Ttk-91 esimerkkitietokone ja sen simulaattori

Konekielinen ohjelmointi

Esimerkkitietokone ttk-91

Ttk-91 koneen rakenne ja käskykanta-arkkitehtuuri

Tietokoneen simulaattori

Ttk-91 ohjelmien suorittaminen simulaattorissa

Tässä luennossa esitellään esimerkkitietokone ttk-91 ja siihen liittyvä simulaattorijärjestelmä. Ensi alkuun perustelemme, miksi konekieltä yleensä kannattaa opiskella, ja sitten, miksi juuri ttk-91 olisi hyvä lähtökohta käskykanta-arkkitehtuuriin tutustumiseen. Pääosa luennosta on kuitenkin ttk-91 tietokoneen käskykannan ja toiminnan kuvausta. Itse konekieliseen ohjelmointiin tälle koneelle pääsemme vasta seuraavalla luennolla. Lopuksi esittelemme simulaattorin idean ja toiminnan yleispiirteissään, ja erityisesti tällä kurssilla käytettävän Titokoneen käytön perusasiat.

Miksi konekieltä?

Koneen toiminnan ymmärtäminen

Oman ohjelman toiminnan ymmärtäminen

Koneenläheinen ohjelmointi (konekielinen ohjelmointi)

Kääntäjän tekeminen

- kääntäjä kääntää lausekielisen ohjelman konekielelle

Ohjelman tehokkuus

- osia ohjelmasta ohjelmoidaan suoraan konekielellä

Aloittelevat ohjelmoijat väittävät joskus, että konekielen opiskelu on turhaa, koska kaikki ohjelmat voi kuitenkin toteuttaa korkean tason kielten avulla. Tämäkin ajatus on väärin, mutta konekielen ymmärtämiseen ja käyttöön on myös muita paljon tärkeämpiä perusteluja kuin pelkkä ohjelmointinäkökulma. Tietokone ihan oikeasti toimii vain konekäskyjä suorittaen, joten jokaisen tietotekniikan ammattilaisen tulee ymmärtää ohjelmien suoritus tällä tasolla. Suoritusvaiheessa ohjelmien alkuperäisellä kuvaustavalla korkean tason kielen avulla ei ole merkitystä. Ymmärtääksemme ohjelman suoritusta meidän tulee nimenomaan ajatella konekäskytason suoritusta. Ohjelmia on helppo suunnitella ja toteuttaa korkean tason kielien avulla, mutta suoritus tapahtuu aina konekielen tasolla.

Miksi konekieltä?

Koneen toiminnan ymmärtäminen

Oman ohjelman toiminnan ymmärtäminen

Koneenläheinen ohjelmointi (konekielinen ohjelmointi)

Kääntäjän tekeminen

- kääntäjä kääntää lausekielisen ohjelman konekielelle

Ohjelman tehokkuus

- osia ohjelmasta ohjelmoidaan suoraan konekielellä

Joissakin tapauksissa, esimerkiksi yhä yleisempien pienten sulautettujen laitteiden kanssa on mahdollista, että ohjelmointi tapahtuu suoraan konekielellä. Yleensä näin ei kuitenkaan tehdä, koska korkean tason ohjelmointikielet tarjoavat parempia apuvälineitä ohjelmien toteuttamiseen. Yhä edelleen on kuitenkin mahdollista, että jokin isonkin ohjelmiston kriittinen laiteläheinen osa toteutetaan konekielellä. Korkean tason ohjelmointikielet on suunniteltu yleiseen ongelmanratkaisuun, eikä spesifiseen laitteistoon. Uusien oheislaitteiden laiteajurit vaativat myös ainakin osittain yleensä konekielitason toteutusta.

Miksi konekieltä?

Koneen toiminnan ymmärtäminen

Oman ohjelman toiminnan ymmärtäminen

Koneenläheinen ohjelmointi (konekielinen ohjelmointi)

Kääntäjän tekeminen
- kääntäjä kääntää lausekielisen ohjelman konekielelle

Ohjelman tehokkuus
- osia ohjelmasta ohjelmoidaan suoraan konekielellä

Kääntäjien toteutuksessa tarvitaan aina konekieltä. Kääntäjän viimeinen vaihe (koodin generointi) luo konekielistä koodia alkuperäisestä korkean tason kielisestä ohjelmasta, joten kääntäjän koodin generointivaiheen toteuttajan tulee tietenkin osata kohdelaitteiston konekieltä hyvin. Aikaisemmin riitti ymmärtää kohdelaitteiston käskykanta-arkkitehtuuri, mutta nykyisten suorittimien ohjelmointi vaatii velä tarkempaa laitteiston ymmärtämistä, kuin mitä ainoastaan konekäskyjen toiminnallisuus antaa. Esimerkiksi, koodin generoijan tulee ymmärtää mitä suorittimen sisäisiä resursseja kukin konekäsky käyttää, milloin niitä käytetään ja kuinka kauan käyttö kestää. Nämä asiat kuuluva: jo tosin jatkokurssille.

Miksi konekieltä?

Koneen toiminnan ymmärtäminen

Oman ohjelman toiminnan ymmärtäminen

Koneenläheinen ohjelmointi (konekielinen ohjelmointi)

Kääntäjän tekeminen

- kääntäjä kääntää lausekielisen ohjelman konekielelle

Ohjelman tehokkuus

- osia ohjelmasta ohjelmoidaan suoraan konekielellä

Aikaisemmin, muutama vuosikymmen sitten, ohjelmien suorituskyvyn eli suoritusnopeuden parantaminen oli tärkeä peruste ohjelman osan toteuttamiseen konekielellä. Nykyään tätä ei yleensä tarvita, koska aikakriittisten ohjelmien toteuttamisessa käytetyt kääntäjät ovat niin hyviä koodin optimoinnissa eli nopeasti suoritettavan koodin generoinnissa. Esimerkiksi, numeerisessa laskennassa käytetyn Fortran-kääntäjän päihittämiseen tarvitaan todella hyvä ohjelmoija, joka ymmärtää täysin kohdelaitteiston sisäisen arkkitehtuurin. Toisaalta, ohjelmoijan tulee edelleenkin ymmärtää konekielitason tapahtumat laitteistossa, jotta hänen ohjelmansa olisi helposti kääntäjän optimoitavissa.

Miksi ei oikeata konekieltä?

Oikeat kielet huomattavasti monimutkaisempia
- niiden opiskeluun tarvitaan oma kurssi

Vaikea valita sopivinta
- paljon erilaisia konekieliä

Keskitytään vain opetuksen kannalta oleellisiin asioihin
- tarvittaessa oikea konekieli "helppo" oppia
  (ainakin, jos ensin osaa jotain konekieltä, vaikkapa opetustarkoitukseen suunniteltua helppoa konekieltä)

Miksi sitten emme opiskelisi todellista konekieltä sen sijaan, että käytämme olemattoman koneen konekieltä? Vastaus on helppo. Oikeat suorittimet ovat huomattavasti monimutkaisempia kuin ttk-91 ja niiden toiminnan kuvaamiseen menisi paljon enemmän aikaa, kuin mitä meillä on käytettävissä tämän kurssin osalta. Vastaavasti oikeiden suorittimien konekielen opiskelu vaatisi ihan oman koko lukukauden kestävän kurssinsa. Todellisen konekielen oppiminen ei vastaa tämän kurssin tavoitteita, koska tarkoituksena on tutustua konekieleen ainoastaan välineenä tietokoneen perustoimintojen kuvaamisessa.

Miksi ei oikeata konekieltä?

Oikeat kielet huomattavasti monimutkaisempia
- niiden opiskeluun tarvitaan oma kurssi

Vaikea valita sopivinta
- paljon erilaisia konekieliä

Keskitytään vain opetuksen kannalta oleellisiin asioihin
- tarvittaessa oikea konekieli "helppo" oppia
  (ainakin, jos ensin osaa jotain konekieltä, vaikkapa opetustarkoitukseen suunniteltua helppoa konekieltä)

Jos kuitenkin käyttäisimme oikeata konekieltä tai vaikkapa vain sen osajoukkoa, niin sopivan kielen valinta olisi silti ongelmallista. Konekieliä on huomattava määrä ja ne ovat hyvin heterogeeninen joukko. Ei siis ole mitään oikein hyvää geneeristä konekieltä, joka olisi edustava näyte, ellei sitten lasketa mukaan juuri opetuskäyttöön suuunniteltuja konekieliä.

Miksi ei oikeata konekieltä?

Oikeat kielet huomattavasti monimutkaisempia
- niiden opiskeluun tarvitaan oma kurssi

Vaikea valita sopivinta
- paljon erilaisia konekieliä

Keskitytään vain opetuksen kannalta oleellisiin asioihin
- tarvittaessa oikea konekieli "helppo" oppia
  (ainakin, jos ensin osaa jotain konekieltä, vaikkapa opetustarkoitukseen suunniteltua helppoa konekieltä)

Opetuskäyttöön suunnitellussa suoritinarkkitehtuurissa ja sen konekielessä on se verraton etu, että siinä keskitytään ainoastaan opetuksen kannalta oleellisiin seikkoihin. Määrittelystä voi jopa puuttua kokonaisuuden kannalta tärkeitä piirteitä, jotka eivät kuitenkaan ole keskeisiä opetettavien asioiden suhteen. Kun arkkitehtuurin ja konekielien perusasiat ovat hallussa, niin todellisten uusien arkkitehtuurien ja konekielien oppiminen on sen jälkeen jos ei nyt ihan triviaalia, niin ainakin kohtuullisen helppoa.

Tietokone ttk-91

Laitteisto, hardware (HW)

- suoritin, muisti, väylät, oheislaitteiden liitännät

Käskykanta - konekieliarkkitehtuuri

- käyttöliittymä laitteistoon
- konekäskyt, tiedon esitysmuodot

Symbolinen konekieli

- luettavampi muoto konekielestä
- kullakin symbolilla yksikäsitteiset arvot (symbolitaulu)

Ttk-91tietokoneen simulaattori

- ttk-91 laitteiston simulaattori
- symbolisen konekielen kääntäjä
- graafinen käyttöliittymä, debugger ympäristö

Ttk-91 tietokoneen laitteisto sisältää ainoastaan peruskomponentit: suorittimen, muistin, niitä yhdistävän väylän ja muutaman oheislaitteen. Ttk-91:stä puuttuu esimerkiksi välimuisti, koska laitteiston toimintaa ei tällä kurssilla käsitellä niin yksityiskohtaisesti. Oheislaitteista on toteutettu ainoastaan kortinlukija (card reader, CR) ja näyttö (Cathode Ray Tube, CRT). Simulaattorissa nämä on toteutettu näppäimistöllä ja käyttöliittymän ikkunoilla. Simulaattorissa Titokone on lisäksi toteutettu myös levymuisti.

Tietokone ttk-91

Laitteisto, hardware (HW)

- suoritin, muisti, väylät, oheislaitteiden liitännät

Käskykanta - konekieliarkkitehtuuri

- käyttöliittymä laitteistoon
- konekäskyt, tiedon esitysmuodot

Symbolinen konekieli

- luettavampi muoto konekielestä
- kullakin symbolilla yksikäsitteiset arvot (symbolitaulu)

Ttk-91tietokoneen simulaattori

- ttk-91 laitteiston simulaattori
- symbolisen konekielen kääntäjä
- graafinen käyttöliittymä, debugger ympäristö

Ttk-91 koneen käskykanta muodostaa käyttöliittymän laitteistoon. Käymme läpi kaikki konekäskyt yksityiskohtaisesti. Kukin konekäsky on hyvin yksinkertainen, jolloin myöskin sen toteuttavat piirit ovat yksinkertaisia. Käymme myös läpi laitteiston ymmärtämät tietotyypit eli selvitämme, minkä tyyppistä tietoa konekäskyt voivat käsitellä. Tyypillisesti jokaiselle tietotyypille on omat konekäskynsä. Tietotyyppeihin liittyy olennaisesti kunkin tietotyypin esitysmuoto eli millä tavoin kyseisen tietotyypin sisältämä tieto koodataan biteiksi.

Tietokone ttk-91

Laitteisto, hardware (HW)

- suoritin, muisti, väylät, oheislaitteiden liitännät

Käskykanta - konekieliarkkitehtuuri

- käyttöliittymä laitteistoon
- konekäskyt, tiedon esitysmuodot

Symbolinen konekieli

- luettavampi muoto konekielestä
- kullakin symbolilla yksikäsitteiset arvot (symbolitaulu)

Ttk-91tietokoneen simulaattori

- ttk-91 laitteiston simulaattori
- symbolisen konekielen kääntäjä
- graafinen käyttöliittymä, debugger ympäristö

Puhtaan konekielen lisäksi esittelemme ttk-91 koneen symbolisen konekielen, jota on huomattavasti kätevämpi meidän ihmisten lukea ja kirjoittaa. Symbolinen konekieli vastaa hyvin suoraviivaisesti puhdasta konekieltä, koska kaikelle symbolisen tiedolle on yksikäsitteinen numeerinen vastaavuus. Symbolisessa konekielessä näyttää olevan useita erilaisia muistiinviittaustapoja, vaikka ne konekielen tasolla onkin toteutettu kaikki samalla tavalla. Ttk-91 koneen symbolisesta konekielestä puuttuu ohjelmointia helpottavat makrot, koska haluamme tuoda kaikki yksityiskohdat esille esimerkkikoodinpätkissä.

Tietokone ttk-91

Laitteisto, hardware (HW)

- suoritin, muisti, väylät, oheislaitteiden liitännät

Käskykanta - konekieliarkkitehtuuri

- käyttöliittymä laitteistoon
- konekäskyt, tiedon esitysmuodot

Symbolinen konekieli

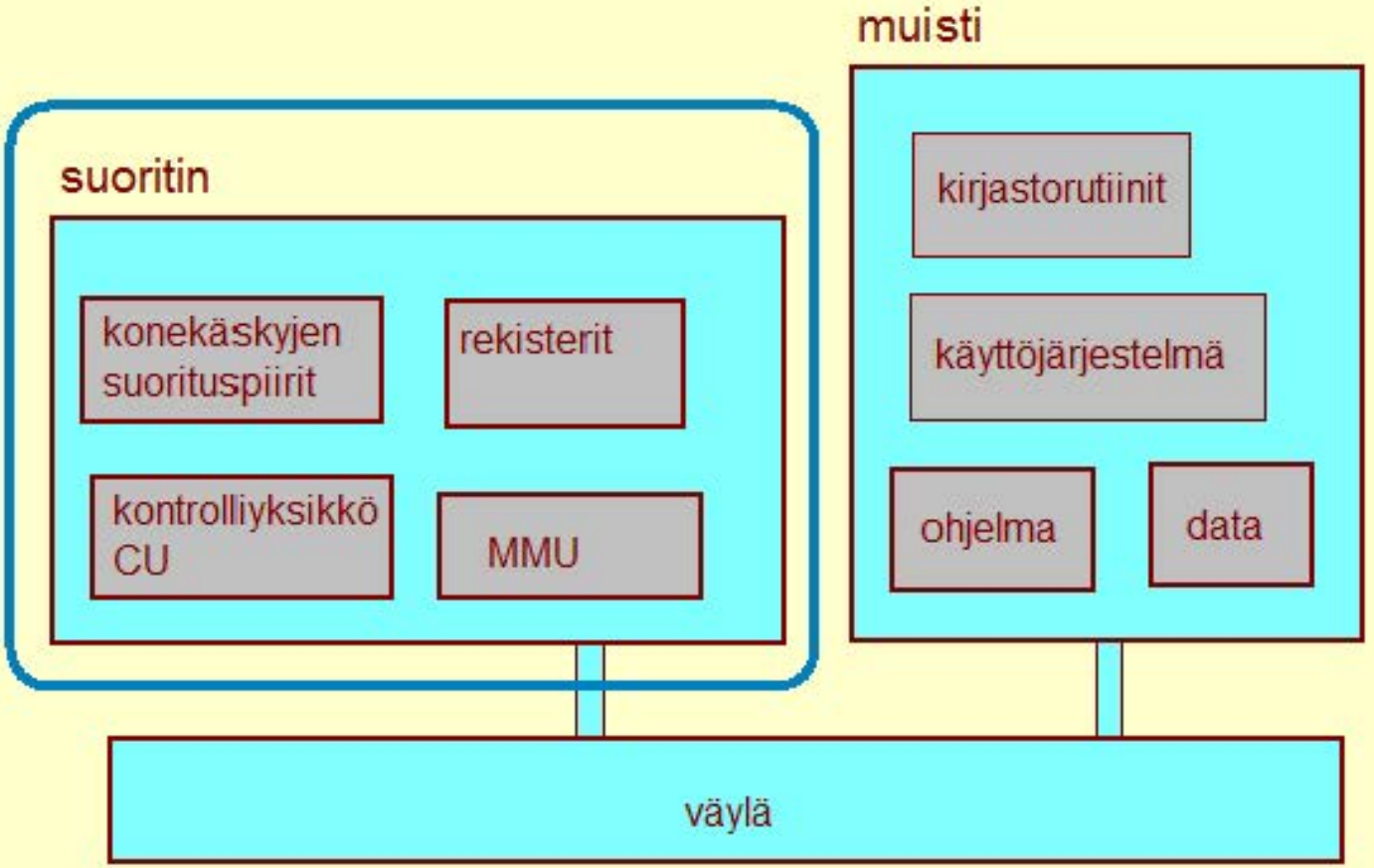
- luettavampi muoto konekielestä
- kullakin symbolilla yksikäsitteiset arvot (symbolitaulu)

Ttk-91tietokoneen simulaattori

- ttk-91 laitteiston simulaattori
- symbolisen konekielen kääntäjä
- graafinen käyttöliittymä, debugger ympäristö

Lopuksi esittelemme ttk-91 järjestelmän simulaattorin Titokone, johon sisältyy paljon muutakin kuin pelkkä tietokoneohjelman suorituksen simulaattori. Järjestelmään kuuluu symbolisen konekielen kääntäjä, ohjelmien kehitysympäristö ja hieno graafinen käyttöliittymä. Ttk-91 ohjelmia voidaan suorittaa laitteistossa esimerkiksi yksi konekäsky kerrallaan siten, että jokaisen käskyn suoritusta animoidaan ja kaikkien rekistereiden ja muistipaikkojen sisältö on tarkasteltavissa joka konekäskyn suorituksen jälkeen.

Ttk-91 laitteisto

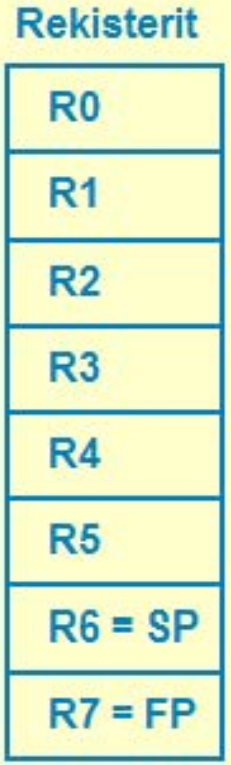


Tarkastelemme aluksi suoritinta. Suorittimen rekisterit sisältävät kaiken laskennan aikaisen datan. Konekäskyissä nimettyjen rekistereiden lisäksi laitteistoon kuuluu muutama suorittimen sisäiseen käyttöön varattu rekisteri. Suorituspiirit tekevät varsinaisen laskentatyön ja muistinhallintayksikkö (Memory Management Unit, MMU) tekee kaikki muistiinviittaukset ja oheislaitteiden ohjaamisen. Kontrolliyksikkö (Control Unit, CU) on tässä työssä 'orkesterinjohtaja', joka jokaisella kellopulssilla kertoo kaikille muille, mitä niiden pitää tehdä.

Ttk-91 rekisterit

8 yleisrekisteriä

- kaikkia rekistereitä voi teoriassa käyttää mihin vain
  (vs. liukulukurekisteriä voi käyttää vain liukulukulaskentaan)
- vain näitä rekistereitä voi koskettaa suoraan konekäskyillä
- kaikki laskenta tapahtuu näiden rekistereiden avulla
- R6 ja R7 varattu aliohjelmien toteuttamiseen: R6=SP, R7=FP
  Pino osoitin (Stack Pointer, SP), Kehysosoitin (Frame Pointer)
- R1-R5 yleisiä työ- ja indeksirekistereitä
  (käyttötapa riippuu rekisterin sijainnista konekäskyssä ja
  muistiinviittaustavasta)



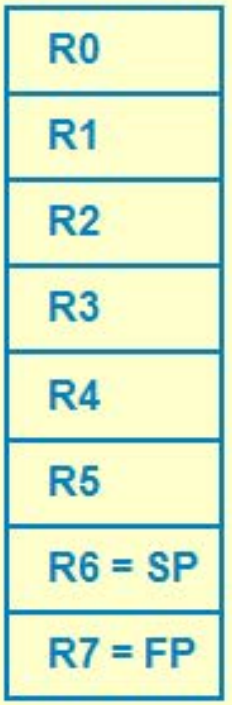
- R0 ainoastaan työrekisteri
  (numero 0 indeksirekisterin kohdalla tarkoittaa, että indeksirekisteriä ei käytetä)

Konekäskyissä voi viitata vain 8 yleisrekisteriin, jotka on nimetty R0-R7. Kaikki laskenta tapahtuu näiden rekistereiden avulla ja laskennan tulos sijoitetaan aina johonkin näistä rekistereistä. Kaikki rekisterit ovat 32 bittisiä. Rekisterit R6 ja R7 on varattu aliohjelmien toteuttamiseen ja niiden käyttö käsitellään myöhemmin. Rekisterit R1-R5 sopivat kaikkeen laskentaan kaikissa tapauksissa. Käyttätapoja on kaksi: tavallisen laskutoimituksen operandi tai tulos, tai sitten muistinviittauksessa käytettävä indeksointi. Käyttötapa selviää siitä, missä kohtaa konekäskyä rekisteri on ja kuinka muistiinviittaus tapahtuu.

Ttk-91 rekisterit

8 yleisrekisteriä

- kaikkia rekistereitä voi teoriassa käyttää mihin vain
  (vs. liukulukurekisteriä voi käyttää vain liukulukulaskentaan)
- vain näitä rekistereitä voi koskettaa suoraan konekäskyillä
- kaikki laskenta tapahtuu näiden rekistereiden avulla
- R6 ja R7 varattu aliohjelmien toteuttamiseen: R6=SP, R7=FP
  Pino osoitin (Stack Pointer, SP), Kehysosoitin (Frame Pointer)
- R1-R5 yleisiä työ- ja indeksirekistereitä
  (käyttötapa riippuu rekisterin sijainnista konekäskyssä ja
  muistiinviittaustavasta)



- R0 ainoastaan työrekisteri
  (numero 0 indeksirekisterin kohdalla tarkoittaa, että indeksirekisteriä ei käytetä)

Rekisteri R0 on erikoistapaus. Indeksirekisterin käyttö konekäskyssä on valinnaista ja käskyssä pitää jotenkin ilmaista indeksirekisterin käyttö. Yksi tapa sen ilmaisemiseen olisi erillinen (1 bitin) indeksirekisterin käyttökenttä, jonka arvo olisi 0, jos indeksirekisteriä ei käytetä. Toinen vaihtoehto (Ttk-91:ssä käytetty) on varata jokin tietty indeksirekisterin numero ilmaisemaan, että indeksirekisteriä ei käytetä. Tietenkään juuri tätä rekisteriä ei sitten voi käyttää indeksirekisterinä. Ttk-91:ssä rekisteri R0 on varattu tähän tarkoitukseen.

Ttk-91 kontrolliyksikkö

Käskynosoitin (Program Counter, PC)
- seuraavaksi suoritettavan käskyn osoite

Käskyrekisteri (Instruction Register, IR)
- suorituksessa oleva konekäsky

Apurekisteri (Temporary Register, TR)
- tilapäinen talletuspaikka käskyn suoritusaikana

Tilarekisteri (State Register, SR)
- suorittimen tila ja rajoitukset tällä hetkellä

Kontrolliyksikössä on neljä sisäistä (32-bittistä) rekisteriä. Käskynosoitinrekisteri eli PC osoittaa aina seuraavaksi suoritettavaan konekäskyyn. Ei siis tällä hetkellä suorituksessa olevaan, vaan seuraavaan. PC osoittaa siis aina johonkin kohtaa suoritettavaa konekielistä ohjelmaa. Konekäskyn suorituksen alussa PC osoittaa nykykäskyä seuraavaan konekäskyyn, koska oletusarvoisesti konekäskyjä suoritetaan muistissa peräkkäisjärjestyksessä. Hyppykäskyjen yhteydessä PC:n arvo kuitenkin vaihdetaan käskyn suoritusaikana, ja seuraava käsky voikin sitten olla muistissa missä päin tahansa.

Ttk-91 kontrolliyksikkö

Käskynosoitin (Program Counter, PC)

- seuraavaksi suoritettavan käskyn osoite

Käskyrekisteri (Instruction Register, IR)

- suorituksessa oleva konekäsky

Apurekisteri (Temporary Register, TR)

- tilapäinen talletuspaikka käskyn suoritusaikana

Tilarekisteri (State Register, SR)

- suorittimen tila ja rajoitukset tällä hetkellä

Käskyrekisterissä IR on tällä hetkellä suorituksessa oleva konekäsky, joka haettiin muistista ennen käskyn suoritusta PC:n osoittamasta paikasta. IR on 32-bittinen, koska kaikki konekäskyt ovat 32 bitin pituisia. IR on toteutettu erikoislaitteistolla, joka on suunniteltu siten, että siinä olevien kenttien tietoja on helppo käyttää. Esimerkiksi, käskyssä oleva 8-bittinen operaatiokoodi (OPER) on suoraan haettavissa käskyrekisteristä.

Ttk-91 kontrolliyksikkö

Käskynosoitin (Program Counter, PC)

- seuraavaksi suoritettavan käskyn osoite

Käskyrekisteri (Instruction Register, IR)

- suorituksessa oleva konekäsky

Apurekisteri (Temporary Register, TR)

- tilapäinen talletuspaikka käskyn suoritusaikana

Tilarekisteri (State Register, SR)

- suorittimen tila ja rajoitukset tällä hetkellä

Apurekisteri TR on yleinen aputila käskyn suorituksen aikana. Jos toinen yhteenlaskukäskyn operandeista on muistissa, niin se pitää hakea muistista johonkin rekisteriin ennen yhteenlaskun suorittamista. Ttk-91 koneessa rekisteriä TR käytetään juuri tähän tarkoitukseen. Todellisissa koneissa on usea suorittimen sisäiseen käyttöön varattu rekisteri, joita kaikkia TR edustaa esimerkkikoneessa. TR:n pituus on 32 bittiä, kuten kaikkien muidenkin rekistereiden.

Ttk-91 kontrolliyksikkö

Käskynosoitin (Program Counter, PC)

- seuraavaksi suoritettavan käskyn osoite

Käskyrekisteri (Instruction Register, IR)

- suorituksessa oleva konekäsky

Apurekisteri (Temporary Register, TR)

- tilapäinen talletuspaikka käskyn suoritusaikana

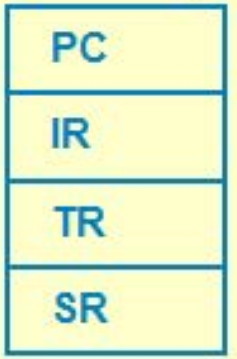
Tilarekisteri (State Register, SR)

- suorittimen tila ja rajoitukset tällä hetkellä

Tilarekisterissä on yleistä tilatietoa laitteistosta eli se kertoo mikä on laitteiston kokonaistila tällä hetkellä. Siellä on myös viimeksi suoritetun vertailun tulos tallennettuna, jotta jokin jäljempänä tuleva konekäsky voi tehdä ehdollisen hypyn aikaisemmin tapahtuneen vertailun perusteella. Todellisissa koneissa vertailujen tulokset talletetaan erilliseen vertailurekisteriin, mutta ttk-91:ssä vertailurekisteri on osa tilarekisteriä. Tilarekisterin pituus on 32 bittiä, vaikka vain osa siitä on nyt käytössä.

Ttk-91 tilarekisteri SR

Kontrolliyksikkö
Control Unit, CU



Tilatietoa siitä, mitä suorittimella tapahtui viimeisen suoritetun käskyn aikana

- virhetilanteet, poikkeukset
- konekäsky olikin käyttöjärjestelmän palvelupyyntö
- vertailun tulos

Tilatietoa siitä, mitä systeemissä tapahtui viime aikoina

- käsittelemättömät laitteiden antamat signaalit
  (laitekeskeytykset, device interrupts)

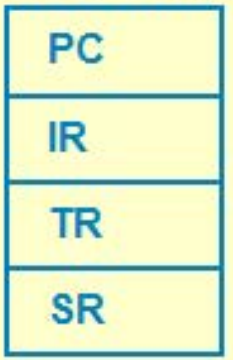
Tilatietoa siitä, mitä suoritin saa tehdä jatkossa

- etuoikeutettu tila: kaikki käskyt OK, kaikki muistialueet OK
- normaali suoritustila: tavalliset käskyt, oma muistialue
- poikkeusten ja keskeytysten käsittely sallittua vai ei?

Tilarekisterin tietoja on kolmenlaisia. Ensinnäkin, siellä on viimeksi suoritetun konekäskyn suorituksen aikana syntynyttä tietoa, kuten esimerkiksi indikaatiot mistä tahansa virhetilanteesta. Käyttöjärjestelmän palvelupyynnöt käsitellään hyvin samalla tavalla kuin virhetilanteetkin, joten myös niiden käyttö koodataan tilarekisteriin samalla tavalla kuin itse virhetilanteetkin. Ttk-91 koneessa myös vertailukäskyjen tulokset (isompi, yhtäsuuri, pienempi) talletetaan tilarekisteriin.

Ttk-91 tilarekisteri SR

Kontrolliyksikkö
Control Unit, CU



Tilatietoa siitä, mitä suorittimella tapahtui viimeisen suoritetun käskyn aikana

- virhetilanteet, poikkeukset
- konekäsky olikin käyttöjärjestelmän palvelupyyntö
- vertailun tulos

Tilatietoa siitä, mitä systeemissä tapahtui viime aikoina

- käsittelemättömät laitteiden antamat signaalit (laitekeskeytykset, device interrupts)

Tilatietoa siitä, mitä suoritin saa tehdä jatkossa

- etuoikeutettu tila: kaikki käskyt OK, kaikki muistialueet OK
- normaali suoritustila: tavalliset käskyt, oma muistialue
- poikkeusten ja keskeytysten käsittely sallittua vai ei?

Toiseksi, tilarekisterissä on tietoa koko järjestelmän lähihistoriasta. Oheislaitteet voivat signaloida tilastaan käyttöjärjestelmälle erityisten laitekeskeytysten avulla ja tilarekisteri pitää kirjaa siitä, mitkä laitekeskeytykset ovat vielä käsittelemättä.

Ttk-91 tilarekisteri SR

Kontrolliyksikkö
Control Unit, CU

PC
IR
TR
SR

Tilatietoa siitä, mitä suorittimella tapahtui viimeisen suoritetun käskyn aikana

- virhetilanteet, poikkeukset
- konekäsky olikin käyttöjärjestelmän palvelupyyntö
- vertailun tulos

Tilatietoa siitä, mitä systeemissä tapahtui viime aikoina

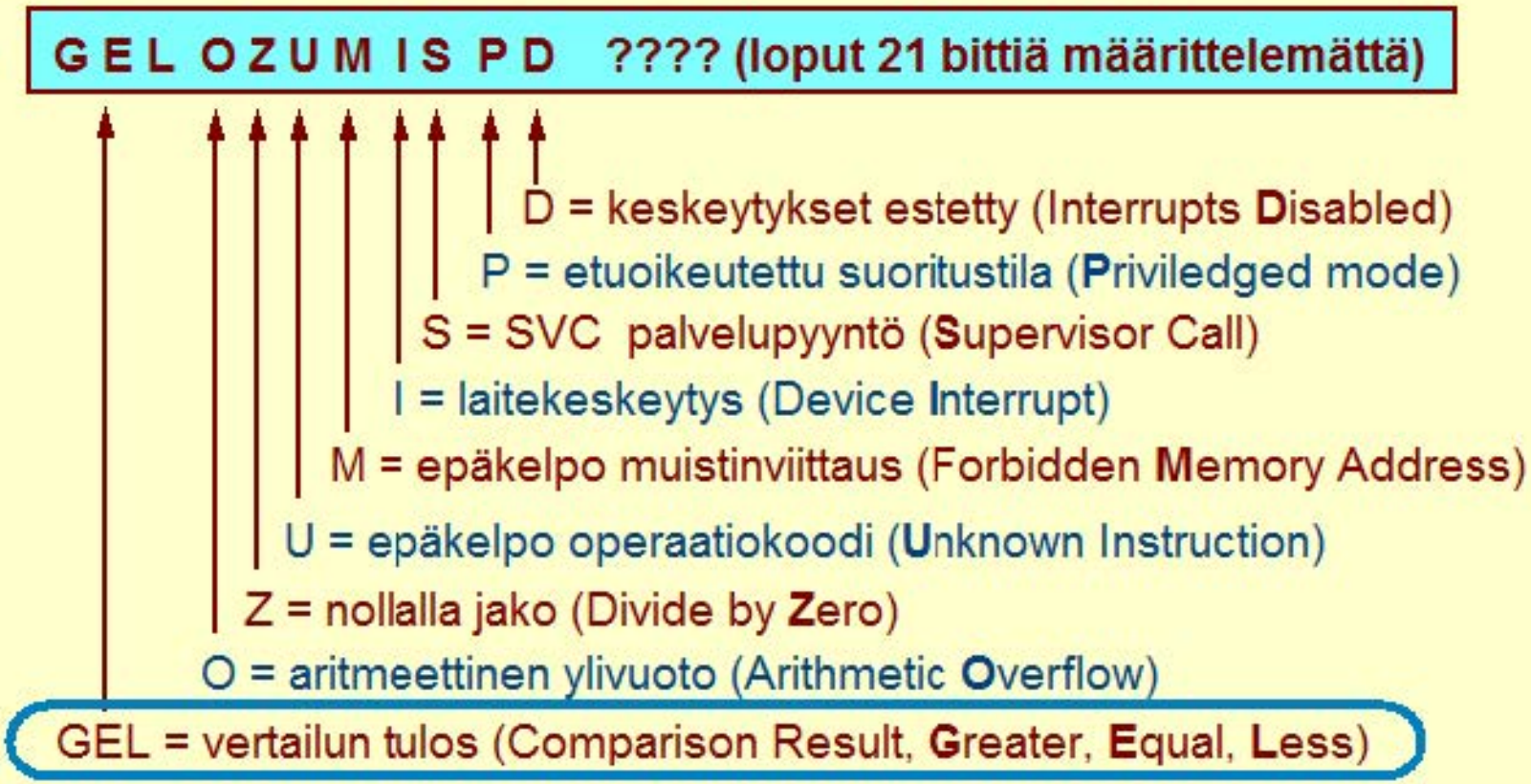
- käsittelemättömät laitteiden antamat signaalit
  (laitekeskeytykset, device interrupts)

Tilatietoa siitä, mitä suoritin saa tehdä jatkossa

- etuoikeutettu tila: kaikki käskyt OK, kaikki muistialueet OK
- normaali suoritustila: tavalliset käskyt, oma muistialue
- poikkeusten ja keskeytysten käsittely sallittua vai ei?

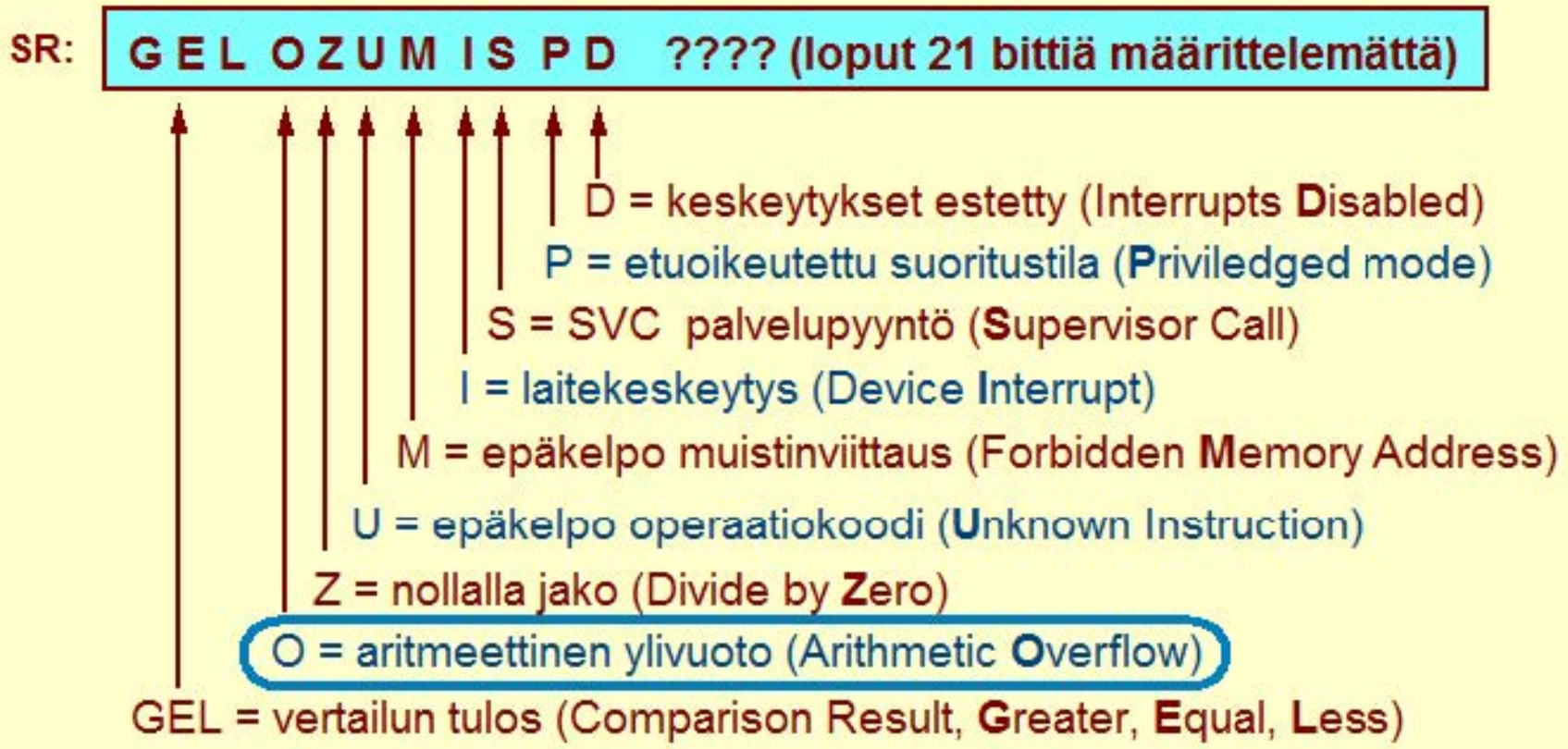
Tämänhetkiset rajoitteet ovat tärkeä osa tilarekisteriä. Suojattujen käyttöjärjestelmien toteutuksen mahdollistaa suorittimen etuoikeutettu tila (supervisor state), jossa suoritin pystyy käyttämään kaikkia konekäskyjä ja viittaamaan mihin tahansa muistialueelle. Normaalitilassa suoritin saa käyttää vain 'turvallisia' käskyjä ja käsitellä vain nykyisen ohjelman omia muistialueita. Esimerkkikone ttk-91 toimii aina normaalitilassa. Samanaikaisuuden aiheuttamien ongelmien ratkaisemiseksi suoritin voi olla lyhyitä aikoja tilassa, jossa mitään keskeytyksiä ei käsitellä. Tällöin olisi parasta, että mitään virheitäkään ei sitten sattuisi!

Tilarekisteri SR



Tilarekisteri on toteutettu 32 bittisenä, mutta vain 11 bittiä on nyt käytössä. Loput on varattu arkkitehtuurin myöhempiä laajennuksia varten. Käytännössä vain 7 ensimmäisellä bitilla on merkitystä, viimeiset neljä ovat mukana esimerkin vuoksi, koska todellisissa koneissa niitä tarvitaan. Ensimmäiset 3 bittiä sisältävät viimeisimmän vertailun tuloksen. Anoastaan yksi biteistä G, E tai L on tosi eli 1 ja muut ovat epätosia eli 0. Seuraava konekäsky voi sitten tutkia vaikkapa, onko bitti E ykkönen vai ei.

Tilarekisteri SR

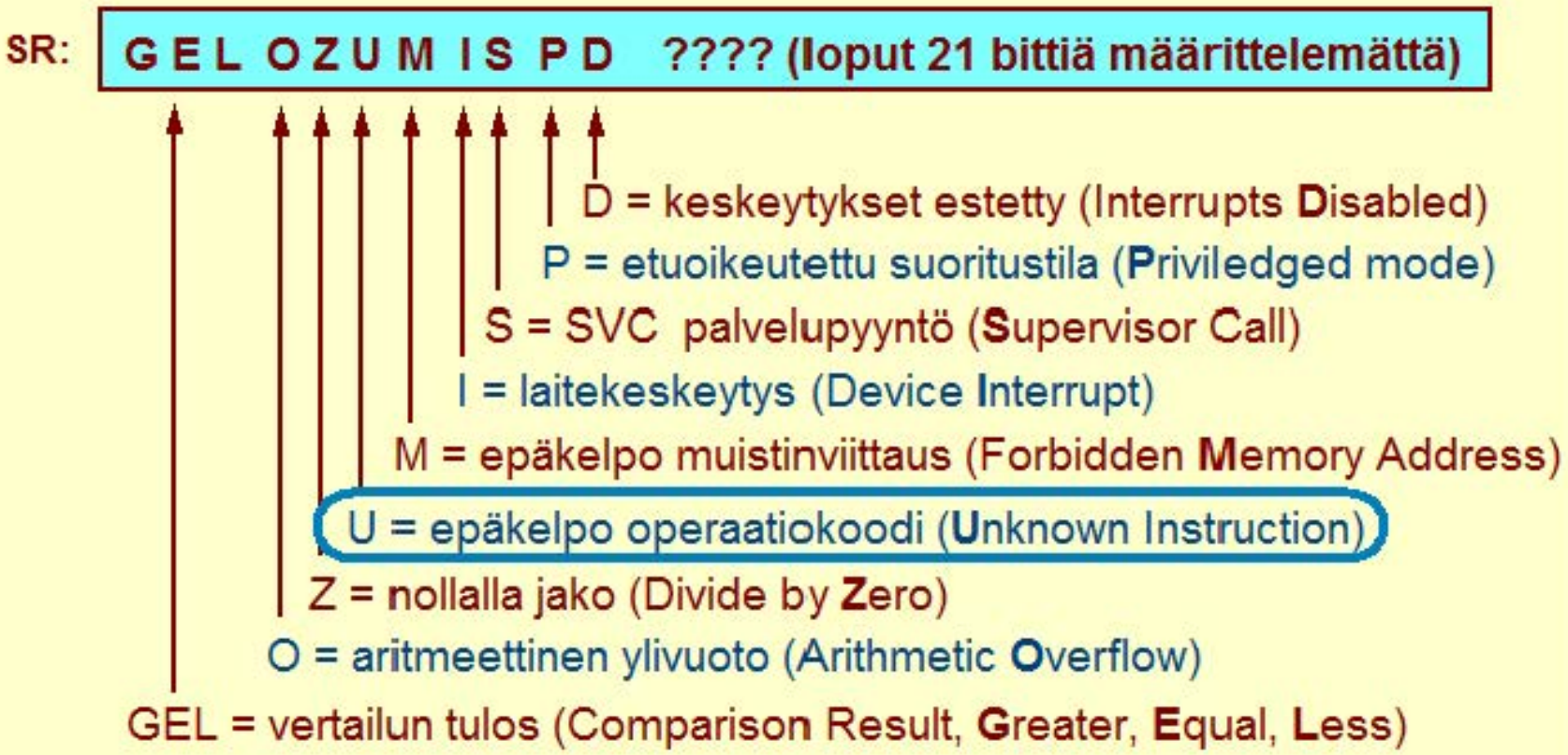


Bitti O asetetaan ykköseksi, jos esimerkiksi yhteenlaskukäskyn tulos on niin suuri, että se ei mahdu 32-bittiseen rekisteriin. Tämä on yleensä virhetilanne ja johtaa ensin käyttöjärjestelmän asiaan puuttumiseen ja sitten meneillään olevan ohjelman suorituksen päättymiseen asianmukaisen virheilmoituksen kera. Joissakin todellisissa arkkitehtuureissa tällaista virhetilannetta ei aina huomioida, jolloin Overflow-bitin asetus on valinnaista. Esimerkiksi Java-ohjelmien suorituksessa ei haluta, että ohjelman suoritus päättyisi tässä tilanteessa virheeseen.

Tilarekisteri SR

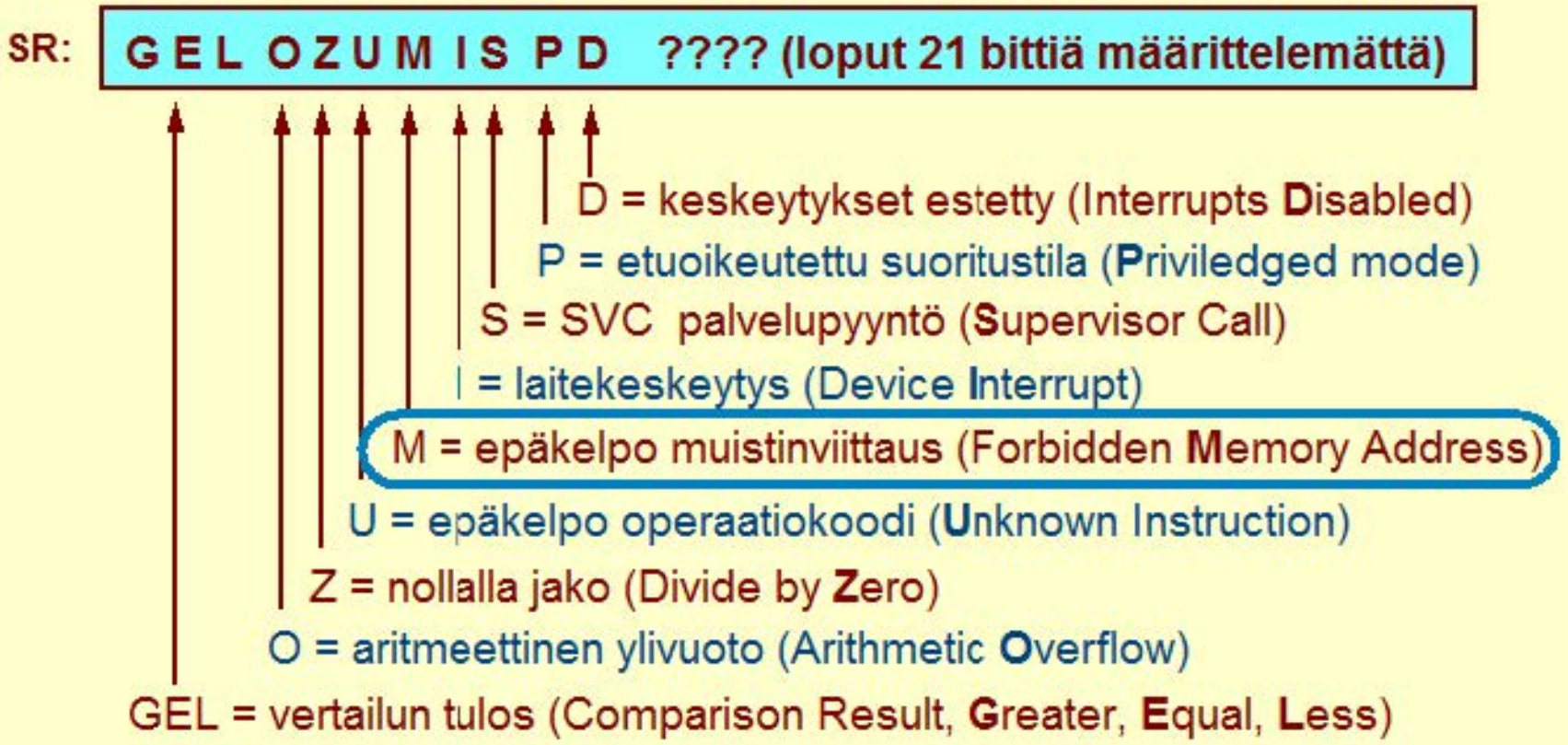
Nollalla jako aiheuttaa myös yleensä virhetilanteen, joka koodataan Z-bitillä tilarekisterissä. Todellisissa koneissa kokonaislukujen nollalla jako on yleensä aina virhetilanne, mutta liukulukujen yhteydessä tuloksena voi olla myös plus tai miinus ääretön! Tästä myöhemmin lisää. Ttk-91 koneessa sekä ylivuoto että nollalla jako ovat aina fataaleja virhetilanteita, jotka johtavat ohjelman suorituksen päättymiseen.

Tilarekisteri SR



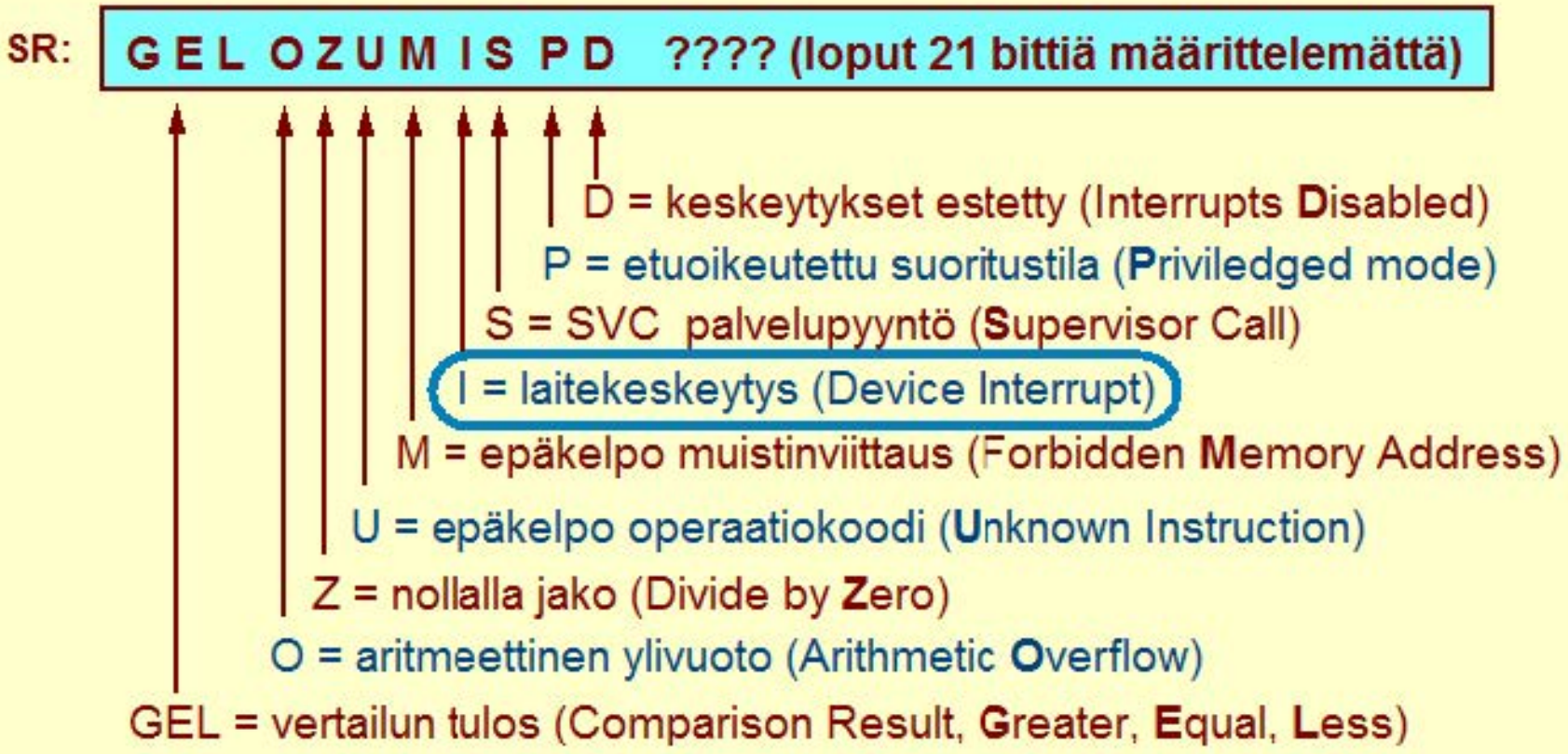
Jos operaatiokoodi on virheellinen, niin se yleensä tarkoittaa, että ohjelman suoritus on siirtynyt koodialueelta data-alueelle. Jos valitaan summittainen data-alkio, niin on hyvin mahdollista, että sen alussa olevat 8 bittiä eivät vastaa mitään käytössä olevaa laillista operaatiokoodia. Esimerkiksi pienien 32-bittisten kokonaislukujen kohdalla ensimmäiset 8 bittiä ovat nollia, jolloin niitä vastaava operaatiokoodi 0 olisi epäkelpo.

Tilarekisteri SR



Jos ohjelma yrittää viitata muistialueelle, jota ei ole tai johon se ei saa viitata, niin suoritus keskeytyy epäkelpoon muistiosoitteeseen. Esimerkiksi, muistiosoitteesta -43 on aika vaikea hakea seuraavaa käskyä tai yhteenlaskussa tarvittavaa arvoa. Luvussa -43 ei sinällään ole vikaa, kunhan sitä ei vain käytetä muistin osoittamiseen.

Tilarekisteri SR



Suorittimen ulkopuolelta tuleva laitekeskeytys koodataan ttk-91 koneessa yhden bitin (I-bitti) avulla. Käyttöjärjestelmän keskeytyskäsittelijä sitten selvittää tarkemmin, mikä laite on kyseessä. I-bitti on ttk-91 koneen tilarekisterissä ihan periaatteen vuoksi - käytetyissä esimerkeissä sitä ei käytetä. Todellisissa koneissa on usein useita keskeytysbittejä - yksi kutakin laitetyyppiä varten.

Tilarekisteri SR

SVC konekäsky näyttää tavalliselta konekäskyltä, mutta sen avulla kutsutaan etuoikeutettua käyttöjärjestelmärutiinia. Kutsu on hyvin saman kaltainen kuin keskeytysten käsittelyn aloitus, joten kutsun toteutus on tehty samalla tavalla keskeytysten käsittelyn kanssa. Ttk-91 koneessa tärkein SVC-komento on 'SVC SP,=HALT', jonka avulla tämän ohjelman suoritus lopetetaan normaalisti.

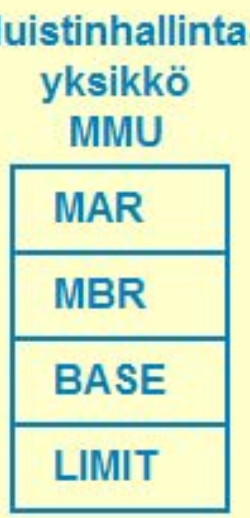
Tilarekisteri SR

Suorittimen ollessa etuoikeutetussa tilassa P-bitin arvo on 1. Todellisissa koneissa on muutamia etuoikeutettuja käskyjä, joita voidaan suorittaa vain tässä tilassa. Välimuistin tyhjennyskäsky tai keskeytyskäsittelyrutiinista paluukäsky ovat hyviä esimerkkejä konekäskyistä, joita ei tarvita tavallisissa sovelluksissa. Luotettavan käyttöjärjestelmän toteuttamisessa ne ovat kuitenkin olennaisia. Esimerkkiohjelmissa P-bitti on aina nolla.

Tilarekisteri SR

Käyttöjärjestelmän toteutuksessa tarvitaan joskus tilanteita, joissa tietty lyhyehkö koodinpätkä täytyy varmasti voida suorittaa ilman mitään keskeytyksiä. Tämä vastaa vähän tilannetta, jossa organisaation johtaja ilmoittaa, että 'hänellä on tärkeä asia kesken, älkää häiritkö!'. Keskeytysten estäminen on yksi menetelmä samanaikaisuuden aiheuttamien ongelmien ratkaisemiseen käyttöjärjestelmien toteuttamisessa. Esimerkkiohjelmat toimivat aina keskeytykset sallittuna eli D==0.

TTK-91 muistinhallintayksikkö (Memory Management Unit, MMU)

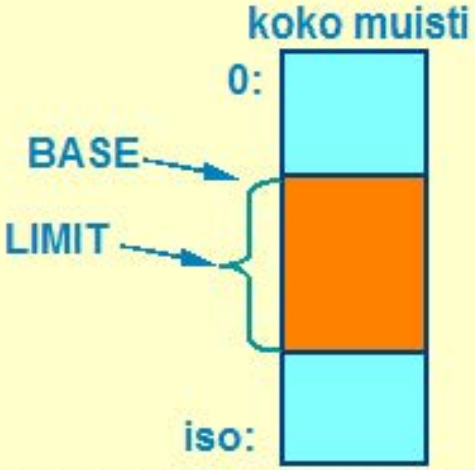


Muistiinviittausrekisterit

- MAR - muistisoiterekisteri (Memory Address Register)
- MBR - muistin puskurirekisteri (Memory Buffer Register)
  sisältää muistista luettavan/muistiin kirjoitettavan arvon

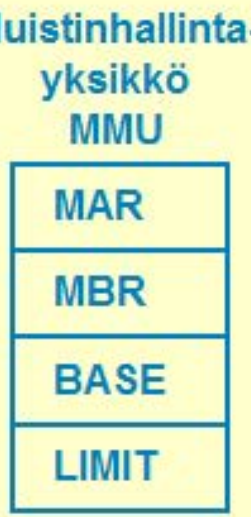
Ohjelman käytössä oleva muistialue (muistisegmentti)

- vain tähän muistialueeseen voi (normaalitilassa) viitata
- rajarekisteripari {BASE, LIMIT}
- BASE - muistialueen alkuosoite
- LIMIT - muistialueen koko
- kaikki ohjelmassa olevat muistiosoitteet ovat suhteellisia BASE-rekisterin arvoon
- Käyttöjärjestelmä valvoo ja asettaa (etuoikeutetussa tilassa, etuoikeutetuilla käskyillä)



Muistinhallintayksikössä on neljä sisäistä rekisteriä. Muistista lukua varten muistiosoite laitetaan MAR'iin ja muistista luettu arvo sitten löytyy aikanaan MBR'stä. Muistiin kirjoitettaessa kopioidaan arvo ensin MBR'ään, osoite MAR'iin ja vähän ajan kuluttua MBR'ssä oleva arvo on sitten kirjoitettu MAR'in osoittamaan muistipaikkaan. Ainoastaan konekäskyjen suorituspiirit käyttävät näitä rekistereitä - ne ei ole mitenkään viitattavissa itse konekäskyissä.

TTK-91 muistinhallintayksikkö (Memory Management Unit, MMU)

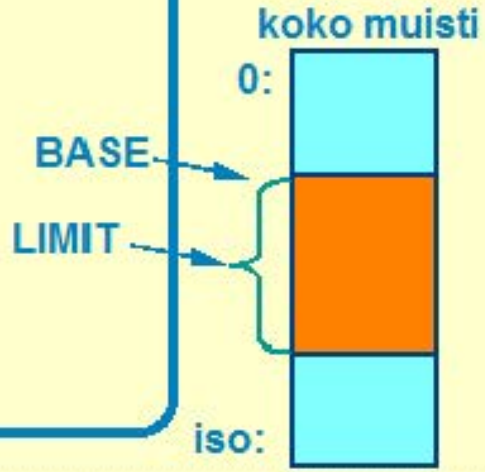


Muistiinviittausrekisterit

- MAR - muistisoiterekisteri (Memory Address Register)
- MBR - muistin puskurirekisteri (Memory Buffer Register)
  sisältää muistista luettavan/muistiin kirjoitettavan arvon

Ohjelman käytössä oleva muistialue (muistisegmentti)

- vain tähän muistialueeseen voi (normaalitilassa) viitata
- rajarekisteripari {BASE, LIMIT}
- BASE - muistialueen alkuosoite
- LIMIT - muistialueen koko
- kaikki ohjelmassa olevat muistiosoitteet ovat suhteellisia BASE-rekisterin arvoon
- Käyttöjärjestelmä valvoo ja asettaa (etuoikeutetussa tilassa, etuoikeutetuilla käskyillä)



Yksi ohjelma ei voi käyttää koko muistia. Ohjelmalle varattu muistialue ilmaistaan ttk-91 koneessa rajarekisteriparilla BASE ja LIMIT. BASE rekisteri ilmaisee käytössä olevan muistialueen alun ja LIMIT sen pituuden. Käskyissä olevat muistiosoitteet ovat nyt kaikki suhteellisia BASE-arvoon ja alkavat siis nollasta. Esimerkiksi, kun ohjelma lukee muistiosoitteestaan 43 jonkin luvun, niin MMU ensin tarkistaa, että 43 on pienempi kuin LIMIT ja vähintään nolla, ja sitten suorittaa itse muistista luvun fyysisen muistin osoitteesta BASE+43.

Ttk-91 käskykannan rakenne

Tietotyypit

Konekäskyjen tyypit

Konekäskyn rakenne

- montako bittiä, minkälaiset kentät

Muistissa olevan tiedon osoitustavat

- konekielessä
- symbolisessa konekielessä

Operaatiot

Pseudo-operaatiot

Minkä tahansa tietokoneen käskykantaan liittyy usea rakenteellinen piirre. Tietotyypit määrittelevät, minkälaisen tiedon käsittelyyn on omia konekäskyjä. Konekäskyn rakenne kuvaa bittitasolla konekäskyn sisäisen rakenteen: mitä kenttiä käskyssä on ja kuinka monta bittiä kukin kenttä on. Tärkeä piirre on muistin osoitustavat. Jos tapoja on useita, niin konetta on helpompi käyttää. Kaikki operaatiot toimintoineen pitää tietenkin esitellä ja lisäksi käskykantaan mielletään kuuluvaksi symbolisen konekielen kääntäjän ohjauskäskyt eli pseudokäskyt.

Ttk-91 koneen tietotyypit

32-bittinen kokonaisluku

- noin 10-numeroinen kokonaisluku

65432 +87654321 -32

Ei:

- pitkiä 64-bitin kokonaislukuja
- lyhyitä 16-bitin kokonaislukuja
- liukulukuja
- merkkejä tai merkkijonoja
- totuusarvoja
- kuvia, ääniä, ...

-43200011123456
-32 (16-bit)
65.125
'a' 'c' Teemu
TRUE FALSE

Ainoa ttk-91 koneen tietotyyppi on 32-bittinen kokonaisluku. Tämä helpottaa koneen rakennetta huomattavasti, mutta rajoittaa jonkin verran sen käyttökelpoisuutta. Esimerkiksi, ttk-91'llä ei voi tehdä 'Hello World' -ohjelmaa, joka on tyypillinen minkä tahansa järjestelmän esimerkkiohjelma. Sitä ei voi toteuttaa, koska ttk-91'ssä ei ole merkkejä tai merkkijonoja. Koneelle voi syöttää ja sillä voi käsitellä ja tulostaa ainoastaan kokonaislukuja. Useimmissa oikeassa koneissa on myös muita tietotyyppejä, esimerkiksi liukulukuja.

TTK-91 koneen käskytyypit

Vain yksi muoto, käsky aina 32 bittiä

NOP-käskyllä voi olla osoite ja sen suoritukseen kuluu aikaa!

Aina kaksi rekisteriä nimetty käskyssä

- niitä ei aina käytetä, vaikka ne ovatkin käskyssä
- esim. NOP-käsky ei tee mitään, eikä käytä operandeja

ADD R1, R2

JUMP here
NOP

Ensimmäinen operandi aina rekisterissä

Toinen operandi rekisterissä, konekäskyssä tai muistissa

- luku rekisteristä/konekäskystä nopeampaa kuin muistista
- muistista arvo haetaan aina ensin sisäiseen TR-rekisteriin

ADD R1, x

ADD R2, =5

ALU-operaation tulos aina 1. operandin paikalle rekisteriin

- korvaa 1. operandin arvon, tuhoaa 1. operandin

ALU (Arithmethical Logical Unit) eli aritmeettis-looginen yksikkö toteuttaa lähes kaikki varsinaista laskentaa tekevät operaatiot suorittimella.

Ttk-91 koneen käskyillä on aina sama muoto. Todellisilla koneilla voi olla useammankin muotoisia käskyjä, mutta on myös sellaisia, joilla on vain yksi muoto. Joka käskyssä on nimetty kaksi rekisteriä, vaikka esimerkiksi NOP-käsky ei käytä niistä kumpaakaan. Laitteisto on helpompi toteuttaa, kun kaikki käskyt ovat saman muotoisia. Ne on myös helppo hakea muistista, kun kaikki käskyt ovat saman muotoisia ja siten myös saman mittaisia.

TTK-91 koneen käskytyypit

Vain yksi muoto, käsky aina 32 bittiä

Aina kaksi rekisteriä nimetty käskyssä

- niitä ei aina käytetä, vaikka ne ovatkin käskyssä
- esim. NOP-käsky ei tee mitään, eikä käytä operandeja

```
ADD  R1, R2

JUMP  here
NOP
```

Ensimmäinen operandi aina rekisterissä

Toinen operandi rekisterissä, konekäskyssä tai muistissa

- luku rekisteristä/konekäskystä nopeampaa kuin muistista
- muistista arvo haetaan aina ensin sisäiseen TR-rekisteriin

```
ADD R1, x
ADD  R2, =5
```

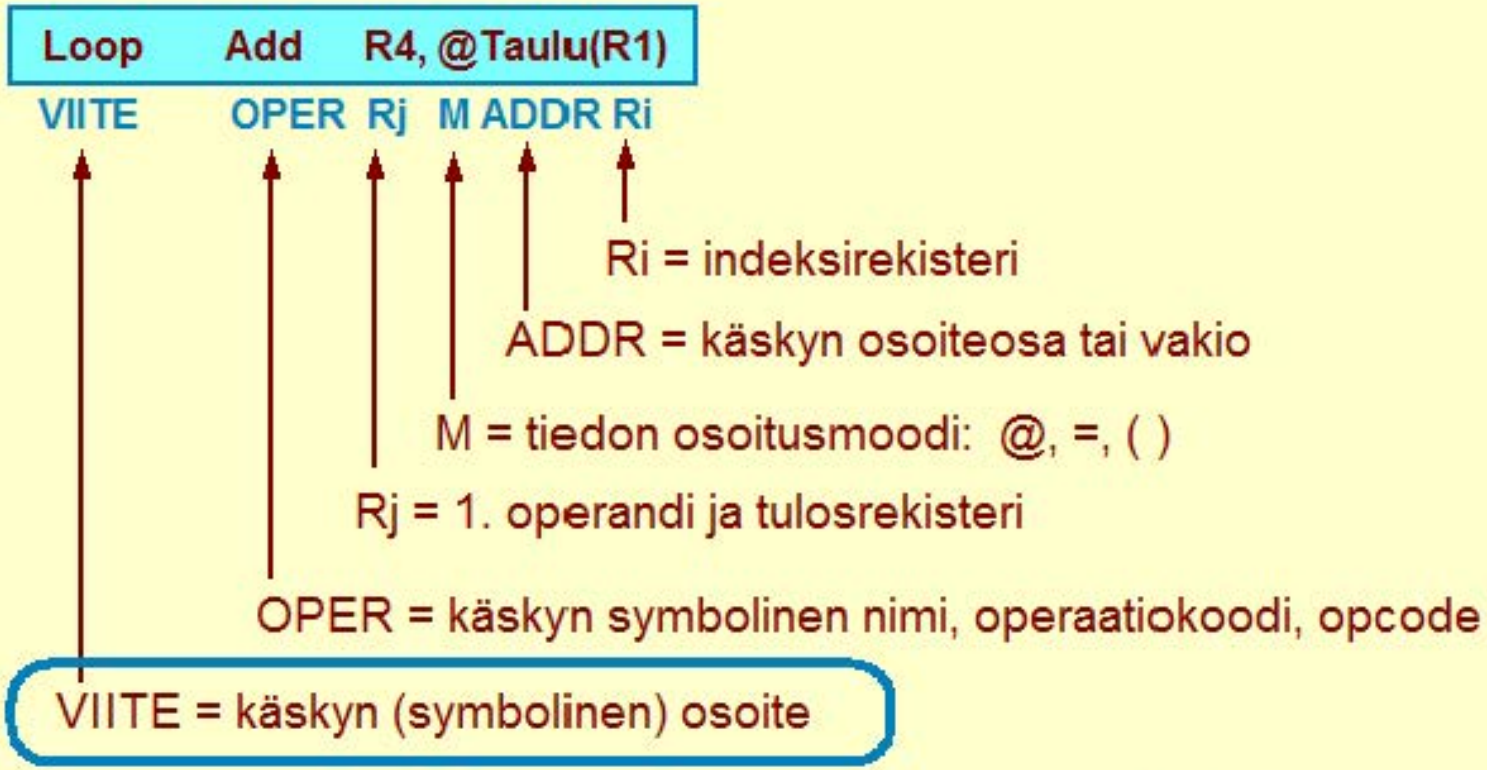
ALU-operaation tulos aina 1. operandin paikalle rekisteriin

- korvaa 1. operandin arvon, tuhoaa 1. operandin

ALU (Arithmethical Logical Unit) eli aritmeettis-looginen yksikkö toteuttaa lähes kaikki varsinaista laskentaa tekevät operaatiot suorittimella.

Käskyissä on yleensä kaksi operandia, joista ensimmäinen on aina käskyn ensiksi mainitussa rekisterissä. Toinen operandi voi löytyä eri paikoista tai se voidaan jopa saada laskemalla yhteen käskyssä mainittu jälkimmäinen rekisteri ja käskyssä oleva pieni vakio. Toinen operandi voi myös olla suoraan käskyssä oleva jälkimmäinen rekisteri tai käskyssä usealla eri tavalla viitattu muistipaikka. ALU-operaatioiden tulos talletetaan aina käskyn ensiksi mainittuun rekisteriin, jonka alkuperäinen arvo siten tuhoutuu. Tämä pitää pitää mielessä ohjelmoitaessa.

Ttk-91 koneen symbolinen konekäsky



Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

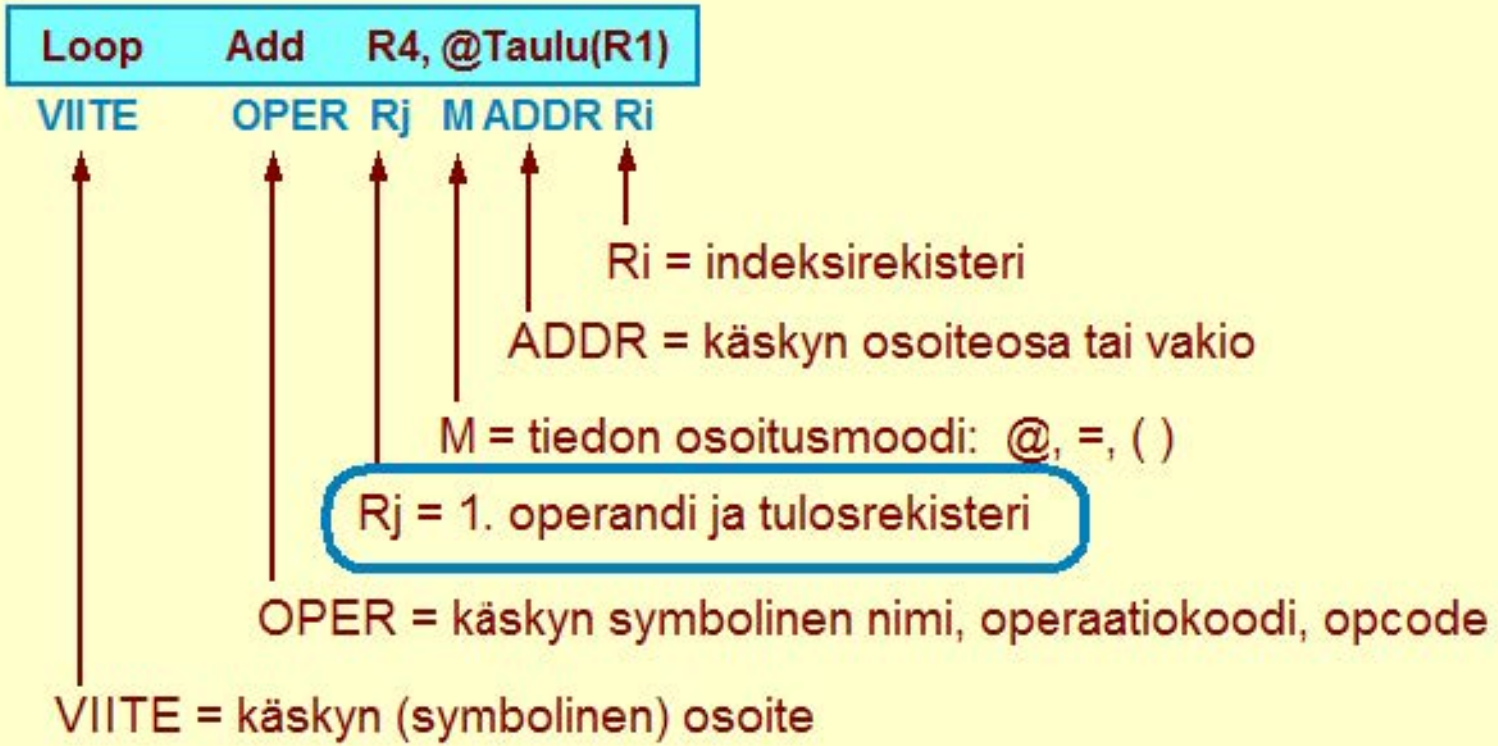
Symbolista konekieltä käyttäen konekäskyt voi kirjoittaa aika hyvin ihmisen luettavaan muotoon. Kaikki tieto voi olla symbolista, mutta myös puhdasta numeerista tietoa voi käyttää. VIITE-kenttä sisältää tämän konekäskyn osoitteen symbolina. Viite-symbolien pitää kaikkien olla erilaisia, uniikkeja ja mitään varattuja sanoja ei saa käyttää. Viite-kenttä voi myös puuttua ja usemmiten se puuttuukin.

Ttk-91 koneen symbolinen konekäsky

Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

Operaatiokoodi ilmaisee, mikä konekäsky on kyseessä. Operaatiokoodit on kaikki määritelty etukäteen ja vain sallittuja symbolisia koodeja voi käyttää. Operaatiokoodit kuten kaikki muutkin symbolit voi kirjoittaa sekä pienillä että isoilla kirjamilla.

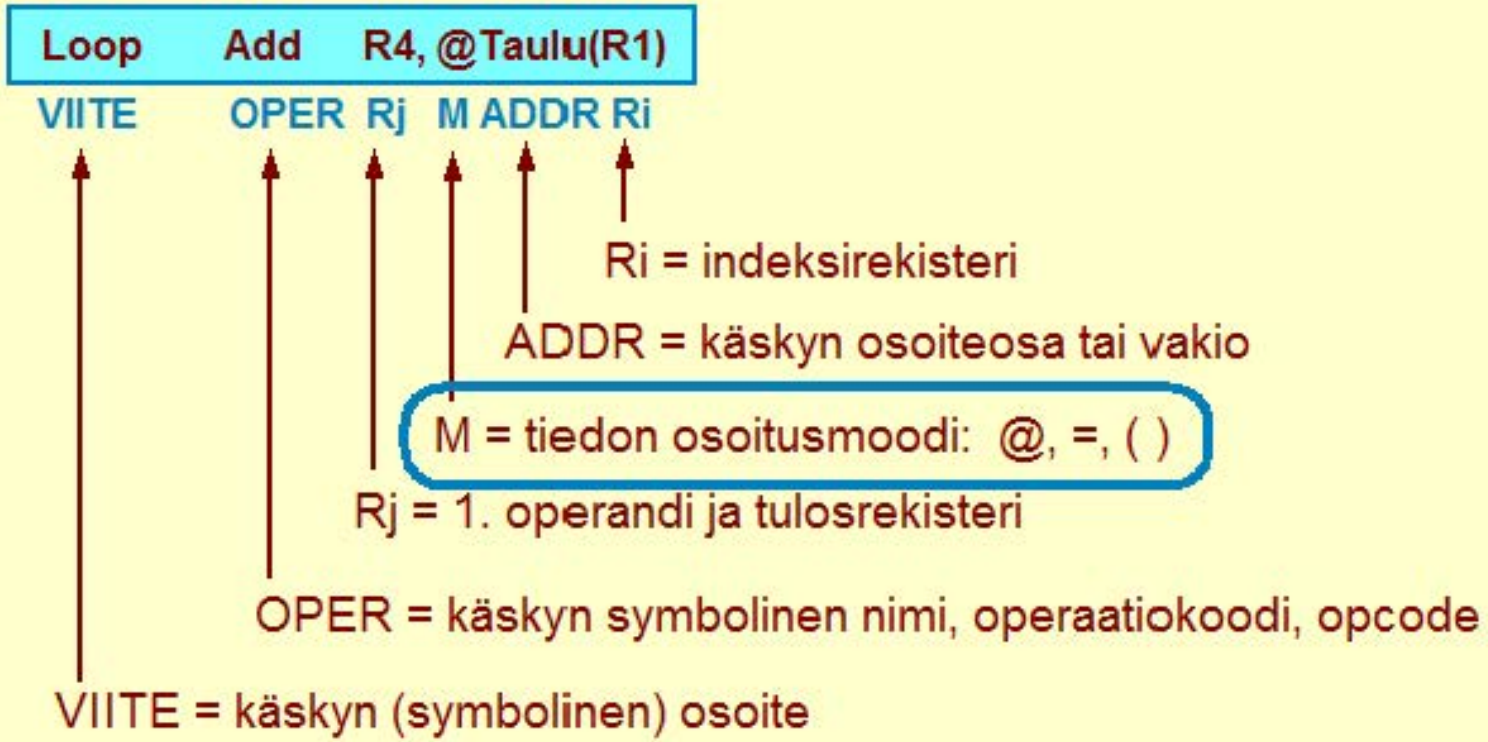
Ttk-91 koneen symbolinen konekäsky



Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

Ensiksi mainittu rekisteri on Rj-kertässä. Tämä on ALU-operaatioiden ensimmäinen operandi ja niiden tulosrekisteri. Sallittuja rekistereiden nimiä ovat R0-R7 ja erikoistapauksira SP ja FP, jotka kuitenkin ovat synonyymejä rekistereille R6 ja R7.

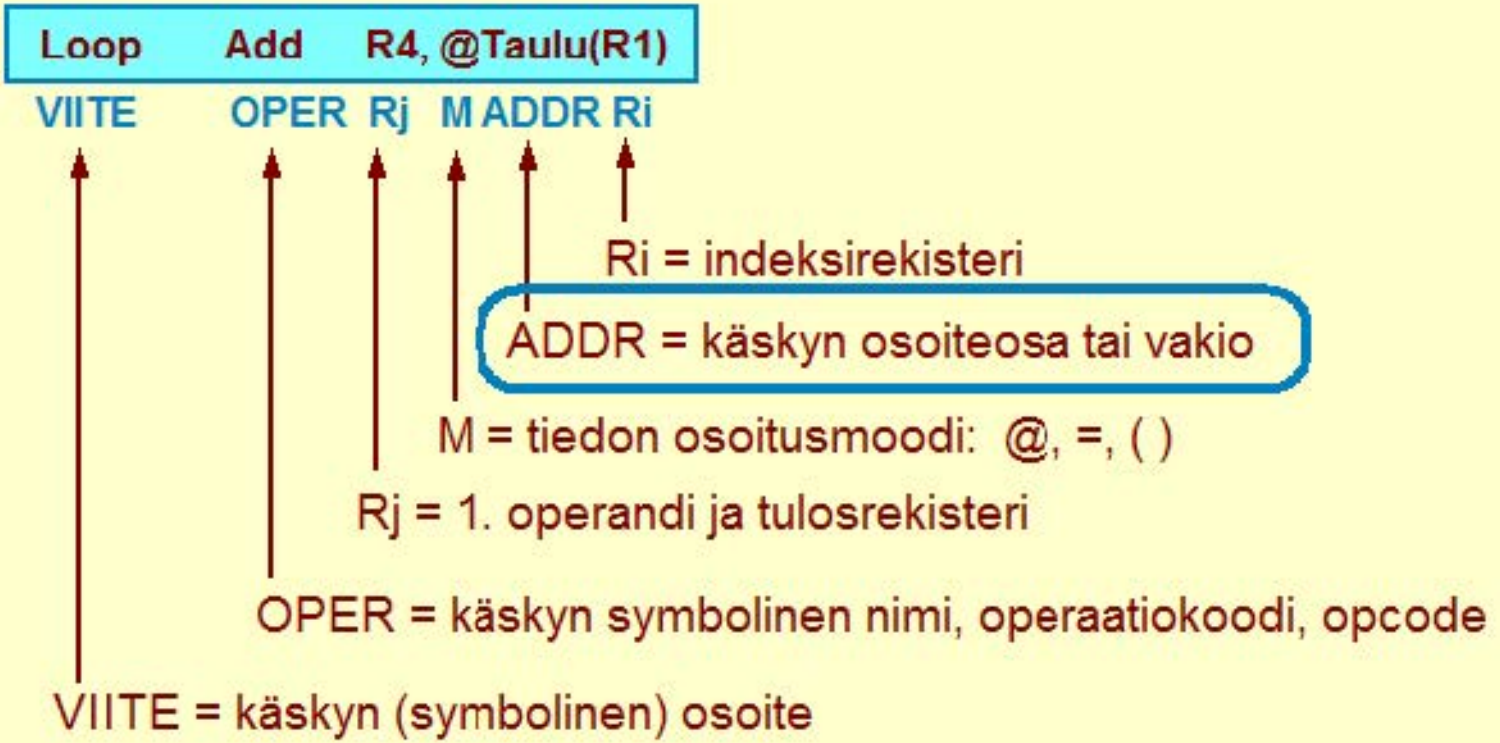
Ttk-91 koneen symbolinen konekäsky



Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

Tiedonosoitusmoodi koostuu symbolisessa konekielessä itse asiassa kahdesta kentästä, ennen vakio-osaa olevasta = tai @-merkistä (tai niiden puuttumisesta) ja indeksirekisterin ympärillä mahdollisesti olevista sulkumerkeistä. Täsmennämme tiedonosoitusmoodin käyttöä myöhemmin. Sen avulla kuitenkin määritellään, miten ja mistä jälkimmäinen operandi löytyy käskyn vakio-osan ja jälkimmäisen rekisterin Ri avulla.

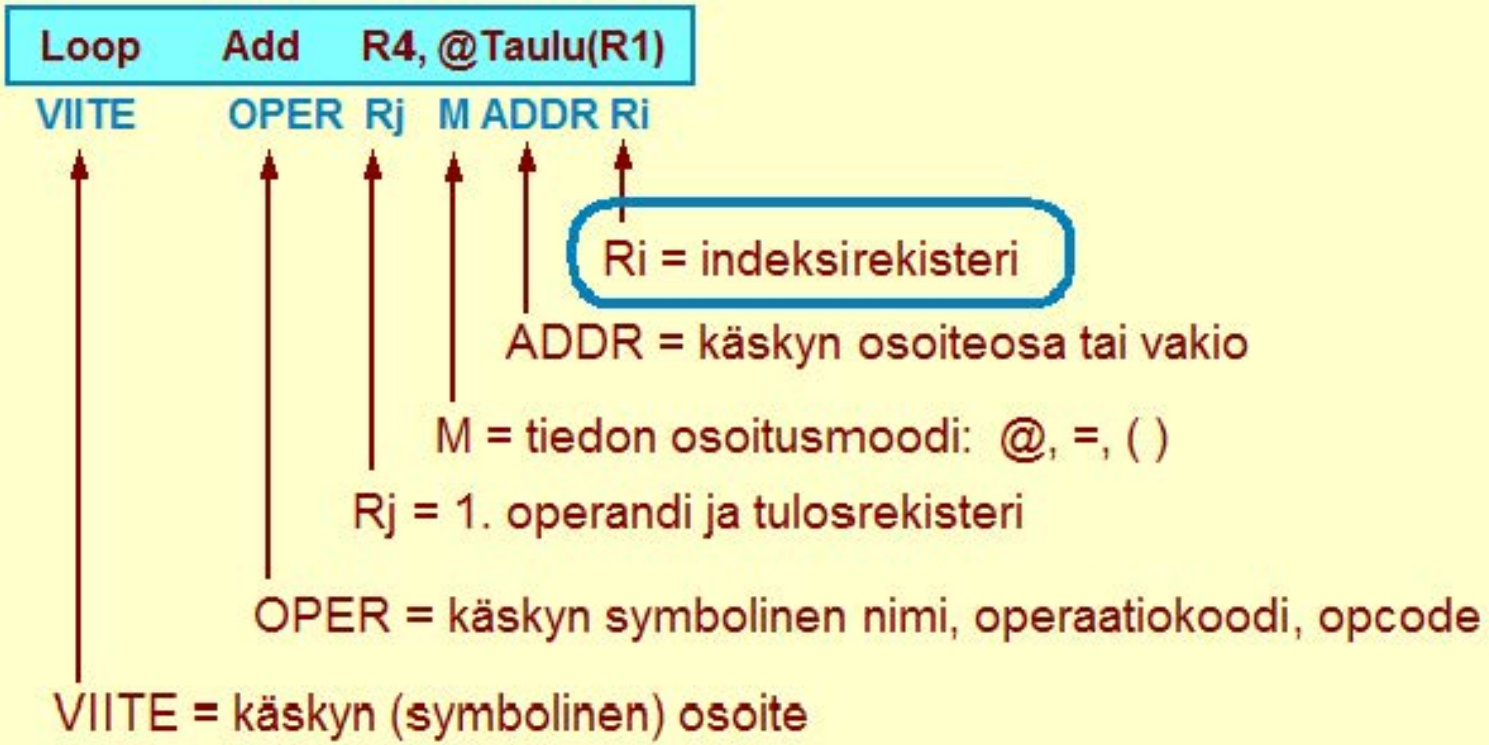
Ttk-91 koneen symbolinen konekäsky



Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

Käskyn osoiteosassa ADDR voidaan antaa mikä tahansa symboli, joka käskyn tyypistä riippuen voi olla käskyn tai datan osoite, tai vain numeerista arvoa vastaava symboli. Osoiteosassa voi myös olla pelkkä numeerinen vakio, mutta ei kovin iso, koska se pitää puhtaassa konekielessä tallettaa 16-bittiseen kenttään. Jos osoite-osa puuttuu (eli on nolla), niin se useissa tapauksissa voidaan jättää kokonaan kirjoittamatta.

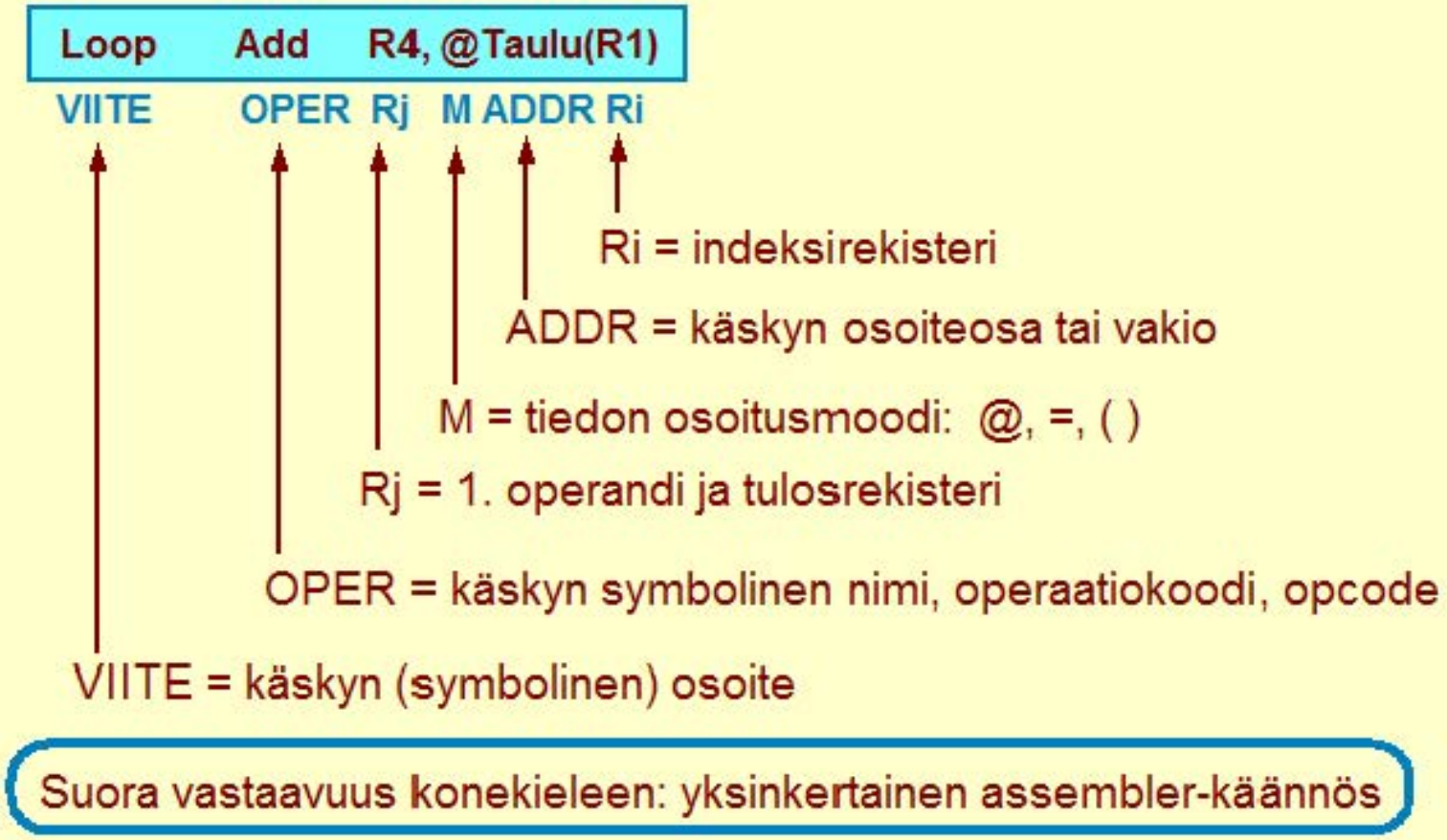
Ttk-91 koneen symbolinen konekäsky



Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

Jälkimmäinen rekisteri Ri on nimeltään indeksirekisteri, koska sitä usein käytetään taulukoiden indeksointiin. Tiedonosoitusmoodista riippuen jälkimmäinen operandi voi olla suoraan rekisterissä Ri tai sitten sen arvo tai osoite voidaan laskea Ri'n ja käskyn osoiteosan avulla. Jos Ri on R0, niin sitä ei kirjoiteta näkyville. Tämä on koodaustapa tilanteelle, jossa indeksirekisteriä ei käytetä lainkaan.

Ttk-91 koneen symbolinen konekäsky

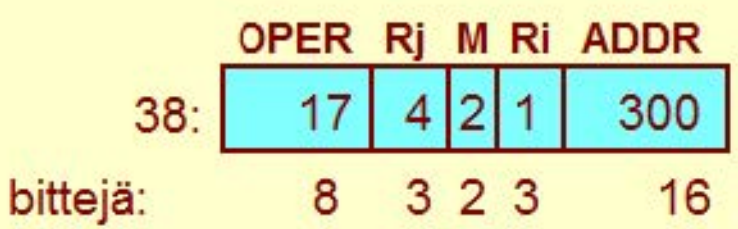


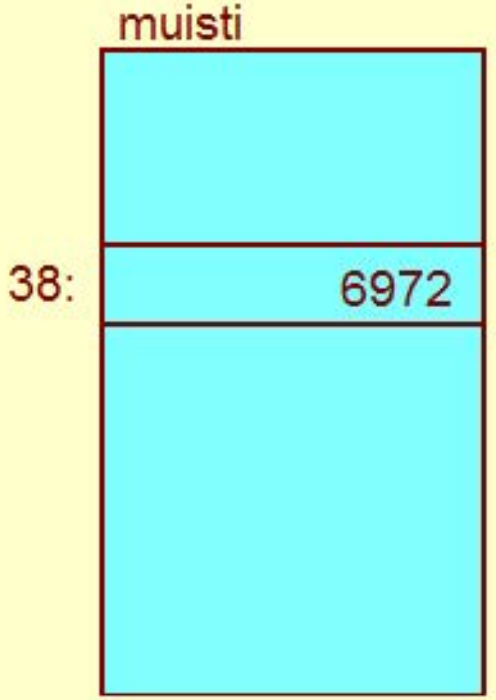
Suora vastaavuus konekieleen: yksinkertainen assembler-käännös

Symbolisella konekielellä on hyvin suoraviivainen vastaavuus puhtaaseen konekieleen. Viite-osa on käskyn osoite, operaatiokoodit on etukäteen kaikki määritelty, rekistereiden nimiä on vain muutama etukäteen määritelty, ja osoiteosan symboli vastaa jonkin muun käskyn viite-osassa määriteltyä symbolia. Ainoa vähän monimutkaisempi on osoitusmoodi. Symbolisessa konekielessä on 8 ohjelmointia helpottavaa osoitusmoodia, mutta konekielen tasolla niitä onkin vain kolme erilaista.

Ttk-91 koneen todellinen konekäsky "puhtaalla" numeerisella konekielellä

Loop	Add	R4, @Taulu(R1)
VIITE	OPER Rj	M ADDR Ri





Symbolitaulu

symboli	arvo
Loop	38
Taulu	300

Symbolista konekäskyä vastaava 'puhtaan' konekielen numeerinen käskyn esitysmuoto saadaan hyvin suoraviivaisesti symbolisesta konekäskystä. Symboleilla 'Add', 'R4' ja 'R1' on tunnetut vakioarvot 17, 4 ja 1. Tiedonosoitusmoodi '@'-merkin kanssa koodataan M-kenttään arvolla 2. Käskyn symbolinen osoite on 'Loop' ja ohjelmaa läpikäymällä on käynyt ilmi, että kyseinen käsky talletetaan muistipaikkaan 38. Vastaavasti symbolisen konekielen kääntäjä on koodia läpikäydessään päätynyt sijoittamaan taulukon 'Taulu' osoitteeseen 300. Nämä arvot löytyvät kääntäjän tekemästä symbolitaulusta.

Symbolinen konekieli vs. 'puhdas' numeerinen konekieli

Load-käskyn operaatiokoodi on 2. Ensimmäinen rekisteri Rj on 1. Indeksirekisteriä ei käytetä, mikä indikoidaan 0:lla kentässä Ri. Osoitekentän arvo on 10. Muistinviittaustapa on suora muistista nouto, mikä koodataan M-kentän arvolla 1. Jälkimmäistä operandia varten haetaan muistista jokin arvo yhden kerran.

Symbolinen konekieli vs. 'puhdas' numeerinen konekieli

Add-käskyn operaatiokoodi on 17. Ensimmäinen rekisteri on rekisteri 2 ja jälkimmäinen rekisteri on rekisteri 3. Osoite- tai vakio-osaa ei ole, joten osoitekenttään tulee luku 0. Jälkimmäinen operandi löytyy suoraan rekisteristä 3, joten sitä varten ei tarvita muistinoutoja. M-kentän arvo on siten 0.

Symbolinen konekieli vs. 'puhdas' numeerinen konekieli

Kertolaskun operaatiokoodi 19 ja ensimmäinen rekisteri on rekisteri 4. Jälkimmäinen operandi on nyt aika monimutkaisesti saatavilla. Taulukon Salary alkuosoite eli symbolin Salary arvo 3020 löytyy symbolitaulusta ja se talletetaan ADDR-kenttään. Indeksirekisterinä on rekisteri 1. Kyseessä on epäsuora muistiviite, jossa tarvitaan kaksi muistista noutoa jälkimmäistä operandia varten. Täten M-kentän arvo on 2.

Tiedon osoitusmuodot symbolisessa konekielessä

8 eri tiedon osoitusmoodia (2. operandille)

```
ADD   R5, R4
MUL   R3, X
LOAD  R1, @Method(R3)
```

Tekstuaalisesti koodattuna

osoitusmoodimerkki ennen jälkimmäistä rekisteriä

- '=' 2. operandi: indeksirekisterin arvo + osoiteosan vakio
- tyhjä 2. operandi: muuttuja tai vakio muistissa, tai rekisterin arvo
- '@' 2. operandi: epäsuora viite muistiin, osoite muistissa tai rekisterissä

sulkumerkit

- ei sulkuja käytä rekisterin arvoa sellaisenaan
- sulut käytä rekisterin arvoa muistiosoitteena

nolla-arvoa ei kirjoiteta näkyville

- indeksirekisterinä R0 tai vakiona osoitekentässä luku 0
  (R0 tarkoittaa, että indeksirekisteriä ei käytetä)

Tiedon osoitusmuotoja on symbolisessa konekielessä 8 kpl. Aluksi niitä tuntuu olevan liikaa ja ne tuntuvat sekavilta. Vähän aikaa kuitenkin ohjelmoituaan kuitenkin havaitsee, että niitä kaikkia tarvitaan. Osoitusmoodit on koodattu tekstuaalisesti ehkä vähän monimutkaisesti. Niissä on myös jonkin verran näennäistä epäkonsistenssisuutta, joten olkaa aluksi varovaisia. Käytännön esimerkit kuitenkin valaisevat tilanteen hyvin yksinkertaiseksi.

Tiedon osoitusmuodot symbolisessa konekielessä

8 eri tiedon osoitusmoodia (2. operandille)

```
ADD   R5, R4
MUL   R3, X
LOAD  R1, @Method(R3)
```

Tekstuaalisesti koodattuna

osoitusmoodimerkki ennen jälkimmäistä rekisteriä

- '=' 2. operandi: indeksirekisterin arvo + osoiteosan vakio
- tyhjä 2. operandi: muuttuja tai vakio muistissa, tai rekisterin arvo
- '@' 2. operandi: epäsuora viite muistiin, osoite muistissa tai rekisterissä

sulkumerkit

- ei sulkuja käytä rekisterin arvoa sellaisenaan
- sulut käytä rekisterin arvoa muistiosoitteena

nolla-arvoa ei kirjoiteta näkyville

- indeksirekisterinä R0 tai vakiona osoitekentässä luku 0
  (R0 tarkoittaa, että indeksirekisteriä ei käytetä)

Tärkein tiedonosoitusmoodin indikaatio on jälkimmäistä rekisteriä edeltävä merkki tai sen puuttuminen. '='-merkki tarkoittaa, että toinen operandi saadaan ilman muistiviitteitä. Puuttuva merkki ilmaisee yleensä, että tieto pitää hakea muistista. Poikkeuksen muodostaa tilanne, jossa jälkimmäinen rekisteri on ilman sulkuja, jolloin toinen operandi löytyy suoraan sieltä. '@'-merkki tarkoittaa epäsuoraa viitettä, jolloin tiedon osoite haetaan ensin muistista ja sitten vasta itse tieto, toisella muistiviitteellä.

Tiedon osoitusmuodot symbolisessa konekielessä

8 eri tiedon osoitusmoodia (2. operandille)

```
ADD   R5, R4
MUL   R3, X
LOAD  R1, @Method(R3)
```

Tekstuaalisesti koodattuna

osoitusmoodimerkki ennen jälkimmäistä rekisteriä

- '=' 2. operandi: indeksirekisterin arvo + osoiteosan vakio
- tyhjä 2. operandi: muuttuja tai vakio muistissa, tai rekisterin arvo
- '@' 2. operandi: epäsuora viite muistiin, osoite muistissa tai rekisterissä

sulkumerkit

- ei sulkuja käytä rekisterin arvoa sellaisenaan
- sulut käytä rekisterin arvoa muistiosoitteena

nolla-arvoa ei kirjoiteta näkyville

- indeksirekisterinä R0 tai vakiona osoitekentässä luku 0
  (R0 tarkoittaa, että indeksirekisteriä ei käytetä)

Sulkumerkit rekisterin ympärillä tarkoittavat yleensä, että rekisterin sisältöä (plus mahdollista vakiota) käytetään muistiosoitteena. Ilman sulkuja olevaa rekisterin arvoa käytetään sellaisenaan. Tässä yhteydessä (siis ilman sulkuja) vakio-osa puuttuu eli on nolla. Yleensäkin pitää paikkansa, että puuttuvaa tiedon osoittamisessa käytettyä komponenttia ei tarvitse kirjoittaa näkyville. Tämä pätee sekä puuttuvaan indeksirekisteriin että vakioarvoon 0.

Indeksointi

LOAD R4, =Tbl(R3)

(tässä Tbl on taulukon alkuosoite, indeksi on R3:ssa)

Laske aina ensin 'alkuperäinen tehollinen muistiosoite' (effective address, EA)

EA = Tbl + (R3) = 201
TR ← EA

Sitten katso moodia ja tee niin monta muistinoutoa kuin tarvitaan.
Yleensä:

'='	→	0 kpl, operandi
tyhjä	→	1 kpl, suora muistiosoitus
'@'	→	2 kpl, epäsuora muistiosoitus

OPER	Rj	M	Ri	ADDR
2	4	0	3	195

R4 ← 201

R4 ← Mem[201] = 11

R4 ← Mem[Mem[201]]
← Mem[11] = 300

pelkkä rekisterin numero @-merkin jälkeen ⇨ vain yksi muistinouto

STORE-käsky ⇨ 1 kpl vähemmän muistinoutoja ja yksi tallennus

Kaikki tiedonosoitustavat 2. operandille toteutetaan itse asiassa samalla tavalla indeksoinnin avulla. Aina aluksi lasketaan yhteen indeksirekisterin sisältö ja käskyn osoiteosa (vakio). Tätä arvoa kutsutaan 'teholliseksi muistiosoitteeksi' ja se talletetaan laitteiston tilapäisrekisteriin TR. Tiedonosoitusmoodi on tulkittu käännösaikana symbolisesta konekielestä ja koodattu konekäskyn M-kenttään pieneksi luvuksi, joka on 0, 1 tai 2.

Indeksointi

LOAD R4, =Tbl(R3)

(tässä Tbl on taulukon alkuosoite, indeksi on R3:ssa)

Laske aina ensin 'alkuperäinen tehollinen muistiosoite' (effective address, EA)

EA = Tbl + (R3) = 201
TR ← EA

Sitten katso moodia ja tee niin monta muistinoutoa kuin tarvitaan.
Yleensä:

'='	→	0 kpl, operandi
tyhjä	→	1 kpl, suora muistiosoitus
'@'	→	2 kpl, epäsuora muistiosoitus

OPER	Rj	M	Ri	ADDR
2	4	0	3	195

R4 ← 201

R4 ← Mem[201] = 11

R4 ← Mem[Mem[201]]
← Mem[11] = 300

pelkkä rekisterin numero @-merkin jälkeen ⇨ vain yksi muistinouto

STORE-käsky ⇨ 1 kpl vähemmän muistinoutoja ja yksi tallennus

'='-merkki vastaa jonkinlaista välitöntä operandia, jolloin arvoa EA käytetään sellaisenaan 2. operandina. R4 saa siis arvon 201. Puuttuva moodimerkki vastaa yleisintä muistiinviittaustapaa, jolloin muistista haetaan EA:n osoittamasta paikasta 2. operandi. R4 saa arvon 11. '@'-merkki on epäsuora muistiinviittaus, jolloin muistista haetaan ensin tiedon osoite EA:n osoittamasta paikasta, ja sitten tuota osoitetta käyttäen muistista haetaan varsinainen tieto. R4 saa siis arvon 300.

Indeksointi

LOAD R4, =Tbl(R3)

(tässä Tbl on taulukon alkuosoite, indeksi on R3:ssa)

Laske aina ensin 'alkuperäinen tehollinen muistiosoite' (effective address, EA)

EA = Tbl + (R3) = 201
TR ← EA

Sitten katso moodia ja tee niin monta muistinoutoa kuin tarvitaan.

OPER	Rj	M	Ri	ADDR
2	4	0	3	195

Yleensä:

'='	→	0 kpl, operandi
tyhjä	→	1 kpl, suora muistiosoitus
'@'	→	2 kpl, epäsuora muistiosoitus

R4 ← 201

R4 ← Mem[201] = 11

R4 ← Mem[Mem[201]]
← Mem[11] = 300

pelkkä rekisterin numero @-merkin jälkeen ⇨ vain yksi muistinouto

STORE-käsky ⇨ 1 kpl vähemmän muistinoutoja ja yksi tallennus

Edellämainitut tilanteet selostavat yleistapauksen. Käytännössä asia on vähän monimutkaisempi. Esimerkiksi, jos '@'-merkin jälkeen tulee pelkkä rekisterin numero, niin silloin kyseessä on epäsuora viite, mutta vain rekisterin perusteella ja muistinviitteitä tulee vain yksi. Muistiin kirjoittaminen myös vaikuttaa tilanteeseen. STORE-käskyllä on aina 1 kpl vähemmän muistinoutoja, mutta käskyn lopussa siihen liittyy yksi muistiin tallentaminen. M-kentän arvo kuitenkin kertoo vain muistinoutojen lukumäärän ennen itse operaation suorittamista.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit

R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti

0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu

Tbl:	200
X:	10
One:	1

```
LOAD  R4, =10          ; R4 ←   10
LOAD  R4, X            ; R4 ←  200
LOAD  R4, @X           ; R4 ← 6000

LOAD  R4, R2           ; R4 ←  201
LOAD  R4, @R2          ; R4 ←   11
LOAD  R4, =R2

LOAD  R4, =Tbl(R3)     ; R4 ←  201
LOAD  R4, Tbl(R3)      ; R4 ←   11
LOAD  R4, @Tbl(R3)     ; R4 ←  300
```

välitön operandi
suora muistiosoitus
epäsuora muistiosoitus

suora rekisteriosoitus
epäsuora rekisteriosoitus
~~välitön rekisteriosoitus~~

indeksoitu välitön operandi
indeksoitu muistiosoitus
indeksoitu epäsuora muistiosoitus

Tässä kuvataan esimerkkien avulla kaikki ttk-91 koneen symbolisen konekielen tiedonosoitusmuodot. Lähtökohtana kaikissa esimerkeissä on sama tilanne. Rekisterin R0 arvo on 104, R1'n arvo on 10 jne. Muistipaikan 10 arvo 200 ja muistipaikan 201 arvo on 11. Symbolin Tbl arvo on 200, symbolin X arvo on 10 ja symbolin One arvo on 1. Koodissa on siis aivan sama, kirjoittaako käskyn osoitekenttään symbolin X tai luvun 10. Tämä tietenkin olettaen, että tiedämme symbolin X arvon olevan 10. Yks tapa saada symbolille arvo 10 on käyttää pseudokäskyä 'X EQU 10'. Useimmiten kuitenkin kääntäjä antaa symbolien arvot.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit	
R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti	
0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu	
Tbl:	200
X:	10
One:	1

```
LOAD  R4, =10          ; R4 ←   10    välitön operandi
LOAD  R4, X            ; R4 ←  200    suora muistiosoitus
LOAD  R4, @X           ; R4 ← 6000    epäsuora muistiosoitus

LOAD  R4, R2           ; R4 ←  201    suora rekisteriosoitus
LOAD  R4, @R2          ; R4 ←   11    epäsuora rekisteriosoitus
LOAD  R4, =R2                         välitön rekisteriosoitus

LOAD  R4, =Tbl(R3)     ; R4 ←  201    indeksoitu välitön operandi
LOAD  R4, Tbl(R3)      ; R4 ←   11    indeksoitu muistiosoitus
LOAD  R4, @Tbl(R3)     ; R4 ←  300    indeksoitu epäsuora muistiosoitus
```


Ensimmäisessä kolmen käskyn ryhmässä jälkimmäinen operandi määräytyy ainoastaan käskyn osoiteosan (eli vakio-osan) avulla. Indeksirekisteriä ei käytetä. Kun käskyn osoiteosassa olevaa arvoa käytetään sellaisenaan 2. operandina, kyseessä on 'välitön operandi', joka merkitään symbolisessa konekielessä '='-merkillä. Esimerkin luvun 10 asemesta voisi yhtä hyvin käyttää mitä tahansa tunnusta, jonka arvo on 10 - esimerkiksi tunnusta 'X'. Rekisterin R4 arvoksi tulee luku 10.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit

R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti

0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu

Tbl:	200
X:	10
One:	1

LOAD R4, =10	; R4 ←	10	välitön operandi
LOAD R4, X	; R4 ←	200	suora muistiosoitus
LOAD R4, @X	; R4 ←	6000	epäsuora muistiosoitus
LOAD R4, R2	; R4 ←	201	suora rekisteriosoitus
LOAD R4, @R2	; R4 ←	11	epäsuora rekisteriosoitus
~~LOAD R4, =R2~~			~~välitön rekisteriosoitus~~
LOAD R4, =Tbl(R3)	; R4 ←	201	indeksoitu välitön operandi
LOAD R4, Tbl(R3)	; R4 ←	11	indeksoitu muistiosoitus
LOAD R4, @Tbl(R3)	; R4 ←	300	indeksoitu epäsuora muistiosoitus

Suora muistiosoitus 'merkitään' puuttuvalla moodimerkinnällä ja se on ns. tavallinen muistiinviittaustapa. Osoitekentän arvoa 10 (eli symbolin X arvoa 10) käytetään muistiosoitteena ja muistipaikan 10 sisältö eli luku 200 on toisena operandina, joka talletetaan rekisteriin R4.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit	
R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti	
0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

```
LOAD  R4, =10          ; R4 ←   10
LOAD  R4, X            ; R4 ←  200
LOAD  R4, @X           ; R4 ← 6000

LOAD  R4, R2           ; R4 ←  201
LOAD  R4, @R2          ; R4 ←   11
LOAD  R4, =R2

LOAD  R4, =Tbl(R3)     ; R4 ←  201
LOAD  R4, Tbl(R3)      ; R4 ←   11
LOAD  R4, @Tbl(R3)     ; R4 ←  300
```

välitön operandi
suora muistiosoitus
epäsuora muistiosoitus

suora rekisteriosoitus
epäsuora rekisteriosoitus
~~välitön rekisteriosoitus~~

indeksoitu välitön operandi
indeksoitu muistiosoitus
indeksoitu epäsuora muistiosoitus

symbolitaulu	
Tbl:	200
X:	10
One:	1

'@'-merkki indikoi epäsuoraa muistinosoitusta. Nyt muistipaikan X (eli muistipaikan 10) sisältämää arvoa 200 ei käytetäkkään 2. operandina vaan vasta 2. operandin osoitteena. Varsinainen operandi löytyy siis muistipaikasta 200, josta saadaan luku 6000 rekisterin R4 uudeksi arvoksi. Tässä yhteydessä tarvitaan siis kaksi muistinoutoa 2. operandia varten: ensin haetaan sen osoite käskyn osoitekentän perusteella ja sitten sen arvo juuri haetun osoitteen perusteella.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit

R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti

0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu

Tbl:	200
X:	10
One:	1

```
LOAD  R4, =10          ; R4 ←   10
LOAD  R4, X            ; R4 ←  200
LOAD  R4, @X           ; R4 ← 6000

LOAD  R4, R2           ; R4 ←  201
LOAD  R4, @R2          ; R4 ←   11
LOAD  R4, =R2

LOAD  R4, =Tbl(R3)     ; R4 ←  201
LOAD  R4, Tbl(R3)      ; R4 ←   11
LOAD  R4, @Tbl(R3)     ; R4 ←  300
```

välitön operandi
suora muistiosoitus
epäsuora muistiosoitus

suora rekisteriosoitus
epäsuora rekisteriosoitus
~~välitön rekisteriosoitus~~

indeksoitu välitön operandi
indeksoitu muistiosoitus
indeksoitu epäsuora muistiosoitus

Seuraava kahden käskyn ryhmä käsittelee tilannetta, jolloin symbolisen konekielen käskyssä 2. operandissa on ainoastaan rekisteri-osa ja osoite-osa puuttuu eli on 0. Suorassa rekisteriviittauksessa 2. operandi löytyy valmiina jälkimmäisestä rekisteristä, jolloin rekisterin R2 arvo 201 kopioidaan rekisteriin R4.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit

R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti

0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

```
LOAD  R4, =10          ; R4 ←   10
LOAD  R4, X            ; R4 ←  200
LOAD  R4, @X           ; R4 ← 6000

LOAD  R4, R2           ; R4 ←  201
LOAD  R4, @R2          ; R4 ←   11
LOAD  R4, =R2

LOAD  R4, =Tbl(R3)     ; R4 ←  201
LOAD  R4, Tbl(R3)      ; R4 ←   11
LOAD  R4, @Tbl(R3)     ; R4 ←  300
```

välitön operandi
suora muistiosoitus
epäsuora muistiosoitus

suora rekisteriosoitus
epäsuora rekisteriosoitus
välitön rekisteriosoitus

indeksoitu välitön operandi
indeksoitu muistiosoitus
indeksoitu epäsuora muistiosoitus

symbolitaulu

Tbl:	200
X:	10
One:	1

Epäsuorassa rekisteriosoituksessa jälkimmäisessä rekisterissä olevaa arvoa ei käytetä sellaisenaan, vaan muistiosoitteena paikkaan, josta 2. operandi löytyy. Jälkimmäinen operandi haetaan siis muistipaikasta 201 ja sieltä löytynyt arvo 11 talletetaan lopulta rekisteriin R4.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit	
R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti	
0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu	
Tbl:	200
X:	10
One:	1

```
LOAD  R4, =10          ; R4 ←   10
LOAD  R4, X            ; R4 ←  200
LOAD  R4, @X           ; R4 ← 6000

LOAD  R4, R2           ; R4 ←  201
LOAD  R4, @R2          ; R4 ←   11
LOAD  R4, =R2

LOAD  R4, =Tbl(R3)     ; R4 ←  201
LOAD  R4, Tbl(R3)      ; R4 ←   11
LOAD  R4, @Tbl(R3)     ; R4 ←  300
```

välitön operandi
suora muistiosoitus
epäsuora muistiosoitus

suora rekisteriosoitus
epäsuora rekisteriosoitus
~~välitön rekisteriosoitus~~

indeksoitu välitön operandi
indeksoitu muistiosoitus
indeksoitu epäsuora muistiosoitus

Ei ole olemassa 'välitöntä rekisteriviitettä', koska tämähän tarkottaisi, että jälkimmäinen operandi olisi rekisterin R2 osoite eli luku 2. Rekistereihin ei voi viitata niiden osoitteen (indeksin) perusteella, vaan aina on käytettävä niiden symbolisia nimiä kuten 'R1' tai 'R2'. Kääntäjä antaa virheilmoituksen tällaisesta tiedonosoituksesta.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit	
R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti	
0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

LOAD R4, =10	; R4 ←	10	välitön operandi
LOAD R4, X	; R4 ←	200	suora muistiosoitus
LOAD R4, @X	; R4 ←	6000	epäsuora muistiosoitus
LOAD R4, R2	; R4 ←	201	suora rekisteriosoitus
LOAD R4, @R2	; R4 ←	11	epäsuora rekisteriosoitus
~~LOAD R4, =R2~~			~~välitön rekisteriosoitus~~
LOAD R4, =Tbl(R3)	; R4 ←	201	indeksoitu välitön operandi
LOAD R4, Tbl(R3)	; R4 ←	11	indeksoitu muistiosoitus
LOAD R4, @Tbl(R3)	; R4 ←	300	indeksoitu epäsuora muistiosoitus

symbolitaulu	
Tbl:	200
X:	10
One:	1

Viimeinen rykelmä konekäskyjä liittyy tilanteeseen, jossa jälkimmäinen operandi määräytyy sekä indeksirekisterin että osoiteosan avulla. Indeksoidussa välittömässä operandissa indeksirekisteri ja käskyn osoiteosa lasketaan yhteen ja näin saatu arvo otetaan suoraan jälkimmäiseksi operandiksi. Esimerkissä R3'n arvo on 1 ja käskyn osoiteosassa on muuttujan Tbl arvo 200. R4'n uudeksi arvoksi tulee siis 201.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit

R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti

0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu

Tbl:	200
X:	10
One:	1

LOAD R4, =10	; R4 ← 10	välitön operandi
LOAD R4, X	; R4 ← 200	suora muistiosoitus
LOAD R4, @X	; R4 ← 6000	epäsuora muistiosoitus
LOAD R4, R2	; R4 ← 201	suora rekisteriosoitus
LOAD R4, @R2	; R4 ← 11	epäsuora rekisteriosoitus
~~LOAD R4, =R2~~		~~välitön rekisteriosoitus~~
LOAD R4, =Tbl(R3)	; R4 ← 201	indeksoitu välitön operandi
LOAD R4, Tbl(R3)	; R4 ← 11	indeksoitu muistiosoitus
LOAD R4, @Tbl(R3)	; R4 ← 300	indeksoitu epäsuora muistiosoitus

Indeksoitu muistisoitus käyttää edellisessä kohdassa laskettua indeksirekisterin ja käskyn osoiteosan summaa muistiosoitteena ja siten muistipaikasta 201 haetaan R4:lle uusi arvo 11.

Ttk-91 koneen tiedonosoitusmuodot symbolisessa konekielessä

rekisterit	
R0:	104
R1:	10
R2:	201
R3:	1
R4:	?

muisti	
0:	
10:	200
11:	300
200:	6000
201:	11
LIMIT:	

symbolitaulu	
Tbl:	200
X:	10
One:	1

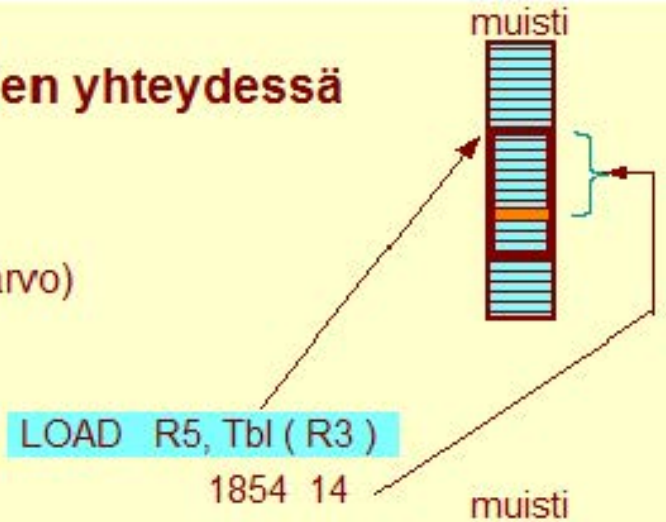
```
LOAD  R4, =10          ; R4 ←   10    välitön operandi
LOAD  R4, X            ; R4 ←  200    suora muistiosoitus
LOAD  R4, @X           ; R4 ← 6000    epäsuora muistiosoitus

LOAD  R4, R2           ; R4 ←  201    suora rekisteriosoitus
LOAD  R4, @R2          ; R4 ←   11    epäsuora rekisteriosoitus
LOAD  R4, =R2                         välitön rekisteriosoitus

LOAD  R4, =Tbl(R3)     ; R4 ←  201    indeksoitu välitön operandi
LOAD  R4, Tbl(R3)      ; R4 ←   11    indeksoitu muistiosoitus
LOAD  R4, @Tbl(R3)     ; R4 ←  300    indeksoitu epäsuora muistiosoitus
```


Viimeisessä tapauksessa kyseessä on indeksoitu epäsuora muistiosoitus. Tällä kertaa indeksirekisterin ja käskyn osoiteosan summa antaakin vasta varsinaisen operandin osoitteen sijaintipaikan muistissa. Haemme siis ensin osoitteen 11 muistipaikasta 201, ja sitten varsinaisen operandin 300 muistipaikasta 11. Rekisterin R4 arvoksi tulee siis 300.

Indeksoinnin käyttö taulukoiden ja tietueiden yhteydessä

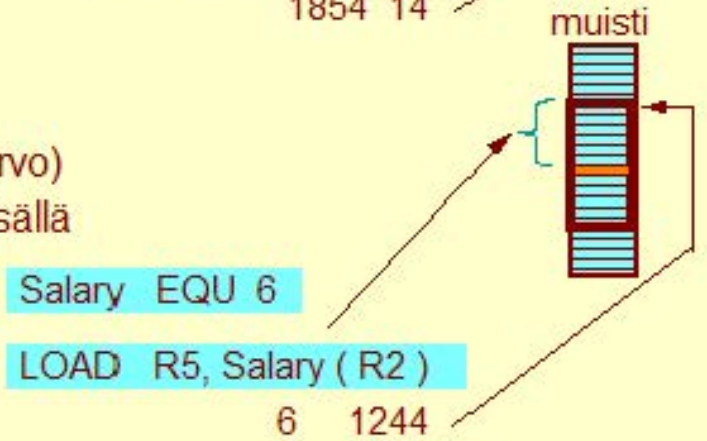
Taulukot

- Taulukon alkuosoite vakiona osoitekentässä ('iso' arvo)
- Taulukon indeksi indeksirekisterissä ('pieni' arvo)



Tietueet

- Tietueen alkuosoite indeksirekisterissä ('iso' arvo)
- Tietueen kentän suhteellinen osoite tietueen sisällä vakiona osoitekentässä ('pieni' arvo)

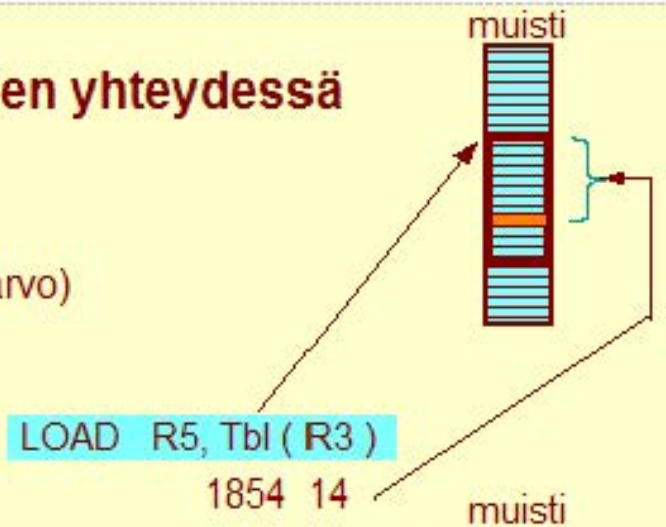


Indeksoitu tiedonosoitusmuoto on lähes kaikissa tietokonearkkitehtuureissa, koska sen avulla voidaan kätevästi toteuttaa kaksi yleisintä rakenteista tietotyyppiä: taulukot ja tietueet. Taulukkojen yhteydessä taulukon alkuosoite annetaan käskyn osoitekentässä ja taulukon indeksi indeksirekisterissä. Juuri tämän viittaustavan vuoksi ttk-91 koneen jälkimmäinen rekisteri on nimeltään 'indeksirekisteri'. Tällä tavoin saadaan helposti toteutettua yksiulotteiset taulukot, mutta on myös olemassa arkkitehtuureja, joissa on käytössä useampi indeksirekisteri samaan aikaan.

Indeksoinnin käyttö taulukoiden ja tietueiden yhteydessä

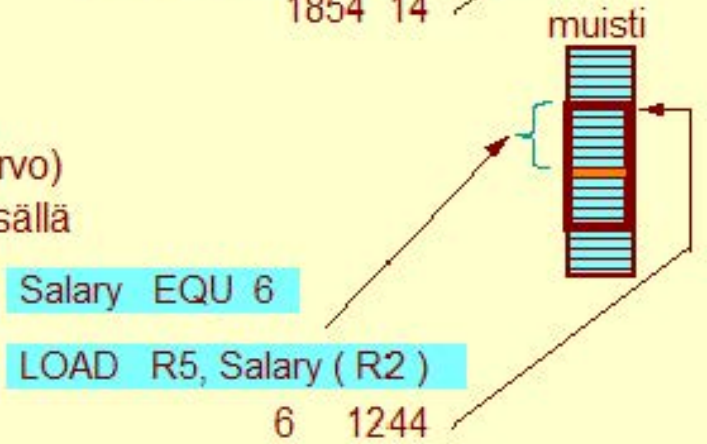
Taulukot

- Taulukon alkuosoite vakiona osoitekentässä ('iso' arvo)
- Taulukon indeksi indeksirekisterissä ('pieni' arvo)



Tietueet

- Tietueen alkuosoite indeksirekisterissä ('iso' arvo)
- Tietueen kentän suhteellinen osoite tietueen sisällä vakiona osoitekentässä ('pieni' arvo)



Tietueiden osalta toteutus on päinvastainen. Tietueen alkuosoite on nyt konekäskyn jälkimmäisessä rekisterissä eli tämä rekisteri osoittaa tietueen alkuun. Toisin sanoen kyseinen rekisteri toimii osoitinrekisterinä tietueeseen. Tietue muodostuu kentistä ja kullakin kentällä on vakiopaikka tietueen sisällä ja tämä suhteellinen sijainti sitten ilmaistaan käskyn vakiokentässä. Yleensä tietueen määrittelyn yhteydessä jokaiselle kentälle on määritelty vakiosymboli, jonka arvo on kyseisen kentän suhteellinen sijainti tietueen sisällä.

Ttk-91 koneen konekäskyjen operaatiot

Muistiinviittaukset

- Tavalliset: load, store
- Pino-operaatiot aliohjelmien toteuttamista varten

Syöttö ja tulostus (I/O)

Kokonaislukuoperaatiot

Bittimanipulaatiot

Kontrollin siirto

- Mistä löytyy seuraavaksi suoritettava käsky?
  (ellei siis oletusarvoisesti nykykäskyä seuraavassa muistipaikassa)

Muut operaatiot

Pseudokäskyt

Ttk-91 koneen, kuten kaikkien muidenkin arkkitehtuurien, konekäskyjen operatiot vuodaan luokitella muutamaan luokkaan. Muistinviittauskäskyihin kuuluvat kaikki ne hitaahkot operaatiot, joilla viitataan muistiin. Useissa koneissa on omia konekäskyjä I/O:n toteuttamiseen - niin myös ttk-91:ssä. Kaikki normaali työ tehdään kokonaisluku- ja bittimanipulaatioilla. Kontrollin siirtokäskyihin kuuluvat kaikki erilaiset hyppykäskyt sekä aliohjelmien kutsut ja niistä paluut. Sen lisäksi mukana on pieni joukko muita sekalaisia käskyjä ja kääntäjän ohjauskäskyjä eli pseudokäskyjä.

Ttk-91 muistinviittausoperaatiot

LOAD

```
LOAD  R1, X
LOAD  R5, @ptrX
LOAD  R5, =Nimi(R3)
```

- käskyä käytetään myös pelkkään rekistereiden kopiointiin laiterekisteristä tai käskyrekisteristä laiterekisteriin (useissa arkkitehtuureissa on oma MOVE käsky rekistereiden kopiontia varten)

```
LOAD  R0, R5
LOAD  R0, =34
```

STORE

- Tallettaa aina muistiin

```
STORE  R2, X
STORE  R5, @ptrX
STORE  R3, Tbl(R4)
```

~~STORE R0, R5~~

~~STORE R0, =34~~ ~~STORE R0, =X~~

~~STORE R5, =Nimi(R3)~~

PUSH, POP, PUSHR, POPR

- Aliohjelmien toteuttamista varten

```
PUSH  SP, R1
PUSH  SP, =5
```

- Voi kopioida tietoa muistista muistiin tai muistista rekisteriin

```
PUSH  SP, X
POP  SP, R1
```

- Käsitellään myöhemmin (luento 4)

```
PUSHR  SP
```


Load-käskyllä yleensä kopiodaan tietoa muistista rekisteriin käyttäen erilaisia muistinviittaustapoja, joita opiskelimme juuri äsken. Joissakin todellisissa koneissa on erikseen oma konekäskynsä rekistereiden väliseen kopiointiin, mutta ttk-91'ssä Load-käskyä käytetään myös tähän. Sen avulla voidaan siis kopioda myös tietoa laiterekisteristä toiseen, tai käskyrekisterin osoite-kentästä laiterekisteriin. Huomaa kuitenkin, että rekisterissä R0 olevaa tietoa ei voida kopioda Load-käskyllä muualle, koska R0 jälkimmäisenä rekisterinä tarkoittaa, että indeksirekisteriä ei käytetä!

Ttk-91 muistinviittausoperaatiot

LOAD

```
LOAD  R1, X
LOAD  R5, @ptrX
LOAD  R5, =Nimi(R3)
```

- käskyä käytetään myös pelkkään rekistereiden kopiointiin laiterekisteristä tai käskyrekisteristä laiterekisteriin

```
LOAD  R0, R5
LOAD  R0, =34
```

(useissa arkkitehtuureissa on oma MOVE käsky rekistereiden kopiontia varten)

STORE

- Tallettaa aina muistiin

```
STORE  R2, X
STORE  R5, @ptrX
STORE  R3, Tbl(R4)
```

~~STORE R0, R5~~
~~STORE R0, =34~~
~~STORE R0, =X~~
~~STORE R5, =Nimi(R3)~~

PUSH, POP, PUSHR, POPR

- Aliohjelmien toteuttamista varten

```
PUSH  SP, R1
PUSH  SP, =5
```

- Voi kopioida tietoa muistista muistiin tai muistista rekisteriin

```
PUSH  SP, X
POP  SP, R1
```

- Käsitellään myöhemmin (luento 4)

```
PUSHR  SP
```


Store-käsky tallettaa aina muistiin. On virhe sijoittaa pelkkä rekisteri Store-käskyn jälkimmäiseksi operandiksi. Rekisteristä toiseen kopionti tulee aina tehdä Load-käskyllä. Store-käskyllä ei myöskään voi muuttaa minkään luvun (esim 34) tai symbolin arvoa. Symbolit ja niiden arvot ovat käännösaikaisia käsitteitä, joten symbolien arvoja ei voi muuttaa suoritusaikana. Huomaa, että STORE-käsky vain kopioi tiedon työrekisteristä muistiin - data jää edelleen käytettäväksi työrekisteriin.

Ttk-91 muistinviittausoperaatiot

LOAD

```
LOAD  R1, X
LOAD  R5, @ptrX
LOAD  R5, =Nimi(R3)
```

- käskyä käytetään myös pelkkään rekistereiden kopiointiin laiterekisteristä tai käskyrekisteristä laiterekisteriin

```
LOAD  R0, R5
LOAD  R0, =34
```

(useissa arkkitehtuureissa on oma MOVE käsky rekistereiden kopiontia varten)

STORE

- Tallettaa aina muistiin

```
STORE  R2, X
STORE  R5, @ptrX
STORE  R3, Tbl(R4)
```

~~STORE R0, R5~~
~~STORE R0, =34~~
~~STORE R0, =X~~
~~STORE R5, =Nimi(R3)~~

PUSH, POP, PUSHR, POPR

- Aliohjelmien toteuttamista varten

```
PUSH  SP, R1
PUSH  SP, =5
```

- Voi kopioida tietoa muistista muistiin tai muistista rekisteriin

```
PUSH  SP, X
POP  SP, R1
```

- Käsitellään myöhemmin (luento 4)

```
PUSHR  SP
```


Push-, Pop-, Pushr- ja Popr-käskyt manipuloivat muistissa olevaa isoa tietorakennetta nimeltään 'Pino'. Pinon pinnalle osoittaa aina pinorekisteri SP eli rekisteri R6. Push-käskyjen avulla pinon pinnalle kopiodaan alkiota ja Pop-käskyjen avulla niitä kopioidaan sieltä pois, aina automaattisesti SP-rekisteriä päivittäen. Nämä ovat tavanomaista monimutkaisempia käskyjä, koska niillä voi kopioida tietoa yhdestä paikasta muistia toiseen paikkaan muistia eli pinoon. Nämä käskyt on suunniteltu aliohjelmien toteuttamisen tueksi ja niitä käsitellään aliohjelmien toteutuksen yhteydessä tarkemmin.

Ttk-91 syöttö- ja tulostus (I/O) operaatiot

```
IN   R3, =KBD
```

IN

- lue arvo (kokonaisluku) annetulta laitteelta rekisteriin

OUT

- tulosta arvo (kokonaisluku) rekisteristä annetulle laitteelle

```
OUT  R2, =CRT
```

Laitteet?

- syöttö: KBD, keyboard, näppäimistö
- tulostus: CRT, cathode ray tube, näyttö
- Titokoneessa myös tiedostoja
- ei muita.

Ttk-91 koneessa on kaksi konekäskyä primitiivi I/O:ta varten. Kummallakin voi tietenkin käsitellä vain kokonaislukuesitysmuotoa. IN-käsky lukee näppäimistöltä yhden kokonaisluvun ja sijoittaa sen annettuun rekisteriin. OUT-käsky vastaavasti tulostaa näytölle rekisterissä olevan arvon. I/O-konekäskyt operoivat vain kahdella laitteella: KBD ja CRT. Simulaattorissa molemmat laitteet on toteutettu näppäimistön ja käyttöliittymän ikkunoinnin avulla.

Ttk-91 kokonaislukuoperaatiot

LOAD ("move")

```
LOAD  R3, R1   ; R3 ← R1
```

ADD, SUB

```
ADD  R3, R1   ; R3 ← R3+R1
SUB  R3, =1   ; R3 ← R3-1
```

Tilarekisterin bitti O (overflow)?

MUL

```
MUL   R3, Tbl(R1)   ; R3 ← R3 * Mem[Tbl+(R1)]
```

Tilarekisterin bitti O (overflow)?

DIV, MOD

```
LOAD R1, =14
DIV    R1, =3      ; R1 ← 4

LOAD R2, =14
MOD   R2, =3       ; R2 ← 2
```

Tilarekisterin bitti Z (divide by Zero)?

Tilarekisterin bitti Z (divide by Zero)?

Ttk-91'ssä on viisi varsinaista kokonaislukuoperaatiota, mutta rekisterien kopiointiin käytettävää LOAD-käskyä voisi pitää kuudentena. LOAD-käskyä voidaan siis käyttää tavalliseen tiedonsiirtoon rekistereiden välillä sen lisäksi, että sillä tehdään muistinoudot rekistereihin. Todellisissa koneissa nämä kaksi aivan eri tyyppistä toimintoa onkin yleensä toteutettu omilla konekäskyillään. Rekistereiden välisen kopioinnin hoitaa yleensä jonkin sortin 'move'-käsky.

Ttk-91 kokonaislukuoperaatiot

LOAD ("move")

```
LOAD  R3, R1   ; R3 ← R1
```

ADD, SUB

```
ADD  R3, R1   ; R3 ← R3+R1
SUB  R3, =1   ; R3 ← R3-1
```

Tilarekisterin bitti O (overflow)?

MUL

```
MUL   R3, Tbl(R1)   ; R3 ← R3 * Mem[Tbl+(R1)]
```

Tilarekisterin bitti O (overflow)?

DIV, MOD

```
LOAD R1, =14
DIV    R1, =3      ; R1 ← 4

LOAD R2, =14
MOD   R2, =3       ; R2 ← 2
```

Tilarekisterin bitti Z (divide by Zero)?

Tilarekisterin bitti Z (divide by Zero)?

ADD- ja SUB-käskyissä ei ole mitään erikoista. Jälkimmäinen operandi valikoidaan normaalitapaan ja sitten sen arvo joko lisätään tai vähennetään ensimmäisenä operandina olevasta rekisteristä. Jos tulos ei mahdu rekisteriir, niin tilarekisterin O-bitti asetetetaan arvoon yksi ja ohjelman suoritus keskeytyy.

Ttk-91 kokonaislukuoperaatiot

LOAD ("move")

```
LOAD  R3, R1   ; R3 ← R1
```

ADD, SUB

```
ADD  R3, R1   ; R3 ← R3+R1
SUB  R3, =1   ; R3 ← R3-1
```

Tilarekisterin bitti O (overflow)?

MUL

```
MUL   R3, Tbl(R1)   ; R3 ← R3 * Mem[Tbl+(R1)]
```

Tilarekisterin bitti O (overflow)?

DIV, MOD

```
LOAD R1, =14
DIV    R1, =3      ; R1 ← 4

LOAD R2, =14
MOD   R2, =3       ; R2 ← 2
```

Tilarekisterin bitti Z (divide by Zero)?

Tilarekisterin bitti Z (divide by Zero)?

Kertolaskukäsky on samalla tavalla triviaali, mutta ylivuodon mahdollisuus on huomattavasti suurempi kuin esimerkiksi yhteenlaskussa. Esimerkiksi, jo kahden 16-bittisen luvun kertolasku tuottaa ylivuodon. Käytännön esimerkiksi ylivuototilanteesta kelpaa vaikkapa kertolasku 256000 * 127000. Yhteenlaskun ja kertolaskun ylivuodot aiheuttavat siis aina ylivuotokeskeytyksen ttk-91'ssä. Java-ohjelmien modulo-semantiikan toteuttaminen tähän koneeseen on vähän vaikeata, mutta kyllä sekin onnistuu ylivuotokeskeytyksen käsittelijää sopivasti 'viilaamalla'.

Ttk-91 kokonaislukuoperaatiot

LOAD ("move")

```
LOAD  R3, R1   ; R3 ← R1
```

ADD, SUB

```
ADD  R3, R1   ; R3 ← R3+R1
SUB  R3, =1   ; R3 ← R3-1
```

Tilarekisterin bitti O (overflow)?

MUL

```
MUL   R3, Tbl(R1)   ; R3 ← R3 * Mem[Tbl+(R1)]
```

Tilarekisterin bitti O (overflow)?

DIV, MOD

```
LOAD R1, =14
DIV    R1, =3      ; R1 ← 4

LOAD R2, =14
MOD   R2, =3       ; R2 ← 2
```

Tilarekisterin bitti Z (divide by Zero)?

Tilarekisterin bitti Z (divide by Zero)?

Jakolasku noudattaa koulusta opittuja sääntöjä, eikä siinä voi tapahtua ylivuotoa. Jakolaskun tulos mahtuu aina rekisteriin. Jakolasku operaatiota ei ole määritelty tilanteessa, jossa jakaja on nolla, joten siinä yhteydessä operaatio päättyy nollalla jako -keskeytykseen, joka ilmaistaan tilarekisterin bitillä Z. Jakojäännös saadaan esille omalla MOD-konekäskyllään. Toisin kuin koulussa on opittu, me emme saa samalla kertaa ttk-91'ssä jakolaskun osamäärää ja jakojäännöstä. Jakolasku pitää tosiaankin suorittaa kahteen kertaan, jos sekä osamäärä että jakojäännös halutaan käyttöön.

Ttk-91 bittioperaatiokäskyt

AND, OR, XOR, NOT

- kaikille 32 bitille yhtäaikaa
- yksi bitti (bittipari) kerrallaan
- NOT'illa vain yksi operandi (jälkimmäisellä ei merkitystä)

```
LOAD  R1, =12        ; R1 = 000 ... 000 1 1 0 0
LOAD  R2, =5         ; R2 = 000 ... 000 0 1 0 1

AND   R1, R2         ; R1 = 000 ... 000 0 1 0 0

OR    R1, R2         ; R1 = 000 ... 000 1 1 0 1

XOR   R1, R2         ; R1 = 000 ... 000 1 0 0 1

NOT   R1             ; R1 = 111 ... 111 0 0 1 1
```


Ttk-91'ssä on perusvalikoima bittimanipulaatiokäskyjä. Näihin sisältyvät tässä esitetyt bittioperaatiot ja seuraavalla sivulla esitetyt bittien siirtokäskyt. Kaikki bittioperaatiot tehdään kaikille 32 bitille, bitti tai bittipari kerrallaan siten, että tulosrekisterin kukin bitti lasketaan operandirekisterien vastaavista biteistä. Operandeja ei siis käsitellä tässä kokonaislukuina vaan ainoastaan bitteinä! And-operaation tulos kullekin bitille on siten 1 jos ja vain jos vastaavat bitit molemmissa operandeissa ovat ykkösiä.

Ttk-91 bittioperaatiokäskyt

AND, OR, XOR, NOT

- kaikille 32 bitille yhtäaikaa
- yksi bitti (bittipari) kerrallaan
- NOT'illa vain yksi operandi (jälkimmäisellä ei merkitystä)

```
LOAD  R1, =12      ; R1 = 000 ... 000 1 1 0 0
LOAD  R2, =5       ; R2 = 000 ... 000 0 1 0 1

AND   R1, R2       ; R1 = 000 ... 000 0 1 0 0
OR    R1, R2       ; R1 = 000 ... 000 1 1 0 1
XOR   R1, R2       ; R1 = 000 ... 000 1 0 0 1
NOT   R1           ; R1 = 111 ... 111 0 0 1 1
```


Or-operaation tulos on yksi, jos jompikumpi tai molemmat operandien vastaavista biteistä on yksi.

Ttk-91 bittioperaatiokäskyt

AND, OR, XOR, NOT

- kaikille 32 bitille yhtäaikaa
- yksi bitti (bittipari) kerrallaan
- NOT'illa vain yksi operandi (jälkimmäisellä ei merkitystä)

```
LOAD  R1, =12        ; R1 = 000 ... 000 1 1 0 0
LOAD  R2, =5         ; R2 = 000 ... 000 0 1 0 1

AND   R1, R2         ; R1 = 000 ... 000 0 1 0 0

OR    R1, R2         ; R1 = 000 ... 000 1 1 0 1

XOR   R1, R2         ; R1 = 000 ... 000 1 0 0 1

NOT   R1             ; R1 = 111 ... 111 0 0 1 1
```


Xor-operaation tulos on yksi, jos jompikumpi operandien vastaavista biteistä on yksi, mutta ei molemmat.

Ttk-91 bittioperaatiokäskyt

AND, OR, XOR, NOT

- kaikille 32 bitille yhtäaikaa
- yksi bitti (bittipari) kerrallaan
- NOT'illa vain yksi operandi (jälkimmäisellä ei merkitystä)

```
LOAD  R1, =12        ; R1 = 000 ... 000 1 1 0 0
LOAD  R2, =5         ; R2 = 000 ... 000 0 1 0 1
```

```
AND   R1, R2         ; R1 = 000 ... 000 0 1 0 0

OR    R1, R2         ; R1 = 000 ... 000 1 1 0 1

XOR   R1, R2         ; R1 = 000 ... 000 1 0 0 1

NOT   R1             ; R1 = 111 ... 111 0 0 1 1
```


NOT on unaarinen operaatio, eli sillä on vain yksi operandi, rekisteri Rj. Not-operaatio kääntää kaikki rekisterissä Rj olevat bitit. Huomaa, että ttk-91'ssä käytettyjen kokonaislukujen esitysmuodon vuoksi Not-operaatiota ei voi käyttää negaatio-operaationa. Eli, jos rekisterin R4 kokonaislukuarvo on +54, niin 'NOT R4' käskyn jälkeen rekisterin R4 arvo ei ole -54.

Ttk-91 bittien siirtokäskyt

SHL, SHR, SHRA (SHift Right Arithmetic)

- siirrä bittejä vasemmalle, täytä nollilla

```
LOAD  R1, =13      ; R1 = 000 ... 00 01101
SHL   R1, =1       ; R1 = 000 ... 00 11010
```

- yhden bitin siirto vasemmalle SHL'lla = kerro arvo kahdella
- siirrä bittejä oikealle, täytä nollilla (SHR) tai etumerkillä (SHRA)

```
LOAD  R1, =13      ; R1 = 000 ... 00 01101   = 13
SHR   R1, =1       ; R1 = 000 ... 00 00110   =  6
```

```
LOAD   R1, =-13    ; R1 = 111 ... 11 10011   = -13
SHRA   R1, =1      ; R1 = 111 ... 11 11001   = -7
```

```
LOAD   R1, =-13    ; R1 = 111 ... 11 10011
SHR    R1, =1      ; R1 = 011 ... 11 11001   = +134217721
```

- yhden bitin oikealle SHRA'lla = jaa arvo kahdella

SHL-käsky siirtää työrekisterissä olevia bittejä jälkimmäisen operandin verran vasemmalle, ja täyttää oikealta 0-biteillä. SHL-käskyä voidaan joskus myös käyttää kertolaskun korvikkeena, koska yhden bitin siirto vasemmalle vastaa kokonaisluvuilla luvun kertomista kahdella. Tätä piirrettä on oikeissa koneissa edullista hyödyntää, koska bittien siirto on huomattavasti helpompaa ja nopeampaa toteuttaa kuin kertolasku.

Ttk-91 bittien siirtokäskyt

SHL, SHR, SHRA (SHift Right Arithmetic)

- siirrä bittejä vasemmalle, täytä nollilla

```
LOAD  R1, =13      ; R1 = 000 ... 00 01101
SHL   R1, =1       ; R1 = 000 ... 00 11010
```

- yhden bitin siirto vasemmalle SHL'lla = kerro arvo kahdella
- siirrä bittejä oikealle, täytä nollilla (SHR) tai etumerkillä (SHRA)

```
LOAD  R1, =13      ; R1 = 000 ... 00 01101   = 13
SHR   R1, =1       ; R1 = 000 ... 00 00110   =  6
```

```
LOAD   R1, =-13    ; R1 = 111 ... 11 10011   = -13
SHRA   R1, =1      ; R1 = 111 ... 11 11001   = -7
```

```
LOAD   R1, =-13    ; R1 = 111 ... 11 10011
SHR    R1, =1      ; R1 = 011 ... 11 11001   = +134217721
```

- yhden bitin oikealle SHRA'lla = jaa arvo kahdella

Bittien siirto oikealle SHR-käskyllä toimii aivan vastaavasti kuin SHL toimii vasemmalle. Täyttöbitit vasemmalta ovat aina nollia. Bittien siirto oikealle toimii nyt jakolaskun tapaan, mutta vain positiivisille luvuille, koska vasemmalta täytetään aina nollia ja vasemmanpuolinen bitti on etumerkkibitti. Sitä varten on olemassa toinen bittiensiirtokäsky SHRA, joka toimii muuten SHR'n tavoin, mutta täyttääkin vasemmalta aina entistä etumerkkibittiä eli vasemmanpuolista bittiä. Nyt yhden bitin siirto oikealle toimii jakolaskuna 2:lla myös negatiivisille luvuille.

Ttk-91 kontrollinsiirtokäskyt

JUMP

JUMP Loop **JUMP AfterElse**

COMP

- asettaa tilarekisteriin vertailun tuloksen: L, E tai G

COMP R3, =27 **COMP R2, X**

JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE

- perustuu tilarekisterin arvoon, joka taas perustuu viimeksi suoritettuun COMP käskyyn

JGRE Loop **JNEQU Ohoh**

JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS

- perustuu työrekisterin arvoon

JPOS R1, Loop

CALL, EXIT **aliohjelman kutsu ja sieltä paluu - käsitellään myöhemmin**

SVC, IRET **käyttöjärjestelmäpalvelun kutsu ja sieltä paluu**

SVC SP, =HALT ; ohjelman suoritus päättyy normaalisti

Normaalisti seuraavaksi suoritettava käsky on aina nykyistä konekäskyä seuraavassa muistipaikassa. Kontrollinsiirtokäskyillä muutetaan tätä oletusarvoista toimintaa. Ehdottomalla hyppykäskyllä JUMP kontrolli siirtyy aina käskyssä nimettyyn paikkaan koodia. Tätä käytetään yleensä looppien ja if-then-else -rakenteiden toteutuksessa. Myös konekielisessä ohjelmoinnissa on huonoa tyyliä hyppiä sinne-tänne koodissa jump-käskyllä ilman jotain ylevämpää perustelua. Koodista tulisi tällöin hyvin vaikealukuista ja mahdollisten virheiden havaitseminen tulisi vaikeaksi.

Ttk-91 kontrollinsiirtokäskyt

JUMP

```
JUMP Loop
JUMP AfterElse
```

COMP

- asettaa tilarekisteriin vertailun tuloksen: L, E tai G

```
COMP  R3, =27
COMP  R2, X
```

JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE

- perustuu tilarekisterin arvoon, joka taas perustuu viimeksi suoritettuun COMP käskyyn

```
JGRE  Loop
JNEQU  Ohoh
```

JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS

- perustuu työrekisterin arvoon

```
JPOS  R1, Loop
```

CALL, EXIT — **aliohjelman kutsu ja sieltä paluu - käsitellään myöhemmin**

SVC, IRET — **käyttöjärjestelmäpalvelun kutsu ja sieltä paluu**

```
SVC  SP, =HALT  ; ohjelman suoritus päättyy normaalisti
```


Yleinen tapa suorittaa mikä tahansa vertailu on ensin vertailla kahta arvoa COMP-käskyllä ja sitten seuraavalla ehdollisella hyppykäskyllä tutkia vertailun tulos ja mahdollisesti haarautua koodissa ehdon mukaan. Haarautumisvaihtoehtoja on kuusi ja niitä kaikkia varten on oma konekäskynsä.

Ttk-91 kontrollinsiirtokäskyt

JUMP

JUMP Loop **JUMP AfterElse**

COMP

- asettaa tilarekisteriin vertailun tuloksen: L, E tai G

COMP R3, =27 **COMP R2, X**

JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE

- perustuu tilarekisterin arvoon, joka taas perustuu viimeksi suoritettuun COMP käskyyn

JGRE Loop **JNEQU Ohoh**

JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS

- perustuu työrekisterin arvoon

JPOS R1, Loop

CALL, EXIT **aliohjelman kutsu ja sieltä paluu - käsitellään myöhemmin**

SVC, IRET **käyttöjärjestelmäpalvelun kutsu ja sieltä paluu**

SVC SP, =HALT ; ohjelman suoritus päättyy normaalisti

Edellisessä menetelmässä on se huono puoli, että vertailuun tarvitaan aina kaksi konekäskyä. Toinen vertailtavista arvoista on usein nolla, ja tätä erikoistilannetta varten on oma käskyjoukkonsa. Tämän avulla nämä usein esiintyvät vertailutilanteet voidaan toteuttaa yhdellä konekäskyllä, joka vertaa työrekisterin arvoa nollaar ja haarautuu sen mukaan. Jälleen kaikkia kuutta eri vertailuvaihtoa vastaa kutakin oma konekäskynsä.

Ttk-91 kontrollinsiirtokäskyt

JUMP

JUMP Loop **JUMP AfterElse**

COMP

- asettaa tilarekisteriin vertailun tuloksen: L, E tai G

COMP R3, =27 **COMP R2, X**

JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE

- perustuu tilarekisterin arvoon, joka taas perustuu viimeksi suoritettuun COMP käskyyn

JGRE Loop **JNEQU Ohoh**

JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS

- perustuu työrekisterin arvoon

JPOS R1, Loop

CALL, EXIT **aliohjelman kutsu ja sieltä paluu - käsitellään myöhemmin**

SVC, IRET **käyttöjärjestelmäpalvelun kutsu ja sieltä paluu**

SVC SP, =HALT ; ohjelman suoritus päättyy normaalisti

Aliohjelmia kutsutaan CALL-käskyllä ja niistä palataan EXIT-käskyllä. Näiden käskyjen käyttö on vähän monimutkaisempaa, joten käsittelemme ne kokonaisuudessaan vasta aliohjelmien esittelyn yhteydessä.

Ttk-91 kontrollinsiirtokäskyt

JUMP

JUMP Loop **JUMP AfterElse**

COMP

- asettaa tilarekisteriin vertailun tuloksen: L, E tai G

COMP R3, =27 **COMP R2, X**

JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE

- perustuu tilarekisterin arvoon, joka taas perustuu viimeksi suoritettuun COMP käskyyn

JGRE Loop **JNEQU Ohoh**

JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS

- perustuu työrekisterin arvoon

JPOS R1, Loop

CALL, EXIT **aliohjelman kutsu ja sieltä paluu - käsitellään myöhemmin**

SVC, IRET **käyttöjärjestelmäpalvelun kutsu ja sieltä paluu**

SVC SP, =HALT ; ohjelman suoritus päättyy normaalisti

SVC-käskyllä kutsutaan käyttöjärjestelmäpalvelua, joka nimi (tai tietenkin tunnusta vastaava numero) annetaan käskyn osoitekentässä. Käyttöjärjestelmäpalvelusta palataan etuoikeutetulla erikoiskäskyllä IRET, joka vastaa paljolti aliohjelmista paluukäskyä EXIT. Kaikki ohjelmat lopulta oikein päättyessään kutsuvat HALT-käyttöjärjestelmäpalvelua, jossa kyseinen ohjelma siististi poistetaan järjestelmästä.

Ttk-91 muut suoritettavat käskyt

NOP

- No OPeration, tyhjä käsky, älä tee mitään
- operandeilla ei merkitystä
- voi olla hypyn kohdeosoite
- varaa kuitenkin muistia yhden sanan (32 bittiä)
- suoritetaan kuten muutkin käskyt (kuluttaa aikaa)

```
        ADD  ...
        JZER   R1, Pois
        SUB ...
        DIV ...
         ...
Pois  NOP
```


Oikeissa tietokoneissa on jonkin verran myös muita käskyjä, kuten laskutoimituksia muilla tietotyypeillä tai tietotyyppien tyypinmuunnoskäskyjä. Kaikissa koneissa on kuitenkin myös NOP-käsky, joka voidaan suorittaa, mutta se ei tee mitään. Aina joskus on tilanne, jolloin on tärkeä vain odottaa tekemättä mitään. NOP-käsky on hyvä myös geneerisenä käskynä, jolle voidaan antaa osoite, mutta se ei tee muuten mitään. Suoritusaikana NOP-käsky kuitenkin haetaan muistista tavallisten käskyjen tapaan.

Ttk-91 assembler-kääntäjän ohjauskäskyt

Käsitellään käännöksen aikana, eivät generoi suoritettavia konekäskyjä

EQU - Equal

```
Sata EQU  100
N    EQU  432
```

```
LOAD R1, =N
LOAD R2, =432
```

- antaa symbolille halutun arvon, joka talletetaan symbolitauluun

DC - Data Constant

```
Raja DC  100
X    DC  0
```

- varaa yksi sana muistista data-alueelta, anna sille alkuarvo ja aseta sen osoite annetun symbolin arvoksi
- globaalin muuttujien määrittelyt alkuarvoineen, vakion määrittely

```
LOAD R1, Raja ; DC:llä
LOAD R2, =N   ; EQU:lla
```

DS - Data Segment

```
Tbl  DS 20
```

- varaa monta peräkkäistä sanaa muistista data-alueelta, aseta ensimmäisen sanan osoite annetun symbolin arvoksi, alkuarvot tuntemattomia
- globaalin taulukon tai tietueen tilan varaus

Simulaattorin (esim. Titokone) omat ohjauskäskyt?

- simuloidun muistin koko, input-tiedoston stdin määrittely, jne???

Kaikkiin symbolisiin konekieliin (kuten myös korkean tason kieliin) sisältyy jonkinlainen joukko kääntäjän ohjauskäskyjä. Ne siis ohjaavat kääntäjän toimintaa, eikä niistä generoidu suoritettavia konekäskyjä. Tämän vuoksi niitä sanotaan myöskin 'pseudokäskyiksi' tai 'valekäskyiksi'. Ttk-91'ssä pseudokäskyillä tehdään staattisten muuttujien, vakioiden ja muiden tietorakenteiden tilanvaraukset, jotka toteutetaan ohjelman latausvaiheessa kääntäjän antamien ohjeiden perusteella. Todellisissa konekielissä pseudokäskyillä voi määritellä makroja ja tiettyjen ehtojen vallitessa myös käännettäviä konekäskysarjoja.

Ttk-91 assembler-kääntäjän ohjauskäskyt

Käsitellään käännöksen aikana, eivät generoi suoritettavia konekäskyjä

EQU - Equal

```
Sata EQU  100
N    EQU  432
```

```
LOAD R1, =N
LOAD R2, =432
```

- antaa symbolille halutun arvon, joka talletetaan symbolitauluun

DC - Data Constant

```
Raja DC 100
X    DC 0
```

- varaa yksi sana muistista data-alueelta, anna sille alkuarvo ja aseta sen osoite annetun symbolin arvoksi
- globaalin muuttujien määrittelyt alkuarvoineen, vakion määrittely

```
LOAD R1, Raja ; DC:llä
LOAD R2, =N   ; EQU:lla
```

DS - Data Segment

```
Tbl  DS 20
```

- varaa monta peräkkäistä sanaa muistista data-alueelta, aseta ensimmäisen sanan osoite annetun symbolin arvoksi, alkuarvot tuntemattomia
- globaalin taulukon tai tietueen tilan varaus

Simulaattorin (esim. Titokone) omat ohjauskäskyt?

- simuloidun muistin koko, input-tiedoston stdin määrittely, jne???

EQU-valekäskyllä annetaan tunnukselle haluttu arvo. Tunnus voisi olla vaikkapa ohjelmassa käytetyn taulukon koko tai tietueen kentän suhteellinen sijainti tietueen sisällä. Joka tapauksessa tunnuksen arvo on sellainen vakio, että sitä ei haluta missään vaiheessa vaihtaa ohjelman suoritusaikana. EQU-määrittelyn jälkeen on aivan sama, kirjoitatko symbolisen konekielen koodiin kyseisen tunnuksen vai sen arvon.

Ttk-91 assembler-kääntäjän ohjauskäskyt

Käsitellään käännöksen aikana, eivät generoi suoritettavia konekäskyjä

EQU - Equal

```
Sata EQU  100
N    EQU  432
```

```
LOAD  R1,  =N
LOAD  R2,  =432
```

- antaa symbolille halutun arvon, joka talletetaan symbolitauluun

DC - Data Constant

```
Raja DC  100
X    DC  0
```

- varaa yksi sana muistista data-alueelta, anna sille alkuarvo ja aseta sen osoite annetun symbolin arvoksi
- globaalin muuttujien määrittelyt alkuarvoineen, vakion määrittely

```
LOAD  R1,  Raja  ; DC:llä
LOAD  R2,  =N    ; EQU:lla
```

DS - Data Segment

```
Tbl  DS  20
```

- varaa monta peräkkäistä sanaa muistista data-alueelta, aseta ensimmäisen sanan osoite annetun symbolin arvoksi, alkuarvot tuntemattomia
- globaalin taulukon tai tietueen tilan varaus

Simulaattorin (esim. Titokone) omat ohjauskäskyt?

- simuloidun muistin koko, input-tiedoston stdin määrittely, jne???

DC-valekäskyä käytetään yleisesti globaalin eli kaikkialla näkyvän muuttujan tai vakion määrittelyyn. Muuttujan erottaa vakiosta siitä, että sen arvoa ei saa vaihtaa! Todellisissa konekielissä on usein eri pseudokäsky muuttujien ja vakioiden määrittelyyn, ja kääntäjä voi valvoa, että vakion arvoa ei voi muuttaa. Aliohjelmien paikalliset muuttujat määritellään eri tavoin, mutta ne käsitellään sitten myöhemmin. Ohjelmakoodista ei mitenkään käy selville kyseisen symbolin arvo, vaan kääntäjä päättää sen itsenäisesti.

Ttk-91 assembler-kääntäjän ohjauskäskyt

Käsitellään käännöksen aikana, eivät generoi suoritettavia konekäskyjä

EQU - Equal

```
Sata EQU  100
N    EQU  432
```

```
LOAD  R1, =N
LOAD  R2, =432
```

- antaa symbolille halutun arvon, joka talletetaan symbolitauluun

DC - Data Constant

```
Raja DC  100
X    DC  0
```

- varaa yksi sana muistista data-alueelta, anna sille alkuarvo ja aseta sen osoite annetun symbolin arvoksi
- globaalin muuttujien määrittelyt alkuarvoineen, vakion määrittely

```
LOAD  R1, Raja  ; DC:llä
LOAD  R2, =N    ; EQU:lla
```

DS - Data Segment

```
Tbl  DS  20
```

- varaa monta peräkkäistä sanaa muistista data-alueelta, aseta ensimmäisen sanan osoite annetun symbolin arvoksi, alkuarvot tuntemattomia
- globaalin taulukon tai tietueen tilan varaus

Simulaattorin (esim. Titokone) omat ohjauskäskyt?

- simuloidun muistin koko, input-tiedoston stdin määrittely, jne???

DS-valekäskyllä varataan isompi alue kerrallaan staattiselta data-alueelta. Sen tyypillinen käyttö on globaali, kaikkialla käytettävissä oleva taulukko. Varatun tietorakenteen alkuarvoja ei ole määritelty, joten koodissa ei pitäisi olettaa niiden olevan esimerkiksi nollia. Taulukko pitää ennen käyttöä alustaa sopivalla koodinpätkällä.

Ttk-91 assembler-kääntäjän ohjauskäskyt

Käsitellään käännöksen aikana, eivät generoi suoritettavia konekäskyjä

EQU - Equal

```
Sata EQU 100
N    EQU 432
```

```
LOAD R1, =N
LOAD R2, =432
```

- antaa symbolille halutun arvon, joka talletetaan symbolitauluun

DC - Data Constant

```
Raja DC 100
X    DC 0
```

- varaa yksi sana muistista data-alueelta, anna sille alkuarvo ja aseta sen osoite annetun symbolin arvoksi
- globaalin muuttujien määrittelyt alkuarvoineen, vakion määrittely

```
LOAD R1, Raja ; DC:llä
LOAD R2, =N   ; EQU:lla
```

DS - Data Segment

```
Tbl DS 20
```

- varaa monta peräkkäistä sanaa muistista data-alueelta, aseta ensimmäisen sanan osoite annetun symbolin arvoksi, alkuarvot tuntemattomia
- globaalin taulukon tai tietueen tilan varaus

Simulaattorin (esim. Titokone) omat ohjauskäskyt?

- simuloidun muistin koko, input-tiedoston stdin määrittely, jne???

Eri kääntäjillä voi tietenkin olla omia ohjauskäskyjään. Ttk-91 koneen simulaattorin, Titokone-ohjelmistoon liittyvällä kääntäjällä on muutama ylimääräinen ohjauskäsky esimerkiksi simuloidun muistin koon määrittelyyn ja standarditiedostojen stdin ja stdout määrittelyyn. Titokoneessa on myös toteutettu käyttöjärjestelmäpalveluna standarditiedostojen luku ja kirjoitus. Tämän kurssin osalta riittää kuitenkin perus-pseudokäskyjen EQU, DC ja DS tunteminen.

Ttk-91 symbolinen konekielinen ohjelma Hello

hello.k91

```
X       DC   13
Y       DC   15

MAIN LOAD R1, X
       ADD   R1, Y
       OUT   R1, =CRT
       SVC   SP, =HALT
```

käännös

hello.b91

```
___b91___
___code___
0 3
36175876
287834117
69206016
1891631115
___data___
4 5
13
15
___symboltable___
main 0
y 5
halt 11
crt 0
x 4
___end___
```

lataus

```
0:    36175876
1:   287834117
2:     6920616
3:  1891631115
4:          13
5:          15
```


Kaikki ttk-91 symbolisen konekielen ohjelmat tunnistetaan tiedoston loppuliitteestä 'k91'. Hello-ohjelmassa määritellään ensin kaksi muuttujaa, X ja Y, alkuarvoineen. Pääohjelma alkaa aina ensimmäisestä todellisesta käskystä (ei siis pseudokäskystä). Tässä kyseisen käskyn osoitteena on symboli 'MAIN' vielä tämän painottamiseksi. Pääohjelmassa lasketaan muuttujien X ja Y arvot yhteen ja tulostetaan niiden summa. Ohjelman suoritus päätetään sitten normaalisti SVC-käskyllä. Tämä ja kaikki muutkin esimerkkiohjelmat löytyvät kurssin verkkomateriaalista.

Ttk-91 symbolinen konekielinen ohjelma Hello

hello.k91

```
X       DC   13
Y       DC   15

MAIN LOAD R1, X
       ADD   R1, Y
       OUT   R1, =CRT
       SVC   SP, =HALT
```

käännös

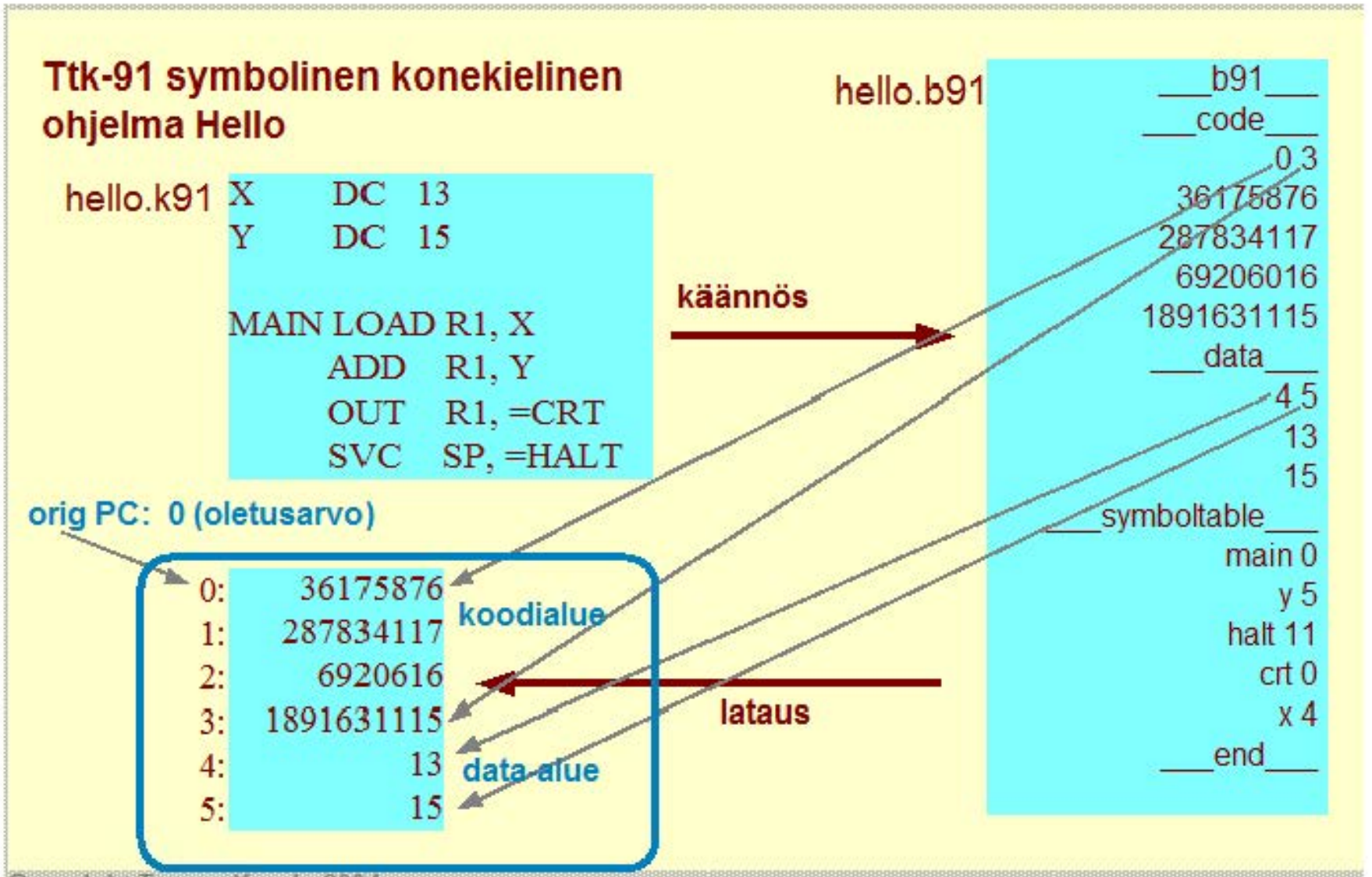
hello.b91

```
___b91___
___code___
0 3
36175876
287834117
69206016
1891631115
___data___
4 5
13
15
___symboltable___
main 0
y 5
halt 11
crt 0
x 4
___end___
```

lataus

```
0:   36175876
1:   287834117
2:   6920616
3:   1891631115
4:   13
5:   15
```


Kun symbolinen konekielinen ohjelma käännetään Titokoneen kääntäjällä, siitä saadaan ladattava 'binääri'-tiedosto eli objektimoduuli. Ttk-91'n objektimoduuli tunnistetaan loppuliitteestä 'b91'. Objektimoduulissa on ensin otsaketieto. Sitä seuraa konekielinen koodialue, joka on merkitty avainsanalla '___code___' ja koodialueen sijainnilla suoritettavassa ohjelmassa. Tämän ohjelman koodi talletetaan siis muistipaikkoihin 0-3. Data alue ja sen alkuarvot ladataar vastaavasti muistipaikkoihin 4-5. Lopuksi on järkevien virheilmoitusten antamista varten tämän ohjelman symbolitaulu.



Ohjelman latauksen jälkeen se on talletettu muistiin (ohjelman omalle muistialueelle), objektimoduulin ilmaisemiin muistipaikkoihin. Koodialue on siis muistipaikoissa 0-3 ja data-alue muistipaikoissa 4-5. Järjestelmä pitää yllä symbolitaulua ohjelmankehitysinformaation ja virheilmoitusten tekemistä varten. Muistiin suoritusta varten talletettu ohjelma muodostuu puhtaasti kokonaisluvuista, vaikka se vastaakin täysin alkuperäistä esitysmuotoaan symbolisella konekielellä.

Ttk-91 symbolinen konekielinen ohjelma Sum

sum.k91

```
; sum - laske annettuja lukuja yhteen, luku 0 on loppumerkki
Luku    DC 0     ; nykyinen luku, alkuarvo 0
Summa   DC 0     ; nykyinen summa, alkuarvo 0

Sum     IN    R1, =KBD     ; ohjelma Sum alkaa käskystä 0
        STORE R1, Luku
        JZER  R1, Done     ; luvut loppu?

        LOAD  R1, Summa  ; Summa <- Summa+Luku
        ADD   R1, Luku
        STORE R1, Summa    ; summa muuttujassa, ei rekisterissä?

        JUMP  Sum

Done    LOAD  R1, Summa    ; tulosta summa ja lopeta
        OUT   R1, =CRT
        SVC   SP, =HALT
```


Tässä on jo vähän monimutkaisempi ohjelmaesimerkki. Sekin löytyy kurssin verkosssa olevasta viitemateriaalista, jos haluat itse kääntää ja suorittaa sen vaikkapa Titokoneella. Ohjelma lukee käyttäjän antamia syötteitä ja laskee niiden summan. Syöte '0' toimii lopetusmerkkinä, jonka jälkeen ohjelma tulostaa aikaisempien lukujen summan. Huomaa tyhjät rivit ja kommentoinnit, joiden avulla koodista saadaan helpommin luettavaa. Laita omaan koodiisikin sopivasti kommentteja, mutta ei liikaa. Huomaa, että koodi käyttää vain yhtä rekisteriä, rekisteri R1'stä. Konekielisessä koodissa rekistereiden käytön optimointi on tärkeätä.

Ttk-91 koneen simulaattori Titokone

Toimii kuten oikea 'kone' toimisi, eli suorittaa ladattuja ttk-91 ohjelmia

Graafinen käyttöliittymä

I/O käyttöliittymän kautta

Ohjelmien valinta, käännös (+ automaattinen linkitys), lataus ja suoritus

Ohjelmien editointi

- voi olla parempi editoida 'off-line' tavallisella tekstieditorilla

Suoritus mahdollista myös käsky kerrallaan, kommentoinnin kera ja/tai animoinnin kera

Ttk-91 koneen simulaattori Titokone on aika laaja ohjelmisto. Sen avulla voidaan esimerkiksi suorittaa ttk-91 ohjelmia samalla tavalla kuin mitä niitä suoritettaisiin todellisessa laitteistossa todellisessa käyttöjärjestelmässä. Titokoneeseen sisältyy siis myös käyttäjärjestelmäpalikoita, kuten kääntäjä ja lataaja. Ohjelmamme ovat niin yksinkertaisia, että mitään linkitystä ei tarvita. Käännöksessä ja käskyjen suorituksessa on useita eri optiota sen mukaan, miten paljon ylimääräisiä detaljeja halutaan näkyviin. Titokone ja sen käyttö on tarkemmin speksattu kurssin harjoitustehtävissä ja verkkomateriaalissa

Yhteenveto

Ttk-91 tietokoneen yleiskuva ja käskykanta-arkkitehtuuri

Ttk-91 ohjelmat

Titokone simulaattori ttk-91 tietokoneelle

Seuraavaksi

Luento 3: ohjelmointi ttk-91 symbolisella konekielellä

Luento 4: aliohjelmien toteutus ttk-91 symbolisella konekielellä

Olemme nyt käyneet läpi esimerkkitietokoneen yleiskuvan ja sen käskykanta-arkkitehtuurin. Näytimme esimerkin vuoksi muutaman yksinkertaisen ohjelman, mutta itse ohjelmointiin pääsemme vasta myöhemmin. Annoimme myös lyhyen yleiskuvan Titokone simulaattorista, johon tutustutaan harjoitustehtävien myötä paljon tarkemmin jatkossa. Seuraavassa luennossa 3 käsittelemme ohjelmoinnin peruskäsitteiden toteuttamista ttk-91 symbolisella konekielellä ja luennossa 4 aliohjelmien toteutuksen.