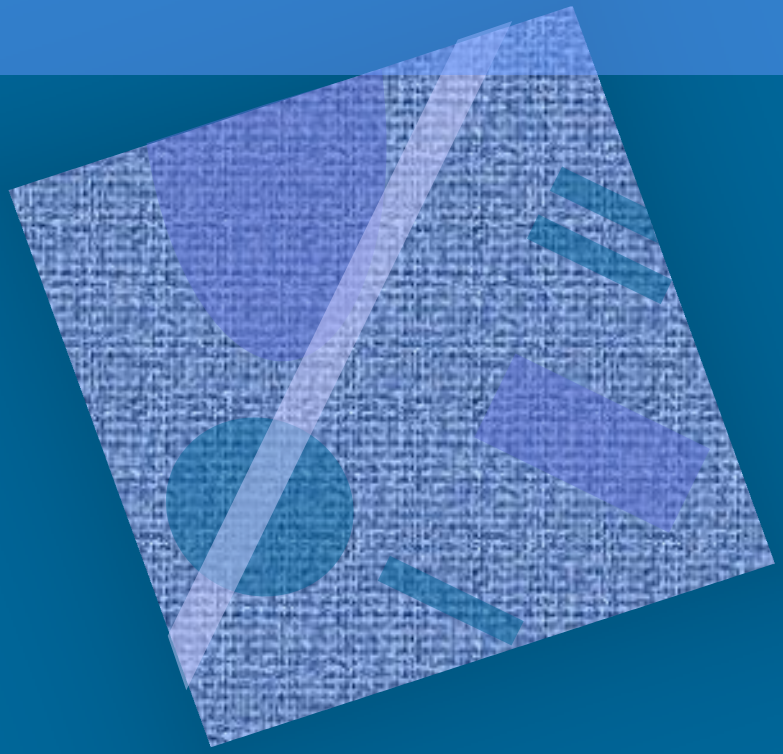
Luento 11 (verkkoluento 11)

Tulkinta ja emulointi



Java ohjelman suoritus

Tavukoodi

JVM

Tulkinta

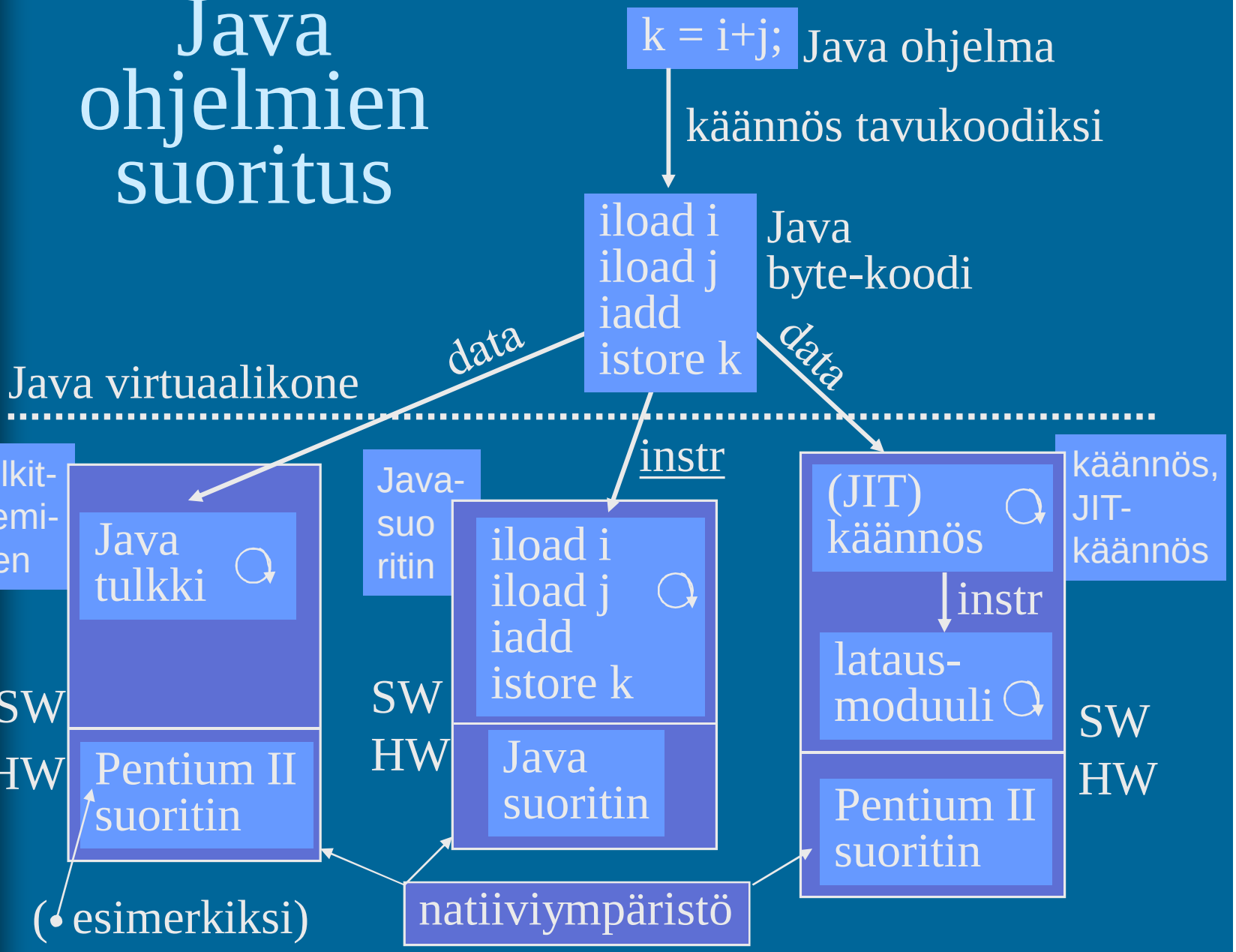
Java-suoritin

Käännös ja JIT-käännös

JVM vs. Titokone

Copyright 2018 Teemu Kerola

Java ohjelmien suoritus



Copyright 2018 Teemu Kerola

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 2018 Teemu Kerola

JVM:n tietorakenteet

- JVM pino

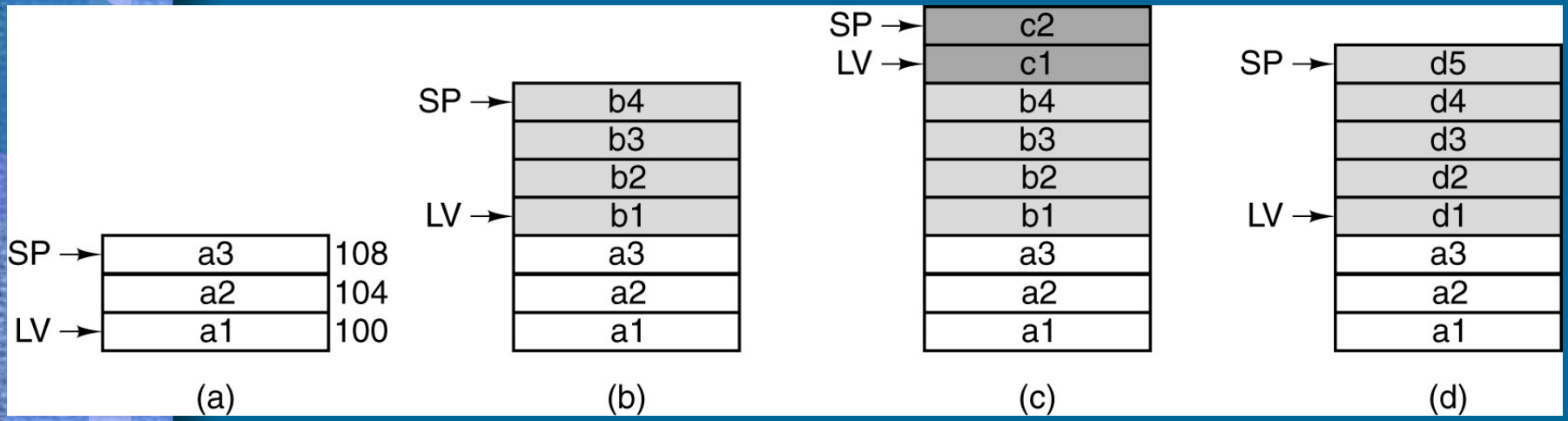
Figs 4-8, 4-9, 4-10 [Tane13]

  - kuten tavallinen aktivointitietuepino
  - koostuu useista *kehyksistä* (frames) (vrt. aktivointitietue) ja operandipinosta
  - käyttö: kehyksille ainoastaan push/pop operaatiot, operandipinon alkioille myös push/pop
  - ei tarvita yhtenäistä muistialuetta
  - allokoidaan keosta (heap)
  - toteutuksesta riippuen rajallinen tai dynaamisesti laajennettavissa
  - tila loppu Þ StackOverflowError, OutOfMemoryError

http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html

Copyright 2018 Teemu Kerola

Fig 4-8 [Tane13]. Stacks (1)

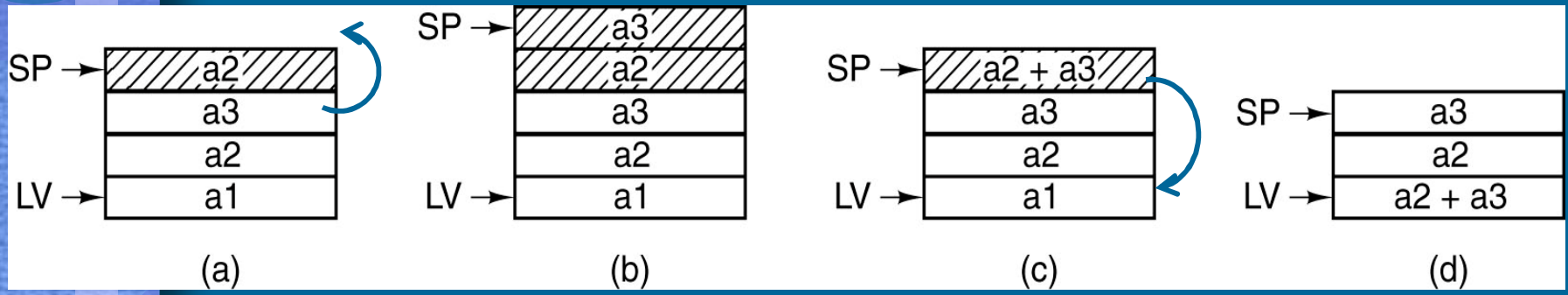


Use of a stack for storing local variables.

a) While *A* is active.
b) After *A* calls *B*.
c) After *B* calls *C*.
d) After *C* and *B* return and *A* calls *D*.

Copyright 2018 Teemu Kerola

Fig 4-9 [Tane13]. Stacks (2)

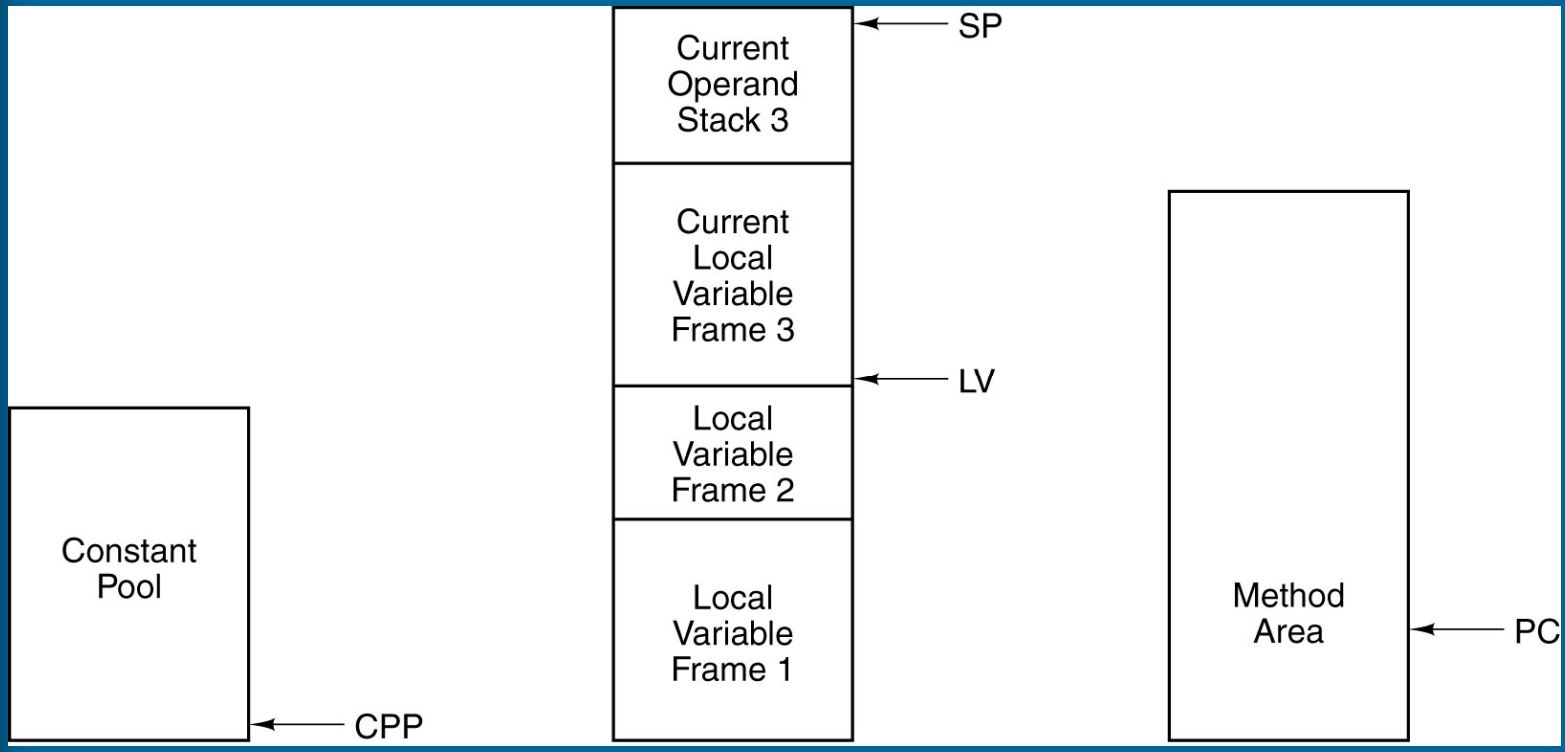


LOAD	ADD	STORE
iLOAD 2	*iADD*	*iSTORE 0*

Use of an operand stack (not registers) for doing an (e.g., integer) arithmetic computation.

Copyright 2018 Teemu Kerola

Fig 4-10 [Tane13]



The various parts of the IJVM memory.

Copyright 2018 Teemu Kerola

JVM:n tietorakenteet (jatkuu)

- JVM keko (JVM heap)
  - yhteinen kaikille saman virtuaalikoneen säikeille
  - automaattinen roskienkeruu (garbage collector)
    - ei-käytössä (implisiittisesti ”vapautettu”) oleva muistialue palautetaan uusiokäyttöön (vapaaksi)
    - ei tarvita erikseen *free* operaatiota Java ohjelmassa
    - voi hidastaa suoritusta milloin vain (ongelma ainakin tosiaikajärjestelmissä)
  - toteutuksesta riippuen kiinteän kokoinen tai dynaamisesti laajennettavissa
  - ei tarvitse muodostaa yhtenäistä muistialuetta natiivijärjestelmän keossa
  - tila loppu Þ OutOfMemoryError

Copyright 2018 Teemu Kerola

JVM:n tietorakenteet (jatkuu)

ks. Fig. 4-10 [Tane13]

- JVM metodialue (JVM Method Area)
  - yhteinen kaikille JVM säikeille
  - vastaa tavallista kääntäjän tuottamaa koodisegmenttiä
  - loogisesti osa JVM kekoa
  - toteutuksesta riippuen kiinteän kokoinen tai dynaamisesti laajennettavissa
  - tila loppu Þ OutOfMemoryError

Copyright 2018 Teemu Kerola

JVM:n tietorakenteet (jatkuu)

ks. Fig. 4-10 [Tane13]

- Javan suoritusaikainen vakioallas (runtime constant pool)
  - joka luokalle (class) ja liittymälle (interface)
  - suoritusaikainen esitystapa tiedoston *class constant_pool* -taulukolle
  - vastaa vähän tavallista symbolitaulua
  - useita erilaisia vakioita (käännösaikaiset literaalit, suor. aikana ratkottavat attribuutit, …)
  - talletetaan JVM metodialueelle
  - tila loppu Þ OutOfMemoryError

Copyright 2018 Teemu Kerola

JVM:n tietorakenteet (jatkuu)

- 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
  - tila loppu Þ StackOverflowError, OutOfMemoryError

Copyright 2018 Teemu Kerola

JVM:n tietorakenteet (jatkuu)

ks. Fig. 4-10 [Tane13]

- JVM rekisterit
  - PC osoittaa johonkin JVM metodialueelle
  - CPP osoittaa vakioaltaaseen
  - LV on paikallisten muuttujien kantaosoite (vähän kuten FP ttk-91:ssä)
  - SP osoittaa JVM operandipinon huipulle
  - kaikki rekisterit implisiittisiä, niitä ei erikseen nimetä JVM konekäskyissä

Copyright 2018 Teemu Kerola

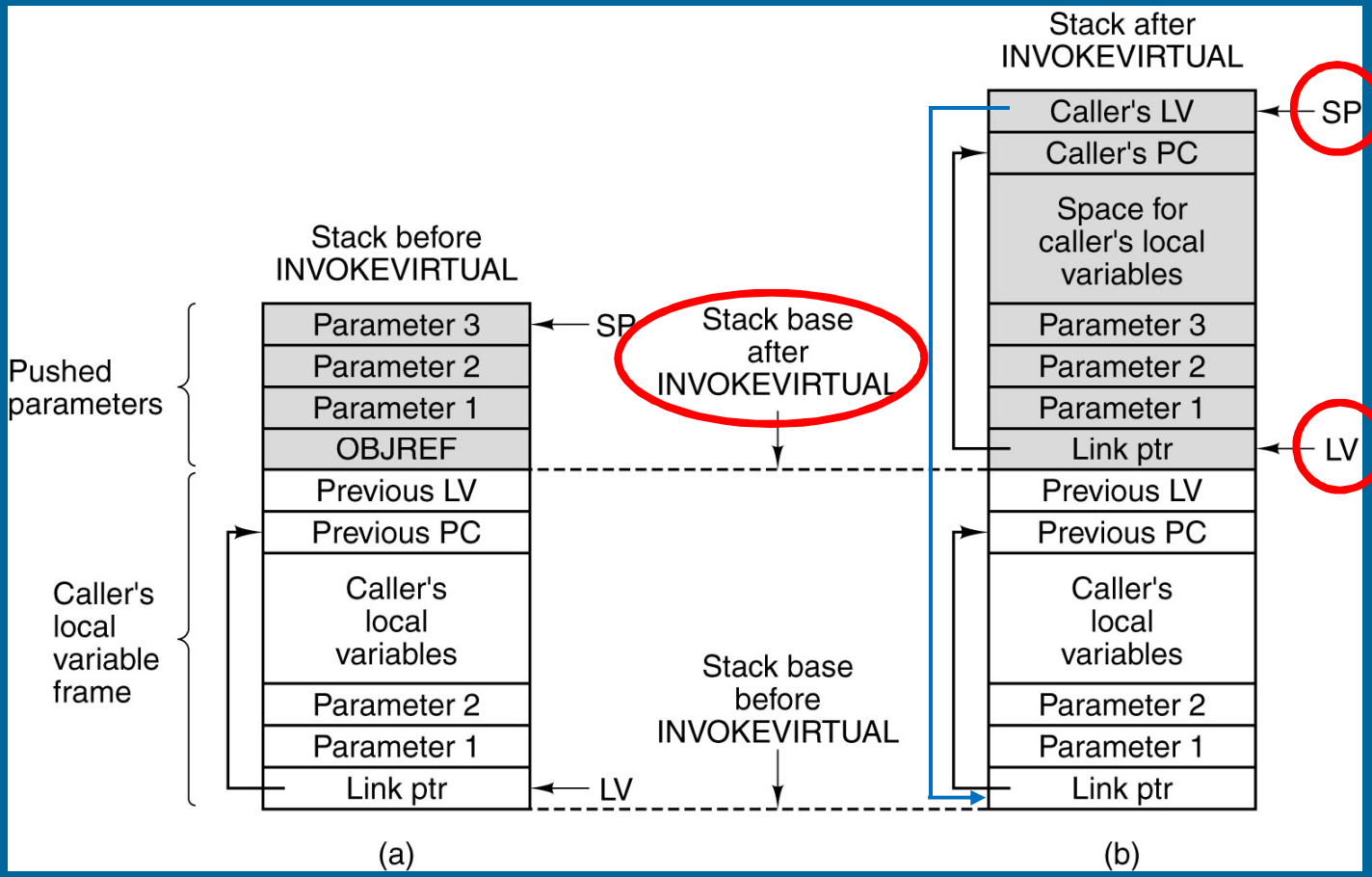
JVM:n tietorakenteet (jatkuu)

ks. Figs 4-12, 4-13 [Tane13]

- JVM kehys (frame, raami)
  - talletetaan JVM pinoon, luodaan metodin kutsun yhteydessä, vapautetaan metodista poistuttaessa
  - paikalliset muuttujat
  - parametrit, paluuarvo ja välitulokset
  - dynaamisen linkityksen toteutusväline
  - keskeytysten toteutusväline

Copyright 2018 Teemu Kerola

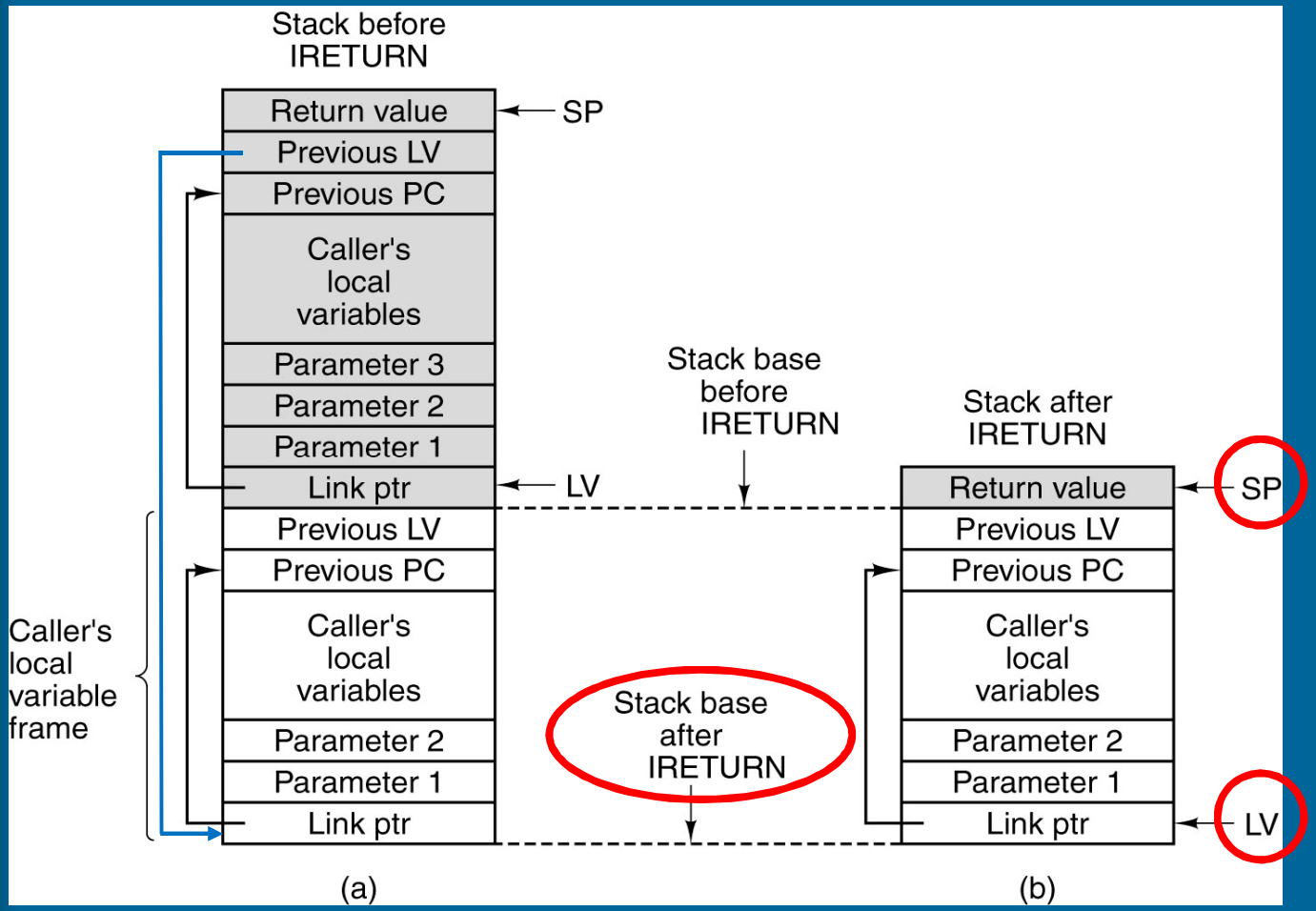
Fig 4-12 [Tane13]
The IJVM Instruction Set (2)



- Memory before executing INVOKEVIRTUAL.
- After executing it.

Copyright 2018 Teemu Kerola

Fig 4-13 [Tane13]
The IJVM Instruction Set (3)



- Memory before executing IRETURN.
- After executing it.

Copyright 2018 Teemu Kerola

JVM kehyksen data

- Paikalliset muuttujat sisältävä taulukko

ks. Fig. 4-13 [Tane13]

  - viittaukset indeksoituna (0, 1, 2, …) rekisterin LV suhteen
  - indeksit sanoina
  - kaksi sanaa vaativa muuttuja (long, double) sijoitetaan kahteen peräkkäiseen (32 bittiseen) sanaan
  - big-endian talletus
- Parametrit, paluuarvon ja välitulokset sisältävä operandipino
  - SP osoittaa pinon huipulle
  - pinoarkkitehtuuri (vs. rekisteriarkkitehtuuri)

Copyright 2018 Teemu Kerola

JVM:n tiedon osoitusmoodit (4)

- Välitön operandi

iINC 2 34

Java: xLoc += 34;

- Indeksoitu

iINC 2 34

tehollinen muistiosoite (LV) + 2

- Pino-osoitus

iADD

Java: a1 = a2+a3;

korvaa pinon kaksi päällä olevaa kokonaislukua niiden summalla

ks. Fig. 4-9 [Tane13]

- Taulukko-osoitus pinon kautta

aLOAD 1
iLOAD 2
iALOAD
iSTORE 3

Korvaa pinon pinnalla olevat taulukon alkuosoitteen ja indeksin k.o. taulukon alkiolla

Java: a = T[i];

Copyright 2018 Teemu Kerola

JVM käskyt

- Peruslaskutoimitukset
  - add, sub, mul, div, rem, neg
- Boolean
  - 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

ks. Fig. 4-11 [Tane13]

Copyright 2018 Teemu Kerola

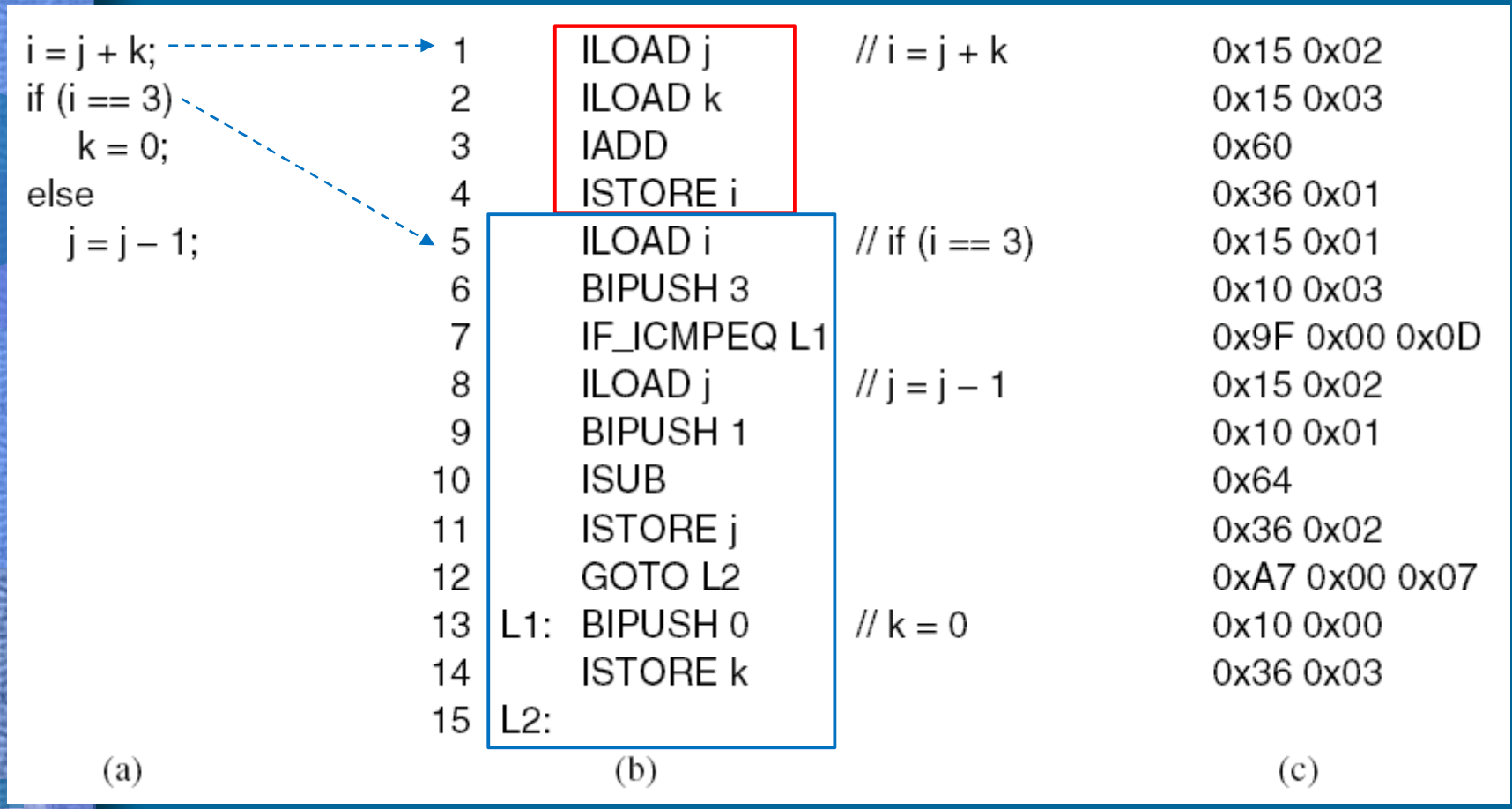
Fig 4-11 [Tane13]
The IJVM Instruction Set (1)

Hex	Mnemonic	Meaning
0x10	BIPUSH *byte*	Push byte onto stack
0x59	DUP	Copy top word on stack and push onto stack
0xA7	GOTO *offset*	Unconditional branch
0x60	IADD	Pop two words from stack; push their sum
0x7E	IAND	Pop two words from stack; push Boolean AND
0x99	IFEQ *offset*	Pop word from stack and branch if it is zero
0x9B	IFLT *offset*	Pop word from stack and branch if it is less than zero
0x9F	IF_ICMPEQ *offset*	Pop two words from stack; branch if equal
0x84	IINC *varnum const*	Add a constant to a local variable
0x15	ILOAD *varnum*	Push local variable onto stack
0xB6	INVOKEVIRTUAL *disp*	Invoke a method
0x80	IOR	Pop two words from stack; push Boolean OR
0xAC	IRETURN	Return from method with integer value
0x36	ISTORE *varnum*	Pop word from stack and store in local variable
0x64	ISUB	Pop two words from stack; push their difference
0x13	LDC_W *index*	Push constant from constant pool onto stack
0x00	NOP	Do nothing
0x57	POP	Delete word on top of stack
0x5F	SWAP	Swap the two top words on the stack
0xC4	WIDE	Prefix instruction; next instruction has a 16-bit index

The IJVM instruction set. The operands *byte*, *const*, and *varnum* are 1 byte. The operands *disp*, *index*, and *offset* are 2 bytes.

Copyright 2018 Teemu Kerola

Fig 4-11 [Tane10], Compiling Java to IJVM (1)



```
i = j + k;                 1     ILOAD j          // i = j + k        0x15 0x02
if (i == 3)                2     ILOAD k                           0x15 0x03
    k = 0;                 3     IADD                              0x60
else                       4     ISTORE i                          0x36 0x01
   j = j – 1;              5     ILOAD i          // if (i == 3)     0x15 0x01
                           6     BIPUSH 3                          0x10 0x03
                           7     IF_ICMPEQ L1                      0x9F 0x00 0x0D
                           8     ILOAD j          // j = j – 1       0x15 0x02
                           9     BIPUSH 1                          0x10 0x01
                          10     ISUB                              0x64
                          11     ISTORE j                          0x36 0x02
                          12     GOTO L2                           0xA7 0x00 0x07
                          13  L1: BIPUSH 0         // k = 0           0x10 0x00
                          14     ISTORE k                          0x36 0x03
                          15  L2:
```

(a) (b) (c)

a) A Java fragment.

b) The corresponding Java assembly language.

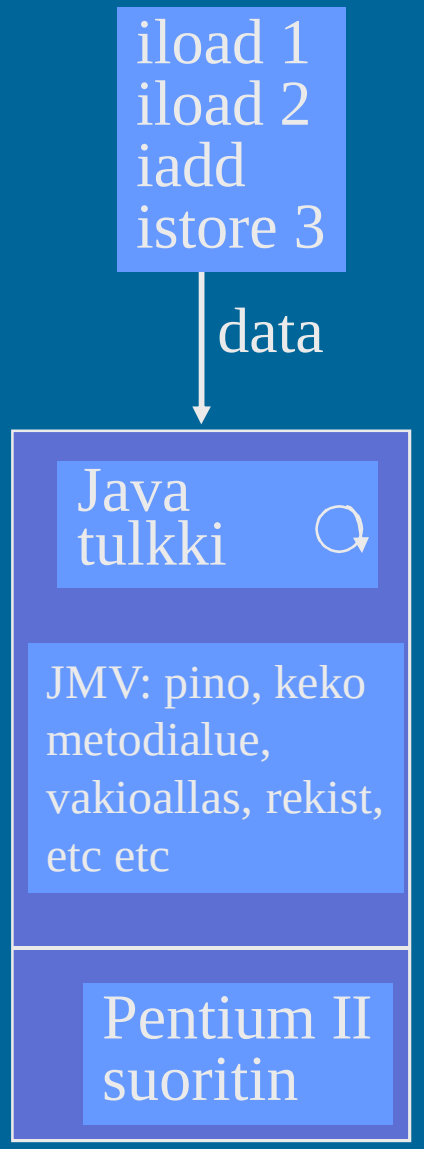
c) The IJVM program in hexadecimal.

Copyright 2018 Teemu Kerola

Tulkin nimi voi olla JVM !!

Java tulkki

- Emuloi JVM konekielen käskyjä (tavukoodia)
- Yksi (tavukoodi) käsky kerrallaan
- JVM rekisterit ja muistialueet emuloitu tulkin tietorakenteina muistissa
  - vrt. Titokone ja ttk-91
- Hidasta, mutta joustavaa

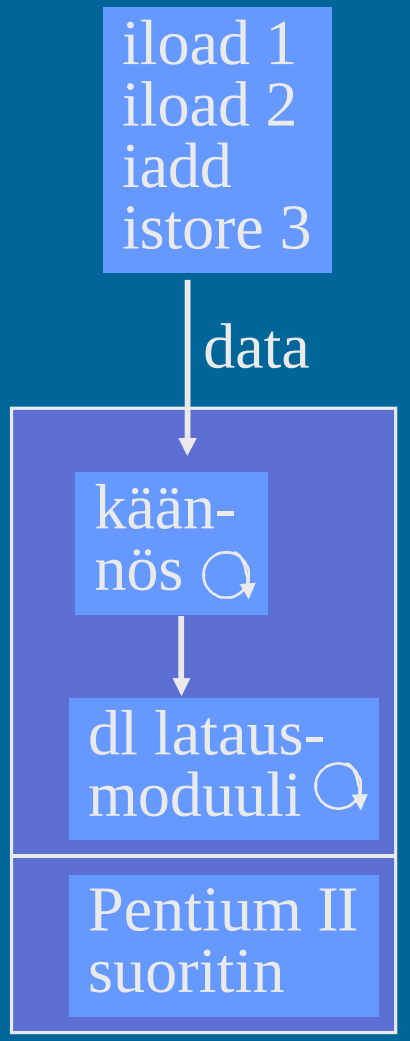


Copyright 2018 Teemu Kerola

Käännös natiivikoneelle

Eri tavat toteuttaa kääntäjä

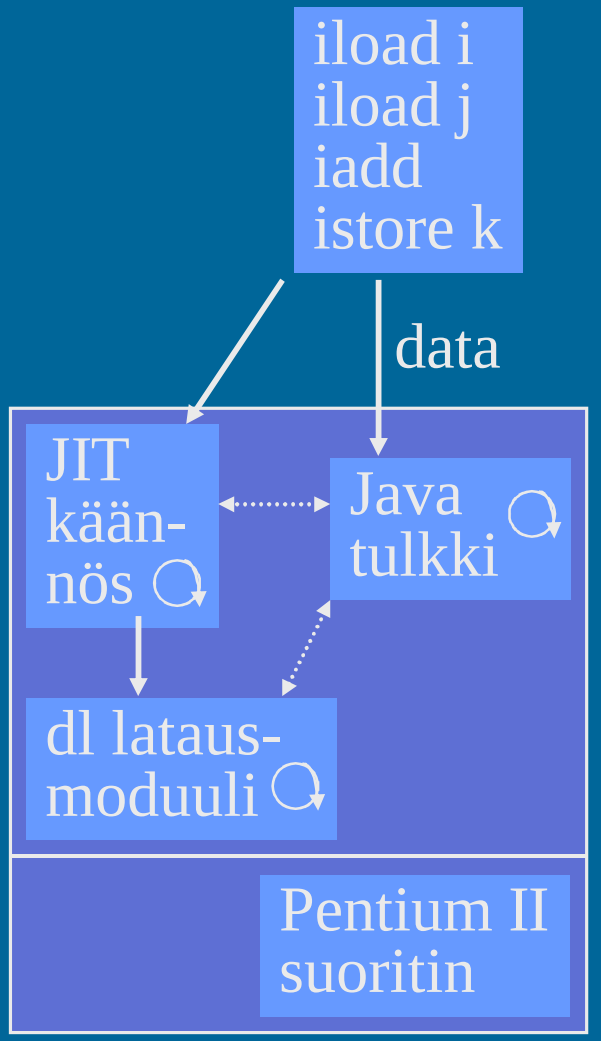
- (a) Käännetään tavukoodi suoraan natiivikoneen konekielelle ja suoritetaan se normaalin ohjelman tapaan
- (b) Käännetään tavukoodi ensin korkean tason kielelle (esim C), joka sitten käännetään standardi C-kääntäjällä natiivikoneen konekielelle
  - alkuosa riittää tehdä kerran
  - loppuosa on jo valmiina yleensä
- Ongelma: ei dynaamista linkitystä



Copyright 2018 Teemu Kerola

Java JIT käännös

- JIT = Just-in-Time
- Emulointi ja/tai käännös tilanteesta riippuen
- Käännä luokka natiivikonekielelle dynaamisesti linkitettäväksi moduuliksi, mutta vasta juuri ennen luokan metodin kutsua
- Tarvitsee paljon muistia
- Voi hidastaa suoritusta, jos käännökseen menee enemmän aikaa kuin tulkitsemiseen
  - käännös vasta 2. kutsukerralla?
- JVM rekisterit ja muistialueet emuloitu tulkin tietorakenteina, joita natiivikoodi myös käyttää



Copyright 2018 Teemu Kerola

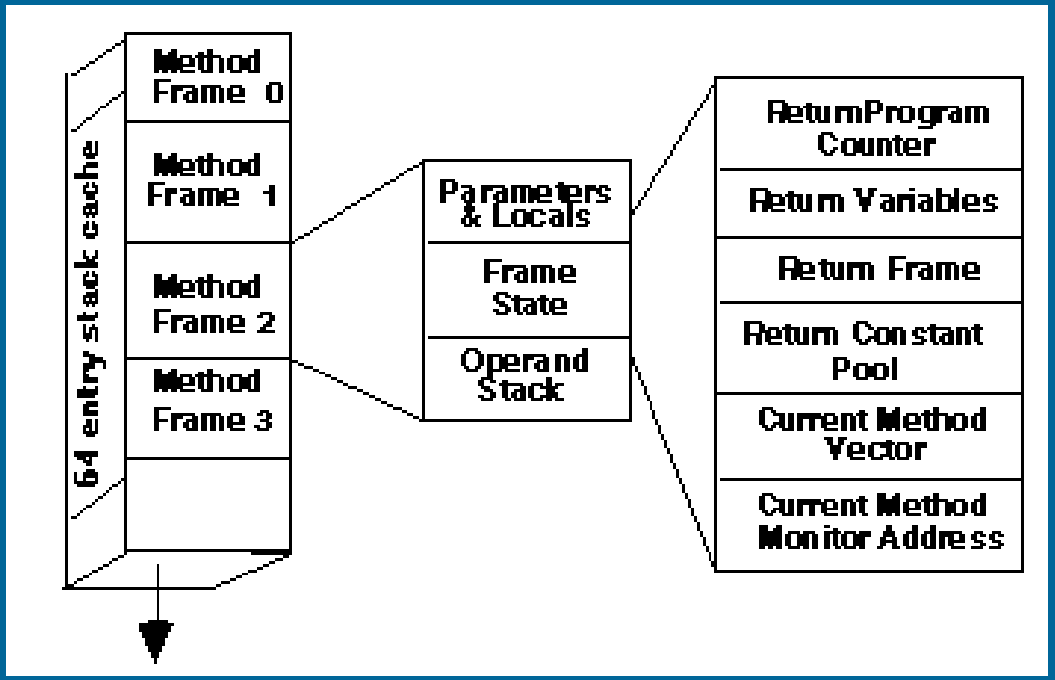
Java suoritin: Sun PicoJAVA II

- 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 2018 Teemu Kerola

PicoJAVA II pino

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



Shawn Lauzon,
Survey of the JavaChip

Copyright 2018 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 2018 Teemu Kerola

PicoJAVA ylim. käskyt

- 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 2018 Teemu Kerola

PicoJAVA toteutuksia (2)

- Sun microJAVA 701
  - valinnainen välimuisti
  - oma muistiväylä
  - PCI väylä muille laitteille
  - 16 ohjelmoitavaa I/O johdinta
    - näppäimet, LEDit, …
  - 3 ohjelmoitavaa ajastinta (Þ kellolaitekeskeytykset)
  - suunnattu halpoihin kannettaviin laitteisiin (kämmenmikro, PDA - Personal Digital Assistant)
- Sun ultraJAVA
  - nopeampi, parempi, kalliimpi, …
  - suunnattu grafiikka- ja multimediasovelluksiin

Copyright 2018 Teemu Kerola

Muita Java-suorittimia

- JEM (Rockwell Collins)
- PSC1000 (Patriot Scientific)
  - dSys (Saksa), lääketieteellisiä laitteita
- MJ501 (LG Semicon)
  - TV, älykortit
- JSR-001, Real-Time Specification for Java (Java Community Process, ”Sun Microsystems”)
  - aJile: aJ-80, aJ-100, älykkäät liikkuvat laitteet
- Komodo, SHAP, jHISC, Cjip, ARM926EJ-S, ObjectCore, …

Copyright 2018 Teemu Kerola

Sun MAJC

- MAJC - Microprocessor Architecture for Java Computing
  - suoritinarkkitehtuurin määrittely
  - tavoitteena suuri nopeus Java, C ja C++ sovelluksille
  - suunnattu multimediasovelluksiin verkossa
  - tukee hyvin JIT-käännöstä

Copyright 2018 Teemu Kerola

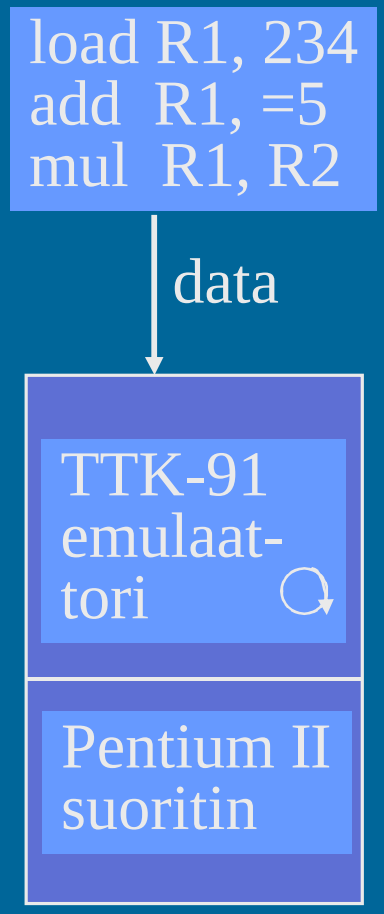
MAJC toteutus: MAJC 5200

- 1-4 suoritinta (2 suorittimen lastu, v. 1999)
- Useiden (peräkkäin kutsuttavien) metodien samanaikainen suoritus eri suorittimilla
  - ennakoivalle (speculative) suoritukselle oma kasa
  - peruutus (rollback), jos ennakoitu suoritus meni pieleen
- 4 säiettä suorituksessa per suoritin
  - säikeen vaihto nopeampaa kuin muistista luku!
  - laiterekisterit 4:lle säikeelle! (hyper-threaded processor)
  - välimuistin hudin aikana suoritetaan muita säikeitä
- VLIW arkkitehtuuri – 4 konekäskyä samanaikaisesti
- Suunnattu interaktiiviseen TV:hen, virtuaalitodellisuussovelluksiin, ...

Copyright 2018 Teemu Kerola

TTK-91 Emulointi

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

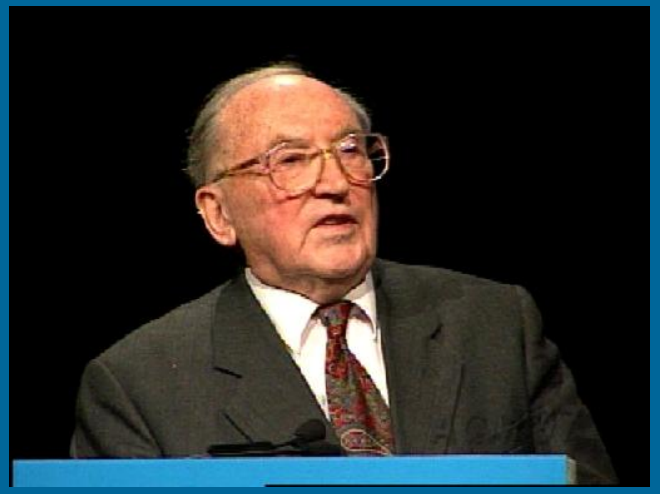


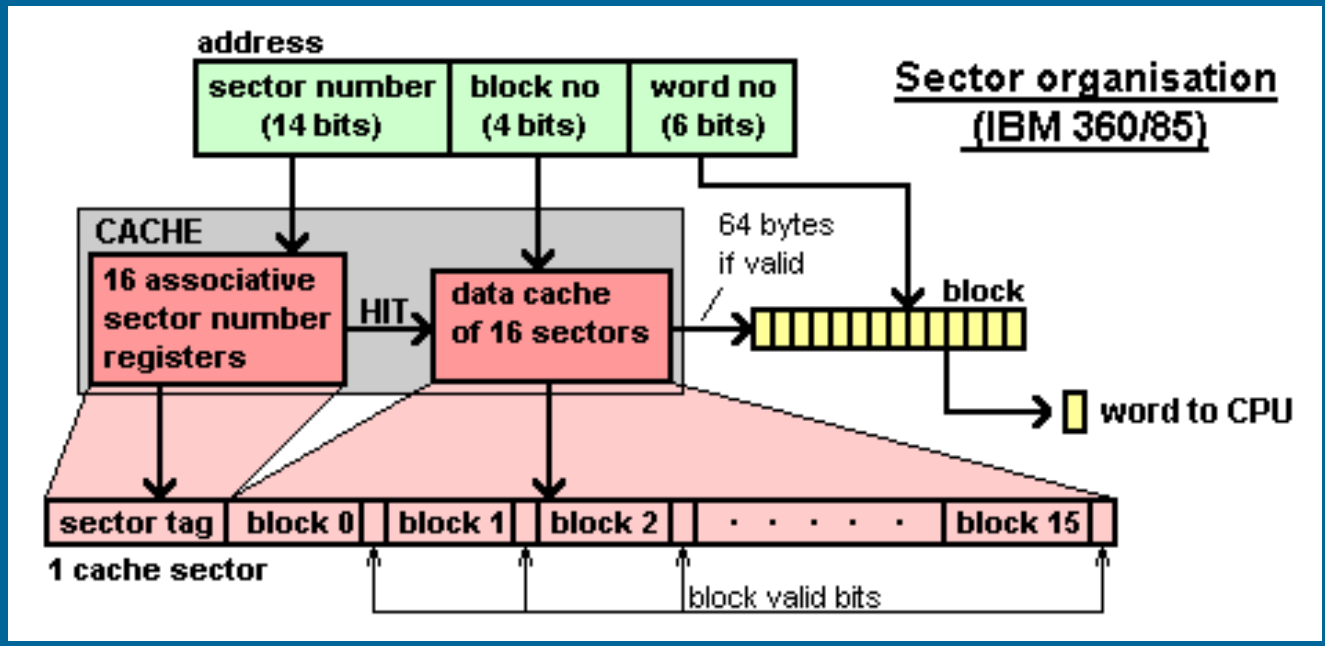
ks. simulaattorin koodi, proj. Titokone:
http://www.cs.helsinki.fi/group/nodes/kurssit/tito/2012s/Interpreter.java
http://www.cs.helsinki.fi/group/nodes/kurssit/tito/2012s/Processor.java

Copyright 2018 Teemu Kerola

-- Loppu --

- Välimuisti (1965, MauriceWilkes)
  - IBM S/360 Model 85
    - 1968
    - 256 lohkoa á 64 tavua





Copyright 2018 Teemu Kerola