# ARVUSÜSTEEMID

meenutame Diskreetsest Matemaatikast :

arvu numbrid asuvad kindlatel asukohtadel — *arvujärkudes* $\mathbf{a_i}$ :

$$\ldots\ \mathbf{a_5}\ \ \mathbf{a_4}\ \ \mathbf{a_3}\ \ \mathbf{a_2}\ \ \mathbf{a_1}\ \ \mathbf{a_0}\ \ \mathbf{a_{-1}}\ \ \mathbf{a_{-2}}\ \ \mathbf{a_{-3}}\ \ \mathbf{a_{-4}}\ \ldots\ \mathbf{a_i}\ \ldots$$

arvusüsteemi **alus** ; järgu**kaal**

arvusüsteemi täisarvuline **alus** $\mathbf{p}$ .

Igal järgul $\mathbf{a_i}$ on **kaal** $\mathbf{p_i}$ , mis arvutub arvusüsteemi aluse $\mathbf{p}$ täisarvastmena: $\mathbf{p_i} = \mathbf{p^i}$

Järgukaalud: $\ldots\ \mathbf{p^5}\ \ \mathbf{p^4}\ \ \mathbf{p^3}\ \ \mathbf{p^2}\ \ \mathbf{p^1}\ \ \mathbf{p^0}\ \ \mathbf{p^{-1}}\ \ \mathbf{p^{-2}}\ \ \mathbf{p^{-3}}\ \ \mathbf{p^{-4}}\ \ldots\ \mathbf{p^i}\ \ldots$

Kui alus $\mathbf{p = 10}$ , siis on **kümnendsüsteem** , kus järkude kaaludeks on:

$$\ldots\ \mathbf{10^3}\ \mathbf{10^2}\ \mathbf{10^1}\ \mathbf{10^0}\ \mathbf{10^{-1}}\ \mathbf{10^{-2}}\ \mathbf{10^{-3}}\ \ldots$$

$$\ldots\ \mathbf{100}\ \ \mathbf{10}\ \ \mathbf{1}\ \ \mathbf{0.1}\ \ \mathbf{0.01}\ \ldots$$

*täisosa* • *murdosa*

*kõrgemad järgud* ⟷ *madalamad järgud*

*Koma* näitab, kus lähevad täisarvulised järgukaalud üle murdarvulisteks (ehk kus lõppeb täisosa ja algab murdosa).

Igas järgus $\mathbf{a_i}$ saab olla $\mathbf{p}$ erinevat *numbrimärki* ehk järguväärtust.

Kui $\mathbf{p = 10}$ , siis $\mathbf{a_i} \in \{\ \mathbf{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}\ \}$

Igal 10ndnumbril on tema traditsiooniline *väärtus* 0 ..... 9.

*Järgu väärtus* on selles arvujärgus asuva *numbri* väärtus.

*Arv* koosneb *numbritest.*

*näide:* arv **1024** koosneb neljast *numbrist:* '**1**' '**0**' '**2**' '**4**'

**Arvu väärtus**

Mistahes positsioonilises arvusüsteemis (ehk iga aluse $\mathbf{p}$ korral) avaldub arvu **väärtus** $\mathbf{N}$ järgneva **korrutiste summana** :

$$\mathbf{N} = \ldots + \mathbf{a_3} \cdot \mathbf{p^3} + \mathbf{a_2} \cdot \mathbf{p^2} + \mathbf{a_1} \cdot \mathbf{p^1} + \mathbf{a_0} \cdot \mathbf{p^0} + \mathbf{a_{-1}} \cdot \mathbf{p^{-1}} + \mathbf{a_{-2}} \cdot \mathbf{p^{-2}} + \ldots$$

*näide:*

10ndsüsteemne arv $\mathbf{123}_{10}$ on väärtusega "*sada kakskümmend kolm*" ainult sellepärast, et järgnev tehe annab sellise tulemuse:

$$\mathbf{123}_{10} = \mathbf{1} \cdot 100 + \mathbf{2} \cdot 10 + \mathbf{3} \cdot 1 = 123_{10}$$

Mõiste "*arvu väärtus*" on eranditult seotud ainult 10ndsüsteemiga.

"*Väärtuse leidmine*" ja "*10ndsüsteemi teisendamine*" on sünonüümid.

Pole olemas "*kahendsüsteemset väärtust*" ega "*kaheksandsüsteemset väärtust*"; on olemas 2ndsüsteemne **esitus** ja 8ndsüsteemne **esitus**.

**"kasutamata" arvujärgud $\mathbf{a_i}$ on täidetud 0-dega:**

Täisosa ees ja murdosa järel asuvad '**0**'-d ei mõjuta arvu väärtust:

$\mathbf{123.45}_{10}$ = . . . . 00000**123.45**0000000 . . . . $_{10}$

**Tüvenumbrid**

Arvu *tüvenumbrid* on arvu numbrid alates kõrgeimast mittenullisest numbrist kuni madalaima mittenullise numbrini.

Kuigi madalaim ja kõrgeim tüvenumber pole kumbki 0 , võivad nende "vahel" olla tüvenumbriteks ka '0'-d.

*näide:* arvus **0.0000120003000** on tüvenumbriteks **120003** .

Üleskirjutatud arvu süsteemikuuluvuse täpsustamiseks lisame talle süsteemi näitava indeksi: $\mathbf{372}_8$ ei ole mitte *"kolmsada seitsekümmend kaks"* vaid on 8ndsüsteemne arv *"kolm-seitse-kaks"*

*nüüd lahkume **10nd**süsteemist ja siseneme muudesse arvusüsteemidesse*

Asendades harjumuspärase arvusüsteemi aluse **p = 10** alusega **2** koos kõigi sellega kaasnevate tagajärgedega, saame **kahendsüsteemi**:

## KAHENDSÜSTEEM

*Kahendsüsteem* on lihtsaim võimalik positsiooniline arvusüsteem:

$$p = 2 \qquad a_i \in \{0, 1\}$$

Kuna positsioonilises arvusüsteemis peab olema tema alusega võrdne arv numbrimärke, siis kahendsüsteemsed arvud koosnevad ainult kahest numbrist: **0** ja **1**.

Arvusüsteemi aluse muutmisega kaasneb ka *järgukaalude* muutus, mis kahendsüsteemis on arvu 10 astmete asemel arvu **2** täisarvastmed:

**2nd**süsteemi **järgukaalud**: $\ldots\ 2^5\ \ 2^4\ \ 2^3\ \ 2^2\ \ 2^1\ \ 2^0\ \ 2^{-1}\ \ 2^{-2}\ \ 2^{-3}\ \ldots$

$\qquad\qquad\qquad\qquad\quad 32\ \ 16\ \ 8\ \ 4\ \ 2\ \ 1\ \ 0.5\ \ 0.25\ \ 0.125$

Järgnevalt on loetletud kõik kuni 6-järgulised kahendarvud (ehk 2ndarvud väärtusega **0** kuni **63** ) :

| | | | |
|---|---|---|---|
| $0_2 = 0_{10}$ | $10000_2 = 16_{10}$ | $100000_2 = 32_{10}$ | $110000_2 = 48_{10}$ |
| $1_2 = 1_{10}$ | $10001_2 = 17_{10}$ | $100001_2 = 33_{10}$ | $110001_2 = 49_{10}$ |
| $10_2 = 2_{10}$ | $10010_2 = 18_{10}$ | $100010_2 = 34_{10}$ | $110010_2 = 50_{10}$ |
| $11_2 = 3_{10}$ | $10011_2 = 19_{10}$ | $100011_2 = 35_{10}$ | $110011_2 = 51_{10}$ |
| $100_2 = 4_{10}$ | $10100_2 = 20_{10}$ | $100100_2 = 36_{10}$ | $110100_2 = 52_{10}$ |
| $101_2 = 5_{10}$ | $10101_2 = 21_{10}$ | $100101_2 = 37_{10}$ | $110101_2 = 53_{10}$ |
| $110_2 = 6_{10}$ | $10110_2 = 22_{10}$ | $100110_2 = 38_{10}$ | $110110_2 = 54_{10}$ |
| $111_2 = 7_{10}$ | $10111_2 = 23_{10}$ | $100111_2 = 39_{10}$ | $110111_2 = 55_{10}$ |
| $1000_2 = 8_{10}$ | $11000_2 = 24_{10}$ | $101000_2 = 40_{10}$ | $111000_2 = 56_{10}$ |
| $1001_2 = 9_{10}$ | $11001_2 = 25_{10}$ | $101001_2 = 41_{10}$ | $111001_2 = 57_{10}$ |
| $1010_2 = 10_{10}$ | $11010_2 = 26_{10}$ | $101010_2 = 42_{10}$ | $111010_2 = 58_{10}$ |
| $1011_2 = 11_{10}$ | $11011_2 = 27_{10}$ | $101011_2 = 43_{10}$ | $111011_2 = 59_{10}$ |
| $1100_2 = 12_{10}$ | $11100_2 = 28_{10}$ | $101100_2 = 44_{10}$ | $111100_2 = 60_{10}$ |
| $1101_2 = 13_{10}$ | $11101_2 = 29_{10}$ | $101101_2 = 45_{10}$ | $111101_2 = 61_{10}$ |
| $1110_2 = 14_{10}$ | $11110_2 = 30_{10}$ | $101110_2 = 46_{10}$ | $111110_2 = 62_{10}$ |
| $1111_2 = 15_{10}$ | $11111_2 = 31_{10}$ | $101111_2 = 47_{10}$ | $111111_2 = 63_{10}$ |

☺ ☻ arvu **väärtuse N** leidmine osutub **2nd**arvude jaoks eriti lihtsaks: teisendus lihtsustub nende järgukaalude summeerimiseks, kus asub järguväärtus **1**:

*näide:*

$$101011_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 =$$
$$= 32 + 8 + 2 + 1 = 43_{10}$$

*ülesanne:*

Leida järgnevate positiivsete 2ndarvude **väärtus**
(ehk teisendada 10ndsüsteemi)

$1\,0\,1_2 = \ldots_{10}$
$0\,1\,1\,0_2 = \ldots_{10}$
$0\,0\,0\,1\,1\,0\,1_2 = \ldots_{10}$
$0\,0\,0\,0\,1\,0\,0\,1\,1_2 = \ldots_{10}$
$0\,1\,0\,0\,0\,1\,0\,1\,1_2 = \ldots_{10}$
$0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0_2 = \ldots_{10}$

$1\,0\,1_2 = \mathbf{5}_{10}$
$0\,1\,1\,0_2 = \mathbf{6}_{10}$
$0\,0\,0\,1\,1\,0\,1_2 = \mathbf{13}_{10}$
$0\,0\,0\,0\,1\,0\,0\,1\,1_2 = \mathbf{19}_{10}$
$0\,1\,0\,0\,0\,1\,0\,1\,1_2 = \mathbf{139}_{10}$
$0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0_2 = \mathbf{44}_{10}$

**TÄISARVU teisendus 10ndsüsteemist 2ndsüsteemi**

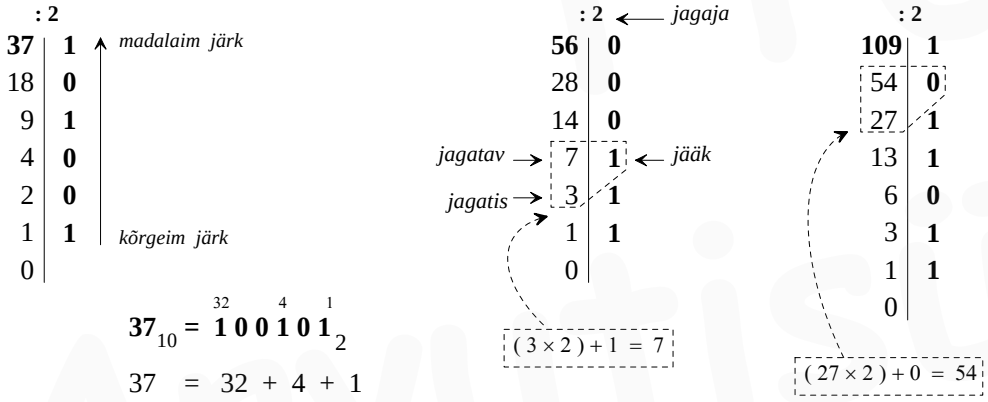
**Täisarvu** teisendus ühest arvusüsteemist teise toimub
**uue alusega jagamise teel** kus jagamine on *täisarvuline*:
murdarvu asemel saame *jagatise* ja *jäägi:*

**7 : 2 = 3** ( jääk **1** )

Väärtuse **N** leidmise suhtes vastupidine teisendus ehk 10ndsüsteemse täisarvu teisendamine 2ndsüsteemi toimub **2**-ga jagamise teel, kusjuures (täisarvulise) jagamise jäägid (0 ja 1) on saadava 2ndarvu järkude väärtusteks.

*näide:*

Teisendame 10ndtäisarvud **37**$_{10}$ **56**$_{10}$ **109**$_{10}$ 2ndkujule:




$37_{10} = {}^{32}\,1\,0\,0\,1\,0\,{}^{1}\,1_2$

37 = 32 + 4 + 1

**37**$_{10}$ = **100101**$_2$ **56**$_{10}$ = **111000**$_2$ **109**$_{10}$ = **1101101**$_2$

☺ ☻ väikeste **2ndarvude kiirkoostamine** **1**de "**sobitamise**" **teel õigetesse järkudesse**:

Vajaliku arvu kahendkuju saab koostada ka järguväärtuste **1** paigutamise teel vajalikesse 2ndjärkudesse :

. . . . . **64 32 16 8 4 2 1**

peast arvutades täidame (kõrgeimast järgust alates) vajalikud järgud "*ühtedega*" nii, et **1**-ga täidetud järkude kaalude summa võrduks soovitud 10ndarvuga.

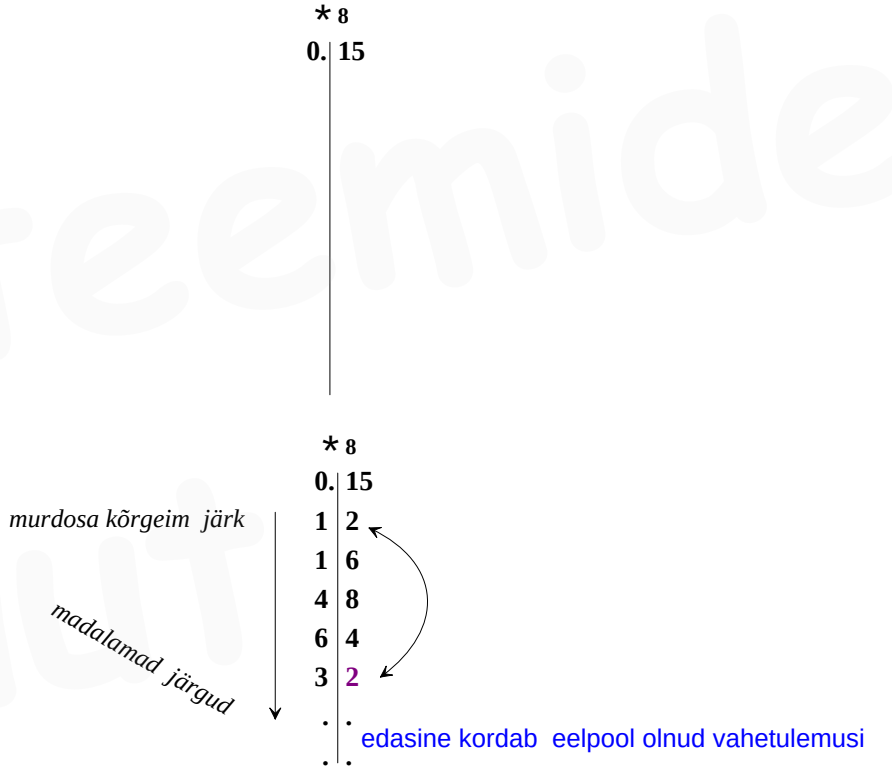
Arvu *murdosa* teisendusmeetod erineb oluliselt *täisarvu* teisendusest.

## MURDARVU (arvu murdosa) teisendus 10ndsüsteemist 8ndsüsteemi

Murdosa teisendatakse uue alusega **korrutamise** teel.

*näide:*

Teisendame 10ndmurdosa **0.15**$_{10}$ **8nd**kujule: **0.15**$_{10}$ = **0.???**$_8$




saime teisendusel :

**0.15**$_{10}$ = **0.11463**...$_8$ . . . . tekkib *lõpmatu perioodiline* murdosa

**0.15**$_{10}$ = **0.1146311463**...$_8$

*näide:*

Teisendame 10ndmurdosa **0.6875**$_{10}$ **8nd**kujule: **0.6875**$_{10}$ = **0.???**$_8$

| * 8 | |
|---|---|
| **0.** | **6875** |

| * 8 | |
|---|---|
| **0.** | **6875** |
| **5** | **5** |
| **4** | **0** |

teisendub uude arvusüsteemi **täpselt**

**0.6875**$_{10}$ = **0.54**$_8$

**Murdosa** teisendus lõppeb, kui

— ilmneb järkude (lõpmatult) kordumajääv osa ehk *periood*

või

— saame vahetulemuseks **0** millega edaspidi korrutades tuleksid ka kõik järgnevad madalamad järgud 0 ; (seljuhul teisendus murdosa **täpselt** )

"**puhtmurdarv**"

kui arvul on olemas ainult murdosa (ehk **täisosa** on arvul **0** ) siis sellist arvu nimetame **puhtmurdarvuks**.

**?**

kuidas teisendada arvu, millel on olemas nii **täisosa** kui ka **murdosa** **?**

. . . . ehk kuidas teisendada nn. üldist / tavalist **murdarvu** :

kui teisendataval arvul on olemas nii *täisosa* kui ka *murdosa*, siis **täisosa** teisendatakse eraldi ja **murdosa** teisendatakse eraldi.

*ülesanne:*

Teisendada järgnevad 10ndtäisarvud **2ndkujule** :

2 0 $_{10}$ = . . . $_2$

8 5 $_{10}$ = . . . $_2$

1 3 1 $_{10}$ = . . . $_2$

2 1 0 $_{10}$ = . . . $_2$

3 2 $_{10}$ = . . . $_2$

2 0 $_{10}$ = **1 0 1 0 0** $_2$

8 5 $_{10}$ = **1 0 1 0 1 0 1** $_2$

1 3 1 $_{10}$ = **1 0 0 0 0 0 1 1** $_2$

2 1 0 $_{10}$ = **1 1 0 1 0 0 1 0** $_2$

3 2 $_{10}$ = **1 0 0 0 0 0** $_2$

Lisaks alustele **p = 10** ja **p = 2** on olulisemateks arvusüsteemide alusteks veel **8** ja **16**, kuna nad on mõlemad arvu 2 astmed: $2^3$ ja $2^4$ .

## KAHEKSANDSÜSTEEM

**8**ndsüsteemi alus on 8 ja seega peab seal olema 8 võimalikku järguväärtust, milleks kasutatakse kaheksat esimest araabia numbrit **0** . . . **7** :

**p = 8** $\quad a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\}$

8ndsüsteemi järgukaalud: . . . $8^4$ $8^3$ $8^2$ $8^1$ $8^0$ $8^{-1}$ $8^{-2}$ $8^{-3}$ . . .

4096 512 64 8 1 0.125

Kaheksandarve nimetatakse ka *oktaalarvudeks*.

*ülesanne:*

Leia nende 8ndarvude **väärtus** :

**2 5**$_8$ = . . . $_{10}$

**7 4**$_8$ = . . . $_{10}$

**1 2 3**$_8$ = . . . $_{10}$

**2 5**$_8$ = $2\times8^1 + 5\times8^0$ = **21**$_{10}$

**7 4**$_8$ = $7\times8^1 + 4\times8^0$ = **60**$_{10}$

**1 2 3**$_8$ = $1\times8^2 + 2\times8^1 + 3\times8^0$ = **83**$_{10}$

## Teisendus 10ndsüsteemist 8ndsüsteemi

10ndtäisarvude teisendus 8ndsüsteemi toimub 8-ga jagamise teel, kusjuures igal jagamissammul saadakse jäägina arvu järgmine 8ndnumber **0 . . . . 7**.

*ülesanne:*

Teisenda 10ndtäisarvud **236**$_{10}$ **109**$_{10}$ 8ndkujule:

| : 8 | | |
|---|---|---|
| **236** | **4** | ↑ *madalaim järk* |
| 29 | **5** | |
| 3 | **3** | *kõrgeim järk* |
| 0 | | |

**236**$_{10}$ = **3 5 4**$_8$ (64 8 1)

| : 8 | | |
|---|---|---|
| **109** | **5** | |
| 13 | **5** | |
| 1 | **1** | |
| 0 | | |

**109**$_{10}$ = **1 5 5**$_8$ (64 8 1) ← *järgukaalud*

*( kontrollimisvõimalus )*

## KUUETEISTKÜMNENDSÜSTEEM

*hexadecimal (hex)*

Kuna **16**ndsüsteemis on arvusüsteemi alus 16, siis peab seal olema ka 16 võimalikku järguväärtust ja sellest tulenevalt ka 16 numbrimärki nende esitamiseks.

Lisaks 10-le araabia numbrile 0 . . . 9 on ülejäänud kuueks numbrimärgiks võetud ladina tähestiku algustähed A . . . F :

**p = 16** $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C\ D\ E\ F\}$

16ndsüsteemi järgukaalud: . . . **$16^4$ $16^3$ $16^2$ $16^1$ $16^0$ $16^{-1}$ $16^{-2}$** . . .

**4096 256 16 1 0.0625**

**16nd**numbrid **A . . . F** omavad järgnevaid väärtusi:

**A** = 10 **B** = 11 **C** = 12 **D** = 13 **E** = 14 **F** = 15

*ülesanne:*

Teisenda järgnevad 16ndarvud **10ndkujule** (ehk leia väärtus) :

**1 2**$_{16}$ = . . . $_{10}$

**4 D**$_{16}$ = . . . $_{10}$

**A 6**$_{16}$ = . . . $_{10}$

**F F**$_{16}$ = . . . $_{10}$

**1 0 0**$_{16}$ = . . . $_{10}$

**1 C D**$_{16}$ = . . . $_{10}$

1 2$_{16}$ = $1\times16^1 + 2\times16^0$ = **18**$_{10}$

4 D$_{16}$ = $4\times16^1 + 13\times16^0$ = **77**$_{10}$

A 6$_{16}$ = $10\times16^1 + 6\times16^0$ = **166**$_{10}$

F F$_{16}$ = **255**$_{10}$

1 0 0$_{16}$ = **256**$_{10}$

1 C D$_{16}$ = **461**$_{10}$

16ndsüsteem on "suurim" praktiliselt kasutatav arvusüsteem. Võimalik on koostada arvusüsteeme ka suurema alusega kui 16 , kuid selliseid suuremaid arvusüsteeme pole vaja.

## Teisendus 10ndsüsteemist 16ndsüsteemi

10ndtäisarvude teisendus 16ndsüsteemi toimub **16**-ga jagamise teel, kusjuures igal jagamissammul saadakse jäägina arvu järgmine 16ndnumber **0** . . . . **F**.

*ülesanne:*

Teisendada järgnevad 10ndarvud **16ndkujule** :

$77_{10} = \ldots_{16}$

$32_{10} = \ldots_{16}$

$256_{10} = \ldots_{16}$

$77_{10} = \mathbf{4D}_{16}$

$32_{10} = \mathbf{20}_{16}$

$256_{10} = \mathbf{100}_{16}$

Edaspidi vajame nendest arvusüsteemidest kõige rohkem *kahendsüsteemi*.

## Teisendus 2ndsüsteemist 8ndsüsteemi või 16ndsüsteemi

| **10**nd | **16**nd | **2**nd | **8**nd | **2**nd |
|---|---|---|---|---|
| **0** | 0 | 0000 | **0** | **000** |
| **1** | 1 | 0001 | **1** | **001** |
| **2** | 2 | 0010 | **2** | **010** |
| **3** | 3 | 0011 | **3** | **011** |
| **4** | 4 | 0100 | **4** | **100** |
| **5** | 5 | 0101 | **5** | **101** |
| **6** | 6 | 0110 | **6** | **110** |
| **7** | 7 | 0111 | **7** | **111** |
| **8** | 8 | **1000** | 10 | |
| **9** | 9 | **1001** | 11 | |
| 10 | A | **1010** | 12 | |
| 11 | B | **1011** | 13 | |
| 12 | C | **1100** | 14 | |
| 13 | D | **1101** | 15 | |
| 14 | E | **1110** | 16 | |
| 15 | F | **1111** | 17 | |

**2nd**süsteemi, **8nd**süsteemi ja **16nd**süsteemi alused on arvu **2** täisarvastmed: $\mathbf{2^1}$ $\mathbf{2^3}$ $\mathbf{2^4}$ . See annab neile kasuliku lisaomaduse, võimaldades nende süsteemide omavahelisi arvuteisendusi teha ka **numbrimärkide asendamise teel** ehk ilma "uue alusega" jagamata.

2ndarvu on võimalik teisendada (ümber kirjutada) tema 8ndkujule, asendades (alates 2ndarvu madalamatest järkudest) iga tema järkudekolmiku **000** .... **111** vastava 8ndnumbriga **0** .... **7** .

*näide:*

Võtame suvalise 2ndarvu: $\mathbf{1011010\,100111}_2$ eesmärgiga viia see arv **8nd**kujule ja seejärel ka **16nd**kujule.

võimalik oleks teisendada 2nd → 10nd → 8nd kuid see oleks asjatu töö

*parim teisendusviis:* Grupeerime 2ndarvu järgud 3-järgulistesse gruppidesse alates madalamatest järkudest, lisades vajadusel arvu ette 0-lle:

001|**011**|**010**|**100**|**111**

Asendame 2ndjärkude iga grupeeritud kolmiku temaga väärtuselt võrdse **8nd**numbriga **0** .... **7** : (eelpoolne vastavustabel **000** kuni **111**; **0** kuni **7** )

**1 3 2 4 7**

001|**011**|**010**|**100**|**111**

Seega $\mathbf{1011010\,100111}_2 = \mathbf{13247}_8$

Järgnevalt viime sellesama 2ndarvu ka **16nd**kujule.

Selleks grupeerime 2ndarvu järgud 4 järgu kaupa alates madalamatest järkudest ja asendame iga 2ndjärkude neliku temaga väärtuselt võrdse 16ndnumbriga 0 .... F :

$$\mathbf{1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,1}_2$$

| **1** | **6** | **A** | **7** |
|---|---|---|---|
| 0001 | **0110** | **1010** | **0111** |

Seega $\mathbf{1011010\,100111}_2 = \mathbf{16A7}_{16}$

*ülesanne:*

Leia eelmise näite **2nd**arvu, **8nd**arvu ja **16nd**arvu **väärtused** :

$\mathbf{1011010\,100111}_2 = \ldots_{10}$

$\mathbf{13247}_8 = \ldots_{10}$

$\mathbf{16A7}_{16} = \ldots_{10}$

$\mathbf{1011010\,100111}_2 = 2^{12} + 2^{10} + 2^9 + 2^7 + 2^5 + 2^2 + 2^1 + 2^0 = \ldots._{10}$

$\mathbf{13247}_8 = \mathbf{1}\times 8^4 + \mathbf{3}\times 8^3 + \mathbf{2}\times 8^2 + \mathbf{4}\times 8^1 + \mathbf{7}\times 8^0 = \ldots._{10}$

$\mathbf{16A7}_{16} = \mathbf{1}\times 16^3 + \mathbf{6}\times 16^2 + \mathbf{10}\times 16^1 + \mathbf{7}\times 16^0 = \ldots._{10}$

$\mathbf{1011010\,100111}_2 = 2^{12} + 2^{10} + 2^9 + 2^7 + 2^5 + 2^2 + 2^1 + 2^0 = \mathbf{5799}_{10}$

$\mathbf{13247}_8 = \mathbf{1}\times 8^4 + \mathbf{3}\times 8^3 + \mathbf{2}\times 8^2 + \mathbf{4}\times 8^1 + \mathbf{7}\times 8^0 = \mathbf{5799}_{10}$

$\mathbf{16A7}_{16} = \mathbf{1}\times 16^3 + \mathbf{6}\times 16^2 + \mathbf{10}\times 16^1 + \mathbf{7}\times 16^0 = \mathbf{5799}_{10}$

( kõik 3 arvu on **võrdsed** )

Numbrite **vastupidise** asendamisega kahendjärkude kolmikuteks või nelikuteks saab arvu **8nd**kujult või **16nd**kujult kergesti üle minna tema **2nd**kujule.

Sarnast arvujärkude asendamist saab rakendada ka 4ndsüsteemiga tegeledes, sest arvusüsteemi alus $4 = 2^2$ .

4ndsüsteem ei ole oluline arvusüsteem ja praktikas teda ei kasutata.

## 16ndsüsteemi tähtsus

Arvutimälus hoitakse andmeid **baitides**, mis on **8**-järgulised kahendkoodid. 16ndsüsteem võimaldab esitada (näidata) baitide sisu ( ja üldse igasuguseid kahendkoode) palju kompaktsemalt võrreldes nende "vahetu" esitamisega kahendkujul.

vaatleme kõikvõimalikke koode mis saavad olla *baidis* :

$\mathbf{00000000}_2$

$\mathbf{00000001}_2$

$\mathbf{00000010}_2$

$\mathbf{00000011}_2$

.

.

$\mathbf{01111001}_2$

$\mathbf{01111010}_2$

$\mathbf{01111011}_2$

$\mathbf{01111100}_2$

.

.

$\mathbf{11111100}_2$

$11111101_2$

$11111110_2$

$11111111_2$

jaotame baidi kõrgemaks ja madalamaks **poolbaidiks** :

$00000000_2$

$00000001_2$

$00000010_2$

$00000011_2$

.

.

$01111001_2$

$01111010_2$

$01111011_2$

.

.

$11111101_2$

$11111110_2$

$11111111_2$

Mõlema *poolbaidi* saab asendada vastava **16ndnumbriga** **0 . . . . F** :

$00000000_2 = 00_{16}$

$00000001_2 = 01_{16}$

$00000010_2 = 02_{16}$

$00000011_2 = 03_{16}$

.

.

$01111001_2 = 79_{16}$

$01111010_2 = 7A_{16}$

$01111011_2 = 7B_{16}$

.

.

$253_{10} = 11111101_2 = FD_{16} = 253_{10}$

$11111110_2 = FE_{16}$

$11111111_2 = FF_{16}$

*Baidi* mistahes võimalikku sisu / koodi saab seega esitada **kahejärgulise 16ndarvuna**: ( *suvalised juhuslikud näitebaidid* )

$10110111_2 = B7_{16}$ $\quad$ $00000101_2 = 05_{16}$

$11100100_2 = E4_{16}$ $\quad$ $01101010_2 = 6A_{16}$

$11111110_2 = FE_{16}$ $\quad$ $11111111_2 = FF_{16}$

Kui arvutimälu sisu tuleb kuidagi visuaalselt näidata, siis eelistatakse mälus tegelikult asuvate **1**-de ja **0**-de näitamise asemel esitada mälubaitides asuvate **2nd**arvudega võrdseid **16nd**arve.

**16nd**süsteemi kasutatakse **2nd**arvude kompaktsemaks esitamiseks

## ARVUSÜSTEEMID

kokkuvõttev loetelu

$2 \le p \le 16$

**2ndsüsteem:** $p = 2$ $\quad a_i \in \{ 0 \ 1 \}$

3ndsüsteem: $p = 3$ $\quad a_i \in \{ 0 \ 1 \ 2 \}$

4ndsüsteem: p = 4 $a_i \in \{0\ 1\ 2\ 3\}$

5ndsüsteem: p = 5 $a_i \in \{0\ 1\ 2\ 3\ 4\}$

6ndsüsteem: p = 6 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\}$

7ndsüsteem: p = 7 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\}$

**8ndsüsteem: p = 8** $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\}$

9ndsüsteem: p = 9 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\}$

**10ndsüsteem: p = 10** $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\}$

11ndsüsteem: p = 11 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\}$

12ndsüsteem: p = 12 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\}$

13ndsüsteem: p = 13 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C\}$

14ndsüsteem: p = 14 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C\ D\}$

15ndsüsteem: p = 15 $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C\ D\ E\}$

**16ndsüsteem: p = 16** $a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C\ D\ E\ F\}$

**olulised arvusüsteemid:** $p = 2 = 2^1$ ( kus $p = 2^n$ )

$p = 8 = 2^3$

$p = 10$

$p = 16 = 2^4$

*ülesanne:*

Esitada 8ndarv $7433_8$ **2nd**süsteemis ja **16nd**süsteemis:

$7433_8 = ?_2 = ?_{16}$ Leida selle arvu väärtus.

*ülesanne:*

Esitada 2ndarv $1101101101_2$ **4nd** , **8nd** ja **16nd**süsteemis:

$1101101101_2 = ?_4 = ?_8 = ?_{16}$ Leida selle arvu väärtus.

*ülesanne:*

Teisendada **3nd**arv $12101_3$ **5nd**süsteemi: $12101_3 = ?_5$

kuna need alused $p = 3$ ja $p = 5$ ei ole *2-täisarvastmed* : $p \neq 2^n$

siis selline teisendus sobib teha üle **10nd**süsteemi: 3nd → 10nd → 5nd

$1\ 2\ 1\ 0\ 1_3 = ?_{10}$

$1\ 2\ 1\ 0\ 1_3 = 145_{10}$

leitud **10nd**arvu (ehk **väärtuse**) teisendame **5nd**süsteemi 5-ga jagades :

| | : 5 | |
|---|---|---|
| 145 | | |

| | : 5 | |
|---|---|---|
| 145 | 0 | ↑ |
| 29 | 4 | |
| 5 | 0 | |
| 0 | 1 | |

ülesanne lahendatud: $12101_3 = 1040_5$

*kontrollimisvõimalus* : $1\ 0\ 4\ 0_5 = 145_{10}$

*ülesanne:*

Koostada **6ndsüsteemi korrutustabel** ja teha selle abil **6ndsüsteemis** korrutamistehe $44_{10}$ * $113_{10}$

Kontrollida leitud **6ndkujulist tulemust** väärtuste võrdlemise kaudu.

**6nd**numbrid on { **0 1 2 3 4 5** }

6ndsüsteemi korrutustabel on maksimaalselt **6** x **6** suurusega tabel, mis tasub vähendada 5 x 5 tabeliks (sest **0**-ga pole mõtet tabelis korrutada) :

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **1** | | | | | |
| **2** | | | | | |
| **3** | | | | | |
| **4** | | | | | |
| **5** | | | | | |

tabeli igasse lahtrisse märgime kahe **6nd**numbri korrutise — **6nd**arvuna

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **1** | $1_6$ | $2_6$ | $3_6$ | $4_6$ | $5_6$ |
| **2** | $2_6$ | $4_6$ | $10_6$ | $12_6$ | $14_6$ |
| **3** | $3_6$ | $10_6$ | $13_6$ | $20_6$ | $23_6$ |
| **4** | $4_6$ | $12_6$ | $20_6$ | $14_6$ | $32_6$ |
| **5** | $5_6$ | $14_6$ | $23_6$ | $32_6$ | $41_6$ |

$44_{10}$ * $113_{10}$ 6ndsüsteemis korrutamiseks teisendame operandid 6ndkujule :

**: 6**

**44**

**: 6**

| | |
|---|---|
| **44** | **2** ↑ |
| **7** | **1** |
| **1** | **1** |
| **0** | |

$44_{10} = 112_6$

**: 6**

**113**

**: 6**

| | |
|---|---|
| **113** | **5** ↑ |
| **18** | **0** |
| **3** | **3** |
| **0** | |

$113_{10} = 305_6$

**6nd**kujul korrutamine :

$1\,1\,2_6$ * $3\,0\,5_6$

$$
\begin{array}{rr}
1\,1\,2_6 & * \quad 3\,0\,5_6 \\
\hline
 & 1\,4 \\
 & 1\,1\;\;0 \\
+ & 1\,0 \\
 & 3\,0\,5 \\
 & 3\,0\,5 \\
\hline
 & 3\,5\,0\,0\,4_6
\end{array}
$$

*kontroll:* **6nd***süsteemi järgukaalude abil arvutame väärtuse ehk* **10nd***kuju* :

$$3 * 1296 + 5 * 216 + 4 = 4972_{10}$$

$$44_{10} * 113_{10} = 4972_{10}$$

*ülesanne:*

Korrutada **8ndkujul** 8ndarvud $24_8 * 37_8$

( 8ndsüsteemi korrutustabelit ei pea eelnevalt koostama )

Kontrollida tulemust **väärtuste** kaudu:

- leides mõlema teguri **väärtuse**;
- korrutades need väärtused omavahel (kalkulaatoriga);
- leides saadud **8ndkujulise korrutise** väärtuse;
- võrdleme kas väärtused (ehk 10ndkujud) võrduvad;

. . . . kuigi 6ndsüsteemi jaoks koostasime eespool ka "korrutustabeli" — ei ole selline korrutustabel mitte ilmtingimata vajalik:

vajalike 8ndnumbrite 0 . . . 7 omavahelised korrutised saame leida ka tegeliku korrutamise käigus (*osakorrutiste* leidmise käigus)

## KAHENDARITMEETIKA

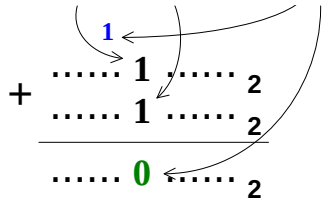
### LIITMINE 2ndsüsteemis

lihtsaim liitmistehe:

$1_{10} + 1_{10} = 2_{10}$ (**10nd**süsteemis)

$1_2 + 1_2 = 10_2$ (sama liitmine **2nd**süsteemis)

$1_2 + 1_2 = 10_2$ **2nd**liitmise järgud:



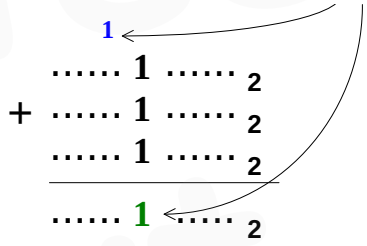

**jooksva summajärgu** ja **ülekande** tekkimine

veidi "keerulisem" liitmistehe:

$1_{10} + 1_{10} + 1_{10} = 3_{10}$ (**10nd**süsteemis)

$1_2 + 1_2 + 1_2 = 11_2$ (sama liitmine **2nd**süsteemis)

$1_2 + 1_2 + 1_2 = 11_2$ ( summajärk ja ülekanne )



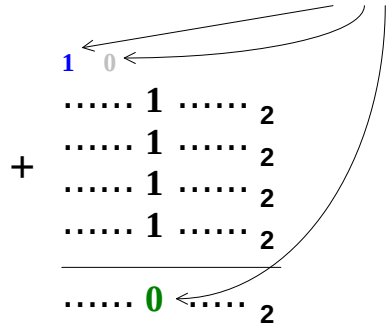

**jooksva summajärgu** ja **ülekande** tekkimine

veel "keerulisem" liitmistehe:

$1_{10} + 1_{10} + 1_{10} + 1_{10} = 4_{10}$ (**10nd**süsteemis)

$1_2 + 1_2 + 1_2 + 1_2 = 100_2$ (sama liitmine **2nd**süsteemis)

$1_2 + 1_2 + 1_2 + 1_2 = 100_2$ ( summajärk ja ülekanded )




**jooksva summajärgu**
ja
**ülekande**
tekkimine

## LAHUTAMINE 2ndsüsteemis

$1_2 - 1_2 = 0_2$

$1_2 - 0_2 = 1_2$

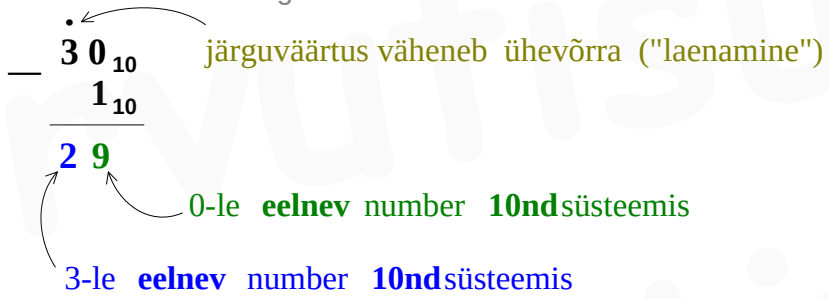
meenutame, et **10nd**süsteemi numbrid (ehk "**10nd**numbrid") on :

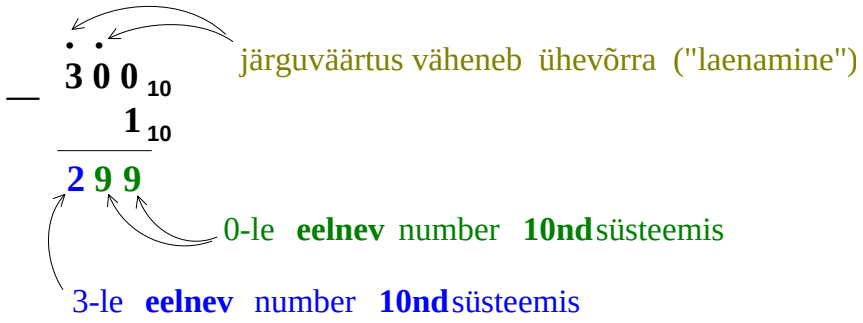
. . . . . 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 . . .

eelneval real on neid korduvana nii palju selleks, et rõhutada :

**9**-le järgneb **0** ja **0**-le eelneb **9** ( **10nd**süsteemis )

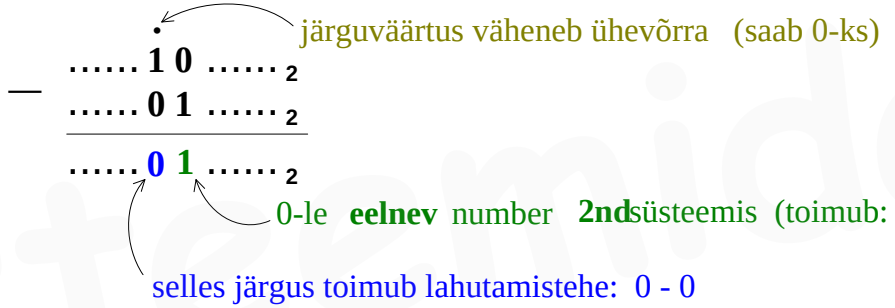
võrdlus 10ndsüsteemi lahutamisega:

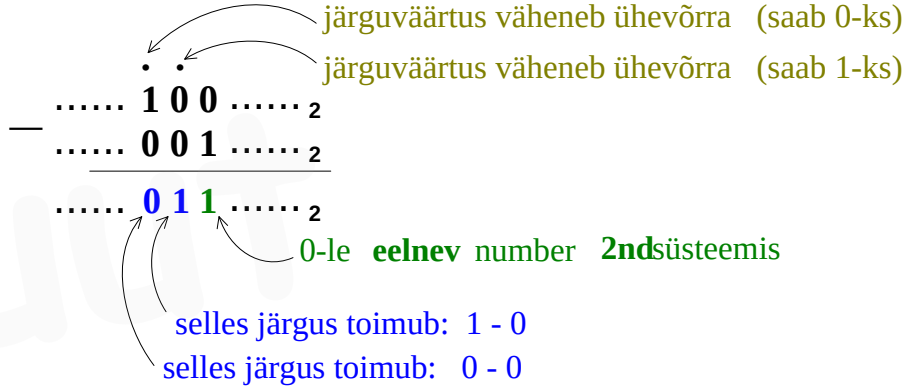



sama põhimõte rakendatuna **2nd**süsteemis

( kus **2nd**numbrid on . . . . . **0 1 0 1 0 1 0 1 0** . . . . .
ehk **0**-le **järgneb 1** ja **0**-le **eelneb 1** ) :



## ÜMARDAMINE erinevates arvusüsteemides

**( *NB! alati ümardatakse MURDOSA, mitte täisosa* )**

**ümardamine kui p on paarisarv ?**

*numbrite väiksem pool* *suurem pool*

**10nd**süsteem: **p = 10** $a_i \in \{ 0\ 1\ 2\ 3\ 4 \quad 5\ 6\ 7\ 8\ 9 \}$

*täisosa* . *murdosa* . . . 234999999999999999. . . $_{10} \approx$ . . . $23_{10}$

*täisosa* . *murdosa* . . . 235000000000000000. . . $_{10} \approx$ . . . $24_{10}$

**8nd**süsteem: **p = 8** $a_i \in \{ 0\ 1\ 2\ 3 \quad 4\ 5\ 6\ 7 \}$

*täisosa* . *murdosa* . . . 12377777777777777. . . $_8 \approx$ . . . $12_8$

*täisosa* . *murdosa* . . . 12400000000000000. . . $_8 \approx$ . . . $13_8$

**6nd**süsteem: **p = 6** $a_i \in \{ 0\ 1\ 2 \quad 3\ 4\ 5 \}$

*täisosa* . *murdosa* . . . 1425555555555555. . . $_6 \approx$ . . . $14_6$

*täisosa* . *murdosa* . . . 1430000000000000000. . . $_6 \approx$ . . . $15_6$

**16nd**süsteem: **p = 16**

$a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7 \quad 8\ 9\ A\ B\ C\ D\ E\ F\}$

*täisosa* . *murdosa* . . . 567FFFFFFFFFFF. . . $_{16} \approx$ . . . $56_{16}$

*täisosa* . *murdosa* . . . 568000000000000000. . . $_{16} \approx$ . . . $57_{16}$

kui **p** = *paarisarv* , siis ümardamiseks on alati piisav arvestada ainuüksi **ESIMEST mittemahtuvat** / ärajäävat järku

ehk ümardamine toimub alati **üheainsa** (ehk "esimese ärajääva") järgu alusel

## ÜMARDAMINE erinevates arvusüsteemides

***kui* p on paaritu arv ?**

*numbrite väiksem "pool"* *suurem "pool"*

**9nd**süsteem: **p = 9** $a_i \in \{ 0\ 1\ 2\ 3 \quad 4 \quad 5\ 6\ 7\ 8 \}$

*täisosa* . *murdosa* . . . 564444444444444443. . . $_9 \approx$ . . . $56_9$

*täisosa* . *murdosa* . . . 564444444444444445. . . $_9 \approx$ . . . $57_9$

**7nd**süsteem: **p = 7** $a_i \in \{ 0\ 1\ 2 \quad 3 \quad 4\ 5\ 6 \}$

*täisosa* . *murdosa* . . . 453333333333333332. . . $_7 \approx$ . . . $45_7$

*täisosa* . *murdosa* . . . 453333333333333334. . . $_7 \approx$ . . . $46_7$

**5nd**süsteem: **p = 5** $a_i \in \{ 0\ 1 \quad 2 \quad 3\ 4 \}$

**15nd**süsteem: **p = 15**

$a_i \in \{0\ 1\ 2\ 3\ 4\ 5\ 6 \quad 7 \quad 8\ 9\ A\ B\ C\ D\ E \}$

**11nd**süsteem: **p = 11**

$a_i \in \{0\ 1\ 2\ 3\ 4 \quad 5 \quad 6\ 7\ 8\ 9\ A \}$

kui **p** = *paaritu arv* , siis ümardamisel võib vajalik olla **MITME mittemahtuva** / ärajääva järgu arvestamine.
(ehk seljuhul *ümardamine* võib vahel toimuda **mitme** järgu järgi )

## ÜMARDAMISREEGEL 2ndSÜSTEEMI JAOKS

arvestades et **p = 2** on paarisarv :

*numbrite väiksem pool* *suurem pool*

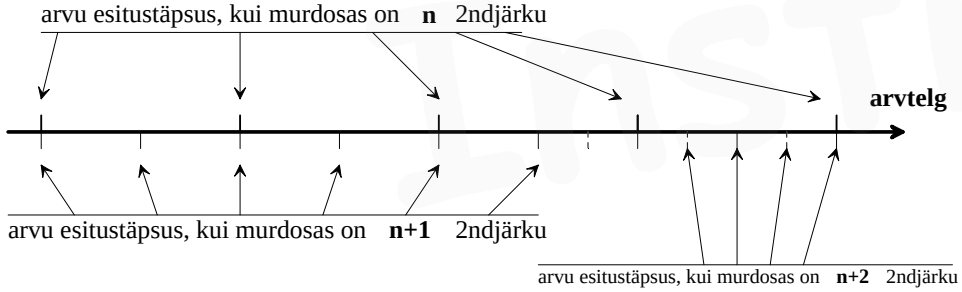
**2nd**süsteem: **p = 2** $a_i \in \{ \; 0 \qquad 1 \; \}$

*täisosa . murdosa . . .* **000** *. . .* $_2 \approx$ *täisosa . murdosa . . .* **00** $_2$

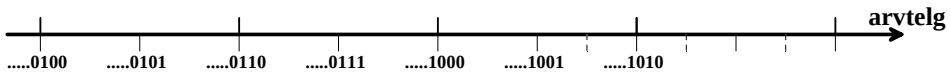
*täisosa . murdosa . . .* **001** *. . .* $_2 \approx$ *täisosa . murdosa . . .* **01** $_2$

**2nd**arvu ümardamise *reegliks* sõnastub:
esimene formaadist väljajääv järguväärtus ( 0 või 1 ) **liidetakse juurde** allesjääva arvuformaadi madalaimasse järku, arvestades ka sellel liitmisel tekkivat ülekannet.

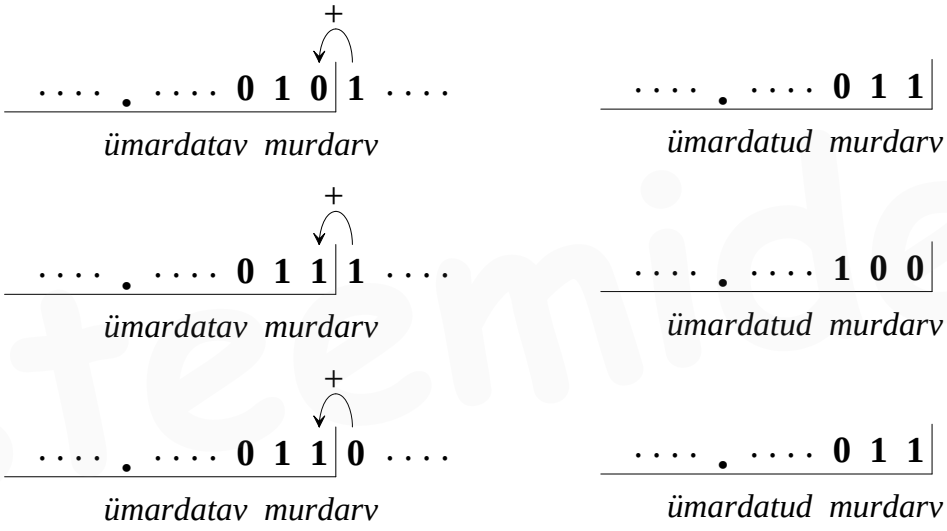
### Kahendarvude murdosa ümardamise olemus




esitustäpsus: **k** järku murdosas:




esitustäpsus: **k-1** järku murdosas: .... kui madalaim järk kaob ümardamisel


ehk veelkord:
esimene formaadist väljajääv järguväärtus ( 0 või 1 ) **liidetakse juurde** allesjääva arvuformaadi madalaimasse järku, arvestades ka sellel liitmisel tekkivat ülekannet.

*ülesanne:*

Teisendada kümnendarv **45.67**$_{10}$ **12ndsüsteemi**
(alusele **p** = 12 ) täpsusega **3** kohta murdosas.

*ülesanne:*

Teisendada kümnendarv **45.67**$_{10}$ **7nd**süsteemi
(alusele **p** = 7 ) täpsusega **4** kohta murdosas.

*ülesanne:*

Teisendada **2ndkujule** arvud

**73.4**
**16.6**
**5.5**
**6.25**

. . . . täpsusega **6 järku** 2ndarvude (ümardatud) murdosas**.**

Arvude teisendus 2ndsüsteemi üle 8ndsüsteemi: **10**nd → **8**nd → **2**nd

*edasi :*

arvuta **2nd**kujul samadest leitud operandidest koosnev järgnev avaldis :

*ülesanne:*

Arvutada **2ndkujul** [ **( 73.4 - 16.6 ) : 5.5** ] × **6.25** = . . . .
täpsusega **6** järku 2ndarvude murdosas (eelmise ülesande oper.)

kontrollides kalkulaatoriga :

**( 73.4 - 16.6 ) : 5.5** ] × **6.25** = **64.54545454 . . .**

*ülesanne:*

Jagada **2ndkujul** **1110.101**$_2$ **:** **11.01**$_2$ **=** . . . . . $_2$

( jagub täpselt !)
kontrollida tulemust 10ndkujul (väärtuste võrdlemise teel)

**TÄIENDKOOD** **PÖÖRDKOOD** **NEGATIIVSETE ARVUDE ESITAMINE**

- 0-ga algavat 2ndkoodi ( **0**..... ) nimetame **otsekood**iks.
  Otsekood esitab alati *positiivset* väärtust, milleks on tema enda kui 2ndkoodi väärtus. (*"otsekood esitab iseennast")*

**NB!** seni oleme tegelenud ainult **otsekoodidega** ehk positiivsete 2ndarvudega

seni esitasime positiivseid 2ndarve ka nii, et nad tohtisid alata numbriga **1** :

+ **17**$_{10}$ = **10001**$_2$

nüüdsest edasi on rangelt tähtis, et **positiivne** arv peab algama **0**ga ! :

+ **17**$_{10}$ = **010001**$_2$
+ **17**$_{10}$ = **00010001**$_2$ ( kui täisarv on esitatud 8-järgulise 2ndkoodina)
+ **17**$_{10}$ = **0000000000010001**$_2$ (esitatud 16-järgulise 2ndkoodina)

- 1-ga algav 2ndkood ( **1**..... ) on **täiendkood** või **pöördkood**.
  ( kumb nendest ta tegelikult on, peab olema ette üteldud: koodile peale vaadates me ei tunne ära, kas ta on täiendkood või pöördkood )
  **täiend**kood ja **pöörd**kood esitavad *negatiivset* väärtust.
  täiendkood ja pöördkood on olemas ainult **2nd**süsteemis

Kõrgeimat järku nimetatakse *märgijärguks*, kuid tegelikult esitab ta samaaegselt nii <u>väärtust</u> kui ka <u>märki</u> — mitte ainult märki**!**

+ **17**$_{10}$ = **010001**$_2$

*tüüpiline esmane eksimus :*

Kuigi kõrgeimat järku nimetatakse **märgijärguks** ja märgijärk **1** on **negatiivse** arvu tunnuseks, siis negatiivset 2ndarvu ei saada positiivsest 2ndarvust tema märgijärgu 0 lihtsa asendamisega 1-ks:

kuigi:

$+ 17_{10} = 010001_{2}$

siis:

$— 17_{10} \neq 110001_{2}$

---

Negatiivset arvu esitatakse *pöördkoodina* või *täiendkoodina*.
Nendest tuleb kumbki esitusviis valida — ja kui valik on tehtud siis teist koodi samas arvutiarhitektuuris enam kasutada ei saa / ei tohi.

♦ otsekoodist saame **pöördkoodi**, kui inverteerime kõik järgud vastupidiseks

$+ 17_{10} = 010001_{2}$

Kui kasutada **-17** esitamiseks **pöördkoodi**, siis

$— 17_{10} = 101110_{pk}$

**täiendkoodi** saamise 2 võimalust :

♦ otsekoodist saame **täiendkoodi**, kui liidame ta *pöördkoodile* **+ 1**

♦ otsekoodist saame **täiendkoodi**, kui kirjutame otsekoodi madalamad järgud ümber kuni esimese **1**-ni (kaasaarvatud) ja ülejäänud kõrgemad järgud inverteerime

$+ 17_{10} = 010001_{2}$

Kui kasutada **-17** esitamiseks **täiendkoodi**, siis

$— 17_{10} = 101111_{tk}$

♦ täiendkoodi täiendkood on **otsekood**

♦ pöördkoodi pöördkood on **otsekood**

♦ Pöörates mingi 2ndkoodi täiendkoodi (või pöördkoodi) saame tema vastandarvu esitava 2ndkoodi.

meenutame:

otsekoodi ette tohib kirjutada **0**-le ilma otsekoodi väärtust sellega muutmata järelikult :

♦ täiendkoodi ja pöördkoodi ette tohib kirjutada **1**-sid
(ilma et koodi poolt esitatav arvväärtus seeläbi muutuks)

$— 17_{10} = 101110_{pk} = 11101110_{pk} = 1111111111101110_{pk}$

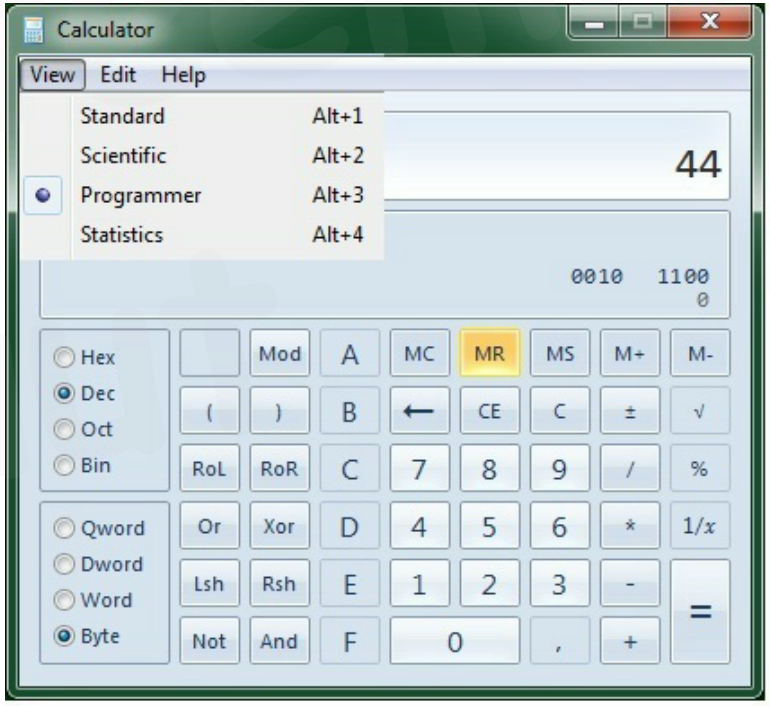
$— 17_{10} = 101111_{tk}$

$— 17_{10} = 11101111_{tk}$ ( neg. täisarv esitatud 8-järgulisena)

$— 17_{10} = 1111111111101111_{tk}$ ( neg. täisarv esitatud 16-järgulisena)

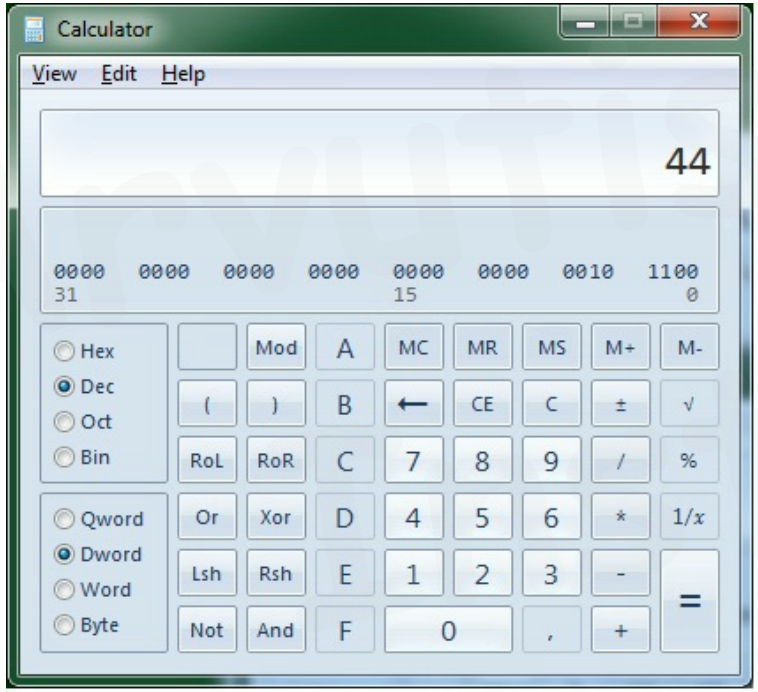
Kuigi negatiivsete arvude esitamiseks arvutis sobib valida nii **täiendkood** kui ka **pöördkood**, siis kõikides arvutiarhitektuurides on valitud selleks **täiendkood** kuna täiendkoodi aritmeetikareeglid on veidi lihtsamad.

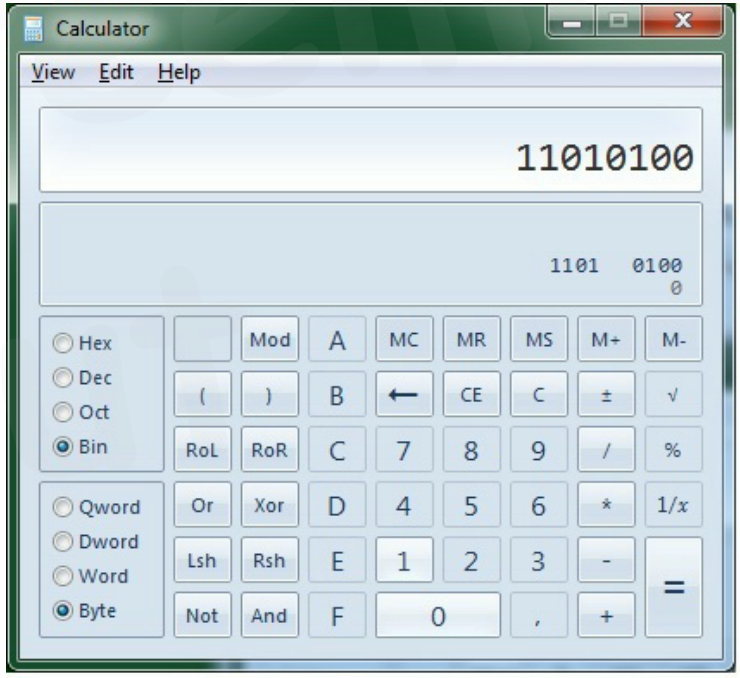
**Windows Calculator** abil saab vaadata arvutimälus salvestatud negatiivse täisarvu kahendkuju, kus ilmneb et arvuti hoiab negatiivseid arve just **täiendkoodis** :

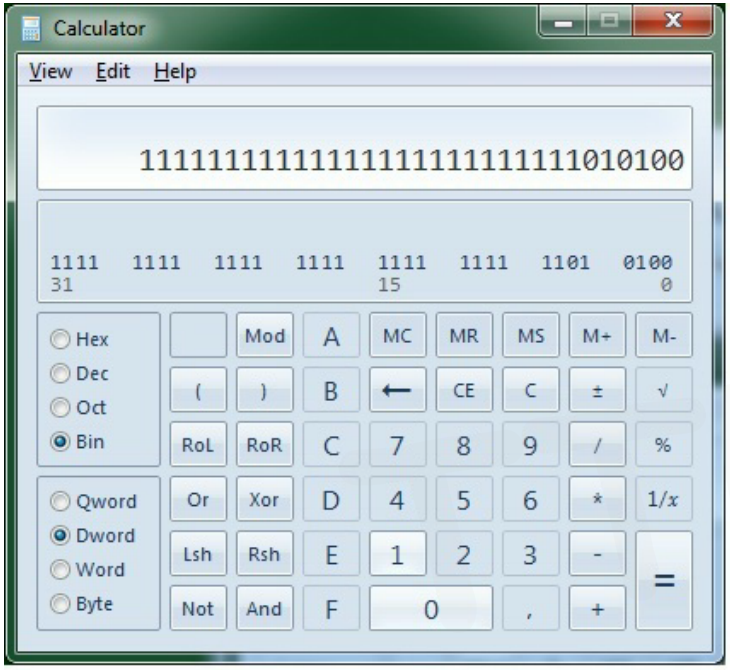



Calculator
View Edit Help
44
0000 0000 0010 1100
15 0
Hex Dec Oct Bin
Qword Dword Word Byte
Mod A MC MR MS M+ M-
( ) B ← CE C ± √
RoL RoR C 7 8 9 / %
Or Xor D 4 5 6 * 1/x
Lsh Rsh E 1 2 3 -
Not And F 0 , + =


Calculator
View Edit Help
-44
1101 0100
0
Hex Dec Oct Bin
Qword Dword Word Byte
Mod A MC MR MS M+ M-
( ) B ← CE C ± √
RoL RoR C 7 8 9 / %
Or Xor D 4 5 6 * 1/x
Lsh Rsh E 1 2 3 -
Not And F 0 , + =


Calculator
View Edit Help
44
0000 0000 0000 0000 0000 0000 0010 1100
31 15 0
Hex Dec Oct Bin
Qword Dword Word Byte
Mod A MC MR MS M+ M-
( ) B ← CE C ± √
RoL RoR C 7 8 9 / %
Or Xor D 4 5 6 * 1/x
Lsh Rsh E 1 2 3 -
Not And F 0 , + =


Calculator
View Edit Help
11010100
1101 0100
0
Hex Dec Oct Bin
Qword Dword Word Byte
Mod A MC MR MS M+ M-
( ) B ← CE C ± √
RoL RoR C 7 8 9 / %
Or Xor D 4 5 6 * 1/x
Lsh Rsh E 1 2 3 -
Not And F 0 , + =




**modifitseeritud** täiendkood / pöördkood

kaitseb *mitteavastatava* ületäitumise vastu

- **modifitseeritud** täiendkoodi või **modifitseeritud** pöördkoodi kahes kõrgemas järgus peab olema sama järguväärtus:

  **11**..... : *negatiivne arv*

  **00**..... : *positiivne arv* (ehk esitub otsekoodina )

modifitseeritud koodi kasutamine välistab tehte tulemuse mitteavastatava ületäitumise:

**10**..... : *"ületäitunud" negatiivne arv* (modif. koodi kasutamisel)

**01**..... : *"ületäitunud" positiivne arv* (modif. koodi kasutamisel)

Sellist "ületäitunud" resultaati ei tohi enam kasutada järgmise tehte operandiks, kuid *ületäitunud modifitseeritud* koodi väärtus (ehk tehte tulemus) on välja loetav õigesti — ehk tegelikult ületäitumist pole.

## ARITMEETIKA TÄIENDKOODiga ja PÖÖRDKOODiga

*ülesanne:*

## PUHTMURDARVUD

Teha **2nd**kujul tehted: murdosa esitustäpsus: **7** kahendkohta

— $0.13_{10}$ + $0.52_{10}$ = ( täiendkoodis )

— $0.52_{10}$ + $0.13_{10}$ = ( modif. täiendkoodis )

— $0.52_{10}$ — $0.13_{10}$ = ( pöördkoodis )

— $0.52_{10}$ + $0.13_{10}$ = ( modif. pöördkoodis )

$\mathbf{0.13}_{10} \approx 0.102_8 \approx 0.0010001_2$ $\mathbf{0.52}_{10} \approx 0.412_8 \approx 0.1000011_2$

$-\mathbf{0.13}_{10} = -0.0010001_2 = 1.1101111_2$ ( tk )

$-\mathbf{0.13}_{10} = 1.1101110_2$ ( pk )

$-\mathbf{0.52}_{10} = -0.1000011_2 = 1.0111101_2$ ( tk )

$-\mathbf{0.52}_{10} = 11.0111101_2$ ( mtk )

$-\mathbf{0.52}_{10} = 1.0111100_2$ ( pk )

$-\mathbf{0.52}_{10} = 11.0111100_2$ ( mpk )

$-\mathbf{0.13}_{10} + (+\mathbf{0.52}_{10}) = 0.0110010_2 = 0.31_8 = 0.39_{10}$

$-\mathbf{0.52}_{10} + (+\mathbf{0.13}_{10}) = 11.1001110_{mtk} = -00.0110010_2 = -0.31_8$

$-\mathbf{0.13}_{10} + (-\mathbf{0.52}_{10}) = 1.0101011_{pk} = -0.52_8 = -0.65_{10}$

$-\mathbf{0.52}_{10} + (+\mathbf{0.13}_{10}) = 11.1001101_{mpk} = -00.0110010_2 = -0.31_8$

*ülesanne:*

TÄISARVUD

Teha **2nd**kujul tehted:

$71_{10}$ — $40_{10}$ = . . . ( täiendkoodis )

— $71_{10}$ + $40_{10}$ = . . . ( modif. täiendkoodis )

— $71_{10}$ — $40_{10}$ = . . . ( modif. pöördkoodis )

**71**$_{10}$ = $107_8$ = $001000111_2$

**40**$_{10}$ = $0101000_2$

— **71**$_{10}$ = $110111001_{mtk}$

— **40**$_{10}$ = $1011000_{tk}$

— **40**$_{10}$ = $11011000_{mtk}$

— **40**$_{10}$ = $11010111_{mpk}$

— **71**$_{10}$ = $110111000_{mpk}$

+ **71**$_{10}$ + ( — **40**$_{10}$ ) = $000011111_2$ = + $37_8$ = + $31_{10}$

— **71**$_{10}$ + **40**$_{10}$ = $111100001_{mtk}$ = — $000011111_2$ = — $31_{10}$

— **71**$_{10}$ + (— **40**$_{10}$) = $110010000_{mpk}$ = — $001101111_2$ = — $111_{10}$

*ülesanne:*

MURDARVUD

Teha **2nd**kujul tehted, esitades negatiivsed arvud **täiendkoodis** :

$17.625_{10}$ + ( — $25.75_{10}$ ) =

$36.25_{10}$ + ( — $25.75_{10}$ ) =

ümardamist pole siin vaja kuna need operandid esituvad 2ndkujul **täpselt**

**17.625**$_{10}$ = $010001.101_2$ **36.25**$_{10}$ = $0100100.01_2$

— **25.75**$_{10}$ = — $011001.11_2$ = $100110.01_2$

$17.625_{10}$ — $25.75_{10}$ = $110111.111_2$ = — $001000.001_2$ = — $8.125_{10}$

$36.25_{10}$ — $25.75_{10}$ = $001010.100_2$ = $10.5_{10}$

*ülesanne:*

Teha **2nd**kujul tehted, esitades negatiivsed arvud **täiendkoodis** :

$13.57_{10}$ + ( — $28.26_{10}$ ) =

$28.26_{10}$ + ( — $13.57_{10}$ ) =

. . . . murdosad teisendada läbi **8nd**süsteemi, täpsusega 8 järku **2nd**arvu murdosas

*iseseisvaks lahendamiseks:*

Teha **2nd**kujul liitmistehe, esitades negatiivset operandi **täiendkoodis** :

$34_{10}$ + ( — $57_{10}$ ) =

Teha **2nd**kujul liitmistehe, esitades negatiivset operandi **pöördkoodis** :

$34_{10}$ + ( — $57_{10}$ ) =

*ülesanne:*

Leida järgnevate baidipikkuste 2ndarvude **väärtused**

(teades et negatiivsete väärtuste esitamiseks kasutatakse **täiendkoodi** ) :

**00000000**$_2$ = . . . . . **10**

**00000001**$_2$ = . . . . . **10**

$00000010_2$ = . . . . . $_{10}$

$00000011_2$ = . . . . . $_{10}$

.

.

$01111110_2$ = . . . . . $_{10}$

$01111111_2$ = . . . . . $_{10}$

$10000000_2$ = . . . . . $_{10}$

$10000001_2$ = . . . . . $_{10}$

$10000010_2$ = . . . . . $_{10}$

.

.

$11111101_2$ = . . . . . $_{10}$

$11111110_2$ = . . . . . $_{10}$

$11111111_2$ = . . . . . $_{10}$

Järelda eelnevast, milline on baidipikkuse täisarvuformaadi **esitusdiapasoon** ?
ehk millise väärtusega on *minimaalseim* ja *maksimaalseim* täisarv, mis "mahub" ühte baiti ehk on esitatav 8-järgulise 2ndarvuna ?

*ülesanne:*

Leida **10**-järgulise 2ndarvu väärtuste diapasoon (positiivseim ja negatiivseim täisarv (väärtus), kui negat. arve esitatakse

❶ täiendkoodis ?

❷ modif. täiendkoodis ?

$1000000000_{tk} \leq N \leq 0111111111_{tk}$

$-512_{10} \leq N \leq +511_{10}$

$1100000000_{mtk} \leq N \leq 0011111111_{mtk}$

$-256_{10} \leq N \leq +255_{10}$

## 2ndkoodide NIHUTAMINE

2ndkoodi **nihutamisel** iga tema järguväärtus **0 / 1** nihkub ehk "astub" tema naaberjärku.
Nihutatav 2ndkood on kindla pikkusega ja asub teda hoidvas **registris**.

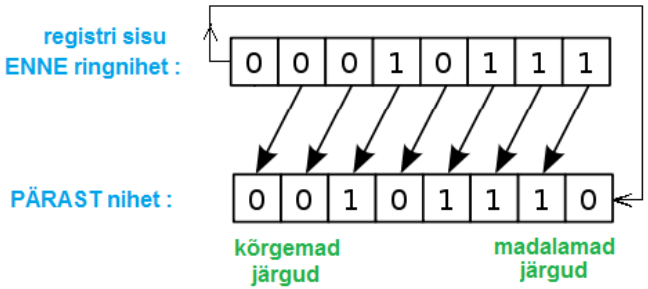
näiteks 8-järgulise 2ndkoodi (ehk 8-järgulise **registri**) korral :
[**0**][**1**][**1**][**1**][**0**][**1**][**0**][**1**]

**järgud** ehk *ühtede-nullide* asukohad (ehk kogu *register* ) on "paigal";
*ühed-nullid* ehk **järguväärtused** nihkuvad / astuvad järgust (naaber)järku.
nihe **vasakule** tähendab nihet **kõrgematesse** naaberjärkudesse;
nihe **paremale** tähendab nihet **madalamatesse** naaberjärkudesse;

on olemas 2 tüüpi nihet:
**aritmeetiline** (ehk "tavaline") nihe ; *arithmetic shift*
**ring**nihe ; *circular shift*

**Ringnihe** seletab juba oma nimega, millega on tegemist:
registrist väljanihkuv järguväärtus **0 / 1** (taas)siseneb registrisse selle "vabanevasse" järku.
*Ringnihe* on vähemtähtis.

RINGnihe vasakule




**aritmeetilise** nihke korral lähevad "formaadist" ehk registrist väljanihkuvad järgud alati kaduma.

*meenutame:* **2nd**süsteemi *järgukaalud :*

**. . . . 1024 512 256 128 64 32 16 8 4 2 1 . . .**

ilmne, et

**2nd**koodi nihutamisel (**1** järgu võrra) **vasakule** sattuvad tema koosseisus olevad kõik **1**-d 2 korda **suurema** kaaluga järkudesse, misjuhul arvu väärtus muutub 2 korda suuremaks ( toimub arvu **korrutamine** **2**ga )

ja

**2nd**koodi nihutamisel (**1** järgu võrra) **paremale** sattuvad tema koosseisus olevad kõik **1**-d 2 korda **väiksema** kaaluga järkudesse, misjuhul arvu väärtus muutub 2 korda väiksemaks ( arv **jagatakse** 2ga ehk korrutatakse **0.5**-ga )

*siit järeldub* **2nd***koodi nihutamise olemus* :

nihutamisel **n** järgu võrra **vasakule** peab 2ndkoodi poolt esitatav väärtus muutuma $2^n$ korda **suuremaks** (arv korrutub $2^n$ -ga )

(ilma et arvu **märk** seejuures muutuks)

nihutamisel **n** järgu võrra **paremale** peab 2ndkoodi poolt esitatav väärtus muutuma $2^n$ korda **väiksemaks** (arv jagub $2^n$ -ga )

(ilma et arvu **märk** seejuures muutuks)

kuna selline nihutamine toob kaasa arvu väärtuse *korrutamise*, siis nimetatakse nihet *aritmeetiliseks* ( *arithmetic shift* )

*meenutame:*

**2nd**koode on olemas 3 "liiki" : **otsekood** **täiendkood** **pöördkood**

**otse**kood : **0 . . . . . .**
**täiend**kood : **1 . . . . . .**
**pöörd**kood : **1 . . . . . .**

Seega leidub **6** võimalikku kombinatsiooni **aritmeetilist** nihet :

**otse**koodi nihe **vasakule** ;
**otse**koodi nihe **paremale** ;
**pöörd**koodi nihe **vasakule** ;
**pöörd**koodi nihe **paremale** ;
**täiend**koodi nihe **vasakule** ;
**täiend**koodi nihe **paremale** ;

igal nihutamisel jääb registri "äärmine" järk "vabaks".
Kas sellesse vabanevasse järku "siseneb" **1** või **0** **?**

Milline järguväärtus ( **0** / **1** ) siseneb nihkel registri vabanevasse järku igal konkr. juhul nendest kuuest **?**

[ *otsekood* ]
[ *pöördkood* ]
[ *täiendkood* ]

selle äratundmisel / järeldamisel vaja arvestada, et arvu **märk** ( pos / neg ) ei tohi nihutamisel muutuda: muutub ainult arvu **väärtus**

kõik **6** võimalikku nihkejuhtumit *:*

>>>  [ *otsekood* ]  >>>  (nihe **paremale**)
<<<  [ *otsekood* ]  <<<  (nihe **vasakule**)

>>>  [ *pöördkood* ]  >>>  (nihe **paremale**)
<<<  [ *pöördkood* ]  <<<  (nihe **vasakule**)

>>>  [ *täiendkood* ]  >>>  (nihe **paremale**)
<<<  [ *täiendkood* ]  <<<  (nihe **vasakule**)

**sissenihkuvad järguväärtused** ( **0** või **1** ) igal konkr. nihkejuhtumil :

**0** >>> [ *otsekood* ] >>> (nihe **paremale**)
[ *otsekood* ] <<< **0** (nihe **vasakule**)

**1** >>> [ *pöördkood* ] >>> (nihe **paremale**)
[ *pöördkood* ] <<< **1** (nihe **vasakule**)

**1** >>> [ *täiendkood* ] >>> (nihe **paremale**)
[ *täiendkood* ] <<< **0** (nihe **vasakule**)

*ülesanne:*

Arvuta aritmeetilise **nihutamise** kaasabil :

**A** / 4 + - ( 4 * **B** )

kui A = - 72 B = 13

**BCD - koodid** (**B**inary **C**oded **D**ecimal)

Arvude kodeerimisviis, kus iga 10ndnumber **0** .... **9** esitatakse 4-järgulise 2ndkoodiga ( *poolbaidiga* ehk *tetraadiga*).

( . . . . oli sobiv põhimõte arvude salvestamiseks ja aritmeetikateheteks **varajastes arvutiarhitektuurides** . . . . )

**1971**: **4**bitine esimene laiatarbe-mikroprotsessor **Intel 4004**
BCD oli sobiv arvude andmeformaat just 4bitise andmesõnaga ( *4-bit word length* ) arvutiarhitektuurides;
veelgi varasemad "suurarvutid" töötlesid samuti BCD-arve

uuemas tarkvaras ja riistvaras BCD-koode tavaliselt enam ei kasutata, kuigi kaasaegsed protsessorid endiselt omavad/toetavad BCD-käske ja BCD-arvuformaati

Olulisemad BCD-koodid:
— **"loomulike kaaludega" BCD-kood** (järgukaaludega 8421)
— **"liiase kolmega" BCD-kood** ( XS3 )
( "liiane 3": tetraadi väärtus on esitatavast numbrist 3 võrra suurem)

| decimal | BCD-koodis **8421** tetraad | BCD-koodis **8421(+3)** tetraad |
|---|---|---|
| **0** | 0000 | 0011 |
| **1** | 0001 | 0100 |
| **2** | 0010 | 0101 |
| **3** | 0011 | 0110 |
| **4** | 0100 | 0111 |
| **5** | 0101 | 1000 |
| **6** | 0110 | 1001 |
| **7** | 0111 | 1010 |
| **8** | 1000 | 1011 |
| **9** | 1001 | 1100 |

BCD-koodide aritmeetika : **PARANDUSLIIKMED liitmisel**

**parandusliige** loomulike kaaludega **BCD-koodis** (8421) :
jooksvale tetraadisummale liidetakse juurde parandusliige **+ 6 = $0110_2$**
kahel juhul:
— kui jooksev tetraadisumma on *keelatud kood* ehk **1010 . . . 1111**
— kui tetraadide liitmisel tekkis *ülekanne* kõrgemasse tetraadi

(mõlemad eelnevad tingimused ei saa samaaegselt esineda **!**)

Kui kumbki tingimus pole täidetud, siis parandusliiget ei rakendu.
Parandusliikmete liitmisel tekkivaid tetraadidevahelisi ülekandeid ei ignoreerita.

---

☐ **parandusliikmed** "liiase kolmega" **BCD-koodis** 8421(+3) :

— jooksvale tetraadisummale liidetakse juurde parandusliige **+ 3 = $0011_2$**
kui tekkis ülekanne kõrgemasse tetraadi

— jooksvale tetraadisummale liidetakse juurde parandusliige **– 3 = $1101_2$**
kui ei tekkinud ülekannet kõrgemasse tetraadi

Parandusliikmete liitmisel tekkivaid tetraadidevahelisi ülekandeid ignoreeritakse.

---

☐ **negatiivse arvu leidmine** loomulike kaaludega **BCD-koodis** (8421) :

— *1. samm*: igale tetraadile liidetakse **+ 6 = $0110_2$**
— *2. samm*: BCD arvu kahendkood tervikuna pööratakse *täiendkoodi*

---

☐ **negatiivse arvu leidmine** "liiase kolmega" **BCD-koodis** 8421(+3) :

— BCD arvu kahendkood tervikuna pööratakse *täiendkoodi*

## BCD-koodide aritmeetika ( liitmine )

*ülesanne:*

Arvutada loomulike kaaludega **BCD-koodis** (8421) :

$4492_{10} + 219_{10} =$

$4492_{10} — 219_{10} =$

| | | | | | |
|---|---|---|---|---|---|
| 4492 : | 0 | **0100** | **0100** | **1001** | **0010** |
| 0219 : | 0 | **0000** | **0010** | **0001** | **1001** |
| | 0 | 0100 | 0110 | 1010 | 1011 |
| *parandus*: | | 0000 | 0000 | 0110 | 0110 |
| $4492_{10} + 219_{10}$ : | 0 | **0100** | **0111** | **0001** | **0001** |
| | | 4 | 7 | 1 | 1 |

$4492_{10} + 219_{10} = 4711_{10}$

| | | | | | |
|---|---|---|---|---|---|
| 0219 : | 0 | **0000** | **0010** | **0001** | **1001** |
| (+ 6) : | 0 | 0110 | 1000 | 0111 | 1111 |
| — 0219 : | 1 | **1001** | **0111** | **1000** | **0001** |
| 4492 : | 0 | **0100** | **0100** | **1001** | **0010** |
| | 1 | 1101 | 1100 | 0001 | 0011 |
| *parandus*: | 0 | 0110 | 0110 | 0110 | 0000 |
| $4492_{10}$ — $219_{10}$ : | 0 | **0100** | **0010** | **0111** | **0011** |
| | | 4 | 2 | 7 | 3 |

$4492_{10}$ — $219_{10}$ = $4273_{10}$

Arvutada "liiase 3-ga" **BCD-koodis** 8421(+3) ehk XS3 :

$9336_{10}$ + $726_{10}$ =

$9336_{10}$ — $726_{10}$ =

liitmise ületäitumise vältimiseks tuleb operandid võtta 5-järgulised

| | | | | | | |
|---|---|---|---|---|---|---|
| 09336 : | 0 | **0011** | **1100** | **0110** | **0110** | **1001** |
| 00726 : | 0 | **0011** | **0011** | **1010** | **0101** | **1001** |
| | 0 | 0111 | 0000 | 0000 | 1100 | 0010 |
| *parandus*: | 0 | 1101 | 0011 | 0011 | 1101 | 0011 |
| $9336_{10}$ + $726_{10}$ : | 0 | **0100** | **0011** | **0011** | **1001** | **0101** |
| | | 1 | 0 | 0 | 6 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| 0726 : | 0 | **0011** | **1010** | **0101** | **1001** |
| — 0726 : | 1 | **1100** | **0101** | **1010** | **0111** |
| 9336 : | 0 | **1100** | **0110** | **0110** | **1001** |

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 1000 | 1100 | 0001 | 0000 |
| *parandus*: | 0 | 0011 | 1101 | 0011 | 0011 |
| $9336_{10}$ — $726_{10}$ : | 0 | **1011** | **1001** | **0100** | **0011** |
| | | 8 | 6 | 1 | 0 |

abstraktne arvusüsteem:

## KAHENDSÜSTEEM alusega -2

( —**2nd**süsteem )

$$p = -2 \qquad a_i \in \{0, 1\}$$

$$\ldots\ a_5\ \ a_4\ \ a_3\ \ a_2\ \ a_1\ \ a_0\ \ a_{-1}\ \ a_{-2}\ \ a_{-3}\ \ a_{-4}\ \ldots$$

kuna järgukaalud on *aluse astmed*, siis aluse **-2** korral osutuvad järgukaaludeks:

| $-2^5$ | $-2^4$ | $-2^3$ | $-2^2$ | $-2^1$ | $-2^0$ | $-2^{-1}$ | $-2^{-2}$ | $-2^{-3}$ | $-2^{-4}$ |
|---|---|---|---|---|---|---|---|---|---|
| **-32** | **16** | **-8** | **4** | **-2** | **1** | **-0.5** | **0.25** | **-0.125** | |

Selles arvusüsteemis ei kasutata **täiendkoodi** negat. arvude esitamiseks.

Selles arvusüsteemis ei ole olemas ei **täiendkoodi** ega **pöördkoodi**.

Alusel **-2** on kasutusel ainult üksainus liik / tüüp koodi, mida nimetame *"miinuskahendkood"* : **-2ndkood**

Negatiivseid arve esitatakse **-2nd**süsteemis (ehk **-2nd**koodis) negatiivsete järgukaalude abil.

Ühejärguline **nihe** on siin samaväärne —**2**ga korrutamisega | jagamisega.

*ülesanne:*

On **10**-järguline täisarvuformaat -2ndsüsteemis $p = -2 \quad a_i \in \{0,1\}$

$$a_9\ \ a_8\ \ a_7\ \ a_6\ \ a_5\ \ a_4\ \ a_3\ \ a_2\ \ a_1\ \ a_0$$

Leida arvude **esitusdiapasoon**: negatiivseim ja positiivseim -2ndarv

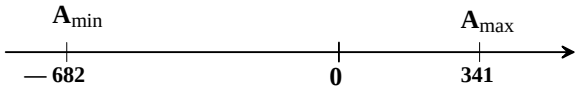
$$\mathbf{1\,0\,1\,0\,1\,0\,1\,0\,1\,0}_{-2} \le A \le \mathbf{0\,1\,0\,1\,0\,1\,0\,1\,0\,1}_{-2}$$

$$(-512) + (-128) + (-32) + (-8) + (-2) \le A \le 1 + 4 + 16 + 64 + 256$$

$$-682 \le A \le 341$$

$$\sum_{i=0}^{4} (-2)^{2i+1} \le A \le \sum_{i=0}^{4} (-2)^{2i}$$

10 järgu korral on -2ndarvu diapasoon **negatiivsele poolele** väljavenitatud, sest formaadi kõrgeima järgu $a_9$ kaal on negatiivne (-512) :




## —2ndARITMEETIKA

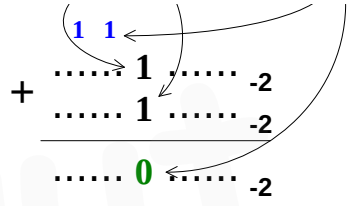
### LIITMINE -2ndsüsteemis

lihtsaim liitmistehe:

$1_{10} + 1_{10} = 2_{10}$ (**10nd**süsteemis)

$\mathbf{1}_{-2} + \mathbf{1}_{-2} = \mathbf{110}_{-2}$ (sama liitmine **-2nd**süsteemis)

$\mathbf{1}_{-2} + \mathbf{1}_{-2} = \mathbf{110}_{-2}$ **-2nd**liitmise järgud:




**jooksva summajärgu**
ja
**ülekannete**
tekkimine -2ndliitmisel

juba lihtsaimal liitmisel tekkib ülekandeid mitmesse kõrgemasse naaberjärku.

veidi "keerulisem" liitmistehe:

$1_{10} + 1_{10} + 1_{10} = 3_{10}$ (**10nd**süsteemis)

$\mathbf{1}_{-2} + \mathbf{1}_{-2} + \mathbf{1}_{-2} = \mathbf{111}_{-2}$ (sama liitmine **-2nd**süsteemis)

**-2nd**aritmeetika võimaldab grupeerida ja kasutada *"koonduvaid kolmikud "*

$1_{10}$ + $(-1_{10})$ = $0_{10}$

**$01_{-2}$ + $11_{-2}$ = $00_{-2}$**

"koonduvate kolmikute" valikuvõimalus :

$$\begin{array}{r} \cdots\cdots \mathbf{11} \cdots\cdots_{-2} \\ +\ \cdots\cdots \mathbf{01} \cdots\cdots_{-2} \\ \hline \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \end{array}$$

...on sama mis :

$$\begin{array}{r} \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \\ +\ \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \\ \hline \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \end{array}$$

"koonduvate kolmikute" sama valikuvõimalus — vastupidine paigutus :

$$\begin{array}{r} \cdots\cdots \mathbf{01} \cdots\cdots_{-2} \\ +\ \cdots\cdots \mathbf{11} \cdots\cdots_{-2} \\ \hline \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \end{array}$$

...on sama mis :

$$\begin{array}{r} \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \\ +\ \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \\ \hline \cdots\cdots \mathbf{00} \cdots\cdots_{-2} \end{array}$$

! *tüüpiline viga :*

SELLINE paiknemine ei ole "koonduv kolmik" ! :

$$\begin{array}{r} \cdots\cdots \mathbf{10} \cdots\cdots_{-2} \\ +\ \cdots\cdots \mathbf{11} \cdots\cdots_{-2} \\ \hline \cdots \mathbf{1101} \cdots\cdots_{-2} \end{array}$$

$$\begin{array}{r} \cdots\cdots \mathbf{11} \cdots\cdots_{-2} \\ +\ \cdots\cdots \mathbf{10} \cdots\cdots_{-2} \\ \hline \cdots \mathbf{1101} \cdots\cdots_{-2} \end{array}$$

. . . . sest siin on summa -3 mitte 0

*ülesanne:*

Teha -2ndsüsteemis **p = -2** $\mathbf{a_i \in \{0,1\}}$ tehted

$43_{10}$ — $12_{10}$

$43_{10}$ + $12_{10}$

— $12_{10}$ × $9_{10}$

teisendada operandid -2ndkujule:

$43_{10}$ = $1111111_{-2}$ $9_{10}$ = $11001_{-2}$

**negatiivse** arvu teisendamiseks -2ndsüsteemi on 2 võimalust:

☺ **vahetu teisendus**;

😐 *nihutamise* ja *liitmise* kaudu teisendus kasutades võrdust:

**— A = — 2A + A**

$-12_{10}$ = $0110100_{-2}$ $12_{10}$ = $11100_{-2}$

$43_{10} - 12_{10} = 1100011_{-2} = 31_{10}$

$43_{10} + 12_{10} = 1001011_{-2} = 55_{10}$

$(-12)_{10} \times 9_{10} = 10010100_{-2} = -108_{10}$

*ülesanne:*

Teha -2ndsüsteemis **p = -2** $\mathbf{a_i \in \{0,1\}}$ korrutamistehted

$21_{10}$ × $(-3_{10})$

$(-6_{10})$ × $(-3_{10})$

*pane tähele:* arvu **$21_{10}$** 2ndkuju **$\mathbf{10101_2}$** on sama nii tavalises **2nd**süsteemis kui ka **-2nd**süsteemis :

**$10101_2$** = **$10101_{-2}$**

*iseseisvaks lahendamiseks:*

Esitada -2ndsüsteemis $p = -2$ $a_i \in \{0,1\}$ arvud

$19_{10}$ $25_{10}$

$-19_{10}$ $-25_{10}$

ja teha tehted: $19_{10} + 25_{10}$

$-19_{10} - 25_{10}$

---

$19_{10} = 10111_{-2}$ $25_{10} = 1101001_{-2}$

$-19_{10} = 111101_{-2}$ $-25_{10} = 111011_{-2}$

$19_{10} + 25_{10} = 1111100_{-2} = 44_{10}$

$-19_{10} - 25_{10} = 11010100_{-2} = -44_{10}$

---

abstraktne arvusüsteem:

**KOLMENDSÜSTEEM** $p = 3$ järguväärtustega $a_i \in \{1, 0, -1\}$

*ülesanne:*

On **3ndsüsteem** $p = 3$ järguväärtustega $a_i \in \{1, 0, -1\}$

Leida 10-järgulise täisarvuformaadi

$a_9$ $a_8$ $a_7$ $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$

**esitusdiapasoon**: negatiivseim ja positiivseim **3nd**arv

$$\underline{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}$$

$$1111111111_3 \le A \le 1111111111_3$$

$$(-1 - 3 - 9 - 27 - 81 - \ldots - 3^9) \le A \le (1 + 3 + 9 + 27 + 81 + \ldots + 3^9)$$

kasutades matemaatikavalemeid RIDADE kohta

( $3^n$ astmete summa ) :

$$3^0 + 3^1 + 3^2 + 3^3 + \ldots + 3^n = \frac{3^{n+1} - 1}{2}$$

. . . . saab esitada sellise 3ndsüsteemi 10-järgulise täisarvuformaadi diapasooni ka avaldise abil :

$$-\frac{3^{10} - 1}{2} \le A \le \frac{3^{10} - 1}{2}$$

**?** kuidas saab selles 3ndsüsteemis **vastandarvu** ehk vastupidise märgiga arvu ?

selleks tuleb asendada arvu koosseisus:

kõik numbrid **1** tuleb asendada numbriga **-1**
kõik numbrid **-1** tuleb asendada numbriga **1**
(numbrid **0** jäävad muutumatuna alles)

negatiivsed arvud esituvad seega **negatiivse järguväärtuse -1** abil
*meenutame:* **-2nd**süsteemis esitusid negatiivsed arvud
**negatiivsete järgukaalude** kaudu / abil

*ülesanne:*

Teisendada arv $\mathbf{0.7}_{10}$ **3ndsüsteemi** järguväärtustega $\mathbf{a}_i \in \{ \mathbf{1}, \mathbf{0}, \mathbf{-1} \}$
täpsusega **7** kohta murdosas.

**?**

probleem: *selles arvusüsteemis ei saa mitte kõiki murdosaväärtusi esitada.*
"vahetult" saab esitada murdosi väärtusega kuni **1/2**
murdosad väärtusega üle **1/2** ei ole **vahetult** esitatavad

murdosad väärtusega üle 1/2 esitatakse "täisosa kaasabil" :
$0.7_{10} = 1 - 0.3$

seega tuleb $0.7_{10}$ esitamiseks leida kõigepealt $-0.3$ selles 3ndsüsteemis

**teisendus "üle tavalise" 3ndsüsteemi $\mathbf{a}_i \in \{ \mathbf{0}, \mathbf{1}, \mathbf{2} \}$**

jõudmiseks süsteemi **{ 1, 0, -1 }** tuleb "kaotada" numbrid **2**
numbritest "2" vabanemiseks on asendusvõimalus :

$\mathbf{0\ 2}_3 = \mathbf{1\ \bar{1}}_3$

**?** kuidas selles 3ndsüsteemis **ümardatakse** ?
ümardamine toimub siin süsteemis alati "*allapoole*"
ehk murdosa võib suvalise järgu juures lihtsalt "äralõigata"
"*ülespoole*" ümardamist pole siin arvusüsteemis olemas

teisendame $0.3_{10}$ esmalt "tavalisse" 3ndsüsteemi **{ 0, 1, 2 }**

$0.3_{10} = 0.0220022_3 = 0.1\bar{1}01\bar{1}01\bar{1}_3$

$-0.3_{10} = 0.\bar{1}01\bar{1}01\bar{1}01_3$

$\mathbf{0.7}_{10} = 1.\bar{1}01\bar{1}01\bar{1}01_3$

*ülesanne:*

Korrutada $(-12_{10}) \times 14_{10}$ **3ndsüsteemis** järguväärtustega
$\mathbf{a}_i \in \{ \mathbf{1}, \mathbf{0}, \mathbf{-1} \}$

teisendame operandid 3ndsüsteemi { 1, 0, -1 }
**täisarvu** saab teisendada siia süsteemi ka "vahetult" ehk ilma "tavalist"
3ndsüsteemi { 0, 1, 2 } läbimata.
Selleks tuleb valida teisendussammudel jäägi 2 asemel jääk **-1**

$12_{10} = 110_3$

$-12_{10} = \bar{1}\ \bar{1}0_3$

$14_{10} = 1\ \bar{1}\ \bar{1}\ \bar{1}_3$

$(-12_{10}) \times 14_{10} = \overline{1}\ 1\ 0\ \overline{1}\ 1\ 0_3$

## UJUPUNKTARVUD — UPA (ujukomaarvud)
( *Floating Point Numbers* )

kõik senivaadeldud 2ndarvud on olnud **kinnispunktarvud** (KPA)

*kinnispunktarvud* ehk "tavalised murdarvud" eelnevates ülesannetes :

$\mathbf{6.25}_{10} = 0110.01_2$

$\mathbf{5.5}_{10} = 0101.1_2$

$-\mathbf{25.75}_{10} = 100110.01_2$

. . . . . .

*kinnispunktarvud* on *ühekomponendilised* ja "esitavad iseennast".

terminit **kinnispunktarv** läheb vaja alles siis, kui tuleb kasutusse termin **ujupunktarv** / ujukomaarv: *floating point number*

"*kinnispunktarv*" vs. "*ujupunktarv*"

nimetus **kinnispunktarv** rõhutab, et see arv pole *ujupunktarv*

**UJUPUNKTARV** (UPA) on arvu kaheosaline esitus, mis koosneb kahest kinnispunktarvust: *mantissist* **m** ja *astendajast* **p**
kusjuures UPA **väärtus** arvutub nendest avaldisega:

*mantissa* $\times\ 2^{exponent}$ ehk $\mathbf{m} \times 2^{\mathbf{p}}$

vaatleme võrdluseks: ujupunktarvu olemus **10nd**süsteemis :

arvu kaheosaline esitus kujul $\mathbf{m} \times 10^{\mathbf{p}}$

$\mathbf{7250}_{10} = 7250 \times 10^{0} =$

$= 7.250 \times 10^{3} =$

$= 0.7250 \times 10^{4}$

või samaväärne esitus (tekstireziimis) arvutiekraanil: **7.250E+3**
(**10nd**eksponent)

ilmneb, et samale arvule leidub palju erinevaid ujupunkt-esituskujusid
ehk palju erinevaid paare **[ mantiss astendaja ]**

. . . . või väikeste arvude korral ( **n** << **1** )

$\mathbf{0.000082663}_{10} = 8.2663 \times 10^{-5} = 0.82663 \times 10^{-4}$

. . . mis esituks arvutiekraanil: **8.2663E-5** (**10nd**eksponent)

$\mathbf{0.00000000000000047098}_{10} =$ **4.7098E-16**

$\mathbf{47098000000000000}_{10} =$ **4.7098E+16**

näeme, et arvu esitus *ujupunktarvuna* ehk kahekomponendilisel kujul ("astendamise ja korrutamise kaudu") on seda õigustatum, mida **suurem** või **väiksem** on esitatav arv ( **n** << **1** **n** >> **1** )
hiigelsuurte ja väga väikeste arvude esitamine "iseendana" ehk ühekomponendilistena ehk *kinnispunktarvudena* on ebamugav / mahukas.

ujupunktkuju **mantissi** esitamiseks piisab selle arvu **tüvenumbritest**

tagasi **2nd**süsteemi juurde :

**mantissi** järgukaalud: 1 . 1/2 1/4 1/8 1/16 1/32 1/64 . . . . . .
**astendaja** järgukaalud: . . . . . 32 16 8 4 2 1

eelnäidatud järgukaaludest järeldub, et *ujupunktarvu* . . .
**mantiss** on PUHTMURDARV ( +/- 0 . ??????. . . . )
**astendaja** on TÄISARV

*näide:*

+ **6.25**$_{10}$ = 0110.01$_2$

**10nd**arvu **6.25**$_{10}$ jaoks:

kui *mantissiks* on **2nd**arv 0110.01 siis kaasnevaks *astendajaks* on 00000
kui *mantissiks* on **2nd**arv 011.001 siis kaasnevaks *astendajaks* on ...00001
kui *mantissiks* on **2nd**arv 01.1001 siis kaasnevaks *astendajaks* on ...00010
kui *mantissiks* on **2nd**arv 01100.1 siis kaasnevaks *astendajaks* on ..11111
kui *mantissiks* on 011001.00.... siis kaasnevaks *astendajaks* on ...111110

kõik eelnevad paarid [ *mantiss — astendaja* ] on *ujupunktarvud*,
mis sobivad esitama *kinnispunktarvu* 0110.01$_2$ ehk väärtust + **6.25**$_{10}$

*ujupunktarvu* tegelik väärtus saadakse **mantissi nihutamisel** astendaja poolt näidatud järkude võrra (ehk "*astendaja rakendamisega mantissile*")

*astendaja rakendamisega mantissile* saame **ujupunktarvust** jälle tagasi **kinnispunktarvu**

**mida näitab astendaja p MÄRK ( + / - ) ?**

astendaja **p märk** (posit / negat ) näitab, kumbas suunas on vaja mantissi **m** nihutada (UPA tegeliku väärtuse saamiseks);

astendaja **p väärtus** ( täisarv ) näitab, mitu järku on vaja mantissi **m** nihutada (UPA tegeliku väärtuse saamiseks);

teguriga × **2** $^{astendaja}$ korrutatakse mantissi
"0-llist kaugele" suureks või "0-llile lähedale" väikseks.
Selliselt korrutatud (nihutatud) mantiss esitabki kogu ujupunktarvu väärtust.

**ujupunktarvu ESITUSTÄPSUS ja DIAPASOON**

**mantissi** järkude arv ("mantissi pikkus") määrab ujupunktarvu **esitustäpsuse** (ehk tüvenumbrite arvu mida mantissi sisse mahub salvestada);

**astendaja** järkude arv määrab ujupunktarvu **esitusdiapasooni** ;

suurema astendaja korral saab mantissi korrutada "0-llist kaugemale" suureks või "0-llile lähemale" väikseks. Sellest tuleneb ka tinglik nimetus "*ujukoma*" ehk koma justnagu liigub / "ujub" mantissi järkude suhtes.

UPA väärtust väljendavast avaldisest

**m** × 2 $^{p}$

järeldub, et kui *astendaja* **p = 0**, siis mantissi väärtus ise ongi kogu UPA väärtuseks: **m** × 2$^{0}$ = **m** × 1 = **m**

**ujupunktarvu MÄRK**

**mantissi märk** on samas kogu UPA märgiks.

**!** mistahes *murdarve* hoitakse arvutis ainult **ujupunktarvudena** **!**
arvutis ei hoita murdarve "ühekomponendilistena" ehk *kinnispunktarvudena*

murdarvude hoidmist/salvestamist *kinnispunktarvudena* pole realiseeritud üheski arvutiarhitektuuris — olukorras kus murdarvude esitus UPA-na oli omal ajal niikuinii vajalik, ei ole enam mõtet selle kõrvale luua murdarvude veel ühte alternatiivset esitust — 1-komponendilist vahetut esitust "iseendana" ehk *kinnispunktarvuna*

Ujupunktarvu hoidmisel arvutis paigutatakse ta mõlemad komponendid ( mantiss **m** astendaja **p** ) kokku ühte füüsilisse registrisse.
Register jaguneb mõtteliselt kaheks loogiliseks osaks —
ühes on *mantiss* **m** (enamus reg. järkudest) ja teises on *astendaja* **p**

**[ . . . . . m a n t i s s . . . . . ][ . . astendaja . . ]**

või olenevalt konkr. arvutiarhitektuurist võib paiknemine registris olla ka vastupidi: **[ . . astendaja . . ][ . . . . . m a n t i s s . . . . . ]**
(üleskirjutatuna ujupunktarve esitades näitame *mantissi* ja *astendaja* eraldi)

kaasaegsetes protsessorites hoitakse ujupunktarve 10-nes baidis

(mõlemale UPA komponendile kokku **80** kahendjärku).

**UPA** olemasolu põhjus ja **eelis KPA ees** :
väheste arvujärkude abil saab esitada **väga suuri** arve ja
**väga väikseid** 0-lilähedasi arve — kuid seejuures kaotame **esitustäpsuses** !

**80**-järguline (10-baidine) standardne UPAformaat võimaldab esitada/salvestada väärtust, mille 10ndkujus on **18 tüvenumbrit**

## NORMALISEERITUD MANTISS

UPA **mantiss** salvestatakse alati **normaliseeritud** kujul.
Kui tehte tulemusel tekkib *mittenormaliseeritud* mantiss, siis ta kohe *normaliseeritakse*.
normaliseeritud mantissi **tunnus :**
**normaliseeritud mantissi** esimene murdosa järguväärtus peab erinema täisosa järguväärtustest

**positiivne** *normaliseeritud* **mantiss** saab olla vahemikus :

$$00.10000000...._2 \le \mathbf{m} \le 00.11111111....._2$$

järelikult asub **positiivse normaliseeritud** mantissi väärtus vahemikus:

$$0.5_{10} \le \mathbf{m} < 1_{10}$$

**negatiivne** *normaliseeritud* **mantiss** saab olla vahemikus :

$$11.00000000\ ....\ _{tk} \le \mathbf{m} \le 11.01111111....\ _{tk}$$

järelikult asub **negatiivse normaliseeritud** mantissi väärtus vahemikus:

$$-1_{10} \le \mathbf{m} < -0.5_{10}$$

## mantissi NORMALISEERIMINE

mantissi on alati võimalik **normaliseerida** , milleks tuleb

**1**. mantissi *nihutada* vajalik arv järke paremale või vasakule, nii et *normaliseeritud mantissi tunnus* saaks täidetuks;
**2**. korrigeerida astendaja väärtust suuremaks või väiksemaks niimitme võrra, mitu järku mantissi nihutati *normaliseerimisel* ;

seejuures :
*normaliseerimine* ei muuda ("ei riku") UPA väärtust ehk astendaja korrigeerimine kompenseerib mantissi nihutamist.

*ülesanne:*

On antud UPA formaat alusel 2 ehk UPA väärtus arvutub :

$$\mathbf{A} = \mathbf{m} \times 2^{\mathbf{p}} = mantiss \times 2^{astendaja}$$

mõlemad **mtk**-s. *mantiss*: 10 kahendjärku *astendaja*: 7 kahendjärku.
Esitada absoluutväärtuselt **suurima** ja **nullilähedaseima posit**. ja **negat**. UPA *mantiss* ja *astendaja*.

arvestades *mantissi* **normaliseerituse** nõuet
on *maksimaalsed* ja *minimaalsed* etteantud formaadis ujupunktarvud:

$$\text{pos } \mathbf{A}_{max} \approx 00.11111111 \times 2^{\mathbf{0011111}} \approx \mathbf{1} \times 2^{\mathbf{31}}$$
$$\text{pos } \mathbf{A}_{min} = 00.10000000 \times 2^{\mathbf{1100000}} = \mathbf{0.5} \times 2^{\mathbf{-32}}$$
$$\text{neg } \mathbf{A}_{min} \approx 11.01111111 \times 2^{\mathbf{1100000}} \approx \mathbf{-0.5} \times 2^{\mathbf{-32}}$$
$$\text{neg } \mathbf{A}_{max} = 11.00000000 \times 2^{\mathbf{0011111}} = \mathbf{-1} \times 2^{\mathbf{31}}$$

*küsimus:*

Kuidas muutuks **UPA diapasoon** eelmises ülesandes, kui :
**mantiss m** oleks 10 järgu asemel **11** kahendjärku
ja

astendaja **p** oleks 7 järgu asemel **8** kahendjärku ?
....ehk mis juhtub kui **m** või **p** esitamiseks lisandub veel üks **2nd**järk ?

*ülesanne:*

Esitada normaliseeritud UPA-na $-\mathbf{0.125}_{10}$ ja $\mathbf{9.375}_{10}$ negatiivsed: **mtk**-ga. Mantissi ja astendaja pikkused valida vabalt: vaja kasutada vähemalt niipalju järke, kui on vajalik väärtuse täpseks esitamiseks.

---

$-\mathbf{0.125}_{10} = -00.001_2 = 11.111_{tk}$ ( *kinnispunktarvuna* )

saadud *kinnispunktarvu* kuulutame UPA *normaliseerimata mantissiks* ja "komplekteerime" talle juurde astendaja 0 :

**m** = $11.111_{tk}$ **p** = 00000

lõpuks *normaliseerime* mantissi mille käigus korrigeerime ka astendajat

$-\mathbf{0.125}_{10}$ : **m** = $11.000_{tk}$ **p** = $11101_{tk}$

$\mathbf{9.375}_{10} = 001001.011_2$ ( *kinnispunktarvuna* )

**m** = $001001.011_2$ **p** = $00000_2$ ( *normaliseerimata UPA-na* )

$\mathbf{9.375}_{10}$ : **m** = $00.1001011_2$ **p** = 00100 (*UPA-na* )

*ülesanne:*

Sama eespool olnud UPA formaat **mtk**-s:

*mantiss*: 10 kahendjärku *astendaja*: 7 kahendjärku.

Teisendada UPA-kujule ja **arvutada** sellises formaadis UPA-dega :

**A** = $-0.3_{10}$ **B** = $2.7_{10}$

**A + B**
**A — B**
**|A| × B** ( *positiivsed operandid* )
**|A| : B** ( *positiivsed operandid* )

leiame operandid ujupunktarvudena (ehk leiame nende **m** ja **p** )
UPA võib leida kahel viisil :
1. eelnäidatud teel :
*kinnispunktarv >>> normaliseerimata UPA >>> normaliseeritud UPA*
või
2. eksponentkuju "matemaatilisel kaasabil", mis võimaldab kiiremini teadasaada *normaliseeritud UPA* **m** ja **p** väärtused :

**A** = $-0.3 = -0.3 \times 2^{0} = -0.6 \times 2^{-1}$

normaliseeritud **m** absoluutväärtus on teatavasti väärtusvahemikus

$0.5_{10} \leq \mathbf{m} \leq 1_{10}$

ilmneb, et arvu **- 0.3** normaliseeritud mantiss on väärtusega **- 0.6** ja tema astendaja on väärtusega **-1**

$+\mathbf{0.6}_{10} \approx 0.463_8 = 00.100110011_2 \approx 00.10011010_2$ (ümardatud)

$\mathbf{m}_A = 11.01100110_{tk}$ $\mathbf{p}_A = 1111111_{tk}$

analüüsides sama mõttekäiguga arvu B = 2.7 leiame tema *normaliseeritud mantissiks* ja *astendajaks* :

$$\mathbf{B} = 2.7 = \frac{2.7}{4} \times 2^{2}$$

$\mathbf{0.7}_{10} \approx 0.546_8 = 00.101100110_2$

$\mathbf{2.7}_{10} \approx 0010.1011001_2$
( ümardatud KPA, mille loeme *normaliseerimata mantissiks* ja normaliseerime ta )

$\mathbf{m_B} = \frac{2.7}{4} = 00.101011001 \approx 00.10101101$ $\qquad \mathbf{p_B} = 0000010$

## ARITMEETIKATEHTED UJUPUNKTARVUDEGA

**UPA aritmeetikatehete emulatsioon kinnispunktarvude aritmeetika kaudu**

**liitmine**
ja
**lahutamine** (ehk negatiivse liidetava liitmine)
tähistame *summa* : $\mathbf{C = A + B}$
liidame A ja B väärtusi esitavad avaldised :

$$\mathbf{C = A + B} = m_A \times 2^{p_A} + m_B \times 2^{p_B} =$$

ja teisendame saadud avaldist, tuues **kahe astme** sulgude ette:

$$= 2^{p_A} ( \quad ? \quad ) =$$

või

$$= 2^{p_B} ( \quad ? \quad )$$

$$= 2^{p_A} \left( m_A + m_B \times 2^{p_B - p_A} \right) =$$

või

$$= 2^{p_B} \left( m_A \times 2^{p_A - p_B} + m_B \right)$$

$$= 2^{p_A} \left( m_A + m_B \times 2^{p_B - p_A} \right) =$$

$p_C$ ; $m_C$

$$= 2^{p_B} \left( m_A \times 2^{p_A - p_B} + m_B \right)$$

operandide kahest astendajast **suurem** on summa astendajaks $\mathbf{p_C}$ ;
siin: $\mathbf{p_B}$

väiksema astendaja (siin: $\mathbf{p_A}$ ) valik summa astendajaks annaks summa mantissi ületäitumise, kuna sulgudes olevat ühte mantissi tuleks seljuhul "korrutada suuremaks"

summaks oleva UPA **C** *mantiss* ja *astendaja* :
$\mathbf{m_C} = 00.10011001$ $\qquad \mathbf{p_C} = 0000010$

$\mathbf{A + B} = 10.011001_2 \approx 2.4_{10}$

tähistame *vahe* : $\mathbf{C = A - B}$

$$\mathbf{C = A - B} = m_A \times 2^{p_A} - m_B \times 2^{p_B} =$$

$$= 2^{p_A} \left( m_A - m_B \times 2^{p_B - p_A} \right) =$$

$p_C$ ; $m_C$

$$= 2^{p_B} \left( m_A \times 2^{p_A - p_B} - m_B \right)$$

ilmneb, et UPA lahutamise ainus erinevus (liitmise suhtes) on :
$+ \mathbf{m_B}$ asemel osaleb resultaadi mantissi arvutamisel $- \mathbf{m_B}$
.... ehk **A - B** *mantissi* leidmisel (sulgudes) tuleb liita $\mathbf{m_B}$ *täiendkood*

**A - B = C** resultaadi **astendaja** on endiselt $p_C = p_B$

(korrektselt ümardatud operandidega) lahutamisel saame

$m_C = -0.75_{10}$ ( täpselt ! )

$A - B = -0.75_{10} \times 2^2 = -3$ ( UPA väärtus **m** ja **p** kaudu )

**korrutamine**

tähistame *korrutise* : **C = A × B**

korrutame A ja B väärtusi esitavad eksponent-avaldised :

$$C = A \times B = m_A \times 2^{p_A} \times m_B \times 2^{p_B} =$$

see avaldis osutub *nelja teguri* korrutiseks . . . . järjestame tegurid ümber :

$$= m_A \times m_B \times 2^{p_A} \times 2^{p_B} =$$

*. . . võrdsete alustega astmete korrutamisel astendajad liidetakse* :

$$= m_A \times m_B \times 2^{p_A + p_B}$$

**korrutise** mantiss ja astendaja arvutuvad seega operandide mantissidest ja astendajatest nii:

$$= \underbrace{m_A \times m_B}_{m_C} \times 2^{\overbrace{p_A + p_B}^{p_C}}$$

*. . . . osutub, et UPA* **korrutamine** *on lihtsam kui nende* liitmine . . .

meie ülesandes seega vaja (positiivsed) mantissid korrutada

korrutise *mantissi* $m_C$ saamiseks: $m_C = m_A \times m_B$

korrutise *astendaja* : $p_C = p_A + p_B$

korrutis *normaliseeritud* UPA-na :

$m_C$ = 00.11001100 $p_C$ = 0000000

**jagamine**

tähistame *jagatise* : **C = A : B**

jagame A ja B väärtusi esitavad eksponent-avaldised :

$$C = A : B = m_A \times 2^{p_A} : m_B \times 2^{p_B} =$$

$$= \frac{m_A \times 2^{p_A}}{m_B \times 2^{p_B}} =$$

$$= \frac{m_A}{m_B} \times \frac{2^{p_A}}{2^{p_B}} =$$

*. . . võrdsete alustega astmete* **jagamisel** *astendajad* **lahutatakse** :

$$= \frac{m_A}{m_B} \times 2^{p_A - p_B}$$

$$\underbrace{\frac{m_A}{m_B}}_{m_C} \times 2^{\overbrace{p_A - p_B}^{p_C}}$$

$m_A$ = 11.01100110 $_{tk}$

$|\mathbf{m}_A| = 00.10011010_2$

$\mathbf{m}_B = 00.10101101_2$

(positiivsete) mantisside **jagamine** ( jagatise mantissi $\mathbf{m}_C$ saamiseks) :

$|\mathbf{m}_A| : \mathbf{m}_B = 00.10011010 : 00.10101101$

$\mathbf{m}_C = 00.11100011_2$

jagatise astendaja $\mathbf{p}_C$ :

$\mathbf{p}_C = 1111111_{tk} - 0000010_2 = -1_{10} - 2_{10} = -3_{10}$

$\mathbf{p}_C = 1111101_{tk}$

jagatis **kinnispunktarvuna** :

$\mathbf{C = A : B} = 0.00011100011_2 = \mathbf{0.1111111111}...._{10}$

kaasaegsetes arvutites ei kasutata enam sellist *emulatsiooni* UPA-dega arvutamisel vaid UPA aritmeetikatehted teostatakse **matemaatika kaasprotsessori** poolt *riistvaras* :
suured *loogikaskeemid* arvutavad tehte tulemuseks oleva UPA kõik järgud

**JAGAMISALGORITMID**

Jagamine **jäägi taastamisega**
Jagamine **jäägi taastamiseta**

*peab kehtima: jagatav < jagaja.*
*Jagatis on kahendpuhtmurdarv:* $\mathbf{0} < jagatis_2 < \mathbf{1}$

Leida jagatise **9 : 13** kahendkuju, jagades **jäägi taastamisega**.
*ja järgnevalt ka :*
Leida jagatise **9 : 13** kahendkuju, jagades **jäägi taastamiseta**.

need jagamisalgoritmid annavad tulemuse / jagatise **2nd**kujul

ehk tegemist on 2ndjagamise algoritmidega
kuid algoritmi samme saab rakendada ka **10nd**kujul andmetele
(kus nii operandid kui ka vahetulemused on **10nd**kujul)

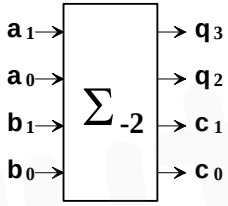
---

$9 : 13 = \mathbf{0.1011000100111}..._2$

*Jäägi TAASTAMISETA jagamisalgoritm sisaldab vähem samme ja seega ta teostab jagamise "kiiremini".*

*ülesanne:*

On **—2ndsummaator** , mis liidab **2** 2-järgulist **-2nd**arvu.




$(\mathbf{a_1a_0})_{-2} + (\mathbf{b_1b_0})_{-2} = (\mathbf{q_3q_2c_1c_0})_{-2}$

Leida summa üksikuid järke arvutavad loogikafunktsioonid (MDNK):

$$\begin{cases} q_3 = f(a_1\,a_0\,b_1\,b_0) \\ q_2 = f(a_1\,a_0\,b_1\,b_0) \\ c_1 = f(a_1\,a_0\,b_1\,b_0) \\ c_0 = f(a_1\,a_0\,b_1\,b_0) \end{cases}$$

---

(mõnede) 2-järguliste **-2nd**arvude summad:

$01_{-2} + 01_{-2} = 0110_{-2} = 2_{10}$

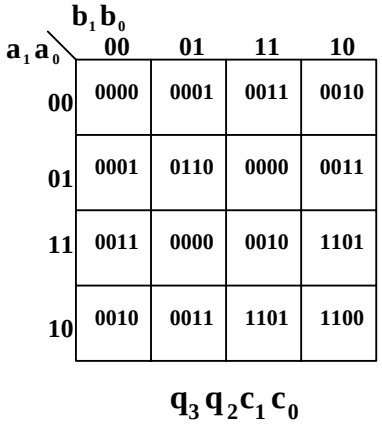
$01_{-2} + 10_{-2} = 0011_{-2} = -1_{10}$

$01_{-2} + 11_{-2} = 0000_{-2} = 0_{10}$

$10_{-2} + 10_{-2} = 1100_{-2} = -4_{10}$

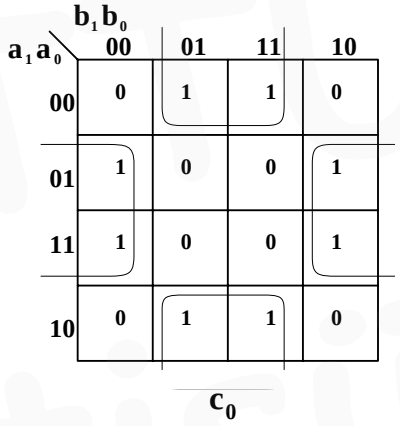
( . . . . meile on vajalikud **kõik**võimalikud 2-järguliste **-2nd**arvude summad — osad puuduvad siin loetelus . . . . )

**4 tõeväärtustabelit** samal kaardil:



| $a_1a_0$ \ $b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0000 | 0001 | 0011 | 0010 |
| 01 | 0001 | 0110 | 0000 | 0011 |
| 11 | 0011 | 0000 | 0010 | 1101 |
| 10 | 0010 | 0011 | 1101 | 1100 |

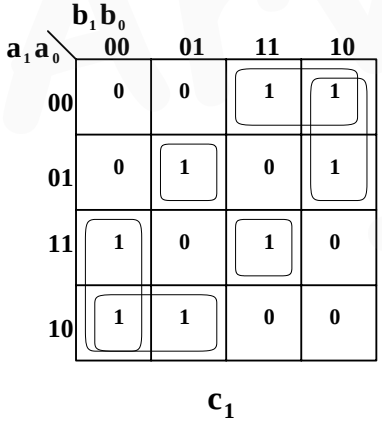
$q_3\,q_2\,c_1\,c_0$



| $a_1a_0$ \ $b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 0 | 1 | 1 | 0 |

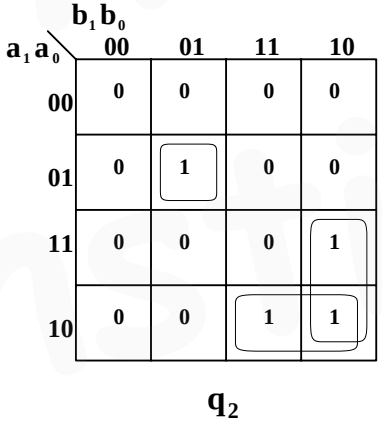
$c_0$

MDNK: $c_0 = a_0\bar{b}_0 \vee \bar{a}_0 b_0$



| $a_1a_0$ \ $b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 1 | 1 | 0 | 0 |

$c_1$



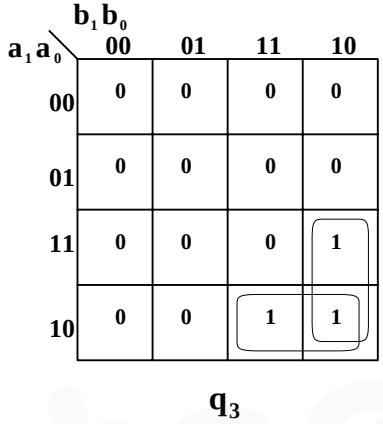
| $a_1a_0$ \ $b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 1 | 1 |

$q_2$

MDNK: $c_1 = a_1\bar{b}_1\bar{b}_0 \vee a_1\bar{a}_0\bar{b}_1 \vee \bar{a}_1 a_0\bar{b}_1 b_0 \vee$

$\vee \bar{a}_1\bar{a}_0 b_1 \vee \bar{a}_1 b_1\bar{b}_0 \vee a_1 a_0 b_1 b_0$

$q_2 = \bar{a}_1 a_0\bar{b}_1 b_0 \vee a_1\bar{a}_0 b_1 \vee a_1 b_1\bar{b}_0$



| $a_1a_0$ \ $b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 1 | 1 |

$q_3$

$q_3 = a_1\bar{a}_0 b_1 \vee a_1 b_1\bar{b}_0$

*ülesanne:*

Teisenda **2nd**arv $110110101_2$ **10nd**arvuks, saades otsitava arvu 10ndnumbrid *jagamise jääkidena* , uue alusega ehk 10-ga ($\mathbf{1010}_2$ ) jagamise teel **2nd**süsteemis

*meenutame:*

täisarvu teisendus ühest süsteemis teise saab toimuda uue arvusüsteemi alusega jagamise teel , kusjuures uue saadava arvu järgud ehk numbrid ilmnevad jagamise **jääkidena**.

Tavaliselt ei ole mõtet teisendada **2nd**arvu **10nd**kujuks (ehk *väärtuseks* ) niimoodi *jagamise* teel, kuna leidub palju kiirem/mugavam tee — kuid

näitame, et ka niimoodi 2ndsüsteemis jagamise teel on teisendus **10nd**kujule võimalik

## ALGORITMIDE GRAAFSKEEMID (AGS)

*ülesanne:*

Koostada algoritm *Algoritmi Graafskeemina* ( **AGS** ) , mis **korrutab** registris A ( **RgA** ) sisalduva arvu **23**-ga, kasutades **nihutamist** ja **summeerimist**.

Tulemus (korrutis) võib tekkida mujale registrisse, mitte samasse **RgA**

Koostada ka seda algoritmi realiseeriva **operatsioonseadme struktuurskeem** koos *juhtsignaalidega* / *juhtkäskudega* $y_i$ .

Riistvara püüda kasutada mitte rohkem kui minimaalselt on vajalik.

*. . . . esitame 23 "kahe astmete" summana — see on alati võimalik :*

**RgA × 23 = RgA × ( 16 + 4 + 2 + 1 ) =**

*. . . . korrutame sulud lahti :*

**= RgA × 16 + RgA × 4 + RgA × 2 + RgA =**

. . . . arvestades tegevuste järjekorda tulevases algoritmis, soovime siin avaldises järjestada liidetavad ümber vastupidisesse järjekorda :

**= RgA + RgA × 2 + RgA × 4 + RgA × 16 = RgA × 23**

selline avaldis ongi arvutatav **nihutamisega** ja **summeerimisega**

. . . AGS on siin slaidil puudu ja OPseadme struktuurskeem kah puudu . . .

*ülesanne:*

Koostada algoritm (AGS-ina), mis teisendaks loomulike kaaludega (8421) **BCD-koodis** positiivse arvu tavaliseks **2ndarvuks**.

*näide:* **0001 0011 0100 → $10000110_2$**

*pöörame tähelepanu* **10nd***arvu üksikutele järkudele :*

**N = ( $a_n$ $a_{n-1}$ . . . . $a_2$ $a_1$ $a_0$ )$_{10}$**

selle **10nd**arvu **N** väärtus on avaldatav / arvutatav avaldisena :

**N = [( $a_n$ × 10 + $a_{n-1}$ ) × 10 + $a_{n-2}$ ] × 10 + . . . . + $a_0$ =**

**= [( $a_n$ × (8+2) + $a_{n-1}$ ) × (8+2) + $a_{n-2}$ ] × (8+2) + . . . . + $a_0$ =**

kui selline avaldis....

a) kujundada **algoritmiks** (AGS näiteks) ;

b) **luua** seda algoritmi teostav **digitaalseade** ;

c) seade "tööle panna" ehk lasta loodud seadmel see algoritm korra läbida ;

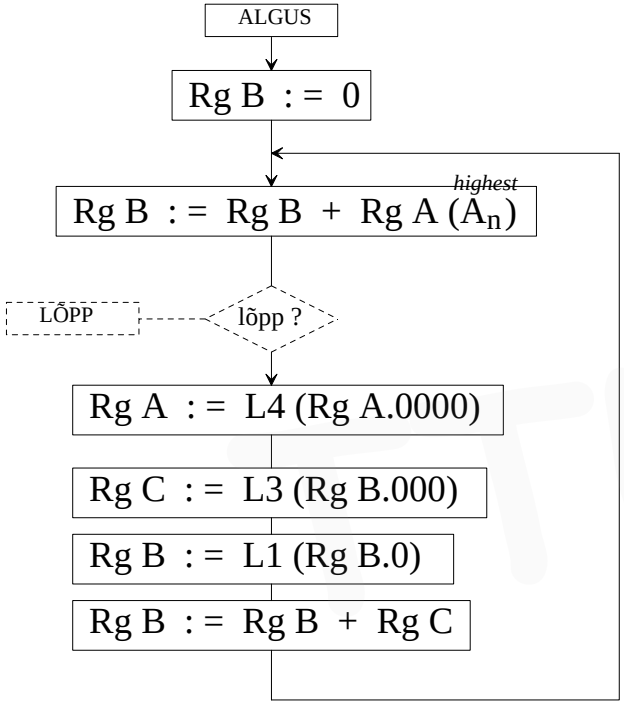
. . . . siis tema töötulemus ongi vajalik resultaat **2nd**kujul seadme mingis registris

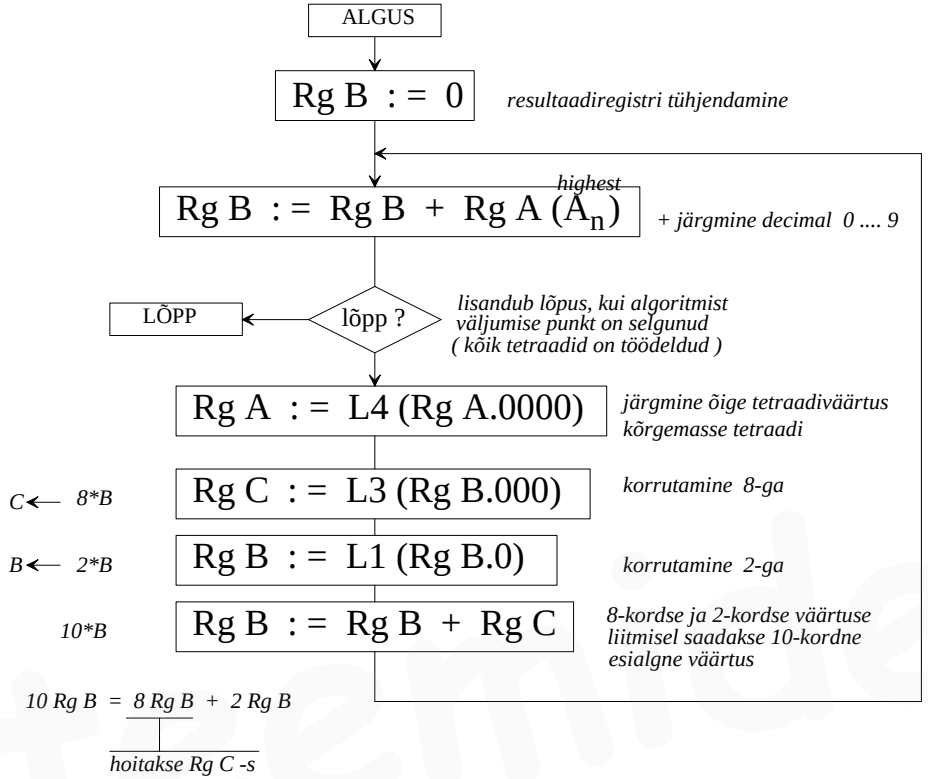
algoritmi koostamise käigus selgub, et vaja on **3 registrit** :

**Rg A** : algne **BCD** (teisendatav)

**Rg B** : **binary** (tulemus)

**Rg C** : *abiregister*




eelnev **AGS** korratud koos sammude selgitustega / kommentaaridega :

*ülesanne:*

Koostada ja esitada AGS-ina vastupidise teisenduse algoritm:
Teisendada **2ndarv** liiase 3-ga **BCD koodi** :
**binary** → **BCD** 8421(+3)

*näide:* **$10000110_2$** → **0100 0110 0111**

*koostatava algoritmi põhimõte :*

arvu **10nd**kuju üksikud numbrid saab **2nd**arvust genereerida tema (korduva) **jagamise teel 10**ga, kus nad tekkivad **jagamise jääkidena** :

*näide arvu* **134** *teisenduse jaoks:*

134 : 10 = 13 ( jääk **4** )
13 : 10 = 1 ( jääk **3** )
1 : 10 = 0 ( jääk **1** )

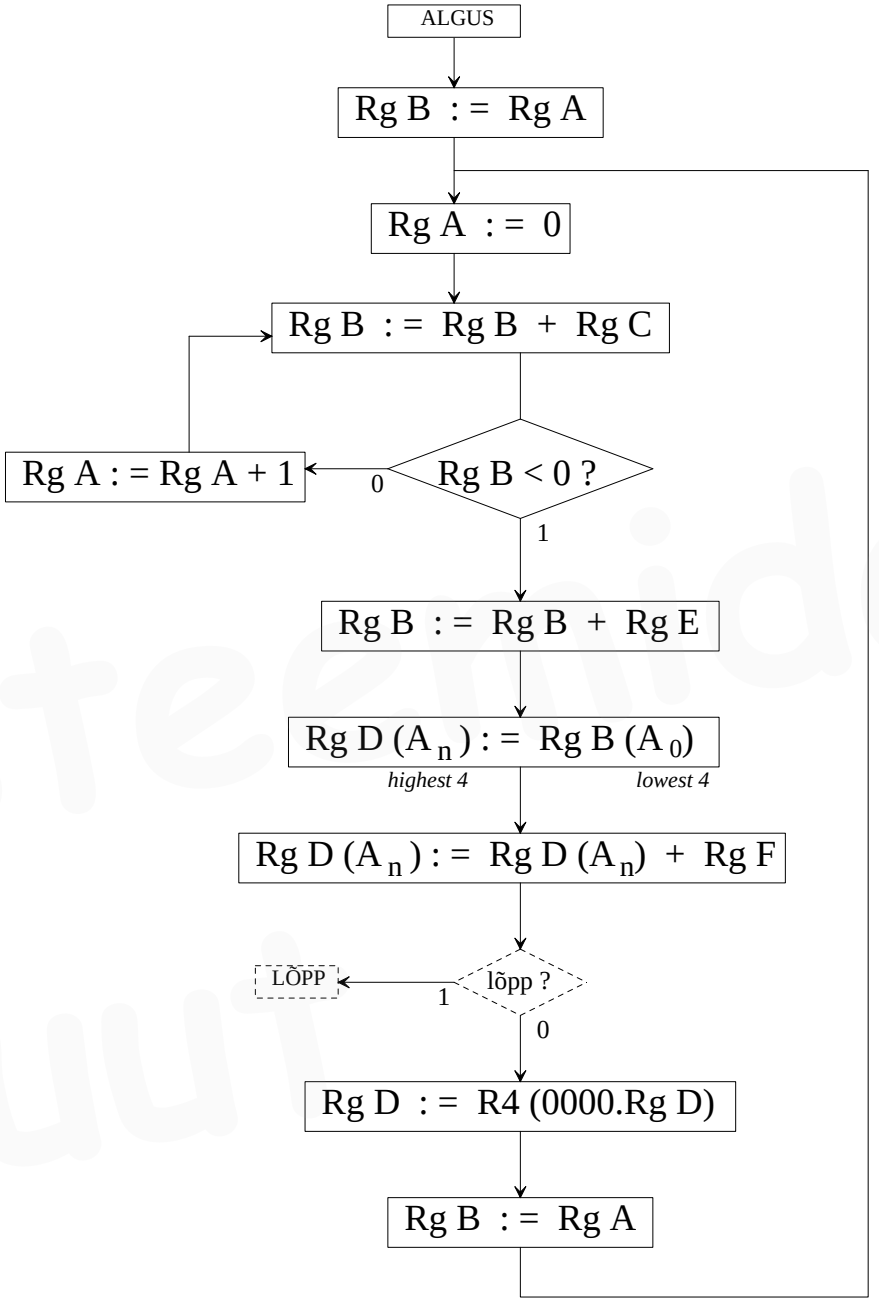
**?** kuidas algoritm jagab **10**ga ? — oletades, et meil pole kasutada spetsiaalset **jagajat** eraldi moodulina

**10**-ga **jagada** saab: **lahutades** korduvalt 10-t ja loendades lahutamise kordi, kuni lahutamise tulemus saab < 0

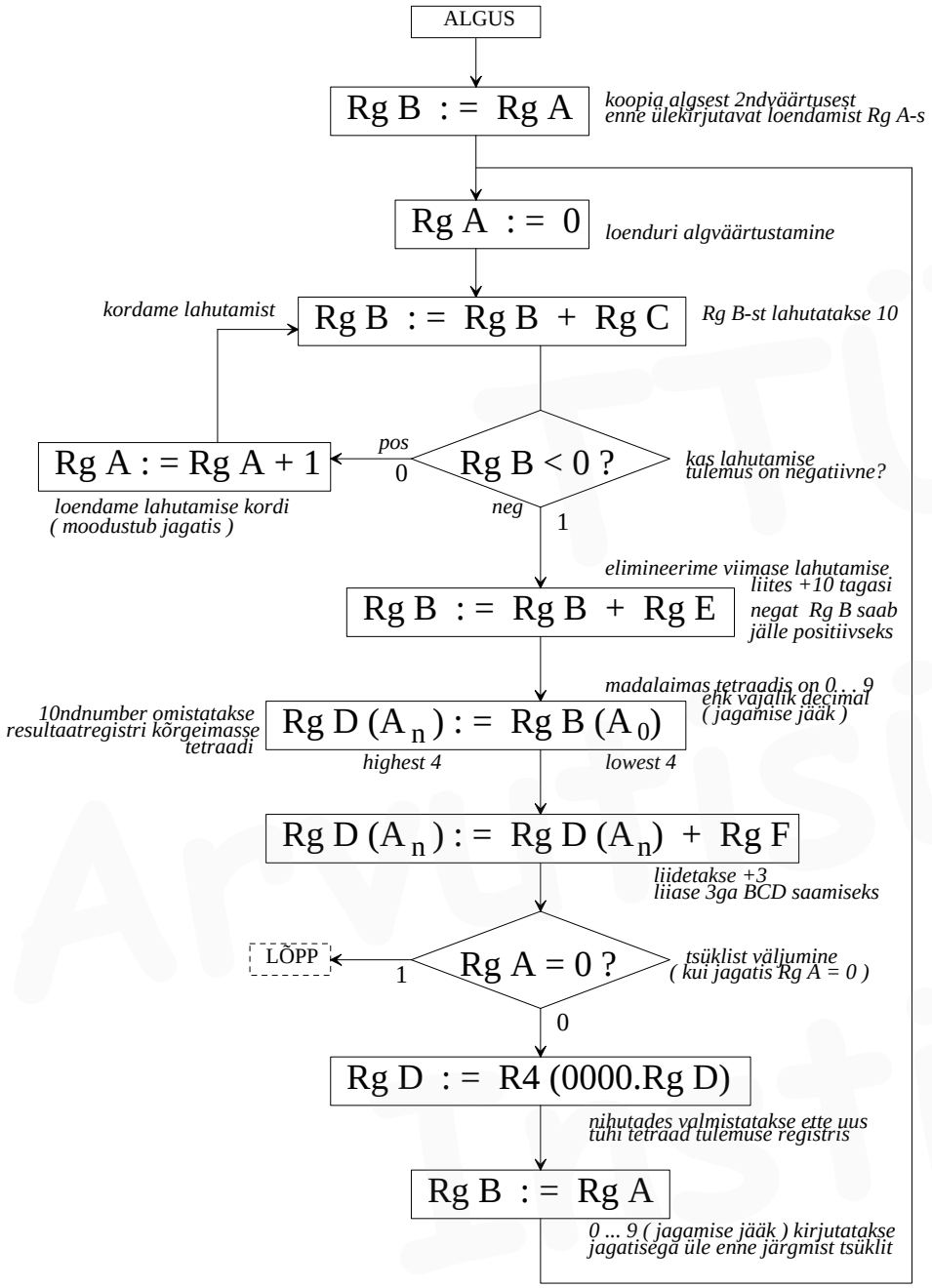
algoritmi koostamisel selgub, et vaja on **6 registrit** :

( *3 registrit nendest sisaldavad konstanti* )

**Rg A** : algne **binary** (teisendatav)
**Rg B** : jäägi akumulaator
**Rg C** : *konstant* $-10_{10}$ = . . . . $1110110_{tk}$
**Rg D** : resultaat **BCD** (8421) + 3 ( XS3 )
**Rg E** : *konstant* $+10_{10}$ = . . . . $001010_2$
**Rg F** : *konstant* $+3_{10}$ = . . . . $0011_2$




eelnev **AGS** korratud koos sammude selgitustega / kommentaaridega :


ALGUS
Rg B : = Rg A
koopia algsest 2ndväärtusest enne ülekirjutavat loendamist Rg A-s
Rg A : = 0
loenduri algväärtustamine
kordame lahutamist
Rg B : = Rg B + Rg C
Rg B-st lahutatakse 10
Rg A : = Rg A + 1
loendame lahutamise kordi ( moodustub jagatis )
pos
0
Rg B < 0 ?
kas lahutamise tulemus on negatiivne?
neg
1
elimineerime viimase lahutamise liites +10 tagasi negat Rg B saab jälle positiivseks
Rg B : = Rg B + Rg E
madalaimas tetraadis on 0 . . 9 ehk vajalik decimal ( jagamise jääk )
10ndnumber omistatakse resultaatregistri kõrgeimasse tetraadi
Rg D (A n ) : = Rg B (A 0)
highest 4
lowest 4
Rg D (A n ) : = Rg D (A n) + Rg F
liidetakse +3 liiase 3ga BCD saamiseks
LÕPP
1
Rg A = 0 ?
tsüklist väljumine ( kui jagatis Rg A = 0 )
0
Rg D : = R4 (0000.Rg D)
nihutades valmistatakse ette uus tühi tetraad tulemuse registris
Rg B : = Rg A
0 ... 9 ( jagamise jääk ) kirjutatakse jagatisega üle enne järgmist tsüklit