

Kertausluento 1 (lu01, lu02, lu03)

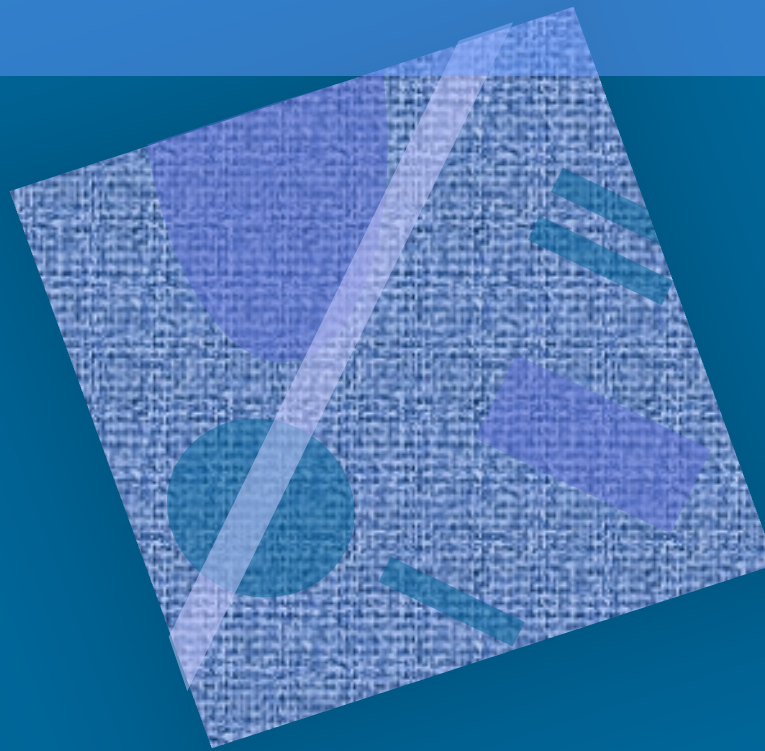
Tietokonejärjestelmän rakenne ttk-91 ja sillä ohjelmointi



Järjestelmän eri tasot
Laitteiston nopeus
ttk-91 rakenne ja käskykanta-
arkkitehtuuri
Konekielinen ohjelmointi
ttk-91:llä
(Titokone, TitoTrainer)

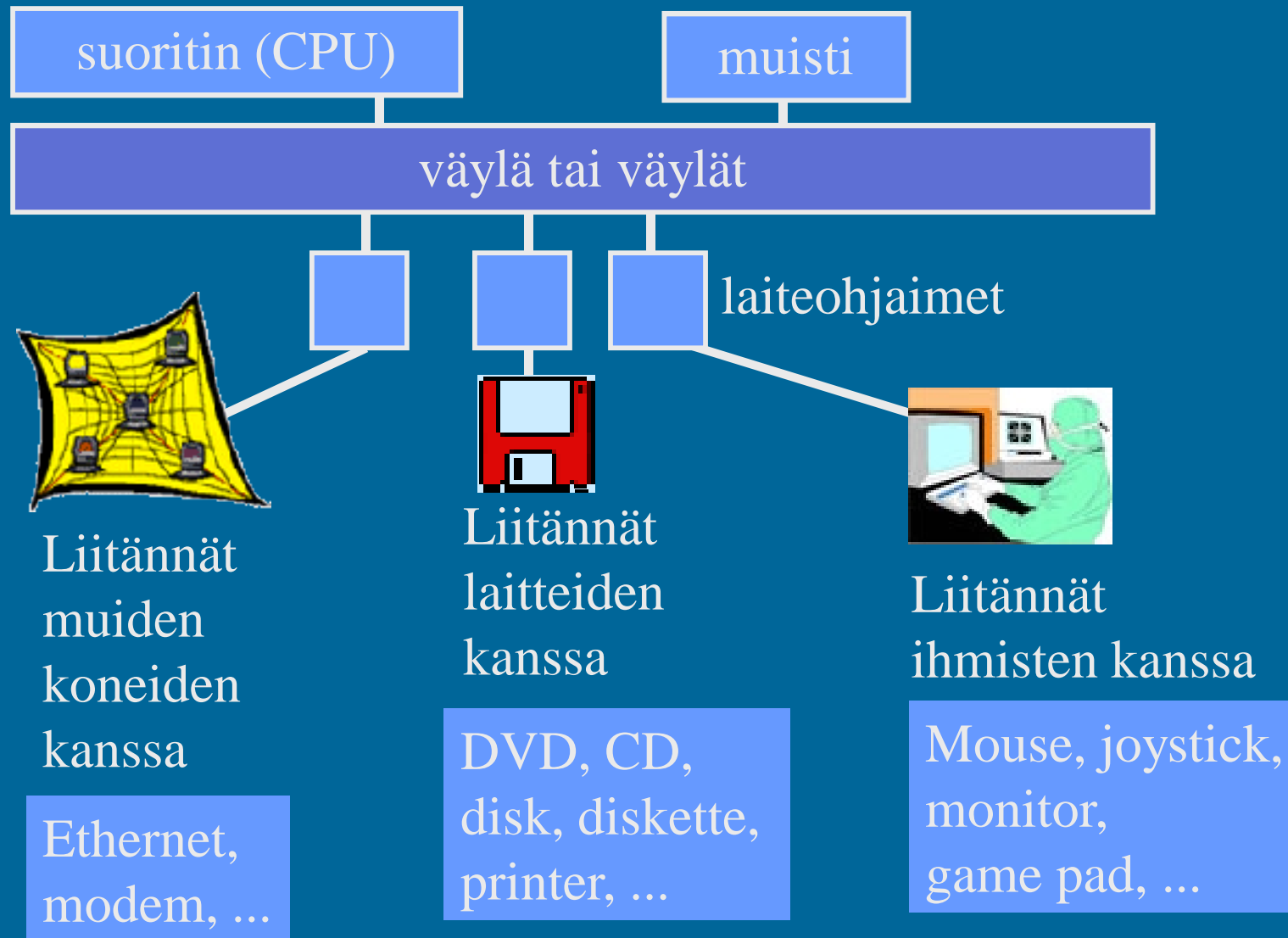
Luento 1

Tietokonejärjestelmän rakenne



Järjestelmän eri tasot
Laitteiston nopeus

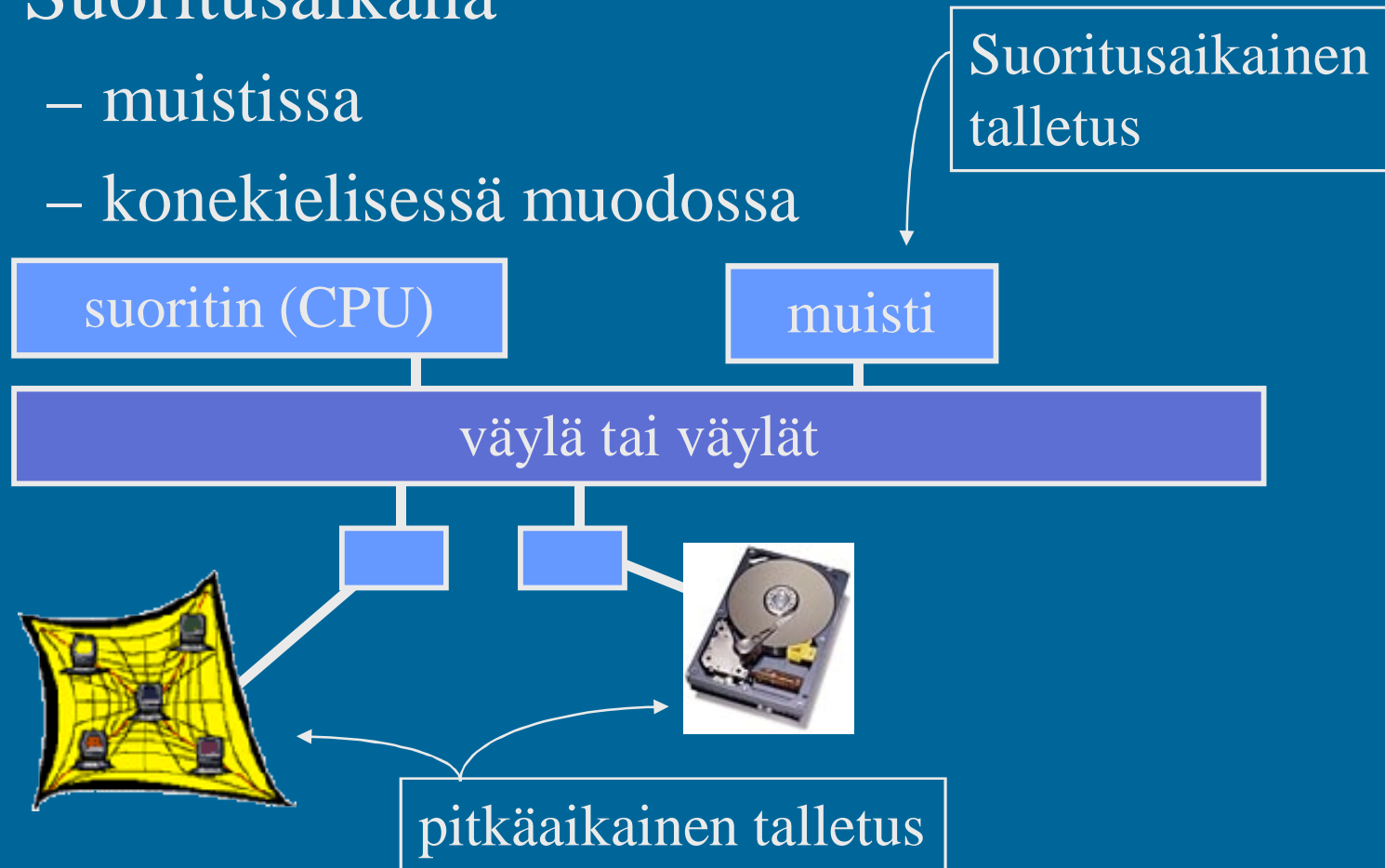
Tietokone






Tietokoneohjelman sijainti ja esitysmuoto (jatkuu)

- Suoritusaikana

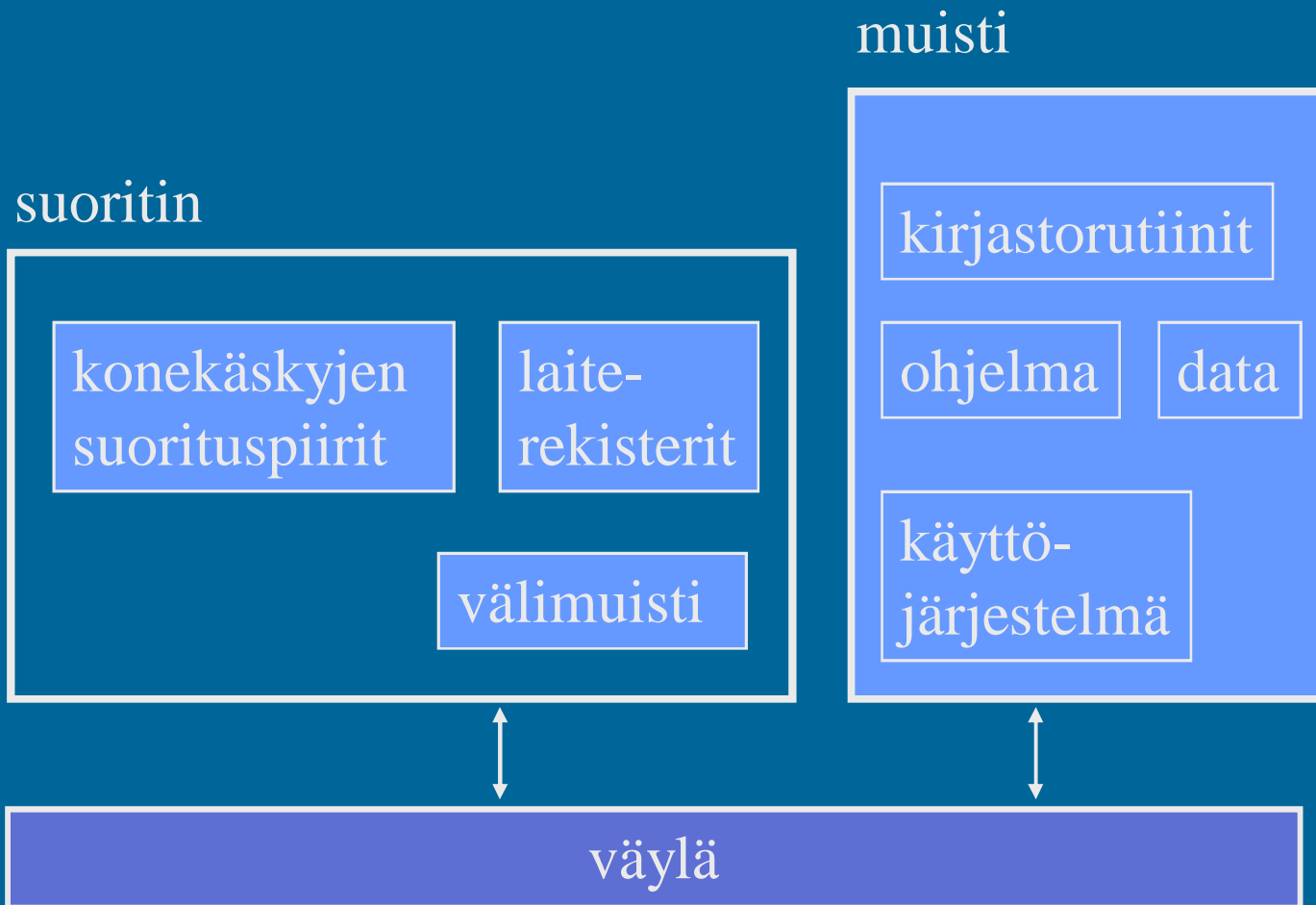
- muistissa
- konekielisessä muodossa



Konekieli

- Suorittimen konekielen käskykanta määrittelee tietokoneen käskykanta-arkkitehtuurin
 - ISA - Instruction Set Architecture
- Kukin käsky on esim. 10-numeroinen kokonaisluku 

- Usein esitetty symbolisella konekielellä
 - käsky jaettu osiin (kenttiin) 
 - joidenkin kenttien arvot kuvattu symboleilla
 - helpompi ihmisten lukea ja kirjoittaa

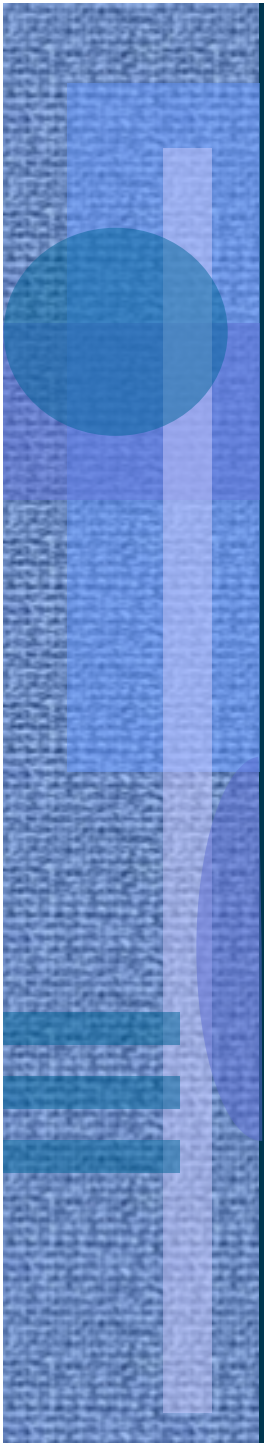
Suorittimen ja muistin sisältö



Teemun juustokakku (5)

Rekisterien, välimuistin, muistin, levymuistin ja magneettinauhan nopeudet suhteutettuna juuston haku aikaan juustokakkuja tehdessä?

<i>käsi</i>	<i>pöytä</i>	<i>jääkaappi</i>	<i>kuu</i>	<i>Europa (Jupiter)</i>
				
0.5 sek (rekisteri)	1 sek (väli- muisti)	10 sek (muisti)	12 pv (levy)	4 v (nauha, ihminen)
			oikea: 10 ms?	



22.3.2010

Copyright Teemu Kerola 2010

9

Luento 2

TTK-91 tietokone ja sen KOKSI simulaattori

Miksi TTK-91?

TTK-91 rakenne ja
käskykanta-arkkitehtuuri

Mikä on simulaattori?

Miten TTK-91 ohjelmia
suoritetaan simulaattorissa?

Tietokone TTK-91

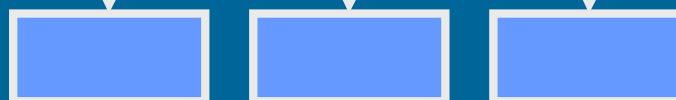
- Laitteisto, hardware (HW)
 - suoritin, muisti, väylät, oheislaitteiden liitännät
- Käskykanta - konekieliarkkitehtuuri
 - käyttöliittymä laitteistoon
 - konekäskyt, tiedon esitysmuodot, tietotyypit
- Symbolinen konekieli
 - luettavampi muoto konekielestä
 - kullakin symbolilla yksikäsitteiset arvot
- Tietokone Simulaattori
 - TTK-91 koneen laitteiston simulaattori
 - symbolisen konekielen kääntäjä
 - graafinen käyttöliittymä, debugger-ympäristö

TTK-91 laitteisto

suoritin - CPU



muisti



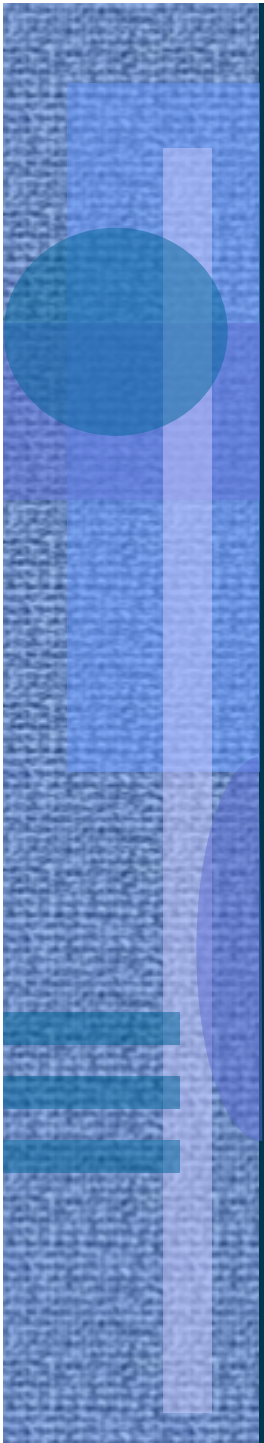
laiteohjaimet

TTK-91 Käskykanta

- Tietotyypit
- Konekäskyjen tyypit
- Konekäskyn rakenne
 - montako bittiä, minkälainen sisäinen rakenne
- Muistissa olevan tiedon osoitustavat
 - konekielessä
 - symbolisessa konekielessä
- Operaatiot

TTK-91 tietotyypit

- 32 bittinen kokonaisluku
 - noin 10-numeroinen desimaaliluku
- EI:
 - liukulukuja
 - merkkejä
 - totuusarvoja
 - ...



22.3.2010

Copyright Teemu Kerola 2010

15

Luento 3

Konekielinen ohjelmointi (TTK-91, KOKSI)



Muuttujat
Tietorakenteet
Kontrolli
Optimointi
Tarkistukset

Tiedon sijainti suoritusaikana

- 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?

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
(heksadesimaali) (desimaali)
- 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. nro 6 tai 18) avulla
 - symbolinen konekieli: R3, FP, F5, I2, jne
- Tieto konekäskyssä (vakiona)
 - oletusarvoisesti, käskyssä on vain yksi paikka tiedolle

Tieto ja sen osoite (2)

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

```
int x =12;
```

symbolin X arvo
muuttujan X osoite?

muuttujan X arvo

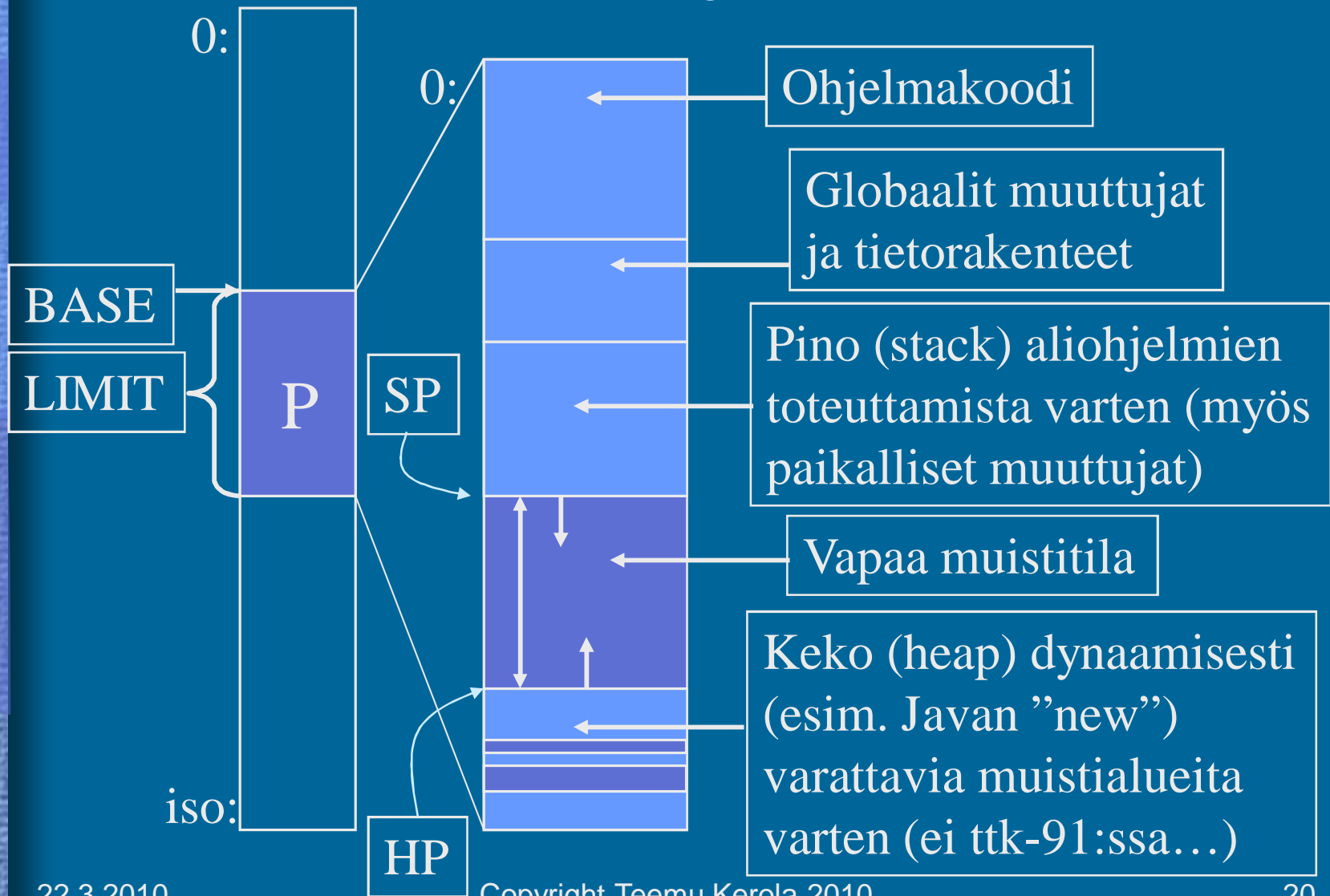
muisti

230
12345
12556
128765
12222
12
12998

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

muuttujan X osoite on symbolin X arvo

Muistitilan käyttö yhdelle ttk-91 ohjelmalle P ⁽⁶⁾



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

Ohjelmoinnin peruskäsitteet

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

For lauseke (4)

```
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?

```
I      DC      0
```

```
...
```

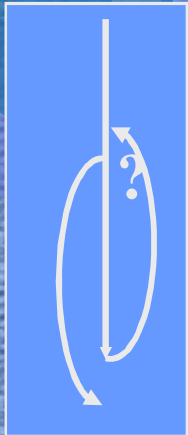
```
LOAD R1, =20  
STORE R1, I
```

```
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
```

While-do -lauseke (2)



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

Mitä kannattaa pitää muistissa?

Mitä kannattaa pitää missä rekisterissä ja milloin?

X in R3?

```
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
```


Koodin generointi ⁽⁷⁾

- 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?
 - joskus R1:ssä, joskus R5:ssä?

Taulukon indeksitarkistus

```
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
```

Moni-ulotteiset taulukot ⁽³⁾

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

T:

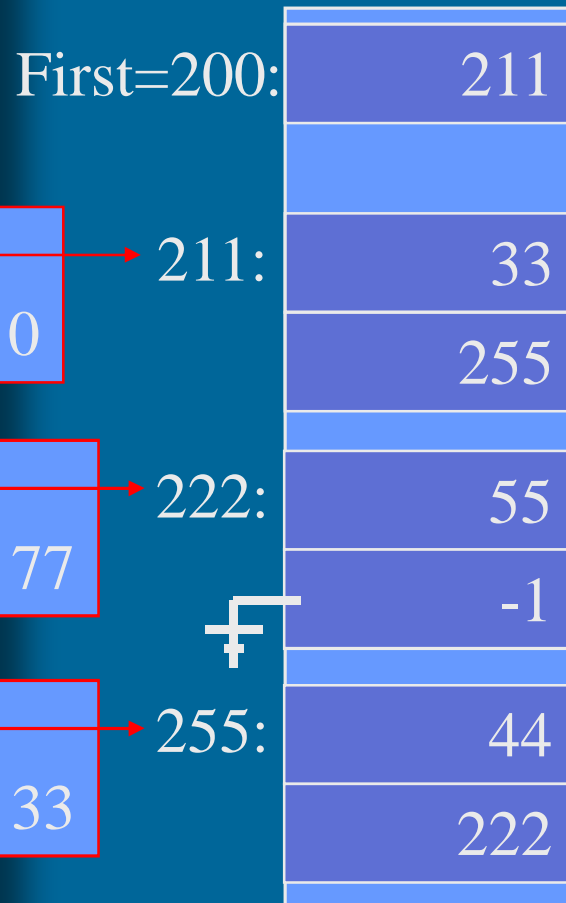
T[0][0]
T[0][1]
T[0][2]
T[1][0]
T[1][1]
T[1][2]
T[2][0]
T[...][...]

T:

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

R1: -1
R2: 132

Linkitetty lista (8)



list_sum.k91

```

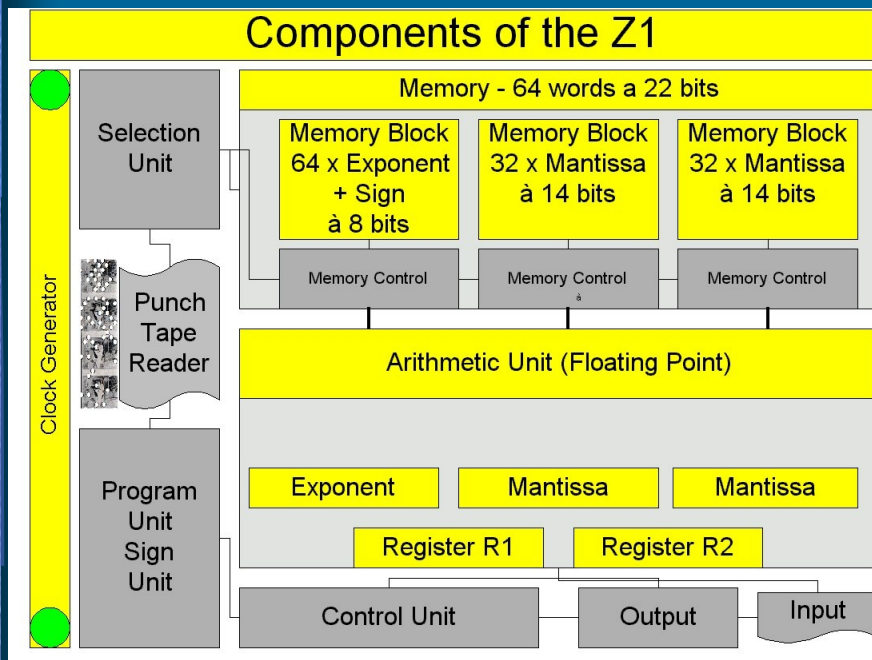
Data EQU 0 ; suht. osoite
Next EQU 1
Sum DC 0
Main LOAD R1, First ; ptrRec
      JNEG R1, Done
      LOAD R2, =0 ; sum
Loop  ADD R2, Data(R1)
      LOAD R1, Next(R1)
      JNNEG R1, Loop
Done  STORE R2, Sum
      SVC SP, =HALT
  
```

Virhe, bugi! Missä?

-- Luennon loppu --

Konrad Zuse: Z1 (1938)

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



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