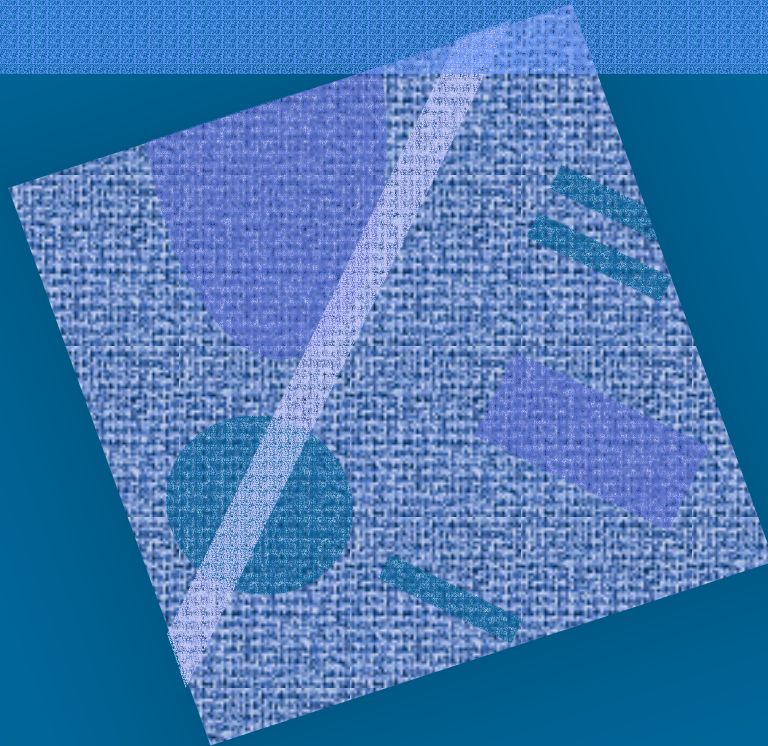


Jakso 4

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

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

- käsittely hyvin samalla tavalla kuin parametreillekin

- 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);  
x = Laske( y+234);
```

Parametryypit

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

```
Swap(i,k);
```

```
Swap(i,T[i]);
```


Arvoparametri (10)

Tulosta (A+3, B)

- Välitetään todellisen parametrin arvo
 - muuttuja, vakio, lauseke, pointeri, olioviite
- Aliohjelma ei voi muuttaa mitenkään todellisena parametrina käytettyä muuttujaa
 - muuttujan (esim. Y) arvo
 - olioviitteen arvo
 - lausekkeen arvo
 - muuta arvoparametrin 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

arvon
kopio

```
Laske (int y, *ptrX)
{
    ...
    y = 5;
    *ptrX = 10;
}
```

Viiteparametri (4)

Summaa (54, Sum)

- Välitetään todellisen parametrin osoite
– muuttujan osoite

pointteri

- 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_1km)

Nimiparametri (4)

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

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

Swap(i,j);
OK!

```
swap (n, A[n]) % n ↔ A[n]
```

```
t := n; n := A[n]; A[n] := t
```

Ei käsitellä
enää jatkossa



”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 arvojen halutaan säilyvän!
 - pääohjelma, toinen aliohjelma, sama aliohjelma, metodi, ...
 - käytettyjen rekistereiden arvot pitää aluksi tallettaa muistiin ja lopuksi palauttaa ennalleen



Aliohjelmat voivat olla sisäkkäisiä

- aliohjelmaan siirtyminen
- aliohjelmasta paluu
- aliohjelman tarvitsemat tiedot



(Ali)ohjelman ympäristö eli **aktivointitietue**

Aktivointitietue

(activation record,
activation frame)

```
int funcA (int x,y);
```

- Aliohjelman toteutusmuoto (ttk-91)

- kaikkien ulostuloparametrien arvot

- kaikkien (sisäänmeno) parametrien arvot

- paluuosoite

- kutsukohdan aktivointitietue **FP**

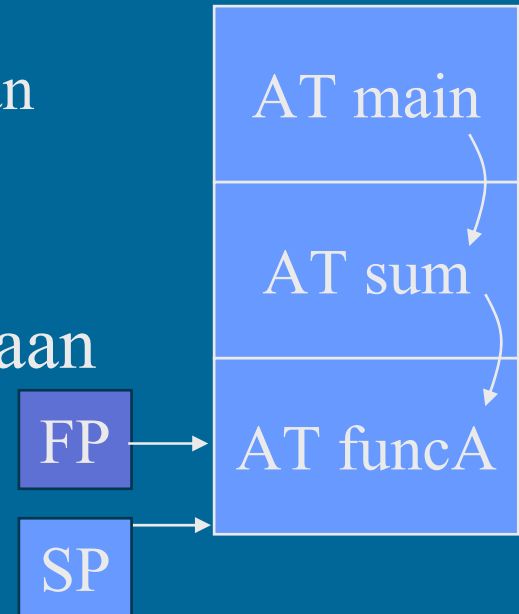
- kaikki paikalliset muuttujat ja tietorakenteet

- aliohjelman ajaksi talletettujen rekistereiden arvot



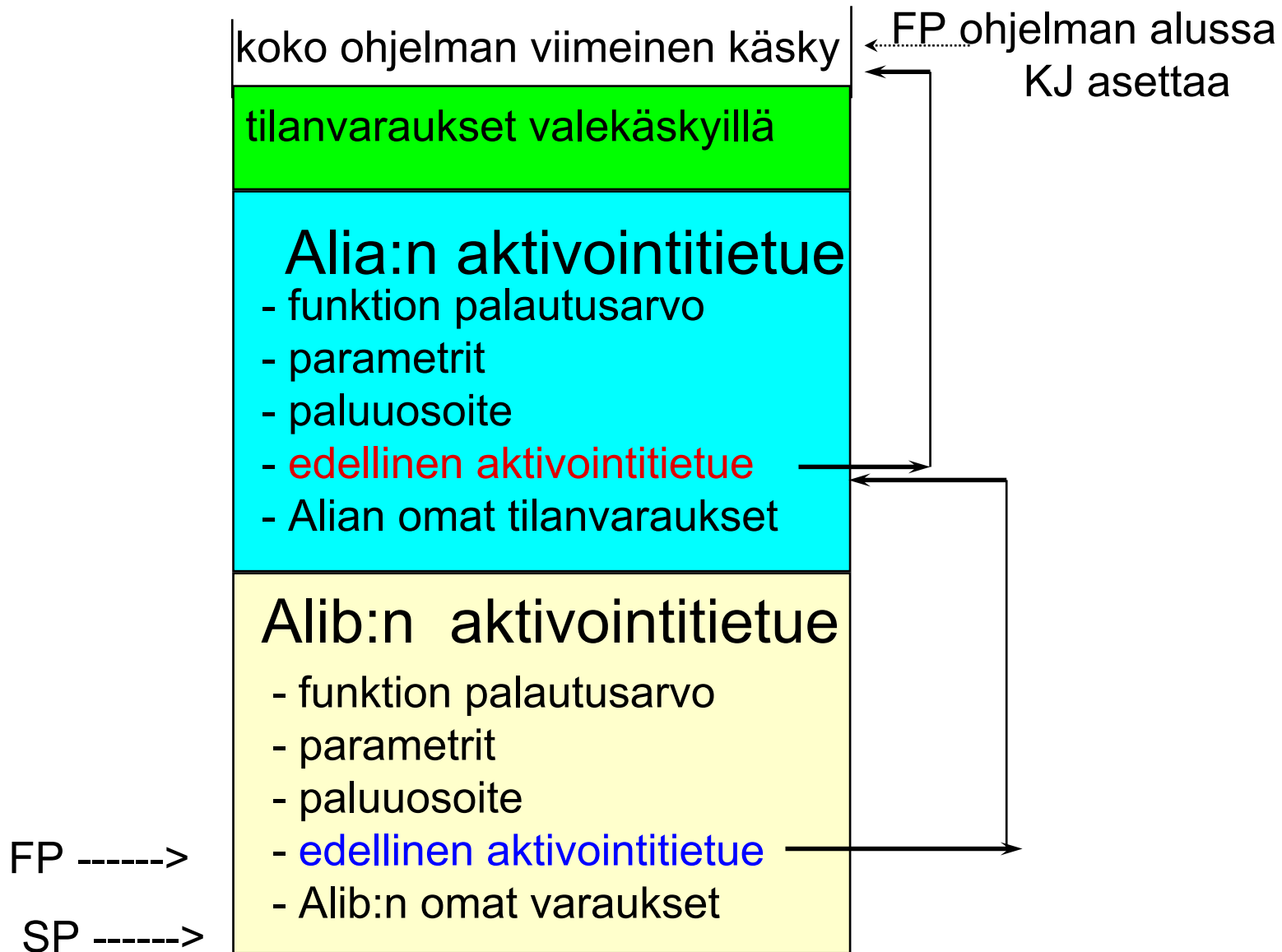
Aktivointitietueiden hallinta (4)

- 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



Talleta R0-R5 pinoon

kasvava
muistiosoite



Pinon tila ennen Alib:sta poistumista

Aliohjelman käytön toteutus (12)

- Toteutus jaettu eri yksiköille

Kutsuva
rutiini

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

CALL
käsky

- talleta vanha PC ja FP; aseta uudet PC ja FP

Kutsuttu
rutiini

- varaa tilaa paikallisille muuttujille
- talleta käytettyjen rekistereiden arvot pinoon

prolog

- (itse aliohjelman toteutus)

- palauta rekistereiden arvot

- vapauta paikallisten muuttujien tila

epilog

EXIT
käsky

- palauta PC ja FP
- vapauta parametrien tila

Kutsuva
rutiini

- ota paluuarvo pinosta

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

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

T = fA (200, R);

aliohjelman käyttö (pää)ohjelmasta:

R DC 24

...

PUSH SP,=0 ; tilaa paluuarvolle

PUSH SP, =200

PUSH SP, R

muistista
muistiin!!

CALL SP, fA

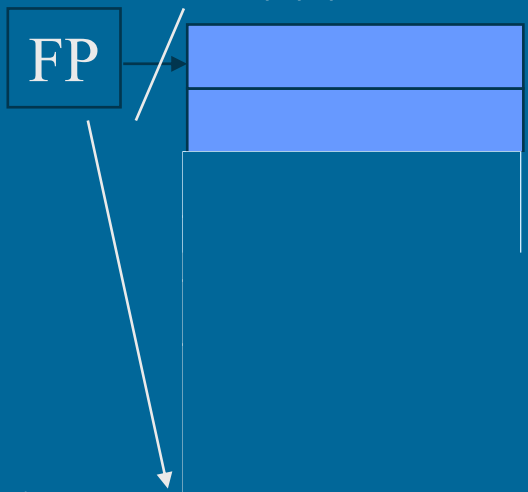
talleta PC, FP
asetta PC,
kutsu & paluu
palauta FP, PC

POP SP, R1

STORE R1, T

2. operandi
aina rekisteri

...



tämän-
hetkinen,
nykyinen
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

muistista
muistiin!!

PUSH SP, R

CALL SP, fA

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

POP SP, R1

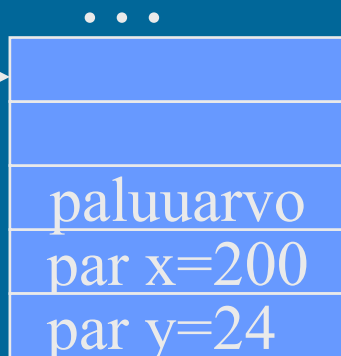
2. operandi
aina rekisteri

STORE R1, T

...

tämän-
hetkinen,
nykyinen
FP

FP



Aliohjelma- esimerkki (11)

```

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

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

...

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

aliohjelman toteutus:

```

retfA EQU -4 ; paluuarvo
parX EQU -3 ; 1. param. = x
parY EQU -2 ; 2. param. = y
locZ EQU 1 ; paikall. muutt
    
```

```

fA PUSH SP, =0 ; tila Z:lle
   PUSH SP, R1 ; R1 talteen
    
```

```

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

```

LOAD R1, parX (FP)
MUL R1, locZ (FP)
ADD R1, parY (FP)
    
```

```

STORE R1, retfA (FP)
    
```

```

POP SP, R1 ; palauta R1
SUB SP, =1; vapauta Z
EXIT SP, =2 ; 2 param.
    
```

Kaikki viitteet
päihin tehdään
suhteessa FP:hen

paluuarvo

FP

Aliohjelma- esimerkki (ei ali-ohjelman 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 1
```

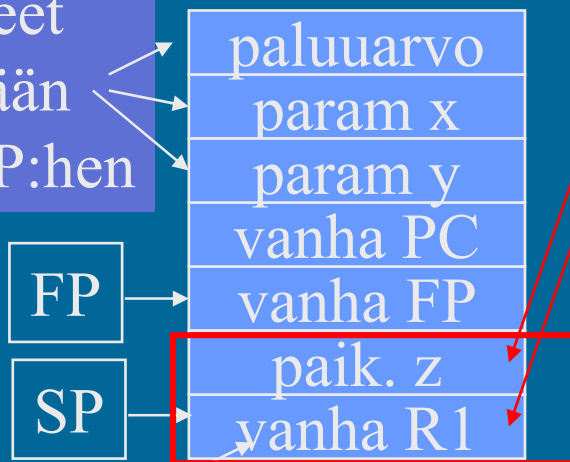
ks. fA.k91

```
fA PUSH SP, =0 ; tilaa Z:lle
    PUSH SP, R1 ; talleta R1
```

```
LOAD R1, =5; alusta 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.
```

Kaikki viitteet
päihin tehdään
suhteessa FP:hen



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 arvoa T, vaan T:n osoite.

Ainoa tapa monisanaiselle parametrille (taulukko, tietue) tai ulostuloparametreille.

Viiteparam. (jatk) ⁽¹⁾

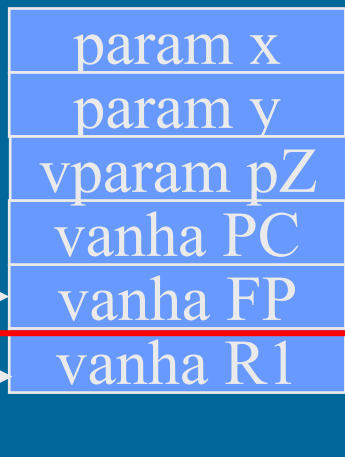
```
procB (x, y: int, var pZ:int)
{
    pZ = x * 5 + y;
    return;
}
```

...

```
procB (200, R, T);
```

FP

SP



aliohjelman toteutus:

```
parX EQU -4 ; suhteessa FP:hen
parY EQU -3
parpZ EQU -2
```

```
procB PUSH SP, R1 ; R1 talteen
```

```
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.
```

ks. procB.k91

prolog

epilog

Aliohjelma kutsuu funktiota (1)

itse aliohjelman
käyttö kuten ennen:

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

```
...
PUSH SP, =200
PUSH SP, R
PUSH SP, =T ; T's address

CALL SP, procC

; T has new value
...
```


Aliohjelma kutsuu funktiota (2)

aliohjelman toteutus:

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

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

ks. procC.k91

```
procC PUSH SP, R1 ; save R1
      ; call fA(parXc, parYc)
```

```
      PUSH SP,=0 ; ret. value
      PUSH SP, parXc(FP)
      PUSH SP, parYc(FP)
```

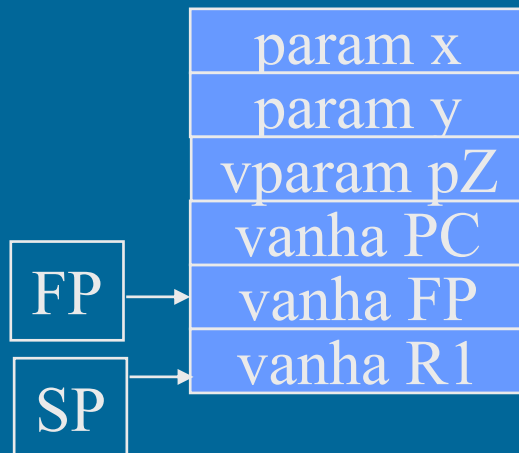
```
      CALL SP, fA
```

```
      POP SP, R1
```

```
      STORE R1, @parpZ (FP)
```

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

AT kuten ennen:



Rekursiivinen aliohjelman (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);
```

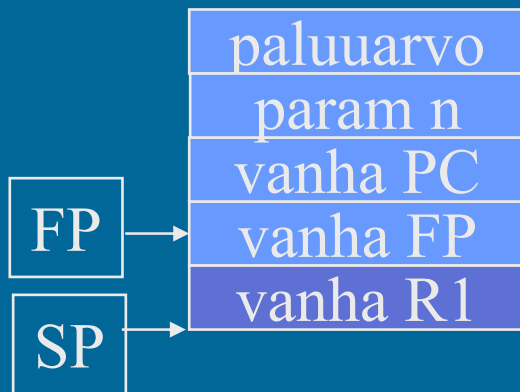
kutsu:

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

Rekursio

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 ; R1 talteen
```

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

```
; paluuarvoksi fPow(N-1) * N
SUB R1, =1 ; R1 = N-1
```

```
PUSH SP, =0 ; tilaa paluuarvol.
PUSH SP, R1
CALL SP, fPow
POP SP, R1 ; R1 = fPow(N-1)
```

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

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


pääohjelma

fPow(4):n suoritus

fPow(3):n suoritus

fPow(2):n suoritus

fPow(4):n suoritus

.....

syöttöparametriksi 4

CALL SP, fPow

paluuarvo = **24**

.....

syöttöparametriksi 3

CALL SP, fPow

paluuarvo = **6 * 4 = 24**

.....

syöttöparametriksi 2

CALL SP, fPow

paluuarvo = **2 * 3 = 6**

.....

syöttöparametriksi 1

CALL SP, fPow

paluuarvo = **1 * 2 = 2**

koska parametri =1, niin paluuarvoksi 1

Rekursion kulku

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

...

PUSH SP, =0; paluuarvo

PUSH SP, =FileBuffer

PUSH SP, CharCnt

PUSH SP, FilePtr

SVC SP, =ReadFile

POP SP, R1

JNZER R1, =FileTrouble

....

-- Jackson 4 loppu --

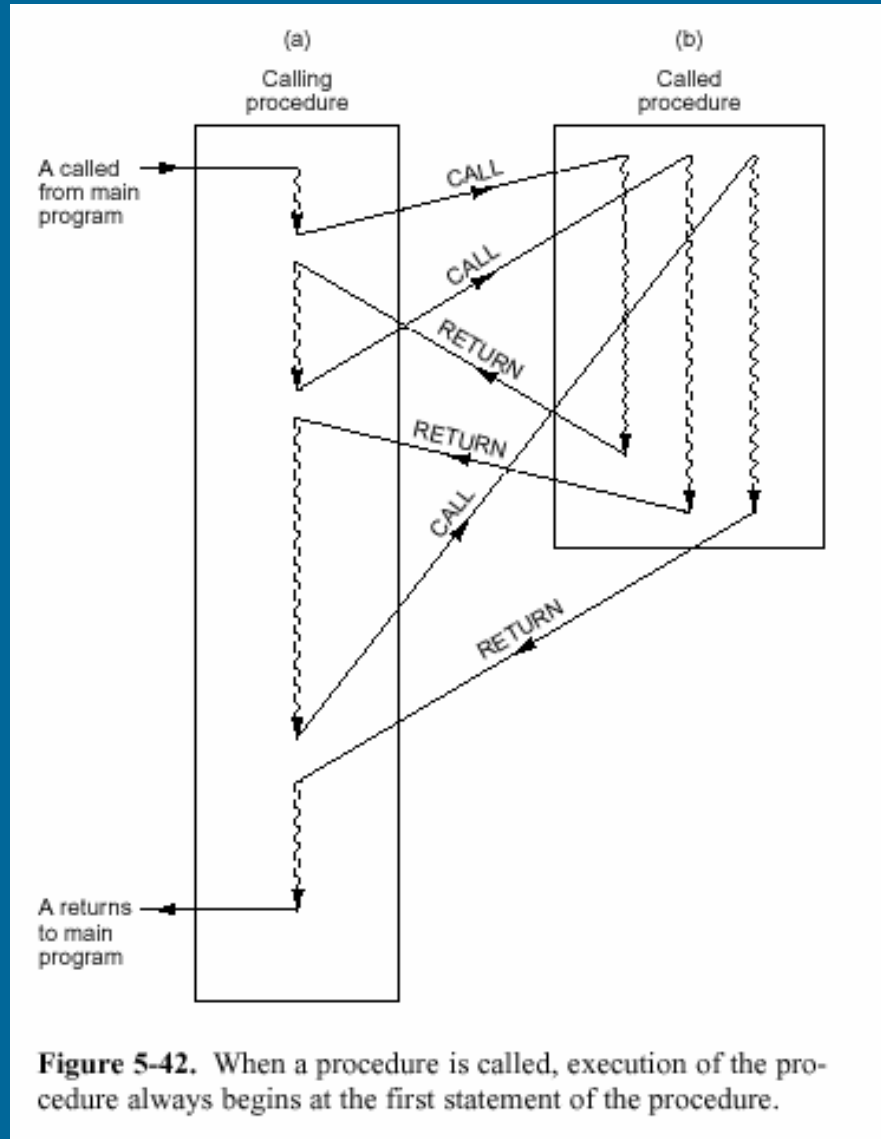


Figure 5-42. When a procedure is called, execution of the procedure always begins at the first statement of the procedure.

[Tane99]