

Luento 3

Konekielinen ohjelmointi (TTK-91, KOKSI)

Muuttujat
Tietorakenteet
Kontrolli
Optimointi
Tarkistukset

09/08/2002

Copyright Teemu Kerola 2002

1

Tiedon sijainti suoritusaikana ⁽³⁾

- Muistissa (=keskusmuistissa)
 - iso Esim. 256 MB, tai 64 milj. 32 bitin sanaa
 - hidas Esim. 10 ns
 - data-alueella vai konekäskyssä vakiona?
- 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?

09/08/2002

Copyright Teemu Kerola 2002

2

Miten tietoon viitataan? ⁽⁴⁾

- 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
- 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
- Tieto konekäskyssä (vakiona)
 - oletusarvoisesti, käskyssä on vain yksi paikka tiedolle

09/08/2002

Copyright Teemu Kerola 2002

3

Tieto ja sen osoite ⁽³⁾

muuttujan X osoite on symbolin X arvo

symbolin X arvo muuttujan X osoite?

muuttujan X arvo

muisti

230
12345
12556
128765
12222
12
12998

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

09/08/2002

Copyright Teemu Kerola 2002

4

Tieto ja sen osoite ⁽⁶⁾

```

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

```

muisti	
Xptr=225:	230
	12345
	12556
	128765
	12222
X=230:	12
	12998

- 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

C-kieli: Y = *ptrX

09/08/2002
Copyright Teemu Kerola 2002
5

Osoitinmuuttujat ⁽⁵⁾

- 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
 - 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

09/08/2002
Copyright Teemu Kerola 2002
6

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;
```

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

– tilan varaus

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

– viittaaminen

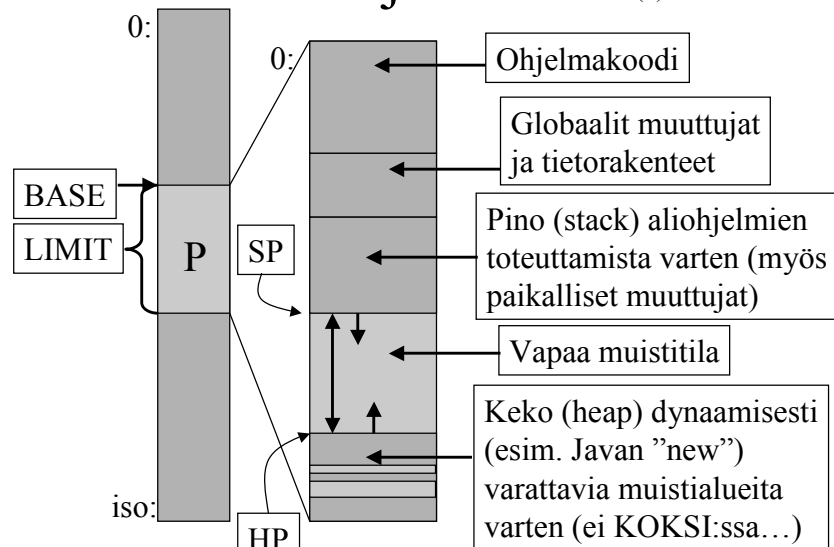
```
LOAD      R1, X
STORE     R2, Taulu(R1)
```

09/08/2002

Copyright Teemu Kerola 2002

7

Muistitilan käyttö yhdelle ttk-91 ohjelmalle P ⁽⁶⁾

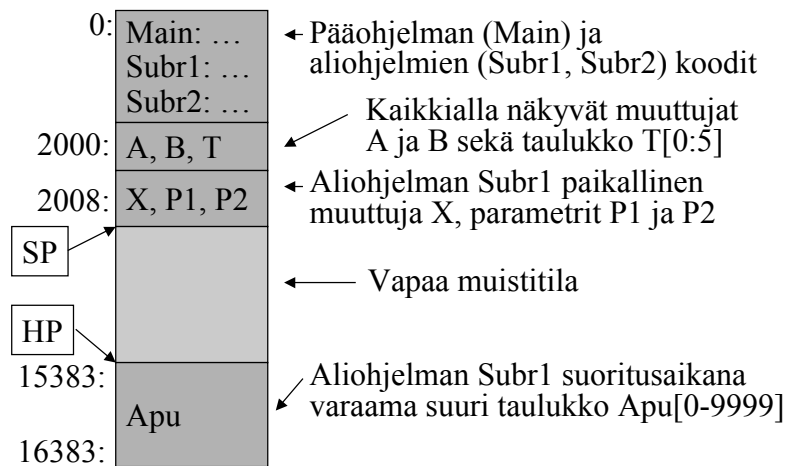


09/08/2002

Copyright Teemu Kerola 2002

8

Esimerkki: ohjelman muistin (osoiteavaruuden) käyttö aliohjelman Subr1 suorituksen aikana



09/08/2002

Copyright Teemu Kerola 2002

9

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

09/08/2002

Copyright Teemu Kerola 2002

10

Tiedon sijainti suoritusaikana ⁽⁴⁾

- 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 suoritusaikana
 - aliohjelmien paikalliset muuttujat, parametrit
 - käyttöjärjestelmä sijoittaa suoritusaikana
 - dynaaminen data keossa suorituksen aikana
- Levy, levypalvelin (liian hidas, ei mahdollista)
 - vaatii käyttöjärjestelmän varusohjelmien apua

09/08/2002

Copyright Teemu Kerola 2002

11

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

09/08/2002

Copyright Teemu Kerola 2002

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
```

09/08/2002
Copyright Teemu Kerola 2002
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		

X:
Y:
Taulu:

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

09/08/2002
Copyright Teemu Kerola 2002
14

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	
Tauno	DS	2
Pituus	EQU	0
Paino	EQU	1

Tietueen osoite on sen ensimmäisen sanan osoite

X:

Tauno:

pituus
paino

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

09/08/2002
Copyright Teemu Kerola 2002
15

Kontrolli - valinta konekielellä ⁽³⁾

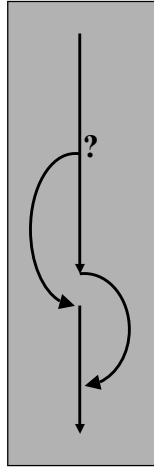
- Ehdoton hyppy
 - JUMP, CALL ja EXIT, SVC ja IRET
- Hyppy perustuen laiterekisterin arvoon (vrt. 0)
 - JZER, JPOS, ...
- Hyppy perustuen aikaisemmin asetetun tilarekisterin arvoon
 - COMP
 - JEQU, JGRE, ...
 - Ongelma vai etu: ttk-91:ssä kaikki ALU käskyt asettavat tilarekisterin
 - ADD, SUB, MUL, DIV,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

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

09/08/2002
Copyright Teemu Kerola 2002
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:

09/08/2002

Copyright Teemu Kerola 2002

17

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
```

09/08/2002

Copyright Teemu Kerola 2002

18

Toistolausekkeet (2)

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

ehto
silmukan
alussa

ehto
silmukan
lopussa

09/08/2002
Copyright Teemu Kerola 2002
19

For lauseke (3)

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

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		

Olisiko parempi pitää i:n arvo rekisterissä?
 Miksi? Milloin?
 Mikä on i:n arvo lopussa?
 Onko sitä olemassa?

09/08/2002
Copyright Teemu Kerola 2002
20

While-do -lauseke (2)

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

Mitä kannattaa pitää muistissa?

Mitä kannattaa pitää rekisterissä ja milloin?

```
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
```

09/08/2002

Copyright Teemu Kerola 2002

21

Koodin generointi (9)

- Kääntäjän viimeinen vaihe
 - voi olla 50% käänösajasta
- Tavallisen koodin generointi
 - alustukset, lausekkeet, kontrollirakenteet
- Optimoidun koodin generointi
 - käänös 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?

09/08/2002

Copyright Teemu Kerola 2002

22

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 suoritettua 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
```

09/08/2002 Copyright Teemu Kerola 2002 23

Virhetilanteisiin varautuminen ⁽³⁾

- Suoritin tarkistaa käskyn suoritusaikana
 - ”automaattinen”
 - integer overflow, divide by zero, ...

```
ADD R1, R2 ; overflow??
DIV R4, =0 ; divide-by-zero
```
- Generoidut konekäskyt tarkistavat ja explisiittisesti 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
SVC SP, =BadIndex ; käyttöjärj. huolehtii
IndexOK STORE R2, Taulu(R1) ; R1 = 12 345 000 ??
```

09/08/2002 Copyright Teemu Kerola 2002 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!

```

Loop
LOAD R1, =20
STORE R1, I
LOAD R2, =0
LOAD R1, I
ok1
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
    
```

09/08/2002
Copyright Teemu Kerola 2002
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 alkiota
Tlow	DC	20 ; alaraja
Thigh	DC	50 ; yläraja+1
...		

indeksitarkistukset...

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

09/08/2002
Copyright Teemu Kerola 2002
26

Taulukon alaindeksi ei ala nolasta ⁽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
```

09/08/2002

Copyright Teemu Kerola 2002

27

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 osoitusmoodia tiedon viittaukseen

09/08/2002

Copyright Teemu Kerola 2002

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 ?

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

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

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

fyysinen

Tarkistukset.... ?

09/08/2002
Copyright Teemu Kerola 2002
29

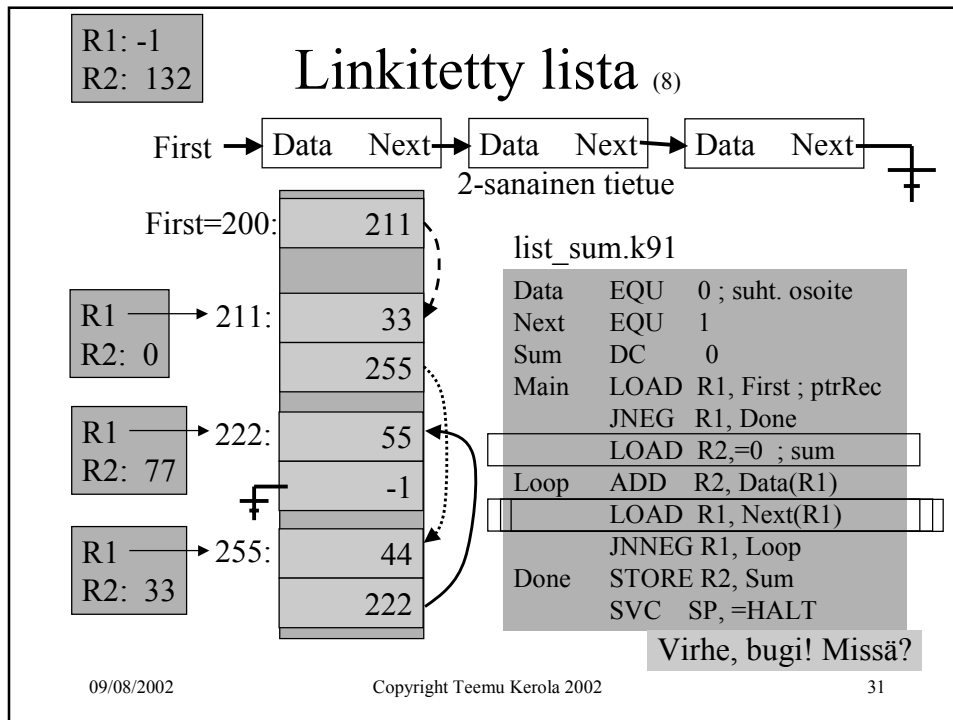
Moni-ulotteiset taulukot (3)

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

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

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

09/08/2002
Copyright Teemu Kerola 2002
30



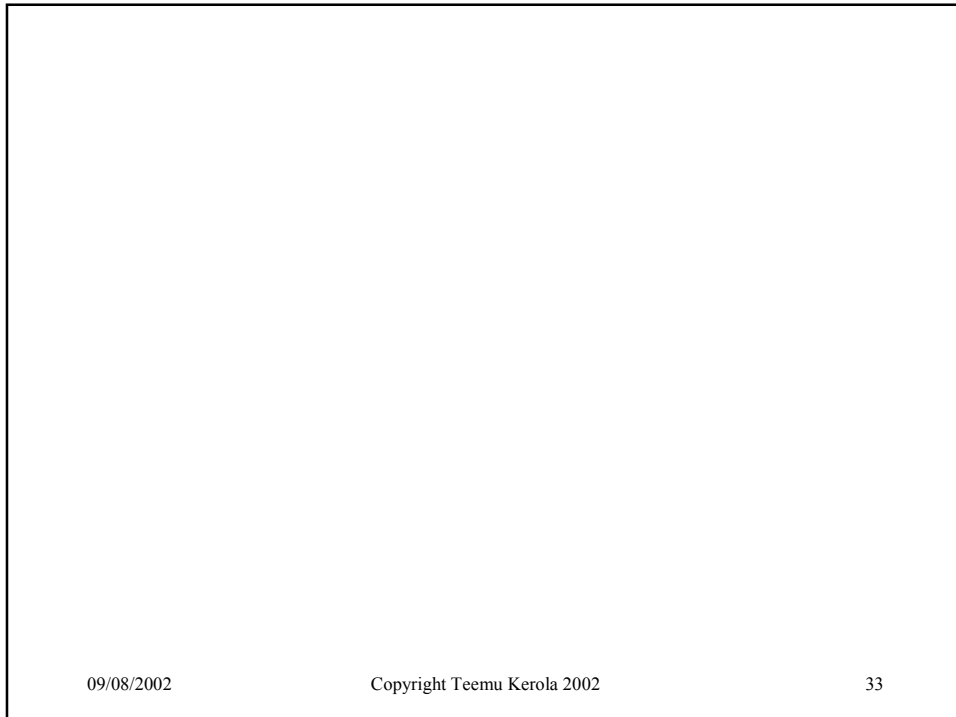
Monimutkaiset tietorakenteet

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

09/08/2002

Copyright Teemu Kerola 2002


32



EDSAC

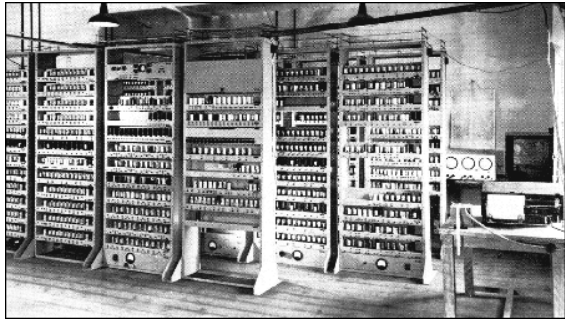

(Electronic Delay Storage Automatic Computer)

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



09/08/2002 Copyright Teemu Kerola 2002 34

EDSAC

(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.

Laitteisto
Muisti

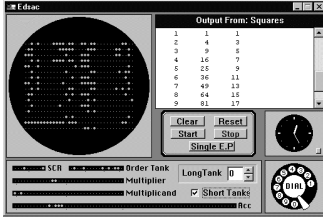
09/08/2002
Copyright Teemu Kerola 2002
35

EDSAC Simulator

Symbolinen konekieli

```

PRINT SQUARES
31 T 123 S ] As required
enter → 32 E 84 S ] initial in
33 || P S ] Jump to 84
34 || P S ] Used to kee
] of subtrac
35 P1000 S ] Power of 10
36 P 1000 S ] subtracted
37 P 100 S ] For use in
38 P 10 S ] binary con
39 P 1 S ]
40 Q S ]
41 π S ] Figures
42 A 40 S ]
                
```



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

Konekieli

```

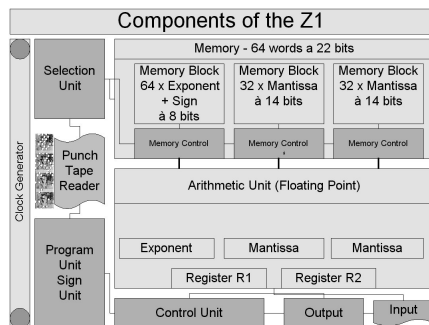
[ Squares ]
T123S E84SPSP1000SP1000SP1008P10SP1S
QS#SA40S1S4S8S043S033SPSA46S
T65ST129SA35ST34S E61ST48SA47ST65SA33SA40S
T93SA488S34S E55SA34SPST48ST33SA52SA4S
USCS#42SGS1SA1178T6ZSPFSPFSPSPS
E110S E118SP100S E95S041ST12S044S045SA76SA4S
U76ST48SA3ST75S E4S043S043SH76SV76S164S
L32SU778S78ST79SA77SU78ST48SA80ST75S E49S
043S043SA79ST48SA81ST75S E49SA3S SA76S8S2S
G85S041S2S
                
```

09/08/2002
Copyright Teemu Kerola 2002
36

-- Luennon 3 loppu --

Konrad Zuse: Z1 (1938)

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



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

09/08/2002

Copyright Teemu Kerola 2002

37