

Luento 4 (verkkoluento 4)

Aliohjelmien toteutus



Tyypit, Parametrit
Aktivaatitietue (AT)
AT-pino, rekursio

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)

Parametrit ja paluuarvo

- Muodolliset parametrit
 - määritelty aliohjelmassa ohjelmointihetkellä
 - tietty järjestys ja tyyppi
 - paluuarvot

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

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

- 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+1);
x = Laske( y+234);
```

Parametryypit

- 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
- Nimiparametri (*yleensä vain tulkittavissa skriptikielissä*)
 - Välitetään todellisen parametrin nimi (merkkijono)
 - Nimi (merkkijono) kuvataan arvoksi kutsuhetkellä
 - Semantiikka määräytyy vasta kutsuhetkellä

swap(i, j) -- makro, nimiparametrit

```
tmp = i;  
i = j;  
j = tmp;
```

```
swap(x, y);
```

```
swap(k, T[k]);
```

```
tmp = x;  
x = y;  
y = tmp;
```

```
tmp = k;  
k = T[k];  
T[k] = tmp;
```

Aliohjelmien toteutuksen osat

- Paluusoite
 - kutsukohtaa seuraavan käskyn osoite
- Parametrien välitys
- Paluarvon 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

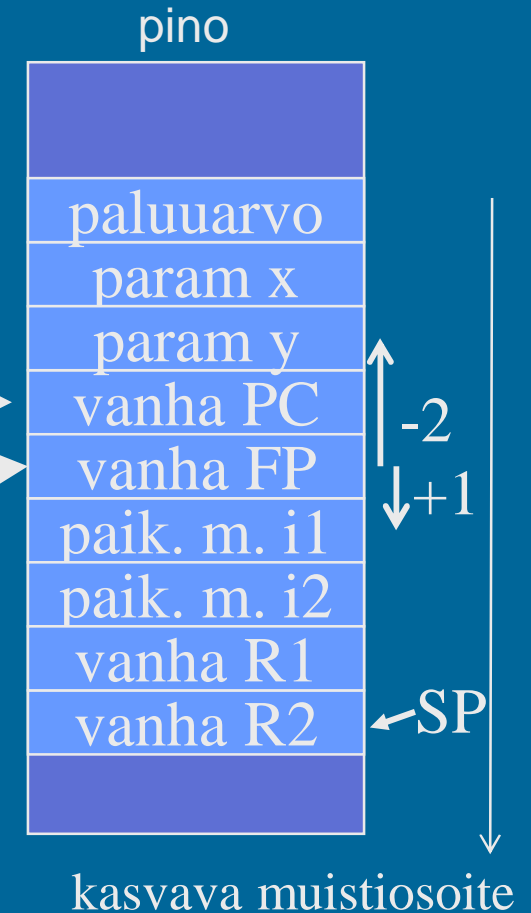
Aktivaatietue (Aktivoitietue)

(activation record,
activation frame)

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

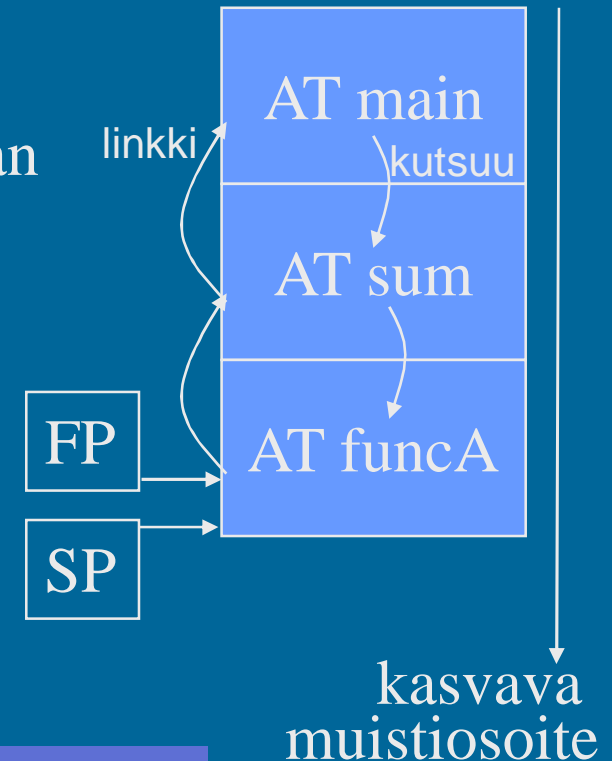
- Aliohjelman toteutus (ttk-91)

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



Aktivaatietuepino muistissa

- Aktivaatietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta (muistista)
 - SP (eli R6) osoittaa pinon pinnalle
- Aktivaatietuepino
 - 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 (SVC, IRET)



Talleta arvo pinoon

Talleta R0-R5 pinoon

Aliohjelmakutsun toteutus

- Toteutus jaettu eri yksiköille

Kutsuva
rutiini

CALL
käsky

Kutsuttu
rutiini

EXIT
käsky

Kutsuva
rutiini

- 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

prolog

epilog

Aliohjelmaesimerkki (13)

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
CALL SP, fA
POP SP, R1
STORE R1, T
...
```

muistista
muistiin!!

talleta PC, FP
asetta PC,
kutsu & paluu
palauta FP, PC
vapauta pari tila

2. operandi
aina rekisteri

tämän-
hetkinen,
nykyinen
FP



Aliohjelmaesimerkki

(ei animointia)

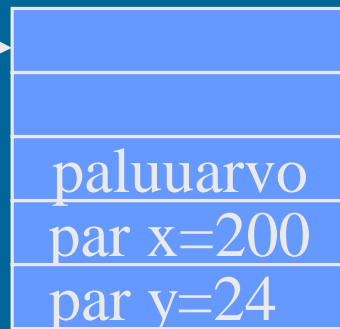
```
int fA (int x, y)
{
    int z = 5;

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

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

tämän-
hetkinen,
nykyinen
FP

FP



käyttö:

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

Aliohjelma- esimerkki ⁽¹²⁾

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

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

...

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

Kaikki viitteet
näihin tehdään
suhteessa FP:hen

paluarvo

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

Aliohjelma- esimerkki

(ei animointia)

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

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

aliohjelman toteutus:

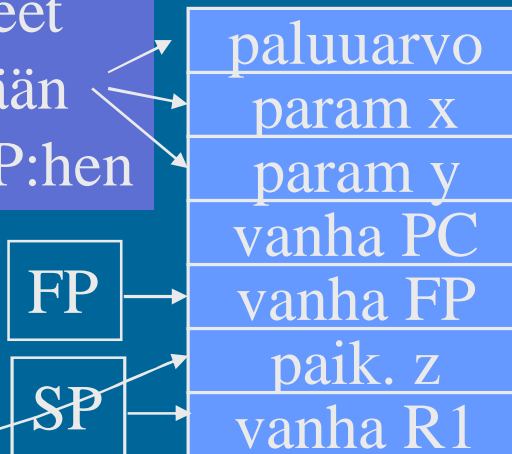
```
retfA EQU -4
parX EQU -3
parY EQU -2
locZ EQU 1
```

```
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.
```

Kaikki viitteet
näihin tehdään
suhteessa FP:hen



prolog

epilog

Viiteparametri

Kutsu $T = fB(R, \underline{S}, \underline{T})$

```
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)

- (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  
...
```

Aliohjelmakutsu

- Toteutus jaettu eri yksiköille

Kutsuva
rutiini

CALL
käsky

Kutsuttu
rutiini

EXIT
käsky

Kutsuva
rutiini

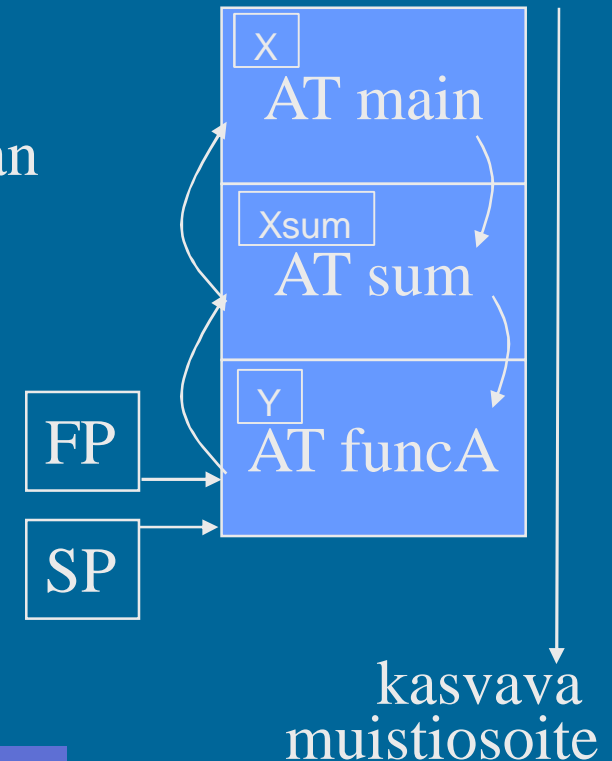
- varaa tilaa paluuarvolle pinosta
- laita parametrit (arvot tai osoitteet) pinoon
- talleta vanha PC ja FP, aseta uudet PC ja FP
- varaa tilaa paikallisille muuttujille
- talleta käytettävien rekistereiden vanhat arvot pinoon
- (itse aliohjelman toteutus – varsinainen työ)
- palauta rekistereiden arvot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

prolog

epilog

Aktivaatietuepino muistissa

- Aktivaatietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta (muistista)
 - SP (=R6) osoittaa pinon pinnalle
- Aktivaatietuepino
 - 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 (SVC, IRET)

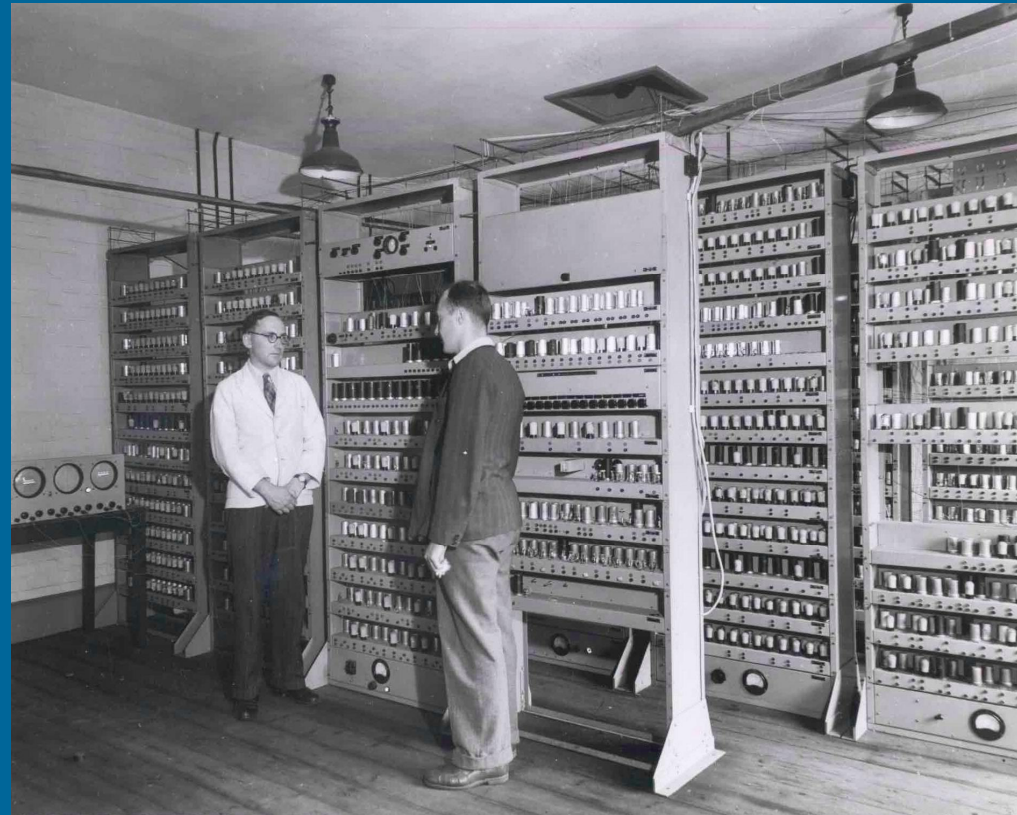


Talleta R0-R5 pinoon

-- Loppu --

M. Wilkes: EDSAC (1949)

- rekisterit (6 kpl), tyhjiöputkilla
- käsky- ja datamuisti, 32 elohopea-viiveputkea, kukin 32 kpl 18b sanaa (n. 2 Kb)
- kertolasku 5.4ms, 650 IPS
- ensimmäinen ”stored program” –tietokone
- 3000 tyhjiöputkea, sähkökulutus 12 kW, tila 5x4m
 - Tyhjiöputket kesti pidempään, jos niitä ei sammutettu koskaan



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