

Luento 3

Konekielinen ohjelmointi (TTK-91, KOKSI)

Muuttujat
Tietorakenteet
Kontrolli
Optimointi
Tarkistukset

25/03/2004 Copyright Teemu Kerola 2003 1

Tiedon sijainti suoritusaikana

- Muistissa (=keskusmuistissa)
 - iso Esim. 256 MB, tai 64 milj. 32 bitin sanaa
 - hidas
 - data-alueella Esim. 10 ns kysä vakiona?
- Rekisterissä
 - pieni
 - nopea Esim. 256 B, tai 64 kpl 32 bitin sanaa
- Probleemi: muuttujat muistissa ja n Esim. 1 ns steri TTK-91: 8 kpl + PC + ...
 - missä päin muistia? miten siihen viitataan?

25/03/2004 Copyright Teemu Kerola 2003 2

Miten tietoon viitataan? (3)

- Tieto muistissa
 - muistiosoitteen (esim. 0x6F123456 tai 3459321) avulla
 - symbolin (esim. HenkTunn tai X) avulla symbolista konekieltä käytettäessä – symbolin arvo on muistiosoite
 - HenkTunn = 0x6F123456, X = 3459321
(heksadesimal) (desimal)
- Tieto välimuistissa
 - samalla tavalla kuin jos tieto olisi muistissa
 - viittaushetkellä ei tiedetä, kummasta paikasta tieto lopulta löytyy tai kauanko viittaamiseen kuluu aikaa!
- Tieto rekisterissä
 - rekisterin osoitteen (esim. 6 tai 18) avulla
 - syb. konekieli: R3, FP, F5, I2, jne
- Tieto konekäskyssä (vakiona)
 - oletusarvoisesti, käskyssä on vain yksi paikka tiedolle

25/03/2004 Copyright Teemu Kerola 2003 3

Tieto ja sen osoite (2)

```

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

muuttujan X osoite on symbolin X arvo

muuttujan X arvo

muisti	
230	
12345	
12556	
128765	
12222	
12	X=230:
12998	

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

25/03/2004 Copyright Teemu Kerola 2003 4

Tieto ja sen osoite (3)

```

Xptr DC 0
X DC 12
LOAD R1, =X ; R1 ← 230
STORE R1, Xptr
LOAD R2, X ; R2 ← 12
LOAD R3, @Xptr ; R3 ← 12
            
```

Xptr=225

muisti	
230	
12345	
12556	
128765	
12222	
12	X=230:
12998	

- Muuttujan X osoite on 230
- Muuttujan X arvo on 12
- Osoitinmuuttujan (pointterin) Xptr osoite on 225 X=230:
- Osoitinmuuttujan Xptr arvo on 230
 - jonkun tiedon osoite (nyt X:n osoite)
- Osoitinmuuttujan Xptr osoittaman kokonaisluvun arvo on 12
 - C-kieli: Y = *ptrX /* ei prtX:n arvo, mutta ptrX:n osoittaman muuttujan arvo */

25/03/2004 Copyright Teemu Kerola 2003 5

Osoitinmuuttujat (4)

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

25/03/2004 Copyright Teemu Kerola 2003 6

Globaali, kaikkialla näkyvä data ⁽²⁾

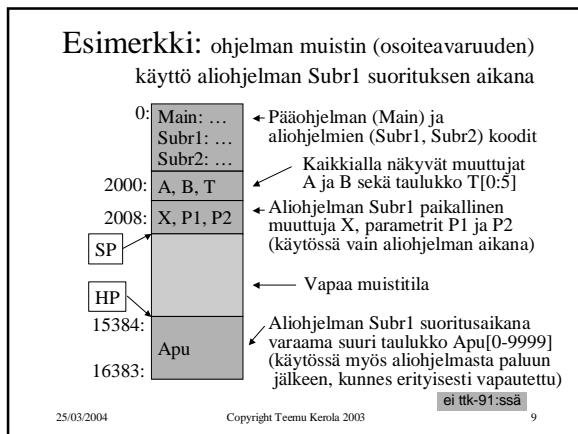
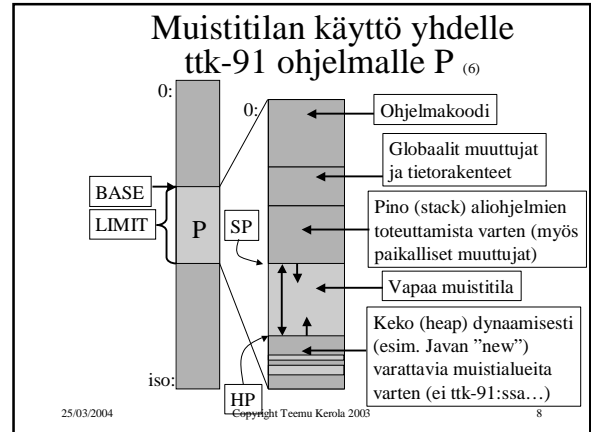
- Globaalit muuttujat ja muut globaalit tietorakenteet sijaitsevat ttk-91 koneen muistissa ohjelmakoodin jälkeen
 - muuttujat

int X = 25;	char Ch;
short Y;	char Str[] = "Pekka";
float Ft;	boolean fBig;
 - tilan varaus

X	DC	15 ; alkuarvo 15
Taulu	DS	20 ; 20 sanaa
fBig	DC	1 ; 1=true, 0=false
 - viittaaminen

LOAD	R1, X
STORE	R2, Taulu(R1)

25/03/2004 Copyright Teemu Kerola 2003 7



- ### Muistissa oleva data ⁽²⁾
- Globaali data `int X; function Print();`
 - varataan ohjelman latauksen yhteydessä
 - kaikkialla viitattavissa nimen (osoitteen) avulla
 - Dynaaminen data `Mach m = new Mach();`
 - (ei ttk-91:ssä) varataan tarvittaessa keosta suorituksen aikana
 - vapautetaan kun ei enää tarvita
 - viittaus varauksen jälkeen osoitteen avulla
 - Aliohjelmien paikallinen data `parametrit, paik. muuttuja`
 - varataan pinosta kutsuhetkellä
 - vapautetaan rutiinista paluun yhteydessä
 - viittaus aliohjelman sisällä osoitteen avulla
- 25/03/2004 Copyright Teemu Kerola 2003 10

- ### Tiedon sijainti suoritusajana ⁽³⁾
- Rekisteri (nopein)
 - kääntäjä päättää milloin muuttujan arvo on rekisterissä
 - Välimuisti (nopea)
 - laitteisto hoitaa automaattisesti joillekin muistialueille
 - Muisti (hidas)
 - kääntäjä/lataaja valitsee sijaintipaikan
 - globaali data ohjelman latauksen yhteydessä
 - vakiot konekäskyssä
 - ohjelma sijoittaa suoritusajana
 - aliohjelmien paikalliset muuttujat, parametrit
 - käyttöjärjestelmä sijoittaa suoritusajana
 - dynaaminen data keossa suorituksen aikana
 - Levy, levypalvelin (liian hidas, ei mahdollista)
 - vaatii käyttöjärjestelmän varusohjelmien apua
- 25/03/2004 Copyright Teemu Kerola 2003 11

- ### Ohjelmoinnin peruskäsitteet
- 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, virheilanteet
 - Monimutkaiset tietorakenteet
 - listat, moniulotteiset taulukot
- 25/03/2004 Copyright Teemu Kerola 2003 12

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	

25/03/2004 Copyright Teemu Kerola 2003 13

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	

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

25/03/2004 Copyright Teemu Kerola 2003 14

Globaalien tietueiden tilan varaus ja käyttö (3)

```

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

X	DC	0
Tauno	DS	2
Pituus	EQU	0
Paino	EQU	1
...		
	LOAD R1, =Tauno	
	LOAD R2, Paino(R1)	
	STORE R2, X	

Kentän "Paino" suhteellinen osoite tietueen Tauno sisällä

Tietueen osoite on sen ensimmäisen sanan osoite

25/03/2004 Copyright Teemu Kerola 2003 15

Kontrolli - valinta konekielellä (2)

- Ehdoton hyppy
 - JUMP, CALL ja EXIT, SVC ja IRET
- Hyppy perustuen laitekisterin arvoon (vrt. 0)
 - JZER, JPOS, ...
- Hyppy perustuen aikaisemmin asetettuun tilarekisterin arvoon
 - COMP R2, LIMIT
 - JEQU LOOP
 - JZER, JGRE, ...
 - Ongelma vai etu: ttk-91:ssä kaikki ALU käskyt asettavat tilarekisterin (implisiittinen "COMP Rj, =0")
 - ADD, SUB, MUL, DIV, NOT, AND, OR, XOR, SHL, SHR

25/03/2004 Copyright Teemu Kerola 2003 16

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:

25/03/2004 Copyright Teemu Kerola 2003 17

Case lauseke (2)

```

switch (lkm) {
    case 4: x = 11;
            break;
    case 0: break;
    default: x = 0;
            break;
}
    
```

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

Onko case-tapausten järjestyksellä väliä?

25/03/2004 Copyright Teemu Kerola 2003 18

Toistolausekkeet (2)

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

ehto silmukan alussa

ehto silmukan lopussa

25/03/2004 Copyright Teemu Kerola 2003 19

For lauseke (4)

```

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
    
```

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?

Entä jos toisenlainen loop-semantiikka?

25/03/2004 Copyright Teemu Kerola 2003 20

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 missä rekisterissä ja milloin? X in R3?

25/03/2004 Copyright Teemu Kerola 2003 21

Koodin generointi (7)

- Kääntäjän viimeinen vaihe
 - voi olla 50% käännösajasta
- Tavallisen koodin generointi
 - alustukset, lausekkeet, kontrollirakenteet
- Optimoitun koodin generointi
 - käännös kestää (paljon) kauemmin
 - suoritus tapahtuu (paljon) nopeammin
 - milloin globaalin/paikallisen muuttujan X arvo kannattaa pitää rekisterissä ja milloin ei?
 - missä rekisterissä X:n arvo kannattaa pitää?
 - joskus R1:ssä, joskus R5:ssä?

25/03/2004 Copyright Teemu Kerola 2003 22

Optimoitu For lauseke (2)

```

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?

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
    
```

6 vs. 11 konekäskyä (koodin koko)
122 vs. 272 suoritetua käskyä!
muuttujan i arvo lopussa?
152 vs. 452 muistiviitettä!

25/03/2004 Copyright Teemu Kerola 2003 23

Virhetilanteisiin varautuminen (3)

- Suoritin tarkistaa käskyn suoritusajana
 - "automaattinen", HW
 - integer overflow, ADD R1, R2 ; overflow??
 - divide by zero, DIV R4, =0 ; 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!

```

COMP R1, Tsize ; indeksin rajatarkistus
JLES IndexOK ; Taulu:n koko Tsize:ssä
SVC SP, =BadIndex ; käyttöjärj. huolehtii
IndexOK STORE R2, Taulu(R1) ; R1 = 12 345 000 ??
    
```

25/03/2004 Copyright Teemu Kerola 2003 24

Taulukon indeksitarkistus (1)

```

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

I DC 0

T DS 50 ; data
Tsize DC 50 ; koko
...

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

```

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
ok2 STORE R2, T(R1)
LOAD R1, I
ADD R1, =1
STORE R1, I ; 50 OK!
LOAD R3, I
COMP R3, =50
JLES Loop
    
```

25/03/2004 Copyright Teemu Kerola 2003 25

Taulukon alaindeksi ei ala nolasta (ei animoitu)

```

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

I DC 0

T DS 30 ; 30 alkia
Tlow DC 20 ; alaraja
Thigh DC 50 ; yläraja+1
...

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

indeksitarkistukset...

25/03/2004 Copyright Teemu Kerola 2003 26

Taulukon alaindeksi ei ala nolasta (3)

```

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

I DC 0

T DS 30 ; 30 alkia
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
    
```

25/03/2004 Copyright Teemu Kerola 2003 27

Moni-ulotteiset taulukot (2)

- Ohjelmointikieli voi tukea suoraan moni-ulotteisia taulukoita
 $X = Tbl[i, j]; \quad Y = Arr[k][6][y+2];$
- Toteutus konekielitasolla aina (useimmissa arkkitehtuureissa) yksilotteinen taulukko
 - vain yksi indeksirekisteri konekäskyssä
- Moniosainen toteutus
 - laske alkion osoite yksi-ulotteisessa taulukossa ja käytä indeksoitua tiedonosoitusmoodia
 - TAI: laske alkion osoite muistissa ja käytä epäsuoraa tiedonosoitusmoodia

25/03/2004 Copyright Teemu Kerola 2003 28

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 ?

T: T[0][0]
T[0][1]
T[0][2]
T[1][0]
T[1][1]
T[1][2]
T[2][0]
T[...][...]

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

looginen fyysinen

```

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

Tarkistukset.... ?

25/03/2004 Copyright Teemu Kerola 2003 29

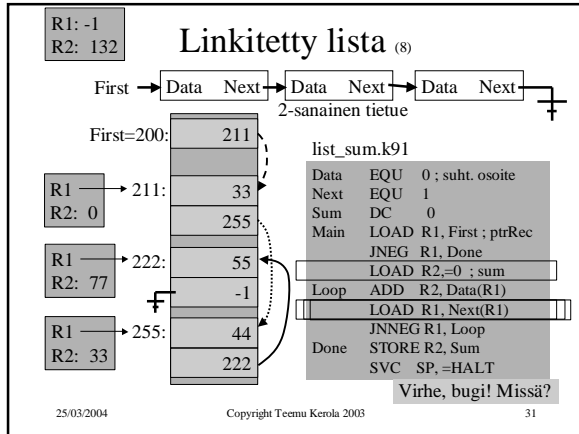
Moni-ulotteiset taulukot (3)

- Talletus riveittäin
 - C, Pascal, Java?
- Talletus sarakeittain
 - Fortran
- 3- tai useampi ulotteiset
 - vastaavalla tavalla!

T: T[0][0]
T[0][1]
T[0][2]
T[1][0]
T[1][1]
T[1][2]
T[2][0]
T[...][...]

T: T[0][0]
T[1][0]
T[2][0]
T[3][0]
T[0][1]
T[1][1]
T[2][1]
T[...][...]

25/03/2004 Copyright Teemu Kerola 2003 30



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?
- Tarkistukset?

$$X = T[\text{yliopNum}][\text{opNum}].\text{palkka}[\text{kk}];$$

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

EDSAC

Laitteisto

Muisti

EDSAC Simulator

Symbolinen konekieli

```

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

Konekieli

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

-- Luennon 3 loppu --

Konrad Zuse: Z1 (1938)

- mekaaninen ”laskin”, kellotaajuus 1 Hz (käännä kampea!)
- kertolasku 5 s
- datamuisti 64W à 24b
- ohjelma reikänauhalta (filmiltä)

Components of the Z1

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