# Luento 4
# Aliohjelmien toteutus

Tyypit
Parametrit
Aktivointitietue (AT)
AT-pino
Rekursio

 Copyright Teemu Kerola 2002 

## 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)

 Copyright Teemu Kerola 2002 

# 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
  - 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);
```

08/08/2002 Copyright Teemu Kerola 2002 3

# Parametrityypit (3)

- 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

08/08/2002 Copyright Teemu Kerola 2002 4

# 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 (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

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

- Javassa ja C:ssä vain arvoparametreja

08/08/2002 Copyright Teemu Kerola 2002 5

# Viiteparametri (5)

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;
        …
}
```

Summaa (6, Kok_lkm)

vrt. C:ssä arvoparametrina välitetyn osoitinmuuttujan osoittaman arvon (PtrX, ed. kalvo) muuttaminen

08/08/2002 Copyright Teemu Kerola 2002 6

# Nimiparametri (6)

- 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)
t := i, i :=j; j := t;
```

Ei käsitellä enää jatkossa.

entä: swap (n, A[n]) % n ↔ A[n]

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

”väärä” n



# Aliohjelmien toteutuksen osat (5)

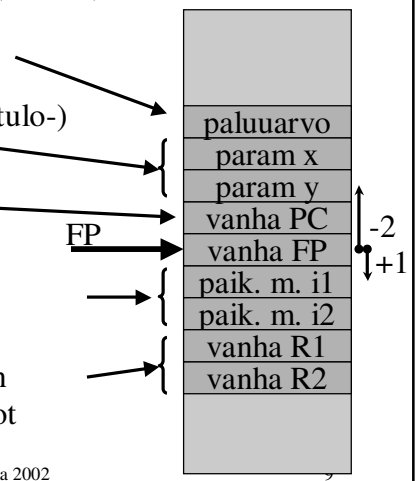
- Paluuosoite
  - kutsukohtaa seuraava käskyn osoite
- Parametrien välitys
- Paluuarvon välitys
- Paikalliset muuttujat
- Rekistereiden allokointi (varaus)
  - kutsuvalla ohjelman osalla voi olla käytössä rekistereitä, joiden arvon halutaan säilyä!
    - pääohjelma, toinen aliohjelma, sama aliohjelma,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

# Aktivointitietue (7)

(activation record, activation frame)

int funcA (int x,y);

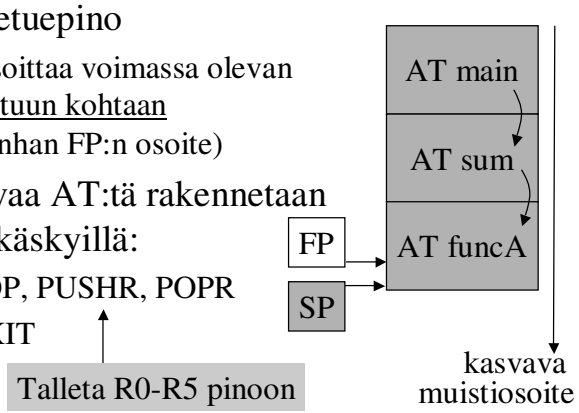
- Aliohjelman toteutusmuoto (ttk-91)
  - funktion paluuarvo (tai kaikki paluuarvot)
  - kaikkien (sisäänmeno- ja ulostulo-) parametrien arvot
  - paluuosoite
  - kutsukohdan aktivointitietue
  - kaikki paikalliset muuttujat ja tietorakenteet
  - aliohjelman ajaksi talletettujen rekistereiden alkuperäiset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






# Aktivointitietueiden hallinta (4)

- Aktivointitietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta (muistista)
  - 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

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 käytettävien rekistereiden vanhat arvot pinoon

prolog

- (itse aliohjelman toteutus)

Kutsuttu rutiini
- palauta rekistereiden arvot
- vapauta paikallisten muuttujien tila

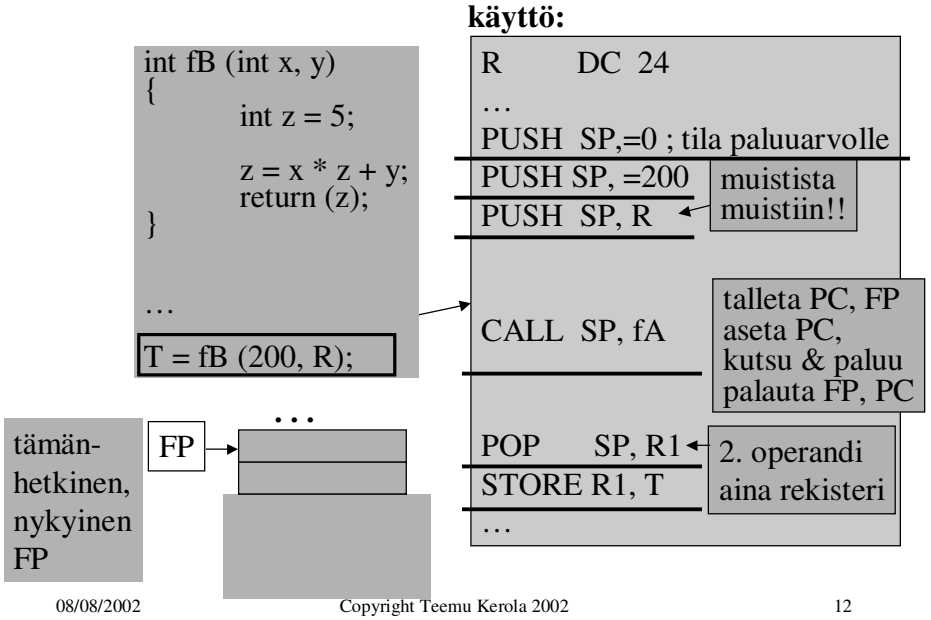
EXIT käsky
- palauta PC ja FP
- vapauta parametrien tila

epilog

Kutsuva rutiini
- ota paluuarvo pinosta

08/08/2002 Copyright Teemu Kerola 2002 11

# Aliohjelmaesimerkki (13)




08/08/2002 Copyright Teemu Kerola 2002 12

# Aliohjelmaesimerkki (ei anim)

**käyttö:**

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

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

…

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

```
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

FP

…

paluuarvo
par x=200
par y=24



# 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  # params
parX    EQU  -3
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.
```

ks. fA.k91

prolog

epilog

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluuarvo

# Aliohjelma-esimerkki (ei anim)

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

        z = x * z + y;
        return (z);
}
…
T = fA (200, R);
```

aliohjelman toteutus:

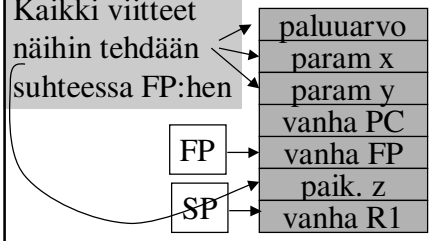
ks. fA.k91

```
retfA   EQU  -4
parX    EQU  -3
parY    EQU  -2
locZ    EQU  1

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






# Viiteparametri esimerkki (2)

(Pascal)

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

…
procB (200, R, T);
```

käyttö:

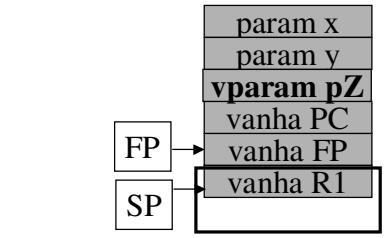
```
…
PUSH SP, =200
PUSH  SP, R
PUSH  SP, =T ; T's address!

CALL  SP, procB

        ; T has new value
…
```

Ei välitetä arvoa T, vaan T:n osoite.
Ainoa tapa monisanaiselle parametrille (taulukko, tietue) tai ulostuloparametreille

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);
```

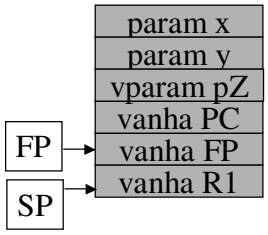
itse aliohjelman käyttö kuten ennen:

```
…
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:

```
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 (5)

- Aliohjelma, joka kutsuu itseään
- Ei mitään erikoista muuten
- Aktivointitietue hoitaa tilanvarauksen automaattisesti paikallisille muuttujille joka kutsukerralla
- Rekursio ei onnistu, jos paikallisten muuttujien tilanvaraus 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 * fPower (n-1));
}

…

k = fPow (4);
```

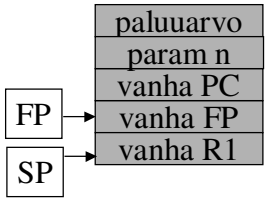
kutsu:

```
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);
```




ks. fPow.k91

```
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 (7)

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

fOK = ReadBlock (fp, 64)

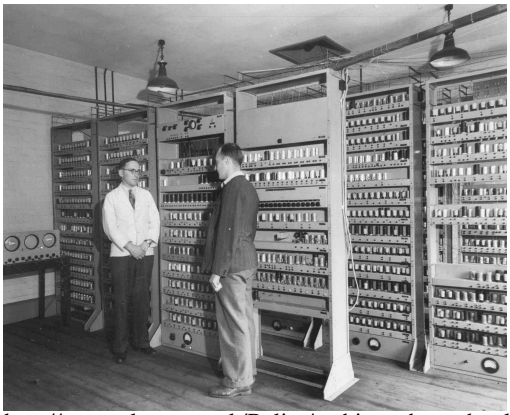
```
...
PUSH  SP, =0 ;paluuarvo
PUSH  SP, =FileBuffer
PUSH  SP, CharCnt
PUSH  SP, FilePtr

SVC   SP, =ReadFile

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

08/08/2002 Copyright Teemu Kerola 2002 23

# -- Luennon 4 loppu --

M.Wilkes:
EDSAC I (1949)

- rekisterit (6W), tyhjiöputkilla
- käsky- ja datamuisti, 32 elohopea -viiveputkea, 512W à 36b
- kertolasku 5.4ms, 650 IPS



http://www.cl.cam.ac.uk/Relics/archive_photos.html

- ensimmäinen ”stored program” –tietokone
- 3000 tyhjiöputkea, sähkökulutus 12 kW, tila 5x4m

08/08/2002 Copyright Teemu Kerola 2002 24