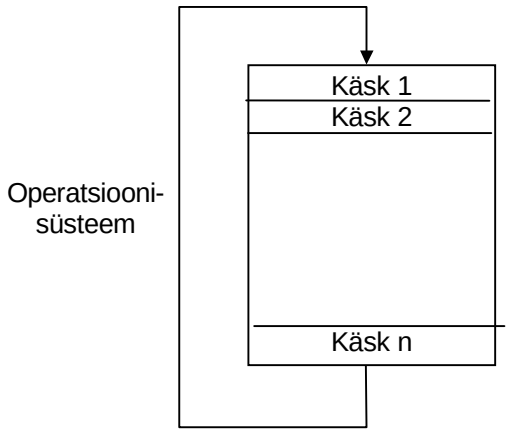
8 Programmeerimise alused

8.1 Kasutajaprogrammide struktuur

Kõikide juhtseadmete protsessorites, sh. kontrollerites töötavad kahte tüüpi programmid: *operatsioonisüsteem* ja *kasutajaprogrammid*. Igal protsessoril on operatsioonisüsteem, mis organiseerib protsessori tööd ning pole otseselt seotud juhitava protsessiga ehk juhtimisülesandega, mis hõlmab vastavate sisendite olekutele põhinevaid väljundite reaktsioone. Operatsioonisüsteem organiseerib näiteks protsessori käivitust, protsessikuvade sisend- ja väljundtabelite moodustamist, kasutajaprogrammi väljakutsumist, määrab katkestused, töötleb vigu, korraldab tööd mälupiirkondadega ning kommunikeerub programmaatorite ja teiste seadmetega. Kasutajaprogramm tuleb kasutajal endal programmeerida ja protsessorisse laadida. Kasutajaprogramm on kindla protsessi juhtimiseks, mis tähendab seda, et selles programmis saab määrata reaktsioonid vastavatele katkestustele, töödelda protsessist tulevaid andmeid, määrata tingimused protsessori lülitamiseks talitlusse ning programmi talitluskõrvalekallete töötlemiseks.

Kasutajaprogrammi võib omakorda jaotada *lineaarseks* ja *jaotatud* programmiks. Siit tulenevalt eristatakse ka lineaarset ja jaotatud programmeerimist.

Lineaarseks programmeerimiseks nimetatakse programmi koostamist, milles kõik programmikäsud on üksteise järel reas ning nende täitmine toimub samas järjekorras (joonis 8.1). Lineaarset programmeerimist kasutatakse lihtsate automaatikaseadmete, nagu loogikamoodulite (nt. LOGO!) puhul, millega juhitakse ka suhteliselt lihtsaid protsesse.



Joonis 8.1. Lineaarne programmeerimine

Jaotatud programmeerimiseks nimetatakse sellist programmi koostamist, milles koostaja jaotab juhtimisprotsessi operatsioonideks ja osaülesanneteks, mis on omavahel seotud kindlate tingimustega. Jaotatud programmeerimise eelisteks on

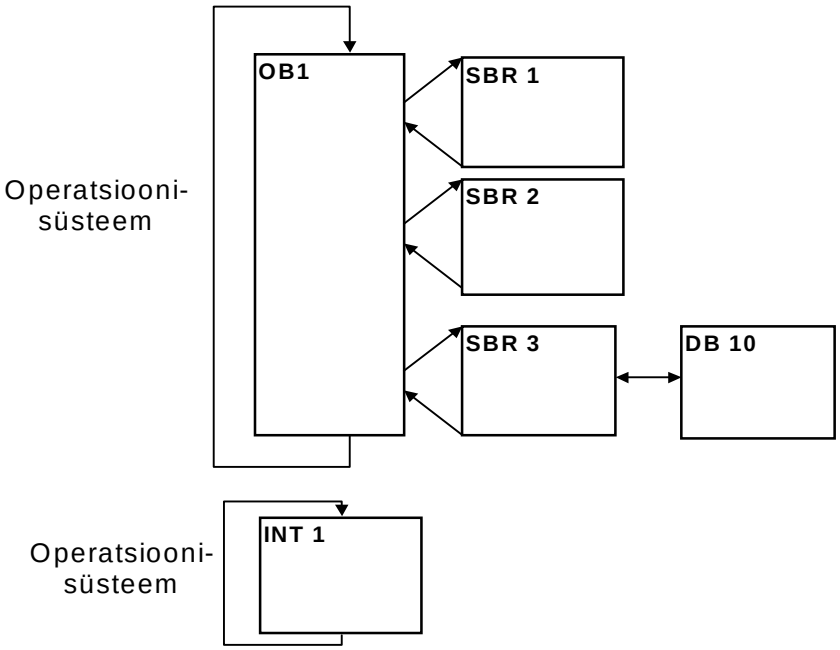
- programmeerimise lihtsus,
- alamprogrammide korduva kasutamise võimalus,
- programmi lugemise hõlpsus,

- programmi redigeerimise lihtsus,
- programmi testimise lihtsus, sest programmi erinevaid osi ehk plokke saab testida eraldi,
- protsessi määratlemise ja programmeerimise lihtsus.

Jaotatud programmeerimist kasutakse nt. programmeeritavate kontrollerite ja teiste kõrgkeeles programmeeritavate seadmete juures. Näiteks SIMATIC S7 kontrolleritarkvara võimaldab jaotatud programmeerimist, milles programmi üksikuid operatsioone ehk jaotisi nimetatakse plokkideks. SIMATIC S7 kasutajaprogramm koosneb plokkidest (nn. kasutaja programmplokkidest), käskudest ja aadressidest. Tabel 8.1 [12] annab ülevaate sellest, missuguseid plokke saab STEP 7 keeles kasutada ja millised on nende funktsioonid.

Enne kasutaja programmploki töötlemist peab toimuma vastav päring ehk *siire*. Programmi siirdumiseks kasutatakse STEP 7 erikäske. Plokki siirdumist saab programmeerida mõne teise ploki sees. Sellist siirdumist ühest plokist teise võib võrrelda siirdumisega (ülem)programmist alamprogrammi. Iga selline siire tähendab ploki vahetust. Päringu esitanud ploki (kust siirduti) aadress säilitatakse operatsioonisüsteemis, et alamprogrammi lõppedes saaks siirduda tagasi põhiprogrammi. Plokkidesse siirdumise korda ehk järjestust nimetatakse *hierarhiaks*. Teisiti öeldes, korda mis määrab, millised plokid võivad olla ühtedele plokkidele alam- ja teistele ülemprogrammplokkideks, nimetatakse hierarhiaks (joonis 8.2). Plokkide arv hierarhias sõltub protsessori tüübist.

Hierarhias kutsub alamprogrammplokke välja OB- juhtprogrammplokk.



Joonis 8.2. Hierarhiline jaotatud programmeerimine

Tabel 8.1. STEP 7 MicroWIN tarkvaras kasutatavad programmiplokid

Plokid	Kirjeldus
OB	Juhtprogrammplokk, mille poole vahetult operatsioonisüsteem pöördub ja kus määratakse siirded alamprogrammiplokkidesse või katkestuste programmiplokkide käivitamine
SBR	Alamprogrammiplokk, mille käivitab juhtprogrammiplokk kui käivitamiseks vajalik tingimus on tõene.
INT	Katkestuste programmploki käivitab operatsioonisüsteem kui juhtprogramm- või alamprogrammiplokis on antud vastav käsklus ja sellele eelnev tingimus on tõene.

8.2 Käsurea struktuur

Juhtimisülesanne programmeeritakse üksikute käskude kaupa. Käsk on iseseisev ja sõltumatu programmi osa ning sisaldab juhtsüsteemi tööks vajalikku informatsiooni. Käsk on kasutajaprogrammi väikseim osa nii käsulisti kujul programmeerides kui ka programmimällu salvestatuna. Käsk on määratud käsuvorminguga (joonis 8.3) ning koosneb *operatsiooni koodist* ja *operandist*.

Juhtkäsk		
Operatsioon (mida teha?)	Operand (millega ja kus teha?)	
	Tunnus	Parameeter
A	*I*	*0.5*

Joonis 8.3. Juhtkäsk

Tähis *A* vastab STEP7 keeles NING-tehtele, *I* vastab sisendile ja *0.5* on aadress. Sõnades võib öelda, et ning sisendis aadressiga 0.5 on olek “1”. Sisendaadressi *I 0.5* korral *0* tähistab sisendbaiti ja *5* selle sama baidi viiendat bitti.

Käsu operatsiooni- ehk tehtekood määrab, mida tuleb teha. Operand sisaldab operatsiooni ehk juhtkäsu jaoks täpsustavat teavet, s.t. vastab küsimusele "kus ja millega teha?". Operandi moodustavad operandi tunnus ja parameeter. Operandi tunnuseks võib olla sisend (*I*), väljund (*Q*), märgend (*M*) jne. erinevates informatsioonihulkades, nagu bitt, bait, sõna ja topeltsõna. Seega operandi tunnus määrab informatsiooni asukoha protsessori mälus (tabel 8.2). Parameeter on operandi aadress, mis määrab informatsiooni asukoha aadressi mälus.

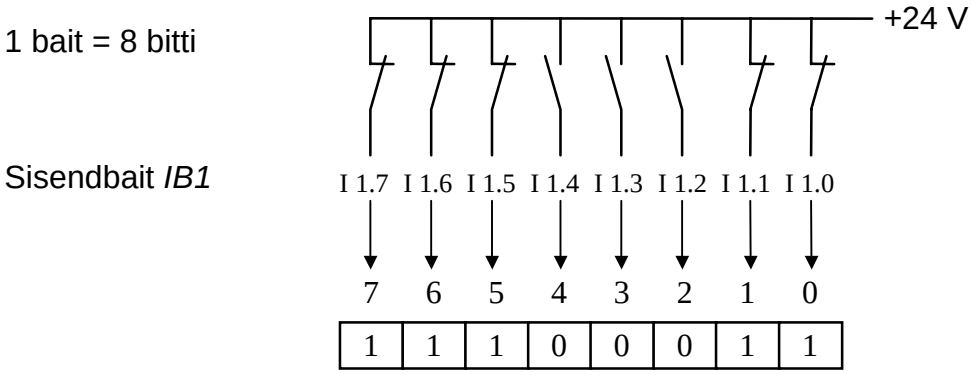
Programmeeritavas kontrolleris kasutatakse bitt-, bait-, sõna- ja topeltsõnaaadresse. Operandi tunnustena kasutatakse sisendi puhul sõltuvalt sõnapikkusest vastavalt tähised *I*, *IB*, *IW* ja *ID*. Programmeeritavas kontrolleris toimivad vaid kahendsignaalid. Olek “0” tähistab pinge puudumist (0 V), olek “1” tähistab pinge olemasolu (24 või 220 V). Igal bitil on oma järjekorranumber ehk aadress. Baidi parempoolne bitt on aadressiga 0, ning vasakpoolne aadressiga 7. Samuti omab aadressi iga bait.

Tabel 8.2 Operandide kirjeldus

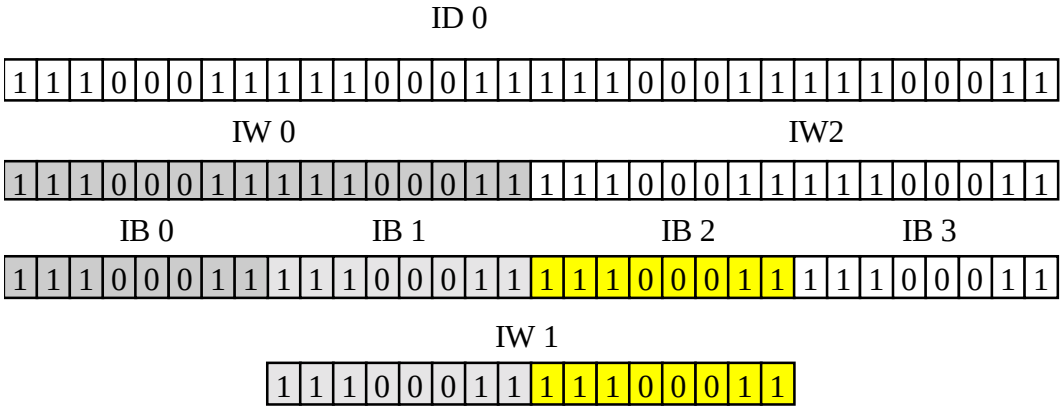
Operand	Nimetus	Kirjeldus
I	Sisendite protsessikuva ehk vahemälu	Tsükli alguses loeb operatsioonisüsteem sisendsignaalide olekud ja asetab need sisendite protsessikuvasse.
Q	Väljundite protsessikuva ehk vahemälu	Tsükli kestel määrab programm väljundsuurused või -olekud ning asetab need väljundite protsesskuvasse. Tsükli lõppedes loeb operatsioonisüsteem väljundite protsessikuvas paiknevad andmed ning olekud ja kannab need väljunditesse.
V	Muutujate mälu	Kasutab programmeerija protsessi juhtimiseks vajalike vaheolekute ja andmete säilitamiseks ja lugemiseks.
M	Bitimälu	Mälupiirkond vaheolekute operatsioonideks. Sarnaneb eelmisele, kuid mällu salvestatakse info biti kaupa.
S	Sammprogramm-juhtimise mälu	Kasutatakse vastavate käskude abil, et realiseerida protsesside sammprogramm-juhtimist.
SM	spetsiaalmälu	Seadme valmistaja poolt ettenähtud mälupiirkonnad spetsiaalsete operatsioonide teostamiseks. Vastavalt seadme käsiraamatule iga mälu piirkond omab kindlat funktsiooni, mida on võimalik kasutada spetsiifiliste toimingute teostamiseks nagu näiteks andmeside jne.
T	Taimerid	Mälupiirkond viivitus- ehk aja-operatsioonideks.
C	Loendurid	Mälupiirkond loendusoperatsioonideks.
AI	Analoogsisendid	Mälupiirkond otseseks kiireks protsessist mõõdetavate sisendsuurustele juurdepääsuks
AQ	Analoogväljundid	Mälupiirkond otseseks kiireks protsessi saadetavate väljundsuurustele juurdepääsuks.
AC	Akumulaatorid ehk vahemälud	Vahemälu programmis jooksvalt töödeldavate andmete jaoks.
HC	Kõrgsagedus-loendurid	Kõrge sagedusega impulsside loendamiseks sõltumata protsessori töökiirusest.
DB	Andmeplokid	Andmetele reserveeritud mälupiirkond, kuhu võib programmi plokkidest pöörduda.
L	Kohtmälu	Mälupiirkond programmiplokkide sees andmete töötlemiseks, säilitamiseks ja lugemiseks. Kui programmploki ehk alamprogrammploki töötlus on lõppenud, ei ole need andmed enam kättesaadavad. Sarnaneb olemuselt V-mälule.
2000, 16#A5, 'tere', +1.0E+02, 2#01101110	Konstandid	Operandid on ka konstandid, mida võib esitada 10nd arvuna, 16nd arvuna, ASCII-koodina, reaalarvuna ja 2nd arvuna.

Operandide I, Q, V, M, L, S, SM tunnuseks võib olla bitt, bait, sõna ja topeltsõna. Operandide AC, HC, T, C, DB tunnuseks on vastava seadme või registri number. Operandide AI, AQ tunnuseks on sõna.

Joonisel 8.4 on toodud sisendbait aadressiga 1 (*IB 1*). Baiti kuuluvaid bitte tähistatakse *I 1.0 - I 1.7*. Sisendite ja väljundite tähistamiseks kasutatakse vastavalt inglise keeles tähti *I* ja *Q*. Sõnade ja topeltsõnade kasutamisel tuleb arvestada, et nad omavad alati neisse kuuluva noorima baidi aadressi (joonis 8.5).



Joonis 8.4. Sisendbait



Joonis 8.5. Sõnavormingud

8.3 Andmetüübid ja adresseerimisviisid

Programmeerimiskeeles STEP 7 kasutatakse andmeid biti-, baidi-, sõna- ja topeltsõnapikkustena. Järgnevalt vaadeldakse mõningaid andmetüüpe (tabel 8.3), mida saab esitada eri andmeühikutes.

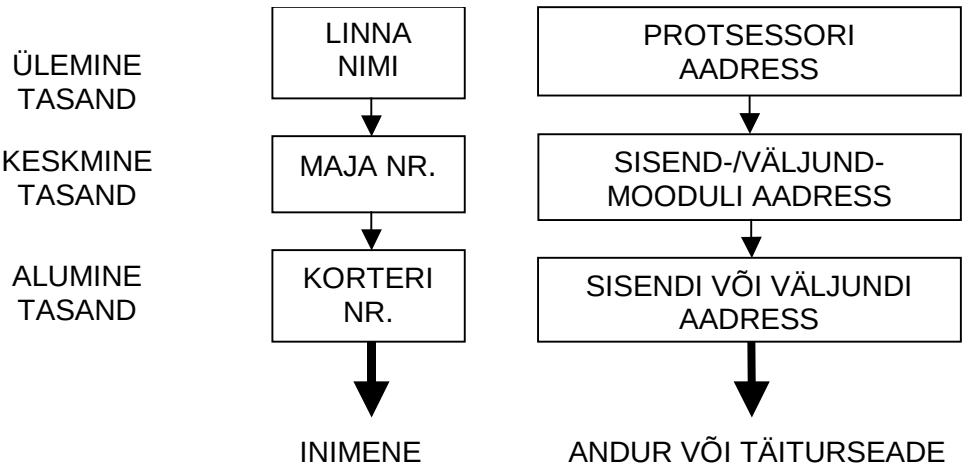
Tabel 8.3 Andmetüübid

Andmetüüp (andmeühik)	Andmevormingud	Piirkond
BOOL (bitt)	kahendarv	0, 1
BYTE (bait)	märgita bait, märgiga bait	0…255, -128…+127
WORD (sõna)	märgita täisarv	0…65535
INT (sõna)	märgiga täisarv	-32768…+32767
DWORD (topeltsõna)	märgita topeltsõna täisarv	0…4294967295
DINT (topeltsõna)	märgiga topeltsõna täisarv	-2147483648…+2147483647
REAL (topeltsõna)	ujukoma ehk reaalarv	+1.175495E-38…+3.402823E+38 -1.175495E-38…-3.402823E+38

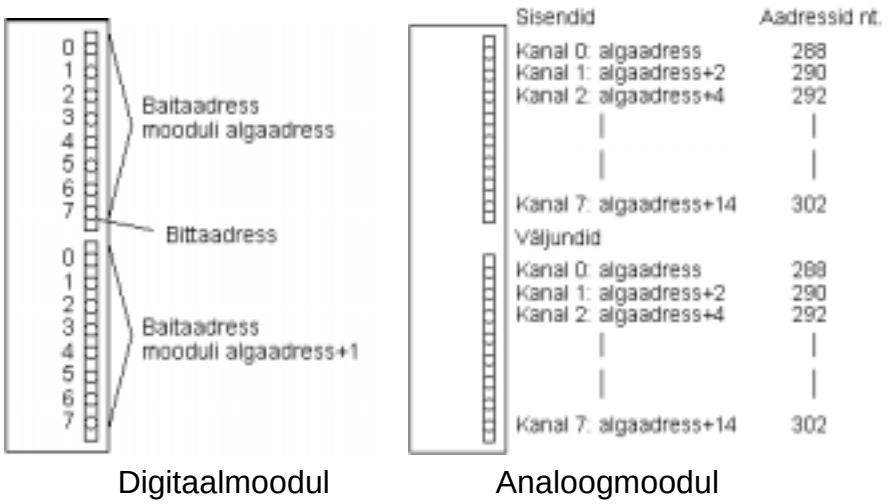
Peale elementaarandmetüüpide võimaldab STEP 7 – Micro/WIN 32 kasutada ka kompleksandmetüüpe (nt. loendurid ja viivitusahelad).

Enamik kaasaegseid sisend-/väljundmooduleid adresseeritakse tarkvaraliselt programmeerimispaketi abil. Riistvaralise adresseerimise korra antakse moodulitele aadress mitmepositsiooniliste ümberlülitite abil, millede positsioonid vastavad konkreetsetele aadressidele. Programmeeritavate kontrollerite sisend- ja väljundmoodulite adresseerimine sh. sisenditele ja väljunditele adresside sätestamine on vajalik nende külge ühendatavate andur- ja täiturseadmete identifitseerimiseks. Samuti sätestatakse aadressid protsessormoodulitele (sh. kommunikatsiooni-protsessorid ja erimoodulid), mis on vajalik nende tuvastamiseks andmesidevõrkudes. Neid aadresse võib nimetada kõrgema tasandi aadressideks.

Programmeeritavate kontrollerite moodulite aadresseerimist võib võrrelda inimesele elukoha järgi aadressi omistamist, mida ilmestab ka järgmine joonis 8.6.



Joonis 8.6. Adresseerimise selgitus



Joonis 8.7 Sisend- ja väljundmoodulite adresseerimine

Digitaalsisend- ja digitaalväljundmoodulite aadress koosneb baitaadressist ja bitt aadressist. Mooduli baitaadressi määrab tema algaadress ja bittaadressi määrab mooduli peale trükitud või kataloogis antud number (joonis 8.7).

Analoogsisend ja väljundid ehk kanalid omavad alati sõnaaadressi ja iga kanali aadress sõltub mooduli algaadressist. Analoogsisend-/-väljundmoodulil sisendite algaadress vastab väljundite algaadressile. Puhtalt analoogsisend- või -väljundmoodulil ei ole eri kanalitel sarnaseid aadresse ja iga järgneva kanali sõnaadress on 2 võrra suurem eelmisest (joonis 8.7).

Programmide kirjutamisel eristatakse otsest- ja kaudset adresseerimist. Otsese adresseerimise korral käsureas esitatud operandi tunnus viitab otse mälupiirkonda kust vastavalt operandi parameetrile loetakse mälupesast vastav suurus, mida kasutatakse programmi töös. Otsese adresseerimise kirjeldamiseks olgu toodud järgmine käsurida LD M 0.0, mis tähendab, et mälu aadressile 0.0 salvestatud olek loetakse vahemällu. Algajale programmeerijale on kaudne adresseerimine keerukam.

kontrollerite sarja S7-200 puhul saab kasutada kaudset adresseerimist järgmiste operandi tunnuste korral: I, Q, V, M, S, T (jooksev väärtus), and C (jooksev väärtus). Kaudset adresseerimist ei saa kasutada bittide või analoogsuuruste puhul. Et kasutada kaudset adresseerimist, tuleb kõige pealt defineerida viit mis osutab vastavasse mälu piirkonda. Viidad on topeltsõnapikkused mälu piirkonnad, mis sisaldavad mälu aadressi, kus opereeritav suurus paikneb. Viitadena saab kasutada ainult V-mälu, M-mälu ja aku registreid (AC1, AC2, AC3).

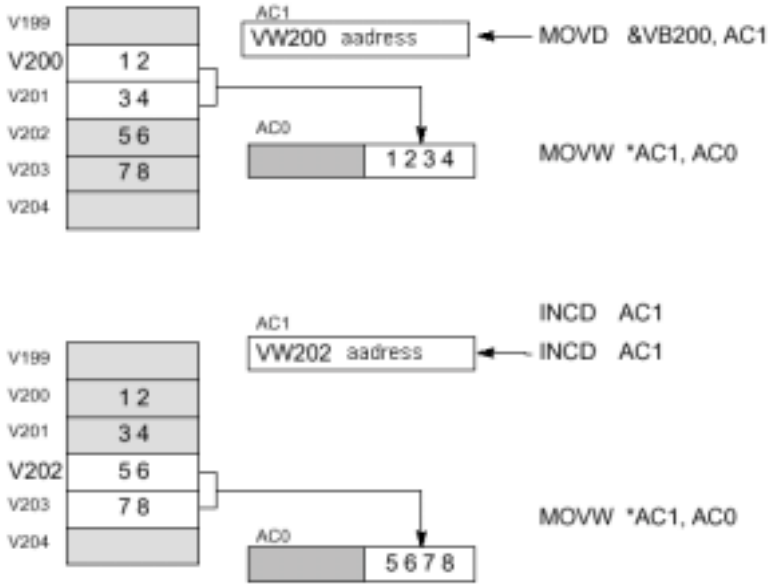
Viida defineerimiseks kasutatakse MOVD-käsku, mis laeb aadressi kaudselt adresseeritud mälupiirkonnast viidaga määratud mälupiirkonda. Näiteks: ***MOVD &VB100, VD204; MOVD &MB4, AC2; MOVD &C4, LD6*** kus sisendoperandile eelneb sümbol &, mis kirjeldab milliselt mälu aadressilt laetakse vastav suurus ja väljundoperand on viidaks sellele mäluaadressile.

Tärn (*) operandi ees määrab ära et seda operandi kasutatakse viidana. Näiteks *AC1 tähendab seda, et vastavalt käsule MOVW on AC1 viidaks sõnapikkusele väärtusele. Nagu joonisel 8.8 on näidatud aadressidele V200 ja V201 laetakse akusse ehk registrisse AC0.

Viida väärtus on muudetav. Kuna viidad on topeltsõnapikkused, tuleb kasutada topeltsõnade jaoks ettenähtud käske, et viida väärtust muuta. Selleks võib kasutada lihtsaid matemaatika käske nagu liitmine ja suurendamine, et muuta viida väärtust. Viida väärtuse muutmisel tuleb arvestada alljärgnevaga:

- Kui tegemist on baitsuurustega, tuleb viida väärtust suurendada ühe võrra.
- Kui tegemist on sõna, taimeri või loenduriga, tuleb liita või suurendada viida väärtust kahe võrra.
- Kui tegemist on topeltsõnapikkuste suurustega, tuleb liita või suurendada viida väärtust nelja võrra.

Joonisel 8.8 on toodud üks näide kuidas saab luua kaudse aadressi viita, s.t., kuidas andmed on kaudselt kättesaadavad ja kuidas suurendatakse viida väärtust.



Joonis 8.8 Kaudse adresseerimise näide