(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2007-226784
(P2007-226784A)
(43)公開日　平成19年9月6日(2007.9.6)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/44	(2006.01)	G06F	9/44	530A	5B081
G06F	9/45	(2006.01)	G06F	9/44	320C	

審査請求 有　請求項の数 20　OL　（全 15 頁）

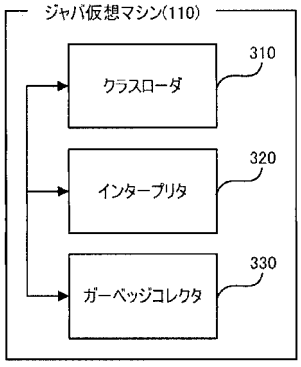
(21)出願番号　特願2007-26490(P2007-26490)
(22)出願日　平成19年2月6日(2007.2.6)
(31)優先権主張番号　10-2006-0018412
(32)優先日　平成18年2月24日(2006.2.24)
(33)優先権主張国　韓国(KR)

(71)出願人　390019839
三星電子株式会社
Ｓａｍｓｕｎｇ　Ｅｌｅｃｔｒｏｎｉｃｓ　Ｃｏ．，Ｌｔｄ．
大韓民国京畿道水原市八達区梅灘洞４１６番地
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(74)代理人　100107766
弁理士　伊東　忠重

最終頁に続く

(54)【発明の名称】インラインされたメソッドの呼出方法およびそれを用いたジャバ仮想マシン

(57)【要約】
【課題】　ジャバ仮想マシンにおいてインタープリタのメソッド呼出性能を向上させるインラインされたメソッドの呼出方法およびそれを用いたジャバ仮想マシンを提供する。
【解決手段】　第１メソッドが呼び出される場合、前記第１メソッドの実行に必要な情報を含むフレームを生成するステップと、所定の条件を満たす第２メソッドが呼び出される場合、前記生成されたフレームを用いて前記第２メソッドを実行するステップとを含む。
【選択図】　図３



【特許請求の範囲】
【請求項１】
第１メソッドが呼び出される場合、前記第１メソッドの実行に必要な情報を含むフレームを生成するステップと、
所定の条件を満たす第２メソッドが呼び出される場合、前記生成されたフレームを用いて前記第２メソッドを実行するステップとを含む、インラインされたメソッドの呼出方法。
【請求項２】
前記フレームは、実行中のバイトコードの位置を指示するプログラムカウンタと、実行中のメソッドが終了する時にリターンするバイトコードの位置を指示するリターンアドレスと、前記実行中のメソッドが用いるオペランドスタックとを含む、請求項１に記載のインラインされたメソッドの呼出方法。
【請求項３】
前記第２メソッドを実行するステップは、
前記第２メソッドを呼び出すバイトコードの次のバイトコードの位置情報を前記リターンアドレスとして格納するステップと、
前記第２メソッドの１番目のバイトコードの位置に前記プログラムカウンタを設定するステップと、
前記オペランドスタックを用いて前記第２メソッドを実行するステップとを含む、請求項２に記載のインラインされたメソッドの呼出方法。
【請求項４】
前記第１メソッドを呼び出すステップと、
前記第１メソッドが用いるオペランドスタックを割り当てるステップとをさらに含む、請求項１に記載のインラインされたメソッドの呼出方法。
【請求項５】
前記オペランドスタックは、前記第１メソッドの実行に必要な大きさより所定の大きさ分さらに大きい、請求項４に記載のインラインされたメソッドの呼出方法。
【請求項６】
前記第２メソッドは前記第１メソッドから呼び出され、他のメソッドは呼び出さないメソッドである、請求項１に記載のインラインされたメソッドの呼出方法。
【請求項７】
前記第２メソッドは前記第１メソッドから派生したメソッドの呼出経路上に存在するメソッドであり、前記メソッドの呼出経路は臨界レベル以下のメソッドの呼出レベルで構成される、請求項１に記載のインラインされたメソッドの呼出方法。
【請求項８】
前記第２メソッドは、前記第２メソッドを実行するために必要なオペランドスタックの大きさが臨界大きさ未満のメソッドである、請求項７に記載のインラインされたメソッドの呼出方法。
【請求項９】
所定のクラスファイルをロードするステップと、
前記クラスファイルに含まれたメソッドのうち前記第１メソッドとして動作できるメソッドと、前記第２メソッドとして動作できるメソッドに関する情報を提供するステップとをさらに含む、請求項１に記載のインラインされたメソッドの呼出方法。
【請求項１０】
所定のクラスファイルをロードするクラスローダと、
前記ロードされたクラスファイルに含まれたメソッドのうち、第１メソッドが呼び出される場合は前記第１メソッドの実行に必要な情報を含むフレームを生成し、所定の条件を満たす第２メソッドが呼び出される場合は前記生成されたフレームを用いて前記第２メソッドを実行するインタープリタとを含む、ジャバ仮想マシン。
【請求項１１】

前記フレームは、実行中のバイトコードの位置を指示するプログラムカウンタと、実行中のメソッドが終了する時にリターンするバイトコードの位置を指示するリターンアドレスと、前記実行中のメソッドが用いるオペランドスタックとを含む、請求項１０に記載のジャバ仮想マシン。

【請求項１２】

前記インタープリタは、前記第２メソッドを呼び出すバイトコードの次のバイトコードの位置情報を前記リターンアドレスとして格納し、前記第２メソッドの１番目のバイトコードの位置に前記プログラムカウンタを設定し、前記オペランドスタックを用いて前記第２メソッドを実行する、請求項１１に記載のジャバ仮想マシン。

【請求項１３】

前記インタープリタは、前記第１メソッドを呼び出し、前記第１メソッドが用いるオペランドスタックを割り当てる、請求項１０に記載のジャバ仮想マシン。

【請求項１４】

前記オペランドスタックは、前記第１メソッドの実行に必要な大きさより所定の大きさ分さらに大きい、請求項１３に記載のジャバ仮想マシン。

【請求項１５】

前記第２メソッドは、前記第１メソッドから呼び出され、他のメソッドは呼び出さないメソッドである、請求項１０に記載のジャバ仮想マシン。

【請求項１６】

前記第２メソッドは前記第１メソッドから派生したメソッドの呼出経路上に存在するメソッドであり、前記メソッドの呼出経路は臨界レベル以下のメソッド呼出レベルで構成される、請求項１０に記載のジャバ仮想マシン。

【請求項１７】

前記第２メソッドは、前記第２メソッドを実行するために必要なオペランドスタックの大きさが臨界大きさ未満のメソッドである、請求項１６に記載のジャバ仮想マシン。

【請求項１８】

前記クラスローダは、前記クラスファイルに含まれたメソッドのうち前記第１メソッドとして動作できるメソッドと、前記第２メソッドとして動作できるメソッドに関する情報を提供する、請求項１０に記載のジャバ仮想マシン。

【請求項１９】

ジャバ仮想マシンと、

前記ジャバ仮想マシンが用いるメモリを提供する主記憶部と、

前記ジャバ仮想マシンに所定のデータを入力し、前記ジャバ仮想マシンから所定の作業結果が出力される入出力部とを含み、

前記ジャバ仮想マシンは、

所定のクラスファイルをロードするクラスローダと、

前記ロードされたクラスファイルに含まれたメソッドのうち、第１メソッドが呼び出される場合は前記第１メソッドの実行に必要な情報を含むフレームを生成し、所定の条件を満たす第２メソッドが呼び出される場合は前記生成されたフレームを用いて前記第２メソッドを実行するインタープリタとを含む、デジタル演算装置。

【請求項２０】

請求項１ないし請求項９のうちいずれか１項に記載の方法を実行するためのコンピュータによって読取可能なプログラムを記録した記録媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、ジャバ仮想マシンに関し、より詳しくは、インラインされたメソッドの呼出方法およびそれを用いたジャバ仮想マシンに関するものである。

【背景技術】

【０００２】

ハードウェアプラットフォームに依存するアプリケーションプログラムを開発するためには、同一作業を行うアプリケーションプログラムであっても各々のハードウェアプラットフォームに適するように別途に設計しなければならない不便さがある。よって、ハードウェアプラットフォームに依存しないアプリケーションプログラムに対する要求があり、このような要求を充足させるために、ハードウェアプラットフォームを抽象化した仮想マシンが開発された。仮想マシンのうち代表的なものがジャバ仮想マシン（Ｊａｖａ（登録商標）ｖｉｒｔｕａｌ　ｍａｃｈｉｎｅ）である。

【０００３】

ジャバは、プラットフォーム独立の実行ファイルを生成するために、ソースコードをジャバだけの独特のバイトコード形式でコンパイルし、ジャバ仮想マシンにて行わせる。

【０００４】

従来のジャバ仮想マシンを、実行エンジンを基準に分類すれば、インタープリート方式のジャバ仮想マシン、ジャストインタイム（ＪＩＴ）コンパイル方式のジャバ仮想マシン、インタープリートとＪＩＴコンパイルを共に用いる方式のジャバ仮想マシン、またはアヘッドオフタイム（ＡＯＴ）コンパイル方式のジャバ仮想マシンなどに分類することができる。

【０００５】

このうちインタープリート方式のジャバ仮想マシンでは、行うべきメソッドのバイトコードをインタープリタが１つずつ解析することによってアプリケーションプログラムが動作することができる。インタープリタは実行中であるメソッドに関する情報をフレームという資料構造によって管理する。

【０００６】

呼び出されるメソッドが十分長くて複雑である場合には、新たなフレームを作ることにより、各メソッドに対して独立したフレームを格納する必要が充分であると言える。しかし、呼び出されるメソッドが非常に簡単なメソッドである場合（例えば、セッター関数やゲッター関数のように単にクラス内のフィールドにアクセスして必要である値を読み取ったり書き込んだりするメソッド）、このようなメソッドを呼び出すために毎度新たなフレームを作る作業が行われるのは非効率的であり、全体作業時間を延ばせる結果を招く。

【０００７】

特許文献１では、ジャバ仮想マシンにおいてクラスファイルのロード過程とコンパイル過程を同時に行い、アプリケーションプログラムの実行性能を向上させる技術を開示している。しかし、特許文献１では、コンパイラを用いてジャバプログラムの処理速度を向上させる技術であるため、インタープリタを用いるジャバ仮想マシンには適用し難いところがある。

【特許文献１】韓国公開特許第１０－２００５－００７４７６６号公報

【発明の開示】

【発明が解決しようとする課題】

【０００８】

本発明の目的は、ジャバ仮想マシンにおいてインタープリタのメソッド呼出性能を向上させることにある。

【０００９】

本発明の目的は、以上で言及した目的に制限されず、言及していないまた他の目的は下記記載によって当業者が明確に理解できるものである。

【課題を解決するための手段】

【００１０】

前記目的を達成するために、本発明の一態様に係るインラインされたメソッドの呼出方法は、第１メソッドが呼び出される場合、前記第１メソッドの実行に必要な情報を含むフレームを生成するステップと、所定の条件を満たす第２メソッドが呼び出される場合、前記生成されたフレームを用いて前記第２メソッドを実行するステップとを含む。

【００１１】

前記目的を達成するために、本発明の他の態様に係るジャバ仮想マシンは、所定のクラスファイルをロードするクラスローダと、前記ロードされたクラスファイルに含まれたメソッドのうち第１メソッドが呼び出される場合は前記第１メソッドの実行に必要な情報を含むフレームを生成し、所定の条件を満たす第２メソッドが呼び出される場合は前記生成されたフレームを用いて前記第２メソッドを実行するインタープリタとを含む。

【発明の効果】

【００１２】

本発明のインラインされたメソッドの呼出方法およびそれを用いたジャバ仮想マシンによれば、ジャバ仮想マシンにおいてインタープリタのメソッド呼出性能を向上させる効果がある。

【発明を実施するための最良の形態】

【００１３】

その他、実施形態の具体的な事項は、詳細な説明および図面に含まれている。
本発明の利点および特徴、そしてそれらを達成する方法は、添付する図面と共に詳細に後述する実施形態を参照すれば明確になる。しかし、本発明は以下にて開示する実施形態に限定されず、互いに異なる多様な形態によって実施され、単に本実施形態は本発明の開示が完全なものとなるようにし、本発明が属する技術分野で通常の知識を有する者に発明の範疇を完全に知らせるために提供されるものであって、本発明は請求項の範疇によってのみ定義されるものである。明細書の全体にわたり、同一参照符号は同一構成要素を示す。

【００１４】

以下、添付する図面を参照し、本発明の好ましい実施形態についてより詳細に説明する。

【００１５】

図１は、本発明の一実施形態に係るジャバ仮想マシンが搭載されたデジタル演算装置の構成図である。デジタル演算装置１００は、ジャバ仮想マシン１１０、主記憶部１２０、入出力部１３０、および制御部１４０を含む。

【００１６】

ジャバ仮想マシン１１０は、デジタル演算装置１００において必要とする特定機能を行う。例えば、デジタル演算装置１００が放送受信装置である場合、ジャバ仮想マシン１１０は受信した放送情報を見せる機能を行うことができる。他の例として、デジタル演算装置１００がインターネット通信装置である場合、ジャバ仮想マシン１１０は、ウェブブラウザを駆動させたり、アプレット、ミドレット、エクスレット等を実行する機能を提供したりすることができる。この他にもジャバ仮想マシン１１０は、多様な形態のジャバアプリケーションを実行することができる。

【００１７】

ジャバ仮想マシン１１０は、その動作を行うためにクラスファイル２００をロードするが、本発明の一実施形態に係るクラスファイル２００は図２に示している。

【００１８】

クラスファイル２００は、クラスファイル２００の基本属性を表わすクラスヘッダ２１０、メソッドを実行するための常数を含む常数領域（コンスタントプール）２２０、クラスファイル２００のインターフェースとフィールドに関する情報を含むインターフェース部分２３０とフィールド部分２４０、および１つ以上のメソッド２５０を含む。各メソッドは所定のオペコードとオペランドを含む。

【００１９】

この他にもクラスファイル２００は公知の他の情報をさらに含むことができる。

【００２０】

再び図１を参照すれば、主記憶部１２０はジャバ仮想マシン１１０の動作時に必要なスタックとバッファなどが具現され得るメモリを提供する。本発明の一実施形態によれば、主記憶部１２０はラムとして具現され得、ラムの種類としてはＤＲＡＭ、ＳＲＡＭ、ＳＤＲＡＭなどを挙げることができる。

【００２１】
入出力部１３０は、ジャバ仮想マシン１１０の動作に必要なデータを提供し、ジャバ仮想マシン１１０の作業結果が出力される。例えば、クラスファイル２００は、ネットワーク、外部装置、またはデジタル演算装置１００の補助記憶装置（図示せず）等、多様な場所に存在することができるが、クラスファイル２００を得るために入出力部１３０は、ネットワークインターフェース、外部装置インターフェース、補助記憶装置インターフェースなどを含むことができる。また、入出力部１３０は、ジャバ仮想マシン１１０の作業結果を所定のディスプレイやスピーカによって出力させることもできる。
【００２２】
制御部１４０は、前述した構成要素１１０～１３０間の動作を制御する。
【００２３】
図１を参照して説明したデジタル演算装置１００は、携帯電話、ＰＤＡ、デジタルＴＶ、セットトップボックス、ノートパソコン、デスクトップパソコンなどのデータ格納能力とプロセシング能力を有する多様な電子製品として具現され得る。以下、本発明の一実施形態に係るジャバ仮想マシン１１０についてより具体的に説明する。
【００２４】
図３は、本発明の一実施形態に係るジャバ仮想マシン１１０を示すブロック図である。ジャバ仮想マシン１１０は、クラスローダ３１０、インタープリタ３２０、およびガーベッジコレクタ３３０を含む。
【００２５】
クラスローダ３１０は、クラスファイル２００をロードし、クラスファイル２００に含まれた各メソッド２５０に対応するメソッドブロックを生成する。メソッドブロックは、メソッドに関する情報を含むが、本発明の一実施形態に係るメソッドブロックは図４に示している。
【００２６】
図４に示すメソッドブロック４００は、メソッドを構成するバイトコード４１０、メソッドが属するクラス情報４２０、メソッドの大きさ４３０、メソッドの開始位置４４０、およびメソッド類型情報４５０を含む。ここで、メソッド類型情報４５０は、メソッドブロックに対応するメソッドが一般メソッド、メインメソッド、およびインラインされたメソッドのうち、どのメソッドであるかを示す情報である。ここで、一般メソッドは自身のみのためのフレームを有する。また、メインメソッドはインラインメソッドと共有するフレームを有し、インラインメッソドはメインメソッドのフレームを共有するようになる。ここでメインメソッドは構成や動作の面で一般メソッドと実質的に異ならないが、説明の便宜上、インラインされたメソッドとフレームを共有するメソッドとそうでないメッソドとを区分できるように表すために一般メソッドと異なる名称（メインメソッド）を用いる。但し、一般メソッドには一般メソッドを実行するために必要な大きさのオペランドスタックが割り当てられるが、メインメソッドにはメインメソッドを実行するために必要な大きさより所定の大きさ分さらに大きいオペランドスタックが割り当てられるという点で相異する。
【００２７】
メソッドブロック４００を生成するために、クラスローダ３１０はロードされたクラスファイルに対して検証作業を行うことができる。検証作業の時、クラスローダ３１０はクラスファイル２００の構造と内容が正しく作成されているか否かを判別することができる。特に、クラスローダ３１０は、クラスファイル２００に含まれたメソッド２５０に対する検証により、各メソッドに含まれたバイトコードが全て有効であるか否かを検査する。
【００２８】
また、本発明の一実施形態によれば、クラスローダ３１０は、クラスファイル２００を検証する時、各メソッドがメインメソッドとインラインされたメソッドとして動作できるか否かを判断し、メインメソッドやインラインされたメソッドのメソッドブロックには各々メインメソッドやインラインされたメソッドであることを示すメソッド類型情報を設定

し、それ以外のメソッドのメソッドブロックには一般メソッドであることを示すメソッド類型情報を設定する。どのようなメソッドがメインメソッドとインラインされたメソッドとして設定されるかに関する内容は図６～図９の説明によって明らかになる。

【００２９】

再び図３を参照すれば、インタープリタ３２０は、クラスローダ３１０がロードするクラスファイル２００に含まれたメソッド２５０を実行させる。より具体的には、インタープリタ３２０はメソッドを構成するバイトコードを認識し、認識されたバイトコードに対応するハンドラを実行させる。

【００３０】

インタープリタ３２０は、現在実行中のメソッドに関する情報をフレームという資料構造を用いて管理する。したがって、メソッドを呼び出す時、インタープリタ３２０は新たなフレームを生成し、メソッドの実行に必要な情報を管理する。ところが、呼び出されるメソッドがインラインされたメソッドであれば、インタープリタ３２０は新たなフレームを生成せず、メインメソッドのフレームを用いてインラインされたメソッドを実行する。

【００３１】

本発明の一実施形態に係るフレームの構造は図５に示している。フレーム５００は、メソッドブロックポインタ５１０、プログラムカウンタ情報５２０、リターンアドレス５３０、トップオブスタック（ＴｏｐＯｆＳｔａｃｋ）情報５４０、およびオペランドスタック５５０を含む。

【００３２】

メソッドブロックポインタ５１０は実行中のメソッドに対応するメソッドブロックを指示し、プログラムカウンタ情報５２０は実行中のバイトコードの位置情報を格納する。リターンアドレス５３０は現在実行中のメソッドを終了する時にリターンするバイトコードの位置を指示し、トップオブスタック情報５４０はインタープリタ３２０が用いるオペランドスタック５５０の最上位の位置情報を格納する。オペランドスタック５５０はメソッドを実行するために必要なオペランドの格納空間である。

【００３３】

プログラムカウンタ情報５２０とトップオブスタック情報５４０とは、メソッドの呼出やガーベッジコレクタ３３０の呼出、またはジャバの例外処理のようにメソッドの実行をしばらく中断して他の作業を処理しなければならない場合、該当メソッドの実行中の状態を保持するためにフレーム５００に格納されるものであるため、プログラムカウンタ情報５２０とトップオブスタック情報５４０とは、バイトコードを実行する度毎にフレーム５００に格納される必要はない。

【００３４】

呼び出されるメソッドがメインメソッドであれば、インタープリタ３２０はフレームを生成する時、メインメソッドを実行するために必要な大きさより所定の大きさ分さらに大きいメモリ空間にオペランドスタックを割り当てる。これは、メインメソッドのフレームを用いてインラインされたメソッドを実行する場合、インラインされたメソッドの実行に必要な大きさのオペランドスタックを確保するためである。メインメソッドのフレームを生成する時、メインメソッドの実行に必要な大きさよりどれ程さらに大きいオペランドスタックを割り当てるかの問題は、本発明を限定せず、実施形態によって多様な具現が可能である。

【００３５】

一方、呼び出されるメソッドがインラインされたメソッドである場合、インタープリタ３２０はインラインされたメソッドを実行するためにメインメソッドのフレームを使用する。すなわち、インラインされたメソッドはメインメソッドのフレームを共有する。このためにインタープリタ３２０は、インラインされたメソッドを呼び出す時、実行中であったメソッドのバイトコードのうちインラインされたメソッドを呼び出したバイトコードの次のバイトコードの位置情報をフレームにてリターンアドレスとして格納し、インラインされたメソッドの１番目のバイトコードの位置にプログラムカウンタを移動させる。その

後、インタープリタ３２０は、インラインされたメソッドを実行するためにメインメソッドのフレームに含まれたオペランドスタックを用いる。

【００３６】

呼び出されるメソッドが一般メソッドであれば、インタープリタ３２０は従来の技術のようにフレームを生成し、呼び出されたメソッドを実行する。

【００３７】

再び図３を参照すれば、ガーベッジコレクタ３３０はインタープリタ３２０がメソッドを実行する時に用いるメモリのうち必要のないメモリを再使用できるように収集する。

【００３８】

この他にもジャバ仮想マシン１１０は、所定機能を実行する公知の構成要素をさらに含むことができる。

【００３９】

以上図３を参照して説明したジャバ仮想マシン１１０の構成要素（３１０～３３０）は一種のモジュールとして具現され得る。ここでモジュールは、ソフトウェア、ＦＰＧＡ（Ｆｉｅｌｄ Ｐｒｏｇｒａｍｍａｂｌｅ Ｇａｔｅ Ａｒｒａｙ）または特定用途向け集積回路（Ａｐｐｌｉｃａｔｉｏｎ Ｓｐｅｃｉｆｉｃ Ｉｎｔｅｇｒａｔｅｄ Ｃｉｒｃｕｉｔ、ＡＳＩＣ）のようなハードウェア構成要素を意味し、モジュールはある機能を行う。ただし、モジュールはソフトウェアまたはハードウェアに限定される意味ではない。モジュールは、アドレッシングできる格納媒体にあるように構成することもでき、１つまたはそれ以上のプロセッサを再生させるように構成することもできる。したがって、一例としてモジュールはソフトウェア構成要素、オブジェクト指向ソフトウェア構成要素、クラス構成要素およびタスク構成要素と同じ構成要素と、プロセス、関数、属性、プロシージャ、サブルーチン、プログラムコードのセグメント、ドライバ、ファームウェア、マイクロコード、回路、データ、データベース、データ構造、テーブル、アレイ、および変数を含む。構成要素とモジュールの中で提供されている機能は、さらに小さい数の構成要素およびモジュールに結びついたり追加的な構成要素とモジュールにさらに分離できる。

【００４０】

図６は、本発明の一実施形態に係るメソッドブロックの提供過程を示すフローチャートである。

【００４１】

クラスローダ３１０は、ネットワーク、外部装置、または補助記憶装置等からクラスファイル２００をロードする（Ｓ６１０）。

【００４２】

その後、クラスローダ３１０は、ロードされたクラスファイル２００に含まれたメソッド２５０のメソッド類型を判別する（Ｓ６２０）。これによってメソッド２５０は、一般メソッド、メインメソッド、およびインラインされたメソッドのうちいずれか１つに設定され得る。

【００４３】

ここでメソッドの類型を判別する基準は実施形態によって多様に設定され得る。本発明の一実施形態として、メインメソッドは自身そのものから派生し、臨界レベル以下のメソッド呼出レベルからなるメソッドの呼出経路が存在するメソッドであり得る。ここで、臨界レベルは実施形態によって多様な値に設定され得る。また、インラインされたメソッドは、メインメソッドから派生し、臨界レベル以下のメソッド呼出レベルからなるメソッドの呼出経路上に存在するメソッドであり得る。

【００４４】

例えば、図７Ａに示すメソッドの呼出構造を参照すれば、メソッドＡはメソッドＢを呼び出す。したがって、メソッドＢはメソッドＡから派生したメソッドの呼出経路上に存在し、メソッドＡからメソッドＢへつながるメソッドの呼出経路のメソッド呼出レベルは「１」である。

【００４５】

他の例として、図７Ｂに示すメソッドの呼出構造を参照すれば、メソッドＣはメソッドＤを呼出し、メソッドＤはメソッドＥを呼び出す。したがって、メソッドＤとメソッドＥは、メソッドＣから派生した第１メソッドの呼出経路上に存在する。ここで第１メソッドの呼出経路はメッソドＣからメソッドＤを経てメソッドＥまで続くため、第１メソッドの呼出経路のメソッドの呼出レベルは「２」となる。図７Ｂに示すメソッドの呼出構造においてメソッドＤとメソッドＣのみの関係を考慮すれば、メソッドＥはメソッドＤから派生した第２メソッドの呼出経路上に存在することもある。ここで第２メソッドの呼出経路のメソッド呼出レベルは１になる。

【００４６】

臨界レベルが「１」と設定されていれば、図７Ｂに示す実施形態において臨界レベル以下のメソッド呼出レベルからなるメソッドの呼出経路は第２メソッドの呼出経路である。この場合、メソッドＤはメインメソッドになり、メソッドＥは第２メソッドの呼出経路によってメソッドＤに対してインラインされたメソッドになることができる。

【００４７】

しかし、臨界レベルが「２」と設定されていれば、図７Ｂに示す実施形態において第１メソッドの呼出経路と第２メソッドの呼出経路が共に臨界レベル以下のメソッド呼出レベルからなる。この場合、第２メソッドの呼出経路を派生させたメソッドＤは、第１メソッドの呼出経路を派生させたメソッドＣから呼び出されるため、クラスローダ３１０は、第１メソッドの呼出経路に基づいてメインメソッドとインラインされたメソッドを決定する。すなわち、臨界レベル以下のメソッド呼出レベルからなる複数のメソッドの呼出経路が重なる場合、クラスローダ３１０は、メソッド呼出レベルが最も高いメソッドの呼出経路を基準にし、メインメソッドとインラインされたメソッドを決定する。これにより臨界レベルが「２」と設定されていれば、図７Ｂに示す実施形態においてメソッドＣがメインメソッドと設定され、メソッドＤとメソッドＥはメソッドＡに対してインラインされたメソッドと設定される。ここで、第２メソッドの呼出経路を独立的に見ると、メソッドＤがメソッドＥに対してメインメソッドになり得ると見られるが、前述したようにメソッドＤは、既にメソッドＣにインラインされたメソッドであるため、メインメソッドとして設定されることができない。

【００４８】

本発明の他の実施形態として、メソッドの類型を判別する基準としてメソッドの実行に必要なオペランドスタックの大きさが考慮され得る。すなわち、クラスローダ３１０は、実行に必要なオペランドスタックが閾値の大きさ以下であり、他のメソッドから呼び出されるメソッドをインラインされたメソッドと設定でき、インラインされたメソッドを呼び出すメソッドをメインメソッドと設定することもできる。この場合、特定値を読み取るようにするゲッター関数や特定値を変更させるセッター関数のような簡単なメソッドを呼び出す場合は、フレームを生成するのに必要な時間と資源を節約できる。

【００４９】

好ましくは、クラスローダ３１０は、これ以上他のメソッドを呼び出さないメソッドであるリーフメソッド（ｌｅａｆ　ｍｅｔｈｏｄ）をインラインメソッドと設定し、リーフメソッドを直接呼び出すメソッドをメインメソッドと設定することができる。

【００５０】

再び図６を参照すれば、クラスローダ３１０は、メソッド類型の判別結果に応じ、メソッド類型情報を含んで各メソッドに対応するメソッドブロックを生成する（Ｓ６３０）。生成されたメソッドブロックは、主記憶部１２０が提供する所定のメモリ領域に格納され得、メソッドブロックの実施形態は図４を参照して説明した通りである。

【００５１】

図６の過程に示してはないが、クラスローダ３１０は、検証作業、準備作業、および分解（ｒｅｓｏｌｕｔｉｏｎ）作業を行うことができる。

【００５２】

検証作業は、クラスファイル２００の構造と内容が正しく作成されているか否かを判別

する過程であり、検証作業によってクラスファイル２００内のメソッド２５０のメソッド呼出関係および必要なオペランドスタックの大きさなどが分かる。

【００５３】

準備作業はクラスファイル２００に定義された常数や変数のデフォルト値を指定する過程であり、分解過程は常数領域２２０に明示されたシンボルをクラスの実行中に必要なリソースの物理的な位置情報に置き換えるステップである。

【００５４】

前述した過程が終了すれば、所定のメモリ領域にロードされたクラスファイルを初期化した後にメソッド２５０を実行する作業が行われるが、これについては図８を参照して説明する。

【００５５】

図８は、本発明の一実施形態に係るメソッドの呼出過程を示すフローチャートである。図８の過程では、特定メソッドのために生成されたフレームが既に存在する場合を仮定する。

【００５６】

所定のメソッドが呼び出されれば（Ｓ８１０）、インタープリタ３２０は呼び出されたメソッドがインラインされたメソッドであるか否かを判断する（Ｓ８１５）。インラインされたメソッドであるか否かは、呼び出されたメソッドに対応するメソッドブロックのメソッド類型情報によって確認できる。

【００５７】

呼び出されたメソッドがインラインされたメソッドでなければ、インタープリタ３２０は呼び出されたメソッドにオペランドスタックを割り当てる（Ｓ８２０）。ここで呼び出されたメソッドが一般メソッドであれば、インタープリタ２０は呼び出されたメソッドの実行に必要な大きさのオペランドスタックを割り当てることができる。しかし、呼び出されたメソッドがメインメソッドであれば、インタープリタ３２０は呼び出されたメソッドの実行に必要な大きさより所定サイズほどさらに大きいオペランドスタックを割り当てることができる。

【００５８】

その後、インタープリタ３２０は呼び出されたメソッドのためのフレームを生成し（Ｓ８２５）、生成されたフレームのプログラムカウンタ情報５２０とトップオブスタック情報５４０としてメソッド呼出を指示したバイトコードの次のバイトコードの位置情報と実行中であったメソッドに割り当てられたオペランドスタックの最上位の位置情報とを各々格納する（Ｓ８３０）。格納された情報は呼び出されたメソッドの終了後リターン過程に用いられる。

【００５９】

その後、インタープリタ３２０は、プログラムカウンタとトップオブスタックを呼び出されたメソッドの１番目のバイトコードと呼び出されたメソッドに割り当てられたオペランドスタックの最上位の位置に移動させ（Ｓ８３５）、呼び出されたメソッドを実行する（Ｓ８４０）。

【００６０】

一方、ステップＳ８１５の判断結果、呼び出されたメソッドがインラインされたメソッドであれば、インタープリタ３２０は新たなメソッドの呼出を指示したバイトコードの次のバイトコードの位置情報を既存フレームのリターンアドレス５３０として格納し（Ｓ８４５）、呼び出されたメソッドの１番目のバイトコードの位置にプログラムカウンタを移動させる（Ｓ８５０）。ここで、トップオブステクは移動させないが、これはインラインされたメソッドがメインメソッドのオペランドスタックを共有するためである。インラインされたメソッドを呼び出したメソッドも又インラインされたメソッドであったのであれば、既存フレーム上に既にリターンアドレス５３０が格納されているはずであって、インタープリタ３２０は既に設定されたリターンアドレスと区別されるまた他の１つのリターンアドレスを既存のフレーム上に格納する。すなわち、１つのメインメソッドに対して複

数のインラインされたメソッドが存在すれば、インタープリタ３２０はメインメソッドのフレームにリターンアドレス５３０を複数格納することもできる。

【００６１】

その後、インタープリタ３２０は既存のフレームのオペランドスタックを用い、呼び出されたメソッドを実行する（Ｓ８５５）。

【００６２】

図８は、前述したように、特定メソッドのために生成されたフレームが既に存在する場合を仮定したものである。最初のメソッドが実行される場合であれば、最初のメソッドはメインメソッドと一般メソッドのうちいずれか１つであるため、図８のステップＳ８１０～Ｓ８４０を参照すれば、最初メソッドの実行過程を理解できる。

【００６３】

一方、呼び出されたメッセージの実行が終了して既存のメッセージにリターンする過程は、一般メソッドとメインメソッドに対しては従来のように遂行され得る。その反面、インラインされたメソッドの場合は、リターン過程においてフレームのリターンアドレス５３０を参照する。

【００６４】

図９は、本発明の一実施形態に係るメソッドの呼出状態とそれに伴うフレームの管理状態を示す図面である。示された実施形態においてメソッドａは一般メソッドで、メソッドｂはメインメソッドで、メソッドｃはインラインされたメソッドであり、各メソッド内のブロックはバイトコードを示す。

【００６５】

メソッドａが呼び出された場合、インタープリタ３２０はメソッドａのためにフレームａを生成し、フレームａを用いてメソッドａを実行する。メソッドａを呼び出す時、インタープリタ３２０はメソッドａの実行のために必要な大きさのオペランドスタックを割り当てる。

【００６６】

メソッドａの実行中にメソッドｂを呼び出すバイトコード９１０によってメソッドｂが呼び出されれば、インタープリタ３２０はメソッドｂのためにフレームｂを生成し、フレームｂを用いてメソッドｂを実行する。メソッドｂはメインメソッドであるため、メソッドｂを呼び出す時、インタープリタ３２０はメソッドｂの実行のために必要な大きさより所定の大きさ分さらに大きいオペランドスタックを割り当てる。

【００６７】

一方、インタープリタ３２０は、メソッドａからメソッドｂを呼び出すバイトコード９１０の次のバイトコード９２０の位置情報をフレームｂのプログラムカウンタ情報９３０として格納する。また、インタープリタ３２０は、プログラムカウンタをメソッドｂの１番目のバイトコード９４０に移動させ、メソッドｂに割り当てられたオペランドスタックの最上位の位置にトップオブスタックを移動させる。

【００６８】

メソッドｂの実行中にメソッドｃを呼び出すバイトコード９５０によってメソッドｃが呼び出されれば、インタープリタ３２０はフレームｂをそのまま用いてメソッドｃを実行する。メソッドｃはインラインされたメソッドであるため、メソッドｃのためのフレームは別途生成しない。

【００６９】

メソッドｃを呼び出す時、インタープリタ３２０はフレームｂからメソッドｃを呼び出すバイトコード９５０の次のバイトコード９６０の位置情報をリターンアドレス９７０として格納する。また、インタープリタ３２０はメソッドｃの１番目のバイトコード９８０にプログラムカウンタを移動させる。メソッドｃを実行する時にはフレームｂのオペランドスタック９９０が用いられる。

【００７０】

メソッドｃの実行が終了すれば、インタープリタ３２０はプログラムカウンタをフレー

ムｂのリターンアドレス９７０として格納されたバイトコードの位置情報を用いて移動させ、該当バイトコード９６０からメソッドｂを実行する。

【００７１】

メソッドｂの実行が終了すれば、インタープリタ３２０はフレームｂのプログラムカウンタ情報９３０に格納されたバイトコードの位置情報を用いてプログラムカウンタを移動させ、該当バイトコード９２０からメソッドａを実行する。この時フレームｂは削除され得る。

【００７２】

本発明の一実施形態によれば、図６～図９を参照して説明したジャバ仮想マシン１１０の動作過程を実行するためのコンピュータによって読取可能なプログラムをフラッシュメモリ、ＣＤ、ハードディスク等所定の記録媒体に書き込み、記録媒体に書き込まれたプログラムをコンピュータが実行することによって本発明を具現することも可能である。

【００７３】

以上、添付した図面を参照して本発明の実施形態を説明したが、本発明が属する技術分野で通常の知識を有する者であれば、本発明がその技術的思想や必須の特徴を変更せず、他の具体的な形態によって実施することができるということを理解できる。したがって、以上で記述した実施形態はすべての面で例示的なものであり、限定的なものではないことを理解しなければならない。

【図面の簡単な説明】

【００７４】

【図１】本発明の一実施形態に係るジャバ仮想マシンが搭載されたデジタル演算装置を示す構成図である。

【図２】本発明の一実施形態に係るクラスファイルを示す図である。

【図３】本発明の一実施形態に係るジャバ仮想マシンを示すブロック図である。

【図４】本発明の一実施形態に係るメソッドブロックを示す図である。

【図５】本発明の一実施形態に係るフレームの構造を示す図である。

【図６】本発明の一実施形態に係るメソッドブロックの提供過程を示すフローチャートである。

【図７Ａ】本発明の一実施形態に係るメソッドの呼出構造を示す図である。

【図７Ｂ】本発明の一実施形態に係るメソッドの呼出構造を示す図である。

【図８】本発明の一実施形態に係るメソッドの呼出過程を示すフローチャートである。

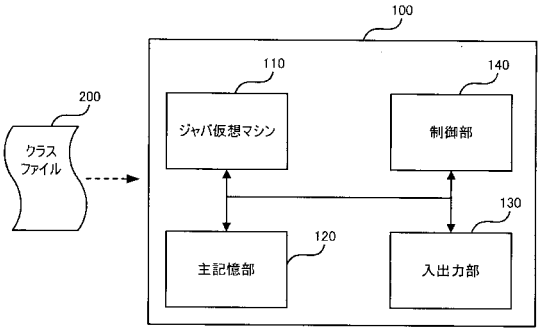
【図９】本発明の一実施形態に係るメソッドの呼出状態とそれに伴うフレームの管理状態を示す図である。

【符号の説明】

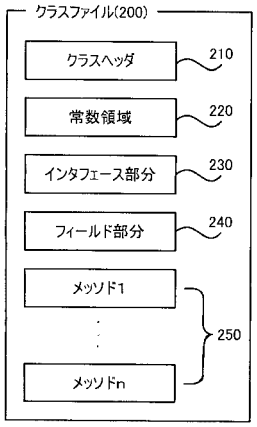
【００７５】

３１０　クラスローダ
３２０　インタープリタ
３３０　ガーベッジコレクタ

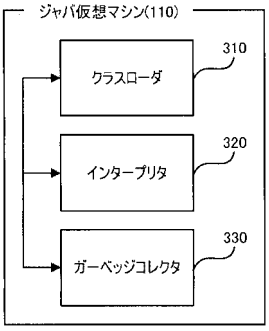
【図１】



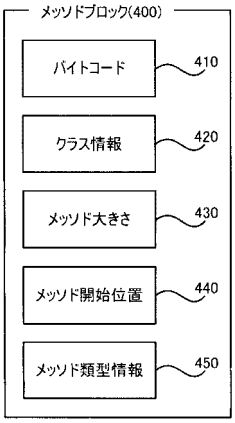
【図２】



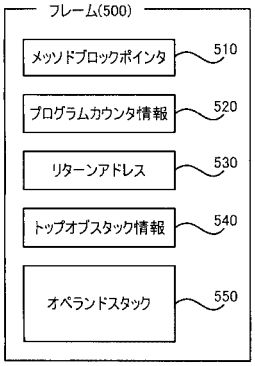
【図３】



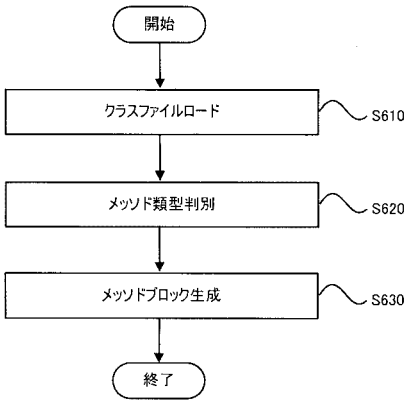
【図４】



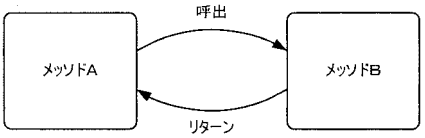
【図５】



【図６】

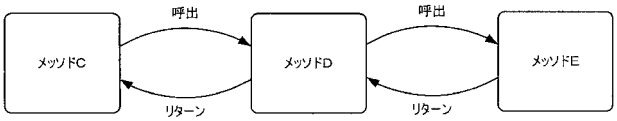


【図７Ａ】



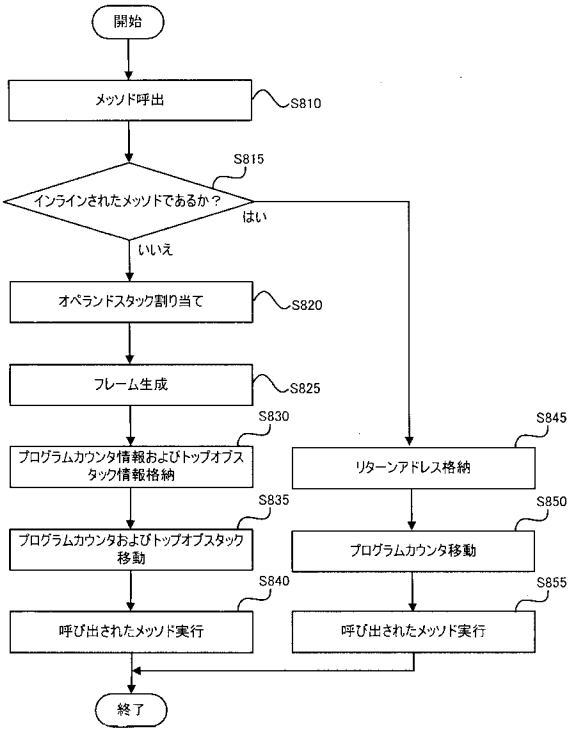
メッソド呼出経路：メッソドA－メッソドB

【図７Ｂ】

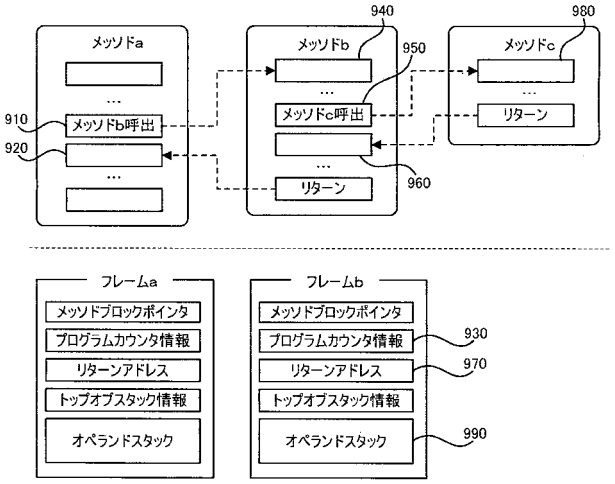


第１メッソド呼出経路：メッソドC－メッソドD－メッソドE
第２メッソド呼出経路：メッソドD－メッソドE

【図８】


開始
メッソド呼出
S810
S815
インラインされたメッソドであるか？
はい
いいえ
オペランドスタック割り当て
S820
フレーム生成
S825
S830
プログラムカウンタ情報およびトップオブスタック情報格納
S835
プログラムカウンタおよびトップオブスタック移動
S840
呼び出されたメッソド実行
S845
リターンアドレス格納
S850
プログラムカウンタ移動
S855
呼び出されたメッソド実行
終了

【図９】


メッソドa
メッソドb
メッソドc
940
950
980
910
メッソドb呼出
920
メッソドc呼出
リターン
960
リターン
フレームa
メッソドブロックポインタ
プログラムカウンタ情報
リターンアドレス
トップオブスタック情報
オペランドスタック
フレームb
メッソドブロックポインタ
プログラムカウンタ情報
リターンアドレス
トップオブスタック情報
オペランドスタック
930
970
990

フロントページの続き

(72)発明者　鄭　承　範
大韓民国京畿道水原市八達区仁溪洞１１２４－１　ジーエス仁溪ザイアパート１３０２号

Ｆターム(参考) 5B081 AA09 DD02