



Arvuti **Tartu**
kasutaja juhend

Arvuti **Tartu**
kasutaja juhend

Tartu Ülikool 1989

Sisukord

Sissejuhatus	3
Arvuti töökorda seadmine	5
Klaviatuur ja ekraan	8
Keele T-BASIC interpretaator	
Interpretaatori töörežiimid	14
Sisendrea formaat	14
Tähestik	15
Konstandid	16
Muutujad	17
Avaldised	19
Keele T-BASIC konstruktsioonide loetelu	22
Tähistused	22
Direktiivid	23
Funktsioonid	44
Masinkoodis kirjutatud alamprogrammide kasutamine	51
Interpretaatori mäluaotus	55
Interpretaatori teated	56
Monitor	
Monitori direktiivid	60
Mälu sisu trükkimine	62
Mäluosade võrdlemine	64
Mälu sisu muutmine	66
Mäluosa kopeerimine	67
Välisseadmete juhtimine	68
Programmide käivitamine	69
Ekraanidirektiivid	69
Magnetofoni kasutamine	70
Arvuti mäluaotus	71
Ekraani formaat	74
Monitori mäluaotus	75
Monitori alamprogrammid	76
Indeks	81
Lisa 1. Magnetofoniga seotud teated	93
Lisa 2. Laienduspesa X2 kirjeldus	94
Lisa 3. Arvuti Tartu tähestik	97

Sissejuhatus

Personaalalarvuti Tartu on loodud Tartu Ülikoolis 1984. aastal. Arvutis kasutatakse kaheksabitist mikroprotsessorit KP580BM80A. Operatiivmälu on 64K baiti, millest 12K on eraldatud graafikaekraani tarbeks. Püsimälu võib arvutis olla 6K kuni 16K baiti.

Arvutiga ühendatud televiisori ekraanile saame joonistada 384*256 punktist koosnevat pilti ning trükkida 25 rida tähti, numbreid ja muid sümboleid, igas reas 64 täheruumi. Trükkali või plotteri olemasolul võime pilte ja tekste ka paberile jäädvustada.

Arvuti välismäluna saame tarvitada kassettmagnetofoni, kuid vastavate kontrollerite olemasolul ka 5- või 8-tolliseid ümbrikkettaseadmeid ning EC-tüüpi arvutite 7 või 29 megabaidise mahuga ketassalvesteid.

Välismaailmaga on lihtne sidet pidada interfeisi RS232 abil.

Ilma mingisugust välismälu kasutamata saame töötada programmeerimiskeeltes T-BASIC, mille interpretaator paikneb arvuti püsimälus, samuti kui arvuti juhtprogramm -- Monitor, mille

abil saame koostada ning siluda lihtsamaid masinkoodis kirjutatud programme. Kui tahame oma töö tulemusi säilitada, võime need kirjutada magnetofonile.

Kui meil on võimalik kasutada kettaseadet (-seadmeid), avar-duvad arvuti kasutusvõimalused veelgi. Sel juhul töötame operatsioonisüsteemis CP/M, milles on aastate jooksul kirjutatud palju kasulikke programme. Näiteks saame programmeerida järgmistes keeltes: *Assemblier*, *Basic*, *C*, *Forth*, *Fortran*, *Lisp* ja *Pascal*. Töötavad ka andmetötlussüsteemid *dBase II* ja *SuperCalc* ning tekstiredaktor *WordStar*. Spetsiaalselt arvuti Tartu tarvis on kirjutatud tekstiredaktor *TE*, mis eriti hästi sobib programmitekstide sisestamiseks ja redigeerimiseks.

Arvuti töökorda seadmine

Arvuti töökorda seadmiseks tuleb:

- 1) ühendada arvutiga monitori kaabel;
- 2) ühendada monitoriga toiteploki kaabel;
- 3) ühendada arvutiga magnetofoni kaablid;
- 4) ühendada vooluvõrguga magnetofoni, monitori ja arvuti toitejuhtmed

(pistikupesade ja lülitite paigutus on näidatud arvuti tagaseina joonisel lk. 7).

Seejärel peame sisse lülitama monitori toiteploki ning lõpuks arvuti (tagaseinal paiknevast toitelülitist). Kui paarikümnne sekundi jooksul pole ekraanile mingit pilti ilmunud (proovime pilti kätte saada ka monitori heledus- ning kontrastsusnuppusid pöörates), oleme kas midagi valesti ühendanud või on arvuti korraast ära. Kui ekraanile ilmub veidi mälelauda meenutav pilt, peame vajutama arvuti tagaseinal asuvale nupule RESET. Märgime siinkohal, et alati on võimalik imiteerida arvuti sisselülitamist, hoides klahvi CTRL alla vajutatuna ning vajutades siis nupule RESET.

Korralikult töötava arvuti ekraani ülaossa peab tekima tekst "Tere!" ning sellest veidi alla ja vasakule sümbol mille järel paikneb vilkuv ristkülik -- kursor. Nüüd on arvuti valmis vastu võtma ning täitma programmi Monitor käske (selle programmiga tutvume põhjalikult veidi hiljem). Programmeerimiskeele T-BASIC interpretaatori (lühemalt: keele BASIC) käivitamiseks peame kas teistkordsest vajutama nupule

RESET või sisestama klaviatuurilt rea

d000g

(ärme unustame rea lõpus vajutamast klahvile RETURN; väike-tähtede d ja g asemel võime tarvitada ka suurtähti D ja G; numbri 0 ja tähe O eristamiseks kujutatakse nulli nii klaviatuuril kui ka ekraanil läbikriipsutatuna). Seejärel peab arvuti kogu ekraani puhastama ning ekraani ülemisse vasakusse nurka kirjutama teksti

***. T-BASIC ***

OK

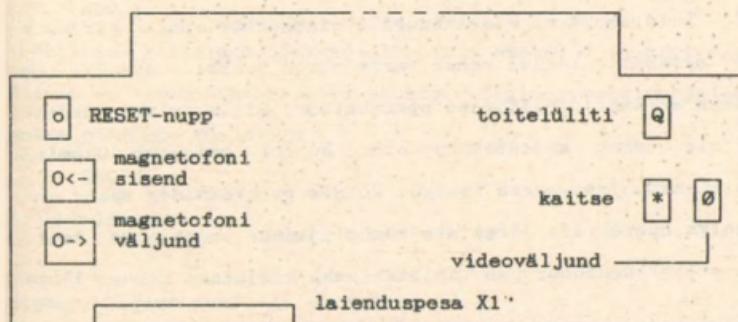
Kursor paigutub sõna OK alla. Nüüd on T-BASIC valmis meiega suhtlemiseks.

Käesolevas juhendis ei hakka me õppima ei programmeerimist üldse ega ka programmeerimist keeles BASIC, vaid tutvume selle keele arvutil Tartu töötava versiooniga T-BASIC eeldusel, et meil on programmeerimise algteadmised olemas. Edaspidi kasutame nime T-BASIC asemel nime BASIC. Kõigepealt aga uurime lähemalt arvuti ekraani ning klaviatuuri tööd.

Arvuti Tartu klaviatuur



Arvuti tagasein



Laienduspesa X2 paikneb arvuti parempoolsel külgsinal

Arvuti ekraanil on 25 rida (numbritega 0, ..., 24) ning 48 või 64 veergu (numbritega 0, ..., 47 või 63). Programm Monitor kasutab "kitsamat" ekraani, keele BASIC interpretaator (ja praktiliselt kõik teised programmid) aga "laiemat", millele mahub 1600 sümbolit. Väljastamise kohta ekraanil näitab (siis, kui arvuti parajasti cotab meilt klahvivajutust) vili- kuv ristkülik -- kurSOR.

Ekraanile väljastatakse sümboleid harilikult vasakult paremale; pärast rea viimase sümboli trükkimist minnakse üle järgmissele reale. Pärast kõige alumise rea viimase sümboli väljastamist nihkub kogu tekst ekraanil ühe rea vörra ülespoole, sealjuures läheb kõige ülemine rida kaduma ning ekraani allserva tekib tühi rida.

Soovi korral saame kursoori paigutada soovitud kohale ekraanil, kuid seejärel klaviatuurilt sisestatav tekst kirjutab üle ekraanil sellel kohal varem olnud teksti. Arvutis on olemas ekraani puastamise operatsioon, mille toimel ekraanil olev tekst kustutatakse ning kurSOR paigutub ülemise rea vasakpoolseimassse veergu. Kõigis programmis saame ekraaniga opereerida järgmiste sümbolijadade trükkimise teel. (Kuueteistkümnendarvude tähistamiseks kirjutame nende lõppu tähe h.)

Ekraani puastamiseks ning kursoori viimiseks ekraani ülemisse vasakusse nurka tuleb programmis ekraanile väljastada sümbolid

ESC ja *

(ESC on juhtsümbol kuueteistkümnendkoodiga 1Bh või kümnendkoodiga 27). Sümbol kümnendkoodiga

30 (1Eh)

paigutab kursori ülemisse vasakusse nurka ekraanipilti muutmata. Kursori paigutamiseks soovitud positsiooni tuleb väljastada 4 sümbolit:

ESC ja =

seejärel soovitud rea number, millele on liidetud 32 (20h) ning lõpuks soovitud veeru number, millele on samuti liidetud 32. Kursorit liigutavad veel järgmised juhtsümbolid: 11 (ØBh) -- ühe rea võrra ülespoole; 12 (ØCh) -- ühe veeru võrra paremale; 8 (Ø8h) -- ühe veeru võrra vasakule.

Tavaliselt joonistatakse sümbolid ekraanile heledatena tumedal taustal (seda nimetame positiivrežiimiks). Sümbolite

ESC ja (

trükkimise järel aga ilmuvald järgnevad sümbolid negatiivrežiimis -- tumedatena heledal taustal. Tagasi positiivrežiimi saame pöörduda sümbolite

ESC ja)

trükkimisega.

Lõpuks märgime veel, et sümbolite

ESC ja T

väljastamine kustutab rea kursorist lõpuni ja paigutab kursori järgmise rea algusesse.

Arvuti klaviatuuri joonis on toodud leheküljel 7. Klaviatuur

riilt saab sisestada kõiki tähestiku ASCII (vt. lisa 3) sümboleid ning lisaks eesti tähestiku tähti õÄÜÜõõõ ja kirillitsa tähti (need puuduvad jooniselt, küll aga on graveeritud klahvidele). Ühekordne lühike vajutus klahvile sisestab arvutisse ühe sümbole. Kui hoiamo klahvi pikemalt vajutatuna, ootab arvuti umbes ühe sekundi ning seejärel käitub nii, nagu me vajutaksime sedasama klahvi umbes 5 kuni 10 korda sekundis.

Ladina tähtede režiimi valikuks on kasutusel klaviatuuri keskmise rea paremas servas paiknev klahv pealdisega LAT, kirillitsa režiimi valib aga vajutus sama rea vasakus servas asuvale klahvile RUS.

Täheklahvile vajutamisel sisestame arvutisse klahvile graveeritud väiketähe (muidugi vastavalt registriklahvidele LAT ja RUS). Suurtähe saamiseks on kaks võimalust. Esiteks võime vajutada täheklahvile, hoides registriklahvi SH (SHIFT) vajutatuna. Kui meil on tarvis kirjutada pikemat suurtähtedest koosnevat teksti, võime (üks kord) vajutada klaviatuuri alumiises reas vasakult teisel kohal asuvale klahvile CAPS. Seejärel saame täheklahvide abil sisestada suurtähti (ja koos klahviga SH -- väiketähti). Teistkordne vajutus klahvile CAPS taastab klaviatuuri esialgse seisundi. Peame meeles, et me ei tohi hoida klahvi CAPS vajutatuna liiga kaua -- siis arvab arvuti, et me tahame seda režiimi korduvalt muuta.

Klaviatuuril on veel 20 klahvi, millele on graveeritud tähtedest erinevad sümboleid (numbrid, kirjavahemärgid jms.).

Neile klahvidele ei mõju registriklahvid LAT, RUS ega CAPS. Klahvi vajutamisel sisestame arvutisse alumisena graveeritud sümboli, kui aga hoiame klahvi SH vajutatuna, siis ülemise sümboli.

Klaviatuuri ülemise rea vasakus servas asuv klahv ESC ning sama rea paremas servas paiknev klahv RUB sisestavad arvutisse tähestiku ASCII sümbolid kuueteistkümnendkoodidega vastavalt 1Bh ja 7Fh. Alumise rea vasakpoolseim klahv TAB tekitab sümboli koodiga Ø9h. Neid kolme klahvi kasutavad eeskätt operatsioonisüsteemis CP/M töötavad programmid.

Klahvi RUB all paiknev klahv RETURN (mis annab arvutisse sümboli kuueteistkümnendkoodiga ØD) on praktiliselt kõigis programmides kasutusel rea sisestamise lõpu tunnusena.

Klaviatuuri paremas servas paiknevad klahvid DEL, COPY ning neli nooltega tähistatud klahvi on ette nähtud sisestatava rea redigeerimiseks ning töötavad järgmiselt. Nooleklahvide abil saame ekraanil liikuda soovitud kohta, sisendreas seejuures midagi ei muutu. Klahvi DEL vajutamisega saame sisendreast kustutada viimatisestatud sümboli (kui me oleme ekraanil ringi liikunud, siis ei pruugi see sugugi enam olla kursrist vahetult vasakul paiknev sümbol). Klahvi COPY abil saame sisendrea lõppu lisada kursori all paikneva sümboli (nagu oleksime vajutanud vastavale tähe- või numbriklahvide). Toome siinkohal ühe näite redigeerimisklahvide kasutamisest. Olgu meil tarvis sisestada kaks järjestikust BASIC-programmi rida

```
100 IF X$ = "y" THEN GOTO 200
110 IF X$ <> "n" THEN GOTO 90
```

Sisestame kõigepealt rea number 100, misjärel ekraanipilt näeb välja niisugune:

```
100 if x$ = "y" then goto 200
```

(allkriips tähistagu kursori asukohta). Paneme nüüd tähele, et juba sisestatud reas on tükke, mis kõlbaksid ka järgmise rea koosseisu. Redigeerimisklahvide abil saamegi neid ära kasutada ning ei pea kõiki sümboleid klaviatuurilt taga ajama. Liigume kõigepealt klahvi "nool üles" abil selmisele reale, saame ekraanile järgmise pildi:

```
100 if x$ = "y" then goto 200
```

Nüüd alustame sisestamist: vajutame klahvile COPY; sisendritta tekib kursori all olnud sümbol 1. Järgmise sümboli peame asendama numbriga 1 (me sisestame ju rida number 110), niisiis vajutame klahvile 1. Saame niisuguse pildi:

```
110 if x$ = "y" then goto 200
```

Nüüd võime klahvi COPY abil liikuda kuni võrdusmärgini, selle asemel aga vajutame klahvile < (ärme unustame klahvi SH all hoida). Ekraan näeb nüüd välja niisugune:

```
110 if x$ < "y" then goto 200
```

Kursori asukhta on meil vaja lisada sümbol >. Liigume vasa-ku noole klahviga ekraanil ühe täheruumi vasakule (sisend-reas sellest midagi ei muutu, seal jääb viimaseks sümboliks ikka <) ja vajutame klahvile >. Seejärel sisestame (2 korda klahvile COPY vajutades) tühiku ja jutumärgid, vajutame klahvile N ning oleme ekraanil saanud niisuguse pildi:

```
110 if x$ > "n" then goto 200
```

Klahvi COPY abil saame sisestada sõnad THEN ja GOTO, samuti neile järgnevad tühikud. Lõpuks sisestame klaviatuurilt numbrid 9 ja Ø. Nüüd on ekraanil järgmine pilt:

```
110 if x$ > "n" then goto 900
```

Vajutame klahvile RETURN (kursori all paiknev number Ø jääb sisestamata). Kontrolliks võime sisestada direktiivi LIST ning peame siis nägema järgmist kahte rida:

```
100 IF X$ = "y" THEN GOTO 200
```

```
110 IF X$ <> "n" THEN GOTO 90
```

Interpretaatori töörežiimid

Keel BASIC interpretaatoril on kaks töörežiimi. Esimeses neist -- direktiivide vahetu täitmise režiimis -- täidab interpretaator talle antud direktiivi(d) kohe, kui me oleme sisestamise lõpetanud (vajutanud klahvile RETURN). Reanumberiga algavat direktiivi aga ei täideta, vaid salvestatakse mällu. Sel viisil saame koostada kõeles BASIC kirjutatud programmi (lühemalt: BASIC-programmi). Oma teises töörežiimis -- programmi täitmise režiimis -- tegelebki interpretaator eelnevalt salvestatud programmi täitmisega.

Peaaegu kõiki direktiive tohib kasutada mõlemas režiimis; esinevaid piiranguid vaatleme vastavate direktiivide kirjeldamisel.

Sisendrea formaat

Klaviatuurilt sisestatakav rida tohib koosneda kuni 250 sümbolist ning võib sisalda ühe või mitu direktiivi. Direktiivid eraldatakse üksteisest kooloniga. Programmi koosseisu sisestatakava rea alguses peab paiknema reanumber -- täisarv piirkonnast 0 kuni 65529. Sisendrea üldkuju on seega järgmine:

[reanumber] direktiiv [: direktiiv ...]

(nurksulgudega ümbritsetud konstruktsiooniühikud võivad puu-

duda; kordumise tunnus ... tähendab, et viimane konstruktiooniühik võib esineda praktiliselt piiramatu arv kordi). Read salvestatakse programmi reanumbrite kasvamise järvkorras. Samas järvekorras toimub ka programmi täitmine (kui juhtimisdirektiividega ei ole teisiti määratud). Direktiivi sees on soovitatav võtmesõnad ümbritseda tähikutega.

Tähestik

Programmi kirjutamisel tohime kasutada tähti, numbreid ja kirjavahemärke. Tähtedeks loetakse inglise tähestiku 26 tähte

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

kusjuures väljaspool tekstikonstante samastab interpretaator suur- ja väiketähed. Numbriteks on kümme araabia numbrit Ø kuni 9 ning kirjavahemärkide hulka kuuluvad järgmised sümbolid:

- = võrdumis- või omistamismärk
- + plussmärk
- miinusemärk
- * korrutusmärk (tärn)
- / jagamismärk (kaldkriips)
- ~ astendamismärk
- (alustav sulg
-) lõpetav sulg
- \$ dollarimärk (tekstitüübi tähis)
- , koma

· punkt
; semikoolon
: koolon
? küsimärk
" jutumärgid
< alustav noolsulg (suhte "väiksem kui" tähis)
> lõpetav noolsulg (suhte "suurem kui" tähis)

Trükkikujuta sümbolitest kasutatakse tühikut ning reavahetuse sümbolit (saadaval klahvi RETURN abil).

Kõiki teisi arvuti tähestikus leiduvaid sümboleid (ülejäänud kirjavahemärke, kirillitsa tähti jne.) tohime kasutada ainult tekstikonstantides.

Konstandid

Konstante on keeles **BASIC** kahte tüüpi: teksti- ja arvulised konstandid.

Tekstikonstant on jutumärkides paiknev suvalistest sümbolitest koosnev sõne, mille pikkus on piiratud sisendrea pikku-sega. Siin tohime kasutada kõiki arvuti tähestikus leiduvaid sümboleid, v. a. jutumärke. Tekstikonstantideks on näiteks

"see on tekstikonstant"

"3556.789"

"jaääärne"

kuid pole

"ajaleht "Edasi""

(sisaldab jutumärke)

'suur reheahi'

(ei paikne jutumärkides)

Aryulised konstandid võivad olla esitatud harilikus kirjutusviisis või poollogaritmilisel kujul. Esimesel juhul koosneb arvuline konstant punktiga eraldatud täis- ja murdosast (nii üks kui ka teine võib ka puududa). Arvu ette võime kirjutada pluss- või miinusmärgi. Teisel juhul lisandub täis- ja/või murdosa (ehk mantissi) järele täht E (inglisekeelsest sõnast *exponent* - järk), mille järele omakorda kirjutame astendaja -- märgiga või märgita täisarvu. Arvulised konstandid on näiteks kirjutised

1

-3556.789

.3556789E+4

(see on sama, mis 3556.789)

Kõik arvud kujutatakse arvutis ühtviisi ja nimelt poollogaritmilisel kujul. Arvu mantiss sisaldab umbes 7 õiget tüvenumbrat, arvu järk aga tohib olla -38 kuni +38.

Muutujad

Muutujad jagunevad kahte liiki: lihtmuutujad ja massiivid. Mõlematel liikidel on veel kaks alamliiki: teksti- ning arvulised muutujad.

Lihtmuutujate ning massiivi elementide väärtsused võivad olla samasugused nagu vastavat tüüpi konstantidel (vt. ülalt). Teksttitüüpi muutujad võivad sisaldada kuni 255 sümbolit.

Igal muutujal on nimi, mis peab algama tähega ning koosnema tähtedest ja numbritest. Muutuja nime sisestamisel teisendab interpretaator väiketähed suurtähtedeks, seetõttu pole näiteks nimed ab ja AB erinevad. Nime pikkus pole küll piiratud, kuid translaator eristab nimesid kahe esimese sümboli järgi (seega saame ühes programmis kasutada kuni 936 ühte tüüpi muutujat). Teksttimuutuja nime lõppu tuleb lisada tunnus \$. Seega on lihtmuutujate nimedeks näiteks

A	(arvuline)
B2	(arvuline)
B2UUS	(arvuline; sama mis B2)
B2\$	(teksttitüüpi)

Massiivi nime järele tuleb sulgudesse kirjutada tema indeksite loetelu (vt. ka direktiivi DIM), näiteks

A(34)	(ühemõõtmeline arvumassiiv)
SUUR(I,J,K)	(kolmemõõtmeline arvumassiiv)
TEATED\$(5)	(ühemõõtmeline tekstimassiiv)

Avaldised

Avaldiseks loeb interpretaator

- 1) konstanti,
- 2) lihtmuutujat,
- 3) standard- või defineeritud funktsiooni poole pöördumist (vt. allpool),
- 4) aavaldiste ja tehtemärkide kombinatsiooni, mille tullemuseks on üks väärthus.

Vastavalt olemasolevatele muutujatüüpidele võivad aavaldised olla kas teksti- või arvutüüpi, kuid on olemas ka loogilist tüüpi aavaldised.

Arvulist tüüpi aavaldistes tohime kasutada järgmisi tehteid:

-	astendamine	a ^ 3
-	unaarne miinus	-bc
*	korrutamine ja jagamine	5 * r b / .3
+	- liitmine ja lahutamine	ab + cd 5 - 3.4

Selles loetelus ülalpool asuvad tehted teostatakse aavaldisse väärtsuse arvutamisel kõigepealt (neil on loetelus allpool asuvate tehete suhtes prioriteet). Võrdse prioritesiga tehete teostatakse aavaldises vasakult paremale. Tehete järjekorra muutmiseks tohib kasutada ümarsulgusid. Toome mõned näited harilikus kirjaviisis ning keele BASIC reeglite kohaselt kirjutatud arvutüüpi aavaldistest:

harilik kirjutusviis

keele BASIC kirjutusviis

$x + 2y$

$x + 2*y$

y

$x = -$

$x = y/2$

2

xy

$--$

$x*y/z$

z

$(x^2)y$

$(x^2)^y$

$x(-y)$

$x*(-y)$

Tekstitüpi avaldistes saame tekstikonstantidele, -muutujatele ning tekstitüpi funktsioonidele rakendada ühtainust tehet -- konkatenatsiooni, mida tähistatakse plussmärgiga, näiteks

`"Tere " + "hommikust" (= "Tere hommikust")`

`A$ + " ja " + B$`

Loogilist tüpi avaldise saame konstrueerida kahe aritmeetilist tüpi avaldise vahelle võrdlusemärgi paigaldamisega. Sellise avaldise väärtuseks on kas "tõene" või "väär". Keeles BASIC on töeväärtused kodeeritud arvudega -- "väär" on \emptyset ning "tõene" on -1 (või üldiselt nullist erinev). Võrdlus-

märgid on järgmised:

=	"võrdub"	$x = y$
\leftrightarrow	"ei võrdu"	$x \leftrightarrow y$
<	"on väiksem"	$x < y$
>	"on suurem"	$x > y$
\leq	"on väiksem või võrdne"	$x \leq y$
\geq	"on suurem või võrdne"	$x \geq y$

Keerulisemaid loogilisi avaldisi saame lihtsamate kombinererimisel loogiliste teheteega, milledest keeles on olemas negatsioon, konjunksioon ja disjunksioon, tehtemärkidega vastavalt NOT, AND ja OR. Need tehted on kõik madalama prioriteediga (täidetakse hiljem) kui aritmeetilised; negatsioon on kõrgema prioriteediga disjunksiooni ja konjunksiooni suhtes. Ka loogiliste tehete järjekorra muutmiseks tohime kasutada ümbersulgusid. Loogilisteks avaldisteks on seega näiteks

$3 < 4$

(alati tõene)

$-5.7 < x \text{ AND } x < .95$

$Q = \emptyset \text{ AND NOT}(w < 1 \text{ OR } w \geq 2)$

Teksti- ja arvutüüpi objekte ühes avaldises koos kasutada ei ole lubatud (keeles on olemas tüübiteisehodusfunksioonid, mida vaatleme hiljem). Küll tohib aga loogilist avaldist vaadelda arvulisena ning vastupidi, pidades meeles, et nul-list erinev arvuline väärus vastab loogilisele väärusele "tõene", "väär" aga kodeeritakse arvuna \emptyset .

Keelle T-BASIC konstruktsioonide loetelu

Tähistused

Keelekonstruktsioonide kirjapanekul kasutatakse järgmisi tähistusi:

1. Suurtähtedega kirjutatud konstruktsiooniühikud kuuluvad vahetult direktiivi koosseisu.
2. Väiketähtedega kirjutatud ning noolsulgudesse < ja > paigutatud konstruktsiooniühikud tuleb direktiivi kirjapanekul asendada konstandi, muutuja nime, avaldise või muu sama tüüpi konstruktsiooniga (täpsemalt on asendusvariante loetletud direktiivide kirjeldustes).
3. Nurksulgudesse [ja] paigutatud konstruktsiooniühikud võivad puududa.
4. Loogelistes sulgudes { ja } paiknevatest ning püstkriipsuga ; eraldatud konstruktsiooniühikutest tuleb valida täpselt üks.
5. Kõik kirjavahemärgid peale püstkriipsu, noolsulgude, nurksulgude ja loogeliste sulgude kuuluvad vahetult direktiivide koosseisu.
6. Termin ctrl-C on kasutusel tähistamaks kahe klahvi, ctrl ja C, üheaegset vajutamist.

Direktiivid

BEEP

Süntaks: BEEP <kõrgus>, <kestus>

Tegevus: Tekitab arvuti valjuhääldis antud kõrguse ja kestusega heli.

Kõige kõrgema tooni annab parameetri <kõrgus> väärthus 0 ning kõige madalama 65535. Heli kestus sõltub lineaarselt parameetri <kestus> väärustusest, mis võib olla 0 kuni 65535 (viimane väärthus annab umbes 2 sekundit kestva heli).

Näide:

```
10 BEEP 1000*RND(1), 50000*RND(1): GOTO 10
```

CLEAR

Süntaks: CLEAR [<tekstide piirkonna suurus>]

Tegevus: Kustutab kõik muutujad ja tühistab massiivikirjeldused. Sõna CLEAR järelle kirjutatud aavaldisi väärthus määrab tekstimuutujate ja -avaldiste värtuste jaoks eraldatud mäluosa pikkuse (vaikimisi on see 5120 baiti e. sümbolit).

CLS

Süntaks: CLS

Tegevus: Puhastab ekraani ja viib kursoori ülemisse vasakuse nurka.

Direktiivid NORMAL ja INVERSE määravad, missuguse värviga täidab CLS ekraani: kui neist ei viimaseks kasutatud NORMAL, värvib CLS ekraani mustaks, vastasel korral valgeks.

CONT

Süntaks: CONT

Tegevus: Jätkab programmi täitmist pärast direktiivi STOP või pärast programmi töö katkestamist klahvidega ctrl-C.

Programmi tööd ei saa jätkata pärast veateadet ja pärast paranduste tegemist programmi tekstis (küll aga on enne programmi uuestikäivitamist lubatud muuta muutujate väärtusi).

CUR

Süntaks: CUR <rida>, <veerg>

Tegevus: Paigutab kursoori antud ritta ning antud veergu ekraanil. Lubatud on reanumbrid 0, ..., 24 ning vee runumbrid 0, ..., 63 (kitsama ekraani korral 47).

Näide:

```
10 CLS
20 FOR RIDA = 0 TO 24
30   CUR RIDA, 2*RID
40   PRINT "*";
50 NEXT RIDA
```

DATA

Süntaks: DATA <konstantide loetelu>

Tegevus: Direktiivi DATA ei täideta, vaid ta on kasutusel konstantide hoidmiseks programmi tekstis; vt. ka direktiivide READ ja RESTORE kirjeldusi.
<konstantide loetelu> võib sisaldada arvulisi ja tekstikonstante, nende arv on piiratud ainult programmi rea pikkusega. Tekstikonstantide ümber on jutumärgid kohustuslikud ainult siis, kui need konstandid sisaldavad komasid, kooloneid või teksi alguses või lõpus paiknevaid tühikuid.

Näide: Vt. näidet direktiivi READ kohta.

DEF FN

Süntaks: DEF FN<nimi>[(<parameetrite loetelu>)]=<kirjeldus>

Tegevus: Direktiivi DEF FN ei täideta, vaid kasutatakse programmeerija loodud (arvulist tüüpi) funktsioonide kirjeldamiseks.

Funktsiooni nimi peab vastama muutuja nimele esitavatele nõuetele (algama tähega, koosnema tähtedest ja numbritest ning olema eristatav teistest nimedest esimese kahe sümboli järgi). Funktsiooni poole pöördumisel tuleb tema nime ette kirjutada tähed FN.

Parameetrite loetelu sisaldab nende muutujate nimed funktsionikirjeldusest, mis funktsiooni poole pöördumisel asendatakse mingite teiste muutujate nimedega. Parameetrite loetelus antud muutujatel (nn. formaalsetel parameetritel) ei ole seetõttu midagi ühist programmis kasutatavate muutujatega. Formaalsed parameetrid eraldatakse loetelus üksteisest komadega.

Funktsionikirjeldus peab (programmi täitmise järelkorras) paiknema eespool selle funktsiooni kasutamist. Vahetu täitmisse režiimis ei tohi funktsioone kirjeldada.

Näide:

```
10 DEF FNSEC(X) = 1/COS(X)
      .
100 F = FNSEC(Y/2)
```

DIM

Süntaks: DIM <massiivide loetelu>

Tegevus: Kirjeldab loetelus esitatud massiivid. Interpretaator saab teada massiivide dimensioonid ning re-

serveerib nende jaoks mälu.

Massiivide loetelu iga element kirjeldab ühe massiivi. Kirjeldus algab massiivi nimega, millele ümarsulgudes järgneb massiivi ülemiste rajade loetelu. Alumine raja võetakse alati võrdseks nulliga. Lubatud on kirjeldada muutuvate rajadega massiive, kasutades kirjeldamise hetkeks väärtsuse saanud muutujaid suvalise keerukusega aavaliste koosseisus.

Kui programmis pöördutakse kirjeldamata massiivi poole, siis käitub interpretaator nii, nagu oleks see massiiv kirjeldatud ülemis(t)e raja(de)ga 10.

Näide:

```
10 DIM A(15), NIMED(3,N+2)
```

FOR . . . NEXT

Süntaks: FOR <muutuja> = <av1> TO <av2> [STEP <av3>]

```
NEXT [<muutuja>]
```

Tegevus: Kasutatakse tsüklite programmeerimiseks.

Direktiivis antud (kindlasti arvutüüpi) muutuja on kasutusel tsükliloendajana. Tsüklidirektiivi FOR... esmakordsel täitmisel omistab interpretaator sellele muutujale aavalise <av1> väärtsuse, seejärel täidetakse järgmisi direktiive kuni tsükli lõpudirektiivini NEXT. Siis liidetakse tsükliloendajale aavalise <av3> väärtsus (täpsemalt see

täitmisel; <av3> puudumisel võetakse selleks +1) ning kontrollitakse, kas tsükliloendaja väärthus on saanud

- a) avalidise <av3> positiivse väärtsuse korral suuremaks avalidise <av2> väärtsusest;
- b) avalidise <av3> negatiivse väärtsuse korral väiksemaks avalidise <av2> väärtsusest..

Selle tingimuse täidetuse korral antakse juhtimine direktiivi NEXT taga paiknevale direktiivile, mitte täidetuse korral aga direktiivi FOR taga asuvale direktiivile (s.t. korratakse tsüklis sisalduvaid direktiive).

Tuleb meeles pidada, et FOR ... NEXT tsüklis paiknevad direktiivid täidetakse üks kord ka siis, kui lõpetamise tingimus on juba tsüklisse sisenemise hetkel rahuldatud.

Näide:

```
10 REM"tähestiku trükkimine ekraanile
20 CLS
30 REM"trükime tähestiku kaheteistkümnesse ritta,
40 REM  igas reas 16 sümbolit
50 FOR RIDA = 0 TO 11
60   FOR VEERG = 0 TO 15
70     KOOD = 32 + 16*RIDÄ + VEERG
80     PRINT CHR$(KOOD);
90   NEXT VEERG
100 PRINT
110 NEXT RIDA
```

GOSUB . . . RETURN

Süntaks: GOSUB <reanumber>

RETURN

Tegevus: Kasutatakse korduvalt tarvitatavate programmilõi-kude vormistamiseks alamprogrammidena (eeskätt põhiprogrammi ülevaatlikkuse ning lühiduse huvides). Alamprogrammiks loetakse programmilõiku, mille esimesele reale viatab mõni direktiiv GOSUB ning mis lõpeb direktiiviiga RETURN. Alamprogramm ei tohi juhitimist saada otse (ilma direktiivita GOSUB). Selle ärahoidmiseks võib alamprogrammi teksti paigutada direktiivide STOP või GOTO järele.

GOTO

Süntaks: GOTO <reanumber>

Tegevus: Kasutatakse direktiivide täitmise loomuliku järjekorra muutmiseks.
Direktiivis antud numbriga rida peab programmis leiduma, kuid ei pruugi sisaldada täidetavat direktiivi (sel juhul jätkub programmi täitmine esimesest suurema numbriga reast, kus paikneb täide-tav direktiiv).

IF . . . THEN ja IF . . . GOTO

Süntaks: IF <tingimus> THEN <direktiiv(id)>

või IF <tingimus> {THEN ; GOTO} <reanumber>

Tegevus: Kasutatakse programmi täitmise loomuliku järjekorra muutmiseks tingimuse järgi.

Interpretaator leiab sõna IF järel asuva aavaldisse <tingimus> väärtsuse. Kui see on tõene (s.t. nul-list erinev), jätkub programmi täitmine sõna THEN järelt, vastasel korral aga direktiivile IF ... järgnevalt reakt.

Näide:

```
10 REM"Leiame muutujate A ja B väärustest vähimma.
20 REM Kuna keelus pole konstruktsiooni ELSE, siis
30 REM" eeldame, et A on väiksem kui B; vaatame,
40 REM" kas tõesti on, ja kui pole, omistame muutu-
50 REM" jale MIN muutuja B väärtsuse.

60 MIN = A

70 IF B < A THEN MIN = B
```

INPUT

Süntaks: INPUT ["kutsung";] <muutujate loetelu>

Tegevus: Kasutajal palutakse klaviatuurilt sisestada loete-

lus mainitud muutujate väärтused.

Direktiivi täitmisel trükib interpretaator kõige-pealt ekraanile kutsungi (see on programmeerija määratud tekst, mis võib sisaldada suvalisi sümboleid), selle puudumisel aga küsimärgi ning jäab siis ootama, et programmi kasutaja sisestaks kõigi loetelus antud muutujate väärтused. Neid võib sisestada kas ühes reas, komadega eraldatult, või mitmes reas; sel juhul trükib interpretaator iga uue rea algusesse kaks küsimärki näitamaks, et kõik väärтused pole veel käes.

Kui sisestati liiga palju väärтusi või arvutüpi väärтustes oli vigu, kirjutab interpretaator ekraanile teksti "? Redo from start" ning palub kõik väärтused uuesti sisestada.

Näide:

```
10 PI = 3.1415927
20 PRINT "Mina oskan arvutada ringi pindala."
30 PRINT "Lõpetan siis, kui Sa annad raadiuse"
40 PRINT "0 või veel väiksema."
50 INPUT "Palun ütle raadius:"; R
60 IF R <= 0 THEN STOP
70 PRINT "Ringi pindala on " PI * R * R " ühikut."
80 GOTO 50
```

INVERSE

Süntaks: INVERSE

Tegevus: Määrab edaspidi väljastatavate sümbolite värviks musta valgel taustal ning edaspidi joonistatavate punktide ja joonte värviks musta. Vt. ka direktiivide CLS, PLOT, LINE ja NORMAL kirjeldusi.

LINE

Süntaks: LINE <x1>, <y1>, <x2>, <y2>

Tegevus: Joonistab ekraanile sirglõigu, mille otspunktide koordinaadid on (x1; y1) ja (x2; y2).

Joone värv on määratud direktiividega NORMAL ja INVERSE (vt.). Ekraani ülemises vasakus nurgas asuva punkti koordinaadid on (0; 0) ja alumises paremas nurgas asuva punkti koordinaadid (383; 255). Koordinaatide väärтused tohivad olla mitte-negatiivsed ja väiksemad kui 65536; x-koordinaati kasutatakse modulo 384 ning y-koordinaati modulo 256. Seetõttu ilmub näiteks üle ekraani parema serva joonistatud joon nähtavale vasakus servas ja üle ülemise serva joonistatud joon alumises servas. Võime ette kujutada, et ekraani vasak ja parem serv ning ülemine ja alumine serv on omavahel kokku ühendatud.

Näide:

10 CLS

```
20 REM"raam ümber ekraani
30 LINE 0,0, 383,0
40 LINE 0,0, 0,255
50 LINE 383,0, 383,255
60 LINE 0,255, 383,255
```

LIST

Süntaks: LIST [<reanumber>]

Tegevus: Väljastab ekraanile programmi teksti alates antud numbriga reast.

Kui <reanumber> puudub, väljastatakse kogu programmi tekst. Kui antud numbriga rida puudub, alustatakse väljastamist esimesest suurema numbriga reast.

Direktiivi LIST võib tarvitada ka programmis, kuid peab arvestama, et selle täitmise järel läheb interpretaator alati direktiivide vahetu täitmise režiimi. Lisaks tuleb tähele panna, et LIST üldil tab sisse ekraani positiivrežiimi (nagu direktiiv NORMAL).

LOAD

Süntaks: LOAD [<programmi nimi>]

Tegevus: Loeb magnetofonilt antud kuni 8-sümbolilise nimega programmi (kui nimi on õra jäetud, siis esimese,

mille lindilt leiab).

Direktiivi täitmisel kustutab interpretaator kõigepealt mälust senise programmi ning seejärel palub kasutajal käivitada magnetofon ja vajutada suvalisele klahvile. Kõigi otsimise käigus leitud programmide nimed trükkitakse ka ekraanile. Lugemise vea korral trükkitakse ekraanile teade "LOAD error". Vt. ka jaotist "Magnetofoni kasutamine" ning lisa 1.

NEW

Süntaks: NEW

Tegevus: Kustutab arvuti mälust programmi ja muutujad.
Kasutada äärmise ettevaatusega!

NORMAL

Süntaks: NORMAL

Tegevus: Määrab edaspidi väljastatavate sümbolite värviks valge (heleda) mustal (tumedal) taustal ning edaspidi joonistatavate punktide ja joonte värviks valge (see režiim kehtestatakse ka interpretaatori käivitamisel). Vt. ka direktiivide CLS, PLOT, LINE ja INVERSE kirjeldusi.

ON ... GOSUB ja ON ... GOTO

Süntaks: ON <avaldis> {GOSUB ; GOTO} <reanumbrite loetelu>

Tegevus: Kasutatakse programmi täitmise loomuliku järjekorra muutmiseks. Direktiiv ON sobib eriti kasutada neil juhtudel, kui on vaja valida mitme tegevusvariandi vahel.

Direktiivi täitmisel leiab interpretaator kõigepealt avaldise väärtsuse täisosa ning valib selle järgi loetelust järgmisena täitmisele tuleva rea numbri. Kui direktiivis antud avaldise väärthus on null või suurem kui reanumbrite arv loetelus, tulub järgmisena täitmisele vahetult järgmine direktiiv (s.t. valikut tegelikult ei toimu). Kui avaldise väärthus on suurem kui 256, trükitakse veateade "Illegal function call" ning programmi täitmine katkestatakse.

Direktiiv ON ... GOSUB erineb direktiivist ON ... GOTO selle poolest, et direktiivi RETURN toimel antakse juhtimine ON ... GOSUB järel paiknevale reale; ON ... GOTO korral mingit automaatsset tagasipöörдumist aga ei toimu.

OUT

Süntaks: OUT <pordi number>, <väärthus>

Tegevus: Protsessori KM580BM80A porti (välisseadmele) kirjutatakse parameetriga <väärthus> antud arv, mis ei tohi olla negatiivne ning peab olema väiksem kui 256. Pordi numbri määrab esimene parameeter, mis samuti tohib olla 0, ..., 255.

Direktiivi OUT tuleb tarvitada ettevaatlikult ning soovitatavalalt alles pärast arvuti ehitusega tutvumist, kuna tema juhuslik kasutamine võib viia andmete ja programmi hävimisele arvuti mälust!

Näide:

```
10 REM" valjuhäldi pordi number on 255
20 REM" teeme häält
30 OUT 255,0: GOTO 30
```

PLOT

Süntaks: PLOT <x>, <y>

Tegevus: Joonistab ekraanile punkti koordinaatidega (x; y). Punkti värv on määratud direktiividega NORMAL ja INVERSE (vt.). Ekraani ülemises vasakus nurgas asuva punkti koordinaadid on (0; 0) ja alumises paremas nurgas asuva punkti koordinaadid (383; 255). Koordinaatide väärtsused tohivad olla mitte-negatiivsed ja väiksemad kui 65536. Koordinaatide väärustest võetakse täisosa; x-koordinaati kasu-

tatakse modulo 384 ning y-koordinaati modulo 256.

Näide:

```
10 REM" öhtune taevas
20 CLS
30 X = 384 * RND(1): Y = 256 * RND(1)
40 PLOT X, Y
50 GOTO 30
```

POKE

Süntaks: POKE <aadress>, <väärthus>

Tegevus: Kirjutab esimese parameetriga määratud aadressiga baiti (mälupessa) teise parameetri väärtsuse. Aadress tohib olla 0 kuni 65535 ning väärthus 0 kuni 255.

Direktiivi POKE tuleb tarvitada ettevaatlikult ning soovitatavalalt alles pärast arvuti ehitusega tutvumist, kuna tema juhuslik kasutamine võib viia andmete ja programmi hävimisele arvuti mälust!

PRINT

Süntaks: PRINT [<avaldiste loetelu>]

Tegevus: Ekraanile trükitakse loetelus esitatud aavaldiste väärtsused.

Kui aavaldiste loetelu puudub, tehakse ekraanil

reavahetus, vastasel korral aga arvutatakse ning trükitakse kõigi loetelus antud avaldiste väärtsed.

Loetelu elemendid võivad olla üksteisest eraldatud kas koma või semikooloniga. Koma puhul tabuleeritakse järgmise avaldise väärthus, s.t. trükitakse kõigepealt tühikuid kuni veeruni \emptyset , 16, 32 või 48 (esimeseni neist) ning alles seejärel väljastatakse avaldise väärthus. Semikooloniga eraldatud avaldiste väärtsused trükitakse vahetult teineteise järel (semikooloni võib ka ära jäätta, kui interptaator on suuteline üles leidma eelmise avaldise lõpu ja järgmiste alguse; näiteks ei tohi semikoolon puududa kahe muutuja nimede vahelt). Pärast viimase avaldise väärtsuse trükkimist

- a) tehakse reavahetus, kui loetelu ei lõpe koma ega semikooloniga,
- b) trükitakse tühikuid kuni järgmise tabulatsiooniveeruni, kui loetelu lõpeb komaga,
- c) jäädakse samasse ritta ja samasse veergu, kui loetelu lõpeb semikooloniga.

Tekstitüüpi avaldised trükitakse täpselt nii mitmele täheruumile, kui palju nad vajavad. Arvude järel väljastatakse üks tühik ning positiivsete arvude ette tühik, negatiivsete ette aga miinusmärk. Absoluutväärtsuselt suuremad kui 10^{-7} ja väiksemad kui 10^{-7} arvud väljastatakse üldjuhul poollogaritmilisel kujul.

READ

Süntaks: READ <muutujate loetelu>

Tegevus: Loeb direktiivi(de)ga DATA kirjeldatud väärтused ning omistab need loetelus esitatud muutujatele.

Muutujate väärтused võivad olla esitatud ühe või mitme direktiivi DATA abil. Programmi töö alguses paneb interpretaator väärтuste lugemise viida kõige väiksema numbriga reas asuvale direktiivile DATA ning edaspidi nihutab viita edasi vastavalt sellele, mitu väärтust luges sisse järjekordne direktiiv READ. Vajaduse korral saab viita algseisu seada direktiiviga RESTORE (vt.).

Arvuliste muutujate väärтused peavad olema kirjeldatud arvuliste konstantide esitamise reeglite järgi. Kui direktiiv READ avastab arvulise muutuja väärтuse lugemisel vea, trükib interpretaator veateate "Syntax error" ning katkestab programmi töö. Kui väärтusi on (üle kõigi direktiivide DATA) vähem kui nõuab direktiiv READ, trükib interpretaator veateate "Out of DATA" ning katkestab programmi töö. Liigseid väärтusi ignoreeritakse.

Näide:

```
10 DATA 34, 46.6, 23
20 DIM D(5)
30 FOR I = 1 TO 5
40 READ D(I)
50 PRINT D(I),
```

60 NEXT I

70 DATA -12, 3E7

REM

Süntaks: REM <suvaline sümbolijada>

Tegevus: Direktiivi REM ei täideta, samuti ei kirjelda ta mingeid objekte, vaid on ette nähtud selgitavate märkuste lisamiseks programmi teksti.

Eesti tähestiku ja kirillitsa tähtede kasutamise korral on soovitav sõna REM järelle kirjutada jutumärgid.

Näide:

```
100 REM peame meeles, et kommentaariks loetakse
110 REM"kogu tekst sõnast REM kuni rea lõpuni
```

RENUMBER

Süntaks: RENUMBER [<samm>]

Tegevus: Nummerdab programmi read ümber. Esimese rea number jäääb muutmata, iga järgmisse rea number saab olema eelmisest .avaldise <samm> väärtsuse võrra suurem. Vaikimisi võetakse sammuks 10. Direktiiv RENUMBER korrigeerib ka suunamistes (GOTO, GOSUB) kasutataid reanumbreid.

RESTORE

Süntaks: RESTORE

Tegevus: Paneb (direktiivi READ jacks kasutatava) viida esimese direktiivi DATA algusse, võimaldades nii programmis kirjeldatud andmehulka korduvalt lugeda. Vt. ka direktiivide DATA ja READ kirjeldusi.

Näide:

```
10 READ A, B, C
20 PRINT A, B, C
30 RESTORE
40 READ D, E, F
50 PRINT D, E, F
60 DATA 34, 67, 23
```

RUN

Süntaks: RUN [<reanumber>]

Tegevus: Alustab mälus paikneva programmi täitmist (kui reanumbrit pole antud, siis vähima numbriga reast). Kõigi muutujate eelmised väärtused ning kõik massiivid hävivad; uuteks algväärtusteks on arvutüüpi muutujatel Ø ning tekstitüüpi muutujatel "" (tühisöne).

SAVE

Süntaks: **SAVE <programmi nimi>**

Tegevus: Kirjutab magnetofonile antud (kuni 8-sümbolilise) nimega programmi. Direktiivi täitmisel palub kasutajal käivitada magnetofon ja vajutada suvalisele klahvile. Kasutaja peab ise olema lindil leidnud vaba koha. Vt. ka jaotist "Magnetofoni kasutamine" ning lisa 1.

STOP

Süntaks: **STOP**

Tegevus: Peatab programmi täitmise ning viib interpretaatori direktiivide vahetu täitmise režiimi. Ekraanile trükitakse teade "Break in xxxxx" (kus xxxxx on direktiivi STOP sisaldatava rea number). Programmi täitmist saab vajaduse korral jätkata direktiiviga **CONT** (vt.).

VERIFY

Süntaks: **VERIFY [<programmi nimi>]**

Tegevus: Loeb magnetofonilt antud kuni 8-sümbolilise nimega programmi (kui nimi on ära jäetud, siis esimese,

mille lindilt leiab) ning võrdleb seda mälus paiknevaa programmiga.

Direktiivi täitmisel palub kasutajal käivitada magnetofon ja vajutada suvalisele klahvile. Köigi otsimise käigus leitud programmide nimed trükitakse ka ekraanile. Kui lindilt loetud programm ei vasta täpselt mälus paiknevale, trükitakse veateade "VERIFY error". Mälus paiknev programm jäääb muutmata. Vt. ka jaotist "Magnetofoni kasutamine" ning lisa 1.

Funktsioonid

ABS

Süntaks: ABS(<arvavaldis>)

Funktsiooni väärtsuseks on tema argumendi absoluutväärtus.

ASC

Süntaks: ASC(<tekstiavaldis>)

Funktsiooni väärtsuseks on tema argumendi esimese sümboli arvuline kood.

ATN

Süntaks: ATN(<arvavaldis>)

Funktsiooni väärtsuseks on tema argumendi arkus-tangens.

CHR\$

Süntaks: CHR\$(<arvavaldis>)

Funktsiooni väärtsuseks on ühesümboliline tekstivaldis. Sümboli koodi annab funktsiooni argument, mis tohib seetõttu olla 0, ..., 255.

COS

Süntaks: COS(<arvavaldis>)

Funktsiooni väärtsuseks on argumendi koosinus.

EXP

Süntaks: EXP(<arvavaldis>)

Funktsooni väärtsuseks on arvu e argumendiga määratud aste. Argument ei tohi olla suurem kui 87.3365.

FRE

Süntaks: FRE(<arvavaldis>)

või FRE(<tekstiavaldis>)

Funktsooni väärtsuseks on vaba mälu arv baitides (esimesel juhul käib see programmile ja arvutüüpi muutujatele eraldatud mäluosa kohta, teisel juhul aga tekstmuutujatele eraldatud mäluosa kohta). Argumendi väärtsuse ei ole mingit tähtsust.

INP

Süntaks: INP(<arvavaldis>)

Funktsooni väärtsuseks on protsessori KM580BM80A pordist (välisseadmel) loetud väärthus. Pordi numbriga määrab funktsiooni argument, mis tohib olla \emptyset , ..., 255.

Funktsooni INP tuleb tarvitada ettevaatlikult ning soovitatavalalt alles pärast arvuti ehitusega tutvumist, kuna tema juhuslik kasutamine võib viia andmete ja programmi hävimisele arvuti mälust!

INT

Süntaks: INT(<arvavaldis>)

Funktsooni väärtsuseks on tema argumendi täisosa,

s.t. suurim täisarv, mis ei ületa argumendi väärust. Näiteks $\text{INT}(3.7) = 3$, kuid $\text{INT}(-3.7) = -4$.

LEFT\$

Süntaks: `LEFT$(<tekstiavaldis>, <arvavaldis>)`

Funktsiooni vääruseks on tema esimese argumendi "vasakpoolne" (esimesest sümbolist algav) alamsõne, mille pikkuse määrab teise argumendi väärus. Sümbolid on tekstiavaldistes nummerdatud ühest alates.

LEN

Süntaks: `LEN(<tekstiavaldis>)`

Funktsiooni vääruseks on tema argumendi pikkus sümbolites.

LOG

Süntaks: `LOG(<arvavaldis>)`

Funktsiooni vääruseks on tema argumendi naturaalarithm. Argument peab olema positiivne.

MIDS\$

Süntaks: `MID$(<tekstiavaldis>, <arvav1>, <arvav2>)`

Funktsiooni vääruseks on tema esimese argumendi alamsõne, mille alguse määrab teise argumendi väärus ja mille pikkuse (sümbolites) määrab kolmanda argumendi väärus. Vt. ka funktsioone LEFT\$ ja RIGHT\$.

PEEK

Süntaks: PEEK(<arvavaldis>)

Funktsooni väärтuseks on arvuti mälust loetud baidi (mälupesa) sisu. Baidi aadressi määrab funktsiooni argument, mis võib olla \emptyset , ..., 65535.

POS

Süntaks: POS(<arvavaldis>)

Funktsooni väärтuseks on kursori asukohaveeru number (vasakpoolseim veerg on number 1). Argumendi väärтusel ei ole mingit tähtsust.

RIGHT\$

Süntaks: RIGHT\$(<tekstiavaldis>, <arvavaldis>)

Funktsooni väärтuseks on tema esimese argumendi "parempoolne" (viimase sümboliga lõppev) alamsõne, mille pikkuse määrab teise argumendi väärтus.

Sümbolid on tekstiavaldistes nummerdatud ühest alates.

RND

Süntaks: RND(<arvavaldis>)

Funktsooni väärтuseks on pseudojuhuslik arv pool-lõigult $[\emptyset; 1]$.

SGN

Süntaks: SGN(<arvavaldis>)

Funktsooni väärтuseks on argumendi märk, mis on kodeeritud järgmiselt:

SGN(x) = -1, kui x < 0,
SGN(x) = 0, kui x = 0,
SGN(x) = +1, kui x > 0.

SIN

Süntaks: SIN(<arvavaldis>)

Funktsiooni väärtsuseks on tema argumendi siinus.

SPC

Süntaks: SPC(<arvavaldis>)

Ekraanile väljastatakse argumendiga määratud arv tühikuid (0 kuni 255). Peale direktiivi PRINT ei tohi funktsiooni SPC kuskil mujal tarvitada.

SQR

Süntaks: SQR(<arvavaldis>)

Funktsiooni väärtsuseks on tema argumendi ruutjuur (argument ei tohi muidugi olla negatiivne).

STR\$

Süntaks: STR\$(<arvavaldis>)

Funktsiooni väärtsuseks on argumendi värtus teisendatuna tekstivaldiseks (argumendi värtuse "kirjapilt"). Vt. ka funktsiooni VAL kirjeldust.

TAB

Süntaks: TAB(<arvavaldis>)

Viib kursori ekraanil argumendi värtusega määratud veergu (vasakpoolseima veeru number on 1). Kui

kursor on juba määratud veerus või sellest paremal, liigub ta järgmise rea vastavasse veergu. Funktsiooni TAB tohib kasutada ainult direktiivis PRINT.

TAN

Süntaks: TAN(<arvavaldis>)

Funktsiooni väärtsuseks on tema argumendi tangens.

USR

Süntaks: USR(<arvavaldis>)

Funktsioon pöördub protsessori KM580BM80A käskudes kirjutatud alamprogrammi poole. Programmi algus- aadressi määrab argumendi väärthus, mis tohib olla 0 kuni 65535. Tagasipöördumine alamprogrammist peab toimuma protsessori käsuga RET. Funktsiooni väärtsuseks on protsessori A-registri (akumulaatori) väärthus käsu RET täitmise hetkel. Alamprogramm peab säilitama protsessori kõigi ülejäänud registrite väärtsused.

Funktsiooni USR tuleb tarvitada ettevaatlikult ning soovitavalt alles pärast arvuti ehitusega tutvumist, kuna tema juhuslik kasutamine võib viia andmete ja programmi hävitamisele arvuti mälust!

VAL

Süntaks: VAL(<tekstiavaldis>)

Funktsiooni väärtsuseks on tema argumendi arvuks teisendamisel saadud väärthus (kui argumendi esime-

ne sümbol pole number või pluss- või miinusmärk, on teisendamise tulemuseks 0). Tühikud jäetakse teisendamisel vahel. Vt. ka funktsiooni STR\$ kirjeldust.

Masinkoodis kirjutatud alamprogrammide kasutamine

Programmeerimiskeele BASIC kõige suuremaks puuduseks tuleb pidada programmeerimiskeele töökiirust (BASIC-programm töötab masinkoodis kirjutatud programmist ligikaudu 100 korda aeglasemalt). Mõnel juhul (näiteks välisseadmete juhtimisel) on vaja oluliselt suuremat kiirust. Lisaks sellele ei saa keele BASIC vahenditega kõiki arvuti võimalusi täielikult ära kasutada, küll saab seda aga teha masinkoodis (assembleris) programmeerides. Muidugi on kõrgema taseme programmeerimiskeeltes märksa kergem programmeerida kui assembleris, seetõttu on mõistlik peaaegu kogu programm kirjutada keeles BASIC ning ainult häavajalikud osad programmeerida assembleris või lausa masinkoodis.

Arvutil Tartu on masinkoodis kirjutatud programmeerimiseks olemas keele BASIC direktiiv POKE ning funktsioonid PEEK ja USR. Nende kirjeldused on antud vastavalt direktiivide ja funktsioonide kirjelduste osas; siinkohal illustreerime mõne näitega nende kasutusvõimalusi. Märgime, et mitmesuguste kasulike alamprogrammide aadressid on antud jaotises "Monitori alamprogrammid".

Pöörduda alamprogrammi poole saame funktsiooni USR abil. Funktsiooni argumendiks peab olema väljakutsutava alamprogrammi aadress. Alamprogramm peab säilitama registrite BC, DE, HL ja SP väärтused ning lõppema käsuga RET. Funktsiooni väärтuseks võetakse protsessori A-registri (akumulaatori) sisu käsu RET täitmise hetkel ja see teisendatakse täisar-

vuks vahemikust 0 kuni 255.

Olgu programmis näiteks vaja oodata ühe (suvalise) klahvi vajutust. Selle saab realiseerida direktiiviga

100 QQ = USR(51187)

(aadressil 51187 ehk C7F3h paikneb Monitori alamprogramm, mis vilgutab ekraanil cursorit, ootab klahvivajutust ning jätab vajutatud klahvi koodi akumulaatorisse). Funktsiooni USR töö tulemusena saab muutuja QQ uueks väärtsuseks alamprogrammi poolt akumulaatorisse salvestatud arv. Seejuures peame arvestama, et muutuja QQ eelmine väärus kaob.

Sageli on tarvis saada alamprogrammilt rohkem kui ühe baidi väärust, aga ka pöördumisel mingiid parameetreid ette anda. Selleks peame tarvitama direktiivi POK (parameetrite ette-andmiseks) ning funktsiooni PEEK (tulemuste lugemiseks). Nii sisend- kui ka väljundparameetrite jooks peab alamprogrammis olema reserveeritud vajaliku pikkusega mäluosa. Kahebaidiste väärustete ülekandmisel peame meeles pidama, et masinkoodis kirjutatud programmid salvestavad neid harilikult "pööratud" kujul. Näiteks neljakohaline kuuteistkümnendarv abcd kujutub mälus kahe järjestikuse baidina

CD AB

(aadresside kasvamise järjekorras.)

Paiknegu näiteks mälus alates aadressist A000h programm, mis leiab ühega võrduvate bittide arvu etteantud kahebaidises

(16-bitises) arvus, mis paikneb baditides A100h ja A101h.
Selle alamprogrammi pool saame pöörduda näiteks järgmiselt:

```
100 REM" olgu vaja lugeda kokku ühed muutuja RT vääruses
110 COUNT = 10$4096: REM alamprogrammi aadress
120 XL = 10$4096+256: REM parameetri aadress
130 QH = INT(RT/256): REM eraldame "vanema" baidi
140 QL = RT - 256$QH: REM eraldame "noorema" baidi
150 REM" anname ette parameetri vääruse
160 POKE XL,QL: POKE XL+1,QH
170 REM" pöördume alamprogrammi ja saame ühtede arvu
180 REM muutujasse YH
190 YH = USR(COUNT)
```

Kasutaja enda kirjutatud alamprogrammide paigutamiseks võib kasutada mälupiirkonda aadressidega B000h kuni BFFFh. Lühemad alamprogrammid saame kirjutada BASIC-programmi koosseisu direktiivide DATA abil. Siis peab BASIC-programmi alguses paiknema lõik, mis kirjutab masinkoodis alamprogrammi tema jooks ettenähtud kohale arvuti mälus.

Olgu meil näiteks tarvis alamprogrammi, mis trükitib üheba-diste arvude kuueteistkümnendväärusi. Monitoris on selline alamprogramm PRBYTE olemas (vt. lk. 77), kuid selle sisend-parameeter peab olema antud protsessori akumulaatoris. Me võime kirjutada programmilõigu, mis kannab sisendparameetri mälust akumulaatorisse ja siis pöördub programmi PRBYTE poolle. Paigutame selle "pealisehituse" aadressile B001h, sisendparameeter olgu baidis B000h. Meie assemblerprogramm

näeb siis välja näiteks niisugune (esitame siin osa transleerimisprotokollist, kus on näha ka käskude kuueteistkümnendkoodid):

```
F2D3      prbyte    equ  0f2d3h ;Monitori alamprogramm
;
B000  00      parm:     ds   1      ;ruum parameetri jaoks
B001  3A 00 B0  bytepr:   lda  parm   ;parameeter akumulaatorisse
B004  C3 D3 F2      jmp  prbyte ;trükkima
```

Oma põhiprogrammi algusesesse kirjutame järgmise programmi-lõigu:

```
100 REM"assemblerprogrammi kirjutamine mällu
110 REM programm algab B001h = 45057
120 REM parameeter baiti B000h = 45056
130 PARM = 45056: BYTEPR = PARM+1
140 REM"loeme DATA'st ja kirjutame mällu
150 REM" programmi, olles ta käsitsi kümnnendsüsteemi
160 REM  teisendanud
170 FOR I = 0 TO 5
180  READ Q
190  POKE BYTEPR+I, Q
200 NEXT I
210 DATA 58, 0, 176, 195, 211, 242
220 REM  3A 00 B0 C3 D3 F2
```

Muutuja xx väärтuse trükkimiseks kuueteistkümnendsüsteemis peame programmis kirjutama järgmised direktiivid:

500 POKE PARM, xx

510 NUL = USR(BYTEPR)

520 REM" muutuja NUL väärthus läheb siin kaduma

Interpretaatori mälujaotus

Interpretaator kasutab baite 0000h kuni 02FFh oma tööpesade ja puhvrite tarbeks. Kasutajale võivad siit huvi pakkuda baidid 0245h ja 0246h, kus hoitakse viita BASIC-programmi lõpule (ehk lihtmuutujatele eraldatud mäluosa algusele), 0247h ja 0248h, kus asub viit lihtmuutujate mäluosa lõpule (ehk massiividele eraldatud mälu algusele) ning 0249h ja 024Ah, kus paikneb viit massiivide mälu lõpule ehk vaba mälu algusele.

BASIC-programm salvestatakse mällu alates aadressist 0300h, tema tekstile järgnevad lihtmuutujad ja seejärel massiivid. Interpretaatori jooks lõpeb operatiivmälu baidiga AFFFh, millele eelnevat 5120 baiti kasutatakse tekstimuutujate väärustuse hoidmiseks.

Baidid B000h kuni BFFFh jäavad interpretaatori poolt kasutamata ning seal võib näiteks hoida masinkoodis kirjutatud alamprogramme. Alates aadressist C000h on operatiivmälu reserveeritud Monitori vajadusteks.

Interpretaatori teated

Töö käigus trükitib keele BASIC interpretaator ekraanile mitmesuguseid teateid, millest enamik kirjeldavad programmi täitmisel tekkinud veasituatsioone. Ainult kaks teadet ilmuvad ka programmi korraliku töötamise ajal. Kirjeldamegi neid kõigepaalt.

OK

Teate OK trükitib interpretaator, kui ta läheb programmi täitmise režiimist üle direktiivide vahetu täitmise režiimi.

Break

Selle teate väljastab interpretaator, minnes direktiivide vahetu täitmise režiimi kas direktiiv STOP toimel või klahvide ctrl-C vajutamise avastamisel. Kui katkestus toimus programmi täitmise režiimis, trükitakse teate järele fraas

in xxxxxxx

kus xxxxxx on programmi selle rea number, mille töötlemisel katkestus toimus (see kehtib ka järgnevalt kirjeldatavate teadete kohta).

Loetleme nüüd veateated ja selgitame lühidalt nende tähen-dusi.

NEXT without FOR

Täitmisele tuli direktiiv NEXT <var>, millele ei olnud eel-nenud direktiivi FOR <var>=

Syntax error

Interpretaator ei suuda rea sisust aru saada (tõenäoliselt on rea sisestamisel tehtud vigu).

RETURN without GOSUB

Täitmisele tuli direktiiv RETURN, millele ei olnud eelnenud direktiivi GOSUB.

Out of DATA

Direktiiviga READ üritati lugeda rohkem väärtsusi, kui sisal-dus direktiivides DATA.

Illegal value

Direktiivi või funksiooni parameetril on lubamatu väärthus (näiteks negatiivne argument funksioonidel SQR või LOG, suurem kui ühebaidine väärthus direktiivi POKE teisel para-meetril, suurem kui kahebaidine väärthus direktiivi POKE esi-mesel parameetril, funksiooni PEEK argumendil vms.).

Overflow

Aritmeetilise tehte teostamisel või elementaarfunktsiooni arvutamisel saadi tulemuseks suurem arv kui tohib olla muu-taja vääruseks (absoluutväärtuselt suurem kui 10^{38}).

Out of memory

Interpretaatorile ei jätku mälu: programm on liiga pikk (või kasutab liiga suuri massiive).

Undefined statement

Direktiivis RUN, GOTO või GOSUB esines olematu rea number.

Märkus: vahetu täitmise režiimis on arusaamatuste ärahoidmiseks soovitav sõna RUN järele lisada tühik.

Bad subscript

Massiivi elemendi indeksitel on lubamatud väärtsused või pole massiivi dimensioon vastavuses kirjeldusega.

ReDIM'd array

Massiivi katsutakse teistkordset kirjeldada või on massiivi elemente kasutatud enne selle massiivi kirjeldamist.

Division by zero

Avaldise väärtsuse arvutamisel toimus jagamine nulliga.

Illegal direct

Seda direktiivi ei tohtinud tarvitada vahetu täitmise režiimis.

Type mismatch

Avaldises üritatakse seepäri (tüübiteisendusfunktsioone kasutamata) tarvitada arvu- ja tekstitüüpi muutujaid või konstante.

String too long

Tekstitüüpi avaldise väärthus tuli pikem kui 255 sümbolit.

Can't CONTinue

Direktiiviga CONT katsuti taaskäivitada programmi, mida kat-kestuse ajal parandati või mis lõpetas oma töö veateatega.

Undefined function

Esines pöördumine kasutaja funktsiooni poole, mida ei ole kirjeldatud (või on muutuja nimeks valitud FN).

Lõpuks kirjeldame veel direktiivi INPUT täitmisel tekkida võivaid teateid, mis programmi täitmist ei katesta.

? Reenter

Sisestamisel tehti viga (näiteks ei olnud arvutüüpi väärthusid kirjutatud korrektsest). Kogu direktiivis INPUT antud muutujate loetelu tuleb uuesti väärustada.

Extra ignored

Sisestati rohkem väärtsusi kui oli kirjas direktiivis INPUT. Ulejäänud väärtsusi ignoreeritakse.

??

Sisestati vähem väärtsusi kui oli kirjas direktiivis INPUT. Selle teate järel tuleb sisestada puunduvad väärtsused.

Monitor

Monitoriks nimetatakse arvuti juhtprogrammi, mis (tavaliselt) paikneb püsimalus ning võtab enda hooleks arvuti välisseadmete juhtimisega seotud ülesanded. Tihti kuuluvad monitori koosseisu ka masinkoodis kirjutatud programme silumisvahendid.

Järgnevalt vaatlemegi arvuti Tartu monitori kasutamise võimalusi. Kuna monitor on arvutis köige "sisemise" taseme programm (teised programmid saavad näiteks klaviatuuri ning ekraani kasutada monitori alamprogrammide poole pöördumise abil) ning seetõttu tihedalt seotud arvuti aparaaturse osa ülesehitusega, siis on käesolevas peatükis sobilik ära tuua ka arvuti Tartu mäluaotus.

Monitori direktiivid

Monitor pakub kasutajale järgmisi võimalusi:

- mälu sisu trükkimine;
- mälu sisu muutmine;
- välisseadmete juhtimine;
- programmide käivitamine;
- mälu sisu salvestamine magnetofonile ning tagasi-lugemine.

Monitori direktiivide keel on (püsimälu mahu piiratuse tõttu) üsnagi primitiivne. Direktiiv peab tervikuna paiknema ühel real, mille pikkus võib olla kuni 191 sümbolit. Direktiiv koosneb üldjuhul kolmest osast:

aadress;
ühesümboliline direktiivi kood;
direktiivi lõpp.

Kui monitor on direktiivi ära tundnud, ignoreerib ta ülejäävud sümboleid kuni rea lõpuni.

Direktiivides kasutatavad arvud peavad kõik olema esitatud kuueteistkümnendsüsteemis. Monitor kasutab (arvu semantikast olenevalt) tema kahte või nelja viimast numbrit ning ignoreerib liigseid esimesi numbreid. Neljakohalistena sisestatakse mäluaadressid, kahekohalistena aga välisseadmete (portide) numbrid ehk aadressid ning ka operatiivmällu kirjutatavad väärtsused.

Monitori direktiivide kirjeldamisel kasutame järgmist tähisustusi. Nurksulgudesse [ja] paigutatud konstruktsiooniühikud võivad puududa või esineda üks kord, loogelistes sulgudes { ja } paiknevad konstruktsiooniühikud võivad esineda üks või mitu korda. Noolsulgudesse < ja > paigutatud konstruktsiooniühikud tuleb asendada arvu või sümbolijadaga (lubatud asendused on loetletud iga direktiivi kirjelduses). Märgime veel, et suurtähtede asemel võib kasutada ka väike tähti.

Kui monitor ei suuda direktiivist aru saada, annab ta heli-signaali ning jäab ootama järgmist (ning loodetavasti korrektelt sisestatud) direktiivi.

Mälu sisu trükkimine

Kuueteistkümnendsüsteemi arvudena ning tähestiku ASCII sümboleina lubab arvuti mälu sisu uurida direktiiv Print, mille üldkuju on järgmine:

```
[<aadress>] [ P [<baitide arv>]]
```

Selle direktiivi toimel trükitakse ekraanile arvuti mälu sisu, alates direktiivis antud aadressist. Kui <baitide arv> on direktiivist ära jäetud, trükitakse ekraanile 128 baidi väärтused. Kui aga puudub ka täht P, trükitakse üheainsa baidi sisu. Märgime siin veel, et kui väljastatava mäluosa pikkuseks anda null, trükitakse 65536 baidi väärтused.

Monitor jätab meelde viimasena väljastatud baidi aadressi ning kui direktiivist Print (või ka direktiivist List; vt. allpool) on ära jäetud algusaadress, jätkatakse mälu sisu trükkimist kohast, kus see viimatisidetud direktiivi Print või List juures pooleli jäi.

Näide:

```
235p12                                (selle sisestame meie)
0235: 10 AB 10  .+.
0238: 0C 43 4C D3 46 4F D2 4E  .CLSFORN
0240: 45 58 D4 44 42 54 C1  EXTDATA
p9                                (järgmine direktiiv)
0247: 49  I
0248: 4E 50 55 D4 44 49 CD 52  NPUTDIMR
                                    (vajutasime ainult klahvi RETURN)
0250: 45  E
                                    (jällegi RETURN)
0251: 41  A
```

Direktiivi List, mille üldkuju on

[<aadress>] L [<käskude arv>]

saame mälu sisu vaadata kuueteistkümnendarvudena ning prot-
sessori KM580BM80A käskudena. Kui <käskude arv> on ära jäe-
tud, väljastatakse 21 käsku (paneme tähele, et see on üldju-
hul rohkem kui 21 baiti). Käskude arv ei saa olla suurem kui
256 (meie antud parameetrit kasutatakse modulo 256; nulli
korral trükitakse 256 käsku). Aadressi puudumise korral jäät-
kab ka direktiivi List väljastamist sellest baidist, mis eel-
misse direktiivi Print või List täitmisel veel väljastamata
jääi.

Näide:

```
e00015
E000  CALL  $F9A0      CD A0 F9
E003  LXI  B,$E5C1  01 C1 E5
E006  LXI  D,$E623  11 23 E6
E009  LXI  H,$B200  21 00 B2
E00C  CALL  $F9F3      CD F3 F9
13
E00F  XRA  A          AF
E010  STA  $0004      32 04 00
E013  MVI  A,$40      3E 40
```

Mäluosade võrdlemine

Kahe mäluosa võrdlemiseks kasutatakse direktiivi Verify, mille üldkuju on järgmine:

```
<algusaadress1> V <algusaadress2>,<lõppaadress2>
```

Direktiivis on antud "teise" mäluosa algus- ja lõppaadressid ning "esimese" mäluosa algusaadress; kuna võrrelda saab siinult ühepiikkuste mäluosade sisusid, on monitoril "esimese" mäluosa lõppaadressi võimalik arvutada. Märgime siinkohal, et mõisted "esimene" ja "teine" ei sea mingeid piiranguid selle kohta, kumb võrreldavatest mäluosadest on tegelikult väiksemate aadressidega; nad võivad isegi kattuda.

Mäluosasid vörreldakse bait-baidilt ning iga leitud lahkumi-nek trükitakse ekraanile järgmisel kujul:

- 1) aadress "esimesest" mälupiirkonnast;
- 2) selle aadressiga baidi sisu;
- 3) vastava baidi sisu "teisest" mälupiirkonnast.

Näide:

100p10

0100: 40 41 42 43 FF FF 23 E3 @ABC..#c

0108: 7E E3 C3 06 01 C2 30 04 ~cC..B0.

212p10

0212: 40 41 42 43 FF 44 @ABC.D

0218: 23 E3 7E E3 C3 06 01 C2 #c~cC..B

0220: 39 04 9.

100v212,221

0105: FF (44)

010E: 30 (39)

Märgime veel, et direktiivi Verify abil saab kontrollida, kas mingi mäluosa on täidetud ühesuguste arvudega (ühe- või mitmebaidistega). Näiteks direktiiv

100v101,1ff

kontrollib, kas kõigis baitides aadressidega 0100 kuni 01FF sisaldub ühesugune värtus.

Mälusisu muutmine

Arvuti operatiivmälusisu muutmiseks on monitoris kaks direktiivi. Kuueteistkümnendarve saame sisestada direktiiviga

[<aadress>] : [<väärustus>]

(kui sisestatavaid arve on üle ühe, peavad nad üksteisest olema eraldatud tühikutega). Tähestiku ASCII sümboleid saame aga mällu paigutada direktiivi

[<aadress>] T <tekst>

abil, kus <tekst> võib sisaldaada kõiki klaviatuurilt saada-olevaid sümboleid peale juhtsümbolite. Kui neis direktiivides on sisestamise algusaadress ära jäetud, kasutab monitor viimasena täidetud direktiivist Print või List meelete jätetud aadressi. Korraga sisestatavate väärustete hulk on piiratud ainult Monitori sisendrea pikkusega.

Näide:

```
100:40 41 42 43
100p10
0100: . 40 41 42 43 FF FF 23 E3  @ABC..#c
0100: 7E E3 C3 06 01 C2 30 04  ~cC..B0.
                                              (RETURN)
0110: 23  #
:56
```

(RETURN)

0110: 56 V

5000tABCDEFGHIJKLMNPQRSTUVWXYZ

5000p20

5000: 41 42 43 44 45 46 47 48 ABCDEFGH

5008: 49 4A 4B 4C 4D 4E 4F 50 IJKLMNOP

5010: 51 52 53 54 55 56 57 58 QRSTUVWX

5018: 59 5A FE 50 7D CA 08 50 YZ~P;J.P

Mäluosa kopeerimine

Mäluosa sisu kopeerimiseks teise mäluossa on Monitoris ette nähtud direktiiv Move üldkujuga

<algusaadressi> M <algusaadress2>,<lõppaadress2>

See direktiiv loeb ühekaupa kõigi baitide sisu "teisest" mäluviirkonnast (s.t. alates aadressist <algusaadress2> kuni aadressini <lõppaadress2>) ja kirjutab selle "esimese" mäluosa vastavatesse baitidesse. Direktiiv lubab ka mäluosade kattumist, seega saab tema abil etteantud mäluosa täita etteantud (ühe- või mitmebaidise) väärtsusega.

Näide:

3000p10

3000: 0D 35 20 4E 4F 52 4D 41 .5 NORMA

3008: 4C 3A 43 4C 53 0D 37 20 L:CLS.7

100p10

0108: 7E E3 C3 06 01 C2 30 04 ~cC..B0.
30000m100,107
30000p10
3000: 40 41 42 43 FF FF 23 E3 @ABC..#c
3008: 4C 3A 43 4C 53 00 37 20 L:CLS.7
3000:50 51
3002m3000,3007
30000p10
3000: 50 51 50 51 50 51 50 51 PQPQPQPQ
3008: 50 51 43 4C 53 00 37 20 PQCLS.7

Välisseadmete juhtimine

Arvuti välisseadmete juhtimiseks sisalduvad Monitoris direk-
tiivid In ja Out. Direktiivil In on kuju

[<pordi number>] I

ning ta trükitib ekraanile antud pordi numbriga ning sealt loe-
tud väärtsuse. Direktiivil Out on kuju

[<pordi number>] O <väärthus>

ning ta kirjutab antud väärtsuse antud porti (trükkides mõle-
mad arvud ka ekraanile). Nii pordi number kui ka loetud väi-
kirjutatav väärthus on kahekohalised kuuteistkümnendarvud.

Kui pordi number on ära jäetud, kasutavad direktiivid eel-mist numbrit (sisend- ja väljundpordi numbrid peetakse mee-les eraldi). Tuleb ainult tähele panna, et arvuti käivitami-se järel on portide numbritel, nagu ka vaikimisi tarvitata-val mäluaadressil, määramata väärthus.

Programmide käivitamine

Arvuti operatiiv- või püsimälus paikneva programmi käivita-miseks kasutatakse direktiivi Go üldku juga

[<aadress>] G

Kui käivitatav programm oma töö lõpuks säilitab magasini viida sisu ning lõpeb käsuga RET, antakse juhtimine Monito-riile tagasi.

Ekraanidirektiivid

Monitoris sisalduvad ka mõned ekraanipilti juhtivad direk-tiivid, millest igaüks koosneb ainult ühesümbolilisest koo-dist.

Direktiiv Z puastab ekraani ning viib kurSORI ülemisse va-sakusse nurka.

Direktiiv N lülitab sisse ekraani positiivrežiimi (trükitak-

se heledaid sümboleid tumedal taustal).

Direktiiv X lülitab sisse ekraani negatiivrežiimi (trükitakse tumedaid sümboleid heledal taustal).

Direktiiv 1 lülitab ümber rea pikkust (48 või 64 sümbolit) ning kirjutab ekraani vastavalt uuele formaadile ümber. Monitori käivitamisel valitakse rea pikkuseks 48 sümbolit.

Magnetofoni kasutamine

Mälu sisu magnetofonile salvestamiseks on Monitoris direktiiv W üldkujuga

<algusaadress> W <lõppaadress> <faili nimi>

Selle direktiivi toimel kirjutatakse lindile mäluosa sisu, mis algab aadressilt <algusaadress> ning lõpeb aadressil <lõppaadress> - 1. Kirjutatud faili nimeks saab parameetriga <faili nimi> antud kuni 8-sümboliline tekst. Parameetri <algusaadress> väärtus kirjutatakse koos andmetega lindile. Direktiivi sisestamisel peab faili nimele eelnema tühik. Faili lõppu kirjutatakse kahesekundise pausi järel veel failide lõpu tunnus märkimaks vaba koha algust lindil. Uue faili lisamisel tuleb seetõttu jälgida, et eelmise lõputunnus kirjutatakse üle.

Magnetofonilt lugemiseks kasutatakse direktiivi R:

[<algusaadress>] R [<faili nimi>]

Selle direktiivi toimel loetakse magnetofonilt fail nimega <faili nimi> ning paigutatakse mällu alates aadressist <algusaadress>. Vaikimisi võetakse algusaadressiks aga lindile kirjutatud aadress. Parameetri <faili nimi> puudumisel

loetakse sisse esimene lindil ettejuhtunud fail. Faili otsimise käigus trükitakse ekraanile senileitud failide nimed.

Magnetofonile kirjutamise õigsust saab kontrollida direktiiviga S:

[<algusaadress>] S [<faili nimi>]

See direktiiv võrdleb mälu sisu magnetlindil oleva koopiaga. Kui parameeter <algusaadress> puudub, võetakse algusaadress lindilt. Faili nime puudumisel võrreldakse mälu sisu esimese ettejuhtunud faili sisuga. Lugemisel ja võrdlemisel trükitakse ekraanile otsimise käigus leitud failide nimed. Otsimine lõpeb failide lõpu tunnuse leidmisel. Vt. ka lisa 1.

Arvuti mälujactus

Arvutis Tartu on 64K baiti operatiivmälu, millest 12K võtab enda alla ekraan, 2K baiti on eraldatud Monitori tööpiirkonnaks ning 2K baiti reserveeritud süsteemseks otstarbekks (näiteks mittestandardsete välisseadmete juhtprogrammide paigutamiseks). Kasutaja programme jaoks jäab seega 48K baiti vaba operatiivmälu. Püsimälu on arvutis vähemalt 6K baiti (nii palju võtab enda alla Monitor), kuid seda saab laiendada 16K baidini (Monitor + keele BASIC interpretaator + 5-tollise kettaseadme juhtprogramm). Adresseeritavat mälu saab seega kokku olla kuni 80K baiti. Arvutis kasutatav mikroprotsessor KM580BM80A suudab aga adresseerida vaid 64K baiti mälu, seepärast peavad olema ette nähtud mälu ümberlülitamise võimalused. Toome järgneval joonisel ära arvuti

Tartu mäluaotuse ja siis selgitame, kuidas mälу lülitamine toimub.

Operatiivmälу	Püsimälу
FFFF:	FFFF:
	Monitor
	F000:
Ekraan	-----
	EFFF:

D000:	D000:
-----	-----
CF00:	CF00:
Monitori tööpiirkond	Ketta juhtprogramm
C7F0:	C800:
-----	-----
C7EF:	C7FF:
Süsteemi reserv	Monitori konstandid
C000:	C000:
-----	-----
BFFF:	
Vaba mälу	
0100:	
-----	-----
00FF:	
Null-lehekülg	
0000:	

Mälu poole pöördutakse järgmiste reeglite järgi:

- 1) operatiivmälus töötav programm loeb andmeid alati (andmete aadressist sõltumata) operatiivmälust;
- 2) püsimälus töötav programm loeb andmeid operatiivmälust juhul, kui need paiknevad aadressidel 0000h, ..., BFFFh või C800h, ..., CFFFh, või (aadressist sõltumata) magasinioperatsiooniga (POP, RET); muudel juhtudel loetakse andmed püsimälust;
- 3) kui käsuvaliku aadress (protsessori käsuloendaja sisu) on väiksem kui C800h, loeb protsessor järgmisena täide-tava käsu operatiivmälust, vastasel korral püsimälust (sellel reeglil on üks erand, mida vaatleme allpool).

Niisiis saavad operatiivmälus töötavad programmid lugeda andmeid operatiivmälust ilma mingite mälu ümberlülitamise operatsioonideta, kuid ei saa midagi lugeda püsimälust. Viimase operatsiooni jaoks on vaja pöörduda püsimälus paikneva andmete ümbersalvestamise programmi poole (niisugune alamprogramm on Monitoris olemas). Palju sagedamini aga on püsimälus asuvatel programmidel tarvis lugeda andmeid neile kätesaamatust operatiivmälupiirkonnast (näiteks ekraanilt). Selleks võib küll kasutada magasinioperatsioone, kuid see on üsna kohmakas viis ja ei lase pealegi end rahulda vält kasutada koos katkestustega (katkestus kirjutab magasini kaks baiti ning kui magasini viit on parajasti ekraanil, läheb sealt kaks baiti kaduma).

Ulalkirjeldatud probleemi lahendamiseks on arvutis Tartu võimalik ümber lülitada mälu null-lehekülge (aadressidega 0000h, ..., 00FFh). Käsu valik sellel leheküljel toimub kas operatiivmälist aadressidega 0000h, ..., 00FFh või püsimälist aadressidega C000h, ..., C0FFh. Püsimälu sellele leheküljele on paigutatud need Monitori alamprogrammid, mis töötavad harilikult püsimälule kättesamatute operatiivmälu- piirkondadega (andmeid loeb sellel leheküljel töötav programm alati operatiivmälist).

Null-lehekülg on ümberlülitamiseks valitud seetõttu, et siselülitamise järel hakkab protsessor täitma programmi aadressilt 0000h. Kuna arvuti sisselülitamisega on aparatuur-selt seotud ka püsimälu valik null-leheküljel, hakkab arvuti alati täitma püsimäluprogrammi -- ja on seega juhitav.

Ekraani formaat

Arvutis Tartu on ainult monokroomne (must-valge) graafikaekraan, millel saab kujutada 256 rida, igaühes 384 punkti. Ekraanil kujutatav pilt loetakse arvuti operatiivmälist aadressideelt D000, ..., FFFF. Igale mälubitile vastab ekraanil üks punkt, mis helendab siis, kui mälu vastavas bitis on vääratus 1. Ekraani punktide ja mälubitide vastavus on kujutatud järgneval joonisel.

FFFF:FEFF	...	D1FF:D0FF
FFFF:FEFE	...	D1FE:D0FE
...
...
FF01:FE01	...	D101:D001
FF00:FE00	...	D100:D000

Me võime öelda, et ekraani laius on 48 baiti (igaüks 8 punkti). Iga baidi vanim bitt (d7) kujutatakse ekraanil vasakpoolseimana ning noorim bitt (d0) parempoolseimana.

Ekraanil paiknevast tekstist hoitakse arvuti mälus koopiat baitides C9C0h kuni CFFFh; read paiknevad seal kasvavas järeljekorras ning igale reale on eraldatud 64 baiti.

Monitoori väljaantud

Monitor jätab kasutajale vabaks operatiivmälupiirkonna aadressidega 0100h kuni BFFFh (vt. ka arvuti mäluaotuse joonist). Monitori muutujatest võiksid huvi pakkuda järgmised.

C7F0h suunamiskäsk akumulaatorist sümboli väljastamise alamprogrammile (standardselt on seal käsk JMP F81Ch ehk suunamine ekraanile väljastamise alamprogrammi):

- C7F3h suunamiskäsk akumulaatorisse sümboli sisestamise alamprogrammille (standardselt on seal käsk JMP FA45 ehk suunamine klaviatuurilt sisestamise alamprogrammi);
- C807h kursoori horisontaalaadress (veeru number 0 kuni 47. või 63);
- C808h kursoori vertikaalaadress (rea number 0 kuni 24);
- C809h ekraani positiiv/negatiivrežiimi tunnus (väärtused vastavalt 00h või FFh);
- C80Ah kursoori aadress tekstiekraanil (2 baiti);
- C819h tähepiltide algusaadress (2 baiti, standardselt C100h);

Monitori alamprogrammid

Toome järgnevalt härra mõnedesse Monitori koosseisu kuuluvate alamprogrammide algusaadressid ja lühikirjeldused. Märgime, et kõik siintoodud aadressid, v.a. INVEC ja OUTVEC, võivad Monitori järgmistes versioonides muutuda, seetõttu ei maksa nende kasutamisega liialdada (näiteks võib ekraanioperatsioone teha ka juhtsõnade trükkimise abil; vt. ekraani ja klaviatuuri kirjeldust).

* Rea sisestamine puhvrisse aadressidel C901h, ..., C9BFh
a) koos reavahetusega ning kutsungisümboli trükkimisega
baidist C60Dh:

* GETLNZ equ 0F756h

b) kutsungisümboli trükkimisega:

GETLN equ 0F759h

c) kutsungisümboli trükkimiseta:

GETLN1 equ 0F75Fh

* Akumulaatori nelja madalama biti trükkimine kuueteist-
kümnendnumbrina:

PRNIB equ 0F2E0h

* Akumulaatori sisu trükkimine kahekohalise kuueteistkü-
nendarvuna:

PRBYTE equ 0F2D3h

* Registripaari HL sisu trükkimine neljakohalise kuue-
teistkümnendarvuna:

PRADR equ 0F2CAh

* Ekraani viimine positiivrežiimi (heledad tähed tumedal taustal):

SETNOR equ 0F7BCh

* Ekraani viimine negatiivrežiimi (tumedad tähed heledal taustal):

SETINV equ 0F7B9h

* Kursori paigutamine akumulaatoris antud numbriga veergu (\emptyset , ..., 47 või 63, olenevalt ekraani laiusest):

HTAB equ 0F7DFh

* Kursori paigutamine akumulaatoris antud numbriga ritta (\emptyset , ..., 24):

VTAB equ 0F7FDh

* Sümbooli väljastamine ekraanile:

OUTVEC equ 0C7F0h

COUT@ equ 0F81Ch

Kuna aadress OUTVEC paikneb operatiivmälus, siis saab seal paikneva käsu JMP muutmisega väljundseadmeks määrata ka teisi välisseadmeid peale ekraani. Programm COUT@ aga väljastab

sümboli igal juhul ekraanile.

* Sümboli väljastamine ekraanile, kurSOR jäääb paigale:

COUT0 equ 0F8C1h

* Rea lõpu kustutamine ja üleminek järgmisele reale:

RETCLR equ 0F8B9h

* Ekraani puhastamine ja kurSORI paigutamine ekraani üle-
missee vasakusse nurka:

HOMCL equ 0F9A0h

* Klaviatuuri kontroll (akumulaatorisse tekib vajutatud
klahvi kood või FFh, kui ühtegi klahvi pole vajutatud):

KEYIN equ 0FB70h

KEYINP equ 0FBE2h

(esimene neist kontrollib klaviatuuri umbes 10 korda sekun-
dis, teine aga maksimaalkiirusega).

* Sümboli lugemine klaviatuurilt akumulaatorisse (koos
kurSORI näitamisega):

INVEC equ 0C7F3h

RDKEY@ equ 0FA45h

Kuna aadress INVEC paikneb operatiivmälus, siis saab seal paikneva käsu JMP muutmisega sisendseadmeks määrata ka teisi välisseadmeid peale klaviatuuri. Programm RDKEY@ aga sises- tab sümboli igal juhul klaviatuurilt.

- * Mäluosa täitmine etteantud ühebaidise väärtsusega:
B-registris on salvestatav väärthus;
registripaaris DE on baitide arv;
registripaaris HL on esimese baidi aadress;

FILL equ 0F9EAh

- * Mäluosa ümbersalvestamine:
registripaaris BC on lähtekoha algusaadress;
registripaaris DE on lähtekoha lõppaadress + 1;
registripaaris HL on sihtkoha algusaadress;

MOVE equ 0F9F3h

Indeks

ABS, 44
Absoluutväärtus, 38, 44, 57
Akumulaator (A-register), 49, 51,
Alamprogramm, 29, 49, 51-54
Alamsõne, 46, 47
Algusaadress, 64, 67, 70, 71, 80
Algväärtus, 41
AND, 21
Andmed, 73, 74
Andmehulk, 41
Andmetöötlussüsteemid, 4
Argument, 44-49, 51, 57
Aritmeetiline, 57
Arkustängens, 44
Arvavaldis, 44-49
Arvumassiiv, 18
Arvutüüp, 19, 21
ASC, 44
ASCII, 10, 62, 66
Assembler, 4, 51
Assemblerprogramm, 53
Astendamine, 19
Astendamismärk, 15
ATN, 44

Avaldis, 19-21

Basic, 4, 14

BEEP, 23

Break, 56

CAPS, 10, 11

CHR\$, 44

CLEAR, 23

CLS, 24

CONT, 24

COPY, 11-13

COS, 44

CTRL, 5

Ctrl-C, 22

CUR, 24

DATA, 25

DBase II, 4

DEF, 25

DEL, 11

DIM, 26

Dimensioon, 26

Direktiivid, 23

Disjunktsioon, 21

Dollarimärk, 15

Ekraan, 8

Ekraanidirektiivid, 69

ELSE, 30

ESC, 11

EXP, 45

Exponent, 17

FN, 25

FOR, 27

Forth, 4

Fortran, 4

FRE, 45

Funktsoonid, 44

G, 69

GOSUB, 29, 35

GOTO, 29, 30, 35, 37

Graafikaekraan, 74

Heli, 23

I, 68

IF, 30

Indeks, 18

INP, 45

INPUT, 30

INT, 45

Interpretaator, 3, 14, 55, 71

INVERSE, 32

Jagamine, 19

Jagamismärk, 15
Joon, 32
Juhtsümbol, 9
Jutumärgid, 16
Järk, 17

Kahebaidine, 52
Kaldkriips, 15
Kestus, 23
Kirillitsa, 10
Klaviatuur, 8
Koma, 15
Kommentaar, 40
Konjunktsioon, 21
Konkatenatsioon, 20
Konstant, 25, 39, 72
arvuline, 16-19
teksti-, 16, 20, 25
Konstruktsiooniühik, 22, 61
Kood, 11, 44, 61, 69, 79
Koolon, 16
Koordinaadid, 32, 36
Koosinus, 44
Kopeerimine, 67
Korrutamine, 19
Korrutusmärk, 15
Kursor, 5
Kustutamine, 8, 11
Kutsung, 30

Kuueteistkümnendarv, 8, 66
Kävitamine, 5
Küsimärk, 16

L, 63
Ladina, 10
Lahutamine, 19
LAT, 10
LEFT\$, 46
LEN, 46
Lihtmuutujad, 17-19, 55
Liitmine, 19
LINE, 32
Lisp, 4
LIST, 33, 63
LOAD, 33
LOG, 46
Loogelised sulud, 22, 61
Loogiline, 20, 21
Lõppaadress, 64, 67, 80
Lõpudirektiiv, 27

M, 67
Magasin, 73
Mantiss, 17
Masinkood, 51
Massiiv, 17, 18, 26, 27, 41, 55
Massiivikirjeldus, 23
MID\$, 46

Miinusmärk, 15
Mikroprotsessor, 3
Monitor, 80
Murdosa, 17
Muutuja, 17, 18, 26, 27, 30, 34, 39, 41, 75
MÄlu, 41, 45, 55, 60, 62, 63, 66, 71-74
MÄluadress, 61, 69
MÄlujactus, 55, 71, 75
MÄluosa, 23, 45, 52, 55, 62, 64, 65, 67, 80
MÄlupesa, 37, 47
MÄlupiirkond, 53, 65, 67
Märk, 17, 47

N, 69
Naturaallogaritm, 46
Negatiivne, 38
Negatiivrežiim, 9, 70, 76, 78
Negatsioon, 21
NEW, 34
NEXT, 27, 28
Nooleklahvid, 11
Noolsulud, 16, 22, 61
NORMAL, 34
NOT, 21
Null-lehekülg, 72, 74
Numbriklahv, 11
Nurksulud, 22, 61

O, 68
OK, 6, 56
Omistamismärk, 15
Operatiivmälu, 55, 66, 71-74
Operatiivmälupiirkond, 73-75
Operatsioonisüsteem, 4, 11
OR, 21
Otspunktid, 32
OUT, 36, 68
Overflow, 57

P, 62
Parameeter, 25, 26, 52, 63
Pascal, 4
PEEK, 47, 51, 52
Pikkus, 16, 46, 47, 61, 70
Pistikupesa, 5
PLOT, 36
Plussmärk, 15
POKE, 37, 51, 52
Poollogaritmiline kuju, 17, 38
Port, 36, 45, 61, 68
POS, 47
Positiivne, 38
Positiivvrežiim, 9, 70, 78
PRINT, 37, 62
Prioriteet, 19, 21
Programm, 14
Pseudojuhuslik, 47

Punkt, 16, 17, 32, 36, 74, 75
Püsimalü, 3, 71-74
Püsimaluprogramm, 74
Püstkriips, 22

R, 71
Raja, 27
READ, 39, 41
Reanumber, 14, 24, 29, 33, 35, 40, 41
Reavahetus, 16
Redigeerimine, 11
Redigeerimisklahvid, 11
Registriklahvid, 10, 11
Register, 51
REM, 40
RENUMBER, 40
RESET, 5
RESTORE, 39, 41
RETURN, 6, 11, 14, 16, 29, 35
Rida, 8, 14, 24, 29, 33, 74
RIGHT\$, 47
RND, 47
RUB, 11
RUN, 41
RUS, 10
Ruutjuur, 48

S, 71
Salvestamine, 60

SAVE, 42
Semikoolon, 16
SGN, 47
SH, 10
SHIFT, 10
Sinus, 48
SIN, 48
Sirglöök, 32
Sisendparameeter, 53
Sisendrida, 14
Sisestamine, 10, 14, 18, 30, 61, 66, 76
Sisselülitamine, 5, 74
SPC, 48
SQR, 48
STEP, 27
STOP, 24, 42, 56
STR\$, 48, 50
Sulud, 15, 22, 61
SuperCalc, 4
Suurtäht, 10
Sõne, 16
Sümbol, 15
Sümbolijada, 8, 40, 61

T, 66
T-BASIC, 3, 6, 22
TAB, 11, 48
Tagasipördumine, 35, 49
TAN, 49

Tangens, 49
Tehe, 19-21
Tehitemärgid, 19, 21
Teisendamine, 18, 48, 49
Tekst, 33, 55, 66, 75
Tekstiavaldis, 44-49
Tekstikonstant, 16, 20, 25
Tekstimassiiv, 18
Tekstimuutuja, 18, 23, 45, 55
Tekstiredaktor, 4
Tekstitüüp, 18, 20
THEN, 30
Tingimus, 28, 30
TO, 27
Toitejuhtmed, 5
Toitelüiliti, 5
Toiteplok, 5
Trükikuju, 18
Trükkimine, 37, 38, 62
Tsüklidirektiiv, 27, 28
Tsükliloendaja, 27, 28
Töevärtused, 20
Täheklahv, 10
Tähestik, 15
Täht, 15
Täisarv, 17
Täisosa, 45
Tärn, 15
Tühik, 16, 48

Tühisõne, 41
Tüvenumber, 17

USR, 49, 51
Uuestikävitamine, 24

V, 64
Vahetu täitmise režiim (direktiivid), 14, 26, 33, 42
VAL, 49
Veateated, 56
Veerg, 8, 9, 24, 38, 47, 48, 76, 78
Veerunumbrid, 24
VERIFY, 42, 64, 65
Viga, 31, 57, 59
Võrdlemine, 42, 64
Võrdlus, 21
Võrdlusemärk, 20
Väärtus, 18-21, 24, 31, 36-39, 41, 44-49, 55, 62, 66-69, 74, 80
Väiketäht, 10
Välisseadmed, 36, 45, 60, 61, 68, 71, 78, 80
Väljastamine, 8, 9, 33, 38, 48, 62, 63, 75, 78, 79

W, 70
WordStar, 4

Umarsulud, 15, 19, 21, 22, 61
Umberlülitamine, 70, 73, 74

X, 70

X-koordinaat, 32, 36

Y-koordinaat, 32, 36

Z, 69

Lisa 1. Magnetofoniga seotud teated

Magnetofoni kasutamist juhtivate direktiivide (vt. keele BASIC ja Monitori vastavaid direktiive) käivitamisel ilmub lugemise korral ekraanile teade

Press PLAY on tape

& any key on keyboard

ning salvestamise puhul teade

Press RECORD & PLAY on tape

& any key on keyboard

Magnetofoni tööd juhtivate direktiivide täitmise ajal ilmuvad ekraanile veel teated

Searching

Loading

Verifying

Saving

Iga lindilt leitud faili nimi trükitakse ekraanile kujul

Found <faili nimi>

Teade

End of files

ilmub ekraanile failide lõpu tunnuse sisselugemise järel.

Lindilt lugemisel ja võrdlemisel võivad esineda teated

LOAD error

VERIFY error

OK

milledest viimane teatab magnetofonioperatsiooni edukast täitmisest.

Lisa 2. Laienduspesa X2 kirjeldus

Arvuti küljel asuvat laienduspesa X2 saab kasutada välisseadmete ühendamiseks. Välisseadme toiteks võib kontaktidel A31 ja B31 tarbida voolu kuni 200 mA. Lahtise kollektoriga väljundid ei ole arvuti plaadil varustatud koormustakistitega; peale selle on nende funktsioneerimiseks vaja anda toitepinge +5 V kontaktile A1 (näiteks ühendussilla A1-A31 abil). Lahtise kollektoriga väljundid on koormatavad vooluga kuni 40 mA ja taluvad pinget kuni 30 V. Kõik ülejäävad väljundid on koormatavad ühe standardse TTL-sisendiga. Sisenditest on 4.7 kilo-oomiste koormustakistitega varustatud pordi FDh kõik bitid ning pordi F9h bitid d0 ja d7. Laienduspesaga ühendamiseks tuleb kasutada 64-kontaktelist pistikut CH063-64. Järgnevas tabelis on kontaktide jooks kasutatud selle pistiku tähistusi.

Lahtise kollektoriga väljundid on tabelis märgitud tärniga. Minusmärgiga on tähistatud inverteeritud signaalid, s.t. signaalid, millel väärtsusele "üks" vastab madal pingenvoo.

Lisaks staatilistele signaalidele portide väljundregistritest on laienduspesale X2 toodud veel mõned impulsssignaalid, mida vajaduse korral saavad kasutada yahetult peasse ühendatud seadmed (ühenduskaabliga neid signaale koormata ei tohi). Need on portide F0h (või F8h), F4h (või FCh) ja F6h (või FEh) strobeerimissignaalid, mis tekivad vastava aadressiga (pordi numbriga) sisestamis- või väljastamiskäsu täitmise hetkel (signaalide kestus on 0.5 mikrosekundit ja aktiivne nivoo madal), ja aadress a3, mis võimaldab omavahel eristada strobeerimissignaale F0h ja F8h, F4h ja FCh ning F6h ja FEh. Signaal a3 on kehtiv strobeerimissignaalide matala nivoo ajal.

Väljundpört F0h (F8h) on täiesti vabalt kasutatav, pordi F4h (FCh) kasutamise juures tuleb aga meeles pidada, et selle väärtsust muudavad klaviatuuriga tegelevad programmid (kasutades aadressi FCh). Bitid d7 kuni d4 nullitakse, bitid d3 kuni d0 aga jävad määramata seisu. Et klaviatuuriprogrammid ei tööta katkestusrežiimis, on bittide d7 kuni d4 kasutamine siiski täiesti võimalik.

Kontakt Suund

A1		väljundpuhvrite toide
A2		vaba
A3		vaba
A4		vaba
A5		vaba
A6		vaba
A7		vaba
A8		vaba
A9		vaba
A10		maa
A11 -	sisend	pordi F1h bitt d0
A12 -	sisend	pordi F1h bitt d6
A13 -	sisend	pordi F1h bitt d5
A14	väljund	pordi F0h (F8h) bitt d3
A15	väljund	pordi F0h (F8h) bitt d2
A16	väljund	pordi F0h (F8h) bitt d1
*A17 -	väljund	pordi F0h (F8h) bitt d6
*A18 -	väljund	pordi F0h (F8h) bitt d4
*A19 -	väljund	pordi F0h (F8h) bitt d2
*A20 -	väljund	pordi F0h (F8h) bitt d0
*A21 -	väljund	pordi F4h (FCh) bitt d3
*A22 -	väljund	pordi F4h (FCh) bitt d2
A23	väljund	pordi F4h (FCh) bitt d2
A24	väljund	pordi F4h (FCh) bitt d0
A25 -	sisend	pordi F9h bitt d5
A26 -	sisend	pordi FDh bitt d6 (4.7k takistiga)
A27 -	sisend	pordi FDh bitt d4 (4.7k takistiga)
A28 -	sisend	pordi FDh bitt d2 (4.7k takistiga)
A29 -	sisend	pordi FDh bitt d0 (4.7k takistiga)
A30 -	sisend	pordi F9h bitt d0 (4.7k takistiga)
A31		+5V
A32		maa

Kontakt Suund

B1		vaba
B2	-	väljund
B3	-	väljund
B4		
B5		
B6	-	väljund
B7		
B8	-	väljund
B9	-	väljund
B10		
B11		
B12		
B13		väljund
B14		väljund
B15		väljund
B16		väljund
*B17	-	väljund
*B18	-	väljund
*B19	-	väljund
*B20	-	väljund
*B21	-	väljund
*B22	-	väljund
B23		väljund
B24		väljund
B25		sisend
B26	-	sisend
B27	-	sisend
B28	-	sisend
B29	-	sisend
B30	-	sisend
B31		+5V
B32		maa

Lisa 3. Arvuti Tartu tähestik

Dec	Hex	Ctrl	Tähistus	Dec	Hex	Ctrl	Tähistus
0	00	^@	NUL	16	10	^P	DLE
1	01	^A	SOH	17	11	^Q	DC1
2	02	^B	STX	18	12	^R	DC2
3	03	^C	ETX	19	13	^S	DC3
4	04	^D	EOT	20	14	^T	DC4
5	05	^E	ENQ	21	15	^U	NAK
6	06	^F	ACK	22	16	^V	SYN
7	07	^G	BEL	23	17	^W	ETB
8	08	^H	BS	24	18	^X	CAN
9	09	^I	HT	25	19	^Y	EM
10	0A	^J	LF	26	1A	^Z	SUB
11	0B	^K	VT	27	1B	^_	ESC
12	0C	^L	FF	28	1C	^`	FS
13	0D	^M	CR	29	1D	^]	GS
14	0E	^N	SO	30	1E	^~	RS
15	0F	^O	SI	31	1F	^_	US

Dec	Hex	Süm									
32	20	!	48	30	Ø	64	40	@	80	50	P
33	21	!	49	31	1	65	41	A	81	51	Q
34	22	"	50	32	2	66	42	B	82	52	R
35	23	#	51	33	3	67	43	C	83	53	S
36	24	\$	52	34	4	68	44	D	84	54	T
37	25	%	53	35	5	69	45	E	85	55	U
38	26	&	54	36	6	70	46	F	86	56	V
39	27	'	55	37	7	71	47	G	87	57	W
40	28	(56	38	8	72	48	H	88	58	X
41	29)	57	39	9	73	49	I	89	59	Y
42	2A	*	58	3A	:	74	4A	J	90	5A	Z
43	2B	+	59	3B	:	75	4B	K	91	5B	[
44	2C	,	60	3C	<	76	4C	L	92	5C	\
45	2D	-	61	3D	=	77	4D	M	93	5D]
46	2E	.	62	3E	>	78	4E	N	94	5E	^
47	2F	/	63	3F	?	79	4F	O	95	5F	—

Dec	Hex	Süm	Dec	Hex	Süm
96	60	'	112	70	P
97	61	a	113	71	q
98	62	b	114	72	r
99	63	c	115	73	s
100	64	d.	116	74	t
101	65	e.	117	75	u
102	66	f	118	76	v
103	67	g	119	77	w
104	68	h	120	78	x
105	69	i	121	79	y
106	6A	j	122	7A	z
107	6B	k	123	7B	{
108	6C	l	124	7C	—
109	6D	m	125	7D	}
110	6E	n	126	7E	~
111	6F	o	127	7F	☺

Dec	Hex	Симв									
128	80	Ӯ	144	90	Ӯ	160	A0	Ӯ	176	B0	Ӯ
129	81	Ӑ	145	91	Ҭ	161	A1	Ӑ	177	B1	Ҭ
130	82	Ӯ	146	92	Ӯ	162	A2	Ӯ	178	B2	Ӯ
131	83	Ӯ	147	93	Ӯ	163	A3	Ӯ	179	B3	Ӯ
132	84	Ӯ	148	94	Ӯ	164	A4	Ӯ	180	B4	Ӯ
133	85	Ӯ	149	95	Ӯ	165	A5	Ӯ	181	B5	Ӯ
134	86	Ӯ	150	96	Ӯ	166	A6	Ӯ	182	B6	Ӯ
135	87	Ӯ	151	97	Ӯ	167	A7	Ӯ	183	B7	Ӯ
136	88	Ӯ	152	98	Ӯ	168	A8	Ӯ	184	B8	Ӯ
137	89	Ӯ	153	99	Ӯ	169	A9	Ӯ	185	B9	Ӯ
138	8A	Ӯ	154	9A	Ӯ	170	AA	Ӯ	186	BA	Ӯ
139	8B	Ӯ	155	9B	Ӯ	171	AB	Ӯ	187	BB	Ӯ
140	8C	Ӯ	156	9C	Ӯ	172	AC	Ӯ	188	BC	Ӯ
141	8D	Ӯ	158	9D	Ӯ	173	AD	Ӯ	189	BD	Ӯ
142	8E	Ӯ	158	9E	Ӯ	174	AE	Ӯ	190	BE	Ӯ
143	8F	Ӯ	159	9F	Ӯ	175	AF	Ӯ	191	BF	Ӯ

Tr. antud 10.10.89.a. 60x84/16 T-200
Tr.p. 6,25. Arvp. 4,5. Tell. nr. 335-39

PI "EKE Projekt" rotaprint

