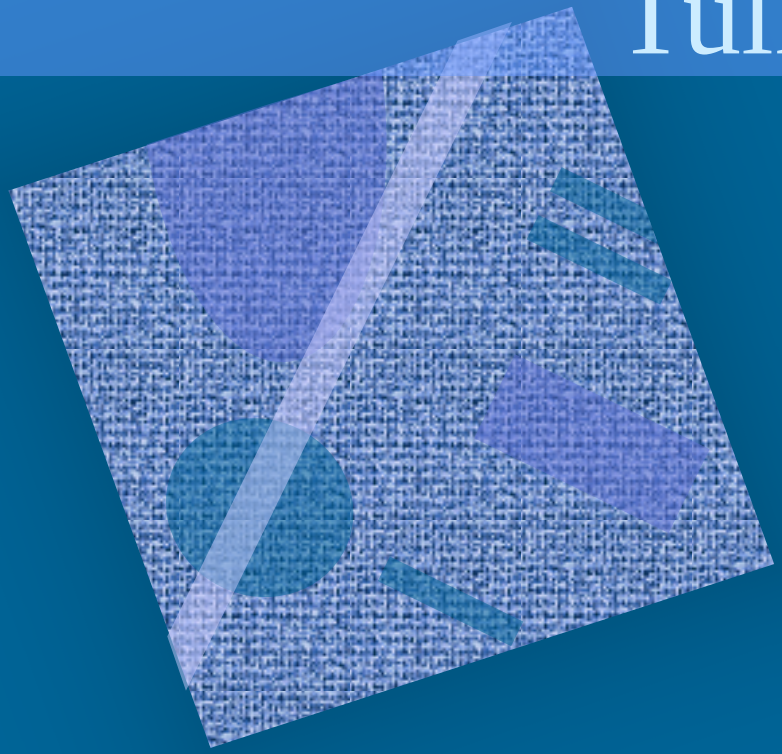
Luvut 9 ja 10
Kertausluento 5

Käännös, linkitys ja lataus Tulkinta ja emulointi



Kääntämisen vaiheet

Staatt. ja dyn. linkitys

Prosessin luonti

JVM

Javan suoritustavat

Suoritus emuloimalla

Copyright 2019 Teemu Kerola

Lausekielisestä ohjelmasta prosessiksi

Käännös lausekielestä

Käännösyksikkö
Lausekielinen ohjelma tai moduuli
Viittaukset: symboleilla

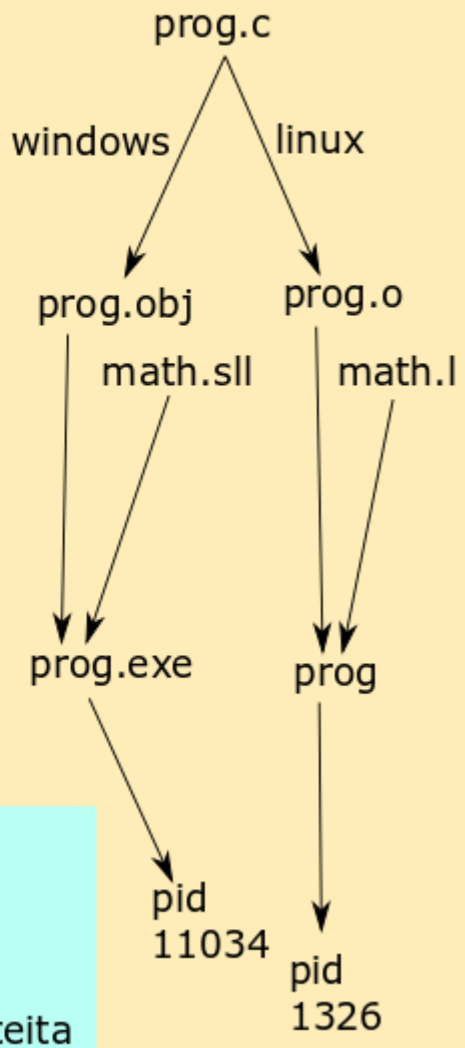
Linkitys muihin moduuleihin

Objektimoduuli
Konekielinen ohjelman osa
Viittaukset: moduulin omat
muistiosoitteet (0 - N)

Lataus muistiin

Ajomoduuli
Konekielinen kokonainen ohjelma
Viittaukset: ohjelman omat
muistiosoitteet (0 - iso-N),
osa voi puuttua!

Prosessi
Suorituskelponen kokonainen ohjelma
Viittaukset: ohjelman omat
muistiosoitteet (0 - iso-N),
voivat olla virtuaalisia osoitteita



Keskustele

Copyright 2019 Teemu Kerola

Assembler käännös

- 1. vaihe (koodin läpikäynti)
  - laske käskyjen tilanvaraukset
  - generoi symbolitaulu, muut taulut
- 2. vaihe (koodin läpikäynti)
  - generoi lopullinen objektimoduuli
  - tulosta symbolinen konekielinen listaus
  - generoi taulut linkitystä varten
  - anna virheilmoitukset
- 3. vaihe
  - koodin generointi ja optimointi
  - voi olla oikeasti ennen 2. vaihetta tai sen yhteydessä

Copyright 2019 Teemu Kerola

Korkean tason kielen käännös

- Enemmän vaiheita
  - Syntaktisten alkioiden etsintä
    - Syntaksipuun generointi ja jäsennys
  - Lauseiden tunnistaminen syntaksipuun avulla
  - Välikielen (välikoodin) generointi (ei aina)
  - Koodin optimointi
  - Koodin generointi
    - ei (yleensä) Java-ohjelmille

BEGIN **123.45** **IF** **(**

(front end)

P-code, bytecode, …

Välikieliesitys ja symbolitaulut

(back end)

Lisää tietoa?  **Kääntäjien ja ohj. kielten kurssit**

Copyright 2019 Teemu Kerola

GameX moduulit ennen linkitystä

GameX

```
    0: int x
...    ...
  234: store r2, x
..     ...
3333: call  Stats.Report (...)
...    ...

8000:
```

Uudelleensijoitustaulu (GameX)

SYMBOL	arvo	viitteet
x:	0	234, ...

EXPORT
(tyhjä)

IMPORT

Stats.Report:	?	3333, ...

...

Stats

```
    0:  int a
...     ...
  302:  load r1, a
...     ...
  800:  Report (...)
...     {   ...
  840:      call  Math.Aver (...)
...       ...
  850:      store r4, a
...       ...
          }
          ...

6000:
```

Uudelleensijoitustaulu (Stats)

SYMBOL	arvo	viitteet
a:	0	302, 850, ...

EXPORT

Report:	800	...

...

IMPORT

Math.Aver:	?	840, ...

...

Math

```
    0:  int sum
...     ...
    5:  load r3, sum
...     ...
2400:   Aver (...)
...     {
            ...
2450:       store r5, sum
...         ...
          }
          ...

5000:
```

Uudelleensijoitustaulu (Math)

SYMBOL	arvo	viitteet
sum:	0	5, 2450, ...

...

EXPORT

Aver:	2400	...

...

IMPORT
(tyhjä)

Uudelleensijoitusvakiot

GameX: 0
Stats: 8000
Math: 14000

Copyright 2019 Teemu Kerola

GameX moduulit ennen linkitystä

GameX

```
    0: int x
...    ...
  234: store r2, x
..     ...
3333: call  Stats.Report (...)
...    ...

8000:
```

Uudelleensijoitustaulu (GameX)

```
SYMBOL    arvo   viitteet
x:           0     234, ...

EXPORT
(tyhjä)

IMPORT
Stats.Report:  ?    3333, ...
...
```

Stats

```
    0: int a
...    ...
  302: load r1, a
...    ...
  800: Report (...)
...    {   ...
  840:     call  Math.Aver (...)
...        ...
  850:     store r4, a
...        ...
         }
         ...

6000:
```

Uudelleensijoitustaulu (Stats)

```
SYMBOL    arvo   viitteet
a:           0    302, 850, ...

EXPORT
Report:     800    ...
...

IMPORT
Math.Aver:     ?   840, ...
...
```

Math

```
    0:  int sum
...     ...
    5:  load r3, sum
...     ...
2400:   Aver (...)
...     {
            ...
2450:       store r5, sum
...         ...
        }
        ...

5000:
```

Uudelleensijoitustaulu (Math)

```
SYMBOL    arvo   viitteet
sum:         0     5, 2450, ...
...

EXPORT
Aver:     2400      ...
...

IMPORT
(tyhjä)
```

Staattisesti linkitetty GameX latausmoduuli

GameX

```
    0: int x
...    ...
  234: store r2, 0
..     ...
3333: call  8800 (...)
...    ...
```

Stats

```
 8000: int a
...    ...
8302:  load r1, 8000
...    ...
8800:  Report (...)
...    {   ...
8840:      call  16400 (...)
...        ...
8850:      store r4, 8000
...        ...
           }
           ...
```

Math

```
14000: int sum
...    ...
14005: load r3, 14000
...    ...
16400: Aver (...)
...    {
           ...
16450:     store r5, 14000
...        ...
           }
           ...

19000:
```

Uudelleensijoitusvakiot

```
GameX:      0
Stats:   8000
Math:   14000
```

Uudelleensijoitustaulu (kaikki)

```
SYMBOL    arvo   viitteet
x:           0     234, ...
a:        8000    8302, 8850, ...
sum:     14000   14005, ...
...

EXPORT
Stats.Report:   8800
Math.Aver:     16400
..

IMPORT
(tyhjä)
```


Copyright 2019 Teemu Kerola

Staattinen ja dynaaminen linkitys

- Staattinen linkitys
  - Kaikki ohjelmakoodissa viitatut moduulit ja kirjastorutiinit on linkitetty ennen suoritusta
  - Iso ajomoduuli
    - mukana moduuleja, joihin ei yhdellä suorituskerralla tule lainkaan viittauksia
- Dynaaminen linkitys
  - Kutsukohdat muihin moduuleihin jätetään auki
  - Pienempi ajomoduuli, mutta hitaampi suorittaa
  - Viittaus ”ratkaisemattomaan” (eli ei-linkitettyyn) moduuliin ratkotaan suoritusaikana
    - suoritus keskeytyy ja puuttuva moduuli linkitetään paikalleen (kaikki viittaukset siihen korjataan kuntoon)

Copyright 2019 Teemu Kerola

Dynaamisesti linkitetty GameX latausmoduuli

```
     0: int x                    GameX
 ...    ...
   234: store r2, 0
 ..     ...
  3333: call  8800 (...)
 ...    ...

 8000: int a                     Stats
...     ...
 8302: load r1, 8000
...     ...
 8800: Report (...)
...     {   ...
 8840:    call  -23 (...)
...       ...
 8850:    store r4, 8000
...       ...
         }
         ...

14000:
```

Uudelleensijoitusvakiot

```
GameX:      0
Stats:   8000
```

Uudelleensijoitustaulu (kaikki)

```
SYMBOL          arvo   viitteet
x:                 0    234, ...
a:              8000    8302, 8850, ...
...

EXPORT
Stats.Report:   8800
..

IMPORT
Math.Aver:        -23    8840  (dyn. link.)
```

Keskustele

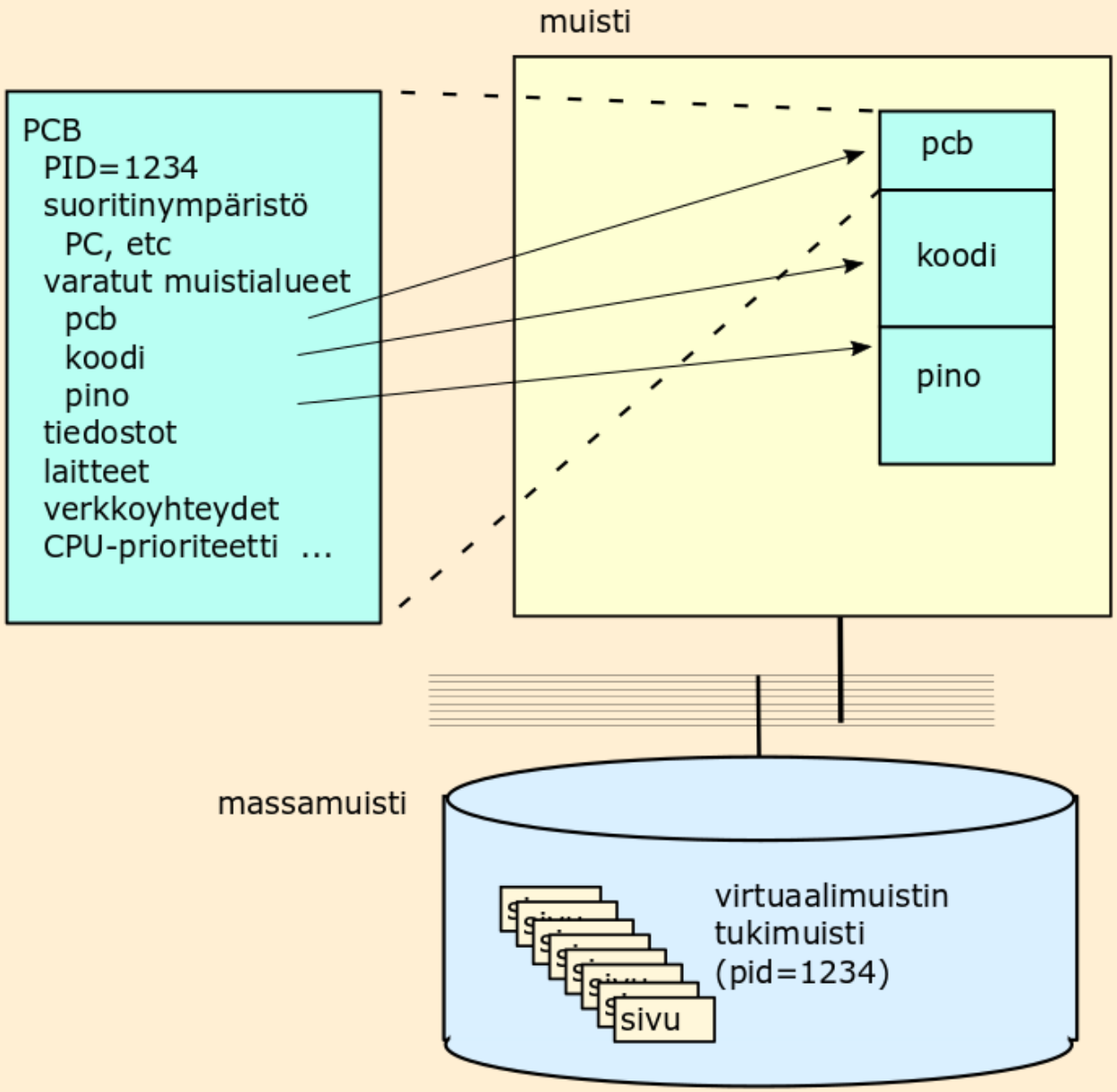
Copyright 2019 Teemu Kerola

Lataus

- Ajomoduulista luodaan suorituskelpoinen prosessi (rakennetaan PCB ja sen viitteet kuntoon)
- Prosessin koodi- ja data-alueet ladataan muistiin, prosessi siirretään jonoon Valmis suoritukseen (Ready, Ready-to-Run)
- Eri tyyppejä
  - Absoluuttinen – aina samaan paikkaan muistia
  - Uudelleensijoitettava – joustava sijainnin valinta muistissa
    - Milloin ja miten osoitteet muutetaan?
  - Dynaaminen sijoitus ajoaikaina
    - Milloin ja miten osoitteet muutetaan?

Copyright 2019 Teemu Kerola

Prosessi ja sen kuvaaja (PCP)



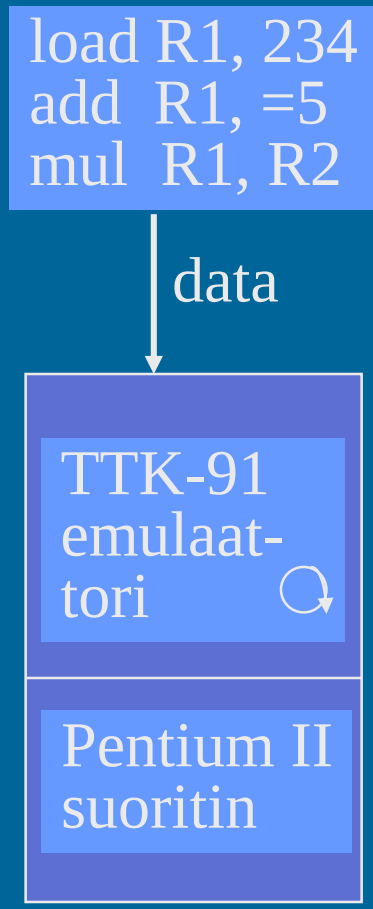
Suoritus tulkitsemalla tai emuloinalla

- Tulkittavat skriptikielet
  - Bourne shell, C-shell, Cmd, …
- Emulaattori
  - Jäljittelee funktionaalisesti jonkun muun järjestelmän toimintaa halutulla tasolla
  - Ttk-91, Intelin tai Transmetan x86-emulaattorit, …
- Simulaattori
  - Jäljittelee jonkun muun järjestelmän tiettyjä toimintoja halutulla tasolla
  - Lentosimulaattorit, analysaattorit, …

Copyright 2019 Teemu Kerola

TTK-91 Emulointi

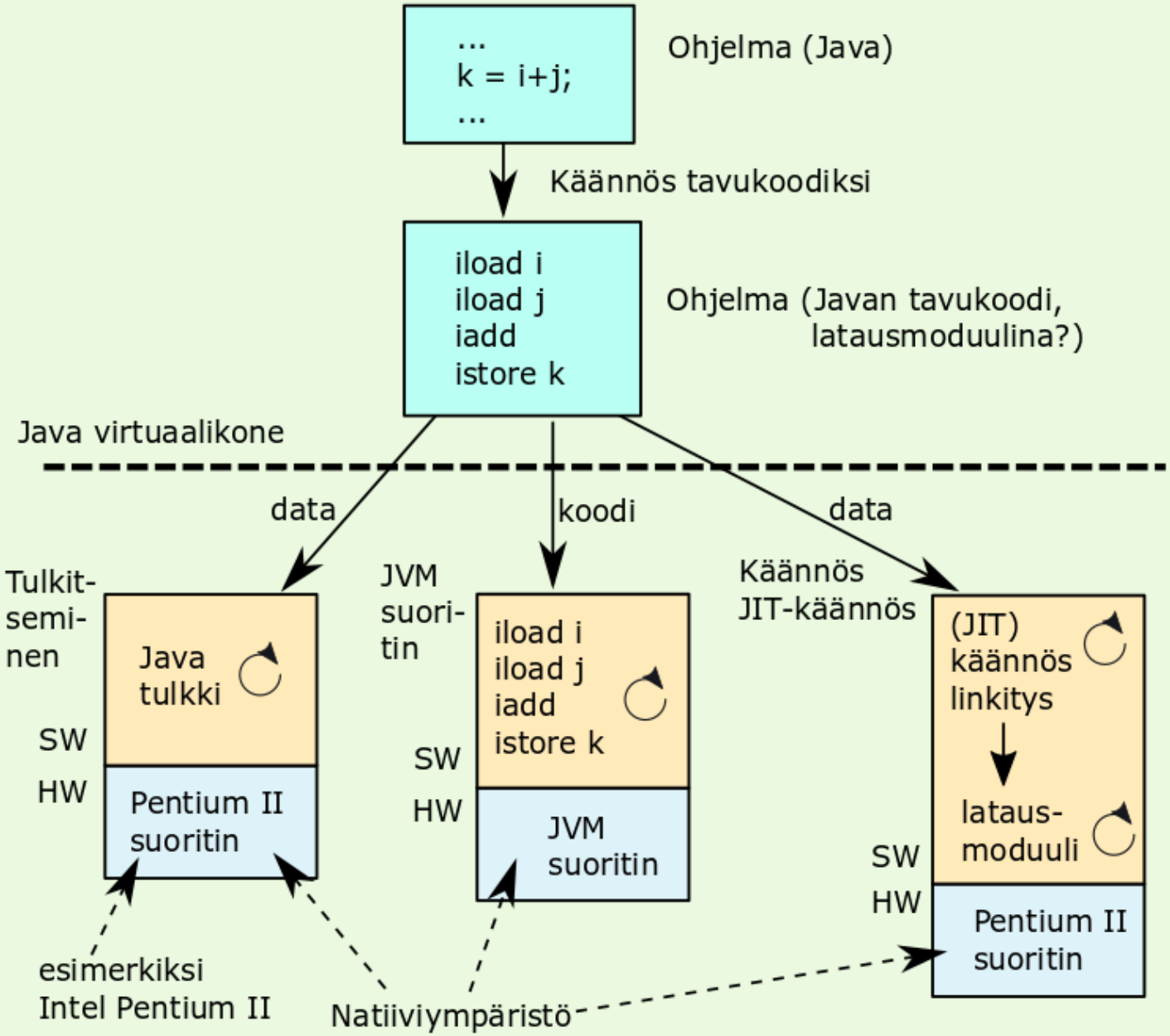
- TTK-91 konekielen emulointi
- Titokoneen komponentti
- Yksi käsky kerrallaan
- TTK-91 koneen rekisterit ja muisti emuloitu tulkin (Titokone) tietorakenteina



ks. simulaattorin koodi, proj. Titokone:
http://www.cs.helsinki.fi/group/nodes/kurssit/tito/2012s/Interpreter.java
http://www.cs.helsinki.fi/group/nodes/kurssit/tito/2012s/Processor.java

Copyright 2019 Teemu Kerola

Java-ohjelmien suoritustavat

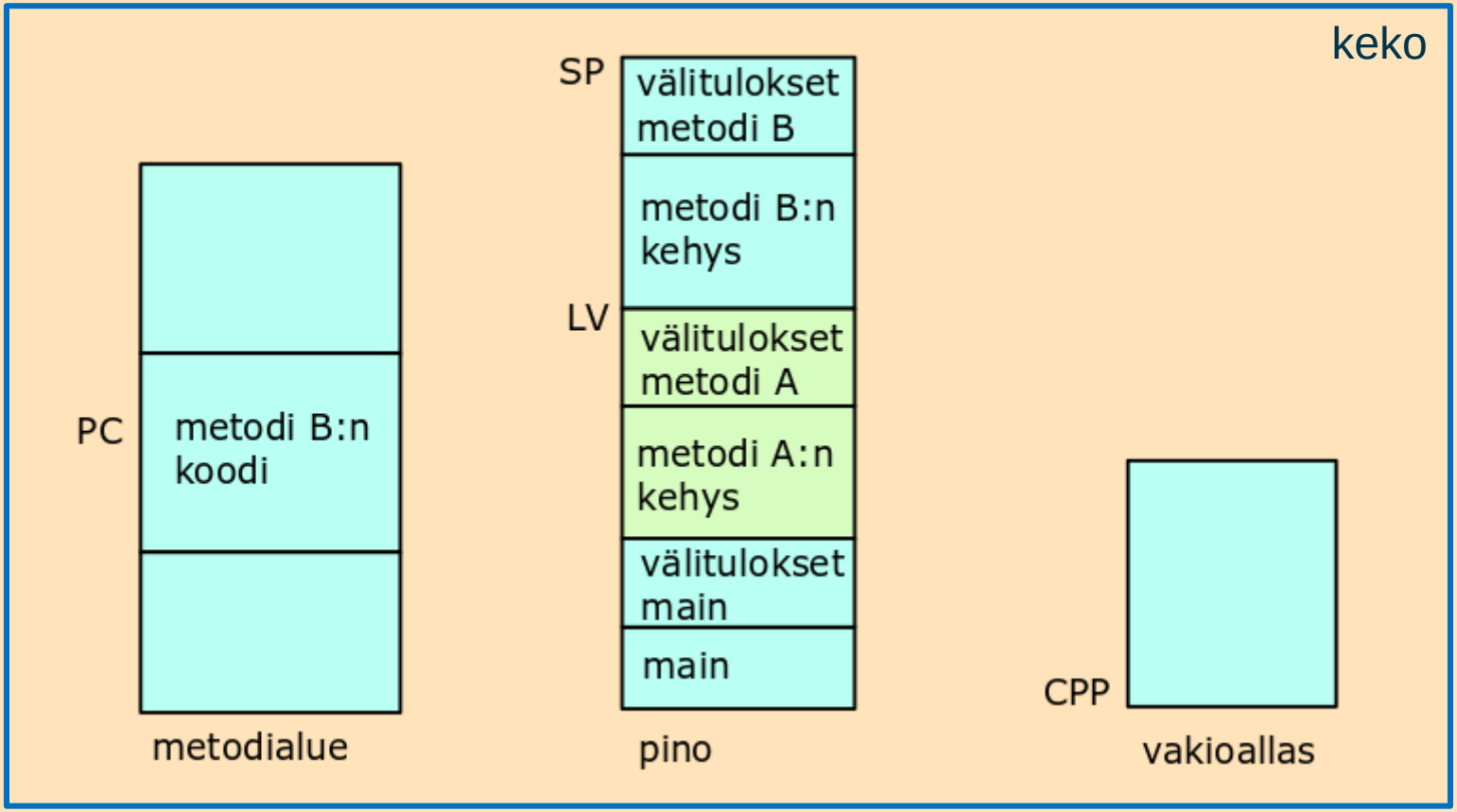


Java virtuaalikone (JVM)

- Hypoteettinen suoritin, toteutus eri tavoilla
- Geneerinen. Sitä on ”helppo” simuloida kaikilla todellisilla suorittimilla
  - Käännökseen tai tulkitsemiseen perustuva suoritus
- Useita säikeitä (thread) voi olla samanaikaisesti suorituksessa
  - vuorotellen tai eri ytimillä todella samanaikaisesti
- Tietorakenteet
  - Mm. virtuaalikoneen suorittimen ”rekisterit”
  - Luodaan JVM:n käynnistämisen yhteydessä
- Käskyt
  - Virtuaalikoneen (JVM) suorittimen konekäskyt
  - 226 käskyä

Copyright 2019 Teemu Kerola

JVM:n muistialueet ja rekisterit



- Kaikki rekisterit implisiittiä
  - Niitä ei ole nimetty konekäskyissä

Copyright 2019 Teemu Kerola

Natiivimetodien pinot (Native Method Stacks)

- toteutus voi käyttää tavallisia pinoja (”C stacks”) sellaisten natiivimetodien tukena, jota ei ole kirjoitettu Javalla
- käytetään myös Java tulkin toteutuksessa
- ei JVM toteutuksissa, joissa ei natiivimetodeja
- toteutuksesta riippuen kiinteän kokoinen tai dynaamisesti laajennettavissa

Copyright 2019 Teemu Kerola

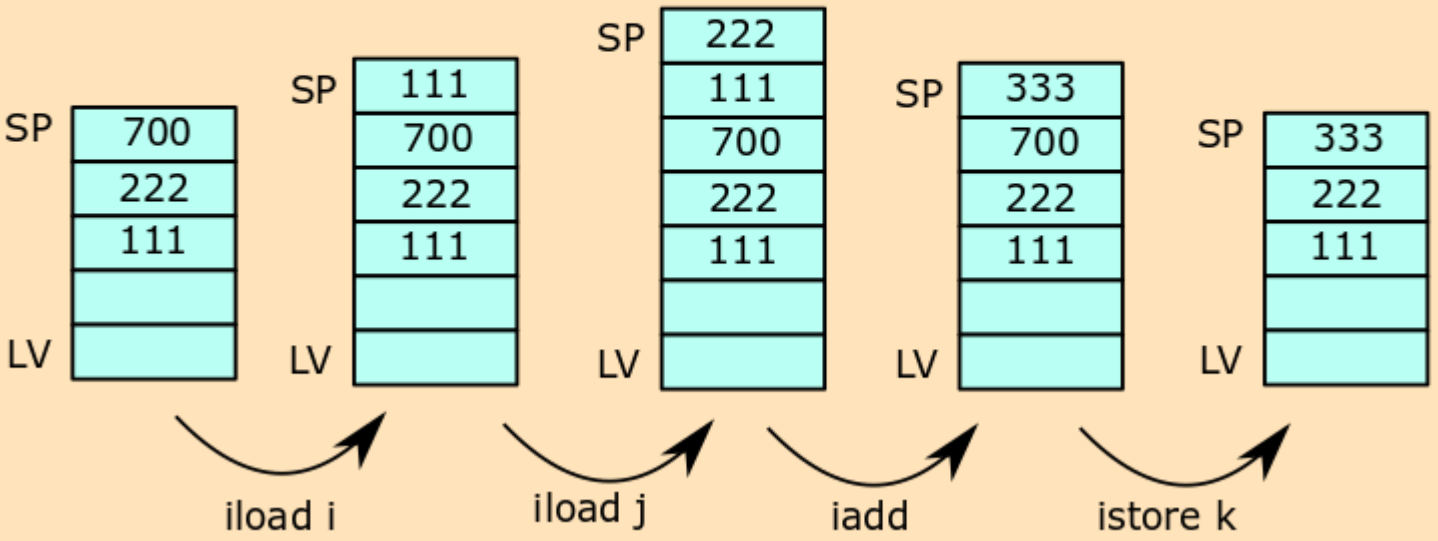
Suoritinarkkiteht. luokittelu aritm./loog.-käskyjen operandien lukumäärän mukaan

z = (x+y)(x+5)

Oper lkm: **0**	**2**	**3**	**1** (1 akkurek)
Pinokone	ttk-91	load-store	akkurek. kone
push x	load r1, x	load r1, x	load x
push y	add r1, y	load r2, y	add y
add	load r2, x	load r3, =5	store z
push x	add r2, =5	add r8, r1, r2	load x
push 5	add r1, r2	add r9, r1, r3	add =5
add	store r1, y	mul r10, r8, r9	mul z
mul		store r10, z	store z
pop z			

Copyright 2019 Teemu Kerola

Java: Yhteenlasku pinossa

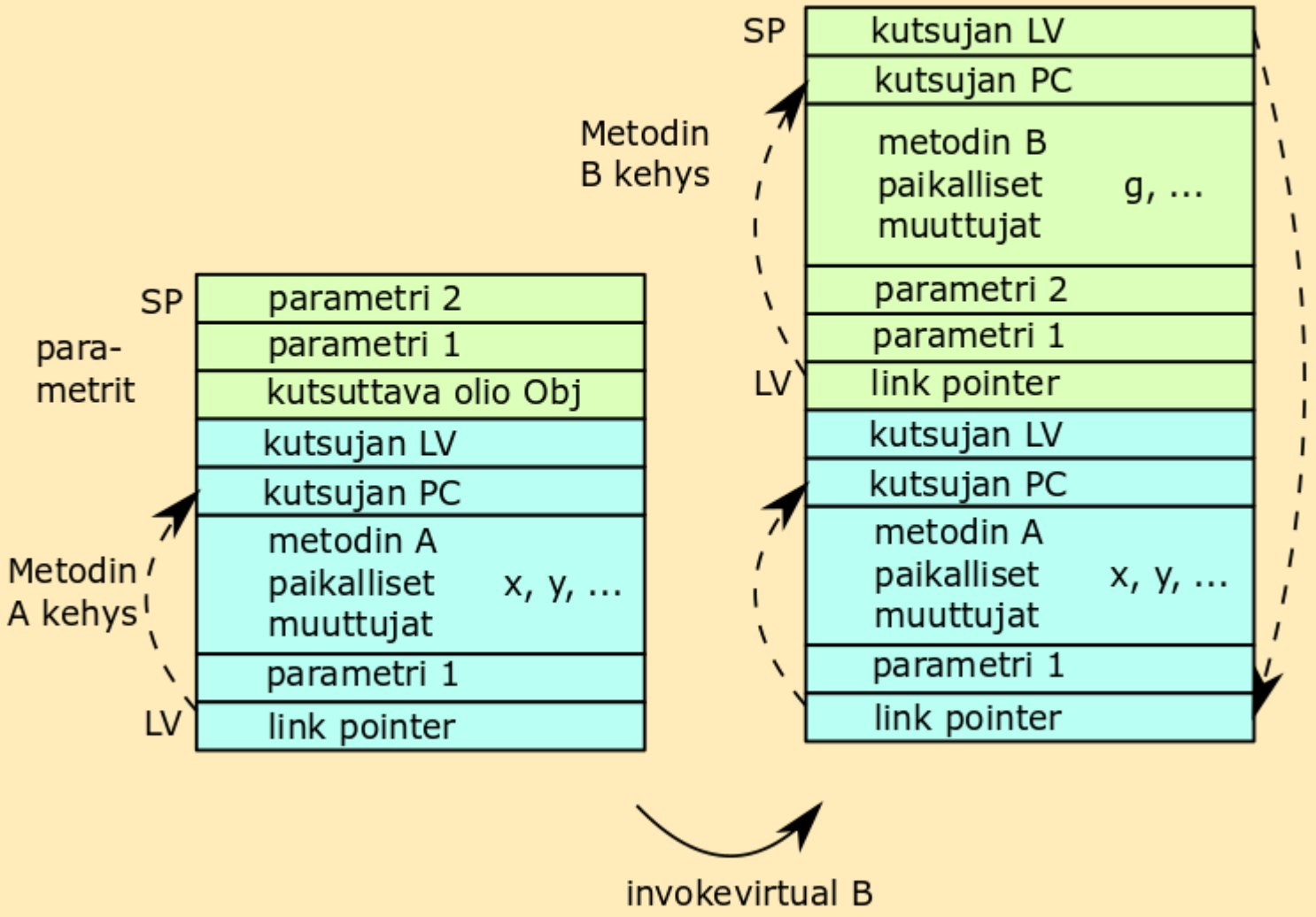


Tavukoodi tekstuaalisena	tavuina	
iload i	0x15 0x02	i:n osoite on LV+2
iload j	0x15 0x03	j:n osoite on LV+3
iadd	0x60	
istore k	0x36 0x04	k:n osoite on LV+4

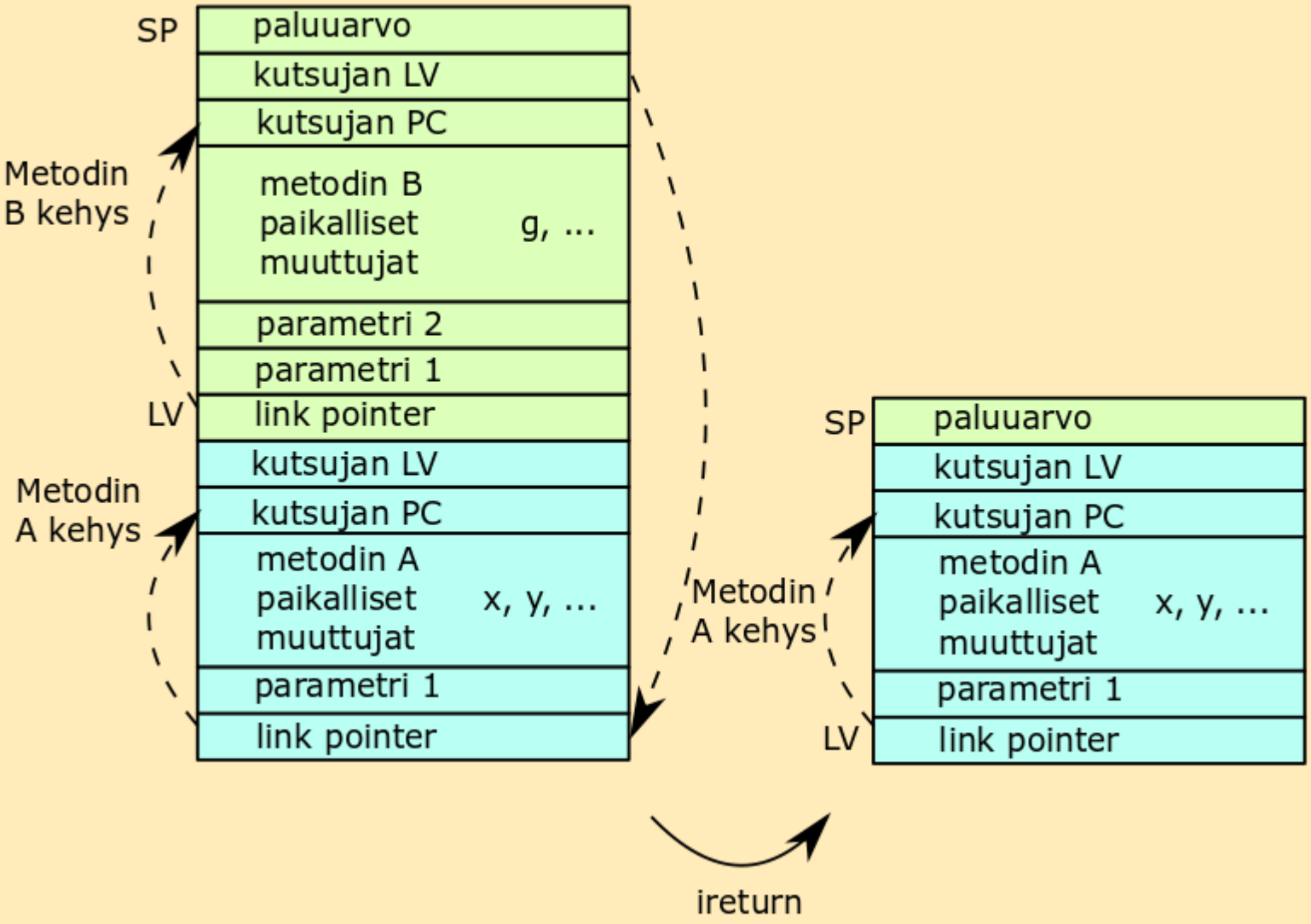
Keskustele

Copyright 2019 Teemu Kerola

Java: Metodin kutsun toteutus



Java: Metodista paluun toteutus



JVM tiedonosoitus

iadd	0x60	implisiittiset viittaukset pinoon, osoitteet SP, SP-1
bipush 5	0x10 0x05	viite vakioarvoon 5 (välitön operandi)
iconst_1	0x04	implisiittinen viite kokonaislukuvakioon 1
fconst_1	0x0c	implisiittinen viite liukulukuvakioon 1.0
iload 6	0x15 0x06	viite dataan osoitteessa LV+6 (indeksoitu viite)
invokevirtual #37	0xb6 0x00 0x25	viite dataan osoitteessa CPP+37 (indeksoitu viite)

Copyright 2019 Teemu Kerola

JVM käskyt

- Peruslaskutoimitukset (int, long, float, double)
  - add, sub, mul, div, rem, neg
- Boolean (int 32b, long 64b)
  - and, or, xor, shl, shr, ushr
- Pinon hallinta
  - dup, pop, swap, tauluk. luonti, esitystavan muutokset
- Load/Store
  - load, aload, store, astore, push-käskyt
- Vertailut
- Kontrollinsiirrot
- Muut

Keskustele

Copyright 2019 Teemu Kerola

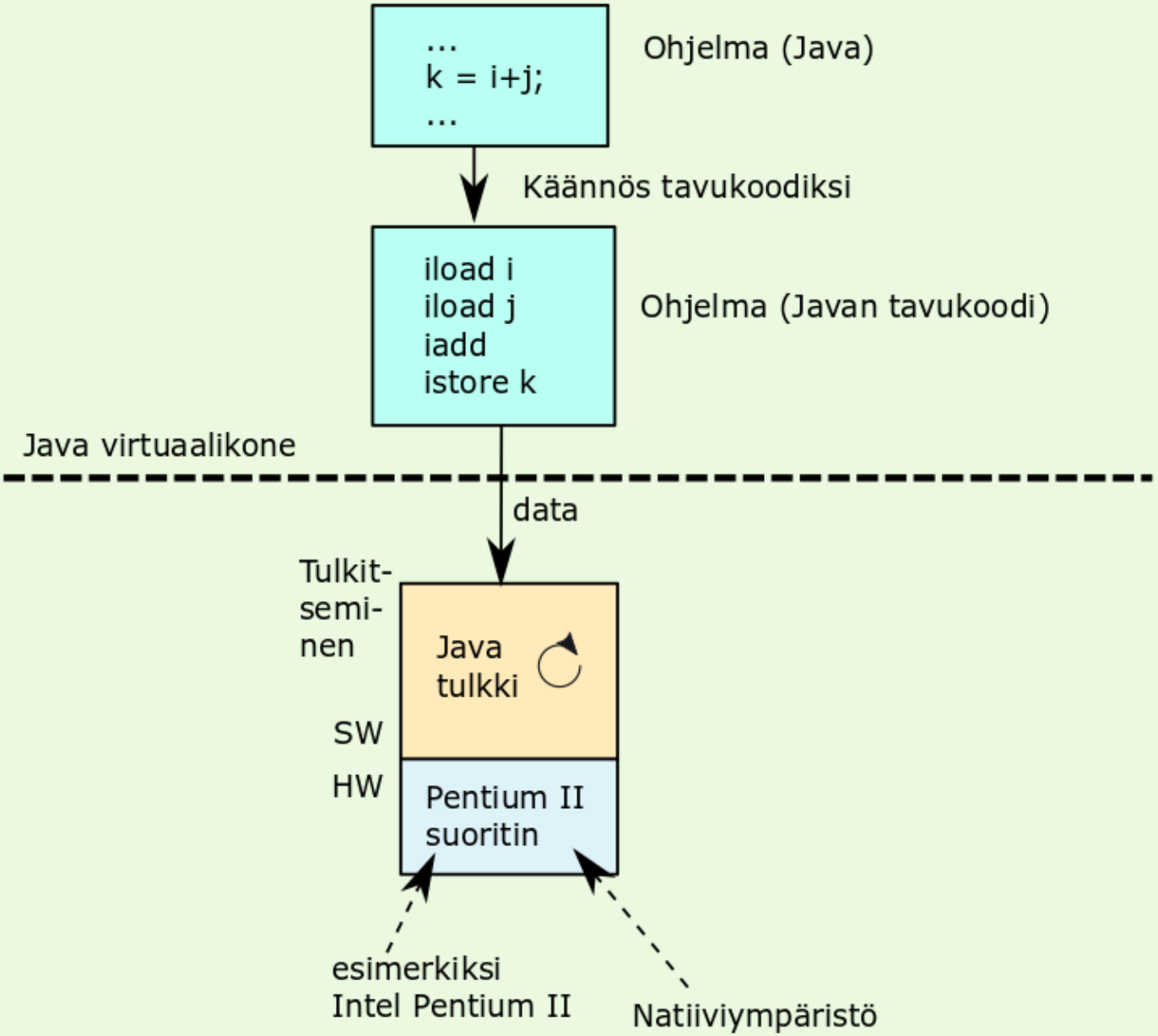
Tavukoodiesimerkki

```
k = i+5;

if (k=10)
    j=i;
else
    j=k;
```

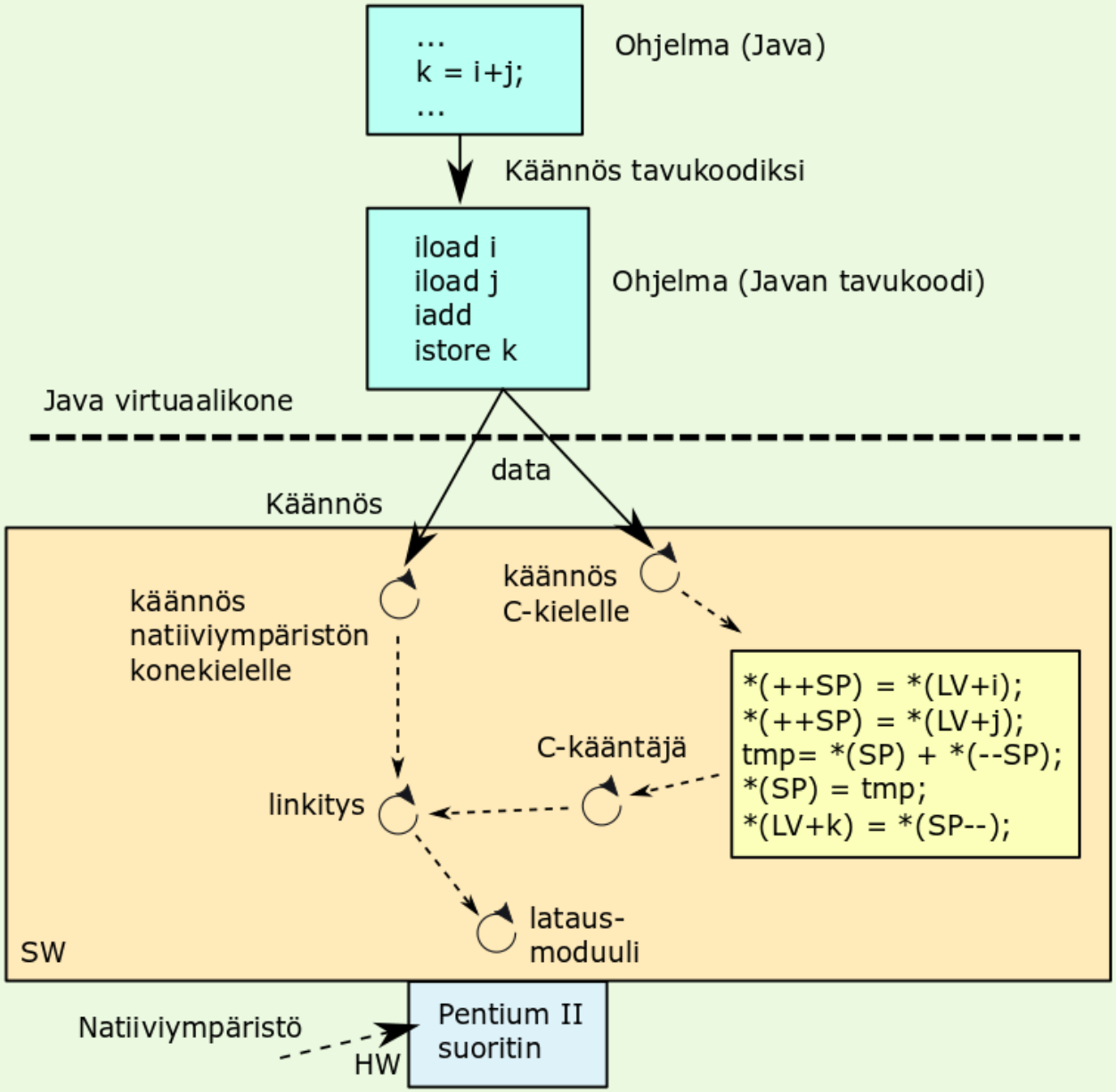
```
 tavukoodi tekstinä       heksadesimaalina

strt  iload i            100:   0x15 0x07
      bipush 5           102:   0x10 0x05
      iadd               104:   0x60
      dup                105:   0x59            k tarvitaan kohta taas
      istore k           106:   0x36 0x09

      bipush 10          108:   0x10 0x0A       k oli pinossa jo
      if_icmpeq else     110:   0x0f 0x00 0x0A  110+10=120
if10  iload i            113:   0x15 0x07
      istore j           115:   0x36 0x08
      goto done          117:   0xa7 0x00 0x07  117+7=124
else  iload k            120:   0x15 0x09
      istore j           122:   0x36 0x08
done  nop                124:   0x00
```


Copyright 2019 Teemu Kerola

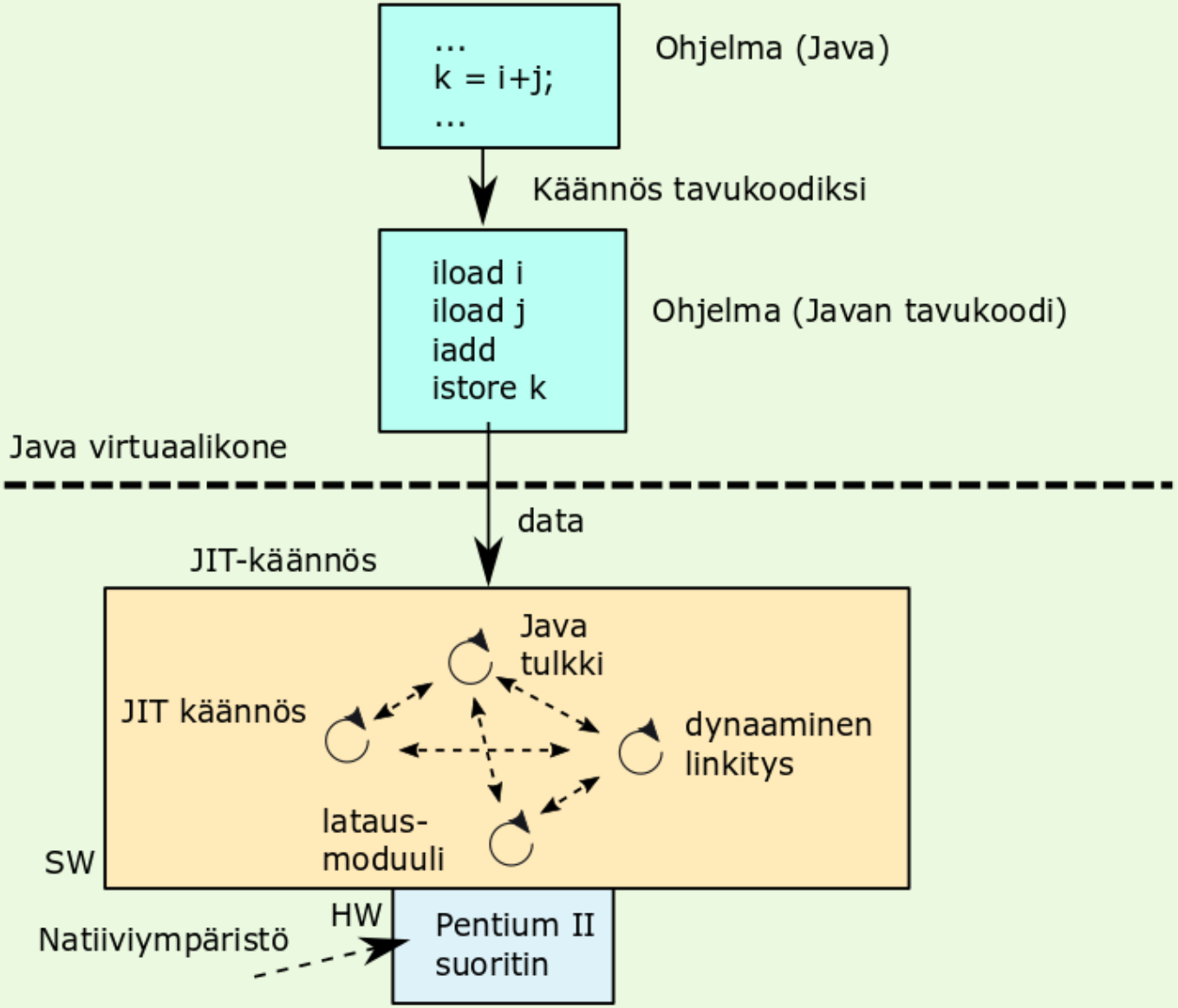
Java-ohjelmien suoritus tulkitsemalla



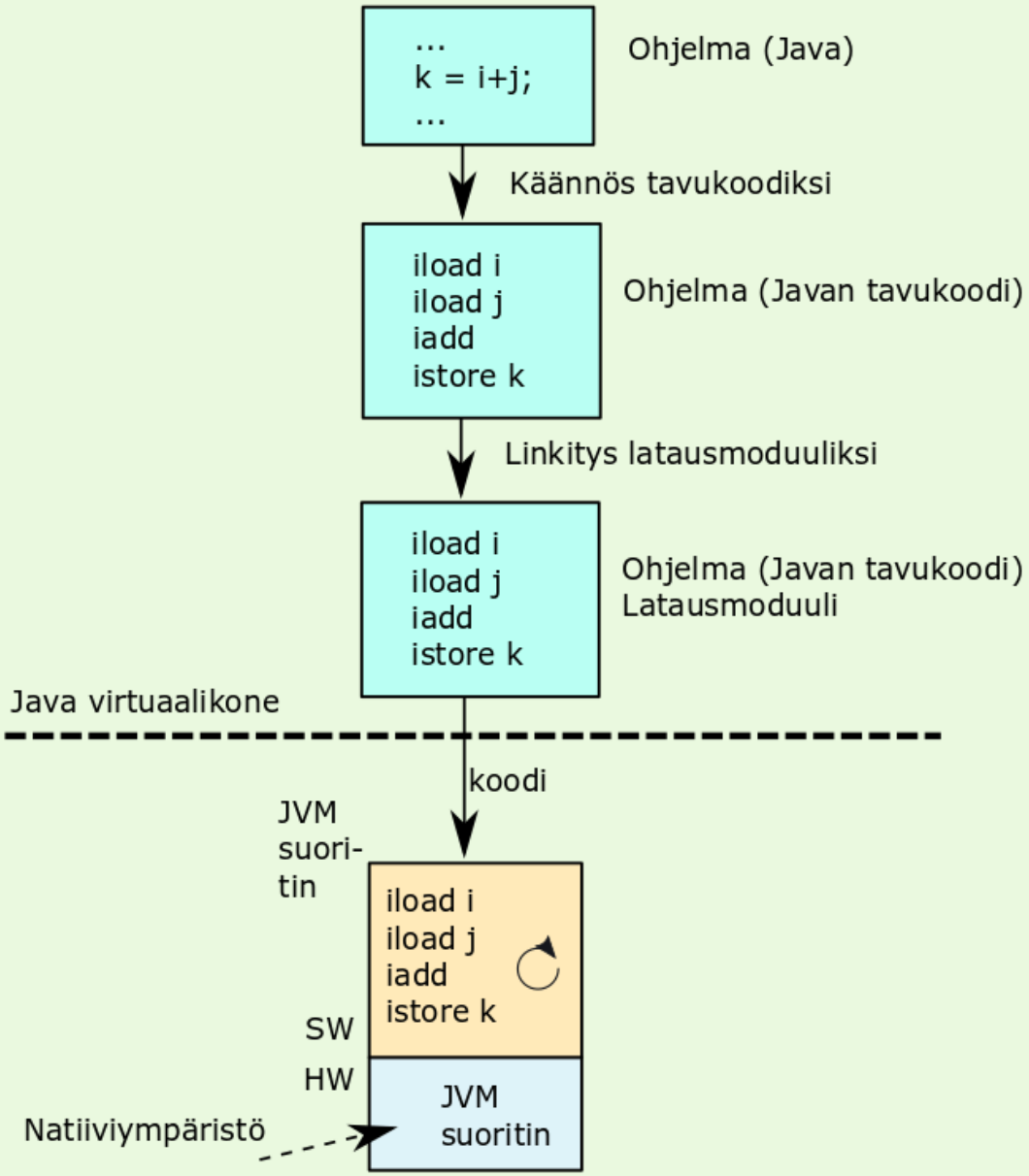
Java-ohjelmien suoritus kääntämällä ja linkittämällä



Java-ohjelmien suoritus JIT-kääntämällä ja dynaamisesti linkittämällä



Java-ohjelmien suoritus natiivisuorittimella



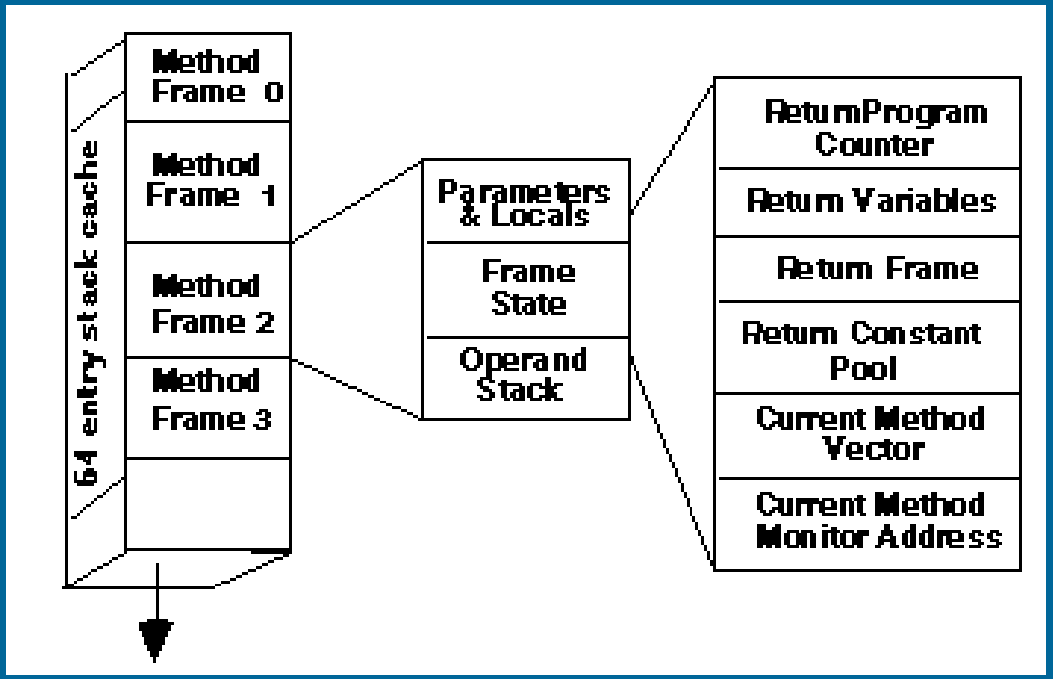
Java suoritin: Sun PicoJAVA II

- Suorittimen määrittely, jonka mukaisessa koneessa byte-koodi -muodossa olevia ohjelmia voidaan sellaisenaan suorittaa
- Valinnainen välimuisti ja liukulukusuoritin
- Kaikki 226 JVM konekäskyä
  - jotkut käskyt toteutettu aliohjelmilla, jotka aktivoidaan keskeytyskäsittelymekanismin avulla
- Myös 115 muuta konekäskyä käyttöjärjestelmän ja muiden ohjelmointikielten toteuttamiseksi
  - C ja C++

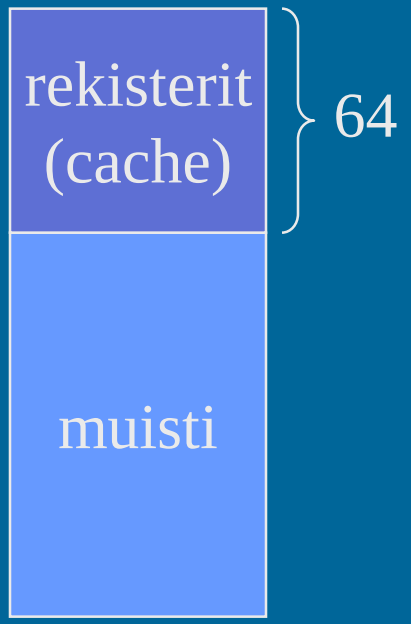
Copyright 2019 Teemu Kerola

PicoJAVA II pino

- 64 (välimuisti-) laiterekisteriä JVM pinon huipun talletukseen
  - loput JVM pinosta muistissa



Shawn Lauzon, Survey of the JavaChip



Copyright 2019 Teemu Kerola

PicoJAVA II rekisterit

- 25 rekisteriä á 32 bittiä
  - PC, LV, CPP, SP (pino kasvaa alaspäin)
  - OPLIM alaraja SP:lle; alitus aiheuttaa keskeytyksen
  - FRAME osoittaa paikallisten muuttujataulukon jälkeen talletettuun metodin paluuosoitteeseen
  - PSW (tilarekisteri)
  - rekisteri, joka kertoo pinon välimuistirekistereiden tämänhetkisen käytön
  - 4 rekisteriä keskeytysten ja break-point'ien käsittelyyn
  - 4 rekisteriä säikeiden hallintaan
  - 4 rekisteriä C ja C++ ohjelmien toteutukseen
  - 2 rajarekisteriä sallitun muistialueen rajoittamiseen
  - suorittimen version numero ja konfiguraatiorekisterit

Copyright 2019 Teemu Kerola

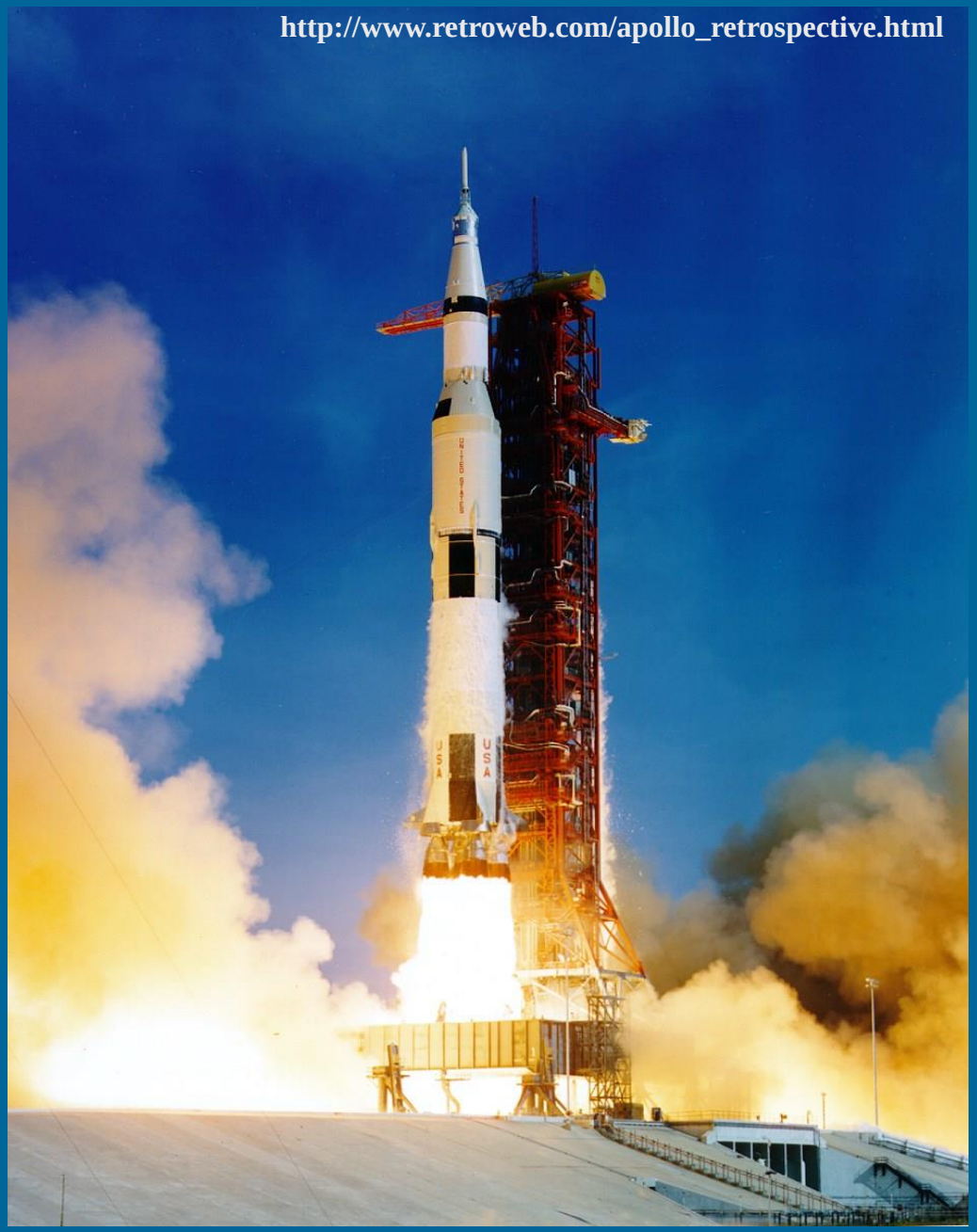
PicoJAVAn 115 ylim. käskyä

- Read/write ylimääräisille rekistereille
- Osoittimien manipulointikäskyt
  - mitä tahansa muistialuetta voidaan suoraan lukea/kirjoittaa
  - tarvitaan C/C++ varten
- C/C++ aliohjelmien kutsu ja paluukäskyt
- Natiivi HW manipulointi
  - tyhjennä välimuisti (osittain? kokonaan?), ...
- Muut käskyt
  - power on/off, ...

Copyright 2019 Teemu Kerola

--
Kurssin loppu
--

http://lue.kokeeseen.edu/ajoissa.html



http://www.retroweb.com/apollo_retrospective.html

Copyright 2019 Teemu Kerola