Développement d'un générateur d'interpréteur de bytecodes pour une JVM embarquée

Mustapha Tachouct

Sous la direction de :

Alexandre Courbot

Moulay-Driss Benchiboun

Plan de la présentation

- I. Présentation du LIFL et de RD2P
  - LIFL : Laboratoire d'Informatique Fondamentale de Lille
  - RD2P : Recherche et Développement sur le Dossier Portable
- II. JITS : Java In The Small
  - Implémentations actuelles
  - Spécificité du projet JITS
- III. Sujet du stage
  - Bytecodes
  - Syntaxe & Utilisation
  - Générateur

I. Présentation du LIFL et de RD2P (½)

- Laboratoire d'Informatique Fondamentale de Lille
  - Créé en 1983
  - Unité de recherche
  - Associé au Dept. STIC du CNRS
  - 160 personnes
  - 3 axes : CBS, CIM et SCOPE

Présentation du LIFL et de RD2P (2/2)

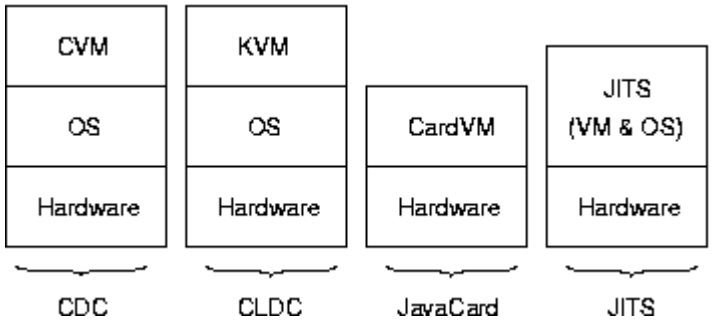
- Recherche et Développement de Dossier Portable
  - Gemplus et USTL
  - Axes de recherche :
    - Réseaux mobiles
    - Protection de l'information
    - Systèmes embarqués

Java dans les POPS (½)

- Implémentations actuelles
  - CDC :
    - PDAs haut de gamme
    - API quasi-complète
    - CVM
  - CLDC :
    - Téléphones portables, PDAs,...
    - Nouvelles restrictions (types, ...)
    - KVM
  - JavaCard :
    - Carte à processeur
    - Type codés sur 8/16 bits
    - CardVM

Java dans les POPS (2/2)

- JITS : Java In The Small
  - Faciliter le développement en Java
  - JVM + OS
  - Customizable
  - API, Garbage Collector, Green Threading, ...
  - Romizer



Générateur d'interpréteur

- Machine Virtuelle Java
- Bytecodes
  - Quelques exemples
- Syntaxe & utilisation
  - XML
  - Attributs, Modularité et sécurité
- Générateur d'interpréteur
  - Bytecodes typiques
- Optimisation de l'interpréteur
  - Plateforme, Compilateur et Résultat

JVM : Java Virtual Machine

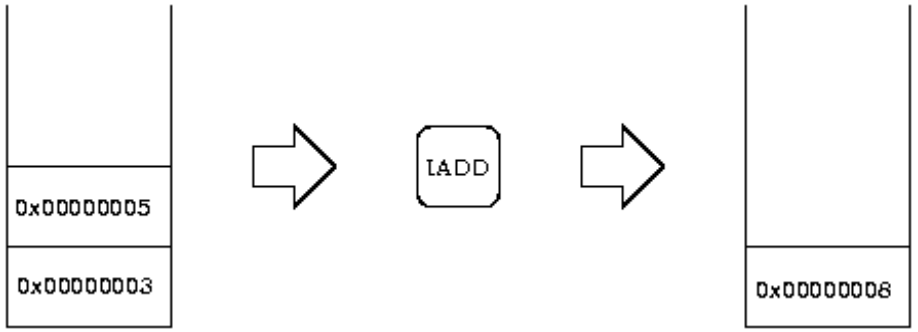
- Machine à pile
- Bytecodes ?
- API java
- Autre fonctionnalités
  - Garbage Collector
  - Green Threading
  - ClassLoader

Bytecodes : principe et fonctionnement

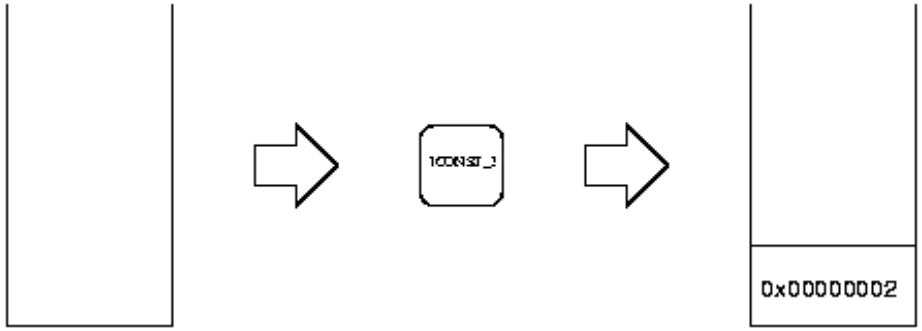
- Paramètres :
  - Sur le sommet de pile
  - Variables locales
  - Constantes directes
  - Constantes Pool
- Résultats :
  - Sur le sommet de pile
  - Modification d'une variables locales

Bytecodes : exemples

- Addition de deux entiers : *iadd*

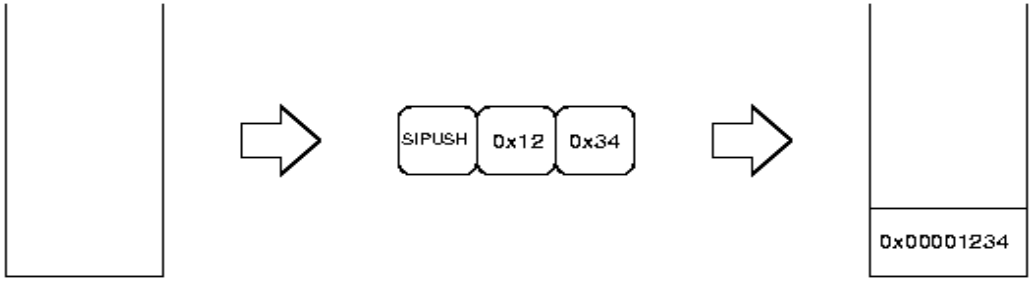


- Constante implicite : *iconst_2*

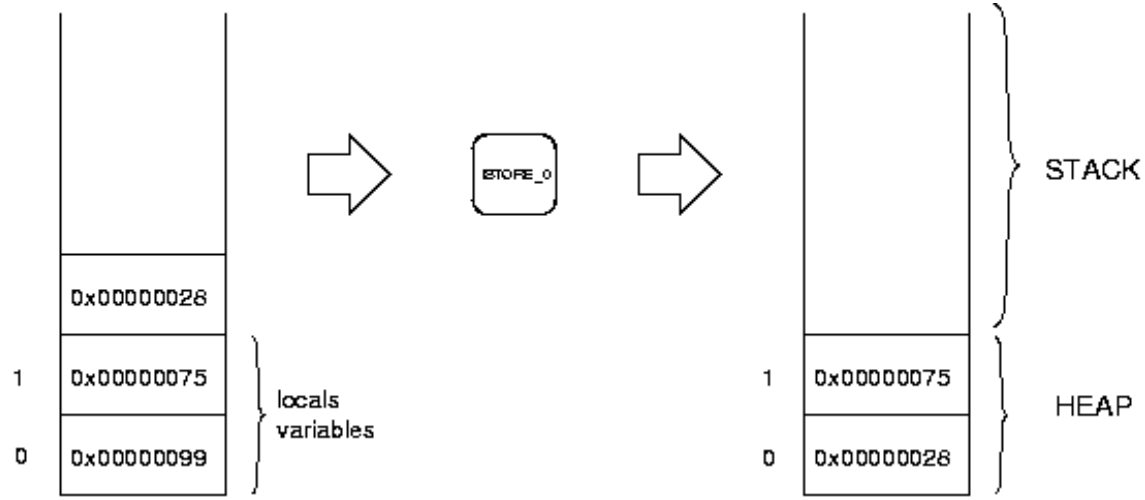


Bytecodes : exemples avec paramètres

- Constante explicite : *sipush*

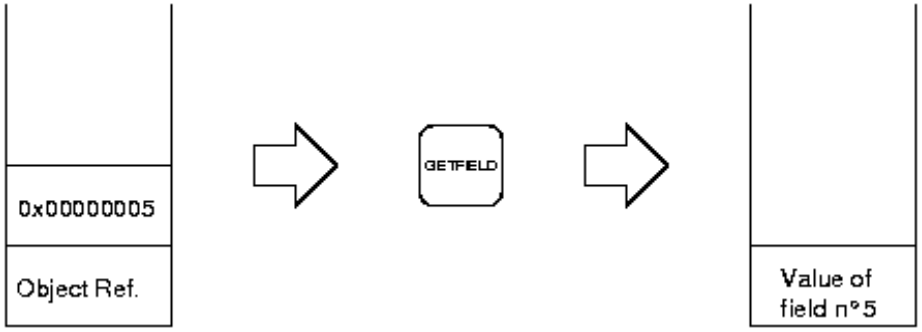


- Modification d'une variable locale : *istore_0*

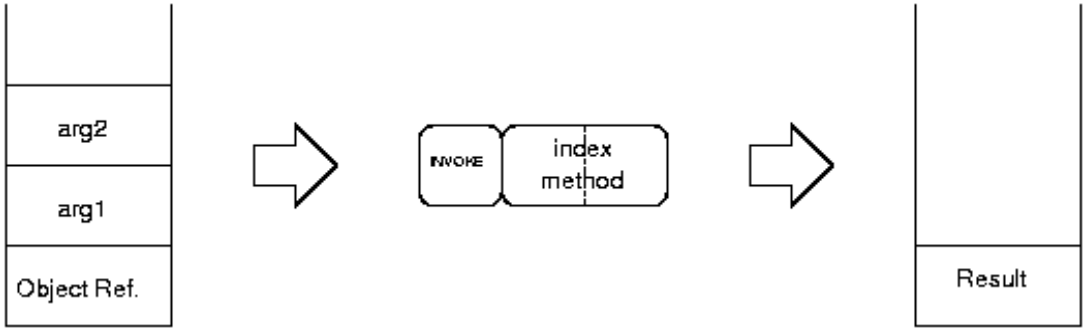


Bytecodes : exemples (objets)

- Obtenir la valeur d'un attribut d'un objet : *getfield*

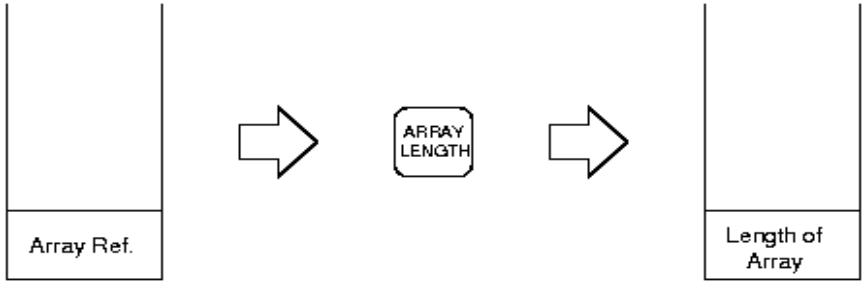


- Appeler une methode « non-static » : *invoke*

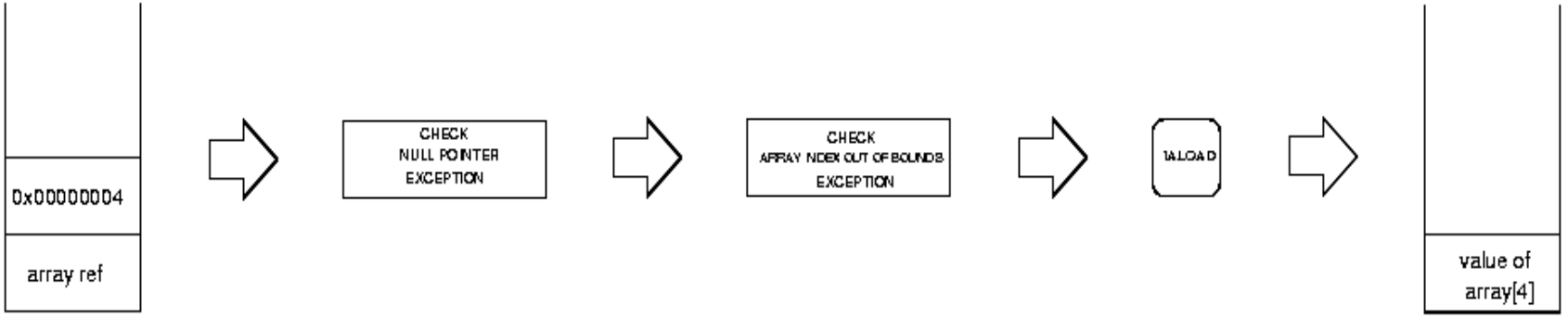


Bytecodes: exemples (tableaux)

- Obtenir la taille d'un tableau : *arraylength*

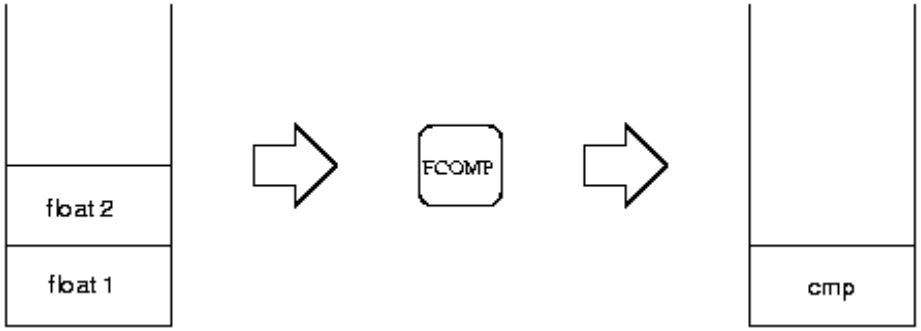


- Obtenir la valeur d'un élément : *iaload*

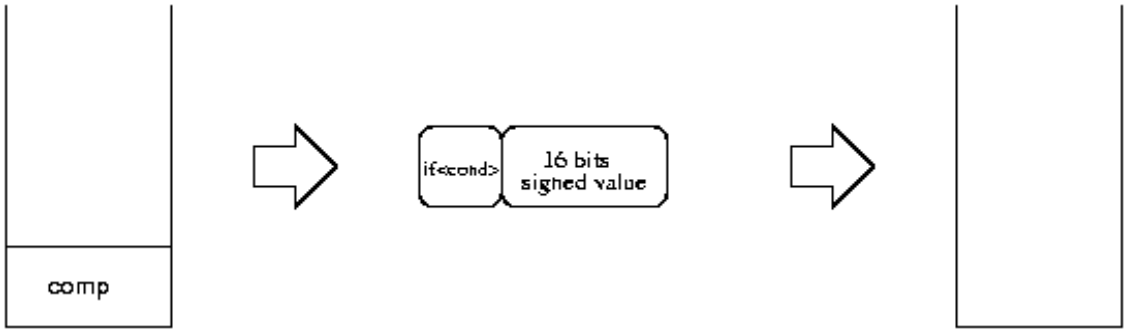


Bytecodes : sauts conditionnels

- Comparaison de deux nombres à virgules flottantes:

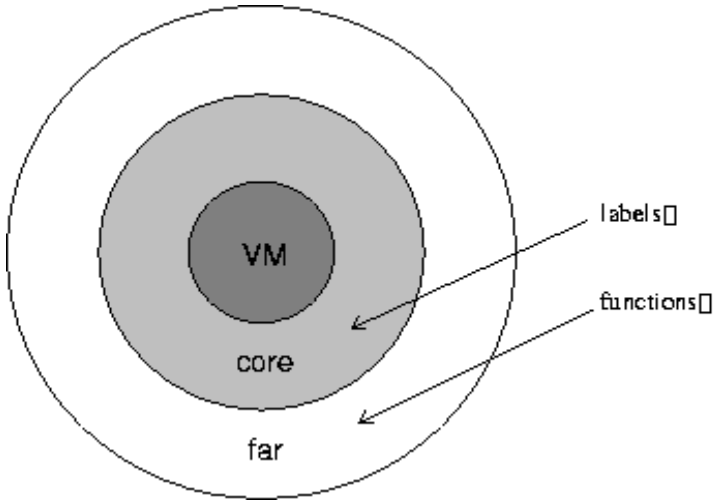


- Saut conditionnel : *if<cond>*



Modelisation des bytecodes

- XML : eXtensible Markup Langage
- Attributs
  - Bc, name, pop/push, inc_pc, inCore



Modelisation des bytecodes

- Modularité
  - Configuration de la VM
    - Personnaliser sa VM
    - Ajout ou suppression de fonctionnalités
    - <vmconfig> ...</vmconfig>
  - Dépendance des bytecodes
    - Supprimer les bytecodes inutiles
    - <require></require>

Modelisation des bytecodes

- Sécurité
  - Verifications
    - NullPointerException
    - ArrayIndexOutOfBoundsException
    - NegativeArraySizeException
    - ArithmeticException
    - ClassCastException
  - Arguments
    - Type : pile, constante directe, constante pool
    - <arg> <cst type= « short » index= « 1 »/> </arg>
    - <arg> <stack type= « Array » index= « 2 »/> </arg>

Générateur d'intepréteur de bytecodes

- Tableaux de fonctions et de labels(ou branchements)
- pile & incrémentation du pc
- Vérifications
- Exemple de syntaxe XML :

```
<bytecode bc= « 46 »  name= »iaload » pop= »2 » push = »1 » inCore= »1 »>
        <exception name= »NullPointerException »>
                <arg><stack type= »array » index= »2 »/></arg>
        </exception>
        <exception name= »ArrayIndexOutOfBoundsException »>
                <arg><stack type= »array » index= »2 »/></arg>
                <arg><stack type= »int »  index =»1 »/></arg>
        </exception>
</bytecodes>
```

Générateur d'intepréteur de bytecodes

- Source généré :

Exemple : *iaload*

```
DEF_BC(IALOAD)
    BEGIN
        CHECK_NULLPOINTEREXCEPTION(ATOS(2))
        CHECK_ARRAYINDEXOUTOFBOUNDSEXCEPTION(ATOS(2),ITOS(1))
        DEF_IALOAD
        TOS--;
    END
DEF_END
```

Optimisation de l'interpréteur

- Plateforme matérielle
  - Intel x86 32 bits (CISC)
  - ARM7TDMI (RISC)
- Compilateur
  - Gcc 3.2
  - Gcc 3.4
  - Icc (problème de compatibilité)
- Optimisation
  - Optimisation du compilateur
  - Optimisation manuelle en assembleur (x86 et ARM)

Optimisation de l'interpréteur

- Tests :
  - boucles de 1000000
  - Bytecodes variés (iinc, comparaison, saut, ...)

- Résultat :
  - Gcc 3.2 : 340 ms
  - Gcc 3.4 : 185 ms
  - Optimisation manuelle : 67ms
  - HotSpot (interpréteur) : 79 ms

CONCLUSION