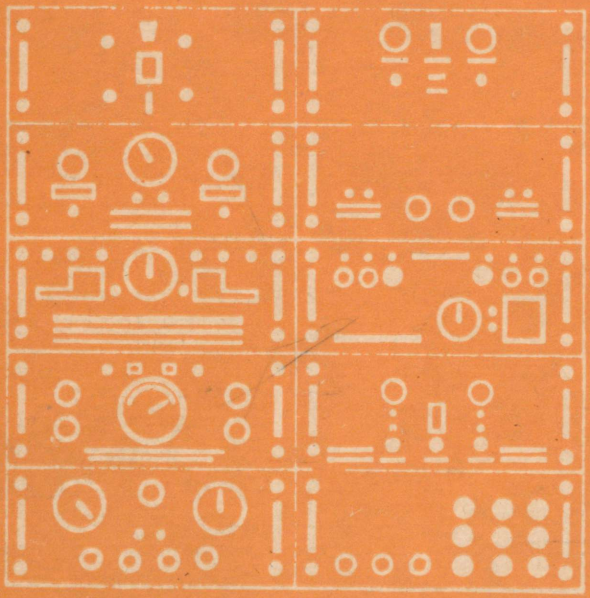
M. KRULL

KOSMOSE-SAJANDIL



ELEKTRON-ARVUTID JA PROGRAMMEERIMINE

MATI KRULL

ELEKTRONARVUTID
JA PROGRAMMEERIMINE

KIRJASTUS „VALGUS" · TALLINN 1966

Kaanekujundus *G. Pant*

SISSEJUHATUS

Kuni käesoleva sajandi alguseni olid teadlaste ja inseneride pingutused suunatud peamiselt ainult füüsilist tööd kergendavate masinate ja seadmete loomisele. Ajapikku kasvas lahendust vajavate probleemide hulk sedavõrd, et oldi sunnitud mõtlema ka mõtlemise enda „mehhaniseerimisele". Peaaegu kõik täppisteaduste ja tehnika saavutused on ühel või teisel määral seotud arvutustega ning just viimaste mehhaniseerimise vajadus kujunes selleks liikumapanevaks hoovaks, mis lõppkokkuvõttes viis elektronarvutite ilmumisele. Hiljem aga selgus, et arvutite kasutamisvõimalused on esialgselt ettenähtust tunduvalt laiemad. Nii näiteks on elektronarvuti suuteline mängima mitmesuguseid mänge, diagnoosima haigusi, pidama raamatukogus kartoteeki, reguleerima tänavaliiklust, juhtima keeruka tehnoloogilise protsessiga tootmist, hoidma lennukit ettemääratud kursil, viima orbiidile tehiskaaslasi ja ballistilisi rakette jne. jne.

Elektronarvutite osatähtsust on raske ülehinnata. Sooritades sadu tuhandeid ja isegi miljoneid operatsioone sekundis, vabastavad masinad tohutu hulga inimesi tüütutest mehhaanilistest arvutustest, see aga moodustab tööjõu, mille võib rakendada uute probleemide püstitamisele ja uurimisele. Elektronarvutite loomisega kaasnes mitmete täiesti uute teaduslik-tehniliste uurimissuundade tekkimine. Teadlastele ja inseneridele avanesid hindamatud võimalused paljude seni praktiliselt teostamatuks osutunud ülesannete lahendamiseks.

Elektronarvutil on kindel koht ka mitmesuguste igapäevase elu konkreetsete küsimuste lahendamisel. Neid on hakatud rakendama rahvamajandusharude vajaduste ehk nn. rahvamajanduse bilansi koostamiseks, tehase tootmisprotsessi kalendriliseks planeerimiseks, põllumajandusettevõtete külviplaanide koostamiseks, autotranspordi õigeks organiseerimiseks, mitmesuguste raamatupidamistööde mehhaniseerimiseks, ettevõtte majandusliku tegevuse analüüsiks ja väga paljuks muuks. Kahtlemata oleks elektronarvutite rakendamine senisest palju tõhusam, kui vastavate elualade töötajad ise oskaksid oma tööle vaadata arvutustehnika võimalusi tundva inimese pilguga.

Seoses arvutite kasutamisega on paratamatult tekkinud palju vaidlusi ka selle üle, milleks arvutusmasin on suuteline. Sageli suhtutakse elektronarvuti võimetesse üsna skeptiliselt, nendesse pole nagu usku. Esineb ka vastupidiseid seisukohti: arvatakse, et elektronarvuti on kõigeks suuteline. Elektronarvuti võib tõepoolest palju, kuid sugugi mitte kõike. Nende küsimuste lahendamisel peame alati jääma materialistideks. Igas konkreetses olukorras on tarvis õigesti hinnata „jõudude vahekorda", s. t. teha kindlaks olemasoleva arvutustehnika võimalused antud ülesande lahendamiseks ja alles siis otsustada.

Käesolevas brošüüris esitatakse lugejale algteadmisi elektronarvuti tööpõhimõtetest ja tutvustatakse ühte konkreetset arvutit. See annab laiemale lugejaskonnale kas või minimaalsed võimalused kaasa aidata eelnevalt püstitatud küsimuste lahendamiseks. Raamatu lugemine võib esialgu tunduda raskena, sest materjal esitatakse küllaltki konspektiivses laadis ja seetõttu tuleb olla eriti tähelepanelik. Piiratud mahu tõttu jääb nii mõnegi probleemi käsitlus pealiskaudseks, kuid nõudlikuma lugeja rahuldamiseks viitame erialasele kirjandusele.

Elektronarvuti on võrdlemisi komplitseeritud elektroonikaseade. Sellele vaatamata ei ole arvutusprogrammide koostamiseks tarvilik teada peaaegu ühtegi elektroonika põhitõde ja et raamat on määratud eeskätt just elementaarsete programmeerimisvõtete demonstreerimiseks, siis jätame kõrvale igasugused elektroonikaalased aspektid. Elektronarvuti tööpõhimõtetega tutvume niisiis mitte elektrooniku, vaid matemaatiku-programmeerija seisukohalt. Kõneldes edaspidi mitmesugustest arvuti põhisõl-

medest ja -seadmetest, räägitakse ainult nende ülesannetest ja funktsioonidest, mitte aga konkreetsest ehitusest.

Tänapäeval toodetavaid elektronarvutite tüüpe loendatakse sadades, seetõttu on võrdlemisi raske anda programmeerimise üldisi aluseid. Pealegi kujuneks üldine teooria liiga abstraktseks ja sellest arusaamine nõuaks eriettevalmistust. Käesolevas raamatus valiti arvutiteperest välja üks konkreetne elektronarvuti, „Uraal-1", ja arvestati materjali esitamisel peamiselt selle arvuti tehnilisi võimalusi. Abstraktsioonidesse laskutakse ainult sedavõrd, kui see tuleb abiks probleemistiku käsitluse lihtsustamisele ja silmaringi üldisele laiendamisele.

ELEKTRONARVUTITES KASUTATAVAD ARVUSÜSTEEMID

Elektronarvutite tundmaõppimist tuleb paratamatult alustada mõningate kümnendsüsteemist erinevate arvutussüsteemide käsitlemisega. Nendeks on kaheksand- ja kahendsüsteem. Kaheksandsüsteemi kasutatakse arvutusprogrammide koostamisel, arvuti ise aga sooritab tehteid ainult kahendarvudega. Tunduvalt vähemal määral kasutatakse kolmend- ja kuueteistkümnendsüsteemi.

Arvusüsteemid jaotatakse **positsioonilisteks** ja **mittepositsioonilisteks.** Lugeja on kindlasti juba tuttav vähemalt ühe esindajaga kummastki klassist. Nendeks on tavaline kümnendsüsteem ja rooma numbrite süsteem.

Elektronarvutites kasutatakse ainult positsioonilisi arvusüsteeme.

1. Positsioonilise arvusüsteemi mõiste.

Positsioonilise arvusüsteemi mõiste selgitamisel tugineme kümnendsüsteemile.

Kümnendsüsteemis on arvude üleskirjutamiseks kasutusel kümme erinevat numbrimärki: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, mis tähistavad kümmet esimest järjestikust täisarvu (alates nullist ja lõpetades üheksaga). Arvu kümme nimetatakse kümnendsüsteemi aluseks ja see kirjutatakse üles kahe numbrimärgi kombinatsioonina kujul 10. Üldiselt alati **positsioonilises arvusüsteemis kasutatavate erinevate numbrimärkide arvu nimetatakse arvusüsteemi aluseks.**

Kõik kümnendsüsteemi *arvud* esitatakse süsteemi alust moodustavate numbrimärkide jadadena. Siinjuures jaotatakse arv koma abil kaheks osaks: esimest nendest (komast vasakul) nimetatakse arvu *täisosaks* ja teist (komast paremal) *murdosaks*. Näiteks arvu 1211,121 korral on täisosaks 1211 ja murdosaks 0,121 (sada kakskümmend üks tuhandendikku).

On kerge veenduda, et

$$1211{,}121 = 1 \cdot 1000 + 2 \cdot 100 + 1 \cdot 10 + 1 + {}^{1}/_{10} + {}^{2}/_{100} + {}^{1}/_{1000} = 1 \cdot 10^3 + 2 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2} + 1 \cdot 10^{-3}.$$

Viimati toodu eeskujul võib esitada mistahes kümnendarvu.

Näiteks

$$8051{,}694 = 8 \cdot 10^3 + 5 \cdot 10^1 + 1 \cdot 10^0 + 6 \cdot 10^{-1} + 9 \cdot 10^{-2} + 4 \cdot 10^{-3}.$$

Eelnevate näidete varal selgub, et **positsioonilises arvusüsteemis sõltub numbri väärtus tema positsioonist (asukohast) arvus.** Arvu 1211,121 puhul täisosa esimesele kohale (lugedes vasakult paremale) kirjutatud üks tähendab tegelikult tuhandet; kolmandal kohal olev üks aga kümmet ja neljandal vastavalt ühte ühelist. Samuti pärast koma — esimene üks vastab ühele kümnendikule, viimane aga ühele tuhandendikule. Näeme, et **kahest kõrvuti olevast samast numbrimärgist on vasakul seisev parempoolsest kümme korda suurema väärtusega.**

Kirjeldatud reeglid ei kehti mittepositsioonilistes arvusüsteemides. Näiteks arvu 196 märgib rooma numbrite süsteemis sümbolite jada CXCVI = C + XC + VI. Esimesel kohal olev C tähistab siin sada; kolmandal kohal asuvat sümbolit C tuleb aga vaadelda koos teise sümboliga X (kümme), mis kokku moodustavad arvu üheksakümmend (XC).

Kümnendsüsteem ei ole ainsaks positsiooniliseks arvusüsteemiks, ta on vaid üks võimalikest. Saame täiesti üldiselt kõnelda mistahes p-ndsüsteemist. Niisiis, p-ndsüsteemi aluseks on p erinevat numbrimärki, millisteks tavaliselt valitakse p esimest järjestikust täisarvu tähistavad numbrimärgid (alates nullist ja lõpetades arvuga p — 1). Iga arv selles süsteemis esitatakse süsteemi alust moodustavate sümbolite jadana, mis jaotatakse koma abil täis- ja murdosaks. Jadas kahest kõrvuti olevast samast numbri-

märgist on vasakul seisev parempoolsest p korda suurema väärtusega. Süsteemi alus p on kirjutatud ise kujul 10.

Kui näiteks tähed $a_n, a_{n-1}, \ldots, a_0, a_{-1}, \ldots, a_{-m}$ tähistavad p-ndsüsteemis numbreid, siis jada $a_n a_{n-1} \ldots a_0, a_{-1} a_{-2} \ldots a_{-m}$ tähistab arvu $a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + \ldots + a_{-m} \cdot 10^{-m}$ (10 märgib süsteemi alust p). Selleks et saada p-ndsüsteemis kirjutatud arvule vastet kümnendsüsteemis, tuleb viimases summas kõikide liidetavate p-ndsüsteemi numbrimärgid asendada nendele vastavate kümnendnumbritega (kaasa arvatud ka p) ja sooritada ettenähtud tehted kümnendsüsteemis. Leitud summa on p-ndsüsteemi arvu kümnendvaste. Konkreetse näite öeldu kohta leiame järgmisest punktist.

2. Kaheksandsüsteem.

Kaheksandsüsteemi aluse moodustavad kaheksa erinevat numbrimärki, millisteks valitakse esimest kaheksat järjestikust täisarvu tähistavad kümnendsüsteemi numbrid 0, 1, 2, 3, 4, 5, 6, 7. Arvusüsteemi alus kaheksa kirjutatakse kujul 10. Kõik ülejäänud arvud esitatakse jadade abil, mis komaga jaotatakse täis- ja murdosaks. Näiteks kümnendarvul 274 on kaheksandsüsteemis kuju $422 = 4 \cdot 10^2 + 2 \cdot 10^1 + 2 \cdot 10^0$ (siin kümme tähistab kaheksat). Et kontrollida arvu kaheksandsüsteemi üleskirjutuse õigsust, teisendatakse ta ümber kümnendsüsteemi. Selleks peab eelneva võrduse paremal pool olevad kaheksandnumbrid asendama nendele vastavate kümnendnumbritega (kaasa arvatud ka süsteemi alus 8) ja teostama vajalikud korrutamised ning liitmised kümnendsüsteemis.

$$422_{(8)} = 4 \cdot 10^2 + 2 \cdot 10^1 + 2 \cdot 10^0 = 4 \cdot 8^2 + 2 \cdot 8^1 + 2 \cdot 8^0 =$$
$$= 4 \cdot 64 + 2 \cdot 8 + 2 = 274_{(10)}.*$$

Edaspidise käsitluse huvides tutvume põgusalt ka kaheksandaritmeetikaga. Et esialgu ei olda harjunud sooritama aritmeetikatehteid vahetult kaheksandsüsteemis, siis tuleb kasutada tabeleid.

Kahe kaheksandsüsteemi arvu liitmiseks (korrutamiseks) valitakse liitmise (korrutamise) tabelist esimesele liidetavale (tegurile) vastav rida ja teisele liidetavale (tegurile) vastav veerg ning seejärel nende summa (korrutis)

* Indeks all sulgudes näitab arvusüsteemi.

+	0	1	2	3	4	5	6	7	10
0	0	1	2	3	4	5	6	7	10
1	1	2	3	4	5	6	7	10	11
2	2	3	4	5	6	7	10	11	12
3	3	4	5	6	7	10	11	12	13
4	4	5	6	7	10	11	12	13	14
5	5	6	7	10	11	12	13	14	15
6	6	7	10	11	12	13	14	15	16
7	7	10	11	12	13	14	15	16	17
10	10	11	12	13	14	15	16	17	20

loetakse valitud rea ja veeru ristumiskohalt. Näiteks $4 + 5 = 11$ või $4 \cdot 5 = 24$. Mõlemad vastused leitakse neljanda rea ja viienda veeru ristumiskohalt (või viienda rea ja neljanda veeru ristumiskohalt).

Tuginedes konkreetsetele näidetele, selgitame tabelite kasutamist ka kaheksandarvude lahutamiseks ja jagamiseks. Võrrandi $x = 15 - 7$ lahendamiseks leiame liitmise tabeli seitsmendast reast (veerust) arvu 15. Näeme, et ta asub kuuele vastava veeru (rea) lõikepunktis seitsmenda reaga (veeruga). Järelikult $7 + 6 = 15$ ehk $x = 6$. Kui $x = 7 - 15$, siis ilmselt $x = -(15 - 7)$ ning viimasest $x = -6$. Lahutamisega sarnaselt kasutatakse korrutustabelit jagamiseks. Näiteks võrrandi $x = 36 : 6$ korral, talitades lahutamiseks antud õpetuse kohaselt, veendume, et $x = 5$. Mitmekohaliste kaheksandarvudega sooritatakse aritmeetikatehteid niisamuti nagu vastavaid tehteid kümnendsüsteemis.

Korrutamine.

×	0	1	2	3	4	5	6	7	10
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	10
2	0	2	4	6	10	12	14	16	20
3	0	3	6	11	14	17	22	25	30
4	0	4	10	14	20	24	30	34	40
5	0	5	12	17	24	31	36	43	50
6	0	6	14	22	30	36	44	52	60
7	0	7	16	25	34	43	52	61	70
10	0	10	20	30	40	50	60	70	100

Liitmine

$$\begin{array}{r} 675{,}62105 \\ +\ 23{,}70063 \\ \hline 721{,}52170 \end{array}$$

Lahutamine

$$\begin{array}{r} 1475{,}6003 \\ -\ 274{,}7056 \\ \hline 1200{,}6725 \end{array}$$

Korrutamine

$$\begin{array}{r} 264{,}571 \times 15{,}21 \\ \hline 264571 \\ 551362 \\ 1607535 \\ 264571 \\ \hline 4535{,}45111 \end{array}$$

Jagamine

$$\begin{array}{r|l} 264575 & 1521 \\ 1521 & 155 \\ \hline 11247 & \\ 10225 & \\ \hline 10225 & \\ 10225 & \\ \hline 0 & \end{array}$$

3. Kahendsüsteem.

Valides arvusüsteemi aluseks ainult kaks erinevat numbrimärki 0 ja 1, saadakse kahendsüsteem. Süsteemi aluse, arvu kaks üleskirjutamiseks kasutatakse kombinatsiooni 10. Kõik teised arvud esitatakse sümbolite 0 ja 1 jadadena, mis on koma abil jaotatud täis- ja murdosaks. Näiteks kümnendarv 274 omab kahendsüsteemis kuju

$$100010010_{(2)} = 1 \cdot 10^{1000} + 1 \cdot 10^{100} + 1 \cdot 10^{1} = \\ = 1 \cdot 2^{8} + 1 \cdot 2^{4} + 1 \cdot 2^{1} = 274_{(10)}.$$

Eriti lihtsaks osutuvad kahendsüsteemi aritmeetikatehete tabelid.

Liitmine	*Lahutamine*	*Korrutamine*
$0 + 0 = 0$	$0 - 0 = 0$	$0 \times 0 = 0$
$0 + 1 = 1$	$1 - 0 = 1$	$0 \times 1 = 0$
$1 + 0 = 1$	$1 - 1 = 0$	$1 \times 0 = 0$
$1 + 1 = 10$	$10 - 1 = 1$	$1 \times 1 = 1$

Mitmekohaliste kahendarvudega sooritatakse aritmeetikatehteid sarnaselt kümnendaritmeetikaga (vt. [1] või [6]).

4. Üleminek ühest arvusüsteemist teise.

Vaatame kõigepealt kümnendsüsteemi arvu teisendamist kaheksandsüsteemi arvuks. Sõltumata arvusüsteemist jaotati arv koma abil täis- ja murdosaks. Üleminekul teisendatakse neid eraldi. Terviklikule kümnendarvule vastab kaheksandsüsteemi viidud täis- ja murdosade summa.

Kümnendarvu täisosa kui täisarvu teisendamiseks kaheksandsüsteemi jagatakse ta kümnendsüsteemis kaheksaga. Tulemuseks on jagatis ja jääk (nimetame selle esimeseks jäägiks ja peame meeles). Juhul kui esimesel jagamisel saadud jagatis on kaheksast suurem, siis jagame edasi. Saadakse omakorda jagatis ja jääk (nimetame teiseks jäägiks). Sellist jagatise jagamist kaheksaga ja järjekorras tekkivate jääkide meelespidamist tehakse senikaua, kuni jagatis saab väiksemaks kaheksast. Viimast jagatist nimetame viimaseks jäägiks. Nüüd, kirjutades jagamisel leitud jäägid nende leidmisele vastupidises järjekorras (kõige enne viimane ja kõige viimasena esimene), saadakse kümnendarvu täisosale vastav kaheksandarv.

```
274 | 8
24  | 34 | 8
--- | --- | ---
34  | 32 | 4
32  |  2 | III jääk
--- | ---
 2  | II jääk
I jääk
```

$274_{(10)} = 422_{(8)}$.

```
10759 | 8
----- | ---- | ---- | ---- | ---
  8   | 1344 | 8
----- | ---- | ---- | ---- | ---
 27   |  8   | 168  | 8
 24   | 54   | 16   | 21   | 8
----- | ---- | ---- | ---- | ---
 35   | 48   |  8   | 16   | 2
 32   | 64   |  8   |  5
----- | ---- | ---- | ---- | ---
 39   | 64   |  0
 32   |  0
----- | ----
  7
```

$10759_{(10)} = 25007_{(8)}$.

Arvu murdosa kui kümnendmurru teisendamiseks kaheksandsüsteemi korrutatakse ta kõigepealt kaheksaga. Saadud korrutis koosneb täisosast (mis võib võrduda ka nulliga) ja murdosast. Järgnevalt eraldatakse täisosa murdosast ja peetakse meeles. Korrutise murdosa korrutatakse uuesti kaheksaga ning eraldatakse uuesti täisosa murdosast. Sellist protsessi võib jätkata senikaua, kuni korrutise murdosaks ei tule null. Kümnendmurru kaheksandvasteks on kaheksandmurd, mille tüvenumbriteks on korrutamisel leitud täisosad täpselt samas järjekorras, nagu nad tekkisid. Lahtiseks jääb ainult küsimus, millal lõpetada kaheksaga korrutamine, s. t. mitu kaheksandsüsteemi tüvenumbrit peab leidma antud kümnendmurru õigeks kujutamiseks kaheksandsüsteemis. Võib kasutada reeglit: kui kümnendmurrus on k tüvenumbrit, siis kaheksandmurrus peab neid olema vähemalt $k + 1$ (s. t. ühe võrra rohkem).

```
0 | 54
  | ×8
--- | ---
4 | 32
  | ×8
--- | ---
2 | 56
  | ×8
--- | ---
4 | 48
```

$0{,}54_{(10)} = 0{,}424_{(8)}$

```
0 | 0761
  | ×8
--- | ---
0 | 6088
  | ×8
--- | ---
4 | 8704
  | ×8
--- | ---
6 | 9632
  | ×8
--- | ---
7 | 7056
  | ×8
--- | ---
5 | 6454
```

$0{,}0761_{(10)} = 0{,}04675_{(8)}$

Toodud arvude varal on kerge veenduda, et

$$10759{,}54_{(10)} = 25007{,}424_{(8)},$$
$$274{,}0761_{(10)} = 422{,}04675_{(8)}.$$

Kümnendarvu üleviimine kahendsüsteemi toimub sarnaselt üleminekuga kaheksandsüsteemi. Erinevus algoritmides on ainult arvu kaheksa asendamises arvuga kaks. Kümnendmurru teisendamisel kahendmurruks peame viimases leidma vähemalt 3 (k + 1) kahendkohta.

274	137	68	34	17	8	4	2	1
2	12	6	2	16	8	4	2	
7	1	8	14	1	0	0	0	
6		0	14					
14			0					
14								
0								

$$274_{(10)} = 100010010_{(2)}$$

10759	5379	2689	1344	672	336	168	84	42	21	10	5	2	1
10758	5378	2688	1344	672	336	168	84	42	20	10	4	2	
1	1	1	0	0	0	0	0	0	1	0	1	0	

$$10759_{(10)} = 10101000000111_{(2)}$$

42	21	10	5	2	1
42	20	10	4	2	
0	1	0	1	0	

$$42_{(10)} = 01010_{(2)}$$

$$0{,}54_{(10)} = 0{,}100010100_{(2)}$$
$$0{,}0761_{(10)} = 0{,}000100110111101_{(2)}$$

0	54
1	08
0	16
0	32
0	64
1	28
0	56
1	12
0	24
0	48

0	0761
0	1522
0	3044
0	6088
1	2176
0	4352
0	8704
1	7408
1	4816
0	9632
1	9264
1	8528
1	7056
1	4112
0	8224
1	6448

Tabelis 1 antakse kümnendsüsteemi esimesele viiekümnele täisarvule vastavad kaheksand- ja kahendarvud. Viimasest selgub, et kolmekohalise kahendarvu abil on võimalik üles kirjutada iga kaheksandsüsteemi number.

0 — 000	4 — 100
1 — 001	5 — 101
2 — 010	6 — 110
3 — 011	7 — 111

Kui kaheksandsüsteemis esitatud arvu iga number asendada kolmekohalise kahendarvuga, siis saadud nullide ja ühtede jada annab sama arvu kahendsüsteemi üleskirjutuse.

$$25007{,}424_{(8)} = \underset{2}{010}\ \underset{5}{101}\ \underset{0}{000}\ \underset{0}{000}\ \underset{7}{111}\ ,\ \underset{4}{100}\ \underset{2}{010}\ \underset{4}{100}{}_{(2)}$$

Kehtib ka vastupidine seos. Kui kahendarv jaotada kolmikuteks, kusjuures täisosa jaotatakse komast vasakule, murdosa aga komast paremale liikudes, ning moodustunud kolmikud asendada nendele vastavate kaheksandnumbritega, saame kahendarvule vastava kaheksandarvu.

$$\underset{4}{100}'\underset{2}{010}'\underset{4}{100},\underset{0}{000}'\underset{4}{100}'\underset{6}{110}'\underset{7}{111}'\underset{5}{101}'_{(2)} = 424{,}04675_{(8)}$$

Eeskirjad, mille abil kümnendsüsteemi arv viiakse üle kaheksand- või kahendsüsteemi, on matemaatiliselt rangelt põhjendatavad ja rakendatavad mistahes arvusüsteemist üleminekuks mõnda teise arvusüsteemi. Kui näiteks on

tarvis p-ndsüsteemi täisarv teisendada q-ndsüsteemi arvuks ($p \neq q$), siis peab p-ndsüsteemi arvu jagama p-ndsüsteemis q-ga. Jagamisel tekkinud jäägid, üleskirjutatult vastupidises järjekorras, moodustavad p-ndsüsteemi arvule vastava q-ndsüsteemi arvu (vaata [1] või [6]).

Tabel 1

10-nd	8-nd	2-nd	10-nd	8-nd	2-nd	10-nd	8-nd	2-nd
1	1	1	18	22	10 010	35	43	100 011
2	2	10	19	23	10 011	36	44	100 100
3	3	11	20	24	10 100	37	45	100 101
4	4	100	21	25	10 101	38	46	100 110
5	5	101	22	26	10 110	39	47	100 111
6	6	110	23	27	10 111	40	50	101 000
7	7	111	24	30	11 000	41	51	101 001
8	10	1 000	25	31	11 001	42	52	101 010
9	11	1 001	26	32	11 010	43	53	101 011
10	12	1 010	27	33	11 011	44	54	101 100
11	13	1 011	28	34	11 100	45	55	101 101
12	14	1 100	29	35	11 101	46	56	101 110
13	15	1 101	30	36	11 110	47	57	101 111
14	16	1 110	31	37	11 111	48	60	110 000
15	17	1 111	32	40	100 000	49	61	110 001
16	20	10 000	33	41	100 001	50	62	110 010
17	21	10 001	34	42	100 010			

ELEKTRONARVUTI TÖÖTAMISE ÜLDISED PÕHIMÕTTED

Elektronarvutite klassifitseerimise olulisemaks aluseks on nende töötamispõhimõte, mille kohaselt nad liigitatakse kahte suurde rühma:

1) pideva toimega arvutid,
2) diskreetse toimega arvutid.

Diskreetse toimega elektronarvutid liigitatakse lähtudes

1) arvude kujutamise viisist ja
2) käsus sisalduvate aadresside arvust.

Nendest esimese järgi eristatakse fikseeritud ja liikuva komaga arvuteid ning arvuteid, milles võib arve kujutada nii fikseeritud kui ka liikuva komaga.

Käsus sisalduvate aadresside alusel on peamisteks esindajateks ühe-, kahe-, kolme-, nelja- ja isegi viieaadressilised arvutid. Peale ülalloetletute esineb veel poolteisening kahe ja pooleaadressilisi arvuteid.

1. Pideva toimega arvutid.

Pideva toimega arvutites kujutatakse matemaatilisi suurusi (arve) mitmesuguste füüsikaliste suuruste (nurgad, pikkused, pinged) abil, mis lahendusprotsessis muutuvad sujuvalt, ilma hüpeteta. Selle klassi üheks lihtsamaks esindajaks on arvutuslükati (pidevalt muutuvaks suuruseks on lõigu pikkus).

Pideva toimega arvutid on enamuses ette nähtud kas teatava ülesannete klassi (näiteks kindlat tüüpi võrrandi-

süsteemide) või isegi ainult ühe konkreetse ülesande lahendamiseks. Seetõttu on arvutite kasutamisvõimalused mõnevõrra piiratud. Teiselt poolt toimub aga nende ülesannete lahendamine, milleks pideva toimega arvuti on ehitatud, tuhandeid kordi kiiremini kui sellesama ülesande lahendamine universaalsel diskreetse toimega elektronarvutil.

Pideva toimega arvutite üheks puuduseks on arvutustulemuste suhteliselt väike täpsus. Viimane sõltub oluliselt arvuti ehitamiseks kasutatud raadiodetailide (takistid, kondensaatorid, induktiivsused) täpsusklassist.

Sageli koosneb pideva toimega arvuti reast blokkidest, millest igaüks sooritab ühte kindlat matemaatilist operatsiooni (liidab, korrutab, integreerib, diferentseerib jne.). Ülesande lahendamiseks ühendatakse blokid vajalikus järjekorras ja vastus saadakse siis juba momentaanselt.

2. Diskreetse toimega arvutid.

Diskreetse toimega arvuteid nimetatakse ka numbrilisteks arvutiteks. Arvude kujutamiseks nendes kasutatakse elemente, millel on kindel hulk üksteisest rangelt erinevaid seisundeid. Näiteks aritmomeetrites on sellisteks elementideks kümne hambaga hammasrattad, milliseid võetakse niimitu, kui see parasjagu vajalikuks osutub.

Elektronarvutites, mis töötavad peamiselt kahendsüsteemis, on kasutusel kahte erinevat seisundit omavad elemendid. Arvutuste täpsuse tõstmiseks suurendatakse arvu kujutamiseks vajalike elementide hulka, kusjuures nende elementide eneste valmistamiseks kasutatavate raadiodetailide suur täpsus ei ole eriti oluline.

Diskreetse toimega arvutid on universaalsed, s. t. nad sobivad mistahes matemaatilise ülesande lahendamiseks. Veelgi enam, elektronarvutil on lahendatavad ka algoritmeeritavad ülesanded*.

Kõigis elektronarvutites on olemas **mäluseade.** Selles salvestatakse ülesande lahendamiseks vajalik informatsioon, s. o. lahendusprogramm ja tarvilikud lähte- ning vahetulemused. Mälu jaotatakse teatavaks arvuks osa-

* Algoritm on teataval viisil esitatud eeskirjade kogum, mille järgi tegutsedes jõutakse soovitud eesmärgile.

deks, millest iga üksikut nimetatakse **pesaks.** Näiteks elektronarvuti „Uraal-1" mälu on jaotatud 2048 osaks ehk, teisiti öeldes, tema mälus on 2048 pesa. Pesade arvu nimetatakse sageli ka **mälu mahuks.** Mälupesad nummerdatakse ja vastavat pesa numbrit nimetatakse tema **aadressiks.** Aadressideks valitakse kaheksandsüsteemi arvud ning 2048 pesa korral on nendeks kaheksandarvud 0 kuni 3777 ($2047_{(10)} = 3777_{(8)}$).

Mälupessa kirjutatakse arv üles kahendsüsteemis. Nimelt jaguneb pesa osadeks ehk nn. **kahendkohtadeks.** Igas kahendkohas on võimalik kujutada üks kahendnumber, s. t. kas 0 või 1. Ühte pesa moodustavate kahendkohtade arv, mida nimetatakse ka pesa pikkuseks, sõltub arvuti tüübist. Elektronarvuti „Uraal-1" pesa koosneb kaheksateistkümnest kahendkohast. Järelikult mahutab tema pesa maksimaalselt kaheksateistkümnekohalise kahendarvu.

Elektronarvuti lahendab ülesannet inimese koostatud eeskirjade kogumi, nn. **programmi** järgi. Programm on ülesande lahendusalgoritmi eriline kuju, ta on masinale arusaadavas keeles (arvudes) üleskirjutatud algoritm. Programm koosneb **käskudest.** Käsk on informatsioon, mis määrab masina töö teatava aja vältel. Käsk omakorda koosneb **tehtekoodist** ja kas ühest või enamast aadressist. Elektronarvuti arvutusoskus piirdub aritmeetika esimese nelja tehtega, millele lisanduvad veel mõningad loogilist laadi operatsioonid (nagu arvude võrdlemine) ja operatsioonid, mis on vajalikud informatsiooni vahetamiseks ning arvutusprotsessi automaatseks juhtimiseks. Igale operatsioonile (tehtele) seatakse vastavusse kindel kahekohaline kaheksandarv, mida nimetatakse operatsioonikoodiks (ehk tehtekoodiks). Alljärgnevalt tuuakse näitena kolmeaadressilise arvuti käsk (elektronarvuti „Strela"):

1375	1654	2572	01
I aadress	II aadress	III aadress	tehtekood

Sellise käsu toimel arvuti liidab pesades (aadressidega) 1375 ja 1654 salvestatud arvud ja summa kirjutatakse pessa 2572.

Elektronarvuti „Uraal-1" on üheaadressiline arvuti. Tema käsk koosneb tehtekoodist ja ainult ühest aadressist:

07	1564
tehtekood	aadress

Arvuti enamike käskude toimel sooritatakse tehe arvudega, millest üks asub käsus näidatud aadressiga pesas ja teine **summaatoris.** Pärast tehte sooritamist jääb tulemus jällegi summaatorisse. Summaator on spetsiaalne seade, milles võib salvestada ühe arvu ja mis sealjuures võtab osa tehte sooritamisest. Näitena esitame kolmekäsulise programmilõigu:

I käsk	02 1654
II käsk	01 1754
III käsk	16 2037.

I *käsk:* „Pesas 1654 salvestatud arv tuua summaatorisse!"

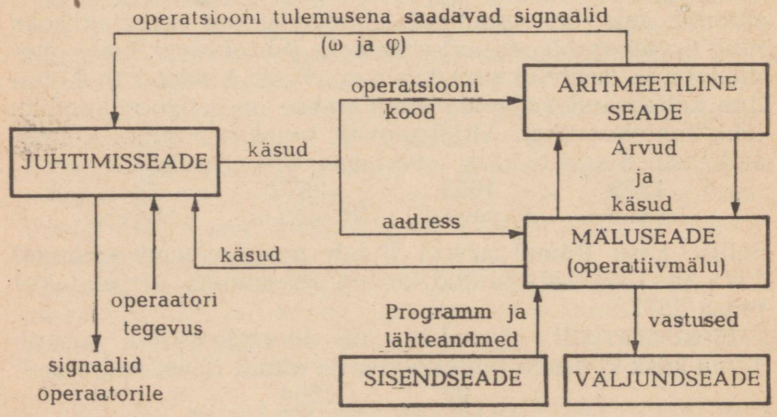
II *käsk:* „Summaatoris olevale arvule liita pesas 1754 salvestatud arv!"

III *käsk:* „Summaatoris olev arv kirjutada mälupessa aadressiga 2037!"

Nende kolme käsuga liidetakse kokku arvud pesadest 1654 ja 1754 ning nende summa salvestatakse pessa aadressiga 2037.

3. Elektronarvuti ehituse üldskeem.

1) aritmeetiline seade,
2) juhtimisseade,
3) mäluseade,
4) sisendseade,
5) väljundseade.



Joon. 1.

Joonisel 1 on kujutatud elektronarvuti lihtsustatud skeem ja näidatud mainitud viie põhisõlme omavahelised seosed.

Aritmeetilise seadme ülesandeks on sooritada aritmeetika- ja loogilisi tehteid nii arvude kui ka käskudega. Sooritatav operatsioon määratakse käsu tehtekoodiga. Peale selle töötab aritmeetiline seade iga operatsiooni korral välja spetsiaalsed signaalid (ω ja φ) ning saadab nad juhtimisseadmele.

Juhtimisseadme ülesandeks on garanteerida ülesande automaatne lahendamine ja arvuti ülejäänud põhisõlmede kooskõlastatud töö. Tegevuse aluseks võtab juhtimisseade lahendusprogrammi ja aritmeetilise seadme poolt välja töötatud signaalid. Juhtimisseade võimaldab ka inimesel (operaatoril) sekkuda automaatsesse lahenduskäiku.

Mäluseade on määratud ülesande lahendamiseks vajaliku informatsiooni salvestamiseks. Mäluseadmele püstitatakse kaks nõuet:

1) ta peab olema küllalt suure mahuga ja
2) võimaldama kiiresti leida vajalikku informatsiooni.

Osutub, et need kaks nõuet on kaasaja elektrotehnika taseme juures vastuolulised. Ei ole õnnestunud ehitada mälu, mis mahutaks palju ja samal ajal töötaks kiiresti. Seetõttu on arvutitel kaht liiki mäluseadmeid:

1) *sisemälu e. operatiivmälu* — võrdlemisi piiratud mahtuvusega, kuid suure töökiirusega;
2) *välismälu* — praktiliselt peaaegu piiramatu mahuga, kuid töötab aeglaselt.

Sisendseade teisendab informatsiooni masinale „suupärasesse" kujju (elektriimpulssideks) ja kirjutab selle mäluseadmesse. Kogu elektronarvutisse viidav informatsioon perforeeritakse vastavate lisaseadmete abil perfolindile või -kaartidele (olenevalt masina sisendseadme konstruktsioonist) teatavate aukude kombinatsioonidena. Sisendseade teisendab „augud" elektriimpulssideks (täpsemalt, nende jadadeks), mida arvuti dešifreerida oskab.

Väljundseadme ülesandeks (vastupidiselt sisendseadmele) on viia arvutis saadud elektriimpulsid, mis tegelikult kujutavad ülesande vastuseid, meie (s. t. inimese) jaoks loetavasse kujju. Väljundseadmeks on tavaliselt trükkimisseade, mis trükib vastused paberilindile arvudena. Sageli aga perforeeritakse ka vastused, olgu siis perfolindile või -kaartidele.

4. Arvude kujutamine fikseeritud komaga arvutis.

Fikseeritud komaga arvutis kujutatakse arve nende tavalisel kujul, s. t. arv esitatakse kahendnumbrite jadana, mis koma abil jaotatakse täis- ja murdosaks. Mälu pesas on arvu ülesmärkimiseks kasutada kindel arv kahendkohti. Need jaotatakse arvu moodustava kolme komponendi vahel: arvu märk, arvu täisosa ja arvu murdosa. Koma pesas ei märgita, vaid arvuti ehitatakse selliselt, et tehte teostamisel arvestab ta koma fikseeritud asukohta. Arvu märgi kujutamiseks reserveeritakse üks kahendkoht. Tavaline märgi esitamise viis on järgmine: „—" kujutatakse märgile ettenähtud kahendkohas numbrina 1 ja „+" vastavalt 0. Näiteks mälupesa, milles arvu kujutamiseks on kasutada 13 kahendkohta, võib olla jaotatud järgmiselt: arvu märk paikneb 0-ndas kahendkohas, arvu täisosa kujutamiseks on kahendkohad 1—4 ja ülejäänud kaheksa (5—12) kahendkohta kuuluvad arvu murdosale.

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	1	1	0	1	1	1	0	1	0	1
märk	täisosa				, murdosa							

Kirjeldatud mälupesas ei ole võimalik kujutada kahendarvust 1111,11111111 absoluutväärtuselt suuremat arvu (vastab kümnendarvule $(16 - 2^{-8})$), s. t. arvu täisosa ei tohi ületada kümnendarvu 15. Ülesande lahendamisel teostatavate aritmeetikatehete tulemuse täisosa võib aga kasvada suuremaks viieteistkümnest (liidetakse näiteks arvud 10 ja 7). Sellisel korral räägitakse, et tekkis ü l e t ä i t u m i n e. Ületäitunud arvud ei kujutu arvutis õigesti, vaid moonutatud kujul. Vaatame näitena arvude 10 ja 7 summat $(10 + 7 = 17_{(10)})$.

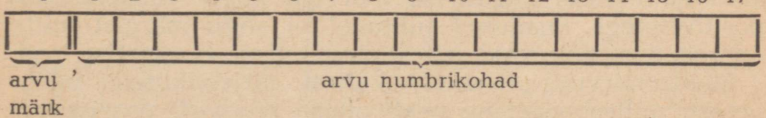
$$\begin{array}{r} 01010{,}00000000 \\ +\ 00111{,}00000000 \\ \hline 10001{,}00000000 \end{array}$$

Saadud tulemus näeb arvuti pesas välja nii:

0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	0	0	0	0	0	0	0	0
märk	täisosa				murdosa							

Ületäitunud kahendjärk kandub arvu märgi kohale ja õige tulemus +17 kujutub pesas nagu täisarv —1. Programmide koostamisel fikseeritud komaga arvutitele peab ületäitumisele pöörama erilist tähelepanu. Ülesande lahendamise programm koostatakse selliselt, et ületäitumist kas üldse ei teki (vastavate mastaapide kasutamise abil) või tehakse osa arvutusi uuesti, kusjuures automaatselt muudetakse lähtesuurusi. Elektronarvutitel on spetsiaalne elektroonikaseade, mis iga sooritatud tehte korral teeb kindlaks, kas ületäitumine esines või mitte. Vastavalt sellele töötatakse aritmeetilises seadmes välja signaal (tavaliselt tähistatakse tähega φ; ületäitumisel $\varphi = 1$), mille alusel saadakse programmis arvestada ületäitumist.

Enamikus fikseeritud komaga arvutites fikseeritakse koma koht vahetult pärast arvu märki. Seega on nendes võimalik kujutada ainult murdarve. Üheks niisuguseks on ka „Uraal-1". Tema pesa kaheksateist kahendkohta jaotatakse arvu kujutamiseks alljärgnevalt:



(„—" vastab 1; „+" vastab 0)

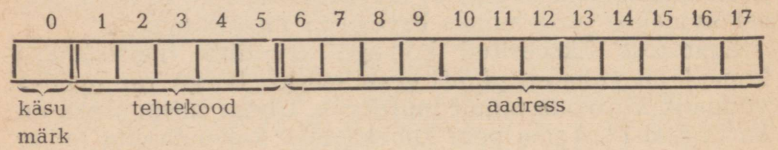
Absoluutselt suurimaks pesas kujutatavaks arvuks on kahendmurd 0,1111111111111111, mis vastab kümnendmurrule $(1 - 2^{-17})$. Absoluutväärtuselt vähimaks aga 0,00000000000000001 (kümnendsüsteemis 2^{-17}). Kõik arvud, mis on absoluutväärtuselt väiksemad kui 2^{-17}, kujutuvad pesas arvuna 0. Selliseid arve nimetatakse **arvuti nullideks.** Järelikult on need arvud, mis tegelikult nulliga ei võrdu, kuid oma suhtelise „väiksuse" tõttu ainult kujutuvad arvutis nullidena. Arvul null on kaks kujutist +0 ja —0. Elektronarvutis „Uraal-1" saadakse „—0" aritmeetikatehete tulemusena.

5. Käskude kujutamine arvutis.

Käsk on informatsioon, mida saab viia arvutisse ja mis määrab arvuti töö teatava ajavahemiku jooksul. Ülesande lahendamise programm kujutab endast kindlas järjekor-

ras üleskirjutatud käskude jada. Käsud, samuti nagu arvud, salvestatakse arvuti sisemälu pesades. Vastavalt arvuti konstruktsioonile on varutud kindel arv kahendkohti käsu tehtekoodi ja aadressi (või aadresside) kujutamiseks. Formaalselt võib iga arvutis kujutatud arvu vaadata kui käsku ja vastupidi — iga käsku võime vaadata kui arvu.

Elektronarvutis „Uraal-1" kasutatakse käsu kujutamiseks 18 kahendkohta, millest esimene (0-is koht) varutakse käsu märgile, järgnevad viis (1—5) määravad tehtekoodi ja ülejäänud 12 vastavalt käsu aadressi.

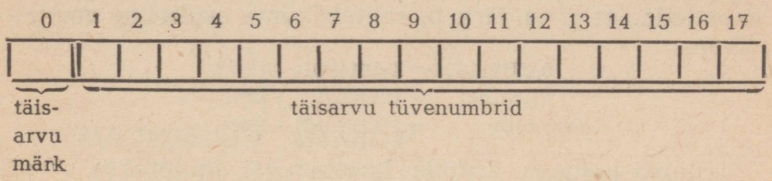


Käsu märgi abil varieeritakse sageli tehtekoodi. Mitte alati ei teostata negatiivse märgiga käske samaväärselt positiivsetega. Enamikul juhtudel ei sõltu märgist mitte operatsiooni sisu, vaid sellest oleneb tema teostamise laad.

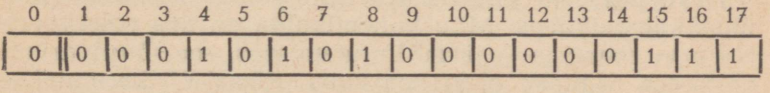
Elektronarvuti „Uraal-1" kahest järjestikusest mälupesast, millest esimene peab olema kindlasti paarisarvulise aadressiga, saab moodustada ühe nn. *pika pesa* (eelnevalt kirjeldatud kaheksateistkümnest kahendkohast koosnevaid mälupesi nimetatakse ka *lühikesteks* pesadeks). Pika pesa aadressiks võetakse temasse kuuluva esimese lühikese pesa nelja tuhande võrra suurendatud aadress. Näiteks pesadest 1754 ja 1755 saadakse pikk pesa aadressiga 5754 (s. t. 4000 + 1754), kusjuures pesa aadress on alati paarisarvuline. Üks pikk pesa koosneb kolmekümne kuuest kahendkohast. Nendest esimene (0-is koht) reserveeritakse märgi kujutamiseks ja ülejäänud 35 arvu tüvenumbritele. Pikki pesi on mälus 1024 ja neis kujutatakse ainult arve. Eriti oluliseks osutuvad nad väikeste arvude kujutamisel, sest siis on arvuti nullideks arvud, mis on väiksemad kümnendarvust 2^{-35}.

6. Täisarvude kujutamine arvutis.

Vaatleme alljärgnevalt täisarvude kujutamist elektronarvuti „Uraal-1" mälupesas. Teame, et viimane koosneb kaheksateistkümnest kahendkohast. Täisarvu korral kasutatakse esimene arvu märgi ja ülejäänud seitseteist kahendkohta — tüvenumbrite kujutamiseks.



Täisarvu kujutamiseks teisendatakse ta kahendsüsteemi ja paigutatakse mälupessa selliselt, et kahendesituse ühelised paikneksid pesa viimases kahendkohas. Näiteks $10759_{(10)} = 25007_{(8)} = 10101000000111_{(2)}$ näeb pesas välja järgmiselt:



Pesas kujutatavaks absoluutselt suurimaks täisarvuks on kahendarv 11111111111111111, millele vastab kümnendarv $131071 = 2^{17} - 1$.

Vaadates pesas salvestatud täisarvu kui fikseeritud komaga arvu, veendume, et ta kujutub oma tõelisest väärtusest 2^{17} korda väiksema arvuna: täisarv x kujutub arvutis samuti nagu murdarv $x \cdot 2^{-17}$ (veenduge, et täisarvu 10759 ja murdarvu 10759 : 131072 kujutised langevad ühte). Seetõttu öeldakse, et täisarve kujutatakse arvutis 2^{-17} ühikutes. Edaspidises tarvitataksegi niisugust terminoloogiat.

7. Arvude kujutamine liikuva komaga arvutis.

Fikseeritud komaga arvutite oluliseks puuduseks on nende mälupesades kujutatavate arvude suhteliselt väike diapasoon. Eriti komplitseerib see programmeerimist, sest peame arvestama ületäitumist ja nägema programmis ette võimalused selle likvideerimiseks. Programmeerimise hõl-

bustamiseks on ehitatud arvuteid, kus arve kujutatakse nn. liikuva koma režiimis. Arvu märkimise aluseks on seos

$$x = m_x \cdot 10^{p_x}, \qquad (1)$$

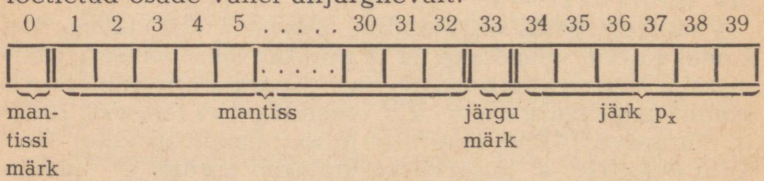
kus $0{,}1 \leqslant |m_x| < 1$ ja p_x on täisarv (sümbol 10 tähendab kahendsüsteemi alust kahte). Suurust m_x nimetatakse arvu **mantissiks** ning täisarvu p_x **järguks.** Arvu üleskirjutust kujul (1) nimetatakse arvu **normaalesituseks.** Kerge on veenduda, et arvu järk p_x näitab koma asukohta mantissis m_x.

$$0{,}001011101 = 0{,}1011101 \cdot 10^{-10}$$
$$0{,}0101001 = 0{,}101001 \cdot 10^{-1}$$
$$1111{,}0101 = 0{,}11110101 \cdot 10^{100}$$

Liikuva komaga arvude kujutamisel jaotatakse arvuti mälupesa kahendkohad nelja komponendi vahel:

1) arvu e. mantissi märk,
2) mantiss,
3) järgu märk,
4) järk.

Arvu märk ja järgu märk kujutatakse kumbki eraldi ühes kahendkohas. Mitu kahendkohta on aga reserveeritud arvu mantissi ja järgu kujutamiseks, see sõltub arvuti ehitusest. Näiteks elektronarvutis „Uraal-4" kasutatakse liikuva komaga arvu salvestamiseks 40-st kahendkohast koosnevat mälupesa, mille kahendkohad jagunevad ülalloetletud osade vahel alljärgnevalt:



Liikuva koma kasutamine tõstab tunduvalt arvutustulemuste täpsust, seda eriti absoluutväärtuselt ühest väiksemate arvude puhul. Liikuva komaga arvutites tekib ületäitumine siis ja ainult siis, kui arvu järk ületab lubatud piirid. Eelneva näite korral signaal $\varphi = 1$, kui $p_x > 63_{(10)}$, s. t. kui x on suurem kümnendarvust 2^{63}.

ELEKTRONARVUTI „URAAL-1"

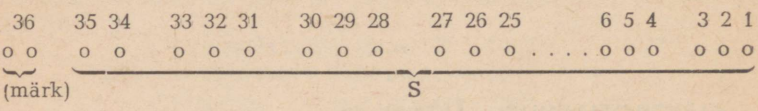
1. Üldiseloomustus ja kasutatav sümboolika.

Elektronarvuti „Uraal-1" on diskreetse toimega universaalne üheaadressiline fikseeritud komaga arvuti. Tema käskude süsteemi kuulub 29 erinevat operatsiooni, millest enamiku täitmiseks kulub $^1/_{100}$ sekundit. Seetõttu kõneldakse, et arvuti sooritab 100 operatsiooni sekundis. Informatsiooni sisseviimiseks arvutisse kasutatakse perfolinti, kusjuures arve loetakse fotoelektrilise seadme abil kiirusega 4500 arvu minutis. Elektronarvuti mäluseadmeks on magnettrummel. Viimane kujutab endast duralumiiniumist ketast, mille pealispind on kaetud spetsiaalse magnetilise materjaliga (ferriidikihiga). Magnettrummel pöörleb kiirusega 6000 p/min. (100 pööret sekundis). Välismäluks kasutatakse magnetlinti. Ühele lindile on võimalik salvestada maksimaalselt 80 000 kaheksateistkümnekohalist e. 40 000 kolmekümne kuue kohalist kahendarvu.

Elektronarvuti „Uraal-1" töötab kahendsüsteemis. Arvutusprogramm koostatakse kaheksandsüsteemis. Arvude üleviimine nii kaheksandsüsteemist kahendsüsteemi kui ka kahendsüsteemist kaheksandsüsteemi toimub automaatselt. Kümnendarvude teisendamiseks kahendsüsteemi ja kahendarvude teisendamiseks kümnendsüsteemi tuleb kasutada spetsiaalseid teisendusprogramme.

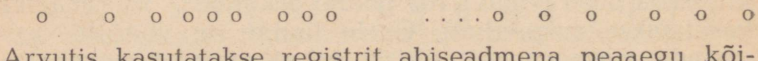
Summaator on nagu omalaadne pesa, mis koosneb 37-st kahendkohast ja milles salvestatud arv võtab osa peaaegu kõikide operatsioonide sooritamisest. Summaatori kahend-

kohad on nummerdatud (kümnendnumbritega) numbrite kahanevas järjekorras (alates 36-st ja lõpetades 1-ga), kusjuures kahendkoht, milles kujutatakse arvu märki, on dubleeritud. Arvuti signalisatsioonipaneelile on paigutatud summaatori kahendkohtadele vastavad signaallambikesed (neoonlambid). Viimased asuvad kolmekaupa rühmades. See võimaldab lugeda summaatoris olevat kahendarvu kaheksandsüsteemis.



Kui summaatori mingis kahendkohas salvestatakse arv 1, siis temale vastavas signaallambikeses põleb tuli. Arvu 0 korral on aga lambike kustunud. Summaatorit tähistatakse edaspidi tähega S. Summaatori üksikut kahendkohta tähistame S^i, kus i on vastava kahendkoha järjekorra number. Muuhulgas tähistatakse ka pesa a üksikut kahendkohta a^i. Edaspidise käsitluse huvides lepime kokku, et arvu märgi kahendkohta vaatleme alati koos kahendkohaga nr. 36, s. t. arv on positiivne siis ja ainult siis, kui nendes mõlemas kujutub kahendarv 0, ning negatiivne, kui seal on 1.

Lisaks summaatorile on elektronarvutil „Uraal-1" veel kolmekümne kuuest kahendkohast koosnev aritmeetilise seadme register R.



Arvutis kasutatakse registrit abiseadmena peaaegu kõikide operatsioonide puhul, otseselt rakendatakse seda aga operatsioonide 05, 11 ja 17 korral.

Seoses tehtekoodidega 25 ning 24 tutvume ka nn. tsüklite loendajaga L. See koosneb kaheteistkümnest kahendkohast, kuhu operatsiooni 25 abil on võimalik kanda maksimaalselt kaheteistkümnekohaline kahendarv (kui täisarv).

Edaspidi tähistatakse pesade aadresse ladina tähestiku tähtedega, millele vajaduse korral lisanduvad ka numbrid (kas indeksitena või „+" märgi abil).

$$a, a_2, B, b+0, k+1, \ldots$$

Pesas a oleva arvu tähistamiseks kasutatakse sümbolit (a). Kirjutis $<x>$ tähistab selle pesa aadressi, kus paikneb arv x. Näiteks kui pesas aadressiga a on kahendarv 0,10 . . . 0, siis (a) = 0,10 . . . 0 ning $<0{,}10 \ldots 0> = a$.

Arvuti käskudesüsteemi kirjeldamisel kasutatakse veel järgmisi tähistusi:

(S_0) — summaatori sisu enne operatsiooni teostamist;
(S) — summaatori sisu pärast operatsiooni teostamist;
(R_0) — aritmeetilise seadme registri sisu enne operatsiooni sooritamist;
(R) — aritmeetilise seadme registri sisu pärast operatsiooni sooritamist;
(L) — tsüklite loendaja sisu pärast operatsiooni teostamist.

Aritmeetilises seadmes töötatakse peale signaali φ iga operatsiooni puhul välja ka veel signaal ω. Selle väärtuseks võib olla kas 0 või 1. Käskude süsteemi kirjelduses näidatakse, millistel tingimustel tekib signaal $\omega = 1$.

2. Käskude süsteem.

Operatsioon 01 — arvude liitmine. Käsu kuju 01 a. Käsu toimel liidetakse (algebraliselt) pesas a olev arv summaatoris oleva arvuga ning summa fikseeritakse summaatoris:

$$(S) = (S_0) + (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 1$, kui $|(S)| \geqslant 1$.

Operatsioon 02 — arvu toomine summaatorisse. Käsu kuju 02 a. Käsu toimel tuuakse pesas a olev arv summaatorisse (arv säilib ka pesas a):

$$(S) = (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati.

Operatsioon 03 — arvude lahutamine. Käsu kuju 03 a. Käsu toimel lahutatakse (algebraliselt) summaatoris olevast arvust pesas a olev arv:

$$(S) = (S_0) - (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 1$, kui $|(S)| \geqslant 1$.

Operatsioon 04 — absoluutväärtuste lahutamine. Käsu kuju 04 a. Käsu toimel lahutatakse (algebraliselt) summaatoris oleva arvu absoluutväärtusest pesas a oleva arvu absoluutväärtus:

$$(S) = (S_0) - (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati.

Operatsioon 05 — korrutamisega liitmine. Käsu kuju 05 a. Käsu toimel korrutatakse aritmeetilise seadme registris R olev arv arvuga pesast a ja saadud korrutis liidetakse (algebraliselt) summaatoris oleva arvuga. Tulemus saadakse summaatorisse.

$$(S) = (S_0) + (R_0) \times (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 1$, kui $|(S)| \geqslant 1$.

Operatsioon 06 — arvude korrutamine. Käsu kuju 06 a. Käsu tulemusena korrutatakse summaatoris olev arv pesas a oleva arvuga ning korrutis fikseeritakse summaatoris:

$$(S) = (S_0) \times (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati.

Operatsioon 07 — arvude jagamine. Käsu kuju 07 a. Käsu toimel jagatakse summaatoris olev arv pesas a oleva arvuga. Jagatis saadakse summaatorisse.

$$(S) = (S_0) : (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 1$, kui $|(S)| \geqslant 1$.

Operatsioon 10 — märgi omistamine. Käsu kuju 10 a. Käsu toimel omistatakse summaatoris olevale arvule pesas a oleva arvu märk (arvu absoluutväärtus ei muutu):

$$(S) = |(S_0)| \cdot \text{sign } (a).$$

Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati.

Operatsioon 11 — arvu nihutamine. Käsu kuju 11 0000 (võib ka 11 a, kui $a < 4000$). Käsu toimel nihutatakse registris R olevat arvu summaatori esimese poole kahte viimasesse kolmikusse (kahendkohad $24 \div 19$) kirjutatud täisarvu kahendkohtade võrra. Nihe tehakse vasakule, kui summaatoris olev arv on positiivne, ja paremale, kui summaatoris kujutub negatiivne arv. Nihkele allub ka registris oleva arvu märk. Nihutatud arv fikseeritakse summaatorisse. Signaal $\omega = 1$, kui pärast nihet $(S) = 0$. Signaal $\varphi = 0$ alati.

Operatsioon 12 — osa eraldamine e. loogiline korrutamine. Loogiline korrutamine kui matemaatilise loogika tehe (tähistamiseks kasutatakse sümbolit $\wedge$) teostatakse alljärgneva tabeli kohaselt:

$$0 \wedge 0 = 0$$
$$1 \wedge 0 = 0$$
$$0 \wedge 1 = 0$$
$$1 \wedge 1 = 1.$$

Käsu kuju 12 a. Arvutis teostatakse operatsioon 12 pesas a ja summaatoris olevate arvude iga üksiku kahendkohaga eraldi:

$$(S^i) = (S_0^i) \wedge (a^i).$$

Tingituna arvuti ehitusest peab aadress a olema paarisarvuline. Kui $a < 4000$, siis $i = 36, 35, \ldots, 19$ ning $a \geqslant 4000$ korral $i = 36, 35, \ldots, 1$. Oletades, et

$$(a) = 00\,000\,000\,111\,111\,000,$$
$$(S_0) = 01\,101\,111\,011\,011\,110,$$

siis pärast operatsiooni 12 a

$$(S) = 00\,000\,000\,011\,011\,000.$$

Signaal $\omega = 1$, kui $(S) = 0$. Signaal $\varphi = 0$ alati.

Operatsioon 13 — formeerimine e. loogiline liitmine. Loogiline liitmine kuulub samuti matemaatilise loogika tehete hulka (tähistamiseks kasutatakse sümbolit $\vee$) ning teostatakse järgmise tabeli alusel:

$$0 \vee 0 = 0$$
$$1 \vee 0 = 1$$
$$0 \vee 1 = 1$$
$$1 \vee 1 = 1.$$

Käsul tehtekoodiga 13 on kuju 13 a ning ta teostatakse pesas a ja summaatoris olevate arvude üksikute kahendkohtade vahel:

$$(S^i) = (S_0^i) \vee (a^i).$$

Käsu aadress a peab olema paarisarvuline. Kui $a < 4000$, siis $i = 36, 35, \ldots, 19$ ning $a \geqslant 4000$ korral $i = 36, 35, \ldots, 1$. Operatsiooni 13 kasutatakse peamiselt kas mitmest käsust või arvust uue käsu või arvu formeerimiseks. Oletades näiteks, et

$$(a) = 00\,100\,000\,000\,000\,000,$$
$$(b) = 00\,000\,001\,111\,011\,101,$$

siis pärast programmiosa

```
02 a
13 b
16 c
```

saame pessa c käsu 04 1735.
Signaal $\omega = 1$, kui $(S) = 0$. Signaal $\varphi = 0$ alati.

Operatsioon 14 — arvude võrdlemine e. mittesamaväärsus. Mittesamaväärsustehe pärineb matemaatilisest loogikast (tähistatakse sümboliga $\approx$)ja seda teostatakse alljärgnevalt:

$$0 \approx 0 = 0$$
$$1 \approx 0 = 1$$
$$0 \approx 1 = 1$$
$$1 \approx 1 = 0.$$

Käsu kuju 14 a. Operatsioon 14 tehakse samuti nagu kaks eelnevat, s. t. pesas a ja summaatoris olevate arvude üksikute kahendkohtade vahel eraldi:

$$(S^i) = (S_0^i) \approx (a^i).$$

Kui $a < 4000$, siis $i = 36, 35 \ldots, 19$ ning $a \geqslant 4000$ korral $i = 36, 35, \ldots, 1$. Käsu aadress a peab olema paarisarvuline.

Operatsiooni 14 kasutatakse arvude võrdlemiseks. Kui võrreldavatel arvudel (s. t. summaatoris ja pesas a) langesid kokku kõik kahendkohad, siis $\omega = 0$. Järelikult $\omega = 1$, kui $(S) \neq 0$. Signaal $\varphi = 0$ alati.

Operatsioon 15 — normaliseerimine. Käsu kuju 15 a. Aadress a peab olema paarisarvuline. Käsu 15 a tulemusena viiakse summaatoris olev arv x kujule $x = m_x \cdot 2^{p_x}$, kusjuures mantiss m_x rahuldab tingimust $0{,}1 \leqslant |m_x| < 1$ ja ta kirjutatakse pessa a. Suurus p_x jääb summaatorisse ja kujutub seal kui täisarv 2^{-17} ühikutes. Kui $|(S_0)| < 1$ (s. t. $|x| < 1$), siis p_x kujutub summaatoris negatiivsena ning $1 \leqslant |(S_0)| < 2$ korral $p_x = 1 \cdot 2^{-17}$. Mantissi m_x märk on alati sama, mis normaliseeritaval arvul x. Näiteks summaatoris olevast arvust 0,000110 . . . 0 saadakse pärast käsku 15 a vastavalt (a) = 0,110 . . . 0 ning (S) = = —00 000 000 000 000 011. Absoluutväärtuselt ühest väiksemate arvude korral nihkub arvu moodul niimitu kahendkohta vasakule, kuimitu nulli on arvus koma ja esimese nullist erineva tüvenumbri vahel. (0,000110 . . . 0 = = 0,110 . . . 0 · 10^{-11}). On võimalik normaliseerida ka ületäitunud arve. Näiteks $(S_0) = 1{,}0110 \ldots 0$ korral (a) = = 0,101100 . . . 0 ja (S) = 00 000 000 000 000 001. Ühest suuremate, kuid kahest väiksemate arvude puhul toimub nihe ühe koha võrra paremale. Signaal $\omega = 1$, kui $(S_0) = 0$, s. t. kui normaliseeritav arv on null. Signaal $\varphi = 0$ alati.

Operatsioon 16 — arvu saatmine pessa. Käsu kuju 16 a. Käsu tulemusena salvestatakse summaatoris olev arv pessa a. Summaatori sisu jääb seejuures muutumatuks:

$$(a) = (S_0) = (S).$$

Signaal $\omega = 1$, kui $(S_0) \leqslant -0$. Signaal $\varphi = 0$ alati.

Operatsioon 17 — arvu saatmine aritmeetilise seadme registrisse. Käsu kuju 17 a. Käsu toimel saadetakse pesas a olev arv aritmeetilise seadme registrisse. Summaatori sisu jääb muutumatuks:

$$(R) = (a) \text{ ja } (S) = (S_0).$$

Signaal $\omega = 1$, kui $(R) = 0$. Signaal $\varphi = 0$ alati. Käsku 17 a kasutatakse enamuses operatsioonide 05 ja 11 teostamiseks.

Operatsioon 20 — arvu saatmine summaatorisse. Käsk võib omada kuju 20 k või 20 4000 + k ($k < 4000$). Käsu 20 k toimel kantakse summaatorisse täisarv k (summaatori esimese poole lõpu koha ühikutes), käsu 20 4000 + k korral aga —k. Näiteks käsu 20 1576 tulemusena saame summaatorisse kahendarvu +00 000 001 101 111 110 . . . 0 ning 20 5576 puhul —00 000 001 101 111 110 . . . 0. Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati.

Operatsioon 21 — tingimuslik suunamine. Käsu kuju 21 a. Käsu sisu sõltub käsule 21 a vahetult eelneva käsu tulemusena saadud signaalist ω. Kui $\omega = 1$, siis järgnevana (s. t. pärast op. 21) täidetakse programmi käsk pesast a. Signaali $\omega = 0$ korral jätkub käskude loomulik täitmise järjekord. Operatsioon 21 ei muuda signaali ω (kõneldakse, et ω säilib) ega ka summaatori sisu.

$$(S) = (S_0).$$

Signaal $\varphi = 0$ alati.

Operatsioon 22 — tingimusteta suunamine. Käsu kuju 22 a ($a < 4000$). Pärast käsku 22 a alustatakse käskude täitmist pesast a. Summaatori sisu ei muutu.

$$(S) = (S_0).$$

Signaal ω säilib. Signaal $\varphi = 0$ alati.

Operatsioon 23 — võtmesse suunamine. Käsu kuju 23 k. Operatsioon 23 on tingimuslik suunamine. Ta arvestab arvuti juhtimispuldil olevate lülitajate seisundit. Seal paikneb muuhulgas seitse tumblerlülitit, mida nimetatakse võtmeteks ja mis on nummerdatud vastavalt 1, . . . , 7.

Aadressi kirjutatud suurus k tähendab võtme numbrit. Näiteks käsk, mis suunab võtmesse nr. 5, omab kuju 23 0005. Käsu 23 k sisu sõltub võtme k asendist (sisse või välja lülitatud). Kui võti nr. k on välja sülitatud, siis käsu 23 k puhul täidetakse temale vahetult järgnev käsk. On see võti aga sisse lülitatud, jäetakse käsule 23 k järgnev käsk vahele ja täidetakse ülejärgnev.

$$(k+0) = 23\,0007,$$
$$(k+1) = 16\ a\ ,$$
$$(k+2) = 16\ b\ .$$

Esitatud programmilõigu abil saadetakse summaatoris olev arv pesadesse a ja b, kui võti 7 on välja lülitatud, ja ainult pessa b, kui ta on sisse lülitatud.

Operatsiooni 23 korral ei muutu ei summaatori sisu ega signaal ω. Signaal $\varphi = 0$ alati.

Operatsioon 24 — tsükli lõpp. Käsu kuju 24 a. Operatsiooni 24 a sisu sõltub tsükli loendaja L sisust. Kui $(L) \neq 0$, siis käsu 24 a toimel vähendatakse tsüklite loendajasse kirjutatud arvu kas ühe või kahe võrra ja pärast seda alustatakse käskude täitmist pesast aadressiga a. Juhul kui $L = 0$, siis tsüklite loendaja sisu ei vähendata ja järgnevana täidetakse käsule 24 a vahetult järgnevas pesas olev käsk (vt. op. 25). Summaatori sisu ja signaal ω ei muutu. Signaal $\varphi = 0$ alati.

Operatsioon 25 — tsükli algus. Käsu kuju 25 n. Suurus n ei ole pesa aadress, vaid nn. tsükli kordamiste arv. Käsu 25 n tulemusena kantakse arv n tsüklite loendajasse L. Summaatori sisu ei muutu.

$$(L) = n \text{ ja } (S) = (S_0).$$

Operatsioone 25 ning 24 kasutatakse peaaegu alati koos ja nad võimaldavad programmeerida tsükliliselt korduvaid arvutusprotsesse. Kõik käsud, mis programmis asuvad käskude 25 n ja 24 a vahel, kuuluvad kordamisele, alates pesast a kuni selle pesani, kus paikneb 24 a. Kordamiste arvu N leiame valemist

$$N = \begin{cases} n+1, \text{ kui } n < 4000, \\ \frac{(n+2)-4000}{2}, \text{ kui } n \geqslant 4000. \end{cases}$$

Käsu 25 n korral, kui $n \geqslant 4000$, peab n olema tingimata paarisarvuline.

Operatsioon 24 vähendab tsüklite loendaja L sisu ühe võrra, kui $n < 4000$, ning kahe võrra, kui $n \geqslant 4000$. Signaal ω säilib. Signaal $\varphi = 0$ alati.

Operatsioon 26 — kontroll-liitmine. Käsu kuju 26 a. Käsu 26 a toimel liidetakse pesas a olev arv summaatoris oleva arvuga ning summa fikseeritakse summaatoris. Operatsioonist 01 erineb 26 sellega, et tehte tulemusena tekkinud ületäitunud kahendkoht kantakse automaatselt summaatori viimasele kahendkohale. Seetõttu ei tarvitse operatsiooni 26 abil saadud arvude summa olla nende arvude algebraline summa, vaid teatav formaalne arv, nn. *kontrollsumma*. Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati (on automaatselt blokeeritud).

Operatsioon 30 — käskude muutmine. Käsu kuju 30 a ($a < 4000$). Käsu tulemusena liidetakse käsule 30 a vahetult järgnevale käsule pesas a olev arv. Summaatori sisu ei muutu. Näiteks käskudepaari

30 a

22 0000 korral antakse juhtimine käsule pesast (a), s. t. pärast käsku 30 a täidetakse käsk

22 0000 + (a).

Signaal ω säilib. Signaal $\varphi = 0$ alati.

Operatsioon 31 — grupioperatsiooni algus. Grupioperatsioonid on määratud informatsiooni vahetamiseks operatiivmälu ja välismälu vahel. Nad koosnevad kolmest käsust, mida arvuti dešifreerib kui ühtset tervikut. Seetõttu peavad kõik operatsiooni käsud järgnema vahetult üksteisele. Esimese grupioperatsiooni abil kirjutatakse perfolindile märgitud informatsioon arvuti operatiivmällu. Operatsioonil on kuju

31 a_1
01 k
00 a_2.

Teine grupioperatsioon kindlustab informatsiooni ümberkirjutamise välismälust operatiivmällu. Operatsiooni kuju

31 a_1
02 k
00 a_2.

Kolmanda grupioperatsiooni abil kirjutatakse arvud ümber operatiivmälust välismällu. Operatsiooni kuju

31 a_1
03 k
00 a_2.

Operatsioon 32 — arvu trükkimine. Käsu kuju 32 0000 (või ka 32 a). Käsu toimel trükitakse summaatoris olev arv paberilindile, kas siis kaheksand- või kümnendsüsteemis, sõltuvalt vastava lüliti asendist arvuti juhtimispuldil. Trükkimise kiirus — 100 arvu minutis. Trükkimisseade ja arvuti töötavad paralleelselt, s. t. arvu trükkimisel töötab arvuti ise edasi. Signaal ω säilib. Signaal $\varphi = 0$ alati.

Operatsioon 34 — reavahe trükkimine. Käsu kuju 34 0000 (või ka 34 a). Käsu 34 0000 abil toimub trükkimisseadmes paberilindi nihutamine ühe rea võrra. Summaatori sisu ei muutu. Ülejäänus langeb operatsioon täpselt ühte operatsiooniga 32.

Operatsioon 37 — arvuti peatumine. Käsu kuju 37 a. Käsu tulemusena lõpetab arvuti ülesande automaatse lahendamise (s. t. peatub) ja summaatorisse kantakse pesas a olev arv. Arvuti uueks käivitamiseks tuleb inimesel vajutada juhtimispuldil olevale käivitusnupule.

$$(S) = (a).$$

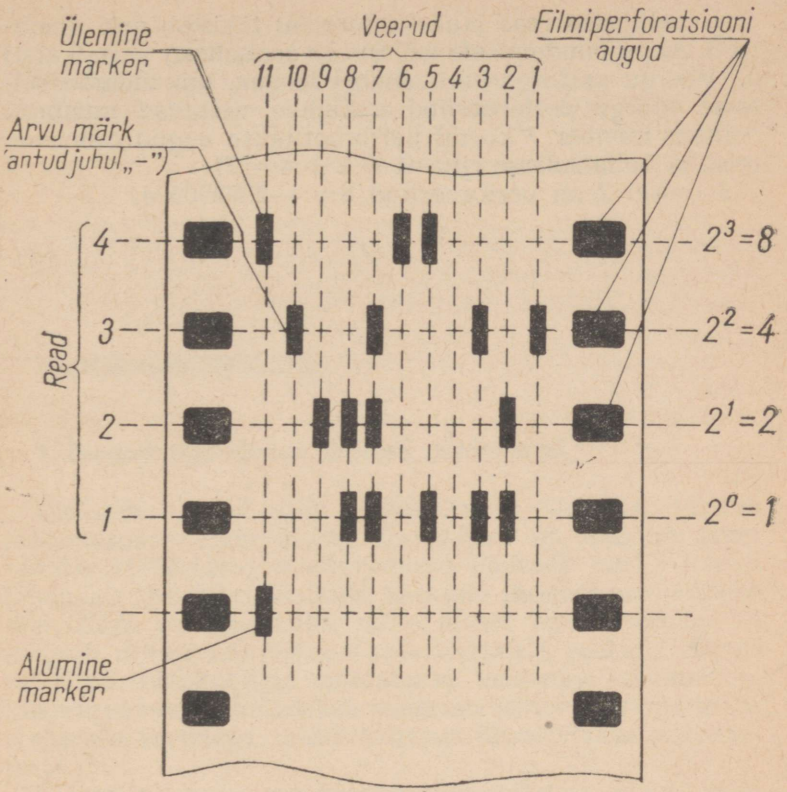
Signaal $\omega = 1$, kui $(S) \leqslant -0$. Signaal $\varphi = 0$ alati.

3. Arvude kujutamine perfolindil.

Nii arvude kui ka käskude sisseviimiseks arvuti mäluseadmesse perforeeritakse nad aukude kujul (vastava lisaseadme, nn. *perforaatori* abil) perfolindile. Viimasena kasutatakse 35 mm laiust läbipaistmatut kinofilmi.

Üks arv kujutatakse kuni nelja ritta ja üheteistkümnesse veergu perforeeritud aukude kombinatsiooni abil. Selgituseks kasutame joonist 2. Ridu perfolindil määravad filmiperforatsiooni augud. Ritta on võimalik perforeerida 11 auku. Järelikult on perfolindil 11 veergu (vertikaalset rida), mis nummerdatakse (kümnendsüsteemis) numbrite kahanevas järjekorras (11, 10, 9, ..., 1). Veergu nr. 11 lööb perforaator automaatselt iga neljanda filmiperforatsiooni kohale augu, mida nimetatakse *markeriks*. Kahe markeri vahelisel alal paikneb üks arv, milleks võib olla kuni kuuekohaline kaheksandarv või kuni üheksakohaline kümnendarv. Markerid liigitatakse (asukoha järgi) *ülemiseks* ja *alumiseks* markeriks (vt. joon. 2).

Ridu nummerdatakse alumise markeri abil nii, et esimene rida asetseb alumisest markerist esimese kõrgemal oleva filmiperforatsiooni kohal.



Joon. 2.

Arvu märk kujutub kolmanda rea kümnendas veerus, kusjuures märgile „—" vastab auk ja märgile „+" selle puudumine. Ülejäänud veerge kasutatakse arvu tüvenumbrite kujutamiseks. Iga veerg on ette nähtud arvu ühe numbrikoha ülesmärkimiseks. Seega veerud määravad numbri positsiooni (asukoha) arvus. Ridade abil moodustatakse erinevatele numbritele vastavad erinevad aukude kombinatsioonid. Sõltumata veerust on ridadesse perforeeritud aukudel järgmised tähendused:

— esimeses reas (ühtede rida) tähistab auk numbrit 1,
— teises reas (kahtede rida) tähistab auk numbrit 2,
— kolmandas reas (neljade rida) tähistab auk numbrit 4,

— neljandas reas (kaheksate rida) tähistab auk numbrit 8. Augu puudumisele (sõltumata asukohast) vastab alati 0. Veergu perforeeritud numbri leiame, kui liidame sellesse veergu perforeeritud aukudele vastavad numbrid. Näiteks number 7 korral perforeeritakse augud esimesse, teise ja kolmandasse ritta ($1 + 2 + 4 = 7$).

Joonisel 2 on perforeeritud arv —237890534.

PROGRAMMEERIMINE

1. Programmjuhtimise üldised põhimõtted.

Ülesande automaatne lahendamine arvutis toimub lahendusprogrammi alusel. Programm on inimese poolt kindlas järjekorras üleskirjutatud käskude jada. Arvuti lahendab ülesannet üksikute käskude järjestikuse täitmise teel. Käsk on signaaliks, mille alusel arvuti töötab. Ta määrab arvuti töö teatava ajavahemiku jooksul. Käsus näidatud tehtekoodiga määratakse teostatav operatsioon, kusjuures tehe sooritatakse enamasti summaatoris ja pesas a olevate arvudega. Tehte tulemus fikseeritakse summaatoris.

Ülesande lahendusprogramm salvestatakse arvuti sisemälu pesades. Lahendusprotsess algab sellest, et arvutile antakse käsk (operaatori poolt või automaatselt programmi abil): alustada käskude täitmist teatavast pesast k. Selle kohta öeldakse, et juhtimine anti pessa k. Pärast juhtimise üleandmist (operatsioon 22) käivitatakse arvuti, vajutades selleks ettenähtud nupule. Käske täidetakse arvutis nn. *loomulikus* järjekorras, s. t. kõigepealt sooritatakse käsk pesast k, siis pesast k + 1 (järgmisest) ja nii edasi, kuni ei ole operatsiooni, mis sunniks seda järjekorda muutma. Käske, mis muudavad käskude täitmise järjekorda, nimetatakse *suunamis-* ehk *juhtimiskäskudeks.* Suunamiskäsud jagunevad tingimusteta ja tingimusega suunamisteks. *Tingimusteta* suunamise (operatsioon 22) korral alustatakse käskude täitmist pesast, mis on näida-

tud käsu aadressis. Näiteks käsu 22 a puhul alustatakse käskude täitmist pesast a ja see jätkub loomulikus järjekorras. *Tingimuslikel* suunamistel (operatsioonid 21, 23, 24) antakse juhtimine edasi kahes võimalikus suunas, sõltuvalt teatud tingimusest (näiteks signaalist ω, võtme sisselülitatusest, tsüklite loendaja sisust). Millises suunas ka juhtimine ei lähe, ikkagi jätkub (alates uuest pesast) käskude loomulik täitmine.

2. Programmi koostamise näide.

Koostame alljärgnevas lineaarse kahe tundmatuga võrrandisüsteemi

$$\begin{cases} ax + by = c \\ dx + my = g \end{cases}$$

põhimõttelise lahendusprogrammi. Siinjuures oletame, et süsteemi determinant on nullist erinev, s. t. $am - bd \neq 0$. Tehtud eeldusel avalduvad lahendid kujul:

$$x = \frac{cm - bg}{am - bd},$$

$$y = \frac{ag - cd}{am - bd}.$$

Programmi koostamiseks on vaja kindlaks määrata programmi käskude ja lähtesuuruste asukohad mälus (mälu jaotus). Alaku programm pesast aadressiga $k + 0$. Süsteemi kordajad ja vabaliikmed paigutame vastavalt pesadesse:

$(k + 30) = a$, $(k + 31) = b$, $(k + 32) = c$,
$(k + 33) = d$, $(k + 34) = m$, $(k + 35) = g$.

Leitud vastused x ja y trükime (kaheksandsüsteemis) paberilindile. Osutub, et niisuguse programmi ratsionaalseks koostamiseks peame kasutama veel kahte mälupesa vahetulemuste salvestamiseks. Olgu nendeks pesad aadressidega A ja B. Pärast ülesande lahendamist peatame arvuti.

Lahendusprogrammi koos selgitustega koondame tabelisse 2.

Pesa		Käsu sisu	Käsu toimel saadakse		
aadress	sisu		summaatorisse	pessa A	pessa B
k+0	02 k+31	Tuua summaatorisse arv b	b	—	—
k+1	06 k+33	Korrutada arv b arvuga d	bd	—	—
k+2	16 A	Salvestada korrutis bd pessa A	bd	bd	—
k+3	02 k+30	Tuua summaatorisse arv a	a	bd	—
k+4	06 k+34	Korrutada arv a arvuga m	am	bd	—
k+5	03 A	Lahutada korrutisest am korrutis bd	am-bd	bd	—
k+6	16 A	Salvestada vahe am-bd pessa A	am-bd	am-bd	—
k+7	02 k+ 31	Tuua summaatorisse arv b	b	am-bd	—
k+10	06 k+35	Korrutada arv b arvuga g	bg	am-bd	—
k+11	16 B	Korrutis bg salvestada pessa B	bg	am-bd	bg
k+12	02 k+32	Tuua summaatorisse arv c	c	am-bd	bg
k+13	06 k+34	Korrutada arv c arvuga m	cm	am-bd	bg
k+14	03 B	Lahutada korrutisest cm korrutis bg	cm-bg	am-bd	bg
k+15	07 A	Jagada vahe cm-bg vahega am-bd	cm-bg / am-bd	am-bd	bg
k+16	32 0000	Trükkida lahend x paberilindile	cm-bg / am-bd	am-bd	bg
k+17	02 k+33	Tuua summaatorisse arv d	d	am-bd	bg
k+20	06 k+32	Korrutada arv d arvuga c	dc	am-bd	bg
k+21	16 B	Salvestada korrutis dc pessa B	dc	am-bd	dc
k+22	02 k+30	Tuua summaatorisse arv a	a	am-bd	dc
k+23	06 k+35	Korrutada arv a arvuga g	ag	am-bd	dc
k+24	03 B	Korrutisest ag lahutada korrutis dc	ag-dc	am-bd	dc
k+25	07 A	Jagada vahe ag-dc vahega am-bd	ag-dc / am-bd	am-bd	dc
k+26	32 0000	Trükkida lahend y paberilindile	ag-dc / am-bd	am-bd	dc

Pesa		Käsu sisu	Käsu toimel saadakse		
aadress	sisu		summaatorisse	pessa A	pessa B
k+27	37 k+0	Peatada arvuti ja summaatorisse kanda arv pesast k+0	02 k+31	am-bd	dc
k+30	a				
k+31	b				
k+32	c	Konstandid (süsteemi kordajad)			
k+33	d				
k+34	m				
k+35	g				

Eespool koostatud programmi nimetatakse *tähelistes* aadressides programmiks. Arvutisse viimiseks selline programm ei kõlba. Tähelistes aadressides koostatud programmist saame tõelise, kui tähed asendada arvudega, s. t. konkreetsete mälupesade numbritega. Valime k=0100. Tehtud valikuga paigutame programmi mällu alates pesast aadressiga 0100. Sellisel juhul salvestatakse süsteemi kordajad pesadesse:

(0130) = a, (0131) = b, (0132) = c,
(0133) = d, (0134) = m, (0135) = g.

Valides veel A = 0136 ning B = 0137, näeb numbrilistes aadressides koostatud programm välja niimoodi:

(0100) = 02 0131
(0101) = 06 0133
(0102) = 16 0136
(0103) = 02 0130
(0104) = 06 0134
(0105) = 03 0136
(0106) = 16 0136
(0107) = 02 0131
(0110) = 06 0135
(0111) = 16 0137
(0112) = 02 0132
(0113) = 06 0134
(0114) = 03 0137
(0115) = 07 0136

(0116) = 32 0000
(0117) = 02 0133
(0120) = 06 0132
(0121) = 16 0137
(0122) = 02 0130
(0123) = 06 0135
(0124) = 03 0137
(0125) = 07 0136
(0126) = 32 0000
(0127) = 37 0100

Pesade 0130÷0135 arvsisu saaksime leida alles pärast suurustele a, b, c, d, m, g arvväärtuste valikut. Et viimast ei ole tehtud, siis puuduvad nad ka programmis.

Edaspidises piirdume peamiselt ainult tähelistes aadressides programmidega, jättes nende numbriliste programmide koostamise lugeja hooleks.

3. Tsükliliselt korduvate arvutusprotsesside programmeerimine.

Paljudes ülesannetes esineb perioodiliselt korduvaid arvutusi. Neid teostatakse ühtede ja samade valemite alusel, kuid (osaliselt või täielikult) uute algandmetega. Näiteks funktsiooni

$$f(x) = \frac{2x^2 + \sin x}{3\cos x + 5}$$

väärtuste arvutamine argumentide $x_1, x_2, \ldots, x_n$ korral. Püstitatud ülesande lahendusprogrammi koostamise üheks viisiks on kirjutada funktsiooni f(x) arvutamiseks vajalik käskuderühm n korda üksteise järel, kusjuures iga kord kasutatakse uut argumendi x väärtust. Selline moodus ei ole aga otstarbekas ja üsna mitmel põhjusel. Näiteks küllalt suure n jaoks ei tarvitse lahendusprogramm üldse mahtuda arvuti sisemällu, s. t. selles on rohkem käske kui arvuti mäluseadmes pesi. Pealegi oleks niisuguse programmi koostamine tülikas ja aeganõudev. Ilmselt on mõistlikum programmeerida ainult funktsiooni f(x) arvutamise valem ja edasi juba organiseerida tema n-kordne kasutamine, muutes automaatselt argumendi x väärtusi.

Tsükliliselt korduvates arvutusprotsessides eristatakse kahte põhimõtteliselt erinevat varianti:

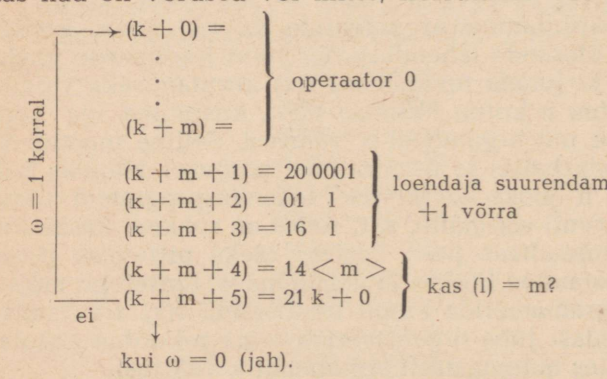
a) arvutuste (nn. tsükli) kordamiste arv on enne vahe-

tut arvutamist teada (fikseeritud kas enne programmi koostamist või arvutatakse programmi jooksul),

b) arvutuste kordamiste arv ei ole ette teada, vaid sõltub arvutusprotsessi enda tulemustest. Näiteks kordub ta senikaua, kuni üks teatav suurus saab suuremaks teisest.

Ülesande lahendamist on võimalik enamasti alati jaotada iseseisvateks osadeks e. etappideks. Loomulikult ei ole selline etappidesse jaotamine teostatav ainult ühel viisil, sest võttes näiteks kaks etappi kokku, saame juba uue jaotuse. Ülesande etappidesse jagamine sõltub nii ülesandest endast kui ka jaotajast, kuid lahendusprogrammis vastab igale etapile kindel programmiosa, mis arvutis realiseerib selle etapi. Nimetame edaspidi seda programmilõiku *operaatoriks.* Tähistame operaatorit tähega 0. Näiteks eelneva funktsiooni f(x) arvutamise programm on operaator. Järelikult tsükliline arvutus on selline, kus üks operaator (neid võib ka rohkem olla) töötab korduvalt.

Alljärgnevas tutvume programmeerimisvõtetega, mille abil on võimalik organiseerida teatava operaatori 0 töötamist m korda. Lihtsam viis on kasutada *loendajat,* kuid seda saab ainult siis, kui kordamiste arv on ette teada. Loendajaks valitakse kindel mälupesa. Operaatori 0 igal järjekordsel kordamisel suurendatakse pesa 1 (ehk lihtsalt loendaja 1) sisu kindla konstandi võrra ja võrreldakse saadud summat etteantud konstandiga. Vastavalt sellele, kas nad on võrdsed või mitte, korratakse või ei korrata

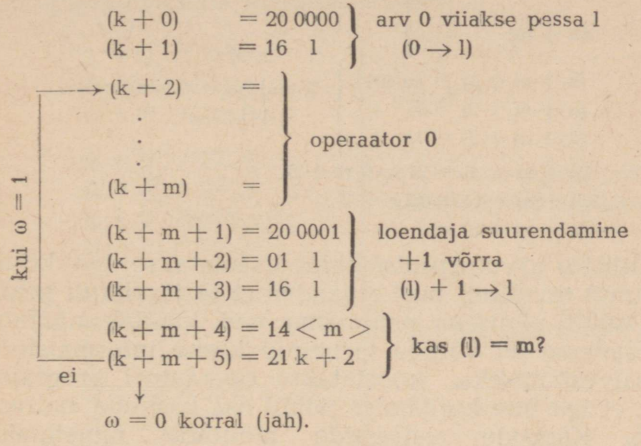


arvutusi. Esitamegi mõningate operaatori 0 kordamist kindlustavate programmilõikude põhiskeemid.

Variant 1. (Pesa 1 sisu peab algul olema võrdne nulliga.)

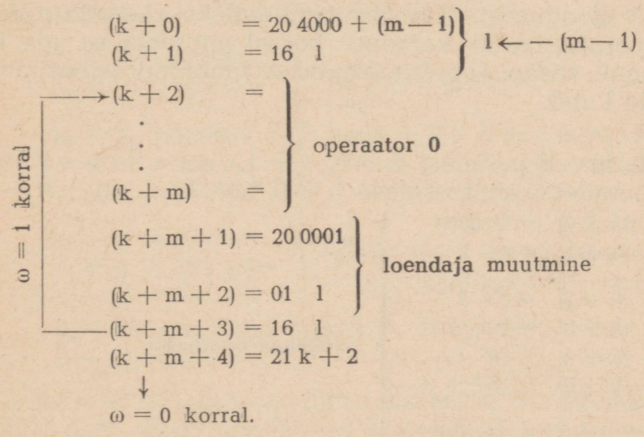
Variant 2.

(Pesa l sisu ei pea võrduma nulliga.)



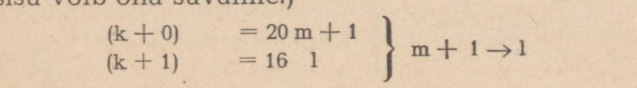
Variant 3.

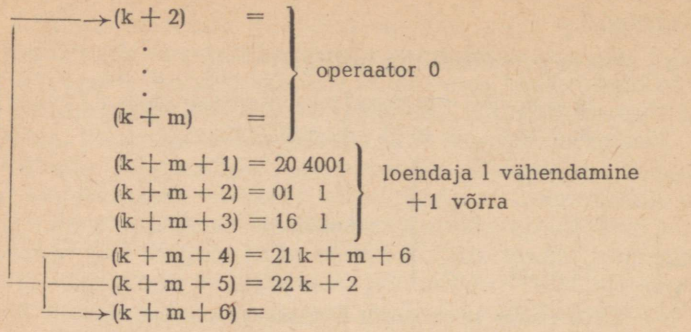
(l sisu võib olla suvaline.)



Variant 4.

(l sisu võib olla suvaline.)





Tsüklilistes arvutusprotsessides kasutatakse iga kordamise puhul tavaliselt uusi algandmeid ja järelikult peame ka iga tsükli alguseks paigutama nad kindlaksmääratud mälupesadesse. Et vältida tarbetuid käske informatsiooni ümbersalvestamiseks, koostatakse operaatori programm niiviisi, et iga uue kordamise puhul osa käskude aadresse muutub. Käskude aadresside muutmist nimetatakse *ümberadresseerimiseks.* Tsüklilist arvutusprotsessi, milles toimub ümberadresseerimine, nimetatakse *ümberadresseerimisega tsükliks.* Ümberadresseerimiseks võib kasutada mitmeid mooduseid. Tavaliselt on selleks käskude muutmise operatsioon 30, vajaduse korral muudetakse aga ka programmi käske. Sageli teostatakse ümberadresseerimine loendaja 1 abil.

Näide 1.

Kanda arvud pesadest a + 0, a + 1, . . . , a + n — 1 vastavalt pesadesse aadressidega b + 0, b + 1, . . . , b + n — 1.

```
(k + 0)  = 20 0000
(k + 1)  = 16  1        } 0 → 1

(k + 2)  = 30  1
(k + 3)  = 02 a + 0
(k + 4)  = 30  1        } ülekande operaator
(k + 5)  = 16 b + 0

(k + 6)  = 20 0001
(k + 7)  = 01  1
(k + 10) = 16  1        } loendaja suurendamine

(k + 11) = 14 <i n>
(k + 12) = 21 k + 2     } kas (l) = n?
```

Näide 2.

Summeerida arvud pesadest a + 0, a + 1, . . . , a + n ning saadud summa salvestada pessa A.

```
(k + 0)  = 20 0000   }
(k + 1)  = 16  1     }  0 → l
(k + 2)  = 16  A     }  0 → A

(k + 3)  = 02  A     }
(k + 4)  = 30  1     }
(k + 5)  = 01 a + 0  }  summeerimise operaator
(k + 6)  = 16  A     }

(k + 7)  = 20 0001   }
(k + 10) = 01  1     }  loendaja suurendamine
(k + 11) = 16  1     }

(k + 12) = 14 < n + 1 >  }  kas (l) = n + 1?
(k + 13) = 21 k + 3      }
```

Näide 3.

Kanda arvud pesadest A + 0, A + 2, . . . , A + 2n — 2 mälupesadesse aadressidega B + 0, B + 4, . . . , B + 4n — 4.

```
(k + 0)  = 02 A + 0   }  ülekande operaator
(k + 1)  = 16 B + 0   }

(k + 2)  = 20 0004    }
(k + 3)  = 01 k + 1   }  pesas k + 1 oleva käsu aadressi suurendamine nelja võrra
(k + 4)  = 16 k + 1   }

(k + 5)  = 20 0002    }
(k + 6)  = 01 k + 0   }  pesas k + 0 oleva käsu aadressi muutumine +2 võrra
(k + 7)  = 16 k + 0   }
(k + 10) = 14 < 02 A + 2n >
(k + 11) = 21 k + 0
```

Toodud ümberadresseerimise moodus ei ole sellepärast eriti sobiv, et pesades k + 0 ja k + 1 olevad käsud on töö lõpuks muutunud, järgneval kasutamisel peame aga nendesse pesadesse kirjutama uuesti (seda nimetatakse ka taastamiseks) vastavalt käsud 02 A + 0 ning 16 B + 0. Allpool esitatakse sama probleemi lahendus operatsiooni 30 abil.

```
(k + 0)  = 20 0000
(k + 1)  = 16  l1      } 0 → l1, l2
(k + 2)  = 16  l2

(k + 3)  = 30  l1
(k + 4)  = 02 A + 0
(k + 5)  = 30  l2      } ülekande operaator
(k + 6)  = 16 B + 0

(k + 7)  = 20 0004
(k + 10) = 01  l2      } (l2) + 4 → l2
(k + 11) = 16  l2

(k + 12) = 20 0002
(k + 13) = 01  l1      } (l1) + 2 → l1
(k + 14) = 16  l1
(k + 15) = 14 < 2n >
(k + 16) = 21 k + 3
```

4. Operatsioonide 25 ja 24 kasutamine.

Operatsioone 25 ning 24 nimetati vastavalt tsükli alguse ja lõpu operatsioonideks ja nad on määratud tsükliliste arvutusprotsesside programmeerimiseks. On teada, et kõik käsud, mis programmis asuvad käskude 25 n ja 24 a vahel, kuuluvad kordamisele N korda. Täpsemalt, korratakse käske, mis asuvad pesa a ja selle pesa vahel, kus paikneb käsk 24 a.

Näide 1.

Arvutada $f(x) = x^{20}$, kui $x = (a + 0)$.

```
(k + 0) = 25 0022
(k + 1) = 02 a + 0        0022(8) = 18(10)
(k + 2) = 06 a + 0
(k + 3) = 24 k + 2
```

Siin käsk 06 a + 0 kordub 19 korda.

Tsükli *sammuks* δ nimetatakse arvu, mille võrra vähendatakse tsükli loendaja L sisu tsükli igakordsel kordamisel. Teame, et tsüklite loendaja sisu vähendab operatsioon 24, kusjuures $n < 4000$ korral $\delta = 1$ ja $n \geqslant 4000$ puhul $\delta = 2$ (n — käsu 25 n aadressi osa).

Tsüklit, kus $\delta = 1$, nimetame *lühikese pesa tsükliks*, $\delta = 2$ korral aga vastavalt *pika pesa tsükliks*. Käskudega 25 n ja 24 a on võimalik teha automaatset ümberadresseerimist. Kõikide nende käskude aadressidest, mis kuuluvad käskude 25 n ja 24 a vahel kordamisele ja mille ette on kirjutatud miinusmärk, lahutatakse arv M, kus

$$M = \begin{cases} (L), \text{ kui } n < 4000 \\ (L) - 4000, \text{ kui } n \geqslant 4000, \end{cases}$$

ja alles siis täidetakse. Järelikult tsüklis olevaid negatiivse märgiga käske ei täideta samaväärselt positiivsetega. Positiivne käsk täidetakse alati samasugusena, nagu ta on kirjutatud programmi, kuna aga negatiivse märgiga käskude aadressid tsükli igakordsel kordamisel muutuvad δ võrra.

Näide 2.

Kanda arvud pesadest $a + 0$, $a + 1$, ..., $a + n$ üle pesadesse $b + 0$, $b + 1$, ..., $b + n$ (kokku $n + 1 < 4000$ ülekannet).

```
      (k + 0) =  25     n
  ┌──→(k + 1) = −02 a + n
  │   (k + 2) = −16 b + n
  └───(k + 3) =  24 k + 1
```

Lihtne on organiseerida tsükleid, kus $\delta > 2$. Selleks peame tsükli lõpu käske programmi kirjutama rohkem kui ühe. Kui $n < 4000$, siis tsükli korral, mis peab töötama sammuga δ, tuleb ka tsükli lõpu käske kirjutada järjest δ korda. Siinjuures annab iga eelnev juhtimise järgmisele ja viimane põhiprogrammi tagasi.

Näide 3.

Liita arvud pesadest $a + 0$, $a + 3$, $a + 6$, ..., $a + 3n$ ($3n < 4000$) ning summa salvestada pessa A. Tsükli samm $\delta = 3$.

```
      (k + 0) =  20 0000
      (k + 1) =  25  3n
  ┌──→(k + 2) = −01 a + 3n
  │ ┌─(k + 3) =  24 k + 4      (käsk 24 k kordub 3 korda)
  │ │ (k + 4) =  24 k + 5
  │ │ (k + 5) =  24 k + 2
  │ └→(k + 6) =  16 A
  └──
```

Kui $n > 4000$, siis sammuga δ töötava tsükli organiseerimiseks peame käske 24 k kirjutama $\frac{\delta}{2}$ korda.

Mitme tsükli lõpu operatsiooni järjestikune kirjutamine ei ole otstarbekas suurte δ väärtuste korral. Toome järgnevalt kaks tsükli organiseerimise skeemi, mis töötavad suvalise sammuga δ.

Variant 1.

Kasutatakse tingimusel, kus $n < 4000$.

```
(k + 1) =  25     n
(k + 2) = -20 4000 + n + (δ - 1)
(k + 3) =  01 k + 1
(k + 4) =  16 k + m
(k + 5) =
   .                 } tsüklis kordamisele kuuluvad käsud
   .
   .
(k + m - 1) =
(k + m) = 00 0000
(k + m + 1) = 24 k + 2
```

Variant 2.

Kasutatakse juhul, kui $n \geqslant 4000$.

```
(k + 1)     =  25     n
(k + 2)     = -20 4000 + n + (δ - 2)
(k + 3)     =  01 k + 1
(k + 4)     =  16 k + m
(k + 5)     =
   .                 } kordamisele kuuluvad käsud
   .
   .
(k + m - 1) =
(k + m)     =  00 0000
(k + m + 1) =  24 k + 2
```

Näide 4.

Olgu tarvis viia arvud pesadest 0511, 0531, 0551, 0571 pesadesse aadressidega 0600, 0620, 0640, 0660. Paigutame vastava programmi mällu alates pesast 1000.

$$n = 0571 - 0511 = 0060$$

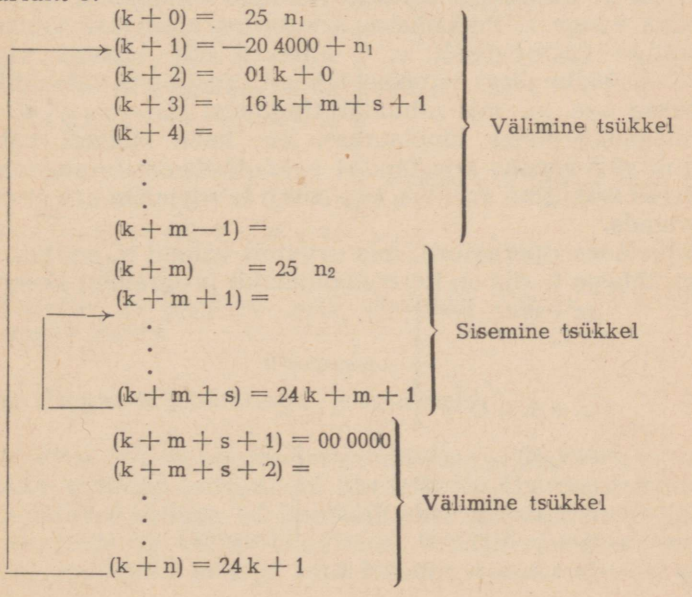
$$\delta = 0531 - 0511 = 0020$$

$$4000 + n + (\delta - 1) = 4000 + 0060 + 0017 = 4077$$

(1000) = 25 0060
(1001) = —20 4077
(1002) = 01 1000
(1003) = 16 1006
(1004) = —02 0571
(1005) = —16 0660
(1006) = 00 0000
(1007) = 24 1001

Elektronarvutis „Uraal-1" on ainult üks tsüklite loendaja L. Seetõttu ei tohi käskude 25 n ja 24 a vahele paigutada teist sama tüüpi tsüklit. Arvutusprotsessis on aga vahel vajalik paigutada üks tsükkel teise sisse, s. t. üks tsükkel sisaldab omakorda veel tsükli. Loomulikult on võimalik kordseid tsükleid programmeerida loendajate abil, kuid osutub, et seda saab teha ka käskudega 25, 24. Tsüklit, mis paigutatakse teise tsükli sisse, nimetatakse *sisemiseks tsükliks*, tsüklit, millesse paigutatakse teine, — *välimiseks tsükliks*. Alljärgnevas tuuakse programmi põhimõtteline skeem, mille alusel on võimalik paigutada kaks (või ka enam) tsüklit teineteise sisse. Oletame, et välimine tsükkel töötab $n_1 + 1$ ning sisemine $n_2 + 1$ korda.

Variant 3.



Esitatud skeemi abil võib paigutada ka erinevate sammudega δ_1 ja δ_2 töötavad tsüklid teineteise sisse, kuid jätame selle juba lugeja enda nuputada.

5. Iteratsioonitsükkel.

Arvutusmeetodit, kus otsitav suurus leitakse tema tõelisele väärtusele järjest lähedasemate väärtuste järkjärgulise arvutamise teel, nimetatakse iteratsioonimeetodiks. Siinjuures arvutatakse iga järgnev eelneva kaudu. Iteratsioonimeetodi korral, lähtudes mingil viisil valitud alglähendist x_0, arvutatakse järgnevad x_i väärtused teatavast seosest

$$x_i = f(x_{i-1}). \qquad (1)$$

Tähistades otsitava suuruse tõelise väärtuse x, konstrueeritakse seose (1) abil jada

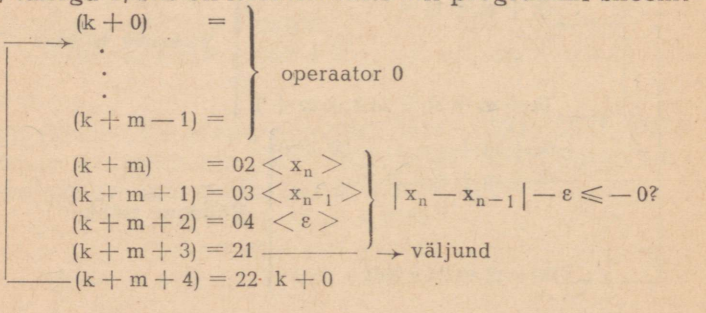
$$x_0, x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n, \ldots,$$

mille korral on teada, et

$$\lim n \to \infty \; x_n = x. \qquad (2)$$

Seosest (2) näeme, et lõpliku n puhul ei saavuta lähend x_n väärtust x. Järelikult leitakse iteratsioonimeetodil lahend teatava veaga ε. Praktilistes arvutustes minnakse viimati mainitule vastupidiselt, s. t. antakse ette lubatav viga $\varepsilon > 0$ ja selle järgi leitakse lahend, kusjuures lahendiks loetakse see x_n, mis rahuldab tingimust $|x_n - x_{n-1}| \leqslant \varepsilon$. Iteratsiooniprotsess lõpetatakse, kui kahe lähendi vahe osutub väiksemaks konstandist ε. Seetõttu on iteratsiooniprotsess tsükliline arvutus, kus tsükli kordamiste arv ei ole ette teada.

Tähistades operaatorit, mis arvutab valemi $x_n = f(x_{n-1})$ järgi, tähega 0, siis on iteratsioonitsükli programmi skeem:



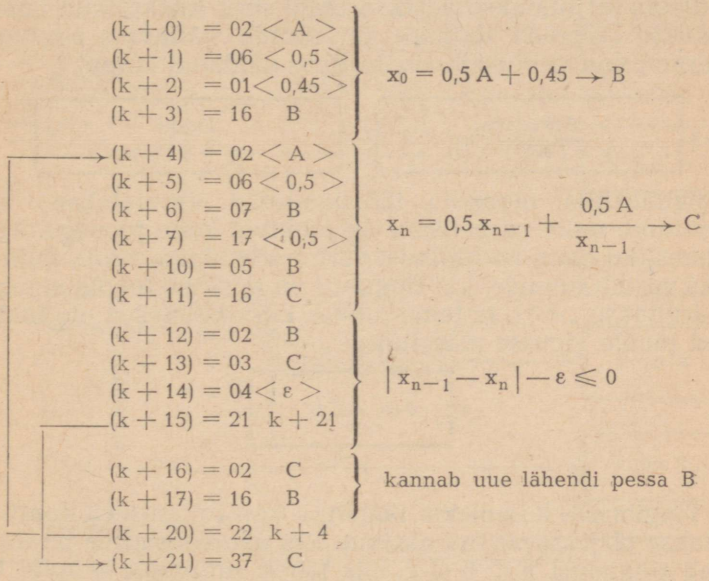
Olgu näiteks tarvis arvutada positiivsest arvust A ($0 < A < 1$) ruutjuur. Selleks kasutatakse nn. Newtoni iteratsioonimeetodit, mille kohaselt lähendid leitakse valemist

$$x_n = 0{,}5\, x_{n-1} + \frac{0{,}5\, A}{x_{n-1}}.$$

Alglähendi x_0 võib arvutada seosest

$$x_0 = \frac{A + 0{,}9}{2}.$$

Programm.



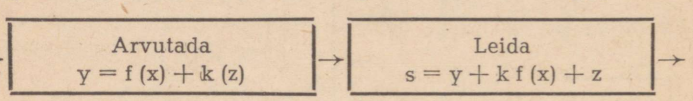
Esitatud programmi järgi saadakse otsitav $\sqrt{A} \approx x_n$ summaatorisse.

6. Hargneva juhtimisega programmid.

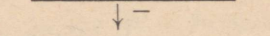
Vaevalt leidub programmi, milles aeg-ajalt ei oleks vaja muuta käskude loomulikku täitmise järjekorda. Juhtimist muudetakse sellega, et kontrollitakse mitmesuguste tingimuste täidetust või mittetäidetust. Tsüklite programmeerimisel puutusime sellega juba kokku. Näiteks korrati seal

arvutusi seni, kuni (l) = m. Alati valitakse kahest võimalikust suunast üks. Arvutis on kasutatav ainult kahevalentne loogika — kas „ei" või „ja". Programmi ei tohi jätta mingisugust kolmandat võimalust, vastasel korral võib arvuti „mõelda" nii, nagu programmeerija seda ei mõelnud.

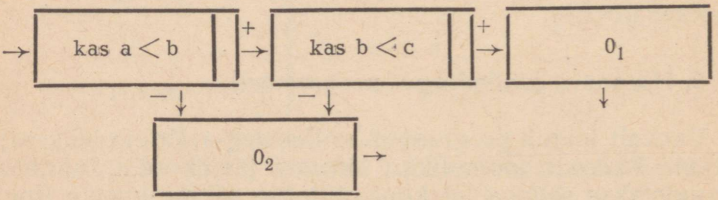
Ülesande programmeerimiseks koostatakse lahenduskäigu *loogiline* e. *blokkskeem*. Selleks jaotatakse lahenduskäik teatavaks arvuks etappideks. (Muuhulgas on omaette etappideks ka tingimuste täidetuse kontrollimine.) Blokkskeemis kasutatakse etapi tähistamiseks ristkülikut, millesse kirjutatakse vastava etapi sisu. Ristkülikud ühendatakse etappide täitmise järjekorda näitavate nooltega (järgmisena täidetakse noole tipus näidatud etapp).



Tingimuslike etappide tähistamiseks kasutatakse ristkülikuid, mille parem serv on joonitud kahe joonega. Tingimusliku etapi ristkülikust väljub alati kaks noolt, millest üks vastab suunale, kui tingimus on täidetud (noole algusse märgitakse „+"), ja teine juhule, kui tingimus ei ole täidetud (noole algusse märgitakse „—").

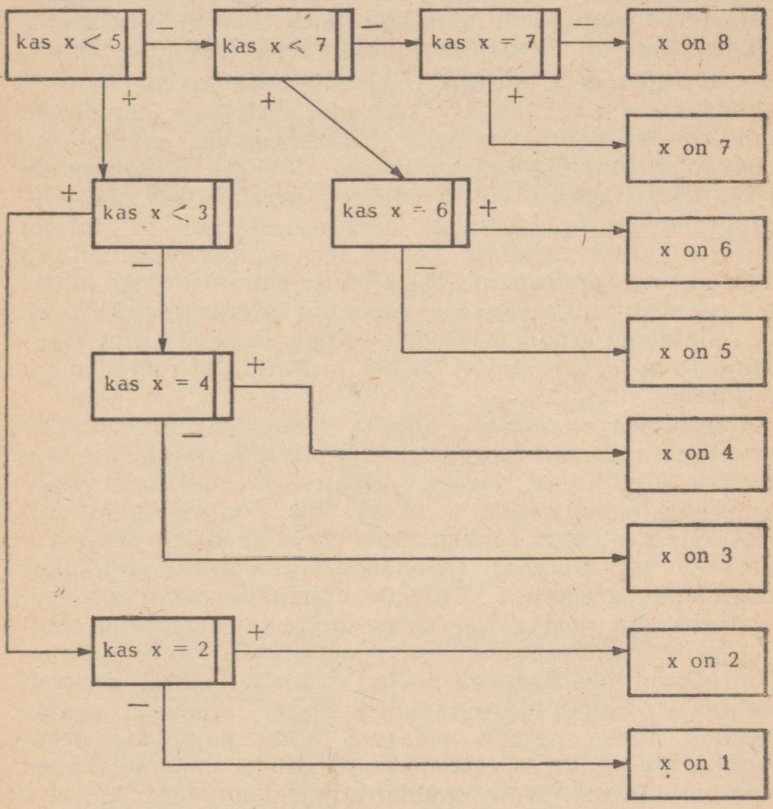


Tingimuste kompleksi täidetuse kontrollimiseks kontrollitakse järjekorras iga üksikut. Kui näiteks suurus b rahuldab tingimust a < b < c, siis läheb juhtimine etapile 0_1, vastasel korral etapile 0_2.



Ulatuslikuma blokkskeemi koostamise näitena vaatame järgmist ülesannet. Mõelge üks numbritest ühest kuni

kaheksani. Osutub, et mõeldud arvu on võimalik kindlaks teha kolme küsimusega. Esitatud küsimustele võib vastata ainult sõnadega „ja" või „ei". Kui esitatud küsimuse vastuseks on „ei", siis läheme blokkskeemis edasi märgiga „—" tähistatud noole suunas, ja „+" suunas, kui vastuseks on „ja".



Toodu eeskujul võiks koostada blokkskeemi, mille abil saame määrata vahemikust $1 \div 2^n$ mistahes täisarvu x, kasutades selleks ainult n küsimust. Näiteks seitsme küsimusega võib leida täisarvu vahemikust $1 \div 128$. Nuputage!

7. Standardprogrammid.

Tihti sisaldavad täiesti erinevate ülesannete lahendusprogrammid ühtesid ning samu operaatoreid. Peaaegu iga lahendatava ülesande korral on vajalik kasutada programmilõiku, mis teisendab vastused kahendsüsteemist kümnendsüsteemi. Samuti ka vastupidi — küllaldane hulk lähteandmeid on mõistlik teisendada kümnendsüsteemist kahendsüsteemi arvuti abil, aga mitte käsitsi. Võiks veel näiteks tuua nn. elementaarfunktsioonide (nagu sinx, cosx, lnx, e^x, tanx, $\sqrt{x}$, arcsinx jne.) väärtuste arvutamise. Et elektronarvuti „Uraal-1" käskude süsteemis puuduvad tehtekoodid nende otseseks leidmiseks, siis arvutatakse elementaarfunktsioonide väärtusi jällegi programmide abil. Kõike kokku võttes jõuame paratamatult mõttele: programmeerida vastavad arvutuseeskirjad ja hiljem kasutada neid vajaduse korral igas programmis. Selleks aga, et programmeeritud algoritm (operaator) oleks lülitatav suvalise programmi koosseisu kui alamprogramm, peab ta rahuldama kindlaid nõudeid, olema sobival kujul koostatud ja vormistatud, s. t. ta peab olema muudetud standardseks.

Käesolevas vaatame ainult elementaarfunktsioonide arvutamise standardprogramme ja tutvume nende ühe võimaliku ülesehituse viisiga. Elementaarfunktsiooni (üldkujul) kirjeldab avaldis $y = f(x)$, kus x on argument, millest leitakse teatud funktsiooni f (x) konkreetne väärtus y (näiteks $y = \text{arctanx}$). Standardprogrammid koostatakse tähelistes aadressides. Viimane võimaldab seda hõlpsasti ulatuslikuma programmi osaks lülitada, kusjuures standardprogrammi esimene käsk paigutatakse põhiprogrammi paarisarvulise aadressiga pessa. Vajalikud töötavad pesad valitakse standardprogrammi lõpust. Samuti peame teadma nende pesade aadresse, kuhu paigutada argument x ning kuhu salvestatakse y (näiteks summaatorisse). Oluline tähtsus on ka standardprogrammi täpsusel, s. t. mitme õige tüvenumbriga (kümnendsüsteemis) arvutatakse funktsiooni väärtus y.

Paigutanud argumendi x temale määratud pessa, antakse juhtimine standardprogrammile (täpsemalt, selle esimesele käsule). Pärast funktsiooni väärtuse y arvutamist peab aga juhtimine tulema põhiprogrammi tagasi. Viimane

garanteeritakse standardprogrammi lõpus olevate käskudega

30 λ
22 0000.

Järelikult enne juhtimise üleandmist standardprogrammile on tarvis viia pessa λ selle pesa aadress (täisarvuna 2^{-17} ühikutes), kuhu peab juhtimine tagasi tulema. Kui ta näiteks peab tulema tagasi pessa 1735, siis väljundpessa λ salvestame kaheksandarvu 001735.

Vaatame nüüd näitena funktsiooni $y = \sin x$, kui $-1 < x < 1$, arvutamise standardprogrammi. Kasutades valemit

$$\sin x = x + x(((-0{,}00198x^2 + 0{,}00833)x^2 - 0{,}16666)x^2)$$

on kindlustatud, et sinx vea absoluutväärtus ei ületa suurust 0,000 003, s. t. ta arvutatakse viie õige tüvenumbriga.

Programm

(k + 0)	= 16 k + 4020	} arvutab y = sinx	
(k + 1)	= 06 k + 4020		
(k + 2)	= 16 k + 4022		
(k + 3)	= 06 k + 0014		
(k + 4)	= 01 k + 0015		
(k + 5)	= 06 k + 4022		
(k + 6)	= 01 k + 0016		
(k + 7)	= 06 k + 4022		
(k + 10)	= 06 k + 4020		
(k + 11)	= 01 k + 4020		
(k + 12)	= 30 k + 0017	} väljund	
(k + 13)	= 22 0000		
(k + 14)	= —00 0037	} konstandid	— 0,00198
(k + 15)	= 00 2110		0,00833
(k + 16)	= —05 2524		— 0,16666
(k + 17)	= 00 0000	} töötavad pesad	
(k + 20)	= 00 0000		
(k + 21)	= 00 0000		
(k + 22)	= 00 0000		
(k + 23)	= 00 0000		

Antud programmi korral peab argument x enne juhtimise üleandmist olema summaatoris. Samuti ka $y = \sin x$ saadakse summaatorisse.

Olgu tarvis arvutada funktsiooni $y_i = 0{,}5 \sin x_i$, kus

$x_i = 0{,}01i$, väärtused iga $i = 0, 1, 2, \ldots, 99$ puhul ning trükkida need (kaheksandsüsteemis) paberilindile. Paigutame programmi mällu alates pesast 0100 ning ta lahendab püstitatud ülesande, kui juhtimine anda käsule pesast 0124.

(0100) =	16	4120	
(0101) =	06	4120	
(0102) =	16	4122	
(0103) =	06	0114	
(0104) =	01	0115	
(0105) =	06	4122	y = sinx
(0106) =	01	0116	
(0107) =	06	4122	
(0110) =	06	4120	
(0111) =	01	4120	
(0112) =	30	0117	väljund
(0113) =	22	0000	
(0114) =	—00	0037	
(0115) =	00	2110	konstandid
(0116) =	—05	2524	
(0117) =	00	0000	
(0120) =	00	0000	
(0121) =	00	0000	töötavad pesad
(0122) =	00	0000	
(0123) =	00	0000	
(0124) =	20	0000	
(0125) =	03	4142	
(0126) =	16	4120	
(0127) =	25	0143	
(0130) =	20	0135	
(0131) =	16	0117	
(0132) =	02	4120	
(0133) =	01	4142	$y_i = 0{,}5 \sin x_i$
(0134) =	22	0100	
(0135) =	06	0124	
(0136) =	32	0000	
(0137) =	24	0132	
(0140) =	37	0000	
(0141) =	00	0000	
(0142) =	00	2436	0,01
(0143) =	53	2173	

8. Ülesande programmeerimise ja lahendamise üldine skeem.

Ülesande lahenduskäigu võime jaotada järgmisteks osadeks:

1) lahendusmeetodi valik,
2) ülesande aritmetiseerimine,
3) blokkskeemi koostamine,
4) lahendusprogrammi koostamine,
5) programmi silumine,
6) ülesande lahendamine.

Ülesande lahendamiseks kasutatav metoodika sõltub kahtlemata ülesandest. Meetodit ei valita, kui arvutatakse ettenähtud valemite kohaselt. Seda tüüpi on paljud insenertehnilised ülesanded, kus kohe ülesande formuleeringus on toodud ka arvutusmeetod. Näiteks leida avaldise

$$f(x, y) = \frac{, x \cos y + 5 \sin^2 x}{\tan x + x \sqrt{y}\, e^x}$$

väärtuste tabel muutujate x ja y teatud väärtuste korral. Sageli kohtame aga teist liiki ülesandeid. Näiteks lahendada kolmekümne tundmatuga lineaarne võrrandisüsteem, kusjuures süsteemi kordajad ja vabaliikmed on tuntud suurused. Lineaarseid võrrandisüsteeme võib lahendada mitmeti, selleks on olemas rida meetodeid. Ühe peame nendest valima, kuid millise? Arvutusmeetodi valikul lähtutakse vähemalt kahest printsiibist:

a) valitud meetodil lahendatud ülesande vastuste viga ei tohi ületada lubatud vea ülemmäära,

b) ülesanne peab lahenduma võimalikult lühikese ajaga.

Et elektronarvutis kujutatakse arve ligikaudselt, s. t. teatud veaga, siis küllalt suure tehete hulga juures võib tekkida ümardamisvigade kuhjumine, mis aga juba oluliselt mõjutab vastuste täpsust. Lahendusaja lühendamine on tähtis arvutusvigade vältimise (arvutis esinevad vead on teatud mõttes aja funktsioonid) ja ülesande lahenduse maksumuse vähendamise seisukohalt.

Ülesande aritmetiseerimine on lahendusalgoritmi teisendamine programmeerimiseks sobivale kujule. Sealjuures peetakse juba silmas arvuti tehnilisi võimalusi. Elektronarvuti „Uraal-1" mälupesades oli võimalik salvestada ainult murdarve, samuti ei tohi arvutustulemused kasvada absoluutväärtuselt ühest suuremateks. Järelikult tuleb

ülesande lahenduskäik taandada kujule, mis püstitatud tingimusi rahuldaks. Selleks kasutatakse nn. *mastaabi* kordajaid. Nende leidmiseks määratakse (ligikaudselt) kindlaks kõigi muutujate muutumispiirid, s. t. kui „suured" või „väikesed" nad on.

Olgu vaja arvutada funktsiooni

$$y = \frac{3x^2 + 6x + 5}{x^3 + 2} \qquad (1)$$

väärtused argumendi $x = 0, 1, 2, \ldots, 9$ korral. Et argumendi x tõelisi väärtusi ei ole võimalik arvutis kujutada, siis kujutame nad 10 korda väiksematena, s. t. $\bar{x} = {}^1/_{10x}$. Järelikult on suuruse x mastaabiks 10^{-1}, seepärast peame teisendama ka valemit (1). Teeme asenduse $x = 10\bar{x}$.

$$y = \frac{+\ 0{,}3\bar{x}^2 0{,}06\bar{x} + 0{,}005}{\bar{x}^3 + 0{,}002}. \qquad (2)$$

Kerge on veenduda, et $x = 1$ (ehk $\bar{x} = 0{,}1$) korral on y maksimaalne väärtus ${}^{14}/_3$, seega saab arvutustulemus aga ühest suuremaks. Peame ka suurusele y valima mastaabi, milleks piisab ühest kümnendikust, $\bar{y} = {}^1/_{10}y$ $(y = 10\bar{y})$.

$$\bar{y} = \frac{0{,}1(0{,}3\bar{x}^2 + 0{,}06\bar{x} + 0{,}005)}{\bar{x}^3 + 0{,}002}. \qquad (3)$$

Programmeerides avaldise (3), oleme veendunud, et arvutatakse ainult murdarvudega ja vastus y saadakse tema tõelisest väärtusest kümme korda väiksem. Lahendusvalemeid ei teisendata mitte ainult mastaabikordajate sissetoomisel, vaid ka otsese lihtsustuse huvides. Valemite lihtsustamine, ühtede suuruste avaldamine teiste kaudu, võib lühendada programmi. Olgu näiteks n astme polünoom

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0. \qquad (4)$$

Sama polünoom avaldub ka kujul

$$p(x) = ((\ldots(a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_1)x + a_0. \qquad (5)$$

Lugeja võib veenduda (valides näiteks $n = 4{,}5\ldots$), et valemi (4) järgi koostatud p(x) arvutamise programm tuleb tunduvalt pikem kui valemi (5) järgi.

Iga ülesande aritmetiseerimise käigus peame silmas pidama (ja otsima) võimalusi ülesande lahenduse õigsuse kontrollimiseks. Tavaliselt kontrollitakse lahendamise õigsust kahekordselt arvutatud tulemuste võrdlemise teel, viimane aga ei välista arvuti poolt süstemaatiliselt tehtavaid vigu. Arvuti võib arvutada kaks (või enam) korda

sama viisi valesti. Aritmetiseerimisel püüame ülesande lahenduse õigsuse kontrollimiseks leida matemaatilisi seoseid, arvutada teatavaid suurusi erinevate valemite abil jne. Kui näiteks on tarvis arvutada argumendist x funktsiooni väärtus $y = \sin x$, siis leiame sealt ka $z = \cos x$ ning kontrollime, kas $y^2 + z^2 = 1$. Tingimuse täidetus räägib arvutuste õigsusest, sest vaevalt mõlema arvutamisel tehti vead, mis teineteist vastastikku kompenseerivad. Enamasti kõikidest ülesannetest on võimalik leida midagi, mis on ainult temale iseloomulik. See „midagi" tulebki avastada ja kasutada ära lahenduse õigsuse kontrollimiseks.

Ülesande programmeerimise aluseks on blokkskeem. Viimase koostamisel tuleb olla eriti kriitiline, kuna sellest sõltub lahendusprogramm. Praktikas koostatakse kas kaks või enam loogilist skeemi. Esimene nendest annab lahendamisest üldise pildi, haarates ülesande suuremaid ja terviklikumaid etappe ning nendevahelisi seoseid. Programmeerimiseks kasutatakse juba täpsemat skeemi, kus arvestatakse kõiki antud ülesande programmeerimiseks vajalikke peensusi. Mida detailsem ja läbimõeldum on blokkskeem, seda hõlpsam on hiljem tema alusel programmi koostamine.

Programmi kirjutamine on suhteliselt lihtsam töö kui ülesande eelnev ettevalmistamine, s. t. meetodi valik, aritmetiseerimine, informatsiooni paigutamine ja kodeerimine ning loogilise skeemi koostamine. On võimalik anda tervete programmilõikude põhimõttelisi skeeme, mida sisaldavad peaaegu kõik programmid. Näiteks loendajaga tsükkel, iteratsioonitsükkel, standardprogrammid, tsüklid käskudega 25, 24 jne. jne. Programmi koostamisel tuleb püüda alati vältida tarbetuid käske, leida kompromiss programmi pikkuse (käskude arvu) ja töötamisaja vahel.

Programmi silumine tähendab, selle sõna otseses mõttes, koostatud programmist vigade otsimist. Nagu praktika näitab, sisaldab peaaegu iga programm (tema algkujul) vähemalt ühe vea (sageli aga kaugelt rohkem). Vead tekivad paratamatult nii blokkskeemi kui ka programmi enda koostamisel (viimase puhul, muide, sagedamini). Silumise eesmärgiks ongi leida kõik vead ja need kõrvaldada. Selleks lahendatakse üks ülesanne (nn. silumisülesanne) käsitsi. See valitakse lihtsustatud kujul (et

oleks kergem arvutada). Siinjuures peame silmas, et ülesanne tuleb valida siiski sedavõrd keerukas, et töötaksid kõik programmi operaatorid. Vastasel korral võime saada arvutist, võrreldes käsitsiarvutatud vastustega, ühtelangevad tulemused, kuigi programm ei tarvitse olla õige. Silumisülesande lahendamisel tuleb arvestada ülesande blokkskeemi. Nimelt on vajalik üles kirjutada rida nn. vahetulemusi, mida peavad välja arvutama programmi üksikud operaatorid. See annab võimaluse programmi kontrollimiseks etappide kaupa, mis omakorda lihtsustab tunduvalt vigade avastamist.

Ülesande lahendamisele asutakse pärast silumist, s. t. siis, kui oleme tõepoolest veendunud, et programmis ei ole vigu. Ülesannet lahendab arvutit teenindav personal. Programmeerija poolt lõpeb ülesande lahendamine programmi silumisega.

KIRJANDUST

1. *Ü. Kaasik, H. Salum, H. Sinisoo.* Elektronarvutusmasinad. Tallinn, 1960.
2. *Е. С. Богомолова, С. А. Гельфгат, Н. И. Ермилова.* Описание системы команд электронной вычислительной машины «УРАЛ». Изд-во АН СССР. Москва, 1961.
3. *Э. Бут, К. Бут.* Автоматические цифровые машины. Изд-во «ФИЗМАТГИЗ». Москва, 1959.
4. *Б. В. Гнеденко, В. С. Королюк, Е. Л. Ющенко.* Элементы программирования. Изд-во «ФИЗМАТГИЗ». Москва, 1961.
5. *Е. А. Жоголев, Г. С. Росляков, Н. П. Трифонов, М. Р. Шура-Бура.* Система стандартных подпрограмм. Изд-во «ФИЗМАТГИЗ». Москва, 1958.
6. *А. И. Китов, Н. А. Криницкий.* Электронные цифровые машины и программирование. Изд-во «ФИЗМАТГИЗ». Москва, 1959.
7. *Н. Я. Смольников.* Основы программирования для цифровой машины «УРАЛ». Изд-во «Советское радио». Москва, 1961.

SISUKORD

Мати Крулль
ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И ПРОГРАММИРОВАНИЕ
На эстонском языке
Издательство «Валгус»
Таллин, Пярнуское шоссе, 10

*

Toimetaja H. Heinoja
Kunstiline toimetaja H. Tikand
Tehniline toimetaja T. Linkvist
Korrektorid A. Kalberg ja H. Kull

Ladumisele antud 9. IX 1965.
Trükkimisele antud 4. VI 1966.
Paber 54×84, 1/16. Trükipoognaid 4,0. Tingtrükipoognaid 3,36. Arvestuspoognaid 2,82. Trükiarv 5000. MB-05533.
Tellimise nr. 3141. Trükikoda «Punane Täht», Tallinn, Pikk tn. 54/58.

Trükipaber nr. 3 — Kohila Paberivabrik.

Hind 11 kop.

2—2—5

11 kop.