# 軽いハードウェアによるJava高速化手法に関する研究

吉兼 寛 (210102)

北陸先端科学技術大学院大学 情報科学研究科

2004年2月13日

**キーワード:** Java，Java仮想機械，バイトコード，スタックアーキテクチャ.

## 1 はじめに

近年，組み込みシステム向けの言語としてマルチプラットホーム，ネットワーク親和性，安全性などの点でJava言語が注目されている．プログラムをネットワークからダウンロードして実行することも可能となってきており，Java 言語の処理機構を組み込んだ携帯端末や家電製品などへの応用が急速に進んでいる．

Java仮想機械はアプリケーションをインタプリタ形式で実行するため実行性能が低く，計算パワーが必要とされるアプリケーションの実行には問題がある．これを解決するために，JIT コンパイラ ，ホットスポット等の提案がなされているが，これらの処理のために必要なメモリ量の確保が困難な組み込み機器には不向きである．また，バイトコードをネイティブコードとして直接実行するJavaチップがあるが，Javaバイトコード以外のアプリケーションを実行することができない問題がある．

本研究では，Java仮想機械においてスタックやローカル変数がメモリ上に確保されていることに着目し，それらをCPU内でレジスタとして実現し高速化する方式を提案する．

## 2 Java仮想機械（JavaVM）

Java仮想機械の仕様においては，1）クラスファイルの読み込み，2）Javaのセマンティクスに従った正しい実行の二点だけが必須機能であり，比較的自由度が高い実装を可能としている．データ構造は主にメソッドエリア（クラス），ヒープ（インスタンス），およびスレッド（Javaスタック）の三種類に分類される．

**メソッドエリア（クラス）**

Javaソースコードをコンパイルして生成されるクラスファイルは，基本的にそのクラスの持つ情報のみを格納する仕様となっているメソッドエリアは，それらのクラスファイルの情報を格納する場所として定義されている．

**ヒープ（インスタンス）**

クラスは抽象化されたものであり，実体化されたものをインスタンスと呼ぶ．Java 仮想機械はインスタンスの状態を保持するためのデータ領域をヒープ領域として確保する．

**スレッド（Java スタック）**

スレッドのデータ構造は Java スタック（pc レジスタおよびフレームを格納）として構成される．pc レジスタはカレントメソッドで現在実行されているバイトコード令を指すポインタであり，フレームはメソッド実行に必要なオペランドスタックとローカル変数を格納する領域である．最初のメソッドを読み出した時点で一つのフレームが生成され，更にネストしてメソッドを呼び出す度に新たなフレームが生成される．

## 3 Java 専用命令

本節では，通常の Java 仮想機械においてフレーム内のオペランドスタックとローカル変数がメモリ上に確保されるのに対し，それらを CPU 内にレジスタ（Java 用レジスタ）として実現する手法を提案する．実現のためには，これらの Java 用レジスタを Java 仮想機械がアクセスする手段が必要となる．その手段として MIPS の実装依存命令を利用して Java 用レジスタにアクセスする Java 専用命令を提供する．これらの Java 専用命令と，スタック，ローカル変数をアクセスするバイトコード命令との対応を確立する．これにより命令の最適化が行なわれる．例えば，バイトコード命令の iadd はオペランドスタック上の二つの値を読み出し，それらを足し合わせ，結果をオペランドスタックに格納する命令である．従来の Java 仮想機械では，以下のように実行されるとする．

```
lw   $r1,  0($op_top)      #第一オペランド
lw   $r2, -4($op_top)      #第二オペランド
add  $r3, $r1, $r2         #加算
sw   $r3, -4($op_top)      #結果格納
add  $op_top, $op_top, -4 #スタックポインタ更新
```

提案する手法では，次のような Java 専用命令により 1 命令で同様の操作が可能となる．

```
Jiadd $op_stack1, $op_stack0, $op_stack1
```

## 4 評価

提案手法を実装したシミュレータで評価をおこなう．シミュレータはパイプライン方式で，2 次キャッシュ/遅延スロットを考慮しない．また，入力は Java 仮想機械をコンパイルしたアセンブラファイルとした．評価対象はループを多用する Java ソースファイルである．

評価結果から，従来手法と比べ最大で局所的な命令の最適化は1350%，総サイクル数は43．38%の性能向上を得られた．また，追加したハードウェア量も全体として十分に小さいものであり，組み込み機器への応用が可能である．

## 5　まとめ

本研究ではJava仮想機械のローカル変数，スタックをレジスタとして用意し，命令数の削減を行ない高速化を実現する手法を提案した．従来のCPU にJava用のレジスタとJava専用命令のデコード／実行機構を追加するのみで高速化を実現し，メモリ制約の厳しい組み込み機器への応用が可能である．