Luento 3 (verkkoluento 3)

Ttk-91 konekielinen ohjelmointi

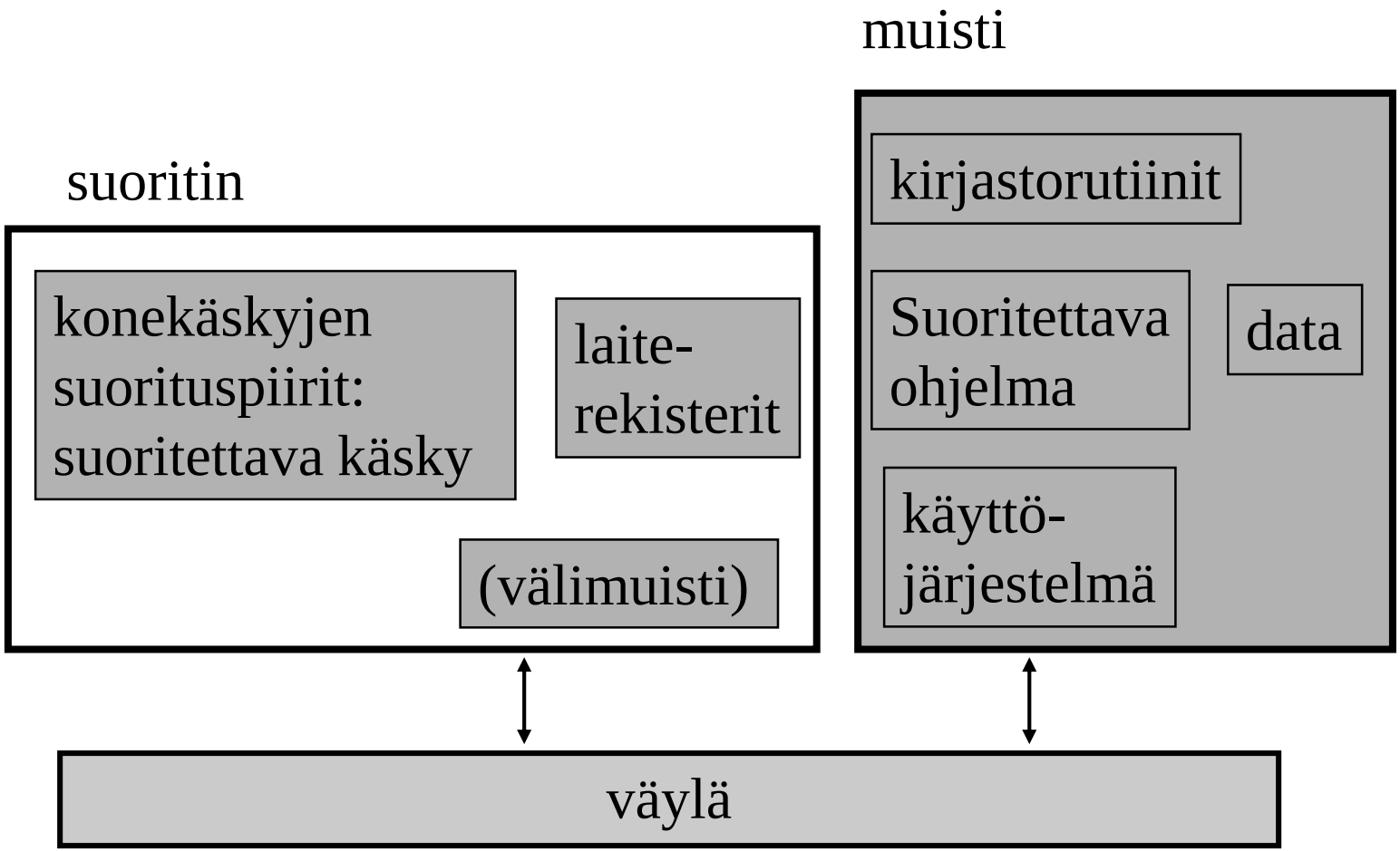
Ohjelman esitysmuoto

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: 012000E4
5:	0000 0000 000 00 000 0000 0000 0000 0000	HEX: 00000000
	- opcode - Rj M Ri - ADDR: arvo, osoite -	

Ohjelman esitysmuoto: symbolinen konekieli

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

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

Keskustele

Ohjelman esitysmuotoja

osoite sisältö **N = I+J+K**

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

osoite

arvo

(c) Symbolic program

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

Symboli (nimi)

(d) Assembly program

(Sta16 Fig 13.14)

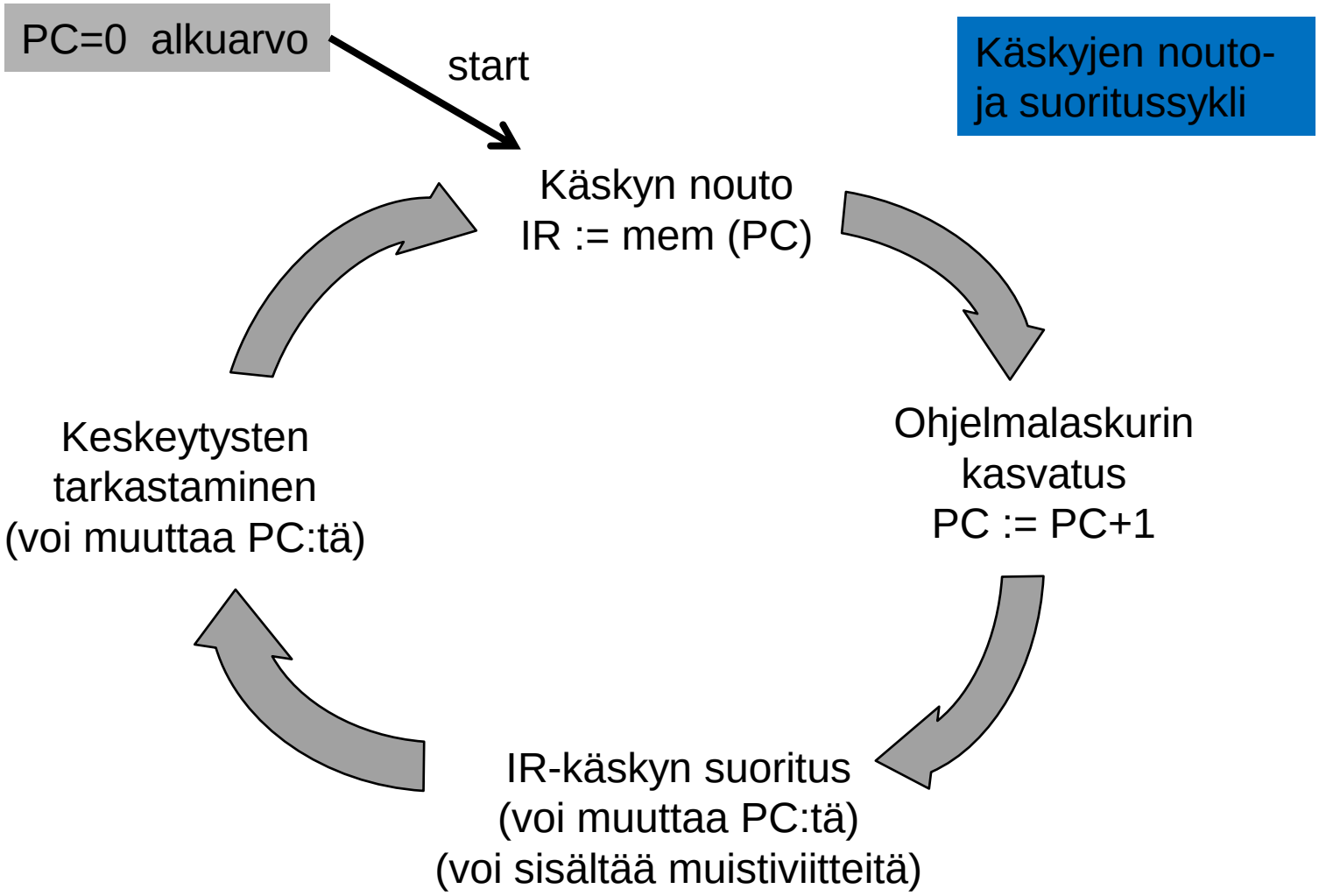
TTK-91 konekieli

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 (tai vain bittejä)
  - EI: merkkejä, liukulukuja, totuusarvoja

Keskustele

Suorittimen toiminta



Keskustele

Tiedon sijainti suoritusaikana

- Rekisteri (nopein)
  - kääntäjä (yleensä) päättää, milloin muuttujan arvo on rekisterissä
- Välimuisti (nopea)
  - laitteisto hoitaa automaattisesti viimeksi käytetyille muistialueille
- Muisti (hidas)
  - kääntäjä/lataaja valitsee sijaintipaikan
    - globaali data ohj. latauksen yhteydessä (data-alue)
    - vakiot konekäskyssä (koodi-alue)
  - ohjelma sijoittaa suoritusaikana
    - aliohjelman paikalliset muuttujat ja parametrit pinossa
  - käyttöjärjestelmä sijoittaa suoritusaikana
    - dynaaminen data keossa suorituksen aikana
- Levy, levypalvelin (liian hidas, ei mahdollista)
  - Ladattava muistiin ennen käyttöä

Miten tietoon viitataan?

- Tieto muistissa
  - Muistiosoitteen (esim. 0x6F123456 tai 3459321) avulla
  - Symbolin (esim. HenkNro tai X) avulla symbolista konekieltä käytettäessä ; symbolin arvo on muistiosoite tai tietueen kentän sijainti
    - HenkNro ≡ 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!
  - Nopeus jossain rekisterin ja muistin nopeuksien välillä
  - Välimuisteja voi olla usea: L1 (pieni, nopein), L2 (isompi), L3 (isoin)
- Tieto rekisterissä
  - Rekisterin osoitteen (esim. nro 6 tai 18) avulla (ttk-91: 0-7)
  - Symbolinen konekieli: R3, FP, F5, PS, SR, I2, jne (ttk-91: R0-R7, SP, FP)
- Tieto konekäskyssä vakiona (suoritushetkellä rekisterissä IR)
  - oletusarvoisesti käskyssä on vain yksi paikka tiedolle (vakio-kenttä)

Tieto ja sen osoite

muut-
tujan
X
osoite
on
sym-
bolin
X
arvo

```
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

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

symb.taulu

…
X 230
…

Ttk-91 ja Ohjelmoinnin peruskäsitteet

Aritmetiikka

Tietorakenteet

Kontrolli, loopit

Monimutkaiset tietorakenteet

Ohjelmoinnin peruskäsitteet

- Aritmeettinen lauseke
  - miten tehdä laskutoimitukset?
- Yksinkertaiset tietorakenteet
  - tiedonosoitusmoodi tukee suoraan
  - yksiulotteiset taulukot, tietueet
- Kontrolli – mistä seuraava käsky?
  - valinta: if-then-else, case
  - toisto: for-silmukka, while-silmukka
  - Aliohjelmat (luento 4), virhetilanteet (keskeytykset)
- Monimutkaiset tietorakenteet
  - listat, moniulotteiset taulukot

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
```

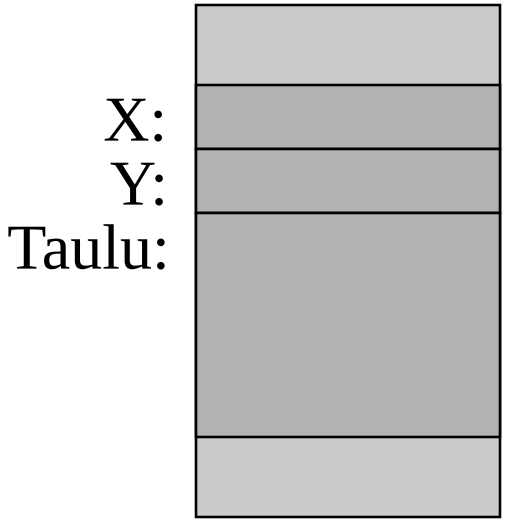
tai:

```
LOAD R1,=5
MUL  R1,C
ADD  R1, B
STORE R1,A
```

Globaalin (kaikkialla näkyvän) 1-ulotteisen 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

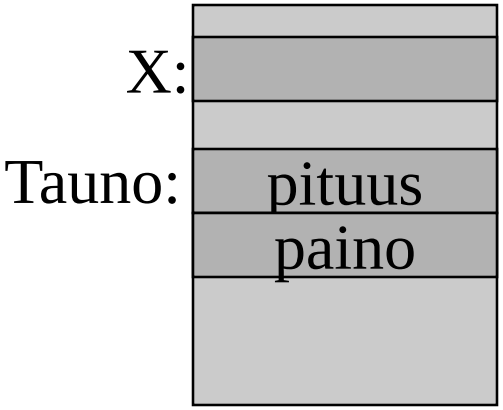
Globaalin tietueen tilan varaus ja käyttö (3)

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

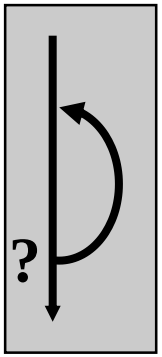
Kentän ”Paino” suhteellinen osoite tietueen Tauno sisällä

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

Tietueen osoite on sen ensimmäisen sanan osoite

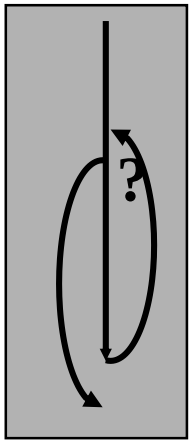


For lauseke




```
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 toisto-semantiikka?

```
T       DS      50
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
```

alku

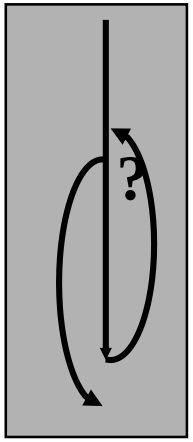
runko

lisää

testaa

Keskustele

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

Taulukon indeksitarkistus

```
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

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]=21
T[1][1]
T[1][2]
T[2][0]
T[...][...]

- Talletus riveittäin
  - C, Pascal, Java?
- Talletus sarakkeittain
  - Fortran

T:

T[0][0]=34
T[1][0]=21
T[2][0]=24
T[3][0]=54
T[0][1]=57
T[1][1]
T[2][1]
T[...][...]

- (On <u>muitakin tapoja</u>, esim.
  - Kukin rivi allokoitu erikseen
  - T[i] on rivin i osoite, jne …)
- 3- tai useampi ulotteiset
  - vastaavalla tavalla!

Monimutkaisten tietorakenteiden viittaukset

```
int T[10, 20];
```

```
T     ds  200
Trows equ 10
Tcols  equ 20
```

```
T[i,j] = 34;
```

- Laske ensin viitatun alkion osoite (tietorakenteen sisällä)
- Tee viittaus indeksointia käyttäen

riveittäin

```
Load r1, i
Mul   r1,  =Tcols
Add   r1, j

Load  r2,=34
Store r2, T(r1)
```

sarakettain

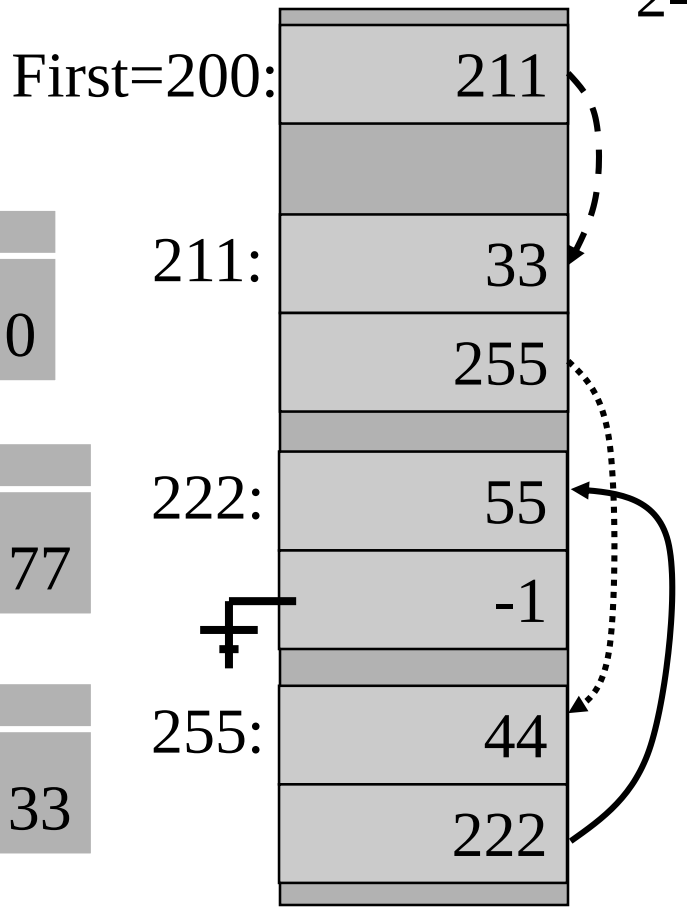
```
Load r1, j
Mul   r1,  =Trows
Add  r1, i

Load  r2,=34
Store r2, T(r1)
```


Keskustele

R1: -1
R2: 132

Linkitetty lista

2-sanainen tietue



R1
R2: 0

R1
R2: 77

R1
R2: 33

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?

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?

-- 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