Luento 2 (verkkoluennot 2-3)

Tietokonejärjestelmän rakenne ttk-91 ja sillä ohjelmointi

Järjestelmän eri tasot

Laitteiston nopeus

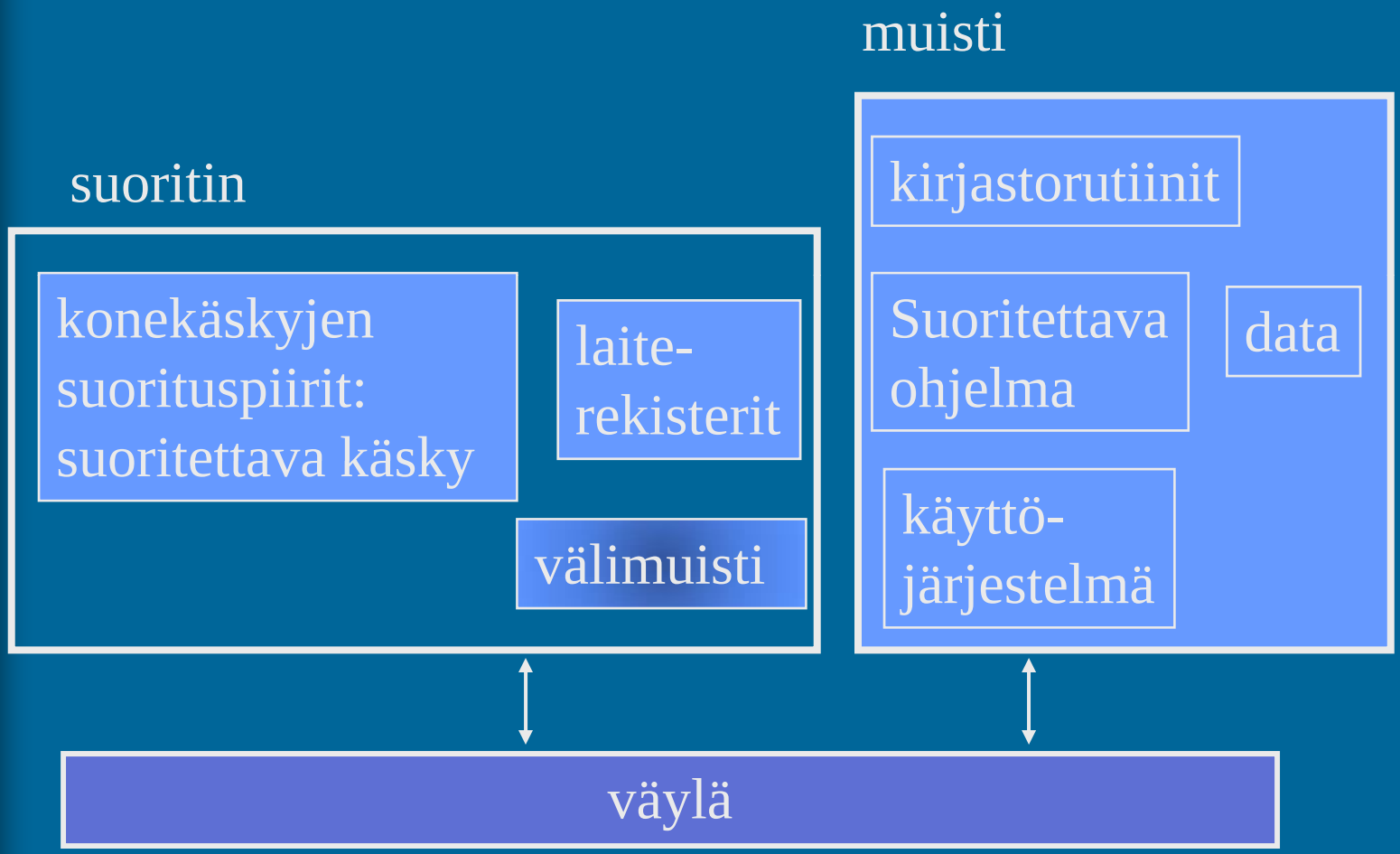
ttk-91 rakenne ja käskykanta-arkkitehtuuri

Konekielinen ohjelmointi

ttk-91:llä

(Titokone, TitoTrainer)

Suorituksenaikainen suorittimen ja muistin sisältö



Ohjelman esitysmuoto: Konekieli

- Suorittimen konekielen käskykanta määrittelee tietokoneen käskykanta-arkkitehtuurin
  - ISA - Instruction Set Architecture
- Ohjelma keskusmuistissa konekielisenä (TTK-91)

	Konekielinen käsky	**lukuna**
0:	0000 0010 000 00 000 0000 0000 0110 0100	DEC: 33554532
1:	0000 0010 001 01 000 0000 0000 0110 0100	DEC: 36175972
2:	0001 0100 001 00 000 0000 0000 0000 0000	DEC: 337641472
3:	0010 0010 000 00 000 0000 0000 0000 0110	HEX: 22000006
4:	0000 0001 001 00 000 0000 0000 1110 0100	HEX: 012000E8
5:	0000 0000 000 00 000 0000 0000 0000 0000	HEX: 00000000

Ohjelman esitysmuoto: symbolinen konekieli

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

Symb. konekieli	Konekielinen käsky
LOAD R2, =100	0000 0010 000 00 010 0000 0000 0110 0100
LOAD R1, 100	0000 0010 001 01 000 0000 0000 0110 0100
DIV R1, R2	0001 0100 001 00 010 0000 0000 0000 0000
JZER 6	0010 0010 000 00 000 0000 0000 0000 0110
STORE R1, 228	0000 0001 001 00 000 0000 0000 1110 0100
NOP	0000 0000 000 00 000 0000 0000 0000 0000

Ohjelman esitysmuotoja

osoite sisältö

käskyt

data

Address	Contents			
101	0010	0010	0000	0001
102	0001	0010	0000	0010
103	0001	0010	0000	0011
104	0011	0010	0000	0100
201	0000	0000	0000	0010
202	0000	0000	0000	0011
203	0000	0000	0000	0100
204	0000	0000	0000	0000

(a) Binary program

Address	Contents
101	2201
102	1202
103	1203
104	3204
201	0002
202	0003
203	0004
204	0000

(b) Hexadecimal program

Address	Instruction	
101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
201	DAT	2
202	DAT	3
203	DAT	4
204	DAT	0

(c) Symbolic program

osoite

arvo

Label	Operation	Operand
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

(d) Assembly program

Symboli (nimi)

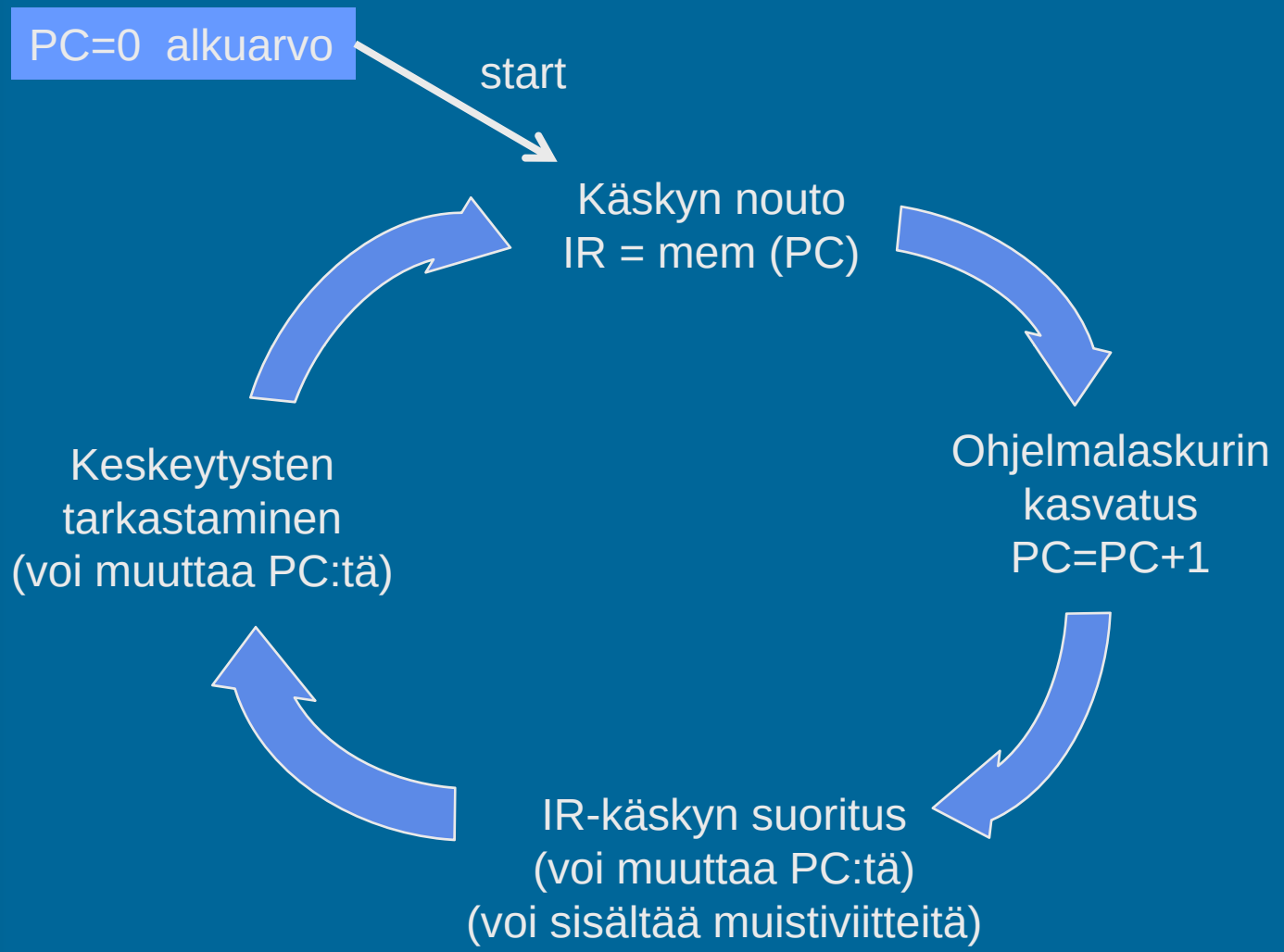
(Sta06 Fig 10.11)

TTK-91 konekieli

Operaatio 8 bit	Rj 3 b	M 2 b	Ri 3 b	Attribuutti (arvo tai osoite) 16 bittiä

- Kukin käsky 32 bittiä. Oma operaatiokoodi kullakin käskyllä
- Käskyn rekisterien ja attribuuttien tulkinta riippuu käskystä ja osoitusmuodosta (M)
- Tietotyyppi:
  - Vain 32-bittinen kokonaisluku
  - EI: merkkejä, liukulukuja, totuusarvoja

Suorittimen toiminta



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 + ...

Milloin muuttujan X arvo pidetään muistissa ja milloin rekisterissä?
Missä päin muistia? Miten siihen viitataan?

Tiedon sijainti suoritusaikana

- 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

Miten tietoon viitataan?

- 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, PS, SR, I2, jne
- Tieto konekäskyssä (vakiona)
  - oletusarvoisesti käskyssä on vain yksi paikka tiedolle

Tieto ja sen osoite

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

int x =12;

muut-
tujan
X
osoite
on
sym-
bolin
X
arvo

symbolin X arvo
muuttujan X osoite?

muuttujan X arvo

muisti

230
12345
12556
128765
12222
12
12998

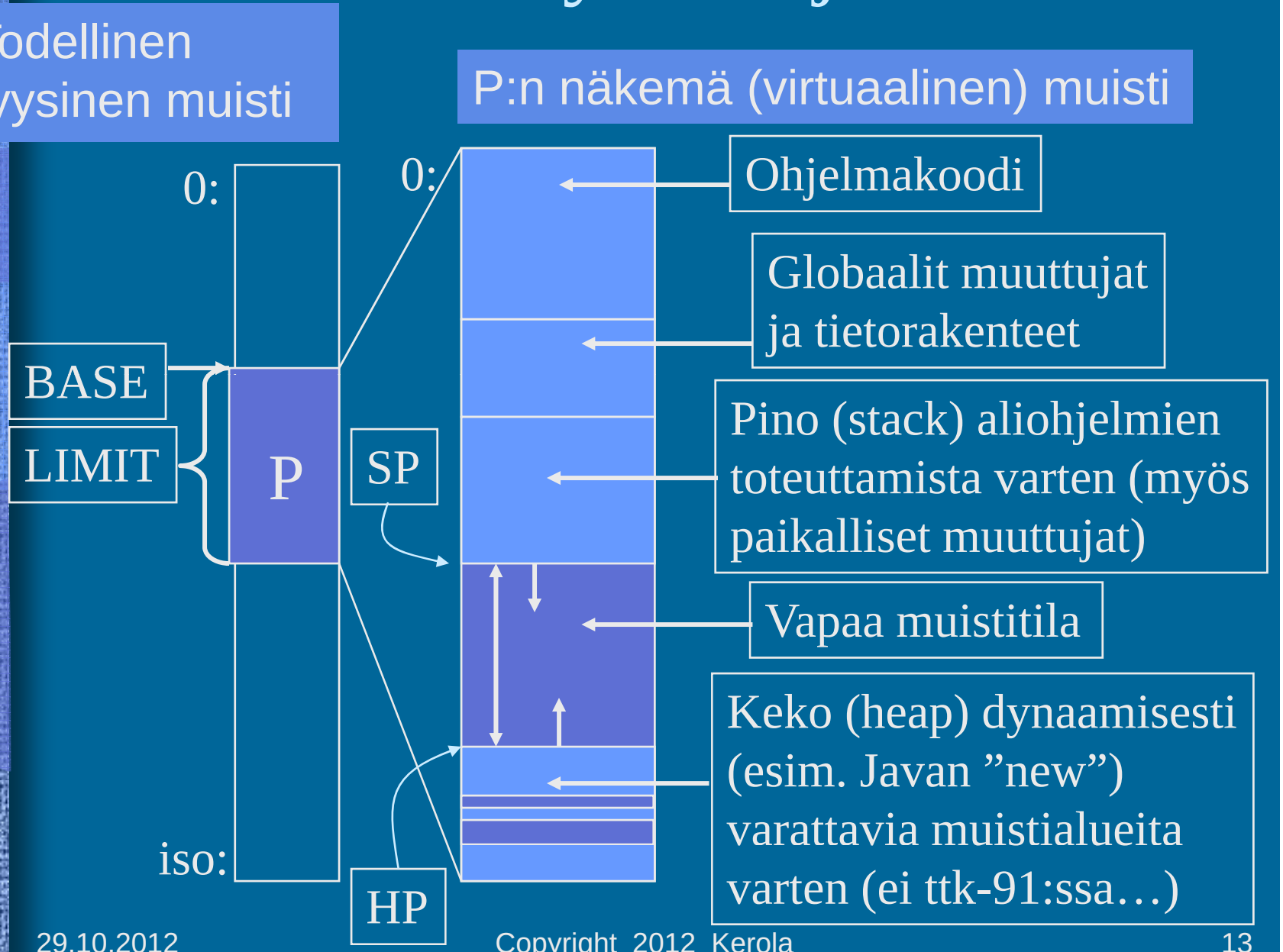
X=230:

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

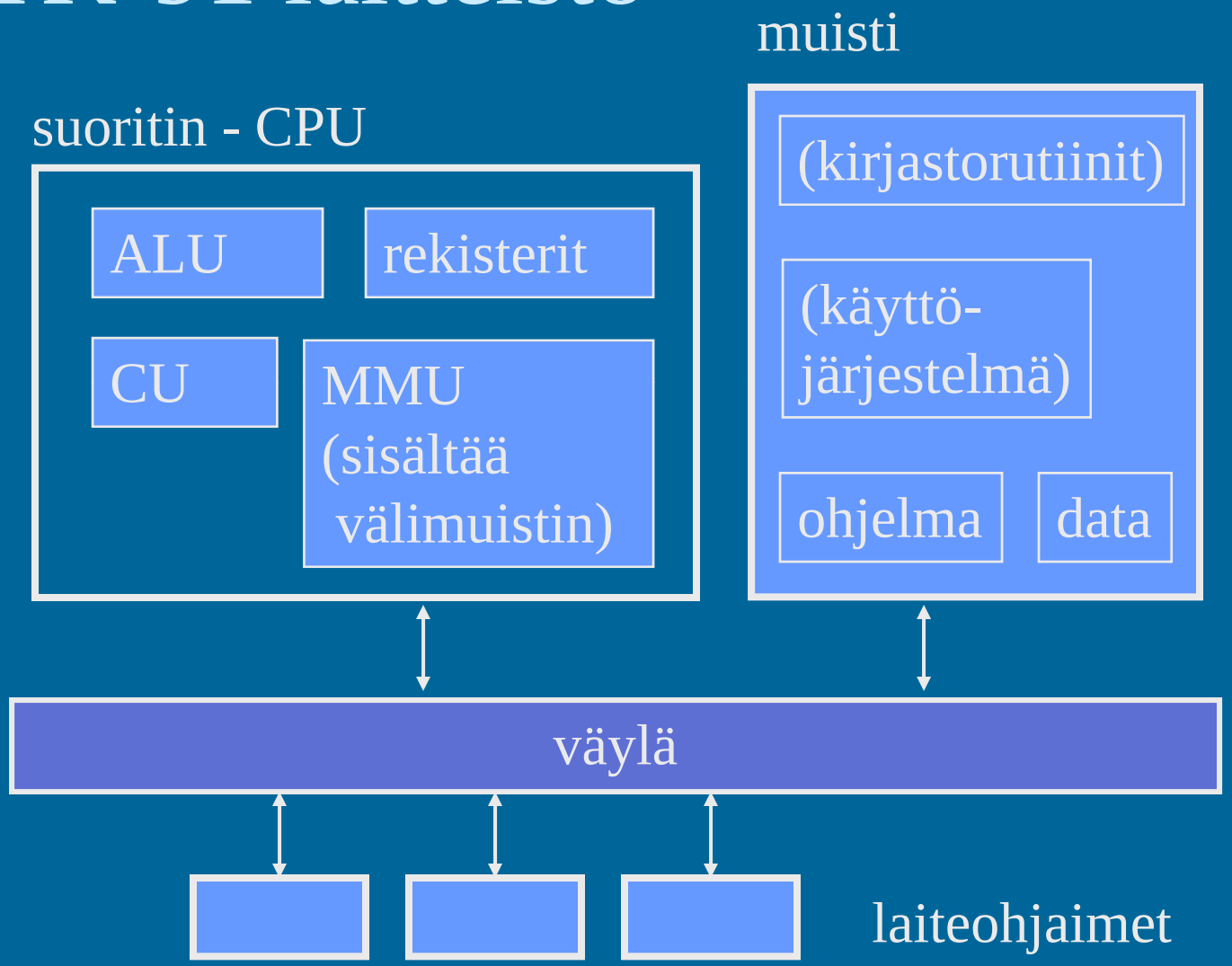
Tiedon osoitusmuodot TTK-91

- Välitön operandi (ei muistiosoitusta)
  - OPER Rj, =ADDR(Ri) M=00
  - Kumpi vain voi puuttua (ADDR, Ri)
- Suora indeksoitu muistiosoitus
  - OPER Rj, ADDR (Ri) M=01
- Epäsuora indeksoitu muistiosoitus
  - OPER Rj, @ADDR(Ri) HUOM: ei R0
    M=10

Muistitilan käyttö ohjelmalle P



TTK-91 laitteisto



Ttk-91 ja Ohjelmoinnin peruskäsitteet

Aritmetiikka

Tietorakenteet

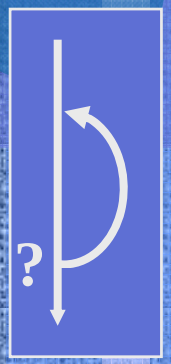
Kontrolli, loopit

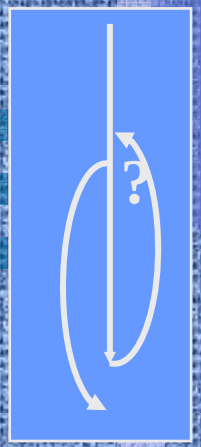
Monimutkaiset tietorakenteet

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, virhetilanteet
- Monimutkaiset tietorakenteet
  - listat, moniulotteiset taulukot

For lauseke






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

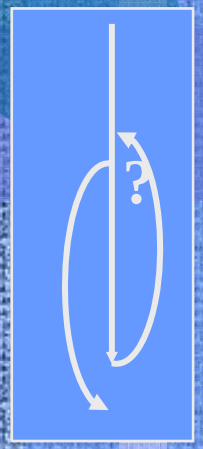
```
I       DC      0
        …
        LOAD  R1, =20
        STORE R1, I

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?

Entä jos toisenlainen toisto-semantiikka?

While-do -lauseke



```
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?

Keskustele

Koodin generointi

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

Taulukon indeksitarkistus (kesk.)

```
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
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
```

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

Moniulotteiset taulukot (3)

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

T:

34	**57**	**76**
21	76	23
24	56	876
54	75	777

T:

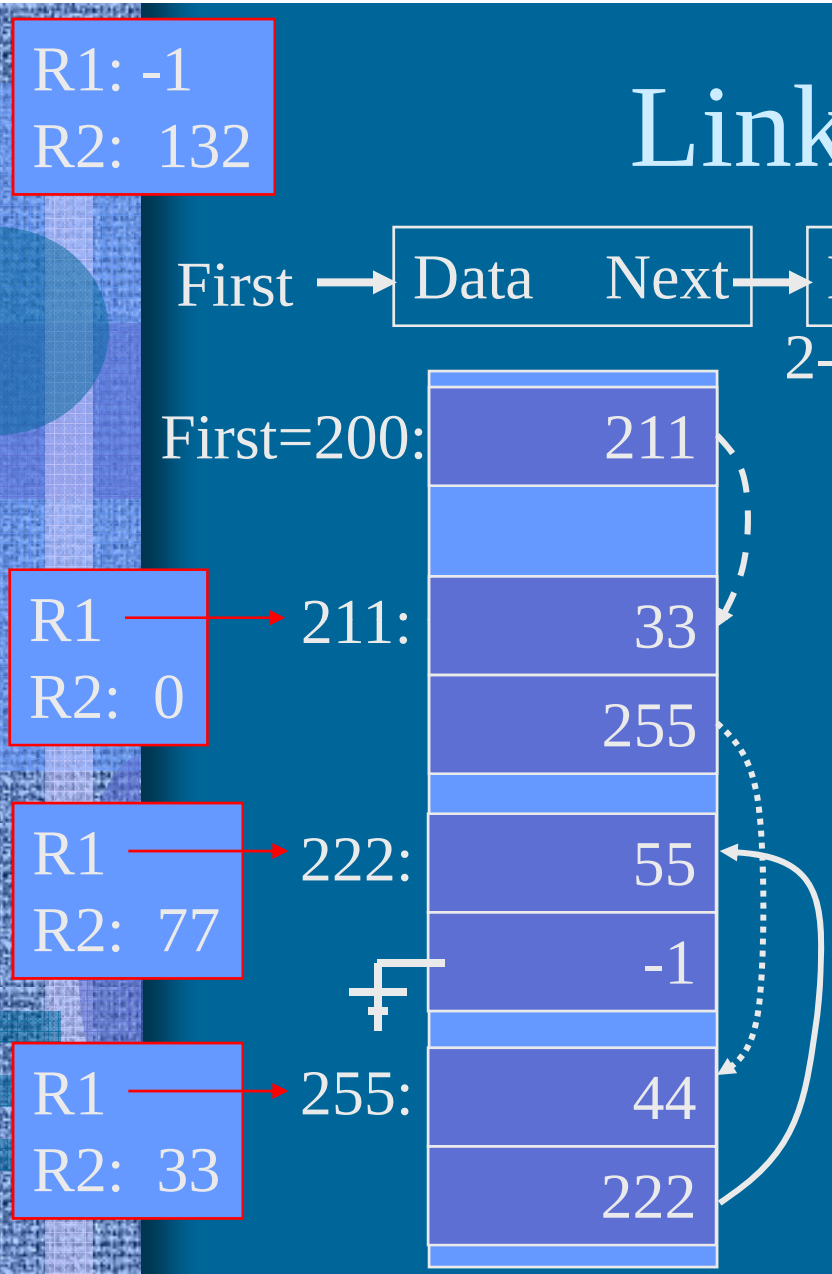
T[0][0]=34
T[0][1]=57
T[0][2]=76
T[1][0]
T[1][1]
T[1][2]
T[2][0]
T[...][...]

T:

T[0][0]=34
T[1][0]=21
T[2][0]=24
T[3][0]=54
T[0][1]
T[1][1]
T[2][1]
T[...][...]

Linkitetty lista

2-sanainen tietue



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?

-- Loppu --

- Elektroniputki
  - logiikka, muisti
- ENIAC, 1945
  - Electronic Numerical Integrator and Computer
  - J.W. Mauchly, J.P. Eckert, J. von Neumann
  - 17 468 elektr. Putkea
  - 5 000 yht.laskua/sek.
  - 357 kertolaskua/sek