

Luento 3

Konekielinen ohjelmointi (TTK-91, KOKSI)

Muuttujat
Tietorakenteet
Kontrolli
Optimointi
Tarkistukset

24/10/2001 Copyright Teemu Kerola 2001 1

Tiedon sijainti suoritusaikana ⁽³⁾

- Muistissa (=keskusmuistissa)
 - iso Esim. 256 MB, tai 64 milj. 32 bitin sanaa
 - hidas Esim. 10 ns
- Rekisterissä
 - pieni Esim. 256 B, tai 64 kpl 32 bitin sanaa
 - nopea Esim. 1 ns TTK-91: 8 kpl + PC + ...
- Probleemi: milloin muuttujan X arvo pidetään muistissa ja milloin rekisterissä?
 - missä päin muistia? miten siihen viitataan?

24/10/2001 Copyright Teemu Kerola 2001 2

Tieto ja sen osoite ⁽³⁾

```

X DC 12
....
LOAD R1, =X
LOAD R2, X
            
```

muuttujan X osoite on symbolin X arvo

symbolin X arvo

muuttujan X arvo

muisti

230
12345
12556
128765
12222
12
12998

```

int x = 12;
            
```

X=230:

- Muuttujan X osoite on 230
- Muuttujan X arvo on 12
- Symbolin X arvo on 230
 - symbolit ovat yleensä olemassa vain käännoaikana!
 - Virheilmoituksia varten symbolitaulua pidetään joskus yllä myös suoritusaikana

24/10/2001 Copyright Teemu Kerola 2001 3

Tieto ja sen osoite ⁽⁵⁾

```

Xptr DC 0
X DC 12
LOAD R1, =X
STORE R1, Xptr
LOAD R2, X
LOAD R3, @Xptr
            
```

muisti

230
12345
12556
128765
12222
12
12998

```

; R1 ← 230
; R2 ← 12
; R3 ← 12
            
```

Xptr=225:

X=230:

- Muuttujan X osoite on 230
- Muuttujan X arvo on 12
- Osoitinmuuttujan (pointterin) Xptr osoite on 225
- Osoitinmuuttujan Xptr arvo on 230
- Osoitinmuuttujan Xptr osoittaman kokonaisluvun arvo on 12

24/10/2001 Copyright Teemu Kerola 2001 4

Osoitinmuuttujat ⁽⁵⁾

- Muuttujia samalla tavoin kuin kokonaislukuarvoiset muuttujatkin
- Arvo on jonkun tiedon osoite muistissa
 - globaalin yksi- tai monisanaisen tiedon osoite
 - muuttuja, taulukko, tietue, olio
 - keosta (heap, joskus ”kasa”) dynaamisesti (suoritusaikana) varatun tiedon osoite
 - Pascalin tai Javan ”new” operaatio palauttaa varatun muistialueen osoitteen (tai virhekoodin, jos operaatiota ei voi toteuttaa)
 - aliohjelman tai metodin osoite
 - osoite ohjelmakoodiin

24/10/2001 Copyright Teemu Kerola 2001 5

Globaali, kaikkialla näkyvä data ⁽²⁾

- Globaalit muuttujat ja muut globaalit tietorakenteet sijaitsevat ttk-91 koneen muistissa ohjelmakoodin jälkeen
 - muuttujat


```

int X = 25;
short Y;
float Ft;
                    
```
 - tilan varaus


```

X DC 25 ; alkuarvo = 25
Y DC 0
fBig DC 1 ; 1=true, 0=false
                    
```
 - viittaaminen

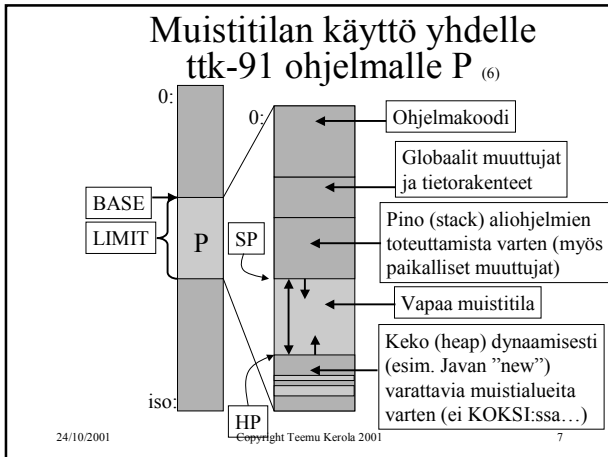

```

LOAD R1, X
STORE R2, Y
                    
```

```

char Ch;
char Str[] = ”Pekka”;
boolean fBig;
            
```

24/10/2001 Copyright Teemu Kerola 2001 6



Muistissa oleva data (3)

- **Globaali data** `int X; function Print();`
 - varataan ohjelman latauksen yhteydessä
 - kaikkialla viitattavissa nimen (osoitteen) avulla
- **Dynaaminen data** `Mach m = new Mach();`
 - varataan tarvittaessa keosta suorituksen aikana
 - vapautetaan kun ei enää tarvita (ei Koksissa)
 - viittaus varauksen jälkeen osoitteen avulla
- **Aliohjelmien paikallinen data** `parametrit, paik. muuttuja`
 - varataan pinosta kutsuhetkellä
 - vapautetaan rutiinista paluun yhteydessä
 - viittaus aliohjelman sisällä osoitteen avulla

24/10/2001 Copyright Teemu Kerola 2001 8

Tiedon sijainti suoritusaikana (4)

- **Rekisteri**
 - nopein, kääntäjä varaa/vapauttaa
- **Välimuisti**
 - nopea, laitteisto hoitaa automaattisesti
- **Muisti**
 - ohjelma varaa/vapauttaa
 - aliohjelmien paik. muuttujat, parametrit
 - käyttäjärj. varaa/vapauttaa (pyydettyä?)
 - globaali data ohjelman latauksen yhteydessä
 - dynaaminen data keosta suorituksen aikana
- **Levy, levypalvelin (verkon takana)**
 - liian hidasta, ei voi käyttää

24/10/2001 Copyright Teemu Kerola 2001 9

Ohjelmoinnin peruskäsitteet (4)

- **Aritmeettinen lauseke**
 - miten tehdä laskutoimitukset?
- **Yksinkertaiset tietorakenteet**
 - yksiulotteiset taulukot, tietueet
- **Kontrolli** - mistä seuraava käsky?
 - valinta: if-then-else, case
 - toisto: for-silmukka, while-silmukka
 - aliohjelmat, virhetilanteet
- **Monimutkaiset tietorakenteet**
 - listat, moniulotteiset taulukot

24/10/2001 Copyright Teemu Kerola 2001 10

Aritmeettinen lauseke (3)

tilan varaus

A	DC	0
B	DC	0
C	DC	0

```

int a, b, c;
...
b = 34;
a = b + 5 * c;
    
```

koodi

```

LOAD R1, =34
STORE R1, B
...
LOAD R1, B
LOAD R2, C
MUL R2, =5
ADD R1, R2
STORE R1, A
    
```

tai:

```

LOAD R1, =5
MUL R1, C
ADD R1, B
STORE R1, A
    
```

24/10/2001 Copyright Teemu Kerola 2001 11

Globaalin taulukon tilan varaus ja käyttö (3)

```

int X, Y;
int Taulu[30];
...
X = 5;
Y = Taulu[X];
    
```

X	DC	0
Y	DC	0
Taulu	DS	30

```

...
LOAD R1, =5
STORE R1, X
LOAD R1, X
LOAD R2, Taulu(R1)
STORE R2, Y
    
```

Taulu:

Optimoiva kääntäjä osaisi jättää pois jälkimmäisen "LOAD R1,X" käskyn

24/10/2001 Copyright Teemu Kerola 2001 12

Globaalien tietueiden tilan varaus ja käyttö (3)

```
int X;
struct Tauno {
    int Pituus;
    int Paino;
}
...
X = Tauno.Paino
```

Kentän "Paino" suhteellinen osoite tietueen Tauno sisällä

X	DC	0
Tauno	DS	2
Pituus	EQU	0
Paino	EQU	1
...		

Tietueen osoite on sen ensimmäisen sanan osoite

Tauno:

pituus
paino

```
LOAD R1, =Tauno
LOAD R2, Paino(R1)
STORE R2, X
```

24/10/2001 Copyright Teemu Kerola 2001 13

Kontrolli - valinta konekielellä (3)

- Ehdoton hyppy
 - JUMP, CALL ja EXIT, SVC ja IRET
- Hyppy perustuen laiterekisterin arvoon (vrt. 0)
 - JZER, JPOS, ...
- Hyppy perustuen aikaisemmin asetetun tilarekisterin arvoon
 - COMP
 - JEQU, JGRE, ...

```
COMP R2, LIMIT
JEQU LOOP
```

Ongelma vai etu: ttk-91:ssä kaikki ALU käskyt asettavat tilarekisterin

- ADD, SUB, MUL, DIV, NOT, AND, OR, XOR, SHL, SHR

24/10/2001 Copyright Teemu Kerola 2001 14

If-then-else -valinta (2)

```
if (a<b)
    x = 5;
else
    x = y;
```

```
LOAD R1, A
COMP R1, B
JNLES Else
LOAD R1, =5
STORE R1, X
JUMP Done
Else LOAD R1, Y
STORE R1, X
Done NOP
```

```
LOAD R2, Y
LOAD R1, A
COMP R1, B
JNLES Else
LOAD R2, =5
ELSE STORE R2, X
```

vai olisiko tämä parempi:

24/10/2001 Copyright Teemu Kerola 2001 15

Case lauseke (2)

```
Switch (lkm) {
    case 4: x = 11;
            break;

    case 0: break;

    default: x = 0;
            break;
}
```

Onko case-tapausten järjestyksellä väliä?

Swi	LOAD R1, Lkm
Vrt4	COMP R1, =4 JNEQ Vrt0 LOAD R2, =11 STORE R2, X JUMP Cont
Vrt0	COMP R1, =0 JNEQ Def JUMP Cont
Def	LOAD R2, =0 STORE R2, X
Cont	NOP

24/10/2001 Copyright Teemu Kerola 2001 16

Toistolausekkeet (2)

- For-step-until -silmukka
- Do-until -silmukka
- Do-while -silmukka
- While-do -silmukka
- ...

ehto silmukan alussa

ehto silmukan lopussa

24/10/2001 Copyright Teemu Kerola 2001 17

For lauseke (3)

```
for (int i=20; i < 50; ++i)
    T[i] = 0;
```

Olisiko parempi pitää i:n arvo rekisterissä? Miksi? Milloin?

Mikä on i:n arvo lopussa? Onko sitä olemassa?

I	DC	0
...		
		LOAD R1, =20 STORE R1, I
Loop		LOAD R2, =0 LOAD R1, I STORE R2, T(R1)
		LOAD R1, I ADD R1, =1 STORE R1, I
		LOAD R3, I COMP R3, =50 JLES Loop

24/10/2001 Copyright Teemu Kerola 2001 18

While-do -lauseke (2)

```

X = 14325;
Xlog = 1;
Y = 10;
while (Y < X) {
    Xlog++;
    Y = 10*Y;
}
    
```

```

LOAD R1, =14325
STORE R1, X
LOAD R1, =1 ; R1=Xlog
LOAD R2, =10 ; R2=Y
While COMP R2, X
      JNLES Done
      ADD R1, =1
      MUL R2, =10
      JUMP While
Done STORE R1, Xlog ; talleta tulos
     STORE R2, Y
    
```

Mitä kannattaa pitää muistissa?

Mitä kannattaa pitää rekisterissä ja milloin?

24/10/2001 Copyright Teemu Kerola 2001 19

Koodin generointi (9)

- Kääntäjän viimeinen vaihe
 - voi olla 50% käännösajasta
- Tavallisen koodin generointi
 - alustukset, lausekkeet, kontrollirakenteet
- Optimoidun koodin generointi
 - käännös kestää kauemmin
 - suoritus tapahtuu nopeammin
 - milloin globaalin/paikallisen muuttujan X arvo kannattaa pitää rekisterissä ja milloin ei?
 - Missä rekisterissä X:n arvo kannattaa pitää?

24/10/2001 Copyright Teemu Kerola 2001 20

Optimoitu For lauseke (3)

```

for (int i=20; i < 50; ++i)
    T[i] = 0;
    
```

```

LOAD R1, =20 ; i
LOAD R2, =0 ; 0
Loop STORE R2, T(R1)
      ADD R1, =1
      COMP R1, =50
      JLES Loop
    
```

Mitä eroja? Onko tämä OK?

122 vs. 272 suoritetta käskyä!
muuttujan i arvo lopussa?
152 vs. 452 muistiviitettä!

alkuperäinen koodi

```

I DC 0
...
LOAD R1, =20
STORE R1, I
Loop LOAD R2, =0
      LOAD R1, I
      STORE R2, T(R1)
      LOAD R1, I
      ADD R1, =1
      STORE R1, I
      LOAD R3, I
      COMP R3, =50
      JLES Loop
    
```

24/10/2001 Copyright Teemu Kerola 2001 21

Virhetilanteisiin varautuminen (3)

- Suoritin tarkistaa käskyn suoritusajana
 - ”automaattinen”
 - integer overflow, divide by zero, ...
- Generoidut konekäskyt tarkistavat ja eksplisiittisesti aiheuttavat keskeytyksen tai käyttöjärjestelmän palvelupyynnön tarvittaessa
 - ”manuaalinen”
 - index out of bounds, bad method, bad operand, ihan mitä vain haluat testata!

```

ADD R1, R2 ; overflow??
DIV R4, =0 ; divide-by-zero
COMP R1, Tsize ; indeksin rajatarkistus
JLES IndexOK
SVC SP, =BadIndex ; käyttöjärj. huolehtii
IndexOK ADD R2, Taulu(R1) ; R1 = 12 345 000 ??
    
```

24/10/2001 Copyright Teemu Kerola 2001 22

Taulukon indeksitarkistus (1)

```

for (int i=20; i < 50; ++i)
    T[i] = 0;
    
```

```

I DC 0
T DS 50 ; data
Tsize DC 50 ; koko
...
LOAD R1, =20
STORE R1, I
Loop LOAD R2, =0
      LOAD R1, I
      JNNEG R1, ok1
      SVC SP, =BadIndex
      COMP R1, Tsize
      JLES ok2
      SVC SP, =BadIndex
      STORE R2, T(R1)
      LOAD R1, I
      ADD R1, =1
      STORE R1, I ; 50 OK!
      LOAD R3, I
      COMP R3, =50
      JLES Loop
    
```

Voisiko loopin kontrollia ja indeksin tarkistusta yhdistää? Optimoiva kääntäjä osaa!

24/10/2001 Copyright Teemu Kerola 2001 23

Taulukon alaindeksi ei ala nollassa (ei animoitu)

```

for (int i=20; i < 50; ++i)
    T[i] = 0;
    
```

```

I DC 0
T DS 30 ; 30 alkiota
Tlow DC 20 ; alaraja
Thigh DC 50 ; yläraja+1
...
    
```

indeksitarkistukset...

T: T[20]
T+1: T[21]
...
T+29: T[49]

24/10/2001 Copyright Teemu Kerola 2001 24

Taulukon alaindeksi ei ala nollassa (3)

```

for (int i=20; i < 50; ++i)
    T[i] = 0;
    
```

I	DC	0
T	DS	30 ; 30 alkiota
Tlow	DC	20 ; alaraja
Thigh	DC	50 ; yläraja+1
...		

indeksitarkistukset...

```

LOAD R1, =20
STORE R1, I
Loop LOAD R2, =0
LOAD R1, I
SUB R1, Tlow
STORE R2, T(R1)
LOAD R4, I
ADD R4, =1
STORE R4, I
LOAD R3, I
COMP R3, =50
JLES Loop
    
```

24/10/2001 Copyright Teemu Kerola 2001 25

Moni-ulotteiset taulukot (3)

- Ohjelmointikieli voi tukea suoraan moni-ulotteisia taulukoita


```
X = Tbl[i, j]; Y = Arr[k][6][y+2];
```
- Toteutus konekielitasolla aina (useimmissa arkkitehtuureissa) yksiulotteinen taulukko
 - vain yksi indeksirekisteri konekäskyssä
- Moniosainen toteutus
 - laske alkion osoite yksi-ulotteisessa taulukossa
 - käytä indeksoitua osoitusmodia tiedon viittaukseen

24/10/2001 Copyright Teemu Kerola 2001 26

2-ulotteiset taulukot (6)

```

int[][] T = new int[4][3];
...
Y = T[i][j];
    
```

T	DS	12
Trows	DC	4
Tcols	DC	3
...		

Esimerkki I=1, J=2 ?

```

LOAD R1, I
R1 MUL R1, Tcols
ADD R1, J
LOAD R2, T(R1)
STORE R2, Y
    
```

Tarkistukset... ?

looginen

T	0,0	0,1	0,2
	1,0	1,1	1,2
	2,0	2,1	2,2
	3,0	3,1	3,2

fyysinen

24/10/2001 Copyright Teemu Kerola 2001 27

Moni-ulotteiset taulukot (4)

- Talletus riveittäin
 - C, Pascal, Java?
- Talletus sarakkeittain
 - Fortran
- 3- tai useampi ulotteiset
 - samalla tavalla!

24/10/2001 Copyright Teemu Kerola 2001 28

Linkitetty lista (7)

R1: -1
R2: 132

```

First → Data Next → Data Next → Data Next → ⚡
                2-sanainen tietue
    
```

First=200:	211
R1 → 211:	33
R2: 0	255
R1 → 222:	55
R2: 77	-1
R1 → 255:	44
R2: 33	222

```

list_sum.k91
Data EQU 0 ; suht. osoite
Next EQU 1
Sum DC 0
Main LOAD R1, First ; ptrRec
JNEG R1, Done
LOAD R2, =0 ; sum
Loop ADD R2, Data(R1)
LOAD R1, Next(R1)
JNEG R1, Loop
STORE R2, Sum
SVC SP, =HALT
Done
    
```

Virhe, bugi! Missä?

24/10/2001 Copyright Teemu Kerola 2001 29

Monimutkaiset tietorakenteet

- 2-ulotteinen taulukko T, jonka jokainen alkio on tietue, jossa neljä kenttää:
 - pituus
 - ikä
 - viime vuoden palkka kunakin kuukautena
 - viime vuoden töissäolopäivien lukumäärä kunakin kuukautena
- Talletustapa?
- Viitteet?


```
X = T[yliopNum][opNum].palkka[kk];
```
- Tarkistukset?

24/10/2001 Copyright Teemu Kerola 2001 30

EDSAC

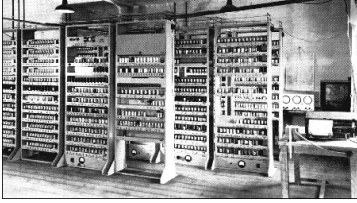

(Electronic Delay Storage Automatic Computer)

- Ensimmäinen toimiva ”todellinen” tietokone
 - ohjelma ja data samassa muistissa
 - Maurice Wilkes, Cambridge University
 - 1949
 - 256 sanan muisti
 - elohopeasäiliötekniologia
 - 35-bitin sanat



24/10/2001
Copyright Teemu Kerola 2001
31

EDSAC

Laitteisto

Muisti

(b) Mercury delay lines or "long tanks" for the main memory, with M. V. Wilkes looking on. The battery of 16 tanks shown here had a capacity of 512 words - the equivalent of a little over 1 Kilobyte.

24/10/2001
Copyright Teemu Kerola 2001
32

EDSAC Simulator

Symbolinen konekieli

```

PRINT SQUARES
31 T 123 S ] As required
enter → 32 E 84 S ] initial is
33 || P S ] Jump to 84
34 || P S ] Used to kee
of subtrac
Power of 10
subtracted
35 P10000 S ] For use in
36 P 1000 S ] binary con
37 P 100 S
38 P 10 S
39 P 1 S
40 Q S
41 π S Figures
42 S 40 S
                
```

Konekieli

```

[Square]
01028884828891000691000810081009108118
02881402154888043003288A65
16512159455275496125495847576584328A05
1332A4889348558A348974887335A325A45
10320420010A111074822087898
3110811881008850A181230548048A748A45
0795T482A8587788498048904887687761648
132877578788788A7707878488A077588A45
04930432A798T488A81877588495A352A7688825
055041823
                
```

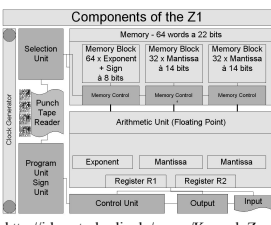

<http://www.dcs.warwick.ac.uk/~edsac/>

24/10/2001
Copyright Teemu Kerola 2001
33

-- Luennon 3 loppu --

Konrad Zuse: Z1 (1938)

- mekaaninen ”laskin”, kelloaajuus 1 Hz
- kertolasku 5 s
- datamuisti 64W à 24b
- ohjelma reikänauhalla (filmiltä)

http://irb.cs.tu-berlin.de/~zuse/Konrad_Zuse/en/Rechner_Z1.html

24/10/2001
Copyright Teemu Kerola 2001
34