨b⟩	⟨o⟩	
1017	↓+;	30	0503	-	
1020	↓x,	33	$\langle \frac{1}{2} \rangle$	-	
1021	↓:	22	⟨a⟩	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 munkarekesze, 505 tartalma pedig egy megállitó utasitás. Erre ugratja a gépet az 1006-os feltételes ugró utasitás, ha a diszkrimináns negativ. Az 1017-es utasitás az 500-as rekeszben lévő paramétertől függ: ha a paraméter értéke 0, ez az utasitás nem változi, ha a paraméter értéke 2^{-30}, akkor kivonás lesz belőle /müveleti jele 31 lesz/. Igy a paraméternek megfelelő gyök kerül a 600-as rekeszbe.

Bemutatunk most egy olyan példát, amelynél a program által összeállitandó, majd végrehajtandó müvelet 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: $a = 0, 1, \ldots, 4$. Az 501-es rekesz tartalma: $b \cdot 2^{-30}$, ahol a b paraméter 8 különböző értéket vehet fel: $b = 0, 1, \ldots, 7$. Adjunk hozzá a 600 + 2a cimü rekesz tartalmához $4b.2^{-30}$-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	$\langle\frac{1}{2}\rangle$	0500	$2a.2^{-30}$
1001	↓+	20	0400	1004	
1002	:,	12	$\langle\frac{1}{4}\rangle$	0501	$4b.2^{-30}$
1003	U1	24	1004	0401	
1004		(. . . . .)			
. . . . .					

Az 1004-es utasitást, amely a kitüzött feladatot végrehajtja, a program állitja össze a 400-as rekeszben tárolt utasitáskonstansból és az a paraméterből. Az 1004-es utasitás elhelyezése /"beültetése"/ az 1001-es utasitás által történik, de első cimének /a 401-es rekesznek/ a tartalmát, 4b-t, csak az 1002-es és 1003-as utasitások képezik és helyezik el. Amikorra az 1004-es utasitás végrehajtására kerül, már minden szükséges előkészület megtörtént. A 401-es rekesz a program munkarekesze, eredeti tartalma közömbös /csakugy mint az1004-es rekesz eredeti tartalma/.

6. Módositsuk az előző feladatunkat a következőképen: 600 + 2a tartalmához adjuk hozzá 700 + 4b tartalmát! Itt tehát a b paraméter is egy cimet határoz meg /ugy, mint az a paraméter/, nem pedig közvetlenül egy számadatot. A módositott feladat programja:

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

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

Előző példánknál a 4b mennyiséget egy bizonyos cimre kellett elhelyezni /a 401-es munkarekeszbe/, hogy mint összeadandó szerepelhessen/ az utasitások nem magukkal a számadatokkal, hanem azok cimével dolgoznak!/ Ennél a példánál erre nem volt szükség, mert mindkét paraméter cimet módositott.

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

A program:

```
                                    /0400/=+00  0700  0600
                                    /0401/=+00  7777  0000
                                    /0402/=+00  0000  7777
1000  ↑,    16   0402   0500
1001  ↓:,   32   (1/2)
1002  ↓+    20   0400   1006
1003  ↑,    16   0401   0500
1004  ↓:,   32   (1/4)
1005  ↓+    20   1006   1006
1006        (--   ----   ----)
```

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

8. Az előző módszerrel oldható meg a következő feladat: Az 1500-tól 1577-ig terjedő rekeszek egy bizonyos programot tartalmaznak. Az 1520-as utasitás müveleti jelét, az 1540-es utasitás első cimét és az 1560-as utasitás második cimét bizonyos paraméterértékek hozzáadásával, a megfelelő utasitások végrehajtása előtt, módositani kell. Célszerü a három módositó paramétert egyetlen szóban, egy utasitáskonstansban egyesiteni, pl. az 500-as rekeszben, ugy hogy az első a müveleti jel helyén, a második az első cim helyén, a harmadik a második cim helyén legyen.

A program:

					/0501/=+77	0000	0000
1000	∧,	16	0501	0500	/0502/=+00	7777	0000
1001	↓+	20	1520	1520	/0503/=+00	0000	7777
1002	∧,	16	0502	0500			
1003	↓+	20	1540	1540			
1004	∧,	16	0503	0500			
1005	↓+	20	1560	1560			
1006	U2	74		1500			

Az 1006-os utasitás, az előkészités után, átadja a vezérlést az 1500-as utasitással kezdődő programnak.

9. 64 szám van elhelyezve a 100-tól 177-ig terjedő cimü rekeszekben. Egyik szám cime az 500-as rekesz második cimrészének a helyén van. Helyezzük el ezt a számot a 600-as rekeszbe, a rákövetkező rekeszben lévőt a 601-be, a rákövetkező rekeszben lévőt a 602-be. Vizsgáljuk meg, hogy az 500-as rekeszben elhelyezett cim nem nagyobb-e 175-nél és ha igen, állitsuk meg a gépet. Ekkor ugyanis nem hajtható végre a feladat.

A program:

1000	|-|,	51	0500	$\langle 175.2^{-30} \rangle$	/0501/=+05	0000	0600
1001	FU	34	0502	1002	/0502/=+04	0000	0000
1002	:,	12	$\langle 2^{-12} \rangle$	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 utasitáskonstanst, amelynek segitségével a program összeállitja az 1006-os, 1007-es és 1010-es utasitásokat. 502 tartalma egy megállitó utasitás, 503 tartalma pedig egy konstans, amelylyel az első változó utasitásból előállitható a másik kettő.

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

Ujból leirjuk a fenti programot

				/0502/=+00 0150 0500
1000	Á	05	⟨0⟩	0500
1001	+	00	0101	0500
1002	+	00	$\langle 2^{-18} \rangle$	1001
1003	-,	11	1001	0502
1004	FU	34	1005	1001
1005	.	.	.	.

Feladatunkat most ugy módositjuk, hogy a 101-es rekesz tartalmától kezdve egy $p \geq 1$ paraméterrel megadott számu rekesz tartalmát adjuk össze. Mint könnyen belátható: fenti programban az 502-es rekeszben elhelyezett utasitáskonstanst kell a paramétertől függően kialakitani, mert ettől a konstanstól függ, hány számot adunk össze /persze ebben az esetben ebből a mennyiségből nem konstanst, hanem változó lesz./ Legyen a paraméter a 600-as rekeszben, az első cim helyén elhelyezve.

Az új program:

/0502/= +00 0100 0500

1000	+	0C	0600	0502
1001	Á	05	⟨0⟩	0500
1002	+	00	0101	0500
1003	+	00	$\langle 2^{-18} \rangle$	1001
1004	-,	11	1001	0502
1005	FU	34	1006	1002
1006	. . . . .			

5. Fejezet.

CIKLIKUS PROGRAMOK VÁLTOZÓ UTASITÁSOKKAL.

5.1. Utasitások visszaállitása beültetéssel.

Miután megismerkedtünk az utasitások módositásának és összeállitásának módszereivel és legfontosabb alkalmazásaival, rátérhetünk az általános /vagy változó utasitásu/ ciklusok tanulmányozására.

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

Vegyük elő ismét a 4. fejezetben ismertetett első példánkat a 101-től 150-ig terjedő cimü 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	$\langle 2^{-18} \rangle$	1001
1003	←,	11	1001	0502
1004	FU	34	1005	1001
1005	. . . . .			

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

A módositott program :

				/0502/	± + 00 0150 0500	
				/0503/	= + 00 0101 0500	
1000	Á	05	⟨0⟩	0500	előkészités	
1001	Á	05	0503	1002		
1002	/	--	----	-----	/ számolás	
1003	+	00	$\langle 2^{-18} \rangle$	1002	módositás	ciklus
1004	↛	11	1002	0502	vizsgálat	
1005	FU	34	1006	1002		
1006	-	-	-	-		

Ez a program egy utasitással hosszabb az előzőnél és egy konstanssal /utasitás-konstánssal, "pszeudo-utasitással"/ többet tartalmaz. A változó utasitás kiinduló alakját az 503-as rekesz tárolja, változatlan állandóként /pszeudo-utasitásként/ és ezt az 1001-es utasitás helyezi el az 1002-es rekeszbe, a következő végrehajtandó utasitás helyére. Az utasitásmódositás az 1002-es helyen tárolt utasitást érinti, minden ismétlődés alkalmával, az ismétlődések az 1002-es utasitással kezdődnek. Ha ezt a részprogramot ujra felhasználjuk, elejétől, az 1000-es utasitástól kezdve, ugy ismét megtörténik az 1002-os változó utasitásnak 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 utasitás a tulajdonképeni ciklusen kivül áll és azt a célt szolgálja, hogy bizonyos kezdeti értékeket helyezzen el. Ez az u.n. előkészitő rész. A többi utasitás mind a ciklushoz tartozik, azaz minden ismétlődés alkalmával végrehajtásra kerül. Az 1002-es változó utasitás a program tulajdonképeni számoló része. A következő utasitás /1003-as/ alkotja a módositási részt, majd az utolsó két utasitás 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 utasitásu ciklusos programban megtalálható. Természetesen mindegyik rész több utasitásból is állhat, mint jelenlegi egyszerü példánkban. Lehetséges a cikluson belüli egyes részeket felcserélni. Pl. fenti példánk esetében:

/0502/ = + 00 0147 0500
/0503/ = + 00 0100 0500

1000	Á	05	⟨0⟩	0500	előkészités	
1001	Á	05	0503	1005		
1002	-,	11	1005	0502	vizsgálat	ciklus
1003	FU	34	1007	1004		
1004	+	00	$\langle 2^{-18} \rangle$	1005	módositás	
1005		/--	----	----/	számolás	
1006	U2	74	----	1002		
1007	. . . . .					

Mint látjuk, ebben az esetben eggyel több utasitásra van szükségünk és az utasitáskonstansokat 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 utasitáskonstans helyesen van-e megállapitva. 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 célszerü mindig megvizsgálni, hogy mi történnék adott esetben, ha a ciklus egyszeri ismétlődést programoznók be.

5.2. A program visszaállitása utókorrekcióval.

Van olyan programozási módszer is, amelynél a változó utasitások nem a program elején /az előkészitő részben/ lesznek kitöltve, hanem: eredeti alakjukban már a helyükön vannak, a ciklusok folyamán változnak, majd a ciklust követő program-rész - az u.n. helyreállitó rész - eredeti alakjukban állitja vissza őket. - Ebben az esetben is lehetséges a program ismételt felhasználása.

Fenti példánk tehát igy is átirható:

/0502/ = + 00 0150 0500
/0503/ = + 00 0050 0000

1000	Á	05	⟨0⟩	0500	előkészités	
1001	+	00	0101	0500	számolás	ciklus
1002	+	00	$\langle 2^{-18} \rangle$	1001	módositás	
1003	-,	11	1001	0502	vizsgálat	
1004	FU	34	1005	1001		
1005	-	01	0503	1001	helyreállitás	

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

Mint látjuk, a ciklusos program előkészitő része csak egyszer kerül végrehajtásra, mig a tulajdonképeni ciklus igen sokszor ismétlődhetik. Ezért esetenként célszerü az előkészitő részt tetemesen megnövelni, ha ezáltal a ciklus maga /esetleg csak kisebb mértékben/

lerövöditheto. Ez olyankor lehetséges, amikor egy bizonyos számitás több alternativ uton /több formula szerint/ végezhető el. Minden esetben bizonyos kompromisszumra kell törekedni a program hossza /utasitások száma/ és végrehajtásának időtartama között. A két követelmény ellentmondó. Maga a ciklusos programozás lényegesen megnöveli a feladat végrehajtásának idejét /esetleg tizszeresére is/ a program rövidsége kedvéért. Ez példánkból is meglátszik, ahol a tulajdonképeni számolást csak egyetlen utasitás végzi, a többi a ciklus érdekében van ott. Direkt programozás esetén 40 összeadó utasitásra lenne szükség, de a végrehajtás ideje sokkal kisebb lenne. A program hossza és végrehajtásának ideje közötti kompromisszumot aszerint kell megállapitani, hogy adott esetben melyik szempont a kritikusabb.

Feladatok:

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

   /n/. /n+1/ - /n+2/ ⟹ n /n = 1201, 1202,..., 1300/

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

+,	10	0101	0102
↓+	20	0500	0500

Ezáltal a ciklus maga egy utasitással hosszabb, de csak fele annyi /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 következőképen:

/0501/ = x_1	/0601/ = y_1
/0502/ = x_2	/0602/ = y_2
⋮	⋮
/0600/ = x_{100}	/0700/ = y_{100}

x és y indexeit, akárcsak a rekeszek címeit, 8-as számrendszerben ábrázoljuk. A táblázat tehát a függvény k. a független változó 64 különböző helyéhez tartozó értékeit tartalmazza. Feltételezzük, hogy az "x-ek" növekvő sorrendben elhelyezett nem negativ-számok, az y-okról semmilyen megszoritást 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 állitania 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, $x_n \leqq x \leqq x_{n+1}$, y_{n+1} és y_n az ezeknek megfelelő függvény értékek.

A számitás 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 0000
			/0302/ = + 11	0300 0501
			/0303/ = + 00	0000 7777
			/0304/ = + 05	0000 0351
			/0305/ = + 00	0001 0001
			/0306/ = + 00	0100 0002
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	1016
1013	↓-	21	0305	1017
1014	↓+	20	0306	1020
1015	↓+	20	0305	1021
1016		/--	----	----/ $x_{n+1} \Longrightarrow$ 351
1017		/--	----	----/ $x_n \Longrightarrow$ 350
1020		/--	----	----/ $y_n \Longrightarrow$ 352
1021		/--	----	----/ $y_{n+1} \Longrightarrow$ 353

1022 - 01 0350 0351 ← $/x_{n+1}-x_n/$

1023 - 01 0352 0353 ← $/y_{n+1}-y_n/$

1024 -, 11 0350 0300 $/x-x_n/$

1025 ↓x, 33 0353

1026 ↓:, 32 0351

1027 ↓+ 20 0352 0400

. . . . . .

A követhetőség kedvéért egyes utasitások mellé odairtuk, mi történik az utasitások révén. Ha az illető utasitás eredménye beiródik a 2. cimre, ezt egy nyillal jelöltük /az 1022-es és 1023-as utasitásnál/. Az első négy utasitás megvizsgálja, hogy x a kérdéses tartományban van-e. Ha nem, ugy a gép megáll a 301-es megállitó utasitás miatt. Az 1004-től 1007-ig terjedő utasitások x_{n+1} megkeresésére szolgálnak /1004-es: előkészitő rész, 1005-1007-ig ciklus/. Az 1010-től 1012-ig terjedő utasitások x_{n+1} cimének leválasztásával képeznek egy átviteli utasitást, az 1013-től 1015-ig terjedő utasitások további három átviteli utasitást képeznek. Ez a négy átviteli utasitás helyezi el az x_n, y_n, x_{n+1}, y_{n+1} értékeket. Végül következik az interpolációs formula kiszámitása.

További példák:

1. A 100-tól 150-ig terjedő cimü rekeszek 41 számot tartalmaznak. Képezzük a szomszédos rekeszekben lévő 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	előkészités	
1001	Á	05	0501	1003		
1002		/--	----	----/	számolás	
1003		/--	----	----/		
1004	+	00	0503	1002	módositás	ciklus
1005	+	00	(2-30)	1003		
1006	-,	11	1003	0502	vizsgálat	
1007	FU	34	1010	1002		
1010	. . . . .					

2. Legyen két 16-dimenziós vektor a memóriában a következőképen: 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 $\underline{x} = /x_1, x_2, \ldots, x_n/$ és $\underline{y} = /y_1, y_2, ---, y_n/$ vektor skaláris szorzata az $x_1y_1 + --- + x_ny_n$ mennyiség.

A program:

/0501/ = + 13 0101 0121
/0502/ = + 00 0001 0001
/0503/ = + 00 0120 0140

1000	Á	05	(0)	0500	előkészités	
1001	Á	05	0501	1002		
1002		/--	----	----/	számolás	
1003	↓+	20	0500	0500		
1004	+	00	0502	1002	módositás	ciklus
1005	-,	11	1002	0503	vizsgálat	
1006	FU	34	1007	1002		
1007	--	--	----	----		

Az 500-as rekeszben nemcsak a végeredményt, de a részletösszeget is elhelyezzük /az 500-as: u.n. gyüjtőrekesz, akkumulátor szerepét játsza./

5.4. Többszörös ciklusok.

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

A müveleti idők becslése céljából megemlitjük, 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 következőképen 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 cimü rekeszekben. Képezzük az első vektor skaláris szorzatát a kilencedikkel, tizedikkel, ..., tizenhatodikkal, majd a második vektor skaláris szorzatát a kilencedikkel, tizedikkel, ..., tizenhatodikkal, sit. 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	$\langle 7.2^{-30} \rangle$	0604
1003	Á	05	$\langle 7.2^{-30} \rangle$	0600
1004	+	00	$\langle 16.2^{-18} \rangle$	0601
1005	↓+	20	0501	1010
1006	Á	05	$\langle 15.2^{-30} \rangle$	0602
1007	Á	05	⟨0⟩	0603
1010		/--	----	----/
1011	↓+	20	0603	0603
1012	+	00	0502	1010
1013	-	01	$\langle 2^{-30} \rangle$	0602
1014	FU	34	1015	1010
1015		/--	----	----/
1016	+	00	$\langle 2^{-30} \rangle$	1015
1017	-	01	$\langle 16.2^{-18} \rangle$	1010
1020	-	01	$\langle 2^{-30} \rangle$	0600
1021	FU	34	1022	1006
1022	-	01	$\langle 2^{-30} \rangle$	0604
1023	FU	34	1024	1003
1024	. . . . .			

C B A

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

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

2. Az alábbi számsémát helyeztük el a gép memóriájában:

$$\begin{array}{llll} a_{11} & a_{12} & \cdots\cdots & a_{1n} \\ a_{21} & a_{22} & \cdots\cdots & a_{2n} \\ a_{31} & a_{32} & \cdots\cdots & a_{3n} \end{array}$$

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.

$n.2^{-30}$ = /0500/

Feladatunk: $\frac{1}{4}\ /a_{1,i}+a_{2,i-1}+a_{2,i+1}+a_{3,i}/ \Longrightarrow a_{2,i}$

/i =,2,... n-1/, azaz: helyettesitsük a számséma belső elemeit a körülöttük lévő négy szám számtani közepével.

A program:

		/0501/ = + 23	$\langle\frac{1}{4}\rangle$	0200
		/0502/ = + 10	0100	0600
		/0503/ = + 00	0101	0001
		/0504/ = + 10	0077	0000
		/0505/ = + 13	0100	0001

1000	Á	05	0200	0600	
1001	Á	05	0501	1011	
1002	Á	05	0502	1005	
1003	-,	11	$\langle 2.2^{-30}\rangle$	0500	
1004	U1	24	1012	0602	←/n-2/. 2^{-30}
1005		/—	----	----/	
1006		/—	----	----/	
1007		/—	----	----/	
1010		/—	----	----/	
1011		/—	----	----/	
1012	+	20	$\langle 2^{-18}\rangle$	1005	
1013	↓+	20	0503	1010	

1014	↓+	20	0504	1006
1015	↓-	21	0505	1007
1016	+	00	$\langle 2^{-30} \rangle$	1011
1017	-	01	$\langle 2^{-30} \rangle$	0602
1020	FU	34	1021	1005
1021	. . . . .			

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

6. Fejezet.

A SZÁMOK FIXPONTOS ÁBRÁZOLÁSÁBÓL ADÓDÓ NUMERIKUS PROBLÉMÁK.

6.1. A fixpontos ábrázolásról.

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

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

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

A bináris pontnak ez a rögzitettsége 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 felirásán alapszik.

Egy ~~x szám~~ lebegőpontos alakján kettes számrendszer esetén az

$$x = d, 2^p \text{ alakot értjük.}$$

ahol a $d = \pm /a_1 2^{-1} + a_2 2^{-2} + \ldots + a_n 2^{-n}/$ $\quad a_i = 0$, vagy

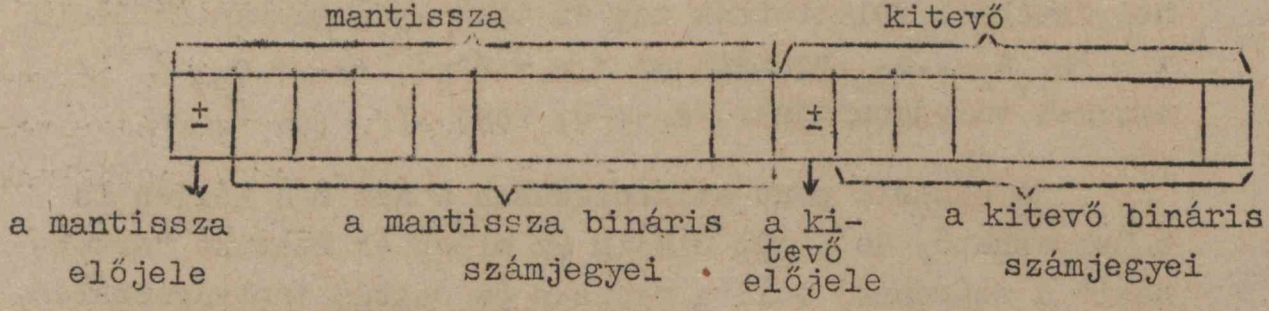
számot az illető szám mantisszájának; a

$$p = \pm /b_1 2^m + \ldots + b_{m+1} 2^0/ \qquad b_i = 0 \text{ vagy}$$

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

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

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



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

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

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

A fixpontból adódó nehézséget lényegében kétféleképpen lehet megkerülni:

a./ lebegőpontos programozás módszerével
b./ transzformációs módszerekkel.

A lebegőpontos programozás lényegében a lebegőpontos gépek müködésének egy program segitségével való utánozását jelenti. Ez esetben a számokat az előzőkben felirt lebegőpontos alakjukban ábrázoljuk, mégpedig vagy ugy, hogy egyetlen szóban ábrázoljuk a mantisszát és kitevőt is, vagy pedig külön-külön szóban ábrázoljuk a mentisszát és a kitevőt. A müveleteket a lebegőpontos alakban felirt 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 mamóriarekesz-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/ ugy alakitjuk át, hogy az abban szereplő adatok - eredmények, részeredmények - abszolut 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 zsugoritjuk.

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,±1,± 2,.../ 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 "zsugoritja" felesleges mértékben az adott számot. /A zsugoritás általában jegyveszteséggel jár./

A szám $\bar{x}$ gépi alakjának előállitása viszont kényelmesebb lo hatványok segitségével. Általában ugy járunk el, hogy a gépben tárolt kiinduló adatokat és a kinyomtatandó adatokat lc hatványaival normáljuk s a gépen belül végzett müveletek-nél fellépő tulcsordulást M-tipusu skálafaktorral kerüljük el. Többnyire célszerü $M = 2^p$-t választani.

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

1. Megállapitjuk a képletben szereplő adatok alsó és felső korlátját s megválasztjuk a megfelelő skálafaktorokat.

2. Felirjuk az adott képletet a számok gépi alakja segitségével s elvégezzük a lehetséges egyszerüsitéseket.

3. Nyilvánvaló, hogy ha a formulában szereplő kiidnuló adattok nem is csordulnak tul, lehetséges, hogy az ezekkel végzett müveletek eredményei tulcsordulnak. Meg kell tehát vizsgálni az eredményként létrejövő adatoknak a müveletben szereplő adatoktól függő alsó és felső korlátját, s ujabb skálafaktorok bevezetésével el kell érni, hogy az adott müvelet 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 módositják; fel kell jegyezni, hogy az eredményhez milyen skálafaktor tartozik. Lássunk egy példát:

Legyen a programozandó képlet a következő:

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

$$\begin{aligned} 1 &< a < 8 \\ 125 &< b < 705 \\ 5 &< c < 10 \\ 1 &< d < 5 \\ 50 &< x < 100 \\ 0{,}01 &< y < 0{,}9 \end{aligned}$$

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

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

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

b./ Felirjuk a képletet a számok normált alakjai segitségével

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

Elvégezzük a lehetséges egyszerüsitéseket.

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

c./ Vizsgáljuk meg a részeredmények tulcsordulási lehetőségeit: számitsuk ki felső korlátjaikat, s válasszuk meg az ujabb skálafaktorokat.

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

$$\max / \frac{10\ \bar{b}}{\bar{d}} + 12{,}1\ \bar{y} / = \frac{10 \max \bar{b}}{\min \bar{d}} + 12{,}1 \max \bar{y} <$$

$$< \frac{10.0{,}705}{0{,}1} + 12{,}1.0{,}9 < 100$$

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

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

$$\min \left(\frac{10\ \bar{b}}{\bar{d}} + 12{,}1\ \bar{y} \right) = \frac{10 \min \bar{b}}{\max \bar{d}} + 12{,}1 \min \bar{y} =$$

$$= \frac{10.0{,}125}{0{,}5} + 12{,}1.0{,}01 > 2.$$

Igy tehát:

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

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

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

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

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

Igy

$$\frac{F}{10} = \frac{\frac{\sqrt{10}}{10.10}\sqrt{\bar{a}\,\bar{x}} + \frac{\bar{c}}{10.100}}{\frac{\bar{b}}{\frac{10}{\bar{d}}} + 0{,}121\,\bar{y}} - \frac{\frac{1}{10}\,\bar{d}}{\bar{a}} =$$

$$= \frac{\frac{\sqrt{10}}{1000}\sqrt{\bar{a}\,\bar{x}} + \frac{\bar{c}}{1000}}{\frac{\bar{b}}{\frac{10}{\bar{d}}} + 0{,}121\,\bar{y}} - \frac{\frac{1}{10}\,\bar{d}}{\bar{a}}$$

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

Irjuk fel a feladat programját:
Legyen a rekeszelosztás a következő:

$$/0100/ = \frac{\sqrt{10}}{1000} \qquad /0106/ = \bar{c}$$

$$/0101/ = \frac{1}{1000} \qquad /0107/ = \bar{d}$$

/0102/ = $\frac{1}{10}$ /0110/ = $\bar{x}$

/0103/ = 0,121 /0111/ = $\bar{y}$

/0104/ = $\bar{a}$ /0120/ = Y_o /A gyökvonás kezdő értéke./

/0105/ = $\bar{b}$

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

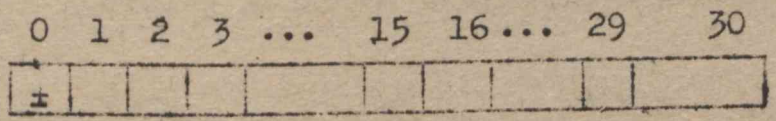
0150	x,	13	0102	0107	$\frac{\frac{1}{10}\bar{d}}{\bar{a}}$
0151	↓:	22	0104	0112 →	
0152	x,	13	0102	0105	
0153	↓:	22	0107	0113	
0154	x,	13	0103	0111	
0155	↓+	20	0113	0113 →	$\frac{1}{10}\frac{\bar{b}}{\bar{d}} + 0{,}121\,\bar{y}$
0156	x,	13	0101	0106	
0157	U1	24	0160	0114 →	$\frac{\bar{c}}{10^3}$
0160	x,	13	0104	0110	
0161	U1	24	0162	0115	
0162	A	05	⟨Y_o⟩	0116	Gyökvonás az
0163	:,	12	0116	0113	$Y_n = \frac{1}{2}\left(\frac{\bar{a}\bar{x}}{Y_{n-1}} - Y_{n-1}\right) + Y_{n-1}$
0164	↓−,	31	0116		
0165	↓x	23	⟨$\frac{1}{2}$⟩	0117	képlet alapján. /A gyök ≈ Y_n /
0166	↓+	20	0116	0116	
0167	↓−1,	51	0117	⟨ε⟩	
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 számitás folyamatát. Külön figyelemmel kell azt is kisérni, hogy a jegyveszteség minimális legyen.

6.3. A bináris pont középre való helyezésének módszere.

Mint mondottuk, 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 $x = \pm a_{14}\, a_{13} \ldots a_0 \ldots a_{-15}$ számhoz az

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

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 egyszerüen 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 abszolut 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.

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

$2 < |c + dx| < 2^7$; $|a + bx| < 2^{15}$; $|bx| < 2^{15}$; $|dx| < 2^{15}$; $|x^2| < 2^1$

Helyezzük el a bináris pontot középre, s legyen a rekeszelosztás /a cimeket számok helyett betükkel jelölve/ a következő:

$/p/ = x.2^{-15}$ $\qquad$ $/p + 3/ = c.2^{-15}$

$/p + 1/ = a.2^{-15}$ $\qquad$ $/p + 4/ = d.2^{-15}$

$/p + 2/ = b.2^{-15}$

Munkarekeszek: m, m+1

Program:

t	x,	13	p+2	p.
t+1	↓:,	32	$\langle 2^{-15} \rangle$	
t+2	↓+	20	p+1	$m \leftarrow /a+bx/2^{-15}$
t+3	x,	13	p+4	p
t+4	↓:,	32	$\langle 2^{-15} \rangle$	
t+5	↓+	20	p+3	$m+1 \leftarrow /c+dx/2^{-15}$
t+6	:	02	$\langle 2^{-8} \rangle$	m+1
t+7	x,	13	$\langle 2^{-7} \rangle$	m
t+10	↓:	22	m+1	$m+1 \leftarrow \frac{a+bx}{c+dx} \quad 2^{-15}$
t+11	x,	13	p	p
t+12	↓:,	32	$\langle 2^{-14} \rangle$	
t+13	+	00	m+1	$m+1 \leftarrow y2^{-15}$.

A t, t+3, és a t+11 utasitások végrehajtása során a bináris-pont a szó végére került, a t+2, t+4, t+12 utasitásokkal tehát "visszatoljuk" középre. A t+6, t+7, utasitások segitségével elértük, hogy az $\frac{a+bx}{c+dx}$ hányadosban is középre kerül a bináris pont. Az eredmény is $y.2^{-15}$ 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-edrendü négyzetes mátrixot: $A = [a_{ij}]$ $B = [b_{ij}]$ /i,j = 1,2,...,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 abszolutértékben kisebb; ugyanis

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

$$= \sum_{v=1}^{n} \frac{1}{n} = n \frac{1}{n} = 1$$

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-adik bit után való elhelyezésével/ elkerülhető a tulcsordulás.

Azokat az n egész számokat, amelyeket számlálásra, cimmódositásra használunk, vagy amelyek valamilyen egészértékü függvény /pl. számelméleti függvények, stb./ adatai, célszerü n. 2^{-30} alakban ábrázolni; ez azt jelenti, hogy a bináris pontot a szó végére képzeljük el.

6.4. Egyéb módszerek.

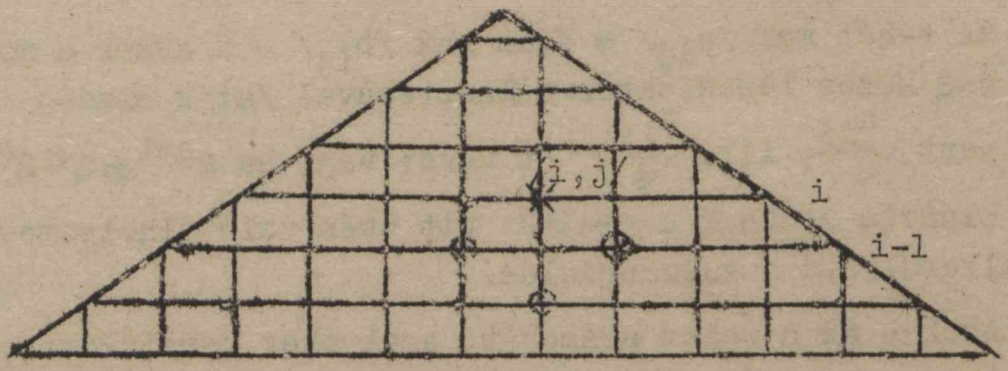
6.4.1. Reciprok érték felhasználása.

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

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

6.4.2. Automatikus normálás módszere.

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



$a_{ij} = a_{i-1,\,j-1} + a_{i-1,\,j+1} - a_{i-2,j}$ képlet adja. /*/

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 $|a_{ij}| < |a_{i-1,\,j-1}| + |a_{i-1\,j+1}| +$

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

tehát a következőképpen járhatunk el. Mielőtt egy uj 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 számitást. Ha nem, akkor előbb a szóbanforgó két sor minden elemét 3-mal osztjuk, de egyuttal megjegyezzük /pl 2^{-30} kinyomtatásával/, hogy az ezután következő a_{ij} értékek helyett azok egyharmadát számoltuk ki. Ha valamelyik következő sor kiszámitásához ujra normálni kell, akkor ujabb 2^{-30} kinyomtatásával jelezzük, hogy az ezután következő értékek kilencszeresét kell venni, hogy a tényleges értéket megkapjuk,...stb.

Másik példa:

Készitsük el az e^{-x} függvény /x o/ kiszámitásának programját az

$$e^{-x} = \frac{1}{/a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 /^4} \qquad /**/$$

közelitőképlet alapján, ahol

$a_0 = 1$

$a_1 = 0{,}2500\ 1093$

$a_2 = 0{,}0311\ 9805$

$a_3 = 0{,}00267325$

$a_4 = 0{,}00012799$

$a_5 = 0{,}00001487$

/ A /**/ közelitőképlet hibája = 0,000002./

A számlálót és a nevezőt is 2-vel osztva:

$$e^{-x} = \left(\frac{\frac{1}{2}}{\frac{a_0}{2} + \frac{a_1}{2}x + \ldots + \frac{a_5}{2}x^5} \right)^4 \qquad /\text{xxx}/$$

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

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

Tegyük fel, hogy

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

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

s igy a nevező sem csordul tul.

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

Készitsünk el egy olyan programot, amely a $0 < x < 16$-ban lévő x-értékekre müködik, mégpedig ugy, 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.i.t. értékkel számol.

Rendezzük át a /xxx/ képlet nevezőjét a Horner séma szerint. Igy:

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

Legyen a rekeszelosztás a következő: /a cimeket betüvel jelölve./

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

Konstansok:

/k/	=	↓+ 2o	k + 3 m + 2
/k+1/	=	↓+ 2o	k + 1o 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/	=	$\frac{a_o}{2}$	

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

A program:

t	Á	05	k	t+12
t+1	Á	05	$\langle 4.2^{-30} \rangle$	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+1o	t+2
t+7	+	00	$\langle 2^{-30} \rangle$	m+1
t+1o	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	↓1-1,	71	k+1	
t+15	FU	34	t+11	t+16

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

A felsorolt módszereken kivül egyes feladatoknál bizonyos speciális módszerek is használhatók. Periódikus függvényeknél pl. eltolási transzformáció alkalmazható. Igy a sin x ill. cos x függvény esetén a sin x = sin /x - 2k π /, cos x = cos /x - 2k π / egyenlőség alapján az u = x - 2k π transzformációval elérhető, hogy

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

Minthogy sin /-u/ = - sin u és cos /-u/ = cos. u. azért elég u pozitiv értékeire szoritkozni.

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

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

Igy a $\sin v = 2 \sin \frac{v}{2} \cos \frac{v}{2}$; $\cos v = \cos^2 \frac{v}{2} - \sin^2 \frac{v}{2}$

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 középre helyezzük.

A transzformációs módszerek segitségével nem lehet minden problémát megoldani. Nem mindig sikerül pl. a feladatban szereplő adatok számára még csak közelitő 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 jegyveszteség igen nagy lesz.

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

7. Fejezet.

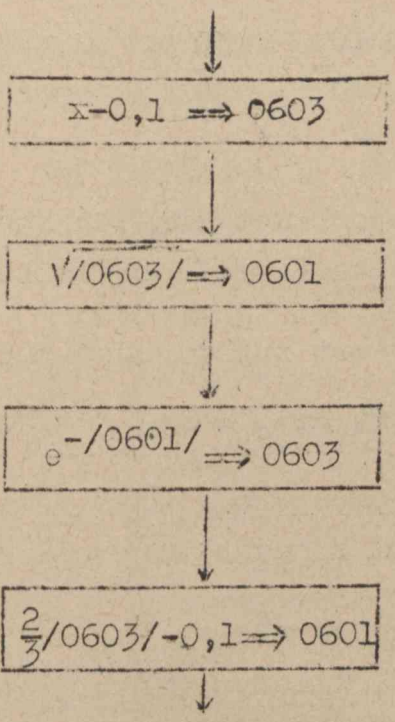
SZUBRUTINOK ALKALMAZÁSA.

7.1. Nyilt és zárt szubrutinok.

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

Készitsünk programot az $y = \frac{2}{3} e^{-\sqrt{x-0,1}} - 0,1$ függvény értékeinek kiszámitásához. A program elkészitésénél tegyük fel, hogy a $0,1 \leq x < 1$ intervallumba eső x érték a 0600 cimü memóriarekeszben van elhelyezve, a hozzátartozó y értéket pedig a 0601 cimü memóriarekeszbe kell bevinni. A programot magát a 0100 cimü rekesztől kezdve, a program konstansait a 0500 cimü rekesztől kezdve helyezzük el, munkarekeszeknek a 0602-0603 rekeszeket használjuk.

A számitás menetrendje igen egyszerü:



Az első blokknak megfelelő utasitáscsoportot azonnal felirhatjuk:

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 elkészitettük. Ez a program[a] /véletlenül választott/ 0011 cimü rekeszben tárolt szám négyzetgyökét képezi az y_0= /0015/ kezdőértékből kiindulva a 0012 cimü rekeszben. A program a következő konstansokból, illetve utasitásokból 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 felirt programot könnyen "beépithetjük" a tekintett példa programjába, ha figyelembevesszük a következőket:

a./ Az y_0 kezdőértéket numerikusan meg kell adni; jelen esetben célszerü 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 továbbá is felhasználhatnók a 0011-0017 rekeszeket /feltéve, hogy ezek a rekeszek nincsenek valami más célra lefoglalva/, azonban az egyöntetüség kedvéért, és szükségtelen átviteli utasitások megtakaritása végett mégis célszerübb a négyzetgyökvonó program konstansait a 0501-0502 és 0515 rekeszekben, az $\underline{x}$ számot a 0603 rekeszekben, az $y_0 = 1-2^{-30}$ értéket a 0503 rekeszben, a négyzetgyökvonás eredményét a 0601 rekeszben tárolni, munkarekesznek pedig a 0602 rekeszt használni. Ennek megfelelően természetesen a négyzetgyökvonás programját teljesen "át kell cimezni": minden mennyiség régi cime helyébe az illető mennyiség uj cimét kell beirni. Az átcimezésnél jól felhasználhatjuk az alábbi "átcimező táblázatot":

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

A második blokknak végeredményben a következő utasitáscsoport 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 kiszámitására készitett programot. Ebben a programban nem tüntettük fel a szereplő mennyiségek valódi /abszolut/ cimeit, hanem relativ cimeket /betü-cimeket, illetve szögletes zárójellel kijelölt cimeket/ alkalmaztunk. A program az m cimü rekeszben tárolt $\bar{x} = \frac{x}{16}$ értékből az e^{-x} értéket az m+2 cimü rekeszben állitja elő, és a következő konstansokból, illetve utasitásokból áll:

/k/ = ↓+20 k+3 m+2; /k+1/ = +20 k+10 m+2;
/k+2/ = +0,00000744; /k+3/ = +0,00006399;
/k+4/ = +0,00133662; /k+5/ = +0,01559902;
/k+6/ = +0,12500546; /k+7/ = $\frac{1}{2}$;

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

t	Á	05	k	t+12
t+1	Á	05	$\langle 4.2^{-30} \rangle$	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	+	09	$\langle 2^{-30} \rangle$	m+1
t+1	Á	05	k+2	m+2
t+11	x	03	m	m+2
t+12	/↓+	20	k+3	m+2/
t+12	+	00	$\langle 2^{-18} \rangle$	t+12
t+14	↓1-1,	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 beépithetjük, ha a relativ cimek helyébe megfelelő abszolut cimeket irunk be. A jelen esetben a t, k és m betü-cimeknek célszerü a t = 0113, k = 0504, m = 0601 értékeket adni. Legyen továbbá $\langle \frac{1}{2} \rangle$ = 0513, $\langle 2^{-18} \rangle$ = 0514, $\langle 2^{-30} \rangle$ = 0515, $\langle 4.2^{-30} \rangle$ = 0516.

A helyettesitéseket elvégezve azonnal felirhatjuk a harmadik blokknak megfelelő utasitáscsoportot:

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	0602
0135	FU	34	0136	0133

A negyedik blokknak megfelelő utasitáscsoport:

0136	x,	13	0517	0003
0137	↓-	21	0500	0601
0140	. . . . .			

ahol /0517/= $\frac{2}{3}$ és /0520/ = 2^{-4}

Az adott programozási feladat ismertetett megoldásában két előre elkészitett szubrutint ugy használunk fel, hogy a kész szubrutinokat mintegy "beleépitettük" az összeállitandó programba, vagyis a szubrutinok utasitásait, konstansait, stb. megfelelő módon átcimezve az összeállitandó program többi részének - az u.n. főprogramnak - utasitásai, konstansai közé helyeztük. A gép mindegyik szubrutin végrehajtását a memóriában közvetlenül a szubrutin utasitásai előtt álló utasitás végrehajtása után kezdi meg, és a szubrutin végrehajtása után a memóriában közvetlenül a szubrutin utasitásai után következő utasitás végrehajtására tér rá. Ilyen esetben azt mondjuk, hogy az előre elkészitett szubrutinokat - példánkban az $y = \sqrt{x}$ és az $y = e^{-x}$ függvény szubrutinját - mint nyilt szubrutinokat használtuk fel az összeállitandó programban. A nyilt szubrutinok tehát a program szerves részévé válnak.

Az adott programozási feladatot azonban másképpen is megoldhatjuk. A két szubrutint ennél a megoldásnál nem épitjük bele a programba, hanem a főprogramtól és egymástól teljesen függetlenül, a memória két tetszőleges helyére visszük 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 cimü 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, $\langle 4.2^{-30}\rangle$ = 0064, m = 0070/ A főprogram a független változó kiszámitott értékét a 0011 /illetve a 0070/ rekeszbe viszi be, és egy-egy feltétlen ugró utasitás segitségével átadja a vezérlést az $y = \sqrt{x}$ /$y = e^{-x}$/ függvény értékeit kiszámitó szubrutinnak /"behivja" a megfelelő szubrutint/. Természetesen mindkét esetben gondoskodni kell arról, hogy a szubrutin végrehajtása után a gép ismét visszatérjen a főprogramra. Ezt a legegyszerübben ugy érhetjük el, hogy mindegyik szubrutin behivása előtt egy-egy feltétlen ugró utasitást vitetünk be a főprogrammal a 0010 /illetve a 0043/ cimü rekeszbe. A főprogram a következő konstansokból illetve utasitásokból 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	
0105	U1	24	0020	0070	/0504/= $\frac{2}{3}$
0106	x,	13	0504	0072	
0107	↓-	21	0501	0601	

Ez a megoldás tipikus példa volt arra, hogy egy program összeállitásánál előre elkészitett szubrutinokat ugy 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 utasitás segitségével adja át a vezérlést a megfelelő szubrutin első utasitásának, és a szubrutin végrehajtása után egy fel-

tétlen ugró utasitás adja vissza a vezérlést a főprogramnak. Ilyenkor azt mondjuk, hogy az előre elkészitett szubrutinokat, mint zárt szubrutinokat használtuk fel az összeállitandó programban.

A kidolgozott példából látható, hogy zárt szubrutinok alkalmazása nyilt 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 megkönnyiti 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 nyilt szubrutinokkal szemben megvan az az előnyük is, hogy ugyanazt a zárt szubrutint többször, a főprogram különböző helyein be lehet hivni. Zárt szubrutin alkalmazásával könnyen megoldhatjuk például az alábbi programozási feladatot:

Készitsünk programot az

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

függvény értékeinek kiszámitásához. A $0,1 \leq x \leq 0,9$ intervallumba eső x érték a 0600 cimü rekeszben van elhelyezve, az y értéket a 0601 cimü rekeszbe kell bevinni. A négyzetgyökvonó szubrutin utasitásai a 0000-0007 cimü 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 készitsünk programot ugyanilyen feltételek mellett az $y = \sqrt[4]{0,9-x} - \sqrt[4]{x-0,1}$ függvény értékeinek kiszámitásához; a negyedik gyökvonást kétszeres négyzetgyökvonással végeztessük el.

7.2. Zárt szubrutinok behivása; visszaugrás a főprogramra.

Az eddig ismertetett példákban a zárt szubrutinokat mindig ugy hivtuk be, hogy először egy átviteli utasitással egy feltétlen ugró utasitást vitettünk be a szubrutint követő rekeszbe /ennek eredeti tartalma irreleváns /majd egy feltétlen ugró utasitással átadattuk a vezérlést a szubrutin kezdőutasitásának. Ha ezt a behivási sémát alkalmazzuk, akkor minden szubrutin behivása a főprogramban általában két-két utasitásba és egy-egy konstansba "kerül".

Sok esetben előnyösebb a zárt szubrutinokat a következő két utasitás segitségével behivni:

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

ahol m a szubrutin behivását közvetlenül megelőző utasitás rekesze, b a szubrutin kezdő utasitásának rekesze, a k rekesz a + 64 0000 0002 konstansot tartalmazza és b' a szubrutin után következő rekesz. A két utasitás közül az első <u>önmagához</u> hozzáadja a + 64 0000 0002 konstansot, és igy a B-regiszterben egy + 74 k a + 3 alaku utasitást hoz létre, a második

ezt az utasitást a b' rekeszben helyezi el, és átadja a vezérlést a szubrutin kezdőutasitásának. A + 64 0000 0002 konstansot célszerü "szabványos" konstansként mindig egy meghatározott memóriarekeszben, például a 0001 cimü 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 behivása ezzel a módszerrel a főprogramban két-két utasitásba, és összesen egy konstansba kerül.

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

1.Példa: Készitsünk programot az

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

függvény értékeinek kiszámitására, 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 abszolut 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 állitja elő a 0025 rekeszben, a szubrutin utasitásai 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 értéket a 0151 cimü rekeszben állitja elő, a szubrutin utasitásai a 0100-0111 rekeszekben vannak elhelyezve.

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

d./ $y = \log x$ szubrutinja. Ez a szubrutin a 0350 rekeszben

elhelyezett x értékhez /$e^{-1} < x < 1$/ tartozó y értéket a 0351 rekeszben állitja elő, a szubrutin utasitásai a 0300-0332 rekeszekben vannak elhelyezve.

A főprogram utasitásait az 1000 cimü rekesztől kezdve, konstansait az 1100 cimü 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/. Készitsünk olyan programot, amely a 200 számot összegezi, az eredményt decimális alakban kinyomtatja, végül pedig megállitja a gépet.

A program elkészitésénél foltesszük, 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 utasitásai 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 utasitásai a 0500-0511 rekeszekben vannak elhelyezve.

Feltesszük továbbá, hogy az összeadandók ugy vannak normálva, hogy az összegezés közben nem lép fel tulcsordulás.

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

/1100/ = 0
/1101/ = 2^{-30}
/1102/ = 200.2^{-30}
/0001/ = +64 0000 0002
/szabványos konstans!/

1000	Á	05	1100	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	FU	34	1002	1010
1010	Á	05	1200	0550
1011	+,	10	0001	1011
1012	U1	24	0500	0512
1013		+77	0000	0000

Az utolsó utasitás 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 szüksóges adatokat a perforált szalagról /binárisan kódolt/ decimális alakban olvassa le, az eredményt docimális alakban kinyomtatja és végül megállitja a gépet.

3.Példa: A perforált szalagra binárisan kódolt decimális alakban folvitettünk 50 mérési adatot. Kiszámitandó 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 állitsa meg a gépet.

A program elkészitésénél 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 tulcsordulás veszélyét ugy 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 számitjuk ki.

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

/1100/ = 0
/1101/ = 2^{-30}
/1102/ = $\frac{1}{50}$
/1103/ = +05 0451 0762
/1104/ = 2^{-18}
/1105/ = +11 0762 1200
/0001/ = +64 0000 0002
/szabványos konstans!/

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

1011	Á	05	1100	0021	
1012	/-,	11	0700	1200/	
1013	U1	24	1014	1201	
1014	↓x,	33	1201	-	
1015	↓x,	33	1102	-	
1016	↓+	20	0021	0021	D kiszámitása
1017	+	00	1104	1012	
1020	↓-,	31	1105	-	
1021	FU	34	1012	1022	
1022	+,	10	0001	1022	
1023	U1	24	0010	0020	
1024	Á	05	0025	0550	
1025	+,	10	0001	1025	
1026	U1	24	0500	0512	
1027	Á	05	1200	0550	$\bar{x}$ és D kinyomtatása
1030	+,	10	0001	1030	
1031	U1	24	0500	0512	
1032	+	77	0000	0000	megállás

Az 1003 és 1012 cimü rekeszek tartalmát szándékosan nem állitottuk helyre; teljes programok változó utasitásait általában nem szoktuk helyreállitani.

Gyakorlásként készitsünk 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 abszolut értékben egynél kisebb számok, $\frac{a-b}{h}$ egész szám/. A program elkészitésénél 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 utasitásait az 1000, konstansait az 1100 cimü memóriarekesztől kezdve helyezzük el.

7.3. Programparaméterek beültetése szubrutinokba.

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

A program a következő volt:

/0502/ = 00 00010001

1006	Á	05	$\langle 15.2^{-30} \rangle$	0602
1007	Á	05	$\langle 0 \rangle$	0603
1010	/x,	13	0000	0200/
1011	↓+	20	0603	0603
1012	+	00	0502	1010
1013	-	01	$\langle 2^{-30} \rangle$	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 behivása előtt mindig át kellene vinni a 0000-0017, illetve a 0200-0217 rekeszekbe. Készithetünk azonban olyan általános szubrutint is, amely össze tudja szorozni skalárisan azt az n dimenziós a vektort, amelynek elemei az a, ..., a+n-1 rekeszben vannak elhelyezve, azzal a szintén n dimenziós b vektorral, amelynek elemei a b,..., b+n-1 cimü rekeszekben vannak tárolva /a, b és n a szubrutin u.n. paraméterei/.

A szubrutin összeállitásánál először tegyük fel azt, hogy az a, b és n paraméter-értékek 2^{-18}-al megszorozva rendre a p, p+1, p+2 cimü rekeszekben vannak elhelyezve.

Az a b skaláris szorzat értékét a szubrutin a p+3 cimü 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: behivása előtt a p, p+1, p+2 rekeszekben mindig el kell helyezni az $a.2^{-18}$, $b.2^{-18}$, $n.2^{-18}$ értékeket. Ez a főprogramban a szubrutin minden egyes behivásánál általában három utasitásba és három konstansba kerül. Ilyen esetekben sokszor célszerü egy memóriarekeszben több paramétert elhelyezni. Igy pl., ha $n \leq 63$, az n paraméter értékét elhelyezhetjük a p cimü rekeszben a müveleti jel helyén; az a és b paramétereket pedig /a és b memóriarekeszek cimeit jelentik és igy 2048-nál kisebbek/ ugyanebben a rekeszben az első, illetve a második cim helyén: /p/ = $n.2^{-6}$ + + $a.2^{-18}$ + $b.2^{-30}$. Az a b skaláris szorzat értékét a szubrutin ebben az esetben a p+1 cimü rekeszben tárolhatja; a p+2 és p+3 cimü paraméter-rekeszekre nincs szükség. Ennek megfelelően természetesen az előbb felirt szubrutint át kell alakitani. Az uj 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	∧,	16	k	p
t+1	↓+	20	k+1	t+6
t+2	∧,	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+8	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-ödrendü $\underline{\underline{A}}$ mátrixot, amelynek elemei "sorfolytonosan" a 0100-2410 rekeszekben vannak tárolva /vagyis az A mátrix a_{ij} eleme /i,j = 1, ..., 35/ a 0100+/i-1/ 45^{+} +/j-1/ rekeszben van elhelyezve/ azzal a $\underline{b}$ vektorral, amelynek elemei a 2500-2542 rekeszben vannak tárolva, Az $\underline{\underline{A}}$ $\underline{b}$ 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 utasitásait elhelyezhetjük például a 0005 rekesztől kezdve /vagyis t = 0005/; legyen továbbá p = 0021, k = 0023, m = 0031.

+ 35 nyolcas rendszerben 43

Az output szubrutin utasitásait a0032 rekesztől kezdve, a főprogram utasitásait 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 memóriarekeszbén helyeztük volna el/, akkor a főprogram két utasitással és egy konstanssal tartalmazna többet.

7.4. Szabványos szubrutinok relativ cimzéssel.

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 elkészitett szubrutinok együttvéve többszörösen kitöltenék a gép memóriakapacitását. Ezért a szubruti-

nok utasitásainak, konstansainak, paramétereinek és munkarekeszeinek célszerü relativ cimeket /betü-cimeket/ adni. A relativ cimeket valóságos /abszolut/ cimekké átalakitani általában könnyebb, mint az abszolut cimekkel elkészitett programot átcimezni. Az eddig kidolgozott szubrutinok közül relativ cimzéssel készitettük el a skaláris szorzás és az $y = e^{-x}$ függvény szubrutinját.

A relativ cimezés egymagában persze még nem oldja meg a szubrutinok racionális felhasználásának problémáját. A relativ cimek átalakitása abszolut cimekké teljesen gépies, de hosszabb szubrutinoknál elég hosszadalmas és fáradságos munka. Ezenkivül a szubrutint minden esetben ujra kell lyukasztani, ami szintén elég sok időt és munkát vesz igénybe. Gyökeres megoldást csak az jelent, ha egy olyan programot készitünk az M-3 gépre, amelynek segitségével a perforált szalagra relativ cimezéssel fellyukasztott szubrutinokat a gép automatikusan a memória kijelölt helyeire viszi be, és közben a relativ cimeket átalakitja megfelelő abszolut cimekké. Az ilyen programot beindító rutinnak, a beindító rutin segitségével a memóriában akárhol elhelyezhető, relativ cimzésü 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 át az M-3-ra.

Ez a beindító rutin minden olyan relativ cimzésü szubrutin bevitelére alkalmas, amelyben a relativ cimzés alapjainak száma /a szereplő "betük" száma/ legfeljebb 15. A relativ cimezés alapjait célszerü az a, b, c,...,o betükkel jelölni. A szabványos szubrutinoknak az a, b, c,...o

alapu relativ cimeken kivül szerepelhetnek abszolut cimek is /pl. a szabványos konstansok cimei/. A szabványos szubrutinok konstansait a memóriában általában közvetlenül a szubrutin utasitásai után helyezzük el.

Egy szabványos szubrutin utasitásait és konstansait ennek megfelelően egy szalagra visszük fel, éspedig először az utasitásokat, utánuk a konstansokat. Egy-egy utasitás, illetve konstans felvitele a perforált szalagra a következő módon történik: minden egyes utasitást /konstanst/ először a relativ cimek alapjainak feltüntetése nélkül lyukasztunk fel /tehát például a +05 b + 7 c + 11 konstansnál először +05 0007 0011-et lyukasztunk/. Az utasitás /konstans/ végét jelző "irás" jel után még öt sort perforálunk a következő szabály szerint: az első két sorba 00-t perforálunk, ha az első cim abszolut, 01-et, ha az első cim relativ, és alapja a,..., 17-et, ha az első cim relativ, és alapja o. A következő két sorba 00-t lyukasztunk, ha a második cim abszolut, 01-et, ha a második cim relativ, és alapja a,...,17-et, ha a második cim relativ, és alapja o. Az ötödik sorba egy "irás" 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 relativ cimezéssel való bevitelén kivül még egy jelentős alkalmazása van. A szubrutinokban azokon a paramétereken kivül, amelyeknek értékét a főprogram helyezi el a szubrutin paraméter-rekeszeiben, gyakran szerepelnek olyan paraméterek is, amelyeknek értékét minden egyes alkalommal a programozó irja be a szubrutin megfelelő helyeire. Az előbbieket program-paramétereknek, az utóbbiakat előre beállitott /preset/ paramétereknek nevezzük. Igy például a skaláris szorzás fent ismertetett szubrutinjában a, b és n program-paraméterek.

Ha a, b és n előre beállitott paraméterek lennének, akkor a szubrutin egyszerüen a következő lenne /a b ==> pl/:

```
/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    ↓-,  31   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 beirni. A beinditó rutin segitségével a szabványos szubrutin bevitelénél a gép elvégzi a relativ cimeknek abszolut cimekké való átalakitásán kivül az előre beállitott paraméterek értékeinek behelyettesitését 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 szubrutinja.

Ennek a függvénynek a kiszámitására 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. Igy $2^{-15} \leq x < 2^{15}$. /Ez az intervallum elég nagy ahhoz, hogy a gyakorlatilag előforduló értékek logaritmusát ki tudjuk számitani./ Ekkor $|\ln x| < 2^4$ lesz. Igy a gép kapacitásának teljes kihasználása céljából az eredménynél a bináris pontot a negyedik helyérték után helyezzük el. Helyettesitsünk $x = 1 + y$-t és fejtsük az $\ln x = \ln /1+y/$ függvényt Taylor sorba.

$$\ln /1+y/ = y - \frac{y^2}{2} + \frac{y^3}{3} - + \ldots + /-1/^{n-1} \frac{y^n}{n} + \ldots$$

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 $x = d.2^q$; $\frac{1}{2} \leq d < 1$; $-15 \leq q < 15$ alakba irjuk. Tudjuk, hogy ez az x számnak az un. lebegőpontos alakja. Mivel a gépben $\bar{x} = x.2^{-15}$ van elhelyezve, ezt hozzuk $\bar{x} = d.2^{q-15} = d.2^k$ alakra, ahol $-30 \leq k < 0$. Ekkor

$$\log x = \log d + q.\ln 2.$$

Tehát a feladat log d kiszámitására redukálódott, ahol $\frac{1}{2} \leq d < 1$.

II. A $t = (\frac{6}{5})^i d$ leképezéssel, ahol $i = 1,2,3,4$ aszerint, 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. Igy

$$\ln t = \ln /1+\frac{z+1}{10}/ = \frac{z+1}{10} - \frac{1}{2} /\frac{z+1}{10}/^2 + \ldots - \frac{1}{10} /\frac{z+1}{10}/^{10} +$$

$$+ R_{10},$$

ahol az R_{10} maradéktagra az alábbi becslés adható:

$$|R_{10}| = \left| \frac{1}{11} \cdot \left(\frac{\frac{z+1}{10}}{1+\xi} \right)^{11} \right| \leq \frac{1}{11} \left(\frac{1}{5} \right)^{11} < 0{,}2.10^{-8}$$

mert

$$0 \leq \xi \leq \frac{z+1}{10} \leq \frac{1}{5}$$

A hatványozásokat elvégezve:

$$\ln t = c_0' + c_1' z + c_2' z^2 + \ldots + c_8' z^8 + c_9' z^9 + c_{10}' z^{10} + R_{10}$$

Az együtthatókat kiszámitva azt kapjuk, hogy

$$|R'| = |c_8' z^8 + c_9' z^9 + c_{10}' z^{10}| < 0{,}08.10^{-8},$$

tehát ez a három tag a gépi számitásnál elhagyható. Ezáltal a közelitő polinóm hetedfokura redukálódott. Ennek a fokszáma még kettővel csökkenthető a Csebisev-polinomok segitségével; z^6 t.i. egy negyedfoku, z^7 pedig egy ötödfoku polinommal pótolható a tekintett intervallumban, éspedig ugy, hogy a hiba R' nagyságrendjével összemérhető. Az n-ed foku Csebisev polinom definiciója 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 felirhatjuk a hatod és hetedfoku Csebisev polinomokat:

$$T_7/z/ = z^7 - \frac{7}{4}z^5 + \frac{7}{8}z^3 - \frac{7}{64}z \leq \frac{1}{64}$$

$$T_6/z/ = z^6 - \frac{3}{2}z^4 + \frac{9}{16}z^2 - \frac{1}{32} \leq \frac{1}{32}$$

Igy

$$c_6' z^6 + c_7' z^7 = c_6' \left(\frac{3}{2}z^4 - \frac{9}{16}z^2 + \frac{1}{32}\right) + c_7' \left(\frac{7}{4}z^5 - \frac{7}{8}z^3 + \frac{7}{64}z\right) + R''$$

$R'' = c_6' T_6 /z/ + c_7' T_7 /z/$ és igy $R'' \leq 0{,}32.10^{-8}$

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

$$|R| \leq |R_{10}| + |R'| + |R''| < 0{,}6.10^{-8}$$

és

$c_0 =$ 0, 095 310 177
$c_1 =$ 0, 090 909 092
$c_2 =$ -0, 004 132 178
$c_3 =$ 0, 000 250 432
$c_4 =$ -0, 000 017 217
$c_5 =$ 0, 000 001 254

$\underline{\ln}\, x \approx q\, \underline{\ln}\, 2 + \underline{\ln}\, d = q\, \underline{\ln\, 2} - i\, \underline{\ln}\, \frac{6}{5} + \underline{c_4} + \ldots + \underline{c_5 z^5}$,

ahol az aláhuzás 2^{-4}-el való szorzást jelent.

Program:

0001	11	-,	0101	0100	$x \leq 0$-ra
0002	34	FU	0003	0004	megállás
0003		megállás			
0004	12	:,	0102	0103	16 $\underline{\ln}$ 2 $\Rightarrow$ 0116
0005	24	U1	0006	0116	
0006	05	Á	0100	0117	
0007	01	-	0103	0116	$d = 2^k x \Rightarrow$ 0117
0010	51	|-|,	0104	0117	
0011	34	FU	0012	0014	/0116/ =
0012	02	:	0104	0117	= /15+k/$\underline{\ln 2}$ = $\underline{g\ln 2}$
0013	24	U1	0007	0117	

0014	03	x	0104	0117	$\frac{d}{2} \Rightarrow$ 0117
0015	01	-	0105	0116	$(\frac{6}{5})^i$ d-1 $\Rightarrow$ 0120
0016	02	:	0106	0117	
0017	21	↓-	0104	0120	
0020	34	FU	0015	0021	/0116/ = q ln 2 -
0021	02	:	0104	0120	- i ln $\frac{6}{5}$
0022	11	-,	0107	0120	$z = \frac{(\frac{6}{5})^i d-1-0,1}{0,1} \Rightarrow$
0023	22	↓:	0107	0120	$\Rightarrow$ 0120
0024	05	Á	0110	0117	
0025	13	x,	0117	0120	
0026	/ 20	↓+	0111	0117/	ln t
0027	00	+	0035	0026	polinom kiszámitása
0030	51	|-|,	0026	0036	Horner elrendezés-
0031	34	FU	0032	0025	sel
0032	01	-	0037	0026	
0033	00	+	0116	0117	$\longrightarrow$ ln x
0034		/kiugrató utasitás helye/			
0035	00	+	0001	0000	
0036	20	↓+	0115	0117	Pseudó utasitások
0037	00	+	0005	0000	

Konstansok:

/0000/ = 0
/0100/ = x
/0101/ = 2^{-30}
/0102/ = 2^{-4}
/0103/ = ln 2
/0104/ = $\frac{1}{2}$
/0105/ = ln $\frac{6}{5}$
/0106/ = $\frac{5}{6}$
/0107/ = 0,1
/0110/ = c_5
/0111/ = c_4
/0112/ = c_3

/0113/ = c_2
/0114/ = c_1
/0115/ = c_0

b./ Abban az esetben, ha $0 \leq x \leq 1$, az ln /1+x/ függvény egy nyolcadfoku polinommal jól közelithető.

$$\ln /1+x/ = a_1x + a_2x^2 + \ldots + a_8x^8$$

ahol

$a_1 = 0,9999964239$ $\quad a_5 = 0,1676540711$

$a_2 = -0,4998741238$ $\quad a_6 = -0,0953293897$

$a_3 = 0,3317990258$ $\quad a_7 = 0,0360884937$

$a_4 = -0,2407338084$ $\quad a_8 = 0,0064535442$

és $|R'| < 4.10^{-8}$

/A képlet megtalálható O. Hastings: "Approximations for Digital Computers" cimü könyvében./

A fenti polinom az ln /1+x/ függvényt a $0 \leq x \leq 1$ intervallumban közeliti jól. Ha $y = 1 + x$ az $1 \leq y \leq 2$ szakaszon kivül esik, akkor az $\bar{y} = \frac{y}{2^k}$ transzformáció segitségével 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 kiszámitani. Mindezek a transzformációk természetesen a főprogramban végzendők el.

Igy a program egy nyolcadfoku polinom kiszámitására korlátozódott. Ezt a nyolcadfoku polinomot a Horner-elrendezés segitségével számitjuk 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 utasitás 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 egyszerübb 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 alkalmazható, 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 közelitő polinomokat.

$\sin \frac{\pi}{2} x$ kiszámitása pl. a következő közelitő 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 számitás relativ hibája kisebb lesz, mint 5.10^{-9}. Az M-3 gépen pedig az eredmény maximális pontossága csak 7 tizedesjegy. Mivel a trigonometrikus függvények argumentuma $-\infty$-től $+\infty$-ig változhat, a számitás megkezdése előtt az argumentumot ugy kell redukálni, hogy abszolut értékben ne legyen nagyobb $\frac{\pi}{2}$-nél.

Ezt a redukciót több lépésben hajtjuk végre. Az eredeti argumentumot y-al jelöljük.

I. Az argumentum abszolut értékét vesszük. Ezt megtehetjük, mivel

$$\sin /-y/ = -\sin y$$
$$\cos /-y/ = \cos y$$

Tehát sin |y| kiszámitása után, ha az eredeti argumentum negativ volt, a kiszámitott függvényértéket ellenkező előjellel kell venni.

II. $|y|$ -ból levonjuk 2π egész számu többszörösét ugy, hogy $-\pi \leq |y| - 2k\pi = y \leq \pi$ 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 számitjuk 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 közelitő polinom $\sin \frac{\pi}{2} x$ kiszámitására 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. Igy $\sin \dot{y}$-ot fogjuk kapni eredményül.

$\cos \dot{y}$ -ot a következő összefüggés segitségével kapjuk meg:

$$\cos \dot{y} = \sin /\frac{\pi}{2} - |\dot{y}|$$

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 közelitést.

A trigonometrikus függvények kiszámitására kétféle szubrutint ismertetünk:

a./ Szubrutin, amely vagy $\sin \dot{y}$ vagy $\cos \dot{y}$ $/ \frac{\pi}{2} \leq \dot{y} \leq \frac{\pi}{2}/$ kiszámitására alkalmas, aszerint, hogy hol kezdi el a gép a program végrehajtását.

Mivel a Hastings-féle közelitő 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:

$$/0040/ = \frac{c_1}{2}$$

$$/0041/ = \frac{c_3}{2}$$

$$/0042/ = \frac{c_5}{2}$$

$$/0043/ = \frac{c_7}{2}$$

$$/0044/ = \frac{c_9}{2}$$

/0045/ = + 20 0043 0057
/0046/ = + 20 0040 0057
/0050/ = + 77 7777 7777
/0051/ = + 00 0001 0000
/0052/ = $\frac{1}{2}$

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 számitásánál 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		kiugrató utasitás helye		

b./ Szubrutin, amely kiszámitja sin x-et, cos x-et és tg x és cotg x közül az egyiket aszerint, hogy $|x|$ kisebb vagy nagyobb $\frac{\pi}{4}$ -mél.

Ugyanis, ha $|x| < \frac{\pi}{4}$, akkor tg x < 1 és cotg x > 1
ha $|x| > \frac{\pi}{4}$, akkor cotg x < 1 és tg x > 1

tg x-et és cotg x-et a következő összefüggések felhasználásával számitjuk:

$$\text{tg } x = \frac{\sin x}{\cos x}; \quad \text{cotg } x = \frac{\cos x}{\sin x}$$

A szubrutin ugyanazokat a konstansokat használja, mint az előbbi, ezeken kivül azonban még további hármat:

$$/0047/ = \frac{2}{\pi}$$

$$/0053/ = 2^{-30} = +\ 00\ 0000\ 0001$$

$$/0054/ = 0 \quad = +\ 00\ 0000\ 0000$$

Az x argumentumot $/-1 < x < 1/$ a 0060-as rekeszben helyezzük el. Az eredmények a következő rekeszbe kerülnek:

sin x ⇒ 0061
cos x ⇒ 0062
tg x ⇒ 0063
cotg x ⇒ 0064

tg x vagy cotg x értéke egynél nagyobb lesz: ennek a helyére a gép 0-t ir, 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	0021	
0021	Á	05	0057	0061	⟵ sin x
0022	U2	74	-	0003	
0023	Á	05	0057	0062	⟵ cos x
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 utasitás helye			

3. A trigonometrikus függvények inverzeinek szubrutinjai:

3.a./ Arc sin x és arc cos x szubrutinja:

Felhasználjuk az

$$\arcsin x = \frac{\pi}{4}\left\{ \operatorname{sign} y_1 + \frac{1}{2} \operatorname{sign} /y_1 y_2/ + \ldots + \right.$$

$$\left. + \frac{1}{2^{k-1}} \operatorname{sign} /y_1 \ldots y_k/ + \ldots \right\}$$

sorfejtést, ahol $y_1 = x$

$$y_{k+1} = 2 y_k^2 - 1$$

Ez a sorfejtés arc sin x-et lépcsősfüggvények limeseként állitja elő. Szoritkozzunk $|x| \leq 1-2^{-30}$-ra. Ez nem jelent lényeges megszoritást, mert $|x| = |\sin y| \leq 1$.

A gép kapacitását figyelembevéve, 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 értékü tagokat a gép már nem tudja figyelembe venni s az ezen tag utáni maradékösszeget sem, mivel az összegezés szukcesszive történik. Igy tehát $k \leq 30$ adódik. A $k \geq 30$ indexü tagok maradékösszege t.i. szintén kisebb, mint 2^{-29}, hiszen

$$|\arcsin x - S_k| = \left| \frac{\pi}{4} / \frac{1}{2^k} \operatorname{sign} /y_1 \ldots y_{k+1}/ + \ldots / \right| \leq$$

$$\leq \frac{\pi}{4} / \frac{1}{2^k} + \frac{1}{2^{k+1}} + \ldots / = \frac{\pi}{4} \frac{1}{2^{k-1}}$$

A $-\frac{\pi}{2} \leq \arcsin x \leq \frac{\pi}{2}$ 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 arc sin /-x/ = - arc sin x tehát akkor sincs baj, ha az argumentum negativ, csak ezt a végeredménynél figyelembe kell venni.

Emellett arc cos x = $\frac{\pi}{2}$ - arc sin x és igy arc sin x kiszámitott értékéből könnyen megkapjuk arc cos x értékét is. Az y_{k+1}-et meghatározó rekurziós formulában y_{k+1} helyett $\frac{1}{4}y_{k+1}$-el számolunk. Ezt megtehetjük, mivel ennek csak az előjelére van szükség. Tehát

$$\frac{y_{k+1}}{4} = 2\left(\frac{y_k}{4}\right)^2 - \frac{1}{2}$$

A felhasználásra kerülő konstansok a következők:

/0000/ = 0

/0010/ = x

/0011/ = $\frac{\pi}{2} \cdot 2^{-1}$

/0013/ = $\frac{1}{2}$

/0014/ = $\frac{1}{4}$

2^{-1} arc sin x ⇒ 0036

2^{-1} arc cos x ⇒ 0037

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	U	24	0025	0040	
0025	+	00	0040	0036	⟵ 2^{-1} 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 utasitás 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 szoritkozzunk $|x| < 2^{15}$-re. Tekintsük tehát az x argumentum helyett az $x.2^{-15}$-öt:

$$\text{arc tg } x = \text{arc sin } \frac{x.2^{-15}}{\sqrt{2^{-30} + x^2.2^{-30}}}, \text{ illetve}$$

$$\text{arc cotg } x = \text{arc cos } \frac{x.2^{-15}}{\sqrt{2^{-30} + x^2.2^{-30}}}$$

Mivel mind arc tg x, mind pedig arc cotg x értékkészlete tullépi az egységet, ezért 2^{-1} arc tg x-et, illetve 2^{-1} arc cotg x-et számitjuk ki.

Legyen a négyzetgyökvonó szubrutin az α-tól $\alpha+k-1$-ig terjedő rekeszekben elhelyezve, és legyen az argumentum helye a δ az eredmény helye a β rekesz.

A felhasznált rekeszek a következők:

/0046/ = $x.2^{-15}$

/0044/ = + 74 0054

/0045/ = 2^{-30}

2^{-1} arc tg x ⇒ 0036; 2^{-1} arc cotg x ⇒ 0037

Program:

0050	Á	05	0044	+k	
0051	x,	13	0046	0046	
0052	+,	30	0045	-	
0053	Ul	24			
0054	:	02		0046	
0055	Ul	24	0015	0010	áttérés arc sin x és arc cos x kiszámitására, amelyek eredményeképp a keresett függvényértékeket kapjuk.

7.5.2. Szubrutin az $\int_a^b f\,/x/\,dx$ kvadraturájához.

A numerikus kvadratura szubrutinja valamely függvény határozott integráljának közelitő értékét számolja a Simpson-formula alapján; igy

$$\int_a^b f\,/x/\,dx = \frac{h}{3}\left[f\,/x_0/ + 4f\,/x_1/ + 2f\,/x_2/ + \ldots + 4f\,/x_{n-1}/ + f\,/x_n/\right] + H$$

ahol a H a közelités hibája és

$$h = \frac{b-a}{n}\ ;\ x_0 = a\ ;\ x_i = x_0 + ih$$ és n páros.

Jelöljük max (|a| , |b|) értékét $\frac{L}{2}$ -vel és legyen $|f(x)| < F$ az/a, b/ intervallumban. Ekkor $|\frac{h}{3}| = |\frac{b-a}{3n}| < \frac{L}{3n}$, és a határozott integrálra a Simpson-formula alapján a következő becslést kapjuk:

$$\left| \int_a^b f(x)\,dx \right| < \frac{h}{3}\Big[|f(x_0)| + 4|f(x_1)| + 2|f(x_2)| + \ldots + 4|f(x_{n-1})| + |f(x_n)| \Big] + |H| < \frac{L}{3n} F\,(1+4+2+\ldots+4+1) +$$

$$+ |H| = \frac{L}{3n} F \cdot 3n + |H| = LF + |H|$$

Legyen az a és b skálafaktora L, f /x/ skálafaktora F.

Legyen tehát $\bar{a} = \frac{a}{L}$; $\bar{b} = \frac{b}{L}$; $\overline{f(x)} = \frac{f(x)}{F}$; $\bar{x} = \frac{x}{L}$.

Ekkor az integrál gépi alakjára a következő becslést kapjuk:

$$\int_{\bar{a}}^{\bar{b}} \overline{f(x)}\,d\bar{x} = \frac{1}{L} \int_a^b \frac{1}{F} f(x)\,dx = \frac{1}{LF} \int_a^b f(x)\,dx < \frac{1}{LF} (LF + |H|) =$$

$$= 1 + \frac{|H|}{LF} .$$

Az n növelésével elérhető, hogy |H| tetszőleges kis szám legyen, L és F pedig megválasztható ugy, hogy az

$$\left| \int_{\bar{a}}^{\bar{b}} \overline{f(x)}\,d\bar{x} \right| < 1$$

egyenlőtlenség is teljesüljön.

A fentiekből következik, hogy sem az integrál értéke, sem a közbenső eredmények nem csordulnak tul.

A szubrutin először n=2m-re számolja a közelitő értéket, majd n=4m-re. /Az m érték valamilyen egész szám, pl. m=2./ Ezután összehasonlitja a két közelitést és ha ezek eltérése nagyobb valamilyen ε küszöbszámnál, akkor megkétszerezi az osztópontok számát és az igy kiszámitott közeli-

tést összehasonlitja az előzővel. Ezt az eljárást addig folytatja, mig a két egymásutáni közelités különbsége ε -nál kisebb nem lesz. Nulladik közelitésnek a gépi 1-est /$1-2^{-30}$/ -at vesszük. ε -t a kivánt pontosságnak megfelelően kell megválasztani.

Legyen a szubrutin kezdőrekeszének cime: 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 $\varepsilon = 4.2^{-30}$. A határozott integrál értéke az i+45-ös rekeszben képződik. A visszaugrató utasitást az i+34-es rekeszben kell elhelyezni.

A kvadratura szubrutin behivja az f /x/ függvényt előállitó szubrutint. Legyen $\bar{x}$ cime: r+1, $\overline{f\,/x/}$ cime: r+2. A szubrutin bemenete: r+3, a visszaugrató utasitás 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+2	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 utasitás helye		

Konstansok:

i+53 = 00 0000 0000 = 0

i+35 = 40 0000 0000 = $\frac{1}{2}$

i+36 = 25 2525 2525 = $\frac{1}{3}$

i+37 = 77 7777 7777 = $1-2^{-30}$

i+40 = 64 0000 0002

i+41 = 00 0000 0004 = 4.2^{-30}

i+42 = 00 0000 0002 = 2.2^{-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 szubrutinkönyvtárunkban. Az elemi függvények kiszámitására az itt ismertetett módszereknél általában jobbak is találhatók és a szubrutinkönyvtárban az általunk ismert legjobb módszer van beprogramozva relativ cimekkel, 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 behivtuk/ a perforált szalagról binárisan kódolt decimális alakban leolvasnak egy vagy több számadatot /eközben 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 abszolut 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 $t = \pm 0, \beta_1 \ldots \beta_7$ hétjegyü 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 átalakitást a megfelelő input a szubrutin a következő képlet alapján végzi el:

$$t = \pm \sum_{k=1}^{7} \frac{\beta_k}{10^k} = \sum_{k=1}^{7} \frac{a_k}{(\frac{10}{16})^k} =$$

$$= (((((((a_7 \frac{16}{10} + a_6) \frac{16}{10} + a_5) \frac{16}{10} + a_4) \frac{16}{10} + a_3) \frac{16}{10} + a_2) \frac{16}{10} + a_1) \frac{16}{10}$$

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 cimü 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+1 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	,	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 kivül néha szükség lehet egy olyan input szubrutinra is, amelynek segitségével az eredetileg tizes számrendszerben megadott $t' = \pm \beta'_6 \ldots \beta'_0$ hétjegyü egész számot fixpontos bináris alakra konvertálva

ugy helyezhetjük el a gépben, hogy a fix bináris pont a szó harmincegyedik, utolsó bitje után álljon, illetve - precizebben megfogalmazva - bináris alakban elhelyezhetjük a gépben a $t^{*} = t' \cdot 2^{-30}$ számot. /Az M-3 gépen a fix bináris pont természetes helye 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:

$$\bar{t} = \pm \sum_{k=0}^{6} \beta_k \, 10^k = \sum_{k=0}^{6} \bar{a}_k \left/\frac{10}{16}\right/^k =$$

$$= \left(\left(\left(\left(\bar{a}_6 \frac{10}{16} + \bar{a}_5\right)\frac{10}{16} + \bar{a}_4\right)\frac{10}{16} + \bar{a}_3\right)\frac{10}{16} + \bar{a}_2\right)\frac{10}{16} + \bar{a}_1\Big)\frac{10}{16} + \bar{a}_0$$

ahol

$$\bar{t} = t' \cdot 2^{-28} \; ; \; \beta_k = \beta'_k \cdot 2^{-28} ; \; \bar{a}_k = \pm \beta_k \cdot 16^k ;$$

a $\bar{t}$ szám bináris alakjából $\frac{1}{4}$-el 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 cimü rekeszben helyezi el. /A relativ cimezés alapjait itt, és az ezután következő szubrutinoknál is ugyanazokkal a betükkel jelöljük, mint az első input szubrutinnnál./

/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	0004	$= 2^{-28}$
/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	/\,	16	m+1	m
a+5	↓ +	20	m+2	m+1
a+6	x	03	k+2	m+2
a+7	↓-1,	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 készithetünk olyan input szubrutinokat, amelyek tizes rendszerben felirt számadatokat bináris alakra konvertálva ugy helyeznek el a gépben, hogy a fix bináris pont az i-edik bit mögött legyen /$1 \leq i \leq 30$/. Könnyen összesze lehet állitani olyan input szubrutinokat is, amelyek egymásután n számadatot visznek be binárisra konvertálva az r,..,r+n-1 cimü rekeszekbe; a két programparamétert, r-et és n-et, célszerüen egy + 00 n r alaku szóval adjuk meg /lásd a 7. fejezetben készitett programokat/. Ezeket a szubrutinokat itt most nem ismertetjük.

8.2. Output decimális nyomtatáshoz.

Az M-3 gép output szubrutinjai /amelyek közül az egyiket szintén többször behivtuk az előző fejezetekben ismertetett programokban /egy vagy több, az M-3 gépben bináris alakban tárolt számot decimális alakra konvertálva a távgépirón kinyomtatnak.

A gyakorlatban itt is az az eset a legfontosabb, amikor a számitás kinyomtatandó végeredménye vagy részeredménye fixpontos bináris alakban keletkezik, és a

fix bináris pont a természetes helyen, vagyis a szó második bitje előtt áll. A tizes számrendszerbe való konverzió ebben az esetben azon a könnyen igazolható tényen alapszik, hogy ha egy kettes számrendszerben felirt, abszolut értékben egynél kisebb számot megszorzunk $\frac{10}{16}$ -al, 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 irva, és ismét $\frac{10}{16}$ -al 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+1
a+7	↓|-|,	71	k+4	-
a+10	FU	34	a+11	a+2
a+11	+n	40	k+3	m+3

A konvertált szám /binárisan kódolt decimális alakban/ megmarad az m+3 cimü rekeszben.

Bizonyos esetekben előfordulhat, hogy egy számitás végeredményei vagy részeredményei a gépben $t'.2^{-30}$ alakuak, ahol t' egy 10^7-nél kisebb egész szám, és a gépnek a $t' = \pm \beta'_6 \ldots \beta'_0$ számot kell tizes számrendszerben kinyomtatni. Az ilyenkor használt konverziós eljárás alapelve a következő: ha

$$t^{*} = t'.2^{-30}; \quad \beta_k^{*} = \beta'_k.2^{-30}; \quad a_k^{*} = \pm \beta_k^{*} 16^k,$$

akkor nyilván

$$t^{*} = \pm \sum_{k=0}^{6} \beta_k^{*} 10^k = \sum_{k=0}^{6} a_k^{*} \left/\frac{10}{16}\right/^k =$$

$$= a_6^{*} \left/\frac{10}{16}\right/^6 + \ldots + a_1^{*} \left/\frac{10}{16}\right/ + a_0^{*}$$

és

$$t^{*} \frac{16}{10} = a_6^{*} \left/\frac{10}{16}\right/^5 + \ldots + a_1^{*} + \frac{a_0^{*}}{10} \cdot 16.$$

A jobboldalon álló első hat tag összege /jelöljük ezt az összeget t^{**}-al/ 2^{-26}-al osztható, a hetedik tag pedig 2^{-26}-nál kisebb; a t^{**} számot tehát a $t^{*} \frac{16}{10}$ szorzatból egyszerüen ugy nyerjük, hogy az utolsó négy bit helyére nullákat irunk. A t^{**} ismeretében viszont a_0^{*}-t azonnal megkaphatjuk:

$$a_0^{*} = t^{*} - \frac{10}{16} t^{**}.$$

Az eljárást ugyanilyen módon tovább folytatva rendre előállithatjuk az $a_1^{*}, \ldots, a_6^{*}$ számokat; ezeknek összegét $\frac{1}{4}$-el 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 cimü rekeszben:

/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	0001 $\approx 2^{-30}$
/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 összeállithatunk 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 egymásután konvertálják és kinyomtatják az r,...r,r+n-1 rekeszekben bináris alakban tárolt számokat /r és n programparaméterek/.

8.3. Relativ cimezésü szubrutinok bevitele.

A 7. fejezetben röviden foglalkoztunk a relativ cimezésü szubrutinok bevitelének kérdésével: megemlitettük, hogy a szabványos alakban elkészitett relativ cimezésü szubrutinokat /az u.n. szabványos vagy könyvtári szubrutinokat/ speciális programok, az u.n. beinditórutinok segitségével lehet a gépbe bevinni, és ismertettük az M-3 gép szubrutinkönyvtárának tervét. Ebben a paragrafusban az M-3 gép két különböző alapelvek szerint müködő beinditórutinját, és a szabványos szubrutinoknak a beinditórutinok segitségével történő bevitelét fogjuk részletesebben tárgyalni.

Az első beinditórutin /amely lényegében az angol EDSAC gép beinditórutinjának elvét követi/ a már ismertetett módon perforált szabványos szubrutinok bevitelére alkalmas: a beviendő szubrutin utasitásait és konstansait a memória megadott cimü rekeszeiben helyezi el, és közben a relativ cimeket megfelelő módon átalakitja abszolut cimekké.

A beinditórutin használata esetén minden egyes szabványos szubrutin bevitele előtt "közölni kell a géppel" a relativ cimezés alapjainak számértékeit. Ez a következő módon történik meg: egy külön szalagra felviszünk először egy

```
+ 77 7777 7777
+ 20 0115 0041
```

alaku u.n. vezérlő kombinációt; ennek hatására a beinditórutin a szalagról leolvasott további szavakat a 0041, 0042, ... rekeszekben fogja elhelyezni. A szalagra ezután ábécé sorrendben felvisszük a relativ cimezés alapjainak számértékeit: minden egyes értéket külön szóban helyezünk el a szó 7-18 bitjében /vagyis az első cim helyén/, a szó

többi bitje zérus lesz. A szó végét jelző "irás" jel után négy zérust, majd ismét egy "irás" jelet perforálunk.

A relativ cimezés alapjainak számértékei után a szalagra egy

```
+ 77 7777 7777
+ 20 0115 m
```

alaku ujabb "vezérlőkombinációt" viszünk fel, ahol m a szabványos szubrutin első utasitásának cime. Ennek hatására a beinditórutin a szabványos szubrutin utasitásait és konstansait az m cimü rekesztől kezdve fogja elhelyezni. Ezt követi egy

```
+ 77 7777 7777
- 77 0000 0000
```

alaku vezérlőkombináció, amelynek leolvasása után a beinditórutin megállitja a gépet. A szabványos szubrutin perforált szalagjának behelyezése után a gépet ujrainditjuk. Ekkor a beinditórutin sorra leolvassa a szalagról a szabványos szubrutin utasitásait /konstansait/; minden egyes utasitásban /konstansban/ a relativ cimeket átalakitja abszolut cimekké, majd az utasitást /konstansot/ elhelyezi az m+r cimü rekeszben, ahol r az illető utasitás /konstans/ relativ cimét jelöli. A szabványos szubrutinok végére szintén egy

```
+ 77 7777 7777
- 77 0000 0000
```

alaku vezérlőkombinációt kell rávinni, ennek hatására a beinditórutin a szubrutin bevitele után megállitja a gépet.

A beinditórutinnal nem lehet a gépbe vinni a ± 77 7777 7777 konstansokat /ezeket egyébként is célszerü szabványos konstansokként állandóan a gépben tárolni/.

A beinditórutin 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      U1    24   0062     0101
0071      /                      /
0072      A,    16   0104     0071
0073      ↓+    20   0106     0077
0074      A,    16   0105     0071
0075      ↓:,   32   0113     --
0076      ↓+    20   0107     0100
0077      /                      /
0100      /                      /
0101      /                      /
0102      +     00   0114     0101
0103      U2    74   --       0062
```

A felirt beinditórutin elvben minden relativ cimezésü szubrutin bevitelére alkalmas; a relativ cimezés alapjainak számát csupán memória-takarékossági okokból maximál-

tuk 15-re. Hátránya azonban, hogy minden egyes címnél külön fel kell tüntetni a relativ címezés alapját, ami megnehezíti a perforálást, és meghosszabbítja a bevitel idejét. A továbbiakban ismertetni fogunk egy másik beinditórutint, amely csak olyan címezésü szubrutinok bevitelére alkalmas, ahol a relativ címeknek egy közös alapjuk van: a szubrutin első utasitásának címe. Ez a rutin azonban az abszolut címekkel való bevitelhez képest nem hosszabbitja meg lényegesen a bevitel idejét. Az alábbiakban közölt beinditórutin, amelyet némileg egyszerüsitett formában fogunk felirni, Pekingben készült az ottani M-3 gépre.

A beinditórutin azon a tényen alapszik, hogy mig a jelenlegi M-3 gépen egy memóriarekesz címe maximálisan 4000 lehet /nyolcas rendszerben/, a szavak első, illetve második címrészének hosszusága 12-12 bit, tehát minden utasitást ábrázoló szóban címrészenként egy-egy bit /t.i. a hetedik, illetve tizenkilencedik bit/ feltétlenül zérus.

A relativ címeket az abszolut címektől ugy különböztetjük meg, hogy ha egy utasitás első címe relativ, akkor a hetedik bit, ha a második cim relativ, akkor a tizenkilencedik bit helyére 1-et irunk, vagyis minden relativ címet 4000-el növelünk. Ha a beinditórutin egy utasitásban 4000-nél nagyobb címet talál, akkor "tudni fogja", hogy az illető cim relativ, és 4000-m levonásával /m-mel jelöljük annak a memóriarekesznek a címét, amelybe a szubrutin első utasitását akarjuk bevinni/ kialakitja a megfelelő abszolut címet. Mindez természetesen csak az utasitásokra és az utasitásjellegü konstansokra vonatkozik, a számkonstansokat, amelyeket a beinditórutin használata esetén mindig az utasitások és utasitásjellegü konstansok címei után következő címekre viszünk be, a beinditórutin minden változtatás nélkül elhelyezi a megfelelő munkarekeszekben.

A szabványos szubrutinok szalagjának elejére mindig egy + 00 p d alaku szót kell lyukasztani, ahol d a szabványos szubrutin utasitásainak és konstansainak együttes számát, p az utasitások és utasitás-jellegü konstansok együttes számát jelöli.

A beinditórutin használata igen egyszerü: a 0101 cimü rekeszbe beirunk egy + 00 0000 m alaku szót, ahol m annak a rekesznek a cime, amelybe a szubrutin első utasitását akarjuk bevinni, a gépadóban elhelyezzük a szubrutin szalagját, majd a 0040 cimü utasitással elkezdjük a beinditórutin végrehajtását.

A beinditórutin 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 4000;
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	FU	34	0063	0067
0067		+77	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.

Emlitettük azonban, hogy a transzformációs módszereken kivül alkalmazhatjuk még a lebegőpontos programozás módszerét, s arról is szó volt, hogy ez lényegében nem más, mint egy lebegőpontos gép müködésének egy program segitségével történő utánozása. Ugyanis a komplex számokkal való müveletek elvégzésére is elkészithetünk egy programot, mondhatnók ugy is, hogy beprogramozzuk egy olyan feltételezett gép müködését, amely komplex számokkal is számol.

Ezeket a programokat célszerü szubrutinokként használni, segitségükkel elérhető, hogy a fixpontos M-3 géppel elvégezhetők legyenek olyan müveletek is, amelyeket a szerkezetileg bonyolultabb lebegőpontos /illetve a feltételezett komplex számokkal dolgozó gépek végeznek el. Ebben az értelemben tehát logikai /technikai/ lehetőségek programok segitségével történő pótlásáról van szó.

9.1. Lebegőpontos számolás az M-3 gépen.

Ha egy lebegőpontos gép működését akarjuk utánozni, lényegében a gép két egységének, az aritmetikai egységnek és a vezérlőegységnek a működését kell programozni. Nevezzük 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 szám kettes számrendszerbeli lebegőpontos /féllogaritmikus/ normalizált alakján az

$$x = d.2^p \text{ alakot értjük,}$$

ahol

$$d = \pm / \alpha_1 2^{-1} + \ldots + \alpha_{30} 2^{-30} / \text{ a mantissza}$$

$$p.2^{-30} = \pm / \beta_1 2^{-1} + \ldots + \beta_{30} 2^{-30} / \text{ a kitevő } 2^{-30} \text{ szorosa}$$

és

$$\alpha_1 = 1 \quad \text{azaz} \quad \frac{1}{2} \leq d < 1.$$

$$\text{Ha } \alpha_1 = 0, \text{ akkor } x = 0, \quad /\text{gépi } 0/$$

Tároljuk az igy felirt számot a memória két rekeszében; megállapodásszerüen a mantisszát egy páros, a kitevőt $/2^{-30}$-as léptékfaktorral megszorozva/ a rákövetkező páratlan cimü rekeszben, s jelöljük ezt az /a; a+1/ = x alakkal, ami tehát azt jelenti, hogy /a/ = d, /a+1/ = $p.2^{-30}$, ha $x = d.2^p$.

Jelöljük ki a memóriában a P, P+1, Q_0 Q+1 rekeszeket. Ezek töltsék be a regiszterek szerepét, mégpedig ugy, hogy

Q; Q+1 feleljen meg az A regiszternek, P; P+1 pedig a B regiszternek /azaz minden müvelet 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 müvelet után átadjuk a vezérlést a normalizáló szubrutinnak.

a./ Normalizáló szubrutin.

Ha két normalizált számmal végzünk valamilyen aritmetikai müveletet az eredmény általában nem lesz normalizált. El kell készitenI tehát egy normalizáló szubrutint, azaz egy olyan szubrutint, amely egy $x = d.2^p$; $d < \frac{1}{2}$ számot olyan alakra hoz, hogy $\frac{1}{2} \leq d < 1$ legyen.

A normalizáló szubrutint ugy készitjük 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 utasitás helye/		

b./ A lebegőpontos szorzás /z = xy/.

Ha /P; P+1/ = x = $a.2^p$ és /Q; Q+1/ = y = $b.2^q$, akkor

z = xy = $a.2^p$ $b.2^q$ = $ab.2^{p+q}$ = $c2^r$ ⟶ /P; P+1/

ahol c = ab r = p+q.

Tehát a szorzást ugy végezzük el, hogy a mantisszákat összeszorozzuk s a kitevőket összeadjuk:

/P/./Q/ ⟶ P

/P+1/ + /Q+1/ ⟶ P+1

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 = $a.2^p$ és /Q; Q+1/ = y = $b.2^q$, akkor

$z = \frac{x}{y} = \frac{a.2^p}{b.2^q} = \frac{a}{b} \cdot 2^{p-q} = c.2^r$ ⟶ /P; P+1/

ahol $c = \frac{a}{b}$ r = p-q.

Minthogy $\frac{1}{2} \leq a < 1$ és $\frac{1}{2} \leq b < 1$, ezért előfordulhat, hogy

$$\left|\frac{a}{b}\right| > 1$$

Ha viszont x = $a.2^p$-t x = $\frac{a}{2}2^{p+1}$ = $a'2^{p+1}$ alakra hozzuk, akkor nyilván

$$\left|\frac{a'}{b}\right| < 1.$$

Tehát az osztást

$z = \frac{x}{y} = \frac{a'}{b}2^{p+1-q}$ ⟶ /P; P+1/

alapján végezzük el, azaz

$$\frac{\frac{1}{2}\ /P/}{/Q/} \longrightarrow P : /P+1/ + 1 - /Q+1/ \longrightarrow P+1$$

Program:

t+13	-,	11	Q+1	P+1
t+14	↓+	20	$\langle 2^{-30} \rangle$	P+1
t+15	x,	13	$\langle \frac{1}{2} \rangle$	P
t+16	↓:	22	Q	P
t+17	U2	74	-	t normalizálás

d./ Lebegőpontos összeadás és kivonás.

Legyen /P; P+1/ = x = $a.2^p$ és /Q; Q+1/ = y = $b.2^q$. Ekkor az összeadást /illetve a kivonást/ a

$$z = x \pm y = a.2^p \pm b.2^q = c.2^r \longrightarrow /P; P+1/$$

képlet alapján célszerü elvégezni, ahol r = max /p,q/ + 1 és

$$c = \begin{cases} \frac{a}{2} \pm \frac{b'}{2} \text{ ha } p > q, \text{ ahol } b' = b.2^{-/p-q/} \\ \frac{a'}{2} \pm \frac{b}{2} \text{ ha } p < q, \text{ ahol } a' = a.2^{-/q-p/} \end{cases}$$

A mantisszák felezésére, s ennek megfelelően a közös kitevő eggyel való növelésére a tulcsordulás 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/ ugy 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 $\geq$ 30, akkor a nagyobb kitevőjü számot tekintjük eredménynek.

A kivonást ugy végezzük el, hogy a kivonandó negativját adjuk hozzá a kisebbitendőhöz.

A blokkdiagram a XVII. táblán látható.

Program:

t+20	x	03	$\langle 2^{-30}-1 \rangle$	Q	
t+21	|-1,	51	$\langle 2^{-30} \rangle$	Q	
t+22	FU	34	t+7	t+23	
t+23	|-1,	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	|-1,	51	$\langle 30.2^{-30} \rangle$	Q+1	
t+37	FU	34	t+40	t+7	
t+40	↓+	20	$\langle 30.2^{-30} \rangle$	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	?
t+47	..U2	74	-	τ

Egy lebegőpontos művelet ezekután a következőképpen végezhető el:

A műveletben szereplő két számot elhelyezzük a két kettős "regiszterben"; a normalizáló szubrutin kiugrató utasitásainak a helyére beirjuk a visszaugrató utasitást, majd átadjuk a vezérlést a lebegőpontos szubrutin megfelelő címére. Ha az eredményt meg kell őriznünk, akkor /P; P+1/-et átvisszük abba a két rekeszbe, amelyben az eredményt tárolni akarjuk. Ezek a lépések fixpontos utasitásokkal programozhatók.

Az előbbieknek megfelelően programozott lebegőpontos müveletekből álló programot lebegőpontos programnak nevezzük.

Lássunk egy példát: kiszámitandó 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+12/ = + 64 0000 0002

Legyen a normalizáló szubrutin r-től n+7-ig elhelyezve. Kezdődjön az összeadás programja az ö utasitással, a kivonásé k-val, az osztásé o-val, a szorzásé sz-el.

Program:

t	x,	13	r+6	r+10	Elvégezzük
t+1	Ul	24	t+2	P	a dx szorzást
t+2	+,	10	r+7	r+11	és normalizá-
t+3	Ul	24	t+4	P+1	lunk
t+4	+,	10	r+12	t+4	
t+5	Ul	24	n	n+7	
t+6	Á	05	r	Q	dx + a képzése.
t+7	Á	05	r+1	Q+1	
t+10	+,	10	r+12	t+10	
t+11	Ul	24	ö	n+7	
t+12	Á	05	P	m	átviszi dx+a-t
t+13	Á	05	P+1	m+1	az m, m+1 re-
t+14	Á	05	r	P	keszckbe
t+15	Á	05	r+1	P+1	
t+16	Á	05	r+2	Q	
t+17	Á	05	r+3	Q+1	$\frac{a}{b}$ képzése
t+20	+,	10	r+12	t+20	
t+21	Ul	24	o	n+7	
t+22	Á	05	P	m+2	$\frac{a}{b} \Rightarrow$ m+2, m+3
t+23	Á	05	P+1	m+3	
t+24	x,	13	r+4	r+6	
t+25	Ul	24	t+26	P	
t+26	+,	10	r+5	r+7	
t+27	Ul	24	t+30	P+1	cx képzése
t+30	+,	10	r+12	t+30	
t+31	Ul	24	n	n+7	
t+32	Á	05	m+2	Q	
t+33	Á	05	m+3	Q+1	cx + $\frac{a}{b}$ képzése
t+34	+,	10	r+12	t+34	
t+35	Ul	24	ö	n+7	

```
t+36   Á    05   m     Q
t+47   Á    05   m+1   Q+1
t+40   +,   10   r+12  t+40
t+41   Ul   24   o     n+7
t+42   /megállás          /
```

$\frac{cx + \frac{a}{b}}{dx \div a}$ képzése

Látható, hogy a lebegőpontos alakban megadott számokkal végzett aritmetikai müveletek végrehajtási ideje lényegesen nagyobb a fixpontos müveletekénél. A kiinduló adatok tárolására szolgáló rekeszek, és a munkarekeszek száma kétszer akkora, mint a fixpontos számitásnál, ehhez jönnek még azok a rekeszek, amelyeket a müveleti előirásoknak megfelelő lebegőpontos szubrutinok kötnek le.

9.2. A komplex aritmetika programozása.

Az alábbiakban olyan szubrutint készitünk, amely komplex számokon végez aritmetikai müveleteket.

A komplex számok tárolására a lebegőpontos számok esetéhez hasonlóan két rekeszre van szükség. Megállapodásszerüen a komplex számok valós részét egy páros, képzetes részét a rákövetkező páratlan cimü rekeszben tároljuk.

Jelöljük ki itt is a /P; P+1/, /Q; Q+1/ rekeszeket, B és A regiszterként.

Természetesen a tulcsordulásra továbbra is ügyelni kell; a következőkben feltesszük, hogy a szereplő adatok abszolut 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/ $= z_1 = a_1 + b_1 i$

/Q; Q+1/ $= z_2 = a_2 + b_2 i$

és

$$z_1 + z_2 = /a_1 + a_2/ + /b_1 + b_2/i \longrightarrow P;\ P+1$$

Ha

$$|a_1| < \tfrac{1}{2};\ |a_2| < \tfrac{1}{2};\ |b_1| < \tfrac{1}{2};\ |b_2| < \tfrac{1}{2},\ \text{akkor}$$

$$|a_1 + a_2| < 1,\quad |b_1 + b_2| < 1,$$

Program:

t	+	00	Q	P
t+1	+	00	Q+1	P+1
t+2	U1	74	-	t+33

b/ Kivonás.

Legyen /P; P+1/ $= z_1 = a_1 + b_1 i$

/Q; Q+1/ $= z_2 = a_2 + b_2 i$

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	-	01	Q+1	P+1
t+5	U1	74	-	t+33

c/ Szorzás.

Legyen $/P; P+1/ = z_1 = a_1 + b_1 i$

$/Q; Q+1/ = z_2 = a_2 + b_2 i$

és

$$z_1 z_2 = /a_1 + b_1 i/ \cdot /a_2 + b_2 i/ =$$

$$= /a_1 \cdot a_2 - b_1 \cdot b_2/ + /a_1 \cdot b_2 + a_2 \cdot b_1/ i =$$

$$= a + bi \longrightarrow (P;\ P+1)$$

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+33	

d/ Osztás.

Legyen $/P; P+1/ = z_1 = a_1 + b_1 i$

$/Q; Q+1/ = z_2 = a_2 + b_2 i$

és

$$\frac{z_1}{z_2} = \frac{a_1 + b_1 i}{a_2 + b_2 i} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + \frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2} \cdot i =$$

$$= a + bi \longrightarrow /P; P+1/$$

Ha $|z_1| < |z_2| < 1$, akkor nyilván $|a| < 1$, $|b| < 1$.

Ugyanis /a számok trigonometrikus alakjával számolva/

$$\frac{z_1}{z_2} = \frac{|z_1|}{|z_2|} \left(\cos /\varphi_1 - \varphi_2/ + i \sin /\varphi_1 - \varphi_2/\right)$$

tehát

$$\left| \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} \right| = \left| \frac{|z_1|}{|z_2|} \cdot \cos /\varphi_1 - \varphi_2/ \right| < 1$$

$$\left| \frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2} \right| = \left| \frac{|z_1|}{|z_2|} i \sin /\varphi_1 - \varphi_2/ \right| < 1$$

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 utasitás helye/			

Ha komplex számokkal végzünk el valamilyen müveletet, akkor ugyanugy kell eljárni, mint lebegőpontos számolás esetén: a két számot el kell helyezni a két kettős regiszterben, s a visszaugrató utasitás elhelyezése után átadni a vezérlést a megfelelő müveletet végző szubrutinra. Természetesen abszolutértékkel történő müveleteknél az adott komplex számok abszolutértékeit külön programmal képezni kell. Valós számok esetén a képzetes rész helyére 0-t kell irni.

9.3. Értelmező szubrutin - lebegőpontos programhoz.

Az egyes problémák megoldását reprezentáló program egészen más felépitésü, ha a feladatban szereplő adatokat lebegőpontos alakban ábrázoljuk, mint a fixpontos ábrázolásnak megfelelő program. Az előbbi mindig lényegesen több utasitásból áll. Nagyon célszerünek látszik tehát egy olyan szubrutin szerkesztése, amely egy "fixpontos" program utasitásait megfelelően transzformálja és igy lebegőpontos programmá alakitja. /Ugyanez a feladat komplex számokkal kapcsolatban is felmerülhet./

Egy olyan szubrutinra van tehát szükség, amely az eredeti program utasitásait "értelmezi", azaz a számokat, amelyekkel a müveletet el kell végeznünk, a lebegőpontos szubrutin két kettős rekeszébe helyezi el alkalmasan transzformálva őket, majd behivja az értelmezett müveleti utasitásnak megfelelő lebegőpontos müvelet szubrutinját, végül elhelyezi az eredményt. A szubrutinnak tehát egy lebegőpontos gép vezérlőegységét kell imitálnia.

Ismeretes, hogy egy adott utasitás lényegében kódolt algoritmus /elemi müveletek sorozata/ alakjában táplál be információt a gépbe; ennek alapján hajtják végre a szerkezeti elemek a szükséges müveleteket a már ismertetett 8-as taktusokban. Egy utasitáshoz, mint információhoz hozzárendelhető azonban egy máskép programozott algoritmus is. Az értelmező szubrutinok valamely utasitás alaku kódolt információhoz egy meghatározott algoritmust /müveletsort/ rendelnek hozzá, s ezt az algoritmust hajtják végre.

Szerkesszünk meg egy olyan értelmező szubrutint, amely valamely fixpontos utasitáshoz a megfelelő - ekvivalens - lebegőpontos algoritmust rendeli hozzá, s azt végrehajtja.

Egy adott fixpontos program utasitásai a következő három csoportba sorolhatók:

1. Vannak olyan utasitások, amelyek adatokon végeznek müveleteket; nevezzük ezeket adat-utasitásoknak.
2. Nevezzük cim-utasitásoknak azokat, amelyek utasitásokkal végeznek müveleteket /pl.: cimmódositás, utasitásbeültetés, stb./ és ezeken kivül a megállási és kinyomtató utasitásokat.
3. Végül vannak vezérlés és adat-átadó utasitások is.

Lássuk el - megállapodásszerüen - a cimutasitásokat negativ előjellel!

Fentiek szerint egy adott program utasitásai a következőképpen különböztethetők meg:

1. Ha egy utasitás negativ előjelü, akkor cimutasitásról van szó, tehát változatlanul - fixpontosan - hajtandó végre. Állapodjunk meg abban, hogy a logikai szorzást csak cimutasitásként használjuk.

2. Ha az utasitás pozitiv előjelü, és müveleti kódjának második jegye 0,1,2,3, akkor adat-utasitás.

3. Ha az utasitás pozitiv előjelü ós müveleti kódjának második jegye 4 vagy 5, akkor vezérlés, vagy adat-átadásról van szó.

Megállapodunk abban is, hogy az egyes fixpontos adatutasitásokba a müveletekben szereplő lebegőpontos alakban adott számok mantisszáinak cimét kell beirni. Egy-egy ilyen fixpontos utasitás tehát: m a' b' alaku, ahol m a müveleti kód, a' az egyik, b' a másik mantissza cime.

Ha egy eredetileg fixpontos alakban használt programot akarunk értelmezni, akkor az értelmezés előtt át kell ezt ugy alakitani, hogy az eredetileg fixpontos alakban tárolt adatok cimei helyett a nekik megfelelő lebegőpontos cimek /mégpedig a mantisszák cimei/ szerepeljenek a programban. Erre a célra külön gépi program készithető; erről a későbbiekben lesz szó.

Maga az értelmezés jelen esetben abban áll, hogy minden pozitiv előjelü utasitáshoz hozzárendeljük a benne szereplő kódjelnek megfelelő művelet lebegőpontos programját. Adatutasitásnál ez a program abból áll, hogy a szóbanforgó utasitás első ciménck tartalmát /a hozzátartozó kitevővel együtt/ átvisszük a lebegőpontos "A regiszterbe": Q; Q+1-be; a második cim tartalmát pedig a lebegőpontos "B regiszterbe": P; P+1-be. /Az m a' b' utasitás ugyan csak a mantisszák rekeszeinek cimeit tartalmazza; a hozzájuk tartozó kitevőkről viszont tudjuk, hogy az a'+1; b'+1 rekeszekben vannak./ Miután a "regiszterekbe" beirtuk a müveletben szereplő két számot, a vezérlést átadjuk a lebegőpontos szubrutin megfelelő részére /végrehajtjuk a megfelelő müveletet/, s ha a müvelet "vesszőtlen", akkor az eredményt beirjuk a második cimre.

Vezérlés-, illetve adatátadó utasitásnál átadjuk a vezérlést illetve átvisszük az adatokat a megfelelő cimekre. /Itt az átvitel kettős müvelet, hiszen a mantisszát és a kitevőt egyaránt át kell vinni./

A negativ előjelü utasitásokat 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/ = 5.2^{-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+12/	=		2^{-18}			

Paraméter:

/a+2/	=	Á	05	u	a

Program:

a		/..	..	....	..../
a+1		+	00	k+13	a+2
a+2	↓	/Á	05	u	a/
a+3		FU	34	a	a+4
a+4		Á	05	k	n+i
a+5		/\,	16	k+1	a

/A program a a+2-nél kezdődik!/

n+i a normalizáló szubrutin kiugró utasitásának helye; u az értelmezendő program első utasitásának cime!

a+6	↓ x,	33	k+14	-
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+4	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	k+10	a
a+33	↓ +	20	k+11	a+35
a+34	↓ +	20	k+7	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	∧ ,	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	∧ ,	16	k+5	a
a+76	↓: ,	32	k+17	
a+77	↓ +	20	a+72	a+2
a+100	U2	74	-	a+2
a+101	∧ ,	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 utasitásban ö az összeadó, k a kivonó, o az osztó, sz a szorzó lebegőpontos szubrutin első utasitásának cime.

Az értelmező szubrutinokból való kiugrás negativ előjelü vezérlésátadó utasitás segitségével történik. /A pozitiv előjelü vezérlésátadó utasitásokat ugyanis a szubrutin értelmezné, ezek a vezérlést további értelmezendő utasitásokra adnák át./

Az értelmezendő fixpontos programok elkészitésénél ügyelni kell arra, hogy a negativ utasitások cimeit ugy módositsuk, hogy eközben szemelőtt tartsuk: a gép ezeket az utasitásokat negativ számokként kezeli.

Látható, hogy egy utasitás értelmezése és lebegőponttal való végrehajtása átlagosan 60-70 utasitás végrehajtását teszi szükségessé. Természetes tehát az a törekvés, hogy lehetőleg inkább a transzformációs módszereket használjuk a fixponttal kapcsolatos nehézség megkerülésére, jóllehet lebegőpontos programozásnál nincsen szükség a tulcsordulás vizsgálatára és a transzformációkra.

Az értelmező szubrutinokkal kapcsolatban felhivjuk a figyelmet arra, hogy ezek "magasabbrendü" programoknak tekinthetők, amelyeknél a kiinduló információ maga is program /az értelmezendő program/.

Kézenfekvő ezzel kapcsolatban felvetni azt a gondolatot, hogy elkészithetők bizonyos u.n. szimbolikus programok is: a gép utasitásrendszerétől függetlenül megállapodásszerüen kódolt algoritmusok. Ezek értelmező szubrutin segitségével a gép utasitásrendszerének megfelelően végrehajthatók. El lehet készitení tehát több gép számára közös szimbolikus programokat is, s külön értelmező szubrutinokkal desiffrirozni és végrehajtani azokat.

9.4. Értelmező szubrutin - a komplex aritmetikához.

A komplex számok aritmetikája éppen ugy kétparaméteres müveletekből épül fel, mint a lebegőpontos számok aritmetikája. Ha tehát egy fixpontos programhoz egy komplex számokkal müködő algoritmust rendelünk hozzá, azaz ha az adott müveletsorozatot komplex számokon kell végrehajtani, akkor a lebegőpontos számitás esetéhez hasonlóan el lehet készíteni egy értelmező szubrutint. Ez a szubrutin nagyrészt megegyező a lebegőpontos szubrutinnal.

A nem abszolut-értékes müveleteknél csak annyi a különbség, hogy az a+104, a+105, a+106, a+107 rekeszekbe olyan utasitásokat helyezünk el, amelyek a komplex aritmetikai szubrutin megfelelő részére adják át a vezérlést.

Tehát

/a+104/ = U2	74	-	komplex összeadás	szubrutinjának kezdő	cime
/a+105/ = U2	74	-	"	kivonás	" "
/a+106/ = U2	74	-	"	osztás	" "
/a+107/ = U2	74	-	"	szorzás	" "

Állapodjunk meg abban, hogy ha egy nem abszolut-értékes müvelet után következik feltételes ugrás, akkor ez a valós rész előjelétől függően adja át a vezérlést.

Minthogy komplex számok esetén az abazolut-érték képzése nem ugy történik, mint lebegőpontos számok esetén, ezért az abszolutérték-képző blokkot ki kell cserélnünk.

A komplex értelmező szubrutin tehát /csak a lebegőpontos szubrutintól különböző részt irjuk le, az egyező részt pusztán jelöljük/ ilyen alaku:

a . . . a+40	Megegyezik a lebegőpontos értelmező szubrutinnal				
a+41	U2	74	-	a+110	
a+42		64	0000	0002	konstansok
a+43		00	Q	P	
a+44					szabad rekesz
a+45 . . . a+103	Megegyezik				
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 utasitása g a be-
a+114	U1	24	gy	gy+i	menete gy+i a kiugrató
a+115	Á	05	e	Q	utasitás helye /e/ = $\sqrt{a}$
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	∧,	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 abszolut-értékes művelet után következik feltételes ugrás, akkor e program attól függően adja át a vezérlést az első vagy a második cimre, hogy az előző - valós számokkal /t.i. a megfelelő komplex számok valós részével/ végzett - művelet eredménye negativ, vagy pozitiv-e.

9.5. Konverziós szubrutinok.

Az előzőkben megjegyeztük, hogy az értelmező szubrutinok a szó bizonyos értelmében "magasabbrendü" 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 utasitásnak megfelelő valóságos utasitáscsoportok/ nem állitunk össze egy teljes programot, hanem a részprogramot mindjárt végre is hajtjuk.

Elkészithető azonban olyan program is, amely miután értelmez egy utasitást nem hajtja azt végre, hanem tárolja a kialakitott utasitáscsoportot, s ezekből az utasitáscsoportokból egy teljes programot állit össze. Azokat a programokat, amelyek egy bizonyos programból valamilyen más programot állitanak elő, konverziós szubrutinoknak hivjuk.

A konverziós szubrutinok tehát lényegében programtranszformációk: egy adott programhoz egy másik programot rendelnek hozzá. Egy program transzformálása utasitásainak transzformálásából áll.

A legegyszerűbb konverziós szubrutin egy átcimező szubrutin, amely egy adott program minden utasitásának első, illetve második ciméhez hozzáad egy s számot, ha ezek a cimek valamilyen y számnál nagyobbak. Az értelmező szubrutin tárgyalásánál megemlitettük, hogy mielőtt egy adott programot értelmeznénk, át kell alakitani a programot ugy, hogy abban a lebegőpontos adatok cimei szerepeljenek. Az ezt végrehajtó program a következőképpen müködik.

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-től v+k-1-ig terjedő rekeszek /itt k=2i/.

A cimutasitásokat /negativ előjelüek/ nem kell módositani, hiszen azok adatcimeket nem tartalmaznak.

A pozitiv előjelü utasitások közül /ezek nem cimutasitások/ át kell cimezni minden adatutasitást, ezeken kivül még az U1 24 vezérlésátadó utasitás második cimét is /ez ugyanis adatcim/.

Program:

t	/			/	
t+1	+	00	$\langle 2^{-18} \rangle$	t+2	
t+2	Á	05	w	t	A t+2 utasitással kezdjük a program végrehajtását.
t+3	FU	34	t+1	t+4	
t+4	∧,	16	t+30	t	
t+5	↓-	21	t+36	m	
t+6	FU	34	t+7	t+20	
t+7	,	11	t+31	t	
t+10	FU	34	t+11	t+1	
t+11	-,	11	t+36	t	
t+12	↓:,	32	$\langle \frac{1}{2} \rangle$	-	

t+13	↓+,	30	t+37	-
t+14	↕Λ	26	t+32	m
t+15	Λ,	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	Λ,	16	t+33	t+2
t+23	↓x,	33	$\langle 2^{-12} \rangle$	-
t+24	↓+	20	t+34	t+27
t+25	Λ,	16	t+35	t
t+26	↓+,	30	m	-
t+27	/			/

Konstansok:

/t+30/	= +	00	7777	7777
/t+31/	= +	34	0000	0000
/t+32/	= +	00	0000	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 utasitásának cime.

Munkarekesz: m.

Megszerkeszthető egy olyan konverziós szubrutin is, amely egy skalármüveletekkel programozott algoritmushoz hozzárendeli a megfelelő vektormüveletekből 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 utasitásonként végigfutja a konvertálandó fixpontos programot s megvizsgálja, hogy egy kiválasztott utasitás adatutasitás, vezérlésátadó utasitás, vagy cimmódositó utasitás-e? Ha a tekintett program-elem nem adatutasitás lássuk el negativ előjellel. A szubrutin kialakitja az adatutasitások végleges formáját, s beülteti a konvertált program soronkövetkező helyére. Kialakitja ezenkivül a cimmódositó utasitásoknak megfelelő utasitáscsoportokat is, s ezeket is végleges formában helyükre teszi. A vezérlésátadó utasitások megfelelő képeit is előállitja, de nem végleges formában, hanem ugy, hogy azokban a fixpontos program cimei szerepeljenek. /Ezeket ugyanis - ha a vezérlést a program következő részére adják át - nem is tudnók kialakitani, hiszen nem tudjuk, hogy a vezérlést a konvertált program melyik utasitására kell átadni./A kialakitott, de fixpontos cimeket tartalmazó utasitáscsoportokat szintén elhelyezzük a konvertált program megfelelő helyére, de a negativ előjelet meghagyjuk.

A későbbiek folyamán szükség van arra az információra, hogy egy adott fixpontos utasitásnak megfelelő utasitáscsoport a konvertált programban hol helyezkedik el. Ezért az egyes utasitások transzformálása után, annak eredeti helyére beirjuk a 00 00rs w alaku információt, ahol r egy bizonyos segédinformáció, s a transzformált utasitáscsoport hossza, w pedig a szóbanforgó utasitáscsoport első utasitásának a cime a konvertált programban.

Az első blokk eredményeként tehát lényegében létrejön a konvertált program, de a vezérlésátadó utasitások még a fixpontos program cimeit tartalmazzák. A második blokk fel-

adata ezek helyett a cimek helyett beirni a konvertált program megfelelő cimeit, felhasználva a megtartott OO OOrs w információt.

Vizsgáljuk meg részletesebben a K szubrutinnak azt a blokkját, amely az adatutasitások képét szerkeszti meg. Ez a szubrutin önállóan is müködik, olyan programok konvertálására alkalmas, amelyekben nincsenek vezérlésátadó és átcimező utasitások.

Jelöljük ki vizsgáló rekesznek a t rekeszt, utasitásszámlálónak a q rekeszt.

Ismeretes, hogy ha egy □ a b alaku fixpontos utasitást lebegőpontosan akarunk végrehajtani, akkor helyette a következő utasitáscsoportot kell felirni. /Nevezzük ezt M-blokknak./

w	Á	05	b'	P	
w+1	Á	05	b'+1	P+1	
w+2	Á	05	a'	Q	/M-blokk/
w+3	Á	05	a'+1	Q+1	
w+4	+,	10	⟨6400000002⟩	w+4	
w+5	Ul	24	m'	n+i	
w+6	Á	05	P	b'	
w+7	Á	05	P+1	b'+1	

ahol m' az m-nek megfelelő lebegőpontos szubrutin első utasitásának cime, n+i a normalizáló szubrutin kiugrató utasitásának a helye, a', a'+1, b', b'+1 a müveletben szereplő számok lebegőpontos alakjának rekeszeit jelentik.

Ha a müvelet "vesszős" /első kódja 1-es/, akkor a w+6, w+7, ha nyilas /első kódja 2-es/, akkor a w w+1, ha nyilvesszős /első kódja 3-as/, akkor w, w+1, w+6, w+7 utasitásokra nincs szükség.

Tehát a K program a következőképpen jár el:

Kialakitja az M blokkot, s ha a müvelet vessző és nyil nélküli, akkor a w-től w+7-ig terjedő, ha vesszős, akkor a w-től w+5-ig terjedő, ha nyilas, akkor a w+2-től w+7-ig terjedő, ha nyilvesszős, akkor a w+2-től w+5-ig terjedő utasitásckat viszi át a konvertált program megfelelő helyére, s legyen az első és második esetben az r segédinformáció: r=4, a harmadik és negyedik esetben r=2,

Természetesen ügyelni kell arra, hogy a w+4 utasitásban a második cimen éppen az a cim szerepeljen, ahová ez az utasitás kerülni fog.

A program:

Konstansok:

/k/	=		00	7777	7777
/k+2/	=		07	0000	0000
/k+3/	=	U1	24	h	n+i
/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   0002
```

Munkarekeszek: m, m+1, s, r, q.

Paraméterek:

```
/p/   = 00  a    a
/p+1/ = 00  A    A
/p+2/ = 00  0000 w
/t+3/ = 05  u-1  t
```

a a fixpontos program kiinduló adatainak első rekesze.
A az a-nak megfelelő rekesz a lebegőpontos programban.
w a lebegőpontos program első utasitásának a helye.
u a fixpontos program első utasitásának 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
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	{			}

A program végrehajtását a t+1 utasitással kezdjük.

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+36
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,	33	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	/			/
t+121	/			/
t+122	/			/
t+123	/			/
t+124	/			/
t+125	/			/
t+126	/			/
t+127	/			/
t+130	/			/

M-blokk helye

Az itt tárgyalt konverziós szubrutinok csak azt akarták bemutatni, hogy a programozás munkájának egy részét át lehet háritani a gépre; csak a gép logikai lehetőségeitől és a mi gyakorlatunktól függ, hogy a programkészités munkájának milyen hányadát végeztetjük magával a géppel.

10. Fejezet.

GYAKORLATI PÉLDÁK.

Az alábbiakban néhány - a gyakorlatban sokszor előforduló - probléma részletes numerikus analizisét és erre támaszkodva a megoldás teljes programját mutatjuk be. Meg kell azonban emlitenünk, hogy a parciális differenciálegyenletekkel csak futólag foglalkozunk, jóllehet ilyen jellegü feladatok gyakran előfordulnak, numerikus szempontból pedig épp ezek megoldása jelenti a fő nehézséget. Mégsem célszerü ezekkel a helyütt részletesen foglalkozni, mert általánosságban semmit sem tudunk mondani még az alkalmazandó - legcélszerübb - 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-számu lineáris egyenletrendszerek megoldása. Igen sok gyakorlati számitási feladat vezethető vissza lineáris egyenletrendszerek megoldására, részint többé-kevésbbé közvetlenül /például egyes statikai és szilárdságtani számitások, a godéziai kiegyenlitő számitások, gazdasági téren különböző tervezési és árrendezési problémák stb./, részint közvetve, /például az áramlási profilok számitásai, rövid időre szóló időjárás-prognozisok készitése, 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

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + \dots + a_{nn}x_n &= b_n \end{aligned} \qquad /1/$$

n ismeretlenes lineáris egyenletrendszer x_i megoldása /i=1,...,n/ az

$$x_i = \frac{D_i}{D}$$

alakban irható fel, ahol D az egyenletrendszer determinánsa /itt és a továbbiakban is feltesszük, hogy $D \neq 0$/és

$$D_i = \begin{vmatrix} a_{11} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1n} \\ & & & & & & \\ a_{n1} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{nn} \end{vmatrix}$$

- általában nem alkalmazható a lineáris egyenletrendszerek /kézi vagy gépi/ numerikus megoldására. Ennek oka egyszerüen az, hogy a numerikus analizis nem ismer olyan eljárást, amelynek segitségével egy általános n-edrendü determináns értékét n^3-nél nagyságrendben kevesebb algebrai müvelettel ki lehetne számitani. A Cramer-szabály alkalmazásánál n+1 determináns értékét kell meghatározni, összesen tehát n^4-nél nagyságrendben nem kevesebb müveletre volna szükség;

ugyanekkor viszont a továbbiakban ismertetett Gauss-féle eliminációs eljárással egy n ismeretlenes lineáris egyenletrendszert kb. n^3 számu müvelettel meg lehet oldani.

Az u.n. direkt eljárások közül /amelyek véges számu 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 tulcsordulás 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

$$\begin{aligned} \alpha_{11}x_1 + \alpha_{12}x_2 + \dots + \alpha_{1n}x_n &= \beta_1 \\ \alpha_{22}x_2 + \dots + \alpha_{2n}x_n &= \beta_2 \\ &\vdots \\ \alpha_{nn}x_n &= \beta_n \end{aligned} \qquad /2/$$

"trianguláris" egyenletrendszert állit 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, ... , n/ és b_k /k = 1, ... , n/ számokat: az r, ... , r+n-1 cimü rekeszekben elhelyezzük az a_{11}, ... , a_{1n} együtthatókat, az r+n cimü rekeszben a b_1 állandót, az r+n+1, ... , r+2n rekeszekben az a_{21}, ... , a_{2n} együtthatókat, az r+2n+1 rekeszben a b_2 állandót, stb.

Az eljárás végén az s, ... , s+n-1 cimü rekeszekben rendre megkapjuk az x_1, ... , x_n ismeretleneket.

Az első eliminációs lépésben a gép a következő müveleteket hajtja végre:

a/ Megnézi, hogy az a_{ij} és b_k számok között van-e abszolut értékben $\frac{1}{2}$-nél nagyobb; ha igen, mindegyik egyenletet végigszorozza $\frac{1}{2}$-del.

b/ Kiválasztja az x_1 együtthatói közül a legnagyobb abszolut értéküt; ha az a k-adik egyenletben van, akkor a_{k1} lesz az un. <u>első főelem.</u>

c/ Az első egyenletet felcseréli a k-adik egyenlettel /az a_{11} elemet az r cimü rekeszből az r+/k-1/. /n+1/ cimü rekeszbe, az a_{k1} elemet az r+/k-1/. /n+1/ cimü rekesztől az r cimü rekeszbe viszi át, stb./

d/ Az igy kapott egyenletrendszer második egyenletéhez hozzáadja az első egyenlet - $\frac{a_{21}}{a_{11}}$ - szeresét /a felcseréléssel kapott egyenletrendszer együtthatóit az egyszerüség kedvéért ugyanugy 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 szükség, hogy a d/ blokkban végrehajtott összeadásoknál ne léphessen fel tulcsordulás. A főelem kiválasztásának és a megfelelő egyenletcserének kettős célja van: egyrészt a d/ blokkban a - $\frac{a_{i1}}{a_{11}}$ hányadosok képzésénél /i=2, ..., n/ elkerüljük a tulcsordulás veszélyét; másrészt, csökkentjük a szorzásoknál és osztásoknál fellépő kerekitési hibák befolyását /ez utóbbi állitásunk bizonyitására 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, ..., n-1/ a gép először megnézi, hogy az i-1-edik lépéssel kapott egyenletrendszer együtthatói és az egyenletrendszer jobboldalán álló állandók között van-e abszolut értékben $\frac{1}{2}$ -nél nagyobb; ha igen, az egész egyenletrendszert végigszorozza $\frac{1}{2}$ -del. Ezután megállapitja, hogy az i-edik, i+1-edik, ..., n-edik egyenlet közül melyikben legnagyobb az x_i ismeretlen együtthatójának abszolut értéke /a legnagyobb abszolut értékü 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 egyszerü: az n-edik egyenletből rögtön megkapjuk az x_n ismeretlen értékét, ezt az értéket az n-1-edik egyenletbe visszahelyettesitve megkapjuk x_{n-1}-et; x_n-et és x_{n-1}-et az n-2-edik egyenletbe visszahelyettesitve, megkapjuk x_{n-2}-t stb. Az ebben a részben végrehajtott összeadásoknál és osztásoknál tulcsordulás léphet fel. Ha a tulcsordulás az x_i ismeretlen értékének kiszámitása közben történt, akkor a gép ujrainditása után egy külön program segitségével megszorozzuk $\frac{1}{2}$ -del a már meghatározott $x_{i+1}, \ldots, x_n$ ismeretleneket, a $\beta_1, \ldots, \beta_i$ állandókat és ismét megpróbáljuk kiszámitani az x_i ismeretlent; ha megint tulcsordulás lép fel, akkor ujból szorzunk $\frac{1}{2}$ -el, stb. Végeredményben az $x_1, \ldots, x_n$ ismeretleneket egy 2^{-m} alaku lépésfaktorral szorozva kapjuk meg, ahol m a fellépett tulcsordulások száma.

A Gauss-féle eliminációs eljárás teljes programja elég hosszu, ezért itt most csak az eljárás egyszerüsitett változatát fogjuk programozni: az eliminációs lépéseknél a/, b/ és c/ blokkokat elhagyjuk, csak a d/ blokkot /a tulajdonképpeni eliminációt/ tartjuk meg. Ez az egyszerüsitett program a gyakorlatban abban az esetben használható, ha az /1/ egyenletrendszer együtthatói közül az a_{ii} együtthatók /i=1, ..., n/ dominálnak /nagyságrendben nagyobbak az összes többi együtthatónál/.

Az eliminációs rész egyszerüsitett programja a következő:

```
/k/   = /+13 k+1 r+n+1/; /k+1/ = - 77 7777 7777;
/k+2/ = /+2o r+n+2 r+n+2/; /k+3/ = + 00 0001 0001;
/k+4/ = + 00 0000 0001; /k+5/ = /x, 13  p  r+n+1/;
/k+6/ = /+ 00 0000 n/; /k+7/ = /+ 00 0001 0001/;
/k+1o/ = + 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+1o	↓↓-1,	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+1o	a+2
a+15	↓↓-1,	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+12	k
a+26	↓|-|,	71	k+11	-
a+27	FU	34	a	a+30

A trianguláris egyenletrendszer megoldásának programja:

/k+15/ = /+ 13 s+n r+n/n+1/-1/; /k+16/ = + 00 n+1 0000;
/k+17/ = + 00 0001 n+2; /k+20/ = + 24 a+43 s;
/k+21/ = /+ 13 s+n r+n/n+1/-1/;

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+32
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	k+21
a+45	-	01	k+14	a+41
a+46	-	01	k+10	k+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 egyszerü felépitésü /az elemek között igen sok zérus értékü lehet/, olyannyira, hogy valamelyik iterációs eljárás lényegesen gyorsabban eredményhez vezetne. Ezt tehát mindig meg kell vizsgálnunk!

10.2. Differenciálegyenletek numerikus integrálása.

Közönséges differenciálegyenletek numerikus integrálása - 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 célszerü adott esetben használni.

Az alábbiakban példaképp bemutatjuk, hogy hogyan lehet elsőrendü egyenletek esetében a Runge-Kutta ill. a differencia-módszert fixpontosan programozni.

10.2.1. A Runge-Kutta módszer elsőrendü egyenleteknél.

A Runge-Kutta módszer az egyik legjobban elterjedt numerikus integrálási módszer. Előnye, hogy eléggé pontos, s elég nagy lépésközzel lehet használni. Kézi számitásnál nagy hátránya viszont, hogy sok számolást igényel. Gépi számitásnál 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 $/x_i = x_{i-1}+h = x_0+i.h/$ pontban vett y_i közelitőértékét a Runge-Kutta módszer szerint az

$y_{i+1} = y_i + k$; /i = 0, 1, 2, ..., n-1; $x_o = a$; $x_n = b$/

képlet alapján számithatjuk ki, ahol a ill. b azon intervallum alsó ill. felső határpontja, ahol a megoldást keressük, és

$$k = \frac{1}{6} /k_1 + 2k_2 + 2k_3 + k_4/$$

$$k_1 = h\ f/x_i, y_i/$$

$$k_2 = h\ f/x_i + \frac{h}{2}; y_i + \frac{k_1}{2} /$$

$$k_3 = h\ f/x_i + \frac{h}{2}; y_i + \frac{k_2}{2} /$$

$$k_4 = h\ f/x_i + h; y_i + k_3 /$$

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 leirása megtalálható L.Collatz: Numerische Behandlung von Differentialgleichungen/Springer-Verlag,Berlin/könyvében az 59-67 oldalon.

Irjuk fel a feladat blokkdiagramját. Minthogy a programozással kapcsolatos irodalomban gyakran találkozhatunk a blokkdiagramoknak az eddig ismert módtól némileg eltérő felirásával, irjuk fel most a blokkdiagramot a következőképpen:

Nevezzük az egyes blokkokat operátoroknak és jelöljük nagybetükkel.

A Runge-Kutta módszer a következő operátorokból áll:

L - operátor: az x_i argumentumérték növelése $\frac{h}{2}$ -el,

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,

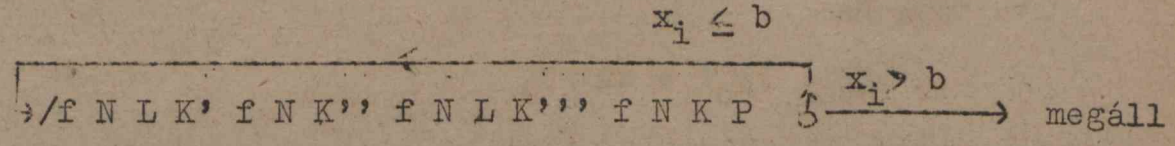
K operátor: k kiszámitása,

f operátor: az f függvény kiszámitása az adott helyen,

N operátor: $k_j = h\, f/x,y/$ kiszámitása /j=1,2,3,4/

P operátor: megvizsgálja, hogy $x_i \leqq b$ teljesül-e?

Igy a számitás menete az



blokkdiagram alapján programozható. Elkészitünk egy fixpontos programot; a tulcsordulásra természetesen ügyelni kell. Tegyük fel, hogy $|y'| = |f/x,y/| \leq K_0 < 1$, $|y| \leq K_1 < 1$ és $|x| < 1$ teljesül. K_0, K_1-et ugy választjuk meg, hogy a közbülső eredményekben se lépjen fel tulcsordulás.

A közbülső eredményekre a következő /durva/ becslés adható:

$$|k_j| \leq |h\, f/x,y/| < h \qquad /j=1,2,3,4/$$

és $|y_i + k_j| \leq |y_i| + |k_j| < |y_i| + h < 1-2^{-30}$

ha

$$|y_i| \leq 1-2^{-30} - h. \qquad /1/$$

A k-ra viszont

$$|k| = \left|\tfrac{1}{6}\,/k_1 + 2k_2 + 2k_3 + k_4/\right| \leq \tfrac{1}{6}\,6h = h \quad \left[h \leq \tfrac{1}{3} \text{ mert } n \geq 3\right]$$

Ha tehát $K_o = \frac{1}{2}$; $K_1 = \frac{1}{2}$, akkor biztosan nem lép fel tulcsordulás.

A tulcsordulás elkerülhető a következő feltételek teljesülése esetén is:

$$|y'| < 1, \quad |y_o| < 1-\delta \quad \text{és} \quad |x| \leq \delta < 1.$$

Ugyanis $|k| < h$ lévén

$|y_i - y_{i-1}| < h$; tehát a megoldás lépésenként legfeljebb h-val növekszik.

Az i-edik pontban vett közelitő értékre tehát:

$$|y_i| \leq |y_o| + ih.$$

Igy, ha

$$|y_o| + ih < 1,$$

azaz

$$|y_o| < 1 - ih,$$

akkor

$$|y_i| < 1.$$

Ha tehát $|y_o| < 1 - \delta \leq$ 1-ik /nh$\leq \delta$/, akkor $\max_{/i/} |y_i| < 1$

Ha a differenciálegyenlet nem teljesiti az előbbi feltételeket, s nem lehet ugy transzformálni, hogy teljesitse, akkor lebegőpontos programra van szükség.

Rekeszelosztás.

Az f/x,y/ függvény megfelelő értékeit külön segédszubrutin felhasználásával számitjuk. Legyen ennek a kezdete q; a visszaugrató utasitás helye: q + w és

$/\alpha/ = x$

$/\beta/ = y$

$/\delta/ = f/x,y/.$

Legyen a kinyomtató szubrutin az r-től r + j-ig terjedő rekeszben olhelyezve, és a kinyomtatandó adat helye legyen s.

Legyenek: δ_1; δ_2; δ_3; a; γ; m; m+1; munkarekeszek.

Program:

t	Á	05	$\langle x_o \rangle$	α
t+1	Á	05	$\langle y_o \rangle$	β
t+2	x,	13	$\langle \frac{1}{2} \rangle$	$\langle h \rangle$
t+3	U1	24	t+4	m $\leftarrow$ $\frac{h}{2}$
t+4	Á	05	$\langle 0 \rangle$	m+1
t+5	Á	05	β	γ
t+6	+,	10	t+44	t+6
t+7	U1	24	q	q+w
t+1o	x	03	m	δ
t+11	↓+	2o	m+1	m+1 $\leftarrow$ $\frac{k_1}{2}$

t+12	+	00	m	α
t+13	+	00	$\langle 6.2^{-30} \rangle$	q+w
t+14	+,	10	δ	γ
t+15	U1	24	q	β
t+16	x	03	$\langle h \rangle$	δ
t+17	↓+	2o	m+1	m+1 $\leftarrow\!\!-\!\!-$ $\frac{k_1}{2} + k_2$
t+2o	+	00	$\langle 6.2^{-30} \rangle$	q+w
t+21	x,	13	$\langle \frac{1}{2} \rangle$	δ
t+22	↓+,	3o	γ	-
t+23	U1	24	q	β
t+24	x	03	$\langle h \rangle$	δ
t+25	↓+	2o	m+1	m+1 $\leftarrow\!\!-\!\!-$ $\frac{k_1}{2} + k_2 + k_3$
t+26	+	00	m	α
t+27	+	00	$\langle 6.2^{-30} \rangle$	q+w
t+3o	+,	1o	δ	γ
t+31	U1	24	q	β
t+32	x,	13	n	δ
t+33	↓+,	3o	m+1	
t+34	↓x,	33	$\langle \frac{1}{3} \rangle$	
t+35	↓+	2o	γ	$\beta \leftarrow\!\!-\!\!-$ $y_i = y_{i-1} + k$
t+36	Á	05	β	δ
t+37	+,	1o	t+44	t+37
t+4o	U1	24	r	r+j
t+41	/-/,	51	α	$\langle b \rangle$
t+42	FU	34	t+43	t+4
t+43	megállási utasitás			
t+44		64	0000	0002

y_i kinyomtatása

Ez a program a számitás kezdetén megválasztott h lépésközzel müködik. Elkészithető 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

$$\left| \frac{k_2 - k_3}{k_1 - k_2} \right| < \varepsilon \ /\varepsilon = 0{,}01/,$$

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 kiszámitjuk az y_i értéket (ezt jelöljük $y_i^{/h/}$-val) majd felezzük a lépésközt, s kiszámitjuk a megoldás közelitő értékét az $x_{i-1} + \frac{h}{2}$ helyen; ennek segitségével mint intervallum-kezdőértékkel ismét kiszámitjuk ujra y_i-t; jelöljük ezt $y_i^{(\frac{h}{2})}$-lel.

Összehasonlitjuk a két lépésben számitott $y_i^{(\frac{h}{2})}$ értéket az egy lépésben nyert y_i értékkel, pontosabban képezzük különbségüket.

Ha $\left| y_i^{/h/} - y_i^{(\frac{h}{2})} \right| \leq \varepsilon$ ahol ε a közelités pontossága / $\varepsilon \approx 15 \frac{\Delta}{N}$, 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 számitást h lépésközzel folytatjuk tovább. Ha $\left| y_i^{/h/} - y_i^{(\frac{h}{2})} \right| > \varepsilon$, akkor $\frac{h}{2}$-et 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 segitségével állitják elő a megoldás közelitő értékeit, mégpedig a szóbanforgó pont körüli pontokban képezett differenciák segitségével. 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 számitani/ az y_1, y_2, y_3 értékeket.

Az y_{i-3}; y_{i-2}; y_{i-1}; y_i értékek segitségével az

$$y_{i+1}^{/1/} = y_{i-3} + / \frac{8h}{3} f_i - \frac{4h}{3} f_{i-1} + \frac{8h}{3} f_{i-2} / \qquad /1/$$

képlet alapján kiszámitjuk y_{i+1} első közelitését.

Ezzel az első közelitéssel kiszámitjuk f_{i+1}-et; s y_{i+1} második, harmadik, stb., ν-edik közelitését az

$$y_{i+1}^{/\nu/} = y_{i-1} + h\left(\frac{1}{3} f_{i+1}^{/\nu-1/} + \frac{4}{3} f_i + \frac{1}{3} f_{i-1}\right) \qquad /2/$$

képlettel számitjuk ki. Az iterációs eljárást a /2/ képlet alapján addig folytatjuk, amig

$$\left| y_{i+1}^{/\nu/} - y_{i+1}^{/\nu-1/} \right| < \varepsilon$$

adódik, ahol ε előre adott. /Lásd: Békéssy András: "Közönséges differenciálegyenletek numerikus integrálása" cimü jegyzet 27-28. oldal, Mérnöki Továbbképző Intézet 2706/.

Elkészitünk egy fixpontos programot a Milne módszerre:

Tegyük fel, hogy $|y| \leq K_o < 1$, $|y'| < 1$, $|x| < 1$. Ha $K_o = \frac{1}{2}$-et választunk, akkor a közbülső eredmények sem csordulnak tul.

Ugyanis:

$$\left| \frac{8}{3} h f_i - \frac{4}{3} h f_{i-1} + \frac{8}{3} h f_{i-2} \right| \leq \frac{20\,h}{3} < 7h$$

$$\left| y_{i+1}^{/1/} \right| \leq y_{i-3} + 7h;$$

ha tehát $|y_{i-3}| \leq \frac{1}{2} \leq 1 - 7h$ /ha $h \leq \frac{1}{14}$; $n \geq 14$/, akkor $\left| y_{i+1}^{/1/} \right| < 1$. Ugyanigy:

$$\left| y_{i+1}^{/\nu/} \right| \leq |y_{i-1}| + \left| \frac{h}{3} f_i^{/\nu-1/} \right| + \frac{5h}{3} \leq \frac{6h}{3} + |y_{i-1}| <$$

$$< 2h + |y_{i-1}| < 1,$$

ha

$$|y_{i-1}| < 1 - 2h.$$

Rekeszelosztás.

Legyen a hf/x,y/-t kiszámitó segédszubrutin az f-től f+k-ig terjedő rekeszekben elhelyezve, s legyen ebben a szubrutinban:

x cime α

y cime β

és hf/x,y/ cime δ .

Legyenek m_1; m_1+1; m_1+2; m_1+3; m, m+1, m+2, M, K munkarekeszek.

Irjuk fel a számitás menetét.

A számitásban előforduló operátorok a következők:

A_0 operátor: y_0; y_1; y_2; y_3; x_0 átvitele az m_1; m_1+1; m_1+2; m_1+3; α rekeszekbe.

F_0 operátor: h f/x_0 + h; y_1/ ==⇒ m; h f/x_0 + 2h; y_2/ ==⇒ m

h f/x_0 + 3h; y_3/ ==⇒ m+2; x_0 + 3h ==⇒ α

A_1 operátor: $y_{i+1}^{/1/}$ képzése:

azaz

$$\frac{8}{3}\,/m/ + \frac{8}{3}\,/m+2/ - \frac{4}{3}\,/m+1/ + /m_1/ \Longrightarrow \beta$$

A_2 operátor: $y_{i-1} + \frac{4}{3} h f_i + \frac{1}{3} h f_{i-1}$ képzése,

azaz

$$\frac{1}{3}\,/m+1/ + \frac{4}{3}\,/m+2/ + /m_1+2/ \Longrightarrow K$$

H_0 operátor: $/\alpha/ + h \Longrightarrow \alpha$ /azaz: az x argumentum növelése h-val/.

F_1 operátor: f/ $/\alpha/; /\beta/ \Longrightarrow \delta$

A_3 operátor: $\frac{1}{3} /\delta/ + K \Longrightarrow M$

P_0 [nem / igen] operátor: $|/M/ - /\beta/| < \varepsilon$ teljesül-e?

F_2 operátor: M átvitele β-ba.

A_4 operátor: $/m_1+1/ \Longrightarrow m_1$-be

$/m_1+2/ \Longrightarrow m_1+1$-be

$/m_1+3/ \Longrightarrow m_1+2$-be

$/M/ \Longrightarrow m_1+3$-ba

$/m+1/ \Longrightarrow m$-be

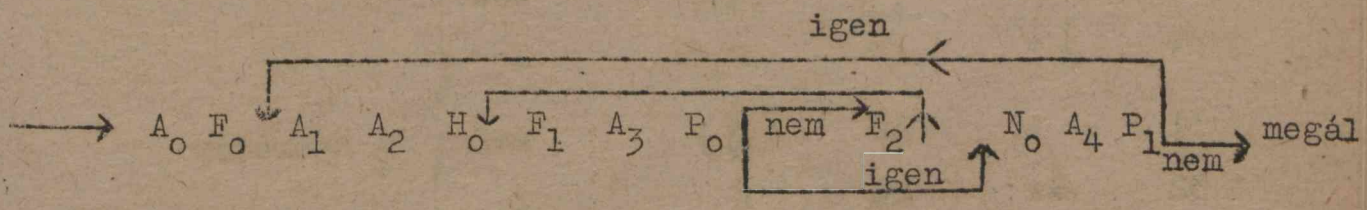
$/m+2/ \Longrightarrow m+1$-be

$/\delta/ \Longrightarrow m+2$-be

N_0 operátor: /M/ kinyomtatása

P_1 [igen / nem] operátor: $/\alpha/ \leq b$ teljesül-e?

A számitás menete:



A program a következő:

t	Á	05	$\langle y_0 \rangle$	m_1	
t+1	Á	05	$\langle y_1 \rangle$	m_1+1	
t+2	Á	05	$\langle y_2 \rangle$	m_1+2	
t+3	Á	05	$\langle y_3 \rangle$	m_1+3	A_0 operátor
t+4	Á	05	$\langle x_0 \rangle$	α	
t+5	Á	05	m_1+1	β	
t+6	Á	05	t+2o	f+k	
t+7	+	00	$\langle h \rangle$	α	
t+1o	U2	74	-	f	
t+11	Á	05	δ	m	
t+12	Á	05	m_1+2	β	
t+13	+	00	$\langle 4.2^{-3o} \rangle$	f+k	
t+14	U2	74	-	t+7	
t+15	Á	05	δ	m+1	F_0 operátor
t+16	Á	05	m_1+3	β	
t+17	U2	74	-	t+13	
t+2o	U2	74	-	t+11	
t+21	Á	05	δ	m+2	
t+22	+,	1o	m+2	m	
t+23	↓:	22	$\langle \frac{3}{8} \rangle$	K	
t+24	:,	12	$\langle -\frac{3}{4} \rangle$	m+1	A_1 operátor
t+25	↓+,	3o	K		
t+26	↓+	2o	m_1	β	
t+27	x,	13	$\langle \frac{1}{3} \rangle$	m+1	
t+3o	↓+	2o	m_1+2	K	
t+31	:,	12	$\langle \frac{3}{4} \rangle$	m+2	A_2 operátor
t+32	↓+	2o	K	K	
t+33	+	00	$\langle h \rangle$	α	H_0 operátor

t+34	+,	10	⟨64 0000 0002⟩	t+34	F_1 operátor
t+35	U1	24	f	f+k	
t+36	x,	13	$\langle\frac{1}{3}\rangle$	δ	A_3 operátor
t+37	↓+	20	K	M	
t+40	↓1-1,	71	β		P_0 operátor
t+41	↓1-1,	71	⟨ε⟩		
t+42	FU	34	t+45	t+43	
t+43	Á	05	M	β	F_2 operátor
t+44	U2	74	-	f	
t+45	Á	05	M	S	
t+46	+,	10	⟨64 0000 0002⟩	t+46	kinyomtatás
t+47	U1	24	r	r+i	/N_0/
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	⟨b⟩	P_1 operátor
t+57	FU	34	t+60	t+21	
t+60					

A differenciamódszereknek nagy hátrányuk, hogy a számitás elkezdéséhez m+1 kezdő értékre van szükség /m a legmagasabb felhasznált differencia fokszáma/, ezeket valamilyen más módszerrel előzőleg meg kell határozni.

10.2.3. Elsőrendü közönséges differenciálegyenletrendszerek.

Az elsőrendü közönséges differenciálegyenletrendszerek numerikus integrálása a közönséges elsőrendü differenciálegyenletek numerikus integrálására vonatkozó ismert módszerek bármelyikével elvégezhető, mégpedig ugy, 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 utasitásokat, amelyek segitségével a programozó által elkészitett - papiron számok formájában felirt - 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 segitségével lehetőségünk van kivülről 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 "sulyát" a nyolcas számrendszerben. Pl.: / ◉-el jelölve az égő lámpát - illetve a benyomott gombot, ○-el a nem égő lámpát, illetve a nem benyomott gombot/. - A

4	2	1	4	2	1	4	2	1	4	2	1
○	◉	◉	◉	○	◉	◉	◉	○	○	○	◉

állapot a 3561 számnak felel meg.

A vezérlő pult nyomógombjai és kapcsolói szerepét a gép müködése közben a XIX.táblán látható elnevezések szerint ismertetjük.

F.1.1. Az adatok lyukasztása és bevitele.

Emlitettük, hogy az M-3 gépen az adatok /utasitások, kiinduló számadatok/ bevitele lyukszalag segitségével történik. Az elkészitett program realizálásának első lépése tehát az adatoknak lyukszalagra történő perforálása. Ez egy bármilyen perforátorral ellátott normál távgépiróval elvégezhető /Siemens/. A /'/ billentyü a"cim vége",a /=/ billentyü a"szó vége"jelet perforálja.

A számok bevitele történhetik cim megadásával vagy program segitségével. Az előző esetben az adatokat a következőképen lyukasztjuk:

1. Beütjük annak a rekesznek a cimét, ahova a szóbanforgó adatot el akarjuk helyezni.

2. Beütjük a"cim vége"billentyüt.

3. Beütjük az adatot /utasitás vagy nyolcas számrendszerben beirt szám/.

4. Beütjük a"szó vége" jelet /=/.

A perforátor a billentyüknek megfelelő jeleket a szalagra lyukasztja és egyuttal, mint irógép, ki is irja 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 utasitást /vagy számot/ a 0006 rekeszbe fogja beirni. Ha szó vége jelet nem irunk, akkor a szóbanforgó szám nem iródik a memóriába. Ilyen módon érvénytelenithetjük az esetleg tévesen beütött számokat.

Ezután a bevitelt a következő módon hajtjuk végre:

1. A "Folyamatos-Lépésenként" kapcsolót "Lépésenként" helyzetbe állitjuk.

2. A "Bevitel 8-10" kapcsolót "8" helyzetbe állitjuk /minthogy a cim 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. /Lépésenkinti bevitelnél egy-egy szó bevitele után a gép megáll./

4. Beinditjuk a transzmitter motorját a megfelelő kapcsoló segitségével.

5. A bevitel "Be" gombjának benyomása után a transzmitter elkezdi a szalagon lévő adatok beolvasását /bevitel jelzőlámpája kigyullad/.

Azokat az adatokat, amelyeket valamely program végrehajtása során viszünk be /a Be 07 utasitással/ cim és cim vége jel nélkül perforáljuk. Pl.: A + 0,01079 tizes számrendszerben felirt számot a + 0107900 alaknak megfelelően perforáljuk /a megfelelő billentyük 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 utasitás/, akkor a program elinditása 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 állitjuk.

Ilymódon, amikor a gépben már bentlévő programban a végrehajtás során egy beviteli utasitás következik, /Be 07/ akkor a gép beolvassa a szalagról a megfelelő adatot és beirja az utasitás második ciménck megfelelő rekeszébe.

F.1.2. Számok beállitása a C, SzR, PR regiszterekbe.

A vezérlő pultra kivezetett regiszterekbe/C; SzR; PR./ a számokat nyomógombok segitségével is beállithatjuk. Ez ugy történik, hogy az illető regiszter "Beállitás-Munka" kapcsolóját "Beállitás" helyzetbe hozzuk, a regiszter nyomógombjain benyomjuk a megfelelő számot. Az "Impulzus" gomb megnyomásával az adott szám beiródik a megfelelő regiszterbe és leolvasható az illető regiszter jelzőlámpáin.

F.1.3. 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 beállitjuk az első utasitás cimét.

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 előkészitjük a gépet kivitelre.

a/ A megfelelő kapcsolóval bekapcsoljuk a távgépiró motorját.
b/ A "Kivitel 8-10" kapcsolót a megfelelő helyzetbe állitjuk.
c/ A "Kivitel" kapcsolót munkahelyzetbe állitjuk.

5. A "Folyamatos-Lépésenként" kapcsolót "Folyamatos" helyzetbe állitjuk.

6. Ha ezek után az "RI inditás" gombot benyomjuk, a gép elkezdi a program végrehajtását.

F.1.4. A program lépésenkénti végrehajtása.

/A "Folyamatos-Lépésenkénti" kapcsoló "Lépésenként" helyzetel./

Mikor a program végrehajtását elkezdjük, célszerü az első néhány utasitást lépésenkint végrehajtani. A gép müködésének helyességét is ellenőrizhetjük az egyes utasitások lépésenkinti végrehajtásával. Egy adott utasitást három lépsében lehet végrehajtani:

1. Az "RI Inditás" gomb megnyomásával a gép végrehajtja az adott utasitás első és második taktusát és utána leáll. A C regiszterben megjelenik az adott utasitás, az SzR-ben az utasitás cime.

2. Ujra benyomva az "RI Inditás" gombot a gép végrehajtja a harmadik, negyedik, ötödik és hatodik taktust, majd megáll. A C regiszterben megjelenik a második cim tartalma és az SzR-ben maga a cim. A PR-ben megjelenik a soronkövetkező utasitás cime.

3. Ismét benyomva az "RI Inditás" gombot a gép végrehajtja az utasitást. A B és C regiszterben megjelenik a müvelet eredménye.

F.1.5. Beirás a memóriába.

A vezérlő pulton lévő nyomógombok segitségével lehetőség van arra is, hogy a memóriába adatokat vigyünk be. Ez a következőképen történik:

1. Beállitjuk a C regiszterbe a beviendő számot.

2. Beállitjuk az SzR regiszterbe azt a cimet, ahova a számot irni akarjuk.

3. Az "Irás" gomb megnyomásával megtörténik a memóriába való bevitel.

F.1.6. Kihivás a memóriából.

A memóriába való beiráson kivül a memória egy adott rekesze tartalmának a kiolvasására is van lehetőség. Ez a következőképen történik:

1. A "Folyamatos-Lépésenként" kapcsolót "Lépésenként" helyzetbe állitjuk.

2. Az SzR-ben beállitjuk a kiolvasandó rekesz cimét.

3. Az "Olvasás" gomb megnyomásával a C regiszterben megjelenik az SzR-ben beállitott cim tartalma.

F.1.7. Megállitás egy adott cimnél.

A vezérlő pulton lévő kapcsolók segitségével lehetséges a program végrehajtása során a gépnek egy adott cim szerinti megállitása is. Ez a cim lehet egy utasitás vagy egy adat cime. A cim szerinti megállitással a gép helyes müködését ellenőrizhetjük. A cim szerinti megállás beállitása a következőképen történik:

1. A "Folyamatos-Megállás" kapcsolót "Megállás" helyzetbe hozzuk.

2. A vezérlő pult jobboldalán lévő kapcsolókon beállitjuk az adott cimet.

3. Ha egy utasitás cime szerint akarjuk leállitani a gépet akkor a "PR-SzR" kapcsolót PR helyzetbe hozzuk. Ha a program végrehajtása során PR-ben megjelenik az adott utasitás cime, akkor a gép legáll.

4. Ha egy adat cime szerinti megállást akarunk, akkor a "PR-SzR" kapcsolót SzR helyzetbe hozzuk. A gép megáll, ha a szóbanforgó adat cime /egy utasitás első vagy második cime/ az SzR-ben megjelenik.

A vezérlőpulton kivül 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 müködés közben megáll a megállás oka a regiszterek jelzőlámpáin leolvasható; mégpedig

a/ ha a B'0-ban megállási utasitás van, akkor a megállás ennek következménye,

b/ ha összeadás és osztás esetén a D regiszter nulladik helyértékén 1-es áll, akkor a megállás tulcsordulás miatt következik be,

c/ minden más esetben a megállás program vagy géphiba következménye. Ilyen esetben célszerü az inditóregiszterben lévő cim előtti cimen található utasitást megvizsgálni.

F.2. Az M-3 utasitásrendszere.

Kód:	Jel:	Magyarázat:
00	+	Az első cim tartalmához hozzáadódik a második cim tartalma és beiródik a memóriába a második cimre.
10	+,	Az első cim tartalmához hozzáadódik a második cim tartalma, de a memóriába nem iródik be.
20	↓+	Az előző müvelet eredményéhez hozzáadódik az első cim tartalma és az eredmény beiródik a memóriába a második cimre.
30	↓+,	Az előző müvelet eredményéhez hozzáadódik az első cim tartalma, de a memóriába nem iródik be.
40	+ n	Az első cim tartalmához hozzáadódik a második cim tartalma. Az eredmény beiródik a memóriába a második cimre és kinyomtatódik.
50	|+|,	Az első cim tartalmának abszolut értékéhez hozzáadódik a második cim tartalmának abszolut értéke, de a memóriába nem iródik be.
60	↓+ n	Az előző müvelet eredményéhez hozzáadódik az első cim tartalma. Az eredmény beiródik a memóriába a második cimre és kinyomtatódik.
70	↓|+|,	Az előző müvelet eredményének abszolut értékéhez hozzáadódik az első cim tartalmának abszolut értéke, de a memóriába nem iródik be.
01	−	A második cim tartalmából kivonódik az első cim tartalma és az eredmény beiródik a memóriába a második cimre.

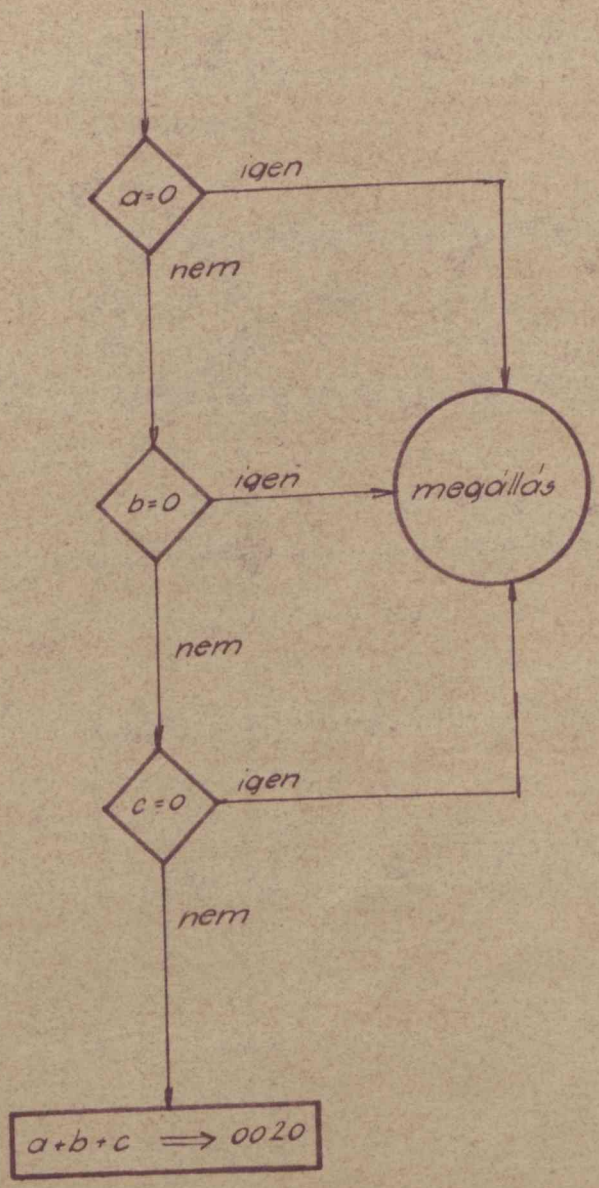
Kód:	Jel:	Magyarázat:
11	-,	A második cim tartalmából kivonódik az első cim tartalma, de a memóriába nem iródik be, - az eredmény megmarad az aritmetikai egységben.
21	↓-	Az előző művelet eredményéből kivonódik az első cim tartalma, az eredmény beiródik a memóriába a második cimre.
31	↓-,	Az előző művelet eredményéből kivonódik az első cim tartalma, az eredmény nem iródik be.
41	-n	A második cim tartalmából kivonódik az első cim tartalma, az eredmény beiródik a második cimre és kinyomtatódik.
51	|-|,	A második cim tartalmának abszolut értékéből kivonódik az első cim tartalmának abszolut értéke, az eredmény nem iródik le.
61	↓- n	Az előző művelet eredményéből kivonódik az első cim tartalma. Az eredmény beiródik a második cimre és kinyomtatódik.
71	↓ |-|,	Az előző művelet eredményének abszolut értékéből kivonódik az első cim tartalmának abszolut értéke, de a memóriába nem iródik be.
02	:	A második cim tartalma elosztódik az első cim tartalmával. Az eredmény beiródik a memóriába a második cimre.
12	:,	A második cim tartalma elosztódik az első cim tartalmával. Az eredmény a memóriába nem iródik be.
22	↓:	Az előző művelet eredménye elosztódik az első cim tartalmával és az eredmény beiródik a memóriába a második cimre.
32	↓:,	Az előző művelet eredménye elosztódik az első cim tartalmával. Az eredmény a memóriába nem iródik be.

Kód:	Jel:	Magyarázat:
42	: n	A második cim tartalma elosztódik az első cim tartalmával. Az eredmény beiródik a memóriába a második cimre és kinyomtatódik.
52	|:|,	A második cim tartalmának abszolut értéke elosztódik az első cim tartalmának abszolut értékével, de a memóriába nem iródik be.
62	↓: n	Az előző müvelet eredménye elosztódik az előző cim tartalmával. Az eredmény beiródik a memóriába a második cimre és kinyomtatódik.
72	↓|:|,	Az előző müvelet eredményének abszolut értéke elosztódik az első cim tartalmának abszolut értékével, de a memóriába nem iródik be.
03	x	A második cim tartalma összeszorzódik az első cim tartalmával. Az eredmény beiródik a memóriába a második cimre.
13	x,	A második cim tartalma összeszorzódik az első cim tartalmával. Az eredmény a memóriába nem iródik be.
23	↓x,	Az előző müvelet eredménye összeszorzódik az első cim tartalmával. Az eredmény beiródik a memóriába a második cimre.
33	↓x	Az előző müvelet eredménye összeszorzódik az első cim tartalmával. Az eredmény a memóriába nem iródik be.
43	x n	A második cim tartalma összeszorzódik az első cim tartalmával. Az eredmény beiródik a memóriába a második cimre és kinyomtatódik.
53	|x|,	A második cim tartalmának abszolut értéke összeszeszorzódik az első cim tartalmának abszolut értékével, de a memóriába nem iródik be.

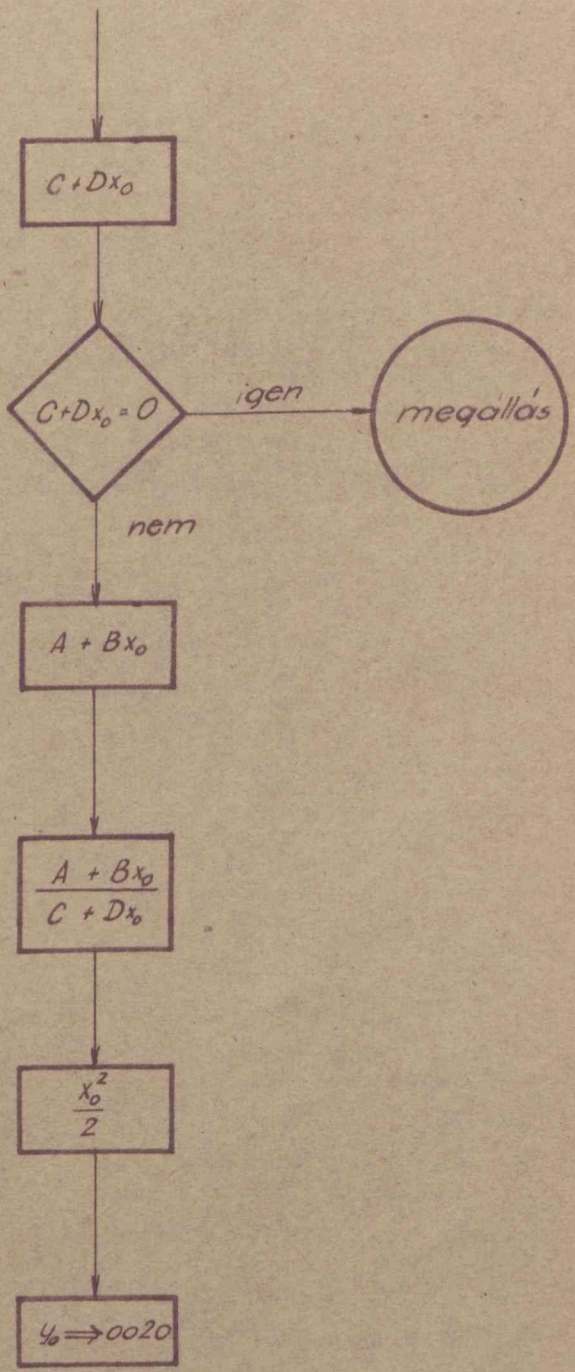
Kód:	Jel:	Magyarázat:
63	↓x n	Az előző müvelet eredménye összeszorzódik az első cim tartalmával. Az eredmény beiródik a második cimre és kinyomtatódik.
73	↓|x|,	Az előző müvelet eredményének abszolut értéke összeszorzódik az első cim tartalmának abszolut értékével, de a memóriába nem iródik be.
06	∧	A második cim tartalma logikailag összeszorzódik az első cim tartalmával, és az eredmény beiródik a memóriába a második cimre.
16	∧,	A második cim tartalma logikailag összeszorzódik az első cim tartalmával. Az eredmény megmarad az aritmetikai egységben.
26	↓∧	Az előző müvelet eredménye logikailag összeszorzódik az első cim tartalmával és az eredmény beiródik a memóriába a második cimre.
36	↓∧,	Az előző müvelet eredménye logikailag összeszorzódik az első cim tartalmával. Az eredmény a memóriába nem iródik be.
46	∧ n	A második cim tartalma logikailag összeszorzódik az első cim tartalmával. Az eredmény beiródik a második cimre és kinyomtatódik.
56	|∧|,	A második cim tartalmának abszolut értéke logikailag összeszorzódik az első cim tartalmának abszolut értékével. Az eredmény a memóriába nem iródik be.
66	↓∧ n	Az előző müvelet eredménye logikailag összeszorzódik az első cim tartalmával. Az eredmény beiródik a memóriába a második cimre és kinyomtatódik.
76	↓|∧|,	Az előző müvelet eredményének abszolut értéke logikailag összeszorzódik az első cim tartalmának abszolut értékével, de a memóriába nem iródik be.

Kód:	Jel:	Magyarázat:
07; 27	Be	Szám bevitele. A szám a perforált szalagról beiródik a második cimre. A regiszterben nem marad meg.
05; 15	Á	Az első cim tartalma átmegy a második cimre és megmarad az aritmetikai egység kettes számu regiszterében.
45; 55	Án	Az első cim tartalma átmegy a második cimre és kinyomtatódik, de az aritmetikai egységben nem marad meg.
24	Ul	Vezérlés átadás. A következő utasitást a gép az első cimről veszi. Az előző müvelet eredménye beiródik a második cimre és megmarad az aritmetikai egységben.
64	Uln	Ugyanaz, mint a 24-es utasitás, de a szám még ki is nyomtatódik.
74	U2	A következő utasitást a gép a második cimről veszi. Az aritmetikai egységben megmarad az előző müvelet eredményének abszolut értéke.
34	FU	Feltételes ugrás. A következő utasitást a gép a második cimről veszi, ha az előző müvelet eredménye pozitiv, és az első cimről ha az előző müvelet eredménye negativ.
04 14 44 54		A megállási utasitások az aritmetikai egység regisztereinek, valamint a kiválasztó és beinditó regiszterek tartalmában különböznek egymástól.
17 37 57 77	megállás	

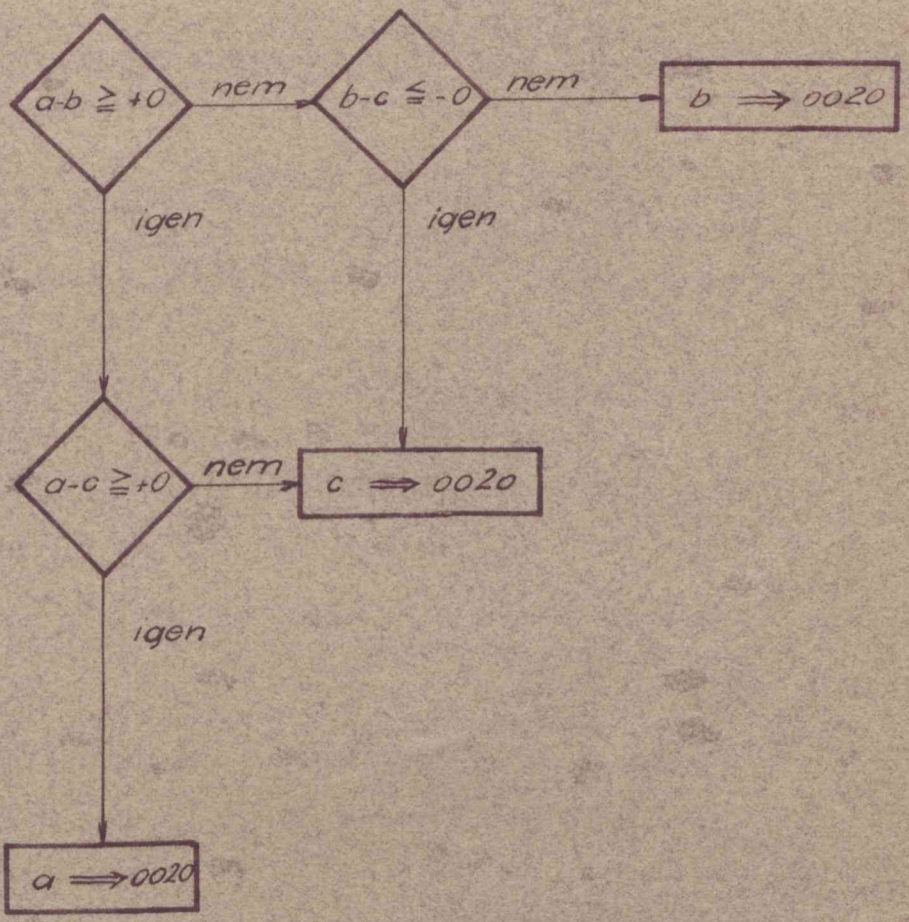
282/D/59.november 9.
V.Iné.
Készült: 150 pld-ban.



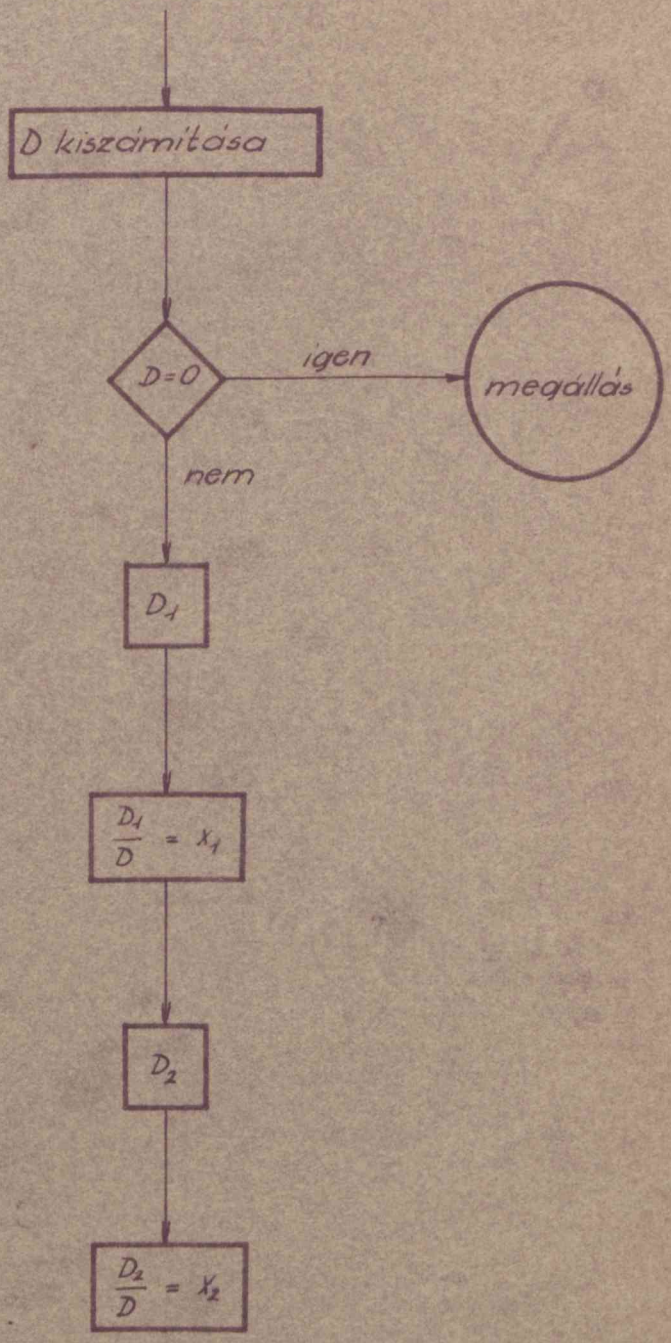
1 tábla



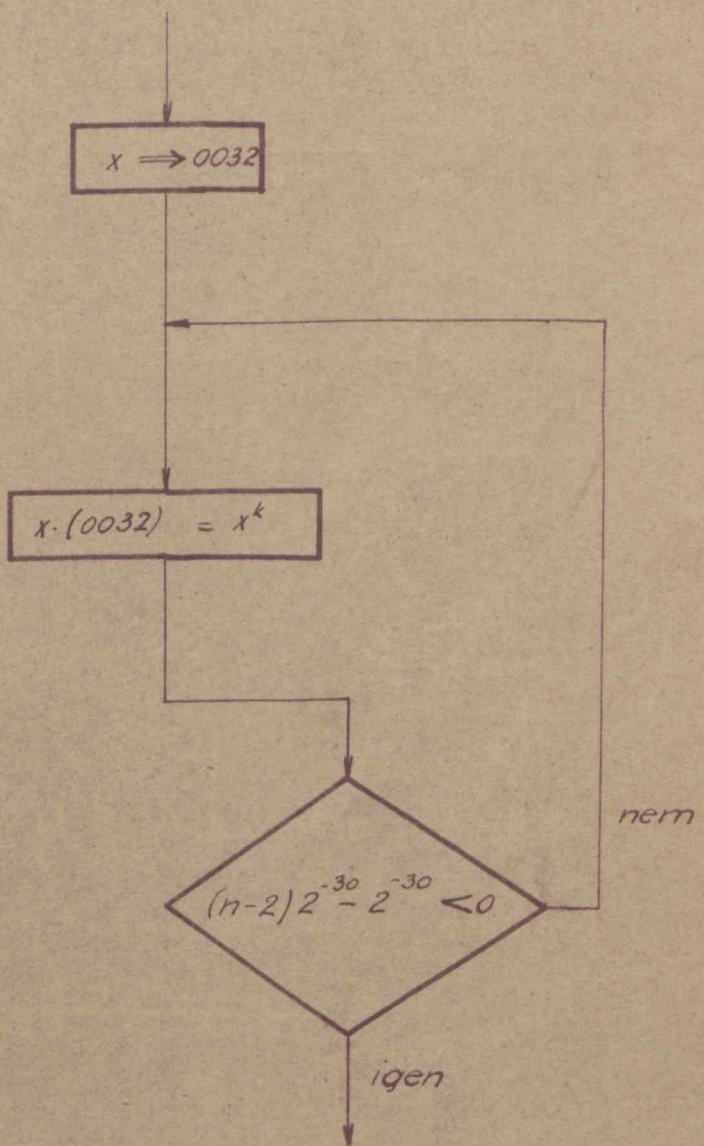
II. tábla



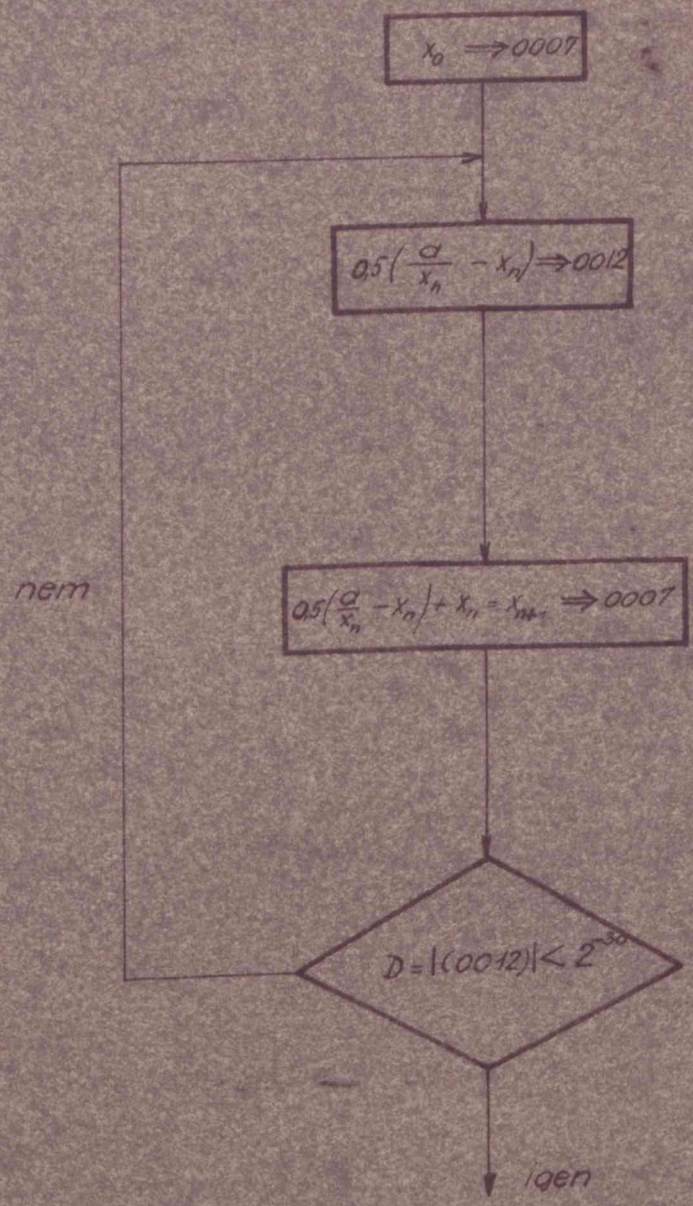
III tábla


D kiszámítása
D=0
igen
megállás
nem
D_1
$\frac{D_1}{D} = X_1$
D_2
$\frac{D_2}{D} = X_2$

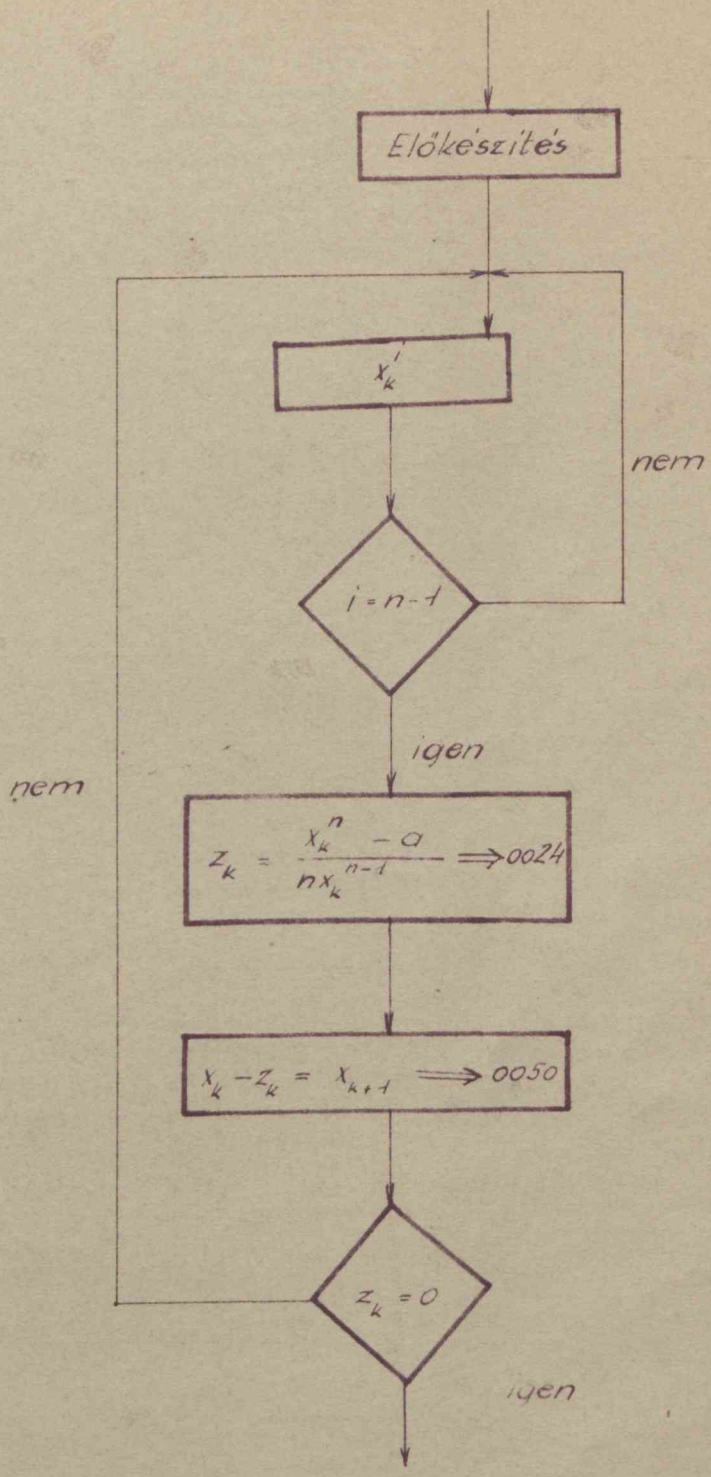
IV. tábla



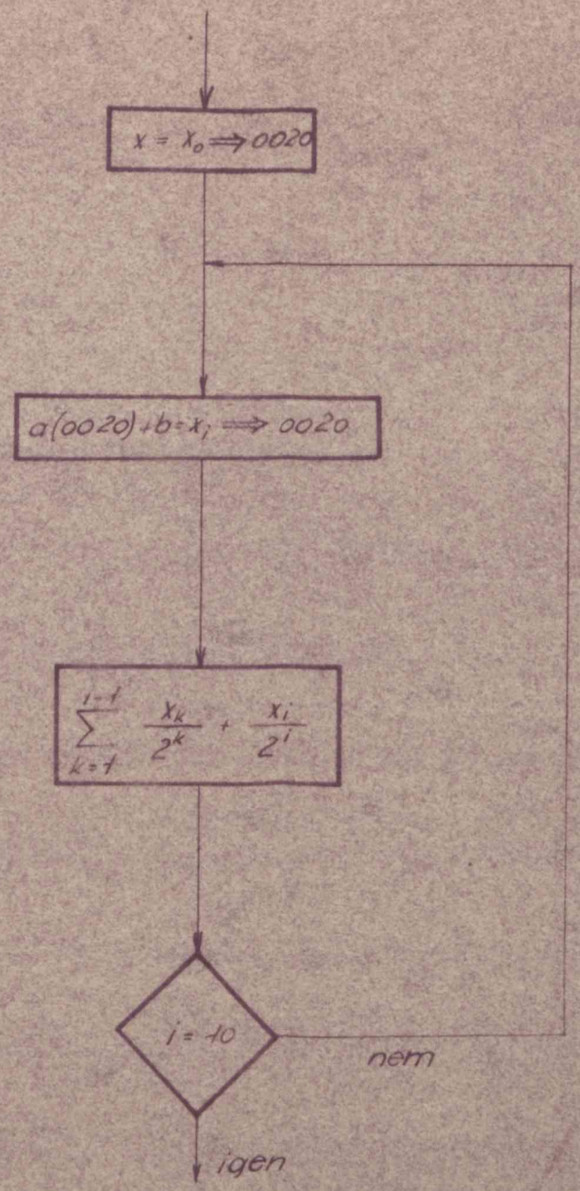
V tábla


$x_0 \Rightarrow 0007$
$0.5\left(\frac{a}{x_n} - x_n\right) \Rightarrow 0012$
$0.5\left(\frac{a}{x_n} - x_n\right) + x_n = x_{n+1} \Rightarrow 0007$
nem
$D = |(0012)| < 2^{-30}$
igen

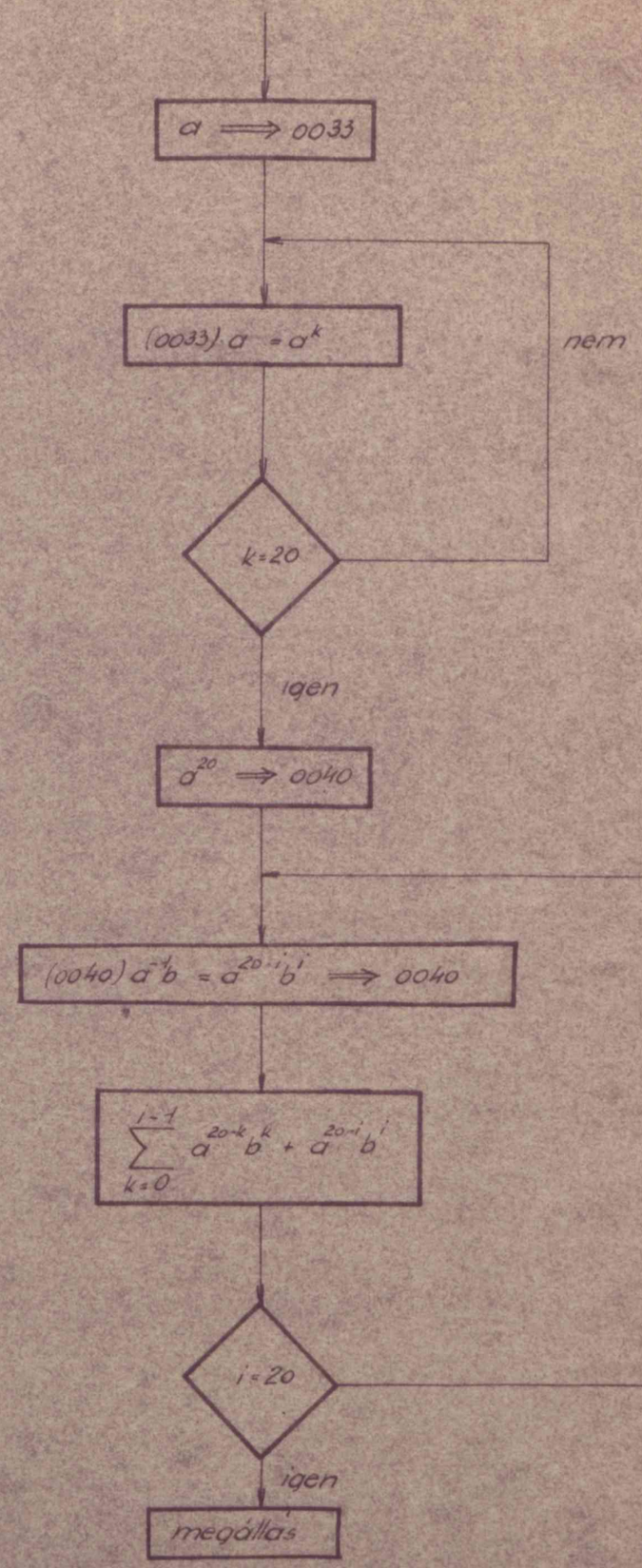
VI tábla


Előkészítés
x_k^i
nem
$i = n-1$
igen
nem
$z_k = \frac{x_k^n - a}{n x_k^{n-1}} \Longrightarrow 0024$
$x_k - z_k = x_{k+1} \Longrightarrow 0050$
$z_k = 0$
igen

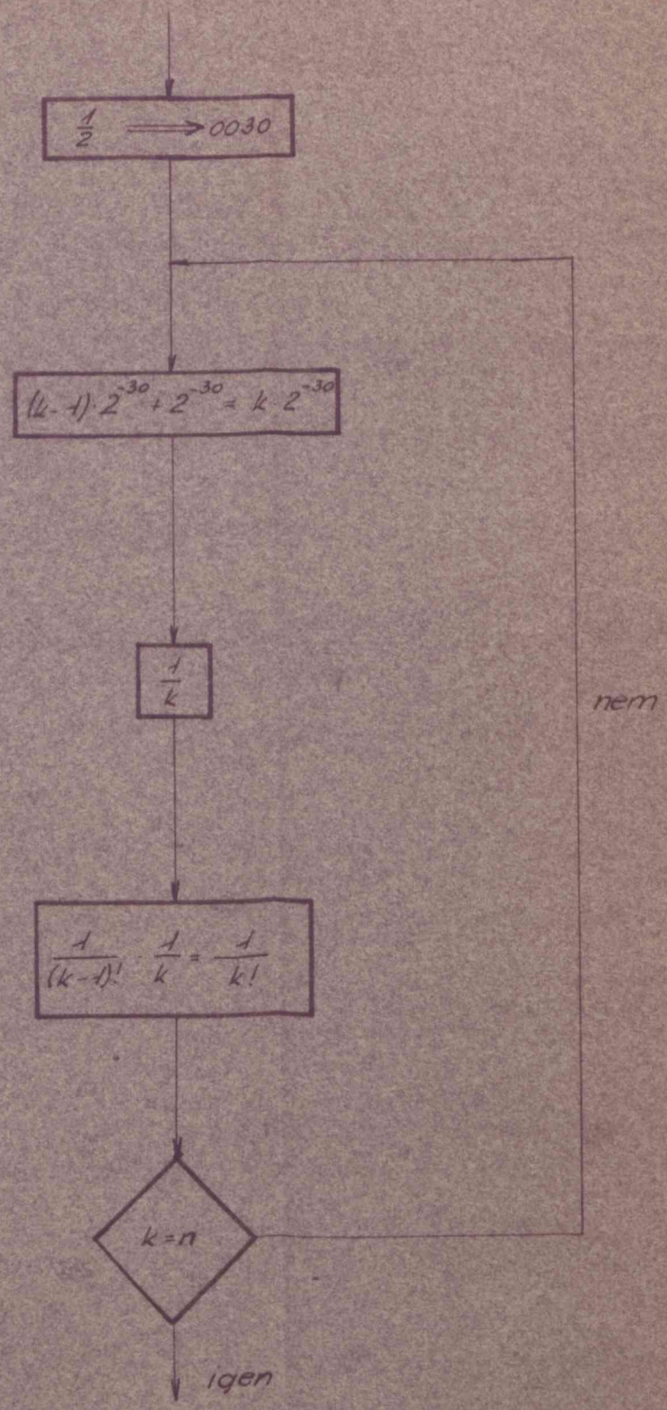
VII. tábla


$x = x_0 \Longrightarrow 0020$
$a(0020)+b=x_i \Longrightarrow 0020$
$\sum_{k=1}^{i-1} \frac{x_k}{2^k} + \frac{x_i}{2^i}$
$i = 10$
nem
igen

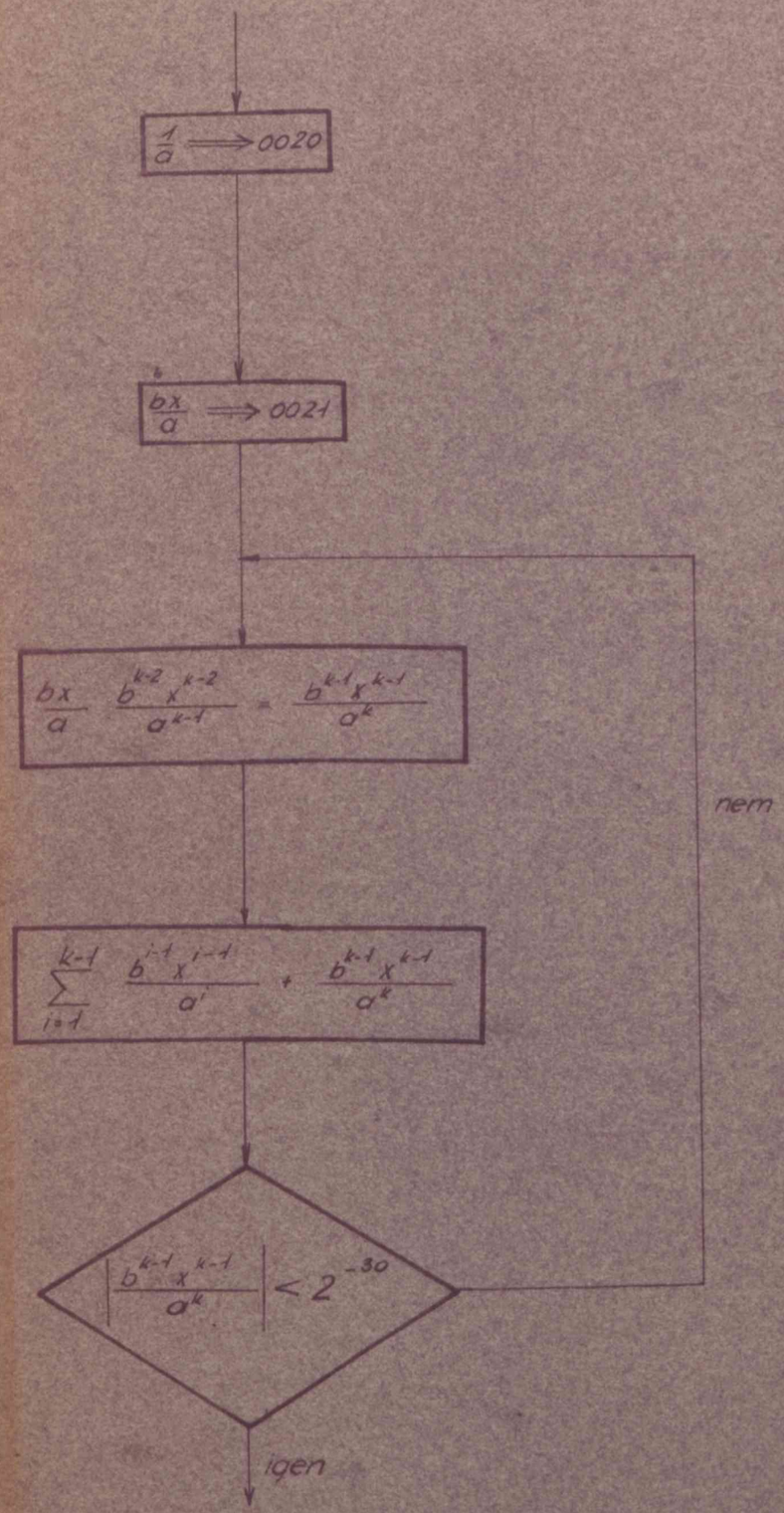
1. ábra


$a \Longrightarrow 0033$
$(0033).a = a^k$
nem
$k=20$
igen
$a^{20} \Longrightarrow 0040$
$(0040)\,a^i b = a^{20-i} b^i \Longrightarrow 0040$
$\sum_{k=0}^{i-1} a^{20-k} b^k + a^{20-i} b^i$
nem
$i=20$
igen
megállás

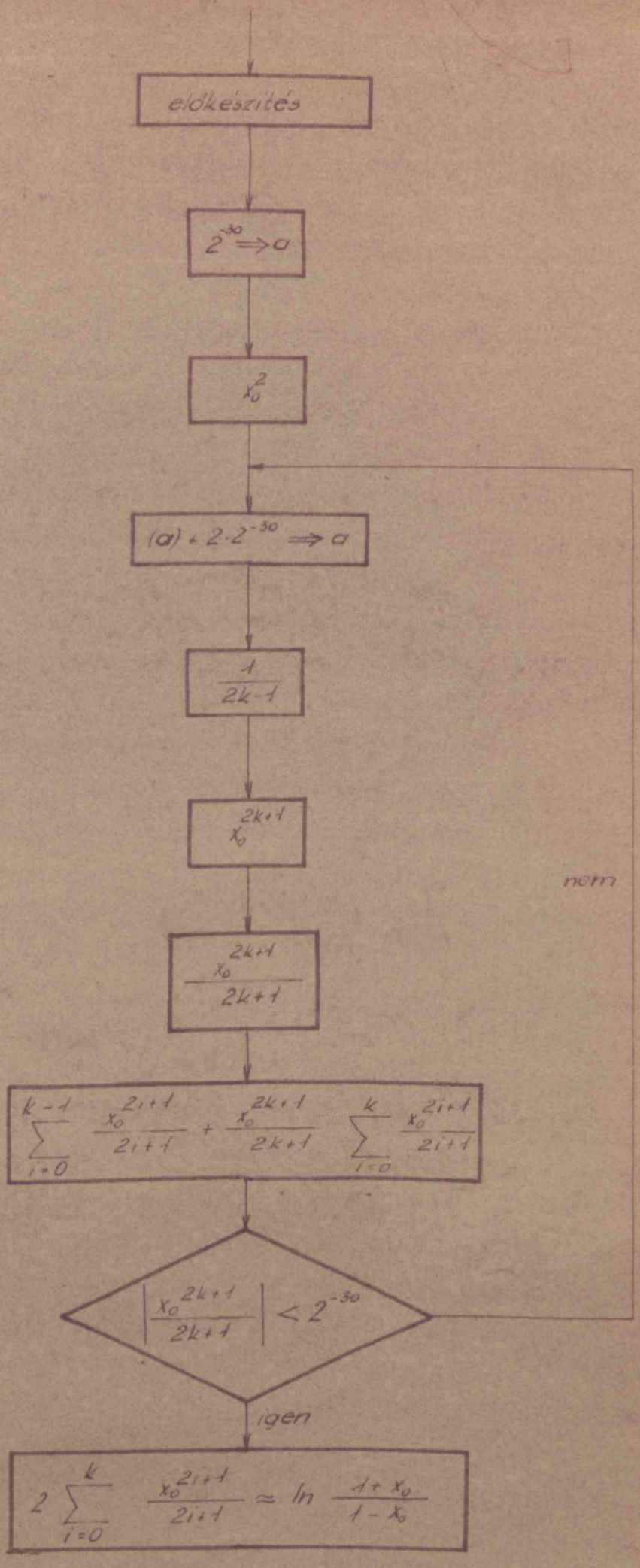
2. ábra


$\frac{1}{2} \Longrightarrow 0030$
$(k-1)\cdot 2^{-30} + 2^{-30} = k\cdot 2^{-30}$
$\frac{1}{k}$
$\frac{1}{(k-1)!}\cdot\frac{1}{k} = \frac{1}{k!}$
$k=n$
nem
igen

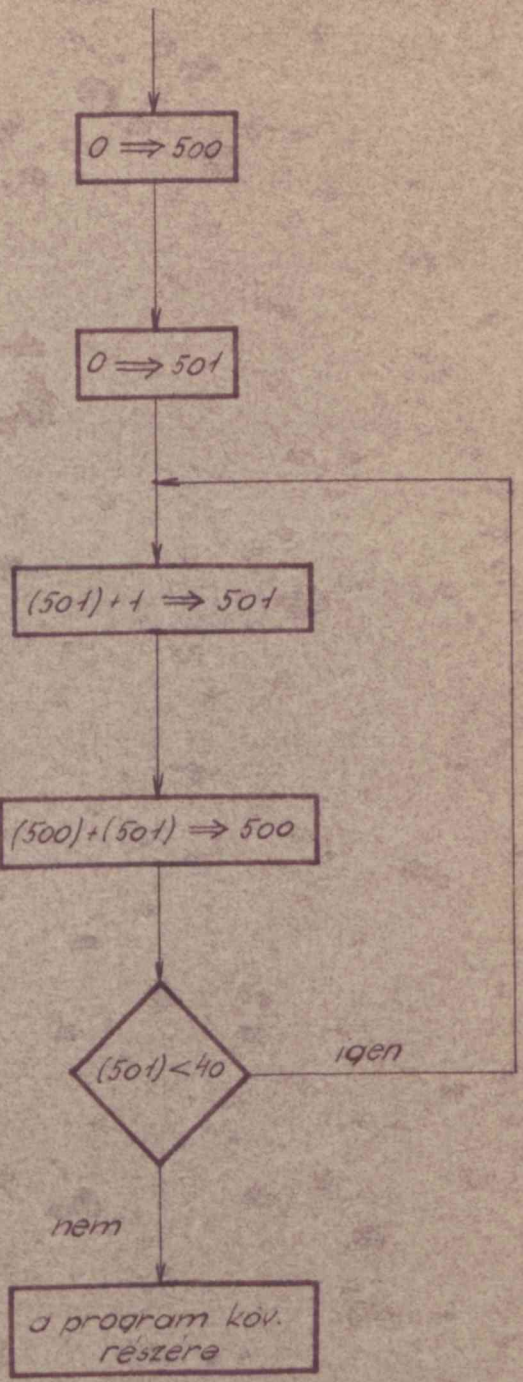
1. ábra


$\frac{1}{a} \Longrightarrow 0020$
$\frac{bx}{a} \Longrightarrow 0021$
$\frac{bx}{a}$ $\frac{b^{k-2}x^{k-2}}{a^{k-1}}$ $\frac{b^{k-1}x^{k-1}}{a^{k}}$
$\sum_{i=1}^{k-1} \frac{b^{i-1}x^{i-1}}{a^{i}} + \frac{b^{k-1}x^{k-1}}{a^{k}}$
$\left|\frac{b^{k-1}x^{k-1}}{a^{k}}\right| < 2^{-30}$
nem
igen

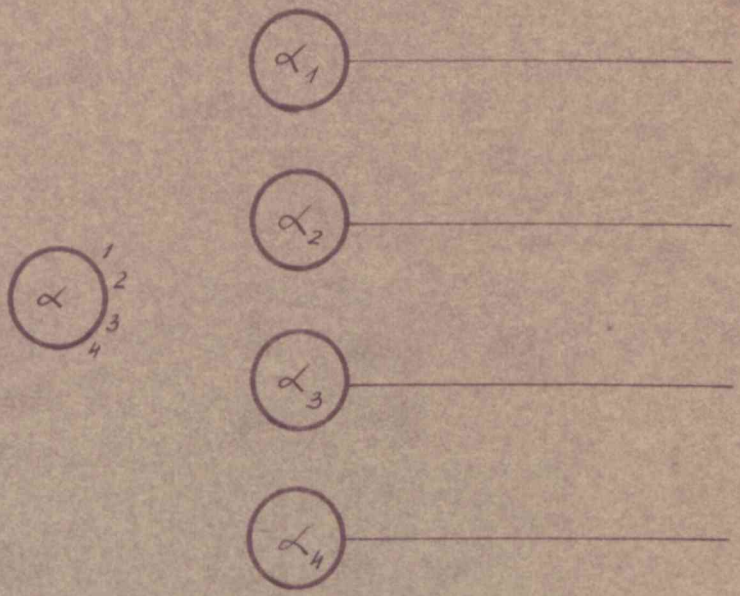
2. ábra


előkészítés
$2^{-30} \Rightarrow a$
x_0^2
$(a) + 2 \cdot 2^{-30} \Rightarrow a$
$\frac{1}{2k-1}$
x_0^{2k+1}
$\frac{x_0^{2k+1}}{2k+1}$
$\sum_{i=0}^{k-1} \frac{x_0^{2i+1}}{2i+1} + \frac{x_0^{2k+1}}{2k+1} \sum_{i=0}^{k} \frac{x_0^{2i+1}}{2i+1}$
$\left| \frac{x_0^{2k+1}}{2k+1} \right| < 2^{-30}$
nem
igen
$2 \sum_{i=0}^{k} \frac{x_0^{2i+1}}{2i+1} \approx \ln \frac{1+x_0}{1-x_0}$

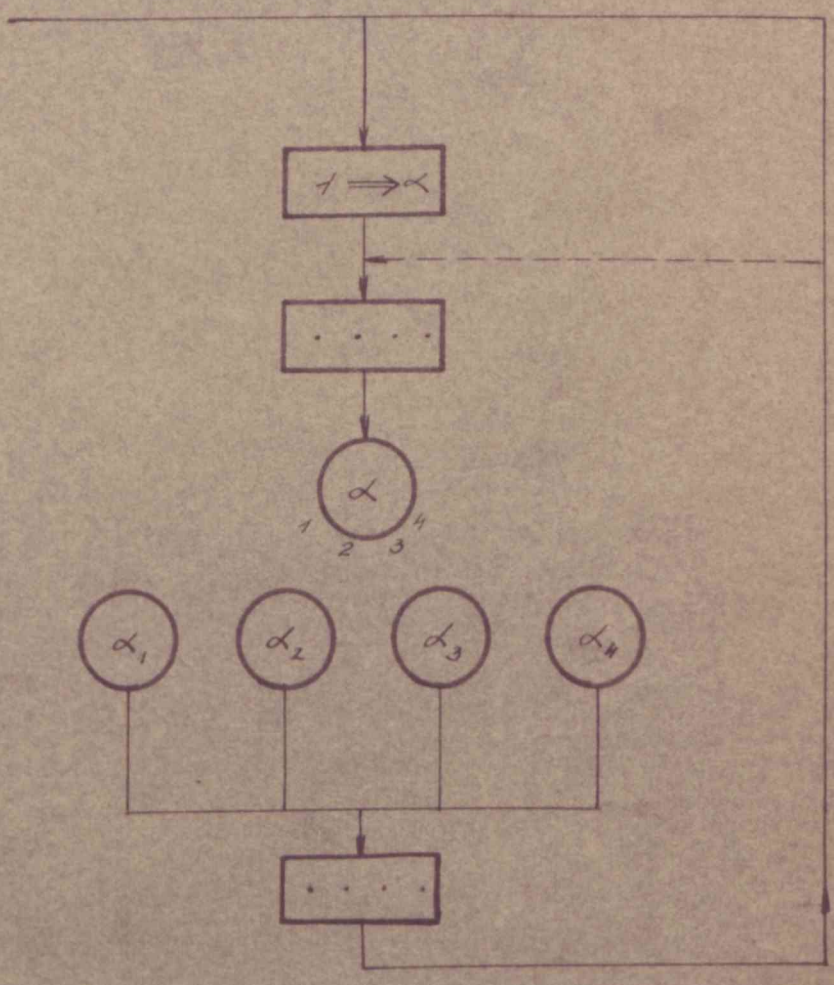
X tábla


0 ⇒ 500
0 ⇒ 501
(501) + 1 ⇒ 501
(500) + (501) ⇒ 500
(501) < 40
igen
nem
a program kov. részére

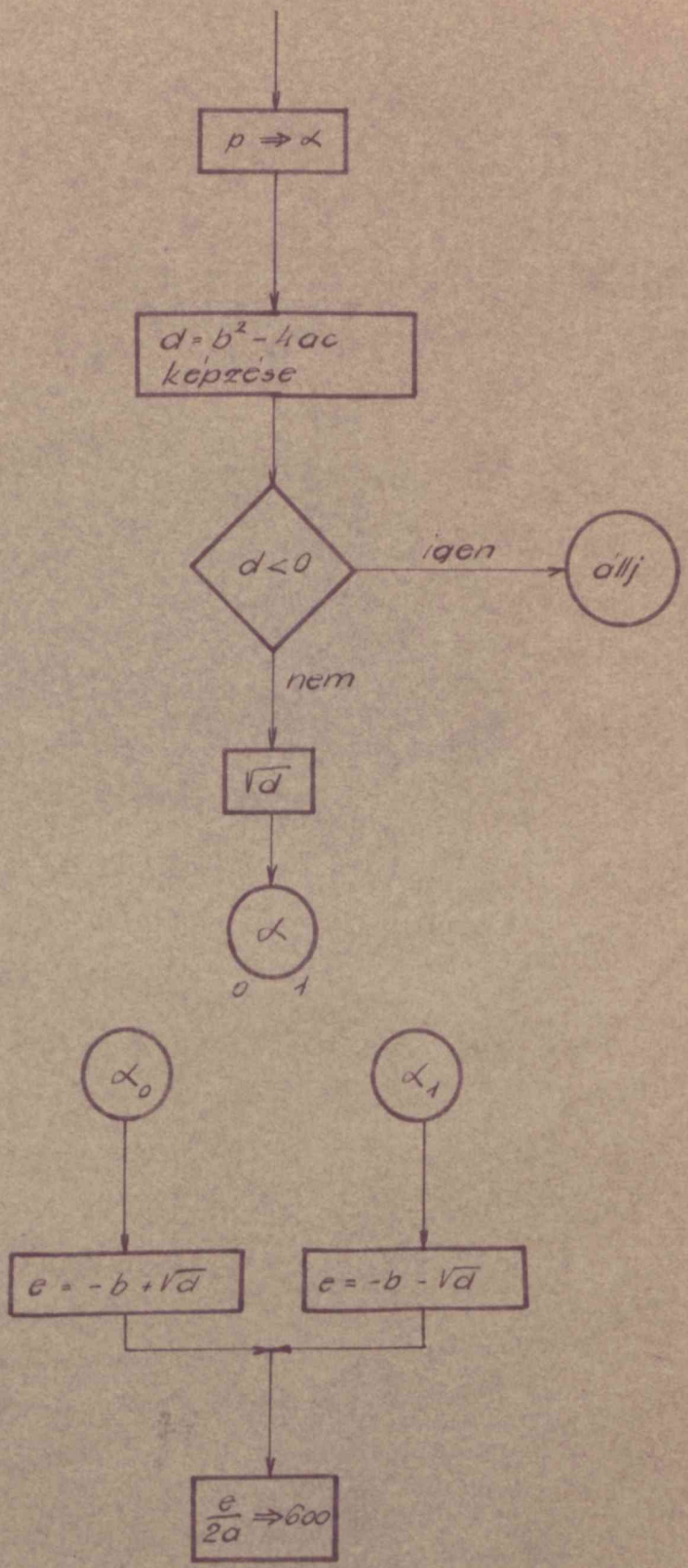
XI tábla



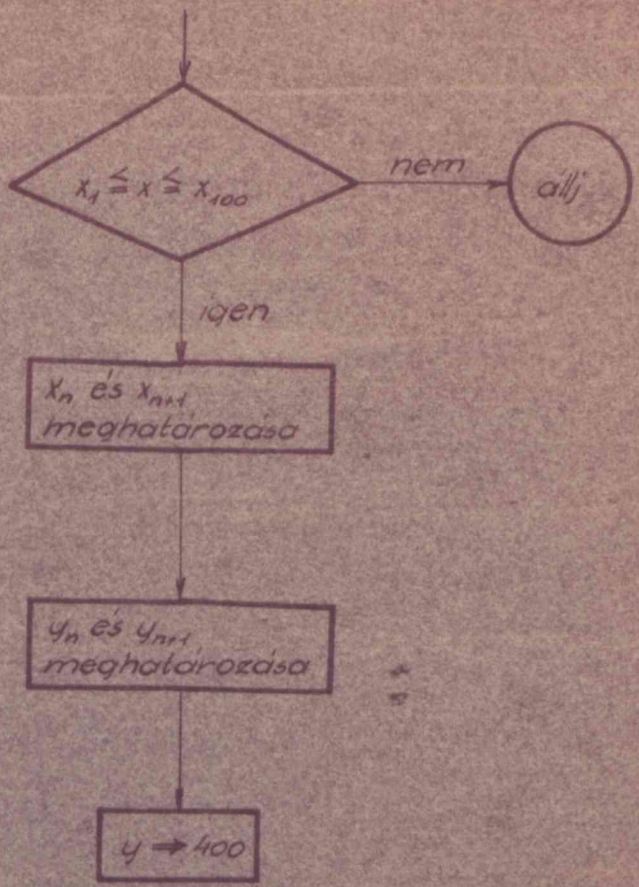
1 ábra



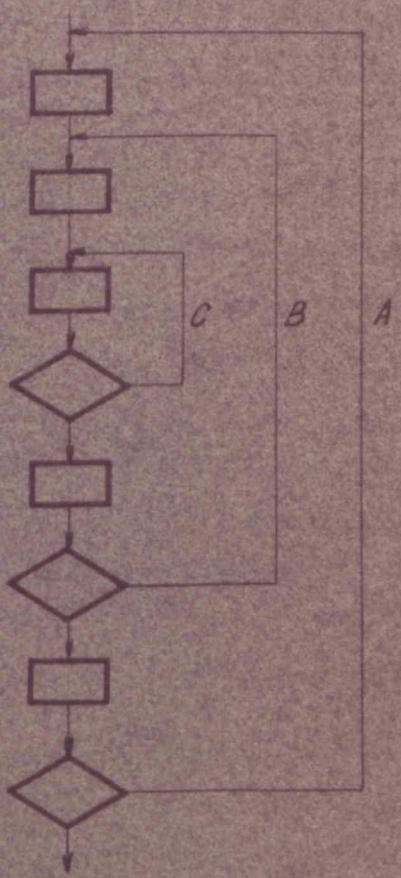
2 ábra



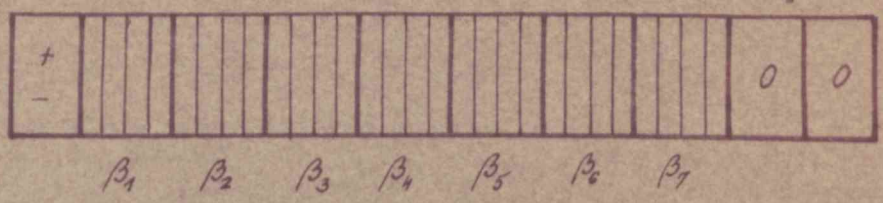
XIII tábla


$x_1 \leq x \leq x_{100}$
nem
állj
igen
x_n és x_{n+1} meghatározása
y_n és y_{n+1} meghatározása
$y \rightarrow 400$

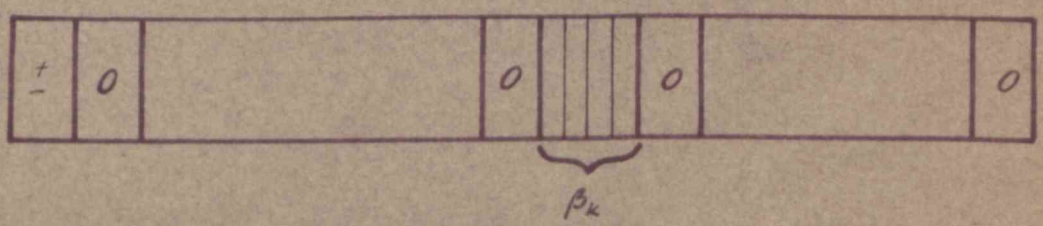
1 ábra


C
B
A

2 ábra

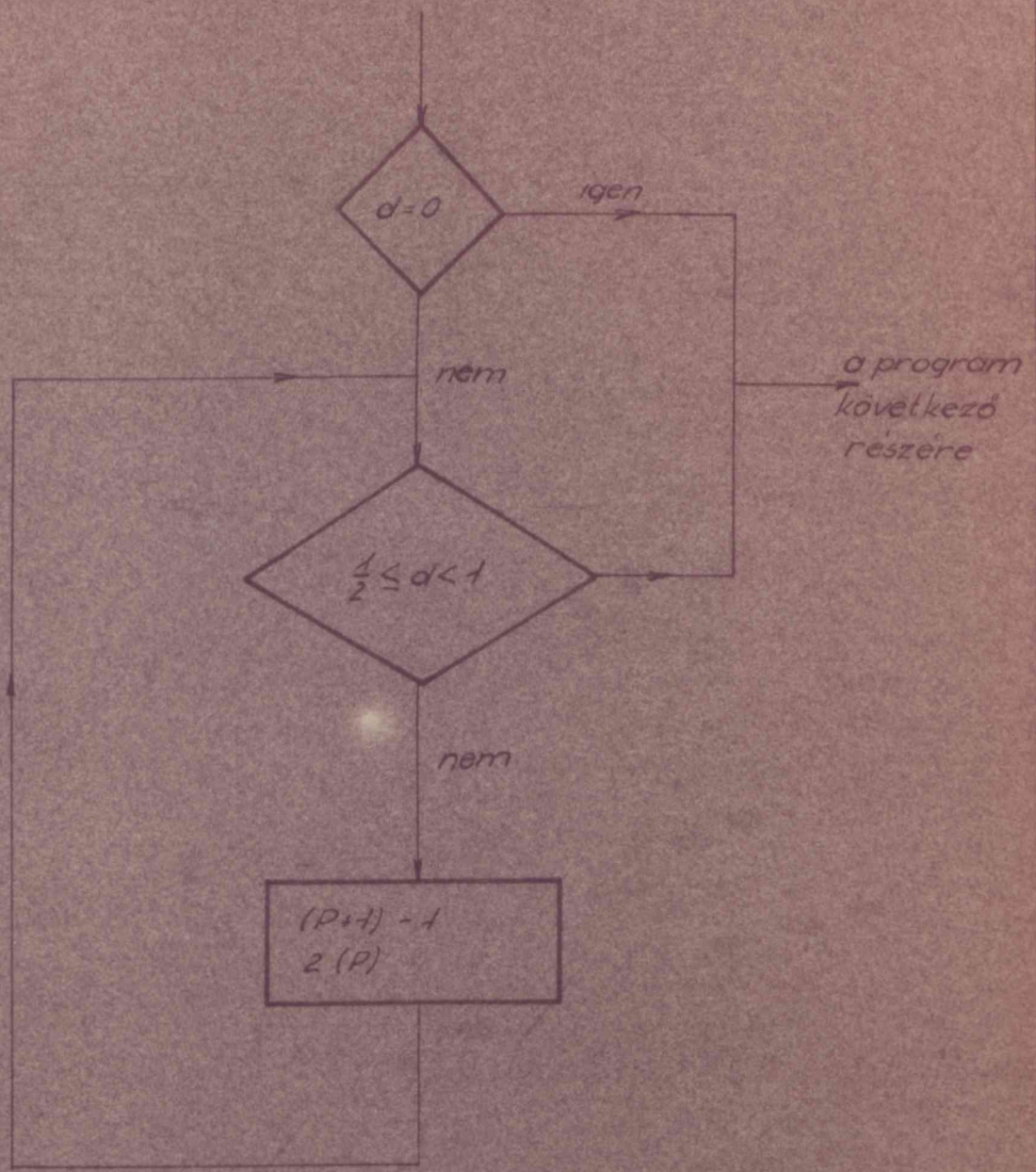


1. ábra

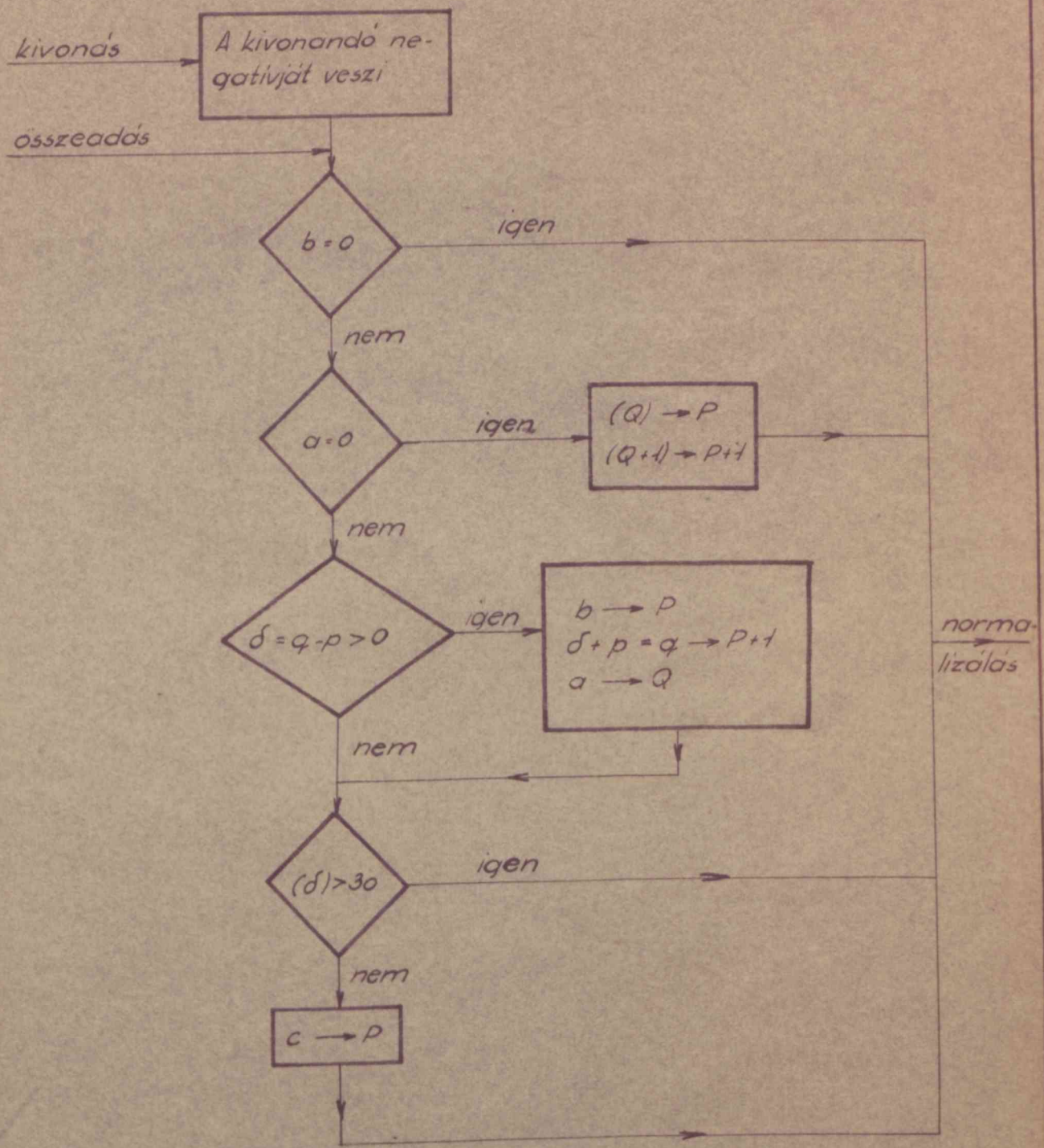


2. ábra

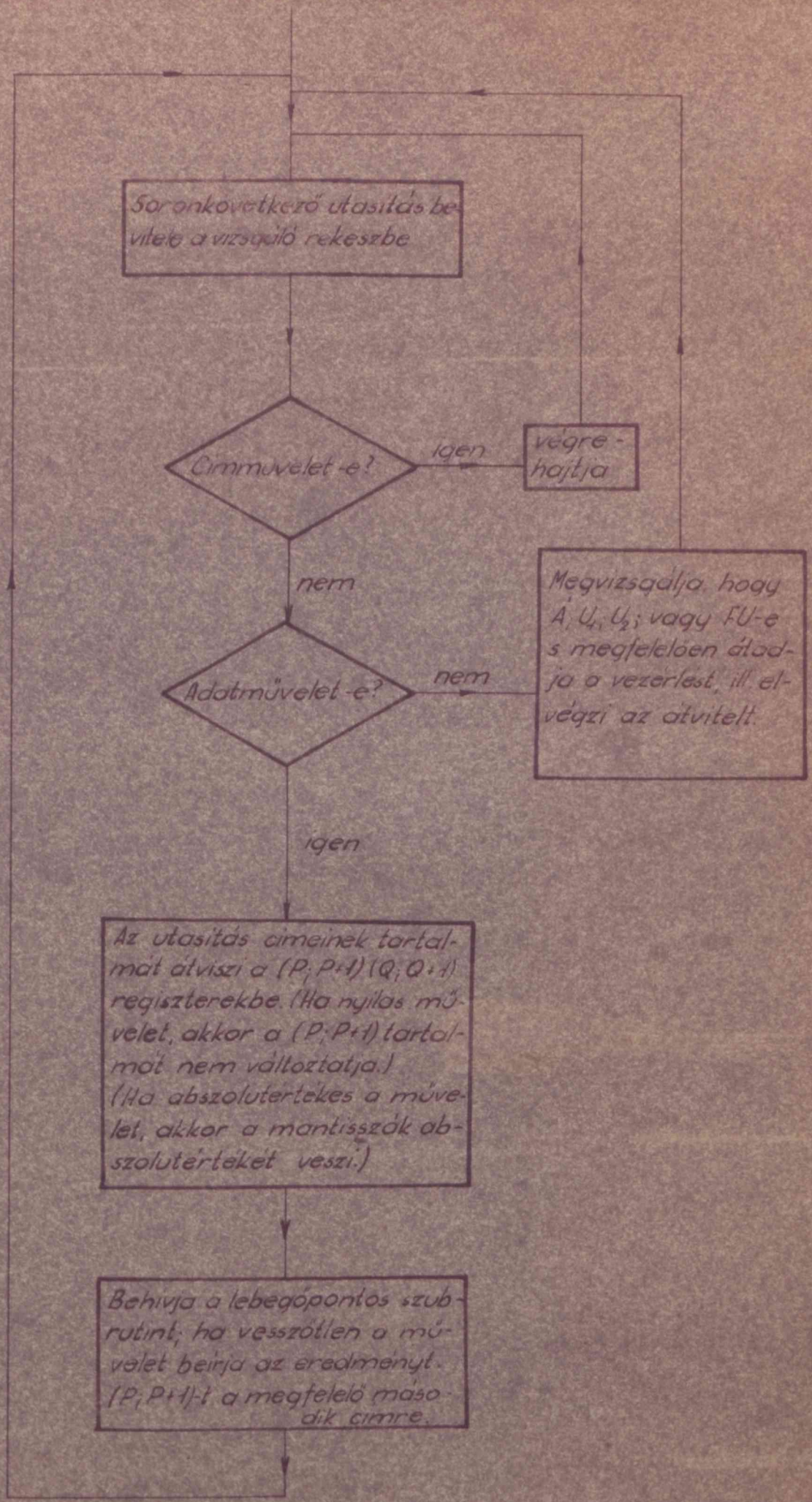
XV. tábla


d=0
igen
nem
a program
következő
részére
$\frac{1}{2} \leq d < 1$
nem
(P+1) - 1
2 (P)

XVI tábla


kivonás
A kivonandó negativját veszi
összeadás
b=0
igen
nem
a=0
igen
(Q) → P
(Q+1) → P+1
nem
δ = q-p > 0
igen
b → P
δ + p = q → P+1
a → Q
normalizálás
nem
(δ) > 30
igen
nem
c → P

XVII tábla


Soronkovetkező utasítás beírása a vizsgáló rekeszbe
Címművelet-e?
igen
végrehajtja
nem
Megvizsgálja, hogy Á, U1, U2; vagy FU-e s megfelelően átadja a vezérlést, ill. elvégzi az átvitelt
Adatművelet-e?
nem
igen
Az utasítás címeinek tartalmát átviszi a (P; P+1) (Q; Q+1) regiszterekbe. (Ha nyílás művelet, akkor a (P; P+1) tartalmát nem változtatja.)
(Ha abszolutértékes a művelet, akkor a mantisszák abszolutértékét veszi.)
Behívja a lebegőpontos szubrutint; ha vesszőtlen a művelet beírja az eredményt (P; P+1)-t a megfelelő második címre.

XVIII tábla

CR

± 4 2 1 4 2 1 | 4 2 1 4 2 1 4 2 1 4 2 1 | 4 2 1 4 2 1 4 2 1 4 2 1

Törlés | Törlés | Törlés

SzR | PR

4 2 1 4 2 1 4 2 1 4 2 1 | 4 2 1 4 2 1 4 2 1 4 2 1

BO utasítás címfő

RI

RI törés léptetőlánc nullázás

Bevitel Folyamatos

Bevitel indítása

Bevitel Lépésenként

Bevitel automatikusan

Bevitel leállítás

Kivitel

Kivitel indítása

Kivitel leállítás

Kivitel leállítás

Motor kód

Motor kód

10

10

indítás

Impulzus

CR SzR PR

Törlés

Folyamatos

Folyamatos

Megállás

Lépésenként

SzR mentő leállás

PR mentő leállás

CR mentő leállás

Beállítás (CR mentő leállás beállítása)

4 2 1 4 2 1 4 2 1 4 2 1

Memória Írás

Memória Olvasás

Memória írás befejeződött jelző

Memória olvasás befejeződött

Írás befejeződött

indítás

▽ = kapcsoló

⊗ = jelzőlámpa

□ ○ = nyomógomb


EX LIBRIS
KALMÁR
LÁSZLÓ