

# Luento 3

## Konekielinen ohjelmointi (TTK-91, KOKSI)

Muuttujat  
Tietorakenteet  
Kontrolli  
Optimointi  
Tarkistukset

24/10/2001

Copyright Teemu Kerola 2001

1

## Tiedon sijainti suoritusaikana <sup>(3)</sup>

- 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 <sup>(3)</sup>

```

X DC 12
...
LOAD R1, =X
LOAD R2, X
int x =12;

```

muuttujan X osoite on symbolin X arvo

symbolin X arvo

muuttujan X arvo

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

- 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 <sup>(5)</sup>

```

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

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

24/10/2001 Copyright Teemu Kerola 2001 4

## Osoitinmuuttujat <sup>(5)</sup>

- 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 <sup>(2)</sup>

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

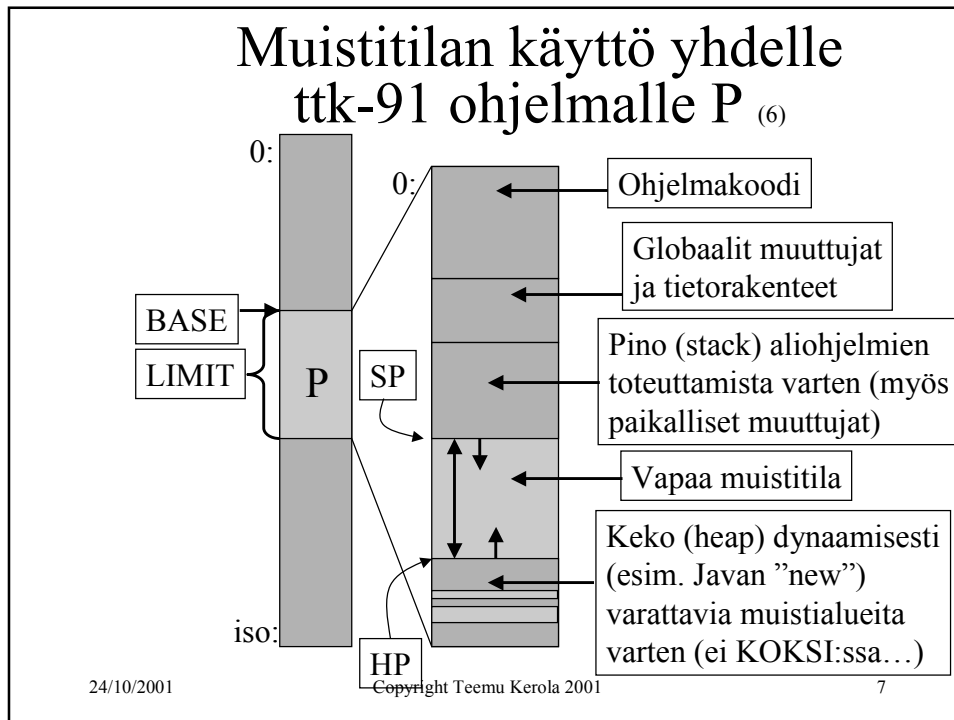
– viittaaminen

```
LOAD      R1, X
STORE     R2, Y
```

24/10/2001

Copyright Teemu Kerola 2001

6



### Muistissa oleva data <sup>(3)</sup>

- **Gloaali 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 <sup>(4)</sup>

- Rekisteri
  - nopein, kääntäjä varaa/vapauttaa
- Välimuisti
  - nopea, laitteisto hoitaa automaattisesti
- Muisti
  - ohjelma varaa/vapauttaa
    - aliohjelmien paik. muuttujat, parametrit
  - käyttöjärj. 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 <sup>(4)</sup>

- 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
```

X:  
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 <sup>(3)</sup>

```
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
```

24/10/2001
Copyright Teemu Kerola 2001
13

## Kontrolli - valinta konekielellä <sup>(3)</sup>

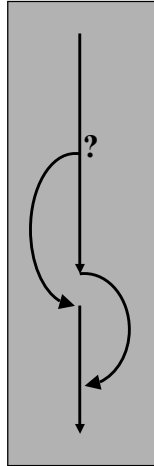
- Ehdoton hyppy
  - JUMP, CALL ja EXIT, SVC ja IRET
- Hyppy perustuen laiterekisterin arvoon (vrt. 0)
  - JZER, JPOS, ...
- Hyppy perustuen aikaisemmin asetetun tilarekisterin arvoon
  - COMP
  - JEQU, JGRE, ...
  - Ongelma vai etu: ttk-91:ssä kaikki ALU käskyt asettavat tilarekisterin
    - ADD, SUB,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

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

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

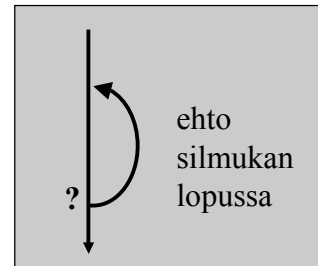
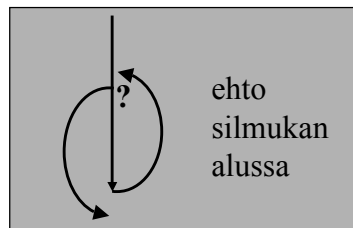
Copyright Teemu Kerola 2001

16



## Toistolausekkeet (2)

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



24/10/2001

Copyright Teemu Kerola 2001

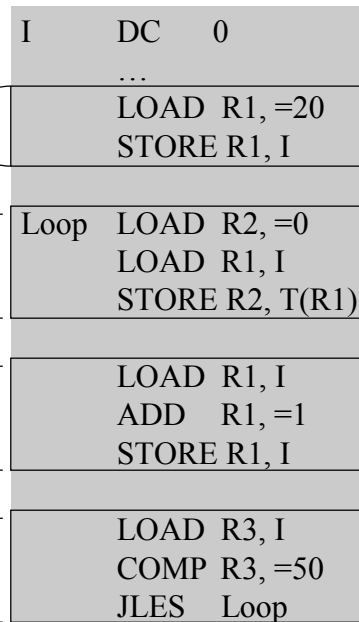
17

## 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?



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;
}
```

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
```

24/10/2001

Copyright Teemu Kerola 2001

19

## Koodin generointi (9)

- Kääntäjän viimeinen vaihe
  - voi olla 50% käänösajasta
- Tavallisen koodin generointi
  - alustukset, lausekkeet, kontrollirakenteet
- Optimoidun koodin generointi
  - käänös 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 <sup>(3)</sup>

```
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
```

24/10/2001 Copyright Teemu Kerola 2001 21

## Virhetilanteisiin varautuminen <sup>(3)</sup>

- 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 ADD R2, Taulu(R1) ; R1 = 12 345 000 ??
```

24/10/2001 Copyright Teemu Kerola 2001 22

## Taulukon indeksitarkistus <sup>(1)</sup>

```
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
ok1   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
```

24/10/2001
Copyright Teemu Kerola 2001
23

## 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]

24/10/2001
Copyright Teemu Kerola 2001
24

## Taulukon alaindeksi ei ala nolasta <sup>(3)</sup>

```
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 <sup>(3)</sup>

- 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

24/10/2001

Copyright Teemu Kerola 2001

26

## 2-ulotteiset taulukot <sup>(6)</sup>

```
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.... ?

24/10/2001
Copyright Teemu Kerola 2001
27

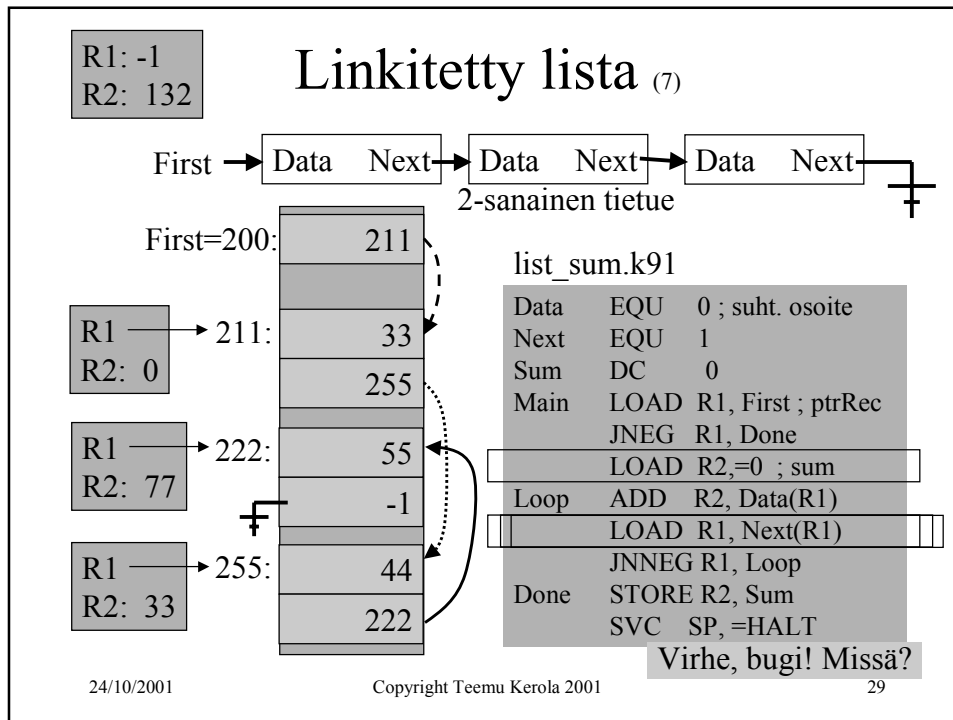
## Moni-ulotteiset taulukot <sup>(4)</sup>

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

T:

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

24/10/2001
Copyright Teemu Kerola 2001
28



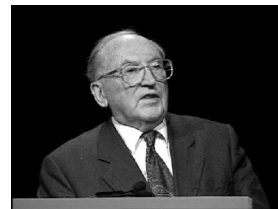
## Monimutkaiset tietorakenteet

- 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?

# 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

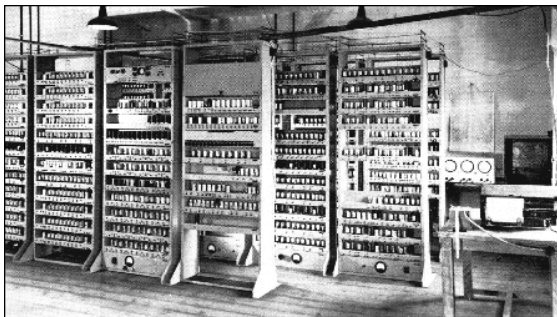


24/10/2001

Copyright Teemu Kerola 2001

31

# EDSAC



Laitteisto



Muisti

24/10/2001

Copyright Teemu Kerola 2001

32

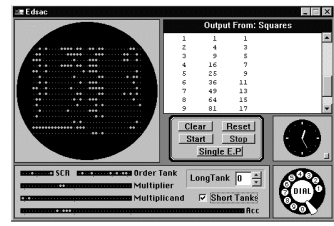


# EDSAC Simulator

## Symbolinen konekieli

```

PRINT SQUARES
      31 | T 123 S | ] As required
enter → 32 | E 84 S | ] initial in
          |-----| ] Jump to 84
      33 | || P S | ] Used to kee
          |-----| ] of subtrac
      34 | || P S | ] Power of 10
          |-----| ] subtracted
      35 | P1000 S | ]
          |-----| ]
      36 | P 1000 S | ] For use in
          |-----| ] binary con
      37 | P 100 S | ]
          |-----| ]
      38 | P 10 S | ]
          |-----| ]
      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 E84SPSP1000SP1000SP100SP10SP1S
QS#SA40S1S6S8S043S033SPSA46S
T65STL29SA35ST34SE61ST48SA47ST65SA33SA40S
T33SA49SP34S53SA34SP48ST33SA52SA4S
US2S9428S61S117ST62SPSPSPSPSPSP
E110SE118SP100SE9S8041STL29S044S045SA76SA4S
U76ST48SA83ST75SE49S043S043SH76SV76S164S
L32SU77S78ST79SA77SU78ST48SA80ST75SE49S
043S043SA79ST48SA81ST75SE49SA35SA76S8S2S
G85S041S2S
          
```

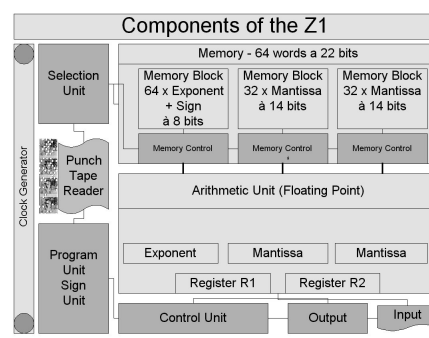
24/10/2001
Copyright Teemu Kerola 2001
33


## -- Luennon 3 loppu --

**Konrad Zuse: Z1 (1938)**

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

### Components of the Z1





[http://irb.cs.tu-berlin.de/~zuse/Konrad\\_Zuse/en/Rechner\\_Z1.html](http://irb.cs.tu-berlin.de/~zuse/Konrad_Zuse/en/Rechner_Z1.html)

24/10/2001
Copyright Teemu Kerola 2001
34