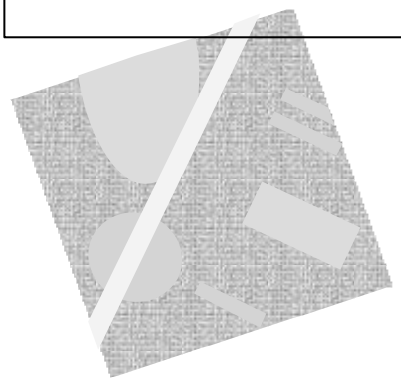
Jakso 4
Aliohjelmien toteutus



Tyypit
Parametrit
Aktivointitietue (AT)
AT-pino
Rekursio

Aliohjelmatyypit (2)

- Korkean tason ohjelmointikielen käsitteet:
  - aliohjelma, proseduuri
    - parametrit
  - funktio
    - parametrit, paluuarvo
  - metodi
    - parametrit, ehkä paluuarvo
- Konekielen tason vastaavat käsitteet:
  - aliohjelma
    - parametrit ja paluuarvo(t)

Parametrit ja paluuarvo (2)

- Muodolliset parametrit
  - määritelty aliohjelmassa ohjelmointihetkellä
  - tietty järjestys ja tyyppi
  - paluuarvot
    - käsittely hyvin samalla tavalla kuin parametreillekin

```
Tulosta (int x, y)
Laske(int x): int
```

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

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

21.5.2002 Copyright Teemu Kerola, K2002 3

Parametrityypit

- Arvoparametri
  - välitetään parametrin arvo kutsuhetkellä
  - arvoa ei voi muuttaa
- Viiteparametri
  - välitetään parametrin osoite
  - arvo voidaan lukea, arvo voidaan muuttaa
- Nimiparametri
  - välitetään parametrin nimi
  - nimi (merkkijono) kuvataan arvoksi kutsuhetkellä
  - semantiikka määräytyy vasta kutsuhetkellä

21.5.2002 Copyright Teemu Kerola, K2002 4

Arvoparametri (10)

Tulosta (A+3, B)

arvon kopio

- Välitetään todellisen parametrin arvo
  - muuttuja, vakio, lauseke, pointteri, olioviite
- Aliohjelma ei voi muuttaa mitenkään todellisena parametrina käytettyä muuttujaa
  - muuttujan X tai B arvo
  - olioviitteen arvo
  - lausekkeen arvo
  - muuta muodollisen parametrin arvoa aliohjelmassa ⇒ muutetaan todellisen parametrin arvon kopiota!
  - Todellisen parametrin ptrX arvoa ei voi muuttaa
  - osoitinmuuttujan osoittamaa arvoa voidaan muuttaa
- Javassa ja C:ssä vain arvoparametreja

```
Tulosta (int y, *ptrX);
{       …
        y = 5;
        *ptrX = 10
}
```


Viiteparametri (4)

~~Summaa (54, Sum)~~

~~pointteri~~

- Välitetään todellisen parametrin osoite
  - muuttujan osoite
- Aliohjelma voi muuttaa parametrina annettua muuttujan arvoa
- Pascalin *var* parametri

Vrt. C:ssä arvoparametrina välitetyn osoitinmuuttujan osoittaman arvon (PrtX, ed. ~~kalvo~~) muuttaminen

```
Summaa (x: int; var cum_sum: int)
{       …
        cum_sum = cum_sum + x;
}
```

~~Summaa(6, Kok_lkm)~~

Nimiparametri (4)

- Välitetään todellisen parametrin nimi
  - merkkijono!
  - Algol 60
  - yleensä makrot
  - sivuvaikutuksia
  - nimiparametri korvataan todellisella parametrilla joka viittauskohdassa

```
void swap (name int x, y)
{
  int t;
  t := x; x := y; y := t;
}
```

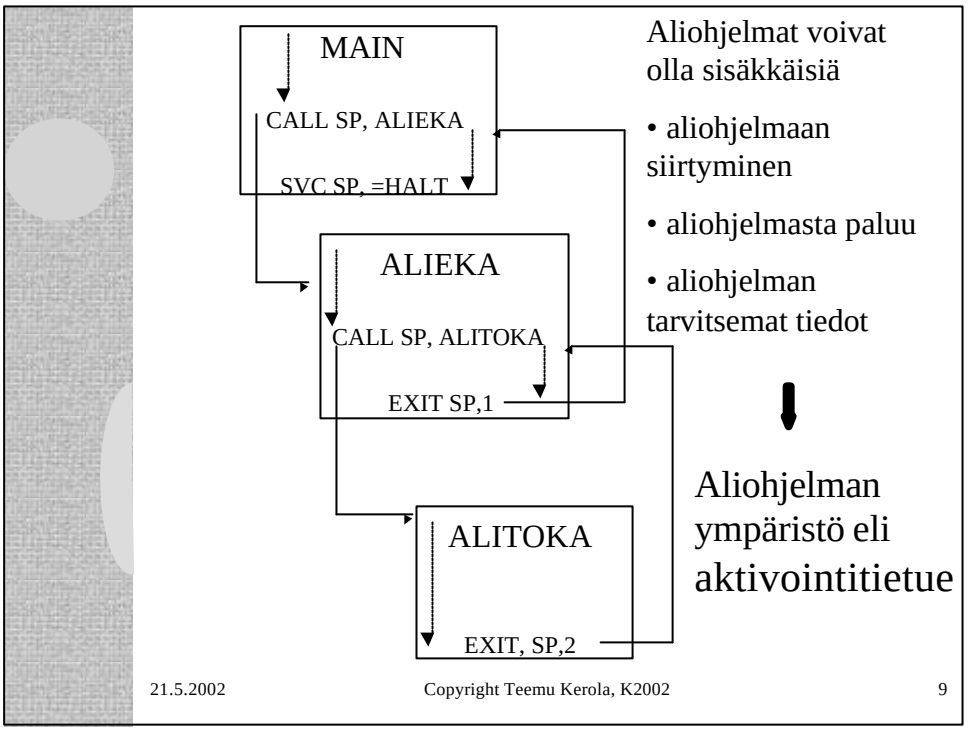
Ei käsitellä enää jatkossa.

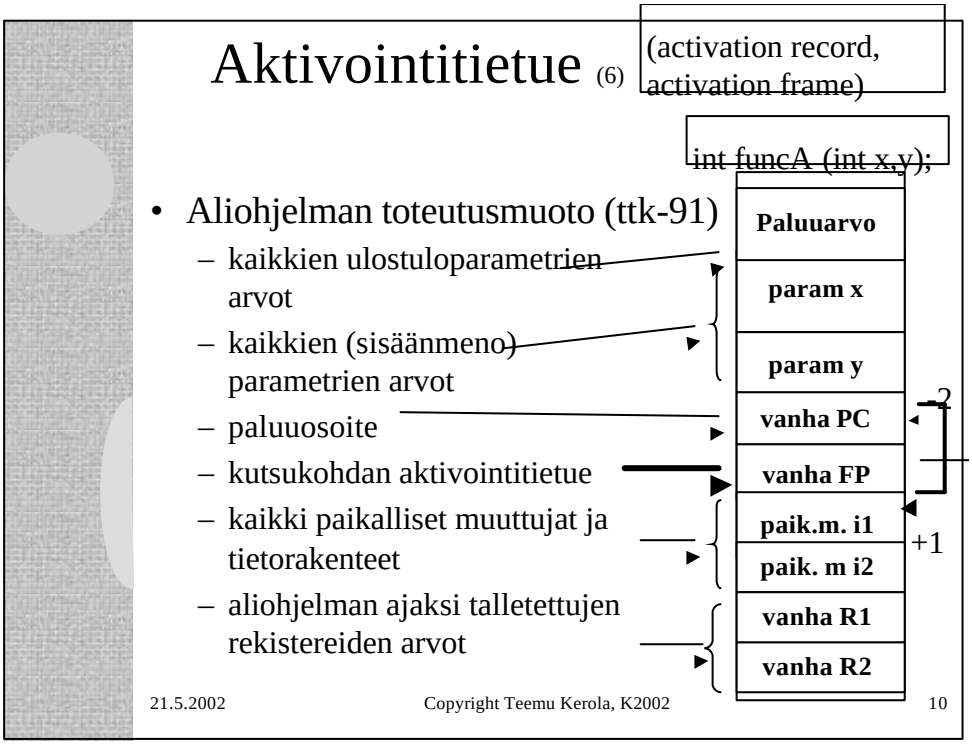

```
swap (n, A[n])   %  n ↔ A[n]
t := n; n := A[n]; A[n] := t;
```

väärä n

Aliohjelmien toteutuksen osat (5)

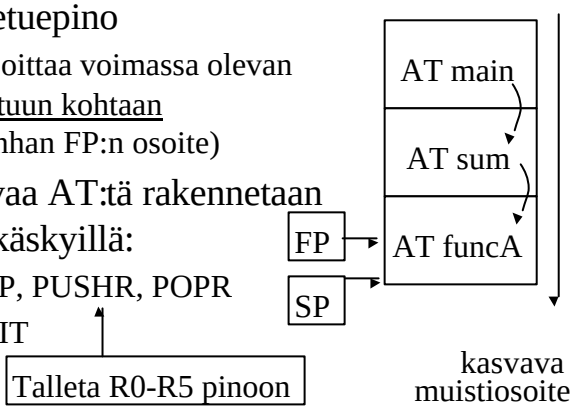
- Paluuosoite
  - kutsukohtaa seuraava käskyn osoite
- Parametrien välitys
- Paluuarvon välitys
- Paikalliset muuttujat
- Rekistereiden allokointi (varaus)
  - kutsuvalla ohjelman osalla voi olla käytössä rekistereitä, joiden arvon halutaan säilyä!
    - pääohjelma, toinen aliohjelma, sama aliohjelma, metodi, …
  - käytettyjen rekistereiden arvot pitää aluksi tallettaa muistiin ja lopuksi palauttaa ennalleen

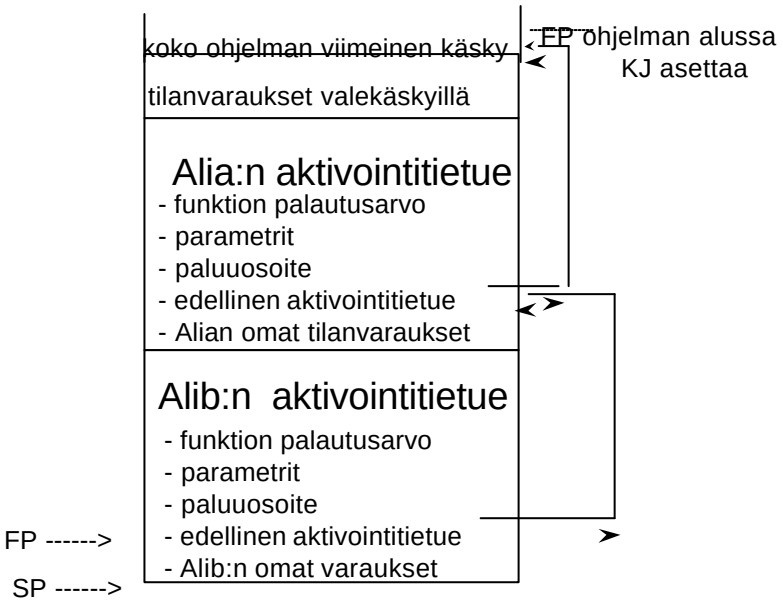

MAIN
CALL SP, ALIEKA
SVC SP, =HALT
ALIEKA
CALL SP, ALITOKA
EXIT SP,1
ALITOKA
EXIT, SP,2
Aliohjelmat voivat olla sisäkkäisiä
• aliohjelmaan siirtyminen
• aliohjelmasta paluu
• aliohjelman tarvitsemat tiedot
Aliohjelman ympäristö eli aktivointitietue
21.5.2002
Copyright Teemu Kerola, K2002
9


Aktivointitietue (6)
(activation record, activation frame)
int funcA (int x,y);
• Aliohjelman toteutusmuoto (ttk-91)
– kaikkien ulostuloparametrien arvot
– kaikkien (sisäänmeno) parametrien arvot
– paluuosoite
– kutsukohdan aktivointitietue
– kaikki paikalliset muuttujat ja tietorakenteet
– aliohjelman ajaksi talletettujen rekistereiden arvot
Paluuarvo
param x
param y
vanha PC
vanha FP
paik.m. i1
paik. m i2
vanha R1
vanha R2
-2
+1
21.5.2002
Copyright Teemu Kerola, K2002
10

Aktivointitietueiden hallinta (4)

- Aktivointitietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta
  - **SP** (=R6) osoittaa pinon pinnalle
- Aktivointitietuepino
  - **FP** (R7) osoittaa voimassa olevan AT:n sovittuun kohtaan (ttk-91: vanhan FP:n osoite)
- Pinossa olevaa AT:tä rakennetaan ja puretaan käskyillä:
  - PUSH, POP, PUSHR, POPR
  - CALL, EXIT





Pinon tila ennen Alib:sta poistumista

12

Aliohjelman käytön toteutus (12)

- Toteutus jaettu eri yksiköille

Kutsuva rutiini
- varaa tilaa paluuarvolle pinosta
- laita parametrit (arvot tai osoitteet) pinoon

CALL käsky
- talleta PC ja FP

Kutsuttu rutiini
- talleta käytettyjen rekistereiden arvot pinoon
- varaa tilaa paikallisille muuttujille

prolog

Kutsuttu rutiini
- (itse aliohjelman toteutus)

EXIT käsky
- vapauta paikallisten muuttujien tila
- palauta rekistereiden arvot
- palauta PC ja FP

epilog

Kutsuva rutiini
- vapauta parametrien tila
- ota paluuarvo pinosta

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 ; tilaa paluuarvolle
PUSH SP, =200
PUSH  SP, R
CALL  SP, fA
POP     SP, R1
STORE R1, T
…
```

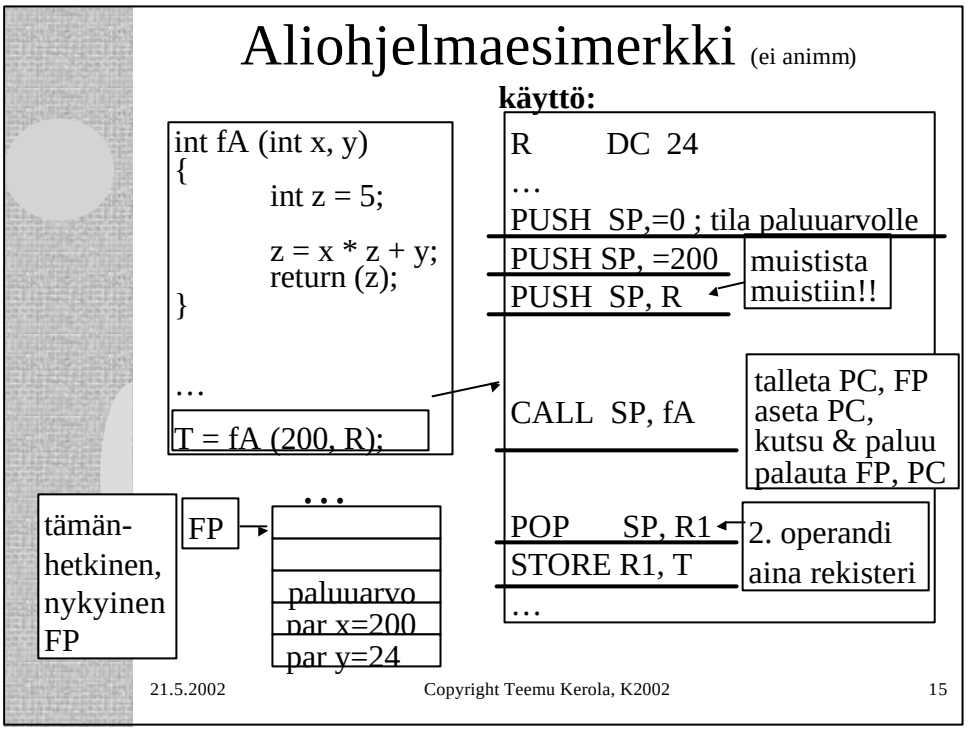
muistista muistiin!!

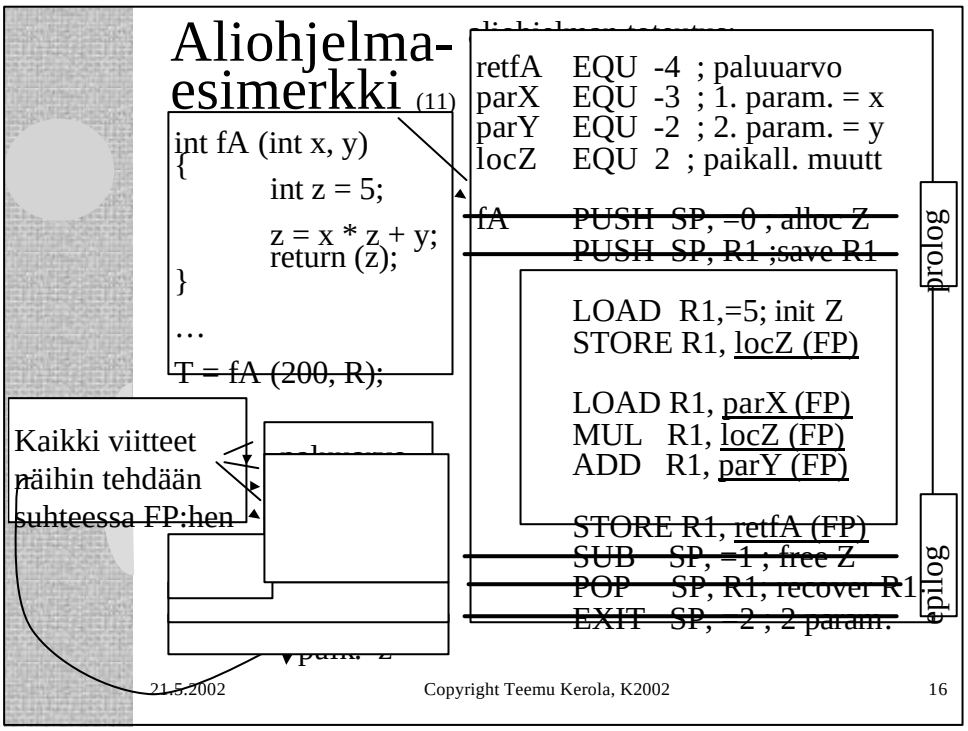
talleta PC, FP
aseta PC,
kutsu & paluu
palauta FP, PC

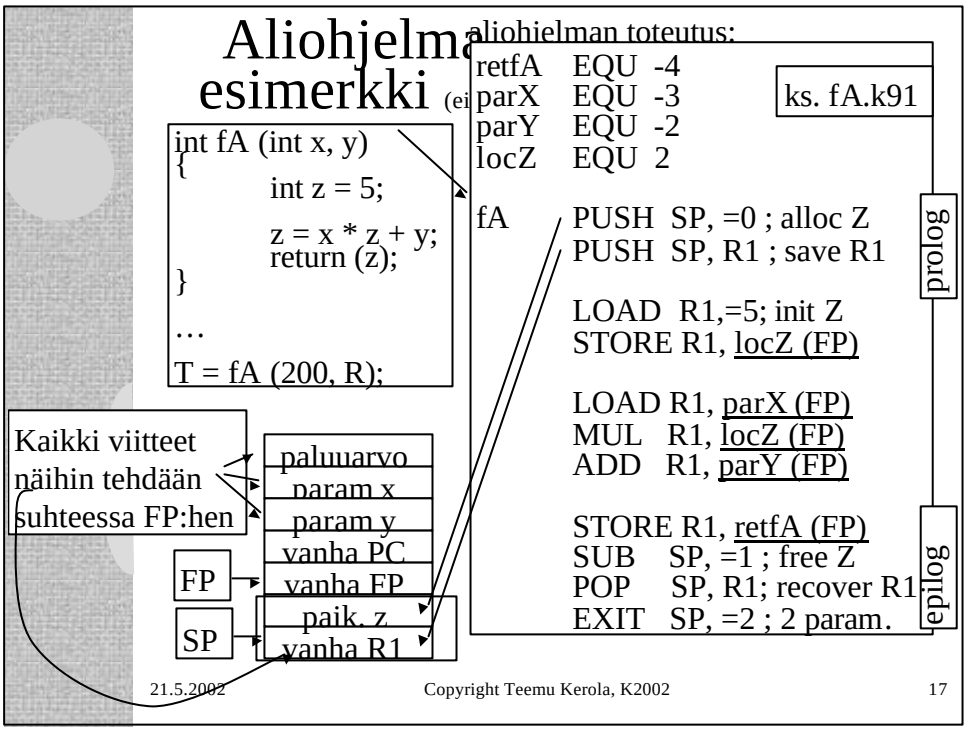
2. operandi aina rekisteri

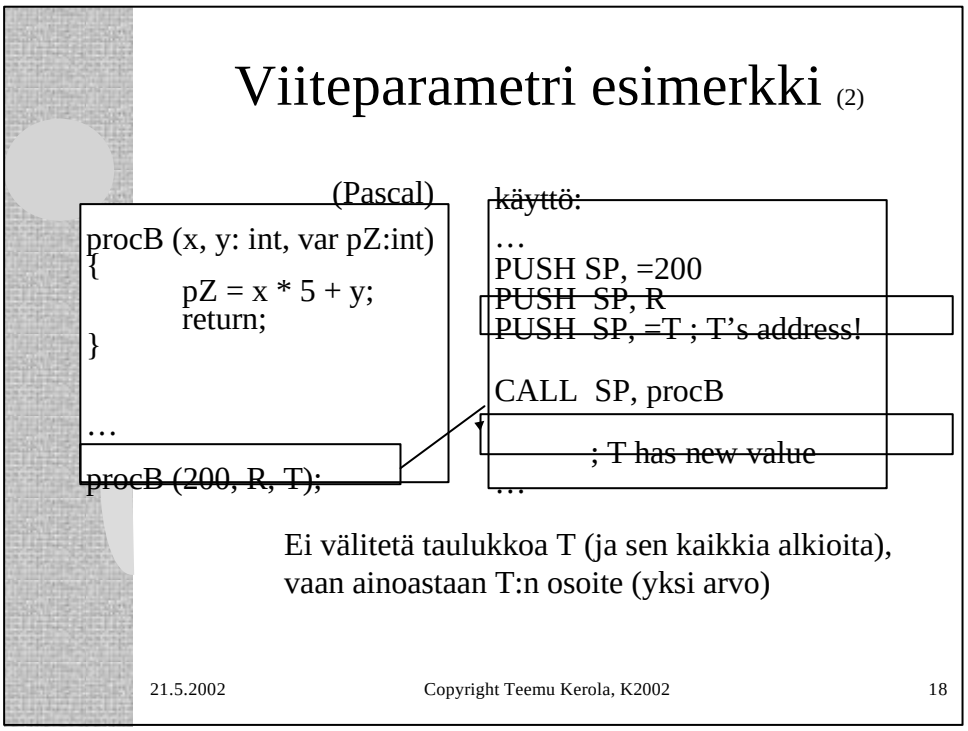
tämän-hetkinen, nykyinen FP

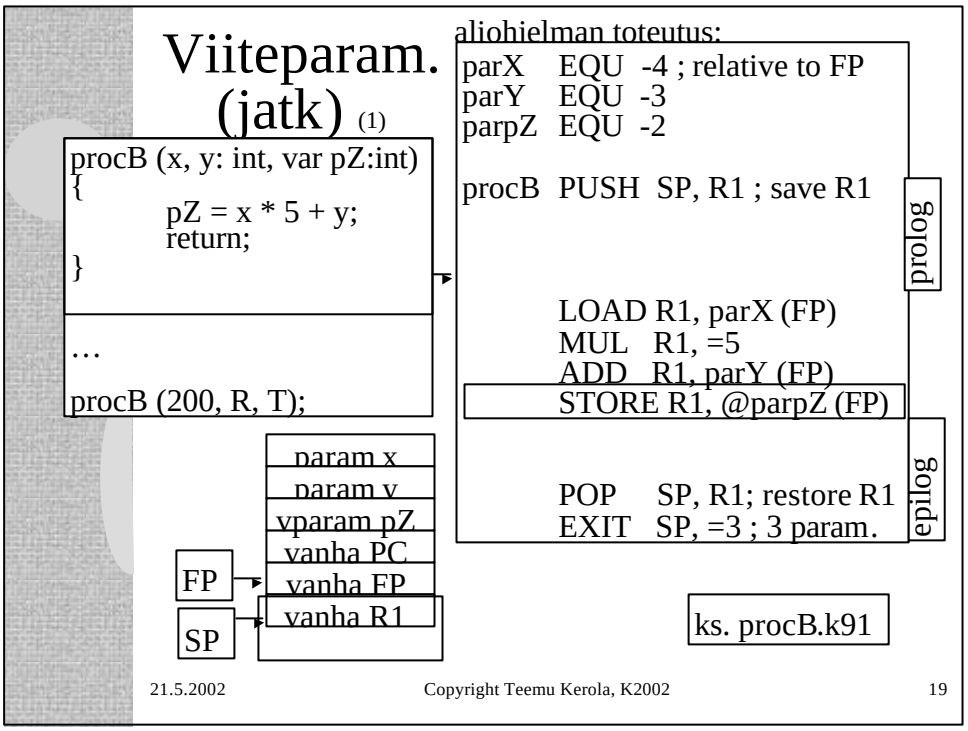
FP

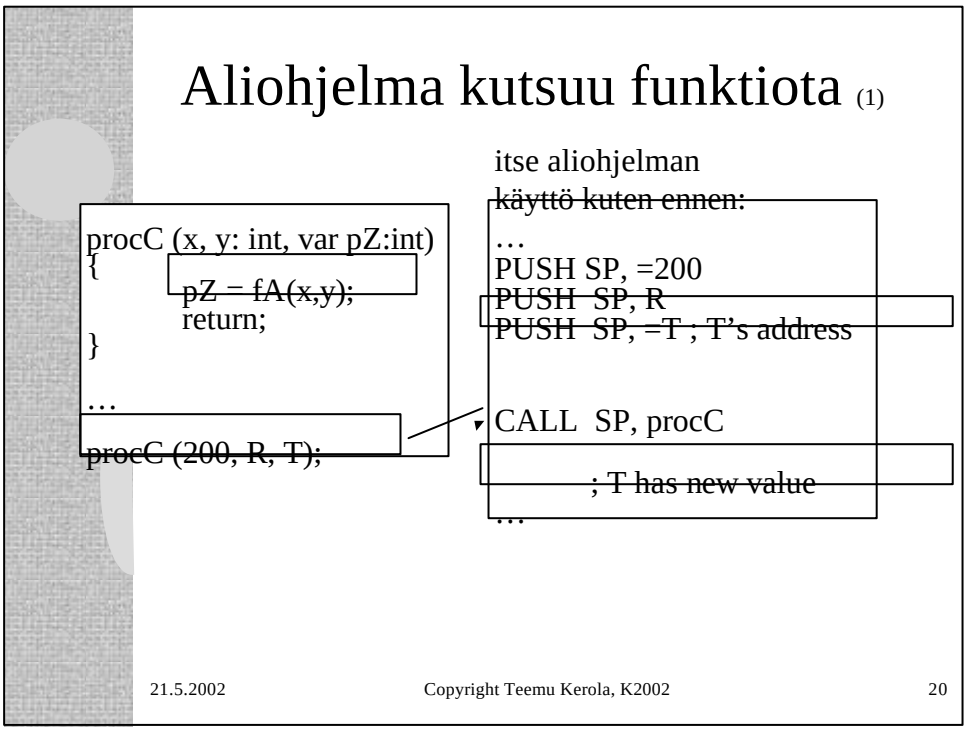

Aliohjelmaesimerkki (ei animm)
käyttö:
int fA (int x, y)
{
int z = 5;
z = x * z + y;
return (z);
}
…
T = fA (200, R);
R DC 24
…
PUSH SP,=0 ; tila paluuarvolle
PUSH SP, =200
PUSH SP, R
muistista muistiin!!
CALL SP, fA
talleta PC, FP
aseta PC,
kutsu & paluu
palauta FP, PC
POP SP, R1
STORE R1, T
2. operandi
aina rekisteri
…
tämän-
hetkinen,
nykyinen
FP
FP
paluuarvo
par x=200
par y=24
21.5.2002
Copyright Teemu Kerola, K2002
15


Aliohjelma-
esimerkki (11)
int fA (int x, y)
{
int z = 5;
z = x * z + y;
return (z);
}
…
T = fA (200, R);
retfA EQU -4 ; paluuarvo
parX EQU -3 ; 1. param. = x
parY EQU -2 ; 2. param. = y
locZ EQU 2 ; paikall. muutt
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, retfA (FP)
SUB SP, =1 ; free Z
POP SP, R1, recover R1
EXIT SP, =2 , 2 param.
prolog
epilog
Kaikki viitteet
näihin tehdään
suhteessa FP:hen
21.5.2002
Copyright Teemu Kerola, K2002
16


Aliohjelma
esimerkki (ei
aliohjelman toteutus:
int fA (int x, y)
{
int z = 5;
z = x * z + y;
return (z);
}
…
T = fA (200, R);
retfA EQU -4
parX EQU -3
parY EQU -2
locZ EQU 2
ks. fA.k91
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, retfA (FP)
SUB SP, =1 ; free Z
POP SP, R1; recover R1
EXIT SP, =2 ; 2 param.
prolog
epilog
Kaikki viitteet
näihin tehdään
suhteessa FP:hen
paluuarvo
param x
param y
vanha PC
vanha FP
paik. z
vanha R1
FP
SP
21.5.2002
Copyright Teemu Kerola, K2002
17


Viiteparametri esimerkki (2)
(Pascal)
procB (x, y: int, var pZ:int)
{
pZ = x * 5 + y;
return;
}
…
procB (200, R, T);
käyttö:
…
PUSH SP, =200
PUSH SP, R
PUSH SP, =T ; T's address!
CALL SP, procB
; T has new value
…
Ei välitetä taulukkoa T (ja sen kaikkia alkioita),
vaan ainoastaan T:n osoite (yksi arvo)
21.5.2002
Copyright Teemu Kerola, K2002
18

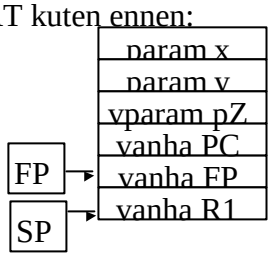

Viiteparam. (jatk) (1)
aliohjelman toteutus:
parX EQU -4 ; relative to FP
parY EQU -3
parpZ EQU -2
procB PUSH SP, R1 ; save R1
LOAD R1, parX (FP)
MUL R1, =5
ADD R1, parY (FP)
STORE R1, @parpZ (FP)
POP SP, R1; restore R1
EXIT SP, =3 ; 3 param.
prolog
epilog
procB (x, y: int, var pZ:int)
{
pZ = x * 5 + y;
return;
}
…
procB (200, R, T);
param x
param y
vparam pZ
vanha PC
vanha FP
vanha R1
FP
SP
ks. procB.k91
21.5.2002
Copyright Teemu Kerola, K2002
19


Aliohjelma kutsuu funktiota (1)
itse aliohjelman
käyttö kuten ennen:
…
PUSH SP, =200
PUSH SP, R
PUSH SP, =T ; T's address
CALL SP, procC
; T has new value
…
procC (x, y: int, var pZ:int)
{
pZ = fA(x,y);
return;
}
…
procC (200, R, T);
21.5.2002
Copyright Teemu Kerola, K2002
20

Aliohjelma kutsuu funktiota (2)

```
procC (x, y: int, var pZ:int)
{
        pZ = fA(x,y);
        return;
}
…
procC (200, R, T);
```

AT kuten ennen:



aliohjelman toteutus:

```
parXc  EQU  -4 ; relative to FP
parYc  EQU  -3
parpZ  EQU  -2

procC  PUSH  SP, R1 ; save R1
       ; call fA(parXc, parYc)
       PUSH  SP,=0 ; ret. value
       PUSH  SP, parXc(FP)
       PUSH  SP, parYc(FP)
       CALL  SP, fA
       POP   SP, R1
       STORE R1, @parpZ (FP)

       POP   SP, R1; restore R1
       EXIT  SP, =3 ; 3 param.
```

ks. procC.k91

Rekursiivinen aliohjelma (4)

- Aliohjelma, joka kutsuu itseään
- Ei mitään erikoista muuten
- Aktivointitietue hoitaa tilanvarauksen automaattisesti paikallisille muuttujille joka kutsukerralla
  - Rekursio ei onnistu, jos paikallisten muuttujien tilanvaraus on aliohjelman ohjelmakoodin yhteydessä
    - jotkut Fortran-versiot
- Joka kutsukerralla suoritetaan sama koodi-alue (aliohjelman koodi), mutta dataa varten on käytössä oma aktivointitietue

Rekursio esimerkki (1)

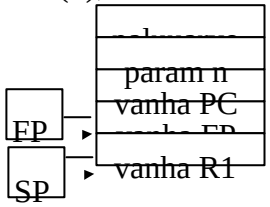
```
fPow (n: int)
{
   if (n=1)
      return (1);
  else
      return (n * fPow (n-1));
}

…

k = fPow (4);
```

kutsu:

```
K       DC   0

        ; k = fPow (4)
        PUSH  SP, =0
        PUSH  SP, =4
        CALL  SP, fPow
        POP     SP, R1
        STORE R1, K
```


Rekursion toteutus (2)

```
fPow (n: int)
{
   if (n=1)
      return (1);
  else
      return (n * fPow (n-1));
}

…

k = fPow (4);
```




```
parRet  EQU  -3
parN    EQU  -2

fPow    PUSH  SP, R1 ; save R1

        LOAD R1,  parN(FP)
        COMP R1,=1
        JEQU  One ; return 1 ?

        ; return fPow(N-1) * N
        SUB    R1, =1  ; R1 = N-1
        PUSH  SP, =0 ; ret. value space
        PUSH  SP, R1
        CALL   SP, fPow
        POP   SP, R1 ; R1 = fPow(N-1)

        MUL  R1, parN(FP)
One     STORE R1, parRet(FP)

        POP    SP, R1; restore R1
        EXIT   SP, =1 ; 1 param.
```

KJ-palvelun kutsu

- Samalla tavalla kuin aliohjelman kutsu
  - CALL-käskyn asemasta SVC
- Tilaa paluuarvolle?
- Parametrit pinoon
- SVC-kutsu
- IRET-paluu
- Paluuarvo (OK, virhe) pois pinosta tarkistusta varten

-- Jakson 4 loppu --

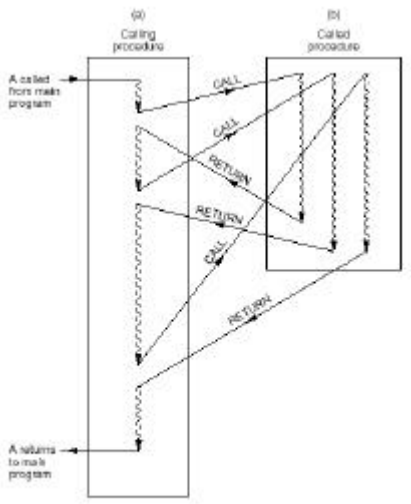


Figure 5-42. When a procedure is called, execution of the procedure always begins at the first statement of the procedure.

[Tane99]