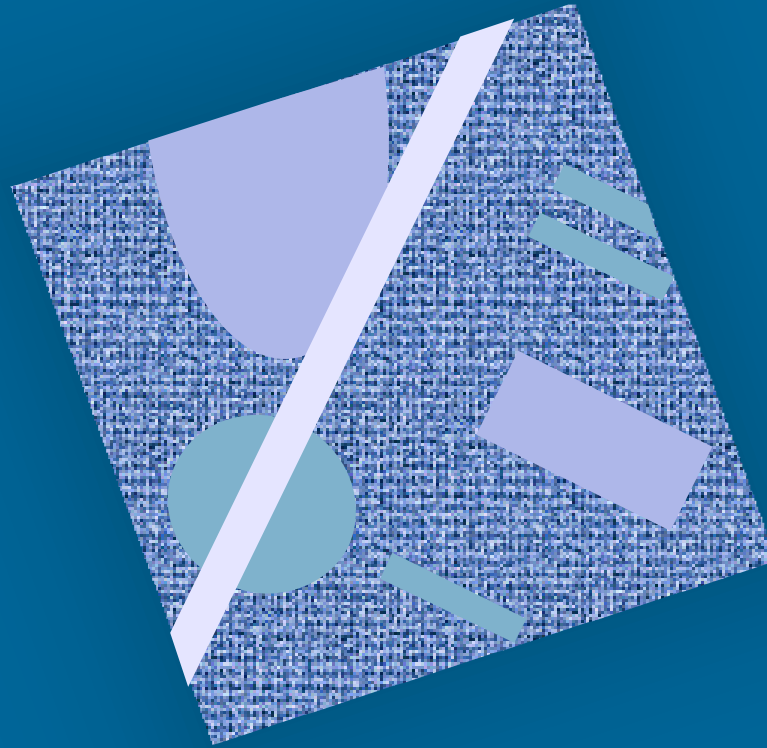


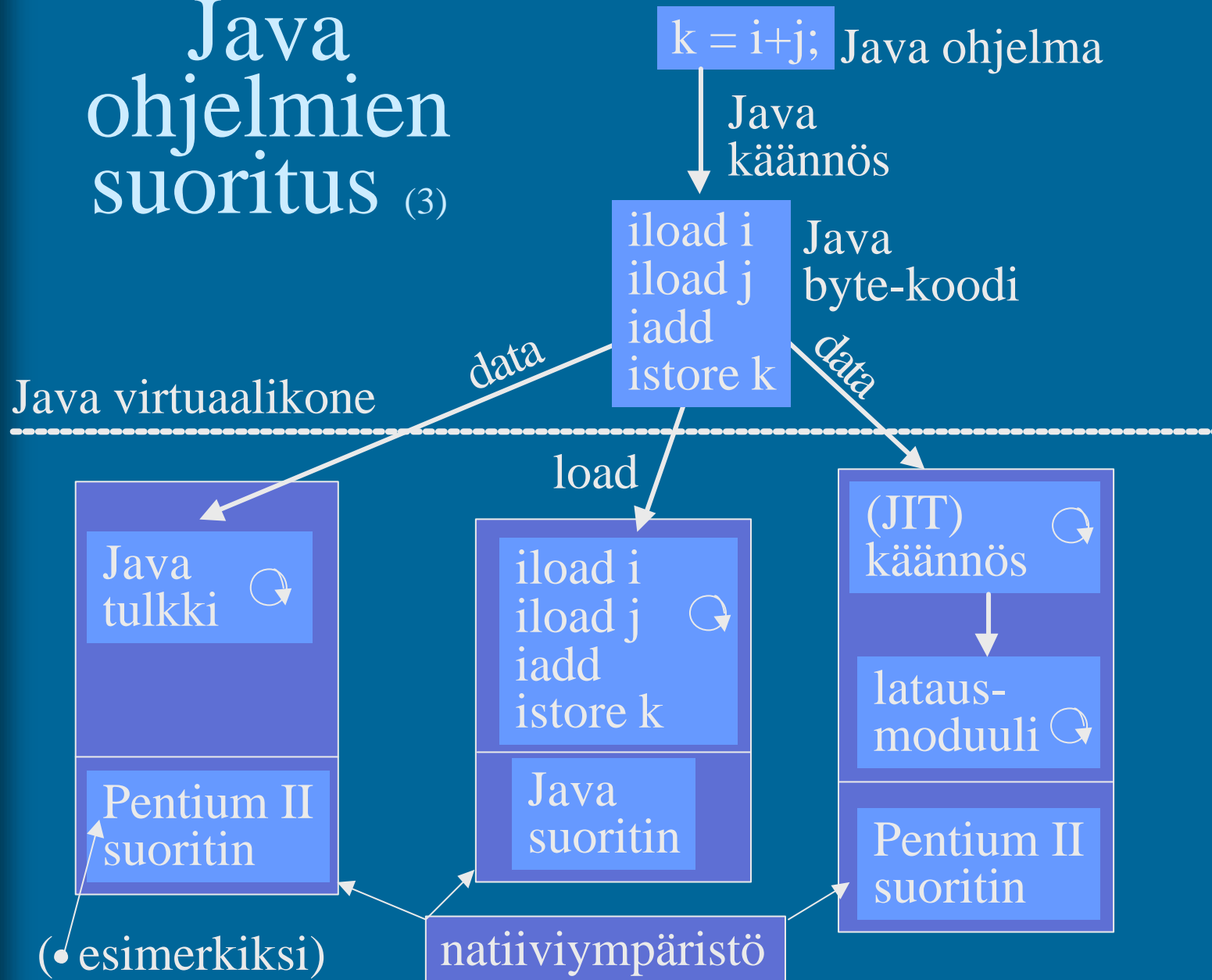
# Luento 11

## Tulkinta ja emulointi



Tulkinta ja emulointi  
Java ohjelman suoritus,  
tulkinta ja kääntäminen  
Suorittimen emulointi

# Java ohjelmien suoritus (3)



# Java virtuaalikone (JVM) (5)

- Hypoteettinen suoritin
- Geneerinen, sitä on ”helppo” simuloida kaikilla todellisilla suorittimilla
- Useita säikeitä (thread) voi olla samanaikaisesti suorituksessa
- Tietorakenteet
  - virtuaalikoneen suorittimen ”rekisterit”
- Käskyt
  - virtuaalikoneen suorittimen konekäskyt
  - 226 käskyä á 32 bittiä

# JVM:n tietorakenteet (8)

- JVM pino
  - vähän kuten tavallinen AT-pino
  - koostuu useista *raameista* tai *kehyksistä* (frames) (vrt. aktivointitietue)
  - ei tarvita yhtenäistä muistialuetta
  - allokoidaan natiivi-arkkitehtuurin kasasta (heap)
  - toteutuksesta riippuen rajallinen tai dynaamisesti laajennettavissa
  - tila loppu  $\Rightarrow$  StackOverflowError, OutOfMemoryError

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

# JVM:n tietorakenteet (jatkuu) (6)

- JVM kasa (JVM heap)
  - yhteinen kaikille saman virtuaalikoneen säikeille
  - automaattinen roskienkeruu (garbage collector)
    - ei-käytössä oleva muistialue palautetaan uusiokäyttöön
    - ei tarvita erikseen *free* operaatiota Java ohjelmassa
    - voi hidastaa suoritusta milloin vain
  - toteutuksesta riippuen kiinteän kokoinen tai dynaamisesti laajennettavissa
  - ei tarvitse muodostaa yhtenäistä muistialuetta natiivijärjestelmän kasassa
  - tila loppu  $\Rightarrow$  OutOfMemoryError

# JVM:n tietorakenteet (jatkuu) <sup>(6)</sup>

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

# JVM:n tietorakenteet (jatkuu) (7)

- Javan suoritusaikainen vakioallas (runtime constant pool)
  - joka luokalle (class) tai liittymälle (interface) omansa
  - suoritusaikainen esitystapa tiedoston *class constant\_pool* -taulukolle
  - vastaa vähän tavallista symbolitaulua
  - useita erilaisia vakioita (käännösaikaiset literaalit, suoritusaikana ratkottavat attribuutit, ...)
  - talletetaan JVM metodialueelle
  - tila loppu  $\Rightarrow$  OutOfMemoryError

# JVM:n tietorakenteet (jatkuu) (6)

- 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 tarvita JVM toteutuksissa, joissa ei voi ladata natiivimetodeja
  - toteutuksesta riippuen kiinteän kokoinen tai dynaamisesti laajennettavissa
  - tila loppu  $\Rightarrow$  StackOverflowError, OutOfMemoryError



# JVM:n tietorakenteet (jatkuu) <sup>(6)</sup>

ks. Fig. 4-10 [Tane99]

- 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

# JVM:n tietorakenteet (jatkuu) <sup>(6)</sup>

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

# JVM raamin data <sup>(8)</sup>

- Paikalliset muuttujat sisältävä taulukko
  - 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)

# JVM:n tiedon osoitusmoodit <sup>(4)</sup>

- Välitön operandi `iINC 2`  $\textcircled{34}$
- Indeksoitu `iINC`  $\textcircled{2}$  `34` tehollinen muistiosoite (LV) + 2
- Pino-osoitus `iADD` korvaa pinon kaksi päällä olevaa kokonaislukua niiden summalla
- Taulukko-osoitus pinon kautta `iALOAD` Korvaa pinon pinnalla olevat taulukon alkuosoite ja indeksi k.o. taulukon alkiolla

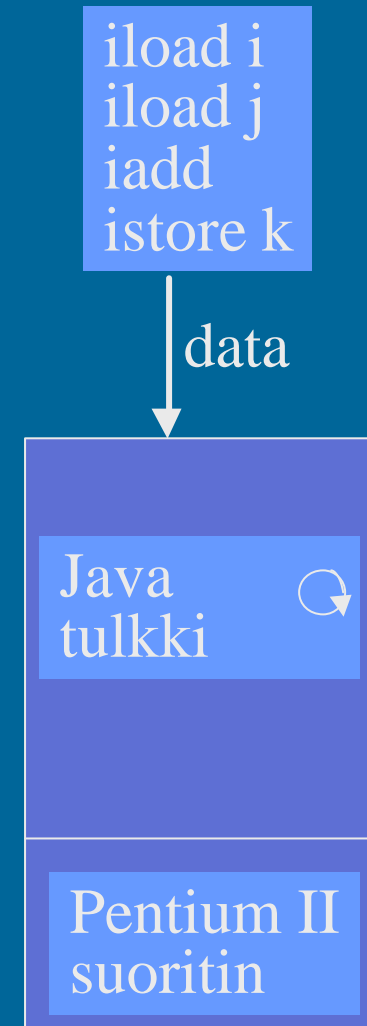
# JVM käskyt (7)

- 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. 5-36 [Tane99]

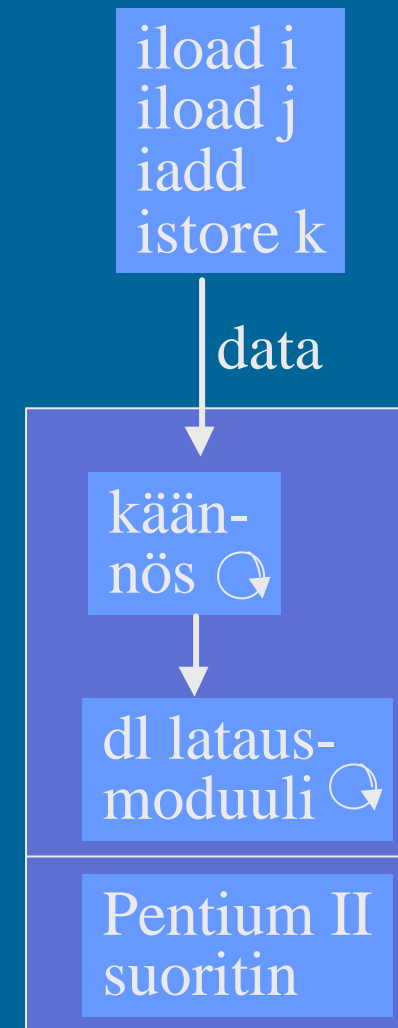
# Java tulkki (4)

- Emuloi JVM konekielen käskyjä (byte-koodia)
- Yksi käsky kerrallaan
- JVM rekisterit ja muistialueet emuloitu tulkin tietorakenteina
- Hidasta, mutta joustavaa



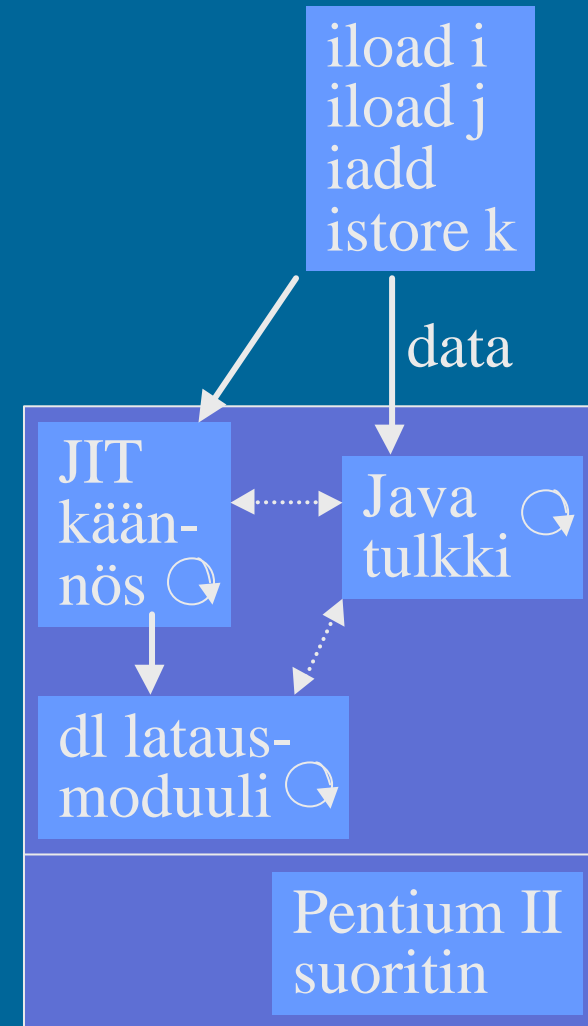
# Käännös natiivikoneelle (3)

- Käännetään tavukoodi natiivikoneen konekielelle, ja suoritetaan normaalin ohjelman tapaan
- Käännetään tavukoodi korkean tason kielelle, joka sitten käännetään natiivikoneen konekielelle
- ongelma: dynaaminen linkitys



# Java JIT käänнос (6)

- JIT = Just-in-Time
- Emuloi JVM konekielen käskyjä (byte-koodia)
- Kääntää luokan natiivikonekielelle dynaamisesti linkitettäväksi moduuliksi, juuri ennen luokan metodin kutsua
- Tarvitsee paljon muistia
- Voi hidastaa suoritusta, jos käännoökseen menee enemmän aikaa kuin tulkitsemiseen
- JVM rekisterit ja muistialueet emuloitu tulkin tietorakenteina





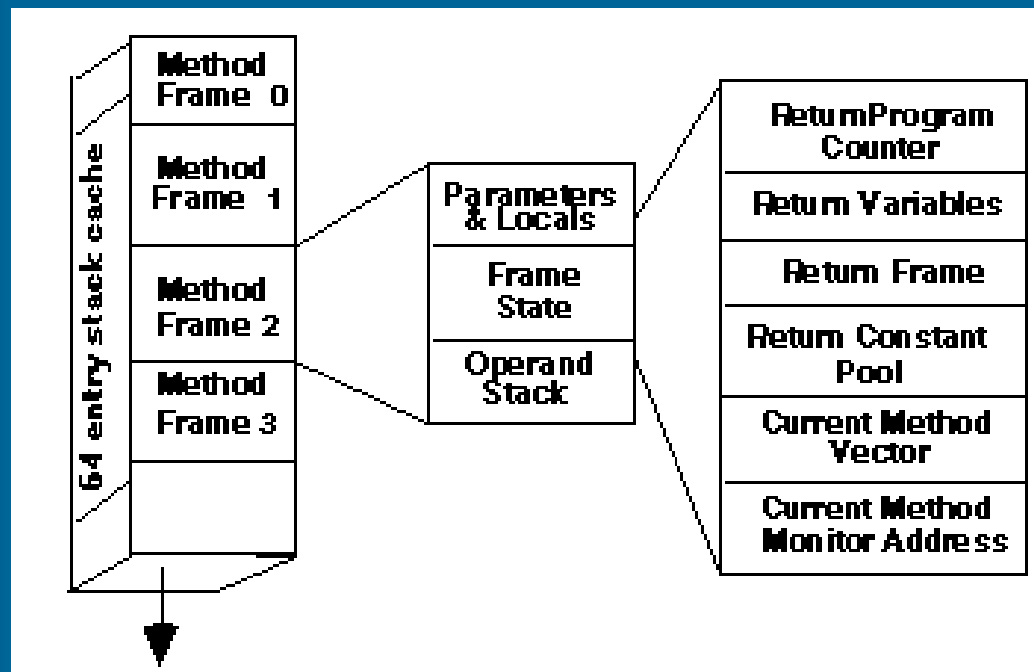
# Java suoritin:

## Sun PicoJAVA II <sup>(4)</sup>

- Suoritinmäärittely, jossa byte-koodi -muodossa olevia ohjelmia voidaan sellaisenaan suorittaa
- Valinnainen välimuisti ja liukulukusuoritin
- Kaikki 226 JVM konekäskyä
  - jotkut käskyt toteutettu aliohjelmilla, jotka aktivoidaan keskeytyskäsittelemekanismin avulla
- Myös 115 muuta konekäskyä käyttöjärjestelmän ja muiden ohjelmointikielten toteuttamiseksi

# PicoJAVA II pino

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



Shawn Lauzon,  
Survey of the JavaChip

<http://www.cs.berkeley.edu/~neefe/ntu.fa98/shawn.project/javaChip.html>

# PicoJAVA II rekisterit (11)

- 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 metodista paluu osoitteeseen
  - PSW
  - 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

# PicoJAVA ylim. käskyt (5)

- 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
  - tyhjä välimuisti (osittain? kokonaan?), ...
- Muut käskyt
  - power on/off, ...

# PicoJAVA toteutuksia (2)

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

# Muita Java suorittimia

- JEM (Rockwell Collins)
- PSC1000 (Patriot Scientific)
- MJ501 (LG Semicon)

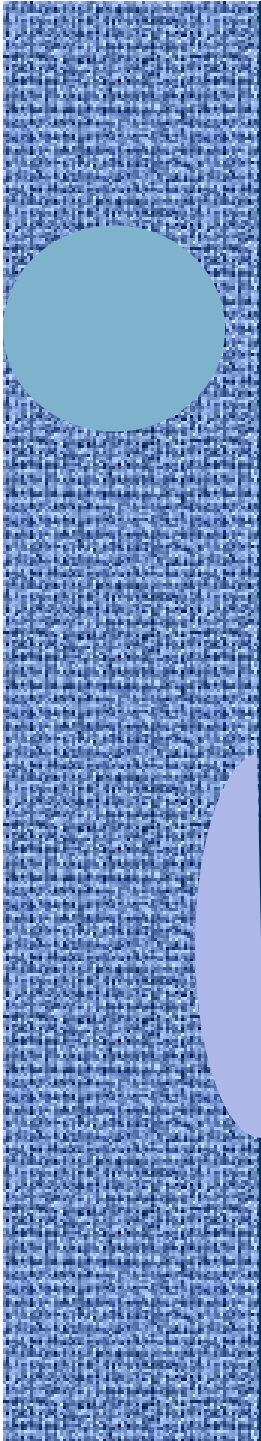
# Sun MAJC

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

# MAJC toteutus: MAJC 5200 <sup>(4)</sup>

- 1-4 suoritinta
- Useiden (peräkkäin kutsuttavien) metodien samanaikainen suoritus eri suorittimilla
  - ennakoiva (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!
  - välimuistin hudin aikana suoritetaan muita säikeitä
- Suunnattu interaktiiviseen TV:hen, virtuaalitodellisuussovelluksiin, ...





23.8.2000

Teemu Kerola, K2000

25

# TTK-91 Emulointi (5)

- TTK-91 konekielen emulointi
- KOKSI simulaattorin osa
- Yksi käsky kerrallaan
- TTK-91 koneen rekisterit ja muisti emuloitu tulkin tietorakenteina

ks. simulaattorin koodi

```
load R1, 234  
add R1, =5  
mul R1, R2
```

data

TTK-91  
emulaat-  
tori

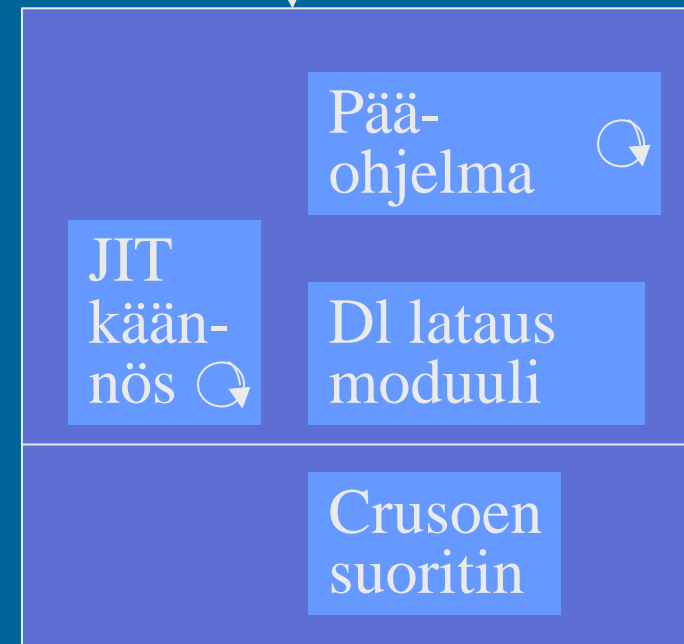
Pentium II  
suoritin

# Transmetan Crusoe suoritin (8)

- x86 konekielen emulointi, JIT käännös
- Natiivi käskykanta ei ole julkistettu
- ”nopeampi, sama teknologia”?
- ”yhtä nopea, vähemmän virtaa”
- Monta x86 käskyä yhtäaikaan, sikin sokin emuloinnissa
- x86 rekisterit emuloitu natiivijärjestelmän laiterekistereillä
- x86 muisti emuloitu suojaattuna tietorakenteina
- Tarkat keskeytykset:
  - suorituksen peruutus
  - uusi, hidas JIT käännös
  - hidas mutta tarkka emulointi

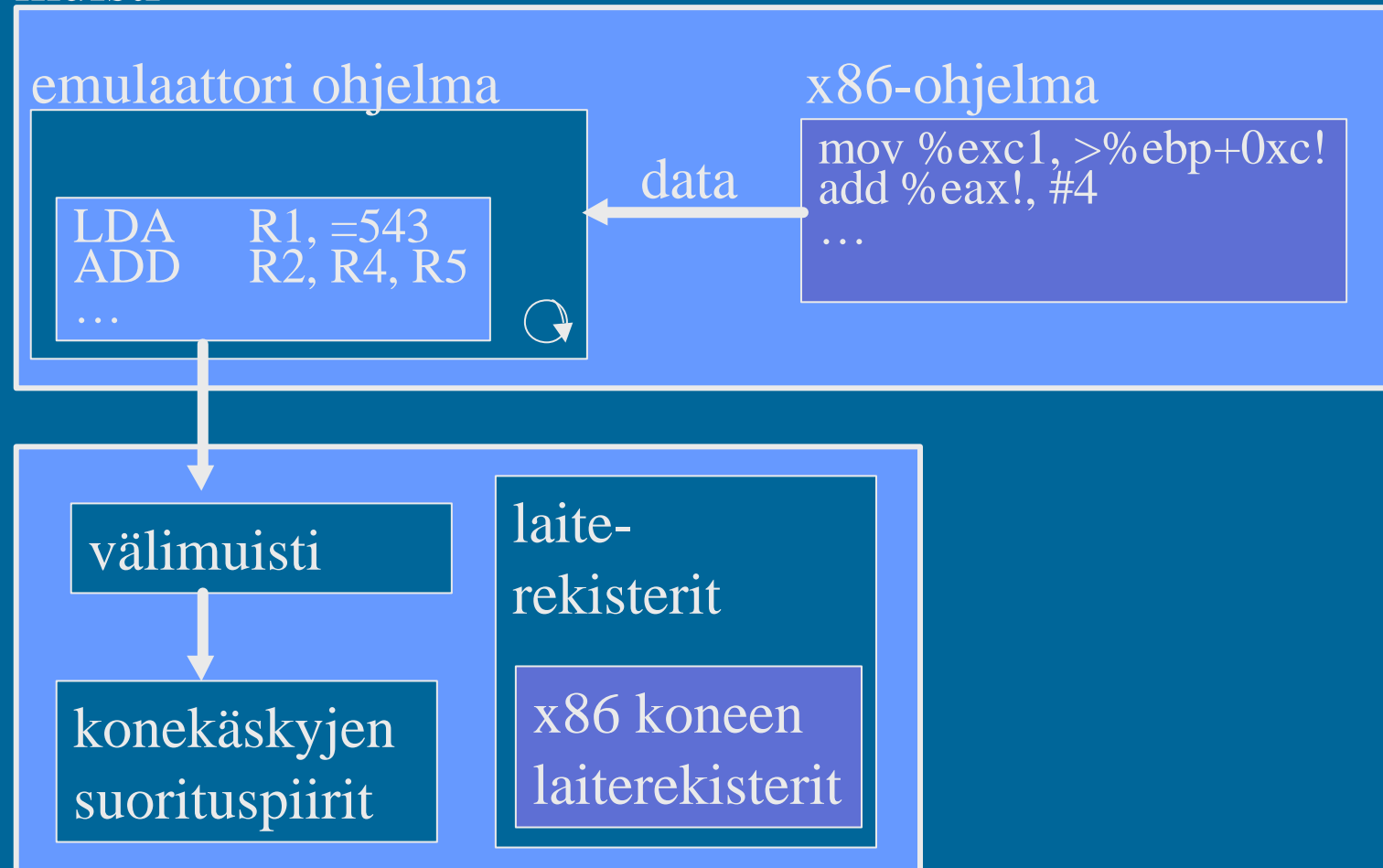
```
movl %esp, %ebp  
subl $4, %esp  
pushl %eax
```

data



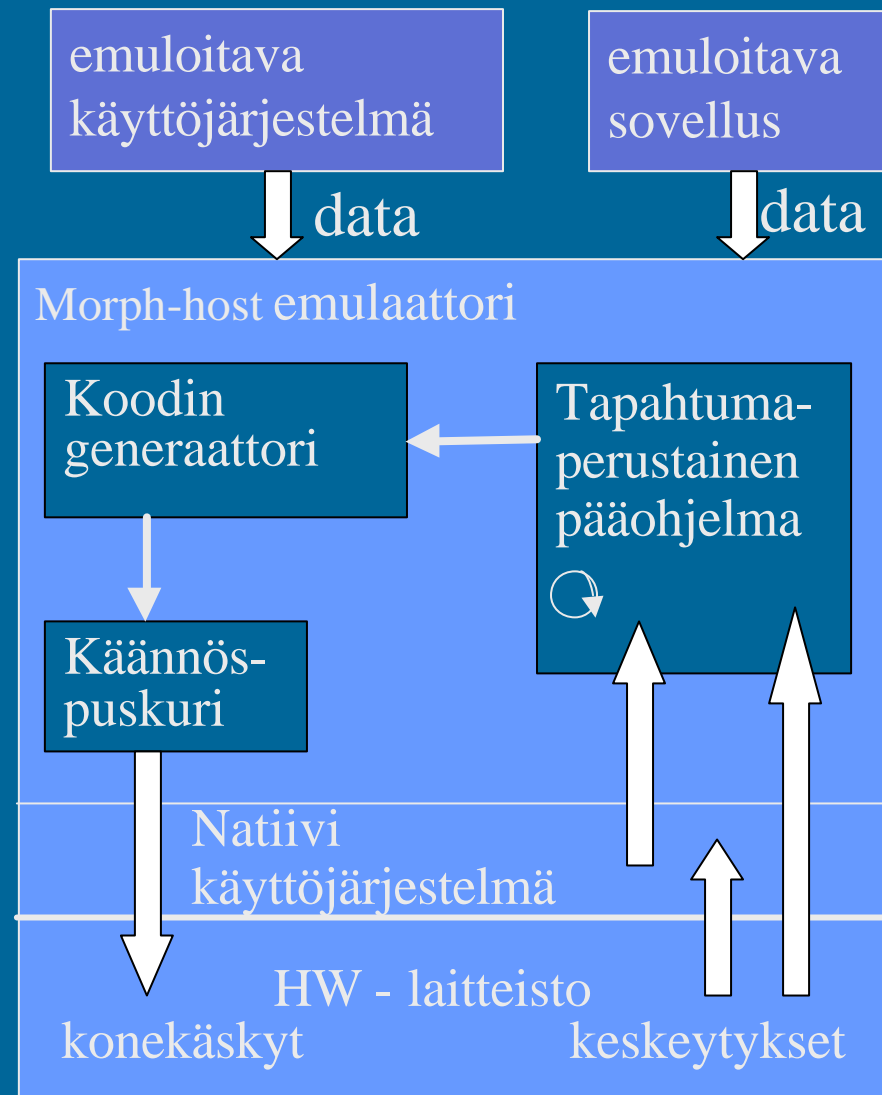
# Crusoe emulaattorin suoritus

muisti

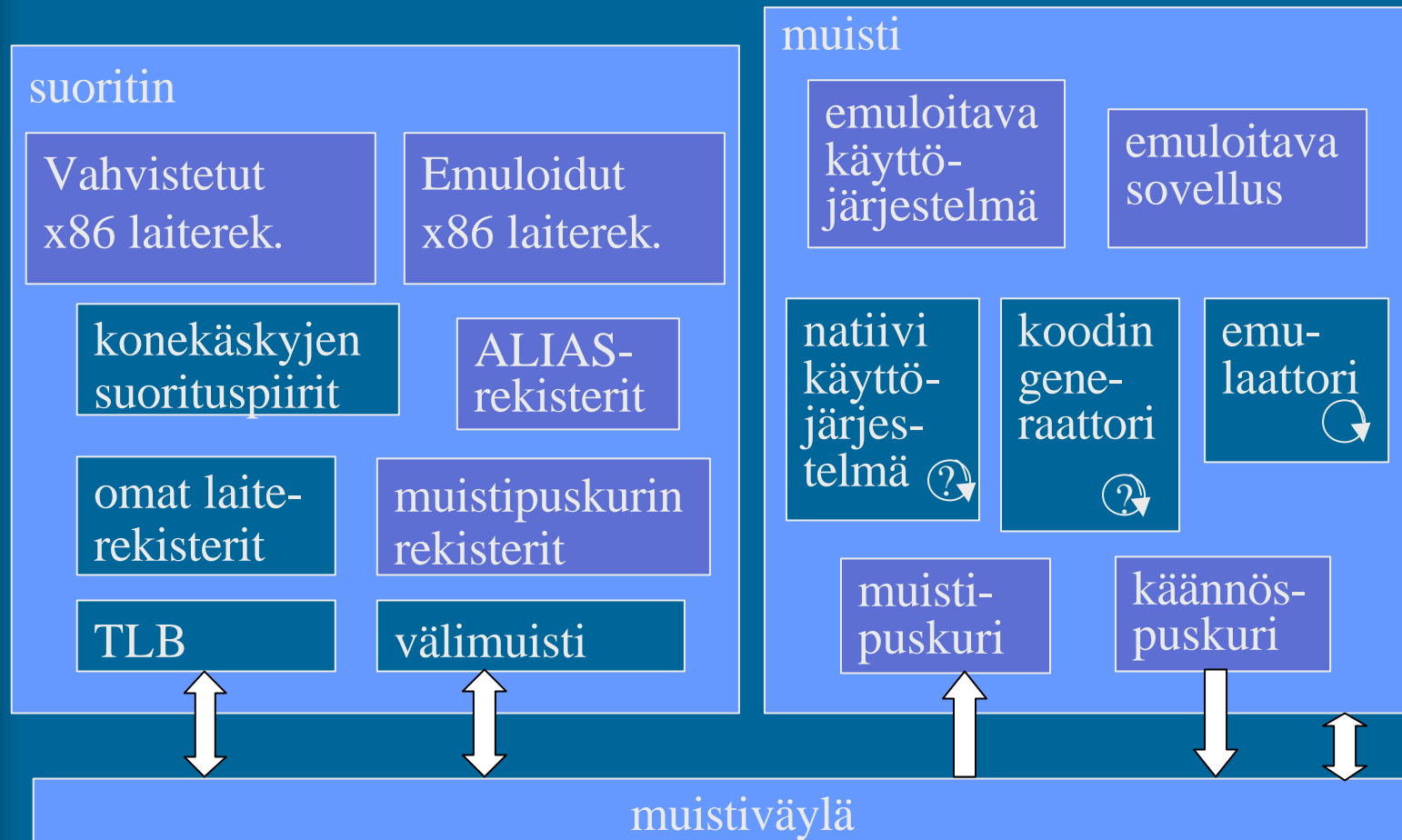


Crusoe suoritin

# Crusoe suorittimen looginen rakenne



# Crusoe suorittimen fyysinen rakenne



# -- Luennon 11 loppu --

