



I. KULL

# ARVUTID JA PROGRAM-MEERIMINE



TARTU RIIKLIK ÜLIKOOL
Arvutusmatemaatika kateeder

I. Kull

# ARVUTID JA PROGRAMMEERIMINE

I

Tartu 1965

TARTU ÜLIKOOLI
RAAMATUKOGU

I p e a t ü k k.

# ARVUTUSMASINATES KASUTATAVA ARITMEETIKA ALUSED.

## § 1. Positsioonilised arvusüsteemid.

Et arvudega opereerida, on vaja neid kõigepealt mingil viisil tähistada. Arvude mistahes tähistamise süsteemi nimetatakse arvusüsteemiks. Nii kujutab näiteks rooma numbrite süsteem, kus arvud kodeeritakse sümbolite I, V, X, L, C, D ja M abil, üht arvusüsteemi. Et aritmeetiliste tehete teostamise algoritmid tulevad selles süsteemis küllaltki keerulised, siis arvutamisel seda ei kasutata.

Tänapäeval kasutatakse arvutamisel positsioonilisi arvusüsteeme. Positsioonilistes arvusüsteemides, täpsemini positsioonilises p-ndsüsteemis ($p \geqslant 2$) tähistatakse p esimest naturaalarvu (alates arvust null) eri sümbolite, nn. numbrite abil. Vastavaid sümboleid ehk numbreid võib muidugi valida mitmeti. Nii näiteks kasutatakse positsioonilises 10-ndsüsteemis numbreid

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Arvu p nimetatakse p-ndsüsteemi aluseks. Selle tähistamiseks tuleb p-ndsüsteemis kasutada juba numbrite kombinatsiooni 10. Üldiselt numbrite kombinatsiooni

$$\alpha_m \alpha_{m-1} \cdots \alpha_1 \alpha_0, \beta_1 \beta_2 \cdots \beta_n \qquad (1)$$

(kus $\alpha_m, \alpha_{m-1}, \ldots, \beta_n$ on p-ndsüsteemi numbrid) abil tähistatakse arvu

$$\alpha_m \cdot 10^m + \alpha_{m-1} \cdot 10^{m-1} + \ldots + \alpha_1 \cdot 10^1 + \alpha_0 \cdot 10^0 + \beta_1 \cdot 10^{-1} + \beta_2 \cdot 10^{-2} + \ldots + \beta_n \cdot 10^{-n}. \qquad (2)$$

Avaldistest (1) ja (2) nähtub, et sama numbri all võib tähistada erinevaid arve olenevalt selle numbri asukohast kirjutises (1). Selle asjaolu tõttu ongi positsioonilised arvusüsteemid saanud oma nimetuse. Et edaspidi vaatleme ainult positsioonilisi arvusüsteeme, jätame termini "positsiooniline" kirjutamata.

Teeme nüüd mõned märkused arvusüsteemide kohta. Kõigepealt märgime, et arvutusmasinates saab kasutada ainult lõpliku pikkusega (s.t. lõpliku p-ndkohtade arvuga) arvude kirjutisi kujul (1). Nii näiteks ei saa kasutada kirjutisi

$$0,333 \ldots 3 \ldots \qquad (3)$$

ja

$$1,2727 \ldots 27 \ldots, \qquad (4)$$

kuigi need tähistavad vastavalt ratsionaalarve $\frac{1}{3}$ ja $\frac{14}{11}$.

Edasi, igal arvul kujul (2) on vähemalt üks p-ndsüsteemi esitus kujul (1). Väite kehtivus järeldub vahetult kirjutise (1) definitsioonist. Eeldades, et kirjutisę (1) alguses ja lõpus asuvad nullid jäetakse kirjutamata, võib väita veelgi rohkem: arvu (2) esitus p-ndsüsteemis kujul (1) on ühene.

Kui lubada aga ka lõpmatuid kirjutisi [nagu (3) ja (4)], siis pole arvu kirjutis p-ndsüsteemis enam ühene. Nii saame 10-ndsüsteemis kirjutada

$$0,5 = 0,4999 \ldots,$$
$$0,78 = 0,77999 \ldots,$$

2-ndsüsteemis

$$0,1 = 0,0111\ldots,$$
$$0,01 = 0,00111\ldots$$

jne.

Kerge on näidata, et arvu a korrutamine arvuga $10^k$ või $10^{-k}$ (kus 10 tähistab p-ndsüsteemi alust "p") on samaväärne koma nihutamisega arvu a p-ndsüsteemi esituses (1) k koha võrra vastavalt paremale või vasakule. Esitatud väited tõestada lugejatel.

Vaatleme lähemalt 2-nd-, 8-nd-, 10-nd- ja 16-ndsüsteeme, sest need leiavad laialdasemat kasutamist tänapäeva arvutusmasinates. Peale nende arvusüsteemide kasutatakse ka teisi

arvusüsteeme (3-nd-, 4-nd-, 5-nd- jt. süsteeme), kuid kõikide nende süsteemide vaatlemine pole vajalik kasvõi seetõttu, et nende põhiomadused on samad ja põhimõtteliselt ei erine ka tehete sooritamine nendes süsteemides.

2-nd- ja 8-ndsüsteemis on otstarbekohane kasutada samu numbreid, mis 10-ndsüsteemis (arvusüsteemi alus, vastavalt 2 või 8 tuleb muidugi tähistada "10" abil). 16-ndsüsteemis tuleb aga kasutada uusi numbreid arvude 10, 11, 12, 13, 14 ja 15 tähistamiseks. Tähistame need järgmiselt:

$10 = \overline{0}$, $11 = \overline{1}$, $12 = \overline{2}$, $13 = \overline{3}$, $14 = \overline{4}$, $15 = \overline{5}$.

Tabelis 1 on toodud esimeste naturaalarvude kirjutised 10-nd, 2-nd-, 8-nd- ja 16-ndsüsteemis.

Tabel 1.

| 10-nd-süsteem | 2-nd-süsteem | 8-nd-süsteem | 16-nd-süsteem | 10-nd-süsteem | 2-nd-süsteem | 8-nd-süsteem | 16-nd-süsteem |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | |
| 1 | 1 | 1 | 1 | 17 | 10001 | 21 | 11 |
| 2 | 10 | 2 | 2 | 18 | 10010 | 22 | 12 |
| 3 | 11 | 3 | 3 | 19 | 10011 | 23 | 13 |
| 4 | 100 | 4 | 4 | 20 | 10100 | 24 | 14 |
| 5 | 101 | 5 | 5 | 21 | 10101 | 25 | 15 |
| 6 | 110 | 6 | 6 | 22 | 10110 | 26 | 16 |
| 7 | 111 | 7 | 7 | 23 | 10111 | 27 | 17 |
| 8 | 1000 | 10 | 8 | 24 | 11000 | 30 | 18 |
| 9 | 1001 | 11 | 9 | 25 | 11001 | 31 | 19 |
| 10 | 1010 | 12 | $\overline{0}$ | 26 | 11010 | 32 | $1\overline{0}$ |
| 11 | 1011 | 13 | $\overline{1}$ | 27 | 11011 | 33 | $1\overline{1}$ |
| 12 | 1100 | 14 | $\overline{2}$ | 28 | 11100 | 34 | $1\overline{2}$ |
| 13 | 1101 | 15 | $\overline{3}$ | 29 | 11101 | 35 | $1\overline{3}$ |
| 14 | 1110 | 16 | $\overline{4}$ | 30 | 11110 | 36 | $1\overline{4}$ |
| 15 | 1111 | 17 | $\overline{5}$ | 31 | 11111 | 37 | $1\overline{5}$ |
| 16 | 10000 | 20 | 10 | 32 | 100000 | 40 | 20 |

Toome arvude kirjutiste kohta veel mõningaid teisi näiteid, tähistades indeksitega vastavad arvusüsteemid. Nii kehtivad võrdused

$100_{10} = 1\ 100\ 100_2 = 144_8 = 64_{16}$ ,

$381_{10} = 101\ 111\ 101_2 = 575_8 = 17\bar{3}_{16}$ ,

$1964_{10} = 11\ 110\ 101\ 100_2 = 3654_8 = 7\bar{0}\bar{2}_{16}$.

Kontrollime näiteks viimase võrduste rea kehtivust:

$11\ 110\ 101\ 100_2 = 1\cdot10^{1010} + 1\cdot10^{1001} + 1\cdot10^{1000} + 1\cdot10^{111} +$

$+ 0\cdot10^{110} + 1\cdot10^{101} + 0\cdot10^{100} + 1\cdot10^{11} + 1\cdot10^{10} + 0\cdot10^{1} +$ (2-ndsüst.)

$+ 0\cdot10^{0} = 1\cdot2^{10} + 1\cdot2^{9} + 1\cdot2^{8} + 1\cdot2^{7} + 1\cdot2^{6} + 1\cdot2^{5} + 0\cdot2^{4} +$

$+ 1\cdot2^{3} + 1\cdot2^{2} + 0\cdot2^{1} + 0\cdot2^{0} = 1024 + 512 + 256 + 128 + 32 +$ (10-ndsüst.)

$+ 8 + 4 = 1964_{10}$

$3654_8 = 3\cdot10^{3} + 6\cdot10^{2} + 5\cdot10^{1} + 4\cdot10^{0} = 3\cdot8^{3} + 6\cdot8^{2} + 5\cdot8 +$ (8-ndsüst.) (10-nd-

$+ 4 = 1536 + 384 + 40 + 4 = 1964_{10}$ -süst.)

$7\bar{0}\bar{2}_{16} = 7\cdot10^{2} + 0\cdot10^{1} + 2\cdot10^{0} = 7\cdot16^{2} + 10\cdot16 + 12 = 1792 +$ (16-ndsüst.) (10-ndsüst.)

$+ 160 + 12 = 1964_{10}$

Toodud näidetest selgub, et ühes arvusüsteemi aluse suurendamisega lühenevad üldiselt ka arvude esitused kujul (1).

Arvude (õigemini: arvude kirjutiste või esituste) teisendamist ühest arvusüsteemist teise käsitleme paragrahvis 2. Praegu vaatleme arvutuste teostamist 2-nd-, 8-nd- ja 16-ndsüsteemi arvudega. Märgime, et üldiselt on arvutuste teostamine seda lihtsam, mida väiksem on arvusüsteemi alus.

Kõige lihtsam on aritmeetiliste tehete teostamine 2-ndsüsteemis. Aritmeetiliste tehete sooritamiseks on vaja teada ühekohaliste arvude liitmis- ja korrutamistabeleid (lahutamine ja jagamine on teostatavad samuti nende abil), mis 2-ndsüsteemi korral on ainult kaherealised

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 10 |

| × | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Toome mõned näited tehete teostamise kohta 2-ndsüsteemis:

$$\begin{array}{r} 1011101{,}101 \\ +\ 111011{,}111 \\ \hline 10011001{,}100 \end{array} \qquad \begin{array}{r} 1011101{,}101 \\ -\ 111011{,}111 \\ \hline 100001{,}110 \end{array}$$

$$\begin{array}{r} \underline{110{,}101 \times 11{,}01} \\ 110101 \\ 110101\phantom{0} \\ 110101\phantom{00} \\ \hline 10101{,}10001 \end{array} \qquad \begin{array}{l} 110111 : 101 = 1011 \\ \underline{101}\phantom{00} \\ \phantom{0}111 \\ \phantom{0}\underline{101} \\ \phantom{00}101 \\ \phantom{00}\underline{101} \end{array}$$

Seletusteks tehete kohta märgime ainult niipalju, et liitmisel ja lahutamisel tuleb vastavad numbrikohad asetada kohakuti, korrutamisel kehtib nn. komareegel ja osakorrutiste nihutamise reegel, jagamisel toimub numbrite juurdekirjutamine lahutamisel tekkivatele vahedele samuti nagu kümnendsüsteemis. Kõik need märkused kehtivad tehete korral mistahes p-ndsüsteemis ($p \geq 2$).

8-ndsüsteemi arvude liitmis- ja korrutamistabelid tulevad järgmised (alljärgnevates tabelites on ära jäetud read ja veerud arvu 0 jaoks):

| + | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
| 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 |
| 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 |
| 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 |
| 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 |
| 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

| X | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 2 | 4 | 6 | 10 | 12 | 14 | 16 |
| 3 | 3 | 6 | 11 | 14 | 17 | 22 | 25 |
| 4 | 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 7 | 16 | 25 | 34 | 43 | 52 | 61 |

Toome mõned näited tehete teostamise kohta 8-ndsüsteemis:

$$\begin{array}{r} 143{,}271 \\ +\ 26{,}544 \\ \hline 172{,}035 \end{array} \qquad \begin{array}{r} 143{,}271 \\ -\ 26{,}544 \\ \hline 114{,}525 \end{array}$$

$$\begin{array}{r} 21{,}74 \times 15{,}3 \\ \hline 6564 \\ 13154 \\ 2174 \\ \hline 357{,}724 \end{array} \qquad \begin{array}{l} 1450{,}22 : 43 = 27{,}06 \\ 106 \\ 370 \\ 365 \\ 322 \\ 322 \end{array}$$

Toome mõned näited tehete teostamise kohta 16-ndsüsteemis:

$$\begin{array}{r} \bar{2}74{,}5\bar{0} \\ +\ 96\bar{3}{,}\bar{1}8 \\ \hline 15\bar{4}2{,}12 \end{array} \qquad \begin{array}{r} \bar{2}74{,}5\bar{0} \\ -\ 96\bar{3}{,}\bar{1}8 \\ \hline 306{,}\bar{0}2 \end{array} \qquad \begin{array}{r} 45{,}3\bar{2} \times 7{,}\bar{1} \\ \hline 36794 \\ 22\bar{0}\bar{0}4 \\ \hline 261{,}1\bar{3}4 \end{array}$$

$$\begin{array}{l} 98\bar{3}\bar{0}{,}98 : \bar{3}8 = \bar{1}5{,}29 \\ 948 \\ 450 \\ 438 \\ 229 \\ 110 \\ 798 \\ 798 \end{array}$$

16-ndsüsteemi arvude liitmis- ja korrutamistabel on juba ülatuslikum:

| + | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 |
| 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 |
| 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 |
| 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | $1\bar{0}$ |
| $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | $1\bar{0}$ | $1\bar{1}$ |
| $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | $1\bar{0}$ | $1\bar{1}$ | $1\bar{2}$ |
| $\bar{4}$ | $\bar{5}$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | $1\bar{0}$ | $1\bar{1}$ | $1\bar{2}$ | $1\bar{3}$ |
| 5 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | $1\bar{0}$ | $1\bar{1}$ | $1\bar{2}$ | $1\bar{3}$ | $1\bar{4}$ |

| X | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ |
| 2 | 2 | 4 | 6 | 8 | $\bar{0}$ | $\bar{2}$ | $\bar{4}$ | 10 | 12 | 14 | 16 | 18 | $1\bar{0}$ | $1\bar{2}$ | $1\bar{4}$ |
| 3 | 3 | 6 | 9 | $\bar{2}$ | $\bar{5}$ | 12 | 15 | 18 | $1\bar{1}$ | $1\bar{4}$ | 21 | 24 | 27 | $2\bar{0}$ | $2\bar{3}$ |
| 4 | 4 | 8 | $\bar{2}$ | 10 | 14 | 18 | $1\bar{2}$ | 20 | 24 | 28 | $2\bar{2}$ | 30 | 34 | 38 | $3\bar{2}$ |
| 5 | 5 | $\bar{0}$ | $\bar{5}$ | 14 | 19 | $1\bar{4}$ | 23 | 28 | $2\bar{3}$ | $3\bar{2}$ | 37 | $3\bar{2}$ | 41 | 46 | $4\bar{1}$ |
| 6 | 6 | $\bar{2}$ | 12 | 18 | $1\bar{4}$ | 24 | $2\bar{0}$ | 30 | 36 | $3\bar{2}$ | 42 | 48 | $4\bar{4}$ | 54 | $5\bar{0}$ |
| 7 | 7 | $\bar{4}$ | 15 | $1\bar{2}$ | 23 | $2\bar{0}$ | 31 | 38 | $3\bar{5}$ | 46 | $4\bar{3}$ | 54 | $5\bar{1}$ | 62 | 69 |
| 8 | 8 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| 9 | 9 | 12 | $1\bar{1}$ | 24 | $2\bar{3}$ | 36 | $3\bar{5}$ | 48 | 51 | $5\bar{0}$ | 63 | $6\bar{2}$ | 75 | $7\bar{4}$ | 87 |
| $\bar{0}$ | $\bar{0}$ | 14 | $1\bar{4}$ | 28 | $3\bar{2}$ | $3\bar{2}$ | 46 | 50 | $5\bar{0}$ | 64 | $6\bar{4}$ | 78 | 82 | $8\bar{2}$ | 96 |
| $\bar{1}$ | $\bar{1}$ | 16 | 21 | $2\bar{2}$ | 37 | 42 | $4\bar{3}$ | 58 | 63 | $6\bar{4}$ | 79 | $8\bar{4}$ | $8\bar{5}$ | $9\bar{0}$ | $\bar{0}5$ |
| $\bar{2}$ | $\bar{2}$ | 18 | 24 | 30 | $3\bar{2}$ | 48 | 54 | 60 | $6\bar{2}$ | 78 | 84 | 90 | $9\bar{2}$ | $\bar{0}8$ | $\bar{1}4$ |
| $\bar{3}$ | $\bar{3}$ | $1\bar{0}$ | 27 | 34 | 41 | $4\bar{4}$ | 51 | 68 | 75 | 82 | $8\bar{5}$ | $9\bar{2}$ | $\bar{0}9$ | $\bar{1}6$ | $\bar{2}3$ |
| $\bar{4}$ | $\bar{4}$ | $1\bar{2}$ | $2\bar{0}$ | 38 | 46 | 54 | 62 | 70 | $7\bar{4}$ | $8\bar{2}$ | $9\bar{0}$ | $\bar{0}8$ | $\bar{1}6$ | $\bar{2}4$ | $\bar{3}2$ |
| $\bar{5}$ | $\bar{5}$ | $1\bar{4}$ | $2\bar{3}$ | $3\bar{2}$ | $4\bar{1}$ | $5\bar{0}$ | 69 | 78 | 87 | 96 | $\bar{0}5$ | $\bar{1}4$ | $\bar{2}3$ | $\bar{3}2$ | $\bar{4}1$ |

## § 2. Arvude teisendamine ühest positsioonilisest arvusüsteemist teise.

Eelmises paragrahvis esitasime rea võrdusi kujul

$$100_{10} = 1\ 100\ 100_2 = 144_8 = 64_{16}.$$

Nende kehtivust on lihtne kontrollida. Käesolevas paragrahvis anname meetodi niisuguste võrduste leidmiseks, s.t. arvude (täpsemini: arvude esituste) teisendamiseks ühest positsioonilisest arvusüsteemist teise.

Arvude teisendusi on otstarbekohane vaadelda eraldi täis- ja murdarvude ( $|a|<1$) puhul.

1. Täisarvude teisendamine.

Olgu arv A esitatud p-ndsüsteemis. Tahame teda teisendada q-ndsüsteemi (arv q on esitatud p-ndsüsteemis). Arvu A saab esitada kujul

$$A = \alpha_m \cdot q^m + \alpha_{m-1} q^{m-1} + \ldots + \alpha_1 \cdot q^1 + \alpha_0 , \qquad (1)$$

kus $\alpha_m, \alpha_{m-1}, \ldots \alpha_1, \alpha_0$ on mittenegatiivsed arvust q väiksemad täisarvud (p-ndsüsteemis). Nagu eespool märgitud, on esitus kujul (1) ühene. Seega annavadki kordajad $\alpha_m, \alpha_{m-1}, \ldots$ $\ldots, \alpha_1, \alpha_0$ arvu A esituse q-ndsüsteemis (kordajad tuleb ainult asendada vastavate q-ndsüsteemi numbritega).

Kordajad avaldises (1) leiame järkjärgulise jagamise teel arvuga q. Jagamine toimub p-ndsüsteemis. Jagades võrduse (1) mõlemad pooled arvuga q, saame

$$\frac{A}{q} = \alpha_m \cdot q^{m-1} + \alpha_{m-1} \cdot q^{m-2} + \ldots + \alpha_1 + \frac{\alpha_0}{q} ,$$

s.t. jagamisel tekkinud jääk $\alpha_0$ annabki arvu A q-ndsüsteemi esituse viimase koha. Edasi vaatleme jagamisel tekkinud täisosa

$$A' = \alpha_m \cdot q^{m-1} + \alpha_{m-1} \cdot q^{m-2} + \ldots + \alpha_1 .$$

Jagades selle arvuga q, saame jäägi $\alpha_1$, s.t. arvu A q-ndsüsteemi esituse eelviimase koha. Teostades sellist jagamist m korda, leiame arvu A kõik q-ndsüsteemi kohad. Protsessi lõpu määrab viimane täisosa

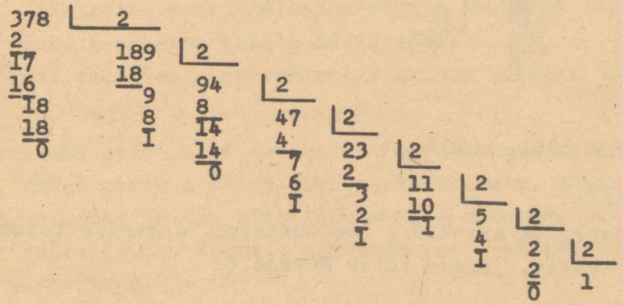
$$A^{(m)} = \alpha_m ,$$

mis osutub väiksemaks arvust q.

Toome nüüd näiteid arvude teisendamise kohta ühest süsteemist teise.

Näide 1.

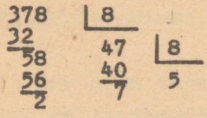
Teisendada arv $378_{10}$ 2-ndsüsteemi.




Seega $378_{10} = 101\ 111\ 010_2$.

Näide 2.

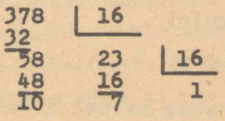
Teisendada arv $378_{10}$ 8-ndsüsteemi.




Seega $378_{10} = 572_8$.

Näide 3.

Teisendada arv $378_{10}$ 16-ndsüsteemi.




Seega $378_{10} = 17\overline{0}_{16}$ (jäägile 10 vastab 16-ndsüsteemis number $\overline{0}$).

Näide 4.

Teisendada arv $1361_8$ 10-ndsüsteemi. Jagamine toimub siin 8-ndsüsteemis, jagada tuleb arvuga $12_8 = 10_{10}$.

```
1361 | 12
12     113 | 12
 16    106   7
 12      5
  41
  36
   5
```

Seega $1361_8 = 753_{10}$.

Näide 5.

Teisendada arv $\bar{2}39_{16}$ 8-ndsüsteemi. Jagamine toimub siin 16-ndsüsteemis, jagada tuleb arvuga 8.

```
Z39 | 8
8     187 | 8
 43   18    30 | 8
 40    7    30   6
  39         0
  38
   1
```

Seega $\bar{2}39_{16} = 6071_8$.

2. Murdude teisendamine.

Olgu lihtmurd A ( $|A| < 1$) esitatud p-ndsüsteemis lõpliku p-ndnumbrite jadaga

$$A = 0, \alpha_1 \alpha_2 \dots \alpha_m$$

Tahame arvu A teisendada q-ndsüsteemi (arv q on esitatud p-ndsüsteemis). Arvu A saab esitada kujul

$$A = \beta_1 \cdot q^{-1} + \beta_2 \cdot q^{-2} + \dots + \beta_n \cdot q^{-n} + \dots , \qquad (2)$$

kus $\beta_1, \beta_2, \dots, \beta_n, \dots$ on mittenegatiivsed arvust q väiksemad täisarvud (p-ndsüsteemis). Seejuures võib rida (2) tulla ka lõpmatu. Viimase väite kinnituseks olgu võrdus

$$0,1_3 = \frac{1}{3}_{10} = 0,333\dots_{10} .$$

Esitus (2) on ühene, kui sellised kirjutised nagu $0,999\dots_{10}$, $0,777\dots_8$, $0,111\dots_2$ asendame kirjutisega $1,000\dots$ .

Kordajad avaldises (2) leiame järkjärgulise korrutamise teel arvuga q. Korrutamine toimub p-ndsüsteemis. Korrutades võrduse (2) mõlemad pooled arvuga q, saame

$$A \cdot q = \beta_1 + \beta_2 \cdot q^{-1} + \ldots + \beta_n \cdot q^{-n+1} + \ldots .$$

Seega - lihtmurru korrutamisel arvuga q tekkinud täisosa $\beta_1$ annab murru A esimese koha q-ndsüsteemis.

Edasi vaatleme korrutamisel tekkinud murdosa

$$A' = \beta_2 \cdot q^{-1} + \beta_3 \cdot q^{-2} + \ldots + \beta_n \cdot q^{-n+1} + \ldots$$

ja korrutame seda jälle arvuga q. Korrutamisel tekkinud täisosa $\beta_2$ annab murru A teise koha q-ndsüsteemis. Samasugusel viisil jätkates leiame kõik järgnevad q-ndkohad. Leitud suurused $\beta_i$ tuleb ($p < q$ korral) asendada veel vastavate q-ndsüsteemi numbritega.

Kuid ülaltoodud murdude teisendamise menetlus võib osutuda lõpmatuks (juhul, kui rida (2) on lõpmatu). Seega peab meil olema mingi lisakriteerium teisendamisprotsessi lõpetamiseks.

Lisakriteeriumi tuletamisel tuleb arvestada kaht juhtumit:

1) arv A on täpne (näiteks mingi mastaabikordaja),
2) arv A on ligikaudne.

Esimesel juhtumil tuleb silmas pidada, et masinas opereeritakse kindla pikkusega arvudega. Täpsete arvude teisendamisel tuleb teisendamisel leida maksimaalne võimalik kohtade arv, s.t. parajasti sama palju kohti kui mitme kohaliste arvudega see masin opereerib.

Ligikaudsete arvude puhul eeldame, et arvud on antud kujul, kus viga ei ületa poolt viimase koha ühikut. Ligikaudsete arvude teisendamisel ei tohi arvu täpsust muidugi vähendada, kuid seda pole võimalik ka suurendada. Seega, kui arv A on antud p-ndsüsteemis m kohaga, siis on otstarbekohane q-ndsüsteemis esitada see n kohaga, kus n on vähim võrratust

$$\frac{1}{2} \cdot p^{-m} \geq \frac{1}{2} \cdot q^{-n} \qquad (3)$$

rahuldav täisarv. Võrratusest (3) saame

$$n \geq \frac{m \log p}{\log q} . \qquad (4)$$

Seega: leitav kohtade arv n on vähim võrratust (4) rahuldav täisarv. Praktiliselt arvutame ühe koha enam ja teostame ümardamise.

Toome nüüd näiteid murdude teisendamise kohta ühest süsteemist teise.

Näide 6.

Teisendada ligikaudne arv $0,7250_{10}$ 2-nd-, 8-nd- ja 16-ndsüsteemi.

Kõigepealt leiame numbrikohtade arvu valemi (4) põhjal

2-ndsüsteemis: $n \geq \frac{4}{\log 2} \approx \frac{4}{0,301} \approx 13,3$, $n = 14$;

8-ndsüsteemis: $n \geq \frac{4}{\log 8} \approx \frac{4}{0,903} \approx 4,4$, $n = 5$;

16-ndsüsteemis: $n \geq \frac{4}{\log 16} \approx \frac{4}{1,2} \approx 3,3$, $n = 4$.

Arvutused teostame 10-ndsüsteemis järgmise skeemi kohaselt (2-, 8- ja 16-ga korrutatakse ainult murdosa):

| | 0,7250 × 2 |
|---|---|
| 1 | 4500 |
| 0 | 90 |
| 1 | 8 |
| 1 | 6 |
| 1 | 2 |
| 0 | 4 |
| 0 | 8 |
| 1 | 6 |
| 1 | 2 |
| 0 | 4 |
| 0 | 8 |
| 1 | 6 |
| 1 | 2 |
| 0 | 4 |
| 0 | 8 |

| | 0,7250 × 8 |
|---|---|
| 5 | 8000 |
| 6 | 4 |
| 3 | 2 |
| 1 | 6 |
| 4 | 8 |
| 6 | 4 |

| | 0,7250 × 16 |
|---|---|
| 11 | 6 |
| 9 | 6 |
| 9 | 6 |
| 9 | 6 |
| 9 | 6 |

Seega $0,7250_{10} = 0,10111001100110_2$,

$0,7250_{10} = 0,56315_8$,

$0,7250_{10} = 0,\overline{1990}_{16}$.

Näide 7. Teisendada ligikaudne arv $0,51403_8$ 10-ndsüsteemi. Numbrikohtade arvu leiame võrratuse põhjal:

$n \geqslant \frac{5 \cdot \log 8}{\log 10} \approx 5 \cdot 0,903 \approx 4,5$. Seega n = 5. Arvutused tuleb teostada 8-ndsüsteemis: murdosi on vaja korrutada arvuga $12_8 = 10_{10}$.

| 0 | 5 1 4 0 3 × 12 |
|---|---|
| 6 | 3 7 0 3 6 |
| 4 | 6 6 4 5 4 |
| 10 | 4 1 6 7 0 |
| 5 | 2 2 4 6 0 |
| 2 | 7 1 7 4 0 |
| 11 | 0 3 3 0 0 |

Kirjutades saadud täisosad 10-ndsüsteemi numbrite abil, saame

$$0,51403_8 = 0,64853_{10}.$$

3. Kui arv A koosneb täisarvulisest ja murrulisest osast, siis selle arvu teisendamisel mingisse teise arvusüsteemi võib teisendada eraldi täis- ja murdosa ning seejärel need teisendatud osad liita. Kuid täis- ja murdosa võib teisendada ka koos.

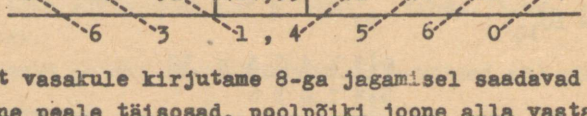
Näide 8. Teisendame ligikaudse arvu $417,59_{10}$ 8-ndsüsteemi. Teisendades eraldi täisosa, saame $471_{10} = 631_8$:

```
417 | 8
40    52 | 8
 17   48   6
 16    5
  1
```

Murdosa teisendamisel saame $0,59_{10} = 0,456_8$.

| 0 | 5 9 × 8 |
|---|---|
| 4 | 7 2 |
| 5 | 7 6 |
| 6 | 0 8 |
| 0 | 6 4 |

Kõiki neid arvutusi saab teha aga korraga järgmise skeemi kohaselt:




Komast vasakule kirjutame 8-ga jagamisel saadavad tulemused (joone peale täisosad, poolpõiki joone alla vastavad jäägid), komast paremale 8-ga korrutamisel saadavad tulemused (joone peale murdosad, poolpõiki joone alla vastavad täisosad). Teisendatud arv tuleb joone alla tavalisel kujul.

Negatiivsete arvude - C (C > 0) teisendamisel teisendame vastavasse süsteemi arvu C ja asetame tulemuse ette märgi "-".

4. Lõpuks vaatleme erijuhtu arvude teisendamisel, kus arvusüsteemi aluste vahel kehtib seos

$$p = q^k. \qquad (5)$$

Sel juhul on üleminek p-ndsüsteemist q-ndsüsteemi eriti lihtne. Nimelt vastab igale p-ndsüsteemi numbrile kindel ka numbrist koosnev q-ndsüsteemi numbrite rühm.

Näitame seda juhul, kui p = 8 ja q = 2. Kõigepealt märgime, et ühekohaliste 8-ndsüsteemi numbritele vastavad järgmised 2-ndsüsteemi arvude kolmikud:

| | |
|---|---|
| 0 = 000, | 4 = 100, |
| 1 = 001, | 5 = 101, |
| 2 = 010, | 6 = 110, |
| 3 = 011, | 7 = 111. |

Olgu nüüd mingi täisarvu A esitus 8-ndsüsteemis kujul

$$A = \alpha_m \alpha_{m-1} \cdots \alpha_2 \alpha_1 \alpha_0$$

ja 2-ndsüsteemis kujul

$$A = \beta_n \beta_{n-1} \cdots \beta_2 \beta_1 \beta_0 .$$

Jagades arvu A arvuga 8, saame ühel puhul jäägi $\alpha_0$, teisel puhul $\beta_2 \cdot 2^2 + \beta_1 \cdot 2^1 + \beta_0 \cdot 2^0$, mis peavad olema võrdsed, s.t.

$$\alpha_0 = \beta_2 \cdot 2^2 + \beta_1 \cdot 2^1 + \beta_0 \cdot 2^0,$$

Seega avaldub arvu A viimane 8-ndkoht 2-ndsüsteemi arvude kolmiku abil vastavalt ülaltoodud tabelile. Et arvu A jagamisel 8-ga tekivad ka võrdsed täisosad, siis neid omakorda 8-ga jagades saame, et

$$\alpha_1 = \beta_5 \cdot 2^2 + \beta_4 \cdot 2^1 + \beta_3 \cdot 2^0,$$

s.t. eelviimane 8-ndkoht avaldub vastava 2-ndsüsteemi arvude kolmiku abil jne.

Lihtmurdude puhul saame analoogilise tulemuse teisendatava arvu A arvuga 8 korrutamise teel.

Arvu teisendamisel 2-ndsüsteemist 8-ndsüsteemi tuleb teisendatava arvu numbrid komast alates jaotada 3-numbrilisteks rühmadeks. Seejärel tuleb iga numbrite rühm asendada vastava 8-ndsüsteemi numbriga.

Toome mõned näited arvude teisendamise kohta 8-ndsüsteemist 2-ndsüsteemi ja vastupidi.

Näide 9. Teisendada arv $357{,}641_8$ 2-ndsüsteemi. Asendades iga 8-ndsüsteemi numbri kolme 2-ndsüsteemi numbriga, saame

| 3 | 5 | 7 | , | 6 | 4 | 1 |
|---|---|---|---|---|---|---|
| 011 | 101 | 111 | , | 110 | 100 | 001 |

ehk (tarbetut nulli ära jättes)

$$357{,}641_8 = 11\ 101\ 111,\ 110\ 100\ 001_2.$$

Näide 10. Teisendada arv $1\ 111\ 010\ 110\ 100\ 11_2$ 8-ndsüsteemi. Rühmitades numbrid komast alates 3-kaupa, saame (fiktiivseid nulle algusesse ja lõppu juurde lisades)

| (00)1' | 111' | 010' | 110', | 011' | 100' | 11(0) |
|---|---|---|---|---|---|---|
| 1 | 7 | 2 | 6 , | 3 | 4 | 6 . |

Seega

$$1\ 111\ 010\ 110,\ 011\ 100\ 11_2 = 1726{,}346_8.$$

Lõpuks teeme veel ühe praktilise märkuse. Nagu eespool nägime, on arvude vahetu teisendamine 2-ndsüsteemi küllaltki aeganõudev protsess. Sellepärast on sageli kasulikum teisen-



3.

dada arv esialgu 8-ndsüsteemi, arvestades seda, et 8-ndsüsteemist 2-ndsüsteemi teisendamine toimub täiesti triviaalselt.

Näide 11. Teisendada arv $679,4825_{10}$ 2-ndsüsteemi. Teisendame arvu $679,4825_{10}$ kõigepealt 8-ndsüsteemi (murdosas arvutame 5 kohta). Saame

| | | | : 8 x | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 84 | 679,4825 | 8600 | 88 | 04 | 32 | 56 | 48 |
| 1 | 2 | 4 | 7, 3 | 6 | 7 | 0 | 2 | (4) | |

Seega $679,4825_{10} = 1247,36702_8 = 1\ 010\ 100\ 111,\ 011\ 110\ 111\ 000\ 010_2$.

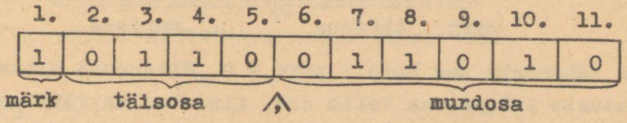
## § 3. Arvude kujutamine arvutusmasinais.

Arvude kujutamisel (salvestamisel) arvutusmasinate seadmetes kasutatakse enamikus 2-ndsüsteemi. Selle põhjuseks on asjaolu, et kahe stabiilse seisundiga elemendid (võrreldes elementidega, millel on seisundite hulk suurem) on konstruktsioonilt kõige lihtsamad. Samuti on aritmeetiliste tehete sooritamine kõige lihtsam 2-ndsüsteemis. Ühe arvu kujutamiseks reserveeritakse arvutusmasinates kindel kahendkohtade hulk, mida nimetatakse pesaks (v.k. ячейка). Nii on näiteks arvutusmasina БЭСМ pesas 39 kahendkohta, "Strela" pesas 43 kahendkohta, "Ural-1" 36, "Ural-2" ja "Ural-4" 40 kahendkohta.

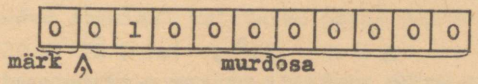
Vastavalt arvutusmasina seadmete ehitusele jaotatakse kõik arvutusmasinad 3 liiki: fikseeritud komaga, liikuva komaga ja nn. segakomaga masinad.

Fikseeritud komaga arvutusmasinates on arvu täis- ja murdosa jaoks eraldatud pesas kindlad kahendkohad. Näiteks võib masina 11-kahendkohalises pesas olla kahendkohad nr. 2.-5. reserveeritud täisosa jaoks, kahendkohad nr.6.-11. murdosa jaoks, esimene kahendkoht aga märgi jaoks. Märgi "+" kodeerime sümboliga 0, märgi "-" aga sümboliga 1. Nii

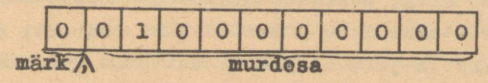
kujutatakse arvu -110,01101 selles pesas järgmiselt:




Koma arvutusmasina pesas muidugi eraldi kujutada pole vaja: arvutusmasina seadmed ise on selliselt konstrueeritud, et tehted pesades olevate arvudega teostatakse nii, nagu asuks koma pärast viiendat kahendkohta. Põhimõtteliselt võib koma fikseerida mitmesuguste kahendkohtade järel. Koma fikseerimine mõjustab õieti ainult korrutamist ja jagamist, liitmine-lahutamine toimuvad koma asukohast sõltumatult. Nii näiteks, kui koma on fikseeritud pärast märgikohta, annab pesades

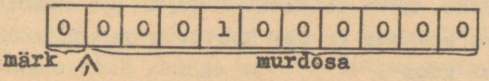



(1)

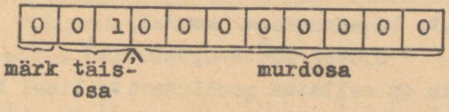



(2)

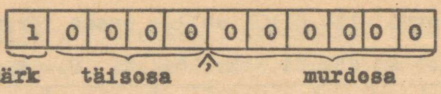
olevate arvude korrutamine tulemuse:




Kui koma oleks fikseeritud pärast kolmandat kahendkohta (märgikoht kaasa arvatud), siis annaks pesades (1) ja (2) olevate arvude korrutamine tulemuse:




Kui aga koma oleks fikseeritud pärast viiendat kahendkohta, siis annaks pesades (1) ja (2) olevate arvude korrutamine tulemuse: 10000 (=100 × 100). Seda arvu ei saa aga pesas enam kujutada, sest seal on täisosa jaoks reserveeritud ainult 4 kohta. Kui seda arvu kujutada pesas nii:




siis tähendaks see hoopis arvu - 0. Niisugust olukorda, kus arvutuste tulemusena tekib arv, mis absoluutväärtuse poolest ületab maksimaalse pesas kujutatava arvu, nimetatakse ületäitumiseks. Ületäitumine võib täielikult moonutada tehte tulemuse ja neid tuleb seetõttu vältida. Põhimõtteliselt võib ületäitumine tekkida kõigi 4 aritmeetilise tehte puhul. Et ületäitumiste võimaluste arvestamist kergendada, fikseeritakse koma tavaliselt kohe pärast märgikohta. See tähendab seda, et pesas kujutatakse arve A, kus $|A| < 1$. Nii on see näiteks arvutusmasinas "Ural-1" ja paljudes teistes masinates.

Arvude kujutamine liikuva komaga masinates põhineb seosel

$$A = m \cdot 10^p , \quad |m| < 1, \qquad (3)$$

kus 10 tähistab mingi positsioonilise q-ndsüsteemi alust. Suurust m nimetatakse arvu A mantissiks, täisarvu p arvu A järguks. Mantiss ja järk on samuti esitatud q-ndsüsteemis. Kujul (3) esitatud arve nimetatakse poollogaritmilisel kujul esitatud arvudeks. Arvu esitamisel kujul (3) määrab järk p sisuliselt koma asukoha arvus. Seetõttu on põhjendatud ka nimetus "liikuv koma".

Arvu esitamine kujul (3) pole ühene. Nii on näiteks (q = 2, s.t. 2-ndsüsteemi korral)

$$0{,}10100 \cdot 10^1 = 0{,}01010 \cdot 10^{10}$$

või (q = 10, s.t. 10-ndsüsteemi korral)

$$0{,}0592 \cdot 10^5 = 0{,}5920 \cdot 10^4 .$$

Arvu A, mis on esitatud poollogaritmilisel kujul, kus mantissi esimene koht peale koma on nullist erinev

$$0{,}1 \ldots \leq m < 1 \qquad (4)$$

nimetatakse normaliseeritud arvuks. Arvu (3) teisendamist kujule, kus on täidetud ka tingimus (4), nimetatakse arvu normaliseerimiseks.

Edaspidi vaatleme poollogaritmilisel kujul esitatud arve, kus q = 2. Ülevaatlikkuse huvides kirjutame aluse "kaks"

harilikul kujul "2". Seega kirjutame seose (3) lihtsalt nii:

$$A = m \cdot 2^p, \qquad |m| < 1 \ . \qquad (3')$$

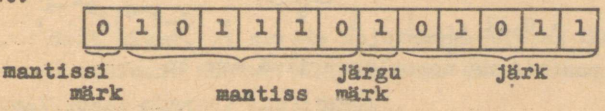
Tingimust (4) võib 2-ndsüsteemi korral kirjutada

$$10^{-1} \leqslant m < 1$$

ehk lihtsalt

$$\frac{1}{2} \leqslant m < 1 \ .$$

Liikuva komaga masinate pesades on reserveeritud kindel arv kahendkohti arvu mantissi ja arvu järgu kujutamiseks. Kahendkohtade jaotus pesas näeb põhimõtteliselt välja selliselt:

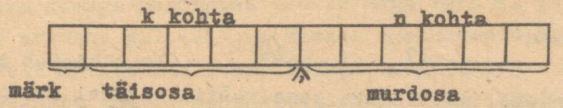



Ülaltoodud joonisel kujutatud pesas salvestatakse praegu arv $+0{,}10111_0 . 2^{-1011}$.

Uuemates arvutusmasinates (näiteks "Ural-2" ja "Ural-4") on võimalik arvutada nii fikseeritud kui ka liikuva koma režiimis. Selliseid arvutusmasinaid nimetatakse segakomaga masinateks.

Edasi vaatleme fikseeritud ja liikuva komaga arvutusmasinate pesades kujutatavate arvude diapasooni.

Kui fikseeritud komaga arvutusmasina pesas on k kahendkohta täisosa ja n kahendkohta murdosa kujutamiseks:




siis selles pesas kujutatav maksimaalne arv $A_{max}$ on

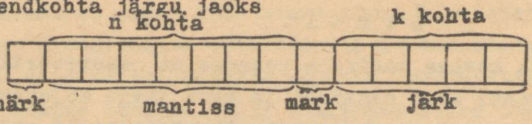
$$A_{max} = \underbrace{11 \ldots 1}_{k}, \ \underbrace{11 \ldots 1}_{n} = 2^k - 2^{-n} \ .$$

Minimaalne positiivne arv, mida arvutusmasina pesas saab kujutada, on

$$|A|_{min} = 0,\underbrace{00 \ldots 01}_{n} = 2^{-n} \ .$$

Arvust $|A|_{min}$ absoluutväärtuselt väiksemad arvud kujutatakse arvutusmasina pesas juba nullidena. Need on nn. masina-nullid. Masina-nullideks nimetatakse nullist erinevaid arve, mis masinas kujutatakse nullidena.

Kui liikuva komaga masina pesas on n kahendkohta mantissi ja k kahendkohta järgu jaoks




siis selles pesas kujutatav maksimaalne arv on

$$A_{max} = 0,\underbrace{11\ldots1}_{n} \cdot 2^{\overbrace{1\ldots1}^{k}} = (1-2^{-n})2^{2^k-1}.$$

Minimaalne positiivne normaliseeritud arv on

$$|A|_{min\ norm} = 0,\underbrace{10\ldots0}_{n} \cdot 2^{\overbrace{1\ldots1}^{k}} = 2^{-1} \cdot 2^{-2^k+1} = 2^{-2^k}.$$

Seega asuvad nullist erinevad normaliseeritud arvude absoluutväärtused diapasoonis

$$2^{-2^k} \leqslant |A| \leqslant (1-2^{-n})\ 2^{2^k-1}.$$

Arvutusmasinatel "Ural-2" ja "Ural-4" on liikuva koma režiimi korral n=32 ja k=6. Seega asuvad normaliseeritud arvud diapasoonis

$$2^{-64} \leqslant |A| \leqslant (1-2^{-32})\ 2^{63}$$

ehk ligikaudu

$$10^{-19} \leqslant |A| \leqslant 10^{19}.$$ (10 tähistab kümmet)

Nagu näeme, on liikuva koma režiimi korral masina pesas kujutatavate arvude diapasoon küllaltki suur, mistõttu ületäitumist praktiliselt ei esine.

Vaatleme nüüd arvude kujutamise täpsust fikseeritud ja liikuva komaga arvutusmasinates. Seejuures arvestame, et absoluutne viga arvude kujutamisel ei ületa poolt viimase koha (resp. mantissi viimase koha) ühikut. Seega tuleb absoluutne viga fikseeritud komaga maisnais

$$|\Delta| \leqslant \frac{1}{2} \cdot 2^{-n}$$

ja liikuva komaga masinais

$$|\Delta| \leq \frac{1}{2} \cdot 2^{-n} \cdot 2^{p}.$$

Arvu kujutamise suhteline viga (ehk täpsus) tuleb siis fikseeritud komaga masinais

$$\delta_{min} = \frac{|\Delta|}{A_{max}} = \frac{\frac{1}{2} \cdot 2^{-n}}{2^{k} - 2^{-n}} \approx 2^{-n-k-1},$$

$$\delta_{max} = \frac{|\Delta|}{|A|_{min}} = \frac{\frac{1}{2} \cdot 2^{-n}}{2^{-n}} = 2^{-1}.$$

Siit näeme, et arvude kujutamise täpsus fikseeritud komaga masinais muutub väga suureks diapasoonis ja võib halvimal juhul ulatuda kuni 50%-ni.

Arvu kujutamise täpsus liikuva komaga masinates (vaatleme ainult normaliseeritud arve) on järgmistes piirides:

$$\delta_{min} = \frac{|\Delta|}{A_{max}} = \frac{\frac{1}{2} \cdot 2^{-n} \cdot 2^{2^{k}-1}}{(1 - 2^{-n})2^{2^{k}-1}} \approx 2^{-n-1},$$

$$\delta_{max} = \frac{\Delta}{|A|_{\substack{min \\ norm}}} = \frac{\frac{1}{2} \cdot 2^{-n} : 2^{-2^{k}+1}}{2^{-2^{k}}} = 2^{-n}.$$

Siit näeme, et normaliseeritud arvude kujutamise täpsus liikuva komaga masinates muutub väga vähe ja praktiliselt ei sõltu arvust. Kui liikuva komaga masinates opereerida aga üldiselt mittenormaliseeritud arvudega, siis muutub täpsus samasuguses suures diapasoonis nagu fikseeritud komaga masinate puhul.

Kuna arvutuste täpsus määrab lõpptulemuse täpsuse, siis tuleb arvutuste teostamisel tähelepanu pöörata arvutuste täpsuse tõstmisele ja säilitamisele. Liikuva komaga masinate puhul tähendab see seda, et tuleb opereerida normaliseeritud arvudega. Fikseeritud komaga masinate puhul tuleb aga võimaluse korral vältida nulle koma ja esimese nullist erineva numbrikoha vahel. Seda on enamikus võimalik saavutada arvudele sobivate mastaabikordajate leidmise teel.

Käesoleva paragrahvi tulemustest nähtuvad liikuva komaga masinate mitmed eelised fikseeritud komaga masinate ees. Tekib küsimus: milleks üldse konstrueerida fikseeritud komaga masinaid. Põhjus on lihtne - fikseeritud komaga masinad on lihtsama ehitusega ja loomulikult ka odavamad. Pealegi on väga paljud ülesanded kergesti kohandatavad fikseeritud komaga masinate suhteliselt väiksematele võimalustele. Viimase asjaolu tõttu on loodud ka eespool nimetatud segakomaga masinad.

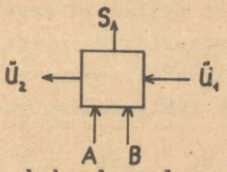
## § 4. Kahendarvude summaatorid. Arvude liitmine nende abil.

Selgitades aritmeetiliste tehete teostamist arvutusmasinates, on paratamatult vaja alustada kahendarvude summaatori põhimõttelisest ehitusest, sest see seade võtab suuremal või vähemal määral osa kõikide aritmeetiliste tehete sooritamisest.

Esmalt vaatleme ühekohalist kahendarvude summaatorit. See on seade 3 ühekohalise kahendarvu liitmiseks. Tal on 3 sisendit (arvude A, B ja ülekande $Ü_1$ jaoks) ja 2 väljundit (tulemuse madalama koha S ja kõrgema koha (ülekande) $Ü_2$ jaoks). Tema tööd iseloomustab järgmine tabel:

| A | B | $Ü_1$ | S | $Ü_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

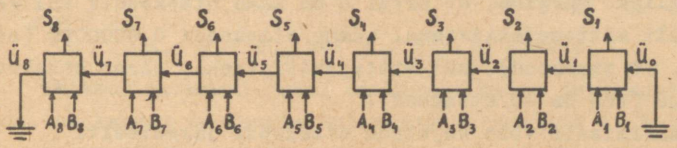
Skemaatiliselt kujutame seda seadet nii:




Ühekohalise kahendarvude summaatori üksikasjalikumat ehitust vaatleme paragrahvis 10.

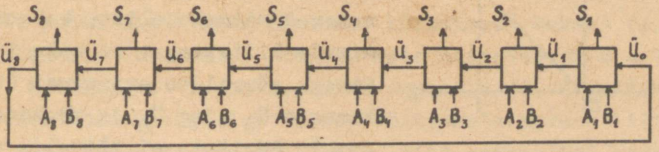
Mitme ühekohalise kahendarvude summaatori ühendamisel saame mitmekohalise kahendarvude summaatori. Olenevalt ühekohaliste summaatorite ühendamise viisist saame kaht tüüpi mitmekohalisi summaatoreid:

1. mitmekohaline summaator ilma ülekandeta kõrgeimast järgust (ehk 1. tüüpi summaator)




Ülekande $\ddot{U}_8$ maandamine tähendab seda, et see ülekanne ei leia kasutamist, $\ddot{U}_0$ maandamine tähendab, et $\ddot{U}_0 = 0$ samaselt.

2. mitmekohaline summaator tsüklilise ülekandega kõrgeimast järgust madalaimasse järku (ehk 2. tüüpi summaator)




Tähistame 1. ja 2. liiki summaatoris teostatava tehte vastavalt tehtemärkidega $\oplus$ ja $\boxplus$. Nii näiteks

$$\begin{array}{rr} & 0\,1\,1\,0\,1\,1\,1\,0 \\ \oplus & 1\,0\,1\,0\,1\,0\,1\,1 \\ \hline (1) & 0\,0\,0\,1\,1\,0\,0\,1 \end{array} \qquad \begin{array}{rr} & 0\,1\,1\,0\,1\,1\,1\,0 \\ \boxplus & 1\,0\,1\,0\,1\,0\,1\,1 \\ \hline (1) & 0\,0\,0\,1\,1\,0\,0\,1 \\ & 0\,0\,0\,1\,1\,0\,1\,0 \end{array}$$

Eeldame nüüd, et koma on fikseeritud pärast märgikohta ja arvud on 2-ndsüsteemis. Et selliseid arve summaatoris liita, on mingil viisil vaja kodeerida ka märki. Kui kodeerime märgi "+" 0 abil, märgi "-" 1 abil, saame nn. otsekoodi. Täpne definitsioon on järgmine: lihtkahendmurru x ( $|x| < 1$ ) otsekood

$$[x]_{ok} = \begin{cases} x, & \text{kui } x \geqslant 0, \\ 1-x, & \text{kui } x \leqslant 0. \end{cases}$$

Nii on näiteks

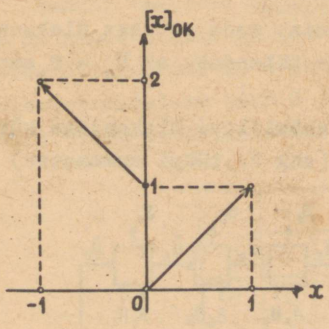
$$[+\,0,\,1\,0\,1\,1\,0\,1\,0]_{ok} = 0,\,1\,0\,1\,1\,0\,1\,0$$

$$[-\,0,\,1\,0\,1\,1\,0\,1\,0]_{ok} = 1-(-0,\,1\,0\,1\,1\,0\,1\,0) =$$

$$= 1,\,0\,1\,1\,0\,1\,0,$$

mis on tõepoolest kooskõlas otsekoodi definitsioonile eelnenud reegliga. Märgime, et arvul 0 on kaks otsekoodi: kui vaatleme nulli mittenegatiivsena, saame otsekoodi 0,0000000 (see on n.-ö. +0 otsekood), kui mittepositiivsena, saame otsekoodi 1,0000000 (see on -0 otsekood).

Graafiliselt võib kujutada otsekoodi järgmiselt:




Otsekoodi definitsioonist ja jooniselt võime järeldada, et lihtkahendmurdude otsekoode võib vaadelda kui arve poollõigul $[0, 2)$.

Kuigi tehted $\oplus$ ja $\boxplus$ arvude otsekoodidega mõnevõrra sarnanevad mitmekohaliste kahendarvude liitmisega, on siin liitmisega võrreldes erinevusi (ülekannete $Ü_0$ ning $Ü_8$ ja märkide osas). Seetõttu ei võimalda need summaatorid arvude liitmist otsekoodis. Väite kinnituseks olgu järgmine näide:

$$\begin{array}{rr} & -0,\,1\,0\,0\,1\,0\,1\,0 \\ + & 0,\,0\,1\,1\,0\,1\,1\,1 \\ \hline & -0,\,0\,0\,1\,0\,0\,1\,1 \end{array}$$

(tulemuse otsekood on 1,0010011).

Liites 1. ja 2. tüüpi summaatorite abil liidetavate otsekoodid, saame

$$\begin{array}{rl} \oplus & 1,\,1\,0\,0\,1\,0\,1\,0 \\ & 0,\,0\,1\,1\,0\,1\,1\,1 \\ \hline (1) & 0,\,0\,0\,0\,0\,0\,0\,1 \end{array} \qquad \text{ja} \qquad \begin{array}{rl} \boxplus & 1,\,1\,0\,0\,1\,0\,1\,0 \\ & 0,\,0\,1\,1\,0\,1\,1\,1 \\ \hline & 0,\,0\,0\,0\,0\,0\,1\,0, \end{array}$$

mis pole kumbki võrdsed tõelise tulemuse -0, 0 0 1 0 0 1 1 otsekoodiga 1, 0 0 1 0 0 1 1.

Kui 1. ja 2. liiki summaatoris on koma fikseeritud pärast esimest kohta (esimene koht on üheliste koht), siis võime väita järgmist.

Kahest väiksemate mittenegatiivsete arvude x ja y puhul annab 1. liiki summaator tulemuse:

$$x \oplus y = \begin{cases} x + y, & \text{kui } x + y < 10 \text{ (kaks)}, \\ x + y - 10, & \text{kui } x + y \geqslant 10 \text{ (kaks)}. \end{cases} \qquad (1)$$

Tulemus esimesel juhul on ilmne. Teise juhu kohta märgime, et kahest suuremate tulemuste (olenevalt liidetavatest on tulemus alati neljast väiksem) puhul läheb kaduma ülekanne kõrgeimast kohast, s.t. 10 (kaks).

Kahest väiksemate mittenegatiivsete arvude x ja y puhul annab 2. liiki summaator tulemuse:

$$x \boxplus y = \begin{cases} x + y, & \text{kui } x + y < 10 \text{ (kaks)}, \\ x + y - 10 + 10^{-n}, & \text{kui } x + y \geqslant 10 \text{ (kaks)}. \end{cases} \qquad (2)$$

Teise juhu kohta tähendame, et kahest suuremate tulemuste puhul liidetakse ülekanne kõrgeimast kohast (s.t. 10) madalaimale (n-indale) kohale. Meie näidetes on n = 7.

Näitame, et 1. ja 2. liiki summaatorid võimaldavad absoluutväärtuse poolest ühest väiksemate arvude liitmist, kui need arvud on kodeeritud 1. liiki summaatori korral nn. täiendkoodis, 2. liiki summatori korral nn. pöördkoodis ja kui liitmisel ei teki ületäitumist.

Esmalt tuleb aga esitada täiendkoodi ja pöördkoodi definitsioonid.

Lihtkahendmurru x (s.t. $|x| < 1$) <u>täiendkoodi</u> $[x]_{tk}$

defineerime järgmiselt:

$$[x]_{tk} = \begin{cases} x, & \text{kui } x \geq 0, \\ 10 + x, & \text{kui } x < 0. \end{cases}$$

Lihtkahendmurru x pöördkoodi $[x]_{pk}$ defineerime järgmiselt:

$$[x]_{pk} = \begin{cases} x, & \text{kui } x \geqslant 0, \\ 10 - 10^{-n} + x, & \text{kui } x \leqslant 0. \end{cases}$$

Nii on näiteks

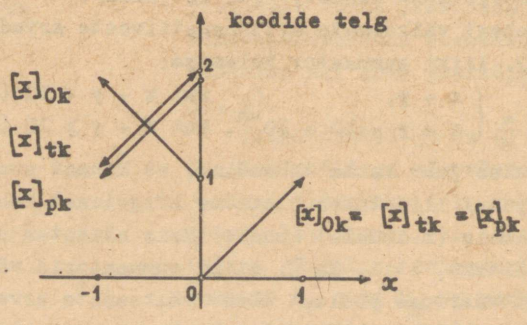
$$[+0,1101011]_{tk} = [+0,1101011]_{pk} = 0,1101011$$

$$[-0,0101101]_{tk} = 10+(-0,0101101) = 1,1010011$$

$$[-0,0101101]_{pk} = 10-0,0000001+(-0,0101101) = 1,1010010.$$

Arvul 0 on üksainus täiendkood, see on 0,0000000. Pöördkoode on arvul 0 kaks: $[+0,0000000]_{pk} = 0,0000000$ ja $[-0,0000000]_{pk} = 1,1111111$.

Üldiselt öeldes: positiivse arvu otsekood, täiendkood ja pöördkood ühtivad. Negatiivsete arvude korral on nad aga kõik erinevad.




Arvude otse-, täiend- ja pöördkoodi graafiline kujutamine.

Negatiivsete arvude täiendkoodi saab leida praktiliselt kõige lihtsamini valemist

$$[-0,x_1x_2\ldots x_n]_{tk} = 1,\bar{x}_1\bar{x}_2\ldots\bar{x}_n+0,\underbrace{00\ldots01}_{n-1}, \qquad (3)$$

negatiivsete arvude pöördkoodi aga valemist

$$[-0,x_1x_2\ldots x_n]_{pk} = 1,\bar{x}_1\bar{x}_2\ldots\bar{x}_n, \qquad (4)$$

kus $x_1, x_2, \ldots, x_n$ on kahendnumbrid ja $\bar{0} = 1$ ning $\bar{1} = 0$. Valemid (3) ja (4) tuletada lugejatel.

Nagu ka joonisest nähtub, võime täiend- ja pöördkoodi vaadelda teisendustena, mis teisendavad vahemikus $(-1,1)$ asuvad arvud poollõigul $[0,2)$ asuvateks arvudeks (koodideks). Teisendusi võime vaadelda üks-ühestena (täiendkoodi puhul jätame vaatlusest välja koodi 1,000000; pöördkoodi puhul vaatleme +0 ja -0 kui erinevaid arve. Lihtkahendmurru täiend- ja pöördkoodi võib soovi korral vaadelda kahest väiksemate mittenegatiivsete arvudena.

Tõestame nüüd eespool nimetatud põhilise tulemuse ühe osa:

Teoreem 1. Kui $|x| < 1$, $|y| < 1$ ja $|x + y| < 1$, siis

$$[x]_{tk} \oplus [y]_{tk} = [x + y]_{tk}. \qquad (5)$$

Seega: 1. liiki summaator võimaldab absoluutselt ühest väiksemate arvude liitmist, kui arvud on täiendkoodis ja kui liitmisel ei teki ületäitumist.

Tõestus. Võrduse (5) kehtivus tuleb näidata 4 erijuhul.

1) Mõlemad liidetavad on mittenegatiivsed. Seega: $0 \leqslant x < 1$, $0 \leqslant y < 1$ ja $0 \leqslant x + y < 1$. Antud juhul $[x]_{tk} = x$, $[y]_{tk} = y$, $[x + y]_{tk} = x + y$. Kontrollime võrduse (5) kehtivust

$$[x]_{tk} \oplus [y]_{tk} = x \oplus y = \ldots$$

Et $x + y < 10$, siis saame seost (1) kasutades jätkata

$$\ldots = x + y = [x + y]_{tk}.$$

2) Üks liidetav on mittenegatiivne, teine negatiivne, summa on mittenegatiivne. Ilma üldsust kitsendamata võime eeldada, et $0 \leqslant x < 1$, $-1 < y < 0$ ja $0 \leqslant x + y < 1$. Antud juhul $[x]_{tk} = x$, $[y]_{tk} = 10 + y$, $[x + y]_{tk} = x + y$. Kontrollime võrduse (5) kehtivust

$$[x]_{tk} \oplus [y]_{tk} = x \oplus (10 + y) = \ldots$$

Et $x + y \geq 0$, siis $x + (10 + y) \geq 10$ ja kasutades seost (1), saame jätkata

$$\ldots = x + (10 + y) - 10 = x + y = [x + y]_{tk}.$$

3) Üks liidetav on mittenegatiivne, teine negatiivne, summa on negatiivne. Ilma üldsust kitsendamata võime eeldada, et $0 \leq x < 1$, $-1 < y < 0$ ja $-1 < x + y < 0$. Antud juhul $[x]_{tk} = x$, $[y]_{tk} = 10 + y$, $[x + y]_{tk} = x + y + 10$. Kontrollime võrdust (5)

$$[x]_{tk} \oplus [y]_{tk} = x \oplus (10 + y) = \ldots$$

Et $x + y < 0$, siis $x + y + 10 < 10$ ja kasutades seost (1), saame jätkata

$$\ldots = x + y + 10 = [x + y]_{tk}.$$

4) Mõlemad liidetavad on negatiivsed. Seega: $-1 < x < 0$, $-1 < y < 0$ ja $-1 < x + y < 0$. Antud juhul $[x]_{tk} = 10 + x$, $[y]_{tk} = 10 + y$, $[x + y]_{tk} = 10 + x + y$. Kontrollime võrdust (5)

$$[x]_{tk} \oplus [y]_{tk} = (10 + x) \oplus (10 + y) = \ldots$$

Et $x + y > -1$, siis $x + y + 10 + 10 > 11$ ja kasutades seost (1) saame jätkata

$$\ldots = x + 10 + y + 10 - 10 = x + y + 10 = [x + y]_{tk}.$$

Sellega on võrdus (5) kõigil neljal juhul tõestatud.

Teoreemi illustreerimiseks toome mõned näited.

Näide 1.

$x = 0,0001001$
$y = -0,1100001$
$x + y = -0,1011000$

$[x]_{tk} = 0,0001001$
$\oplus\ [y]_{tk} = 1,0011111$
$1,0101000$

$[x + y]_{tk} = 1,0101000.$

Näide 2.

$x = -0,0001001$
$y = -0,1100001$
$x + y = -0,1101010$

$[x]_{tk} = 1,1110111$
$\oplus\ [y]_{tk} = 1,0011111$
$(1)1,0010110$

$[x + y]_{tk} = 1,0010110.$

Näitame ka seda, kuidas koodide "liitmisel" (tehte $\oplus$ mõttes) saab kindlaks teha ületäitumist. Ületäitumine saab muidugi esineda ainult samamärgiliste liidetavate korral.

Näide 3.

$$x = -0{,}0001001 \qquad [x]_{tk} = 1{,}1110111$$
$$y = -0{,}1111101 \qquad \oplus\ [y]_{tk} = 1{,}0000011$$
$$x + y = -1{,}0000110 \qquad (1)0{,}1111010$$

$[x + y]_{tk}$ pole defineeritud.

Koodide "liitmisel" saime positiivse tulemuse, kuigi liidetavad olid negatiivsed. See asjaolu ongi ületäitumise tunnuseks.

Teoreem 2. Kui $|x| < 1$, $|y| < 1$ ja $|x + y| < 1$, siis

$$[x]_{pk} \boxplus [y]_{pk} = [x + y]_{pk}. \qquad (6)$$

Seega: 2.liiki summaator võimaldab absoluutselt ühest väiksemate arvude liitmist, kui arvud on pöördkoodis ja kui liitmisel ei teki ületäitumist.

Tõestus. Tõestusel tuleks vaadelda jälle nelja juhtu, analoogiliselt teoreemi 1 tõestusele. Käesoleval juhul tuleb arvestada aga seda, et nullil on 2 pöördkoodi 0,0000000 ja 1,1111111. Seega nullide korral analüüs mõnevõrra komplitseerub. Et nullide liitmise tulemused on olulised ka käskude süsteemi käsitlemisel, siis on otstarbekohane juhtumeid nullidega vaadelda eraldi.

1) Mõlemad liidetavad on nullid. Vahetult on kontrollitav, et

$$[+0]_{pk} \boxplus [+0]_{pk} = 0{,}0000000 = [+0]_{pk},$$
$$[+0]_{pk} \boxplus [-0]_{pk} = 1{,}1111111 = [-0]_{pk},$$
$$[-0]_{pk} \boxplus [-0]_{pk} = 1{,}1111111 = [-0]_{pk}.$$

2) Mõlemad liidetavad on nullist erinevad, summa on võrdne nulliga. Olgu $x > 0$, $y = -x$. Tähistame $[x]_{pk} = 0{,}x_1x_2\ldots x_n$, $[-x]_{pk} = 1{,}\bar{x}_1\bar{x}_2\ldots\bar{x}_n$ ($x_i$ - kahendnumbrid).

Et võrdus $x_i + \bar{x}_i = 1$ kehtib iga i korral, saame

$$\begin{array}{rl} & [x]_{pk} = 0,x_1x_2\ldots x_n \\ \boxplus & [-x]_{pk} = 1,\bar{x}_1\bar{x}_2\ldots\bar{x}_n \\ \hline & 1,1\ 1\ \ldots 1 \quad = [-0]_{pk}. \end{array}$$

Seega saame nullist erineva arvu ja tema vastandarvu liitmisel teist liiki summaatoris tulemuseks -0.

3) Üks liidetav on nullist erinev, teine liidetav on võrdne nulliga. Olgu $x \neq 0$ ja $y = 0$. Vahetult on kontrollitav, et

$$[x]_{pk} \boxplus [+0]_{pk} = [x]_{pk}\ .$$

Kui $y = -0$, siis tähistame

$$[x]_{pk} = x_o,x_1x_2\ldots x_i\ldots x_n,$$

Olgu $x_i$ esimene 0 erinev kahendnumber (lõpust arvates). Seega on $[x]_{pk} = x_o,x_1x_2\ldots 100\ldots 0$. Vahetult on kontrollitav, et

$$\begin{array}{rl} & [x]_{pk} = x_o,x_1x_2\ldots x_{i-1}100\ldots 0 \\ \boxplus & [-0]_{pk} = 1,\ 1\ 1\ \ldots 1\quad 111\ldots 1 \\ \hline & \underbrace{(1)x_o,x_1x_2\ldots x_{i-1}011\ldots 1} \\ & x_o,x_1x_2\ldots x_{i-1}100\ldots 0 = [x]_{pk}. \end{array}$$

4) Mõlemad liidetavad on positiivsed. Seega $0 < x < 1$, $0 < y < 1$, $0 < x + y < 1$. Antud juhul $[x]_{pk} = x$, $[y]_{pk} = y$, $[x + y]_{pk} = x + y$. Kontrollime võrduse (6) kehtivust

$$[x]_{pk} \boxplus [y]_{pk} = x \boxplus y = \ldots$$

Et $x + y < 1$, siis saame seosest (2)

$$\ldots = x + y = [x + y]_{pk}.$$

5) Üks liidetav on positiivne, teine negatiivne, summa on positiivne. Olgu $0 < x < 1$, $-1 < y < 0$, $0 < x + y < 1$. Sel juhul $[x]_{pk} = x$, $[y]_{pk} = 10 - 10^{-n} + y$, $[x + y]_{pk} = x + y$. Kontrollime võrduse (6) kehtivust.

$$[x]_{pk} \boxplus [y]_{pk} = x \boxplus (10 - 10^{-n} + y) = \ldots$$

Et $x + y > 0$, siis $x + y \geq 10^{-n}$ ja $x + (10 - 10^{-n} + y) \geq 10$ ning seose (2) põhjal saame

$$\ldots = x + (10 - 10^{-n} + y) - 10 + 10^{-n} = x + y = [x + y]_{pk}.$$

6) Üks liidetav on positiivne, teine negatiivne, summa on negatiivne. Olgu $0 < x < 1$, $-1 < y < 0$, $-1 < x + y < 0$. Sel juhul $[x]_{pk} = x$, $[y]_{pk} = 10 - 10^{-n} + y$, $[x + y]_{pk} = 10 - 10^{-n} + (x + y)$. Kontrollime võrduse (6) kehtivust

$$[x]_{pk} \boxplus [y]_{pk} = x \boxplus (10 - 10^{-n} + y) = \ldots$$

Et $x + y < 0$, siis $x + y + 10 - 10^{-n} < 10$ ja seose (2) põhjal saame

$$\ldots = x + (10 - 10^{-n} + y) = 10 - 10^{-n} + (x + y) = [x + y]_{pk}.$$

7) Mõlemad liidetavad on negatiivsed. Seega $-1 < x < 0$, $-1 < y < 0$, $-1 < x + y < 0$. Antud juhul $[x]_{pk} = 10 - 10^{-n} + x$, $[y]_{pk} = 10 - 10^{n} + y$, $[x + y]_{pk} = 10 - 10^{-n} + (x + y)$. Kontrollime võrduse (6) kehtivust

$$[x]_{pk} \boxplus [y]_{pk} = (10 - 10^{-n} + x) \boxplus (10 - 10^{-n} + y) = \ldots$$

Et $x + y > -1$, siis $(10 - 10^{-n} + x) + (10 - 10^{-n} + y) > 10$ ja seose (2) põhjal saame

$$\ldots = (10 - 10^{-n} + x) + (10 - 10^{-n} + y) - 10 + 10^{-n} = 10 - 10^{-n} + (x + y) = [x + y]_{pk}.$$

Esitame veel mõned näited arvude liitmise kohta 2.liiki summaatori abil.

Näide 4.

| | | |
|---|---|---|
| $x$ | $= 0,1011001$ | $[x]_{pk} = 0,1011001$ |
| $y$ | $= -0,0101110$ | $\boxplus [y]_{pk} = 1,1010001$ |
| $x + y$ | $0,0101011$ | $(1)0,0101010$ |
| $[x + y]_{pk}$ | $= 0,0101011$ | $0,0101011$ |

Näide 5. (ületäitumise kohta)

$$\begin{array}{rlrl} x &= 0{,}1011001 & [x]_{pk} &= 0{,}1011001 \\ y &= 0{,}0101110 & [y]_{pk} &= 0{,}0101110 \\ \hline x + y &= 1{,}0000111 & & 1{,}0000111 \end{array}$$

$[x + y]_{pk}$ pole defineeritud.

Koodide liitmisel saime negatiivse tulemuse, kuigi liidetavad olid positiivsed. Sellest asjaolust ilmnebki, et on tegemist ületäitumisega.

Kasutades 1. ja 2. liiki summaatoris vastavalt täiend- ja pöördkoodi, saame ületäitumise kindlaks teha liidetavate ja tulemuse märkide võrdlemisel. Kuid ületäitumist võib põhimõtteliselt kindlaks teha ka ainuüksi lõpptulemuse põhjal. Selleks tuleb täiendkoodi ja pöördkoodi asemel kasutada vastavalt modifitseeritud täiendkoodi ja modifitseeritud pöördkoodi. Defineerime need koodid.

Lihtkahendmurru x modifitseeritud täiendkoodi $[x]_{mt}$ defineerime järgmiselt:

$$[x]_{mt} = \begin{cases} x, & \text{kui } x \geqslant 0, \\ 100 + x, & \text{kui } x < 0. \end{cases}$$

Lihtkahendmurru x modifitseeritud pöördkoodi $[x]_{mp}$ defineerime järgmiselt:

$$[x]_{mp} = \begin{cases} x, & \text{kui } x \geqslant 0, \\ 100 - 10^{-n} + x, & \text{kui } x \leqslant 0. \end{cases}$$

Definitsioonist nähtub, et mittenegatiivsete arvude puhul ühtivad modifitseeritud koodid nende arvude hariliku kujuga 2-ndsüsteemist (ainult enne koma kirjutatakse kaks nulli). Kerge on näha, et negatiivsete arvude korral saab modifitseeritud koode arvutada järgmiselt:

$$[-0,x_1x_2 \dots x_n]_{mt} = 11,\bar{x}_1\bar{x}_2 \dots \bar{x}_n + 10^{-n} \qquad (7)$$

ja

$$[-0,x_1x_2 \dots x_n]_{mp} = 11,\bar{x}_1\bar{x}_2 \dots \bar{x}_n. \qquad (8)$$

Toome mõned näited modifitseeritud koodide kohta:

$$[0,1101101]_{mt} = 00,1101101,$$

$$[-0,1101101]_{mt} = 11,0010011,$$

$$[-0,1101101]_{mp} = 11,0010010.$$

Arvul 0 on üksainus modifitseeritud täiendkood 00,0000000 ja 2 modifitseeritud pöördkoodi $[+0]_{mp} = 00,0000000$, $[-0]_{mp} = 11,1111111$.

Modifitseeritud koode võib vaadelda $100_2$ väiksemate mittenegatiivsete arvudena.

Kui 1. ja 2. liiki summaatoris on koma fikseeritud pärast teist kohta (esimene koht on kaheliste, teine üheliste märkimiseks), siis võime väita analoogiliselt eespool toodud tulemustele (1) ja (2) järgmist.

Neljast väiksemate mittenegatiivsete arvude x ja y korral annab 2. liiki summaator tulemuse:

$$x \oplus y = \begin{cases} x + y, & \text{kui } x + y < 100, \\ x + y - 100, & \text{kui } x + y \geqslant 100. \end{cases} \qquad (9)$$

Neljast väiksemate mittenegatiivsete arvude x ja y korral annab 2. liiki summaator tulemuse:

$$x \boxplus y = \begin{cases} x + y, & \text{kui } x + y < 100, \\ x + y = 100 + 10^{-n}, & \text{kui } x + y \geqslant 100. \end{cases} \qquad (10)$$

Tulemusi (9) ja (10) saab põhjendada analoogiliselt tulemuse (1) ja (2) põhjendustega.

Eeldades, et $|x| < 1$, $|y| < 1$ ja $|x + y| < 1$ saab tõestada järgmised võrdused

$$[x]_{mt} \oplus [y]_{mt} = [x + y]_{mt} \qquad (11)$$

ja

$$[x]_{mp} \boxplus [y]_{mp} = [x + y]_{mp}. \qquad (12)$$

Seega: 1.liiki summaator võimaldab teostada arvude liitmist modifitseeritud täiendkoodis, 2. liiki summaator aga modifitseeritud pöördkoodis.

Et võrduste (11) ja (12) tõestus on analoogiline vasta-

valt teoreemide 1 ja 2 tõestustega, siis jätame need esitamata. Märgime ainult seda, et tõestustes tuleb kasutada vastavalt seoseid (9) ja (10).

Esitame ainult mõned näited liitmise kohta modifitseeritud koodides. Eriti huvipakkuvad on muidugi ületäitumise juhud.

Näide 6.

$$\begin{aligned} x &= 0{,}1100001 \\ y &= -0{,}0011101 \\ \hline x+y &= 0{,}1000100 \end{aligned} \qquad \begin{aligned} [x]_{mt} &= 00{,}1100001 \\ [y]_{mt} &= 11{,}1100011 \\ \hline &(1)00{,}1000100 \end{aligned}$$

$$[x+y]_{mt} = 00{,}1000100$$

Näide 7.

$$\begin{aligned} x &= 0{,}0101110 \\ y &= -0{,}0010101 \\ \hline x+y &= -0{,}1000011 \end{aligned} \qquad \begin{aligned} [x]_{mp} &= 11{,}1010001 \\ [y]_{mp} &= 11{,}1101010 \\ \hline &(1)11{,}0111011 \end{aligned}$$

$$[x+y]_{mp} = 11{,}0111100 \qquad 11{,}0111100$$

Näide 8. (positiivne ületäitumine)

$$\begin{aligned} x &= 0{,}1110111 \\ y &= 0{,}1100011 \\ \hline x+y &= 1{,}1011010 \end{aligned} \qquad \begin{aligned} [x]_{mt} &= 00{,}1110111 \\ [y]_{mt} &= 00{,}1100011 \\ \hline & 01{,}1011010 \end{aligned}$$

$[x+y]_{mt}$ puudub.

Ületäitumise kriteeriumiks on erinevad numbrid kahel esimesel kohal enne koma. Numbrite "01" korral kahel esimesel kohal on tegemist ületäitumisega, mis tekkis positiivsete arvude liitmisel (nn. positiivse ületäitumisega), numbrite "10" korral kahel esimesel kohal on tegemist ületäitumisega, mis tekkis negatiivsete arvude liitmisel (nn. negatiivse ületäitumisega).

Näide 9. (negatiivne ületäitumine)

| | | | |
|---|---|---|---|
| $x =$ | $-0,1110111$ | $[x]_{mp}$ | $= 11,0001000$ |
| $y =$ | $-0,1100011$ | $\boxplus\ [y]_{mp}$ | $= 11,0011100$ |
| $x + y =$ | $-1,1011010$ | | $(1)10,0100100$ |
| $[x + y]_{mp}$ | puudub. | | $10,0100101.$ |

Täpselt samuti nagu modifitseeritud täiendkoodi korral, on ka modifitseeritud pöördkoodi korral ületäitumise kriteeriumiks erinevad numbrid kahel esimesel kohal. Seejuures on numbrite "01" korral kahel esimesel kohal tegemist positiivse ületäitumisega, numbrite "10" korral aga negatiivse ületäitumisega.

Nagu nendest näidetest selgub, on ületäitumise kindlakstegemine modifitseeritud koodide puhul palju mugavam kui hariliku täiend- ja pöördkoodi korral. Seetõttu kasutatakse arvutusmasinais enamasti modifitseeritud koode. Arvutusmasinas "Ural-1" kasutatakse näiteks modifitseeritud pöördkoodi.

Lõpuks veel märkus arvude lahutamise kohta. Arvude x ja y lahutamine taandatakse liitmisele

$$x - y = x + (-y)$$

arvu y märgi muutmise teel. Seega ei paku see tehe liitmisega võrreldes midagi uut.

## § 5. Korrutamine ja jagamine fikseeritud komaga arvutusmasinates. Aritmeetiliste tehete teostamisest liikuva komaga arvutusmasinates.

Korrutamine toimub fikseeritud komaga arvutusmasinates (nagu näiteks arvutusmasinas "Ural-1") põhimõtteliselt järgmisel viisil.

Kõigepealt tehakse kindlaks spetsiaalses ühekohalises summaatoris korrutise märk (märk "+" kodeeritakse 0, märk "-" kodeeritakse 1 abil):

$$0 + 0 = 0,$$
$$0 + 1 = 1,$$
$$1 + 0 = 1,$$
$$1 + 1 = 0.$$

Seejärel viiakse korrutatav arv x aritmeetilise seadme registrisse, kusjuures talle omistatakse korrutise märk. Korrutaja y viiakse otsekoodis korrutaja registrisse. Aritmeetilise seadme registris toimub korrutatava järkjärguline nihutamine ühe koha võrra paremale (s.t. osakorrutiste moodustamine). Kui korrutaja y otsekoodi vastaval kahendkohal asub number 1, siis liidetakse summaatoris olevale osakorrutiste summale (modifitseeritud pöördkoodis) antud sammul saadud osakorrutis. Vastupidisel juhul liidetakse summaatoris olevale osakorrutiste summale 0. Suurema täpsuse tagamiseks on aritmeetilise seadme registris ja summaatoris lisakohad (arvutusmasinal "Ural-1" on näiteks 6 lisakohta). Märgime, et korrutamisel ületäitumist ei saa esineda.

Selgitame öeldut mõne näite varal.

<u>Näide 1</u>. Olgu x = 0,1011001, y = 0,1001101. Korrutise märk (0 + 0 = 0) on "+". Osakorrutiste liitmine toimub järgmiselt:

$$\begin{array}{l} 0{,}1\,0\,0\,1\,1\,0\,1 \\ \quad\uparrow \\ 0{,}1\,0\,0\,1\,1\,0\,1 \\ \quad\;\uparrow \end{array}$$

$$\begin{array}{rr|l} & & \text{lisakohad} \\ \boxplus & 0\,0{,}0\,1\,0\,1\,1\,0\,0 & 1\,0\,0\,0\,0\,0 \\ & 1\,1{,}1\,1\,1\,1\,1\,1\,1 & 1\,1\,1\,1\,1\,1 \\ \hline (1) & 0\,0{,}0\,1\,0\,1\,1\,0\,0 & 0\,1\,1\,1\,1\,1 \\ & 0\,0{,}0\,1\,0\,1\,1\,0\,0 & 1\,0\,0\,0\,0\,0 \end{array}$$

(edaspidi jätame nulliga võrduva osakorrutise liitmata).

$$0{,}1\,0\,0\,1\,1\,0\,1 \quad (\uparrow)$$

$$\begin{array}{rr|l} & & \text{lisakohad} \\ \boxplus & 0\,0{,}0\,1\,0\,1\,1\,0\,0 & 1\,0\,0\,0\,0\,0 \\ & 0\,0{,}0\,0\,0\,0\,1\,0\,1 & 1\,0\,0\,1\,0\,0 \\ \hline & 0\,0{,}0\,1\,1\,0\,0\,1\,0 & 0\,0\,0\,1\,0\,0 \end{array}$$

0,1 0 0 1 1 0 1

$$\begin{array}{r|l} \boxplus\ 0\,0{,}0\,1\,1\,0\,0\,1\,0 & 0\,0\,0\,1\,0\,0 \\ 0\,0{,}0\,0\,0\,0\,0\,1\,0 & 1\,1\,0\,0\,1\,0 \\ \hline 0\,0{,}0\,1\,1\,0\,1\,0\,0 & 1\,1\,0\,1\,1\,0 \end{array}$$

0,1 0 0 1 1 0 1

$$\begin{array}{r|l} \boxplus\ 0\,0{,}0\,1\,1\,0\,1\,0\,0 & 1\,1\,0\,1\,1\,0 \\ 0\,0{,}0\,0\,0\,0\,0\,0\,0 & 1\,0\,1\,1\,0\,0 \\ \hline 0\,0{,}0\,1\,1\,0\,1\,0\,1 & 1\,0\,0\,0\,1\,0 \end{array}$$

Pärast ümardamist saame tulemuse

$$[x \cdot y]_{mp} = 00{,}0110110,$$

millest saame, et $x \cdot y = 0{,}0110110$.

Märgime, et positiivse arvu ümardamine tähendab ümardatavale arvule koodi

0 0,0 0 0 0 0 0 0 1 0 0 0 0 0

liitmist (tehte $\boxplus$ mõttes) ja lisakohtade ärajätmist.

<u>Näide 2.</u> Olgu $x = 0{,}1011011$, $y = -0{,}1101001$. Korrutise märk $(0 + 1 = 1)$ on "-". $[-x]_{mp} = 11{,}0100100\,\underbrace{111111}_{\text{lisakohad}}$. Osakorrutiste liitmisel on 0 liitmine ära jäetud.

0,1101001

0,1101001

$$\begin{array}{r|l} \boxplus\ 11{,}1010010 & 011111 \\ 11{,}1101001 & 001111 \\ \hline (1)\,11{,}0111011 & 101110 \end{array}$$

0,1101001

$$\begin{array}{r|l} \boxplus\ 11{,}0111011 & 101111 \\ 11{,}1111010 & 010011 \\ \hline (1)\ 11{,}0110110 & 000010 \end{array}$$

0,1101001

$$\begin{array}{r|l} \boxplus\ 11{,}0110110 & 000011 \\ 11{,}1111111 & 010010 \\ \hline (1)\ 11{,}0110101 & 010101 \\ \hline 11{,}0110101 & 010110 \end{array}$$

Pärast ümardamist saame tulemuse

$$[x \cdot y]_{mp} = 11{,}0110100,$$

millest saame, et $[x \cdot y] = -0{,}1001011$.

Märgime, et negatiivse arvu ümardamine tähendab ümardatavale arvule koodi

11,1111111 011111

liitmist (tehte ⊞ mõttes) ja lisakohtade ärajätmist.

Selgitame, millise põhimõtte järgi leitakse arvutusmasinas jagatis.

Kõigepealt määratakse ühekohalises summaatoris jagatise märk nii, nagu see toimus korrutise puhul. Seejärel selgitatakse, kas jagamisel tekib ületäitumine või mitte. Oletades, et jagatav x ja jagaja y on näiteks positiivsed, saame ületäitumise esinemise kindlaks teha vahe x - y märgi abil (kui x - y > 0, tekib ületäitumine, vastupidisel juhul ületäitumist ei esine). Kui ületäitumist ei esine, tuleb asuda jagatise kahendkohtade leidmisele. Selleks tuleb kõigepealt taastada x (jagaja y liitmise teel vahele x - y), nihutada see ühe koha võrra vasakule (millega saame 2x) ja lahutada saadud tulemusest y. Kui saadud vahe 2 x - y on positiivne, saame jagatise vastavaks kahendkohaks "1", kui negatiivne, siis saame jagatise vastavaks kahendkohaks "0". Negatiivse vahe korral tuleb eelmise sammu tulemus taastada y liitmise teel.

Pärast jagatise mingi kahendkoha leidmist (ja mõnel juhul taastamist) tuleb saadud jääk uuesti nihutada ühe koha võrra vasakule. Tulemusest tuleb jälle lahutada y, kindlaks teha vahe märk jne.

Märgime, et tegelikult pole negatiivse vahe korral eelmise sammu tulemust vajagi taastada. Oletame, et mingil sammul saime negatiivse vahe r - y. Pärast taastamist saame (r - y) + y = r. Nihutades seda arvu ühe koha võrra vasakule, saame 2 r, millest y lahutamise järel saame 2 r - y. Sama tulemuse saame aga selle negatiivse jäägi r - y nihutamisel ühe koha võrra vasakule ja jagaja y liitmisel. Tõepoolest, 2(r - y) + y = 2 r - y.

Juhtudel x > 0, y < 0; x < 0, y > 0 ja x < 0, y < 0 tuleb ülal-

kirjeldatud algoritmis ette võtta mõningad muudatused (näiteks y lahutamise asemel tuleb y liita jms.), mida soovitame sõnastada lugejatel enestel.

Toome ülalõeldu selgituseks mõned näited.

Näide 3. Leiame jagatise x : y, kui x = 0,1001010 ja y = 0,1011010. Arvutused teostame modifitseeritud pöördkoodis $[-y]_{mp}$ = 11,0100101. Jagatis on positiivne (0 + 0 = 0).

| | |
|---|---|
| ⊞ 00,1001010 | ületäitumise kontroll |
| 11,0100101 | x - y |
| 11,1101111 | ületäitumist ei esine |
| ⊞ 11,1011111 | nihutus vasakule (nihutus vasakule toimub tsüklilise ülekandega kõrgeimast madalaimasse järku) |
| 00,1011010 | y liitmine |
| (1)00,0111001 | |
| 00,0111010 | tulemus positiivne, "1" |
| ⊞ 00,1110100 | nihutus vasakule |
| 11,0100101 | -y liitmine |
| 00,0011010 | tulemus positiivne, "1" |
| ⊞ 00,0110100 | nihutus vasakule |
| 11,0100101 | -y liitmine |
| 11,1011001 | tulemus negatiivne, "0" |
| ⊞ 11,0110011 | nihutus vasakule |
| 00,1011010 | y liitmine |
| 00,0001110 | tulemus positiivne, "1" |
| ⊞ 00,0011100 | nihutus vasakule |
| 11,0100101 | -y liitmine |
| 11,1000001 | tulemus negatiivne, "0" |
| ⊞ 11,0000011 | nihutus vasakule |
| 00,1011010 | y liitmine |
| 11,1011101 | tulemus negatiivne, "0" |

| | |
|---|---|
| 11,0111011 | nihutus vasakule |
| 00,1011010 | y liitmine |
| 00,0010110 | tulemus positiivne, "1" |
| 00,0101100 | nihutus vasakule |
| 11,0100101 | -y liitmine |
| 11,1010001 | tulemus negatiivne, "0". |

Seega on jagatis (peale ümardamist) x : y = 0,1101001.

Näide 4. Leiame jagatise x : y, kui x = -0,1010101 ja y = -0,1101101. Arvutused teostame modifitseeritud pöördkoodis. $[x]_{mp}$ = 11,0101010, $[y]_{mp}$ = 11,0010010, $[-y]_{mp}$ = = 00,1101101. Jagatis on positiivne (1 + 1 = 0).

| | |
|---|---|
| 11,0101010 | ületäitumise kontroll |
| 00,1101101 | x - y |
| 00,0011000 | ületäitumist ei esine |
| 00,0110000 | nihutus vasakule |
| 11,0010010 | y liitmine |
| 11,1000010 | tulemus negatiivne, "1" |
| 11,0000101 | nihutus vasakule |
| 00,1101101 | -y liitmine |
| 11,1110010 | tulemus negatiivne, "1" |
| 11,1100101 | nihutus vasakule |
| 00,1101101 | -y liitmine |
| 00,1010011 | tulemus positiivne. "0" |
| 01,0100110 | nihutus vasakule |
| 11,0010010 | y liitmine |
| 00,0111001 | tulemus positiivne, "0" |
| 00,1110010 | nihutus vasakule |
| 11,0010010 | y liitmine |
| 00,0000101 | tulemus positiivne, "0" |

| | |
|---|---|
| ⊞ 00,0001010 | nihutus vasakule |
| 11,0010010 | y liitmine |
| 11,0011100 | tulemus negatiivne, "1". |
| ⊞ 10,0111001 | nihutus vasakule |
| 00,1101101 | -y liitmine |
| 11,0100110 | tulemus negatiivne, "1" |
| ⊞ 10,1001101 | nihutus vasakule |
| 00,1101101 | -y liitmine |
| 11,0111010 | tulemus negatiivne, "1" |

Seega on jagatis (peale ümardamist) x : y = 0,1100100.

Aritmeetiliste tehete sooritamine liikuva komaga arvutusmasinates toimub normaliseeritud arvudega

$$x_1 = m_1 \cdot 2^{p_1}, \quad \frac{1}{2} \leqslant m_1 < 1,$$

$$x_2 = m_2 \cdot 2^{p_2}, \quad \frac{1}{2} \leqslant m_2 < 1$$

järgmisel viisil.

Liitmise korral tuleb enne mantisside liitmist arvude järgud võrdustada. Seda saab teha ainult väiksema järgu suurendamise ja vastava mantissi paremale nihutamise teel.

$$x = x_1 + x_2 = m_1 \cdot 2^{p_1} + m_2 \cdot 2^{p_2} =$$

$$= \begin{cases} (m_1 + 2^{p_2-p_1} m_2) \cdot 2^{p_1}, & \text{kui } p_1 \geqslant p_2, \\ (2^{p_1-p_2} m_1 + m_2) \cdot 2^{p_2}, & \text{kui } p_1 < p_2. \end{cases} \qquad (1)$$

Ei tule arvata, et summa järk (esimesel juhul) on $p_1$ ja mantiss on $m_1 + 2^{p_2-p_1} m_2$. Arv $m_1 + 2^{p_2-p_1} m_2$ ei tarvitse olla normaliseeritud. Arvu $m_1 + 2^{p_2-p_1} m_2$ puhul võib esineda ületäitumine, kuid see ei saa ületada arvu 2. Samuti võib arvu $m_1 + 2^{p_2-p_1} m_2$ puhul esineda alatäitumine. Seega tulevad valemid (1) korrigeeritud kujul järgmised:

$$x = x_1 + x_2 = \begin{cases} 2^{\Delta}\,(m_1 + 2^{p_2-p_1}m_2)\cdot 2^{p_1-\Delta}, & \text{kui } p_1 \geqslant p_2 \\ 2^{\Delta}\,(2^{p_1-p_2}m_1 + m_2)\cdot 2^{p_2-\Delta}, & \text{kui } p_1 < p_2 \end{cases}$$

kus $\Delta = -1$ ületäitumise korral, $\Delta = 1, 2, 3, \ldots$ ( $\Delta$ on nullide arv koma ja esimese 1 vahel) alatäitumise korral. Kui $m_1 + 2^{p_2-p_1}m_2$ (resp. $2^{p_1-p_2}m_1 + m_2$) liitmisel ei teki ei üle- ega alatäitumist, siis võetakse $\Delta = 0$. Kui üks liidetav on võrdne nulliga, siis võetakse summaks teine liidetav. Lahutamine teostatakse kui vastandarvu liitmine.

Näide 5. Liita arvud $x_1 = 0{,}1000010\cdot 2^{101}$ ja $x_2 = -0{,}1010011\cdot 2^{10}$.

Suurendame teise arvu järku 3 võrra, samal ajal nihutades selle arvu mantissi 3 koha võrra paremale. Saame $x_2 = -0{,}0001010\cdot 2^{101}$. Liidame mantissid

$$\begin{array}{r} 0{,}1000010 \\ -0{,}0001010 \\ \hline 0{,}0111000 \end{array}$$

Saadud arv pole normaliseeritud (pole seega summa tõeliseks mantissiks). Normaliseerimiseks tuleb teda nihutada ühe koha võrra vasakule (s.t. $\Delta = 1$). Samaaegselt tuleb järku vähendada 1 võrra. Seega arvude $x_1$ ja $x_2$ summa tuleb normaalkujus

$$x = 0{,}1110000\cdot 2^{100}$$

Siin toodud näites liitsime mantissid otsekoodis. Tegelikult toimub mantisside liitmine - nagu fikseeritud komaga arvude liitmine - üldiselt mingisuguses modifitseeritud koodis.

Korrutamise korral saame:

$$x = x_1 \cdot x_2 = m_1 \cdot 2^{p_1}\, m_2 \cdot 2^{p_2} = m_1 \cdot m_2 \cdot 2^{p_1+p_2} \qquad (3)$$

Et mantisside korrutamisel võib esineda alatäitumine (mantisside korrutis ei saa tulla aga väiksem kui 0,01), siis ei tarvitse $m_1 \cdot m_2$ olla korrutise tõeliseks mantissiks. Seetõttu tuleks võrdust (3) täiendada nii:

$$x = (m_1 \cdot m_2 \cdot 2^{\Delta})\, 2^{p_1+p_2-\Delta}, \qquad (4)$$

kus $\Delta = 0$ või 1 (alatäitumise korral).

Jagamise korral saame:

$$x = x_1 : x_2 = (m_1 \cdot 2^{p_1}) : (m_2 \cdot 2^{p_2}) = \frac{m_1}{m_2} \cdot 2^{p_1-p_2}. \qquad (5)$$

Et mantisside jagamisel võib tekkida ületäitumine, mis ei saa ületada arvu 2, siis tuleb valemit (5) täiendada nii

$$x = x_1 : x_2 = \left(\frac{m_1}{m_2} : 2^{\Delta}\right) : 2^{p_1-p_2-\Delta}, \qquad (6)$$

kus $\Delta = 0$ või $-1$ (ületäitumise korral).

Et mantisside korrutamine ja jagamine toimuvad nagu fikseeritud komaga arvude korrutamine ja vastavalt jagamine, siis me arvulisi näiteid nende tehete kohta ei esita.

## II p e a t ü k k.

## ARVUTUSMASINATE KONSTRUKTSIOONI LOOGILISED ALUSED.

### §6. Lausearvutuse põhimõisteid.

Lausearvutus on üks matemaatilise loogika arvutussüsteeme, mida laialdaselt rakendatakse elektronsüsteemide analüüsis ja sünteesis, samuti ka programmide loogiliste skeemide teisendamisel.

Lausearvutuse põhimõisteteks on lause, tõeväärtuse ja loogilise tehte mõiste. Lausearvutuse lauseks võib olla tavalise keele iga väitev lause, nagu näiteks:

"Tartu on ülikoolilinn.", (1)

" 2+2=4", (2)

"2+2=5". (3)

Lausearvutuse lauseteks pole aga järgmised keelelised laused või väljendid:

"Tere hommikust!",

"Kas täna sajab?",

" 2+2 ",

sest need ei väida midagi (nende väljendite puhul pole võimalik kõnelda nende sisu vastavusest või mittevastavusest tegelikkusele).

Lauseid tähistame tähtedega A, B, C, X, Y, Z. Kui lause sisu vastab tegelikkusele, siis ütleme, et lause on tõene; vastupidisel korral ütleme, et lause on väär. Nii on näiteks laused (1) ja (2) tõesed, lause (3) aga väär. Lausearvutuse lause võib olla kas tõene või väär, kolmas võimalus on välistatud. Tõeväärtusi "tõene" ja "väär" võib tähistada mitmel viisil, näiteks t ja v abil. Tehnilistes rakendustes tähis-

tatakse neid sageli aga 1 ja 0 abil. Ka siin kasutame viimast tähistusviisi. Asjaolu, et lause A on tõene (resp. väär), tähistame A=1 (resp. A=0) abil.

Enne loogiliste tehete defineerimist lepime kokku järgmises: 1) loogilisi tehteid võib sooritada lausetega, millel on mistahes sisu; 2) loogiliste tehete tõeväärtus oleneb ainult komponentide (komponentlausete) tõeväärtustest, mitte aga nende sisust.

Kokkulepped 1) ja 2) võimaldavad loogiliste tehete defineerimisel lausete sisu üldse kõrvale jätta - oluline on ainult lausete tõeväärtus. Et lausearvutuse teoreetiline käsitlus tugineb aga tõeväärtustel ja loogilistel tehetel, siis võib üldse lausete sisu vaatlemisest loobuda, arvestades ainult tõeväärtusi. Seega võib lauseid vaadelda kui kahendmuutujaid, s.t. muutujaid, mis võivad omandada väärtusi kaheelemendilisest hulgast $\{1; 0\}$. Lausearvutus ise taandub aga kahendmuutuja funktsioonide teooriaks, kus peale argumentide $X_1, X_2, \ldots, X_n$ ka funktsioonid $f(X_1, X_2, \ldots, X_n)$ võivad omandada ainult kaht väärtust 1 ja 0.

Loogilistest tehetest vaatleme konjunktsiooni (tehtemärk &), disjunktsiooni ($\vee$), implikatsiooni ($\rightarrow$), ekvivalentsi ($\sim$), mitteekvivalentsi ($\not\sim$), alternatiivset eitust[1] ($/$) ja eitust (tehtemärk $\neg$ või $^-$). Kõik tehted peale viimase on binaarsed, s.t. rakendatud kahele lausele. Eitus on unaarne tehe, s.t. rakendatav ühele lausele.

Tehete defineerimisel tuleb seega defineerida tehete tõeväärtus komponentlausete kõikvõimalike tõeväärtuste korral.

Defineerime loogilised tehted järgmiste tabelite abil:

| X | Y | X & Y | X ∨ Y | X → Y | X ∼ Y | X ≁ Y | X / Y |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

[1] Nimetatakse ka Shefferi tehteks ehk Shefferi kriipsuks (tehtemärgi kuju tõttu).

| X | ¬X |
|---|---|
| 0 | 1 |
| 1 | 0 |

Tabelite põhjal on kerge anda ka loogiliste tehete keelelised vasted. Nii on konjunktsiooni keeleliseks vasteks sidesõna "ja", disjunktsioonil mittevälistav "või", implikatsiooni ligikaudseks keeleliseks vasteks on väljend "kui ..., siis ...", ekvivalentsil väljend " ... on samatõene ...", mitteekvivalentsil väljend " ... pole samatõene ..." ehk välistav "või", alternatiivsel eitusel "kas ... või ... ei kehti", eitusel " ... ei kehti". Nii tuleb lausete (1) ja (2) konjunktsioon sõnastada "Tartu on ülikoolilinn ja 2+2=4". Et mõlemad komponentlaused on tõesed, siis on tõene ka nende konjunktsioon. Lausete (2) ja (3) ekvivalents kõlab ""2+2=4" on samatõene väitega "2+2=5"" ja on väär, sest komponentlausetel on erinev tõeväärtus.

Loogiliste tehete korduval rakendamisel saame avaldised, mida nimetatakse loogilisteks valemiteks. Loogilise valemi mõiste saab induktiivselt defineerida järgmisel viisil:

1) sümbolid 1, 0, A, B, C, X, Y, Z, $A_1$, $B_1$, $C_1$, $X_1$, $Y_1$, $Z_1$,... on loogilised valemid,

2) kui $\mathcal{A}$ ja $\mathcal{B}$ on loogilised valemid, siis on loogilised valemid ka $(\mathcal{A} \& \mathcal{B})$, $(\mathcal{A} \vee \mathcal{B})$, $(\mathcal{A} \to \mathcal{B})$, $(\mathcal{A} \sim \mathcal{B})$, $(\mathcal{A} \approx \mathcal{B})$ ja $(\mathcal{A} / \mathcal{B})$;

3) kui $\mathcal{A}$ on loogiline valem, siis on ka $\neg\mathcal{A}$ loogiline valem.

Nii on näiteks loogilisteks valemiteks avaldised

$$((X \vee Y) \to (Z \sim \neg Y)) \qquad (4)$$

ja

$$((A \approx 1) \sim (\neg 0 \,/\, B)),$$

aga mitte $X \to 1 \to 0$.

Sulgude arvu vähendamiseks lepime kokku, et: 1) valemi kõige välimise sulgude paari võib jätta kirjutamata, 2) tehetest tuleb sooritada kõigepealt eitus, seejärel konjunktsioon ja disjunktsioon (eeldusel, et sulud ei nõua teistsugust

tehete järjekorda), 3) lausete $X_1, X_2, \ldots, X_n$ (laused võivad olla ka eitustega) konjunktsiooni ja disjunktsiooni võib kirjutada kujul $X_1 \& X_2 \& \ldots \& X_n$ ehk $X_1 X_2 \ldots X_n$ ja vastavalt $X_1 \vee X_2 \vee \ldots \vee X_n$. Viimane kokkulepe tugineb konjunktsiooni ja disjunktsiooni assotsiatiivsusel (vt. samasusi (9) ja (10)).

On kerge näha, et iga loogiline valem defineerib teatava kahendmuutuja funktsiooni. Nii näiteks defineerib valem (4) kahendmuutuja funktsiooni f (X, Y, Z) (vt. tabelit 1)

Tabel 1.

| X | Y | Z | X (1) | $\vee$ (2) | Y (1) | $\rightarrow$ (5) | (Z (1) | $\sim$ (4) | $\neg$ (3) | Y) (1) | f(X,Y,Z) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

Argumentide X,Y,Z fikseeritud väärtuste korral leiame °unktsiooni f(X,Y,Z) väärtuse valemist (4), kui teostame selles valemis ettenähtud loogilised tehted (tehete järjekord on tabelis märgitud numbritega (1),(2),...). Valemi (4), seega ka funktsiooni f(X,Y,Z) väärtus tuleb viimase tehtemärgi $\rightarrow$ alla.

Loogiliste valemitega opereerimisel on vajalik valemite (loogilise) samaväärsuse mõiste. Loogiliste valemite $\mathcal{A}$ ja $\mathcal{B}$ samaväärsust mõistame kui valemitele $\mathcal{A}$ ja $\mathcal{B}$ vastavate kahendmuutuja funktsioonide väärtuste võrdust ühe ja sama argumentide (komponentlausete) väärtuste kombinatsiooni korral. Valemite $\mathcal{A}$ ja $\mathcal{B}$ samaväärsust tähistame

$$\mathcal{A} = \mathcal{B}.$$

Nii on näiteks valemid $X \& \neg Y$ ja $\neg(Y \vee \neg X)$ samaväärsed, s.t.

$$X \& \neg Y = \neg(Y \vee \neg X), \qquad (5)$$

mida võib kontrollida nn. tabelimeetodi abil

Tabel 2.

| X | Y | X | & | ¬ | Y | ¬(Y ∨ ¬X) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

Tabelist nähtub, et komponentlausete X ja Y samade väärtuste korral on ka võrduse (5) vasaku ja parema poole väärtused vastavalt võrdsed.

Märgime, et samaväärsetel valemitel ei tarvitse tingimata olla ühed ja samad komponentlaused. Nii näiteks kehtib võrdus

$$X = X \,\&\, (Y \vee \neg Y).$$

Kergesti näeme, et muutujal Y võrduse paremal poolel üldse mingisugust funktsionaalset tähendust ei ole. Selletõttu nimetatakse teda nn. fiktiivseks muutujaks.[2]

Loogiliste valemite lihtsustamiseks, samuti nende teisendamiseks spetsiaalsetele erikujudele on vaja teada tähtsamaid loogilisi samaväärsusi. Need on järgmised:

$$\neg\neg X = X \tag{6}$$

(kahekordse eituse võib ära jätta);

$$X \,\&\, Y = Y \,\&\, X \tag{7}$$

$$X \vee Y = Y \vee X \tag{8}$$

(konjunktsiooni ja disjunktsiooni kommutatiivsus);

$$(X \,\&\, Y) \,\&\, Z = X \,\&\, (Y \,\&\, Z) \tag{9}$$

$$(X \vee Y) \vee Z = X \vee (Y \vee Z) \tag{10}$$

(konjunktsiooni ja disjunktsiooni assotsiatiivsus);

$$X \,\&\, (Y \vee Z) = (X \,\&\, Y) \vee (X \,\&\, Z) \tag{11}$$

$$X \vee (Y \,\&\, Z) = (X \vee Y) \,\&\, (X \vee Z) \tag{12}$$

---

[2] Fiktiivne muutuja defineeritakse järgmiselt: loogilise valemi või kahendmuutuja funktsiooni $f(X_1,\ldots,X_{i-1},X_i,X_{i+1},\ldots,X_n)$ muutujat $X_i$ nimetatakse fiktiivseks, kui võrdus $f(X_1,\ldots,X_{i-1},0,X_{i+1},\ldots,X_n) = f(X_1,\ldots,X_{i-1},1,X_{i+1},\ldots,X_n)$ kehtib muutujate $X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n$ kõikide väärtuste korral.

(konjunktsiooni distributiivsus disjunktsiooni suhtes ja disjunktsiooni distributiivsus konjunktsiooni suhtes);

$$\neg(X \,\&\, Y) = \overline{X} \vee \overline{Y} \qquad (13)$$

$$\neg(X \vee Y) = \overline{X} \,\&\, \overline{Y} \qquad (14)$$

(konjunktsiooni ja disjunktsiooni duaalsus[3]);

$$X \to Y = \overline{X} \vee Y \qquad (15)$$

$$X \sim Y = (X \,\&\, Y) \vee (\overline{X} \,\&\, \overline{Y}) = (X \vee \overline{Y}) \,\&\, (\overline{X} \vee Y) \qquad (16)$$

$$X \not\sim Y = \neg(X \sim Y) = (X \,\&\, \overline{Y}) \vee (\overline{X} \,\&\, Y) = (X \vee Y) \,\&\, (\overline{X} \vee \overline{Y}) \qquad (17)$$

$$X \,/\, Y = \neg(X \,\&\, Y) = \overline{X} \vee \overline{Y} \qquad (18)$$

$X \,\&\, X = X$ (19) $\qquad X \vee X = X$ (23)

$X \,\&\, \overline{X} = 0$ (20) $\qquad X \vee \overline{X} = 1$ (24)

$X \,\&\, 1 = X$ (21) $\qquad X \vee 1 = 1$ (25)

$X \,\&\, 0 = 0$ (22) $\qquad X \vee 0 = X$ (26)

$$X \sim 1 = X \qquad (27)$$

$$X \sim 0 = \overline{X} \qquad (28)$$

$$0 \to X = 1 \qquad (29)$$

$$X \to 0 = X \qquad (30)$$

Võrduste (6) - (30) kehtivust võib tõestada tabelimeetodi abil. Paljud nendest, nagu (6), (7) ja mõned teised, järelduvad vahetult loogiliste tehete definitsioonist. Muidugi võib selliste võrduste loetelu veelgi jätkata.

Näitame nüüd, kuidas võrduste (6) - (30) abil saab loogilisi valemeid lihtsustada.

Näide 1. Lihtsustada valem

$$(\,(X \,/\, \overline{Y}) \to 0) \sim (1 \to X) \qquad (31)$$

---

[3] Konjunktsiooni ja disjunktsiooni duaalsus on veelgi üldisema iseloomuga. Võib tõestada, et kui mingi valem $\mathcal{A}(X_1, X_2, \ldots, X_n; \,\&, \vee, \neg)$ on moodustatud ainult konjunktsiooni, disjunktsiooni ja eituse tehte abil, siis selle eitus $\neg\mathcal{A}(X_1, X_2, \ldots X_n; \,\&, \vee, \neg)$ on loogiliselt samaväärne valemiga $\mathcal{A}(\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_n; \vee, \&, \neg)$, s.t. muutujad tuleb asendada nende eitustega ning konjunktsiooni- ja disjunktsioonimärgid vahetada.



TARTU ÜLIKOOLI
RAAMATUKOGU

Võrduste (15) ja (26) põhjal saame kirjutada $1 \to X = \overline{1} \vee X = 0 \vee X = X$. Samade võrduste abil saame valemi $(X/\overline{Y}) \to 0$ teisendada kujule $\neg(X/\overline{Y})$. Seega saame valemist (31) temaga samaväärse valemi

$$\neg(X/\overline{Y}) \sim X \qquad (32)$$

Võrduse (18) põhjal kehtib $X / \overline{Y} = \neg(X \,\&\, Y)$, millega valemist (32) saame samaväärse valemi

$$X \,\&\, \overline{Y} \sim X. \qquad (33)$$

Kasutades võrdust (16), distributiivsuse ja duaalsuse seoseid, saame valemist (33)

$$((X \,\&\, \overline{Y}) \vee \overline{X}) \,\&\, (\neg(X \,\&\, Y) \vee X) = ((X \vee \overline{X}) \,\&\, (\overline{Y} \vee \overline{X})) \,\&\, ((\overline{\overline{X} \vee Y}) \vee X)$$

Arvestades võrdusi (25), (21), (6), disjunktsiooni assotsiatiivsust ja kommutatiivsust, saame viimase avaldise teisendada kujule

$$\overline{X} \vee \overline{Y}.$$

Näide 2. Lihtsustada valem

$$(\overline{X} \,\&\, Z \to (X \sim 0)) \,\&\, \overline{Y} \to (X \sim Y).$$

Asendame $X \sim 0$ võrduse (28) alusel ja kasutame esimese implikatsiooni puhul võrdust (15) ning seejärel võrdust (13). Saame

$$(\overline{\overline{X} \vee Z \vee \overline{X}}) \,\&\, \overline{Y} \to (X \sim Y).$$

Osavalemit $\overline{\overline{X} \vee Z \vee \overline{X}}$ saame võrduste (6), (10) ja (25) põhjal teisendada nii: $X \vee \overline{Z} \vee \overline{X} = (X \vee \overline{X}) \vee \overline{Z} = 1 \vee Z = 1$. Liikme 1 võib aga võrduse (21) põhjal konjunktsioonis üldse kirjutamata jätta. Seega saame

$$\overline{Y} \to (X \sim Y).$$

Kasutades võrdusi (15) ja (17), saame

$$Y \vee (X \,\&\, Y) \vee (\overline{X} \,\&\, \overline{Y})$$

ehk konjunktsiooni märki ära jättes

$$Y \vee X\,Y \vee \overline{X}\,\overline{Y}.$$

Viimast valemit võib muide veelgi lihtsustada, mis aga pole praeguses näites eriti oluline. Loogiliste valemite teisendamist, eriti aga lihtsustamist, käsitleme üksikasjalikumalt paragrahvides 7 ja 8.

## §7. Lausearvutuse valemite normaalkujud.

Paljude ülesannete lahendamisel on otstarbekohane kasutada üldkujuliste loogiliste valemite asemel teatud erikujulisi valemeid nn. konjunktiivseid ja disjunktiivseid normaalkujusid.

Käesolevas paragrahvis vaatlemegi nende normaalkujude leidmist ja nende omadusi.

Kõigepealt esitame mõned tähised ja mõisted.

Sümboleid 1 ja 0 astmenäitaja kohal mõistame järgmiselt:

$$X^1 = X,\ X^0 = \bar{X}.$$

Seega kehtivad võrdused

$$1^1 = 1,\ 1^0 = 0,\ 0^1 = 0,\ 0^0 = 1$$

ehk kokkuvõtlikult

$$\alpha^\beta = \begin{cases} 1, & \text{kui } \alpha = \beta \\ 0, & \text{kui } \alpha \neq \beta \end{cases}$$

kus $\alpha$ ja $\beta$ on kahendmuutujad, mis võivad omandada väärtusi 1 ja 0.

Konjunktsiooni

$X_1^{\alpha_1} X_2^{\alpha_2} \dots X_n^{\alpha_n}$ ($\alpha_1, \alpha_2, \dots, \alpha_n$ on kahendmuutuja)(1)

nimetame elementaarkonjunktsiooniks, kui selles iga muutuja $X_k$ (kas kujul $X_k$ või $\bar{X}_k$) esineb mitte üle ühe korra. Konjunktsiooni (1) nimetame täielikuks elementaarkonjunktsiooniks, kui selles iga muutuja (mingist kindlast muutujate hulgast) esineb täpselt üks kord.

Analoogiliselt võime defineerida elementaardisjunktsiooni ja täieliku elementaardisjunktsiooni mõiste.

Teoreem 1. Iga loogilist valemit $\mathcal{A}(X_1, X_2, \dots, X_k X_{k+1}, \dots, X_n)$, kus $n \geqslant 1$, võib esitada kujul[4]

---

[4] Kirjutiste $\bigvee_{\alpha_1,\dots,\alpha_k} \mathcal{A}(\alpha_1,\dots,\alpha_k)$ ja $\&_{\alpha_1,\dots,\alpha_k} \mathcal{A}(\alpha_1,\dots,\alpha_k)$ abil tähistame vastavalt disjunktsiooni ja konjunktsiooni liikmetest $\mathcal{A}(\alpha_1,\dots,\alpha_k)$, kusjuures muutujad $\alpha_1,\dots,\alpha_k$ omandavad kõikvõimalikud väärtused 1 ja 0 üksteisest sõltumatult.

$$\mathcal{A}(X_1, X_2, \ldots, X_k\, X_{k+1}, \ldots X_n) =$$

$$= \bigvee_{\alpha_1, \ldots, \alpha_k} X_1^{\alpha_1} X_2^{\alpha_2} \ldots X_k^{\alpha_k}\ \mathcal{A}(\alpha_1, \alpha_2, \ldots, \alpha_k, X_{k+1}, \ldots, X_n). \quad (2)$$

Tõestus. Näitame, et võrdus (2) kehtib muutujate mingite suvaliste väärtuste $X_1 = \beta_1$, $X_2 = \beta_2, \ldots, X_k = \beta_k$ $X_{k+1} = \beta_{k+1}, \ldots$ $\ldots, X_n = \beta_n$ korral. Ilmselt kehtib võrdus

$$X_1^{\alpha_1} X_2^{\alpha_2} \ldots X_k^{\alpha_k} = \begin{cases} 1, \text{ kui } X_1 = \alpha_1,\ X_2 = \alpha_2, \ldots,\ X_k = \alpha_k; \\ 0, \text{ teistel juhtudel.} \end{cases} \quad (3)$$

Anname muutujatele $\alpha_1, \alpha_2, \ldots, \alpha_k$ kõikvõimalikud väärtused ja moodustame konjunktsioonid

$$\beta_1^{\alpha_1}\ \beta_2^{\alpha_2} \ldots \beta_k^{\alpha_k}\ \mathcal{A}(\alpha_1, \alpha_2, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n). \quad (4)$$

Võrduse (3) tõttu on kõik liikmed (4) väärtusega 0, välja arvatud vahest üksainus, kus $\alpha_1 = \beta_1, \alpha_2 = \beta_2, \ldots, \alpha_k = \beta_k$. Arvestades seda, et disjunktsioonis võib ära jätta nulliga võrduvad liikmed (§6, võrdus (26), saame võrduse (2) paremal pool ainult ühe liikme, kus $\alpha_1 = \beta_1, \alpha_2 = \beta_2, \ldots, \alpha_k = \beta_k$. Et sel juhul $\beta_1^{\alpha_1} \beta_2^{\alpha_2} \ldots \beta_k^{\alpha_k} = \beta_1^{\beta_1} \beta_2^{\beta_2} \ldots \beta_k^{\beta_k} = 1$, siis võib selle konjunktsiooni kirjutamata jätta (§6, võrdus (21)), millega saame võrduse (2) paremale poole

$$\mathcal{A}(\beta_1, \beta_2, \ldots, \beta_k, \beta_{k+1}, \ldots, \beta_n), \text{ m.o.t.t.}$$

Teoreem 2. Kui $\mathcal{A}(X_1, X_2, \ldots, X_n)$ pole samaselt võrdne nulliga, siis kehtib võrdus

$$\mathcal{A}(X_1, X_2, \ldots, X_n) = \bigvee_{\substack{\alpha_1, \ldots, \alpha_n \\ \mathcal{A}(\alpha_1, \ldots, \alpha_n) = 1}} X_1^{\alpha_1} X_2^{\alpha_2} \ldots X_n^{\alpha_n}. \quad (5)$$

Tõestus. Teoreem 1 põhjal võime kirjutada

$$\mathcal{A}(X_1, X_2, \ldots, X_n) = \bigvee_{\alpha_1, \ldots, \alpha_n} X_1^{\alpha_1} X_2^{\alpha_2} \ldots X_n^{\alpha_n} \mathcal{A}(\alpha_1, \alpha_2, \ldots, \alpha_n) \quad (6)$$

Et liikmed $X_1^{\alpha_1} X_2^{\alpha_2} \dots X_n^{\alpha_n} \mathcal{A}(\alpha_1, \alpha_2, \dots, \alpha_n)$ võrduvad nulliga, kui $\mathcal{A}(\alpha_1, \alpha_2, \dots, \alpha_n) = 0$, siis võime nad disjunktsioonis (6) kirjutamata jätta. Arvestades ainult neid liikmeid, kus $\mathcal{A}(\alpha_1, \alpha_2, \dots, \alpha_n) = 1$, saamegi võrduse (5).

Võrduse (5) paremat poolt nimetatakse loogilise valemi $\mathcal{A}(X_1, X_2, \dots, X_n)$ täielikuks disjunktiivseks normaalkujuks. Teoreemi 2 põhjal võime väita, et igal valemil $\mathcal{A}(X_1, X_2, \dots, X_n) \neq 0$ on olemas täielik disjunktiivne normaalkuju. Võrdusest (5) nähtub, et täieliku disjunktiivse normaalkuju iga liige (täielik elementaarkonjunktsioon) vastab ühele kindlale muutujate väärtuste valikule - nn. väärtustele, mille korral see valem on väärtusega 1. See asjaolu võimaldab täieliku disjunktiivse normaalkuju kohe üles kirjutada, lähtudes näiteks loogilise funktsiooni esitusest tabelina. Muuhulgas saame järeldada ka, et iga kahendmuutuja funktsiooni esitamiseks piisab konjunktsiooni, disjunktsiooni ja eituse tehtest.

Teoreem 3. Iga loogilist valemit $\mathcal{A}(X_1, X_2, \dots, X_k, X_{k+1}, \dots, X_n)$, kus $n \geqslant 1$, võib esitada kujul

$$\mathcal{A}(X_1, X_2, \dots, X_k, X_{k+1}, \dots, X_n) =$$

$$= \&_{\alpha_1, \dots, \alpha_k} (\bar{X}_1^{\alpha_1} \vee \bar{X}_2^{\alpha_2} \vee \dots \vee \bar{X}_k^{\alpha_k} \vee \mathcal{A}(\alpha_1, \dots, \alpha_k, X_{k+1}, \dots, X_n)).$$

Tõestuseks rakendame valemile $\neg\mathcal{A}(X_1, \dots, X_n)$ teoreemi 1 ja duaalsust (§6, viide 3).

Teoreem 4. Kui $\mathcal{A}(X_1, X_2, \dots, X_n)$ pole samaselt võrdne ühega, siis kehtib võrdus

$$\mathcal{A}(X_1, X_2, \dots, X_n) =$$

$$= \&_{\substack{\alpha_1, \dots, \alpha_n \\ \mathcal{A}(\alpha_1, \dots, \alpha_n) = 0}} (\bar{X}_1^{\alpha_1} \vee \bar{X}_2^{\alpha_2} \vee \dots \vee \bar{X}_n^{\alpha_n}) \qquad (8)$$

Tõestus on analoogiline teoreemi 3 tõestusega, teoreemi 1 asemel tuleb rakendada aga teoreemi 2. Teoreemid 3 ja 4 tõestada lugejatel.

Võrduse (8) paremat poolt nimetatakse loogilise valemi $\mathcal{A}(x_1, x_2, \ldots, x_n)$ täielikuks konjunktiivseks normaalkujuks. Teoreemi 4 põhjal võime väita, et igal valemil $\mathcal{A}(x_1 x_2, \ldots, x_n) \neq 1$ on olemas täielik konjunktiivne normaalkuju. Võrdusest (8) nähtub, et täieliku konjunktiivse normaalkuju iga liige (täielik elementaardisjunktsioon) vastab ühele kindlale muutujate väärtuste valikule, mille korral see valem on väärtusega 0. See asjaolu võimaldab mõnel juhul ka täielikku konjunktiivset normaalkuju üles kirjutada. Nagu näeme, on olukord täielike disjunktiivsete ja täielike konjunktiivsete normaalkujude puhul ansloogiline.

Toome nüüd näiteid täielike normaalkujude leidmise kohta.

Näide 1. Leiame tabeliga

| X Y Z | f (X, Y, Z) |
|---|---|
| 0 0 0 | 1 |
| 0 0 1 | 0 |
| 0 1 0 | 0 |
| 0 1 1 | 1 |
| 1 0 0 | 0 |
| 1 0 1 | 1 |
| 1 1 0 | 0 |
| 1 1 1 | 1 |

defineeritud funktsiooni f(X, Y, Z) täieliku disjunktiivse ja täieliku konjunktiivse normaalkuju. Valemite (5) ja (8) põhjal saame vastavalt

$$f(X, Y, Z) = X^0 Y^0 Z^0 \vee X^0 Y^1 Z^1 \vee X^1 Y^0 Z^1 \vee X^1 Y^1 Z^1 = \bar{X}\,\bar{Y}\,\bar{Z} \vee \bar{X}\,Y\,Z \vee X\,\bar{Y}\,Z \vee X\,Y\,Z \qquad (9)$$

ja

$$f(X, Y, Z)$$

$$(\bar{X}^0 \vee \bar{Y}^0 \vee \bar{Z}^1) \& (\bar{X}^0 \vee \bar{Y}^1 \vee \bar{Z}^0) \& (\bar{X}^1 \vee \bar{Y}^0 \vee \bar{Z}^0) \& (\bar{X}^1 \vee \bar{Y}^1 \vee \bar{Z}^0) =$$

$$= (X \vee Y \vee \bar{Z}) \& (X \vee \bar{Y} \vee Z) \& (\bar{X} \vee Y \vee Z) \& (\bar{X} \vee \bar{Y} \vee Z).$$

Näide 2. Leiame valemi

$$X \sim (Y \approx Z)$$

täieliku disjunktiivse ja täieliku konjunktiivse normaalkuju.

Kasutame täieliku disjunktiivse normaalkuju leidmiseks valemit (2).

Leiame

$$X \sim (Y \approx Z) = \underset{X=1}{X}\,[1 \sim (Y \approx Z)] \vee \overline{X}\,\underset{X=0}{[0 \sim (Y \approx Z)]} =$$

$$= X\,(Y \approx Z) \vee \overline{X}\,(Y \sim Z) = X\,[\underset{Y=1}{Y}\,(1 \approx Z) \vee \overline{Y}\,\underset{Y=0}{(0 \approx Z)}] \vee$$

$$\vee \overline{X}\,[\underset{Y=1}{Y}\,(1 \sim Z) \vee \overline{Y}\,\underset{Y=0}{(0 \sim Z)}] =$$

$$= X\,Y\,\overline{Z} \vee X\,\overline{Y}\,Z \vee \overline{X}\,Y\,Z \vee \overline{X}\,\overline{Y}\,\overline{Z}$$

Täieliku konjunktiivse normaalkuju leidmiseks kasutame valemit (7)

$$X \sim (Y \approx Z) =$$

$$= \Big\{ X \vee \underset{X=0}{[0 \sim (Y \approx Z)]} \Big\} \;\&\; \Big\{ \overline{X} \vee \underset{X=1}{[1 \sim (Y \approx Z)]} \Big\} =$$

$$= [X \vee (Y \sim Z)] \;\&\; [\overline{X} \vee (Y \approx Z)] =$$

$$= [X \vee Y \vee \underset{Y=0}{(0 \sim Z)}] \;\&\; [X \vee \overline{Y} \vee \underset{Y=1}{(1 \sim Z)}] \;\&$$

$$\&\; [\overline{X} \vee Y \vee \underset{Y=0}{(0 \approx Z)}] \;\&\; [\overline{X} \vee \overline{Y} \vee \underset{Y=1}{(1 \approx Z)}] =$$

$$= (X \vee Y \vee \overline{Z}) \;\&\; (X \vee \overline{Y} \vee Z) \;\&$$

$$\&\; (\overline{X} \vee Y \vee Z) \;\&\; (\overline{X} \vee \overline{Y} \vee \overline{Z}).$$

Lõpuks märgime, et paljudel juhtudel on võimalik täielikke normaalkujusid lihtsustada. Nii näiteks saame lihtsustada täielikku disjunktiivset normaalkuju (9). Tõepoolest, kaht viimast täielikku elementaarkonjunktsiooni saab teisendada nii

$$X\,\overline{Y}\,Z \vee X\,Y\,Z = X\,Z\,(\overline{Y} \vee Y) = X\,Z \;\&\; 1 = X\,Z.$$

Seega saame

$$f(X, Y, Z) = \overline{X}\,\overline{Y}\,\overline{Z} \vee \overline{X}\,Y\,Z \vee X\,Z \qquad (10)$$

Võrduse (10) paremat poolt nimetatakse disjunktiivseks nor-

maalkujuks. Disjunktiivse normaalkuju liikmetele (elementaarkonjunktsioonidele) vastavad loomulikult samuti muutujate väärtuste valikud, mille korral see valem on väärtusega 1. Kuid erinevalt täielikest elementaarkonjunktsioonidest vastab mittetäielikule elementaarkonjunktsioonile $2^{\ell}$ sellist väärtustust ($\ell$ on vastavas elementaarkonjunktsioonis puuduvate muutujate arv).

Analoogiline on olukord ka konjunktiivse normaalkujuga. Nii näiteks saame täielikku konjunktiivset normaalkuju

$$(X \vee \bar{Y} \vee Z) \,\&\, (X \vee \bar{Y} \vee \bar{Z}) \,\&\, (\bar{X} \vee \bar{Y} \vee Z) \qquad (11)$$

lihtsustada nii

$$[(X \vee \bar{Y}) \vee (Z \,\&\, \bar{Z})] \,\&\, (\bar{X} \vee \bar{Y} \vee Z) =$$

$$= [(X \vee \bar{Y}) \vee 0] \,\&\, (\bar{X} \vee \bar{Y} \vee Z) =$$

$$= (X \vee \bar{Y}) \,\&\, (\bar{X} \vee \bar{Y} \vee Z) \qquad (12)$$

Konjunktiivses normaalkujus (12) vastab elementaardisjunktsioonile $X\vee\bar{Y}$ 2 väärtustust, mille korral see valem on väärtusega 0. Need väärtustused on X=0, Y=1, Z=1 ja X=0, Y=1, Z=0.

Tekib küsimus, kas normaalkujusid (10) ja (11) saaks veelgi lihtsustada. Vaatleme seda küsimust paragrahvis 8.

## §8. Lausearvutuse valemite minimiseerimine.

On ilmne, et loogiliste valemitega opereerimisel on otstarbekohane need valemid esitada võimalikult lihtsal kujul. Eriti oluline on loogiliste valemite lihtsustamine arvutustehnika seadmete konstrueerimisel, kus seade konstrueeritakse vahetult seadme tööd kirjeldava loogilise valemi alusel. Käesolevas paragrahvis pole meil võimalik anda loogiliste valemite ja funktsioonide minimiseerimise teooria ammendavat käsitlust: see võiks olla terve erikursuse teemaks. Vaatleme siin ainult konjunktiivsete ja disjunktiivsete normaalkujude minimiseerimise üht kõige lihtsamat meetodit.

Kõigepealt tuleb selgitada, kuidas mõistame terminit "minimiseerima". Konjunktiivse (resp.disjunktiivse) normaal-

kuju minimiseerimise all mõistame selle normaalkuju taandamist niisugusele konjunktiivsele (resp.disjunktiivsele) normaalkujule, mis 1) on lähtenormaalkujuga loogiliselt samaväärne ja 2) kus lausemärkide koguarv on minimaalne (kõikide teiste tingimuste 1) rahuldavate normaalkujudega võrreldes).

Nii on näiteks disjunktiivses normaalkujus

$$X\,Y \vee X\,\overline{Y} \vee \overline{X}\,\overline{Y} \qquad (1)$$

kokku 6 lausemärki. Arvestades seda, et kahte esimest elementaarkonjunktsiooni saab teisendada nii:

$$X\,Y \vee X\,\overline{Y} = X(Y \vee \overline{Y}) = X \,\&\, 1 = X,$$

saab valemit (1) kirjutada kujul

$$X \vee \overline{X}\,\overline{Y},$$

milles on ainult 3 lausemärki. Kuid see kuju pole veel minimaalne. Valemi (1) minimaalseks disjunktiivseks normaalkujuks osutub

$$X \vee \overline{Y}.$$

Märgime, et konjunktiivsete ja disjunktiivsete normaalkujude minimiseerimise käsitlemisel võib piirduda ainult näiteks disjunktiivsete normaalkujude vaatlemisega. Tõepoolest, kui on vaja leida mingi valemi $\mathcal{A}(X_1,X_2,\ldots,X_n)$ minimaalset konjunktiivset normaalkuju, siis võime leida esmalt valemi $\neg\mathcal{A}(X_1,X_2,\ldots,X_n)$ minimaalse disjunktiivse normaalkuju

$$\bigvee_{\nu_1,\ldots\nu_k} X_{\nu_1}^{\alpha_{\nu_1}}\; X_{\nu_2}^{\alpha_{\nu_2}} \;\ldots\; X_{\nu_k}^{\alpha_{\nu_k}}.$$

Viimase valemi esitamisel ja duaalsust arvestades saamegi valemi $\mathcal{A}(X_1,X_2,\ldots,X_n)$ minimaalse konjunktiivse normaalkuju

$$\&_{\nu_1,\ldots,\nu_k} (\overline{X}_{\nu_1}^{\alpha_{\nu_1}} \vee \overline{X}_{\nu_2}^{\alpha_{\nu_2}} \vee \ldots \vee \overline{X}_{\nu_k}^{\alpha_{\nu_k}}).$$

Ülaltoodud väidet saab põhjendada, arvestades võrdusteid

$$\neg\mathcal{A}(X_1,X_2,\ldots,X_n) = \bigvee_{\nu_1,\ldots,\nu_k} X_{\nu_1}^{\alpha_{\nu_1}}\; X_{\nu_2}^{\alpha_{\nu_2}} \;\ldots\; X_{\nu_k}^{\alpha_{\nu_k}}$$

ja

$$\mathcal{A}(X_1,X_2,\ldots,X_n) = \&_{\nu_1,\ldots,\nu_k} (\overline{X}_{\nu_1}^{\alpha_{\nu_1}} \vee \overline{X}_{\nu_2}^{\alpha_{\nu_2}} \vee \ldots \vee \overline{X}_{\nu_k}^{\alpha_{\nu_k}})$$

samaväärsust.

Esitame nüüd disjunktiivsete normaalkujude minimiseerimise ühe lihtsaima meetodi - nn. minimiseerivate tabelite meetodi. Selgitame seda meetodit §7 esitatud näite varal. Olgu

$$f(X, Y, Z) = \bar{X}\bar{Y}\bar{Z} \vee \bar{X}YZ \vee X\bar{Y}Z \vee XYZ \qquad (2)$$

Et valemis (2) on kolm muutujat X, Y, Z, siis moodustame järgmise tabeli.

Tabel 3.

| X Y Z | XY | XZ | YZ | XYZ | f(X,Y,Z) |
|---|---|---|---|---|---|
| $\bar{X}\bar{Y}\bar{Z}$ | $\bar{X}\bar{Y}$ | $\bar{X}\bar{Z}$ | $\bar{Y}\bar{Z}$ | $\bar{X}\bar{Y}\bar{Z}$ | 1 |
| $\bar{X}\bar{Y}Z$ | $\bar{X}\bar{Y}$ | $\bar{X}Z$ | $\bar{Y}Z$ | $\bar{X}\bar{Y}Z$ | 0 |
| $\bar{X}Y\bar{Z}$ | $\bar{X}Y$ | $\bar{X}\bar{Z}$ | $Y\bar{Z}$ | $\bar{X}Y\bar{Z}$ | 0 |
| $\bar{X}YZ$ | $\bar{X}Y$ | $\bar{X}Z$ | $YZ$ | $\bar{X}YZ$ | 1 |
| $X\bar{Y}\bar{Z}$ | $X\bar{Y}$ | $X\bar{Z}$ | $\bar{Y}\bar{Z}$ | $X\bar{Y}\bar{Z}$ | 0 |
| $X\bar{Y}Z$ | $X\bar{Y}$ | $XZ$ | $\bar{Y}Z$ | $X\bar{Y}Z$ | 1 |
| $XY\bar{Z}$ | $XY$ | $X\bar{Z}$ | $Y\bar{Z}$ | $XY\bar{Z}$ | 0 |
| $XYZ$ | $XY$ | $XZ$ | $YZ$ | $XYZ$ | 1 |

Tabeli sisemises osas esinevad kõikvõimalikud elementaarkonjunktsioonid muutujatest X, Y ja Z. Kuid mitte iga elementaarkonjunktsioon ei saa esineda valemiga (1) loogiliselt samaväärses disjunktiivses normaalkujus. Ilmselt ei saa niisuguses disjunktiivses normaalkujus esineda elementaarkonjunktsioonid $\bar{X}\bar{Y}Z$, $\bar{X}Y\bar{Z}$, $X\bar{Y}\bar{Z}$ ja $XY\bar{Z}$ (s.t. kus $f(X,Y,Z) = 0$). Tõepoolest, kui disjunktiivses normaalkujus on näiteks liige $\bar{X}\bar{Y}Z$, siis on see normaalkuju $X = 0$, $Y = 0$, $Z = 1$ korral väärtusega 1, valem (2) on aga muutujate selliste väärtuste korral väärtusega 0. Seega ei ole need kaks valemit (normaalkuju) loogiliselt samaväärsed. Võib väita veelgi rohkem: valemiga (2) loogiliselt samaväärses disjunktiivses normaalkujus ei saa esineda ühtki 2., 3., 5. ja 7. rea (kus $f(X,Y,Z)=0$) elementaarkonjunktsiooni. Nii näiteks, kui disjunktiivses normaalkujus esineks elementaarkonjunktsioon $\bar{X}Z$, siis saame teisendada

$$\bar{X}Z = \bar{X}Z\&1 = XZ\& (Y \vee \bar{Y}) = \bar{X}YZ \vee \bar{X}\bar{Y}Z,$$

millest näeme, et nimetatud disjunktiivne normaalkuju on X = Y = 0, Z = 1 korral väärtusega 1. Valem (2) on muutujate nendesamade väärtuste korral aga väärtusega 0. Seega tuleb jätta vaatlusest välja (maha tõmmata) kõik 2., 3., 5. ja 7. rea elementaarkonjunktsioonid. Et tabelis esinevad samad elementaarkonjunktsioonid korduvalt, siis tuleb peale selle kontrollida liikmeid ka veergudes, tõmmates maha need elementaarkonjunktsioonid, mis ridades on juba maha tõmmatud. (Näiteks, et $\bar{X}Z$ on teises reas maha tõmmatud, tuleb $\bar{X}Z$ maha tõmmata ka 5. veeru ja 4. rea ristumiskohas). Pärast ülalmärgitud mahatõmbamisi saame tabeli järgmisel kujul:

Tabel 4.

| X | Y | Z | XY | XZ | YZ | XYZ | f(X,Y,Z) |
|---|---|---|---|---|---|---|---|
| ~~$\bar{X}$~~ | ~~$\bar{Y}$~~ | ~~$\bar{Z}$~~ | ~~$\bar{X}\bar{Y}$~~ | ~~$\bar{X}\bar{Z}$~~ | ~~$\bar{Y}\bar{Z}$~~ | (~~$\bar{X}\bar{Y}\bar{Z}$~~) | 1 |
| ~~$\bar{X}$~~ | ~~$\bar{Y}$~~ | ~~$Z$~~ | ~~$\bar{X}\bar{Y}$~~ | ~~$\bar{X}Z$~~ | ~~$\bar{Y}Z$~~ | ~~$\bar{X}\bar{Y}Z$~~ | 0 |
| ~~$\bar{X}$~~ | ~~$Y$~~ | ~~$\bar{Z}$~~ | ~~$\bar{Y}Y$~~ | ~~$\bar{X}\bar{Z}$~~ | ~~$Y\bar{Z}$~~ | ~~$\bar{Y}Y\bar{Z}$~~ | 0 |
| ~~$\bar{X}$~~ | ~~$Y$~~ | ~~$Z$~~ | ~~$\bar{X}Y$~~ | ~~$\bar{X}Z$~~ | ($YZ$) | $\bar{X}YZ$ | 1 |
| ~~$X$~~ | ~~$\bar{Y}$~~ | ~~$\bar{Z}$~~ | ~~$X\bar{Y}$~~ | ~~$X\bar{Z}$~~ | ~~$\bar{Y}\bar{Z}$~~ | ~~$X\bar{Y}\bar{Z}$~~ | 0 |
| ~~$X$~~ | ~~$\bar{Y}$~~ | ~~$Z$~~ | ~~$X\bar{Y}$~~ | ($XZ$) | ~~$\bar{Y}Z$~~ | $X\bar{Y}Z$ | 1 |
| ~~$X$~~ | ~~$Y$~~ | ~~$\bar{Z}$~~ | ~~$XY$~~ | ~~$X\bar{Z}$~~ | ~~$Y\bar{Z}$~~ | ~~$XY\bar{Z}$~~ | 0 |
| ~~$X$~~ | ~~$Y$~~ | ~~$Z$~~ | ~~$XY$~~ | ($XZ$) | ($YZ$) | $XYZ$ | 1 |

Et otsitav (minimaalne) disjunktiivne normaalkuju oleks esialgsega loogiliselt samaväärne, tuleb võtta igast reast, kus f(X,Y,Z) = 1, vähemalt üks (mahatõmbamata!) elementaarkonjunktsioon. Elementaardisjunktsioonid tuleb valida aga nii, et saadav disjunktiivne normaalkuju oleks minimaalne. Antud näites on see kergesti teostatav. Minimaalse disjunktiivse normaalkuju liikmed on ümbritsetud joonega. Seega on f(X,Y,Z) minimaalne disjunktiivne normaalkuju.

$$X\,Z \vee Y\,Z \vee \bar{X}\,\bar{Y}\,\bar{Z} \qquad (3)$$

Valemis (3) on lausemärkide koguarv 7, valemis (2) aga 12. Minimiseerimise tulemusel saime seega lausemärkide üldarvu

vähendada 5 võrra.

Märgime, et informatsiooni võib tabelitesse kanda palju otstarbekohasemalt ja ökonoomsemalt. Et igas veerus asub kindlate muutujate kombinatsioon, siis on vaja tähistada ainult seda, missugune neist on eitusega, missugune mitte. Niisiis võime näiteks 6. veeru elementaarkonjunktsioonide

$$\bar{Y}\bar{Z},\ \bar{Y}Z,\ Y\bar{Z},\ YZ$$

asemel kirjutada lihtsalt

$$00,\ 01,\ 10,\ 11,$$

mida on otstarbekohane ruumi kokkuhoiu mõttes üles kirjutada 8-ndsüsteemis

$$0,\ 1,\ 2,\ 3.$$

Selline lihtsustus minimiseerivate tabelite juures on muidugi eriti oluline, kui muutujate arv on juba suurem kui 4.

Näide 1. Leiame

$$f(X,Y,Z) = (X \vee Y \vee \bar{Z}) \,\&\, (X \vee \bar{Y} \vee Z) \,\&\, (\bar{X} \vee Y \vee Z) \,\&\, (\bar{X} \vee \bar{Y} \vee Z) \qquad (4)$$

minimaalse konjunktiivse normaalkuju. Selleks tuleb leida

$$\neg f(X,Y,Z) = \bar{X}\bar{Y}Z \vee \bar{X}Y\bar{Z} \vee X\bar{Y}\bar{Z} \vee XY\bar{Z} \qquad (5)$$

minimaalne disjunktiivne normaalkuju.

Koostame vastava tabeli ja teostame vajalikud mahakriipsutused.

Tabel 5.

| X Y Z | XY | XZ | YZ | XYZ | f(X,Y,Z) |
|---|---|---|---|---|---|
| ~~0 0 0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | 0 |
| ø ø x | ø | x | x | (1) | 1 |
| ø x ø | x | ø | (2) | 2 | 1 |
| ~~0 1 1~~ | ~~1~~ | ~~1~~ | ~~3~~ | ~~3~~ | 0 |
| x ø ø | x | (2) | ø | 4 | 1 |
| ~~1 0 1~~ | ~~2~~ | ~~3~~ | ~~1~~ | ~~5~~ | 0 |
| x x ø | x | (2) | (2) | 6 | 1 |
| ~~1 1 1~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~7~~ | 0 |

Seega: valemi (5) minimaalne disjunktiivne normaalkuju tuleb

$$X\bar{Z} \vee Y\bar{Z} \vee \bar{X}\bar{Y}Z,$$

millest eitades (ja duaalsust kasutades) saame valemi (4) minimaalse konjunktiivse normaalkuju

$$(\bar{X} \vee Z) \,\&\, (\bar{Y} \vee Z) \,\&\, (X \vee Y \vee \bar{Z})$$

Näide 2. Leiame

$$f(x_1,x_2,x_3,x_4) =$$

$$= \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3x_4 \vee$$
$$\vee x_1x_2\bar{x}_3\bar{x}_4 \vee x_1x_2x_3x_4$$

minimaalse disjunktiivse normaalkuju. Vastav tabel 4 muutuja korral tuleb muidugi mõnevõrra erinev kolme muutuja funktsioonide minimiseerimise tabelist. Kuid selle koostamise põhimõte on täiesti samasugune nagu kolme muutuja tabeli puhul.

Esitame vastava tabeli koos vajalike mahakriipsutustega

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_1x_2$ | $x_1x_3$ | $x_1x_4$ | $x_2x_3$ | $x_2x_4$ | $x_3x_4$ | $x_1x_2x_3$ | $x_1x_2x_4$ | $x_1x_3x_4$ | $x_2x_3x_4$ | $x_1x_2x_3x_4$ | $f$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 | 1 | 0 | 2 | 2 | 2 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 3 | 3 | 3 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 2 | 2 | 0 | 4 | 4 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 5 | 5 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 3 | 2 | 2 | 3 | 2 | 2 | 6 | 6 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 7 | 7 | 1 |
| 1 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 4 | 4 | 4 | 0 | 10 | 1 |
| 1 | 0 | 0 | 1 | 2 | 2 | 3 | 0 | 1 | 1 | 4 | 5 | 5 | 1 | 11 | 1 |
| 1 | 0 | 1 | 0 | 2 | 3 | 2 | 1 | 0 | 2 | 5 | 4 | 6 | 2 | 12 | 0 |
| 1 | 0 | 1 | 1 | 2 | 3 | 3 | 1 | 1 | 3 | 5 | 5 | 7 | 3 | 13 | 1 |
| 1 | 1 | 0 | 0 | 3 | 2 | 2 | 2 | 2 | 0 | 6 | 6 | 4 | 4 | 14 | 1 |
| 1 | 1 | 0 | 1 | 3 | 2 | 3 | 2 | 3 | 1 | 6 | 7 | 5 | 5 | 15 | 0 |
| 1 | 1 | 1 | 0 | 3 | 3 | 2 | 3 | 2 | 2 | 7 | 6 | 6 | 6 | 16 | 0 |
| 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 7 | 7 | 7 | 7 | 17 | 1 |

Mahatõmbamata mittetäielikud elementaarkonjunktsioonid on ümbritsetud kastikestega. Nagu vahetu proovimine näitab, on siin minimaalseid disjunktiivseid normaalkujusid mitu. Esitame neist kaks:

$$\mathcal{H}_1 = \bar{X}_1X_2\bar{X}_3 \vee X_1\bar{X}_2\bar{X}_3 \vee \bar{X}_1X_2X_4 \vee X_1\bar{X}_3\bar{X}_4 \vee X_1X_3X_4,$$

$$\mathcal{H}_2 = \bar{X}_1X_2X_4 \vee X_1\bar{X}_2X_4 \vee X_1\bar{X}_3\bar{X}_4 \vee X_2\bar{X}_3\bar{X}_4 \vee X_2X_3X_4.$$

Selle näite puhul ilmneb ka minimiseerivate tabelite meetodi oluline puudus: suure arvu muutujate korral pole minimaalse normaalkuju leidmine sugugi lihtne ja see taandub sisuliselt elementaarkonjunktsioonidega kombineerimisele. Seepärast on otstarbekohasem suure muutujate arvu korral kasutada teisi meetodeid.

Lõpuks esitame veel ühe minimiseerivate tabelite meetodi rakendusvõimaluse, nimelt mitte kõikjal defineeritud funktsioonide optimaalsel laiendamisel. Niisugused ülesanded tekivad sageli arvutustehnika seadmete konstrueerimisel. Seadme funktsioneerimise seisukohalt on vastava funktsiooni lisadefinitsioonid kõik samaväärsed. Seadme konstrueerimisel on aga olulised niisugused lisadefinitsioonid, mis annavad näiteks kõige lihtsama disjunktiivse normaalkuju. Selgitame seda ülesannet näite varal.

<u>Näide 3</u>. Defineerida täiendavalt mitte kõikjal defineeritud funktsiooni f(X,Y,Z),

Tabel 6.

| X Y Z | f(X,Y,Z) |
|---|---|
| 0 0 0 | 1 |
| 0 0 1 | 0 |
| 0 1 0 | defineerimata |
| 0 1 1 | defineerimata |
| 1 0 0 | defineerimata |
| 1 0 1 | defineerimata |
| 1 1 0 | 0 |
| 1 1 1 | 1 |

millest lähtudes saaks kõige lihtsama disjunktiivse normaalkuju.

Minimiseerivate tabelite meetod võimaldab kohe leida vastava minimaalse disjunktiivse normaalkuju. Võrreldes eespool kirjeldatud meetodiga, tuleb siin teha üks väike muudatus: f(X,Y,Z) väärtuste veerus esineb peale väärtuste 1 ja 0 ka "väärtus" "defineerimata". Muus osas jääb meetod formaalselt muutusteta: maha tuleb tõmmata read, kus $f(X,Y,Z) = 0$ ja seejärel tuleb maha tõmmata ka veergudes need elementaarkonjunktsioonid, mis ridades on juba maha tõmmatud. Ridadest, kus $f(X,Y,Z) = 1$, tuleb võtta vähemalt üks mahatõmbamata elementaarkonjunktsioon, nii et disjunktiivne normaalkuju tuleks minimaalne.

| X Y Z | XZ | XZ | YZ | XYZ | f(X, Y, Z) |
|---|---|---|---|---|---|
| ø ø ø | ø | [0] | [0] | 0 | 1 |
| ~~0 0 1~~ | ~~0~~ | ~~1~~ | ~~1~~ | ~~1~~ | 0 |
| ø X ø | 1 | 0 | X | 2 | def. |
| ø X X | 1 | X | 3 | 3 | def. |
| X ø ø | 2 | X | 0 | 4 | def. |
| X ø X | 2 | 3 | X | 5 | def. |
| ~~1 1 0~~ | ~~3~~ | ~~2~~ | ~~2~~ | ~~6~~ | 0 |
| X X X | 3 | [3] | [3] | 7 | 1 |

Tabelist nähtub, et otsitavaid minimaalseid disjunktiivseid normaalkujusid on 4:

$$\mathcal{N}_1 = \bar{X}\bar{Z} \vee XZ, \quad \text{(valem ei sisalda muutujat Y)}$$

$$\mathcal{N}_2 = \bar{X}\bar{Z} \vee YZ,$$

$$\mathcal{N}_3 = \bar{X}\bar{Z} \vee \bar{Y}\bar{Z},$$

$$\mathcal{N}_4 = \bar{Y}\bar{Z} \vee YZ, \quad \text{(valem ei sisalda muutujat X)}$$

Normaalkujud $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4$ määravad juba kõikjal defineeritud kahendmuutuja funktsioonid, mis ongi f(X,Y,Z) optimaalseteks laienditeks. Märgime, et mõnel juhul (valemid $\mathcal{N}_1$ ja $\mathcal{N}_4$) on võimalik vähendada sel viisil muutujate arvu. Skeemide (seadmete) konstrueerimisel võib osutuda mõnede sisendite ehk

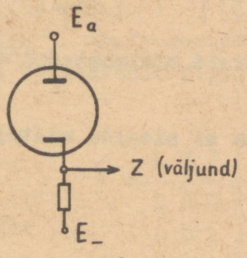
muutujate kõrvaldamine oluliseks sel korral, kui nende töö toimub suhteliselt ebastabiilsemalt.

## §9. Elektroskeemide elemendid ja loogiliste tehete realiseerimine nende abil.

Käesolevas paragrahvis käsitleme elektronskeemide elemente loogiliste tehete realiseerimise seisukohast. Seetõttu jätame vaatlusest välja paljud niisugused elemendid (takistid, kondensaatorid jne.), mis on küll vajalikud seadmete funktsioneerimiseks, kuid ei teosta mingeid loogilisi operatsioone. Samuti jätame vaatlusest välja niisugused (tehnilisest seisukohast olulised) aspektid nagu elementide stabiilsus, impulsside võimendamine ning nende esialgse kuju taastamine, seadme töö ajaline kooskõlastamine jne.

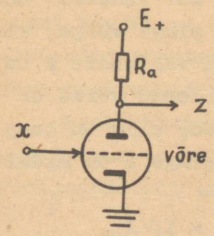
Järgnevas osas lepime kokku tähistada kõrget pinget sümboliga "1", madalat pinget sümboliga "0", samuti tähistame ka positiivset ja negatiivset vooluimpulssi.

### 1. Elektronlambid.




Joon. 1.

Diood (joon. 1) funktsioneerib järgmiselt: kui pinge anoodil on kõrge ($E_a = 1$) ja katoodil madal ($E_- = 0$), siis läbib dioodi elektrivool anoodilt katoodile. Väljundis Z on sel juhul kõrge pinge ($Z=1$). Kui anoodi pinge on madal, siis elektrivool dioodi ei läbi ja väljundis Z on pinge madal ($Z=0$). Praktikas on lampdioodid enamikus asendatud pooljuhtdioodidega. Triood (joon. 2) funktsioneerib järgmiselt: kui võrel X (see on sisendiks) on kõrge pinge, siis läbib lampi vool ja takistil $R_a$ tekib pingelangus. Väljundis Z on sel juhul madal pinge.




Joon. 2.

Kui võrel X on madal pinge, siis on lamp suletud (s.t. lampi ei läbi vool) ja väljundis Z on kõrge pinge. Märgime, et siin ja edaspidi $E_+ \equiv 1$.

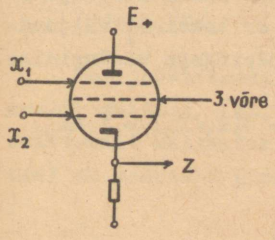
Tähistades trioodi poolt teostatava tehte T abil, saame

$$T(X) = \begin{cases} 1, & \text{kui } X = 0, \\ 0, & \text{kui } X = 1, \end{cases}$$

ehk

$$Z = T(X) = \overline{X}.$$

Seega: triood realiseerib loogilise eituse.

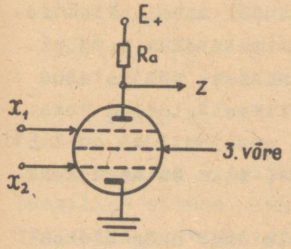



Joon. 3.

Pentood $P_1$ (joon. 3) funktsioneerib järgmiselt: kui 1. ja 2. võrel on kõrge pinge (3. võrel on konstantne kõrge pinge), siis läbib lampi vool ja väljundis Z on kõrge pinge. Kui kas 1. või 2. (või mõlemal) võrel on madal pinge, on lamp suletud ning pinge väljundis Z madal. Seega

$$Z = P_1(X_1, X_2) = X_1 \& X_2,$$

s.t. pentood $P_1$ realiseerib konjunktsiooni tehte.

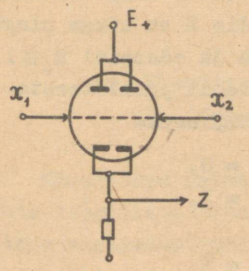



Joon. 4.

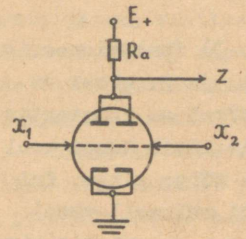
Pentood $P_2$ (joon. 4) funktsioneerib järgmiselt: kui 1. ja 2. võrel on kõrge pinge (3. võrel on konstantne kõrge pinge), siis läbib lampi vool ja takistil $R_a$ tekib pingelangus. Väljundis Z tekib madal pinge. Kui kas 1. või 2. (või mõlemal) võrel on madal pinge, on lamp suletud ning pinge väljundis Z kõrge. Seega

$$Z = P_2(X_1, X_2) = X_1/X_2 = \overline{X}_1 \vee \overline{X}_2,$$

s.t. pentood $P_2$ realiseerib Shefferi tehte.




Joon. 5.




Joon. 6.

Kaksikkatood $K_1$ (joon. 5) funktsioneerib järgmiselt: kui vähemalt ühel võrel $X_1$ või $X_2$ on pinge kõrge, siis läbib lampi vool ja väljundis Z on pinge kõrge. Kui mõlemad võred on madalalt pingestatud, siis on lamp suletud ja väljundpinge Z on madal. Seega

$$Z = K_1(X_1,X_2) = X_1 \vee X_2,$$

s.t. kaksikkatood $K_1$ realiseerib disjunktsiooni.

Kaksikkatood $K_2$ (joon. 6) erineb kaksikkatoodist $K_1$ selle poolest, et tema väljund asub lambi anoodipoolsel osal. Seetõttu on lambi $K_2$ väljundpinge $K_1$ väljundpingega vastupidine. Seega[5]

$$Z = K_2(X_1,X_2) = \neg(X_1 \vee X_2) = \bar{X}_1 \& \bar{X}_2.$$

## 2. Pooljuhtelemendid.

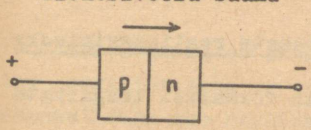
Teatavasti puuduvad pooljuhtides vabad elektronid. Elektrivool tekib pooljuhtides aatomitest lahtirebitud elektronide ja lahtirebimisel tekkinud tühikute (aukude) arvel. Elektronide liikumisest põhjustatud juhtivust nimetatakse n-tüüpi (n- sõnast "negatiivne") juhtivuseks, aukudest põhjustatud juhtivust (augud kujutavad endast positiivselt laetud "osakesi") p-tüüpi juhtivuseks. Et puhastel pooljuhtidel on lahtirebitud elektronide ja aukude arv võrdne, siis on need kaks juhtivust võrdsed.

Lisandite manustamisel pooljuhtidele saab anda ülekaal ühele või teisele juhtivuse tüübile. Kui näiteks lisada 4-va-

[5] Kaksikkatood $K_2$ realiseerib nn. Vebbi tehte, mis defineeritakse nii: $X \circ Y = \bar{X} \& \bar{Y}$.

lentsetele pooljuhtidele germaaniumile või ränile 5-valentset arseeni, muudame need segud n-tüüpi juhtideks, 3-valentse indiumi lisamisel aga p-tüüpi juhtideks. Pooljuhtdioodid ja -trioodid konstrueeritaksegi mitmete erinevat tüüpi juhtivusega pooljuhtide kihtide ühendamisel.

Pooljuhtdioodi (joon. 7a) läbib vool suunas p→n, kui p-piirkond on kõrgelt pingestatud, n-piirkond aga madalalt. Pooljuhtdioodi skemaatiline tähis on esitatud joonisel 7b.

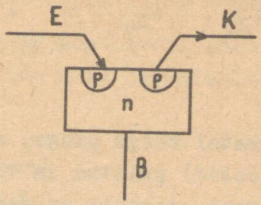



Joon. 7a.



Joon. 7b.

Pooljuhttrioodil ehk transistoril (joon. 8a) on 3 elektroodi: emitter (E), kollektor (K) ja baas (B). Vool liigub emitterilt kollektorile siis, kui baas (mis etendab sisendi osa) on madalalt pingestatud.



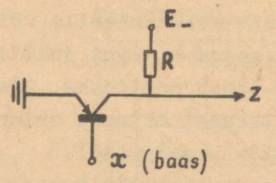

Joon. 8a.



Joon 8b.

Joonisel 9 esitame transistorskeemi, mis realiseerib loogilise eituse. Kui sisend X (baas) on kõrgelt pingestatud, siis on transistor suletud ja pinge väljundil Z madal (võrdne $E_{-}$ pingega). Kui sisend X on pingestatud madalalt, juhib transistor voolu ja pinge väljundil Z on kõrge. Seega toimub esitatud skeemi väljundis tõepoolest sisendpinge

inverteerimine (s.t. eitus).



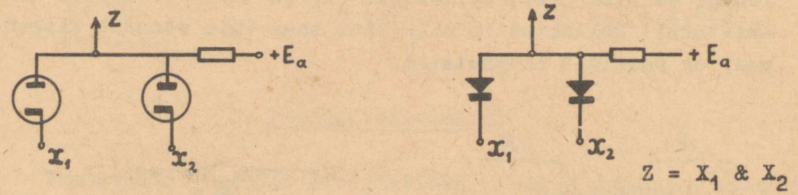

Joon. 9.

Valemites kasutatakse transistori puhul tähist T, samuti nagu lamptrioodi puhul.

## 3. Teisi võimalusi loogiliste tehete realiseerimiseks.

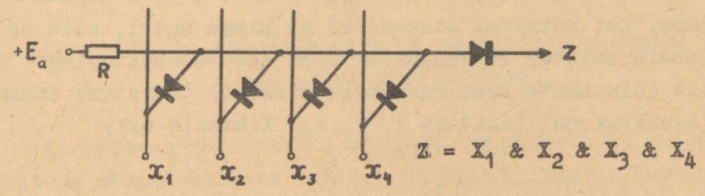
Esitame siin veel mõningaid teisi võimalusi loogiliste tehete realiseerimiseks.

Konjunktsiooni saab realiseerida lamp- või pooljuhtdioodide abil järgmiselt:



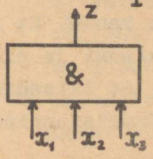

Joon. 10.

Kui sisenditel $X_1$ ja $X_2$ on mõlematel kõrge pinge, siis on mõlemad dioodid (resp. pooljuhtdioodid) suletud ja väljundil Z on kõrge pinge. Kui kasvõi ühel sisendil on madal pinge, on vastav diood avatud. Takistil R toimub pingelangus ja väljundil Z on pinge madal. Märgime, et dioodide abil võib teostada ka enam kui kahe sisendsignaali konjunktsiooni (vt. joon. 11)




Joon. 11.

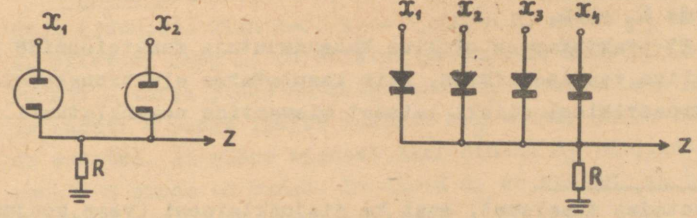
Mõnel juhul on skeemides oluline märkida ainult teostatav loogiline tehe ja pole oluline, missuguste füüsiliste elementidega see realiseeritakse. Sellistel puhkudel tähistatakse näiteks konjunktsioon $X_1 \& X_2 \& X_3$ lihtsalt nii:




Joon. 12.

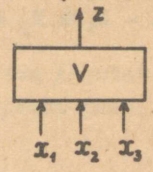
kus $X_1$, $X_2$ ja $X_3$ on sisendid, Z aga on väljund ($Z = X_1 \& X_2 \& X_3$).

Disjunktsiooni saab realiseerida lamp- või pooljuhtdioodide abil järgmiselt:




Joon 13.

On ilmne, et kui vähemalt ühel sisendil on kõrge pinge, siis on vastav diood avatud ja väljundil Z on samuti kõrge pinge. Kui kõikidel sisenditel on pinge madal, siis on kõik dioodid suletud ja väljundil Z on samuti madal pinge. Üldiselt (olenemata realiseerimise viisist) tähistame skeemides disjunktsiooni (näiteks $X_1\ X_2\ X_3$) lihtsalt nii:




($X_1$, $X_2$, $X_3$ - sisendid, Z - väljund).

Joon. 14.

Siinkohal võiks meenutada märkust paragrahvis 7, et iga kahendmuutuja funktsiooni võib esitada konjunktsiooni, disjunktsiooni ja eituse tehte abil.[6] Sellel tulemusel on otsene tähendus ka elektronskeemide kohta. Et eitust saab realiseerida trioodi abil, konjunktsiooni ja disjunktsiooni aga dioodide abil, siis saame väita, et trioodidest ja dioodidest piisab kõigi kahendmuutuja funktsioonide realiseerimiseks.

Et kehtivad võrdused

| | |
|---|---|
| $X / X = \overline{X}$ | $X \circ X = \overline{X}$ |
| $(X/X) / (Y/Y) = X \vee Y$ | $(X \circ X) \circ (Y \circ Y) = X \& Y$ |
| $(X/Y) / (X/Y) = X \& Y$ | $(X \circ Y) \circ (X \circ Y) = X \vee Y,$ |

võib kõiki kahendmuutuja funktsioone esitada ainuüksi Shefferi või Vebbi tehte abil. Skeemide keeles: kõiki kahendmuutuja funktsioone saab realiseerida pentoodide $P_2$ või kaksikkatoodide $K_2$ abil.

Et praktikas on oluline kahendmuutuja funktsioonide ökonoomiline realiseerimine, siis kasutatakse elektronskeemide konstrueerimisel siiski laiemat elementide nomenklatuuri.

---

[6] Kasutades duaalsust, saab ka disjunktsiooni (resp.konjunktsiooni) avaldada eituse ja konjunktsiooni (resp.disjunktsiooni) abil.Seega piisab kõigi kahendmuutuja funktsioonide esitamiseks kahest tehtest: kas konjunktsioonist ja eitusest või disjunktsioonist ja eitusest.

## § 10. Elektronskeemide ja loogiliste skeemide analüüs ning süntees.

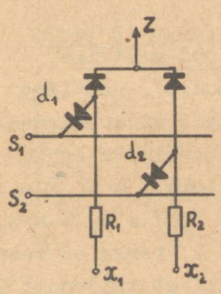
Käesolevas paragrahvis eeldame, et skeemi töö ei sõltu ajast ja skeemi eelnevatest seisunditest. Seega määravad sisendsignaalid üheselt väljundsignaali.

Skeemi analüüsi all mõistame funktsionaalse seose leidmist sisend- ja väljundsignaalide vahel. Tavaliselt esitatakse see seos loogiliste valemite abil.

Toome skeemide analüüsi kohta rida näiteid.

Näide 1. Selgitame, kuidas töötab kaksikventiil (joon.1). Kui sisenditel $X_1$ ja $S_1$ on kõrge pinge, on diood $d_1$ suletud ja kõrge pinge tekib ka väljundil Z. Kui aga $X_1 = 1$ ja $S_1 = 0$, on diood $d_1$ avatud ja takistil $R_1$ tekib pingelangus, mistõttu väljundil Z on madal pinge. Sisendite $X_2$ ja $S_2$ puhul on olukord analoogiline. Seega

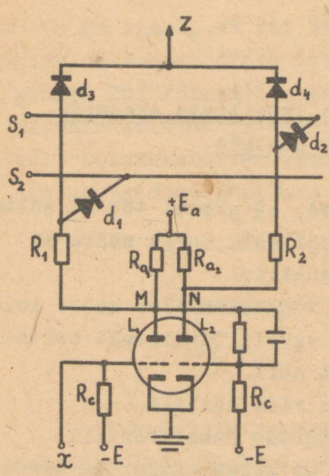
$$Z = (X_1 \& S_1) \vee (X_2 \& S_2).$$




Joon.1.

Näide 2. Selgitame, kuidas töötab ühekohaline teisendaja (joon.2). Olgu X=0, $S_1 = 1$, $S_2 = 0$. Diood $d_1$ on avatud, $d_2$ suletud. Lamp $L_1$ on suletud ja punktis M on kõrge pinge. Punkti M kõrge pinge antakse lambi $L_2$ võrele. Lamp $L_2$ avaneb ja punkti N pinge langeb takisti $Ra_2$ tõttu. Kuigi diood $d_2$ on suletud, ei pääse signaal läbi dioodi $d_4$ väljundile, sest punkti N pinge on madal. Et diood $d_1$ on avatud, ei pääse ka punkti M kõrge pinge väljundile Z.

Olgu $X = 1$, $S_1 = 1$, $S_2 = 0$. Diood $d_1$ on avatud, $d_2$ suletud. Siin saame $M = 0$, $N = 1$. Punkti N kõrge pinge antakse



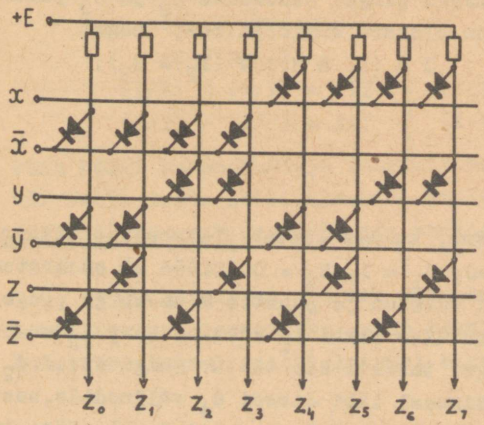

Joon.2.

edasi väljundile Z. Olgu X = 0, $S_1$ = 0, $S_2$ = 1 ($d_1$ on suletud, $d_2$ avatud). Sel juhul on M = 1, N = 0. Punkti M kõrge pinge antakse edasi väljundile Z. Olgu X = 1, $S_1$ = 0, $S_2$ = 1 ($d_1$ on suletud, $d_2$ avatud). Sel juhul on M = 0, N = 1. Et diood $d_2$ on avatud, ei anta punkti N kõrget pinget väljundile Z.

Kerge on näha, et $S_1 = S_2 = 1$ korral on Z = 1 ja $S_1 = S_2 = 0$ korral on Z = 0 olenemata X väärtusest. Seega

$$Z = (X \,\&\, S_1) \vee (\bar{X} \,\&\, S_2).$$

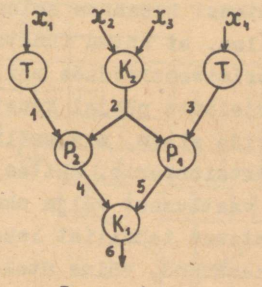
Näide 3. Selgitame, kuidas töötab diood-maatriks (joon.3). Vaatleme mingit sisendsignaalide kombinatsiooni, näiteks X = 0, $\bar{X}$ = 1, Y = 1, $\bar{Y}$ = 0, Z = 0, $\bar{Z}$ = 1. Kerge on veenduda, et väljundil $Z_2$ tekib signaal, kuna teistel väljunditel signaali ei teki. Pole raske kontrollida, et $Z_0 = \bar{X}\,\bar{Y}\,\bar{Z}$, $Z_4 = X\,\bar{Y}\,\bar{Z}$, $Z_1 = \bar{X}\,\bar{Y}\,Z$, $Z_5 = X\,\bar{Y}\,Z$, $Z_2 = \bar{X}\,Y\,\bar{Z}$, $Z_6 = X\,Y\,\bar{Z}$, $Z_3 = \bar{X}\,Y\,Z$, $Z_7 = X\,Y\,Z$. Seega desifreerib antud diood-maatriks sisenditele antava väärtustuse. Diood-maatrik-




Joon. 3.

seid kasutataksegi koodide desifraatoritena.

Näide 4. Selgitame, kuidas töötab joonisel 4 toodud skeem. 1.lülil moodustatakse $T(X_1) = \bar{X}_1$, 2.lülil $K_2(X_2,X_3) = \bar{X}_2 \& \bar{X}_3$, 3.lülil $T(X_4) = \bar{X}_4$, 4.lülil $P_2(\bar{X}_1, \bar{X}_2 \& \bar{X}_3) = X_1 \vee X_2 \vee X_3$, 5.lülil $P_1(\bar{X}_2 \& \bar{X}_3, \bar{X}_4) = \bar{X}_2 \& \bar{X}_3 \& \bar{X}_4$, 6.lülil $K_1(X_1 \vee X_2 \vee X_3, \bar{X}_2 \& \bar{X}_3 \& \bar{X}_4) = X_1 \vee X_2 \vee X_3 \vee \bar{X}_2\bar{X}_3\bar{X}_4$.

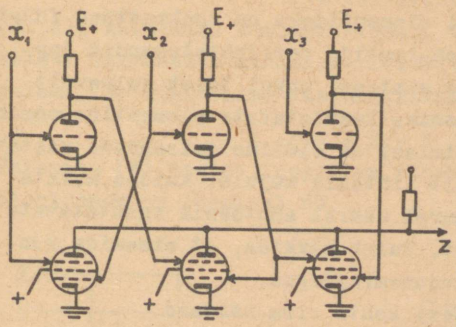



Joon. 4.

Viimast avaldist on võimalik teisendada kujule $X_1 \vee X_2 \vee X_3 \vee \bar{X}_4$. Seega

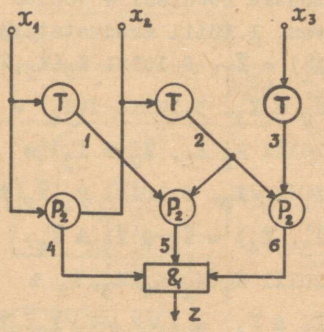
$$Z = X_1 \vee X_2 \vee X_3 \vee \bar{X}_4.$$

Näide 5. Selgitame, kuidas töötab joonisel 5 toodud skeem. Koostame kõigepealt joonisel 5 toodud skeemist lähtudes uue skeemi, kus kasutame tähiseid T ja $P_2$. Märgime, et pentoodide $P_2$ ühise anoodpingega seotud väljund Z tuleb uuel skeemil kujutada nende pentoodide väljundite konjunktsiooni abil.

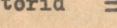



Joon. 5.

Uus skeem on esitatud joonisel 6. Selle skeemi analüüsimisel rakendame eelmises näites kasutatud metoodikat. 1. lülil moodustatakse $T(X_1) = \bar{X}_1$, 2.lülil $T(X_2) = \bar{X}_2$, 3.lülil $T(X_3) = \bar{X}_3$, 4.lülil $P_2(X_1,X_2) = \bar{X}_1 \vee \bar{X}_2$, 5.lülil $P_2(\bar{X}_1,\bar{X}_2) = X_1 \vee X_2$, 6.lülil $P_2(\bar{X}_2,\bar{X}_3) = X_2 \vee X_3$, 7. lülil $Z = (\bar{X}_1 \vee \bar{X}_2) \& (X_1 \vee X_2) \& (X_2 \vee X_3)$.




Joon.6.

Elektronskeemide sünteesi ülesanne on vastupidine analüüsile. Skeemide sünteesi ülesanne seisneb selles, et antud funktsioneerimiseeskirjade või töökirjelduse põhjal konstrueerida skeem, mis selliselt funktsioneerib. Jättes jällegi vaatlusest välja skeemile seatavad tehnilist laadi lisanõuded, võime ütelda, et skeemide töö kirjelduse võib anda loogilise valemiga. Skeemide sünteesi puhul võib kõnelda mitmest tasemest: võime kõnelda loogilisest sünteesist, kus skeemi elementideks on abstraktsed loogilised operaatorid  jne., või sünteesist, kus skeemi elementideks on konkreetsed füüsikalised elemendid: elektron-lambid, pooljuhtelemendid jne. Sellest näeme, et skeemide sünteesi puhul tuleb eelnevalt kindlaks määrata ka sünteesiks kasutatavate elementide nomenklatuur. Skeemide sünteesimisel on oluline teisendada vastav loogiline valem võimalikult lihtsale kujule. Kuidas mõista aga "lihtsust", oleneb suurel määral sünteesil kasutatavate elementide nomenklatuurist. Tuleb märkida, et skeemide sünteesi teooria on praegu kujunemisjärgus.

Toome skeemide sünteesi kohta rida näiteid.

Näide 6. Sünteesida skeem valemi

$$P = [(X \approx Y) \,\&\, Z] \vee [(X \approx Z) \,\&\, Y] \qquad (1)$$

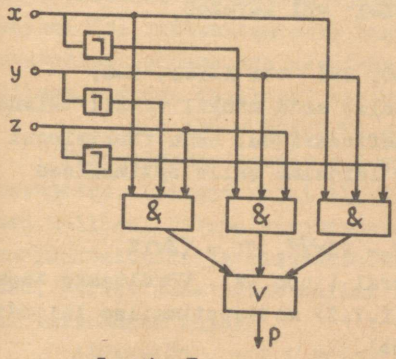
põhjal, kasutades konjunktsiooni, disjunktsiooni ja eituse tehet.

Lihtsustades valemit (1), saame

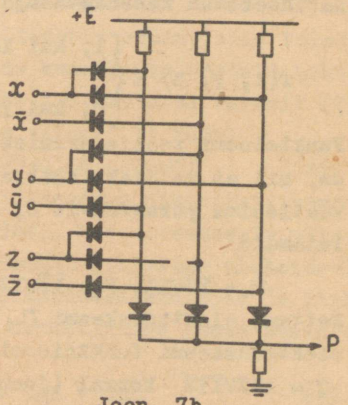
$$P = [(X\bar{Y} \vee \bar{X}Y) \,\&\, Z] \vee [(X\bar{Z} \vee \bar{X}Z) \,\&\, Y] = X\bar{Y}Z \vee \bar{X}YZ \vee XY\bar{Z}. \qquad (2)$$

Minimiseerivate tabelite meetodi abil kontrollides selgub, et disjunktiivset normaalkuju (2) pole võimalik lihtsustada. Vastav skeem loogilistes elementides on toodud joonisel 7a.

Eeldades, et eitused on antud koos sisendsignaalidega, saab vastavat skeemi realiseerida dioodide abil nagu joonisel 7b.




Joonis 7a.




Joon. 7b.

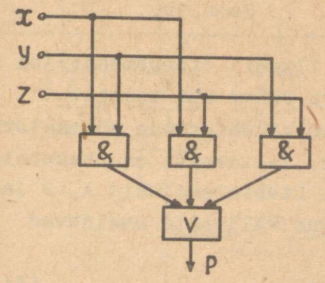
Näide 7. Sünteesida skeem valemi

$$P = XY\bar{Z} \vee X\bar{Y}Z \vee \bar{X}YZ \vee XYZ \qquad (3)$$

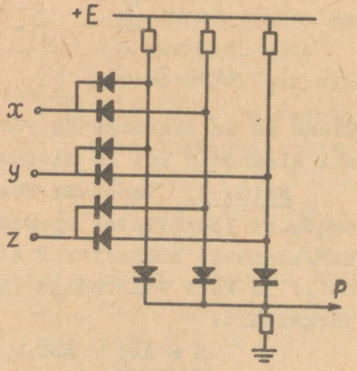
põhjal, kasutades konjunktsiooni, disjunktsiooni ja eituse tehet. Rakendades minimiseerivate tabelite meetodit leiame valemi (3) minimaalse disjunktiivse normaalkuju

$$P = X\,Y \vee X\,Z \vee Y\,Z \qquad (4)$$

Valemile (4) vastav loogiline skeem on toodud joonisel 8a, elektronskeem joonisel 8b.




Joon. 8a.




Joon. 8b.

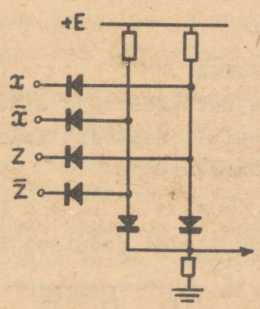
Näide 8. Koostada skeem, mis realiseerib mitte kõikjal defineeritud kahendmuutuja funktsiooni

$$f(X, Y, Z) = \begin{cases} 1, \text{ kui } X=Y=Z=1 \text{ või } X=Y=Z=0, \\ 0, \text{ kui } X=Y=0,\ Z=1 \text{ või } X=Y=1,\ Z=0. \end{cases}$$

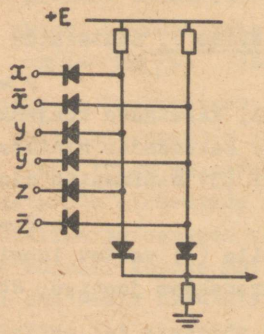
Funktsiooni realiseerimisel tuleb seda mingil viisil täiendada, nii et ta oleks kõikjal defineeritud. Seda funktsiooni vaatlesime paragrahvis 8, kus leidsime selle optimaalsed laiendid

$\mathcal{T}_1 = \bar{X}\bar{Z} \vee XZ$, $\mathcal{T}_2 = \bar{X}\bar{Z} \vee YZ$, $\mathcal{T}_3 = XZ \vee \bar{Y}\bar{Z}$, $\mathcal{T}_4 = \bar{Y}\bar{Z} \vee YZ$.

Esitame elektronskeemi $\mathcal{T}_1$ korral (joon.9a). Võrdluseks toome elektronskeemi funktsiooni f(X,Y,Z) mitteoptimaalse laiendi $\mathcal{T} = XYZ \vee \bar{X}\bar{Y}\bar{Z}$ korral (joon. 9b).




Joon.9a.




Joon.9b.

Skeem 9a on skeemiga 9b võrreldes ilmselt ökonoomsem: tal on nii sisendeid kui ka dioode 2 võrra vähem kui teisel.

Näide 9. Teostame ühekohalise kahendarvude summaatori sünteesi loogilistes operaatorites. Teatavasti on ühekohaline kahendarvude summaator 3 sisendiga (tähistame neid A, B ja C abil) ja kahe väljundiga (S, Ü), kus väljundid avalduvad järgmiselt:

$$S = \bar{A}\bar{B}C \vee \bar{A}B\bar{C} \vee A\bar{B}\bar{C} \vee ABC, \qquad (5)$$

$$Ü = ABC \vee ABC \vee ABC \vee ABC \qquad (6)$$

Valemit (6) saab taandada (vt.näidet 7) lihtsamale kujule

$$Ü = AB \vee AC \vee BC \qquad (7)$$

Valem (5) aga sellisel viisil ei lihtsustu. Et mitme väljundiga skeemide optimaalse sünteesi teooria puudub, siis tuleb valemi (5) lihtsustamiseks kasutada kunstlikke võtteid. Lihtsustamisel püüame ära kasutada juba lihtsustatud valemit (7). Silmas pidades duaalsust kirjutame valemi (5) kujul

$$S = ABC \vee \neg[(A \vee B \vee \bar{C}) \,\&\, (A \vee \bar{B} \vee C) \,\&\, (\bar{A} \vee B \vee C)] \qquad (8)$$

Nurksulgudes asuvat valemit teisendame distributiivsust kasutades. Liikmed kujul $A\bar{A}$, $AB\bar{A}$ jne., mis on samaselt võrdsed nulliga, jätame kirjutamata. Samuti kirjutame ühekordselt konjunktsioonis ja disjunktsioonis korduvad muutujad ja liikmed. Pärast seda saame valemi (8) nurksulgudes asuva osavalemi teisendada kujule

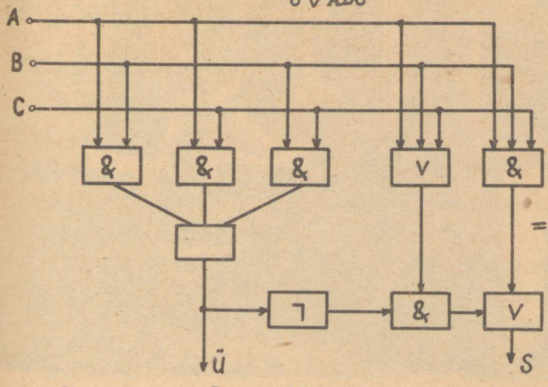
$$AB \vee AC \vee A\bar{B}C \vee ACB \vee BC\bar{A} \vee BC \vee \bar{C}AB \vee \bar{C}B\bar{A} \qquad (9)$$

Valemit (9) saab aga veelgi lihtsustada, sest liikmed $A\bar{B}C$, $ACB$, $BC\bar{A}$ ja $\bar{C}AB$ on seal tegelikult liigsed. Tõepoolest, näiteks elementaarkonjukatsioon $A\bar{B}C$ on liigne valemis (9) esineva liikme $AC$ tõttu: $A\bar{B}C = 1$, kui $A=C=1$ ja $B=0$; sel juhul on ka $AC=1$. Seetõttu $AC \vee A\bar{B}C = AC$. Seega saame valemi (9) esitada kujul

$$A\,B \vee A\,C \vee B\,C \vee \bar{A}\,\bar{B}\,\bar{C}$$

ehk arvestades valemit (7), kujul

$$Ü \vee \bar{A}\bar{B}\bar{C} \qquad (10)$$




Joon. 10.

Asendades valemi (8) nurksulgudes asuva osavalemi valemiga (10) ja kasutades duaalsust, saame

$$S = ABC \vee \neg[Ü \vee \bar{A}\,\bar{B}\,\bar{C}] = A\,B\,C \vee [(A \vee B \vee C) \,\&\, \neg Ü]. \qquad (11)$$

Skeemi sünteesi võib nüüd teostada valemitest (7) ja (11) lähtudes:

Pole raske veenduda, et lihtsustamata valemitest (5) ja (6) lähtudes tuleks vastav skeem märksa keerulisem.

## III p e a t ü k k.

### ÜLEVAADE ARVUTUSMASINA "URAL-1" EHITUSEST.

"Ural-1" tüüpi elektronarvutusmasin, mis oli kodumaises arvutustehnikatööstuses seeriatootmisel kuni 1960.aastani, kuulub väikeste numbriliste universaalsete arvutusmasinate klassi.

Arvutusmasin "Ural-1" on keeruline elektron- ja elektromehhaanilise automaatika seadmete kompleks. Selle kompleksi peamine osa on arvutusmasin (selle sõna otseses tähenduses). Peale selle kuuluvad sellesse seadmete kompleksi veel väljundseadmed (trükkimisseade ja väljundperforaator) ja välised seadmed (klaviatuurseade, kontrolliv-lugev seade ja sisendperforaator). Arvutusmasina "Ural-1" üldvaade on toodud joonisel 1.

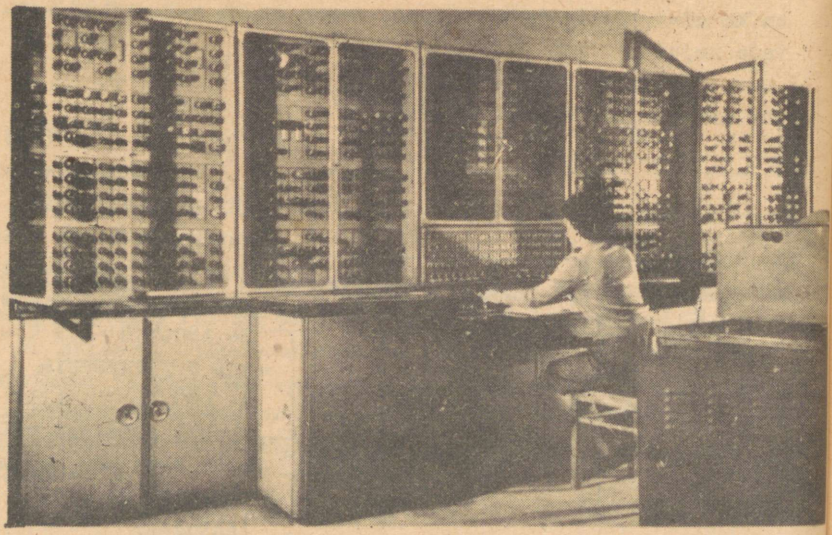
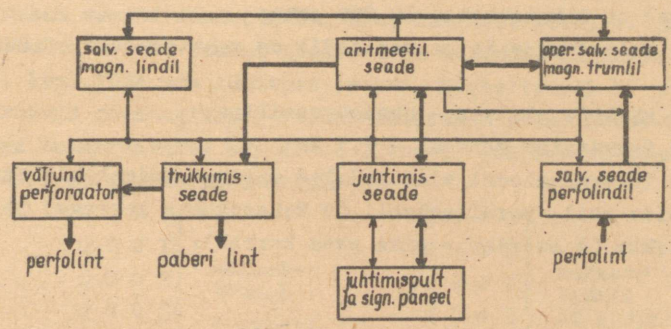


Foto: L. Urm "Ural-1" Joon. 1.

Masina põhimõtteline blokkskeem on toodud joonisel 2.




Joon. 2.

Juhtimisseade on juhtimisimpulsside kaudu (joonisel 2 tähistatud ⟶ abil) ühenduses kõikide teiste seadmetega ja koordineerib nende tööd. Arvulise informatsiooni vahetamine on tähistatud joonisel 2 ⟷ või ⟶ abil. Kui arvulise informatsiooni vahetamine kahe seadme vahel toimub kolmanda seadme vahendusel, siis on see kujutatud joonisel ⟷ või ⟶.

Jooniselt 2 nähtub, et arvutusmasinal "Ural-1" toimub informatsiooni vahetus salvestusseadmete vahel aritmeetilise seadme vahendusel.

Lisaks blokkskeemile toome arvutusmasina "Ural-1" iseloomustamiseks mõned olulisemad andmed:

1) arvutuste kiirus - 100 aritmeetilist või loogilist tehet sekundis;

2) pesa pikkus - nn. pikas pesas on 36 kahendkohta (10--ndsüsteemis vastab sellele 9-kohaline arv), nn. lühikeses pesas 18 kahendkohta;

3) arvude kujutamine masinas ja arvutamine toimub 2-ndsüsteemis, koma on fikseeritud pärast märgikohta;

4) summaator on tsüklilise ülekandega kõrgeimast järgust madalaimasse järku. Summaatoris kasutatav kood on modifitseeritud pöördkood;

5) operatiivne salvestusseade on magnettrumlil - mahu-

tavusega 1024 arvu või 2048 käsku;

6) salvestusseade perfolindil (on ühtlasi ka sisendseadmeks) on mahutavusega 10 000 arvu;

7) salvestusseade magnetlindil on mahutavusega 40 000 arvu;

8) käskude süsteem - üheaadressiline;

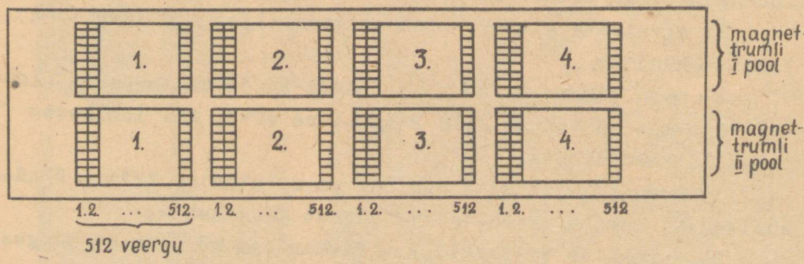
9) tarvitatav võimsus - 7,5 kW;

10) informatsiooni edasiandmine masinas toimub elektriimpulsside abil, paralleelselt (9 kahendkohta korraga) ja järjestikku (4 seerias - pika pesa korral).

## § 11. Salvestusseadmed.

### a) Operatiivne salvestusseade.

Arvutusmasina "Ural-1" operatiivseks salvestusseadmeks on salvestusseade magnettrumlil. Magnettrummel on ferrolakiga kaetud silinder, mille pinnale kantakse informatsioon (sümbolid 1 ja 0) kindla polaarsusega magnetiseeritud piirkondade näol. Arvude kirjutamine magnettrumlile ja lugemine magnettrumlilt toimub ühtede ja samade magnetpeade (lugevate-kirjutavate peade) abil. Informatsioon salvestatakse magnettrumli pinnale järgmiselt (joon. 3):




Joon. 3.

Magnettrumli mõlemas pooles on 4 piirkonda, igas piirkonnas on 512 veergu, igas veerus 9 kahendkohta. Magnettrumli

I poole 1. ja 2. piirkonna esimesed veerud (kokku 18 kahendkohta) moodustavad esimese lühikese pesa, 3. ja 4. piirkonna esimesed veerud moodustavad teise lühikese pesa, 1. ja 2. piirkonna teised veerud moodustavad kolmanda lühikese pesa jne. kuni 1024-nda lühikese pesani. Magnettrumli II pooles paiknevad analoogilisel viisil lühikesed pesad 1025.-2048. Pesad nummerdatakse aga mitte 10-nd-, vaid 8-ndsüsteemis. Numbrid on 4-kohalised arvud:

| | | | |
|---|---|---|---|
| 0 0 0 0 | magnet-trumli I poolel | 2 0 0 0 | magnet-trumli II poolel |
| 0 0 0 1 | | 2 0 0 1 | |
| 0 0 0 2 | | 2 0 0 2 | |
| 0 0 0 3 | | 2 0 0 3 | |
| . . . . | | . . . . | |
| . . . . | | . . . . | |
| 1 7 7 6 | | 3 7 7 6 | |
| 1 7 7 7 | | 3 7 7 7 | |

Arvutusmasinas "Ural-1" võib arvutada peale 18-kohaliste lühikeste pesade ka 36-kohaliste pikkade pesadega. Pika pesa moodustavad kaks järjestikust lühikest pesa, kus esimene pesa on paarisnumbriline. Nii moodustavad esimese pika pesa lühikesed pesad 0000 ja 0001, teise pika pesa pesad 0002 ja 0003 jne. Pikad pesad nummerdatakse samuti 4-kohaliste 8-ndsüsteemi arvudega:

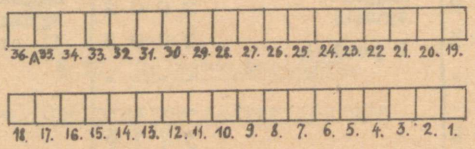
| | | | |
|---|---|---|---|
| 4 0 0 0 | magnet-trumli I poolel | 6 0 0 0 | magnet-trumli II poolel |
| 4 0 0 2 | | 6 0 0 2 | |
| . . . . | | . . . . | |
| 5 7 7 6 | | 7 7 7 6 | |

Pika pesa number saadakse vastava paarisnumbrilise lühikese pesa numbrist selle kõrgeimale järgule 4 liitmise teel. Seega on pikkade pesade numbrid kõik paarisarvud ja mitte väiksemad kui 4000. Need tunnused võimaldavad pesa numbri järgi kohe öelda, kas on tegemist pika või lühikese pesaga - see on aga programmeerimisel väga oluline.

Näiteid. 1) Lühikestest pesadest 2506 ja 2507 moodustatud pika pesa number on 6506, 2) lühikestest pesadest 3023 ja 3024

ei saa moodustada pikka pesa, sest esimene lühike pesa pole paarisarvuline, 3) lühikestest pesadest 1610 ja 1613 ei saa moodustada pikka pesa, sest need lühikesed pesad pole järjestikku (nende vahel on pesad 1611 ja 1612), 4) pikk pesa 7542 moodustatakse lühikestest pesadest 3542 ja 3543, 5) arv 6315 pole ühegi pesa numbriks. Et kõrgem koht on suurem neljast, peaks see olema pika pesa number. See pole aga võimalik, sest pika pesa numbrid peavad olema paarisarvulised.

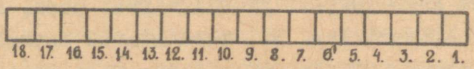
Pikka ja lühikest pesa kujutame skemaatiliselt järgmiselt (vastavalt joonised 4a ja 4b):




Pika pesa I ehk vasak pool (paarisarvuline lühike pesa).

Pika pesa II ehk parem pool (paarituarvuline lühike pesa).

Joon. 4a.




Lühike pesa.

Joon. 4b.

Pesade sisu (s.t. pesades salvestatav arv, käsk vms.) on mingisugune (36- või 18-kohaline) jada sümbolitest 1 ja 0. Niisugust jada, olenemata sellest, kas see kujutab arvu, käsku vms., nimetatakse masina sõnaks. Et masina sõnu kompaktsemalt esitada, on otstarbekohane kodeerida need 8-ndsüsteemi numbrite abil. See kodeerimismoodus on ette nähtud juba masina konstruktsioonis: vastavalt sellele on ehitatud sisendseadmed, trükkimisseade ja klaviatuur juhtimispuldil.[7]

Aritmeetiliste tehete sooritamisel tõlgendab masin tehtest osavõtvate pesade sisusid kui 2-ndsüsteemi arve, kus koma on fikseeritud pärast 36-ndat või 18-ndat kohta, kusjuures 36. või 18. koht on märgikoht. Loogiliste tehete kor-

---

[7] Masinasse saab viia arve ka 10-ndsüsteemis (kümnend-kahendkoodis) ja neid sealt ka 10-ndsüsteemis välja trükkida.

ral manipuleeritakse kõikide kahendkohtadega ühtemoodi ja pesade sisusid vaadeldakse lihtsalt masina sõnadena, s.t. jadadena sümbolitest 1 ja 0.

Toome näiteid masina sõnade ja nende kodeerimise kohta.

Näide 1. Olgu pesas 1540 masina sõna

001 110 111 000 011 010,

mis kujutab käsku (käsud salvestatakse lühikestes pesades). 8-ndsüsteemi numbrite abil kodeerituna tuleb see järgmine:

16 7032.

Näide 2. Olgu pesas 4264 masina sõna

110 101 001 100 011 000 110 111 010 010 011 101,

mis kujutab 2-ndsüsteemi arvu. 8-ndsüsteemi numbrite abil kodeerituna tuleb see masina sõna järgmisel kujul:

651430672235. (1)

Sisuliselt kujutab see 2-ndsüsteemi arvu[8]

- 0,101 010 011 000 110 001 101 110 100 100 111 01,

mis 8-ndsüsteemis tuleb kujul

- 0, 523061564472. (2)

Juhime tähelepanu sellele, et kirjutis (1) on 2-ndsüsteemi arvu kui masina sõna kodeerimise tulemus 8-ndsüsteemi numbrite abil (universaalne masina sõnade kodeerimisviis arvutusmasinal "Ural-1"), kirjutis (2) on aga selle arvu sisuline esitus 8-ndsüsteemis.

Näide 3. Olgu pesas 6052 masina sõna

010 000 111 000 110 010 000 001 100 100 101, 011,

mis kujutagu 10-ndsüsteemi arvu (kümnend-kahendkoodis). 8-ndsüsteemi numbrite abil kodeerituna tuleb see masina sõna järgmine:

207062014453.

Et teada, millist 10-ndsüsteemi arvu see masina sõna tegelikult kujutab, selleks tuleb komast (s.t. 35-ndast kahendkohast) alates rühmitada kahendkohad neljakaupa ja iga nelik dešifreerida järgmiselt:

---

[8] Märgime, et arvud salvestatakse magnettrumlil otsekoodis, s.t. märk "+" kodeeritakse 0, märk "-" 1 abil.

Tabel 1.

| | |
|---|---|
| 0 0 0 0 = 0, | 0 1 0 1 = 5, |
| 0 0 0 1 = 1, | 0 1 1 0 = 6, |
| 0 0 1 0 = 2, | 0 1 1 1 = 7, |
| 0 0 1 1 = 3, | 1 0 0 0 = 8, |
| 0 1 0 0 = 4, | 1 0 0 1 = 9. |

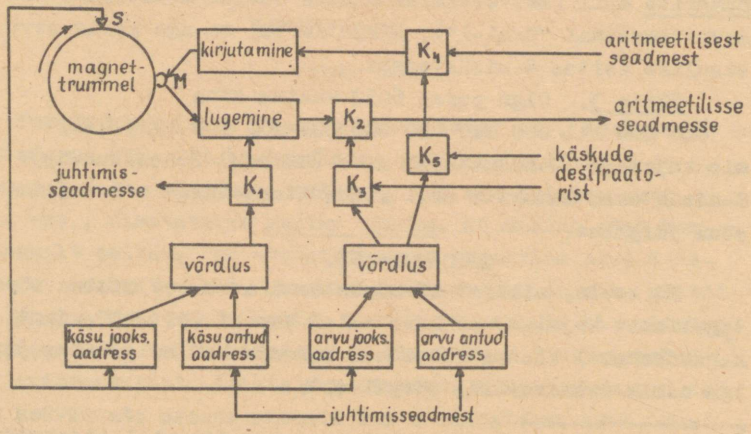
Märgime, et viimase neliku saamiseks võib arvule lisada lõppu numbri "0". Sellest nähtub muide, et 10-ndsüsteemi arvu viimane number on alati paarisnumber.

Antud näites saame järgmise 10-ndsüsteemi arvu

0,871903256.

Tabeliga 1 defineeritud 10-ndsüsteemi arvude kodeerimisviisi (koodi), kus märk "+" kodeeritakse 0, märk "-" 1 abil, nimetataksegi kümnend-kahend koodiks. Kümnend-kahend koodis esitatud arv on ikka 10-ndsüsteemi arv, ainult iga kümnendnumber on kodeeritud kahendnumbrite nelikute abil.

Lõpuks selgitame, kuidas toimub arvude ja käskude lugemine magnettrumlilt ja informatsiooni salvestamine magnettrumlile (vt. joon. 5)




Joon. 5.

Arvutusmasina töötamise ajal pöörleb magnettrummel konstantse nurkkiirusega 100 pööret sekundis. Magnetpesadest M möödub ühe pöörde ajal kogu magnettrumlile salvestatud informatsioon. Kui klapid $K_1$ ja $K_2$ on suletud, ei pääse see informatsioon aga edasi arvutusmasina teistesse seadmetesse. Klapid $K_1$ ja $K_2$ avanevad alles siis, kui magnetpeadest M möödu-vad kindlates pesades olevad käsud või arvud. Need selekteeritakse välja järgmiselt: magnetpea S vahendamisel toimub vastavates blokkides käsu või arvu jooksva aadressi (s.t. pesa numbri) loendamine. Teiselt poolt salvestatakse käsu ja arvu antud aadressi blokkides kindlad, nõutud aadressid (pesa numbrid). Aadresse võrreldakse. Kui käsu jooksev aadress ühtib nõutud aadressiga, avaneb klapp $K_1$ ja väljavalitud käsk viiakse juhtimisseadmesse (käskude registrisse). Arvu aadresside ühtimisel avaneb klapp $K_2$ aga alles siis, kui käskude dešifraatorist on tulnud signaal arvu lugemiseks (s.t. klapp $K_3$ on avatud).

Arvu kirjutamine magnettrumlile toimub analoogiliselt lugemisega, ainult selle erinevusega, et dešifraatorist antakse signaal (klapi $K_3$ kaudu) arvu kirjutamiseks ning et magnetpead M lugemise asemel kirjutavad.

b) Salvestusseade perfolindil.

Salvestusseade perfolindil on arvutusmasina "Ural-1" sisendseadmeks. Üldiselt saab teda kasutada ülesande lahendamise käigus mittemuutuva informatsiooni salvestamiseks (algandmed, alamprogrammid jne.).

Perfolindina kasutatakse 35 mm laiust tumestatud (mittevalgustundlikku) filmilinti. Et perfoauke loetakse fotoelementide abil, siis on oluline, et perfolindil poleks kriimustusi (need võivad anda lisaimpulsse).

Aritmeetilisi ja loogilisi tehteid pole võimalik aga vahetult teostada perfolindil asuvate arvudega. Selleks tuleb nad enne üle kanda magnettrumlile. Arve ja käske võib perfolindilt üle kanda magnettrumlile üldiselt mitte ühekaupa,

vaid kindlates rühmades, mida nimetatakse tsoonideks. Ühes tsoonis ei või koos salvestada 10-ndsüsteemi arve, mis kantakse magnettrumlil pikkadesse pesadesse, ja käske või 6-kohalisi 8-ndsüsteemi arve, mis kantakse lühikestesse pesadesse. Tsoonis võib arve või käske olla suvaline hulk, kuid mitte üle 1000 10-ndsüsteemi arvu või üle 2000 käsu või 6-kohalise 8-ndsüsteemi arvu.[9] Tsoonide maksimaalne arv ühel lindil on 127. Nende numbriteks on 3-kohalised 8-ndsüsteemi arvud 001, 002, 003,..., 177. Maksimaalne lindi pikkus on 250 m. Arvestades vajalikke vahesid tsoonide vahel ja seda, et 1 m pikkusele lindile saab kanda 52 arvu või käsku, saab sellisele lindile kanda umbes 10 000 arvu või käsku. Lindi liikumise kiirus on 1,4 m/sek, arvude lugemise kiirus 4500 ± 10% arvu minutis (u. 75 arvu sekundis). Vajaliku tsooni väljaotsimine 250 m pikkuse lindi korral võtab keskmiselt aega 1,5 minutit ja võib muutuda sekundi murdosast kuni 3 minutini. Perfolint liigub ainult ühes suunas. Et oleks võimalik pöörduda ühe tsooni poole korduvalt, liimitakse lindi algus ja lõpp kokku.

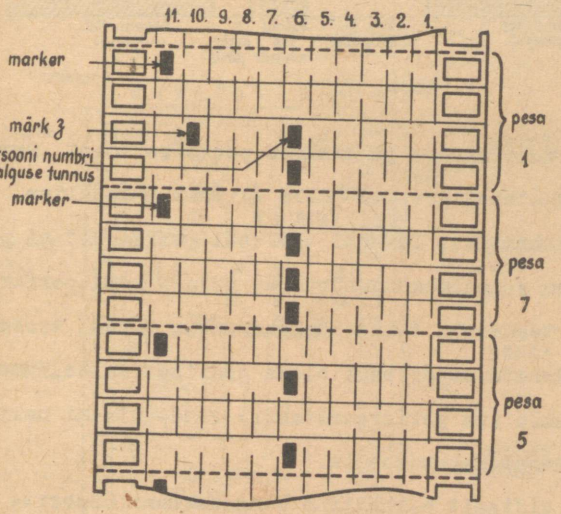
Selgitame, kuidas perforeeritakse lindile tsooni numbrid, käsud ja arvud.

Perfolindi pind on jaotatud üheteistkümneks (mõtteliseks) pikiribaks ehk teekeseks (vt. joonis 6). 11. teekene on ette nähtud spetsiaalse märgi-markeri jaoks, mis jaotab perfolindi 4 perfoaugu laiusteks piirkondadeks. Et see piirkond on vajalik ühe arvu või käsu perforeerimiseks, siis nimetatakse seda sagedasti ka pesaks. 10. teekene on ette nähtud märgi "-" ja märgi "/" (kasutatakse tsooni numbri korral) perforeerimiseks. Teisi teekesi kasutatakse arvude või käskude kindlate numbrikohtade perforeerimiseks.

---

[9] Erandi moodustab tsoon 002, kus võib olla ainult 7 käsku.

Näide tsooni numbri perforeerimise kohta



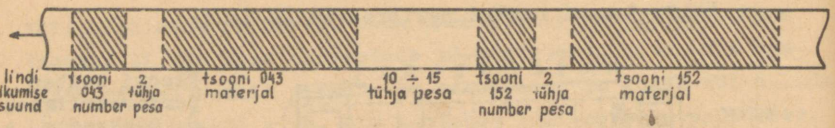

Tsooni number perforeeritakse 6. teekesele, s.o. perfolindi keskele. Tsooni numbri perforeerimiseks on vaja 3 järjestikust pesa, igasse perforeeritakse üks numbrikoht. Antud näites on esimeses pesas (peale märgi ʒ ja tsooni numbri alguse tunnuse) perforeeritud 1, teises pesas 7 ja kolmandas 5. Seega on tsooni number 175.

Joon. 6.

Pärast tsooni numbrit perforeeritakse vastava tsooni materjal (arvud, käsud vms.). Tsooni numbri ja vastava tsooni materjali vahele tuleb jätta perfolindil 2 tühja pesa.

Pärast tsooni materjali lõppu (enne uue tsooni numbrit) on soovitav jätta perfolindil 10 ÷ 15 tühja pesa. Sel korral võtab järgmise tsooni otsimine kõige vähem aega. Asi seisneb selles, et pärast eelmise tsooni materjali sisseviimist liigub lint seadme inertsi tõttu veel veidi edasi. Tühjade pesade puudumise korral võib uue tsooni numbri algus lugemisseadmest mööduda. Selle numbri leidmiseks tuleks nüüd terve perfolint seadmes uuesti läbi kerida. Märgime veel, et tsoonid võivad perfolindil paikneda mistahes järjekorras.

Tsooni numbrite ja muu informatsiooni paigutus perfolindil võib seega olla näiteks järgmine:

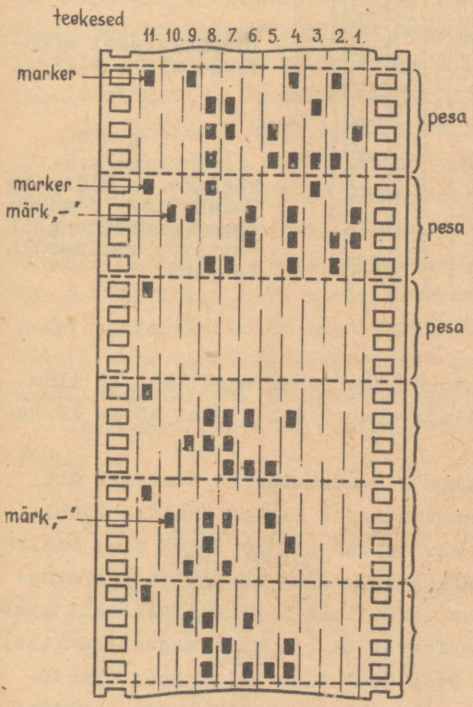



Joon. 7.

Selgitame nüüd arvude ja käskude perforeerimist. Perforeeritav 10-ndsüsteemi arv võib olla ülimalt 9-kohaline. Märk "-" perforeeritakse 10-ndal teekesel (märki "+" ei perforeerita), arvu esimene kümnendnumber pärast koma perforeeritakse 9-ndal teekesel, teine kümnendnumber 8-ndal teekesel jne. Viimane kümnendnumber peab olema paarisarvuline. Kümnendnumbrid esitatakse (ja perforeeritakse) 2-ndsüsteemi nelikute abil nagu kümnendkahendkoodis.

Käsk (või üldiselt 6-kohaline 8-ndsüsteemi numbrite abil esitatud masina sõna) perforeeritakse 9. ÷ 4. teekesele, kusjuures iga 8-ndsüsteemi number perforeeritakse kahendsüsteemi kolmikuna omaette teekesele. Soovi korral võib kasutada ka märki "-", mis perforeeritakse 10-ndal teekesel.

Vastavad perforatsiooninäited on toodud joonisel 8. Joonisel 8 pole kinni peetud nõudest, et 10-ndsüsteemi arvud ja 8-ndsüsteemi arvud ning käsud peavad paiknema eri tsoonides.




pesas on perforeeritud 10-ndsüsteemi arv
+ 0,876039592

pesas on 10-ndsüsteemi arv
- 0,491607836

pesas on perforeeritud arv +0 (tühi pesa)

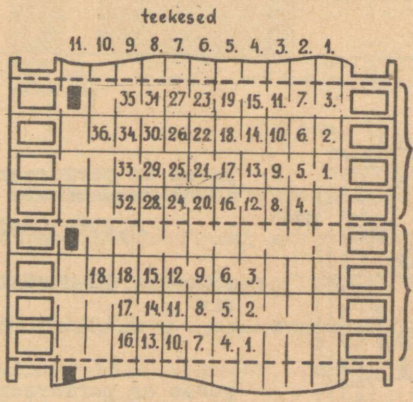
pesas on perforeeritud käsk
26 7514

pesas on perforeeritud käsk
- 16 5042

pesas on perforeeritud 6-kohaline masina sõna
472513

Joon. 8.

Toodud näidetest selgub, et perfolindi igale augukesele vastab kindel kahendkoht pikas või lühikeses pesas. Arvestades pesa kahendkohtade numeratsiooni (vt. joon. 4), võib selle vastavuse esitada järgmiselt:




Joon. 9.

Jooniselt 9 nähtub, et käskude puhul saab läbi ka ilma märgita "-", kasutades selleks numbri 4 liitmist kõige kõrgemale numbrikohale.

Tähendame, et 12-kohaliste 8-ndsüsteemis numbrite abil esitatud masina sõnade sisseviimiseks magnettrumlile (seal salvestatakse need igaüks ühes pikas pesas) tuleb need masina sõnad jaotada kaheks 6-kohaliseks masina sõnaks ja perforeerida järjestikustes pesades perfolindil. Põhimõtteliselt saab 12-kohalisi masina sõnu perforeerida (ja viia magnettrumlile) ka perfolindi ühte pessa. Seda saab teha väljundperforaatorit kasutades, mis perforeerib pikkade pesade kaupa - vastavus pika pesa kahendkohtade ja perfoaukude vahel on nagu 10-ndsüsteemi arvude puhul (vt. joon. 9). Sellist perforeerimist ei võimalda aga sisendperforaator.

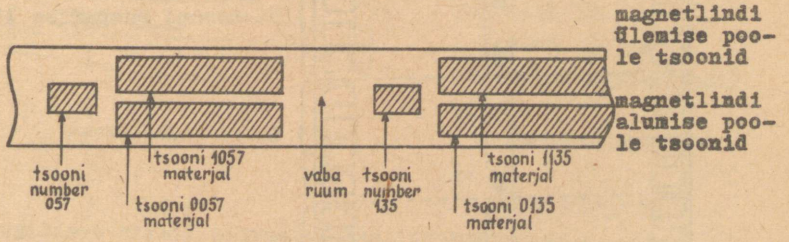
c) Salvestusseade magnetlindil.

Salvestusseade magnetlindil on ette nähtud arvude (eriti vahetulemuste), aga ka käskude salvestamiseks. Magnetlindina kasutatakse standardset 35 mm laiusega magnetlinti. Erinevalt perfolindist võib magnetlindile ülesande lahenda-

mise käigus kanda uusi arve ja neid ka kustutada, nagu see on magnettrumligagi. Erinevus magnettrumlist seisneb aga selles, et magnetlindil asuvate arvudega pole võimalik vahetult tehteid teostada, vaid nad tuleb enne viia magnettrumlile. Selles suhtes sarnaneb magnetlint perfolindile. Magnetlindilt saab informatsiooni kanda magnettrumlile aga jälle ainult rühmiti - tsoonide kaupa.

Magnetlindil võib olla üldse 254 tsooni. Tsoonid nummerdatakse programmis 4-kohaliste 8-ndsüsteemi numbritega 0001 ÷ 0177 (magnetlindi alumise poole tsoonid) ja 1001 ÷ 1177 (magnetlindi ülemise poole tsoonid). Magnetlindile perforeeritakse nad aga 3-kohaliste 8-ndsüsteemi numbritena 001 ÷ 177. Kuigi kahe erineva tsooni (näit. 0057 ja 1057) jaoks perforeeritakse magnetlindile ainult üks ühine number (057), pole tsooniде segiminekut karta, sest need tsoonid asuvad magnetlindi erinevatel pooltel.

Informatsiooni salvestamist magnetlindil selgitab joonis 10.




Joon. 10.

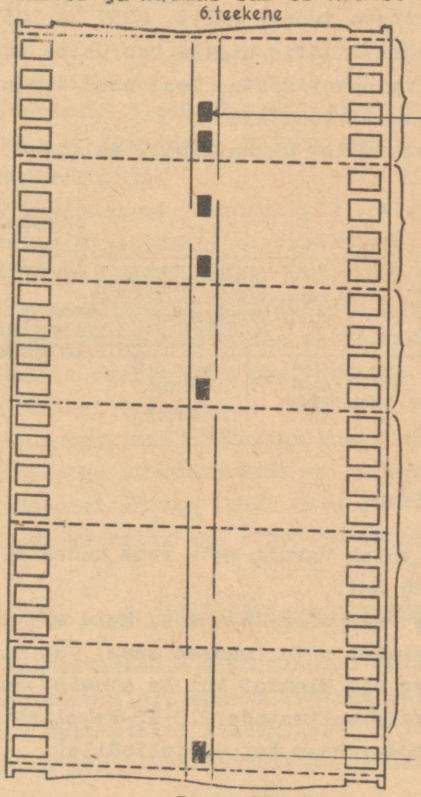
Kahe tsooni vahele tuleb varuda vaba ruum umbes 60 perfoaugu ulatuses (u. 28 cm).

Arvude hulk tsoonis võib olla suvaline, kuid mitte üle 1000 12-kohalise 8-ndsüsteemi arvu (masina sõna) või mitte üle 2000 käsu. Arvestades nii ülemist kui ka alumist poolt, saab magnetlindi 1 meetrile salvestada 210 I2-kohalist 8-ndsüsteemi arvu (4 korda rohkem kui perfolindile).

Maksimaalne lindi pikkus on 250 m. Arvestades vajalikke vahesid tsoonide vahel, saab sellisele lindile salvestada umbes 40 000 arvu või 80 000 käsku.

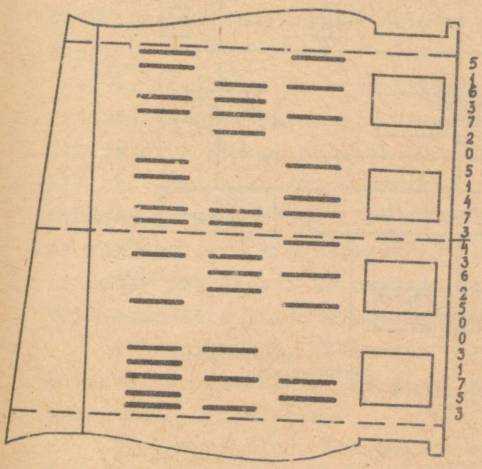
Magnetlindile kirjutamise või magnetlindilt lugemise kiirus on 4500 ± 10% arvu minutis. Lindi liikumise kiirus on umbes 0,7 m/sek. Vajaliku tsooni otsimiseks kulub 250 m pikkuse lindi korral keskmiselt 3 minutit, maksimaalselt aga kuni 6 minutit. Magnetlint liigub ainult ühes suunas. Et oleks võimalik korduvalt pöörduda ühe tsooni poole, liimitakse lindi algus ja lõpp kokku.

Selgitame, kuidas tuleb magnetlindile perforeerida tsooni number ja kuidas salvestatakse informatsioon.




Joon. 11.

Magnetlindi äärtel on samasugused perfoaugud nagu perfolindil. Erinevalt perfolindist pole siin markereid. Tsooni numbrid (joon. 10) perforeeritakse magnetlindi keskele (6. teekesele). Tsooni numbri perforeerimine magnetlindil erineb mõnevõrra tsooni numbri perforeerimisest perfolindil (puudub tunnus $\zeta$, esineb tunnus $\lambda$-2). Magnetlindi tsoonides salvestatakse (resp.loetakse) informatsioon pikkade pesade, s.t. 12-kohaliste 8-ndsüsteemi arvude (masina sõnade) kaupa. Üks 12-kohaline 8-ndsüsteemi arv salvestatakse magnetlindi ühel pool 2 perfoaugu laiuses piirkonnas, nagu näidatud joonisel 12.




Joon. 12.

Antud näites on kujutatud magnetlindil arvud

516372051473

ja

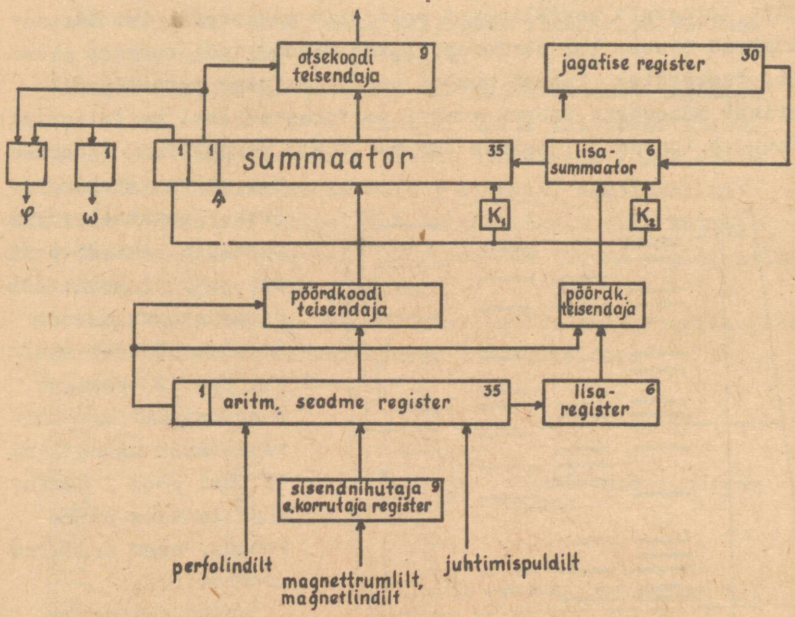
436250031753.

Analoogilisel viisil salvestatakse informatsioon ka magnetlindi teise poole tsoonides.

## § 12. Aritmeetiline seade.

Kõik aritmeetilised ja loogilised tehted sooritatakse arvutusmasina "Ural-1" spetsiaalses, nn. aritmeetilises seadmes. Tehete teostamisest võtavad osa peaaegu kõik aritmeetilise seadme põhilised blokid, kusjuures nad võivad funktsioneerida (vastavalt käsu iseloomule) mitmel eri viisil.

Aritmeetilise seadme põhilised blokid ja nende seosed on kujutatud joonisel 1.




Joon. 1.

1) Summaatoril on 37 kahendkohta, neist 2 märgikohta. Arv kujutatakse summaatoris modifitseeritud pöördkoodis;
2) lisasummaatoril (summaatori lisakohad) on 6 kohta;
3) aritmeetilise seadme registril on 36 kohta, neist 1 märgikoht. Arv kujutatakse aritmeetilise seadme registris otsekoodis;
4) lisaregistril (aritmeetilise seadme registri lisakohad) on 6 kohta;
5) sisendnihutajal (korrutaja registril) on 9 kohta;
6) jagatise registril on 30 kohta;
7) ületäitumise signaali $\varphi$ väljatöötamise blokk;
8) juhtimissignaali $\omega$ väljatöötamise blokk. Signaal $\omega$

võimaldab arvutustulemustest sõltuvalt arvutuste käiku suunata. Millistel juhtudel töötatakse välja signaal $\omega$ , esiteme käskude süsteemi käsitlemisel.

Joonisel 1 on märkimata jäänud blokkide tööd juhtivate impulsside kanalid ja klapid, mis asuvad iga kahe bloki vahelises informatsioonikanalis.

Mõni sõna sellest, kuidas toimub aritmeetilise seadme blokkides aritmeetiliste tehete sooritamine (võrdle § 4 ja § 5).

1. Liitmine. Esimene liidetav asub summaatoris (modifitseeritud pöördkoodis). Lisasummaator on tühi, s.t. positiivse arvu korral on seal 000 000, negatiivse arvu korral 111 111. Teine liidetav tuuakse magnettrumlilt 9-liste kahendkohtade rühmade kaupa läbi sisendnihutaja aritmeetilise seadme registrisse (lisaregister jääb tühjaks). Edasi läbib teine liidetav pöördkoodi teisendaja ja liidetakse summaatoris olevale esimesele liidetavale. Liitmisel saame täpse summa.

2. Lahutamine. Vähendatav arv asub summaatoris. Sellele liidetakse vastupidise märgiga lahutatav. Seega taandatakse lahutamine liitmisele.

3. Korrutamine. Korrutatav arv asub enne korrutamist summaatoris. Sealt viiakse see aritmeetilise seadme registrisse, kusjuures summaator tühjendatakse. Korrutaja tuuakse magnettrumlilt 9-liste rühmade kaupa korrutaja registrisse (sisendnihutajasse). Enne korrutamist selgitatakse korrutise märk. Seejärel nihutatakse korrutatav arv 1 koha võrra paremale ja korrutaja 1 koha võrra vasakule. Kui korrutaja kõrgeimal kohal on 1, liidetakse aritmeetilise seadme sisu (korrutise märgiga) summaatori sisule, vastupidisel juhtumisel liidetakse summaatori sisule 0. Pärast üheksat nihutust tuuakse sisendnihutajasse uus 9-line rühm ja jätkatakse osakorrutiste liitmist, kuni kogu korrutaja on operatsioonist osa võtnud. Korrutamisest võtavad osa ka lisasummaator ja lisaregister. Korrutise viga $|\Delta| \leqslant 0{,}5 \cdot 2^{-35}$.

4. Korrutamine koos liitmisega. Erinevalt korrutamisest on siin korrutatav arv R aritmeetilise seadme registris, kus-

juures summaatoris asub mingi (üldiselt nullist erinev) arv S, millele liidetakse osakorrutised. Osakorrutiste moodustamine ja nende liitmine summaatoris oleva arvuga toimub samuti nagu korrutamise puhul. Tulemuseks saame

$$Z = S + RA,$$

kus A on korrutaja. Tehte viga $|\Delta| \leqslant 2^{-35}$.

5. Jagamine. Jagatav arv on summaatoris, jagaja aritmeetilise seadme registris. Jagatise kohtade leidmine toimub selliselt, et summaatoris olevast arvust lahutatakse aritmeetilise seadme registris olev jagaja. Seejärel selgitatakse saadud vahe märk. Ühes jagatava ja jagaja märgiga määrab see jagatise kahendkoha, mis kantakse jagatise registri viimasele kohale. Peale jagatise kahendkoha leidmist nihutatakse summaatoris olev arv (tsükliline ülekanne toimub läbi klapi $K_1$ summaatori 35-ndasse kohta) ühe koha võrra vasakule. Samuti nihutatakse ühe koha võrra vasakule jagatise registris olev arv. Analoogiliselt leitakse jagatise kõik 36 kahendkohta, mis paiknevad esialgu jagatise registris ja lisasummaatoris. Viimane, 36. koht leitakse ümardamise eesmärgil. Seejärel viiakse saadud tulemus (ühes vastava märgiga) summaatorisse. Jagatise viga $|\Delta| \leqslant 0{,}5 \cdot 2^{-35}$.

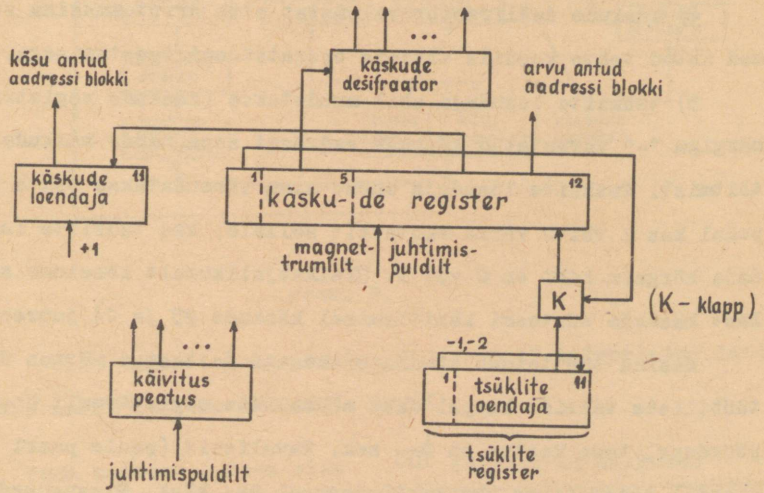
Märgime veel (vt. joon. 1), et arvu toomisel summaatorisse läheb see tingimata läbi aritmeetilise seadme registri. Ületäitumist, mis võib esineda liitmisel, lahutamisel, korrutamisel kooş liitmisega ja jagamisel, saab kindlaks teha nii signaali $\varphi$, aga ka otseselt summaatori kahe kõrgeima nn. märgikoha koha järgi: 01 korral on tegemist positiivse ületäitumisega (tehte tulemus on suurem kui +1), 10 korral negatiivse ületäitumisega (tehte tulemus on väiksem kui -1)

## § 13. Juhtimisseade.

Juhtimisseadme ülesanne seisneb kõikide teiste arvutusmasina seadmete töö juhtimises. Seda teostatakse juhtimisseadmes väljatöötatavate impulsside ja signaalide abil, mis

sünkroniseerivad kõikide masina blokkide tööd ja valmistavad need ette vajalike operatsioonide teostamiseks. Ülesannete automaatne lahendamine on võimalik ainult juhtimisseadme olemasolu tõttu.

Juhtimisseadme põhilistest blokkidest annab ülevaate joonis 1.




Joon. 1.

1) käivituse ja peatuse blokk võimaldavad masinat peatada või tööle rakendada automaatjuhtimise või ühetaktilises režiimis. Blokk on ühenduses kõikide põhiliste seadmetega.

2) käskude loendaja (11 kahendkohta) on ette nähtud järgmisel taktil täitmisele tuleva käsu aadressi väljatöötamiseks. Tavaliselt suureneb käskude loendaja sisu 1 võrra.

Suunamiskäskude korral antakse talle aga täiesti uus sisu.

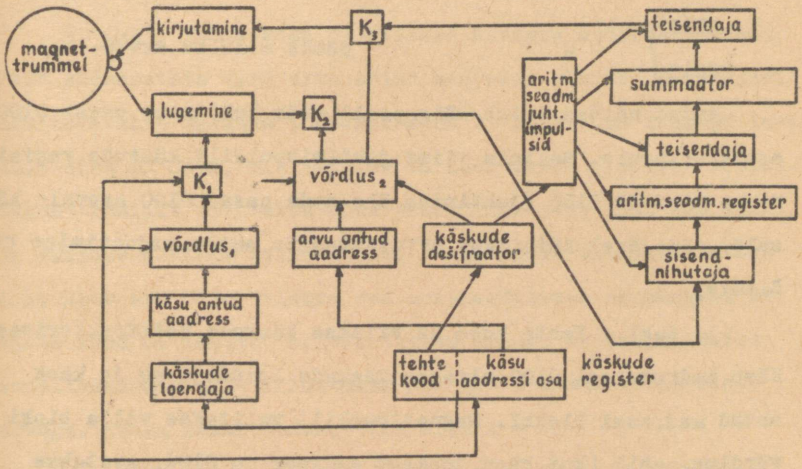
3) käskude register (18 kahendkohta - nendest 1 koht märgi "-", 5 kohta tehte koodi ja 12 kohta tehtest osavõtva arvu aadressi salvestamiseks) on ette nähtud täitmisele tuleva käsu väljatöötamiseks ja käskude loendaja sisu korrigeerimiseks. Andmed täitmisele tuleva käsu kohta saadetakse mitmetesse blokkidesse, nagu näeme jooniselt 1.

4) käskude dešifraator valmistab ette arvutusmasina seadmed antud tehte koodile vastava operatsiooni teostamisega.

5) tsüklite loendaja abil muudetakse (käskude registris) märgiga "-" varustatud käskude aadressi enne nende käskude täitmist. Tsüklite loendaja enese sisu vähendatakse tehte 24 puhul kas 1 või 2 võrra vastavalt sellele, kas tsüklite loendaja kõrgeim koht on 0 või 1. Üksikasjalikumalt kõneleme sellest käskude süsteemi käsitlemisel käskude 25 ja 24 juures.

Masina töö toimub kindla pikkusega ja teatud mõttes ühetüübiliste taktide kaupa. Takt määratakse magnettrumli ühe pöördega, tema kestus on $\frac{1}{100}$ sek. Tavaliselt (peale paari erandi) teostatakse ühe takti jooksul üks käsk. Missugused käsud masin peab täitma, see on antud ülesande lahendamise programmis, mis salvestatakse koos algandmetega magnettrumlil.

Selgitame nüüd näite ja joonise 2 varal, kuidas toimub arvutusmasina töö ülesande lahendamisel. Märgime, et koondskeemis joonisel 2 on esitatud arvutusmasina struktuur vähem detailselt kui paragrahvides 11-12.




Joon. 2.

Olgu magnettrumli pesades 0100 - 0113 salvestatud järgmine informatsioon (käsud ja arvud):

| Pesa aadress | Pesa sisu | Selgitavad märkused |
|---|---|---|
| 0100 | 02 4106 | pesas 0100 on käsk: pesas 4106 asuv arv tuua summaatorisse; |
| 0101 | 01 4110 | pesas 4110 asuv arv liita summaatoris oleva arvuga; |
| 0102 | 06 4112 | pesas 4112 asuv arv korrutada summaatoris oleva arvuga; |
| 0103 | 16 4106 | summaatoris asuv arv viia pessa 4106; |
| 0104 | 32 0000 | summaatori sisu trükkida; |
| 0105 | 37 0000 | peatus; |
| 0106<br>0107 | A | pesas 4106 on arv A; |

0110, 0111 — B — pesas 4110 on arv B;

0112, 0113 — C — pesas 4112 on arv C.

Antud näites tuleb kõigepealt anda juhtimine pesas 0100 asuvale käsule. Selleks viime juhtimispuldilt käskude registrisse käsu 22 0100 (juhtimine üle anda pesas 0100 asuvale käsule). Seejärel käivitame arvutusmasina automaatjuhtimise režiimis.

1. takt. Tehte kood 22 viiakse käskude dešifraatorisse. Käsu aadressiosa 0100 viiakse käskude loendajasse ja käsk antud aadressi blokki. Magnettrumlilt valitakse välja bloki võrdlus$_1$ abil (kui käsu jooksev aadress on 0100, avatakse klapp $K_1$) pesas 0100 asuv käsk (s.t. 02 4106) ja viiakse see käskude registrisse.

2. takt. Tehte kood 02 viiakse käskude dešifraatorisse, käsu aadressiosa 4106 arvu antud aadressi blokki. Samaaegselt suurendatakse käskude loendaja sisu 1 võrra. Uus käskude loendaja sisu on seega 0101, mis viiakse ka käsu antud aadressi blokki. Magnettrumlilt valitakse välja bloki võrdlus$_1$ abil pesas 0101 asuv käsk (01 4110) ja viiakse see käskude registrisse. Samaaegselt valitakse bloki võrdlus$_2$ abil magnettrumlilt ka pesas 4106 asuv arv (koodi 02 korral lubat käskude dešifraator arvu lugeda, seega avatakse klapp $K_2$) ja viiakse see läbi sisendnihutaja ja registri summaatorisse. Märgime, et tehte kood 02 saadeti käskude dešifraatorist aritmeetilise seadme juhtimisimpulsside blokki, mis lülitas

aritmeetilise seadme töösse selliselt, et arv A toodi summaatorisse.

3. takt. Tehte kood 01 viiakse käskude dešifraatorisse, käsu aadressiosa 4110 arvu antud aadressi blokki. Samaaegselt suurendatakse käskude loendaja sisu 1 võrra. Uus käskude loendaja sisu on 0102, mis viiakse ka käsu antud aadressi blokki. Magnettrumlilt valitakse nüüd pesas 0102 asuv käsk (06 4112) ja viiakse käskude registrisse. Samaaegselt valitakse pesas 4110 asuv arv B ja viiakse see aritmeetilisse seadmesse. Et tehte kood 01 lülitab aritmeetilise seadme töösse liitmise sooritamiseks, siis liidetakse see arv B summaatoris oleva arvuga. Seega saame summaatoris arvu A + B.

Tuletame siinkohal meelde, et informatsioon salvestatakse magnettrumlil 9-kohaliste kahendkohtade rühmadesse (veergudesse), kusjuures 512 sellist rühma moodustavad ühe piirkonna. Piirkonnad pole aga paigutatud vahetult üksteise järele, vaid nende vahel asuvad kindla pikkusega tühikud. Nendele tühikutele kuluva pöörlemisaja jooksul sooritataksegi tehted aritmeetilises seadmes. Seega - toimub aritmeetiliste ja loogiliste tehete sooritamine aritmeetilises seadmes praktiliselt paralleelselt tehtest osavõtva arvu toomisega magnettrumlilt.

4. takt. Tehte kood 06 viiakse käskude dešifraatorisse, 4112 arvu antud aadressi blokki. Käskude loendaja sisu suurendatakse 1 võrra. Uus loendaja sisu on seega 0103. Pesas 0103 asuv käsk 16 4106 viiakse käskude registrisse. pesas 4112 asuv arv C viiakse aritmeetilisse seadmesse. Tehte kood

06 lülitab aritmeetilise seadme töösse korrutamiseks. Et summaatoris oli summa A + B, saame pärast korrutamist summaatorisse (A + C) C.

5. takt. Tehte kood 16 viiakse käskude dešifraatorisse, 4106 arvu antud aadressi blokki. Käskude loendaja sisu suurendatakse 1 võrra. Uus loendaja sisu on seega 0104. Pesas 0104 asuv käsk 32 0000 viiakse käskude registrisse. Et käsu 16 korral on kirjutamine lubatud, siis kirjutatakse magnettrumli pessa 4106 summaatoris olev arv (A + B) C. Summaatori sisu seejuures säilib. Uue arvu kirjutamisel pessa 4106 kustub automaatselt selles pesas enne olnud arv A.

6. takt. Tehte kood 32 viiakse käskude dešifraatorisse ja 0000 arvu antud aadressi blokki. Et 32 on summaatori sisu trükkimise käsk, siis pole sellel aadressil aga üldse tähtsust. Käskude loendaja sisu suurendatakse 1 võrra - see on nüüd seega 0105. Pesas 0105 asuv käsk 37 0000 viiakse käskude registrisse. Summaatori sisu (A + B) C trükitakse, kuid sama arv säilib ka summaatoris.

7. takt. Tehte kood 37 viiakse käskude dešifraatorisse ja 0000 arvu antud aadressi blokki. Käskude loendaja sisu suurendatakse 1 võrra - see on nüüd seega 0106. Pesas 0106 asuvat masina sõna tõlgendatakse kui käsku ja tuuakse see käskude registrisse. Tegelikult on selles pesas tulemuse (A + B) C esimene pool. Käsk 37 on peatuse käsk, kusjuures samaaegselt tuuakse summaatorisse arv pesast 0000. Kui oletame, et pesas 0000 on arv +0, siis tühjendatakse summaatorning masina töö peatub.

Esitame alljärgneva tabeli abil kokkuvõtte masina tööst

Tabel 1

| Seade või blokk \ Takt | 1. | 2. | 3. | 4. | 5. | 6. | 7. |
|---|---|---|---|---|---|---|---|
| 1. Käskude dešifraator | 22 | 02 | 01 | 06 | 16 | 32 | 37 |
| 2. Arvu antud aadressi blokk | - | 4106 | 4110 | 4112 | 4106 | 0000 | 0000 |
| 3. Käskude loendaja | 0100 | 0101 | 0102 | 0103 | 0104 | 0105 | 0106 |
| 4. Käskude register | 02 4106 | 01 4110 | 06 4112 | 16 4106 | 32 0000 | 37 0000 | $(A+B)C_{Ip}$ |
| 5. Magnettrumlilt toodav (resp. magn. trumlile viidav arv | - | A | B | C | (A+B)C | - | +0 |
| 6. Summaator | - | A | A+B | (A+B)C | (A+B)C | (A+B)C | +0 |

## § 14. Signalisatsioonipaneel ja juhtimispult.

Signalisatsioonipaneel võimaldab operaatoril visuaalselt jälgida ja kontrollida arvutusmasina tööd (teataval määral ka siis, kui masin töötab automaatjuhtimisrežiimis). Juhtimispuldi kaudu on operaatoril võimalik masina tööd juhtida - anda masinale soovitud tööreżiimi (automaatjuhtimise režiim, ühetaktiline režiim, programmide trükkimise režiim vms.), katkestada või muuta ülesande lahendamise käiku.

Vaatleme kõigepealt signalisatsioonipaneeli (joon.1).

Summaatori ( сумматор ) sisu näitavad signalisatsioonipaneeli esimese rea 37 neoonlambikest, iga lamp tähistab vastavat kahendkohta. Siin ja edaspidi tähistab põlev lambike numbrit 1 või märki "-", mittepõlev lamp numbrit 0 või märki "+". Meenutame, et arvud on summaatoris modifitseeritud pöördkoodis.

Aritmeetilise seadme registri ( регистр арифметического устройства ) sisu näitavad signalisatsioonipaneeli teise rea 36 lambikest. Aritmeetilise seadme registris on arvud otsekoodis. Edaspidi jätame märkuse "arvud on otsekoodis" tegemata.

Kontrollregistri ( контрольный регистр ) sisu näitavad signalisatsioonipaneeli kolmanda rea 36 lambikest. Kontrollregistri abil on võimalik kontrollida magnettruli mistahes pesa sisu (ka masina töö ajal), valides vastavatel tumbleritel[9] juhtimispuldil soovitud pesa aadressi.

[9] Tumbler - kahepositsiooniline lüliti.

Lambike $\varphi$ näitab ületäitumist. Kui vastav lambike põleb (s.t. $\varphi = 1$), esineb ületäitumine. Mittepõleva lambikese $\varphi$ korral (s.t. $\varphi = 0$) pole ületäitumist.

Lambike $\omega$ näitab vastava registri sisu, mille abil on võimalik teostada arvutuskäigu harunemist olenevalt arvutustulemustest. Aritmeetiliste tehete puhul on näiteks tehte positiivse tulemuse korral $\omega = 0$, negatiivse tulemuse korral $\omega = 1$ (lambike põleb).

Operatsiooni number ( номер операции ) 5 neoonlambikest, näitab käskude dešifraatori sisu, s.t. eelmisel taktil teostatud tehte koodi.

Käskude loendaja ( счетчик команд ) 11 lambikest, näitab pesa numbri, milles asub teostamisele tulev käsk.

Käskude register ( регистр команд ) 18 lambikest, näitab teostamisele tulevat käsku.

Edasi on toodud üldpealkirja "Töörežiimid" ( режим работы) all 8 signaallambikest: kirjutamine või lugemine magnetlindilt ( запись, воспроизведение с НМЛ ), lugemine perfolindilt ( считывание с НПЛ ), tulemuste trükkimine (печать результатов ), magnettrumli soendus (подогрев МБ ), programmi trükkimise I režiim (I печать программы ), programmi trükkimise II režiim (II печать программы ), käsu korduv valimine (многократная выдача ), peatus ( остановка), mis vastava töörežiimi korral põlevad. Vastavad töörežiimid antakse masinale juhtimispuldilt.

Pealkirja "Peatuse põhjused" (причины остановки ) all on 3 signaallambikest: toitevoolu ebakorrapärasus (неисправ-

ность питания ), pöördumine magnetlindi poole ( обращение к НМЛ ), pöördumine perfolindi poole ( обращение к НПЛ ). Viimased 2 lambikest põlevad vastavate salvestusseadmete poole pöördumise korral.

Pealkirja "Mitte välja lülitatud" ( не выключено ) all on 5 signaallambikest: profülaktiline kontroll ( профконтроль ), käskude registri tühjendamise blokeerimine ( блокировка сброса РГ команд ), magnettrumlile kirjutamise blokeerimine ( блокировка записи НМБ ), magnetlindile kirjutamise blokeerimine ( блокировка записи НМЛ ), $\varphi$ blokeerimine ( блокировка $\varphi$ ), mis näitavad vastavate tumblerite asendit profülaktilise kontrolli paneelil ja juhtimispuldil (kui tumbler on sisse lülitatud, põleb vastav signaallamp). Nende signaallambikeste põlemine tähistab kõrvalekaldumist arvutusmasina tavalisest tööreziimist ülesande lahendamisel. Mõningatel juhtudel on niisugused kõrvalekaldumised lubatavad. Näiteks, kui ülesande lahendamisel pole ette nähtud magnetlindile kirjutamist, võib põleda signaallamp "magnetlindile kirjutamise blokeerimine". Signaallamp "$\varphi$ blokeerimine" võib põleda siis, kui ülesande lahendusprogramm on koostatud liikuva koma meetodi alusel.

Vaatleme nüüd juhtimispulti (joon. 2). Juhtimispuldil näeme tumblereid ja nuppe. Tumblerid võivad olla püsivalt kas välja lülitatud - suunaga allapoole - või sisse lülitatud - suunaga ülespoole. Instruktsioonide andmine nuppude kaudu toimub nupule vajutamisega, mis annab masinasse ühekordse korralduse.

Juhtimispuldi keskosas asub klaviatuur käskude viimiseks

käskude registrisse (klaviatuuri vasakpoolne osa: nupud "Вз" ja "И" ning 6 nuppude veergu) ja arvude viimiseks summaatorisse ning ühtlasi ka aritmeetilise seadme registrisse (klaviatuuri parempoolne osa: nupud "+", "-", 12 nuppude veergu ja parempoolsed nupud "Вз" ning "И"). Käske ja arve vaadeldakse seejuures 6- või 12-kohaliste 8-ndsüsteemi numbrite abil kodeeritud masina sõnadena. Igale numbrikohale vastab kindel veerg klaviatuuril. Märki "-" võib asendada 4 liitmisega kõrgeimale kohale. Vasakpoolsele nupule "И" (исполнение) vajutamine viib klaviatuurile kantud käsu käskude registrisse. Parempoolsele nupule "И" vajutamine aga klaviatuurile kantud arvu summaatorisse ning aritmeetilise seadme registrisse. Vastavatele nuppudele "Вз" vajutamine taastab klaviatuuri nuppude normaalseisu.

Pealkirja "Magnettrummel" (магнитный барабан) all on tumbler "kirjutamise blokeerimine" (блокировка записи). Kui see tumbler on sisse lülitatud, siis kirjutamist magnettrummelile ei toimu ja signalisatsioonipaneelil pealkirja "Mitte välja lülitatud" all süttib vastav lambike.

Pealkirja "Magnettrummel" all on nupp "kustutamine" (стирание). Teine samasugune nupp on klaviatuuri paremal pool. Üheaegse vajutamisega nendele kahele nupule kustutatakse kogu magnettrumli sisu. Nuppe kustutamiseks on sellepärast kaks, et vältida magnettrumli sisu juhuslikku kustutamist.

Pealkirja "Magnetlint" (магнитная лента) all on kaks tumblerit: "peatus pöördumise korral" (остановка при обраще-

нии) ja "kirjutamise blokeerimine" (блокировка записи ). Esimese tumbleri sisselülitamisel jääb masin magnetlindi poole pöördumise korral seisma ja signalisatsioonipaneelil süttib vastav lambike (vt. "Peatuse põhjused"). Kui nüüd vastav tumbler välja lülitada, jätkab masin tööd, hakates otsima tsooni numbrit. Teise tumbleri sisselülitamisel ei toimu magnetlindile kirjutamist ja signalisatsioonipaneelil pealkirja "Mitte välja lülitatud" all süttib vastav lambike.

Pealkirja "Perfolint" (перфолента) all on tumbler "peatus pöördumise korral". See funktsioneerib analoogiliselt vastava tumbleriga magnetlindi jaoks.

Edasi on juhtimispuldil rida nuppe. Selgitame, mis toimub vastavale nupule vajutamisel.

Algkäivitus (начальный пуск ) viib perfolindilt tsooni nr. 002 materjali (seal võib olla maksimaalselt 7 käsku) magnettrumlile alates pesast 0002. Seejärel antakse automaatselt juhtimine pesas 0002 asuvale käsule ja masin alustab tööd automaatjuhtimise režiimis. Enne algkäivituse kasutamist peavad "operatsiooni number", käskude loendaja ja käskude register olema tühjad.

Algtühjendus (начальный сброс ) tühjendab käskude loendaja ja tsüklite registri sisu.

Peatus ( остановка). Programmis ettenähtud käskude täitmine peatatakse.

Käivitus (пуск). Masin käivitatakse automaatjuhtimise režiimis. Esimesena täidetakse käskude registris asuv käsk. Edasi suurendatakse käskude loendaja sisu 1 võrra ja saadakse

pesa number (aadress), millest võetakse järgmine käsk. Kui esimene käsk on suunamiskäsk, viib see käskude loendajasse täiesti uue pesa numbri, millest võetakse järgmine täitmisele tulev käsk. Nuppudele "peatus" ja "käivitus" vajutamine ei kahjusta programmi täitmist, kui see ei toimu rühmaoperatsioonide ajal.

Ühetaktiline reziim (однотактный режим). Sellele nupule vajutades teostab masin ainult ühe takti (üldiselt öeldes - ühe käsu). Ainult jagamine ja normaliseerimine nõuavad vastavalt 4 ja 2 takti. Neid tehteid saab teostada ka ühe taktiga, kui asetada tumbler "jagamise režiimid" profülaktilise kontrolli paneelil vastavasse asendisse. Muide, normaliseerimine ei tarvitse ühetaktilises režiimis 2 takti korral anda õiget tulemust. Rühmaoperatsioone ühetaktilises režiimis teostada ei saa.

Takti vahelejätmine (пропуск такта). Seda nuppu kasutatakse ühe käsu (takti) vahelejätmiseks ühetaktilises režiimis. Seisva masina puhul kasutatakse käskude registri tühjendamiseks.

Tumbleri "Korduv valimine" (многократная выдача) sisselülitamisel säilitatakse käskude loendaja konstantne aadress, millega valitakse käskude registrisse kogu aeg sama käsk. Seega võib korduvalt teostada üht ja sama käsku.

Tumbleri "Käskude registri tühjendamise blokeerimine" (блокировка сброса регистра команд) sisselülitamisel ei toimu käskude registri tühjendamine käivituse või ühetaktilise režiimi korral.

Kahte viimast tumblerit kasutatakse peamiselt masina profülaktilisel kontrollimisel.

Pealkirja "Peatus aadressi järgi" (остановка по адресу) all on 12 tumblerit. Neist kõige vasakpoolsem lülitab aadressi järgi peatamise mehhanismi tegevusse, ülejäänud 11 on aga lühikese pesa aadressi fikseerimiseks, mis toimub aadressi kahendkohtadele vastavate tumblerite sisselülitamise teel. Olles valinud tumbleritel konkreetse aadressi ja lülitanud sisse äärmise vasakpoolse tumbleri, peatub masin enne selles pesas asuva käsu täitmist (see käsk tuuakse käskude registrisse). Pärast "peatuse aadressi järgi" väljalülitamist (s.t. äärmise vasakpoolse tumbleri väljalülitamist) ja vajutust nupule "käivitus" jätkub töö endisel viisil.

Edasi järgnevad 3 tumblerit: "$\varphi$ blokeerimine" (блокировка $\varphi$), "juhtimise üleandmine $\varphi$ järgi"[10] ja "peatus $\varphi$ järgi" (остановка по $\varphi$). Kui tumbler "$\varphi$ blokeerimine" on sisse lülitatud (teise 2 tumbleri asend pole oluline) ja toimub ületäitumine, jätkab masin käskude täitmist programmis ettenähtud järjekorras. Kui tumbler "$\varphi$ blokeerimine" on välja lülitatud ja tumbler "peatus $\varphi$ järgi" on sisse lülitatud (keskmise tumbleri asend pole oluline), jääb ületäitumise korral masin seisma. Kui ülalnimetatud 2 tumblerit on välja lülitatud ja tumbler "juhtimise üleandmine $\varphi$ järgi" sisse lülitatud, antakse ületäitumise korral juhtimine pesas 0000 asuvale käsule. Kui

[10] See tumbler on monteeritud TRÜ arvutusmasinale "Ural-1" arvutuskeskuse inseneride poolt nupu "сброс $\varphi$" ($\varphi$ kustutamine) asemele. Tumblerile on aga jäänud vana pealkiri "сброс $\varphi$", mis muidugi ei vasta tema uuele funktsioonile.

ka viimane tumbler on välja lülitatud, jäetakse ületäitumise korral üks käsk vahele.

Pealkirja "Programmi trükkimine" ( печать программы) all on 2 tumblerit: "I režiim" ja "II režiim". Tumbleri "I režiim" sisselülitamisel trükitakse 8-ndsüsteemi numbrite abil järjestikuste lühikeste pesade sisu ühes nende pesade aadressidega (numbritega). Tumbleri "II režiim" sisselülitamisel trükitakse käsk, selle aadress ja vastav summaatori sisu ülesande lahendamise käigus.

Programmi trükkimine toimub nii: lülitatakse sisse 8-ndsüsteemi trükkimine; lülitatakse sisse tumbler "I režiim" või "II režiim"; antakse juhtimine pesale, mis vahetult eelneb sellele pesale, millest alates soovitakse programmi trükkida ja vajutatakse nupule "käivitus". Programmi trükkimist võib teostada ka ühetaktilises režiimis. Programmi trükkimine lõpetatakse vajutusega nupule "peatus", mille järele vastav trükkimisrežiimi tumbler välja lülitatakse. Programmi trükkimist võib lõpetada ka peatusega aadressi järgi.

Tumbleri "trükkimine 8-ndsüsteemis-10-ndsüsteemis" (печать восьмеричная-десятичная ) asend määrab summaatori sisu trükkimise viisi trükkimise käsu 32 0000 korral. Kui tumbler on seisus "8-ndsüsteem", vaadeldakse ja trükitakse summaatori sisu 12-kohalise 8-ndsüsteemi numbrite abil kodeeritud masina sõnana. Kui tumbler on seisus "10-ndsüsteem", vaadeldakse summaatori sisu kümnend-kahendkoodis oleva arvuna. Tulemus trükitakse 10-ndsüsteemi numbrite abil.

Nupp "summaatori sisu trükkimine" (печать содержимого

сумматора ) võimaldab trükkida summaatori sisu 8-ndsüsteemi numbrite abil kodeeritud masina sõnana. Nupule vajutus (s.t. summaatori sisu trükkimine) ei kahjusta programmi täitmist.

Pealkirja "Juhtimise üleandmine" (передача управления ) all on 7 tumblerit (nummerdatud 1,2,...,7). Need funktsioneerivad koos käskudega vastavalt 23 0001, 23 0002, ..., 23 0007. Nimelt, kui tumbler number $k$ ( $k = 1,2,...,7$) on sisse lülitatud ja masinas tuleb täitmisele käsk 23 $k$, siis jäetakse järgmine käsk vahele. Kõikidel teistel juhtudel täidetakse käsud tavalises järjekorras.

Pealkirja all "Kontrollregistri aadress" ( адрес контрольного регистра) on 12 tumblerit. Nende abil fikseeritakse pesa aadress, mille sisu tahetakse visuaalselt kontrollida kontrollregistris signalisatsioonipaneelil.

Esitame lõpuks mõned sagedamini esinevad ülesanded, mida on vaja täita operaatoril juhtimispuldi taga.

1. Juhtimise üleandmine pesas 1507 asuvale käsule (asugu pesas 1507 näiteks käsk 02 4330). Selleks talitame järgmiselt:

1) viime käskude registri klaviatuurile käsu 22 1507;
2) vajutame vasakpoolsele nupule "И", millega käsk 22 1507 viiakse käskude registrisse,
3) vajutame nupule "ühetaktiline režiim", millega viiakse käskude loendajasse aadress 1507 ja käskude registrisse käsk 02 4330, mis tulebki järgmisel taktil täitmisele.

2. Masina sõna 425736723041 viimine pesasse 5306. Selleks talitame järgmiselt:

1) viime masina sõna 425736723041 summaatori klaviatuurile,

2) vajutame parempoolsele nupule "И", millega see masina sõna viiakse summaatorisse;

3) viime käskude registri klaviatuurile käsu 16 5306,

4) vajutame vasakpoolsele nupule "И", millega see käsk viiakse käskude registrisse;

5) vajutame nupule "ühetaktiline režiim", millega täidetakse käskude registris olev käsk (s.t. summaatori sisu viiakse pessa 5306).

3. 10-ndsüsteemi arvu -0,578987426 viimine summaatorisse. Selleks talitame järgmiselt:

1) kirjutame arvu -0,578987426 kümnend-kahend koodis

1 0101.0111.1000.1001.1000.0111.0100.0010.011 .
5 7 8 9 8 7 4 2 6

2) rühmitame kahendkohad 3-kaupa ja asendame iga kolmiku vastava 8-ndsüsteemi numbriga:

101.010.111.100.010.011.000.011.101.000.010.011.
5 2 7 4 2 3 0 3 5 0 2 3

Seega tuleb summaatorisse viia masina sõna 527423035023. Kuidas seda teha, selgub punktist 2.

4. 10-ndsüsteemi arvu viimisel mingisse magnettrumli pessa tuleb kasutada punktides 3 ja 2 näidatud võtteid.

5. Pesa 4662 sisu kontrollimine.

Fikseerime aadressi 4662 tumbleritel "kontrollregistri aadress" ja loeme kontrollregistrilt signalisatsioonipanee-

lilt vastava pesa sisu.

Kui kontrollregister pole aga korras, siis võib kasutada teist moodust:

1) viime käskude registrisse käsu 02 4662 (pesa 4662 sisu toomine summaatorisse),
2) vajutame nupule "ühetaktiline režiim", millega pesa 4662 sisu tuuakse summaatorisse ja ühtlasi ka aritmeetilise seadme registrisse. Signalisatsioonipaneelilt (kas summaatori või aritmeetilise seadme neoonlampidelt) loeme nüüd pesa 4662 sisu.

6. Masina ettevalmistamine tsooni nr. 002 sisu toomiseks magnettrumlile.

Kõigepealt tuleb tühjendada "operatsiooni number",käskude loendaja ja käskude register. See teostatakse vajutusega nuppudele "takti vahelejätmine", "ühetaktiline režiim", "algtühjendus" ja "takti vahelejätmine". Seejärel lülitame sisse tumbleri "peatus pöördumise korral" (perfolindi jaoks), asetame perfolindi sisendseadmesse ja vajutame nupule "algkäivitus". Et tumbler "peatus pöördumise korral" on sisse lülitatud, siis lindikerimise mootor ei käivitu. See sünnib aga kohe peale nimetatud tumbleri väljalülitamist.

Kõige lõpuks tähendame, et 10-ndsüsteemi arvude masinasse viimisel juhtimispuldilt võib kasutada järgmist tabelit.

Tabel 2

| I \ II | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 001 | 002 | 003 | 004 | 005 | 006 | 007 | 010 | 011 |
| 1 | 020 | 021 | 022 | 023 | 023 | 025 | 026 | 027 | 030 | 031 |
| 2 | 040 | 041 | 042 | 043 | 044 | 045 | 046 | 047 | 050 | 051 |
| 3 | 060 | 061 | 062 | 063 | 064 | 065 | 066 | 067 | 070 | 071 |
| 4 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 110 | 111 |
| 5 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 130 | 131 |
| 6 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 150 | 151 |
| 7 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 170 | 171 |
| 8 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 210 | 211 |
| 9 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 230 | 231 |

Meetod kümnend-kahend koodis esitatud 10-ndsüsteemi arvude teisendamiseks 8-ndsüsteemi numbrite abil üleskirjutatud masina sõnadeks on järgmine.

Teisendatava 10-ndsüsteemi arvu, näiteks -0,578987426 numbrikohad tuleb rühmitada nelja rühma järgmisel viisil

-0,57'898'742'6.

Esimene märkide rühm teisendatakse nii: märk "-" asendatakse arvuga 400, märk "+" arvuga 000, kahe numbrikoha vasted leitakse tabelist. Vastete liitmine toimub 8-ndsüsteemis. Seega saame

$$\begin{array}{r} -0,57 \\ \hline 400 \\ +127 \\ \hline 527. \end{array}$$

Teise ja kolmanda numbrite rühma teisendamine toimub ühtmoodi. Vastavate kolmikute esimene number asendatakse 8-ndsüsteemi numbrite nelikuga järgmiselt:

0 - 0000 5 - 2400
1 - 0400 6 - 3000

| | | | |
|---|---|---|---|
| 2 | - 1000 | 7 | - 3400 |
| 3 | - 1400 | 8 | - 4000 |
| 4 | - 2000 | 9 | - 4400. |

Kolmiku kahe viimase numbrikoha vasted leitakse aga tabelist. Seega saame

| | |
|---|---|
| 898 | 742 |
| 4000 | 3400 |
| 230 | 102 |
| 4230 | 3502. |

Viimase numbri vaste leiame aga selle numbri kahega jagamisel. Seega saame 6 : 2 = 3.

Kui üksikud rühmad on teisendatud, kirjutame saadud tulemused üles üksteise järel

527423035023,

mis ongi soovitud lõpptulemus.

Meetodi põhjendust me siin ei esita. Seda võib leida aga lugeja ise, kui kümnend-kahendkoodis esitatud arvu kahendkohad rühmitada kolmekaupa ja asendada need seejärel 8-ndsüsteemi numbritega.

Toome veel mõned näited.

Näide 1. 10-ndsüsteemi arv +0,723062894 teiseneb

| +0,72' | 306' | 289' | 4 |
|---|---|---|---|
| 000 | 1400 | 1000 | |
| 162 | 006 | 211 | |
| 162 | 1406 | 1211 | 2 |

Seega saame masina sõna 162140612112.

Näide 2. 10-ndsüsteemi arv -0,817308918 teiseneb

| -0,81' | 730' | 891' | 8 |
|---|---|---|---|
| 400 | 3400 | 4000 | |
| 201 | 060 | 221 | |
| 601 | 3460 | 4221 | 4 |

Seega saame masina sõna 601346042214.

Märgime, et leitud masina sõnad ei kujuta nende 10-nd-süsteemi arvudega võrdseid 8-ndsüsteemi arve. Need on ainult nende 10-nd süsteemi arvude kümnend-kahendkoodidele vastavad masina sõnad.

## § 15. Väljundseadmed. Välised seadmed.

Väljundseadmed on ette nähtud materjali (lõpptulemused, vahetulemused jne.) väljatoomiseks arvutusmasina operatiivsest salvestusseadmest - magnettrumlilt.

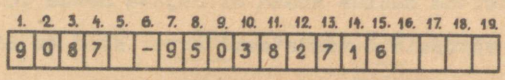
Väljundseadmeid on kaks: trükkimisseade ja väljundperforaator, mis on omavahel seotud. Materjali trükkimine või perforeerimine määratakse trükkimisseadme juhtimispaneelil asuva tumbleri "perforeerimine - trükkimine" (перфорирование - печать) asendiga.

Trükkimisseade võib töötada mitmesuguses režiimis (programmi trükkimise I ja II režiim, summaatori sisu ühekordne trükkimine, trükkimine programmis asuvate käskude 32 0000 põhjal jne.). Seadme töö oleneb ka tumbleri "trükkimine 8-ndsüsteemis - 10-ndsüsteemis" asendist.

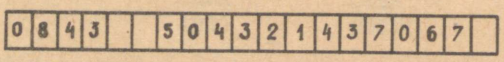
Arvude trükkimiseks on trükkimisseadmes 19 kangikest, igaüks ühe koha trükkimiseks. Esimesed 4 kangikest trükivad (10-ndsüsteemis) trükitava arvu järjekorranumbri. 5.kangike

jääb vahele, 6.kangike trükib arvu märgi (-), 7.-15 kangike trükivad 10-ndsüsteemi arvu numbrikohad. 8-ndsüsteemi trüki korral trükitakse pika pesa sisu 7.-18.kangikese abil. 19. kangike jääb kasutamata.

Kangikeste jaotus arvude trükkimisel on seega järgmine:




10-ndsüsteemi arvu puhul




8-ndsüsteemi trüki puhul

Kangikesi, mida me mingil põhjusel ei soovi kasutada, võib trükkimisel ka välja lülitada. Näiteks, kui meid ei huvita tulemuste loendamine, lülitame välja kangikesed 1 - 4.

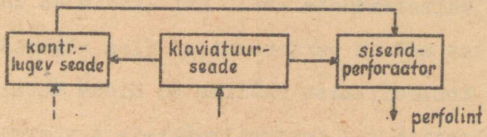
Arvud trükitakse 11 cm laiusele paberlindile, kiirusega 100 ± 10% arvu minutis (1,7 arvu sekundis).

Mõningatel juhtudel (mõne suurema ülesande vahetulemuste, standardsete programmide vms. puhul) on otstarbekohane materjali mitte välja trükkida, vaid välja perforeerida. Kui tumbler "trükkimine 8-ndsüsteemis - 10-ndsüsteemis" lülitada asendisse "10-ndsüsteem" ja tumbler "perforeerimine-trükkimine" trükkimisseadme juhtimispaneelil lülitada asendisse "perforeerimine", siis on võimalik väljundperforaatoriga magnettrumlil asuvat materjali perforeerida pikkade pesade kaupa. Seejuures on pika pesa kahendkohtade ja perfoaukude vastavus samasugune nagu kümnend-kahend koodis esitatud 10-ndsüsteemi arvude puhul. Märgime, et standardsed program-

mid salvestatakse perfolintidel just väljundperforaatori abil perforeeritutena. Põhjuseks on asjaolu, et sel juhtumil perforeeritakse kaks käsku ühte pessa, mistõttu on lindid poole lühemad ja poole lühem on ka nendel lintidel asuva materjali sisseviimise aeg.

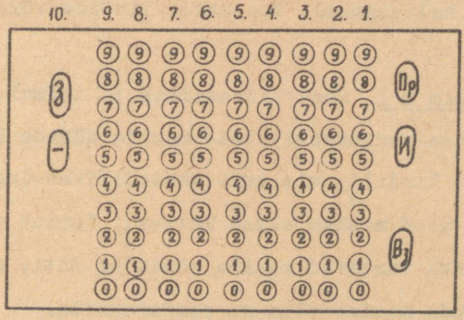
Trükkimisseade ja väljundperforaator on peamasinaga seotud ja temale alluvad seadmed. Välised seadmed: sisendperforaator, klaviatuurseade ja kontrollseade on aga peamasinast sõltumatud. Nende seadmete abil toimubki põhiliselt masinasse sisseviidava informatsiooni perforeerimine.

Väliste seadmete vahekorda iseloomustab joonis 2.




Joon. 2.

Perforeerimisel valitakse klaviatuurseadme klaviatuuril (vt. joon. 3) vajalik käsk või arv. Vajutades nupule "И"




Veergudele vastavad teekesed perfolindil

Joon. 3.

teostatakse selle käsu või arvu perforeerimine lindile. Vajutus nupule "Вз" taastab nuppude algseisu. Nupp "Пр" on seotud kirjutusmasinaga, mis oli mõeldud perforeerimise õigsuse visuaalseks kontrollimiseks perforeerimise ajal. Et selle töö oli aga ebastabiilne, on ta TRÜ arvutuskeskuse klaviatuur seadmel demonteeritud.

Arvude perforeerimise kiirus oleneb perforeerijast. Tavaliselt on see 6 - 10 arvu piires minutis. Peale perforeerimist tuleb tingimata linti kontrollida. Seda võib teha visuaalselt. Seda võib teha aga ka teisiti: lindi perforeerib teiskordselt teine perforeerija ja kahte linti kontrollitakse automaatselt välisseadmete abil. Kui lindid on identsed, siis võib lugeda, et nad on perforeeritud ilma vigadeta.

Lintide kontrollimise režiime on üldse kaks:

Kontrollrežiim A. Kontrollrežiimis A võrreldakse kaht perfolinti, mis on asetatud kontrollivasse-lugevasse seadmesse. Kontrollimise kiirus on 200 ± 10% arvu minutis. Kui kahe lindi vastavate pesade paari puhul nende pesade sisud ei ühti, antakse helisignaal, kusjuures lintide võrdlemine peatatakse.

Kontrollrežiim Б. Kontrollrežiimis Б toimub lindi perforeerimine ühes perforeeritavate masina sõnade kontrollimisega teiselt lindilt, mis asub kontrollivas-lugevas seadmes. Kui vastavad masina sõnad ühtivad, toimub selle perforeerimine. Kui vastavad masina sõnad ei ühti, antakse helisignaal ja seda masina sõna ei perforeerita.

Peale kontrollrežiimide võimaldavad välised seadmed veel linti automaatselt perforeerida (kopeerida) juba olemasoleva perfolindi järgi. Kopeeritav perfolint asetseb seejuures kontrollivas-lugevas seadmes. Uus lint perforeeritakse kiirusega 180 ± 10% arvu minutis.

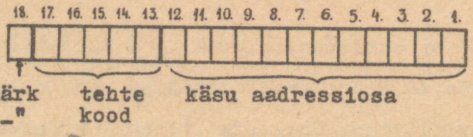
## IV p e a t ü k k.

## PROGRAMMEERIMISE ALUSED.

### § 16. Arvutusmasina "Ural-1" käskude süsteem.

Et koostada käskude jada mingi ülesande lahendamiseks elektronarvutusmasinal ehk selle ülesande lahendamise programmi, peab teadma selle arvutusmasina käskude süsteemi.

Arvutusmasina "Ural-1" käsud on 18-kohalised numbrite 0 ja 1 abil kodeeritud masina sõnad, mis salvestatakse lühikestes pesades. Käsu 18.kahendkoht (vt.joon. 1) on reserveeritud märgi "-" salvestamiseks (märk "-" kodeeritakse numbriga 1), järgmised 5 kahendkohta on ette nähtud tehte koodi, viimased 12 kahendkohta aga käsu aadressiosa[11] jaoks. Enamiku käskude puhul kujutab käsu aadressiosa antud tehtest osavõtva arvu (masina sõna) aadressi.




Joon. 1.

[11] Juhime tähelepanu sellele, et teeme vahet terminite "käsu aadress" ja "käsu aadressiosa" vahel. Esimene tähendab pesa numbrit, kus antud käsk asub, teine aga selle käsu viimasel 12-nel kahendkohal asuvat osa.

Suurema kompaktsuse saavutamiseks kodeeritakse käsud (programmis, perforeerimisel sisendperforaatori abil jne.) aga 8-ndsüsteemi numbrite abil, mis teatavasti on arvutusmasina "Ural-1" masina sõnade universaalseks kodeerimisviisiks.

Näide 1. Olgu pesas 0117 käsk 001 010 011 101 000 110. 8-ndsüsteemi numbrite abil kodeerituna tuleb see käsk kujul 12 3506. Tähendame, et märgi "-" puudumisel märki "+" ei kasutata.

Näide 2. Olgu pesas 2034 käsk 101 110 101 010 111 100. 8-ndsüsteemi numbrite abil kodeerituna tuleb see käsk kujul 56 5274. Kasutades märki "-", saame selle kujul -16 5274.

Käskude süsteemi esitamiseks on vajalik teha veel mõned kokkulepped ja märkused.

1. Teatavasti on matemaatikas (väidete lühikeseks ja täpseks sõnastamiseks) vaja kasutada tähiseid mistahes naturaalarvu, ratsionaalarvu jne. jaoks. Samadel põhjustel on programmeerimise teoorias vaja üldisi tähiseid pesade ja nende sisude jaoks. Lepime kokku, et kasutame tähiseid a, a+2, a+4, ..., b, b+2, b+4, ..., c, c+2, c+4, ... suvalise paarisnumbrilise lühikese pesa tähistamiseks. Paaritunumbriliste lühikeste pesade tähistamiseks kasutame tähiseid a+1, a+3, ..., b+1, b+3, ..., c+1, c+3, ... . Pesad a ja a+1 on seejuures muidugi järjestikused lühikesed pesad. Pikkade pesade puhul kasutame tähiseid a+4000, a+4002 jne. Märgime, et pikk pesa a+4000 moodustatakse lühikestest pesadest a ja a+1, pikk pesa a+4002 lühikestest pesadest a+2 ja a+3 jne. Paarisnumbriliste lühikeste pesade, paaritunumbriliste lühikeste pesade ja pik-

kade pesade tähiseid on otstarbekohane eristada, sest nende pesadega opereerimisel ilmnevad paljude käskude puhul erinevused.

Vajaduse korral kasutame tähte u mistahes pesa (kas lühikese või pika) tähistamiseks, tähte v aga mistahes lühikese pesa (kas paaris- või paaritunumbrilise) tähistamiseks.

Summaatorit ja aritmeetilise seadme registrit tähistame vastavalt s ja r abil.

Lepime kokku, et mingis pesas a, a+1 või a+4000 olevat masina sõna tähistame vastavalt (a), (a+1) või (a+4000) abil, s.t. sama pesa tähisega ümarsulgudes. Analoogiliselt tähistame aritmeetilise seadme registri sisu (r) abil. Summaatori sisu jaoks kasutame kahte tähist: $(s)_0$ - summaatori sisu enne vastava tehte sooritamist, (s) - summaatori sisu pärast vastava tehte sooritamist. Pesade sisude ülalesitatud tähistusviisi kasutame ka konkreetsete numbriliste aadresside korral. Nii võime näidetes 1 ja 2 toodud faktid esitada lühidalt kujul

(0117) = 12 3506

(s.t. pesa 0117 sisu on masina sõna 12 3506) ja

(2034)= - 16 5274

(pesa 2034 sisu on käsk - 16 5274).

2. Summaatoril ja aritmeetilise seadme registril on teatavasti 6 lisakohta. Lisakohtadel on arvutustehniline eesmärk - vajaliku täpsuse tagamine. Programmeerimisel neid arvestada ei tule.

3. Summaatoris on arv alati modifitseeritud pöördkoodis,

teistes seadmetes aga otsekoodis. Programmeerimisel (ka loogiliste tehete puhul) pole aga koodide erinevuse arvestamine üldse vajalik. Aritmeetiliste tehete puhul, mis nii või teisiti taandatakse liitmistehetele summaatoris, teisendatakse teine liidetav automaatselt modifitseeritud pöördkoodi. Loogiliste tehete puhul teisendatakse summaatoris olev arv aga automaatselt otsekoodi. Seega võime programmeerimisel arvestada (peale mõne erandi) ainuüksi arvude otsekoode.

4. Ranget vahet tuleb teha arvude +0 ja -0 vahel. Arvu +0 loeme positiivseks arvuks (see on vähim positiivne arv), arvu -0 loeme negatiivseks (suurim negatiivne arv). Seega on loomulik aktsepteerida ka võrratus $-0 < +0$. Tähendame, et aritmeetilise seadme konstruktsioonist tingitult ei kehti märgireeglid arvudega +0 ja -0 korrutamisel.

5. Märk "-" käsu ees on selle käsu modifitseerimise tunnuseks. Niisugustel käskudel teisendatakse käskude registris (magnettrumlil säilib käsk endisel kujul) enne nende täitmist aadressiosa: sellest lahutatakse tsüklite loendaja sisu. Et selline teisendus toimub ühel ja samal viisil kõigi miinus märgiga käskude puhul, siis vaatleme käskude süsteemi käsitlemisel käske tavalisel kujul, s.t. ilma märgita "-". Miinus märgiga käskudest kõneleme käskude 24 ja 25 juures, üksikasjalikumalt aga paragrahvis 15.

6. Käskude süsteemi esitamisel tuleb iga käsu juures märkida, millistel tingimustel tekib signaal $\omega$ (ehk millal $\omega = 1$). Need andmed on vajalikud tingitud suunamise käsu 21 kasutamisel, mis võimaldavad arvutuskäigu harunemist olenevalt

vahetulemustest.

Samuti on vajalikud andmed ületäitumise signaali $\varphi$ tekkimise kohta.

7. Peale eespool nimetatud asjade mõjustab käskude täitmist veel tumblerite asend juhtimispuldil. Kui näiteks on sisse lülitatud tumbler "kirjutamise blokeerimine" (magnettrumli puhul), siis käsk tehte koodiga 16 (arvu kirjutamine magnettrumlile) jääb täitmata. Et tumblerite funktsioonidest kõnelesime küllaltki üksikasjalikult juhtimispuldi kirjeldamisel, siis pole vajadust vastavate märkuste tegemiseks käskude süsteemi käsitlemisel.

Asume nüüd üksikute käskude vaatlemisele.

Liitmine. Tehte kood 01. Käsu kuju on kas 01 a, 01 a+1 või 01 a+4000, s.t. käsu aadressiosas võib olla mistahes lühikese või pika pesa aadress. Kasutades tähist u, võib selle käsu kirjutada kujul 01 u.

Käsu 01 u korral liidetakse summaatoris olevale arvule pesas u olev arv. Liidetavaid $(s)_0$ ja (u) vaadeldakse seejuures 2-ndsüsteemi arvudena, kus koma on fikseeritud pärast märgikohta. Tehte tulemus - arvude $(s)_0$ ja (u) summa - jääb summaatorisse. Arv pesas u säilib seejuures.[12] Kui käsu aadressiosas on lühikese pesa aadress, siis liidetakse vastavas lühikeses pesas olev arv summaatori esimese poole sisuga. Seega saame

$$(s) = (s)_0 + (u).$$

[12] Et aritmeetiliste tehete puhul vaadeldakse tehtest osavõtvaid masina sõnu alati 2-ndsüsteemi arvudena, kus koma on fikseeritud pärast märgikohta, siis seda edaspidi enam ei korrata. Samuti ei korrata väiteid "tehete tulemus jääb summaatorisse" ja "arv pesas u säilib".

Ületäitumise korral (s.t. kui $|(s)| \geqslant 1$), ja ainult siis, antakse signaal $\varphi$. Signaal $\omega$ antakse $(s) \leqslant -0$ puhul ja ainult siis. Lühiduse huvides esitame signaalide $\varphi$ ja $\omega$ tekkimise tarvilikud ja piisavad tingimused valemitega

$$(\varphi = 1) \equiv (|(s)| \geqslant 1), \quad (\omega = 1) \equiv ((s) \leqslant -0).$$

Tähendame, et kui liitmisel saame tulemuseks nulli, siis on see -0, välja arvatud juhtum (+0) + (+0), mis annab +0 (vt. § 4).

Arvu toomine summaatorisse. Tehte kood 02. Käsu kuju 02 u.

Käsu 02 u korral tühjendatakse summaator ja tuuakse seejärel pesas u olev arv summaatorisse. Kui käsu aadressiosas on lühikese pesa aadress, siis tuuakse (pärast summaatori tühjendamist) vastavas lühikeses pesas olev arv summaatori esimesse poolde. Seega saame

$$(s) = (u).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \quad (\omega = 1) \equiv ((s) \leqslant -0).$$

Lahutamine. Tehte kood 03. Käsu kuju 03 u.

Käsu 03 u korral lahutatakse summaatoris olevast arvust pesas u olev arv. Kui käsu aadressiosas on lühikese pesa aadress, siis lahutatakse vastavas lühikeses pesas olev arv summaatori esimesest poolest. Seega saame

$$(s) = (s)_0 - (u).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$(\varphi = 1) \equiv (|(s)| \geqslant 1), \quad (\omega = 1) \equiv ((s) \leqslant -0).$$

Tähendame, et kui lahutamisel saame tulemuseks nulli,

siis on see - 0, välja arvatud juhtum (+0) - (-0) = +0.

Absoluutväärtuste lahutamine. Tehte kood 04. Käsu kuju 04 u. Käsu 04 u korral lahutatakse summaatoris oleva arvu absoluutväärtusest pesas u oleva arvu absoluutväärtus. Kui käsu aadressiosas esineb lühikese pesa aadress, siis lahutatakse vastavas lühikeses pesas olev arv summaatori esimesest poolest. Seega saame

$$(s) = |(s)_0| - |(u)|.$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) \leqslant -0).$$

Tähendame, et kui absoluutväärtuste lahutamisel saame nulli, siis on see alati -0.

Tehet 04 võib muuhulgas kasutada arvu absoluutväärtuse leidmiseks. Seda saab teha käskudega

02 a

04 b,

kus pesas a on antud arv ja (b) = +0.

Korrutamine koos liitmisega. Tehte kood 05. Käsu kuju 05 u.

Käsu 05 u korral liidetakse summaatoris olevale arvule aritmeetilise seadme registris ja pesas u oleva arvu korrutis. Seega saame

$$(s) = (s)_0 + (r) \cdot (u).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$(\varphi = 1) \equiv (|(s)| \geqslant 1), \qquad (\omega = 1) \equiv ((s) \leqslant -0).$$

Tehte 05 abil on võimalik arvutada avaldisi kujul

$\sum_{i=1}^{n} A_i B_i$ ilma vahetulemuste salvestamiseta magnettrumlil.

Seda saab teha käskudega

```
17 a1
05 b1      (eeldame seejuures, et
17 a2      Ai = (ai),  Bi = (bi),
05 b2          i = 1,2,...,n)
.....
.....
```

Märgime, et üldiselt muutub aritmeetilise seadme registri sisu iga käsuga. Seepärast tuleb arv tuua aritmeetilise seadme registrisse vahetult enne käsku 05 (vt. tehet koodiga 17).

Korrutamine. Tehte kood 06. Käsu kuju 06 u.

Käsu 06 u korral korrutatakse summaatoris olev arv pesas u oleva arvuga. Seega

$$(s) = (s)_0 \cdot (u).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) < -0).$$

Märgireeglid korrutamisel kehtivad (ka neil juhtumitel, kus korrutatakse arvudega +0 ja -0).

Summaatoris oleva arvu korrutamisel arvuga $2^{-n}$ toimub üldiselt summaatori sisu nihutamine n kohta paremale. Umardamise tõttu ei kehti see aga alati. Näiteks korrutades arvu

00,0111 ... 1

arvuga $2^{-2}$, saame

00,0010 ... 0.

Korrutades arvu $2^{-35}$ arvuga $2^{-1}$ saame $2^{-35}$. Tõepoolest, õige tulemus $2^{-36}$ kujutatakse numbriga 1 lisasummaatori kõrgeimal kohal, see aga ümardatakse ülespoole, s.t. arvuks $2^{-35}$.

Jagamine. Tehte kood 07. Käsu kuju 07 u.

Käsu 07 u korral jagatakse summaatoris olev arv pesas u oleva arvuga. Seega

$$(s) = (s)_0 : (u).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$(\varphi = 1) \equiv (|(s)| \geq 1), \qquad (\omega = 1) \equiv ((s) \leq -0).$$

Märgireeglid nullist erinevate arvude jagamisel kehtivad. Erinevused esinevad aga juhtudel $(\pm 0) : (-0) = +0$ ja $(-0) : (+0) = +0$. Muide, viimasel kahel juhul on ka $\varphi = 1$. Juhtumitel $(+0) : (+0)$ ja $(-0) : (-0)$ tekib positiivne ületäitumine.

Jagamistehe teostatakse arvutusmasinas 4 taktiga.

Märgi omistamine. Tehte kood 10. Käsu kuju 10 u.

Käsu 10 u korral antakse summaatoris olevale arvule pesas u oleva arvu märk. Seega

$$(s) = \text{sign}(u) \cdot |(s)_0|, \text{ kus } \text{sign}(u) = \begin{cases} +1, \text{ kui } (u) \geq +0, \\ -1, \text{ kui } (u) \leq -0. \end{cases}$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) \leq -0).$$

Esitatud tehted on aritmeetilised tehted.

Edasi järgnevad neli loogilist tehet.

Nihutamine. Tehte kood 11. Käsu kuju 11 0000 (loogiline nihutamine) või 11 4000 (aritmeetiline nihutamine).

Käsu 11 0000 korral nihutatakse aritmeetilise seadme registris olev masina sõna (kõik kahendkohad, märgikoht kaasa arvatud) k kahendkohta ($-77_8 \leqslant k \leqslant 77_8$) vasakule või paremale. Nihutuskonstant k tuleb salvestada summaatori 24. - 19. kahendkohal, konstandi k märk aga summaatori märgikohal. Summaatori teisi kahendkohti nihutamisel ei arvestata. Nihutuse suuna määrab nihutuskonstandi k märk: $k > +0$ korral toimub nihutus vasakule, $k < -0$ korral paremale poole. Kahendnumbrid, mis nihutamisel väljuvad registri piirest, kustutatakse. Nihutamisel juurdetulevad kahendnumbrid on nullid.

Peale seda kustutatakse nihutuskonstant summaatorist ja tuuakse nihutatud arv aritmeetilise seadme registrist summaatorisse. Arvu ülekandmisel summaatorisse teisendatakse tulemus modifitseeritud pöördkoodi, tõlgendades registri 36-ndat kohta kui märgikohta. Nihutatud arv registris säilib (otsekoodis).

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid,

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) = \pm 0).$$

Käsu 11 4000 korral nihutatakse eespool kirjeldatud viisil ainult aritmeetilise seadme registris oleva arvu numbrikohti (s.t. 1. - 35-ndat kahendkohta), kuna märgikoht jääb paigale ja omistatakse seega lõpptulemusele. Kahendnumbrid, mis nihutamisel väljuvad registri 1. - 35-nda kahendkoha piirest, kustutatakse. Muus osas toimub nihutamine käsu 11 4000 korral samuti nagu käsu 11 0000 puhul.

Signaalid $\varphi$ ja $\omega$ tekivad samadel tingimustel.

Eeldusel, et täisosa ei arvestata ja ümardamist ei teostata, on aritmeetiline nihutamine samaväärne arvuga $2^k$ korrutamisele.

Konjunktsioon. Tehte kood 12. Käsu kuju on kas 12 a+4000, 12 a või 12 a+1.

Käsu 12 a+4000 korral teostatakse summaatoris oleva masina sõna (mis on teisendatud otsekoodi) ja pesas a+4000 oleva masina sõna vastavate kahendnumbrite konjunktsioon. Märgikoht võtab tehtest osa samuti nagu teised kohad. Seega võime kirjutada

$$(s)^i = (s)_0^i \;\&\; (a+4000)^i, \quad (i = 1,2,\ldots,36),$$

kus $(s)^i$ ja $(a+4000)^i$ abil tähistame vastavate pesade i-ndal kohal asuvat numbrit.

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) = +0).$$

Näide.

$$\begin{array}{rl} & (s)_0 = \boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\ \ldots \\ \& & (a+4000) = \boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\ \ldots \\ \hline & (s) = \boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\ \ldots \end{array}$$

Lühikeste pesade puhul teostatakse konjunktsioon järgmiselt.

Käsu 12 a korral teostatakse tehe nii, nagu oleks tegemist pika pesaga, mille esimeses pooles on (a), teises pooles aga +0. Käsu 12 a+1 korral teostatakse tehe nii, nagu oleks tegemist pika pesaga, mille esimeses pooles on +0, teises pooles aga (a+1).

Muus osas toimub konjunktsioon lühikeste pesade puhul samuti nagu käsu 12 a+4000 puhul. Signaalide $\varphi$ ja $\omega$ tekkimise tingimused on kõikidel juhtudel samasugused.

Konjunktsiooni teostamisel tuleb pöörata tähelepanu sellele, et tehe teostatakse ikka vastavate kahendkohtadega. Nii näiteks pole võimalik vahetult konjunktsiooni teostada summaatori esimese poole ja paaritunumbrilise lühikese pesaga.

Konjunktsiooni kasutatakse tavaliselt summaatoris oleva masina sõna kahendkohtade eraldamiseks. Näiteks, kui on vaja eraldada 35. - 33. kahendkohad, tuleb teostada konjunktsioon 12 a, kus pesas a oleva masina sõna kõik kahendnumbrid on nullid, välja arvatud 17. 16. ja 15. kahendkohad, s.t. (a) = $= 01110\ldots0_2 = 340\ 000_8$.

Disjunktsioon. Tehte kood 13. Käsu kuju on kas 13 a+4000, 13 a või 13 a+1.

Käsu 13 a+4000 korral teostatakse summaatoris oleva masina sõna (mis on teisendatud otsekoodi) ja pesas a+4000 oleva masina sõna vastavate kahendnumbrite disjunktsioon. Märgikoht võtab tehtest osa samuti nagu teised kohad. Seega võime kirjutada

$$(s)^i = (s)_0^i \vee (a+4000)^i, \quad (i = 1,2,\ldots,36).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) = +0).$$

Näide.

| | | | | | | |
|---|---|---|---|---|---|---|
| | $(s)_0 =$ | 1 | 0 | 1 | 1 | 0 | ... |
| $\vee$ | $(a+4000) =$ | 0 | 1 | 1 | 0 | 0 | ... |
| | $(s) =$ | 1 | 1 | 1 | 1 | 0 | |

Käskude 13 a ja 13 a+1 teostamisel vaadeldakse pesi a ja a+1 nagu pikki pesi samal viisil, nagu seda tehti konjunktsiooni korral.

Disjunktsiooni kasutatakse sagedasti vajaliku masina sõna moodustamiseks mitmest komponentsõnast.

Mitteekvivalents ehk võrdlus. Tehte kood 14. Käsu kuju on kas 14 a+4000, 14 a või 14 a+1.

Käsu 14 a+4000 korral teostatakse summaatoris oleva masina sõna (mis on teisendatud otsekoodi) ja pesas a+4000 oleva masina sõna vastavate kahendnumbrite mitteekvivalentsi tehe. Märgikoht võtab tehtest osa samuti nagu teised kohad. Seega võime kirjutada.

$$(s)^i = (s)^i_0 \approx (a+4000)^i, \qquad (i = 1,2,\ldots,36).$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) \neq +0).$$

Näide.

$$\begin{array}{rl} & (s)_0 = \boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\ \ldots \\ \approx & (a+4000) = \boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\ \ldots \\ \hline & (s) = \boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\ \ldots \end{array}$$

Käskude 14 a ja 14 a+1 teostamisel vaadeldakse pesi a ja a+1 nagu pikki pesi samal viisil, nagu seda tehti konjunktsiooni korral.

Mitteekvivalentsi tehet kasutatakse tavaliselt arvude (masina sõnade) võrdlemiseks. Kui masina sõnade $(s)_0$ ja (a+4000) kõik kahendnumbrid on vastavalt võrdsed, saame summaatoris pärast tehte 14 sooritamist +0 (signaal $\omega = 0$). Kui aga nendel masina sõnadel kasvõi ühel kahendkohal on erinevad

numbrid, tuleb sellele kahendkohale pärast tehte 14 sooritamist summaatoris 1, s.t. $(s) \neq +0$ ja seega $\omega = 1$.

Tehte 14 a+4000 korral, kus $(a+4000) = 111 \ldots 11_2 = 77 \ldots 7_8$, teostatakse summaatoris asuva masina sõna kõigi kahendnumbrite ümberpööramine (eitus).

Tähendame, et pärast loogiliste tehete sooritamist teisendatakse summaatoris asuv tulemus automaatselt modifitseeritud pöördkoodi. Seejuures vaadeldakse 36-ndat kahendkohta märgikohana.

Normaliseerimine. Tehte kood 15. Käsu kuju on tavaliselt 15 a+4000 (võimalik on ka kuju 15v).

Käsu 15 a+4000 korral toimub summaatoris oleva arvu normaliseerimine. See toimub ületäitumise puhul (kui $|(s)_0| \geq 1$) ja juhul $|(s)_0| < 1$ erinevalt.

Kui summaatoris oli tekkinud ületäitumine, nihutatakse summaatoris olev arv ühe koha võrra paremale. Vasakult juurdetulev kahendnumber on positiivse $(s)_0$ korral 0, negatiivse $(s)_0$ korral 1.

Näiteid.

$(s)_0$ = 0 1 1 0 1 1 0 ... $(s)_0$ = 1 0 1 0 1 1 1 ...

pärast nihutamist 0 0 1 1 0 1 1 ... pärast nihutamist 1 1 0 1 0 1 1 ...

Pärast nihutust viiakse summaatoris olev arv pesasse a+4000 ja summaatoris salvestatakse nihutatud kohtade arv märgiga "+", s.t. +1. Ühik "1" salvestatakse summaatori 19-ndal kahendkohal, märk "+" märgikohal.

Juhul $|(s)_0| < 1$ nihutatakse summaatoris oleva arvu numbrikohad vasakule, kuni 35-ndale kahendkohale tuleb esi-

mene number 1 (arvu vaatleme siin otsekoodis). Arvu märgikohti ei nihutata. Kui esimene 1 on kohe peale koma, siis nihutamist ei toimu. Seejärel viiakse nihutatud arv pessa a+4000, kusjuures summaatoris salvestatakse nihutatud kohtade arv märgiga "-". Nihutatud kohtade arv salvestatakse summaatori esimeses pooles alates 19-ndast kahendkohast, märk "-" salvestatakse märgikohal.

Näide.

$(s)_0$ = 1 1 0 0 0 0 0 1 0 1 1 ...
(otsekoodis)

arv summaatoris pärast nihutamist (otsekoodis)
1 1 1 0 1 1 . . . . . ...

Nihutatud kohtade arvu (-5) salvestamine summaatoris (modifitseeritud pöördkoodis).

| 37. | 36. | 35. | 34. | 33. | 32. | 31. | 30. | 29. | 28. | 27. | 26. | 25. | 24. | 23. | 22. | 21. | 20. | 19. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s)_0 = \pm 0).$$

Normaliseerimise tehe teostatakse arvutusmasinas 2 takti jooksul. Normaliseerimise käsu olemasolu võimaldab arvutada nn. liikuva koma meetodil. Selleks tuleb esitada arvud $x$ kujul $m_x \cdot 2^{p_x}$, kus mantiss $m_x$ on tavaliselt pikas pesas, järk $p_x$ aga lühikeses pesas. Normaliseerimise käsku läheb vaja mantisside normaliseerimiseks. Käsuga 15 a+4000 viiakse normaliseeritud mantiss pessa a+4000, kusjuures summaatorisse jääb arvu järgu "parandus" - arvu järgule liidetav suurus (mis on summaatoris salvestatud juba vajaliku märgiga).

Arvu viimine summaatorist magnettrumlile. Tehte kood 16. Käsu kuju on tavaliselt 16 a+4000, 16 a või 16 a+1.

Käsu 16 a+4000 korral viiakse summaatoris olev arv magnettrumli pessa a+4000. Pesa a+4000 endine sisu kustutatakse seejuures automaatselt. Summaatori sisu aga säilib. Seega võime kirjutada

$$(a+4000) = (s)_0 , \qquad (s) = (s)_0 .$$

Käskude 16 a või 16 a+1 korral viiakse pessa a või vastavalt a+1 summaatori esimese poole sisu.

Tähendame, et ka summaatori teise poole sisu on võimalik viia magnettrumlile, aga ainult paaritunumbrilisse lühikesesse pessa. Sel juhul tuleb aadressiosale liita 4000, nii et vastav käsk tuleb kirjutada kujul 16 a+4001.

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s)_0 \leqslant -0).$$

Arvu toomine registrisse. Tehte kood 17. Käsu kuju on 17 u.

Käsu 17 u korral tühjendatakse aritmeetilise seadme register ja tuuakse seejärel registrisse pesas u olev arv. Kui käsu aadressiosa on lühikese pesa aadress, siis tuuakse (peale registri tühjendamist) vastavas lühikeses pesas olev arv registri esimesse poolde. Summaatori ja pesa u sisu seejuures säilib. Seega saame

$$(r) = (u), \qquad (s) = (s)_0.$$

Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((r) = \pm 0).$$

Tähendame, et enamiku tehete teostamisel ( välja arvatud tehted koodiga 21, 22, 23, 24, 25, 30, 32 ja 34) muutub aritmeetilise seadme registri sisu. Seepärast peab vajaliku arvu toomine registrisse toimuma reeglina vahetult enne seda

käsku, mis kasutab registrisse toodud arvu. Nii näiteks ei saa nihutamisel kasutada sellist käskude järjekorda

17 a+4000 (nihutatava arvu toomine registrisse),

02 b (nihutuskonstandi toomine summaatorisse),

11 0000 (nihutamine),

sest arvu toomisel summaatorisse rikutakse aritmeetilise seadme registri sisu. Vastavad käsud tuleb järjestada nii:

02 b

17 a+4000

11 0000.

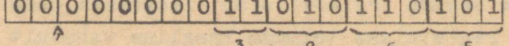
Arvu moodustamine summaatoris. Tehte kood 20. Käsu kuju on 20 k, kus k võib olla mistahes neljakohaline 8-ndsüsteemi numbrite abil esitatav masina sõna (s.t. $0000 \leqslant k \leqslant 7777$).

Käsu 20 k korral tõlgendatakse aadressiosa k kõige kõrgemat kahendkohta märgikohana (numbriga "1" kodeeritakse märk "-", numbriga "0" märk "+") ja see kantakse summaatori märgikohtadele, teised kahendnumbrid kantakse summaatori 29. - 19. kahendkohtadele. Summaatori teistele kahendkohtadele tulevad positiivse k (s.t. $k \leqslant 3777$) korral nullid, negatiivse k (s.t. $k \geqslant 4000$) korral aga ühed. Tehte 20 k korral moodustatakse sama arv ka aritmeetilise seadme registris. Summaatoris on vastav arv muidugi modifitseeritud pöördkoodis, registris otsekoodis.

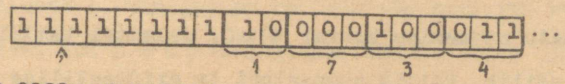
Signaalide $\varphi$ ja $\omega$ tekkimist iseloomustavad valemid

$$\varphi \equiv 0, \qquad (\omega = 1) \equiv ((s) \leqslant -0).$$

Näiteid. Käsu 20 3265 puhul tuleb summaatori sisu järgmine



Käsu 20 5734 puhul tuleb summaatori sisu ($k \geqslant 4000$) järgmine




Käsu 20 0000 puhul tühjendatakse summaator (s.t. moodustatakse seal arv +0).

Käsku 20 k kasutatakse ette teadaolevate konstantide (näiteks nihutuskonstandid, ümberadresseerimiskonstandid jms.) toomiseks summaatorisse. Kui selleks otstarbeks kasutaksime käsku 02 u, peaksime need konstandid enne salvestama magnettrumlile. Käsu 20 k korral pole neid arve eelnevalt vaja magnettrumlil salvestada, millega saab vähendada arvutusmasinasse sisseviidava materjali (pesade) hulka.

Edasi järgnevad neli suunamiskäsku.

Tingitud suunamine. Tehte kood 21. Käsu kuju on kas 21 v või 21 v+4000, kus v on mistahes lühikese pesa aadress.

Käsu 21 v korral sõltub arvutuste käik signaalist $\omega$ järgmiselt. Kui eelmisel taktil oli $\omega = 0$, teostatakse käsule 21 v järgnev käsk, s.t. käskude täitmine toimub tavalises järjekorras. Kui eelmisel taktil oli aga $\omega = 1$, antakse juhtimine üle pesas v asuvale käsule.

Käsu 21 v+4000 korral toimub ümberpöördult: $\omega = 0$ korral antakse juhtimine pesas v asuvale käsule, $\omega = 1$ korral täidetakse aga käsule 21 v+4000 järgnev käsk.

Tingitud suunamise korral $\varphi \equiv 0$, eelmisel taktil väljatöötatud $\omega$ säilib. Säilib ka summaatori ja aritmeetilise seadme registri sisu.

Tingimatu suunamine. Tehte kood 22. Käsu kuju on 22 v.

Käsu 22 v korral antakse juhtimine üle pesas v asuvale käsule.

Tingimatu suunamise korral $\varphi \equiv 0$, eelmisel taktil väljatöötatud $\omega$ säilib. Säilib summaatori ja aritmeetilise seadme registri sisu.

Suunamine tumbleri abil. Tehte kood 23. Käsu kuju 23 k, kus k on kas 0001, 0002, 0003, 0004, 0005, 0006 või 0007.

Käsu 23 k korral sõltub arvutuste käik tumbleri number k asendist juhtimispuldil järgmiselt. Kui tumbler numbriga k on välja lülitatud, teostatakse käsule 23 k järgnev käsk. s.t. käskude täitmine toimub tavalises järjekorras. Kui tumbler numbriga k on aga sisse lülitatud, siis jäetakse käsule 23 k vahetult järgnev käsk vahele ja täidetakse ülejärgnev käsk. Teiste tumblerite asend käsule 23 k mõju ei avalda.

Käsu 23 k korral $\varphi \equiv 0$, eelmisel taktil väljatöötatud $\omega$ säilib. Säilib summaatori ja aritmeetilise seadme registri sisu.

Käsk 23 k võimaldab ülesande lahendamise käigus programmist välja lülitada mõningad osad. Selleks tuleb juhtimise üleandmine vastavale programmi osale kirjutada kujul

23 k

22 v.

Kui tumbler number k on välja lülitatud, siis antakse juhtimine sellele programmi osale. Kui tumbler k on sisse lülitatud, siis selle programmi osa poole ei pöörduta.

Käsku 23 k võib kasutada ka programmi silumise hõlbustamiseks.

Tsükli lõpp. Tehte kood 24. Käsu kuju 24 v.

Käsu 24 v korral oleneb arvutuste käik tsüklite loendaja sisust L. Oluline on ka tsüklite registri kõrgeim kahendnumber R. Kui L = 0, teostatakse käsule 24 v järgnev käsk. Kui L ≠ 0, lahutatakse tsüklite loendaja sisust R + 1 ja antakse seejärel juhtimine pesas v asuvale käsule.

Käsu 24 v korral $\varphi \equiv 0$, eelmisel taktil väljatöötatud $\omega$ säilib. Summaatori ja aritmeetilise seadme registri sisu säilib.

Käsku 24 v kasutatakse koos käsuga 25 m tsükliliste arvutuste teostamisel.

Tsükli algus. Tehte kood 25. Käsu kuju 25 m, kus m = 4000 · R + n, R on kas 0 või 1 ja n on vahemikus $0000 \leqslant n \leqslant 3777$.

Käsu 25 m korral viiakse arv m tsüklite registrisse. Täpsemini: arv R viiakse tsüklite registri kõrgeimale kahendkohale, arv n aga tsüklite loendajasse.

Käsu 25 m korral $\varphi \equiv 0$, säilib eelmisel taktil väljatöötatud $\omega$. Summaatori ja aritmeetilise seadme registri sisu säilib.

Käsule 25 m järgnevad "-" märgiga käsud teisendatakse enne täitmist nii, et nende käskude aadressiosadest lahutatakse tsüklite loendaja sisu. Kui käsud on ilma märgita "-", siis täidetakse need tavalisel kujul.

Selgitame mõne näite varal, kuidas käskude 25 m ja 24 v abil saab organiseerida tsüklilisi arvutusi.

Näide. Koostame programmi suuruse $A \cdot B^{20}$ arvutamiseks. Olgu arv A pesas a+4000, arv B pesas a+4002. Programmi

koostamisel arvestame, et $20_{10} = 24_8$. Alustame programmi pesast 0100. Programm tuleb järgmine:

(0100) = 02 a+4000 (A toomine summaatorisse)
(0101) = 25 0023
(kui L ≠ 0) (0102) = 06 a+4002
(0103) = 24 0102.

Käsk 25 0023 viib tsüklite registri kõrgeimale kahendkohale arvu R = 0 ja tsüklite loendajasse arvu n = 0023, mis iga tsükli korral väheneb R + 1 = 1 võrra. Et iga tsükli korral korrutatakse summaatoris olev arv arvuga B, siis saame

| tsüklite loendaja sisu | 0023 | 0022 | 0021 | ... | 0001 | 0000 |
|---|---|---|---|---|---|---|
| summaatoris olev arv (astendaja on 8-ndsüsteemis) | $A \cdot B$ | $A \cdot B^2$ | $A \cdot B^3$ | ... | $A \cdot B^{23}$ | $A \cdot B^{24}$ |

Pärast seda, kui summaatoris on $A \cdot B^{24}$, tuleb järg jälle käsu 24 0102 kätte. Kuid siis on tsüklite loendaja juba 0. Seega teostatakse käsule 24 0102 järgnev (pesas 0104 salvestatud) käsk ja tsükliline arvutus on lõpetatud.

Ülalesitatud arvutusi võib teostada ka pisut erineva programmi abil. Nimelt

(0100) = 02 a+4000
(0101) = 25 4046
(kui L ≠ 0) (0102) = 06 a+4002
(0103) = 24 0102.

Käsk 25 4046 viib tsüklite registri kõrgeimale kahendkohale arvu R = 1 ja tsüklite loendajasse arvu n = 0046, mis iga tsükli korral väheneb R + 1 = 2 võrra. Analoogiliselt

saame

tsüklite loenda-ja sisu summaatoris olev arv (astendaja on 8-ndsüsteemis)

| 0046 | 0044 | 0042 | ..... | 0002 | 0000 |
|---|---|---|---|---|---|
| $A \cdot B$ | $A \cdot B^2$ | $A \cdot B^3$ | ... | $A \cdot B^{23}$ | $A \cdot B^{24}$ |

Pärast seda, kui summaatoris on $A \cdot B^{24}$, on tsüklite loendaja sisu 0. Seega teostatakse käsule 24 0102 järgnev käsk pesast 0104 ja tsükliline arvutus on lõpetatud.

Üldse, 25 n ($n \leqslant 3777$) puhul teostatakse programmi osa, mis asub käskude 25 n ja 24 v vahel, n + 1 korda, käsu 25 n+4000 puhul (kus n on paarisarv ja $n \leqslant 3776$) aga $\frac{n}{2} + 1$ korda.

Näide. Koostame programmi korrutise $A_0 \cdot A_1 \cdot A_2 \cdot \ldots \cdot A_{10}$ (10 on 8-ndsüsteemi arv - seega kaheksa) arvutamiseks. Olgu arvud $A_0$, $A_1$, ..., $A_{10}$ järjestikustes pikkades pesades

4500, 4502, ... , 4516, 4520.

Vastavat programmi alustame näiteks pesast 0060. See tuleb järgmine:

(0060) = 02 4500 ($A_0$ toomine summaatorisse)
(0061) = 25 4016 (R = 1, n = 0016)
(kui L ≠ 0) (0062) = -06 4520
(0063) = 24 0062.

Et R = 1, vähendatakse tsüklite loendaja sisu igal tsüklil 2 võrra. Selles programmis esineb käsk märgiga "-", mida täitmisel teisendatakse (lahutatakse tema aadressiosast tsüklite loendaja ja sisu L). Programmi tööd iseloomustab alljärgnev tabel:

| tsüklite loendaja sisu | 0016 | 0014 | 0012 | ... | 0002 | 0000 |
|---|---|---|---|---|---|---|
| täitmisele tulev "-"märgiga käsk | 06 4502 | 06 4504 | 06 4506 | ... | 06 4516 | 06 4520 |
| summaatoris olev arv | $A_0\ A_1$ | $A_0 \cdot A_2 \cdot A_3$ | $A_0 \cdot A_1 \cdot A_2 \cdot A_3$ | ... | $A_0 \cdot A_1 \cdot \ldots \cdot A_7$ | $A_0 \cdot A_1 \cdot \ldots \cdot A_{10}$ |

Pärast seda, kui summaatoris on $A_0 \cdot A_1 \cdot \ldots \cdot A_{10}$, on tsüklite loendaja sisu 0. Seega teostatakse käsule 24 0062 järgnev käsk pesast 0064 ja tsükliline arvutus on lõpetatud.

Üldiselt öeldes peavad pesad, mille aadresse soovitakse igal tsüklil teisendada, olema järjestikku (kas järjestikused pikad või järjestikused lühikesed pesad). Käsk märgiga "-" tuleb programmis kirjutada maksimaalse muutuva aadressiga (vaadeldud näites on aadress 4520 tõepoolest muutuvate aadressside hulgas maksimaalne).

Käsu 25 m aadressiosa m saab leida nii nagu eelmises näites - programmi teostamiste arvu järgi. Arvestades seda, et muuta tuleb pikkade pesade aadresse, peab m olema kujul m = n+4000. Kordamisele tulevat programmi osa teostatakse $10_8$ korda. Seega $\frac{n}{2} + 1 = 10$, millest $\frac{n}{2} = 7$ ja $n = 2 \cdot 7 = 16$.

Käsu 25 m aadressiosa saab leida muutuvate aadresside korral aga ka teisiti: maksimaalse muutuva ja minimaalse muutuva aadressi vahena. Kui tsüklite loendaja muutub 2-kaupa (s.t. muutuvad aadressid on pikkade pesade aadressid), siis tuleb liita kõrgeimale kohale 4. Antud näites saame (arv $A_0$ pesast 4500 on juba summaatorisse toodud):

4520 - 4502 = 0016, 4000 + 0016 = 4016.

Märgime veel, et käsk 25 m asetatakse käskude rühma ette, mida tuleb korduvalt teostada. Käsu 25 m ja korduvalt teosta-

tava programmi osa vahel võib olla aga ka teisi käske, mida ei korrata. Käsk 24 v peab aga olema vahetult korduvalt teostatava programmi osa järel.

Kontroll-liitmine. Tehte kood 26. Käsu kuju on 26 u.

Käsu 26 u korral liidetakse summaatoris olevale arvule pesas u olev arv. Ületäitumise signaal $\varphi \equiv 0$. Signaali $\omega$ tekkimist iseloomustab valem $(\omega = 1) \equiv ((s) \leqslant -0)$.

Tehet 26 u kasutatakse kontrollsummade arvutamiseks, milleks on vajalik just ületäitumise signaali puudumine.

Tuleb märkida, et kahe teineteisele järgneva programmi osa või üldse informatsiooni massiivi kontrollsummat ei saa leida osade kaupa.

Käsu muutmine. Tehte kood 30. Käsu kuju on 30 v.

Käsu 30 v korral teisendatakse pesas v olev masina sõna pöördkoodi, vaadeldes seejuures kõrgeimat kahendnumbrit märgina (1 tähendab märki "-", 0 märki "+"). Seejärel liidetakse pöördkoodi teisendatud (v) käsule 30 v järgneva käsuga, mis on otsekoodis. Liitmine toimub käskude registris tsüklilise ülekandega kõrgeimast järgust madalaimasse järku. Järgmisel taktil täidetakse sellisel viisil modifitseeritud käsk. Programmis endas (s.t. magnettrumlil) jääb käsk loomulikult endisele kujule. Summaatori ja aritmeetilise seadme registri sisu selle käsu korral ei muutu. Tehet 30 võib kasutada ka tsüklites "-" märgiga käskude puhul, samuti võib seda tehet rakendada ka korduvalt (üksteise järel).

Signaal $\varphi \equiv 0$, eelmisel taktil väljatöötatud $\omega$ säilib.

Selgitame tehet 30 paari näite varal.

Näide. Olgu programmis järgmised käsud

(0144) = 30 1325

(0145) = 02 4320,

kusjuures pesas 1325 on masina sõna 00 1240. Masina sõna 00 1240 kirjutatud kahendsüsteemi numbrite abil tuleb

000 000 001 010 100 000.

Tõlgendades esimest, kõrgeimat kahendnumbrit "0" märgina "+" ja teisendades selle masina sõna pöördkoodi, saame sama tulemuse, sest see "arv" on positiivne. Liidame nüüd saadud masina sõna käsuga 02 4320

| | | | | | | |
|---|---|---|---|---|---|---|
| | 000 | 000 | 001 | 010 | 100 | 000 |
| käsk 02 4320 (otsekoodis) | 000 | 010 | 100 | 011 | 010 | 000 |
| | 000 | 010 | 101 | 101 | 110 | 000 |
| | 0 | 2 | 5 | 5 | 6 | 0 |

Seega tuleb järgmisel taktil käsu 02 4320 asemel täitmisele käsk 02 5560.

Näide. Olgu programmis järgmised käsud

(0220) = 30 0540

(0221) = 13 4022,

kusjuures pesas 0540 on masina sõna 41 0460.

Masina sõna 41 0460 tuleb kahendsüsteemi numbrites järgmine

100 001 000 100 110 000,

pöördkoodis aga selline (arv on negatiivne)

111 110 111 011 001 111.

Liidame saadud masina sõna käsuga 13 4022

| | | | | | | |
|---|---|---|---|---|---|---|
| | 111 | 110 | 111 | 011 | 001 | 111 |
| käsk 13 4022 (otsekoodis) | 001 | 011 | 100 | 000 | 010 | 010 |
| (1) | 001 | 010 | 011 | 011 | 100 | 001 |
| | 001 | 010 | 011 | 011 | 100 | 010 |
| | 1 | 2 | 3 | 3 | 4 | 2 |

Seega tuleb järgmisel taktil käsu 13 4022 asemel täitmisele käsk 12 3342. Antud juhul muutus ka tehte kood.

Tavaliselt kasutatakse tehet 30 käskude aadressiosade muutmiseks.

Ülekanne perfolindilt. Ülekanne perfolindilt teostatakse kolme käsuga:

31 $u_1$
01 N
00 $u_2$

Nende käskudega tuuakse perfolindi tsooni nr. N sisu magnettrumli järjestikustesse pesadesse alates pesast $u_1$ ja lõpetades pesaga $u_2$ ($u_1 \leq u_2$). Kui perfolindi pesades on perforeeritud 6-kohalised 8-ndsüsteemi arvud (üldiselt masina sõnad), peavad $u_1$ ja $u_2$ olema lühikeste pesade aadressid. Kui perfolindi pesades on perforeeritud 10-ndsüsteemi arvud või 12-kohalised 8-ndsüsteemi masina sõnad (perforeeritud väljundperforaatori abil), peavad $u_1$ ja $u_2$ olema pikkade pesade aadressid.

Tähendame, et kümnendsüsteemi arvude viimisel magnettrumlile on vigade vältimiseks soovitav sisse lülitada tumbler "$\varphi$ blokeerimine".

Ülekanne magnetlindilt. Ülekanne magnetlindilt teostatakse kolme käsuga:

31 $a_1$+4000
02 N
00 $a_2$+4000 ($a_1 \leq a_2$).

Nende käskudega tuuakse magnetlindi tsooni nr. N sisu

magnettrumli järjestikustesse pikkadesse pesadesse alates pesast $a_1$+4000 ja lõpetades pesaga $a_2$+4000.

Magnetlindile viimine. Magnetlindile viimine teostatakse kolme käsuga:

31 $a_1$+4000
03 N
00 $a_2$+4000 $(a_1 \leq a_2)$.

Nende käskudega viiakse magnettrumli järjestikuste pikkade pesade sisu alates pesast $a_1$+4000 ja lõpetades pesaga $a_2$+4000 magnetlindi tsooni nr. N.

Et rühmatehete korral viiakse arvud ühest salvestusseadmest teise läbi summaatori, jääb pärast rühmatehete sooritamist summaatorisse mingi arv. Käsku 30 v võib asetada rühmatehte käskude ette, mitte aga nende käskude vahele.

Tulemuse trükkimine. Tehte kood 32. Käsu kuju on 32 0000.

Käsu 32 0000 korral trükitakse summaatoris olev masina sõna (kas 8-nd või 10-ndsüsteemis) paberlindile või perforeeritakse väljundperforaatori abil.

Enne arvude trükkimist kümnendsüsteemis tuleb need arvud teisendada kahendsüsteemist kümnendkahend koodi.

Ühe arvu trükkimiseks kulub 60 takti. Arvu trükkimisel trükkimisseadmes jätkab masin oma tööd programmi alusel. Seega, kui arve trükitakse mitte sagedamini kui 60 takti järel, siis see ei aeglusta masina tööd.

Reavahe jätmine. Tehte kood 34. Käsu kuju on 34 0000.

Käsu 34 0000 korral keeratakse paberlint trükkimisseadmes ühe rea võrra edasi ilma midagi trükkimata.

Peatus. Tehte kood 37. Käsu kuju 37 u.

Käsu 37 u korral tuuakse pesas u olev arv summaatorisse. Käskude loendaja sisu suurendatakse 1 võrra ja käskude registrisse tuuakse järgmises lühikeses pesas olev käsk. Seejärel peatatakse käskude täitmine.

## § 17. Näiteid ülesannete programmeerimise kohta.

Eelmises paragrahvis kõnelesime muuhulgas "ülesande lahendamise programmist". Käesolevas paragrahvis täpsustame programmi mõistet ja toome mõningaid näiteid programmidest ja nende koostamisest.

Ülesande $\mathcal{A}$ lahendamise programmiks arvutusmasinal $\mathcal{M}$ nimetame ülesande $\mathcal{A}$ lahendamise algoritmi, mis on esitatud masina $\mathcal{M}$ käskude süsteemis.

Algandmeid programmi koostisosaks ei loeta. Programmi koostisosadeks loetakse aga ülesande lahendamiseks vajalikud konstandid, abiarvud, skaalad jne., mis ei sõltu algandmetest, vaid antud ülesande tüübist. Näiteks arvutades $\sin x$ väärtusi valemi

$$\sin x = \sum_{i=0}^{3} C_{2i+1} x^{2i+1}, \quad |x| \leq \frac{\pi}{2}$$

järgi, kus $C_1 = 0{,}999\,9992$, $C_3 = -0{,}166\,6567$, $C_5 = 0{,}008\,3132$, $C_7 = -0{,}000\,1852$, vaadeldakse konstante $C_1$, $C_3$, $C_5$ ja $C_7$ (need ei sõltu argumendist $x$) programmi koostisosana. Argumendi $x$ väärtusi, mille puhul on vaja arvutada $\sin x$, programmi osaks aga ei loeta.

Seega jaguneb ülesande lahendamiseks vajalik informat-

sioon kahte suurde ossa: 1) ülesande lahendamise programm, mis sobib muide teatavale ülesannete klassile (algandmed võivad teataval määral varieeruda), 2) ülesannet lähemalt määravad algandmed.

Programm ise jaguneb samuti kahte ossa: käsud (6-kohalised 8-ndsüsteemi masina sõnad) ja nn. abiarvud (programmi kuuluvad konstandid, skaalad, loendajad jne.). Viimased võivad mõnel juhul olla ka 10-ndsüsteemis. Sellisel juhul peavad need arvud olema perforeeritud programmi käskudest eraldi tsoonis.

Algandmed on matemaatiliste ülesannete puhul tavaliselt 10-ndsüsteemi arvud, loogiliste ülesannete korral aga 8-ndsüsteemi masinasõnad. Kuigi algandmed ja programm (või selle osa) võivad teatavatel juhtudel olla kodeeritud ühes ja samas süsteemis, ei ole seda materjali siiski otstarbekohane perforeerida ühte ja samasse tsooni. Põhjuseks on asjaolu, et vastavat tüüpi ülesannete korduval lahendamisel tuleks sel puhul korduvalt perforeerida ka teatavat mittemuutuvat informatsiooni.

Enne programmide koostamise käsitlemist teeme veel mõned üldised märkused. Nimelt esineb programmi koostamisel teataval määral vasturääkiv olukord: programmi käske ei saa kirja panna enne, kui pole teada pesad algandmete, abiarvude ja vahetulemuste jaoks. Teiselt poolt ei saa aga pesi algandmete, abiarvude ja vahetulemuste jaoks planeerida enne, kui pole teada programmi käskude paigutus magnettrumlil.

Sellest olukorrast võib leida väljapääsu aga sel teel, et programmi koostamisel kasutada mitte konkreetseid numbrilisi,

vaid tähelisi aadresse. Seejuures on otstarbekohane kasutada üksikuid tähti veel eri tähenduses. Nii näiteks võib algandmete pesi tähistada a+4000, a+4002, ..., programmisse kuuluvate abiarvude pesi b+4000, b+4002, ..., vahetulemuste pesi c+4000, c+4002,..., lõpptulemuste pesi d+4000, d+4002,..., programmi enese käskude pesi k+0, k+1, ... . Muidugi oleneb pesade tähistus programmeerija suvast ja võib erineda erinevate ülesannete korral.

Pärast programmi koostamist tähelistes aadressides ja algandmete planeerimist antakse üksikutele tähtedele kooskõlastatud numbrilised väärtused (materjal magnettrumlil ei tohi kattuda) ja kirjutatakse programm koos algandmetega ümber konkreetsetes numbrilistes aadressides.

Tähendame veel, et programmi kirjapanekul tuleb tingimata koos käsuga kirja panna ka vastava pesa aadress, sest vastupidisel juhul pole võimalik üles kirjutada suunamiskäskude aadressiosi. Kui näiteks käsk 02 a+4000 asub pesas k+30, käsk 06 a+4002 asub pesas k+31, siis tuleb § 16 toodud tähistusi arvestades kirjutada nii:

(k+30) = 02 a+4000,

(k+31) = 06 a+4002,

- - - - - - - - - - - .

Programmi üleskirjutuse lihtsustamiseks jätame aga edaspidi ära sulud ja võrdusmärgid, kirjutades lihtsalt:

k+30 02 a+4000

k+31 06 a+4002

- - - - - - - - - - - .

Programmide koostamisel tuleb arvestada veel mitmeid

komplitseerivaid asjaolusid. Nii näiteks tuleb hoolikalt arvestada ületäitumise võimalusi ja ette näha teed ületäitumise vältimiseks või selle tagajärgede likvideerimiseks. Tuleb arvestada täpsuse kao võimalusi ja kindlustada arvutuste nõutav täpsus. Samuti tuleb programmis ette näha algandmete teisendamine 10-ndsüsteemist 2-ndsüsteemi ja tulemuste teisendamine 2-ndsüsteemist 10-ndsüsteemi. Kui algandmed on absoluutselt suuremad kui 1, tuleb kasutada kas sobivaid mastaabikordajaid või muid võtteid. Kuidas kõiki neid asjaolusid arvestada, näeme järgmistes paragrahvides. Alljärgnevates näidetes eeldame, et ülaltoodud komplikatsioonid ei esine, s.t. arvutuste käigus ületäitumist ei esine, algandmed on 2-ndsüsteemis ja neid võib ka 2-ndsüsteemi arvudena masinast välja trükkida jne.

Näide 1. Koostame programmi kahe muutujaga lineaarse võrrandisüsteemi

$$\left.\begin{aligned} A_1x + B_1y &= C_1 \\ A_2x + B_2y &= C_2 \end{aligned}\right\} \qquad (1)$$

lahendamiseks, lähtudes lahendusvalemitest

$$x = \frac{C_1B_2 - C_2B_1}{A_1B_2 - A_2B_1} = \frac{D_1}{D}, \qquad y = \frac{A_1C_2 - A_2C_1}{A_1B_2 - A_2B_1} = \frac{D_2}{D}, \qquad (2)$$

mis eeldavad, et $D \neq 0$.

Et algandmeid on vähe (6 arvu) ja programm tuleb äärmiselt lihtne, siis koostame selle numbrilistes aadressides.

Programmi koostamisel eeldame, et algandmed $A_1$, $A_2$, $B_1$, $B_2$, $C_1$, $C_2$ (2-ndsüsteemis!) asuvad vastavalt pikkades pesades 4400 ÷ 4412. Ehk üksikasjalikumalt: (4400) = $A_1$, (4402) = $A_2$, (4404 = $B_1$, (4406) = $B_2$, (4410) = $C_1$, (4412) = $C_2$.

Koostatava programmi paigutame magnettrumlile alates näiteks pesast 0020. Esitame selle koos selgitavate märkustega.

| Pesa aadress | Käsk | Märkused |
|---|---|---|
| 0020 | 02 4402 | Arvu $A_2$ toomine summaatorisse: (s) = $A_2$; |
| 0021 | 06 4404 | (s) = $A_2B_1$; |
| 0022 | 16 7600 | Arvu $A_2B_1$ viimine pessa 7600; |
| 0023 | 02 4400 | (s) = $A_1$; |
| 0024 | 06 4406 | (s) = $A_1B_2$; |
| 0025 | 03 7600 | (s) = $A_1B_2 - A_2B_1$; |
| 0026 | 16 7600 | Arvu D = $A_1B_2 - A_2B_1$ viimine pessa 7600; |
| 0027 | 17 7600 | Arv D viiakse registrisse. Kui (r) = $\pm$0, siis $\omega$ = 1; |
| 0030 | 21 0051 | Kui (r) = $\pm$ 0, siis antakse juhtimine pesas 0051 olevale käsule, s.t. töö lõpeb; |
| 0031 | 02 4412 | (s) = $C_2$; |
| 0032 | 06 4404 | (s) = $C_2B_1$; |
| 0033 | 16 7602 | Arvu $C_2B_1$ viimine pessa 7602; |
| 0034 | 02 4410 | (s) = $C_1$; |
| 0035 | 06 4406 | (s) = $C_1B_2$; |
| 0036 | 03 7602 | (s) = $C_1B_2 - C_2B_1 = D_1$; |
| 0037 | 07 7600 | (s) = $D_1$ : D = x; |
| 0040 | 32 0000 | x = $D_1$ : D-trükkimine; |
| 0041 | 02 4402 | (s) = $A_2$; |
| 0042 | 06 4410 | (s) = $A_2C_1$; |
| 0043 | 16 7602 | Arvu $A_2C_1$ viimine pessa 7602; |
| 0044 | 02 4400 | (s) = $A_1$; |

| | | |
|---|---|---|
| 0045 | 06 4412 | (s) = $A_1C_2$; |
| 0046 | 03 7602 | (s) = $A_1C_2 - A_2C_1 = D_2$; |
| 0047 | 07 7600 | (s) = $D_2 : D = y$; |
| 0050 | 32 0000 | y = $D_2$ : D trükkimine; |
| 0051 | 37 7600 | peatus. |

Seega paikneb programm pesades 0020 ÷ 0051. Algandmed asuvad pesades 4400 ÷ 4412 ehk, kasutades lühikeste pesade numeratsiooni, pesades 0400 ÷ 0413. Pesi 7600 ja 7602 kasutatakse vahetulemuste säilitamiseks. Niisuguseid pesi nimetatakse tööpesadeks. Tavaliselt valitakse nad magnettrumli lõpuosast.

Programmi algandmete ja tööpesade paigutus magnettrumlil oleneb muidugi programmeerija suvast, kusjuures tuleb aga juhinduda ajalise-ruumilise ökonoomsuse ja otstarbekohasuse kriteeriumist (materjali hõlpus viimine magnettrumlile, selle otstarbekohane paigutus, ülesande võimalikult väike lahendamise aeg jne.).

Selgitame veel, kuidas viia ülesande lahendamiseks vajalik materjal (programm ja algandmed) magnettrumlile. Eeldades, et programm on perforeeritud näiteks tsoonis nr. 3 ja algandmed tsoonis nr. 4, perforeerime tsooni nr. 2 järgmised käsud

| | | |
|---|---|---|
| 0002 | 31 0020 | perfolindi tsooni nr.3 materjal viiakse magnettrumli pesadesse 0020 - 0051; |
| 0003 | 01 0003 | |
| 0004 | 00 0051 | |
| 0005 | 31 0400 | perfolindi tsooni nr.4 materjal viiakse magnettrumli pesadesse 0400 - 0413; |
| 0006 | 01 0004 | |
| 0007 | 00 0413 | |

0010 22 0020 juhtimine antakse pesas 0020 asuvale käsule.

Algkäivitusega viime tsooni nr. 2 sisu magnettrumli pesadesse 0002 ÷ 0010. Seejärel antakse juhtimine automaatselt pesas 0002 asuvale käsule. Edasi viiakse magnettrumlile 3. tsooni, seejärel 4. tsooni materjal ning antakse juhtimine üle pesas 0020 asuvale käsule, s.t. asutakse ülesande lahendamisele.

Näide 2. Koostame programmi funktsiooni

$$y = \frac{C_0x - C_1x^2 + C_2x^5}{C_0 + C_3x^2 + C_4x^4 + C_5x^5} \qquad (3)$$

väärtuste arvutamiseks argumendi väärtustel $x = x_0, x_1, x_2, \ldots, x_n$.

Koostame siin programmi tähelistes aadressides. Olgu algandmed paigutatud järgmiselt:

$(a+4000) = x_0$, $(a+4002) = x_1$, $(a+4004) = x_2, \ldots$, $(a+4000+2n) = x_n$,

$(b+4000) = C_0$, $(b+4002) = C_1$, $(b+4004) = C_2$, $(b+4006) = C_3$,

$(b+4010) = C_4$, $(b+4012) = C_5$, $(b+0014) = 2n$.

Enne programmeerimist teisendame avaldise (3) kujule

$$y = \frac{\left[(C_2x^3 - C_1)x + C_0\right]x}{\left[(C_5x + C_4)x^2 + C_3\right]x^2 + C_0}. \qquad (4)$$

Programm tähelistes aadressides tuleb järgmine:

| | | | |
|---|---|---|---|
| k +0 | 20 0000 | (s) = +0; |
| k+1 | 16 c+0 | pesas c+0 on esialgu +0, hiljem 2,4,..., 2n; |
| k+2 | 30 c+0 | (c+0) abil modifitseeritakse järgmise käsu aadressi; |

| | | | |
|---|---|---|---|
| k+3 | 02 | a+4000 | |
| k+4 | 16 | c+4002 | (c+4002) = $x_i$; |
| k+5 | 06 | c+4002 | (s) = $x_i^2$; |
| k+6 | 16 | c+4004 | (c+4004) = $x_i^2$; |
| k+7 | 02 | b+4012 | (s) = $C_5$; Nimetaja arvutamine |
| k+10 | 06 | c+4002 | (s) = $C_5x_i$; |
| k+11 | 01 | b+4010 | (s) = $C_5x_i + C_4$; |
| k+12 | 06 | c+4004 | (s) = $(C_5x_i + C_4)x_i^2$; |
| k+13 | 01 | b+4006 | (s) = $(C_5x_i + C_4)x_i^2 + C_3$; |
| k+14 | 06 | c+4004 | (s) = $\left[(C_5x_i + C_4)x_i^2 + C_3\right]x_i^2$; |
| k+15 | 01 | b+4000 | (s) = $\left[(C_5x_i + C_4)x_i^2 + C_3\right]x_i^2 + C_0$; |
| k+16 | 16 | c+4006 | pesas c+4006 asub nimetaja; |
| k+17 | 17 | c+4006 | nimetaja viimine registrisse, kui (r) = |
| k+20 | 21 | k+33 | $= \pm 0$, siis $\omega = 1$; |
| k+21 | 02 | b+4004 | (s) = $C_2$; Lugeja arvutamine |
| k+22 | 06 | c+4004 | (s) = $C_2x_i^2$; |
| k+23 | 06 | c+4002 | (s) = $C_2x_i^3$; |
| k+24 | 03 | b+4002 | (s) = $C_2x_i^3 - C_1$; |
| k+25 | 06 | c+4002 | (s) = $(C_2x_i^3 - C_1)x_i$; |
| k+26 | 01 | b+4000 | (s) = $(C_2x_i^3 - C_1)x_i - C_0$; |
| k+27 | 06 | c+4002 | (s) = $\left[(C_2x_i^3 - C_1)x_i - C_0\right]x_i$ |
| k+30 | 07 | c+4006 | (s) = y |
| k+31 | 32 | 0000 | y trükkimine |
| k+32 | 22 | k+34 | |
| k+33 | 34 | 0000 | kui mingi $x_i$ korral on nimetaja 0, jäetakse reavahe; |

k+34 20 0002

k+35 01 c+0

k+36 16 c+0 (c+0) suurendamine 2 võrra;

k+37 03 b+0014 (s) = 2i - 2n, kui i ≤ n on (s) ≤ -0, ja $\omega$ = 1;

k+40 21 k+2 suunatakse järjekordse $y_i$ arvutamisele;

k+41 37 c+0 peatus.

Kui programm on valmis, siis on lihtne määrata tähtedele sobivad arvulised väärtused. Nii näiteks võime võtta k = 20. Programm asub siis pesades 0020 ÷ 0061. Edasi võime võtta b = 70 ja a = 106. Oletades, et n on näiteks $100_8$, mahuvad kõik algandmed magnettrumli pesadesse 4070 ÷ 4306 ehk lühikeste pesade numbreid kasutades, pesadesse 0070 ÷ 0307. Tööpesade c+0, c+4002, c+4004 ja c+4006 puhul võib c väärtuseks võtta mistahes paarisarvu, mis rahuldab tingimust $310 \leqslant c \leqslant 3770$.

Eeldades, et programm on perforeeritud näiteks tsooni nr. 5 ja algandmed tsooni nr. 27, saab selle materjali viia magnettrumlile järgmiselt. Tsooni nr. 2 perforeerime käsud

| | | |
|---|---|---|
| 0002 | 31 0020 | tsooni nr. 5 materjal viiakse magnettrumli pesadesse 0020 ÷ 0061 |
| 0003 | 01 0005 | |
| 0004 | 00 0061 | |
| 0005 | 31 0070 | tsooni nr. 27 materjal viiakse magnettrumli pesadesse 0070 ÷ 0307. |
| 0006 | 01 0027 | |
| 0007 | 00 0307 | |
| 0010 | 22 0020 | juhtimine antakse pesas 0020 asuvale käsule. |

Pärast tsooni nr. 2 viimist magnettrumli pesadesse

0002 + 0010 antakse juhtimise pesas 0002 asuvale käsule. Seega viiakse magnettrumlilt tsoonis 5 ja 27 materjal ning antakse seejärele juhtimine pesas 0020 asuvale käsule, millega algab ka ülesande lahendamine.

Näide 3. Koostame programmi kahe neljarealise ruutmaatriksi korrutamiseks. Korrutatavate maatriksite A ja B elemendid olgu järgmistes pesades:

$a_{11}$ = (a+4000), $a_{12}$ = (a+4002), $a_{13}$ = (a+4004), $a_{14}$ = (a+4006);
$a_{21}$ = (a+4010), $a_{22}$ = (a+4012), $a_{23}$ = (a+4014), $a_{24}$ = (a+4016);
$a_{31}$ = (a+4020), $a_{32}$ = (a+4022), $a_{33}$ = (a+4024), $a_{34}$ = (a+4026);
$a_{41}$ = (a+4030), $a_{42}$ = (a+4032), $a_{43}$ = (a+4034), $a_{44}$ = (a+4036);
$b_{11}$ = (b+4000), $b_{12}$ = (b+4002), $b_{13}$ = (b+4004), $b_{14}$ = (b+4006);
.....
$b_{41}$ = (b+4030), $b_{42}$ = (b+4032), $b_{43}$ = (b+4034), $b_{44}$ = (b+4036).

Maatriksite A ja B korrutamisel korrutatakse teatavasti maatriksi A read maatriksi B veergudega, s.t. korrutismaatriksis element $c_{ij}$ avaldub kujul

$$c_{ij} = \sum_{k=1}^{4} a_{ik} b_{kj}. \qquad (5)$$

Programmi koostamisel on otstarbekohane muuta reaindeksit i järgmiselt: i = 0,10,20,30, mis vastab pesade aadresside vahele eri ridades. Veeruindeksit j on otstarbekohane muuta aga nii: j = 0,2,4,6; indeksit k on otstarbekohane muuta k = 0,1,2,3. Programm tähelistes aadressides tuleb järgmine:

```
k+0   20 0000   (s) = +0
k+1   16 c+0    (c+0) = i
```

```
k+2   16 c+1     (c+1) = j
k+3   16 c+2     (c+2) = j +10 k,  k = 0,1,2,3;
k+4   25 4006
k+5   30 c+0     Elemendi c_ij arvutamine.
k+6 - 02 a+4006
k+7   30 c+2
k+10  06 b+4000  (s) = a_ik b_kj;
k+11  01 c+4004
k+12  16 c+4004  summa (5) liidetavad kogutakse pessa
                                          c+4004;
k+13  20 0010
k+14  01 c+2
k+15  16 c+2
k+16  24 k+5
k+17  02 c+4004
k+20  32 0000    c_ij trükkimine;
k+21  20 0000
k+22  16 c+4004  pesa c+4004 tühjendamine;
k+23  20 0006
k+24  03 c+1
k+25  21 k+32    kui j = 6, siis minnakse üle reaindeksi
                            suurendamisele;
k+26  20 0002
k+27  01 c+1
k+30  16 c+1     j suurendamine 2 võrra;
k+31  22 k+3
k+32  20 0030
```

```
k+33   03 c+0
k+34   21 k+44   kui i = 30, siis lõpetatakse töö;
k+35   20 0010
k+36   01 c+0
k+37   16 c+0    i suurendamine 10 võrra;
k+40   20 0000
k+41   16 c+1    j muudetakse nulliks;
k+42   34 0000   uue rea juurde minekul jätame reavahe
k+43   22 k+3
k+44   37 0000   peatus.
```

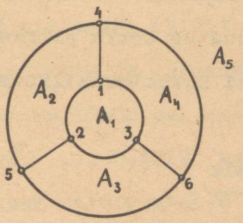
Andes tähtedele k, a, b näiteks väärtused vastavalt 30, 100, 140 (suuruse c võib võtta vahemikus $200 \leqslant c \leqslant 3772$) ning eeldades, et programm on perforeeritud 3. tsooni ja algandmed 5. tsooni, saab selle materjali viia magnettrumlile algkäivitusega, kus tsoonis nr. 2 on käsud

```
0002   31 0030
0003   01 0003
0004   00 0074
0005   31 0100
0006   01 0005
0007   00 0177
0010   22 0030.
```

Näide 4. Koostame programmi, mis arvutab joonisel 1 toodud topoloogilise kaardi[13] kõik kahevärvilahendid.

[13] Topoloogiliseks kaardiks nimetame tasandi jaotust (pidevate kõverate abil) lõplikuks arvuks piirkondadeks, kusjuures üks piirkond on tõkestamata.

Kõigepealt tuleb defineerida kahevärvilahend. Kaardil (joon.1) on 5 piirkonda: $A_1$, $A_2$, $A_3$, $A_4$ ja $A_5$. Omistame igale piirkonnale $A_i$ ($i = 1,2,\ldots,5$) kas väärtuse 1 või 0 (s.t. värvime iga piirkonna $A_i$ kas "värviga" 1 või "värviga" 0). Näiteks kujutab võrduste komplekt $A_1=1$, $A_2=0$, $A_3=1$, $A_4=0$, $A_5=1$ (ehk lihtsalt 10101) üht piirkondade värvimist. Kui osutub, et piirkondade värvimisel ühegi sõlmpunkti juures asuvast 3 piirkonnast kõik kolm pole ühte värvi, siis ongi tegemist ühe kahevärvilahendiga.




Joon. 1.

Kerge on kontrollida, et piirkondade värvimine 10101 on kahevärvilahend, 10110 aga pole (3. sõlmpunkti juures asuvad piirkonnad $A_1$, $A_3$ ja $A_4$ on kõik "värvi" 1). On tõestatud, et igal normaalkaardil[14] leidub kahevärvilahend.

On ilmne, et kahevärvilahendi värvide ümbervahetamisel saame jälle kahevärvilahendi. Nii näiteks annab kahevärvilahend 10101 värvide ümbervahetamisel kahevärvilahendi 01010. Seega võib kahevärvilahendite leidmisel piirduda ainult poolte lahendite leidmisega (s.t. lahenditega, kus näiteks $A_1 = 0$), sest ülejäänud lahendid (kus $A_1 = 1$) saame juba leitud lahenditest värvide vahetamise teel.

Kas piirkondade värvimine osutub kahevärvilahendiks või mitte, selgub kontrollimisel sõlmpunktidega. Iga sõlmpunkt

[14] Normaalkaardiks nimetame sellist topoloogilist kaarti, kus igast sõlmpunktist väljub täpselt 3 lüli (nagu kaardil joonisel 1).

puutub kokku 3 piirkonnaga. 1. sõlmpunkt näiteks piirkonnaga $A_1$, $A_2$ ja $A_4$. Seda informatsiooni sõlmpunktide kohta kodeerime 5-kohalise 2-ndsüsteemi arvuna, kus i-ndal kohal asuv number 1 tähendab, et vastav sõlmpunkt puutub kokku piirkonnaga $A_1$. Sellist arvu nimetame sõlmpunkti karakteristikuks. Seega saame

| Sõlmpunkt | Karakteristik |
|---|---|
| 1. ..... | 11010 |
| 2. ..... | 11100 |
| 3. ..... | 10110 |
| 4. ..... | 01011 |
| 5. ..... | 01101 |
| 6. ..... | 00111 |

Värvimisi $A_1A_2A_3A_4A_5$ vaatleme samuti 5-kohaliste 2-ndsüsteemi arvudena vahemikus 00000 ÷ 10000, suurendades neid 1 liitmise teel viimasele 2-ndkohale. Summaatoris ja pesades kasutame värvimiste ja sõlmpunktide karakteristikute salvestamiseks 2-ndkohti 23 - 19.

Värvimine $A_1A_2A_3A_4A_5$ kujutab kahevärvilahendit siis ja ainult siis, kui ta rahuldab kaht järgmist tingimust.

1) Värvimine $A_1A_2A_3A_4A_5$ ja ühegi sõlmpunkti karakteristiku $B_1B_2B_3B_4B_5$ konjunktsioon ei tohi anda tulemuseks 00000 (ehk +0). Tõepoolest, kui mingi sõlmpunkti karakteristiku korral saaksime konjunktsiooni tulemuseks +0, siis oleksid selle sõlmpunkti juures asuvad 3 piirkonda värvitud kõik värviga 0. Tähendame, et konjunktsioon teostatakse vastavate kahendkohtade $A_i$ ja $B_i$ vahel (nagu tehte 12 puhul).

2) Värvimise $A_1A_2A_3A_4A_5$ ja ühegi sõlmpunkti karakteristiku $B_1B_2B_3B_4B_5$ konjunktsioon ei tohi anda tulemuseks $A_1A_2A_3A_4A_5$. Sel juhul oleksid vastava sõlmpunkti juures asuvad 3 piirkonda värvitud kõik värviga 1.

Abiarvu ja algandmed salvestame järgmiselt: maksimaalse värvimise 10000 salvestame pesas a+4000; pesades b+4000, b+4002, ..., b+4012 aga sõlmpunktide karakteristikud. Informatsioon salvestatakse tegelikult nende pesade esimeses pooles, kuna pesade teine pool on tühi. Informatsiooni kompaktsemat salvestamist ei võimalda asjaolu, et nende pesadega tuleb teostada loogilisi tehteid, mis teatavasti toimuvad erinevalt paaris- ja paaritunumbriliste lühikeste pesade puhul.

Programm tuleb järgmine:

| | | |
|---|---|---|
| k+0 | 20 0001 | uue värvimise moodustamine; |
| k+1 | 01 c+4000 | |
| k+2 | 16 c+4000 | uus värvimine salvestatakse pesas c+4000; |
| k+3 | 12 a+4000 | kui (s) = +0 ($\omega$ = 1), siis antud värvimine pole maksimaalne; |
| k+4 | 21 k+6 | suunatakse värvimise uurimise juurde; |
| k+5 | 37 c+4000 | |
| k+6 | 25 4012 | algab tingimuste 1) ja 2) kontrollimine; |
| k+7 | 02 c+4000 | |
| k+10 | -12 b+4012 | kui (s) = +0 ($\omega$ = 1), siis pole täidetud tingimus 1), |
| k+11 | 21 k+0 | suunab uue värvimise moodustamise juurde; |
| k+12 | -14 b+4012 | kui (s) = +0 ($\omega$ = 0), siis pole täidetud tingimus 2); |
| k+13 | 21 k+4000 | suunab uue värvimise moodustamise juurde; |

```
k+14    24 k+7
k+15    02 c+4000
k+16    32 0000     kahevärvilahendi trükkimine;
k+17    22 k+0      suunab uue värvimise moodustamise juurde.
```

Abiarvu 10 000 = (a+4000) võib paigutada kohe programmi käskude järele. Seega saame a = k+20 ja

```
k+20    00 0020 }
                } maksimaalne värvimine.
k+21    00 0000 }
```

Võttes näiteks k = 30, tuleb a = 50 ja kogu programm paikneb pesades 0030 ÷ 0051. Algandmed võib paigutada näiteks alates pesast 0060 (b = 60). Seega saame

```
0060    00 0032 }
                } 1. sõlmp. karakteristika
0061    00 0000 }

0062    00 0034 }
                } 2. sõlmp.      -"-
0063    00 0000 }

0064    00 0026 }
                } 3. sõlmp.      -"-
0065    00 0000 }

0066    00 0013 }
                } 4. sõlmp.      -"-
0067    00 0000 }

0070    00 0015 }
                } 5. sõlmp.      -"-
0071    00 0000 }

0072    00 0007 }
                } 6. sõlmp.      -"-
0073    00 0000 }
```

Oletades näiteks, et esitatud programm on perforeeritud tsooni nr. 7 ja algandmed tsooni nr. 13, saab selle materjali tuua magnettrumlile algkäivitusega, kus tsoonis nr. 2 on

järgmised käsud:

```
0002  31 0030
0003  01 0007
0004  00 0051
0005  31 0060
0006  01 0013
0007  00 0073
0010  22 0030 .
```

## § 18. Tsüklite programmeerimine.

Matemaatiliste ja loogiliste ülesannete lahendamisel kasutatakse sageli (näiteks polünoomi väärtuste arvutamisel, iteratsioonimenetluste puhul jne.) korduvalt (järjestikku) üht ja sama arvutuseeskirja, kontrollitakse samalaadseid tingimusi.

Programmeerimise seisukohalt on niisugust arvutuskäiku otstarbekohane realiseerida programmi korduvalt kasutatava osa abil. Niisugust korduvalt (järjestikku) kasutatavat programmi osa ühes kordamiste arvu loendava ja kordamist juhtiva osaga nimetataksegi tsükliks.

Ülalöeldut silmas pidades on oluline teada tsüklite programmeerimise mitmesuguseid võimalusi ja võtteid.

Tsüklite programmeerimiseks on arvutusmasinal "Ural-1" spetsiaalsed käsud 25 m ja 24 a, mille kasutamise kohta tõime mõningaid lihtsaid näiteid eespool. Vaatleme nüüd teistsuguseid võimalusi.

1. Tsüklid paaris ja paaritute lühikeste pesade kaupa.

Olgu järjestikustes pesades

0100, 0101, 0102, ... , 0114, 0115

arvud. Nende kõikide korrutis (mingi summaatoris oleva arvuga) on leitav järgmiselt:

k+0 25 0015 ($a_{max}$-$a_{min}$= 0115 - 0100 = 0015.)

k+1 -06 0115

k+2 24 k+1 .

Paarisaadressiliste lühikeste pesade sisu korrutise summaatoris oleva arvuga saame järgmiselt:

k+0 25 4014 ($a^{paaris}_{max}$ - $a^{paaris}_{min}$ + 4000 = 0114 - 0100 + + 4000 = 4014)

k+1 -06 0114 ($a^{paaris}_{max}$)

k+2 24 k+1 .

Paarisaadressilised lühikesed pesad valitakse välja seetõttu, et tsüklite loendaja sisu muutub 2-kaupa (tsüklite registri kõrgeim koht on 1:).

Paarituaadressiliste lühikeste pesade sisu korrutise summaatori sisuga saame analoogiliselt:

k+0 25 4014 ($a^{paaritu}_{max}$ - $a^{paaritu}_{min}$ + 4000 = 0115 - 0101 + + 4000 = 4014)

k+1 -06 0115 ($a^{paaritu}_{max}$)

k+2 24 k+1 .

Arvestades tsüklite loendaja sisu muutumist 2-kaupa: 0014, 0012, 0010, 0006, 0004, 0002, 0000 saame muutuvad aadressid vastavalt 0101, 0103, 0105, 0107, 0111, 0113 ja 0115 m.o.t.t.

## 2. Tsükkel pika sammuga.

Eelmises punktis käsitletud tsükleid paaris või paaritute lühikeste pesade kaupa võib programmeerida ka veel teisiti, nimelt nii:

```
k+0   25 0014
k+1   -06 0114
k+2   24 k+3   } tsüklite loendaja sisu vähendatakse
k+3   24 k+1   } kaks korda 1-kaupa
```

ja analoogiliselt

```
k+0   25 0015
k+1   -06 0115
k+2   24 k+3   } tsüklite loendaja sisu vähendatakse kokku
k+3   24 k+1   } 2-kaupa
```

Niisugust võtet: käskude 24 järjestikust rakendamist

```
      - - - -
      - - - -
      - - - -
24    - - - -
24    - - - -
24    - - - -
```

saab kasutada tsüklite programmeerimisel, kus järjestikuste muutuvate pesade aadressid erinevad üksteisest väikese konstandi võrra. Seda konstanti nimetatakse tsükli sammuks.

Näiteks olgu vaja liita pesade

b+0000, b+0005, b+0012, ..., b+5n

sisu. Tegemist on tsükliga lühikeste pesade kaupa sammuga 5. Kasutades käskude 24 järjestikust rakendamist, saame

```
k+o   20 0000
k+1   25 5n
```

```
k+2   -01 b+5n
k+3    24 k+4 ⎫
k+4    24 k+5 ⎪
k+5    24 k+6 ⎪  tsüklite loendaja sisu vähendatakse
k+6    24 k+7 ⎪  kokku 5 võrra
k+7    24 k+2 ⎭
```

Kui samm on aga suurem kui 5, pole selline moodus enam otstarbekohane, kuna see võtab liig palju masinaaega ja pesi magnettrumlil.

Anname programmi skeemi tsüklilisteks arvutusteks lühikeste pesade kaupa sammuga $\delta$

```
k+0   25   n     (n = a_max - a_min)
k+1  -20 4000+n+δ -1
k+2   01 k+0
k+3   16 k+ℓ
        - - - - - - -  ⎫
        - - - - - - -  ⎬ tsükli käsud; märgiga "-" käsud
        - - - - - - -  ⎭ kirjutatakse kujul - ⊤ a_max.
k+ℓ   00 0000
k+ℓ+1 24 k+1 .
```

Teeme mõned selgitavad märkused programmi skeemi kohta. Käsuga -20 4000+n+$\delta$-1 moodustatakse summaatoris arv -($\delta$-1), mis liidetuna käsuga 25 n annab käsu 25 n-$\delta$+1. See viiakse pessa k+$\ell$. Pärast tsükli käskude teostamist teostatakse käsk 25 n-$\delta$+1, s.t. viiakse tsüklite loendajasse arv n-$\delta$+1, mida järgmise käsuga vähendatakse veel 1 võrra. Seega saame tsüklite loendaja uueks sisuks n-$\delta$ . Järgmisel etapil moodustatakse summaatoris arv -(2$\delta$-1) ja pessa k+$\ell$ viiakse käsk

25 n-2δ+1, mille aadressiosast pärast 1 lahutamist saame tsüklite loendaja uueks sisuks n-2δ. Seega väheneb tsüklite loendaja sisu tõesti sammu δ kaupa.

Näide 1. Koostame programmi, mis viib arvud pesadest

0730, 0750, 0770, 1010, 1030

vastavalt pesadesse

0731, 0751, 0771, 1011, 1031.

Samm δ=0750 - 0730 = 0020, δ-1 = 0017, n = 1030 - 0730 = = 0100.

Programm tuleb järgmine:

```
k+0   25 0100
k+1  -20 4117
k+2   01 k+0
k+3   16 k+6
k+4  -02 1030
k+5  -16 1031
k+6   00 0000
k+7   24 k+1.
```

Kui tegemist on tsükliliste arvutustega pikkade pesade kaupa sammuga δ, tuleb programmi skeem analoogiline

```
k+0   25 4000+n     (n = a_max - a_min)
k+1  -20 4000+n+(δ-2)
k+2   01 k+0
k+3   16 k+ℓ
        - - - -  } tsükli käsud, käsud märgiga "-"
        - - - -  } kirjutatakse kujul - T a_max.
        - - - -  }
k+ℓ   00 0000
k+ℓ+1 24 k+1 .
```

Erinevus eelmise skeemiga on ainult ühes käsus -20 4000+n+(δ-2). Selgitavad märkused on analoogilised eelnenud selgitustega.

### 3. Segatsükkel.

Sageli on vaja moodustada tsükleid, kus esinevad samaaegselt muutuvate aadressidega pikad ja lühikesed pesad. Tingimuseks on, et pikad pesad (samuti nagu lühikesed pesad) oleksid järjestikku ja muutuvate aadressidega pikkade ja lühikeste pesade arv oleks võrdne. Selliseid tsükleid nimetatakse segatsükliteks.

Segatsükli saab moodustada järgmisel viisil:

| | | | |
|---|---|---|---|
| k+0 | 25 | 4000+n | $n = a^{pikk}_{max} - a^{pikk}_{min}$ |
| k+1 | -20 | n | |
| k+2 | 06 | b | $(b) = \frac{1}{4}$ |
| k+3 | 16 | b+1 | |
| | - - - - | | tsükli käsud; muutuvate aadressidega käsud, kus esinevad pikkade pesade aadressid, tuleb kirjutada kujul - $T\, a^{pikk}_{max}$, muutuvate lühikeste pesadega käsud tuleb kirjutada kujul $T\, c_{min}$, kus $c_{min}$ on minimaalne muutuva lühikese pesa aadress. |
| | - - - - | | |
| | - - - - | | |
| | - - - - | | |
| k+ℓ | 24 | k+1 | |

Näide 2. Koostame programmi summa

$$\sum_{i=1}^{6} A_i B_i$$

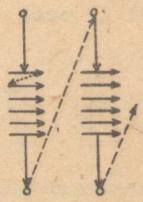
leidmiseks, kus arvud $A_i$ on pikkades pesades 5120 ÷ 5132 ja $B_i$ lühikestes pesades 1344 ÷ 1351.

| | | | |
|---|---|---|---|
| k+0 | 20 | 0000 | |
| k+1 | 16 | a+4000 | |
| k+2 | 25 | 4012 | (n = 5132 - 5120 = 0012) |
| k+3 | -20 | 0012 | |
| k+4 | 06 | b | (b) = ½ |
| k+5 | 16 | b+1 | |
| k+6 | -02 | 5132 | tsükli käsud |
| k+7 | 30 | b+1 | |
| k+10 | 06 | 1344 | |
| k+11 | 01 | a+4000 | |
| k+12 | 16 | a+4000 | |
| k+13 | 24 | k+3 | |

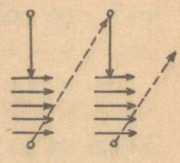
## 4. Tsükkel tsüklis.

Mõnikord on vaja moodustada tsükli ühe etapi raames veel teine tsükkel. Sel korral kõnelemegi tsüklist tsüklis. Arvutuskäik võib sel korral olla mitmesugune, kas

1) väline tsükkel sisemine tsükkel



või 2) väline tsükkel sisemine tsükkel



Peale selle võivad väline ja sisemine tsükkel asuda kas samas massiivis (paikneda programmis üksteise järel) või erinevates massiivides. Tsüklit tsüklis võib programmeerida käskude 25 m ja 24 a abil. Sisemise tsükli puhul mingeid probleeme ei esine, seda võib vaadelda iseseisva tsüklina. Välise tsükli puhul seisneb probleem aga selles, kuidas taastada õige tsük-

lite loendaja sisu pärast tagasipöördumist sisemiselt tsüklilt välisele tsüklile. Mõningal määral muudab programmi ka see, kas väline ja sisemine tsükkel on eri massiivides või mitte ja kas väline tsükkel on pikkade või lühikeste pesade kaupa.

Näide 3. Olgu väline ja sisemine tsükkel ühes massiivis. Programm võib sel juhul olla näiteks selline:

```
k+0   25  n1          }
k+1  -20  4000+n1     }  Välise tsükli I osa. Kui väline tsükkel
k+2   01  k+0         }  on pikkade pesade kaupa, tuleb pesas
k+3   16  k+ℓ         }  k+0 asuv käsk asendada käsuga
                      }  25 4000+n1 .
      - - - - - -     }
      - - - - - -     }
      - - - - - -     }

      25  n2          }
   ┌- - - - - -     }  sisemine tsükkel. Sisemine tsükkel
   |  - - - - - -     }  võib loomulikult olla ka pikkade
   |                  }  pesade kaupa.
   └ 24               }

k+ℓ   00  0000        }  Välise tsükli II osa.
      - - - - - -     }
      - - - - - -     }
      - - - - - -     }
k+ℓ+m 24  k+1         }
```

Käsuga -20 4000+$n_1$ viiakse esimesel sammul summaatorisse -0, mis liidetakse käsuga 25 $n_1$ ja viiakse pessa k+$\ell$, millega tagatakse õige tsüklite loendaja sisu välise tsükli II osa puhul. Järgmisel sammul on tsüklite loendaja sisu $n_1$ -1, summaatorisse viiakse -1, pessa k+$\ell$ käsk 25 $n_1$ - 1, millega tagatakse jälle õige tsüklite loendaja sisu jne. Vastavat programmi pole raske kohandada ka juhule, kus väli-

ne ja sisemine tsükkel on eri massiivides.

Näide 4. Olgu väline ja sisemine tsükkel eri massiivides. Peale selle on väline tsükkel pikkade pesade kaupa.

Programm võib olla näiteks selline:

I massiiv

(väline tsükkel)

k+0 25 4000+$n_1$ (lühikeste pesade puhul asendub käsuga
- - - - - - 25 $n_1$)
- - - - - -
- - - - - -

k+$\ell$ 22 m+0

k+$\ell$+1 24 k+1

II massiiv

(sisemine tsükkel)

m+0 -20 7777

m+1 16 a
25 $n_2$ (sisemine tsükkel võib loomulikult olla
ka pikkade pesade kaupa)
- - - - - -
- - - - - -
- - - - - -
24

m+p 30 a
m+p+1 25 7777 (lühikeste pesade puhul asendub käsuga
25 3777)
m+p+2 22 k+$\ell$+1.

Kontrollime, kas enne käsku 22 k+$\ell$+1 on taastatud õige tsüklite loendaja sisu. Oletame, et välise tsükli loendaja sisu on 0050. Käsk -20 7777 moodustab seega summaatoris arvu 7727, mis tuleb pöördkoodis liita käsule 25 7777. Saame käsu 25 4050, millega taastatakse tõesti õige tsüklite loendaja sisu.

Kui väline tsükkel on lühikeste pesade kaupa, tuleb pesas k+0 asuv käsk asendada käsuga 25 $n_1$ ja pesas m+p+1 asuv käsk käsuga 25 3777.

Programmi pole raske kohandada ka juhule, kus väline ja sisemine tsükkel asuvad ühes massiivis.

Käskude 25 m ja 24 a abil võib moodustada ka rohkem kui kahekordseid tsükleid. Vastavalt kordsusele komplitseerub aga tsüklite loendaja sisu taastamine.

## 5. Tsüklid loendajate abil.

Et vältida lisaprobleeme, mis tekivad näiteks mitmekordsete tsüklite puhul, on sageli otstarbekohane programmeerida tsükleid mitte käskude 25 m ja 24 a abil, vaid nn. loendajate abil.

Toome selle kohta rida näiteid.

Näide 5. Oletame, et pesades k+0 ÷ k+$\ell$ asuvat programmi osa tuleb teostada n korda. Vastav programm võib olla järgmine

| | | |
|---|---|---|
| k+0 | - - - - - - | korduvalt teostatav programmi osa |
| | - - - - - - | |
| | - - - - - - | |
| k+$\ell$ | - - - - - - | |
| k+$\ell$+1 | 20 0001 | kordumist juhtiv programmi osa. Pesa a on nn. loendaja. Seal peab alguses olema arv (1-n). Et vastav arv paigutatakse lühikesesse pesasse viimasest kahendkohast alates, kirjutatakse see sageli kujul $(1-n).2^{-17}$. |
| k+$\ell$+2 | 01 a | |
| k+$\ell$+3 | 16 a | |
| k+$\ell$+4 | 21 k+0 | |
| k+$\ell$+5 | arvutus läheb edasi. | |

Loendaja sisuga $(1-n)2^{-17}$ teostatakse programmi osa k+0 ÷ k+$\ell$ esimene kord, sisuga $(2-n)2^{-17}$ teine kord, ..., sisuga $(n-n)2^{-17} = -0$ n-is kord. Pärast seda muutub loendaja sisu

positiivseks ja juhtimine antakse pesas k+$\ell$+5 asuvale käsule.

Teine näide loendaja kohta:

```
k+0      - - - - - -  )
         - - - - - -  )  korduvalt teostatav programmi osa
         - - - - - -  )
k+ℓ      - - - - - -  )

k+ℓ+1    20 0001      )
k+ℓ+2    01  a        )  kordumist juhtiv programmi osa.
k+ℓ+3    16  a        )  Pesa a on loendaja. Seal peab al-
k+ℓ+4    14  b        )  guses olema +0, (b) = n·2^-17. Kui
k+ℓ+5    21 k+0       )  (a) = n·2^-17, siis on pärast 14 b
                         (s) = +0 ja tekib ω = 0.
                         Teistel juhtudel on ω = 1.
k+ℓ+6    arvutus läheb edasi.
```

Kui vastavaid programmi osi kasutatakse ülesande lahendamise käigus korduvalt, tuleb loendaja algseis taastada.

Sageli kasutatakse tsüklit loendaja abil koos käsuga 30. Näiteks olgu vaja arvutada funktsiooni f(x) väärtused argumendi väärtuste $x_1, x_2, \ldots, x_n$ korral, kus argumendid pole võrdse sammuga, kuid asuvad järjestikustes pikkades pesades c+4000, c+4002, ..., c+4000+2n-2.

Programm tuleb sellise struktuuriga

```
k+0      30  a       )  uue argumendi väärtuse leidmine
k+1      02 c+4000   )
         - - - - - -  )
         - - - - - -  )  f(x) väärtuse arvutamine
         - - - - - -  )
         - - - - - -  )
k+ℓ      20 0002     )
k+ℓ+1    01  a       )  Kordumist juhtiv programmi osa, mis
k+ℓ+2    16  a       )  on analoogiline teist tüüpi loenda-
k+ℓ+3    14  b       )  jaga.
                        Pesas a on alguses +0, pesas b on
k+ℓ+4    21 k+0      )  2n·2^-17.
```

Esitatud programmi osa korduval kasutamisel tuleb loendaja algseis taastada.

Kaks korda teostatava arvutuskäigu korral võib kasutada järgmist võtet:

| | | |
|---|---|---|
| k+0 | - - - - - - | kaks korda teostatav arvutus |
| | - - - - - - | |
| | - - - - - - | |
| k+$\ell$ | - - - - - - | |
| k+$\ell$+1 | 20 0000 | kordamist juhtiv programmi osa. Loendajas a on alguses positiivne arv. Pärast ühekordset kordamist on seal 0 - (a) = - (a), pärast teistkordset kordamist 0 - (-(a)) = = (a). Signaal $\omega$ = 0 tõttu antakse juhtimine edasi käsule k+$\ell$+5. |
| k+$\ell$+2 | 03 a | |
| k+$\ell$+3 | 16 a | |
| k+$\ell$+4 | 21 k+0 | |
| k+$\ell$+5 | arvutus läheb edasi. | |

Loendaja a algseis taastatakse automaatselt.

Teine võimalus programmi kahekordseks taastamiseks:

| | | |
|---|---|---|
| k+0 | - - - - - - | kaks korda teostatav programmi osa |
| | - - - - - - | |
| | - - - - - - | |
| k+$\ell$ | - - - - - - | |
| k+$\ell$+1 | 02 a | kordamist juhtiv programmi osa. Pesas b on mistahes negatiivne arv või käsk. See ei muutu programmi täitmise käigus. Pesa a algseis peab olema +0. Pärast programmi osa korratava ühekordset täitmist on (a) = (b), pärast teiskordset täitmist (a) = +0 ja signaali $\omega$ = 0 tõttu antakse juhtimine käsule k+$\ell$+5. Loendaja a algseis taastub automaatselt. |
| k+$\ell$+2 | 14 b | |
| k+$\ell$+3 | 16 a | |
| k+$\ell$+4 | 21 k+0 | |
| k+$\ell$+5 | arvutus läheb edasi. | |

## 6. Iteratsioonitsüklid.

Iteratsioonimenetluste korral tekkivaid tsükleid nimetame iteratsioonitsükliteks. Siin ei tea me ette, mitu korda tuleb vastavat menetlust rakendada. Sel korral on aga ette antud hinnangud, mille alusel toimub arvutuskäigu suunamine.

Vaatleme järgmist näidet. Olgu vaja arvutada teatav

piirväärtus

$$x = \lim_{n\to\infty} x_n,$$

kusjuures suurused $x_n$ arvutame suurustest $x_{n-1}$ ja kindlatest konstantidest teatud kindlal viisil, s.t.

$$x_n = f(x_{n-1}), \quad (n = 1,2,\ldots). \qquad (1)$$

Piirväärtuse x asemel võtame tema lähisväärtuse $x_n$, kui on täidetud tingimus

$$|x_n - x_{n-1}| \leq \varepsilon$$

ehk

$$|x_n - x_{n-1}| - \varepsilon \leq -0. \qquad (2)$$

Kui tingimus (2) pole täidetud, arvutame valemist (1) suuruse $x_{n+1}$ jne.

Vastav programm on järgmine:

| | | |
|---|---|---|
| k+0 | - - - - - - | } valemi f(x) järgi arvutamine (itereerimine) |
| | - - - - - - | |
| | - - - - - - | |
| k+$\ell$ | - - - - - - | |
| k+$\ell$+1 | 02 a+4002 | (a+4000) = $x_{n-1}$ |
| k+$\ell$+2 | 03 a+4000 | (a+4002) = $x_n$ |
| k+$\ell$+3 | 04 a+4004 | (a+4004) = $\varepsilon$ |
| k+$\ell$+4 | 21 k+4000 | |
| k+$\ell$+5 | arvutus läheb edasi. | |

## 7. Loogilised skaalad.

Esineb ülesandeid, kus arvutusi on tarvis suunata programmi ühe või teise osa juurde ette teadaoleval viisil. Sel juhul saab neid suunamisi teostada nn. loogiliste skaalade abil. Ühes pesas asub skaala, näiteks

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | ... | | 0 |

skaala

Vastavalt skaala "1"-dele ja "0"-dele antakse juhtimine kas ühele või teisele programmi osale. Skaala viimane kahendkoht peab seejuures tingimata olema "1", sest muidu pole võimalik kindlaks teha töö lõppu (töö on lõppenud, kui skaala parempoolne osa on +0).

Arvutuskäigu juhtimist skaala abil võib programmeerida näiteks järgmiselt:

| | | |
|---|---|---|
| k+0 | 20 0001 | $(s) = +1 \cdot 2^{-17}$ |
| k+1 | 17 c | pesas c asub skaala |
| k+2 | 11 0000 | |
| k+3 | 21 k+$\ell$ | kui pärast nihutamist on (s) = +0, on skaala töö lõppenud ja juhtimine läheb väljapoole. |
| k+4 | 16 c | |
| k+5 | 21 k+m | kui skaala 1. kohal on 1, antakse juhtimine pessa k+m, kui 1. kohal on 0, läbib juhtimine pesa k+6. |
| k+6 | - - - - - | |

Vastava arvutuskäigu kordamisel tuleb taastada ka skaala algseis.

## § 19. Konstantsete mastaapide valikust.

Teatavasti sooritab arvutusmasin "Ural-1" aritmeetilisi tehteid (koodid 01, 03, 04, 05, 06 ja 07) arvudega vahemikus $|x| < 1$. Enamikul ülesannetest on aga kas osa algandmeid, vahe- või lõpptulemusi väljaspool seda vahemikku. Et selliseid ülesandeid lahendada arvutusmasinal "Ural-1", tuleb kasutada erivõtteid aritmeetiliste tehete taandamiseks arvudele vahemikus $|x| < 1$. Nendeks võteteks on:

1. Konstantsete mastaapide valimine ülesandes esinevatele suurustele, s.t. arvude $x$ esitamine kujul

$\mathfrak{X} = m\xi$ , ehk $\frac{\mathfrak{X}}{m} = \xi$ (m - mastaabikordaja),

kus $|\xi| < 1$.

2. Muutuvate mastaapide valimine, kus mastaabikordajaid korrigeeritakse automaatselt arvutuste käigus.

3. Programmeerimine nn. liikuva koma režiimis, kus arvud esitatakse kujul

$$\mathfrak{X} = m_x \cdot 2^{p_x}, \quad |m_x| < 1$$

(arvu mantiss $m_x$ asub pikas, järk $p_x$ lühikeses pesas).

4. Programmeerimine nn. transformeeritud koma režiimis, kus kõikidele arvudele $(-\infty < \mathfrak{X} < +\infty)$ seatakse vastavusse nende transformatsioonid

$$x = f(\mathfrak{X}), \quad |f(\mathfrak{X})| < 1.$$

Konstantsete mastaapide meetod on meetodite 1.-4. hulgas kõige lihtsam, kuid selle kasutamisel peab eelnevalt olema võimalik hinnata kõikide suuruste muutumisvahemikke. Peale selle peab mastaapide valik tagama arvutuste nõutud täpsuse. Asi seisneb nimelt selles, et suurte mastaabikordajate kasutamine vähendab arvude tüvenumbrikohti ja seega arvutuste täpsust.

Kui mingi ülesande puhul pole ülaltoodud nõudeid võimalik täita, tuleb kasutada teisi meetodeid 2.-4.

Vaatleme konstantsete mastaapide valikuga seotud probleeme näidete varal. Kõigepealt selgitame tehete järjekorra tähtsust üle- ja alatäitumise (resp. täpsuse kao) seisukohast.

Näide 1. Olgu vaja arvutada

$$z = \frac{u\, v\, 2^{-17}}{w},$$

kus argumendid u, v ja w rahuldavad tingimusi $2^{-6} < |u| < 1$, $2^{-6} < |v| < 1$, $2^{-17} < |w| < 1$.

Kergesti leiame, et

$$2^{-29} < |z| < 1,$$

kuid pole kaugeltki ükskõik, missuguses järjekorras tehteid teostada.

Vaatleme kolme juhtumit:

1) $z = (u \cdot v \cdot 2^{-17}) : w$,

2) $z = [(u \cdot v) : w] \cdot 2^{-17}$,

3) $z = u \cdot v \cdot (2^{-17} : w)$.

Juhtum 1). Kui u ja v on oma alumise piiri lähedal, siis $u \cdot v \cdot 2^{-17} \approx 2^{-29}$. Et pesas on 35 numbrikohta, saame tulemuses 7 tüvenumbrit. Jagamisel arvuga w ei kasva õigete tüvenumbrite arv. Seejuures on kõik argumendid antud vähemalt 19 tüvenumbriga ja tulemuse võiks saada samuti 19 tüvenumbriga. Esineb täpsuse kadu 12 2-ndkohta.

Juhtum 2). Kui u ja v on oma ülemise piiri, w aga oma alumise piiri lähedal, tuleb ületäitumine $(u \cdot v) : w \approx \approx \frac{1}{2^{-17}} = 2^{17}$.

Juhtum 3). Kergesti leiame, et $2^{-12} < |u \cdot v| < 1$ ja $2^{-17} < |2^{-17} : w| < 1$. Suuruste $u \cdot v$ ja $(2^{-17} : w)$ korrutamisel saame z. Arvutamisel ei teki ületäitumist ega täpsuse kadu. Seega on antud tehete järjekord parim.

Näide 2. Olgu vaja arvutada

$$z = \frac{u \cdot v}{w},$$

kus argumendid u, v ja w rahuldavad tingimusi $2^{-15} < |u| < 2^{-8}$, $2^{-15} < |v| < 2^{-8}$, $2^{-10} < |w| < 1$.

Kergesti leiame, et

$$2^{-30} < |z| < 2^{-6}.$$

Vaatleme kolme juhtumit:

1) $z = (u : w)\, v$ ,

2) $z = (v : w)\, u$ ,

3) $z = (u\, v) : w$.

Juhtumitel 1) ja 2) tekib ületäitumine, kui u ja v on oma ülemise, w oma alumise piiri lähedal.

Juhtumil 3) ei teki küll ületäitumist, kuid sellisel viisil arvutades läheb kaduma osa tüvenumbreid. Tõepoolest, kui näiteks $u \approx 2^{-15}$, $v \approx 2^{-8}$ ja $w \approx 2^{-10}$ , saame $z \approx 2^{-13}$. Tulemuses on 23 tüvenumbrit. Et argumendid on antud vähemalt 21 tüvenumbriga, siis peaks tulemuses olema samuti 21 õiget tüvenumbrit. Arvutustes esineb aga korrutis $u\, v \approx 2^{-23}$, kus on ainult 13 tüvenumbrit. Seetõttu ei saa ka lõppetulemuses olla rohkem kui 13 õiget tüvenumbrit. Esineb täpsuse kadu 8 tüvenumbrit.

Paremaid tulemusi annab arvutamine valemi

$$z = \left[(2^{-2}\, u) : w\right] (v : 2^{-2}) \qquad (1)$$

järgi. Siin ületäitumist ei teki , kuid võib siiski tekkida kahekohaline täpsuse kadu. Tõepoolest, kui $u \approx 2^{-15}$ (21 tüvenumbrit), $v \approx 2^{-8}$ (28 tüvenumbrit), $w \approx 2^{-10}$ (26 tüvenumbrit), saame $z \approx 2^{-13}$ (23 tüvenumbrit). Õigete tüvenumbrite arv on seega 21. Kuna arvutusvalemis (1) esineb korrutis $2^{-2}u$, saame antud juhul $2^{-2} \cdot 2^{-15} = 2^{-17}$, kus on ainult 19 tüvenumbrit. Seega ei saa ka lõpptulemuses olla üle 19 õige tüvenumbri.

Suurust z saaks arvutada aga ilma täpsuse kaota. Kuid siis tuleks analüüsida, kas $2^2$ w annab ületäitumise või mitte. Arvutuskäik tuleks sel korral järgmine:

$$z = \begin{cases} [u : (2^2 w)] \cdot (2^2 v), & \text{kui } 2^2 |w| < 1 \\ (u : w)\, v, & \text{kui } 2^2 |w| \geq 1, \text{ s.t. } |w| \geq 2^{-2}. \end{cases}$$

Ülaltoodud arutelust nähtub, et mõningatel juhtudel (nõutud täpsuse tagamiseks) on otstarbekohane kasutada nn. vahepealseid mastaape (antud näites suurused $2^{-2}$ ja $2^2$).

Märkus: Kui valemis esineb ainult korrutamis- ja jagamistehe, siis lõpptulemuse õigete tüvenumbrite maksimaalne võimalik arv on miinimum argumentide ja lõpptulemuse enese tüvenumbrite arvust. Arvutamisel võib aga esineda olukordi, mis seda maksimaalset võimalikku õigete tüvenumbrite arvu vähendab (nagu näiteks valemi (1) puhul). Arvutamisel saadav õigete tüvenumbrite arv on miinimum argumentide, kõikide vahetulemuste ja lõpptulemuse tüvenumbrite arvust.

Näide 3. Leiame 10-ndmastaabid ($m = 10^n$) funktsiooni

$$y = \frac{\sqrt{(931 + 9x^2)x - 41}}{0{,}3x + 8} \qquad (2)$$

väärtuste arvutamiseks, kui argument x on vahemikus [1; 7].

Kerge on kontrollida, et $y'(x)$ on lõigul [1; 7] positiivne, s.t. y on vastavas piirkonnas kasvav funktsioon. Seega saame, et $y_{min} = y(1) \approx -1{,}246$ ja $y_{max} = y(7) \approx 5{,}644$. Järelikult sobib nii x kui ka y mastaabikordajaks 10:

$$y = 10\eta\,, \qquad x = 10\xi\,.$$

Asendades valemisse (2) saame

$$10\eta = \frac{\sqrt{(931 + 900\xi^2)\,10\xi - 41}}{3\xi + 8},$$

millest

$$\eta = \frac{\sqrt{(93{,}1 + 90\,\xi^2)\,\xi} - 4{,}1}{3\,\xi + 8} = \frac{\sqrt{(0{,}931 + 0{,}9\,\xi^2)\,\xi} - 0{,}41}{0{,}3\,\xi + 0{,}8}\,.$$

Kuigi kõik arvud on viimases avaldises ühest väiksemad, võib tekkida ületäitumine vahetulemustes. Juurealuse avaldise puhul ületäitumist ei esine: $(0{,}931 + 0{,}9\,\xi^2)\xi$ maksimaalne väärtus ($\xi = 0{,}7$ korral) on 0,9604. Kuid $0{,}931 + 0{,}9\,\xi^2$ maksimaalne väärtus on 1,372, nimetaja $0{,}3\xi + 0{,}8$ maksimaalne väärtus on 1,01. Ületäitumise kõrvaldamisel viimase kahe avaldise puhul jagame lugeja ja nimetaja kahega, millega valem omandab lõpliku kuju

$$\eta = \frac{\sqrt{(0{,}23275 + 0{,}225\,\xi^2)\,\xi} - 0{,}205}{0{,}15\,\xi + 0{,}4} \qquad (3)$$

Valemi (3) kasutamisel tuleb argument masinasse viia 10 korda vähendatult ning masinast saadavat tulemust tõelise vastuse $y$ leidmiseks korrutada kümnega.

Näide 4. Leiame mastaabikordajad diferentsiaalvõrrandi

$$y' = \frac{10}{x^2 + y^2} + x \qquad (4)$$

erilahendi arvutamiseks lõigul $x \in [1;\ 10]$, kus erilahend peab rahuldama algtingimust

$$y_{x=1} = 1.$$

Argumendi $x$ muutumise sammuks võtame $h = 0{,}01$. Diferentsiaalvõrrandi erilahendi leiame Euleri-Cauchy meetodi abil. Teatavasti tuleb diferentsiaalvõrrandi $y' = f(x,y)$ lahendamisel Euleri-Cauchy meetodiga teostada arvutusi järgmiste valemite järgi:

$$y'_i = f(x_i,\ y_i), \qquad (5)$$

$$\bar{y}_{i+1} = y_i + h\, y'_i, \tag{6}$$

$$x_{i+1} = x_i + h, \tag{7}$$

$$\bar{y}'_{i+1} = f(x_{i+1}, \bar{y}_{i+1}), \tag{8}$$

$$\theta_i = \tfrac{1}{2}(y'_i + y'_{i+1}), \tag{9}$$

$$y_{i+1} = y_i + h\,\theta_i, \tag{10}$$

kus i = 0,1, ... , n-1. Antud näite puhul on $x_0 = 1$, $x_n = 10$, $y_0 = 1$ ja $n = \frac{x_n - x_0}{h} = \frac{9}{0{,}01} = 900$.

Mastaabikordajad tuleb leida nii, et kõik ülesandes esinevad suurused ja arvutuste vahetulemused oleksid absoluutselt ühest väiksemad.

Hindame ülesandes esinevate suuruste muutumispiirkondi. Ülesande formulatsioonist järeldub, et

$$1 \leqslant x \leqslant 10 \tag{11}$$

Hindame y' muutumispiirkonda.

$$y' = \frac{10}{x^2 + y^2} + x \leqslant \frac{10}{x^2} + x.$$

Funktsioonil $\frac{10}{x^2} + x$ on aga maksimum (väärtusega 11) punktis $x = 1$. Et $y' > x$, siis saame, et

$$1 < y' < 11. \tag{12}$$

Suuruse y muutumispiirkonna leidmisel lähtume võrratusest

$$y' \leqslant \frac{10}{x^2} + x$$

ja integreerime x järgi rajades 1 kuni x-ni. Saame

$$\int_1^x y'\,dx \leqslant \int_1^x \frac{10}{x^2}\,dx + \int_1^x x\,dx,$$

millest

$$y(x) - y(1) \leqslant -\frac{10}{x}\Big|_1^x + \frac{x^2}{2}\Big|_1^x =$$

$$= -\frac{10}{x} + 10 + \frac{x^2}{2} - \frac{1}{2} = -\frac{10}{x} + \frac{x^2}{2} + 9{,}5.$$

Arvestades, et $y(1) = 1$, saame

$$y(x) \leqslant -\frac{10}{x} + \frac{x^2}{2} + 10{,}5.$$

Viimase võrratuse parem pool on maksimaalne $x = 10$ korral. Seega saame

$$1 \leqslant y \leqslant 59{,}5. \tag{13}$$

Tuues sisse mastaabikordajad $m$ ja $k$:

$$x = m\xi \quad , \qquad y' = k\eta' \quad ,$$
$$y = m\eta, \quad \text{ja} \quad \theta = k\bar{\theta} \tag{14}$$

Võime öelda, et need peavad rahuldama tingimusi

$$m \geqslant 60, \tag{15}$$

$$k > 11 \tag{16}$$

Kontrollime, kas arvutuste käigus tekib ületäitumine. Valemist (4) saame peale asendusi (14):

$$\eta' = \frac{\frac{10}{m^2 k}}{\xi^2 + \eta^2} + \frac{m}{k}\xi. \tag{17}$$

1) Arvestades, et peab kehtima $\left|\frac{m}{k}\xi\right| < 1$, kusjuures kehtib $|\xi| < 1$, peab kehtima $\frac{m}{k} \leqslant 1$. Siit saame tingimuse

$$m \leqslant k \, . \tag{18}$$

2) $\frac{10}{m^2 k} < 1$ kehtivusega on samaväärne $m^2 k > 10$. Viimane võrratus kehtib (15) ja (16) tõttu.

3) Peab kehtima

$$\frac{\frac{10}{m^2 k}}{\xi^2 + \eta^2} < 1 \quad \text{ehk} \quad \frac{10}{km^2(\xi^2 + \eta^2)} < 1 \quad \text{ehk} \quad \frac{10}{k(x^2 + y^2)} < 1.$$

Et $x^2 + y^2$ minimaalne väärtus on 2 ($1^2 + 1^2 = 2$), siis kehtivad toodud võrratused juba $k > 5$ korral, ammugi (16) korral.

Valem (6) teisendub kujule

$$\bar{\eta}_{i+1} = \eta_i + \frac{k\,h}{m}\eta'_i \,.$$

millest saame, et $\frac{k\,h}{m} < 1$ ehk

$$k\,h < m. \qquad (19)$$

Mastaabikordajad, mis rahuldavad tingimusi (15), (16), (18) ja (19), võimaldavad arvutuste teostamist valemite (5) - (10) järgi ilma ületäitumisteta. Mastaabikordajateks on kõige mugavam võtta 10 astmed, nii et vastavad tingimused (15),(16), (18) ja (19) oleksid rahuldatud. Sel juhtumil oleksid tõelised tulemused leitavad arvutatavatest koma nihutamise teel. Kuid mastaabikordajate suurendamine vähendab arvutuste täpsust. Seepärast loobutakse mõnigi kord kümnendmastaapidest, kasutades sel juhul 2 astmeid.

Võtamegi antud näites $m = k = 2^6 = 64$. Tingimused (15), (16), (18) ja (19) on täidetud. Teisendusvalemid (14) tulevad

$$x = 2^6\xi\,, \qquad y' = 2^6\eta'\,, \qquad (20)$$
$$y = 2^6\eta\,, \qquad \theta = 2^6\bar{\theta}\,.$$

Argumendi $\xi$ muutumispiirkonna saame

$$\frac{1}{2^6} \leqslant \xi \leqslant \frac{10}{2^6} = \frac{5}{2^5}\,.$$

Algtingimus tuleb

$$\eta_{\xi=2^{-6}} = 2^{-6}\,.$$

Arvutusvalemid (5)-(10) omandavad järgmise kuju:

$$\eta'_i = \frac{\frac{5}{2^n}}{\xi_i^2 + \eta_i^2} + \xi_i\,, \qquad (5')$$

$$\eta_{i+1} = \eta_i + 0{,}01\,\eta_i', \qquad (6')$$

$$\xi_{i+1} = \xi_i + \frac{0{,}01}{2^6}, \qquad (7')$$

$$\bar{\eta}'_{i+1} = \frac{\frac{5}{2^{\pi}}}{\xi^2_{i+1} + \bar{\eta}^2_{i+1}} + \xi_{i+1}, \qquad (8')$$

$$\bar{\theta}_i = \tfrac{1}{2}\eta'_i + \tfrac{1}{2}\bar{\eta}'_{i+1}, \qquad (9')$$

$$\eta_{i+1} = \eta_i + 0{,}01\,\bar{\theta}_i. \qquad (10')$$

## § 20. Muutuvate mastaapide meetod. Programmeerimine liikuva koma meetodil.

Kui ülesandes esinevad suurused muutuvad piirides, mida on raske ette määrata, või väga laias diapasoonis, siis pole konstantsete mastaapide kasutamine võimalik. Esimesel juhul pole konstantseid mastaape lihtsalt võimalik määrata, teisel juhul võib konstantsete mastaapide kasutamine tekitada sellise täpsuse kao et arvutustel kaob mõte.

Näiteks, kui suuruse $x$ väärtused on

$$x_1 = 0{,}000\ 015\ 338; \quad x_2 = 0{,}000\ 020\ 779; \ \ldots;$$

$$x_n = 8870{,}2,$$

siis tuleb konstantne (kümnendmastaap) valida

$$m_x = 10^4.$$

Eeldades, et pesas saab kujutada 9-kohalisi arve, tulevad vastavad väärtused pärast mastabeerimist

$$\xi_1 = 0{,}000\ 000\ 002; \quad \xi_2 = 0{,}000\ 000\ 002; \ \ldots;$$

$$\xi_n = 0{,}887\ 020\ 000,$$

millega läheb kaduma arvutuste täpsus ( $\xi_1$ ja $\xi_2$ on antud

ühe tüvenumbriga).

Sellistel juhtudel tuleb kasutada kas muutuvate mastaapide meetodit, programmeerimist liikuva koma või nn. transformeeritud koma meetodil (süsteemis).

Vaatleme kõigepealt muutuvate mastaapide meetodit. Suurused $\mathfrak{X}$, $\mathfrak{Y}$, ... esitatakse siin kujul

$$\mathfrak{X} = m_{\mathfrak{X}}\, x, \qquad \mathfrak{Y} = m_{\mathfrak{Y}}\, y, \quad \ldots .$$

Mastaabid $m_{\mathfrak{X}}$, $m_{\mathfrak{Y}}$ valitakse mingi arvu, tavaliselt kas 10 või 2 astmetena: s.t. kas $m_{\mathfrak{X}} = 10^p$ või $m_{\mathfrak{Y}} = 2^q$ jne. Arve x ja y nimetatakse arvude $\mathfrak{X}$ ja $\mathfrak{Y}$ mantissideks, arve p ja q aga nende arvude järkudeks.

Nullist erineva arvu kümnendmastaapide korral valitakse järk nii, et mantiss rahuldaks võrratust

$$\frac{1}{16} \leqslant |x| < 1, \tag{1}$$

kahendmastaapide korral aga võrratust

$$\tfrac{1}{2} \leqslant |x| < 1 . \tag{2}$$

Mantissi absoluutväärtuste alumised tõkked võrratustes (1) ja (2) on tingitud täpsuse nõuetest: need ei võimalda (suurt) alatäitumist. Põhimõtteliselt võivad alumised tõkked olla ka teistsugused. Kuid alumine tõke $\beta$ peab rahuldama nõuet: kui $|x| < \beta$, siis mantissi suurendamine kümnendmastaabi puhul 10 korda (kahendmastaabi puhul vastavalt 2 korda) ei tohi tekitada ületäitumist. Seega saame, et kümnendmastaabi puhul $\beta \leqslant \frac{1}{10}$, kahendmastaabi puhul $\beta \leqslant \frac{1}{2}$.

Vaatleme nüüd näite varal, kuidas tuleb programmis ette näha mastaapide automaatset muutumist.

Näide 1. Olgu vaja arvutada

$$Z = X + Y .$$

Asendades suurused $X$, $Y$, $Z$ mastaabikordajate ja mantissi-de abil, saame

$$m_z z = m_x x + m_y y$$

ehk

$$z = \frac{m_x}{m_z} x + \frac{m_y}{m_z} y .$$

Mastaabikordajate muutmise eesmärgil võtame kasutusele ühest väiksemad suurused A, B, C ja D, nii et

$$\frac{m_x}{m_z} = \frac{A}{C} , \quad \frac{m_y}{m_z} = \frac{B}{D} .$$

Seega saame

$$z = \frac{A}{C} x + \frac{B}{D} y . \qquad (3)$$

Eeldades, et meil on tegemist näiteks kümnendmastaapi-dega, vaatleme üksikuid mastaapide muutmise juhtumeid. Mastaapide algväärtused leitakse nagu konstantsete mastaapide meetodi puhul.

Muutuvate mastaapide kasutamisel tuleb välja lülitada tumblerid "$\varphi$ blokeerimine" ja "peatus $\varphi$ järgi". Ületäitumise tagajärgi võib likvideerida kas juhtimise üleandmisega pessa 0000 või ühe käsu vahelejätmisega. Nimelt anname vastavasse pessa viidud käsu 22 a juhtimise ületäitumist likvideerivale programmi osale.

Juhtum 1. Mantissi $x$ arvutamisel tekkis ületäitumine. Signaali $\varphi = 1$ abil antakse juhtimine edasi programmi osale, mis selle ületäitumise likvideerib. Teiskordselt arvu-tatav $x$ tuleb 10 korda väiksem eelmisest, kümnendjärku $p_x$ aga suurendatakse 1 võrra. Kordajat C korrutatakse arvuga

$\frac{1}{10}$ . Kui y rahuldab tingimust (1), hakatakse arvutama valemi (3) järgi.

Analoogiliselt toimitakse siis, kui ületäitumine tekkis y arvutamisel.

Juhtum 2. Mantissi x puhul pole täidetud tingimus $\frac{1}{16} \leqslant |x|$. Sel juhul tuleb $p_x$ vähendada 1 võrra ja x uuesti arvutada. Korrutaja A tuleb korrutada arvuga $\frac{1}{10}$ . Kui y rahuldab tingimust (1), hakatakse arvutama valemi (3) järgi.

Juhtum 3. Mantissid x ja y rahuldavad tingimust (1), kuid z arvutamisel tekkis ületäitumine. Signaali $\varphi = 1$ abil antakse juhtimine edasi programmi osale, mis suureneb $p_z$ ühe võrra ja korrutab kordajad A ja B arvuga $\frac{1}{10}$ . Pärast seda hakatakse arvutama uuesti valemi (3) järgi.

Juhtum 4. Mantissid x ja y rahuldavad tingimust (1), kuid z puhul pole täidetud tingimus $\frac{1}{16} \leqslant |z|$. Sel juhul vähendatakse $p_z$ ühe võrra ja korrutatakse kordajad C ja D arvuga $\frac{1}{10}$. Pärast seda arvutatakse z uuesti valemist (3).

Tuleb märkida, et kui mingi suuruse järku muudetakse arvutuste käigus, siis tuleb vastaval viisil muuta ka kõiki arvutuses esinevaid kordajaid A, B, C, D, ... , mis sõltuvad sellest järgust. Näiteks valemis (3) sõltuvad järgust $p_x$ kordajad A ja C, järgust $p_y$ kordajad B, D ja järgust $p_z$ kordajad A, B, C, D.

Kordajate muutmise vajaduse arvestamine teeb aga programmeerimise võrdlemisi komplitseerituks, mistõttu seda meetodit kasutatakse harva. Kui ülesande programmeerimisel ei ole võimalik kasutada konstantsete mastaapide meetodit, siis kasuta-

takse enamasti liikuva koma meetodit.

Liikuva koma meetodi kasutamisel tuleb arvud X, Y, Z,... esitada kujul

$$\mathcal{X} = m_x \cdot 2^{p_x}, \quad \mathcal{Y} = m_y \cdot 2^{p_y}, \quad \mathcal{Z} = m_z \cdot 2^{p_z}, \ldots$$

(mantissid salvestatakse pikkades, järgud lühikestes pesades), kus nullist erinevate arvude mantissid peavad olema normaliseeritud.

Aritmeetilisi tehteid: korrutamist, jagamist ja liitmist saame teostada, sooritades tehteid vastavate arvude mantisside ja järkudega valemite

$$\mathcal{Z} = \mathcal{X} \cdot \mathcal{Y} = m_x \cdot m_y \cdot 2^{p_x+p_y}, \tag{4}$$

$$\mathcal{Z} = \mathcal{X} : \mathcal{Y} = \frac{m_x}{m_y} 2^{p_x-p_y}, \tag{5}$$

$$\mathcal{Z} = \mathcal{X} + \mathcal{Y} = \begin{cases} (m_x + m_y \cdot 2^{p_y-p_x})\, 2^{p_x}, & \text{kui } p_x > p_y, \\ (m_x \cdot 2^{p_x-p_y} + m_y)\, 2^{p_y}, & \text{kui } p_x \leqslant p_y, \end{cases} \tag{6}$$

järgi.

Valemite (5) ja (6) kasutamisel võib tekkida aga ületäitumine (ületäitumise puhul on tulemus siiski alati väiksem kui 2). Ületäitumise korral oleneb arvutuskäik aga $\varphi$ režiimist. Kui tumbler "$\varphi$ blokeerimine" on sisse lülitatud, saame ületäitumise ($|(s)|<2!$) korral normaliseerimisega ikka õige tulemuse. Kui aga ületäitumise korral masin seisma jääb (peatus $\varphi$ järgi) või antakse juhtimine pessa 0000 või jäetakse vahele üks käsk (juhtimise üleandmine $\varphi$ järgi), siis rikub see kogu arvutuskäigu. Et mõnikord on aga kasulik mõned programmi osad programmeerida näiteks konstantsete mastaapide

meetodil (kus ületäitumise korral jääb masin seisma), siis on otstarbekohane arvutusvalemid (5) ja (6) muuta sõltumatuks $\varphi$ režiimist. Kordaja $\frac{1}{2}$ kasutamisega saamegi valemites (5) ja (6) ületäitumise kõrvaldada. Seega

$$Z = X : Y = \frac{\frac{1}{2} m_x 2^{p_x - p_y + 1}}{m_y} \qquad (7)$$

ja

$$Z = X + Y = \begin{cases} (\frac{1}{2} m_x + \frac{1}{2} m_y \cdot 2^{p_y - p_x})\, 2^{p_x + 1} & \text{, kui } p_x > p_y, \\ (\frac{1}{2} m_x \cdot 2^{p_x - p_y} + \frac{1}{2} m_y)\, 2^{p_y + 1} & \text{, kui } p_x \leqslant p_y \cdot \end{cases} \qquad (8)$$

Kui kasutame $\varphi$ blokeerimise režiimi, siis võib muidugi arvutada valemite (5) ja (6) järgi.

Esitame nüüd programmid, mis teostavad arvutusi mantisside ja järkudega valemite (4), (7) ja (8) järgi. Programmide põhimõttelisel üleskirjutamisel on otstarbekohane kasutada sümbolit $\langle\ \rangle$. Kirjutise $\langle A \rangle$ abil tähistame selle pesa aadressi, kus asub arv $A$. Kui arv $A$ asub pesas a + 4000, siis $\langle A \rangle$ = a + 4000. Viimane kirjutis on samaväärne kirjutisega (a + 4000) = $A$, s.t. pesa a + 4000 sisuks on arv $A$.

Korrutamise programm tuleb järgmine (valem (4)).

```
k + 0   02   <m_x>
k + 1   06   <m_y>
k + 2   15   <m_z>   Korrutamisel võib tekkida ala-
k + 3   01   <p_x>   täitumine, seega tuleb tulemus
k + 4   01   <p_y>   normaliseerida
k + 5   16   <p_z>
```

Siin võiks lisada ainult seda, et mantisside korrutis viiakse normaliseerimiskäsuga tulemuse mantissi pessa, tegu-

rite järkude summa ühes parandusega, mis tuleb normaliseerimisest, tulemuse järgu pessa.

Jagamise programm tuleb järgmine (valem (7)):

| | | | |
|---|---|---|---|
| k + 0 | 02 | $\langle m_x \rangle$ | |
| k + 1 | 06 | $\langle \frac{1}{2} \rangle$ | mantissi $m_x$ korrutamine arvuga $\frac{1}{2}$ |
| k + 2 | 07 | $\langle m_y \rangle$ | |
| k + 3 | 15 | $\langle m_z \rangle$ | |
| k + 4 | 01 | $\langle p_x \rangle$ | |
| k + 5 | 03 | $\langle p_y \rangle$ | |
| k + 6 | 01 | $\langle 000001 \rangle$ | järkude vahele $p_x$-$p_y$ arvu 1 liitmine |
| k + 7 | 16 | $\langle p_z \rangle$ | |

Enne jagamisele asumist tuleb kontrollida (17 abil), kas $m_y \neq 0$. Blokeeritud $\varphi$ korral võib programmist käsud 06 $\langle \frac{1}{2} \rangle$ ja 01 $\langle 000001 \rangle$ välja jätta.

Liitmise programm tuleb järgmine (valem (8)):

| | | | | |
|---|---|---|---|---|
| | k + 0 | 17 | $\langle m_x \rangle$ | |
| | k + 1 | 21 | k+33 | kui $m_x = \pm 0$, siis $\omega = 1$ |
| | k + 2 | 17 | $\langle m_y \rangle$ | |
| | k + 3 | 21 | k+37 | kui $m_y = \pm 0$, siis $\omega = 1$ |
| | k + 4 | 02 | $\langle p_x \rangle$ | |
| | k + 5 | 03 | $\langle p_y \rangle$ | (s) = $p_x$-$p_y$ |
| | k + 6 | 21 | k+20 | kui $p_x > p_y$, siis $\omega = 0$ |
| ($p_x > p_y$) | k + 7 | 10 | $\langle -\alpha \rangle$ | $\alpha > 0$, (s) = $p_y$-$p_x$ |
| | k + 10 | 17 | $\langle \frac{1}{2} \rangle$ | |
| | k + 11 | 11 | 0000 | (s) = $\frac{1}{2} \cdot 2^{p_y - p_x}$ |
| | k + 12 | 06 | $\langle m_y \rangle$ | |
| | k + 13 | 17 | $\langle \frac{1}{2} \rangle$ | |
| | k + 14 | 05 | $\langle m_x \rangle$ | |

| | | | | |
|---|---|---|---|---|
| | k + 15 | 15 | $\langle m_z \rangle$ | |
| | k + 16 | 01 | $\langle p_x \rangle$ | |
| | k + 17 | 22 | k+27 | |
| $(p_x \leqslant p_y)$ | k + 20 | 17 | $\langle \frac{1}{2} \rangle$ | (s) $= p_x - p_y \leqslant -0$ |
| | k + 21 | 11 | 0000 | (s) $= \frac{1}{2} \cdot 2^{p_x - p_y}$ |
| | k + 22 | 06 | $\langle m_x \rangle$ | |
| | k + 23 | 17 | $\langle \frac{1}{2} \rangle$ | |
| | k + 24 | 05 | $\langle m_y \rangle$ | |
| | k + 25 | 15 | $\langle m_z \rangle$ | |
| | k + 26 | 01 | $\langle p_y \rangle$ | |
| | k + 27 | 01 | 000001 | |
| | k + 30 | 16 | $\langle p_z \rangle$ | |
| | k + 31 | 30 | $\lambda$ | nende käskude abil toimub |
| | k + 32 | 22 | 0000 | väljasuunamine liitmisprogrammist |
| | k + 33 | 02 | $\langle m_y \rangle$ | kui $m_x = \pm 0$ |
| | k + 34 | 16 | $\langle m_z \rangle$ | |
| | k + 35 | 02 | $\langle p_y \rangle$ | |
| | k + 36 | 22 | k+30 | |
| | k + 37 | 02 | $\langle m_x \rangle$ | kui $m_y = \pm 0$ |
| | k + 40 | 16 | $\langle m_z \rangle$ | |
| | k + 41 | 02 | $\langle p_x \rangle$ | |
| | k + 42 | 22 | k+30 | |

Lugejatel koostada liitmisprogramm, eeldusel, et $\varphi$ on blokeeritud ja masinal on olemas käsk 11 4000.

Lahutamisel $x - y$ muudame $y$ märgi (s.t. mantissi $m_y$ märgi) vastupidiseks ja liidame.

Mõni sõna nende programmide kasutamise kohta. Neid programme võiks kasutada selliselt, et iga kord, kui on vaja korrutada, jagada või liita, viime algandmed vastava programmi osa spetsiaalsetesse algandmete pesadesse $m_1$, $p_1$, $m_2$, $p_2$

02 $\langle m_x \rangle$
16 $\langle m_1 \rangle$
02 $\langle p_x \rangle$
16 $\langle p_1 \rangle$
02 $\langle m_y \rangle$
16 $\langle m_2 \rangle$
02 $\langle p_y \rangle$
16 $\langle p_2 \rangle$

ja valmistame ette tagasipöördumise õigesse pessa (näiteks pesa b) põhiprogrammis:

20 b
16 $\lambda$

(programmi osadest väljatulek toimub sel juhul käskude 30 $\lambda$ ja 22 0000 abil). Selle järel tuleb anda juhtimine programmi osa esimesele käsule

22 k+0 .

Kogu ettevalmistus vajab seega 11 käsku. Et korrutamise ja jagamise vahetu teostamine nõuab ainult 6 + 8 käsku, siis on ilmselt kasulikum iga korrutamise ja jagamise puhul vastav programmi osa programmis uuesti kirja panna. Liitmise ja lahutamise puhul on aga otstarbekohasem kasutada eespool kirjeldatud 11-käsulist ettevalmistavat programmi osa.

Selgitame näite varal ülesande programmeerimist liikuva koma meetodil.

Näide 2. Koostame programmi suuruse

$$y = \frac{x^2 - Ax}{x + B} = \frac{(x - A)x}{x + B}$$

arvutamiseks.

Algandmed olgu paigutatud pesadesse:

$m_x = (a + 4000)$, $p_x = (a + 0006)$,

$m_A = (a + 4002)$, $p_A = (a + 0007)$,

$m_B = (a + 4004)$, $p_B = (a + 0010)$.

Liitmisprogrammi spetsiaalsed pesad olgu algandmete jaoks

$m_1 = (b + 4000)$, $p_1 = (b + 0006)$

$m_2 = (b + 4002)$, $p_2 = (b + 0007)$

ja summa jaoks[15]

$m_3 = (b + 4004)$, $p_3 = (b + 0010)$.

Liitmisprogramm paiknegu pesades k+0 ÷ k+42.

| | | | |
|---|---|---|---|
| $\ell + 0$ | 02 | a + 4000 | liitmise ettevalmistamine: algandmete viimine spetsiaalsetesse pesadesse |
| $\ell + 1$ | 16 | b + 4000 | |
| $\ell + 2$ | 02 | a + 0006 | |
| $\ell + 3$ | 16 | b + 0006 | |
| $\ell + 4$ | 02 | a + 4004 | |
| $\ell + 5$ | 16 | b + 4002 | |
| $\ell + 6$ | 02 | a + 0010 | |
| $\ell + 7$ | 16 | b + 0007 | |
| $\ell + 10$ | 20 | $\ell + 13$ | tagasipöördumise ettevalmistamine |
| $\ell + 11$ | 16 | $\lambda$ | |
| $\ell + 12$ | 22 | k+0 | |

[15] Tähendame, et summa mantissi ja järgu jaoks võib kasutada soovi korral ka ühe (näiteks esimese) liidetava mantissi ja järgu pesi.

| | | | |
|---|---|---|---|
| $\ell$+13 | 17 | b + 4004 | kontrollime, kas nimetaja on |
| $\ell$+14 | 21 | $\ell$ + 52 | nullist erinev, kui (r) = $= \pm 0$, siis $\omega = 1$. |
| $\ell$+15 | 02 | b + 4004 | |
| $\ell$+16 | 16 | a + 4012 | tõõpesa nimetaja mantissi salvestamiseks |
| $\ell$+17 | 02 | b + 0010 | |
| $\ell$+20 | 16 | a + 0011 | tõõpesa nimetaja järgu salvestamiseks |
| $\ell$+21 | 20 | 0000 | |
| $\ell$+22 | 03 | a + 4002 | A märgi muutmine vastupidiseks |
| $\ell$+23 | 16 | b + 4002 | -A mantissi ja järgu viimine |
| $\ell$+24 | 02 | a + 0007 | liitmisprogrammi spetsiaalsetesse pesadesse |
| $\ell$+25 | 16 | b + 0007 | |
| $\ell$+26 | 20 | $\ell$ + 31 | tagasipöördumise ettevalmistamine |
| $\ell$+27 | 16 | $\lambda$ | |
| $\ell$+30 | 22 | k + 0 | |
| $\ell$+31 | 02 | a + 4000 | (£-A)£ arvutamine |
| $\ell$+32 | 06 | b + 4004 | |
| $\ell$+33 | 15 | a + 4014 | tõõpesa tulemuse mantissi jaoks |
| $\ell$+34 | 01 | a + 0006 | |
| $\ell$+35 | 01 | b + 0010 | |
| $\ell$+36 | 16 | a + 0016 | tõõpesa tulemuse järgu jaoks |
| $\ell$+37 | 02 | a + 4014 | jagamine |
| $\ell$+40 | 06 | $\langle \frac{1}{2} \rangle$ | |
| $\ell$+41 | 07 | a + 4012 | |
| $\ell$+42 | 15 | a + 4012 | jagatise mantiss |
| $\ell$+43 | 01 | a + 0016 | |
| $\ell$+44 | 03 | a + 0011 | |

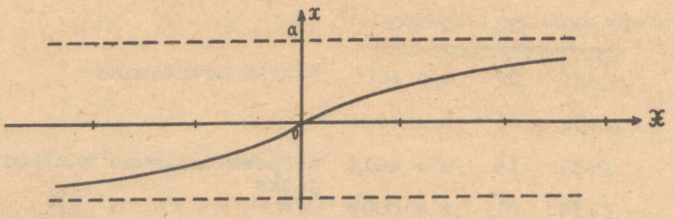
| | | | |
|---|---|---|---|
| $\ell$+45 | 01 | <000001> | |
| $\ell$+46 | 16 | a + 0011 | jagatise järk |
| $\ell$+47 | 32 | 0000 | tulemuse järgu trükkimine |
| $\ell$+50 | 02 | a + 4012 | |
| $\ell$+51 | 32 | 0000 | tulemuse mantissi trükkimine |
| $\ell$+52 | 37 | 0000 | peatus. |

Lõpuks kõneleme lühidalt programmeerimisest transformeeritud koma süsteemis.

Meetodi põhimõte seisneb arvude $\mathfrak{X}$ ( $-\infty < \mathfrak{X} < +\infty$ ) teisendamises vahemikku (- a, a), $0 < a < 1$, kusjuures see teisendus peab olema pööratav ja arvutusmasinas lihtsalt realiseeritav.

Võib esitada näiteks järgmise teisendusvalemi[16] (joon.1)

$$x = f(\mathfrak{X}) = \frac{a\mathfrak{X}}{1 + |\mathfrak{X}|}, \qquad 0 < a < 1,$$




Joon. 1.

kusjuures pöördteisendus avaldub kujul

$$\mathfrak{X} = F(x) = \frac{x}{a - |x|} .$$

16 N.Warmus, Transformed Point Arithmetic for Digital Computers, Bulletin de l'Academie Polonaise des Sciences. Serie des sciences Math.astron. et physiques. Vol. IX. Nº 4, 1961, 237-239.

Kerge on näha, et kui $-\infty < \mathfrak{X} < +\infty$, siis $-a < x < a$.

Tähistame aritmeetilistele tehetele $\mathfrak{X}+\mathfrak{Y}$, $\mathfrak{X}-\mathfrak{Y}$, $\mathfrak{X}\cdot\mathfrak{Y}$, $\mathfrak{X}:\mathfrak{Y}$ vastavad tehted nende arvude transformatsioonide x, y vahel järgmiselt $x \hat{+} y$, $x \hat{-} y$, $x \hat{\cdot} y$, $x \hat{:} y$. Vastavat tehet mõistame seejuures järgmiselt

$$f(\mathfrak{X}+\mathfrak{Y}) = f(\mathfrak{X}) \hat{+} f(\mathfrak{Y}) = x \hat{+} y$$

jne.

Kasutades tähiseid $\overline{x} = a - |x|$, võime üles kirjutada transformeeritud tehete avaldised:

$$x \hat{+} y = \frac{a z_1}{\overline{x}\,\overline{y} + |z_1|}, \text{ kus } z_1 = x\overline{y} + y\overline{x}, \tag{9}$$

$$x \hat{-} y = x \hat{+} (-y), \tag{10}$$

$$x \hat{\cdot} y = \frac{a z_2}{\overline{x}\,\overline{y} + |z_2|}, \text{ kus } z_2 = x y, \tag{11}$$

$$x \hat{:} y = x \hat{\cdot} \overline{y}\,\operatorname{sgn} y, \quad y \neq 0. \tag{12}$$

Valemite (9) - (12) järgi arvutamisel ei saa ületäitumist esineda.

Märgime, et kui $\mathfrak{X}$ transformatsioon on x, siis $-\mathfrak{X}$ transformatsioon on $-x$ ja $\frac{1}{\mathfrak{X}}$ transformatsioon on $a \operatorname{sgn} x - x = \overline{x} \operatorname{sgn} x$.

Arvutuste täpsus transformeeritud koma süsteemis on suurem arvutuste täpsusest liikuva koma süsteemis, kui arvud on lähedased ühele. Absoluutväärtuselt väga suurte ja väga väikeste arvude puhul on aga arvutuste täpsus transformeeritud koma süsteemis väiksem täpsusest liikuva koma süsteemis.

Harjutusülesannetena koostada programmid tehete $x \hat{+} y$, $x \hat{\cdot} y$ ja $x \hat{:} y$ teostamiseks.

## V p e a t ü k k.

## PROGRAMMEERIMISE ÜLDISI KÜSIMUSI.

### § 21. Standardprogrammide süsteem.

Matemaatiliste ülesannete lahendamisel arvutusmasina abil on kõigepealt vaja algandmed teisendada 10-ndsüsteemist 2-ndsüsteemi, tulemuste väljatrükkimisel tuleb aga kasutada vastupidist teisendust. Peale selle on matemaatiliste ülesannete lahendamisel väga sageli vaja arvutada elementaarfunktsioonide (sin x, arcsin x, $\sqrt{x}$, ln x, $e^x$ jne.) ja spetsiaalsete funktsioonide (sfäärilised ja silindrilised funktsioonid, Legendre'i funktsioonid jne.) väärtusi jms.

Selliste sagedamini esinevate arvutuste sooritamiseks on otstarbekohane koostada vastavad programmid (nn. standardsed alamprogrammid) ainult üks kord ja kasutada neid alati just sellisel standardsel kujul. See võimaldab oluliselt kokku hoida programmeerija aega.

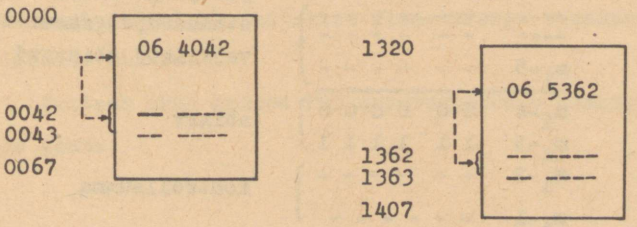
Loomulikult tekib küsimus, kuidas neid standardseid alamprogramme koostada, säilitada ja põhiprogrammi lülitada.

Kõigepealt märgime, et standardseid alamprogramme on otstarbekohane koostada nii, et neis poleks tsükleid käskudega 25 m ja 24 a. Vastupidisel juhul tuleks alamprogrammide poole pöördumisel kasutada mitmesuguseid tsüklite loendaja

sisu taastamise võtteid, mis oleks küllaltki ebamugav.

Teiseks. Standardprogramme võiks põhimõtteliselt koostada ja säilitada tähelistes aadressides. Kuid siis tuleb konkreetse programmi koostamisel standardprogramm ümber kirjutada konkreetsetes numbrilistes aadressides. See võte on aga töömahukas, pealegi võivad tekkida ümberkirjutamisvead. Otstarbekohane on standardsed programmid koostada (ja perforeerida) nii, nagu nad algaksid pesast 0000. Nende programmide paigutamisel magnettrumlile mõnest teisest pesast alates, näiteks alates pesast 1320, tuleb muuta kõiki nn. sisemisi aadresse (käskude aadressiosad, mis on seotud sellesama programmiga) sellesama konstandi 1320 võrra. On ilmne, et programmi muud käsud (nn. mittesisemiste aadressiosadega käsud) jäävad muutmata.

Näide 1. Kui standardses programmis A, mis asub pesades 0000 ÷ 0067, on käsk 06 4042, siis on aadress 4042 sisemine (4042 koosneb lühikestest pesadest 0042 ja 0043). Kui paigutame selle programmi magnettrumlile alates pesast 1320, siis tuleb selle käsu aadressile liita 1320, millega käsk omandab uue kuju 06 5362.




Joon. 1.

Käskude aadressiosi on lihtne muuta masina enda abil, kui on näidatud, missuguste käskude puhul tuleb seda teha. Et standardprogrammides ei kasutata tsükleid käskude 25 m ja 24 a abil, võib selleks tunnuseks võtta märgi "-" käsu ees (pärast teisendamist märk "-" muidugi kustutatakse). Programmi, mis teisendab standardprogrammide sisemisi aadresse ja paigutab standardprogrammid vajalikku kohta magnettrumlil, nimetatakse koostavaks programmiks ( составляющая программа ). Standardseid alamprogramme ühes vastava koostava programmiga nimetataksegi standardprogrammide süsteemiks.

Koostav programm on perforeeritud standardprogrammidega samale lindile (pikkade pesade kaupa). Koostav programm asub tsoonis 3, iga standardprogramm asub omaette tsoonis. Et standardprogramme fikseeritud ja liikuva koma süsteemis kasutatakse eraldi, siis on need perforeeritud eri lintidele.

Standardprogrammi üldine ehitus on toodud joonisel 2.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0000 | - | - | - | - | - | - | | | standardprogrammi |
| 0001 | - | - | - | - | - | - | | | |
| ---- | - | - | - | - | - | - | | | käskude osa |
| ---- | - | - | - | - | - | - | | | |
| ---- | - | - | - | - | - | - | | | |
| ---- | 3 | 0 | | $\lambda_i$ | | | | | juhtimise üleandmine standardprogrammilt põhiprogrammile |
| ---- | 2 | 2 | | 0 | 0 | 0 | 0 | | |
| ---- | - | - | - | - | - | - | | | standardprogrammi tööks vajalikud abiarvud |
| ---- | - | - | - | - | - | - | | | |
| $m_i$-5 | - | - | - | - | - | - | | | |
| $m_i$-4 | 0 | 0 | 0 | 0 | 0 | 0 | | | abiarv |
| $m_i$-3 | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| $m_i$-2 | - | - | - | - | - | - | | | kontrollsumma |
| $m_i$-1 | - | - | - | - | - | - | | | |

Joon. 2.

Standardprogrammist väljumine põhiprogrammi juurde toimub käskude 30 $\lambda_i$; ja 22 0000 abil nagu liitmisprogrammis liikuva koma meetodil. Seega tuleb enne standardprogrammi poole pöördumist pessa $\lambda_i$ viia aadress, kuhu tuleb tagasi pöörduda peale standardprogrammi töö lõppu. 4 viimast lühikest pesa on vajalikud standardprogrammi magnettrumlile viimiseks. Pärast standardprogrammi sisseviimise kontrolli osutuvad need ülearusteks. Magnettrumli pesade planeerimisel tuleb standardprogrammidele reserveerida ainult $m_i$ - 4 lühikest pesa.

Standardprogrammid fikseeritud koma režiimis on koostatud selliselt, et argument saadetaks (loomulikult enne standardprogrammi poole pöördumist) summaatorisse. Tulemus saadakse samuti summaatorist. Liikuva koma süsteemis koostatud standardprogrammide puhul tuleb argumendi mantiss saata pessa 7740 ja järk summaatorisse. Tulemuse mantiss saadakse pesast 7740 ja järk summaatorist. Asjaolu, et argument ja resultaat paiknevad ühtedes ja samades pesades, hõlbustab standardprogrammide järjestikust rakendamist.

Tööpesad on kõikidel standardprogrammidel ühised ja on paigutatud magnettrumli lõppu (pesad 7742 ÷ 7764). Standardprogrammi töö algab juhtimise üleandmisega standardprogrammi esimesele käsule.

Esitame nüüd andmed TRÜ arvutuskeskuse standardprogrammide kohta.

Standardprogrammid fikseeritud koma režiimis.

| Alamprogramm | Tsooni nr. | $m_i$ | $m_i$-4 | Väljumis-pesa $\lambda_i$ | Tööpesad |
|---|---|---|---|---|---|
| 10 → 2 | 4 | 112 | 106 | 3767 | 7756 - 7760 |
| 2 → 10 | 5 | 104 | 100 | 3766 | 7742 - 7760 |
| $\sqrt{x}$ | 7 | 60 | 54 | 3770 | 7754 - 7760 |
| $\frac{1}{4}e^x$ | 10 | 56 | 52 | 3771 | 7760 |
| $\frac{1}{2\pi}\ln x$ | 11 | 70 | 64 | 3772 | 7753 - 7760 |
| kraadimõõdu teisendamine radiaanmõõduks | 6 | 100 | 74 | 3773 | 7752 - 7760 |
| sin x | 12 | 46 | 42 | 3774 | 7756 - 7760 |
| cos x | 13 | 40 | 34 | 3775 | 7760 |
| tan x | 14 | 42 | 36 | 3776 | 7754 - 7760 |
| arctan x | 20 | 64 | 60 | 3777 | 7756 - 7760 |
| sin x ja cos x, kus x on mastaabik.2$\pi$ | 15 | 56 | 52 | 3765 | 7756 - 7760 |
| arc sin x | 16 | 142 | 136 | 3764 | 7750 - 7760 |

Standardprogrammid liikuva koma süsteemis.

| Alamprogramm | Tsooni nr | $m_i$ | $m_i$-4 | Väljumis-pesa | Tööpesad |
|---|---|---|---|---|---|
| 10 → 2 | 22 | 152 | 146 | 3767 | 3742 - 3764 |
| 2 → 10 | 23 | 230 | 224 | 3766 | 7742 - 7744 |
| $\sqrt{x}$ | 24 | 52 | 46 | 3770 | 7742 - 7744 |
| $e^x$ | 25 | 162 | 156 | 3771 | 3742 - 3752 |
| ln x | 26 | 100 | 74 | 3772 | 7742 - 7746 |
| sin x | 27 | 106 | 102 | 3774 | 3742 - 3744 |
| cos x | 30 | 72 | 66 | 3775 | 3743 - 3745 |
| tan x . | 31 | 124 | 120 | 3776 | 3743 - 3751 |
| arctan x | 32 | 122 | 116 | 3777 | 3742 - 3744 |

Märkus: arvud $m_i$ ja $m_i$-4 on 8-ndsüsteemis.

Koostav programm on perforeeritud 3. tsoonis ja viiakse sealt magnettrumlile algkäivitusega. Magnettrumlil paikneb see pesades 0060 ÷ 0275. Pesadesse 0030 ÷ 0057 viiakse informatsioon sisseviidavate standardprogrammide kohta järgmisel kujul:

$$
\begin{array}{l}
(0030) = 2\,P \\
(0031) = k_1 \\
\hline
(0032) = 01\ C_1 \\
(0033) = m_1 \\
\hline
(0034) = 01\ C_1 \\
(0035) = m_2 \\
\hline
\text{----} \quad \text{----} \\
\text{----} \quad \text{----} \\
(0030+2i) = 01\ C_i \\
(0031+2i) = m_i \\
\text{----} \quad \text{----} \\
\text{----} \quad \text{----}
\end{array}
$$

kus i = 1,2,...,P. P tähistab magnettrumlile viidavate standardprogrammide arvu (8-ndsüsteemis). Ühe korraga võib magnettrumlile viia maksimaalselt 11 standardprogrammi (s.t. $P \leqslant 13_8$). $C_i$ abil tähistame tsooni numbri, milles asub i-s sisseviidav standardne programm, $m_i$ abil tähistame i-nda standardprogrammi pesade arvu. Suurused $C_i$ ja $m_i$ leiame eespool toodud tabelitest. $k_1$ abil tähistame esimese standardse programmi esimese pesa aadressi. $k_1$ peab olema paaris aadressiline lühikene pesa, kusjuures peab olema täidetud tingimus

$$k_1 > 0304 + \max_{1 \leq i \leq P} m_i .$$

Kasutades pesadesse 0030 ÷ 0057 viidud informatsiooni, teeb koostav programm järgmist:

1) trükib pesa aadressi, kuhu viiakse teisendatava standardprogrammi esimene käsk;

2) toob järjekordse standardprogrammi magnettrumlile ja kontrollib, kas see on toodud magnettrumlile õigesti;

3) modifitseerib sisemised aadressid ja kaotab märgid "-" käskude ees. Kontrollib modifitseerimise õigsust;

4) moodustab arvu $k_{i+1} = k_i + m_i - 4$ (s.t. (i+1)-se standardprogrammi 1.pesa aadressi).

Vastavaid operatsioone teostatakse seni, kuni kas masin mingil põhjusel peatub või on juba magnettrumlile viidud kõik ettenähtud standardprogrammid. Märgime, et standardprogrammid paigutatakse magnettrumlile üksteise järele ilma ühegi vaheta (standardprogrammide tööpesad asuvad magnettrumli lõpus).

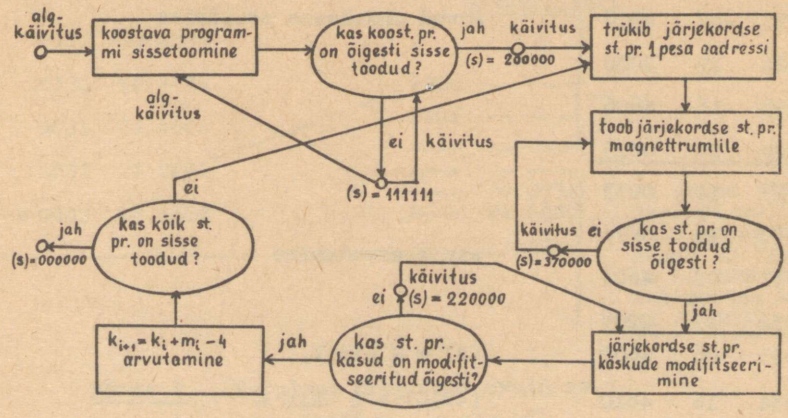
Peatused koostava programmi sisseviimisel ja koostava programmi töö käigus ühes summaatori vastava sisuga näitavad, kuidas kulgeb informatsiooni (standardprogrammide) salvestamine magnettrumlile.

| Summaatori sisu | Peatuse põhjus | Peatuse kõrvaldamine |
|---|---|---|
| 20 0000 | vastav programm sisse viidud õigesti | vajutus nupule "käivitus" |
| 11 1111 | koostav programm sisse viidud valesti (kontrollsummad ei ühti) | "käivitus" - arvutatakse uuesti kontrollsumma "algkäivitus" - koostav programm viiakse uuesti masinasse |

| Summaatori sisu | Peatuse põhjus | Peatuse kõrvaldamine |
|---|---|---|
| 37 0000 | standardprogramm on viidud magnettrumlile valesti | "käivitus" |
| 22 0000 | viga standardprogrammi käskude modifitseerimises | "käivitus" |
| 00 0000 | kõik standardprogrammid on viidud magnettrumlile ja modifitseeritud õigesti | võtta sisendseadmest standardprogrammide lint. Viia sisse lint põhiprogrammiga. |

Pärast standardprogrammide sisseviimist pole informatsioon pesades 0030 ÷ 0057 ega koostav programm enam vajalik. Sellele kohale viiakse tavaliselt põhiprogramm. Pärast põhiprogrammi ja algandmete sisseviimist võib ülesande lahendamine alata.

Illustreerime koostava programmi sisseviimist ja tööd järgmise joonise abil.




Joon. 3.

Vaatleme lõpuks paari näidet standardprogrammide kasutamise kohta.

Näide 2. Koostame programmi (nn.põhiprogrammi) suuruse

$$Z = \frac{1}{3} \sin X + \frac{1}{2} \sqrt{X^2 - A}$$

arvutamiseks fikseeritud koma režiimis. Eeldame, et ületäitumist ei esine. Algandmed ja vajalikud standardprogrammid paigutame järgmiselt:

$\frac{1}{2}$ = (4600), $\frac{1}{3}$ = (4602), x = (4604), A = (4606)

st. pr. "10→2" algab pesast 2000, $\lambda_1$ = 3767,

st. pr. "2→10" algab pesast 2106, $\lambda_2$ = 3766,

st. pr. "sin X" algab pesast 2206, $\lambda_3$ = 3774,

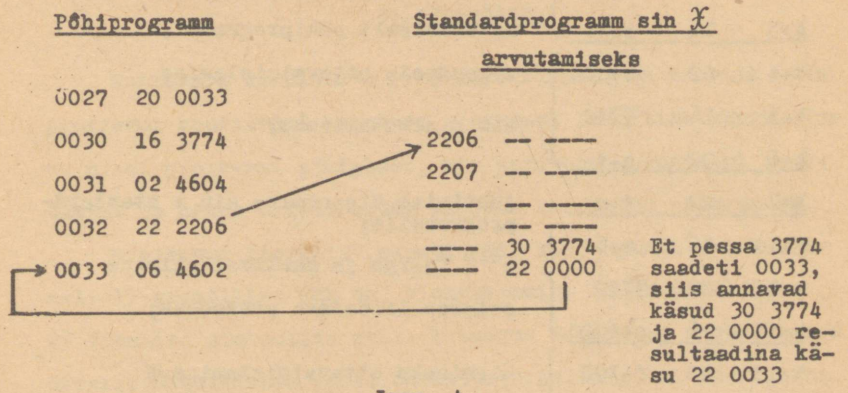
st. pr. "$\sqrt{x}$" algab pesast 2250, $\lambda_4$ = 3770

Paigutame põhiprogrammi magnettrumlile alates pesast 0020. See tuleb järgmine:

| | | | |
|---|---|---|---|
| 0020 | 20 | 0025 | |
| 0021 | 16 | 3767 | Arvude teisendamine 10→2. |
| 0022 | 25 | 4006 | Pärast teisendamist viiakse |
| 0023 | -02 | 4606 | arvud samadesse pesadesse |
| 0024 | 22 | 2000 | |
| 0025 | -16 | 4606 | |
| 0026 | 24 | 0023 | |
| 0027 | 20 | 0033 | |
| 0030 | 16 | 3774 | sin X arvutamine |
| 0031 | 02 | 4604 | |
| 0032 | 22 | 2206 | |
| 0033 | 06 | 4602 | (s) = $\frac{1}{3}$ sin X |
| 0034 | 16 | 4610 | (4610) = $\frac{1}{3}$ sin X |

| | | | |
|---|---|---|---|
| 0035 | 20 | 0045 | tagasipöördumise ettevalmistamine |
| 0036 | 16 | 3770 | ruutjuure arvutamiselt |
| 0037 | 20 | 0050 | tagasipöördumise ettevalmistamine |
| 0040 | 16 | 3766 | standardprogrammist 2 → 10 |
| 0041 | 02 | 4604 | (s) = X |
| 0042 | 06 | 4604 | (s) = $X^2$ |
| 0043 | 03 | 4606 | (s) = $X^2$ - A |
| 0044 | 22 | 2250 | suunatakse ruutjuure standardprogrammi |
| 0045 | 06 | 4600 | (s) = $\frac{1}{2}\sqrt{X^2 - A}$ |
| 0046 | 01 | 4610 | (s) = Z (2-ndsüsteemis) |
| 0047 | 22 | 2106 | suunatakse standardprogrammi 2 → 10 |
| 0050 | 32 | 0000 | tulemuse trükkimine (10-ndsüsteemis) |
| 0051 | 37 | 0000 | peatus |

Juhtimise üleandmist põhiprogrammilt standardprogrammile iseloomustab järgmine joonis:




Joon. 4.

Näide 3. Koostame programmi avaldise

$$Z = e^{\sin x} + e^{\cos \sqrt{x}}$$

väärtuse arvutamiseks liikuva koma meetodil. Algandmed paigu-

tame järgmiselt:

$$m_x = (a + 4000),\ p_x = (a + 0004)$$

Eeldame, et $X > 0$ ja on juba teisendatud 2-ndsüsteemi. Liitmisprogramm liikuva koma meetodi jaoks algab pesast k+40, $\lambda$ = m+11. Algandmete ja lõpptulemuste pesad on järgmised:

$$m_1 = m+4000,\ m_2 = m+4002,\ m_3 = m+4004,$$

$$p_1 = m+6,\ p_2 = m+7,\ p_3 = m+10.$$

Andmed standardsete programmide kohta on järgmised:

sin x standardprogramm algab pesast $\ell+0$, $\lambda = 3774$;

cos x -"- " " $\ell+102$, $\lambda = 3775$;

$e^x$ -"- " " $\ell+170$, $\lambda = 3771$;

$\sqrt{x}$ -"- " " $\ell+346$, $\lambda = 3770$.

Programm tähelistes aadressides tuleb järgmine:

| | | | |
|---|---|---|---|
| k+0 | 20 | $\ell$+170 | ülemineku ettevalmistamine sin x |
| k+1 | 16 | 3774 | arvutamiselt $e^x$ arvutamisele; |
| k+2 | 20 | k+10 | ülemineku ettevalmistamine $e^x$ |
| k+3 | 16 | 3771 | arvutamiselt põhiprogrammi; |
| k+4 | 02 | a+4000 | algandmete ettevalmistamine |
| k+5 | 16 | 7740 | sin x arvutamiseks; |
| k+6 | 02 | a+4 | |
| k+7 | 22 | $\ell$+o | juhtimise üleandmine sin x standardprogrammile; |
| k+10 | 16 | m+6 | $e^{\sin x}$ järgu ja mantissi viimine |
| k+11 | 02 | 7740 | esimese liidetava pesadesse; |
| k+12 | 16 | a+4000 | |
| k+13 | 20 | $\ell$+102 | ülemineku ettevalmistamine $\sqrt{x}$ |
| k+14 | 16 | 3770 | arvutamiselt cos x arvutamisele; |
| k+15 | 20 | $\ell$+170 | ülemineku ettevalmistamine cos x |
| k+16 | 16 | 3775 | arvutamiselt $e^x$ arvutamisele; |

| | | | |
|---|---|---|---|
| k+17 | 20 | k+27 | ülemineku ettevalmistamine $e^x$ |
| k+20 | 16 | 3771 | arvutamiselt põhiprogrammi; |
| k+21 | 20 | k+33 | ülemineku ettevalmistamine liitmis- |
| k+22 | 16 | m+11 | programmist põhiprogrammi; |
| k+23 | 02 | a+4000 | algandmete ettevalmistamine |
| k+24 | 16 | 7740 | $e^{\cos\sqrt{x}}$ arvutamiseks; |
| k+25 | 02 | a+4 | |
| k+26 | 22 | $\ell$+346 | juhtimise üleandmine $\sqrt{x}$ standard-programmile; |
| k+27 | 16 | m+7 | algandmete ettevalmistamine |
| k+30 | 02 | 7740 | liitmiseks; |
| k+31 | 16 | m+4002 | juhtimise üleandmine liitmisprogrammile; |
| k+32 | 22 | k+40 | |
| k+33 | 37 | 0000 | peatus. |

## § 22. Programmi loogiline skeem ja selle teisendamine.

Suuremate ülesannete programmeerimisel on kasulik enne programmi koostamist koostada programmi loogiline skeem. See on n.-ö. programmi põhiplaan, mis kergendab programmi koostamist ja aitab orienteeruda ka juba koostatud programmis.

Programmi loogilist skeemi võib kirja panna kahel viisil: 1) graafilise ehk nn. blokkskeemina (nagu § 21) ja 2) tähelise sümboolika abil. Erinevus loogilise skeemi nende üleskirjutusviiside vahel on mitte sisuline, vaid vormiline. Loogiliste skeemide koostamise põhimõtted ja -nõuded on mõlema mooduse puhul samad.

Vaatleme mingi ülesande lahendamisel teostatavaid tehteid (tehete sooritamise järjekorras) ja võtame nad kokku rühmadeks (blokkideks, operaatoriteks). Muidugi pole selline rühmitamine ühene. Täielikku meelevalda tehete rühmitamisel vähendavad järgmised soovitused ja nõuded:

1) ühte rühma pole otstarbekohane kokku võtta eritüübilisi arvutusi. Näiteks, kui tuleb teostada arvutusi valemite

$$a_1 = F_1(t, u, v)$$

$$a_2 = F_2(t, u, v, a_1)$$

järgi ja seejärel minna edasi ruutvõrrandi

$$x^2 + a_1 x + a_2 = 0$$

lahendamisele, siis on otstarbekohane kordajate arvutamine ja ruutvõrrandi lahendamine eraldada eri blokkidesse. Olenevalt $F_1$ ja $F_2$ kujust võib osutuda otstarbekohaseks ka $a_1$ ja $a_2$ arvutamise eraldamine eri blokkidesse.

Kindlasti on otstarbekohane eraldi blokkidesse rühmitada algandmete toomist kas perfo- või magnetlindilt ja seejärel toimuvaid arvutusi.

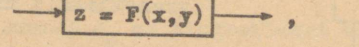
2) Tingimata tuleb eraldi blokina vaadelda mingisuguse tingimuse kehtivuse kontrolli, näiteks

„ kas $f(x) \geqslant + 0$?"

Niisuguste tingimuste põhjal toimub suunamine kas ühe või teise bloki juurde. Niisuguseid tingimuse kontrollimise blokke nimetatakse loogilisteks blokkideks. Loogiliste blokkide olulisem erinevus mitteloogilistest blokkidest seisneb selles, et esimesed võivad anda juhtimise edasi kahte erinevasse kohta, kuna viimased võivad anda juhtimise edasi ainult ühte kohta.

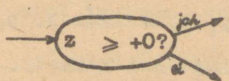
3) Juhtimine antakse väljastpoolt mistahes blokile alati ühte ja samasse kohta (ühele ja samale käsule). Samuti väljub juhtimine blokist alati ühest ja samast kohast (ühest ja samast käsust).

Mitteloogilise bloki tähistame blokkskeemis järgmiselt:




kus nooltega märgitakse juhtimise saamine ja üleandmine. Bloki sisse kirjutatakse tavaliselt (kas valemina või sõnalises tekstis), millist operatsiooni see blokk teostab.

Loogilise bloki tähistame järgmiselt:



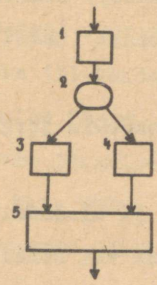

Kui loogilises blokis kontrollitav tingimus $z \geq +0$ on täidetud, siis antakse juhtimine edasi noole $\xrightarrow{\text{jah}}$ suunas, vastupidisel juhul aga noole $\xrightarrow{\text{ei}}$ suunas. Sõnu „jah" ja „ei" võib asendada ka teiste tähistega nagu 1,0 või +, -.

Sagedasti on otstarbekohane veel kõik blokid nummerdada ühtses numeratsioonis. Sel korral võib selgitused blokkide kohta tuua eraldi loetelus vastavate bloki numbrite all ja blokkskeemis need hoopis ära jätta.

Märgime, et blokkskeemi blokke võib vajaduse korral osadeks jaotada või kokku võtta. Seejuures peavad kehtima nõuded 2) ja 3). Märgime, et nõue 2) on suhtelise iseloomuga. Kui programmi osi, kuhu antakse suunamine „jah" ja „ei" korral, ei vaadelda eraldi blokkidena, siis pole võimalik vaadelda eraldi blokina ka vastava tingimuse kehtivuse kontrolli blokki.

Selgitame toodud seisukohti näite varal.

Näide 1. Olgu antud blokkskeem:




Nõude 3) põhjal saab blokk 5 muidugi juhtivuse ühesse ja samasse kohta, kuigi see graafiliselt nii ei näi olevat.

Kõiki blokke 1 - 5 võib kokku võtta üheks blokiks. Kokku ei või võtta aga blokke 1 ja 2 (tingimuse 2) põhjal). Kokku ei või võtta ka blokke 1, 2, 3, 4 või 2, 3, 4 (sel korral antakse juhtimine välja kahest eri kohast). Samuti ei või võtta kokku blokke 3, 4, 5 ega 3, 5 (sel juhul saaks juhtimise uue "bloki" erinevad kohad).

Toome nüüd näiteid ülesannete loogiliste skeemide koostamise kohta.

Näide 2. Koostame tingrefleksi modelleerimise ülesande blokkskeemis.

Tähistame $\alpha$ abil tingimatu ja $\beta$ abil tingitud ärritaja, A abil aga reaktsiooni. Suurused $\alpha$, $\beta$ ja A võivad omandada kaht väärtust: 1 (kui ärritaja toimib või reaktsioon esineb) ja 0 (vastupidisel juhul).

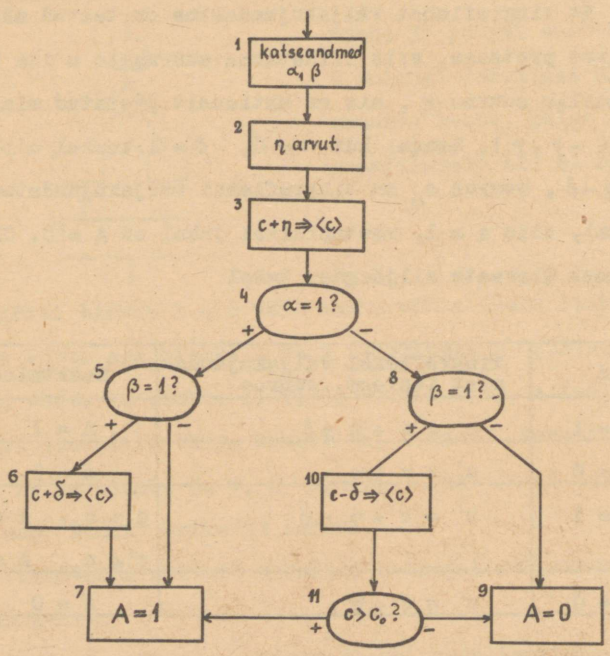
Kui $\alpha = 1$, siis olenemata $\beta$ väärtusest on A = 1. Kui $\alpha = 0$, siis $\beta = 0$ korral on ka A = 0. Kui $\alpha = 0$, $\beta = 1$, siis oleneb reaktsiooni esinemine tingrefleksi väljakujundamise tugevusest. Viimast iseloomustame teatava suuruse c abil.

Kui $\alpha = \beta = 1$, siis liidame suurusele c veel $\delta > 0$ (ting-

refleks tugevneb), kui $\alpha = 0$, $\beta = 1$ lahutame $\delta$ (tingrefleks nõrgeneb). Et tingrefleksi väljakujundamine on teatud määral stohhastiline protsess, siis liidetakse suurusele c iga katse korral juhuslik suurus $\eta$, mis on ühtlaselt jaotatud mingis vahemikus $(-\gamma, \gamma)$. Seega: kui $\alpha = 0$, $\beta = 1$ korral $c' > c_0$ ($c' = c + \eta - \delta$, suurus $c_0$ on tingrefleksi väljakujundatuse läviväärtus), siis A = 1, vastupidisel juhul on A = 0. Olukorrast annab ülevaate alljärgnev tabel

| Ärritajad | Tingrefleksi väljakujundamist iseloom. suurus | Reaktsioon |
|---|---|---|
| $\alpha = 1$, $\beta = 1$ | $c' = c + \eta + \delta$ | A = 1 |
| $\alpha = 1$, $\beta = 0$ | $c' = c + \eta$ | A = 1 |
| $\alpha = 0$, $\beta = 1$ | $c' = c + \eta - \delta$ | $c' > c_0$, A = 1<br>$c' \leq c_0$, A = 0 |
| $\alpha = 0$, $\beta = 0$ | $c' = c + \eta$ | A = 0 |

Vastav blokkskeem tuleb järgmine:




Joon. 5.

Märkus: Kirjutise „c+η ⇒ ⟨c⟩" abil tähistame operatsiooni: suurusele c liidetakse η ja arvutatud summa viiakse pessa, kus oli salvestatud c.

Edasi vaatleme loogiliste skeemide teist eritusvõimalust, kus tehete rühmad tähistatakse tähtede abil. Siin nimetatakse neid rühmi operaatoriteks. Põhinõuded operaatorite puhul on muidugi samad mis blokkide puhul. Seega on oluline loogiliste ja mitteloogiliste operaatorite eristamine. Sageli eristatakse ka mitteloogiliste operaatorite eri liike. Nii tähistab:

P loogilist operaatorit,

A aritmeetilist operaatorit (mis sooritab arvutusi mingi aritmeetilise avaldise järgi),

$\Pi$ ülekande operaatorit (arvude kandmine säilitusseadme ühest osast teise jne.)

F parameetrite muutmise operaatorit,

T trükkimise operaatorit,

S standardprogrammide operaatorit

Я peatust jne.

Operaatorit üldse tähistame Q abil, mitteloogilist operaatorit D abil.

Loogilise skeemi (nn. operaatorskeemi) üleskirjutuse reeglid on järgmised:

1. Skeemis nummerdatakse kõik operaatorid ühtses numeratsioonis, näiteks

$$\Pi_1 S_2 A_3 S_4 A_5 S_6 T_7 Я_8.$$

2. Kui operaator sõltub parameetrist, siis märgitakse see parameeter kas sulgudesse või ülemise indeksina, näiteks

$$A_1(i) \quad \text{või} \quad A_1^i .$$

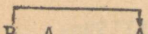
3. Kui kaks operaatorit on üksteise kõrval, siis annab vasakpoolne operaator juhtimise üle parempoolsele. Seega $\Pi_1 S_2 A_3$ tähendab

$$\Pi_1 \rightarrow S_2 \rightarrow A_3.$$

4. Kui mingi operaator ei anna kunagi juhtimist vahetult paremal asuvale operaatorile, siis eraldatakse need operaatorid semikooloniga

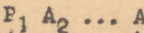
$$A_2 \; ; \; A_3.$$

5. Juhtimise üleandmine märgitakse nooltega järgmisel viisil:



kui $P_1$ on täidetud, siis antakse juhtimine operaatorile $A_k$

kui $P_1$ pole täidetud, siis antakse juhtimine operaatorile $A_2$

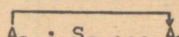


kui $P_1$ pole täidetud, siis antakse juhtimine operaatorile $A_k$

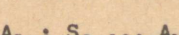
kui $P_1$ on täidetud, siis antakse juhtimine operaatorile $A_k$

Ühe sõnaga: tingimuse täidetuse korral antakse juhtimine ülemise noole kaudu, mittetäidetuse korral alumise noole kaudu.

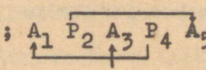
Kui juhtimine antakse üle mitteloogiliselt operaatorilt, võime kirjutada kas



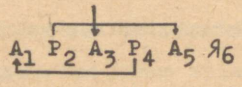
või



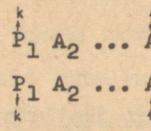
6. Kui skeemis ei alusta tööd mitte esimene operaator, tähistatakse seda järgmiselt:

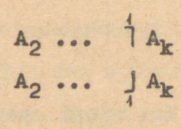


või



Märkus: Mõnedes töödes kasutatakse noolte asemel noolekesi või nurki





Näide 3. Koostame loogilise skeemi programmi jaoks, mis võimaldab arvutada y(x), mis on antud ilmutamata kujul

$$F(x, y) = 0. \qquad (1)$$

Oletame, et valem (1) on teisendatav kujule

$$y = f(x,y), \tag{2}$$

mille puhul on võimalik kasutada iteratsioonimenetlust. Peale selle eeldame, et $y_1$ alglähendiks on a ja iga järgmise $y_{k+1}$ alglähendiks on $y_k$. Seega

$$(y_1)_o = a,$$

$$(y_{k+1})_o = y_k.$$

Ülesande lahendamisel teostame arvutusi valemi

$$(y_k)_{i+1} = f\,(x_k, (y_k)_i) \tag{3}$$

järgi, kus iteratsiooni samme i järgi teostatakse seni, kuni

$$|(y_k)_{i+1} - (y_k)_i| < \varepsilon \tag{4}$$

kus $\varepsilon$ on teatav etteantud positiivne arv. Tingimust (4) rahuldav $(y_k)_{i+1}$ võetakse argumendi väärtusele $x_k$ vastavaks funktsiooni väärtuseks $y_k$.

Arvud

$$a,\ x_1\ ,\ x_2\ ,\ \dots\ ,\ x_n$$

paigutame pesadesse

$$b+4000,\ b+4002,\ b+4004,\ \dots,\ b+4000+2n.$$

Tööpesasse c+4000 viime arvu $(y_k)_i$, pesasse b+4000 arvutame arvu $(y_k)_{i+1}$. Tingimust (4) rahuldava $(y_k)_{i+1}$ viime pesast b+4000 vastava argumendi väärtuse $x_k$ semele pessa b+4000+2k (argumendi väärtust $x_k$ edaspidistes arvutustes enam ei kasutata).

Arvestades veel materjali sisseviimist arvutusmasinasse, algandmete ja lõpptulemuste teisendamist ning lõpptulemuste trükkimist, tuleb programmi loogiline skeem järgmine:

$$\Pi_1\ S_2\ \Pi_3\ A_4^k\ P_5\ \Pi_6^k\ F_7^k\ P_8\ S_9\ T_{10}\ Я_{11}.$$

Operaatorite tähendus on järgmine:

$\Pi_1$ - programmi ja algandmete viimine masinasse,

$S_2$ - algandmete teisendamine 2-ndsüsteemi

$\Pi_3$ - arvu viimine pesast b+4000 pessa c+4000 (ehk lühidalt (b+4000) $\Rightarrow$ c+4000),

$A_4^k$ - arvutamine valemi (3) järgi ehk

f( (b+4000+2k),(c+4000)) $\Rightarrow$ b+4000

(k algväärtus on 1),

$P_5$ - tingimuse (4) kontroll ehk „kas $|(b+4000) - (c+4000)| < \varepsilon$ ?",

$\Pi_6^k$ - (b+4000) $\Rightarrow$ b+4000+2k ,

$F_7^k$ - k+1 $\Rightarrow$ ⟨k⟩,

$P_8$ - „kas k $\neq$ n+1 ?",

$S_9$ - tulemuste teisendamine 10-ndsüsteemi,

$T_{10}$ - tulemuste trükkimine,

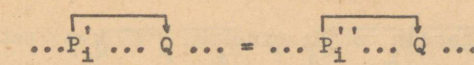
$Я_{11}$ - peatus.

Vaatleme nüüd loogiliste skeemide teisendamist. Skeemide teisendamise eesmärgiks on nende lihtsustamine, kusjuures pärast teisendamist saadud skeem peab olema lähteskeemiga samaväärne. Samaväärsust mõistame funktsioneerimise seisukohalt, s.t. kaht skeemi nimetame samaväärseks, kui nendes skeemides töötavad alati ühed ja samad mitteloogilised operaatorid (resp.blokid) ühes ja samas järjekorras.

Et teisenduspõhimõtted on analoogilised nii operaatorkui ka blokkskeemide puhul, siis vaatleme ainult operaatorskeeme.

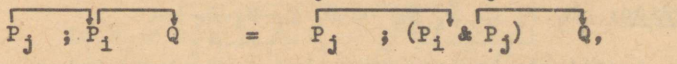
Kõigepealt esitame operaatorskeemide teisendusreeglid.

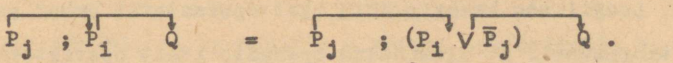
1. Kui $P_i(\alpha_1, \alpha_2, \ldots, \alpha_n) = P_i'(\alpha_1, \alpha_2, \ldots, \alpha_n)$, siis



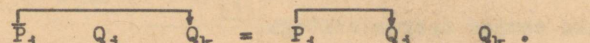
Loogilises skeemis võib ühe loogilise funktsiooni (tingi asendada temaga samaväärse funktsiooniga (tingimusega).

2. Kui $P_i$ saab juhtimise siis ja ainult siis, kui P siis võib $P_i$ asendada $P_i \& P_j$ või $P_i \vee P_j$ abil. Seega

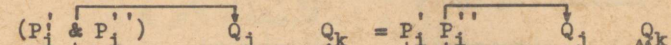




3.Loogilise eituse märgi võib ära jätta, kui selle t gimuse juurest lähtuvad nooled ümber pöörata:



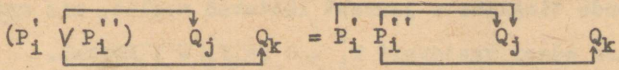
4. Kahe tingimuse konjunktsiooni saab teisendada nii



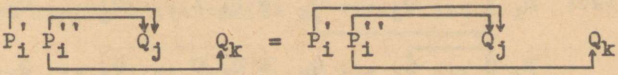
Konjunktsiooni kommutatiivsust kasutades saame



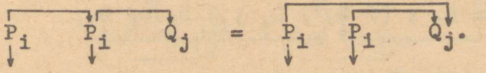
5. Kahe tingimuse disjunktsiooni saab teisendada nii:



Disjunktsiooni kommutatiivsust kasutades saame



6. Samade loogiliste operaatorite korduval rakendamisel saab skeemi teisendada nii:



Kui teine operaator $P_i$ ei saa juhtimist kuskilt mujalt kui esimeselt operaatorilt $P_i$, võib teise hoopis ära jätta:

$$P_i \; ; P_i \; Q_j = P_i \; ; \; Q_j .$$

7. Kui loogilises skeemis mingile operaatorile $Q_j$ üldse juhtimist ei anta, siis võib selle operaatori $Q_j$ ühes temalt lähtuvate nooltega loogilisest skeemist välja jätta.

Näide: $D_1 P_2 Q_3 D_4 D_5 = D_1 P_2 D_4 D_5$ .

8. Loogilises skeemis võib igat operaatorit ümber paigutada tingimusel, et üleminekud operaatoritelt operaatoritele selle ümberpaigutuse juures ei muutu. Piltlikult öeldes: noolte algused ja lõpud peavad enne ja pärast ümberpaigutamist olema seotud ikka samade operaatoritega.

Näide: $P_1 D_2 D_3 = \; ; D_2 \; ; P_1 D_3$

s.t. $P_1 \; D_2 \; D_3 = \; ; D_2 \; ; P_1 \; D_3$.

9. Kui loogilises skeemis esineb samaselt kehtiv ($P_i' \equiv 1$) või samaselt mittekehtiv ($P_i'' \equiv 0$) tingimus, siis võib kustutada nende tingimuste juurest lähtuvad nooled, mis annaksid juhtimise edasi vastavalt $P_i' = 0$ ja $P_i'' = 1$ korral.

Näited: $D_i \; 1 \ldots Q_j \ldots Q_k = D_i \; 1 \ldots Q_j \ldots Q_k$,

$D_i \; 0 \ldots Q_j \ldots Q_k = D_i \; 0 \ldots Q_j \ldots Q_k$.

Näide 4. Teisendame loogilist skeemi

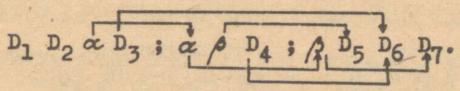
$$D_1 D_2 \overline{\alpha} D_3 \; ; (\alpha \,\&\, \overline{\beta}) D_4 \; ; \beta D_5 D_6 D_7,$$

kus $\alpha$ ja $\beta$ on loogilised muutujad.

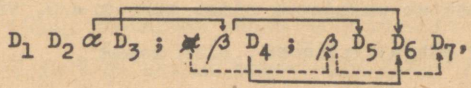
Kõigepealt teisendame konjunktsiooni (reeglid 4 ja 3)

$$(\alpha \,\&\, \overline{\beta}) D_4 \; ; \beta = \alpha \overline{\beta} D_4 \; ; \beta = \alpha \beta D_4 \; ; \beta .$$

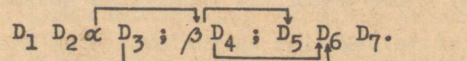
Kasutades reeglit 6 saab ära jätta muutuja $\beta$ korduva kontrollimise. Seega saame



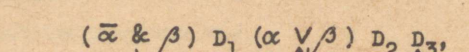
Kasutame $\alpha$ puhul reeglit 6 ja 7. Saame



millest nähtub, et ka muutuja $\beta$ pärast operaatorit $D_4$ on ülearune. Lõpuks saame

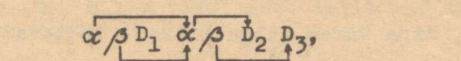


Näide 5. Teisendame loogilist skeemi

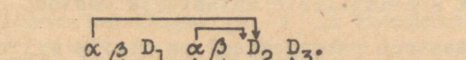


kus mitteloogiline operaator $D_1$ võib muuta ainult loogilise muutuja $\alpha$ väärtust.

Konjunktsiooni, disjunktsiooni ja eitusemärgi asendamisel saame



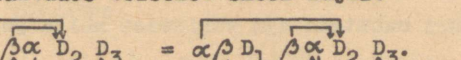
ja pärast reegli 6 kasutamist



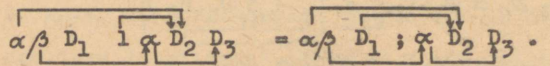
Reegli 5 põhjal saame kirjutada, et



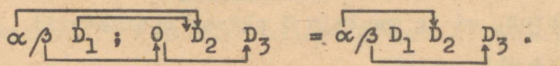
millest lähtudes teiseneb skeem kujule



Et $D_1$ ei muuda $\beta$ väärtust, siis töötab teine $\beta$ parajasti sel korral, kui $\beta = 1$. Reegli 2 põhjal võib selle asendada $\beta \vee \bar{\beta} \equiv 1$. Reega saame



Et teine $\alpha$ töötab parajasti siis, kui $\alpha = 0$, võib selle asendada reegli 2 põhjal $\alpha \& \bar{\alpha} \equiv 0$. Siis saame



Näite 5 põhjal võime öelda, et loogiliste skeemide teisendamisel on iga mitteloogilise operaatori puhul kasulik teada, milliste loogiliste muutujate väärtusi ta võib muuta.

## § 23. Monte-Carlo meetod.

Kaasaegsete arvutusmasinate loomine ja areng on tekitanud arvutusmeetodite valdkonnas mitmeidki ümberhinnanguid. Nii näiteks tunnistatakse tänapäeval efektiivseteks mõned sellised meetodid, mida varem (ilma kiirelt töötavate arvutusmasinateta) polnud mõeldavgi kasutada. Üheks selliseks on Monte-Carlo ehk statistiliste katsetuste meetod.

Monte-Carlo meetodi rakendamisel on tarvis kiirelt töötavat juhuslikkude arvude tekitajat. Selleks on mitu võimalust.

1. Juhuslikkude arvude tabel. Et vältida seaduspärasuste tekkimist valitavate arvude vahel, peab tabel olema suur. Siis tuleb see tabel mahutada aga välisesse mällu ja selle kasutamine võtab palju aega. Sellepärast ei saa seda moodust

eriti soovitada.

2. Juhuslikke arve võib arvutada eriprogrammide abil. Sellisel viisil saadud juhuslikke arve nimetatakse pseudo-juhuslikeks arvudeks. Nii näiteks annab valem

$$\xi_{i+1} = \left\{ \xi_i \cdot 2^{-7} + \left\{ \xi_i \cdot 2^{7} \right\}_{\text{murdosa}} \right\}_{\text{murdosa}} \qquad (1)$$

lõigul [0,1] ühtlaselt jaotatud juhuslike arvude jada. Astendajad -7 ja 7 on saadud proovimise teel. Valemit (1) saab arvutusmasinas "Ural-1" realiseerida umbes 10 käsu abil.

3. Kõige paremaid tulemusi (arvutuste täpsuse ja kiiruse mõttes) annab spetsiaalse juhuslike arvude anduri kasutamine. Anduris kasutatakse juhuslike arvude saamiseks füüsikalisi juhuslikke protsesse nagu raadioaktiivset lagunemist, müra-sid elektronlampides vms.

Juhuslike arvude jada ehk voog on iseloomustatav kahe tunnuse - diapasooni ja jaotusseaduse ehk jaotusfunktsiooni abil.

Juhuslike arvude voo diapasoon määratakse tõketega a ja b ($a < b$), mille vahel asub selle voo iga arv $\xi$, s.t.

$$a \leqslant \xi \leqslant b.$$

Jaotusfunktsiooniks nimetatakse funktsiooni $\Phi(\xi)$, mis on defineeritud lõigul [a, b] ja rahuldab tingimust: iga $\alpha, \beta$ korral ($a \leqslant \alpha \leqslant \beta \leqslant b$) kehtib

$$p(\alpha \leqslant \xi \leqslant \beta) = \int_{\alpha}^{\beta} \Phi(t)\,dt$$

kus $p(\alpha \leqslant \xi \leqslant \beta)$ tähistab sündmuse „$\alpha \leqslant \xi \leqslant \beta$" tõenäosust.

Monte-Carlo meetodi puhul on olulise tähtsusega Tšebõševi võrratus:

$$P\left\{ |\nu - p| < \varepsilon \right\} \geqslant 1 - \frac{p(1-p)}{n\varepsilon^2}, \qquad (2)$$

kus $\varepsilon$ on etteantud positiivne suurus, p - uuritava sündmuse tõenäosus, n - katsete koguarv, $\nu$ - uuritava sündmuse esinemissagedus n katse puhul (s.t. $\nu = \frac{m}{n}$, kus m on uuritava sündmuse esinemiste arv n katse puhul). Kasutades uuritava sündmuse tõenäosuse p asemel selle lähisväärtust $\nu$, on tarvis hinnata ka viga (resp. vea tõenäosust). Valem (2) ütlebki, et võrratuse $|\nu-p|<\varepsilon$ tõenäosus P on mitte väiksem kui $1 - \frac{p(1-p)}{n \cdot \varepsilon^2}$. Tõenäosust P nimetatakse usaldatavuseks (nimelt hinnangu $|\nu-p|<\varepsilon$ usaldatavuseks). Võrratusest (2) järeldub, et n suurenedes suureneb ka usaldatavus P.

Anname nüüd ette teatava usaldatavuse $\delta$ ($0 < \delta < 1$), fikseerime $\varepsilon$ ja leiame katsete koguarvu n, mille puhul

$$P\{|\nu-p|<\varepsilon\} \geq \delta \qquad (3)$$

Arvestades seda, et p(1-p) maksimaalne väärtus $0 \leq p \leq 1$ korral on $\frac{1}{4}$, saame

$$1 - \frac{p(1-p)}{n\varepsilon^2} \geq 1 - \frac{1}{4n\varepsilon^2}. \qquad (4)$$

Võrratustest (2) ja (4) järeldub, et

$$1 - \frac{1}{4n\varepsilon^2} \geq \delta \qquad (5)$$

korral kehtib ka (3). Võrratusest (5) saamegi soovitud hinnangu n jaoks

$$n \geq \frac{1}{4(1-\delta)\varepsilon^2}. \qquad (6)$$

Monte-Carlo meetodi kasutamisel võetakse $\varepsilon > 0$ selline, et see annaks tulemuse vajaliku täpsusega, kuna $\delta$ võetakse vahemikus $0{,}85 \div 0{,}95$.

Selgitame nüüd Monte-Carlo meetodit näidete varal.

Näide 1. Arvutame Monte-Carlo meetodi abil määratud integraali

$$\int_{\alpha}^{\beta} f(t)dt, \qquad (\alpha < \beta)$$

eeldades, et andur tekitab juhuslike arvude voo diapasooniga $[a, b]$ (kusjuures $a \leqslant \alpha < \beta \leqslant b$) ja jaotusfunktsiooniga $f(\xi)$, mis ühtib integreeritava funktsiooniga.

Jaotusfunktsiooni põhiomadusest saame, et

$$\int_{\alpha}^{\beta} f(t)dt = p(\alpha \leqslant \xi \leqslant \beta).$$

Tõenäosuse $p(\alpha \leqslant \xi \leqslant \beta)$ asemel arvutame selle lähisväärtuse $\nu$. Selleks leiame ettevõetud $\varepsilon$ ja $\delta$ puhul valemist (6) väärtuse n ja tekitame juhuslike arvude voo

$$\xi_1, \quad \xi_2, \quad \xi_3, \quad \ldots, \quad \xi_n .$$

Loendame $\xi_i$ $(i=1, 2, \ldots, n)$, mille puhul kehtib $\alpha < \xi_i < \beta$. Olgu nende arv m. Seega saame

$$\int_{\alpha}^{\beta} f(t)dt \approx \nu = \frac{m}{n} .$$

Näitest ilmneb Monte-Carlo meetodi üks voorusi - meetodi lihtsus.

Näide 2. Arvutame Monte-Carlo meetodi abil integraali

$$J = \int_{\alpha}^{\beta} f(t)dt \qquad (\alpha < \beta)$$

eeldades, et andur tekitab juhuslike arvude voo diapasooniga $[0,1]$ ja jaotusfunktsiooniga $\Phi(\xi) = 1$.

Ülesande lahendamisel eeldame, et funktsioon $f(t)$ on tõkestatud, s.t.

$$N \leqslant f(t) \leqslant M, \qquad t \in [\alpha, \beta]$$

Teeme muutujate vahetuse

$$t = (\beta - \alpha)\, z + \alpha, \qquad z \in [0,1],$$

millega integraal $\mathcal{J}$ taandub kujule

$$\mathcal{J} = (\beta - \alpha)\left[(M-N)\int_0^1 \frac{f\left[(\beta-\alpha)z+\alpha\right]-N}{M-N}dz + N\right],$$

kus integreeritav funktsioon

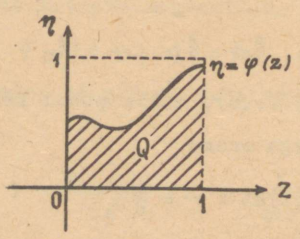
$$\frac{f\left[(\beta-\alpha)z+\alpha\right]-N}{M-N} = \varphi(z)$$

on vahemikus $0 \leq \varphi(z) \leq 1$. Seega saab taandada integraali $\mathcal{J}$ arvutamise erikujulise integraali

$$\mathcal{J}_1 = \int_0^1 \varphi(z)\,dz$$

arvutamisele.

Integraal $\mathcal{J}_1 = \int_0^1 \varphi(z)dz$ on arvuliselt võrdne kujundi Q pindalaga, mis on piiratud sirgetega $z = 0$, $z = 1$, $\eta = 0$ ja kõveraga $\eta = \varphi(z)$




Kujundi Q pindala

$$S_Q \approx \sum_1 S_{R_i},$$

kus $R_i$ on ristkülikud, mis asuvad kujundi Q sees.

Ristküliku $R_i$ pindala

$$S_{R_i} = (a_2 - a_1)(b_2 - b_1) =$$

$$= \int_{a_1}^{a_2} d\xi \int_{b_1}^{b_2} d\eta = p\,(a_1 \leq \xi \leq a_2)\,p\,(b_1 \leq \eta \leq b_2). \qquad (7)$$

Siin eeldame, et andur annab meile kaks juhuslike arvude voogu diapasioonis $[0,1]$ jaotusfunktsiooniga $\Phi_1(\xi) = \Phi_2(\eta) = 1$. Eeldus on kergesti realiseeritav, kuna anduri poolt tekita-

tava juhuslike arvude voo paaris ja paaritu indeksiga liikmeid võib vaadelda eri voogudena. Et sündmused „$a_1 \leq \xi \leq a_2$" ja „$b_1 \leq \eta \leq b_2$" on sõltumatud, võime võrdust (7) jätkata

$$= p(\text{„}a_1 \leq \xi \leq a_2\text{"} \text{ ja } \text{„}b_1 \leq \eta \leq b_2\text{"}) = p\{(\xi, \eta) \in R_1\}.$$

Summeerides pindalad $S_{R_i}$, saame

$$\sum_i S_{R_i} = \sum_i p\{(\xi,\eta) \in R_i\} =$$

$$= p\{(\xi, \eta) \in \bigcup_i R_i\}. \qquad (8)$$

Viimase võrduse puhul märgime, et kuigi ristkülikutel $R_i$ ja $R_j$ $(i \neq j)$ võib olla ühine külg, nii et sündmused $(\xi, \eta) \in R_i$ ja $(\xi, \eta) \in R_j$ pole üksteist välistavad, on tõenäosuste liitmisteoreem rakendatav, sest punkti $(\xi, \eta)$ ristküliku rajajoonele sattumise tõenäosus on 0.

Kui nüüd alajaotust peenendada, läheneb $\bigcup_i R_i$ piiramatult kujundile Q ja me saame võrdusest (8)

$$S_Q = p\{(\xi, \eta) \in Q\}.$$

Punkt $(\xi, \eta)$ asub kujundis Q aga juhul, kui

$$\eta \leq \varphi(\xi). \qquad (9)$$

Seega saame $\int_a^b \varphi(z)\,dz = p[\eta \leq \varphi(\xi)]$.

Viimast tõenäosust saab ligikaudu arvutada järgmiselt. Olles arvutanud võrratusest (6) katsete arvu n, tekitame anduriga juhuslikud arvud

$$\xi_1, \eta_1, \xi_2, \eta_2, \ldots, \xi_k, \eta_k, \ldots, \xi_n, \eta_n.$$

Iga paari $(\xi_k, \eta_k)$ korral kontrollime tingimuse (9) kehtivust. Tingimust (9) rahuldavate paaride arvu m abil leiamegi $\nu = \frac{m}{n}$, mis on tõenäosuse $p[\eta \leq \varphi(\xi)]$ ja integraali $J_1$ ligi-

kaudseks väärtuseks.

Analoogiliselt leiame, et k-kordse integraali puhul, kus $0 \leq \varphi(z_1, z_2, \ldots, z_k) \leq 1$, kehtib valem

$$\int_0^1 \int_0^1 \cdots \int_0^1 \varphi(z_1, z_2, \ldots, z_k)\, dz_1 dz_2 \ldots dz_k = p[\eta \leq \varphi(\xi_1, \xi_2, \ldots \xi_k)].$$

Siin tuleb juhuslike arvude voog jaotada k+1 eri vooks või rühmitada ühe voo liikmed (k+1)-liikmelisteks rühmadeks.

$$\xi_1, \xi_2, \ldots, \xi_k, \eta.$$

Seejärel tuleb selgitada nende rühmade arv m, mille puhul on täidetud tingimus

$$\eta \leq \varphi(\xi_1, \xi_2, \ldots, \xi_k).$$

Sagedus $\nu = \frac{m}{n}$ ongi integraali ligikaudseks väärtuseks.

Märgime, et määratud integraalide arvutamisel ristküliku-, trapetsivalemi ja Simpsoni valemi abil on sõlmpunktide arvu A ja kordsuse k vaheline seos $A = N^k$. Monte-Carlo meetodi puhul on k-kordse integraali arvutamisel üheks statistiliseks katseks vajalik ainult k+1 juhuslikku arvu. Sellest ilmneb Monte-Carlo meetodi efektiivsus just suure kordsusega integraalide arvutamisel.

Näide 3. Arvutame Monte-Carlo meetodi abil pöördfunktsiooni $y = f^{-1}(x)$ väärtusi, lähtudes funktsioonist $f(y) = x$. Seejuures eeldame, et funktsioon on pidev ja monotoonselt kasvav lõigul $[a, a+1]$.

Ülesande lahendamiseks on vajalik juhuslike arvude andur, mis tekitab juhuslike arvude voo diapasooniga $[0,1]$ ja jaotusfunktsiooniga $\Phi(\xi) = 1$. Liites igale juhuslikule arvule a, saame voo diapasoonis

$$a \leq \xi \leq a + 1,$$

kusjuures jaotusfunktsioon jääb samaks.

Fikseerime mingi x vahemikus

$$f(a) \leq x \leq f(a+1)$$

ja arvutame $f^{-1}(x)$.

Mistahes y puhul $y \in [a, a+1]$ kehtib võrdus

$$p(\xi \leq y) = \int_{a}^{y} d\xi = y - a. \qquad (10)$$

Et võrratused $\xi \leq y$ ja $f(\xi) \leq f(y) = x$ on funktsiooni f monotoonse kasvamise tõttu samaväärsed, siis saame võrdusest (10)

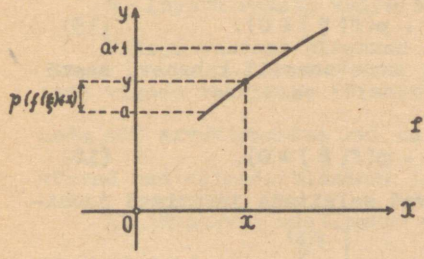
$$p(f(\xi) \leq f(y)) = y - a,$$

millest omakorda $f(y) = x$, $y = f^{-1}f(y) = f^{-1}(x)$ arvestamisel

$$p(f(\xi) \leq x) = f^{-1}(x) - a$$

ehk

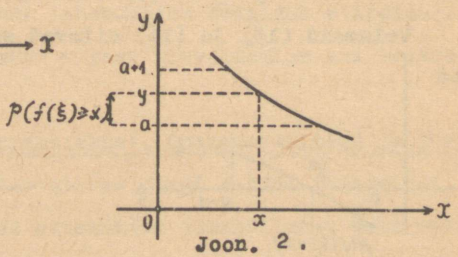
$$f^{-1}(x) = a + p(f(\xi) \leq x). \qquad (11)$$




Joon. 1.

Kui f(x) on pidev ja monotoonselt kahanev, saame valemi kujul

$$f^{-1}(x) = a + p(f(\xi) \geq x). \qquad (12)$$




Joon. 2.

Kui näiteks $f(y) = x$ on konkreetselt $y^k = x$, siis leiame punktile x

$$a^k \leq x \leq (a+1)^k$$

vastava funktsiooni $\sqrt[k]{x}$ väärtuse valemi (11) põhjal järgmiselt

$$\sqrt[k]{x} = a + p(\xi^k \leq x). \qquad (13)$$

Valemi (13) rakendamisel on otstarbekohane võtta a täisarv. Sel korral väljendab a $\sqrt[k]{x}$ täisosa.

Valemeid (11) ja (12) saab kasutada ka võrrandite lahendamisel.

Olgu antud võrrand

$$f(y) = 0, \qquad (14)$$

kus $x = f(y)$ on lõigul $y \in [a, a+1]$ pidev ja monotoonselt kasvav funktsioon. Eeldame, et võrrandil (14) leidub vahemikus $(a, a+1)$ üks lahend $y^*$. Tähistame $x = f(y)$ pöördfunktsiooni $y = f^{-1}(x)$ abil. Seega tahame leida $f^{-1}(0) = y^*$.

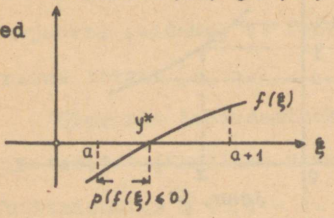
Kasutades valemit (11), saame

$$y^* = f^{-1}(0) = a + p(f(\xi) \leq 0). \qquad (15)$$

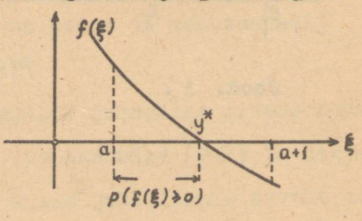
Kui $f(y)$ on lõigul $[a, a+1]$ monotoonselt kahanev, saame valemist (12)

$$y^* = f^{-1}(0) = a + p(f(\xi) \geq 0). \qquad (16)$$

Valemeid (15) ja (16) aitavad selgitada järgmised joonised




Joon. 3.




Joon. 4.

Tšebõševi võrratusest järeldub, et meetodi viga Monte-Carlo meetodi puhul väheneb katsete arvu n suurendamisel. Aritmeetiline viga aga katsete arvust n ei sõltu.

## § 24. Kontrolli meetoditest.

Ülesannete lahendamisel on meie eesmärgiks usaldusväärsete lõpptulemuste saamine. Lõpptulemuste saamiseks peab arvutusmasin teostama mõnikord küllaltki suuremahulise arvutustöö, millele muidugi eelneb programmeerija-arvestusmatemaatiku töö. Kogu selle töö võib jaotada järgmisteks etappideks:

1) ülesande lahendusmeetodi ja algoritmi valimine,
2) programmi loogilise skeemi ja programmi koostamine,
3) programmi ja algandmete perforeerimine,
4) programmi ja algandmete viimine masinasse,
5) ülesande lahendamine arvutusmasinas,
6) lõpptulemuste väljastamine arvutusmasinast.

Vähegi suurema ülesande lahendamisel tuleb alati arvestada vigade tekkimise võimalust. Vead võivad seejuures esineda nii arvutusmasina kui ka programmeerija töös, samuti võivad nad esineda ülesande lahendamise kõikidel etappidel.

Alljärgnevas vaatlemegi vigade kõrvaldamise ehk kontrolli meetodeid.

1. Arvutusmasina korrasolekust. Süstemaatilised ja juhuslikud vead. Teatavasti jaotatakse mingi seadme töötamisel esinevad kõrvalekaldumised ettenähtud tööreŽiimist (ehk vead) kahte liiki: süstemaatilised ja juhuslikud vead. Süstemaati-

lised vead tekivad mõne detaili mittekorrasoleku puhul (näiteks on läbi põlenud mõni lamp).

Siinkohal võiks tuua mõned andmed ühe konkreetse arvutusmasina "Ural-1" tehnilistest riketest ühe aasta jooksul (1958.a. augustist 1959.a. augustini). Nimetatud ajavahemikus oli sellel masinal üldse 238 tehnilist riket, mis jagunesid järgmiselt: juhtimisseadmes 25 riket (10,5%), aritmeetilises seadmes 37 riket (15,6%), magnettrumlil 45 riket (18,9%), trükkimisseadmes 46 riket (19,3%), perfolindil ja magnetlindil 61 riket (25,6%), toitevoolu süsteemis ja juhtimispuldil 24 riket (10,1%).

Süstemaatiliste vigade (tehniliste rikete) avastamine ja kõrvaldamine on tehnilise personali ülesandeks. Seda teostatakse spetsiaalsete kontrolltestide abil, kusjuures arvutusmasin käivitatakse sageli ka raskendatud töörežiimis. Tehniliste rikete asukoha täpseks kindlaksmääramiseks kasutatakse peale selle veel vastavat aparatuuri (ostsillograafid jms.).

Reegliks on, et matemaatikutele tuleb anda töötamiseks tehniliselt täiesti korrasolev masin, mis ei tee süstemaatilisi vigu. Töö käigus tekkinud tehniliste rikete puhul tuleb masina töö peatada ja masin tehnilisele personalile kordaseadmiseks üle anda.

Juhuslikud vead on mitmesuguse iseloomuga, põhjustatud näiteks masina vibratsioonist, toitevoolu väikestest muutustest jms. Tavaliselt ei olegi võimalik täpselt ütelda ühe või teise juhusliku vea põhjust. Juhuslikke vigu võib teha

ja teeb ka täiesti korrasolev masin. Nende esinemisel ei tule masina tööd peatada, kuigi nad võivad rikkuda tulemuste õigsust. Asi seisneb nimelt selles, et juhuslike vigade esinemise võimalust tuleb arvestada juba programmi koostamisel. Vähegi suurema ülesande programm tuleb nii koostada, et juhuslike vigade esinemisest hoolimata saaksime kas õige tulemuse või vähemalt saaksime informatsiooni tulemuse mitteõigsusest.

Edasi vaatleme vigade kõrvaldamise ja arvutuste kontrolli meetodeid üksikutel töötappidel. Et ülesande lahendusmeetodi ja algoritmi valikust kõneleme üksikasjalikumalt paragrahvis 25, siis me neid küsimusi siin ei käsitle.

2. Tuleb silmas pidada, et vead võivad tekkida juba programmi koostamisel ja blankettidele kandmisel. Programmi hoolika kontrollimisega saab need enamikus kõrvaldada. Programmi kontrollimisel on suureks abiks programmi loogiline skeem, magnettrumli pesade jaotuse plaan ja selgitavad märkused programmi olulisemates kohtades.

Programmi lõplik kontroll teostatakse aga arvutusmasina enese abil - programmi silumisel.

3. Edasised vead võivad tekkida programmi perforeerimisel. Perforeerimist võib kontrollida mitmeti. Kõigepealt võib perfolinti kontrollida ja võrrelda programmiga visuaalselt. On mõeldav ka programm teiskordselt perforeerida (soovitav teise isiku poolt) ja kahte perfolinti võrrelda kontrollivas seadmes. Erinevused kahe perfolindi vahel selgitatakse välja automaatselt. Kui kaks perfolinti ühtivad, tuleb lugeda, et need on õigesti perforeeritud (kaks ühtelangevat perforeerimisviga eri lintidel tuleb lugeda äärmiselt ebatõenäoseks).

Lõpuks on võimalik perfolinti nii kontrollida, et perfolindilt masinasse sisseviidud materjal trükitakse masinast 8-ndsüsteemis. Masinast väljatrükitud programmi võrreldakse blankettidele kantud programmiga. Nende ühtimisel tuleb lugeda,et vastav materjal on õigesti perforeeritud.

4. Järgmise etapi moodustab perfolindile kantud materjali viimine masinasse. Ka siin on võimalik vigade tekkimine. Materjali masinasse viimise õigsust kontrollitakse kontrollsummade abil. Pärast materjali masinasse viimist arvutatakse selle materjali kontrollsumma:

```
k+0    20    0000          kontrollsumma leitakse
k+1    25    4000+2n       materjalist, mis asub
k+2   -26    a+4000+2n     pesades
k+3    24    k+2           a+4000,a+4002,...,a+4000+2n.
k+4    16    b+4000        Leitud kontrollsumma viiakse
k+5    37    b+4000        pessa b+4000. Pärast masina
                           peatumist on see loetav
                           signalisatsioonipaneelilt
                           (pealkirja "summaator" all).
```

Seejärel viiakse sama materjal teiskordselt masinasse ja arvutatakse uuesti sisseviidud materjali kontrollsumma. Kui need kaks kontrollsummat ühtivad, siis loetakse, et materjal on viidud masinasse õigesti.

Et vältida materjali kahekordse masinasse viimisega seotud ajakulu, talitatakse järgmiselt. Pärast materjali perforeerimist arvutatakse selle materjali kontrollsumma ja perforeeritakse see samale lindile (näiteks programmi lõppu pessa

b+4000). Ülesande lahendamisel viiakse masinasse perforeeritud materjal ja arvutatakse sisseviidud materjali kontrollsumma. Arvutatud kontrollsummat võrreldakse perforeeritud kontrollsummaga. Nende ühtimisel loetakse, et materjal on viidud masinasse õigesti.

Kontrollsummat arvutav ja võrdlev programmi osa võib olla näiteks järgmine:

| | | | |
|---|---|---|---|
| k+0 | 20 | 0000 | |
| k+1 | 25 | 4000+2n | |
| k+2 | -26 | a+4000+2n | |
| k+3 | 24 | k+2 | |
| k+4 | 16 | b+4002 | arvutatud kontrollsumma viiakse pessa b+4002 |
| k+5 | 02 | b+4002 | |
| k+6 | 14 | b+4000 | perforeeritud kontrollsumma on pesas b+4000 |
| k+7 | 21 | k+4011 | kui kontrollsummad ühtivad, siis $\omega$ =0 ja juhtimine antakse pesas k+11 asuvale käsule |
| k+10 | 37 | b+4002 | kontrollsummade mitteühtimise korral masin peatub |
| k+11 | ..... | | asutakse ülesande lahendamisele |

Märgime, et analoogilist võtet tuleb kasutada ka materjali toomisel magnetlindilt magnettrumlile. Materjali toomisel magnettrumlilt magnetlindile saaks analoogilist võtet kasutada aga siis, kui masinal oleks käsk magnetlindil oleva materjali kontrollsumma arvutamiseks ilma selle materjali toomiseta magnettrumlile (see käsk puudub TRÜ arvutuskeskuse masinal "Ural-1"). Selle käsu puudumisel tuleb magnetlindile

kantud materjal tuua tagasi magnettrumlile (vabadesse pesadesse), arvutada selle kontrollsumma ja võrrelda seda sama materjali kontrollsummaga, mis on arvatud enne selle materjali kandmist magnetlindile. Kui vastavad kontrollsummad ei ühti, viiakse sama materjal esialgsel kujul (see säilitati) uuesti magnetlindile.

5. Materjali perforeerimisel väljundperforaatori abil võib perforeerimise õigsuse kontrollimiseks samuti kasutada kontroll-liitmist. Selleks arvutatakse perforeeritava materjali kontrollsumma ja perforeeritakse see samale lindile. Seejärel viiakse vastav perforeeritud materjal magnettrumlile ja arvutatakse uuesti vastava materjali kontrollsumma. Kontrollsummasid võrreldakse, ja talitatakse nagu eespool vastavatel juhtudel.

Materjali väljastamist trükkimisseadme abil võib samuti kontrollida summeerimise teel. Nii võib masinas arvutada näiteks lõpptulemuste summa murrulise osa. Pärast tulemuste väljatrükkimist leiame tulemuste summa murrulise osa väikese arvutusmasina abil ja võrdleme leitud kontrollsummasid.

Trükkimise kontrollimiseks võib kasutada ka teisi võtteid. Nii näiteks võib 10-ndsüsteemi arvude (kümnend-kahend koodis) trükkimisel neid arve trükkida kaks korda, kusjuures teistkordsel trükkimisel nihutatakse arv ühe 10-ndkoha (ehk nelja 2-ndkoha) võrra paremale poole. Seda saab teha kolme käsu abil.

```
32    0000
06    ⟨2^-4⟩
32    0000
```

Seejuures tuleb silmas pidada, et korrutamine toimub ümardamisega. Näiteks, kui trükitav arv on

- 718390536,

siis pärast korrutamist arvuga $2^{-4}$ saame

- 71839054.

Et iga kümnendkoht (peale viimase koha) trükitakse trükkimisseadme erinevate kangikeste abil, on võimalik avastada trükkimisseadme juhuslikke vigu.

Analoogiliselt võib 8-ndsüsteemi arvude (masina sõnade) trükkimisel trükkida vastavaid arve kaks korda, kusjuures teiskordsel trükkimisel võib arvu nihutada 3 kahendkohta paremale poole. Kui kasutada selleks otstarbeks lihtsuse pärast arvuga $2^{-3}$ korrutamist, tuleb arvestada ümardamist. Peale selle muutuvad siin ka arvu kaks esimest kohta. Nii näiteks saame masina sõnast

663 405 713 274

pärast arvuga $2^{-3}$ korrutamist masina sõna

426 340 571 330.

Ümardamise vältimiseks võib kasutada ka nihutamist kas 11 0000 või 11 4000 abil.

6. Kuigi programm on programmeerija poolt hoolikalt koostatud ja kontrollitud, võib see siiski sisaldada vigu. Seepärast tuleb teda enne ülesande lahendamist kontrollida veel masina abil. Niisugust programmi kontrollimist nimetatakse programmi silumiseks. Programmi silumist võib jaotada kolme ossa:

1. programmi loogilise struktuuri kontroll,

2) programmi aritmeetilise osa kontroll,

3) programmi kompleksne silumine.

Programmi loogilise struktuuri kontrollimisel selgitatakse, kas juhtimise üleandmine blokkidelt blokkidele toimub õigesti. Aritmeetilist osa seejuures tavaliselt ei kontrollita. Aritmeetilise osa vaatlusest väljajätmiseks võib programmis asendada mõned käsud tingimata suunamiskäskudega, viies need vajalikkudesse pesadesse juhtimispuldilt. Sellel silumise etapil võib kasutada "peatust aadressi järgi". Masin peatatakse enne kontrollitavat juhtimise üleandmise käsku. Veendudes, et see funktsioneerib õigesti, asutakse kontrollima järgmist juhtimise üleandmise käsku.

Muide, kontrollida võib ka teisiti, kasutades "ühetaktilist režiimi". Sel juhul pole vaja programmi täiendavaid tingimata suunamise käske paigutada. Põhjuseks on asjaolu,et siis kontrollime niikuinii ainult vastavaid juhtimise üleandmise kohti.

Kõik parandused ja muudatused tuleb täpselt registreerida.

Kui programmi loogiline struktuur on kontrollitud ja vajalikud parandused tehtud, asutakse aritmeetilise osa kontrollimisele. Seda teostatakse üksikute programmilõikude kaupa. Programmilõike kontrollitakse sobivalt valitud kontrollülesannetega, mille vahetulemused ja vastused on juba ette teada (enne välja arvutatud). Sagedasti teisendatakse need tulemused veel 2-ndsüsteemi, et masina töö õigsust oleks võimalik kontrollida visuaalselt (signalisatsioonipaneeli abil). Pro-

grammilõikude aritmeetika kontrollimisel kasutatakse tavaliselt "peatust aadressi järgi". Võib kasutada ka "programmi trükkimise II režiimi" või "ühetaktilist režiimi".

Ka siin tuleb masina töö ja tehtavad parandused täpselt fikseerida. Silumise käigus võib selguda, et mõni programmilõik pole tegelikult üldse vajalik. Sel korral pole harilikult otstarbekohane programmi uuesti ümber kirjutada. Mittevajaliku programmiosa ette kirjutatakse lihtsalt tingimatu suunamise käsk, mis suunab arvutuskäigu sellest programmi osast mööda. Sagedamini esineb olukordi, kus programmeerimisel on mõni vajalik programmi osa vahele jäänud. Ka sel juhul ei ole otstarbekohane programmi uuesti ümber kirjutada. Parandus tehakse järgmiselt: vajalik programmi osa kirjutatakse tühjadesse pesadesse programmi lõppu. Suunamine selle vajaliku programmi osa juurde ja sealt tagasi teostatakse tingimatu suunamise käskude abil.

Programmi silumisel tuleb koostada täpne plaan: tuleb valida ja lahendada kontrollülesanded, tuleb leida üksikute blokkide kontrollimise järjekord, tuleb täpselt kavandada,mida on vaja välja trükkida jne.

Täpselt kavandatud silumisplaani korral pole raske koostada ka vastavat abiprogrammi, mis võimaldaks kontrollida põhiprogrammi funktsioneerimist samuti, nagu seda teeks inimene. Need nn. silumisprogrammid on juba tingimata vajalikud arvutusmasinatest "Ural-1" kiiremate masinate puhul.

7. Masina töö käigus tekkivate juhuslike vigade tõttu tuleb kontrollida peale kõige muu ka arvutuste õigsust. Võtted arvutuste õigsuse kontrollimiseks tuleb valida ja kavan-

dada juba programmi koostamisel.

Üks kõige lihtsamaid, aga ka kõige aeganõudvamaid meetodeid on ülesande üksikute etappide kahe või kahe-kolmekordne lahendamine. Meetodi põhimõte seisneb järgmises: pärast teatud etapi kahekordset lahendamist võrreldakse saadud tulemusi. Kui vastavad tulemused ühtivad, loetakse nad õigeks ja ülesande lahendamist jätkatakse. Kui tulemused ei ühti, jätetakse masin seisma (kahekordse lahendamise puhul), kahe-kolmekordse lahendamise puhul arvutatakse aga veel kolmas kord. Kolmest tulemusest loetakse õigeks see, mis esines kahel korral.

Et need kontrollivõtted nõuavad suurt ajakulu, siis võib neid soovitada ainult sel korral, kui paremad meetodid puuduvad.

Märgime, et mitmekordne lahendamine ei garanteeri alati ka arvutuste s i s u l i s t õigsust. Nimelt võib töö käigus programmi mõni käsk juhuslikult muutuda. Seega tuleb suurte ülesannete puhul kontrollida kontrollsummade abil ka programmi säilimist. Sellejuures tuleb arvestada loomulikult muudatusi, mis on programmis endas ette nähtud.

Arvutuste õigsuse kontrolliks on kõige otstarbekohasem kasutada seoseid, mis kehtivad lahendatava ülesande puhul, kuid mida pole kasutatud programmeerimisel. Näiteks võib $\sin x$, $\cos x$ arvutamisel kasutada kontrolliks seost

$$\sin^2 x + \cos^2 x = 1,$$

mis peab kehtima ligikaudu, s.t. kujul

$$|\sin^2 x + \cos^2 x - 1| < \varepsilon. \qquad (\varepsilon > 0)$$

Lineaarse võrrandisüsteemi lahendamisel Gaussi meetodiga on kasulik moodustada kontrollveerg võrrandite kordajate summadest.

| | kontrollveerg |
|---|---|
| $2x+3y = 5$ | 10 |
| $-5x+4y = 8$ | 7 |

Kontrollveeru elementi teisendatakse samuti nagu vastava võrrandi kordajaid. Ülesande lahendamise käigus kontrollitakse, kas võrrandi kordajate summa on (ligikaudne) võrdne kontrollveeru vastava elemendiga.

Võrrandite, võrrandisüsteemide, diferentsiaalvõrrandite ja diferentsiaalvõrrandisüsteemide lahendamisel on võimalik kontrollida lahendeid nende asendamise teel lahendatavasse võrrandisse või võrrandisüsteemi. Kontrollitav võrdus peab olema täidetud jälle ligikaudu. Nii näiteks peab võrrandi $f(x)=0$ lahend $x^*$ rahuldama tingimust

$$|f(x^*)| < \varepsilon ,$$

kus $\varepsilon$ on küllalt väike positiivne arv.

## § 25. Arvutusmeetodi valikust.

Mingi ülesande lahendamiseks elektronarvutusmasina abil on vaja kõigepealt valida sobiv lahendusmeetod (nn. numbrilise lahendamise meetod). Alljärgnevas vaatlemegi, milliseid asjaolusid tuleb silmas pidada lahendusmeetodite valikul.

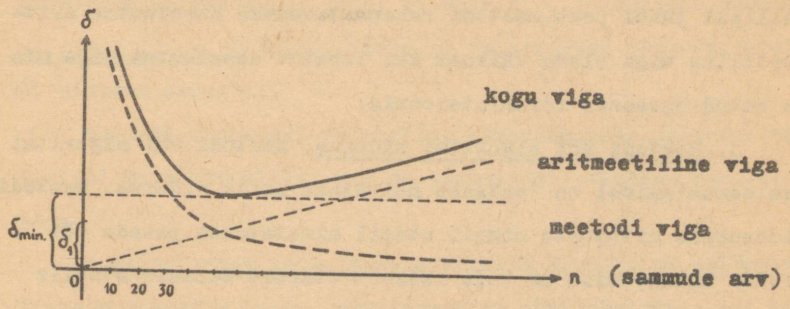
1. Elektronarvutusmasina salvestusseadmest tingitud nõuded. Arvutusmasina "Ural-1" operatiivse salvestusseadme mahutavus on 1024 arvu, magnetlint- ja perfolintsalvestusseadme

mahutavus on vastavalt 40 000 ja 10 000 arvu. Tehete teostamisel välistes salvestusseadmetes olevate arvudega tuleb nad aga enne viia magnettrumlile. Informatsioonivahetus väliste salvestusseadmete ja magnettrumli vahel toimub kiirusega 75 arvu/sek, millele lisandub tsooni otsimise aeg. Et arvude kandmise aeg ühest seadmest teise tuleb aritmeetiliste tehete sooritamise ajale lisaks, siis on soovitav valida selline arvutusmeetod, mille järgi koostatud programm ühes algandmetega mahuks magnettrumlile.

2. Meetod peab rahuldama nõudeid täpsuse kohta. Teatavasti saame ülesannete numbrilisel lahendamisel tulemused ikka ligikaudu, s.t. teatava veaga. Tulemuste viga koosneb meetodi veast ja peale selle veel veast, mis tekib ligikaudsete arvudega opereerimisel. Viimast viga nimetatakse aritmeetiliseks veaks.

Meetodi viga väheneb meetodi sammude arvu (näiteks kordamiste arv iteratsioonimeetodite puhul) suurendamisel. Kuid sammude arvu suurendamine põhjustab ka aritmeetiliste tehete arvu suurenemise ja ühes sellega üldiselt ka absoluutse aritmeetilise vea suurenemise.

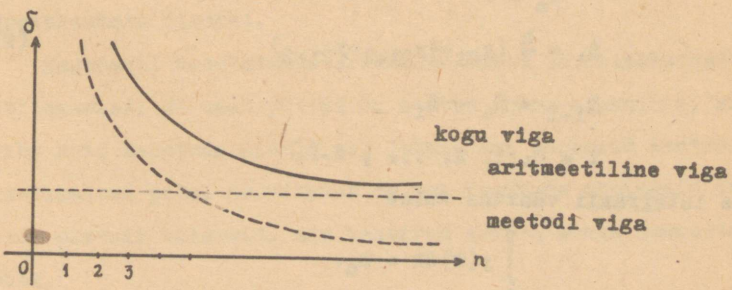
Näiteks diferentsiaalvõrrandi lahendamisel Euleri,Adamsi või Runge-Kutta meetodi abil võib eeldada, et absoluutne aritmeetiline viga on võrdeline sammude arvuga n. Meetodi viga läheneb aga n kasvades monotoonselt nullile.
Kujutame seda olukorda joonise abil.




Joon. 1.

Joonisel kujutatud juhtumil saame parima tulemuse n = 50 korral ja sammude arvu edasine suurendamine ei anna aritmeetilise vea kasvamise tõttu enam paremat täpsust. Kui arvutustelt nõutakse suuremat täpsust, kui seda saame parimal juhul n = 50 korral, siis antud meetod arvutusteks ei sobi. Teisiti öeldes, kui lubatud vea ülemmäär $\delta_1 < \delta_{min}$ , siis antud meetod ei sobi.

Iteratsioonimeetodite ja Monte-Carlo meetodi puhul on absoluutne aritmeetiline viga konstantne suurus (ei sõltu üldse tehete arvust). Meetodi viga aga läheneb ka kasvades nullile.




Joon. 2.

Sellisel juhul peab meetodi rakendatavuseks absoluutne aritmeetiline viga olema väiksem kui lubatav absoluutne viga, mis on antud ülesande formulatsioonis.

3. Meetodi või algoritmi sidusus. Meetodi või algoritmi iseloomustamisel on tähtsaks näitajaks selle sidusus. Meetodi sidususeks arvutuste mingil etapil nimetatakse pesade või arvude hulka, mida on vaja sellelt etapilt teisele etapile üleminekul. Üksikute etappide sidususte maksimumi võib nimetada meetodi kogusidususeks. See annab ülesande lahendamiseks vajalike tööpesade arvu. Meetodi mõningal teisendamisel on sageli võimalik selle sidusust vähendada.

Vaatleme näiteks määratud integraali arvutamist Simpsoni valemi abil.

$$\int_a^b y(x)dx = \frac{h}{3}\,(y_0+4y_1+2y_3++4y_4+\ \ldots+\ 2y_{2n-2}+4y_{2n-1}+y_{2n}) \quad (1)$$

Suuruste $y_i$ arvutamiselt üleminekul integraali arvutamisele valemi (1) järgi on vaja salvestada suurused $y_0, y_1, \ldots, y_{2n}$ kokku $2_{n+1}$. Seega on meetodi sidusus etapil $2_{n+1}$. Kuid Simpsoni meetodi võib esitada ka kujul:

$$S_0 = 0$$

$$S_i = \frac{h}{3}\,(y_{2i}+4y_{2i+1}+y_{2i+2}) \quad (2)$$

$$S_{i+1} = S_i + S_i$$

$$i = 0,\ 1,\ 2,\ \ldots,\ n-2,$$

kus integraali väärtus tuleb

$$\int_a^b y(x)dx = S_n.$$

Arvutades valemite (2) järgi on vaja $S_{i+1}$ leidmiseks suurusi $y_{2i}$, $y_{2i+1}$, $y_{2i+2}$ ja $S_i$. Seega on teisendatud Simpsoni meetodi sidusus ainult 4.

Meetodi sidususel on oluline tähendus materjali salvestamisel magnettrumlile, mis on suhteliselt väikese mahutavusega.

4. Meetod peab võimaldama arvutuste taandamist aritmeetilistele tehetele (resp. arvutusmasina põhitehetele). Kui meetodi abil taandatakse arvutused transtsendentsetele funktsioonidele, aeglaselt koonduvatele ridadele või võrrandite lahendamisele, on nende otstarbekus küsitav. Neid võib kasutada ainult siis, kui teised meetodid puuduvad.

5. Meetod peab võimaldama programmi tsüklite kasutamist. Tsüklite kasutamisel väheneb programmi pikkus ja programmi enese koostamine on hõlpsam.

6. Meetod peab hõlbustama mastaapide valikut ja vajaduse korral ka nende automaatset muutmist.

7. Meetod peab hõlbustama arvutuste kontrolli. Teatavasti esinevad juhuslikud vead ka täiesti korrasoleva masinaga töötamisel, mistõttu tuleb alati kontrollida nii vahe- kui ka lõpptulemuste õigsust.

Sagedasti kasutatakse arvutuste kahe- ja kolmekordset läbitegemist. Et need võtted on seotud suure ajakuluga, siis võiks neid kasutada ainult sel juhul, kui paremaid kontrollimisvõimalusi pole. Soovitavam on aga kasutada mõningaid ette teada olevaid valemeid, mis kehtivad arvutatavate suuruste puhul.

Mõned arvutusmeetodid ei vaja aga üldse arvutuste õigsuse kontrolli - nimelt iteratsioonimeetodid. Juhuslik viga masina töös võib arvutuskäiku ainult mõnevõrra pikendada, mitte aga muuta lõpptulemust. Vahetulemuste kontrolli pole üldiselt vaja ka Monte-Carlo meetodi puhul.

See asjaolu on viimati nimetatud meetodite teiseks heaks omaduseks.

8. Meetodi valikul tuleb silmas pidada ka selle majanduslikku ökonoomsust. Üldiselt öeldes, kui ülesande lahendamise tähtaeg seda võimaldab, tuleb meetod valida selline, et see võimaldaks minimiseerida ülesande lahendamise kogumaksumuse (st. programmeerimise maksumuse ja masina-aja maksumuse summat). Siin tuleb arvestada masina "Ural-1" töötunni maksumust 10 rbl.20 kop., samuti matemaatikute-programmeerijate töötasu.

## KIRJANDUS


1. Kaasik, Ü., Salum,H., Sinisoo,M., Elektronarvutusmasinad, Tallinn 1960.
2. TRÜ Arvutuskeskuse bülletään, 1 vihik, Tartu 1961.
3. Warmus, M. Transformed Point Arithmetic for Digital Computers, Bulletin de l'Academie Polonaise des Sciences. Serie des sciences math.astr.et physiques. Vol. IX. Nr.4. 1961, 237-239.
4. Балуев А.Н. (редактор), Методические указания по работе на универсальной цифровой машине "Урал-I", Ленинград I96I.
5. Богомолова Е.С., Гельфгат С.А., Ермилова Н.И., Описание системы команд электронной вычислительной машины "Урал", Москва I96I.
6. Бондаренко В.Н., Плотников И.Г., Полозов П.П., Программирование для цифровой вычислительной машины "Урал", Москва I958.
7. Бураков М.В., Опыт эксплуатации цифровой вычислительной машины "Урал", Москва I962.
8. Жоголев Е.А.,Трифонов Н.Г , Курс программирования, Москва I964.
9. Китов А.И., Криницкий Н.А., Электронные цифровые машины и программирование, Москва I959.
10. Криницкий Н.А., Миронов Г.А., Фролов Г.Д., Программирование, Москва I963.
11. Куусик В., Интерпретирующая программа для вычисления с трансформируемой запятой, Материалы по вопросам эксплуатации и усовершенствования электронной вычислительной машины М-3, Таллин I962, 46-60
12. Поспелов Д.А., Арифметические и логические основы вычислительных машин дискретного действия, ч. II, III, Москва I96I.

I3. Синтез электронных вычислительных и управляющих схем, Москва I954.
I4. Смольников Н.Я., Основы программирования для цифровой машины "Урал", Москва I96I.
I5. Ющенко Е.Л., Адресное программирование и особенности решения задач на машине "Урал", Киев I960.

SISUKORD.

lk.

I peatükk. ARVUTUSMASINATES KASUTATAVA ARITMEETIKA ALUSED

§ 1. Positsioonilised arvusüsteemid ..... 3

§ 2. Arvude teisendamine ühest positsioonilisest arvusüsteemist teise ..... 10

§ 3. Arvude kujutamine arvutusmasinais ..... 18

§ 4. Kahendarvude summaatorid. Arvude liitmine nende abil ..... 24

§ 5. Korrutamine ja jagamine fikseeritud komaga arvutusmasinates. Aritmeetiliste tehete teostamisest liikuva komaga arvutusmasinates 37

II peatükk. ARVUTUSMASINATE KONSTRUKTSIOONI LOOGILISED ALUSED

§ 6. Lausearvutuse põhimõisteid ..... 46

§ 7. Lausearvutuse valemite normaalkujud ..... 53

§ 8. Lausearvutuse valemite minimiseerimine .... 58

§ 9. Elektronskeemide elemendid ja loogiliste tehete realiseerimine nende abil ..... 66

§ 10. Elektronskeemide ja loogiliste skeemide analüüs ning süntees ..... 73

III peatükk. ÜLEVAADE ARVUTUSMASINA "URAL-1" EHITUSEST

§ 11. Salvestusseadmed ..... 82

a) Operatiivne salvestusseade ..... 82

b) Salvestusseade perfolindil ..... 87

c) Salvestusseade magnetlindil ..... 92

§ 12. Aritmeetiline seade ..... 95

§ 13. Juhtimisseade ..... 98

§ 14. Signalisatsioonipaneel ja juhtimispult .... 106

§ 15. Väljendusseadmed. Välised seadmed ..... 119

lk.

IV peatükk. PROGRAMMEERIMISE ALUSED

§ 16. Arvutusmasina "Ural-1" käskude süsteem ..... 124
§ 17. Näiteid ülesannete programmeerimise kohta 152
§ 18. Tsüklite programmeerimine ..... 167
§ 19. Konstantsete mastaapide valikust ..... 180
§ 20. Muutuvate mastaapide meetod. Programmeerimine liikuva koma meetodil ..... 189 189

V peatükk. PROGRAMMEERIMISE ÜLDISI KÜSIMUSI

§ 21. Standardprogrammide süsteem ..... 202
§ 22. Programmi loogiline skeem ja selle teisendamine ..... 213
§ 23. Monte-Carlo meetod ..... 226
§ 24. Kontrolli meetoditest ..... 235
§ 25. Arvutusmeetodi valikust ..... 245

KIRJANDUS ..... 251

Тартуский государственный университет
ЭССР, г. Тарту, ул. Юликооли, I8

И. Куйль

ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И ПРОГРАММИРОВАНИЕ
I

На эстонском языке

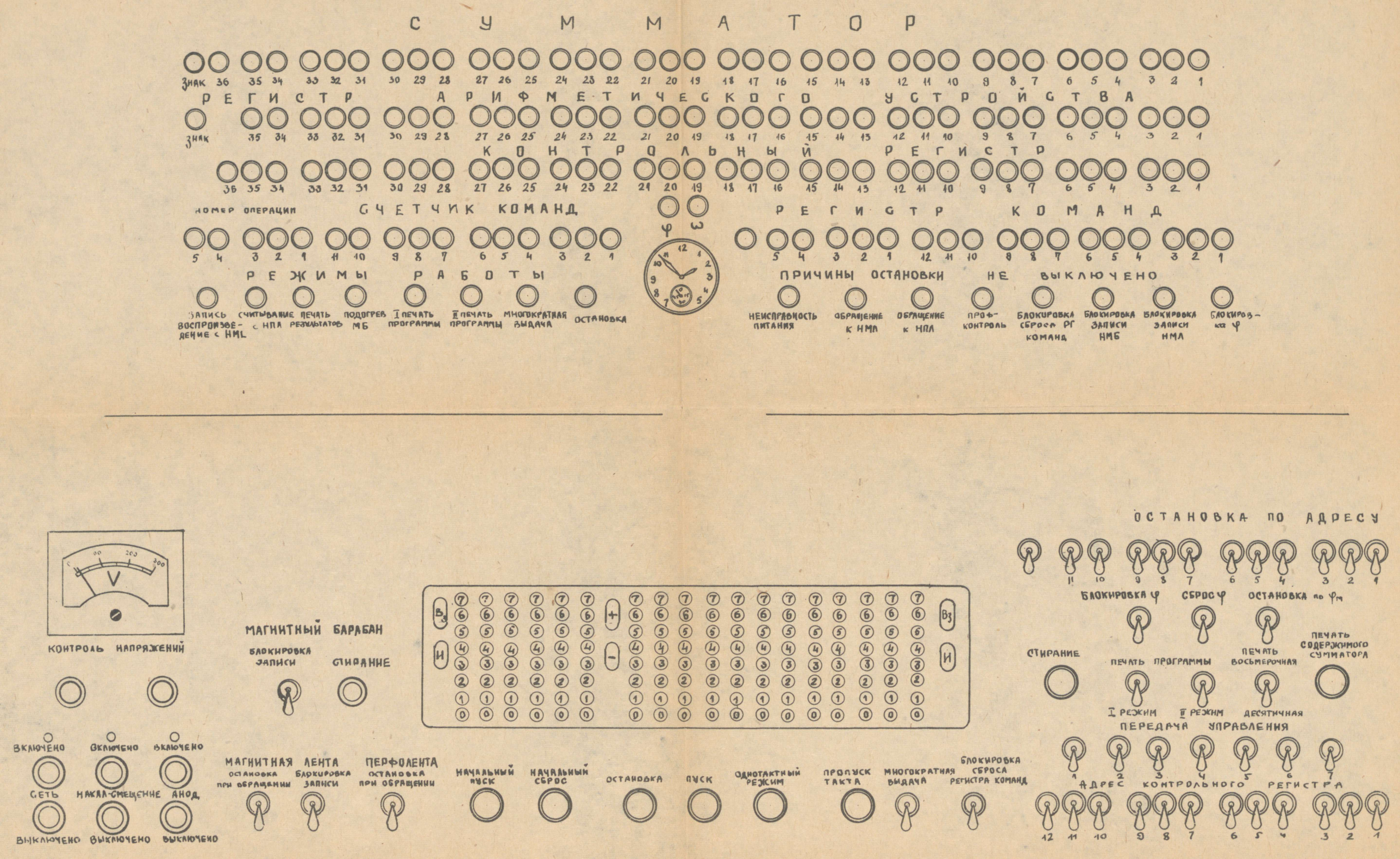
---

Vastutav toimetaja L. Luht
Korrektor E. Oja

==========

TRÜ rotaprint 1965. Trükipoognaid 15,9. Tingtrükipoognaid 14,5. Arvestuspoognaid 13,75. Tiraaž 300.
Paljundamisele antud 14. VI 65. MB 06302.
Tell. nr. 237.

Hind 41 kop.


СУММАТОР
Знак 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
РЕГИСТР АРИФМЕТИЧЕСКОГО УСТРОЙСТВА
Знак 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
КОНТРОЛЬНЫЙ РЕГИСТР
36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
НОМЕР ОПЕРАЦИИ
СЧЕТЧИК КОМАНД
5 4 3 2 1 11 10 9 8 7 6 5 4 3 2 1
φ ω
РЕГИСТР КОМАНД
5 4 3 2 1 12 11 10 9 8 7 6 5 4 3 2 1
РЕЖИМЫ РАБОТЫ
ЗАПИСЬ ВОСПРОИЗВЕДЕНИЕ С НМЛ
СЧИТЫВАНИЕ С НПЛ
ПЕЧАТЬ РЕЗУЛЬТАТОВ
ПОДОГРЕВ МБ
I ПЕЧАТЬ ПРОГРАММЫ
II ПЕЧАТЬ ПРОГРАММЫ
МНОГОКРАТНАЯ ВЫДАЧА
ОСТАНОВКА
ПРИЧИНЫ ОСТАНОВКИ НЕ ВЫКЛЮЧЕНО
НЕИСПРАВНОСТЬ ПИТАНИЯ
ОБРАЩЕНИЕ К НМЛ
ОБРАЩЕНИЕ К НПЛ
ПРОФКОНТРОЛЬ
БЛОКИРОВКА СБРОСА РГ КОМАНД
БЛОКИРОВКА ЗАПИСИ НМБ
БЛОКИРОВКА ЗАПИСИ НМЛ
БЛОКИРОВКА φ
V
КОНТРОЛЬ НАПРЯЖЕНИЙ
МАГНИТНЫЙ БАРАБАН
БЛОКИРОВКА ЗАПИСИ
СТИРАНИЕ
ВЗ
И
+
−
ВЗ
И
ВКЛЮЧЕНО
ВКЛЮЧЕНО
ВКЛЮЧЕНО
СЕТЬ
НАКАЛ-СМЕЩЕНИЕ
АНОД
ВЫКЛЮЧЕНО
ВЫКЛЮЧЕНО
ВЫКЛЮЧЕНО
МАГНИТНАЯ ЛЕНТА
ОСТАНОВКА ПРИ ОБРАЩЕНИИ
БЛОКИРОВКА ЗАПИСИ
ПЕРФОЛЕНТА
ОСТАНОВКА ПРИ ОБРАЩЕНИИ
НАЧАЛЬНЫЙ ПУСК
НАЧАЛЬНЫЙ СБРОС
ОСТАНОВКА
ПУСК
ОДНОТАКТНЫЙ РЕЖИМ
ПРОПУСК ТАКТА
МНОГОКРАТНАЯ ВЫДАЧА
БЛОКИРОВКА СБРОСА РЕГИСТРА КОМАНД
ОСТАНОВКА ПО АДРЕСУ
11 10 9 8 7 6 5 4 3 2 1
БЛОКИРОВКА φ
СБРОС φ
ОСТАНОВКА по φм
СТИРАНИЕ
ПЕЧАТЬ ПРОГРАММЫ
ПЕЧАТЬ ВОСЬМЕРОЧНАЯ
ПЕЧАТЬ СОДЕРЖИМОГО СУММАТОРА
I РЕЖИМ
II РЕЖИМ
ДЕСЯТИЧНАЯ
ПЕРЕДАЧА УПРАВЛЕНИЯ
1 2 3 4 5 6 7
АДРЕС КОНТРОЛЬНОГО РЕГИСТРА
12 11 10 9 8 7 6 5 4 3 2 1

Hind 41 kop.

A
26823
47 27 360

TÜ RAAMATUKOGU
1 0300 00472736 0