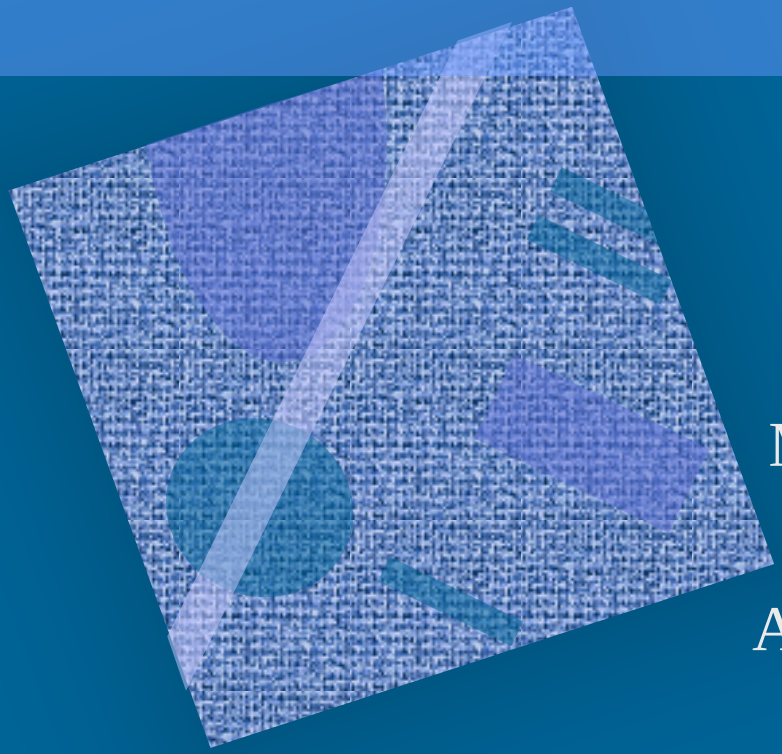
Luvut 5 ja 6
Kertausluento 3

Ttk-91 konekielinen ohjelmointi



Aritmetiikka

Tietorakenteet

Kontrolli, loopit

Monimutkaiset tietorakenteet

Aliohjelman parametrit

Aktivaatiotietue (AT), AT-pino

Aliohjelman kutsu ja paluu

Ohjelmoinnin peruskäsitteet

- Aritmeettinen lauseke
  - Miten tehdä laskutoimitukset?
- Yksinkertaiset tietorakenteet
  - Tiedonosoitusmoodi tukee suoraan
  - Yksiulotteiset taulukot, tietueet
- Kontrolli – mistä seuraava käsky?
  - Valinta: if-then-else, case
  - Toisto: for-silmukka, while-silmukka
  - Aliohjelmat (luento 4), virhetilanteet (keskeytykset)
- Monimutkaiset tietorakenteet
  - Listat, moniulotteiset taulukot
  - Laske ensin tiedon osoite, sitten viittaa tietoon

TTK-91 operaatiot

- Muistiinviittaukset
  - tavalliset: load & store, aritmetiikan yhteydessä
  - pino-operaatiot (aliohjelmien toteuttamista varten)
- I/O käskyt
- Kokonaislukuoperaatiot
- Loogiset operaatiot totuusarvoille
- Bittien siirtokäskyt (shift instructions)
- Kontrollin siirtokäskyt
  - mistä löytyy seuraavaksi suoritettava käsky? (ellei se ole seuraavassa muistipaikassa)
- Muut käskyt

Tiedon osoitusmuodot TTK-91

- Vaihtelee vain jälkimmäiselle operandille
  - Ensimmäinen operandi on aina rekisteri
- Välitön operandi (ei muistiosoitusta)
  - OPER Rj, =ADDR(Ri) M = 0 = 0b00
  - Jälkimmäinen operandi: ADDR+Ri
  - Kumpi vain voi puuttua (ADDR=0 tai Ri=R0)
- Suora (indeksoitu) muistiosoitus
  - OPER Rj, ADDR (Ri) M = 1 = 0b01
  - Jälkimmäinen operandi: Mem(ADDR+Ri)
- Epäsuora (indeksoitu) muistiosoitus
  - OPER Rj, @ADDR(Ri) M = 2 = 0b10
  - Jälkimmäinen operandi: Mem(Mem(ADDR+Ri))

Indeksointi

```
LOAD R4,=Tbl(R3)
LOAD R4,Tbl(R3)
LOAD R4,@Tbl(R3)
```

- Laske aina ensin ns. tehollinen muistiosoite (effective address, EA):

EA= Tbl + (R3) = 201

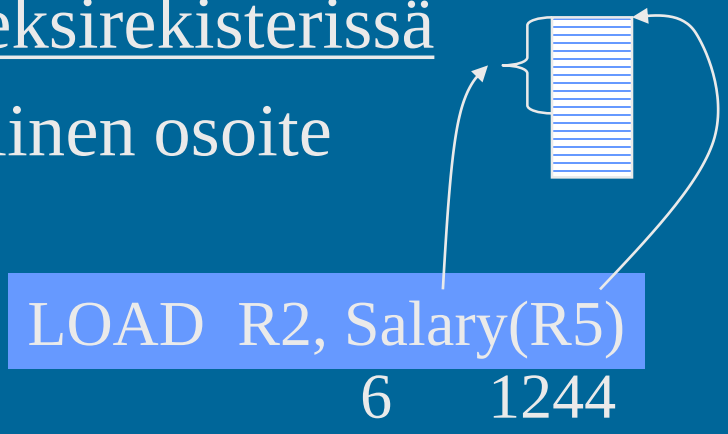
- Sitten katso moodia ja tee niin monta muistinoutoa kuin tarvitaan (tai ei yhtään)
  - ”=”: 0 kpl — R4 ← 201 (välitön)
  - tyhjä: 1 kpl — R4 ← Mem[201] = 11 (indeksoitu)
  - ”@”: 2 kpl — R4 ← Mem[Mem[201]] = Mem[11] = 300 (epäsuora)

pelkkä rekisterin nro @-merkin jälkeen ⇒ vain 1 muistinouto

STORE käsky ⇒ 1 kpl vähemmän noutoja ja yksi tallennus

Indeksoinnin käyttö taulukkojen ja tietueiden yhteydessä

- Taulukot
  - taulukon alkuosoite vakiona
  - taulukon indeksi indeksirekisterissä
- Tietueet
  - tietueen alkuosoite indeksirekisterissä
  - tietueen kentän suhteellinen osoite tietueen sisällä vakiona



olio

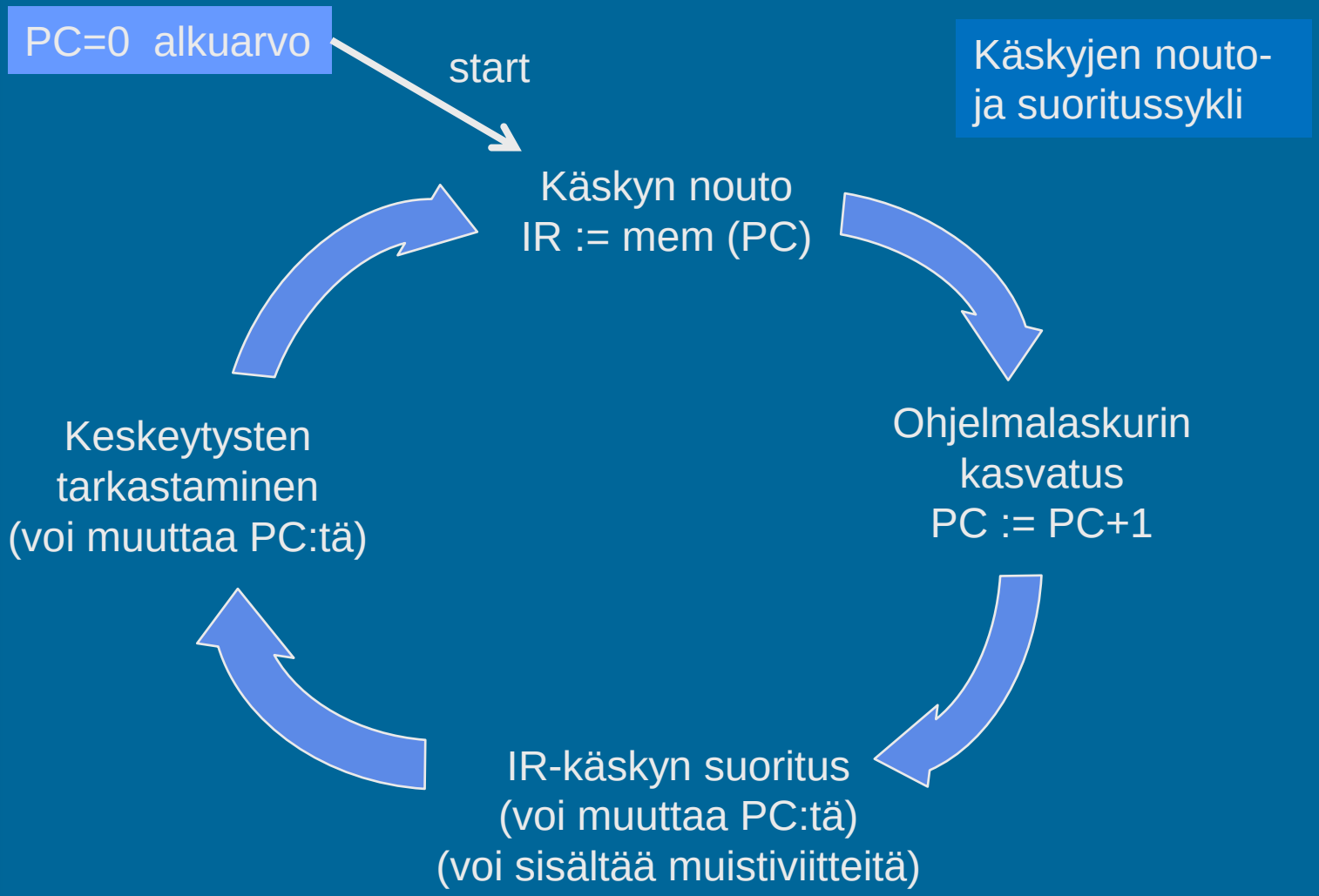
record
object

TTK-91 konekieli

Operaatio 8 bit	Rj 3 b	M 2 b	Ri 3 b	Attribuutti (arvo tai osoite) 16 bittiä

- Kukin käsky 32 bittiä. Oma operaatiokoodi kullakin käskyllä
- Käskyn rekisterien ja attribuuttien tulkinta riippuu käskystä ja osoitusmuodosta (M)
- Tietotyyppi:
  - Vain 32-bittinen kokonaisluku (tai vain bittejä)
  - EI: merkkejä, liukulukuja, totuusarvoja

Suorittimen toiminta



Tiedon sijainti suoritusaikana

- Rekisteri (nopein)
  - kääntäjä (yleensä) päättää, milloin muuttujan arvo on rekisterissä
- Välimuisti (nopea)
  - laitteisto hoitaa automaattisesti viimeksi käytetyille muistialueille
- Muisti (hidas)
  - kääntäjä/lataaja valitsee sijaintipaikan
    - globaali data ohj. latauksen yhteydessä (data-alue)
    - vakiot konekäskyssä (koodi-alue)
  - ohjelma sijoittaa suoritusaikana
    - aliohjelman paikalliset muuttujat ja parametrit pinossa
  - käyttöjärjestelmä sijoittaa suoritusaikana
    - dynaaminen data keossa suorituksen aikana
- Levy, levypalvelin (liian hidas, ei mahdollista)
  - Ladattava muistiin ennen käyttöä

Miten tietoon viitataan?

- Tieto muistissa
  - Muistiosoitteen (esim. 0x6F123456 tai 3459321) avulla
  - Symbolin (esim. HenkNro tai X) avulla symbolista konekieltä käytettäessä
    - symbolin arvo on muistiosoite tai tietueen kentän sijainti
    - HenkNro ≡ 0x6F123456, (heksadesimaali) X ≡ 3459321 (desimaali)
- Tieto välimuistissa
  - Samalla tavalla kuin jos tieto olisi muistissa
  - Viittaushetkellä (konekäskyä suoritettaessa) ei tiedetä, kummasta paikasta tieto lopulta löytyy tai kauanko viittaamiseen kuluu aikaa!
  - Nopeus jossain rekisterin ja muistin nopeuksien välillä
  - Välimuisteja voi olla usea: L1 (pieni, nopein), L2 (isompi), L3 (hyvin iso)
- Tieto rekisterissä
  - Rekisterin osoitteen (esim. nro 6 tai 18) avulla (ttk-91: 0-7)
  - Symbolinen konekieli: R3, FP, F5, SP, SR, I2, jne (ttk-91: R0-R7, SP, FP)
- Tieto konekäskyssä vakiona (suoritushetkellä rekisterissä IR)
  - oletusarvoisesti käskyssä on vain yksi paikka tiedolle (vakio-kenttä)

Tieto ja sen osoite

```
X  DC     12
   ....
   LOAD  R1, =X
   LOAD  R2, X
```

int x =12;

muut-
tujan
X
osoite
on
sym-
bolin
X
arvo

symbolin X arvo
(muuttujan X osoite)

muuttujan X arvo

muisti

230
12345
12556
128765
12222
12
12998

X ≡ 230:

- Muuttujan X osoite on 230
- Muuttujan X arvo on 12
- Symbolin X arvo on 230
  - symbolit ovat yleensä olemassa vain käännösaikana
  - virheilmoituksia varten symbolitaulua pidetään joskus yllä myös suoritusaikana

symb.taulu

…
X 230
…

Aritmeettinen lauseke (3)

```
int  a, b, c;
…
b = 34;


a = b + 5 * c;
```

tilan varaus

```
A       DC      0
B       DC      0
C       DC      0
```

koodi

```
        LOAD    R1, =34
        STORE   R1, B
        ....
        LOAD    R1, B
        LOAD    R2, C
        MUL     R2, =5
        ADD     R1, R2
        STORE   R1, A
```

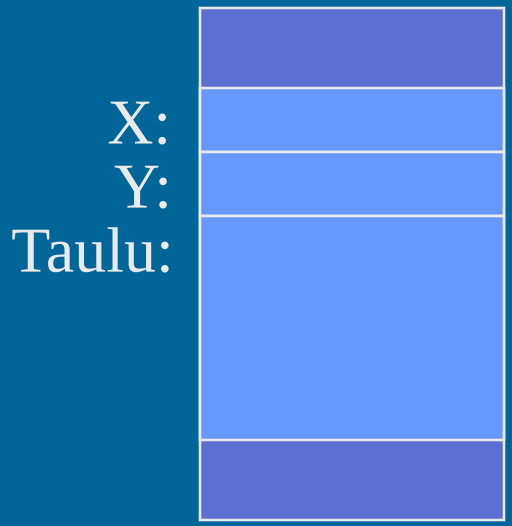
tai:

```
LOAD R1,=5
MUL  R1,C
ADD  R1, B
STORE R1,A
```

Globaalin (kaikkialla näkyvän) 1-ulotteisen taulukon tilan varaus ja käyttö (4)

```
int X, Y;
int Taulu[30];
…
X = 5;
Y = Taulu[X];
```

```
X       DC    0
Y       DC    0
Taulu   DS    30
        …
        LOAD  R1, =5
        STORE R1, X
        LOAD  R1, X
        LOAD  R2, Taulu(R1)
        STORE R2, Y
```



Optimoiva kääntäjä osaisi jättää pois jälkimmäisen ”LOAD R1, X” käskyn

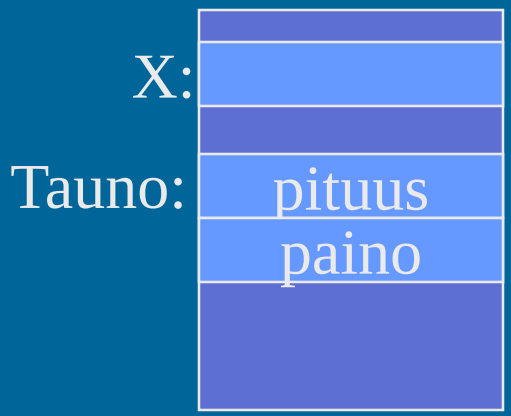
Globaalin tietueen tilan varaus ja käyttö (3)

```
int X;
struct Tauno {
        int Pituus;
        int Paino;
        }
…
X = Tauno.Paino
```

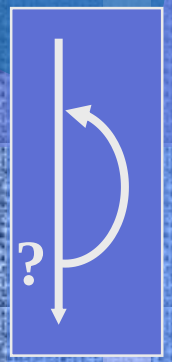
Kentän ”Paino” suhteellinen osoite tietueen Tauno sisällä

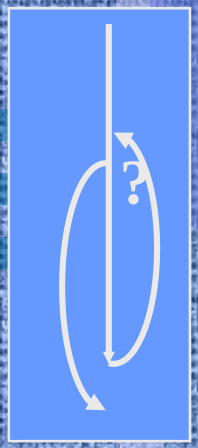
```
X       DC      0
Tauno   DS      2
Pituus  EQU     0
Paino   EQU     1
…
LOAD R1,=Tauno
LOAD R2, Paino(R1)
STORE R2, X
```

Tietueen osoite on sen ensimmäisen sanan osoite



For lauseke






```
for (int i=20; i < 50; ++i)
    T[i] = 0;
```

```
T       DS      50
I       DC      0
        …
        LOAD  R1, =20
        STORE R1, I
Loop    LOAD  R2, =0
        LOAD  R1, I
        STORE R2, T(R1)
        LOAD  R1, I
        ADD   R1, =1
        STORE R1, I
        LOAD  R3, I
        COMP  R3, =50
        JLES  Loop
```

alku

runko

lisää

testaa

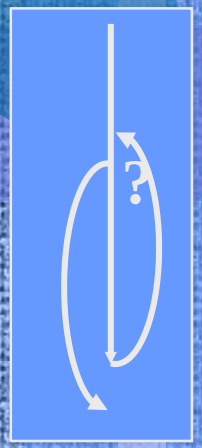
Olisiko parempi pitää i:n arvo rekisterissä? Miksi? Milloin?

Mikä on i:n arvo lopussa? Onko sitä olemassa?

Entä jos toisenlainen toisto-semantiikka?

Keskustele

While-do -lauseke



```
X = 14325;
Xlog = 1;
Y = 10;
while (Y < X) {
        Xlog++;
        Y = 10*Y
        }
```

```
        LOAD   R1, =14325
        STORE  R1, X
        LOAD   R1, =1   ; R1=Xlog
        LOAD   R2, =10  ; R2=Y
While   COMP   R2, X
        JNLES  Done
        ADD    R1, =1
        MUL    R2, =10
        JUMP   While
Done    STORE  R1, Xlog ; talleta tulos
        STORE  R2, Y
```

Mitä kannattaa pitää muistissa?

Mitä kannattaa pitää missä rekisterissä ja milloin?

X in R3?

Taulukon indeksitarkistus

```
for (int i=20; i < 50; ++i)
        T[i] = 0;
```

```
I       DC    0

T       DS    50  ; data
Tsize   DC    50  ; koko
        …
```

Voisiko toiston kontrollia ja indeksin tarkistusta yhdistää? Optimoiva kääntäjä osaa!

```
        LOAD  R1, =20
        STORE R1, I
Loop    LOAD  R2, =0
        LOAD  R1, I
        JNNEG R1, ok1
        SVC   SP,=BadIndex
ok1     COMP  R1, Tsize
        JLES  ok2
        SVC   SP, =BadIndex
ok2     STORE R2, T(R1)
        LOAD  R1, I
        ADD   R1, =1
        STORE R1, I ; 50 OK!
        LOAD  R3, I
        COMP  R3, =50
        JLES  Loop
```

Moniulotteiset taulukot

T:

34	**57**	**76**
21	76	23
24	56	876
54	75	777

T:

T[0][0]=34
T[0][1]=57
T[0][2]=76
T[1][0]=21
T[1][1]
T[1][2]
T[2][0]
T[...][...]

- Talletus riveittäin
  - C, Pascal, Java?
- Talletus sarakkeittain
  - Fortran

T:

T[0][0]=34
T[1][0]=21
T[2][0]=24
T[3][0]=54
T[0][1]=57
T[1][1]
T[2][1]
T[...][...]

- (On muitakin tapoja, esim.
  - Kukin rivi allokoitu erikseen
  - T[i] on rivin i osoite, jne …)
- 3- tai useampi ulotteiset
  - vastaavalla tavalla!

Keskustele

Monimutkaisten tietorakenteiden viittaukset

```
int T[10, 20];
```

```
T     ds  200
Trows equ 10
Tcols  equ 20
```

```
T[i,j] = 34;
```

- Laske ensin viitatun alkion osoite (tietorakenteen sisällä)
- Tee viittaus indeksointia käyttäen
  - Samalla tavalla kuin 1-ulotteiselle taulukolle

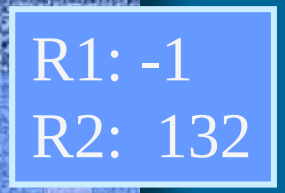
riveittäin

```
Load r1, i
Mul   r1,  =Tcols
Add   r1, j

Load  r2,=34
Store r2, T(r1)
```

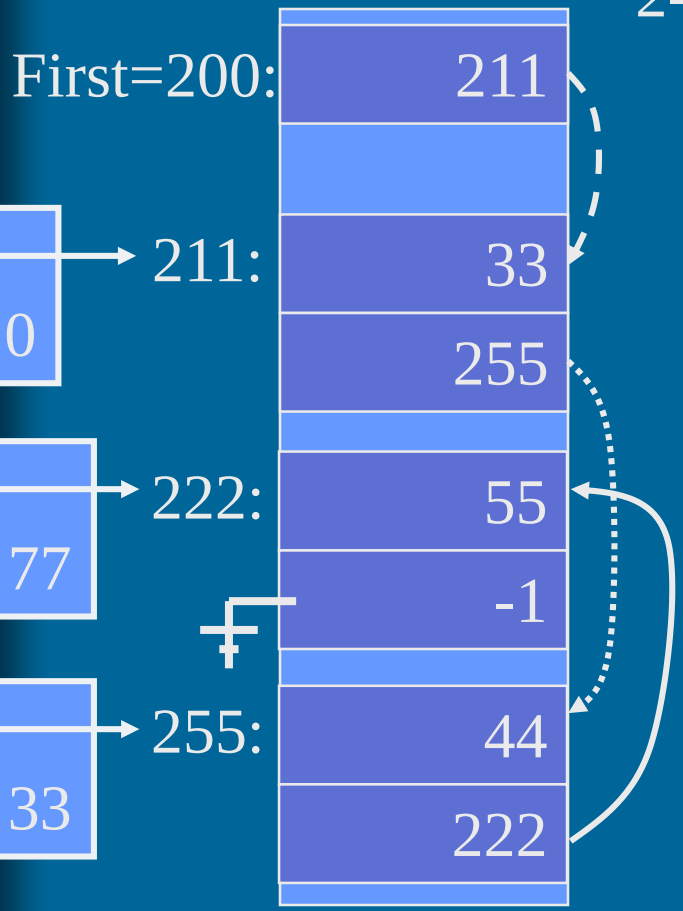
sarakettain

```
Load r1, j
Mul   r1,  =Trows
Add  r1, i

Load  r2,=34
Store r2, T(r1)
```

Linkitetty lista



2-sanainen tietue



list_sum.k91

```
Data    EQU   0 ; suht. osoite
Next    EQU   1
Sum     DC    0
Main    LOAD  R1, First ; ptrRec
        JNEG  R1, Done
        LOAD  R2,=0  ; sum
Loop    ADD   R2, Data(R1)
        LOAD  R1, Next(R1)
        JNNEG R1, Loop
Done    STORE R2, Sum
        SVC   SP, =HALT
```

Virhe, bugi! Missä?

Keskustele

Koodin generointi

- Kääntäjän viimeinen vaihe
  - voi olla 50% käännösajasta
- Tavallisen koodin generointi
  - alustukset, lausekkeet, kontrollirakenteet
- Optimoidun koodin generointi
  - käännös kestää (paljon) kauemmin
  - **suoritus tapahtuu (paljon) nopeammin**
  - milloin globaalin/paikallisen muuttujan X arvo kannattaa pitää rekisterissä ja milloin ei?
  - missä rekisterissä X:n arvo kannattaa pitää?
    - joskus R1:ssä, joskus R5:ssä, joskus ei missään!

Aliohjelmatyypit

- Korkean tason ohjelmointikielen käsitteet
  - Aliohjelma, proseduuri
    - Parametrit (sisäänmeno- ja ulostuloparametrit)
  - Funktio
    - Parametrit, paluuarvo
  - Metodi
    - Parametrit, ehkä paluuarvo
- Konekielitason vastaava käsite
  - Aliohjelma
    - Parametrit ja paluuarvo(t)

Aliohj. parametrit ja paluuarvo

- Muodolliset parametrit
  - Määritelty aliohjelmassa ohjelmointihetkellä
  - Tietty järjestys ja tyyppi
  - Paluuarvot
    - käsittely hyvin samalla tavalla kuin parametreillekin

```
Tulosta (int x, y)
void Tulosta (int x, y)

Laske(int x): int
int Laske(int x)
```

- Todelliset parametrit ja paluuarvo
  - Todelliset parametrit sijoitetaan muodollisten parametrien paikalle kutsuhetkellä suoritusaikana
  - Paluuarvo saadaan paluuhetkellä ja sitä käytetään, kuten mitä tahansa arvoa

```
Tulosta (5, apu+1);
x = Laske( y+234);
```

Aliohjelman parametrityypit

- Arvoparametri
  - Välitetään todellisen parametrin arvo (eli sen kopio) kutsuhetkellä
  - Arvo voidaan lukea
  - Alkuperäistä arvoa ei voi muuttaa, mutta arvon kopiota voi muuttaa
- Viiteparametri
  - Välitetään todellisen parametrin osoite
  - Arvo ja osoite voidaan lukea, arvoa voi muuttaa osoitteen avulla
  - Aliohjelma voi muuttaa pääohjelman dataa

Aliohjelmien toteutuksen osat

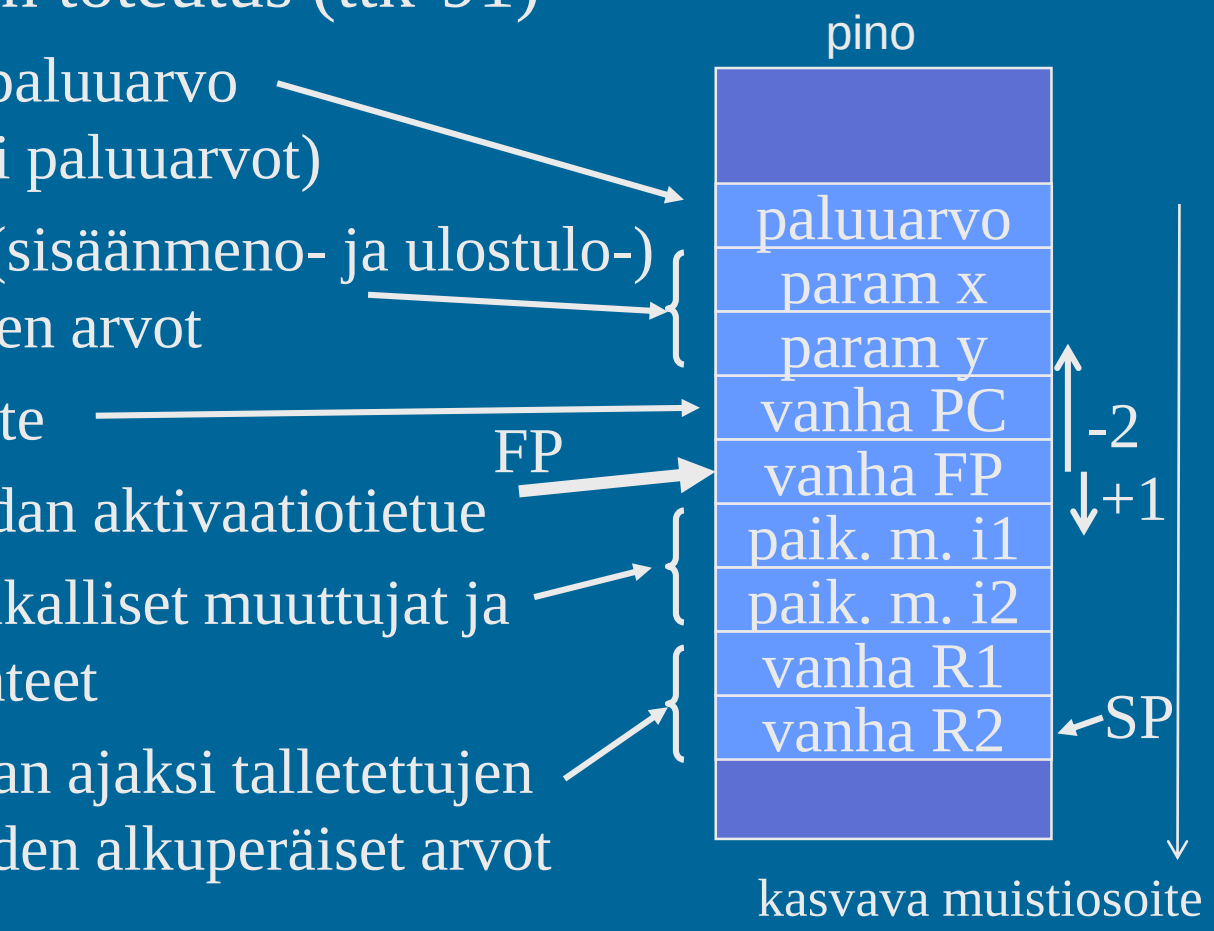
- Paluuosoite
  - kutsukohtaa seuraavan käskyn osoite
- Parametrien välitys
- Paluuarvon välitys
- Paikalliset muuttujat
- Rekisterien varaus (allokointi)
  - Kutsuva ohjelman osa haluaa säilyttää käyttämiensä rekisterien arvot!
    - pääohjelma, toinen aliohjelma, sama aliohjelma, metodi, …
  - Aliohjelman pitää aluksi tallettaa (muistiin) käytettävien rekisterien arvot ja lopuksi palauttaa ne ennalleen

Aktivaatiotietue (Aktivointitietue)

(activation record, activation frame)

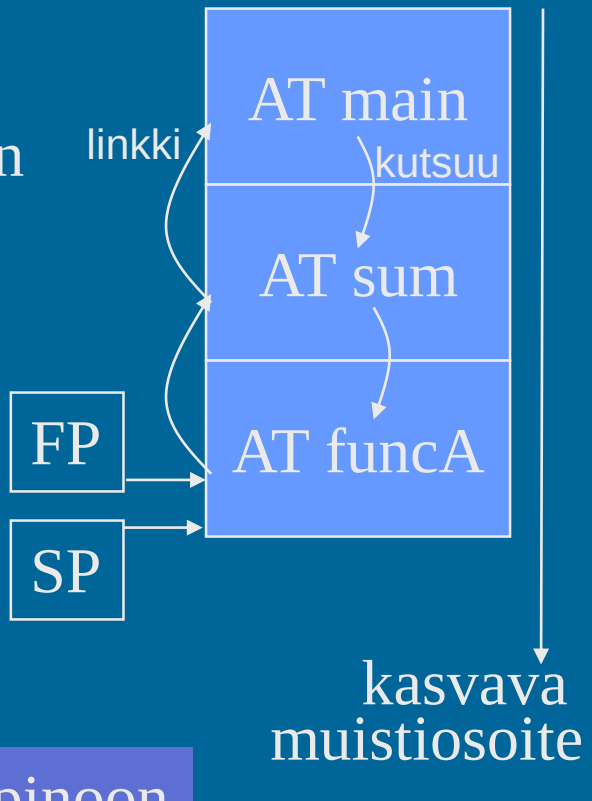
int funcA (int x,y);

- Aliohjelman toteutus (ttk-91)
  - funktion paluuarvo (tai kaikki paluuarvot)
  - kaikkien (sisäänmeno- ja ulostulo-) parametrien arvot
  - paluuosoite
  - kutsukohdan aktivaatiotietue
  - kaikki paikalliset muuttujat ja tietorakenteet
  - aliohjelman ajaksi talletettujen rekistereiden alkuperäiset arvot



Aktivaatiotietuepino muistissa

- Aktivaatiotietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta (muistista)
  - SP (eli R6) osoittaa pinon pinnalle
- Aktivaatiotietuepino
  - FP (R7) osoittaa voimassa olevan AT:n sovittuun kohtaan (ttk-91: vanhan FP:n osoite)
- Pinossa olevaa AT:tä raken-netaan ja puretaan käskyillä:
  - PUSH, POP, PUSHR, POPR
  - CALL, EXIT (SVC, IRET)



Talleta arvo pinoon

Talleta R0-R5 pinoon

Aliohjelmakutsun toteutus

- Toteutus jaettu eri yksiköille
  - varaa tilaa paluuarvolle pinosta
  - laita parametrit (arvot tai osoitteet) pinoon
  - talleta vanha PC ja FP, aseta uudet PC ja FP
  - varaa tilaa paikallisille muuttujille
  - talleta käytettävien rekistereiden vanhat arvot pinoon
  - (itse aliohjelman toteutus – varsinainen työ)
  - palauta rekistereiden arvot
  - vapauta paikallisten muuttujien tila
  - palauta PC ja FP
  - vapauta parametrien tila
  - ota paluuarvo pinosta

Kutsuva rutiini

CALL käsky

prolog

Kutsuttu rutiini

EXIT käsky

epilog

Kutsuva rutiini

Aliohjelmaesimerkki (13)

```
int fA (int x, y)
{
        int z = 5;

        z = x * z + y;
        return (z);
}

…

T = fA (200, R);
```

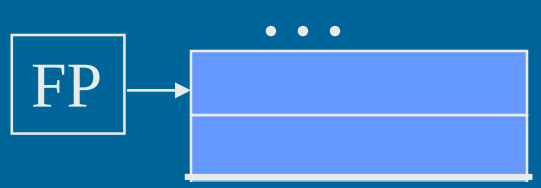
käyttö:

```
R       DC  24
…
PUSH  SP,=0 ; tila paluuarvolle
PUSH SP, =200
PUSH  SP, R

CALL  SP, fA

POP     SP, R1
STORE R1, T
…
```

muistista muistiin!!

talleta PC, FP
aseta PC,
kutsu & paluu
palauta FP, PC
vapauta par tila

2. operandi aina rekisteri



tämän-hetkinen, nykyinen FP

Aliohjelma-esimerkki (12)

```
int fA (int x, y)
{
        int z = 5;

        z = x * z + y;
        return (z);
}

…

T = fA (200, R);
```

aliohjelman toteutus:

```
retfA   EQU  -4 # return value
parX    EQU  -3 # params
parY    EQU  -2
locZ    EQU  1   # local vars

fA      PUSH  SP, =0 ; alloc Z
        PUSH  SP, R1 ; save R1

        LOAD  R1,=5; init Z
        STORE R1, locZ (FP)

        LOAD R1, parX (FP)
        MUL   R1, locZ (FP)
        ADD   R1, parY (FP)
        STORE R1, locZ (FP)
        STORE R1, retfA (FP)
        POP    SP, R1; recover R1
        SUB    SP, =1 ; free Z
        EXIT   SP, =2 ; 2 param.
```

prolog

epilog

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluuarvo

Keskustele

Viiteparametri

Kutsu T = fB(R, S, T)

```
R       DC  24
S       DC 56
T       DC 77
pT      DC  0     ; pointer
…
LOAD   R1, =T     ; initialize pT
STORE  R1, pT
…
PUSH  SP,=0 ; tila paluuarvolle
PUSH SP, R  ; arvoparametri
PUSH  SP, =S ; viiteparametri
PUSH  SP, pT; viiteparametri
CALL  SP, fB
POP       SP, R1
STORE     R1, T
…
```

Y ja Z ovat viiteparametreja:

```
retfB   EQU  -5  ; paluuarvo
parX    EQU  -4  ; arvoparametri
vparY EQU  -3  ; viiteparam
vparZ EQU  -2  ; viiteparam

fB      PUSH  SP, R1 ; save R1

        LOAD R1, parX (FP)
        MUL   R1, @vparY (FP)
        ADD   R1, @vparZ (FP)
        STORE R1, retfB (FP)
        POP    SP, R1; recover R1
        EXIT   SP, =3 ; 3 param.
```

Muuttujan S osoite

Osoitinmuuttujan pT arvo on muuttujan T osoite

KJ-palvelun kutsu (proseduraalisesti)

- Osa palveluista CALL-käskyllä
- Etuoikeutetun palvelun kutsu (esim.) samalla tavalla kuin aliohjelman kutsu
  - CALL käskyn asemesta SVC (SuperVisor Call)
  - Tila paluuarvolle?
  - Parametrit pinossa vai rekistereissä? (palvelukohtaisesti)
  - SVC kutsu
    - Kutsuttavan rutiinin numero operandina
  - IRET paluu
- Paluuarvo (OK, virhe) pois pinosta tarkistusta varten

fOK = ReadBlock (fp, 64)

```
...
PUSH  SP, =0 ;paluuarvo
PUSH  SP, =FileBuffer
PUSH  SP, CharCnt
PUSH  SP, FilePtr

SVC   SP, =ReadBlock

POP     SP, R1
JNZER  R1, FileTrouble
…
```