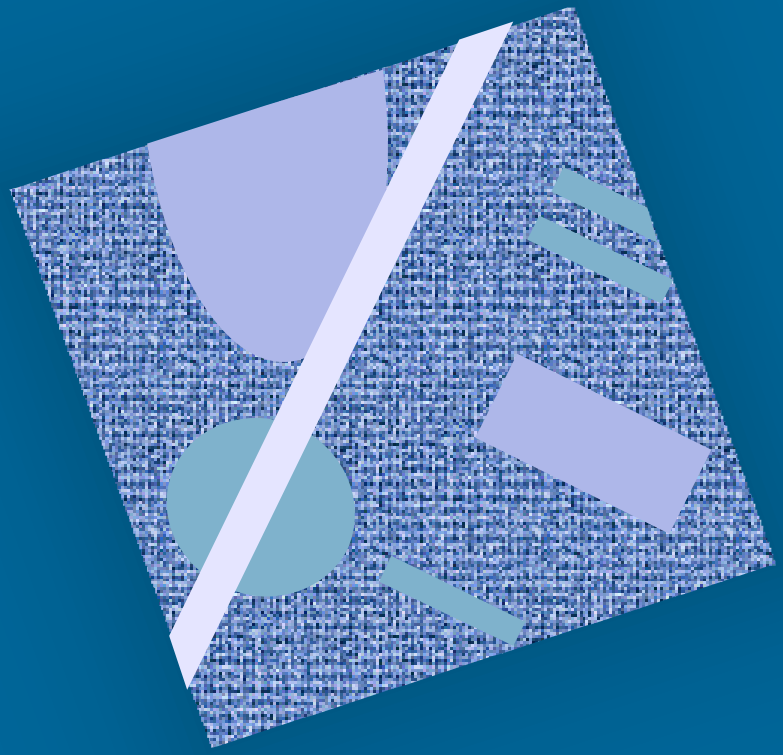
Jakso 4
Aliohjelmien toteutus



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
  - 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
  - paluuarvo saadaan paluuhetkellä ja sitä käytetään kuten mitä tahansa arvoa

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

Arvoparametri (10)

Tulosta (A+3, B)

- Välitetään todellisen parametrin arvo
  - muuttuja, vakio, lauseke, pointteri, olioviite

arvon kopio

- 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!
  - osoitinmuuttuja parametrin ptrX arvoa ei voi muuttaa
  - osoitinmuuttujan osoittamaa arvoa voidaan muuttaa

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

- Javassa ja C:ssä vain arvoparametreja

Viiteparametri (4)

Summaa (54, Sum)

- Välitetään todellisen parametrin osoite
  - muuttujan osoite

pointteri

- Aliohjelma voi muuttaa parametrina annettua muuttujan arvoa
- Pascalin *var* parametri

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

Nimiparametri (4)

- Välitetään todellisen parametrin nimi
  - merkkijono!
  - Algol 60
  - C:n 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.

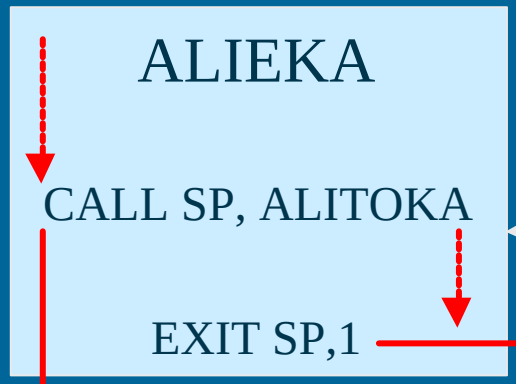

```
swap (n, A[n])   %  n ↔ A[n]
```

```
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



Aliohjelmat voivat olla sisäkkäisiä

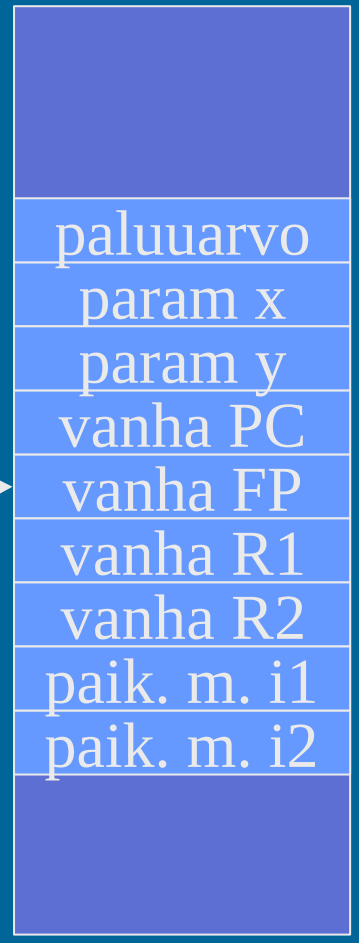
- aliohjelmaan siirtyminen
- aliohjelmasta paluu
- aliohjelman tarvitsemat tiedot

Aliohjelman ympäristö eli aktivointitietue

Aktivointitietue (6)

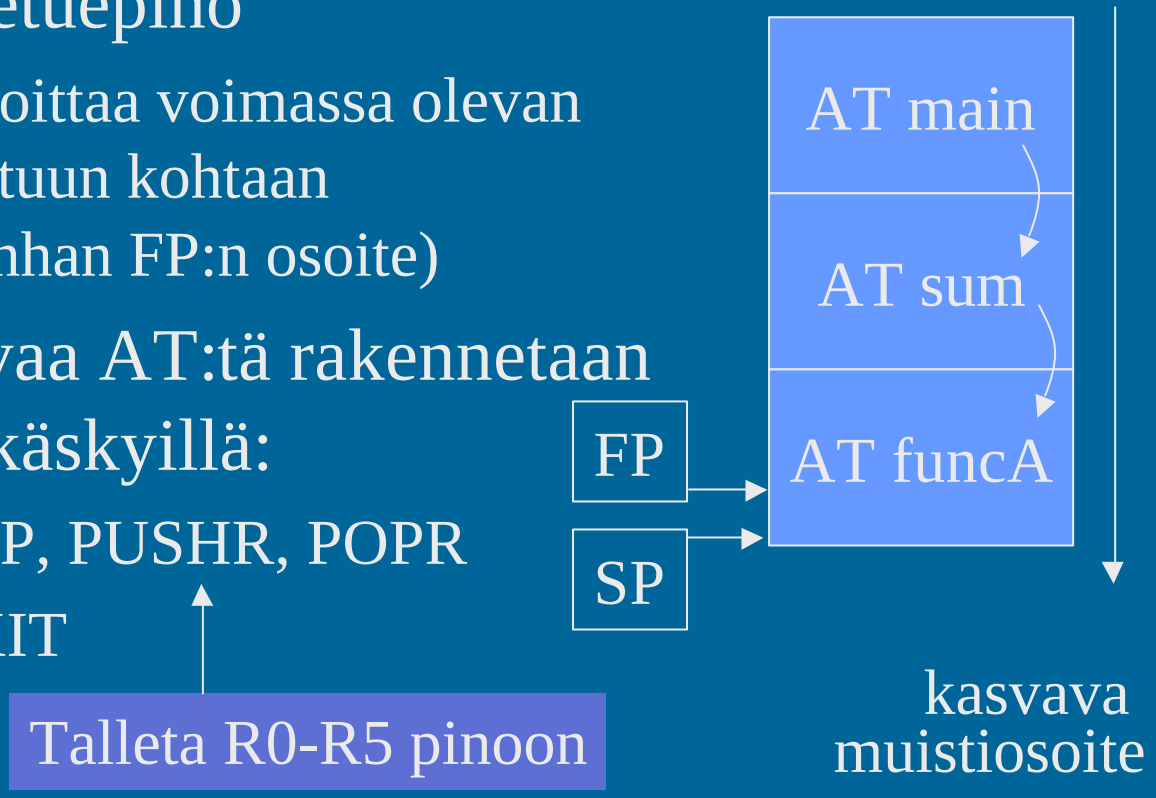
int funcA (int x,y);

- Aliohjelman toteutusmuoto (ttk-91)
  - kaikkien ulostuloparametrien arvot
  - kaikkien (sisäänmeno) parametrien arvot
  - paluuosoite
  - kutsukohdan aktivointitietue
  - aliohjelman ajaksi talletettujen rekistereiden arvot
  - kaikki paikalliset muuttujat ja tietorakenteet



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



Aliohjelman käytön toteutus (12)

- Toteutus jaettu eri yksiköille
  - varaa tilaa paluuarvolle pinosta (Kutsuva rutiini)
  - laita parametrit (arvot tai osoitteet) pinoon (Kutsuva rutiini)
  - talleta PC ja FP (CALL käsky)
  - talleta käytettyjen rekistereiden arvot pinoon (Kutsuttu rutiini, prolog)
  - varaa tilaa paikallisille muuttujille (Kutsuttu rutiini, prolog)
  - (itse aliohjelman toteutus) (Kutsuttu rutiini)
  - vapauta paikallisten muuttujien tila (Kutsuttu rutiini, epilog)
  - palauta rekistereiden arvot (Kutsuttu rutiini, epilog)
  - palauta PC ja FP (EXIT käsky)
  - vapauta parametrien tila (Kutsuva rutiini)
  - ota paluuarvo pinosta (Kutsuva rutiini)

Aliohjelmaesimerkki (13)

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

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

…

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

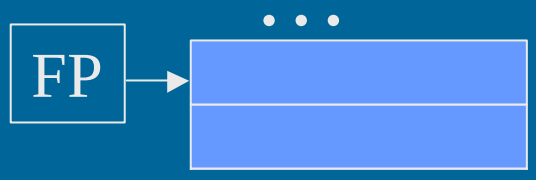
käyttö:

```
R       DC  24
…
PUSH  SP,=0 ; ret. value space
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

Aliohjelmaesimerkki (ei animm)

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

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

…

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

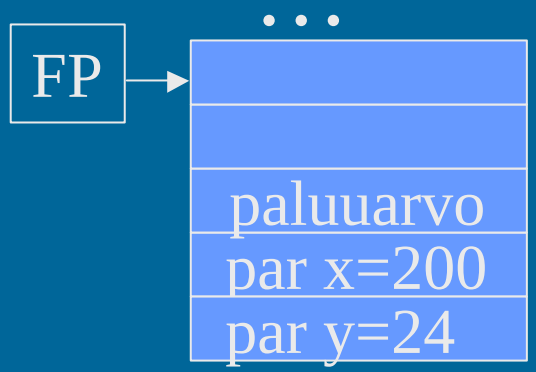
käyttö:

```
R       DC  24
…
PUSH  SP,=0 ; ret. value space
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

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
parX    EQU  -3
parY    EQU  -2
locZ    EQU  2

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

        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.
```

ks. fA.k91

prolog

epilog

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluuarvo

Aliohjelma esimerkki (ei

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

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

…

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

aliohjelman toteutus:

ks. fA.k91

```
retfA   EQU  -4
parX    EQU  -3
parY    EQU  -2
locZ    EQU  2

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

        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

Viiteparametri esimerkki (2)

(Pascal)

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

…

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

käyttö:

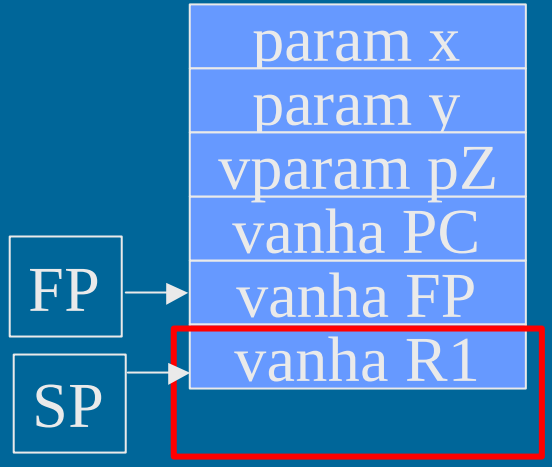
```
…
PUSH SP, =200
PUSH  SP, R
PUSH  SP, =T ; T's address!

CALL  SP, procB

        ; T has new value
…
```

Ei välitetä taulukkoa T (ja sen kaikkia alkioita), vaan ainoastaan T:n osoite (yksi arvo)

Ero C-kieleen: *pZ = x * 5 + y; ?????

Viiteparam. (jatk) (1)

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

…

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



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

ks. procB.k91

Aliohjelma kutsuu funktiota (1)

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

…

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

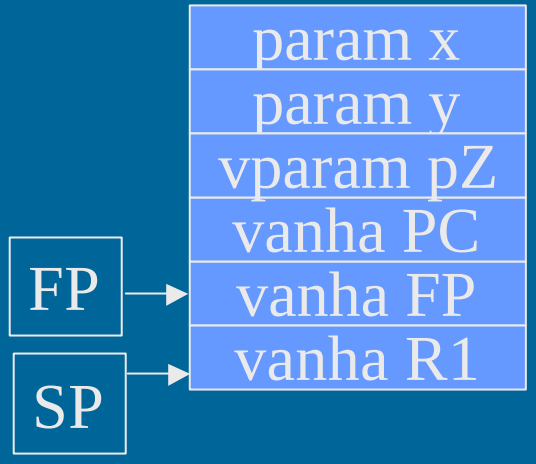
itse aliohjelman
käyttö kuten ennen:

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

CALL  SP, procC

        ; T has new value
…
```

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:

ks. procC.k91

```
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.
```

Rekursiivinen aliohjelma (4)

- Aliohjelma, joka kutsuu itseään
- Ei mitään erikoista muuten
- Aktivointitietue hoitaa tilanvarauksen automaattisesti paikallisille muuttujille joka kutsukerralla
- Joka kutsukerralla suoritetaan sama koodi-alue (aliohjelman koodi), mutta dataa varten on käytössä oma aktivointitietue

Rekursio esimerkki (1)

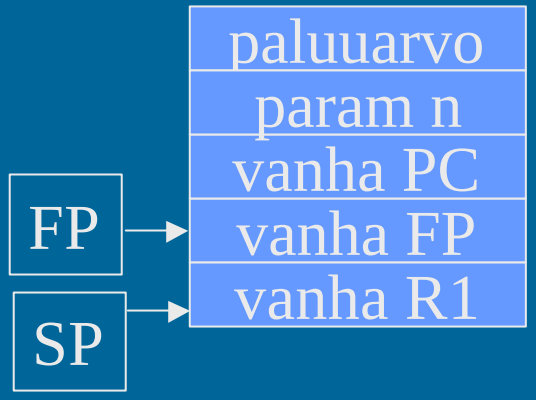
```
fPow (n: int)
{
    if (n=1)
        return (1);
    else
        return (n * fPower (n-1));
}

…

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 = fPower (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.
```

-- Jakson 4 loppu --

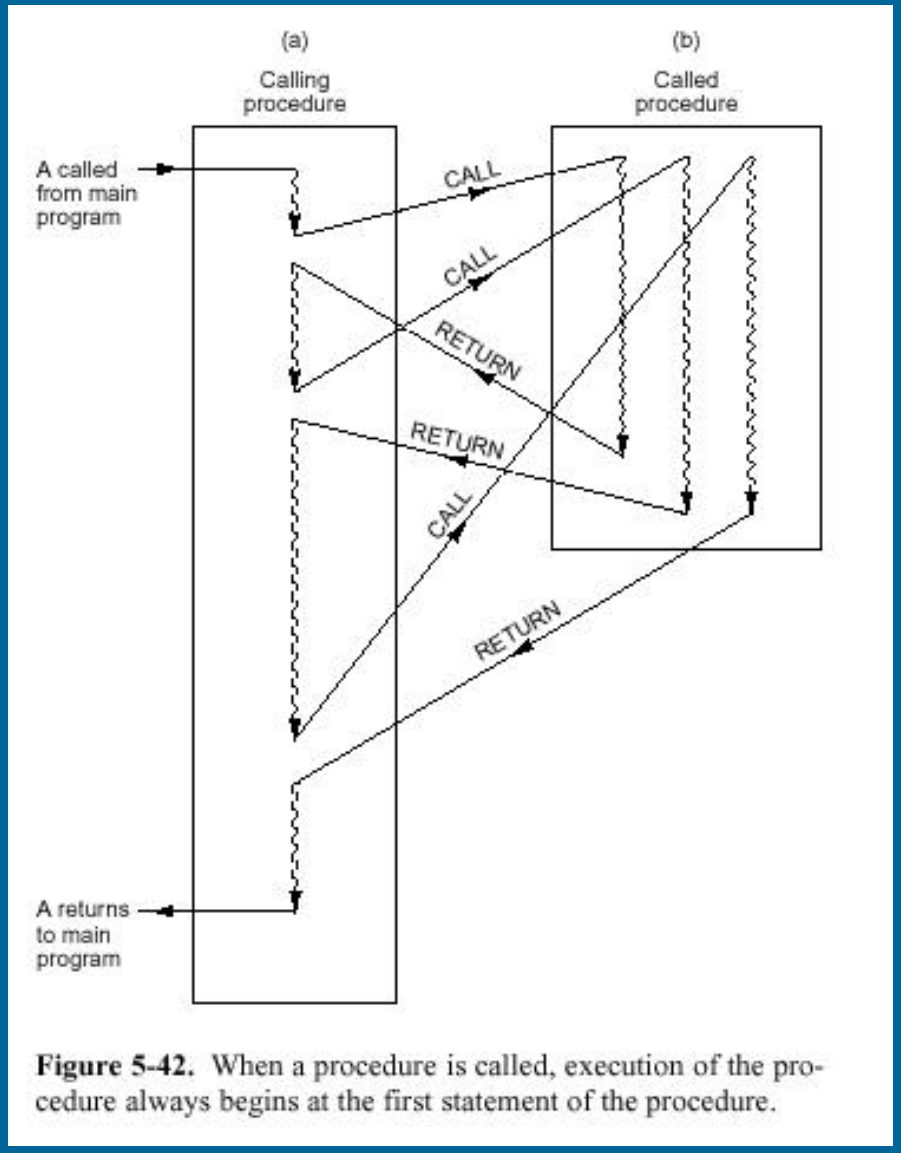


Figure 5-42. When a procedure is called, execution of the procedure always begins at the first statement of the procedure.

[Tane99]