(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4647164号
(P4647164)
(45)発行日　平成23年3月9日(2011.3.9)　(24)登録日　平成22年12月17日(2010.12.17)

(51)Int.Cl.　F I
G06F 9/45 (2006.01)　G06F 9/44 322E
G06F 9/44 322D
G06F 9/44 320C

請求項の数 12　(全 16 頁)

(21)出願番号　特願2001-505281(P2001-505281)
(86)(22)出願日　平成12年6月3日(2000.6.3)
(65)公表番号　特表2003-502774(P2003-502774A)
(43)公表日　平成15年1月21日(2003.1.21)
(86)国際出願番号　PCT/US2000/015247
(87)国際公開番号　WO2000/079381
(87)国際公開日　平成12年12月28日(2000.12.28)
審査請求日　平成19年5月15日(2007.5.15)
(31)優先権主張番号　09/337,814
(32)優先日　平成11年6月22日(1999.6.22)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ジェームズ　エス．ミラー
アメリカ合衆国　１５９５６　ワシントン州　ベルビュー　ノースイースト　４プレイス　１７２１３

最終頁に続く

(54)【発明の名称】中間言語内の不定サイズ変数

(57)【特許請求の範囲】
【請求項１】
記憶装置に記憶されたコンピュータ実行可能命令をプロセッサが実行することによって実施される方法であって、
実行エンジンが、変数のサイズによって分類され、最小サイズ変数、最大サイズ変数およびサイズ不定変数の順でソートされている変数のリストを含む中間言語コードを受信するステップと、
前記実行エンジンが、前記変数のリストの開始位置、終了位置および変数のサイズの境界位置を示す情報を受信するステップと、
前記実行エンジンが、前記中間言語コードに基づいてネイティブコードを生成するステップであって、
前記開始位置、終了位置および変数のサイズの境界位置を示す情報に基づいて、前記変数がサイズ不定変数を含むか否かを決定するステップと、
前記ネイティブコードが実行されるシステムのアーキテクチャに基づき、変換すべきサイズ不定変数のサイズを規定する基準に従って、前記サイズ不定変数に対応するサイズ確定変数を生成するステップと
を含む、ネイティブコードを生成するステップと、
前記実行エンジンが、前記ネイティブコードを出力するステップと
を備えたことを特徴とする方法。
【請求項２】

最初に、コンパイラが、ソースコードを前記中間言語コードにコンパイルするステップをさらに備えたことを特徴とする請求項１に記載の方法。
【請求項３】
サイズ不定変数は、サイズ不定整数、サイズ不定符号なし整数またはサイズ不定実数のいずれかであることを特徴とする請求項１に記載の方法。
【請求項４】
中間言語コードを受信するステップは、前記中間言語コードをコンピュータ可読記録媒体から受信することを特徴とする請求項１に記載の方法。
【請求項５】
記憶装置に記憶されたコンピュータ実行可能命令をプロセッサが実行することによって実施される方法であって、
実行エンジンが、変数のサイズによって分類され、最小サイズ変数、最大サイズ変数およびサイズ不定変数の順でソートされている、中間言語コードの変数のリストを受信するステップと、
前記実行エンジンが、前記変数のリストの開始位置、終了位置および変数のサイズの境界位置を示す情報を受信するステップと、
前記実行エンジンが、前記開始位置、終了位置および変数のサイズの境界位置を示す情報に基づいて、前記変数がサイズ不定変数を含むか否かを決定するステップと、
前記変数がサイズ不定変数を含むと決定すると、前記実行エンジンが、ネイティブコードが実行されるシステムのアーキテクチャに基づき、変換すべきサイズ不定変数のサイズを規定するマシン固有の基準に従って、前記サイズ不定変数に対応するネイティブコードにおけるサイズ確定変数を生成するステップと
を備えたことを特徴とする方法。
【請求項６】
前記ネイティブコードを出力するステップをさらに備えたことを特徴とする請求項５に記載の方法。
【請求項７】
最初に、コンパイラが、前記変数を含むソースコードを前記中間言語コードにコンパイルするステップをさらに備えたことを特徴とする請求項５に記載の方法。
【請求項８】
実行エンジンが、変数のサイズによって分類され、最小サイズ変数、最大サイズ変数およびサイズ不定変数の順でソートされている変数のリストを含む中間言語コードを受信するステップと、
前記実行エンジンが、前記変数のリストの開始位置、終了位置および変数のサイズの境界位置を示す情報を受信するステップと、
前記実行エンジンが、前記中間言語コードに基づいてネイティブコードを生成するステップであって、
前記開始位置、終了位置および変数のサイズの境界位置を示す情報に基づいて、前記変数がサイズ不定変数を含むか否かを決定するステップと、
前記ネイティブコードが実行されるシステムのアーキテクチャに基づき、変換すべきサイズ不定変数のサイズを規定する基準に従って、前記サイズ不定変数に対応するサイズ確定変数を生成するステップと
を含む、ネイティブコードを生成するステップと、
前記実行エンジンが、前記ネイティブコードを出力するステップと
を含む方法をプロセッサに実行させるためのコンピュータ実行可能命令を記録したコンピュータ可読記録媒体。
【請求項９】
サイズ不定変数は、サイズ不定整数、サイズ不定符号なし整数またはサイズ不定実数のいずれかであることを特徴とする請求項８に記載のコンピュータ可読記録媒体。
【請求項１０】

変数のサイズによって分類され、最小サイズ変数、最大サイズ変数およびサイズ不定変数の順でソートされている変数のリストを含む中間言語コードを受信し、前記中間言語コードに基づいてネイティブコードを生成し、前記ネイティブコードを出力する実行エンジンであって、
前記中間言語コードの変数のリストの開始位置、終了位置および変数のサイズの境界位置を示す情報を受信し、
前記開始位置、終了位置および変数のサイズの境界位置を示す情報に基づいて、前記変数がサイズ不定変数を含むか否かを決定し、
前記ネイティブコードが実行されるシステムのアーキテクチャに基づき、変換すべきサイズ不定変数のサイズを規定する基準に従って、前記サイズ不定変数に対応するサイズ確定変数を生成することを特徴とする実行エンジンを備えたことを特徴とするデバイス。
【請求項１１】
実行エンジンが、変数のサイズによって分類され、最小サイズ変数、最大サイズ変数およびサイズ不定変数の順でソートされている、中間言語コードの変数のリストを受信するステップと、
前記実行エンジンが、前記中間言語コードの変数のリストの開始位置、終了位置および変数のサイズの境界位置を示す情報を受信するステップと、
前記実行エンジンが、前記開始位置、終了位置および変数のサイズの境界位置を示す情報に基づいて、前記変数がサイズ不定変数を含むか否かを決定するステップと、
前記変数がサイズ不定変数を含むと決定すると、前記実行エンジンが、ネイティブコードコードが実行されるシステムのアーキテクチャに基づき、変換すべきサイズ不定変数のサイズを規定するマシン固有の基準に従って、前記サイズ不定変数に対応するネイティブコードにおけるサイズ確定変数を生成するステップと
を備えた方法をプロセッサに実行させるためのコンピュータ実行可能命令を記録したコンピュータ可読記録媒体。
【請求項１２】
変数のサイズによって分類され、最小サイズ変数、最大サイズ変数およびサイズ不定変数の順でソートされている変数のリストを受信し、前記変数がサイズ不定変数を含むか否かを決定し、前記変数がサイズ不定変数を含むと決定すると、ネイティブコードが実行されるシステムのアーキテクチャに基づき、変換すべきサイズ不定変数のサイズを規定するマシン固有の基準に従って、前記サイズ不定変数に対応するネイティブコードにおけるサイズ確定変数を生成する実行エンジンであって、
前記中間言語コードの変数のリストの開始位置、終了位置および変数のサイズの境界位置を示す情報を受信し、
前記開始位置、終了位置および変数のサイズの境界位置を示す情報に基づいて、前記変数がサイズ不定変数を含むか否かを決定することを特徴とする実行エンジンを備えたことを特徴とするデバイス。
【発明の詳細な説明】
【０００１】
（発明の分野）
本発明は、一般に中間言語に関し、より詳細には、このような中間言語内の不定サイズ変数に関する。
【０００２】
（発明の背景）
プログラミング言語のための中間言語型モデルがますます普及してきている。中間言語モデルにおいて、ソースコードは一般に、望ましくは実質的にプラットフォーム独立の中間言語にコンパイルされる。コードが特定のプラットフォーム上で動作することを望まれているとき、このプラットフォーム上の実行エンジンは、中間言語を、プラットフォームによって理解可能なネイティブコードにインタプリトあるいはコンパイルする。中間言語を使用するシステムの例には、Java（登録商標）仮想マシンが含まれる。

【０００３】
したがって、中間言語コードの利点は、そのプラットフォームポータビリティである。ソースコードが中間言語コードにコンパイルされると、次に、この後者のコードは、異なるプラットフォームに分散されることが望ましく、これらは、ｘ８６タイプ、PowerPC（登録商標）およびDEC Alphaプロセッサなど、異なる基礎的なプロセッサを有する。そのコードは、正しくネイティブコードにコンパイルまたはインタプリトされ、実行される。すなわち、プログラムのソースコードは、プログラムが動作するように意図されたあらゆるタイプのプラットフォーム毎に、リコンパイルされる必要がないことが望ましい。
【０００４】
しかし、現在の中間言語モデルは、異なるプラットフォーム間の最適なスケーラビリティを提供しない。これに関連したスケーラビリティは一般に、プログラムがローエンドおよびハイエンドプラットフォーム上で十分に動作する能力を指す。ローエンドプラットフォームは、３２ビットアーキテクチャを有することができ、これは、プロセッサが最大３２ビット幅のデータ（つまり、３２ビットのデータ長の「ワード」）を一度に処理できることを意味する。逆に、ハイエンドプラットフォームは、６４ビットアーキテクチャを有することができ、これは、プロセッサが最大６４ビット幅のデータを一度に処理できることを意味する。
【０００５】
たとえば、従来技術では、ソースコードを中間言語コードにコンパイルすることができ、中間言語コードが、符号なし（すなわち、ポインタ）のデータタイプを有する変数など３２ビットの変数を、自動的に指定する。３２ビットアーキテクチャを有するローエンドプラットフォームでは、これはアーキテクチャの全（３２ビット）処理能力が利用されるので問題ではない。しかし、６４ビットアーキテクチャを有するハイエンドプラットフォームでは、３２ビット変数のみの指定は、アーキテクチャの全（６４ビット）処理能力が十分に利用されないことを意味する。
【０００６】
したがって、６４ビットアーキテクチャの全能力は、ソースコードを中間言語コードにコンパイルするとき、あるいはソースコード自体が書かれるとき、変数のサイズ（３２ビット、６４ビットなど）が自動的に固定される中間言語コードにより使用できない可能性がある。全能力を使用するには、ソースコードを中間言語コードにリコンパイルしなければならない可能性があり、あるいはさらに悪いことに、ソースコードを書き直して６４ビット変数を３２ビット変数の代りに指定しなければならない可能性がある。しかし、所望のターゲットプラットフォームに関わらず、６４ビット変数を演繹的に指定することは、実行できる解決策ではない。このことは、中間言語コードが３２ビットプラットフォーム上で動作しない結果となり、たとえば、これらが６４ビット変数を処理できないからである。
【０００７】
これは単に、データタイプおよびデータタイプサイズの一例であることに留意されたい。他のデータタイプには、整数、浮動小数点（すなわち、実数）などが含まれる。データタイプは他のサイズである可能性もあり、これには３２ビットおよび６４ビットのほかに、特に古いプロセッサを扱っているときは８ビットおよび１６ビットが含まれ、ならびにより最新のプロセッサを扱っているときは１２８ビットおよび２５６ビットなどが含まれる。一般に、本明細書で使用されるデータタイプという用語は、あらゆるタイプのデータタイプを指し、データタイプサイズは、あらゆるサイズのｎビットを指し、ｎは制限されない。
【０００８】
データタイプおよびデータタイプサイズの問題は、暗示的に、様々な中間言語によって不利な方法により対処される。他の２つのタイプの中間言語は、Ｏコードを含み、これはコンパイラにより、ＢＣＰＬ（「B Computer Programming Language」という、一般に使用されるＣプログラミング言語以前のもの）で書かれたソースコードから生成される。さら

に、ｐコードが含まれ、これはコンパイラにより、Pascalで書かれたソースコードから生成される。Ｏコードは、データタイプのサポートを有していない。つまり、Ｏコードは整数、浮動小数点、符号なし整数（すなわち、ポインタ）などを区別しない。さらに、すべての変数は、それらのデータタイプに関わらず同じサイズである。ｐコードは、異なるデータタイプを可能にするが、異なるサイズのデータタイプを提供しない。

【０００９】

したがって、変数のデータタイプのサイズを、中間言語コードが動作するプラットフォームに従って決定できないことは、中間言語コードのスケーラビリティを著しく損なう。結果として、中間言語コードの有用性が減る。これは、中間言語コードが一般に最初の場所において、直接ソースコードからコンパイルされるネイティブコードと比較して、より高いポータビリティのために、使用されるからである。これらおよび他の理由により、本発明の必要性がある。

【００１０】

（発明の概要）

本発明は、中間言語内の不定サイズ変数に関する。一実施形態では、コンピュータ実行方法は最初に、サイズ不定変数を有する中間言語コードを入力する。この方法は、ネイティブコードを中間言語コードに基づいて生成し、マシン固有の基準に従って、サイズ不定変数に対応するサイズ確定変数を生成することを含む。次に、この方法は、ネイティブコードを出力し、たとえば一実施形態では、この方法は、ネイティブコードを実行する。

【００１１】

たとえば、一実施形態では、すでに中間言語コードにおけるプログラムは、ポインタ（すなわち、符号なしデータタイプを有するもの）であり、かつサイズ不定である変数を有することができる。ポインタは変数のタイプであり、変数がある範囲のメモリセル内でメモリセルを参照する。ポインタによって参照することができるメモリセルの範囲は、ポインタのサイズによって制限される。３２ビットポインタは、2^{32}ものセルを参照することができるが、６４ビットポインタは2^{64}ものセルを参照することができる。３２ビット、６４ビットなどのポインタは、正確にはサイズ確定ポインタを指し、これらは確定サイズを有する。したがって、サイズ不定ポインタは、確定サイズを有していないポインタである。つまり、変数は、中間言語コード自体において、３２ビットのサイズ、６４ビットのサイズなどとして指定されない。

【００１２】

中間言語コードがネイティブコードに変換されたとき、ネイティブコードが動作する基礎的なプラットフォームが、たとえば、このポインタのサイズを指図する。３２ビットアーキテクチャの場合、サイズ不定ポインタを３２ビットポインタに変換することができ、６４ビットアーキテクチャの場合、サイズ不定ポインタを６４ビットポインタに変換することができる。これは、単一の中間言語コードが、３２ビットおよび６４ビットアーキテクチャ上で十分に動作できることを意味する。

【００１３】

したがって、本発明の実施形態は、従来技術に勝る利点に備える。本発明の実施形態によるプログラムの中間言語コードは、ローエンドおよびハイエンドプラットフォームの双方でスケーリングすることができ、プログラムのソースコードを中間言語コードにリコンパイルしたり、あるいは書き直す必要がない。つまり、プログラムが動作するプラットフォームのマシン固有の基準に応じて、サイズ不定変数は、プログラムが動作するプラットフォームに適切なサイズ確定変数に変換される。

【００１４】

本発明は、コンピュータ実行方法、マシン可読媒体、コンピュータ化されたシステム、デバイスおよび様々な範囲のコンピュータを含む。本明細書に記載されたもの以外に、本発明の他の態様、実施形態および利点は、詳細な説明を読むことによって、かつ図面を参照して明らかになるであろう。

【００１５】

（発明の詳細な説明）
以下の本発明の例示的実施形態の詳細な説明では、その一部を形成する添付の図面への参照を行い、これにおいては、本発明を実施することができる特定の例示的実施形態を示す。これら実施形態を十分詳細に記載して、当業者が本発明を実施できるようにし、本発明の精神または範囲から逸れることなく、他の実施形態を利用できること、および論理的、機械的、電気的および他の変更を行うことができることを理解されたい。したがって、以下の詳細な説明は、限定の意味に解釈されるものではなく、本発明の範囲は、付属の特許請求の範囲によってのみ定義される。
【００１６】
後に続く詳細な説明のいくつかの部分は、コンピュータメモリ内のデータビットにおけるオペレーションのアルゴリズムおよび記号的表現に関して提示される。これらアルゴリズムの記載および表現は、データ処理業界における当業者によって使用されて、もっとも効果的にそれらの働きの実質を他の当業者に伝えるための手段である。アルゴリズムは、本明細書において、かつ一般には、所望の結果に通じる自己矛盾のないステップのシーケンスとして表される。ステップは、物理量の物理的操作を必要とするものである。通常、必須ではないが、これらの量は、電気または磁気信号の形式を取り、格納され、転送され、結合され、比較されかつそうでなければ操作することができる。
【００１７】
これらの信号をビット、値、要素、シンボル、キャラクタ、用語、数などと呼ぶことは、時として好都合であると証明されており、これは主として一般的な使用の理由のためである。しかし、これらの全ておよび類似の用語が、適切な物理量に関連付けられるものであり、これらの量に適用された好都合なラベルでしかないことに留意されたい。具体的に述べられていない限り、そうでない場合は以下の考察から明らかであるように、本発明全体で、処理またはコンピューティングまたは計算または決定または表示などの用語を使用する考察は、コンピュータシステムまたは類似の電子コンピューティングデバイスの動作および過程を指し、これらがコンピュータシステムのレジスタおよびメモリ内で物理（電子）量として表現されたデータを操作し、コンピュータシステムのメモリまたはレジスタまたは他のそのような情報記憶装置、伝送または表示デバイス内で物理量として類似の方法で表現された他のデータに変換することを理解されたい。
【００１８】
（オペレーティング環境）
図１を参照して、本発明の実施形態を協働して実施することができるハードウェアおよびオペレーティング環境を示す。図１の記載は、本発明を協働して実施することができる、適切なコンピュータハードウェアおよび適切なコンピューティング環境の簡単で一般的な記載を提供するように意図される。必要ではないが、本発明を一般に、パーソナルコンピュータなどのコンピュータによって実行される、プログラムモジュールなどのコンピュータ実行可能命令に関連して記載する。一般に、プログラムモジュールには、ルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれ、これらが特定のタスクを実行し、あるいは特定の抽象データタイプを実施する。
【００１９】
さらに、本発明を他のコンピュータシステム構成により実施することができ、これらにはハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベースまたはプログラム可能な家電製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータなどが含まれることは、当業者には理解されよう。本発明はまた、分散コンピューティング環境において実施することもでき、タスクがリモートの処理デバイスによって実行され、これらが通信ネットワークを介してリンクされる。分散コンピューティング環境では、プログラムモジュールをローカルおよびリモートの記憶デバイスに位置付けることができる。
【００２０】
本発明を実施するための図１の例示的ハードウェアおよびオペレーティング環境は、コン

ピュータ２０の形式における汎用コンピューティングデバイスを含み、これは、処理装置２１、システムメモリ２２、および、システムメモリを含む様々なシステムコンポーネントを処理装置２１に動作可能に結合するシステムバス２３を含む。１つのみまたは複数の処理装置２１がある可能性があり、コンピュータ２０のプロセッサは、単一の中央処理装置（ＣＰＵ）、または複数の処理装置を備え、これは一般に並列処理環境と呼ばれる。コンピュータ２０は、従来のコンピュータ、分散コンピュータまたは他のいかなるタイプのコンピュータにすることもでき、本発明はそのように限定されない。

【００２１】

システムバス２３は、いくつかのタイプのバス構造のいずれにすることもでき、これには、メモリバスまたはメモリコントローラ、周辺バス、および様々なバスアーキテクチャのいずれかを使用するローカルバスが含まれる。システムメモリは、単にメモリと呼ばれることもあり、読取り専用メモリ（ＲＯＭ）２４およびランダムアクセスメモリ（ＲＡＭ）２５を含む。基本入出力システム（ＢＩＯＳ）２６は、起動中など、情報をコンピュータ２０内の要素の間で転送するための助けとなる基本ルーチンを含み、ＲＯＭ２４に格納される。コンピュータ２０はさらに、図示しないハードディスクから読み取りかつこれに書き込むためのハードディスクドライブ２７、取外し可能磁気ディスク２９から読み取りあるいはこれに書き込むための磁気ディスクドライブ２８、および、ＣＤ　ＲＯＭまたは他の光媒体などの取外し可能光ディスク３１から読み取りあるいはこれに書き込むための光ディスクドライブ３０をさらに含む。

【００２２】

ハードディスクドライブ２７、磁気ディスクドライブ２８、および光ディスクドライブ３０は、システムバス２３に、それぞれハードディスクドライブインタフェース３２、磁気ディスクドライブインタフェース３３、および光ディスクドライブインタフェース３４によって接続される。ドライブおよびそれらの関連付けられたコンピュータ可読媒体が、コンピュータ２０用のコンピュータ可読命令、データ構造、プログラムモジュールおよび他のデータの不揮発性記憶装置を提供する。磁気カセット、フラッシュメモリカード、デジタルビデオディスク、ベルヌーイカートリッジ、ランダムアクセスメモリ（ＲＡＭ）、読取り専用メモリ（ＲＯＭ）など、コンピュータにより可読であるデータを格納することができるいかなるタイプのコンピュータ可読媒体も例示的オペレーティング環境において使用できることを、当業者には理解されたい。

【００２３】

いくつかのプログラムモジュールをハードディスク、磁気ディスク２９、光ディスク３１、ＲＯＭ２４またはＲＡＭ２５において格納することができ、これには、オペレーティングシステム３５、１または複数のアプリケーションプログラム３６、他のプログラムモジュール３７およびプログラムデータ３８が含まれる。ユーザは、コマンドおよび情報をパーソナルコンピュータ２０へ、キーボード４０およびポインティングデバイス４２などの入力デバイスを介して入力することができる。他の入力デバイス（図示せず）には、マイクロフォン、ジョイスティック、ゲームパッド、衛星放送受信アンテナ、スキャナなどが含むことができる。これらおよび他の入力デバイスは、システムバスに結合されるシリアルポートインタフェース４６を介して、処理装置２１に接続されるが、これを、パラレルポート、ゲームポートまたはユニバーサルシリアルバス（ＵＳＢ）など、他のインタフェースによって接続することができる。モニタ４７または他のタイプの表示デバイスは、ビデオアダプタ４８などのインタフェースを介して、システムバス２３に接続される。モニタに加えて、コンピュータは、典型的には、スピーカおよびプリンタなど、他の周辺出力デバイス（図示せず）を含む。

【００２４】

コンピュータ２０は、ネットワーク環境において、リモートコンピュータ４９など、１つまたは複数のリモートコンピュータへの論理接続を使用して動作することができる。これらの論理接続は、コンピュータ２０に結合された通信デバイスまたはコンピュータ２０の一部によって達成され、本発明は特定のタイプの通信デバイスに限定されない。リモート

コンピュータ４９は、別のコンピュータ、サーバ、ルータ、ネットワークＰＣ、クライアント、ピアデバイスまたは他の共通ネットワークノードにすることができ、典型的には、コンピュータ２０に関して上述した要素の多くまたはすべてを含むが、記憶デバイス５０のみを図１に例示した。図１に示す論理接続は、ローカルエリアネットワーク（ＬＡＮ）５１およびワイドエリアネットワーク（ＷＡＮ）５２を含む。このようなネットワーキング環境は、オフィスネットワーク、企業全体のコンピュータネットワーク、イントラネットおよびインターネットにおいて一般的であり、これらはすべてのタイプのネットワークである。

【００２５】

ＬＡＮネットワーキング環境において使用するとき、コンピュータ２０は、ネットワークインタフェースまたはアダプタ５３を介して、ローカルネットワーク５１に接続され、これは通信デバイスの１つのタイプである。ＷＡＮネットワーキング環境において使用するとき、コンピュータ２０は、典型的には、モデム５４、通信デバイスの１つのタイプ、または他のいかなるタイプの通信デバイスを含み、インターネットなどのワイドエリアネットワーク５２を介して通信を確立する。モデム５４は、内部あるいは外部にすることができ、シリアルポートインタフェース４６を介して、システムバス２３に接続される。ネットワーク環境では、パーソナルコンピュータ２０に関して示したプログラムモジュールまたはその一部を、リモート記憶デバイスに格納することができる。図示のネットワーク接続は例示的であり、コンピュータ間の通信リンクを確立するための他の手段および通信デバイスを使用できることを理解されたい。

【００２６】

（中間言語環境）

このセクションでは、本発明の一実施形態による中間言語環境の概略を提供する。図２を参照して、本発明の一実施形態による中間言語環境を示す。図２の環境は、実行エンジン２００を含む。Visual Basic（ＶＢ）、Visual C++（ＶＣ＋＋）および他のソースなど、多数のソースコード言語のソースが、コンパイラ２０４などのコンパイラによって、中間言語（ＩＬ）コードにコンパイルされる。次に、実行エンジン２００は、コンパイラ、インタプリタまたはジャストインタイムコンパイラなどのメカニズムを介して、ＩＬコードを、特定のプラットフォームに固有の実行コードに（中間コードに基づいて）コンパイル、インタプリトあるいはジャストインタイムコンパイルすると、この実行可能コードを実行する。実行可能コードは、ネイティブコードとも呼ばれる。つまり、実行エンジン２００は、中間コードを実行用の実行可能コードに変換し、ネイティブコードを実行する。さらに、一実施形態では、実行エンジン２００は、ＩＬサイズ不定変数に対応するネイティブのサイズ確定変数を生成するためのメカニズムを含み、詳細な説明は、さらなるセクションにおいて記載する。

【００２７】

一実施形態では、コンパイラ２０４、実行エンジン２００および実行エンジンの構成部分のそれぞれを、プロセッサによってメモリなどのコンピュータ可読媒体から実行されるコンピュータプログラムにすることができる。コンパイラ２０４などのコンパイラは、当技術分野内で知られている。さらに、Ｃ、Visual Basicなどのソースコード言語をネイティブコードにコンパイルする使用可能なコンパイラは、その代りに中間言語コードを生成するように修正可能である。一実施形態では、多数のソースコード言語のソース、ＩＬコードおよび実行可能コードのそれぞれをデータとして、メモリまたはハードディスクドライブなどのコンピュータ可読媒体上に格納することができる。しかし、本発明はそのように限定されない。

【００２８】

実行エンジン２００は、それが動作する特定の基礎的なマシンプラットフォームに基づくこと、すなわち、それが動作する基礎的なマシンプラットフォームについての情報を含むことが望ましいことに留意されたい。これにより、ＩＬコードサイズ不定変数に対応するネイティブコードサイズ確定変数を生成し、その決定を基礎的なマシンプラットフォーム

に基づくようにするメカニズムとすることができる。
【００２９】
一実施形態では、実行エンジン２００は、コンパイラ２０４を含んでいないシステムの一部であり、コンパイラ２０４がソースコードをＩＬコードにプリコンパイルし、これは、たとえば、システム内のコンピュータ可読媒体上に格納される。他の例として、ＩＬコードがシステムによって受信され、ネイティブコードに変換され、動作する。もう１つの実施形態では、実行エンジンおよびその構成部分が、詳細な説明の先行するセクションにおいて記載されたコンピュータなど、デバイスの一部である。本発明に従う他のデバイスには、テレビ用のセットトップボックス、ハンドヘルドデバイス、テレビセット、家電デバイス、ラップトップコンピュータ、ハンドヘルドコンピュータ、アプライアンス、デスクトップコンピュータおよびカーエレクトロニクスデバイスが含まれる。本発明はそのように限定されない。このようなデバイスは、典型的には、プロセッサ、およびメモリなどのマシン可読媒体を含み、実行エンジン２００およびその構成部分がプロセッサによって媒体から実行されるようにする。
【００３０】
（概観）
詳細な説明のこのセクションでは、本発明の少なくともいくつかの実施形態が動作する方法の概略を記載する。この記載は、詳細な説明の以下のセクションにおいて提示される、本発明の異なる実施形態の方法およびシステムを理解するための基礎を提供する。概略を、図３および図４を参照して記載する。
【００３１】
最初に図３を参照して、本発明の一実施形態のオペレーションを例示する。メカニズム３００は、図２の実行エンジン２００の一部にすることができ、サイズ不定変数などのＩＬ変数３０２を有するＩＬコードを入力する（たとえば、コンピュータ可読媒体から）。サイズ不定変数は、ＩＬコードからコンパイルされたソースコードが書かれる時点で、サイズ不定であるとして指定される。すなわち、プログラマは、変数をサイズ不定として指定するための決定を行う。メカニズム３００は、ＩＬコードに対応するネイティブコードを生成し、ＩＬサイズ不定変数に対応するサイズ確定変数などの変数３０２に対応するネイティブ変数３０４を含む。ＩＬコード自体は、図３に図示しないソースコードからコンパイルされる。さらに、生成されると、ネイティブコードを出力することができる。一実施形態では、ネイティブコードを出力することが、ネイティブコードを実行することとして定義される。この概略の記載の残りは、ＩＬコード変数３０２がサイズ不定であり、つまりこれが固定サイズではないデータタイプを有することを仮定し、ネイティブコード変数３０４が、メカニズム３００によって決定されたサイズを有する。一実施形態におけるメカニズム３００は、プロセッサによってコンピュータ可読媒体から実行されるコンピュータプログラム（またはその一部）であるが、本発明はそのように限定されない。
【００３２】
ＩＬコード変数３０２は、いかなるタイプのデータタイプも有することができ、本発明はそのように限定されない。たとえば、変数３０２を、サイズ不定整数、サイズ不定符号なし整数（ポインタにすることもできる）、およびサイズ不定実数（浮動小数点とも呼ばれる）にすることができる。メカニズム３００は、変数３０２のサイズ情報を含む情報３０６を参照することによって、変数３０２のサイズを決定する。つまり、情報３０６は、具体的に変数３０２のサイズを線引きする。
【００３３】
一実施形態では、情報３０６は、シグネチャファイルと呼ばれるものの一部である。シグネチャファイルは、ＩＬコードによって表現されるコンピュータプログラムのオブジェクトクラスのメソッドについて特定の情報を含む。オブジェクトクラスという用語は、当技術分野内で知られており、一般にオブジェクト指向プログラミングに関連して、オブジェクトのクラスの特定の定義を指し、そこから特定のオブジェクトをインスタンス化することができる。オブジェクトは、一般に不透明であり、他のオブジェクトは、そのオブジェ

クトの１または複数のメソッドを介して、そのオブジェクトにアクセスのみできるようにし、オブジェクトは、各メソッドによって定義された特定の機能性を実行するようにオブジェクトに求める。各メソッドは一般に、変数３０２など、１または複数のローカル変数を有する。したがって、シグネチャファイルは、特定のメソッドに関する情報を含む。この情報は、たとえば、変数３０２のサイズを線引きする情報３０６を含むことができる。

【００３４】

したがって、情報３０６を参照することによって、メカニズム３００は、変数３０２のサイズがすでに定義されている（３２ビット整数、６４ビット実数など）か否か、または仮定されたように、サイズが不定である（サイズ不定整数など）か否かを決定することができる。サイズが不定であった場合、メカニズム３００は、さらに基準３０８を参照して、対応するネイティブコード変数３０４がなるべきサイズを決定する。

【００３５】

一実施形態では、この基準３０８はマシン固有であり、ネイティブコードが実行されるシステムのアーキテクチャに関係する。たとえば、典型的なｘ８６タイプのプロセッサでは、基準３０８により、あらゆるサイズ不定変数を３２ビット変数に解決（resolve）すると規定することができ、DEC Alphaプロセッサでは、基準３０８により、あらゆるサイズの不定変数を６４ビット変数に解決すると規定することができる。しかし、本発明は特定の基準３０８に限定されない。

【００３６】

たとえば、特定の基準３０８により、あらゆるサイズ不定変数を可能な最大サイズの変数に解決すると規定することができる。別法として、基準３０８により、サイズ不定整数変数を、たとえばあるサイズに解決し、一方、サイズ不定浮動小数点変数を別のサイズに解決すると規定することができる。

【００３７】

この方法において、メカニズム３００は、不定サイズＩＬコード変数３０２に対応する確定サイズネイティブコード変数３０４を生成することができる。メカニズム３００は、サイズ情報３０６を使用して、ＩＬコード変数３０２が不定サイズであることを決定する。次に、メカニズム３００は、基準３０８を使用して、対応するネイティブコード変数３０４のサイズを決定する。メカニズム３００は最後に、確定サイズネイティブコード変数３０４を生成する。

【００３８】

一例は、本発明の実施形態をさらに理解することにおいて例示的なものである。ソースコードのＩＬコード表現「var x int」を考察すると、これは整数タイプの変数を指定するものである。ソースコード自体は整数のサイズを指定しない。すなわち、たとえば短整数および長整数のそれぞれ対応する「short」または「long」を指定しない。したがって、実行エンジンは、変数ｘをネイティブコードにコンパイルあるいはインタプリトするとき、この整数のサイズを解決する必要がある。実行エンジンが３２ビットプラットフォーム上で動作する場合、たとえば、サイズ不定整数を含むすべてのサイズ不定変数を、対応する３２ビット変数に解決することを規定するための、固有の基準を有することができる。したがって、変数ｘは、３２ビット整数に解決される。逆に、実行エンジンが６４ビットプラットフォーム上で動作する場合、サイズ不定符号なし変数のみを６４ビット符号なし変数に解決し、他のすべてのサイズ不定変数については３２ビット変数に解決することを規定するための、固有の基準を有することができる。この後者の実施例でも、変数ｘは、３２ビット整数に解決される。しかし、上述したように、本発明は、サイズ不定変数を解決するための特定の基準に限定されない。

【００３９】

図４を参照して、本発明の一実施形態にかかる中間言語変数のサイズが決定される方法を具体的に例示する図を示す。最小サイズ変数、最大サイズ変数、およびサイズ不定変数を含み、変数のサイズによって分類された変数４００のリストは、ソースコードからＩＬコードへのコンパイルプロセスの一部として生成され、リスト４００が、実行エンジンに

沿って渡されるＩＬコードの一部となる。言い換えれば、一実施形態では、リスト４００は、図３のメカニズム３００によって依拠された図３のサイズ情報３０６である。しかし、本発明はそのように限定されない。

【００４０】

したがって、図４に示すようなリスト４００は、１６ビット（または「short」）変数４０２、３２ビット（または「long」）変数４０４および不定サイズ変数４０６を含む。リスト４００は、開始４０８および終了４１４、ならびに１６ビット変数４０２を３２ビット変数４０４から分離する第１の境界４１０、および３２ビット変数４０４を不定サイズ変数４０６から分離する第２の境界４１２を有する。

【００４１】

一実施形態では、ＩＬ変数のサイズがこのリストに関して、以下に記載するように決定される。ＩＬコードと共に、リスト４００に関して渡された情報（ＩＬコード自体の生成中に生成された）は、リスト４００における変数の総数を含み、たとえば、配列variables[beginning=1..end=total number of variables]がリストを指定する。ＩＬコードと共に渡された情報は、リスト４００内の１６ビット変数４０２の数も含み、変数variables[beginning ..first boundary =number of 16-bit variables]が１６ビット変数を指定する。ＩＬコードと共に渡された情報は、加えて、リスト４００内の３２ビット変数４０４の数を含み、変数variables[first boundary +1.. second boundary =first boundary +number of 32-bit variables]が３２ビット変数を指定する。したがって、変数[second boundary +1.. end]がサイズ不定変数を指定する。最後に、ＩＬコードと共に渡された情報は、各変数のリスト番号を含み、variables[list number of a particular variable]が特定の変数を指定する。

【００４２】

したがって、この情報を有することにより、特定の変数のリスト番号が分かっている場合には、特定の変数のサイズを容易に決定することができ、このときＩＬコードと共に渡された情報が、各変数のリスト番号も含む。たとえば、合計１０個の変数がメソッドにあり、５個の１６ビット変数および３個の３２ビット変数が含むことができる。したがって、５以下のリスト番号を有する変数は、その変数が１６ビットであることを意味する。５より大きいが８以下であるリスト番号を有する変数は、その変数が３２ビットであることを意味する（８は、５個の１６ビット変数および３個の３２ビット変数を加算することによって決定される）。最後に、８より大きいリスト番号を有する変数は、その変数がサイズ不定であることを意味する。すなわち、この実施形態では、変数リストの変数サイズ境界に対する特定の変数の位置を参照することで、容易に特定の変数のサイズが決定する。

【００４３】

（方法）

詳細な説明のこのセクションでは、本発明の様々な実施形態によるコンピュータ実行方法を記載する。コンピュータ実行方法は、少なくとも部分的には、コンピュータ（図１のコンピュータなど）上で動作する１または複数のプログラム、つまり、メモリなどのコンピュータ可読媒体からコンピュータのプロセッサによって実行されるプログラムとして実現することができる。プログラムは、別のコンピュータ上に分散および実装および実行のために、フロッピー（登録商標）ディスクまたはＣＤ－ＲＯＭなど、マシン可読媒体上に格納可能であることが望ましい。

【００４４】

図５を参照して、本発明の一実施形態にかかるコンピュータ実行方法を示す。５００で、ＩＬ変数が、たとえばＩＬコードを入力することの一部として入力される。ＩＬコードが入力される方法は、本発明によって限定されない。たとえば、ＩＬコードを、インターネットを介して受信することができ、あるいは、ハードディスクドライブまたは不揮発性メモリなど、コンピュータ可読媒体から読み取ることができる。

【００４５】

５０２で、５００で受信されたＩＬ変数がサイズ不定であるかどうかが決定される。一実

施形態では、この決定が、詳細な説明の以前のセクションで記載されたように行われ、たとえば、ＩＬコードと共に渡された情報、記載されたような変数のリストに関する情報などを参照することによる。ＩＬ変数が不定サイズを有した場合、この方法は５０４へ進行し、そうでない場合、この方法は５０６へ進行する。

【００４６】

５０４で、不定サイズ変数が確定サイズ変数に解決される。つまり、ＩＬ変数に対して対応するネイティブ変数のサイズが決定される。一実施形態では、これはマシン固有の基準に従って、詳細な説明の以前のセクションで記載されたように実施される。たとえば、プラットフォームがｘ８６タイプのプラットフォームであった場合、サイズ不定変数は、３２ビットサイズの変数に解決されることができる。しかし、本発明はそのように限定されない。

【００４７】

最後に、５０６で、ＩＬ変数に対して対応するネイティブ変数が、たとえばＩＬコードに対応するネイティブコードを生成することの一部として生成される。５０６へ５０２から進行した場合、ネイティブ変数のサイズが、５００で受信されたＩＬ変数のサイズによって指図される。５０６へ５０４から進行した場合、ネイティブ変数のサイズが、５０４で行われた決定によって指図される。

【００４８】

当業者には理解できるように、本発明の一実施形態による方法は、図５の特定の方法において図示しない他の部分を含むことができる。たとえば、ソースコードからのＩＬコードのコンパイルは、ＩＬ変数が確定サイズを有するか否かを決定するために使用される情報の生成を含み、これを５００の前に実行することができる。もう１つの実施例として、５０６の後に、生成されたネイティブコードを出力することができ、一実施形態では、ネイティブコードを出力することが、コードが動作するプラットフォーム上で、そのコードを実行することを含む。しかし、本発明はそのように限定されない。

【００４９】

これも当業者には理解できるように、図５の方法の１または複数の部分を、図２に関連して記載された中間言語環境のコンポーネントによって実行することができる。たとえば、システム内で、ジャストインタイムコンパイラなどのメカニズムが、中間言語コードをネイティブコードに変換することができ、これは、不定サイズＩＬ変数に対応する確定サイズネイティブ変数を生成することを含む。もう１つの実施例として、デバイス内で、あるメカニズムをプロセッサによってコンピュータ可読媒体から実行して、不定サイズＩＬ変数に対応する確定サイズネイティブ変数を生成することができる。しかし、本発明はそのように限定されない。

【００５０】

（ハードウェアによる実行）

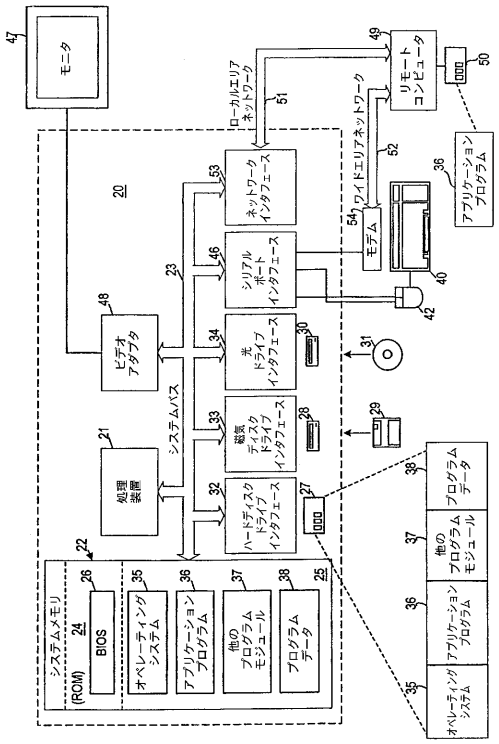
本願においてここまで記載したように、本発明の実施形態を、主としてソフトウェア環境に適用可能として記載した。しかし、当業者には理解できるように、本発明はそのように限定されない。たとえば、別の実施形態において、サイズ不定変数を有する中間言語（ＩＬ）コードを、直接ハードウェアのみのメカニズムに入力することができ、これは１または複数の集積回路（ＩＣ）または「チップ」であり、ＩＬコードに基づいてネイティブコードを生成し、このネイティブコードを実行する。このような実施例では、ハードウェアのみのメカニズムが組み込みの条件を有し、たとえば、それ自体のアーキテクチャに基づいて、サイズ不定変数をサイズ確定変数に解決する。すなわち、ハードウェアのみのメカニズムは、実際には、ハードウェアに実装された専用実行エンジンである。このようなメカニズムの設計者は、サイズ不定変数がサイズ確定変数に変換される方法についての基準を指定する。たとえば、３２ビットハードウェアメカニズムに関連して、すべてのサイズ不定変数を３２ビット変数に変換することができる。別の実施例として、サイズ不定整数を３２ビット整数に変換することができ、サイズ不定ポインタを６４または１２８ビットポインタに変換することができる。

【００５１】
さらに、ある特定のハードウェア実施では、ハードウェアのみのメカニズムは、「ネイティブコード」をそれ自体を有していない可能性があり、これは、ハードウェア自体を、中間言語コードがそのハードウェア用の「ネイティブ」コードであるように設計できるからである。つまり、ハードウェアは中間言語コードを実行用のネイティブコードに変換する必要がなく、これは中間言語コードがそれ自体でハードウェア上で実行可能だからである。この例では、それでもハードウェアがなおサイズ不定変数をサイズ確定変数に変換する必要がある。したがって、本明細書で記載したようなハードウェアのみのメカニズムは、中間言語コードを直接実行し、いかなるサイズ不定変数も最初にサイズ確定変数に変換しなければならないという但し書きを有する。

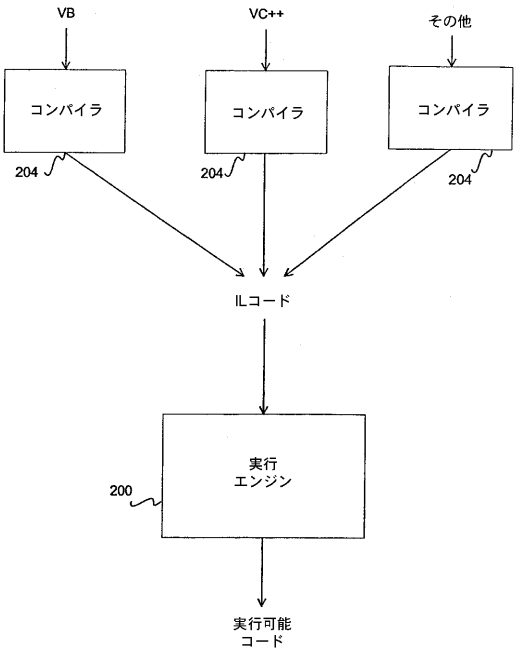
【００５２】
（結論）
特定の実施形態を本明細書で例示し、記載したが、同じ目的を達成するように計算されたいかなる構成も、示した特定の実施形態の代りに使用できることは、当業者には理解されよう。本願は、本発明のいかなる適合または変形形態をも包含するように意図される。したがって、本発明が以下の特許請求の範囲およびそれに相当するものによってのみ限定されることが明白に意図される。

【図面の簡単な説明】
【図１】　本発明の実施形態を協働して実施することができるオペレーティング環境の図である。
【図２】　その一部を本発明の一実施形態にかかるシステムまたは装置内で実施することができる、本発明の一実施形態にかかる中間言語環境の図である。
【図３】　本発明の一実施形態のオペレーションを示す図である。
【図４】　本発明の一実施形態にかかる、中間言語変数のサイズが決定される方法をより明確に示す図である。
【図５】　本発明の一実施形態にかかる流れ図である。

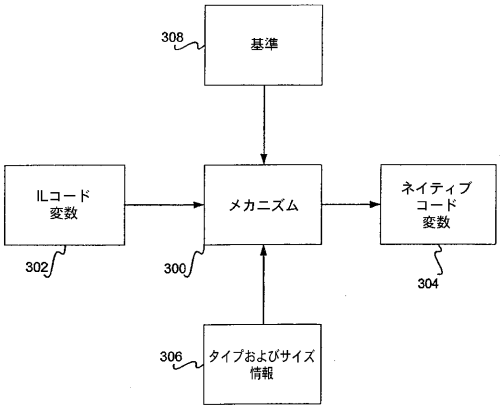
【図１】


モニタ
ローカルエリアネットワーク
リモートコンピュータ
ワイドエリアネットワーク
アプリケーションプログラム
ネットワークインタフェース
モデム
シリアルポートインタフェース
ビデオアダプタ
光ドライブインタフェース
システムバス
磁気ディスクドライブインタフェース
処理装置
ハードディスクドライブインタフェース
システムメモリ
(ROM)
BIOS
オペレーティングシステム
アプリケーションプログラム
他のプログラムモジュール
プログラムデータ

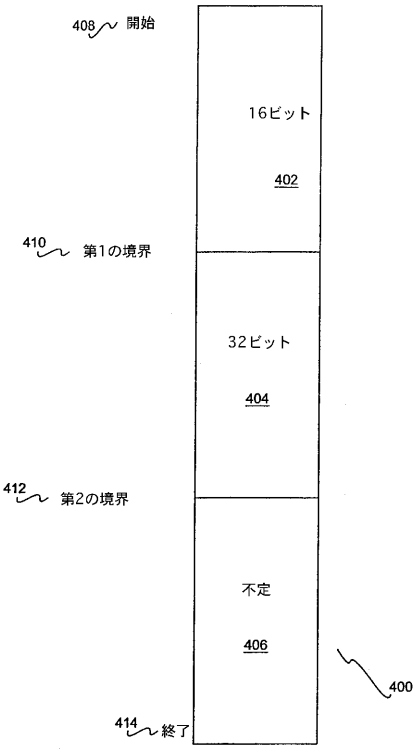
【図２】


VB
VC++
その他
コンパイラ
コンパイラ
コンパイラ
204
204
204
ILコード
実行エンジン
200
実行可能コード

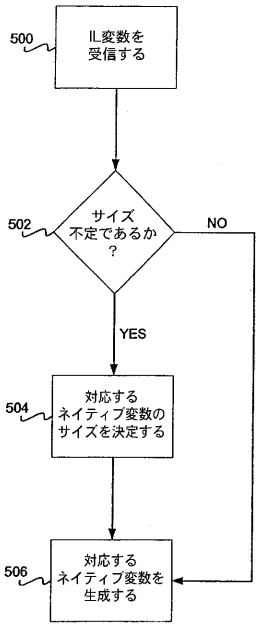
【図３】


基準
308
ILコード変数
メカニズム
ネイティブコード変数
302
300
304
タイプおよびサイズ情報
306

【図４】


408 開始
16ビット
402
410 第1の境界
32ビット
404
412 第2の境界
不定
406
400
414 終了

【図５】


500
IL変数を
受信する
502
サイズ
不定であるか
？
NO
YES
504
対応する
ネイティブ変数の
サイズを決定する
506
対応する
ネイティブ変数を
生成する

フロントページの続き

(72)発明者　ピーター　クコル
アメリカ合衆国　３８４９０　ワシントン州　モンロー　２０８プレイス　サウスイースト　１６５２１
(72)発明者　バンス　ピー．モリソン
アメリカ合衆国　９８０３３　ワシントン州　カークランド　１２０アベニュー　ノースイースト　６１１４

審査官　久保　光宏

(56)参考文献　特開平６－２３６２８２（ＪＰ，Ａ）
欧州特許出願公開第０４６３５８３（ＥＰ，Ａ２）
志村浩也（外１名），「Java JITコンパイラの試作」，情報処理学会研究報告，日本，社団法人情報処理学会，１９９６年１０月３１日，第９６巻，第１０６号（96-ARC-120），第３７～４２頁，ISSN:0919-6072

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F9/45,
CSDB(日本国特許庁),
JSTPlus(JDream2)