修 士 論 文

Java 仮想機械の形式仕様とその検証

指導教官　二木厚吉 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻
言語設計学講座

奥村 滋
学籍番号：810024

2000 年 2 月 15 日

要 旨

Java 言語のセキュリティーモデルにおいて、バイトコード検証系はセキュリーティーを保つために信頼性が求めらるものである。しかし、バイトコード検証系の仕様が自然言語が自然言語を用いた曖昧な形でしか記述されていない。そこで本稿では、公開されている仕様書に基づきバイトコード検証系を形式仕様言語 CafeOBJ により記述し、その仕様を用いてバイトコード検証系の検証を行なう技法について報告をする。

目 次

第 1 章

はじめに

Java 言語は、Java のクラスファイルを Java 仮想機械上で解釈することで実行している。そして、Java のクラスファイルというのは、通常 Java 言語のソースコードから Java のバイトコンパイラにより生成される。そこで、Java 言語の使用形態を考えてみると、ネットワークを介する使い方ができるためクラスファイルが悪意を持った者によって改竄されたクラスファイルであったり欠陥のあるバイトコンパイラによって生成されたクラスファイルである可能性があり、その改竄されたクラスファイルをそのまま Java 仮想機械上で実行してしまうと実行自体が破綻したり、最悪の場合セキュリティー上の問題になることが考えられる。

そこで、Java 仮想機械ではクラスファイルを実際に実行する前に実行するクラスファイルの構造やクラスファイルの内容から実行しても破綻しないクラスファイルであるかどうか検査する。特に、クラスファイルに埋め込まれた命令列を見てデータフロー解析をする機構のことをバイトコード検証系と呼ぶ。

しかし、このバイトコード検証系の仕様が自然言語で記述された曖昧なものでしかない。そこで本稿では、公開されている仕様書に基づきバイトコード検証系を形式仕様記述の一つである代数仕様記述言語 CafeOBJ により記述する。

一般に、ソフトウェアを構築する際、ターゲットとしているシステムを絵などを用いて表現し、自分の中でどのように設計するかをまとめ、その後プログラムのコーディングを始めるという方法が考えられるが、その方法であると、自分で思い描いた設計に曖昧な部分や間違いに気が付くのはゴーディング途中やコーディングを終えた後に実行してから気が付く場合が少なくないと思われる。もちろん、実行し、プログラムの間違いを発見し、デバッガー等でその修正しなくてはいけない個所を発見し、修正するという方法ももちろん重要である。しかし、システムの設計段階で形式的な仕様を記述する道具を用いて設計の間違いを発見できるとすれば、設計の間違いであるのか実装の間違いであるのかが明確になるため有用であると考えられる。

本稿に関しては、実際に動いている Java 仮想機械の一部であるバイトコード検証系について形式仕様を記述しようとしているために、設計段階において形式仕様を記述することと意味合いが違うわけだが、

現状のバイトコード検証系が狙いどうりの設計になっているのかを調べることや、同様の network を介した使用形態の言語の設計をするときの参考にもなると考えられる。

第 2 章

CafeOBJについて

本章では、目標としているシステムを形式的に記述する道具である、形式仕様と CafeOBJ について説明する。

2.1 形式手法

形式手法とは、ソフトウェアの仕様作成から最終的なコーディングまでを、形式的な作業に基づいて進めて行く方法である。形式手法で一番の要となるのは形式仕様の作成である。形式仕様とは、数学を基礎としての意味の裏付けが確立された言語である形式仕様記述言語を用いて作成された仕様の事である。代表的な形式手法では、形式仕様記述言語を用いて記述された抽象度の高い仕様をもとに形式的な段階的詳細化を繰り返し適用して抽象度を下げていき、最終的にはプログラムコードを得るおいう過程で構成される。この作業は、理想的には全て形式的に行われるのが望ましいが、技術的な問題とそれにかかるコストの問題により部分的に非形式的な手法を導入することになる。また、形式手法をソフトウェア開発の工程のある部分に対して適用するということも行われる。形式手法はソフトウェアの信頼性の向上に極めて効果があるのだが、仕様作成や検証にかかるコストはソフトウェア開発の全体と較べても小さいとは言えない。しかし、ソフトウェア開発の上流工程でのバグにより引き起されるコストの増大は、ソフトウェアの規模が大きく、信頼性が求められるものであればあるほど大きくなる。つまり、形式手法の採用は、ある程度の規模を持つソフトウェアの開発に対しては、コストの面においても有効であると考えられる。

2.2 形式仕様

形式仕様とは、形式仕様記述言語を用いて作成された仕様のことである。形式仕様記述言語は数学を基礎として意味が確立されているという点で通常のログラミング言語と区別される。また、通常のプログラミング言語より抽象度の高い記述が可能である。形式仕様に求められる性質には以下のものが考えられる。

- 厳密性 …… 仕様に曖昧な部分がない。
- 非冗長性 …… 仕様に冗長な部分がなり。
- 健全性 …… 仕様から検証できることは常に正しい。

また、形式仕様を作成するためのアプローチには、代数的記述法、公理的仕様記述法、操作的仕様記述法がある。公理的記述法を用いたものには Z や VDM、操作的仕様記述法を用いたものには、Estell、そして、代数的仕様記述法を用いている、CafeOBJ、OBJ、Maude がある。

2.3 CafeOBJ と代数仕様

この節では、CafeOBJ について説明する。CafeOBJ は、代数仕様記述言語 OBJ の流れを汲む実行可能な仕様記述言語である。CafeOBJ は複数の論理の組合せを仕様の意味モデルとして持つことができるという特徴がある。CafeOBJ が仕様の意味モデルとしてもつ論理として、多ソート代数 (many sorted algebra)、順序ソート代数 (order sorted algebra)、隠蔽代数 (hidden algebra)、書換え論理 (rewriting logic) である。これら複数の論理を任意の組合せにより開発対象のシステムに応じた意味モデルの選択が可能である。一方、CafeOBJ は項書換えシステムをその操作的意味論として持っている。したがって、等式論理、書換え論理の推論規則を使って定理の証明を行うことができる。これは、CafeOBJ が仕様の構文と意味を厳密に定める能力のほか、仕様を検証する機能があることを意味する。

2.4 等式仕様と代数、抽象データ型

この節では、データオブジェクトのクラス、すなわち抽象データ型を記述するための CafeOBJ 構文と意味モデルについて説明する。

システムのモジュール化を行う方法の一つとして、抽象データ型に基く方法がある。抽象データ型とは、データ構造を実現方法によって記述するのではなく、そのデータ構造が提供するサービスのリストとそのサービスの特性を示すことによってデータ構造のクラスを記述する手段である。抽象データ型は、あるモジュールは単一のデータ抽象のみを扱い、そのサービスはそのデータ型のみを操作するべきであるという考え型に基づいている。このアプローチはデータのカプセル化 (data encapsulation) と呼ばれている。CafeOBJ は、抽象データ型の仕様に対してその仕様に対応する始代数の同型なクラスをその意味モデルとして与える。

たとえば、自然数のクラスを等式仕様で記述すると、以下のようになる。

```
module! NAT {
```

```
  [ Nat ]
op 0   : -> Nat
op s_  : Nat -> Nat
op _+_ : Nat Nat -> Nat
vars M N : Nat
eq M + 0 = M .
eq M + s N = s(M + N) .
}
```

CafeOBJ では、仕様はモジュールを単位として記述される。module はモジュールの宣言の始まりを意味し、これに続く NAT はこのモジュールの名前である。“[” と “]” で囲まれた Nat がソート名であり、型を表す。op で演算子を宣言する。ここで演算子のアリティ (arity : 引数ソートの列) は ->の前に、演算子のコアリティ (coarity : 結果ソート) は->の後ろに宣言される。また、vars で変数、eq で等式を宣言している。

この等式仕様に対して、さまざまな代数 (集合と集合上の演算との組) を考えることができる。特に、項の同値類で構成される代数のことを Σ 代数と呼ぶ。等式仕様の意味論には、Σ 代数をモデルとする始代数意味論 (initial semantics あるいわ tight semantics) と、全代数をモデルとする緩い意味論 (loose semantics) がある。モジュール宣言の始めにある module! における “!” は、この等式仕様を始代数意味論で解釈することを宣言する。もう一方の緩い意味論を用いて仕様を解釈するということを宣言するには、module* と宣言し、記号 “*” を用いる。

2.5 振舞仕様と隠蔽代数、抽象機械

CafeOBJ はシステムオブジェクトのクラスである抽象機械の仕様に対して、その仕様に対応する隠蔽代数のクラスのを意味モデルとして与えることができる。隠蔽代数ではデータのための「可視」ソート (visible sort) と状態のために「隠蔽」ソート (hidden sort) を明示的に区別する。そして、可視の「属性」(attributes) と不可視の「内部状態」を変更するための「操作」(method) を隠蔽代数は持つ。このような隠蔽代数で意味定義された仕様のことを「振舞仕様」(behavioural specification) と呼ぶ。したがって、振舞仕様は、観測されるデータを記述する可視部とシステム振舞を記述する隠蔽部とに分けられる。前述のシステムオブジェクトの状態を観測する演算である属性、状態を変更するための演算である操作、および初期状態を意味する演算である隠蔽定数は、まとめて「振舞演算」(behavioural operators) と呼ぶ。

ここでは、バイトコード検証系の一部となるオペランドスタックを振舞仕様の形で表現した。

```
module* STACK (Y :: TRIV) {
```

```
*[ Stack ]*
op  null : -> Stack             -- initialize
bop push : Elt.Y Stack -> Stack -- method
bop pop_ : Stack -> Stack       -- method
bop top_ : Stack -> Elt.Y       -- observation
var S : Stack
var E : Elt.Y
beq top (push (E, S)) = E .
beq pop (push (E, S)) = S .
}
```

モジュール名 STACK の次に記述してある（Y :: TRIV）は、このモジュールがパラメータ化モジュールであること宣言している。実際にスタック自身の仕様を記述するときには、スタックに積まれるものには依存していないことを記述するために、このように記述している。“*[” と “]*” で囲まれた Stack が隠蔽ソートで、スタックオブジェクトのクラス名である、意味モデル上ではこのシステムの状態集合と解釈する。op で隠蔽定数を宣言する。演算 null はスタックの初期状態をあらわす「隠蔽定数」(hidden constants) である。bop で振舞演算を宣言する。演算 push,pop は、スタックの状態を変更する操作、演算 top は、スタックが一番上に積んでいるデータの値を示す属性である。

システムオブジェクトの状態は、観測によってしか知ることができない。したがって、あるシステムオブジェクトに対してある状態が別の状態と等しいというのは観測される値すべてを比較しても区別をつけられないことである。

システムオブジェクトに対する操作を $m_1, ..., m_n$ 、属性を att とすると、観測は、att $m_n \cdots m_1 \Box$ ($\Box$ は、観測対象) と表現される。この属性から始まり操作が続く列、att $m_n \cdots m_1$ を観測文脈 (context) と呼ぶ。 観測文脈によりシステムオブジェクトの状態上の等価関係 “∼” を次のように定義する。s,s’ をシステムオブジェクトの状態とするとき、s ∼ s’ iff $\wedge_{oc \in AllOc}(oc[s] = oc[s'])$。ここで、ALLOc は、観測文脈の全体集合を表す。なお、この “∼” のことを振舞等価と呼ぶ。

以上が、形式仕様と CafeOBJ についての説明である。次章では、形式仕様を記述する目標である Java 仮想機械についての説明をする。

第 3 章

Java仮想機械

本章では、Java 仮想機械がどのようなしくみで Java のバイトコードを実行しているかを説明する。

3.1 メソッドエリアとヒープ

Java 仮想機械は、クラスファイルを読み込むとクラスをメモリに展開する。メモリ中のクラスの置き場所をメソッドエリア (method area) と呼ぶ。そのクラスのインスタンス (実体) を生成すると、そのインスタンスはヒープ (heap) という場所に置かれる。(ただし、実装によっては、ヒープの中にメソッドエリアを割り当てる場合もある。)

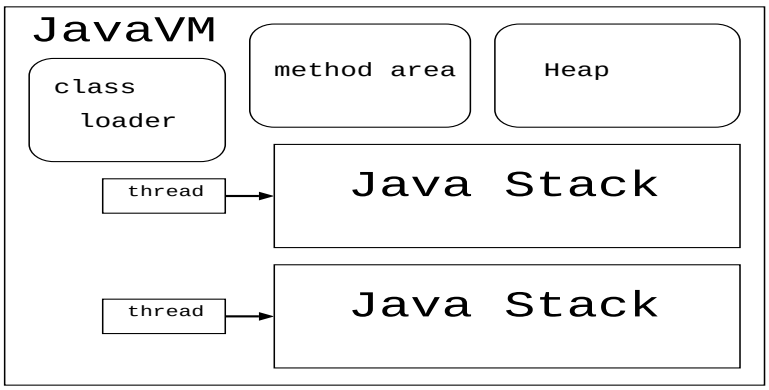


図 3.1: Java 仮想機械のメモリ構成

図 3.2: run() を実行したおきにスレッド A の Java スタック

3.2 Java スタックとフレーム

次に、figure3.1 を見ると、メソッドエリアとヒープの他に、Java スタック (Java Stack) というメモリ領域が Java 言語の処理単位であるスレッド毎に割り当てられている。Java スタックはコールスタックと呼ばれることもある。それは、メソッドがコールされるたびにフレーム (frame) が積まれるスタックであることが理由である。

スレッドが 1 つ生成されると、そのスレッドの Java スタックが 1 つ生成される。スレッドと Java スタックの関係は常に 1 対 1 になっている。Java スタックは、フレームを積み上げていくことになるが、その積まれるフレームは各メソッドの作業用のメモリ領域となる。

ここで、簡単な例を上げて Java スタックの動きについて説明する。Java のスレッド (これをスレッド A と呼ぶことにする) が run() というメソッドを実行したとする。そうすると、フレームが 1 つ作られ (これをフレーム X と呼ぶことにする)、スレッド A の Java スタックに積まれる。3.2

このとき、run() の中で、method1() というメソッドが呼び出されたとする。すると、メモリに method1() 用の領域が取ることが可能ならば method1() 用のフレームが作られ、method1() に割り当てられる。(これをフレーム Y とする。) すると、スレッド A の Java スタックの状態は、figure3.3 のようになる。また、この図 3.3 からフレームはメソッド毎に割り当てるのではなく、メソッドの呼び出し毎に割り当てられていることがわかる。

そして、method1() の実行が終了すると、フレーム Y は破棄され、Java スタック上には何も積まれていない状態となる。つまり、フレームはメソッドの実行開始と共に生成され、メソッドの実行終了と共に破棄される。スタックに積まれたフレームが全て破棄されるとそのスレッドは終了となる。

また、Java スタックの一番上に積まれたフレームのメソッドのことを、カレントスレッドと呼び。カレントメソッドの所属するクラスを、カレントクラスと呼び。そして、スレッドが実際に実行しているの

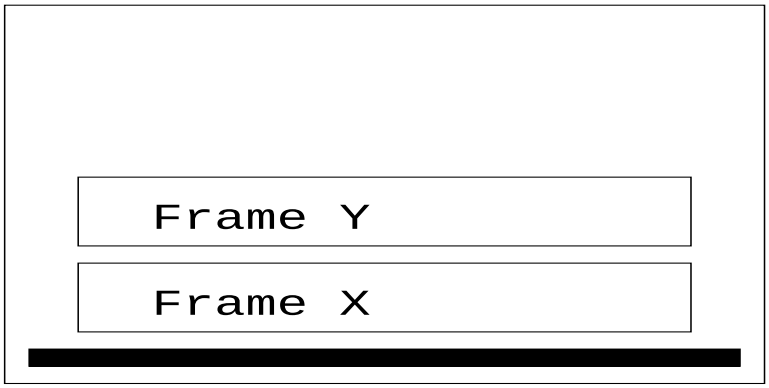


図 3.3: スレッド A の Java スタックの状態

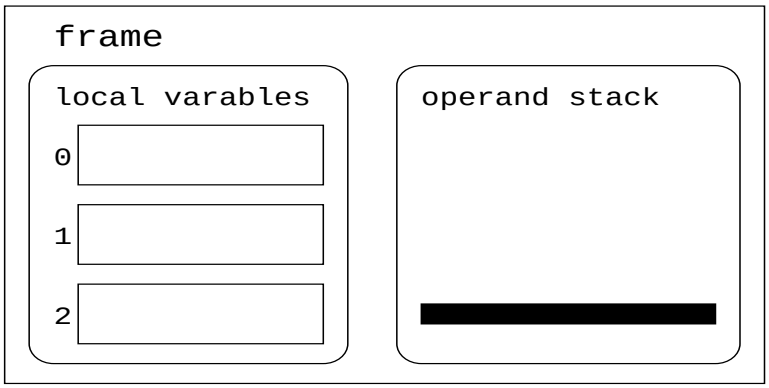


図 3.4: フレームの主な構成要素

は、常にカレントメソッドだけどなる。

3.3 ローカル変数とオペランドスタック

次に、Java スタック上に積まれるフレームの中身を見ていくことにする。フレームの中には、メソッドによって使用される。下図 3.4 のようにローカル変数とオペランドスタックの 2 つが含まれる。

Java 言語や C++言語などの高級言語でメソッドを実行することと、アセンブラ言語やマシン語でインストラクションを実行することは良く似ている。高級言語でメソッドを実行するときには、引数を渡し返し値を受け取る。高級言語のメソッド実行の際の引数は、メソッド引数と呼ばれ、アセンブラ言語のインストラクションの実行の際の引数は、オペランドと呼ばれる。オペランドスタックにオペランドを積むことによってインストラクションにオペランドが渡される場合があり、このときにオペランドスタックが使用される。他にも、インストラクションコードのバイトの隣に定数を埋め込んでオペランドとする場合が

ある。そして、インストラクションの実行にのために用意された一時的な記憶領域であるローカル変数がある。オペランドスタックがスタックであるのに対して、ローカル変数は、0 から始まる番号によって格納場所を指定している。

このローカル変数とオペランドスタックが、メソッドの中で実行されるインストラクションコード (マシン語) を実行したときにどのような変化があるかを説明していく。

例として、以下の code を考えてみる。

```
void method1(int a, int b) {
       int c = a + b;
}
```

このメソッドを Java のバイトコンパイラーによってクラスファイルを生成させ、ある種の逆アセンブラにかけると以下のようなインストラクションの列が見える。ここで、Java の code における method1 に引数である a と b は、それぞれローカル変数の 1 番と 2 番に格納されてるとする。

```
iload_1
iload_2
iadd
istore_3
```

それぞれのインストラクションについて説明すると、`iload_1` という命令は、ローカル変数の 1 番に格納されている `int` 型の変数をオペランドスタック上に積むという命令で、`iload_2` という命令は、ローカル変数の 2 番に格納されている `int` 型の変数をオペランドスタック上に積むという命令である。iadd というのは、オペランドスタック上に積まれた上から 2 つの `int` 型の変数を足すという命令、そして、最後の `istore_3` というのが、オペランドスタックの一番上にある変数をローカル変数の 3 番に格納するという命令である。

この 4 つのインストラクションを実行することによってローカル変数とオペランドスタックがどのように変化するかを図によって示す。以下に示す例は、実行時の動きである。

まず、`iload_1` によってローカル変数の 1 番に格納されている int 型の method1 の第一引数 `a` の値がオペランドスタック上に積まれる。3.5

次に、`iload_2` によってローカル変数の 2 番に格納されている int 型の method1 の第二引数 `b` の値がオペランドスタックの上に積まれる。3.6

そして、`iadd` により、オペランドスタック上に積まれた 2 つの int 型の値を足すことになる。この命令によって、もともと 2 つあった int 型の値は、取り除かれ、足された値 (`a + b`) がオペランドスタックの

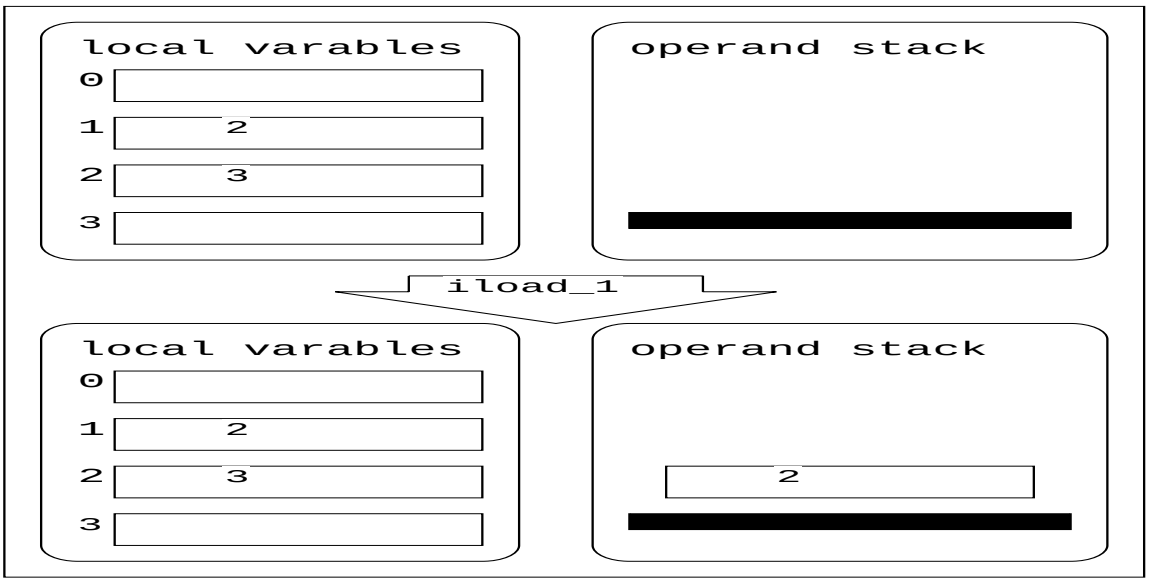


図 3.5: iload_1 を実行したときの変化

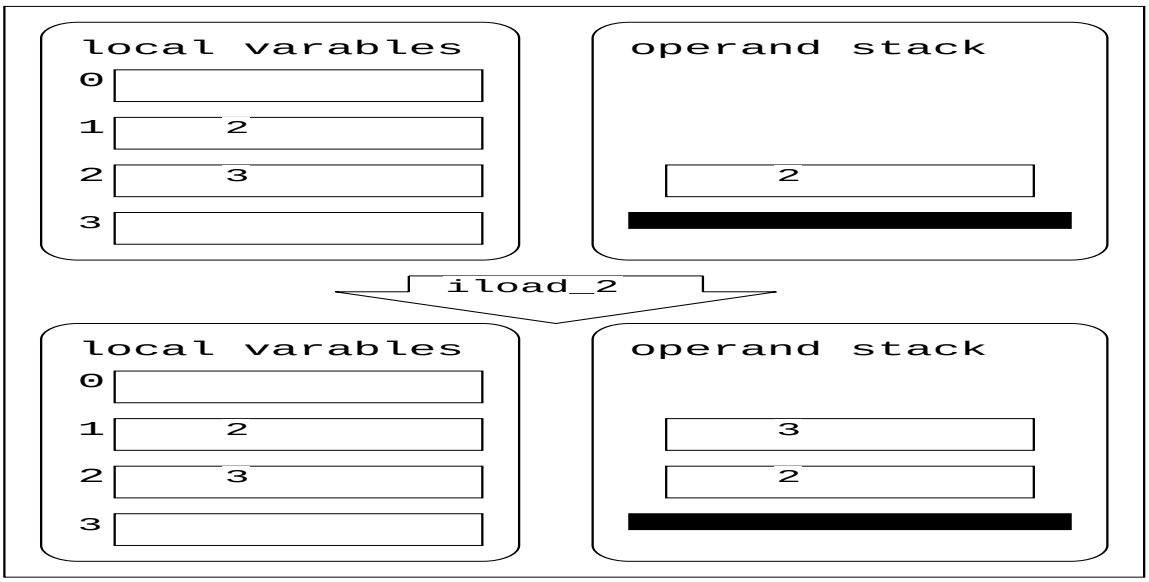


図 3.6: iload_2 を実行したときの変化

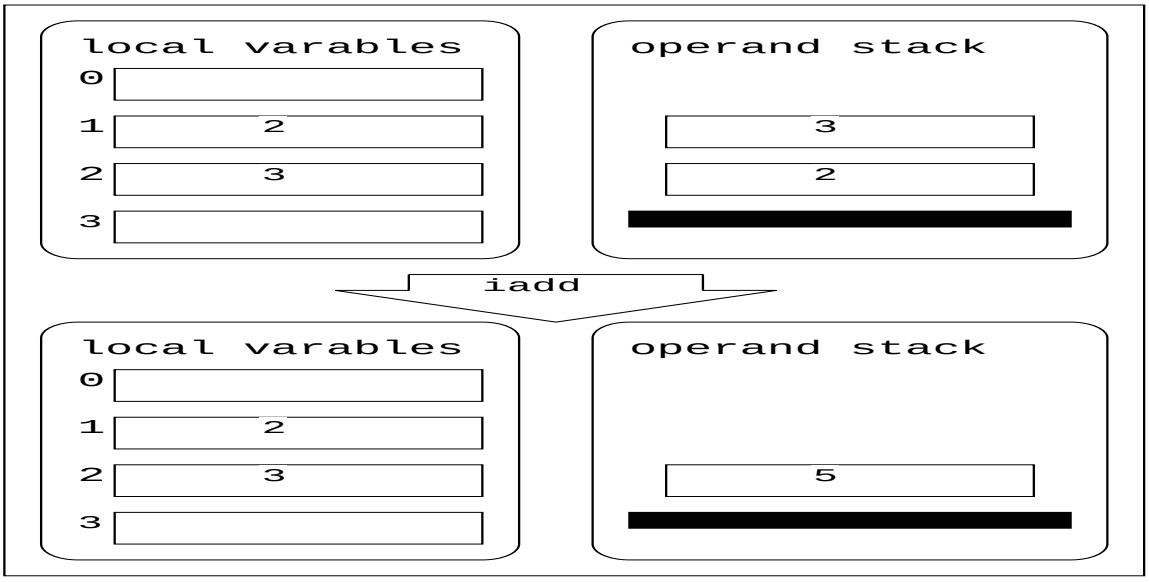


図 3.7: iadd を実行したときに変化

上に積まれる。3.7

最後に、int 型の値がオペランドスタック上に積まれているものをローカル変数の 3 番に格納し、オペランドスタック上の int 型の値を取り除く。3.8

以上、ローカル変数とオペランドスタックが命令によってどのように変化するかを説明した。次章では、これを基にクラスファイルの検査、そしてバイトコード検証系がどのような機構なのかを説明する。

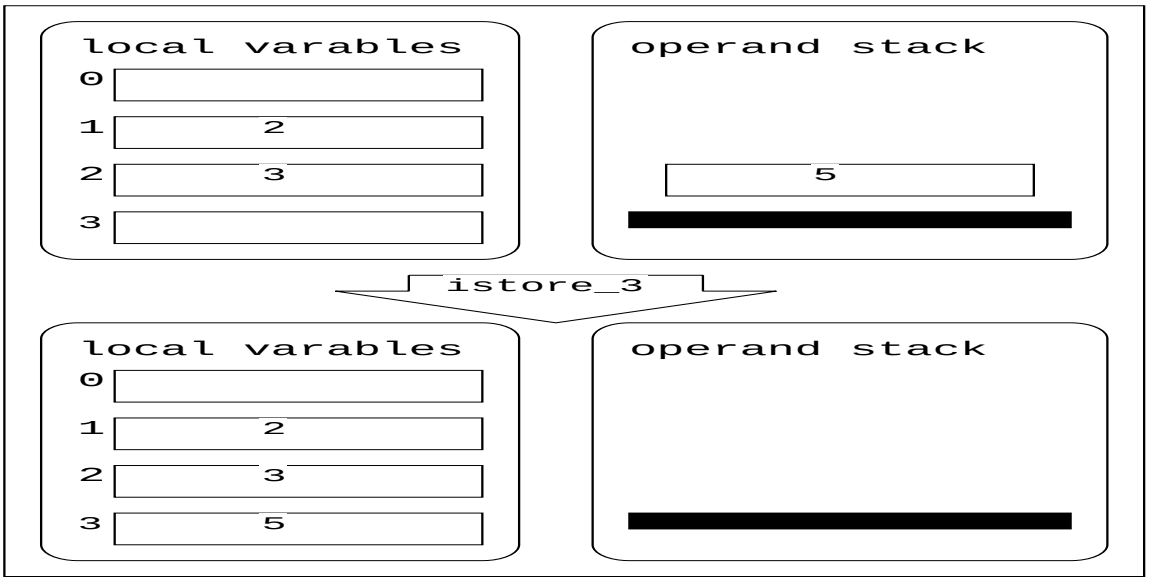


図 3.8: istore_3 を実行したときに変化

第 4 章

バイトコード検証系

本章では、バイトコード検証系の必要性を述べ、その後、どのようにクラスファイルを検査しているのかを説明する。

4.1 バイトコード検証系の必要性

Java 仮想機械は、読み込む時点では、クラスファイルが Sun の Java バイトコンパイラによって生成されたものであるかということや、Java 仮想機械上で実行して以下のことが起らないという保証がない。

- コンピューターのハードウェアやファイルシステムにダメージを与える
- コンピューターをクラッシュさせたり、使えないものになる
- コンピューターについての情報やファイルに保存してあるデータを他のところに漏洩する

また、Sun の HotJava Web ブラウザのようなアプリケーションは、ソースコードをダウンロードした後にコンパイルを行うのではく、バイトコンパイル済みのクラスファイルをダウンロードすることになる。このことにより、クラスファイルをダウンロードした後にそのクラスファイルが Java 仮想機械上で実行しても上に挙げたことにならないものなのか検査する必要がある。

それでは、次の節で実際にどのような検査をしているのかを説明する。

4.2 クラスファイルの検査

クラスファイルの検査は、大きく分けて以下ように、4 つの段階に分けることができると考えられる。[5]

- pass 1 Java 仮想機械はクラスファイルをロードする際、まずそのファイルが Java のクラスファイルの基本的な形式に合致しているかどうかを確認する。ここでいう基本的な形式とは、最初の 4 バ

イトは決められたマジック・ナンバーを保持していなければらなず、認識する属性はすべてクラスファイルに埋め込まれた値通りのフォーマットであること。また、クラスファイルは途中で途切れていたり、最後尾に余分なバイトがあってはいけないというものである。そして、命令を格納している大きさを規定している値等を格納しているコンスタント・プールと呼ばれる場所には int 型や文字型等決められた形式以外の情報は保持していないことを確認する。

- pass 2 クラスファイルに命令が格納されている部分がありそれは配列として表現されているのだがそのクラスファイルのリンク時に、その配列の内容に関係なく行なえる追加の検査をここで行なう。pass 2 では、以下の検査内容が含まれる。

  - final 節がサブクラス化されていないこと、および final メソッドがオーバーライドされていないことを保証する。
  - 各クラス (Object は除く) がスーパークラスを保持していることを検査する。
  - コンスタント・プールが規定した静的制約を見たしていることを保証する。
  - コンスタント・プール中のすべてのフィールド参照とメソッド参照が有効な名前、有効なクラス、有効な型記述子を保持しているかどうかを検査する。

  検査しているクラスのフィールド参照とメソッド参照を検索する際、pass 2 では指定したフィールドやメソッドが指定したクラス中に存在していることを確認するための検査はしない。ここでは、詳細な検査は pass 3 と pass 4 で行なわれる。

- pass 3

  ここでは、リンクの最中に各メソッドのデータフロー解析を行うことによって、クラスの各メソッドの命令内容を検査する。このデータフロー解析では、program counter が同じところには、どのようなコード・パスを通ろうとも、任意の場所で以下のことを保証する。

  - オペランド・スタックは必ずオブジェクトの同じサイズ、同じ型を保持している。
  - 適切な型の値を保持していることがわかっていない限りは、ローカル変数にアクセスしない。
  - メソッドは、適切な引数をともなって呼び出される。
  - フィールドには、適切な型の値のみを代入することができる。
  - すべてのオペコードは、オペランド・スタック上とローカル変数中に適切な型の引数を保存している。

  詳しい内容に関しては、後節の「バイトコード検証系で行う検査」で改めて記述する。

- pass 4

  原則的には pass 3 で行うべきテストを効率性の問題からメソッド・コールの初回呼び出しに持ち越す場合がある。こうすることによって検証系の pass 3 はクラスファイルのロードを可能な限り避けようとしている。例えば、メソッドがクラス A のインスタンスをリターンする別のメソッドを呼び出し、そのインスタンスを同じ型のフィールドのみに代入している場合、検証系としては、クラス A が実際に存在するかどうかをわざわざ検査しない。しかし、そのインスタンスを B 型のフィールドに代入する場合には A が B のサブクラスであることを保証するために A と B の双方の定義をロードしなければならない。pass 4 では、適切な Java 仮想機械の命令によって検査を行う仮想パスとなる。型を参照する命令を最初に実行する際、実行命令は以下のことを行う。

  – まだロードしていなければ、参照型の定義をロードする。
  – 現在実行している型がその型を参照することができることを検査する。
  – 初期化されていなければ、そのクラスを初期化する。

  命令が、最初にメソッドを呼び出すことや、アクセスや、フィールドを更新したりする際に以下のことを行う。

  – 参照されるメソッドやフィールドが、指定したクラス中に存在してることを保証する。
  – 参照されるメソッドやフィールドが、指示されたディスクリプタを保持していることを検査する。
  – 現在実行中のメソッドが、参照されるメソッドやフィールドにアクセスできることを検査する。

  型検査に関しては、pass 3 で既に行っているため、pass 4 において Java 仮想機械はオペランド・スタック上のオブジェクトの型を検査することはしない。また pass 4 で検知したエラーにより、LinkageError というクラスのサブクラスのインスタンスをスローして実行を止める。

4.3 バイトコード検証系で行う検査

本節では、形式仕様記述言語で記述する対象であるバイトコード検証系について、以下に詳しく説明する。

クラスファイルの中にある各メソッドのコードは個別に検証されることになる。まず、コードを形成するバイト列が一連の命令に分割され、各命令の先頭の番地を覚えておく。次に、検証系は再びコードを読み直して命令を解析する。このパスの間に、各 Java 仮想機械命令に関する情報をメソッド中に保持する

ためのデータ構造体が構築される。各命令にオペランドが一つでも存在する場合は、有効性を確認するために検査が行なわれることにある。

検査内容としては、以下のものが存在する。

- 分岐はメソッドの命令列の境界内になければならない
- すべての制御フロー命令のターゲットは各命令の先頭となる。wide 命令の場合には、wide オペコードを命令の先頭とみなし、その命令の先頭としなし、その wide 命令の修飾している処理を指定しているオペコードは命令の先頭とはみなさない。また命令の途中へ分岐することもできない。
- メソッドが使用しているローカル変数の数よりも大きな数を添え時として使ってローカル変数にアクセスしたり、変更したりすることはできない。
- 命令の途中でコードの終了することはない。
- 実行がコードの限界を越えることはできない。
- 各例外ハンドラにおいて、ハンドラが保護するコードの起点と終点は命令の先頭になければならず、起点は終点の前になければならない。例外ハンドラのコードは有効な命令から開始しなければならないが、wide 命令が修飾しているオペコードからは開始することができない。

これらの検査により以下の 3 つが保証されていることになる。

- オペランド・スタックは、オーバーフローもアンダーフローもしない
- 使用、ストアするすべてのローカル変数は有効なものである。
- すべての Java 仮想機械の命令に対する引数は有効な型である。

次に、バイトコード検証系がどのように検査するのかを説明する。バイトコード検証系は、メソッドの各命令の実行に先立ってオペランド・スタックの内容と、ローカル変数の内容を記録する。オペランド・スタックについては、スタックの大きさとそこに格納されているそれぞれの値の型を知る必要がある。各ローカル変数については、そのローカル変数の型や、ローカル変数がローカル変数が使用不可能、あるいは不明な値 (初期化されてない可能性がある) を保持しているかを確認する必要がある。バイトコード検証系がオペランド・スタック上の値の型を決定する際には、整数型 (例えば byte、short、char) を識別する必要がない。

次に、データフロー・アナライザを初期化する。メソッドの最初の命令では、引数を表現したローカル変数には最初にメソッドの型ディスクリプタが示す型の値が保持されていて、オペランド・スタックは空

である。他のローカル変数にはすべて不正な値を保持している。まだ検査していない他の命令ではオペランド・スタックやローカル変数に関する情報を利用することはできない。

そして、データフロー・アナライザを実行する。各命令に対応している「変更」(changed) ビットというものを設けていて、これはその「変更」ビットが立っている場合、その命令に着目する必要があるということを示すものである。まず、最初の命令に対してのみ「変更」ビットをセットする。その後、データフロー・アナライザは以下のループを実行する。

1. 「変更」ビットがセットされている仮想機械命令を選択する。「変更」ビットがセットされている命令が残っていない場合、メソッドを問題なく検証したということになる。さもなければ選択した命令の「変更」ビットをオフにして、次のステップに進む。
2. そして、以下にオペランド・スタック上の命令とローカル変数の影響をモデル化する。
   - 命令がオペランド・スタックからの値を使用する場合、スタックに十分な数の値が存在し、スタックの先頭の値が適切な型であることを確認する。
   - 命令がローカル変数を使用する場合、指定したローカル変数が適切な型を値を保持していることを確認する。さもなければ検証は失敗する。
   - 命令がオペランド・スタックに値をプッシュする場合、新しい値のあめに十分な場所がオペランド・スタックに存在するこおを確認する。モデル化したオペランド・スタックの先頭に指定の型を加える。
   - 命令がローカル変数を変更する場合、ローカル変数は現在新しい型を保持していることを記録する。
3. いま着目している命令の次にくる命令を決定する。次にくる命令とは、以下のいずれかである。
   - カレントの命令が無条件制御移行命令 (例えば、goto、return、athrow) でなかった場合、その次の命令となる。メソッドの最後の命令である場合、検証は失敗する。
   - 条件制御移行、または無条件制御移行やスイッチの分岐先
   - この命令の例外ハンドラ
4. カレントの命令終了時に、オペランド・スタックとローカル変数の状態を、次にくる各命令にマージする。マージすることに関しては、次節「オペランド・スタックとローカル変数のマージ」で説明する。例外ハンドラへの制御移行という特別なケースでは、オペランド・スタックに、例外ハンドラ情報が示す例外型の単一オブジェクトが保持されるようにする。
   - 次にくる命令の実行が初回である場合、ステプ 2 と 3 で操作したオペランド・スタックとローカル変数値が、次の命令を実行する前のオペランド・スタックとローカル変数の状態を記録する。

次にくる命令に対して「変更」ビットをセットする。

- 次にくる命令を以前に実行していた場合、ステップ 2 と 3 で操作したオペランド・スタックとローカル変数値を、以前に実行した際の値にマージする。値を変更していた場合には「変更」ビットをセットする。

5. ステップ 1 から続行する。

4.4 オペランド・スタックとローカル変数のマージ

2 つのオペランド・スタックをマージするには、各スタックにある値の数が同一である必要がある。2 つのスタック上の対応する場所に型の異なる参照型があってもかまわない点を除けば、スタック上の値の型は等しくなければならない。型の異なる参照型の場合、マージするオペランド・スタックは最初の共通スーパークラスのインスタンスへ、あるいは、2 つの型のスーパーインターフェースへの参照を保持している。Object 型のすべてのクラス型とインターフェース型のスーパー型であるため、こういった参照型は必ず存在している。オペランド・スタックをマージすることができない場合は、メソッドの検証は失敗する。

2 つのローカル変数の状態をマージするためにローカル変数の対応する組を比較する。2 つの型が等しくない場合、双方が参照値を保持していない限り、検証系はローカル変数が使用できない値を保持していると記録する。1 組のローカル変数の双方が参照値を保持している場合、マージした状態は 2 つの型の直近の共通スーパークラスのインスタンスへの参照を保持している。

データフロー・アナライザが失敗することなくメソッド上の検証を実行できた場合、クラスファイル検証系のパス 3 は問題なくメソッドを検証したことになる。

以上で、クラスファイルの検査について説明した。次章では、このバイトコード検証系をどうモデル化しどのように形式仕様記述言語 CafeOBJ を用いて記述を行ったのかを説明する。

第 5 章

CafeOBJによる表現

この章では、バイトコード検証系をどのようにモデル化して、どのようにCafeOBJによる記述をしたかについて説明する。

5.1 バイトコード検証系のモデル化とCafeOBJによる記述

目標としていることは、入力としてバイトコードを受け取ってそのバイトコードが決められた条件に合致しているかをチェックすることである。また、バイトコード検証系による検査は、オペランド・スタックやローカル変数には変数の型の情報を積むことで検査をしていて、実際の変数の値は検査では使用していない。

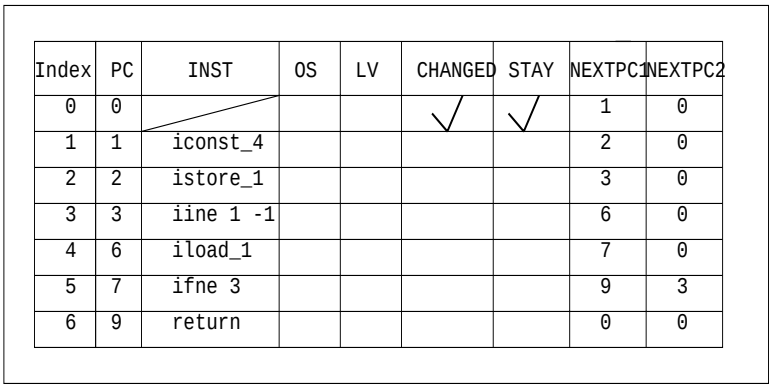
用意するものとして、Program Counter(PC)、命令(INST)、オペランド･スタック(OPERAND-STACK、表中ではOS)、ローカル変数(LOCAL-VARIABLES、表中ではLV)、変更ビット(CHANGEDBIT、表中ではCHANGED)、滞在したかどうか(STAY) そして、次の命令のPCを格納しているNEXTPC1とNEXTPC2というものを用意して、Indexを含めて合計9つの要素を持つ表を以下のように用意した。

この表5.1を使いながら4章で記述した方法により、バイトコード検証系をシミュレートしていく。PCの0番に関してはシミュレートをするのが容易にするために仮想的に用意したものである。

5.2 状態の変化の表現

CafeOBJによる記述について、それぞれの表の内容に対する観測演算として以下の9つを用意することで表を表現した。

```
getpc i R
getinst i R
getstack i R
```



Index	PC	INST	OS	LV	CHANGED	STAY	NEXTPC1	NEXTPC2
0	0				√	√	1	0
1	1	iconst_4					2	0
2	2	istore_1					3	0
3	3	iine 1 -1					6	0
4	6	iload_1					7	0
5	7	ifne 3					9	3
6	9	return					0	0

図 5.1: 使用するグラフ

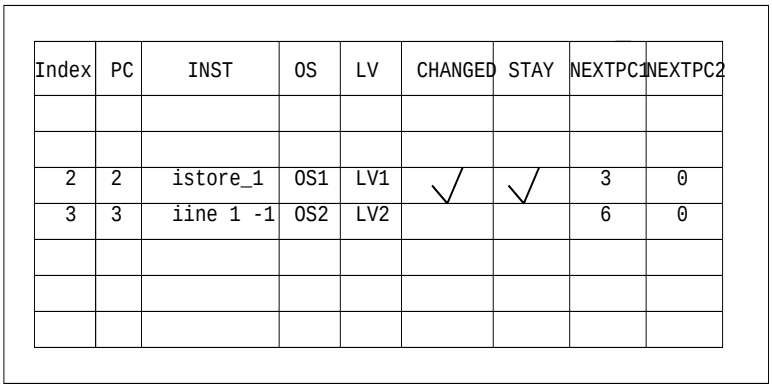
```
getlocalv i R
getstaybit i R
getchangedbit i R
getnextpc1 i R
getnextpc2 i R
getIndexFromPc i R
```

ここで、i は、表における Index の値、R は表における状態を定めた観測対象となる。また、これらの値によってどのように表の内容を変化させるかを決定する。

以下に検証のプロセスについてこの figure 9 を使って説明する。最初に PC の 0 番のの CHANGED をチェックしておき、検証を始める。まず始めに CHANGED のところにチェックがあるものを探す。もし、どの CHANGED にもチェックが無かった場合は、この表に用意したコードに関しては、バイトコード検証系は終了したことになり、検査したコードはバイトコード検証系が検査できるもの関しては全て正常であったことを示したことになる。

次に、CHANGED にチェックがあった場合、そのチェックが付いた Index と、その Index の次にくる Index、すなわち CHANGED にチェックのあった場所の Index に対する NEXTPC1 と NEXTPC2 のものに注目する。ここで、NEXTPC1 もしくは、NEXTPC2 の中身が 0 であるのもについては、行くところがないとする。また、NEXTPC1 と NEXTPC2 が必要な理由として、ジャンプ命令のときに表現しやすいようにするために、2 つ用意することにした。また、NEXTPC1 と NEXTPC2 の二つとも 0 のときにはこのメソッドの終了とすることにした。

そして、STAY の項目にチェックがなかった場合、そのプログラムカウンタの場所には初めてくることになるので CHANGED にチェックが付いた Index の状態のときのオペランドスタックとローカル変数を用いて NEXTPC1 もしくは NEXTPC2 の命令を実行し、NEXTPC1 もしくは NEXTPC2 のオペランド



Index	PC	INST	OS	LV	CHANGED	STAY	NEXTPC1	NEXTPC2
2	2	istore_1	OS1	LV1	√	√	3	0
3	3	iine 1 -1	OS2	LV2			6	0

図 5.2: CHANGED が立っていないとき

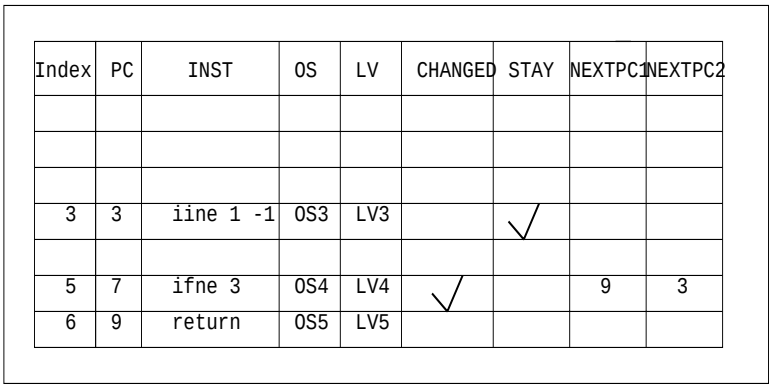
スタックとローカル変数の場所にその状態を格納する。そして、NEXTPC1 もしくは、NEXTPC2 の場所のオペランドスタックとローカル変数の状態に変化があったということで、CHANGED にチェックを入れる。そして、いま CHANGED にチェックの付いていたものを外す。

以下の図を使って説明して、CafeOBJ による表現を示す。

そして、CafeOBJ による表現は、表中の状態を R とし、表のインデックスが Index であるときの命令を getinst Index R と表現できる。また命令によってどのようにオペランドスタックが変化するかを表現するものとして、eval getinst Index R OS と表現し、どのようにローカル変数が変化するかを表現するものとして、eval getinst Index R LV と表現する。また、あるインデックスのときの次のプログラムカウンタを NEXTPC1 から得るという表現を getnextpc1 Index R という記述方法とした。そして、表中の OS1 と LV1 を基に OS2 と LV2 を変化させるということを CafeOBJ により記述すると以下のようになる。

```
bceq mark-check R = cboff 2 staybiton 2
  cbon getIndexFromPc getnextpc1 2 R R
  lvm getIndexFromPc getnextpc1 2 R R
 stm getIndexFromPc getnextpc1 2 R R
  if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R == 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == false
```

まず bceq は、振舞仕様における条件分岐付きである演算であることを示している。そして、この演算の



Index	PC	INST	OS	LV	CHANGED	STAY	NEXTPC1	NEXTPC2
3	3	iine 1 -1	OS3	LV3		√		
5	7	ifne 3	OS4	LV4	√		9	3
6	9	return	OS5	LV5				

図 5.3: CHANGED が立っているとき

名前は mark-check で表中の状態を決定してる。次に if 以下を見てみる。if 以下では、まず表の観測演算である getchangedbit 2 R というものがある。これは、表の中で、Index が 2 である場所の CHANGED の内容を観測している。表中では、チェックマークとしているがチェックマークが付いているときに true を返す観測演算としている。次に、NEXTPC1 と NEXTPC2 が 0 であるかどうか調べている。この Index に対する NEXTPC1 の方は 0 ではないので NEXTPC1 における STAY を調べる。そして、その結果一度も来ていないということで、この mark-check 演算が選ばれたことになる。この演算は、表中 NEXTPC1 の場所である OS2 と LV2 をそれぞれ、今注目している Index の場所の OS1 お LV2 を基に代入するということを表中の状態にに加わえている。このことにより、表の中の OS2 と LV2 の場所にスタックの状態とローカル変数の状態が入れられたということとなる。

次に、一度来たことのある Index に NEXTPC1 や NEXTPC2 がなったときのことを説明する。このときには、NEXTPC1 の場所と NEXTPC2 の場所の STAY に注目する。もし、STAY という項目にチェックが入っているということはオペランドスタックとローカル変数の箱の内容を埋めたことがある NEXTPC1 もしくは NEXTPC2 のプログラムカウンタの場所にある命令を PC のときのオペランドスタックとローカル変数に対して命令を実行する。このときのオペランドスタックとローカル変数の状態が、NEXTPC1 もしくは NEXTPC2 のときのオペランド・スタックとローカル変数の状態と同等であるならば現時点での検査を終了し、また CHNAGED にチェックがあるものを探す。もち、同等でなければ新しく作成したオペランド・スタックの状態とローカル変数の状態を埋める。

以下の図を使って説明をして、CafeOBJ による表現を示す。

そして、ĆafeOBJ による表現は、NEXTPC1 と NEXTPC2 とも 0 でなく、NEXPTC2 の方の Index には一度来ているという状態であるので CafeOBJ による表現は、以下のようになる。

```
bceq mark-check R = cboff 5 staybiton 5
```

```
 cbon getIndexFromPc getnextpc1 5 R R
  lvm getIndexFromPc getnextpc1 5 R R
 stm getIndexFromPc getnextpc1 5 R R
  cboff getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == false
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and true == same? getstack getIndexFromPc getnextpc2 5 R R R
 eval getinst getIndexFromPc getnextpc2 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc getnextpc2 5 R R R
 eval getinst getIndexFromPc getnextpc2 5 R R R getlocalv 5 R .
```

そして、if 以下を見ると、getchangedbit 5 R という Index の 5 の CHANGED がチェックが付いていて、NEXTPC1 と NEXTPC2 ともに 0 でなく、STAY は、NEXTPC2 で指している Index の方にはチェックが付いていて NEXTPC1 で指している Index の方にはチェックが付いていない状態。そして、OS3 と LV3 と OS4 と LV4 に対して NEXTPC2 が指している Index での命令を実行した結果を比較して同じであるので、OS3 と LV3 はそのままとする。そして、NEXTPC1 が指している Index の OS5 と LV5 は、OS4 と LV4 を基に NEXTPC1 が指している Index の命令を実行したものを代入することになる。

5.3 型検査の表現

次に、オペランドスタックとローカル変数の状態を得るときに、また別の検査をしていることについて説明する。オペランドスタックに関しては、スタック上の積む操作と取る操作のときに型が合致しているかを検査している。もし、合致していないのであれば、検証は失敗となる。また、ローカル変数に関しては、ローカル変数に格納する操作のときとローカル変数を取得するときの操作のときに型が合致しているかどうかを検査している。こちらに関しても、合致していないのであれば、検証は失敗となる。この 2 つの検査に関して、それぞれ状態を求めるときに問題がないかそれぞれ検査している。

そして、型検査に対しての CafeOBJ の記述は、以下のように記述した。

```
  bceq getstack i stm j R = eval "error" getstack j R
```

```
if getIndexFromPc getnextpc1 i R R == j
  and false == tcheck getinst i R getstack j R .

bceq getlocalv i lvm j R = eval "error" getlocalv j R
if getIndexFromPc getnextpc1 i R R == j
and false == tcheck getinst i R getlocalv j R .
```

まず、オペランド・スタックに対しての型検査として、tcheck という演算により実行しようとしている命令によって型が違うものによって実行しようとしていないかを検査している。そして、その検査の結果、型がい違うものであった場合今回 CafeOBJ で表現したものでは、オペランド・スタックであれば error という型情報をスタック上に積み、ローカル変数であれば、error という型情報を格納するようにした。

そして、最後に CHANGED にチェックがあるかどうか検査をして、オペランドスタック、ローカル変数を求めるときに、格納や比較するところまでを mark-check というメソッドで、そしてオペランド・スタック、ローカル変数を求めるときに、型の検査が失敗していないかどうかを観測する typecheck という観測演算、また、CHANGED のチェックまだ残っているかどうかを観測する exist?() という観測演算とする。そして、この表自体 (表の状態) を表現しているものを S とすると以下のように表現をして、バイトコード検証系を表現した。

```
bred typecheck mark-check S .
bred exist? mark-check S .
bred typecheck mark-check mark-check S  .
bred exist? mark-check mark-check S  .
bred typecheck mark-check .... mark-check S .
bred exist? mark-check .... mark-check S .
```

このように mark-check に実行していき、typecheck の観測の結果が常に true で、exist?の観測の結果が false すなわち、CHANGED にチェックがない状態になったとき検査しているコードは、バイトコード検証系を通過したものとなる。

以上、バイトコード検証系をどのようにモデル化してどのように CafeOBJ による記述をしたかについて説明した。また、CafeOBJ によるバイトコード検証系の記述の全体は、Appendix に記述した。

第 6 章

まとめと今後の課題

本研究では、Java 仮想機械におけるバイトコード検証系の仕様を代数仕様記述言語 CafeOBJ によって記述した。この記述により、実際にどのようにバイトコード検証系が動作しているかをより形式的に記述できたことになる。しかし、本研究で行った CafeOBJ の記述は、サンプルとなるバイトコードに出てくる命令のみを受けとることのできるバイトコード検証系となっている。この事に関しては、対応する命令を増やすことで、より多くのバイトコードについて検査できると考えられる。

次に、バイコード検証系を通った Java のクラスファイルが保証されていることは以下の 3 つである。

- オペランド・スタックは、オーバーフローもアンダーフローもしない
- 使用、ストアするすべてのローカル変数は有効なものである
- すべての Java 仮想機械の命令に対する引数は有効な型である

このことが Java 仮想機械上でそのクラスファイルを実行しても、コンピュータのハードウェアやファイルシステムにダメージを与えないかどうかか、またコンピュータをクラッシュさせたり、コンピュータを使えない状態にしたりしないか。そして、コンピュータ上にある情報やファイルの内容を漏洩するようなことにならないどうかということが保証されるかどうかは判断できない。このことを保証しているかどうかを検査するには、バイトコード検証系をモデル化し CafeOBJ で形式的に記述したように、Java 仮想機械自体もモデル化し、CafeOBJ で形式的に記述し、これらの保証が確実なものか検証する必要があると考える。

また、今回行った記述は、ソフトウェア開発における上流過程の形式仕様の記述と多少意味合いが違うように見られる。それは、Java 仮想機械というシステムは現時点で存在しており、これから開発をするというものではないということである。しかし、現存するソフトウェアをもう一度再確認するという観点からすれば、意味のあることであると考えられる。特に Java 言語は、ネットワーク上を移動することが可能な言語であり、もし不正なコードを実行できるようなものであれば、セキュリティー上大きな問題とな

ることが考えられ、Java 仮想機械全体の仕様記述をすることによって仕様を確認するということは、重要であると考える。

最後に、代数仕様記述言語 CafeOBJ の特徴でもある実行可能であることで、本研究で記述したバイトコード検証系の記述が受けとれる状態のバイトコードに変換するような、変換プログラムを用意し、`mark-check` と `typecheck()`、`exist?`を並べ、検査をする部分を `CafeOBJ` に渡すプログラムを準備することにより検証の自動化ということも考えられる。

また、本稿で記述したようなある程度大きい仕様を記述する際、特に条件の羅列が多いような場合、仕様記述をサポートするような機構が必要であると強く感じられた。これは、エディタによるサポートもしくは、CafeOBJ 自体の機能で自動的に条件を補完するような機構があると、より大きな仕様を記述する際、記述の負担が減るのではないかと思われる。以上のことも含め、より実際に近いバイトコード検証系の形式仕様記述、そして Java 仮想機械の形式仕様記述等、今後の課題としたい。

第 7 章

謝辞

本修士研究を指導していただいた北陸先端科学技術大学院大学の二木 厚吉教授、渡部卓雄助教授、森 彰助手、緒方 和博助手、天野 憲樹助手に深く感謝いたします。また、研究に関する議論につきあっていただいた言語設計学講座の皆様に感謝致します。

第 8 章

Appendix

以下は、本研究で CafeOBJ を用いて記述したバイトコード検証系の仕様である。

```
-- ------------------------------------------------------------------------
-- array object
-- レジスターを表現するための配列の仕様
-- ------------------------------------------------------------------------

module* ARRAY (X :: TRIV) {
protecting(NAT)

[ Array ]

op null : -> Array
op _[_] : Array Nat -> Elt.X
op _[_<-_] : Array Nat Elt.X -> Array

var A : Array
vars I K : Nat
var J : Elt.X

eq (A [ I <- J ])[I] = J .
ceq (A [ I <- J ])[K] = A[K] if I =/= K .
}

-- ------------------------------------------------------------------------
-- stack object
-- オペランド・スタックを表現するためのスタックの仕様
-- ------------------------------------------------------------------------

module* STACK (Y :: TRIV) {
```

```
[ Stack ]

op null : -> Stack
op push : Elt.Y Stack -> Stack
op pop_ : Stack -> Stack
op top_ : Stack -> Elt.Y

var S : Stack
var E : Elt.Y

eq top (push (E, S)) = E .
eq pop (push (E, S)) = S .
}

-- ------------------------------------------------------------------------
-- operand stack imports from stack object
-- 数個の命令を扱えるオペランド・スタックの仕様
-- ------------------------------------------------------------------------

module* OPERAND-STACK {

protecting(STACK(STRING) + BOOL)

op eval__   : String Stack -> Stack
op tcheck__ : String Stack -> Bool
op same?__ : Stack Stack -> Bool

var S : Stack
vars S1, S2 : Stack

eq eval "" S = S .
eq eval "iconst_4" S = push("int", S) .
eq eval "istore_1" S = S .
eq eval "iinc_1" S = S .
eq eval "iload_1" S = push("int", S) .
eq eval "ifne" S = pop S .
eq eval "return" S = S .

-- error
eq eval "error" S = push("error", S) .

eq tcheck "" S = true .
eq tcheck "iconst_4" S = true .
ceq tcheck "istore_1" S = true if top S == "int" .
```

```
eq tcheck "iinc_1" S = true .
ceq tcheck "iload_1" S = true if top S == "int" .
eq tcheck "ifne" S = true .
eq tcheck "return" S = true .

-- this is fake.
ceq same? S1 S2 = true if top S1 == top S2 .
ceq same? S1 S2 = false if top S1 =/= top S2 .
}

-- ------------------------------------------------------------------------
-- local variables imports from array
-- 数個の命令を扱えるレジスターの仕様
-- ------------------------------------------------------------------------

module* LOCAL-VARIABLES {

protecting(INT + ARRAY(STRING))

op eval __   : String Array -> Array
op tcheck __ : String Array -> Bool
op same? __  : Array Array -> Bool

var LV : Array
vars I J : Int
vars LV1 LV2 : Array

eq eval "" LV = LV .
eq eval "iconst_4" LV = LV .
eq eval "istore_1" LV = LV[1 <- "int"] .
eq eval "iinc_1" LV = LV .
eq eval "iload_1" LV = LV .
eq eval "ifne" LV = LV .
eq eval "return" LV = LV .

-- error
eq eval "error" LV = LV[256 <- "error"] .

eq tcheck "" LV = true .
eq tcheck "iconst_4" LV = true .
eq tcheck "istore_1" LV = true .
ceq tcheck "iinc_1" LV = true if LV[1] == "int" .
ceq tcheck "iinc_1" LV = false if LV[1] =/= "int" .
eq tcheck "iload_1" LV = true .
```

```
eq tcheck "ifne" LV = true .
eq tcheck "return" LV = true .

-- this is fake.
ceq same? LV1 LV2 = true if LV1[1] == LV2[1] .
ceq same? LV1 LV2 = false if LV1[1] =/= LV2[1] .
}

-- ---------------------------------------------------------------------
-- Program Counter
-- プログラム・カウンターを表現するためのデータ表現
-- ---------------------------------------------------------------------

module! PC {
protecting(NAT)
[ Nat < Pc ]
}

-- ---------------------------------------------------------------------
-- Instruction
-- 命令を表現するためのデータ表現
-- ---------------------------------------------------------------------

module! INST {
protecting(STRING)
[ String < Inst ]
}

-- ---------------------------------------------------------------------
-- Stay bit
-- 一度実行したかどうかを判定するフラッグ
-- ---------------------------------------------------------------------
module* STAY {
protecting(BOOL)
[ Bool < Stay ]
}

-- ---------------------------------------------------------------------
-- changed bit
-- オペランド・スタックとレジスターを書換えたかどうか判定するフラッグ
-- ---------------------------------------------------------------------

module* CHANGEDBIT {
protecting(BOOL)
```

```
[ Bool < Changed ]
}

-- ------------------------------------------------------------------------
-- Next PC 1
-- 次にどのプログラムカウンターに飛ぶかを格納する
-- ------------------------------------------------------------------------
module! NEXTPC1 {
protecting(NAT)
[ Nat < NextPC1 ]
}

-- ------------------------------------------------------------------------
-- Next PC 2
-- 次にどのプログラムカウンターに飛ぶかを格納する
-- ------------------------------------------------------------------------
module! NEXTPC2 {
protecting(NAT)
[ Nat < NextPC2 ]
}

-- ------------------------------------------------------------------------
-- Index of Box
-- 表の INDEX
-- ------------------------------------------------------------------------

module! INDEX {
protecting(NAT)
[ Nat < Index ]
}

-- ------------------------------------------------------------------------
-- Rule set
-- 表の表現
-- ------------------------------------------------------------------------

module* RULE {
protecting (PC + INST
+ OPERAND-STACK + LOCAL-VARIABLES
+ CHANGEDBIT + STAY + NEXTPC1
+ NEXTPC2 + INDEX)

*[ Rule ]*
```

```
-- initial state for Rule (initialize)
op init : -> Rule
-- mark-check method for Rule (method)
-- getchangedbit が true なものを見つけて
-- 表の内容を変化させる操作
bop mark-check_ : Rule -> Rule  { memo }

-- mark exist? attribute for Rule (attribute)
-- getchangedbit が true なものがないかどうかを
-- 調べる観測
bop exist?_ : Rule -> Bool
-- type check (attribute)
-- 型エラーがあるかどうかの検査
bop typecheck_ : Rule -> Bool

-- type check attribute for Rule
-- 表がどのように見えるのかを
-- 表現している観測の signature

bop getpc__ : Index Rule -> Pc
bop getinst__ : Index Rule -> Inst
bop getstack__ : Index Rule -> Stack
bop getlocalv__ : Index Rule -> Array
bop getstaybit__ : Index Rule -> Stay
bop getchangedbit__ : Index Rule -> Changed
bop getnextpc1__ : Index Rule -> NextPC1
bop getnextpc2__ : Index Rule -> NextPC2

-- Index From Pc
-- on user CODE (attribute)
bop getIndexFromPc__ : Pc Rule -> Index

-- method
-- changedbit を on にする
bop cbon__ : Index Rule -> Rule
-- changedbit を off にする
bop cboff__ : Index Rule -> Rule
-- stay bit を on にする
bop staybiton__ : Index Rule -> Rule
-- オペランド・スタックを変化させる
bop stm__ : Index Rule -> Rule
-- レジスターの内容を変化させる
bop lvm__ : Index Rule -> Rule
```

```
var R : Rule
vars i j : Index
var n : Int

-- type check
-- 型の検査
bceq typecheck R = false
if top (getstack 0 R) == "error"
or top (getstack 1 R) == "error"
or top (getstack 2 R) == "error"
or top (getstack 3 R) == "error"
or top (getstack 4 R) == "error"
or top (getstack 5 R) == "error"
or top (getstack 6 R) == "error"
or (getlocalv 0 R)[256] == "error"
or (getlocalv 1 R)[256] == "error"
or (getlocalv 2 R)[256] == "error"
or (getlocalv 3 R)[256] == "error"
or (getlocalv 4 R)[256] == "error"
or (getlocalv 5 R)[256] == "error"
or (getlocalv 6 R)[256] == "error" .

bceq typecheck R = true
if top (getstack 0 R) =/= "error"
and top (getstack 1 R) =/= "error"
and top (getstack 2 R) =/= "error"
and top (getstack 3 R) =/= "error"
and top (getstack 4 R) =/= "error"
and top (getstack 5 R) =/= "error"
and top (getstack 6 R) =/= "error"
and (getlocalv 0 R)[256] =/= "error"
and (getlocalv 1 R)[256] =/= "error"
and (getlocalv 2 R)[256] =/= "error"
and (getlocalv 3 R)[256] =/= "error"
and (getlocalv 4 R)[256] =/= "error"
and (getlocalv 5 R)[256] =/= "error"
and (getlocalv 6 R)[256] =/= "error" .

-- rule set of changedbit attribute
-- changedbit の変化
bceq getchangedbit i cbon j R = true
if i == j and j =/= 0 .
bceq getchangedbit i cbon j R = false
if i == j and j == 0 .
```

```
bceq getchangedbit i cbon j R = getchangedbit i R
if i =/= j .
bceq getchangedbit i cboff j R = false
if i == j .
bceq getchangedbit i cboff j R = getchangedbit i R
if i =/= j .

beq getchangedbit i staybiton  j R = getchangedbit i R .
beq getchangedbit i staybitoff j R = getchangedbit i R .
beq getchangedbit i stm j R = getchangedbit i R .
beq getchangedbit i lvm j R = getchangedbit i R .

-- rule set of getstaybit attribute
-- stay bit の変化
bceq getstaybit i staybiton j R = true
if i == j .
bceq getstaybit i staybiton j R = getstaybit i R
if i =/= j .
bceq getstaybit i staybitoff j R = false
if i == j .
bceq getstaybit i staybitoff j R = getstaybit i R
if i =/= j .

beq getstaybit i cbon  j R = getstaybit i R .
beq getstaybit i cboff j R = getstaybit i R .
beq getstaybit i stm j R = getstaybit i R .
beq getstaybit i lvm j R = getstaybit i R .

-- rule set of getinst attribute
-- getinst の観測結果

beq getinst i cbon j R = getinst i R .
beq getinst i cboff j R = getinst i R .
beq getinst i stm j R = getinst i R .
beq getinst i lvm j R = getinst i R .
beq getinst i staybiton j R = getinst i R .
beq getinst i staybitoff j R = getinst i R .

-- rule set of getstack attribute
-- オペランド・スタックの変化
  bceq getstack i stm j R = eval getinst i R getstack j R
if getIndexFromPc getnextpc1 i R R == j
and true == tcheck getinst i R getstack j R .
  bceq getstack i stm j R = eval getinst i R getstack j R
```

```
if getIndexFromPc getnextpc2 i R R == j
  and true == tcheck getinst i R getstack j R .

  bceq getstack i stm j R = eval "error" getstack j R
if getIndexFromPc getnextpc1 i R R == j
  and false == tcheck getinst i R getstack j R .
  bceq getstack i stm j R = eval "error" getstack j R
if getIndexFromPc getnextpc2 i R R == j
  and false == tcheck getinst i R getstack j R .

  bceq getstack i stm j R = getstack i R
if getIndexFromPc getnextpc1 i R R =/= j .
  bceq getstack i stm j R = getstack i R
if getIndexFromPc getnextpc2 i R R =/= j .

beq getstack i cbon j R = getstack i R .
beq getstack i cboff j R = getstack i R .
beq getstack i staybiton j R = getstack i R .
beq getstack i staybitoff j R = getstack i R .
beq getstack i lvm j R = getstack i R .

-- rule set of getlocalv attribute
-- レジスターの変化

bceq getlocalv i lvm j R = eval getinst i R getlocalv j R
if getIndexFromPc getnextpc1 i R R == j
  and tcheck getinst i R getlocalv j R .
bceq getlocalv i lvm j R = eval getinst i R getlocalv j R
if getIndexFromPc getnextpc2 i R R == j
and tcheck getinst i R getlocalv j R .

bceq getlocalv i lvm j R = eval getinst i R getlocalv j R
if getIndexFromPc getnextpc1 i R R =/= j
and tcheck getinst i R getlocalv j R .
bceq getlocalv i lvm j R = eval getinst i R getlocalv j R
if getIndexFromPc getnextpc2 i R R =/= j
and tcheck getinst i R getlocalv j R .

bceq getlocalv i lvm j R = eval "error" getlocalv j R
if getIndexFromPc getnextpc1 i R R == j
and false == tcheck getinst i R getlocalv j R .
bceq getlocalv i lvm j R = eval "error" getlocalv j R
if getIndexFromPc getnextpc2 i R R == j
and false == tcheck getinst i R getlocalv j R .
```

```
beq getlocalv i cbon j R = getlocalv i R .
beq getlocalv i cboff j R = getlocalv i R .
beq getlocalv i staybiton j R = getlocalv i R .
beq getlocalv i staybitoff j R = getlocalv i R .
beq getlocalv i stm j R = getlocalv i R .

-- getIndexFromPc
-- Index からプログラムカウンターを得る
beq getIndexFromPc i cbon j R = getIndexFromPc i R .
beq getIndexFromPc i cboff j R = getIndexFromPc i R .
beq getIndexFromPc i staybiton j R = getIndexFromPc i R .
beq getIndexFromPc i staybitoff j R = getIndexFromPc i R .
beq getIndexFromPc i stm j R = getIndexFromPc i R .
beq getIndexFromPc i lvm j R = getIndexFromPc i R .

-- getnextpc1
-- 一つ目のプログラムカウンター
beq getnextpc1 i cbon j R = getnextpc1 i R .
beq getnextpc1 i cboff j R = getnextpc1 i R .
beq getnextpc1 i staybiton j R = getnextpc1 i R .
beq getnextpc1 i staybitoff j R = getnextpc1 i R .
beq getnextpc1 i stm j R = getnextpc1 i R .
beq getnextpc1 i lvm j R = getnextpc1 i R .

-- getnextpc2
-- 二つ目のプログラムカウンター
beq getnextpc2 i cbon j R = getnextpc2 i R .
beq getnextpc2 i cboff j R = getnextpc2 i R .
beq getnextpc2 i staybiton j R = getnextpc2 i R .
beq getnextpc2 i staybitoff j R = getnextpc2 i R .
beq getnextpc2 i stm j R = getnextpc2 i R .
beq getnextpc2 i lvm j R = getnextpc2 i R .

--
-- rule set of mark-check method
--

-- 今回は、6 つ命令列に限定したもの
-- として表現したので以下のように書いた

-- changed bit 0

bceq mark-check R = cboff 0 staybiton 0 R
```

```
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R == 0
and getIndexFromPc getnextpc2 0 R R == 0 .

bceq mark-check R = cboff 0 staybiton 0
 cboff getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R == 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc2  0 R R R == true
and true == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0
 cbon getIndexFromPc getnextpc2 0 R R
  lvm getIndexFromPc getnextpc2 0 R R
 stm getIndexFromPc getnextpc2 0 R R
  cbon getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R == 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and false == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc2 0 R R
 lvm getIndexFromPc getnextpc2 0 R R
  stm getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R == 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 0 R R R == false .

bceq mark-check R = cboff 0 staybiton 0
 cboff getIndexFromPc getnextpc1 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R == 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and true == same? getstack getIndexFromPc getnextpc1 0 R R R
```

```
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 lvm getIndexFromPc getnextpc1 0 R R stm getIndexFromPc getnextpc1 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R == 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and false == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 lvm getIndexFromPc getnextpc1 0 R R stm getIndexFromPc getnextpc1 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R == 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == false .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 cbon getIndexFromPc getnextpc2 0 R R
  lvm getIndexFromPc getnextpc1 0 R R
 stm getIndexFromPc getnextpc1 0 R R
  lvm getIndexFromPc getnextpc2 0 R R
 stm getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and false == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R
and false == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc2 0 R R
 lvm getIndexFromPc getnextpc2 0 R R
```

```
  stm getIndexFromPc getnextpc2 0 R R
 cboff getIndexFromPc getnextpc1 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and true == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R
and false == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 lvm getIndexFromPc getnextpc1 0 R R
 stm getIndexFromPc getnextpc1 0 R R
  cboff getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and false == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R
and true == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cboff getIndexFromPc getnextpc1 0 R R
 cboff getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and true == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc1 0 R R R
```

```
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R
and true == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc2 0 R R R
eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc2 0 R R
 lvm getIndexFromPc getnextpc2 0 R R
  stm getIndexFromPc getnextpc2 0 R R
 cboff getIndexFromPc getnextpc1 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and getstaybit getIndexFromPc getnextpc2 0 R R R == false
and true == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and true == same? getlocalv getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc2 0 R R
 lvm getIndexFromPc getnextpc2 0 R R stm getIndexFromPc getnextpc2 0 R R
  cbon getIndexFromPc getnextpc1 0 R R lvm getIndexFromPc getnextpc1 0 R R
 stm getIndexFromPc getnextpc1 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == true
and getstaybit getIndexFromPc getnextpc2 0 R R R == false
and false == same? getstack getIndexFromPc getnextpc1 0 R R R
 eval getinst getIndexFromPc getnextpc1 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc1 0 R R R
eval getinst getIndexFromPc getnextpc1 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 lvm getIndexFromPc getnextpc1 0 R R
  stm getIndexFromPc getnextpc1 0 R R
 cboff getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == false
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and true == same? getlocalv getIndexFromPc getnextpc2 0 R R R
```

```
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 lvm getIndexFromPc getnextpc1 0 R R stm getIndexFromPc getnextpc1 0 R R
  cbon getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == false
and getstaybit getIndexFromPc getnextpc2 0 R R R == true
and false == same? getstack getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getstack 0 R
and false == same? getlocalv getIndexFromPc getnextpc2 0 R R R
 eval getinst getIndexFromPc getnextpc2 0 R R R getlocalv 0 R .

bceq mark-check R = cboff 0 staybiton 0 cbon getIndexFromPc getnextpc1 0 R R
 cbon getIndexFromPc getnextpc2 0 R R lvm getIndexFromPc getnextpc1 0 R R
  stm getIndexFromPc getnextpc1 0 R R lvm getIndexFromPc getnextpc2 0 R R
 stm getIndexFromPc getnextpc2 0 R R R
if getchangedbit 0 R == true
and getIndexFromPc getnextpc1 0 R R =/= 0
and getIndexFromPc getnextpc2 0 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 0 R R R == false
and getstaybit getIndexFromPc getnextpc2 0 R R R == false .

-- changed bit 1

bceq mark-check R = cboff 1 staybiton 1 R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R == 0
and getIndexFromPc getnextpc2 1 R R == 0 .

bceq mark-check R = cboff 1 staybiton 1 cboff
 getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R == 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and true == same? getstack getIndexFromPc getnextpc2 1 R R R
 eval getinst getIndexFromPc getnextpc2 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc getnextpc2 1 R R R
 eval getinst getIndexFromPc getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc getnextpc2 1 R R
```

```
 lvm getIndexFromPc getnextpc2 1 R R stm getIndexFromPc getnextpc2 1 R R
  cbon getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R == 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and false == same? getstack getIndexFromPc getnextpc2 1 R R R
 eval getinst getIndexFromPc getnextpc2 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc getnextpc2 1 R R R
 eval getinst getIndexFromPc getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc getnextpc2 1 R R
 lvm getIndexFromPc getnextpc2 1 R R stm getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R == 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 1 R R R == false .

bceq mark-check R = cboff 1 staybiton 1 cboff
 getIndexFromPc getnextpc1 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R == 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and true == same? getstack getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon
 getIndexFromPc getnextpc1 1 R R lvm getIndexFromPc getnextpc1 1 R R
  stm getIndexFromPc getnextpc1 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R == 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and false == same? getstack getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc getnextpc1 1 R R
 lvm getIndexFromPc getnextpc1 1 R R stm getIndexFromPc getnextpc1 1 R R R
if getchangedbit 1 R == true
```

```
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R == 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == false .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc getnextpc1 1 R R
 cbon getIndexFromPc getnextpc2 1 R R lvm getIndexFromPc getnextpc1 1 R R
  stm getIndexFromPc getnextpc1 1 R R lvm getIndexFromPc getnextpc2 1 R R
 stm getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and false == same? getstack getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getlocalv 1 R
and false == same? getstack getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
 getnextpc2 1 R R lvm getIndexFromPc getnextpc2 1 R R
  stm getIndexFromPc getnextpc2 1 R R cboff getIndexFromPc
 getnextpc1 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and true == same? getstack getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getlocalv 1 R
and false == same? getstack getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getlocalv 1 R .
```

```
bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
 getnextpc1 1 R R lvm getIndexFromPc getnextpc1 1 R R
  stm getIndexFromPc getnextpc1 1 R R cboff getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and false == same? getstack getIndexFromPc
getnextpc1 1 R R R eval getinst getIndexFromPc
 getnextpc1 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 1 R R R eval getinst getIndexFromPc
  getnextpc1 1 R R R getlocalv 1 R
and true == same? getstack getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cboff
 getIndexFromPc getnextpc1 1 R R cboff getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 1 R R R eval getinst getIndexFromPc
  getnextpc1 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 1 R R R eval getinst getIndexFromPc
  getnextpc1 1 R R R getlocalv 1 R
and true == same? getstack getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
 getnextpc2 1 R R lvm getIndexFromPc getnextpc2 1 R R
  stm getIndexFromPc getnextpc2 1 R R cboff getIndexFromPc getnextpc1 1 R R R
```

```
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and getstaybit getIndexFromPc getnextpc2 1 R R R == false
and true == same? getstack getIndexFromPc getnextpc1 1 R R R
 eval getinst getIndexFromPc getnextpc1 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 1 R R R eval getinst getIndexFromPc
  getnextpc1 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
 getnextpc2 1 R R lvm getIndexFromPc getnextpc2 1 R R
  stm getIndexFromPc getnextpc2 1 R R cbon getIndexFromPc
 getnextpc1 1 R R lvm getIndexFromPc getnextpc1 1 R R
  stm getIndexFromPc getnextpc1 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == true
and getstaybit getIndexFromPc getnextpc2 1 R R R == false
and false == same? getstack getIndexFromPc
 getnextpc1 1 R R R eval getinst getIndexFromPc
  getnextpc1 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 1 R R R eval getinst getIndexFromPc
  getnextpc1 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
 getnextpc1 1 R R lvm getIndexFromPc getnextpc1 1 R R
  stm getIndexFromPc getnextpc1 1 R R cboff getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == false
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getstack 1 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
```

```
 getnextpc1 1 R R lvm getIndexFromPc getnextpc1 1 R R
  stm getIndexFromPc getnextpc1 1 R R cbon getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == false
and getstaybit getIndexFromPc getnextpc2 1 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getstack 1 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 1 R R R eval getinst getIndexFromPc
  getnextpc2 1 R R R getlocalv 1 R .

bceq mark-check R = cboff 1 staybiton 1 cbon getIndexFromPc
 getnextpc1 1 R R cbon getIndexFromPc getnextpc2 1 R R
  lvm getIndexFromPc getnextpc1 1 R R stm getIndexFromPc
 getnextpc1 1 R R lvm getIndexFromPc getnextpc2 1 R R
  stm getIndexFromPc getnextpc2 1 R R R
if getchangedbit 1 R == true
and getIndexFromPc getnextpc1 1 R R =/= 0
and getIndexFromPc getnextpc2 1 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 1 R R R == false
and getstaybit getIndexFromPc getnextpc2 1 R R R == false .


-- changed bit 2

bceq mark-check R = cboff 2 staybiton 2 R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R == 0
and getIndexFromPc getnextpc2 2 R R == 0 .

bceq mark-check R = cboff 2 staybiton 2 cboff getIndexFromPc
 getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R == 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and true == same? getstack getIndexFromPc getnextpc2 2 R R R
 eval getinst getIndexFromPc getnextpc2 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc getnextpc2 2 R R R
 eval getinst getIndexFromPc getnextpc2 2 R R R getlocalv 2 R .
```

```
bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc2 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R cbon getIndexFromPc getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R == 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and false == same? getstack getIndexFromPc getnextpc2 2 R R R
 eval getinst getIndexFromPc getnextpc2 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc2 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R == 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 2 R R R == false .

bceq mark-check R = cboff 2 staybiton 2 cboff getIndexFromPc
 getnextpc1 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R == 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and true == same? getstack getIndexFromPc
getnextpc1 2 R R R eval getinst getIndexFromPc
 getnextpc1 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc1 2 R R
  stm getIndexFromPc getnextpc1 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R == 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getstack 2 R
```

```
and false == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc1 2 R R
  stm getIndexFromPc getnextpc1 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R == 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == false .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R cbon getIndexFromPc getnextpc2 2 R R
  lvm getIndexFromPc getnextpc1 2 R R stm getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R
and false == same? getstack getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc2 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R cboff getIndexFromPc
 getnextpc1 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
```

```
and true == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R
and false == same? getstack getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc1 2 R R
  stm getIndexFromPc getnextpc1 2 R R cboff getIndexFromPc
 getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R
and true == same? getstack getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cboff getIndexFromPc
 getnextpc1 2 R R cboff getIndexFromPc getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
```

```
  getnextpc1 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
getnextpc1 2 R R R eval getinst getIndexFromPc
 getnextpc1 2 R R R getlocalv 2 R
and true == same? getstack getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc2 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R cboff getIndexFromPc
 getnextpc1 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and getstaybit getIndexFromPc getnextpc2 2 R R R == false
and true == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc2 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R cbon getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc1 2 R R
  stm getIndexFromPc getnextpc1 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == true
and getstaybit getIndexFromPc getnextpc2 2 R R R == false
and false == same? getstack getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 2 R R R eval getinst getIndexFromPc
  getnextpc1 2 R R R getlocalv 2 R .
```

```
bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc1 2 R R
  stm getIndexFromPc getnextpc2 2 R R cboff getIndexFromPc
 getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == false
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getstack 2 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc1 2 R R
  stm getIndexFromPc getnextpc1 2 R R cbon getIndexFromPc
 getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == false
and getstaybit getIndexFromPc getnextpc2 2 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getstack 2 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 2 R R R eval getinst getIndexFromPc
  getnextpc2 2 R R R getlocalv 2 R .

bceq mark-check R = cboff 2 staybiton 2 cbon getIndexFromPc
 getnextpc1 2 R R cbon getIndexFromPc getnextpc2 2 R R
  lvm getIndexFromPc getnextpc1 2 R R stm getIndexFromPc
 getnextpc1 2 R R lvm getIndexFromPc getnextpc2 2 R R
  stm getIndexFromPc getnextpc2 2 R R R
if getchangedbit 2 R == true
and getIndexFromPc getnextpc1 2 R R =/= 0
and getIndexFromPc getnextpc2 2 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 2 R R R == false
and getstaybit getIndexFromPc getnextpc2 2 R R R == false .

-- changed bit 3
```

```
bceq mark-check R = cboff 3 staybiton 3 R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R == 0
and getIndexFromPc getnextpc2 3 R R == 0 .

bceq mark-check R = cboff 3 staybiton 3
 cboff getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R == 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc2 3 R R lvm getIndexFromPc getnextpc2 3 R R
  stm getIndexFromPc getnextpc2 3 R R cbon
 getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R == 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and false == same? getstack getIndexFromPc
getnextpc2 3 R R R eval getinst getIndexFromPc
 getnextpc2 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon
 getIndexFromPc getnextpc2 3 R R lvm getIndexFromPc
  getnextpc2 3 R R stm getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R == 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 3 R R R == false .

bceq mark-check R = cboff 3 staybiton 3 cboff
 getIndexFromPc getnextpc1 3 R R R
```

```
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R == 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc1 3 R R lvm getIndexFromPc getnextpc1 3 R R
 stm getIndexFromPc getnextpc1 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R == 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
getnextpc1 3 R R R eval getinst getIndexFromPc
 getnextpc1 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc1 3 R R lvm getIndexFromPc getnextpc1 3 R R
  stm getIndexFromPc getnextpc1 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R == 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == false .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc1 3 R R cbon getIndexFromPc getnextpc2 3 R R
  lvm getIndexFromPc getnextpc1 3 R R stm getIndexFromPc
 getnextpc1 3 R R lvm getIndexFromPc getnextpc2 3 R R
  stm getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and false == same? getstack getIndexFromPc
```

```
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R
and false == same? getstack getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc2 3 R R lvm getIndexFromPc getnextpc2 3 R R
  stm getIndexFromPc getnextpc2 3 R R cboff getIndexFromPc
getnextpc1 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R
and false == same? getstack getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
getnextpc1 3 R R lvm getIndexFromPc getnextpc1 3 R R
 stm getIndexFromPc getnextpc1 3 R R cboff getIndexFromPc
  getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and false == same? getstack getIndexFromPc
```

```
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R
and true == same? getstack getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cboff getIndexFromPc
 getnextpc1 3 R R cboff getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R
and true == same? getstack getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst
  getIndexFromPc getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc2 3 R R lvm getIndexFromPc getnextpc2 3 R R
  stm getIndexFromPc getnextpc2 3 R R cboff
 getIndexFromPc getnextpc1 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and getstaybit getIndexFromPc getnextpc2 3 R R R == false
and true == same? getstack getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
```

```
and true == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc2 3 R R lvm getIndexFromPc getnextpc2 3 R R
  stm getIndexFromPc getnextpc2 3 R R cbon getIndexFromPc
 getnextpc1 3 R R lvm getIndexFromPc getnextpc1 3 R R
  stm getIndexFromPc getnextpc1 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == true
and getstaybit getIndexFromPc getnextpc2 3 R R R == false
and false == same? getstack getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 3 R R R eval getinst getIndexFromPc
  getnextpc1 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc1 3 R R lvm getIndexFromPc getnextpc1 3 R R
  stm getIndexFromPc getnextpc1 3 R R cboff getIndexFromPc
 getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == false
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getstack 3 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
getnextpc1 3 R R lvm getIndexFromPc getnextpc1 3 R R
 stm getIndexFromPc getnextpc1 3 R R
  cbon getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
```

```
and getstaybit getIndexFromPc getnextpc1 3 R R R == false
and getstaybit getIndexFromPc getnextpc2 3 R R R == true
and false == same? getstack getIndexFromPc
getnextpc2 3 R R R eval getinst
 getIndexFromPc getnextpc2 3 R R R getstack 3 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 3 R R R eval getinst getIndexFromPc
  getnextpc2 3 R R R getlocalv 3 R .

bceq mark-check R = cboff 3 staybiton 3 cbon getIndexFromPc
 getnextpc1 3 R R cbon getIndexFromPc getnextpc2 3 R R
  lvm getIndexFromPc getnextpc1 3 R R
 stm getIndexFromPc getnextpc1 3 R R
  lvm getIndexFromPc getnextpc2 3 R R
stm getIndexFromPc getnextpc2 3 R R R
if getchangedbit 3 R == true
and getIndexFromPc getnextpc1 3 R R =/= 0
and getIndexFromPc getnextpc2 3 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 3 R R R == false
and getstaybit getIndexFromPc getnextpc2 3 R R R == false .

-- changed bit 4

bceq mark-check R = cboff 4 staybiton 4 R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 0 R R == 0
and getIndexFromPc getnextpc2 0 R R == 0 .

bceq mark-check R = cboff 4 staybiton 4
 cboff getIndexFromPc getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R == 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc2 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R cbon getIndexFromPc
```

```
 getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R == 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc2 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R == 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 4 R R R == false .

bceq mark-check R = cboff 4 staybiton 4 cboff
 getIndexFromPc getnextpc1 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R == 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc1 4 R R
  stm getIndexFromPc getnextpc1 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R == 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
```

```
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc1 4 R R
  stm getIndexFromPc getnextpc1 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R == 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == false .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc1 4 R R cbon getIndexFromPc getnextpc2 4 R R
  lvm getIndexFromPc getnextpc1 4 R R stm getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
 getnextpc1 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R
and false == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst
  getIndexFromPc getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc2 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R cboff getIndexFromPc
 getnextpc1 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and true == same? getstack getIndexFromPc
```

```
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R
and false == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc1 4 R R
  stm getIndexFromPc getnextpc1 4 R R cboff getIndexFromPc
 getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and false == same? getstack getIndexFromPc
getnextpc1 4 R R R eval getinst getIndexFromPc
getnextpc1 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R
and true == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cboff getIndexFromPc
 getnextpc1 4 R R cboff getIndexFromPc getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getstack 4 R
```

```
and true == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R
and true == same? getstack getIndexFromPc
getnextpc2 4 R R R eval getinst getIndexFromPc
 getnextpc2 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc2 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R cboff getIndexFromPc
 getnextpc1 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and getstaybit getIndexFromPc getnextpc2 4 R R R == false
and true == same? getstack getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc2 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R cbon getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc1 4 R R
  stm getIndexFromPc getnextpc1 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == true
and getstaybit getIndexFromPc getnextpc2 4 R R R == false
and false == same? getstack getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 4 R R R eval getinst getIndexFromPc
  getnextpc1 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
```

```
 getnextpc1 4 R R lvm getIndexFromPc getnextpc1 4 R R
  stm getIndexFromPc getnextpc1 4 R R cboff getIndexFromPc
 getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == false
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc1 4 R R
  stm getIndexFromPc getnextpc1 4 R R cbon getIndexFromPc
 getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == false
and getstaybit getIndexFromPc getnextpc2 4 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getstack 4 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 4 R R R eval getinst getIndexFromPc
  getnextpc2 4 R R R getlocalv 4 R .

bceq mark-check R = cboff 4 staybiton 4 cbon getIndexFromPc
 getnextpc1 4 R R cbon getIndexFromPc getnextpc2 4 R R
  lvm getIndexFromPc getnextpc1 4 R R stm getIndexFromPc
 getnextpc1 4 R R lvm getIndexFromPc getnextpc2 4 R R
  stm getIndexFromPc getnextpc2 4 R R R
if getchangedbit 4 R == true
and getIndexFromPc getnextpc1 4 R R =/= 0
and getIndexFromPc getnextpc2 4 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 4 R R R == false
and getstaybit getIndexFromPc getnextpc2 4 R R R == false .

-- changed bit 5
```

```
bceq mark-check R = cboff 5 staybiton 5 R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R == 0
and getIndexFromPc getnextpc2 5 R R == 0 .

bceq mark-check R = cboff 5 staybiton 5 cboff
 getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R == 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc2 5 R R lvm getIndexFromPc getnextpc2 5 R R
  stm getIndexFromPc getnextpc2 5 R R cbon getIndexFromPc
 getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R == 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc2 5 R R lvm getIndexFromPc getnextpc2 5 R R
  stm getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R == 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 5 R R R == false .

bceq mark-check R = cboff 5 staybiton 5 cboff getIndexFromPc
 getnextpc1 5 R R R
if getchangedbit 5 R == true
```

```
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R == 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc1 5 R R
  stm getIndexFromPc getnextpc1 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R == 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc1 5 R R
  stm getIndexFromPc getnextpc1 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R == 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == false .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R cbon getIndexFromPc getnextpc2 5 R R
  lvm getIndexFromPc getnextpc1 5 R R stm getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc2 5 R R
  stm getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
```

```
  getnextpc1 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R
and false == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc2 5 R R lvm getIndexFromPc getnextpc2 5 R R
  stm getIndexFromPc getnextpc2 5 R R cboff getIndexFromPc
 getnextpc1 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and true == same? getstack getIndexFromPc
getnextpc1 5 R R R eval getinst getIndexFromPc
 getnextpc1 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R
and false == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc1 5 R R
 stm getIndexFromPc getnextpc1 5 R R cboff
  getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
```

```
  getnextpc1 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R
and true == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
 getnextpc2 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cboff getIndexFromPc
 getnextpc1 5 R R cboff getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R
and true == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc2 5 R R lvm getIndexFromPc getnextpc2 5 R R
  stm getIndexFromPc getnextpc2 5 R R cboff
 getIndexFromPc getnextpc1 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and getstaybit getIndexFromPc getnextpc2 5 R R R == false
and true == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
```

```
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc2 5 R R lvm getIndexFromPc getnextpc2 5 R R
  stm getIndexFromPc getnextpc2 5 R R cbon getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc1 5 R R
  stm getIndexFromPc getnextpc1 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == true
and getstaybit getIndexFromPc getnextpc2 5 R R R == false
and false == same? getstack getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 5 R R R eval getinst getIndexFromPc
  getnextpc1 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc1 5 R R
  stm getIndexFromPc getnextpc1 5 R R cboff getIndexFromPc
 getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == false
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc1 5 R R
  stm getIndexFromPc getnextpc1 5 R R cbon
 getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == false
```

```
and getstaybit getIndexFromPc getnextpc2 5 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getstack 5 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 5 R R R eval getinst getIndexFromPc
  getnextpc2 5 R R R getlocalv 5 R .

bceq mark-check R = cboff 5 staybiton 5 cbon getIndexFromPc
 getnextpc1 5 R R cbon getIndexFromPc getnextpc2 5 R R
  lvm getIndexFromPc getnextpc1 5 R R stm getIndexFromPc
 getnextpc1 5 R R lvm getIndexFromPc getnextpc2 5 R R
 stm getIndexFromPc getnextpc2 5 R R R
if getchangedbit 5 R == true
and getIndexFromPc getnextpc1 5 R R =/= 0
and getIndexFromPc getnextpc2 5 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 5 R R R == false
and getstaybit getIndexFromPc getnextpc2 5 R R R == false .

-- changed bit 6

bceq mark-check R = cboff 6 staybiton 6 R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R == 0
and getIndexFromPc getnextpc2 6 R R == 0 .

bceq mark-check R = cboff 6 staybiton 6 cboff
 getIndexFromPc getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R == 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc2 6 R R lvm getIndexFromPc getnextpc2 6 R R
  stm getIndexFromPc getnextpc2 6 R R cbon getIndexFromPc
 getnextpc2 6 R R R
if getchangedbit 6 R == true
```

```
and getIndexFromPc getnextpc1 6 R R == 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
 getnextpc2 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc2 6 R R lvm getIndexFromPc getnextpc2 6 R R
  stm getIndexFromPc getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R == 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc2 6 R R R == false .

bceq mark-check R = cboff 6 staybiton 6 cboff
 getIndexFromPc getnextpc1 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R == 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc1 6 R R
  stm getIndexFromPc getnextpc1 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R == 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R .
```

```
bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc1 6 R R
  stm getIndexFromPc getnextpc1 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R == 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == false .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc1 6 R R cbon getIndexFromPc getnextpc2 6 R R
  lvm getIndexFromPc getnextpc1 6 R R stm getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc2 6 R R
  stm getIndexFromPc getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R
and false == same? getstack getIndexFromPc
getnextpc2 6 R R R eval getinst getIndexFromPc
 getnextpc2 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc2 6 R R lvm getIndexFromPc getnextpc2 6 R R
  stm getIndexFromPc getnextpc2 6 R R cboff getIndexFromPc
getnextpc1 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
 getnextpc1 6 R R R getstack 6 R
```

```
and true == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R
and false == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc1 6 R R
 stm getIndexFromPc getnextpc1 6 R R cboff getIndexFromPc
  getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R
and true == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cboff getIndexFromPc
 getnextpc1 6 R R cboff getIndexFromPc getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
```

```
  getnextpc1 6 R R R getlocalv 6 R
and true == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc2 6 R R lvm getIndexFromPc getnextpc2 6 R R
 stm getIndexFromPc getnextpc2 6 R R cboff getIndexFromPc getnextpc1 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and getstaybit getIndexFromPc getnextpc2 6 R R R == false
and true == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc2 6 R R lvm getIndexFromPc getnextpc2 6 R R
  stm getIndexFromPc getnextpc2 6 R R cbon getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc1 6 R R
  stm getIndexFromPc getnextpc1 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == true
and getstaybit getIndexFromPc getnextpc2 6 R R R == false
and false == same? getstack getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc1 6 R R R eval getinst getIndexFromPc
  getnextpc1 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc1 6 R R
 stm getIndexFromPc getnextpc1 6 R R cboff
 getIndexFromPc getnextpc2 6 R R R
```

```
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == false
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and true == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst
  getIndexFromPc getnextpc2 6 R R R getstack 6 R
and true == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst
  getIndexFromPc getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon
 getIndexFromPc getnextpc1 6 R R lvm getIndexFromPc
  getnextpc1 6 R R stm getIndexFromPc getnextpc1 6 R R
 cbon getIndexFromPc getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == false
and getstaybit getIndexFromPc getnextpc2 6 R R R == true
and false == same? getstack getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getstack 6 R
and false == same? getlocalv getIndexFromPc
 getnextpc2 6 R R R eval getinst getIndexFromPc
  getnextpc2 6 R R R getlocalv 6 R .

bceq mark-check R = cboff 6 staybiton 6 cbon getIndexFromPc
 getnextpc1 6 R R cbon getIndexFromPc getnextpc2 6 R R
  lvm getIndexFromPc getnextpc1 6 R R stm getIndexFromPc
 getnextpc1 6 R R lvm getIndexFromPc getnextpc2 6 R R
  stm getIndexFromPc getnextpc2 6 R R R
if getchangedbit 6 R == true
and getIndexFromPc getnextpc1 6 R R =/= 0
and getIndexFromPc getnextpc2 6 R R =/= 0
and getstaybit getIndexFromPc getnextpc1 6 R R R == false
and getstaybit getIndexFromPc getnextpc2 6 R R R == false .

-- 全ての changedbit が false のとき
bceq mark-check R = R
if false == getchangedbit 0 R
and false == getchangedbit 1 R
and false == getchangedbit 2 R
```

```
and false == getchangedbit 3 R
and false == getchangedbit 4 R
and false == getchangedbit 5 R
and false == getchangedbit 6 R .

--
-- rule sets of exist? attribute
--

-- all getchangedbit attribution false.
-- getchagedbit が全て false のときは、exist?は true を返す
bceq exist? R = true
if false == getchangedbit 0 R
and false == getchangedbit 1 R
and false == getchangedbit 2 R
and false == getchangedbit 3 R
and false == getchangedbit 4 R
and false == getchangedbit 5 R
and false == getchangedbit 6 R .

-- others this condition
-- getchangedbit のどれかが true のときは、false を返す
bceq exist? R = false
if true == getchangedbit 0 R
or true == getchangedbit 1 R
or true == getchangedbit 2 R
or true == getchangedbit 3 R
or true == getchangedbit 4 R
or true == getchangedbit 5 R
or true == getchangedbit 6 R .

}

-- ---------------------------------------------------------------------
-- pseudo bytecode
-- ---------------------------------------------------------------------

-- 擬似的な bytecode

module* CODE {
protecting(RULE)

-- PC
beq getpc 0 init = 0 .
```

```
beq getpc 1 init = 1 .
beq getpc 2 init = 2 .
beq getpc 3 init = 3 .
beq getpc 4 init = 6 .
beq getpc 5 init = 7 .
beq getpc 6 init = 9 .

beq getIndexFromPc 0 init = 0 .
beq getIndexFromPc 1 init = 1 .
beq getIndexFromPc 2 init = 2 .
  beq getIndexFromPc 3 init = 3 .
beq getIndexFromPc 6 init = 4 .
beq getIndexFromPc 7 init = 5 .
beq getIndexFromPc 9 init = 6 .

-- instruction
beq getinst 0 init = "" .
beq getinst 1 init = "iconst_4" .
beq getinst 2 init = "istore_1" .
beq getinst 3 init = "iinc_1" .
beq getinst 4 init = "iload_1" .
beq getinst 5 init = "ifne" .
beq getinst 6 init = "return" .

-- stack
beq getstack 0 init = null .
beq getstack 1 init = null .
beq getstack 2 init = null .
beq getstack 3 init = null .
beq getstack 4 init = null .
beq getstack 5 init = null .
beq getstack 6 init = null .

-- local variables
beq getlocalv 0 init = null .
beq getlocalv 1 init = null .
beq getlocalv 2 init = null .
beq getlocalv 3 init = null .
beq getlocalv 4 init = null .
beq getlocalv 5 init = null .
beq getlocalv 6 init = null .

-- stay bit
beq getstaybit 0 init = true .
```

```
beq getstaybit 1 init = false .
beq getstaybit 2 init = false .
beq getstaybit 3 init = false .
beq getstaybit 4 init = false .
beq getstaybit 5 init = false .
beq getstaybit 6 init = false .

-- changed bit
beq getchangedbit 0 init = true .
beq getchangedbit 1 init = false .
beq getchangedbit 2 init = false .
beq getchangedbit 3 init = false .
beq getchangedbit 4 init = false .
beq getchangedbit 5 init = false .
beq getchangedbit 6 init = false .

-- next pc 1
  beq getnextpc1 0 init = 1 .
beq getnextpc1 1 init = 2 .
beq getnextpc1 2 init = 3 .
beq getnextpc1 3 init = 6 .
beq getnextpc1 4 init = 7 .
beq getnextpc1 5 init = 9 .
beq getnextpc1 6 init = 0 .

-- next pc 2
beq getnextpc2 0 init = 0 .
beq getnextpc2 1 init = 0 .
beq getnextpc2 2 init = 0 .
beq getnextpc2 3 init = 0 .
beq getnextpc2 4 init = 0 .
beq getnextpc2 5 init = 3 .
beq getnextpc2 6 init = 0 .
}


-- 以上の仕様をもとに
-- シミュレートさせる

open CODE .

--> typecheck
bred typecheck init .
--> exist?
```

```
bred exist? init .
--> typecheck
bred typecheck mark-check init .
--> exist?
bred exist? mark-check init .

--> typecheck
bred typecheck mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check mark-check mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check mark-check mark-check mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check mark-check
  mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check mark-check mark-check mark-check
 mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check mark-check
 mark-check mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check mark-check mark-check mark-check
 mark-check mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check mark-check
 mark-check mark-check mark-check mark-check init .
--> exist?
```

```
bred exist? mark-check mark-check mark-check mark-check mark-check
 mark-check mark-check mark-check init .

--> typecheck
bred typecheck mark-check mark-check mark-check mark-check
 mark-check mark-check mark-check mark-check mark-check init .
--> exist?
bred exist? mark-check mark-check mark-check mark-check mark-check
 mark-check mark-check mark-check mark-check init .

close

-- ------------------------------------------------------------------------
eof
-- ------------------------------------------------------------------------
```

参考文献

[1] 萩谷 昌己、Java 仮想機械手続きのための新しいデータフロー解析について、第一回プログラミングおよび応用のシステムに関するワークショップ論文集、日本ソフトウェア科学会、1998、
http://www.brl.ntt.co.jp/ooc/spa98/proceedings/016_hagiya.pdf
Masami Hagiya: On a New Method for Dataflow Analysis of Java
Virtual Machine Subroutines, IPSJ, PRO-17-3, pp.13-18, 1998.
http://nicosia.is.s.u-tokyo.ac.jp/pub/staff/hagiya/pro98/jvm.ps

[2] Zhenyu Qian,
A Formal Specification of Java(TM) Virual Machine Instructions for Objects, Methods and Subroutines, 1998 http://www.informatik.uni-bremen.de/~qian/abs-fsjvm.html

[3] Ataru T.Nagakawa,Toshimi Sawada,Kokichi Futatsugi, CafeOBJ User's Manual

[4] Tim Lidholm and Frank Yellin. The Java Virtual Machine Specification, Addison-Wesley.

[5] Jon Meyer and Troy Downing. Java Virtual Machine, O'Reilly & Associates, Inc.

[6] Gray McGraw and Edward W.Felten: Java Security: Hostile Applets, Holes and Antidotes, John Wiley and Sons, 1996

[7] Shusaku Iida, Michihiro Matsumoto, Răzvan Diaconescu,
Kokichi Futatsugi, and Dorel Lucanu Concurrent Object Composition in CafeOBJ, JAIST Research Report IS-RR-98-0009S

[8] Joseph A. Goguen and Grant Malcolm, Algebraic Semantics of Imperative Programs, The MIT Press Cambridge, Massachusetts London, England

[9] Low Level Security in Java Second edition, Tim Lindholm and Frank Yellin
http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html

[10] JAVA Virtual Machine The JAVA SERIES(tm),Jon Meyer and Troy Downing,
O'Reilly & Associates, Inc., 1997

[11] Secure Computing with Java: Now and the Future(a whitepaper),
http://www.javasoft.com/marketing/collateral/security.html