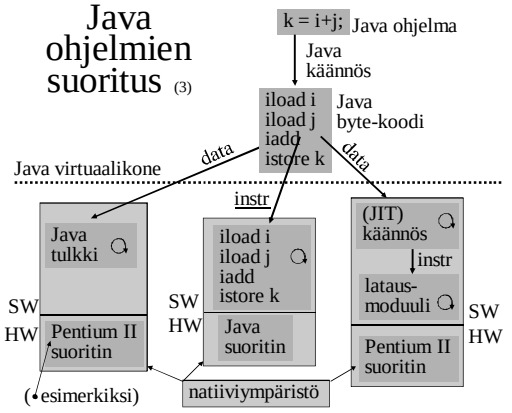
Luento 11
Tulkinta ja emulointi

Tulkinta ja emulointi
Java ohjelman suoritus, tulkinta ja kääntäminen
Suorittimen emulointi
C#, ttk-91, Crusoe

Java ohjelmien suoritus (3)



Java virtuaalikone (JVM) (5)

- 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
  - suorittimen mikroaikaskaalassa vain yksi kerrallaan
- Tietorakenteet
  - mm. virtuaalikoneen suorittimen ”rekisterit”
  - luodaan JVM:n käynnistämisen yhteydessä
- Käskyt
  - virtuaalikoneen suorittimen konekäskyt
  - 226 käskyä á 32 bittiä

JVM:n tietorakenteet

- JVM pino — ks. Fig. 4-10 [Tane99]
  - kuten tavallinen AT-pino
  - 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 ⇒ StackOverflowError, OutOfMemoryError

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

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?)
  - toteutuksesta riippuen kiinteän kokoinen tai dynaamisesti laajennettavissa
  - ei tarvitse muodostaa yhtenäistä muistialuetta natiivijärjestelmän keossa
  - tila loppu ⇒ OutOfMemoryError

JVM:n tietorakenteet (jatkuu)

ks. Fig. 4-10 [Tane99]

- 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 ⇒ OutOfMemoryError

JVM:n tietorakenteet (jatkuu)

ks. Fig. 4-10 [Tane99]

- 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 ⇒ OutOfMemoryError

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 ⇒ StackOverflowError, OutOfMemoryError

JVM:n tietorakenteet (jatkuu)

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)

ks. Figs 4-12, 4-13 [Tane99]

- 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

JVM kehyksen data (1)

- Paikalliset muuttujat sisältävä taulukko

ks. Fig. 4-13 [Tane99]

  - 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 (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 [Tane99]

- Taulukko-osoitus pinon kautta

aLOAD 1
iLOAD 2
iALOAD
iSTORE 3

Korvaa pinon pinnalla olevat taulukon alkuosoite ja indeksi k.o. taulukon alkiolla

Java: a = T[i];

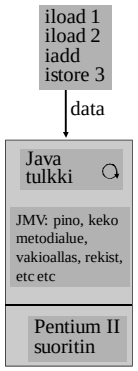
JVM käskyt

- 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. 5-36 [Tane99]

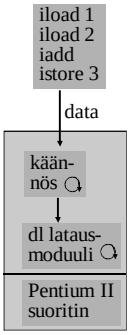
Java tulkki

- Emuloi JVM konekielen käskyjä (byte-koodia)
- Yksi (byte-koodi) käsky kerrallaan
- JVM rekisterit ja muistialueet emuloitu tulkin tietorakenteina muistissa
  - vrt. KOKSI ja TTK-91
- Hidasta, mutta joustavaa



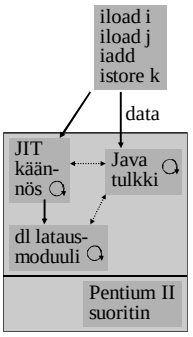
Käännös natiivikoneelle

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



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

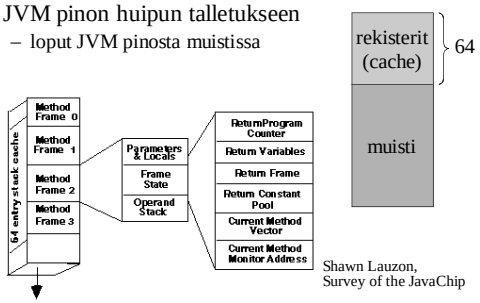


Java suoritin: Sun PicoJAVA II

- 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

PicoJAVA II pino

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



Shawn Lauzon, Survey of the JavaChip

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

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, ...

PicoJAVA toteutuksia (2)

- Sun microJAVA 701
  - valinnainen välimuisti
  - oma muistiväylä
  - PCI väylä muille laitteille
  - 16 ohjelmoitavaa I/O johdinta
    - näppäimet, LEDit, …
  - 3 ohjelmoitavaa ajastinta (⇒ 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)
  - 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

ks. aJ100-WRP kuva

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

- 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, ...

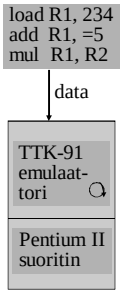
C# eli ”C sharp” (1)

- C#
  - Javan kaltainen kieli
  - kehittäjä: Anders Hejlsberg (Turbo Pascal, Delphi, J++)
  - osa Microsoftin .Net -ympäristöä
  - Mono – open source .Net for Linux www.go-mono.com
  - nivoutuu hyvin Windows XP:n kanssa
  - ECMA (European Computer Manufacturers Association) standardi (MS, HP ja Intel)
- MSIL – virtuaalikoneen konekieli
  - Microsoft Intermediate Language
  - sopiva ”välikieli” kaikille korkean tason kielille: C, C++, Pascal, Java, C#, Visual Basic
  - suoritus ainoastaan (JIT) käännösten avulla
  - CLR virtuaalikone (Common Language Runtime)

TTK-91 Emulointi

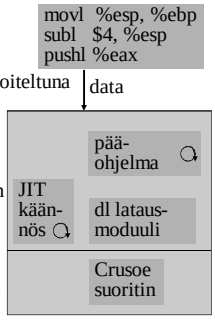
- TTK-91 konekielen emulointi
- KOKSI simulaattorin osa
- Yksi käsky kerrallaan
- TTK-91 koneen rekisterit ja muisti emuloitu tulkin tietorakenteina

ks. simulaattorin koodi, luento 5 (kurssikansio)

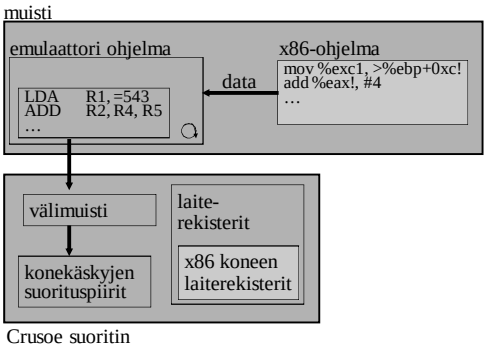


Transmetan Crusoe suoritin (8)

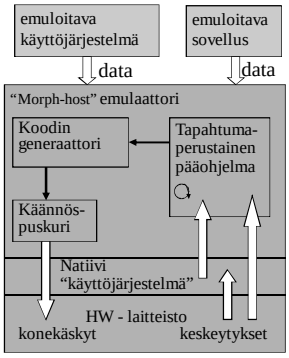
- x86 konekielen emulointi, JIT käännös
- Natiivi käskykantaa ei ole tärkeä
- ”nopeampi, sama teknologia”?
- ”yhtä nopea, vähemmän virtaa”
- Monta x86 käskyä yhtä aikaa paloiteltuna emuloinnissa, sekajärjestyksessä
- x86 rekisterit emuloitu natiivi-järjestelmän laiterekistereillä
- x86 muisti emuloitu rekistereiden avulla suojattuna tietorakenteina
- Tarkat keskeytykset:
  - suorituksen peruutus
  - uusi käännös hitaalle koodille
  - uusi hidas tarkka emulointi



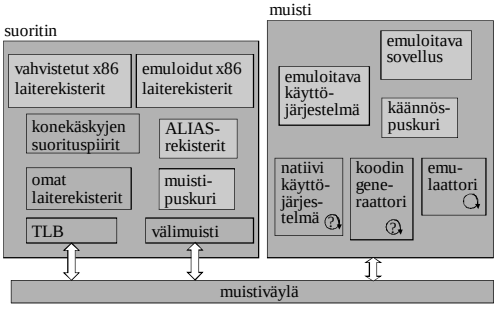
Crusoe emulaattorin suoritus



Crusoe suorittimen looginen rakenne



Crusoe suorittimen fyysinen rakenne



--

Luennon 11 loppu