(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3785596号
(P3785596)
(45)発行日　平成18年6月14日(2006.6.14)　(24)登録日　平成18年3月31日(2006.3.31)

(51)Int.Cl.　F I
G06F 9/42 (2006.01)　G06F 9/42 330A

請求項の数 4 (全 14 頁)

(21)出願番号　特願2002-575788(P2002-575788)
(86)(22)出願日　平成14年3月25日(2002.3.25)
(86)国際出願番号　PCT/JP2002/002888
(87)国際公開番号　WO2002/077802
(87)国際公開日　平成14年10月3日(2002.10.3)
審査請求日　平成16年10月15日(2004.10.15)
(31)優先権主張番号　特願2001-88850(P2001-88850)
(32)優先日　平成13年3月26日(2001.3.26)
(33)優先権主張国　日本国(JP)

(73)特許権者　899000046
関西ティー・エル・オー株式会社
京都府京都市下京区中堂寺粟田町９３番地
(74)代理人　100078868
弁理士　河野　登夫
(72)発明者　山本　晃成
東京都新宿区新宿２丁目４番３号　フォーシーズンビル１０階　株式会社数理システム内
(72)発明者　湯淺　太一
京都府京都市左京区吉田本町　京都大学内

審査官　後藤　彰

最終頁に続く

(54)【発明の名称】関数実行方法、関数実行装置、コンピュータプログラム、及び記録媒体

(57)【特許請求の範囲】

【請求項１】

メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げ、積み上げた関数記録領域を利用して被呼出関数を呼び出し、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄する関数実行方法において、

ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析し、

該解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行し、

かつ解析により呼出関数及び被呼出関数が異なると判別したとき、被呼出関数の形式に基づいて、呼出関数を実行する関数記録領域を変更して利用させる

ことを特徴とする関数実行方法。

【請求項２】

メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げ、積み上げた関数記録領域を利用して被呼出関数を呼び出し、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄する関数実行装置において、

ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析する手段と、

該解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行する実行手段と

を備え、

該実行手段は、解析により呼出関数及び被呼出関数が異なると判別したとき、呼出関数を実行する関数記録領域を、被呼出関数の形式に基づいて変更する様に構成してある

ことを特徴とする関数実行装置。

【請求項３】

コンピュータに、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げさせ、積み上げた関数記録領域を利用して被呼出関数を呼び出させ、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄させるコンピュータプログラムにおいて、

コンピュータに、ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析させる手順と、

コンピュータに、解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行させる手順と、

コンピュータに、解析により呼出関数及び被呼出関数が異なると判別したとき、被呼出関数を呼び出す領域として利用させるべく、呼出関数を実行する関数記録領域を、被呼出関数の形式に基づいて変更させる手順と

を含むことを特徴とするコンピュータプログラム。

【請求項４】

コンピュータに、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げさせ、積み上げた関数記録領域を利用して被呼出関数を呼び出させ、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄させるコンピュータプログラムを記録してあるコンピュータでの読み取りが可能な記録媒体において、

コンピュータに、ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析させる手順と、

コンピュータに、解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行させる手順と、

コンピュータに、解析により呼出関数及び被呼出関数が異なると判別したとき、被呼出関数を呼び出す領域として利用させるべく、呼出関数を実行する関数記録領域を、被呼出関数の形式に基づいて変更させる手順と

を含むコンピュータプログラムを記録してあることを特徴とするコンピュータでの読み取りが可能な記録媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数から呼び出される被呼出関数の形式に基づく関数記録領域を積み上げ、積み上げた関数記録領域を利用して被呼出関数を呼び出し、呼び出された被呼出関数の実行後、当該関数記録領域を破棄する関数実行方法、その方法を適用した関数実行装置、その装置を実現するためのコンピュータプログラム、及びそのコンピュータプログラムを記録した記録媒体に関し、

特にＪＶＭ(Java Virtual Machine)にてＪａｖａ（登録商標）言語等の言語にて記述された関数を実行する関数実行方法、関数実行装置、コンピュータプログラム、及び記録媒体に関する。

【背景技術】

【０００２】

Ｊａｖａ言語等の言語にて記述された複数の命令からなる関数を複数含むプログラムを実行する関数実行方法が近年様々な用途で利用されており、またこのようなＪａｖａ言語等の言語は一般的にＪＶＭのバイトコードにコンパイルされてからＪＶＭのメモリ中の領域にて実行される。

【０００３】

図１はプログラムの実行に必要な関数を記録する関数記録領域を示す概念図である。

プログラムは、一般に関数（メソッド）の呼び出しが入れ子状態になって実行される。即ち図１（ａ），（ｂ），（ｃ）に示すようにプログラムの処理に必要な関数は、実行中の呼出関数から実行すべき被呼出関数を呼び出して、メモリ中のスタック領域に確保された実行中の呼出関数の関数記録領域（フレーム）に、被呼出関数の形式に基づく新たな関数記録領域を積み上げる形で確保し、確保された関数記録領域を利用して呼び出された被呼出関数を実行し、被呼出関数の実行完了後、積み上げられている不要になった関数記録領域を破棄する。

【０００４】

最上位に積み上げられた実行中の関数を記録している関数記録領域は、トップフレームと呼ばれ、図１（ａ）では関数Ｆが実行中であり、関数Ｆの関数記録領域がトップフレームとなる。

この関数Ｆを呼出関数として、被呼出関数である関数Ｇが呼び出されると、図１（ｂ）に示すように、関数Ｇを実行するための関数記録領域がスタック領域に積み上げられてトップフレームとなり、関数Ｇの実行完了後、関数Ｇの実行結果を関数Ｆに渡し、当該関数記録領域は破棄されて、呼び出し前の図１（ａ）の状態となる。

【０００５】

なお図１（ｂ）において、関数Ｇを呼出関数として、更に他の被呼出関数である関数Ｈが呼び出されたときは、図１（ｃ）に示すように、関数Ｈを実行するための関数記録領域がスタック領域に積み上げられてトップフレームとなり、関数Ｈの実行完了後、関数Ｈの実行結果を関数Ｇに渡し、当該関数記録領域は破棄されて、呼び出し前の図１（ｂ）の状態となる。

このとき呼出関数である関数Ｇにより呼び出された被呼出関数である関数Ｈの実行結果を関数Ｇの実行結果として関数Ｆに渡す場合、関数Ｈの呼び出しを末尾呼び出しと呼び、末尾呼び出しでは関数Ｈの実行完了後、関数Ｈ及び関数Ｇの関数記録領域は連続して破棄されることになる。

【特許文献１】特開昭６０－８９４４号公報

【発明の開示】

【発明が解決しようとする課題】

【０００６】

しかしながらスタック領域に数多くの関数記録領域を積み上げていくことにより、スタック領域をオーバーフローするという状況を引き起こし、場合によってはプログラムの実行に支障を来すことがあるという問題がある。

【０００７】

また関数記録領域の積み上げ及び破棄に要する処理負荷が、全体の実行速度の低下を招くという問題がある。

【０００８】

本発明は斯かる事情に鑑みてなされたものであり、呼出関数による呼び出しが末尾呼び出しであると判断した場合に、新たな関数記録領域を積み上げることなく、呼出関数が記録されている関数記録領域を、被呼出関数を記録する関数記録領域として利用することで

、関数記録領域がスタック領域をオーバーフローしてプログラムの実行に支障を来す可能性を低減し、また積み上げ及び破棄に要する処理負荷を低減し、全体の実行速度を向上させる関数実行方法、その方法を適用した関数実行装置、その装置を実現するためのコンピュータプログラム、及びそのコンピュータプログラムを記録した記録媒体の提供を主たる目的とする。

【０００９】

さらにＪＶＭのバイトコード等の実行形式の呼出関数を実行する際に、上述した関数記録領域を再利用する処理を、実行すべき呼出関数を予め準備している代替関数に代替し、代替された代替関数を実行することにより実現するので、Ｊａｖａ言語等の言語によるソースコードの作成時には特別な配慮を行う必要が無く、またソースコードをバイトコード等の実行形式の関数に変換するコンパイラに特別な命令を追加する必要が無い関数実行方法等の提供を他の目的とする。

【００１０】

また呼出関数及び被呼出関数が同じ場合には、確保されている関数記録領域をそのまま利用し、呼出関数及び被呼出関数が異なる場合には、確保されている関数記録領域を被呼出関数の形式に基づいて変更することにより、全体の処理負荷の軽減及び関数記録領域の適正化を行うことが可能な関数実行方法等の提供を他の目的とする。

【課題を解決するための手段】

【００１３】

第１発明の関数実行方法は、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げ、積み上げた関数記録領域を利用して被呼出関数を呼び出し、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄する関数実行方法において、ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析し、該解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行し、かつ解析により呼出関数及び被呼出関数が異なると判別したとき、被呼出関数の形式に基づいて、呼出関数を実行する関数記録領域を変更して利用させることを特徴とする。

【００１４】

第１発明の関数実行方法では、第１の呼出関数による呼び出しが、被呼出関数の実行結果を呼出関数の実行結果とする末尾呼出であると判断した場合に、利用した関数記録領域を被呼出関数の記録領域として再利用する処理を含む新たに用意された関数である第２の呼出関数を、第１の呼出関数の代替関数として実行し、これによりスタック領域に積み上げられる関数記録領域の数を低減して、関数記録領域がスタック領域をオーバーフローしてプログラムの実行に支障を来す可能性を低減し、また関数記録領域の積み上げ及び破棄に要する処理負荷を低減して、全体の実行速度を向上させることが可能であり、しかも上述した第２の呼出関数を第１の呼出関数の代替関数として実行する処理は、Ｊａｖａ言語等の言語により記述されたソースコードをコンパイルして得られたバイトコード等の実行形式の第１の呼出関数を解析することにより行われるので、ソースコードの作成時には特別な配慮を行う必要が無く、またソースコードを変換するコンパイラに特別な命令を追加する必要がない。

【００１８】

さらに、呼出関数及び被呼出関数が異なるときに、呼出関数のために確保されている関数記録領域を被呼出関数の形式に基づいて変更することにより、被呼出関数を再利用した関数記録領域にて問題なく実行することが可能であり、また呼出関数及び被呼出関数が同じ関数であるときには、関数記録領域の形式を変更することなく、そのまま再利用するので形式の変更に要する処理を無くし、全体としての処理速度を向上させることが可能である。

【００２１】
第２発明の関数実行装置は、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げ、積み上げた関数記録領域を利用して被呼出関数を呼び出し、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄する関数実行装置において、ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析する手段と、該解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行する実行手段とを備え、該実行手段は、解析により呼出関数及び被呼出関数が異なると判別したとき、呼出関数を実行する関数記録領域を、被呼出関数の形式に基づいて変更する様に構成してある。

【００２２】
第２発明の関数実行装置では、第１の呼出関数による呼び出しが、被呼出関数の実行結果を呼出関数の実行結果とする末尾呼出であると判断した場合に、利用した関数記録領域を被呼出関数の記録領域として再利用する処理を含む新たに用意された関数である第２の呼出関数を、第１の呼出関数の代替関数として実行し、これによりスタック領域に積み上げられる関数記録領域の数を低減して、関数記録領域がスタック領域をオーバーフローしてプログラムの実行に支障を来す可能性を低減し、また関数記録領域の積み上げ及び破棄に要する処理負荷を低減して、全体の実行速度を向上させることが可能であり、しかも上述した第２の呼出関数を第１の呼出関数の代替関数として実行する処理は、Ｊａｖａ言語等の言語により記述されたソースコードをコンパイルして得られたバイトコード等の実行形式の第１の呼出関数を解析することにより行われるので、ソースコードの作成時には特別な配慮を行う必要が無く、またソースコードを変換するコンパイラに特別な命令を追加する必要がない。

【００２４】
さらに呼出関数及び被呼出関数が異なるときに、呼出関数のために確保されている関数記録領域を被呼出関数の形式に基づいて変更することにより、被呼出関数を再利用した関数記録領域にて問題なく実行することが可能であり、また呼出関数及び被呼出関数が同じ関数であるときには、関数記録領域の形式を変更することなく、そのまま再利用するので形式の変更に要する処理を無くし、全体としての処理速度を向上させることが可能である。

【００２５】
第３発明のコンピュータプログラムは、コンピュータに、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げさせ、積み上げた関数記録領域を利用して被呼出関数を呼び出させ、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄させるコンピュータプログラムにおいて、コンピュータに、ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析させる手順と、コンピュータに、解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行させる手順と、コンピュータに、解析により呼出関数及び被呼出関数が異なると判別したとき、被呼出関数を呼び出す領域として利用させるべく、呼出関数を実行する関数記録領域を、被呼出関数の形式に基づいて変更させる手順とを含む。

【００２６】
第３発明のコンピュータプログラムでは、携帯電話及びパーソナルコンピュータ等の処理装置を用いたＪＶＭにて実行することで、ＪＶＭが関数実行装置として動作することにより、第１の呼出関数による呼び出しが、被呼出関数の実行結果を呼出関数の実行結果とする末尾呼出であると判断した場合に、利用した関数記録領域を被呼出関数の記録領域と

して再利用する処理を含む新たに用意された関数である第２の呼出関数を、第１の呼出関数の代替関数として実行し、これによりスタック領域に積み上げられる関数記録領域の数を低減して、関数記録領域がスタック領域をオーバーフローしてプログラムの実行に支障を来す可能性を低減し、また関数記録領域の積み上げ及び破棄に要する処理負荷を低減して、全体の実行速度を向上させることが可能であり、しかも上述した第２の呼出関数を第１の呼出関数の代替関数として実行する処理は、Ｊａｖａ言語等の言語により記述されたソースコードをコンパイルして得られたバイトコード等の実行形式の第１の呼出関数を解析することにより行われるので、ソースコードの作成時には特別な配慮を行う必要が無く、またソースコードを変換するコンパイラに特別な命令を追加する必要がない。

【００３０】

さらに、呼出関数及び被呼出関数が異なる関数であるときに、呼出関数のために確保されている関数記録領域を被呼出関数の形式に基づいて変更することにより、被呼出関数を再利用した関数記録領域にて問題なく実行することが可能であり、また呼出関数及び被呼出関数が同じ関数であるときには、関数記録領域の形式を変更することなく、そのまま再利用するので形式の変更に要する処理を無くし、全体としての処理速度を向上させることが可能である。

【００３３】

第４発明のコンピュータでの読み取りが可能な記録媒体は、コンピュータに、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げさせ、積み上げた関数記録領域を利用して被呼出関数を呼び出させ、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄させるコンピュータプログラムを記録してあるコンピュータでの読み取りが可能な記録媒体において、コンピュータに、ソースコードをコンパイルして得られた、関数記録領域で実行される第１の呼出関数の実行形式を解析させる手順と、コンピュータに、解析により第１の呼出関数にて呼び出すべき被呼出関数の実行結果が呼出関数の実行結果となると判断した場合に、呼出関数を実行する関数記録領域を、被呼出関数を呼び出す領域として利用させる処理を含む第１と異なる第２の呼出関数を第１の呼出関数の代替関数として実行させる手順と、コンピュータに、解析により呼出関数及び被呼出関数が異なると判別したとき、被呼出関数を呼び出す領域として利用させるべく、呼出関数を実行する関数記録領域を、被呼出関数の形式に基づいて変更させる手順とを含むコンピュータプログラムを記録してある。

【００３４】

第４発明のコンピュータでの読み取りが可能な記録媒体では、記録されているコンピュータプログラムを携帯電話及びパーソナルコンピュータ等の処理装置を用いたＪＶＭにて実行することで、ＪＶＭが関数実行装置として動作することにより、第１の呼出関数による呼び出しが、被呼出関数の実行結果を呼出関数の実行結果とする末尾呼出であると判断した場合に、利用した関数記録領域を被呼出関数の記録領域として再利用する処理を含む新たに用意された関数である第２の呼出関数を、第１の呼出関数の代替関数として実行し、これによりスタック領域に積み上げられる関数記録領域の数を低減して、関数記録領域がスタック領域をオーバーフローしてプログラムの実行に支障を来す可能性を低減し、また関数記録領域の積み上げ及び破棄に要する処理負荷を低減して、全体の実行速度を向上させることが可能であり、しかも上述した第２の呼出関数を第１の呼出関数の代替関数として実行する処理は、Ｊａｖａ言語等の言語により記述されたソースコードをコンパイルして得られたバイトコード等の実行形式の第１の呼出関数を解析することにより行われるので、ソースコードの作成時には特別な配慮を行う必要が無く、またソースコードを変換するコンパイラに特別な命令を追加する必要がない。

さらに呼出関数及び被呼出関数が異なる関数であるときに、呼出関数のために確保されている関数記録領域を被呼出関数の形式に基づいて変更することにより、被呼出関数を再利用した関数記録領域にて問題なく実行することが可能であり、また呼出関数及び被呼出関数が同じ関数であるときには、関数記録領域の形式を変更することなく、そのまま再利

用するので形式の変更に要する処理を無くし、全体としての処理速度を向上させることが可能である。

【発明の効果】

【００３５】

本発明に係る命令実行方法、命令実行装置、コンピュータプログラム、及び記録媒体では、Ｊａｖａ言語等の言語にて記述され複数の関数を含むプログラムを、携帯電話及びパーソナルコンピュータ等の処理装置を用いたＪＶＭにて実行する場合に、メモリ中のスタック領域に他の関数を呼び出す処理を含む第１の呼出関数の実行形式を解析し、第１の呼出関数による呼び出しが、被呼出関数の実行結果を呼出関数の実行結果とする末尾呼び出しであると判断したときに、呼出関数の関数記録領域を被呼出関数の記録領域として再利用する処理を含む新たに用意された関数である第２の呼出関数を、第１の呼出関数の代替関数として実行し、スタック領域に積み上げられる関数記録領域の数を低減することで、関数記録領域がスタック領域をオーバーフローしてプログラムの実行に支障を来す可能性を低減し、また関数記録領域の積み上げ及び破棄に要する処理負荷を低減して、全体の実行速度を向上させることが可能である等、優れた効果を奏する。

【００３６】

また上述した第２の呼出関数を第１の呼出関数の代替関数として実行する処理は、Ｊａｖａ言語等の言語により記述されたソースコードをコンパイルして得られたバイトコード等の実行形式の第１の呼出関数を解析することにより行われるので、ソースコードの作成時には特別な配慮を行う必要が無く、またソースコードを変換するコンパイラに特別な命令を追加する必要がない等、優れた効果を奏する。

【００３７】

さらに本発明では呼出関数及び被呼出関数が異なるときに、呼出関数のために確保されている関数記録領域を被呼出関数の形式に基づいて変更することにより、被呼出関数を再利用した関数記録領域にて問題なく実行することが可能であり、また呼出関数及び被呼出関数が同じ関数であるときには、関数記録領域の形式を変更することなく、そのまま再利用するので形式の変更に要する処理を無くし、全体としての処理速度を向上させることが可能である等、優れた効果を奏する。

【発明を実施するための最良の形態】

【００３８】

以下、本発明をその実施の形態を示す図面に基づいて詳述する。

【００３９】

図２は本発明の関数実行装置を示すブロック図である。

図中１０は携帯電話及びパーソナルコンピュータ等の処理装置を用いた本発明の関数実行装置であり、関数実行装置１０は、本発明の関数実行装置用のコンピュータプログラムＰＧ及びデータ等の情報を記録したＣＤ－ＲＯＭ及びメモリカード等の記録媒体ＲＥＣからコンピュータプログラムＰＧ及びデータ等の情報を読み取る補助記憶手段１２、補助記憶手段１２により読み取られたコンピュータプログラムＰＧ及びデータ等の情報を記録するハードディスク等の記録手段１３、並びに各種情報を一時的に記録するメモリ手段１４を備えている。

【００４０】

そして記録手段１３からコンピュータプログラムＰＧ及びデータ等の情報を読み取り、メモリ手段１４に記録してＣＰＵ１１により実行することで、処理装置（コンピュータ）は本発明の関数実行装置１０として動作する。

【００４１】

また関数実行装置１０は、モデム、ＴＡ(Terminal Adapter)、及びアンテナ等の通信手段１５を備えており、通信手段１５によりインターネット等の通信ネットワークＮＷに接続して、通信ネットワークＮＷに接続するウェブサーバコンピュータ等の記録装置２０が備える記録媒体２１に記録された本発明のコンピュータプログラムＰＧ及びデータ等の情報を取り込み、実行するようにしてもよい。

【００４２】
次に本発明の関数実行方法の処理内容を説明する。
【００４３】
本発明の関数実行方法は、Ｊａｖａ言語等の言語にて記述されたソースコードを、汎用性のコンパイラを用いてＪＶＭのバイトコード等の実行形式にコンパイルすることにより得られた複数の命令からなる関数を複数含むプログラムを実行する方法に適用されるものであり、メモリ手段１４中のスタック領域に、実行すべき関数についての引数の個数及びローカル変数領域の大きさ等の形式に基づく関数記録領域を積み上げ、積み上げた関数記録領域に、関数を呼び出して実行する関数実行方法を基本とする。
【００４４】
図３は本発明の関数実行方法を概念的に示す説明図である。
図３（ａ）はスタック関数領域に確保された関数記録領域にて実行形式の関数Ｆが実行されている状態を示す。
そして関数記録領域にて実行される関数Ｆを呼出関数とし、関数Ｆに呼び出される被呼出関数である実行形式の関数Ｇを呼び出す場合、図３（ｂ）に示すように関数Ｆの関数記録領域に、関数Ｇを呼び出すための関数記録領域を新たに積み上げ、積み上げた関数記録領域を利用して関数Ｇを呼び出し実行する。
【００４５】
さらに関数Ｇを呼出関数とし、関数Ｇに呼び出される被呼出関数である実行形式の関数Ｈを呼び出す場合で、関数Ｈの実行結果が関数Ｇの実行結果となる末尾呼出であるときに、図３（ｃ）に示すように関数Ｈのための関数記録領域を新たに積み上げることなく、関数Ｇが記録されている関数記録領域を、関数Ｈを呼び出すための関数記録領域とする。
そして関数Ｈの実行結果は関数Ｆに渡され、関数Ｈを実行した関数記録領域は破棄される。
【００４６】
次に本発明の関数実行方法における解析処理を図４に示すフローチャートを用いて説明する。
ＪＶＭ等のバイトコードによる関数を解析するにあたり、呼び出すべき関数が、ｉｎｖｏｋｅｓｔａｔｉｃ、ｉｎｖｏｋｅｖｉｒｔｕａｌ、ｉｎｖｏｋｅｓｐｅｃｉａｌ、及びｉｎｖｏｋｅｉｎｔｅｒｆａｃｅ等の被呼出関数を呼び出す命令を含む呼出関数である場合に、当該呼出関数の実行形式を解析し（Ｓ１０１）、解析により、呼出関数が、被呼出関数の実行結果が呼出関数の実行結果となるという条件を満足する末尾呼出関数であると判断したときに（Ｓ１０２：Ｙ）、更に呼出関数及び被呼出関数が同じであるか否かを判別する（Ｓ１０３）。
【００４７】
なおステップＳ１０２において末尾呼出関数であると判断する基準のＪａｖａ言語における具体例としては、被呼出関数を呼び出す命令の直後に任意個のプログラムカウンタ以外を変化させないｎｏｐ及びｇｏｔｏ等の命令を挟んで復帰命令があり、被呼出関数の戻り値の型と、ｉｒｅｔｕｒｎ、ｌｒｅｔｕｒｎ、ｆｒｅｔｕｒｎ、ｄｒｅｔｕｒｎ、ａｒｅｔｕｒｎ、及びｒｅｔｕｒｎ等の復帰命令の型とが一致し、被呼出関数を呼び出す命令から復帰命令の間に例外ハンドラが設定されていない等の基準がある。
【００４８】
ステップＳ１０３において、呼出関数及び被呼出関数が異なると判別したとき（Ｓ１０３：Ｎ）、当該呼出関数（第１の呼出関数）を、被呼出関数を呼び出す命令を予め用意している代替命令に置換した再帰代替関数（第２の呼出関数）に置換する（Ｓ１０４）。
【００４９】
呼出関数及び被呼出関数が同じであると判別したとき（Ｓ１０３：Ｙ）当該呼出関数（第１の呼出関数）を、呼出関数と同じである被呼出関数を呼び出す命令を予め用意している代替命令に置換した自己再帰代替関数（第２の呼出関数）に置換する（Ｓ１０５）。
なお置換すべき命令が複数個含まれる場合、該当する全ての命令を置換した関数に置換

される。

【００５０】

ステップＳ１０２において、末尾呼出関数でないと判断したとき（Ｓ１０２：Ｎ）、ステップＳ１０３～Ｓ１０５の処理は行われない。

なお呼出関数の代替関数として用いられるステップＳ１０４に示す再帰代替関数及びステップＳ１０５に示す自己再帰代替関数として、以下に示す新たな命令を含む関数を用意しておく。

【００５１】

即ち、

ｉｎｖｏｋｅｓｔａｔｉｃ、

ｉｎｖｏｋｅｖｉｒｔｕａｌ、

ｉｎｖｏｋｅｓｐｅｃｉａｌ、

及びｉｎｖｏｋｅｉｎｔｅｒｆａｃｅ

等の呼出命令を含む呼出関数の再帰代替関数として、

ｔａｉｌｉｎｖｏｋｅｓｔａｔｉｃ、

ｔａｉｌｉｎｖｏｋｅｖｉｒｔｕａｌ、

ｔａｉｌｉｎｖｏｋｅｓｐｅｃｉａｌ、

及びｔａｉｌｉｎｖｏｋｅｉｎｔｅｒｆａｃｅ

等の代替命令を用意し、用意した代替命令を含む再帰代替関数を用いる。

【００５２】

また自己再帰代替関数として、

ｓｅｌｆｔａｉｌｉｎｖｏｋｅｓｔａｔｉｃ、

ｓｅｌｆｔａｉｌｉｎｖｏｋｅｖｉｒｔｕａｌ、

及びｓｅｌｆｔａｉｌｉｎｖｏｋｅｉｎｔｅｒｆａｃｅ

等の代替命令を用意し、用意した代替命令を含む自己再帰代替関数を用いる。

【００５３】

これらの命令を含む関数は、解析処理では置換されるだけであり、実際に実行されるのは、置換された関数を実行する実行処理においてであるので、以下の実行処理の説明にてその機能を説明する。

図５及び図６は本発明の関数実行方法における実行処理を示すフローチャートである。

実行形式の解析処理により必要に応じて置換がなされた関数を実行する場合、先ず実行すべき呼出関数が、代替関数である再帰代替関数或いは自己再帰代替関数、又は代替関数で無いかを判別し（Ｓ２０１）、再帰代替関数であると判断した場合（Ｓ２０１：１）、再帰代替関数（第２の呼出関数）の形式に基づく関数記録領域を積み上げ（Ｓ２０２）、積み上げた関数記録領域を利用して再帰代替関数を呼び出し（Ｓ２０３）、ステップＳ２０２にて積み上げた関数記録領域にて再帰代替関数を実行する（Ｓ２０４）。

【００５４】

そして代替関数に含まれるｔａｉｌｉｎｖｏｋｅｓｔａｔｉｃ等の代替命令による処理により、呼出関数（第１の呼出関数）として実行している再帰代替関数（第２の呼出関数）から被呼出関数を呼び出す場合に、新たに関数記録領域を積み上げることなく、再帰代替関数を実行した関数記録領域を、被呼出関数の形式に基づいて変更し（Ｓ２０５）、形式を変更した関数記録領域を、被呼出関数を呼び出す領域として利用して（Ｓ２０６）、被呼出関数を呼び出し（Ｓ２０７）、呼び出した被呼出関数を実行し（Ｓ２０８）、被呼出関数の実行後、被呼出関数の実行に利用した関数記録領域を破棄する（Ｓ２０９）。

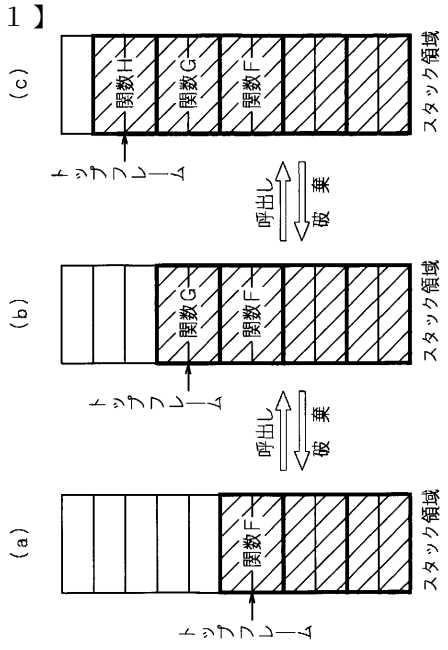
【００５５】

ステップＳ２０１において、自己再帰代替関数であると判断した場合（Ｓ２０１：２）、自己再帰代替関数（第２の呼出関数）の形式に基づく関数記録領域を積み上げ（Ｓ２１０）、積み上げた関数記録領域を利用して自己再帰代替関数を呼び出し（Ｓ２１１）、ステップＳ２１０にて積み上げた関数記録領域にて自己再帰代替関数を実行する（Ｓ２１２）。

【００５６】
そして自己再帰代替関数に含まれるｓｅｌｆｔａｉｌｉｎｖｏｋｅｓｔａｔｉｃ等の代替命令による処理により、呼出関数（第１の呼出関数）として実行している自己再帰代替関数（第２の呼出関数）から被呼出関数を呼び出す場合に、新たに関数記録領域を積み上げることなく、自己再帰代替関数を実行した関数記録領域を、被呼出関数を呼び出す領域として利用して（Ｓ２１３）、被呼出関数を呼び出し（Ｓ２１４）、呼び出した被呼出関数を実行し（Ｓ２１５）、被呼出関数の実行後、被呼出関数の実行に利用した関数記録領域を破棄する（Ｓ２１６）。
【００５７】
ステップＳ２０１において、代替関数でないと判断した場合（Ｓ２０１：３）、呼出関数の形式に基づく関数記録領域を積み上げ（Ｓ２１７）、積み上げた関数記録領域を利用して呼出関数を呼び出し（Ｓ２１８）、ステップＳ２１７にて積み上げられた関数記録領域にて呼出関数を実行する（Ｓ２１９）。
【００５８】
そして呼出関数から被呼出関数を呼び出す場合に、被呼出関数の形式に基づく新たな関数記録領域を積み上げ（Ｓ２２０）、積み上げた関数記録領域を利用して被呼出関数を呼び出し（Ｓ２２１）、ステップＳ２２０にて積み上げた関数記録領域にて被呼出関数を実行し（Ｓ２２２）、被呼出関数の実行後、被呼出関数の実行に利用した関数記録領域を破棄し（Ｓ２２３）、更に呼出関数における被呼出関数実行後の処理を実行して、呼出関数の実行に利用した関数記録領域を破棄する（Ｓ２２４）。
【００５９】
次に本発明の関数実行方法及び従来の関数実行方法の処理速度を比較した結果を図７に示すグラフを用いて説明する。
図７では横軸に末尾呼出関数の呼び出し深さをとり、縦軸にマイクロ秒（ｕｓ）を単位とする処理時間をとって、その関係を示したものであり、×印は呼出関数を呼び出す都度、関数記録領域を積み上げる従来の関数実行方法による呼び出し深さと処理時間との関係を示し、□印は呼出関数が末尾呼出の場合に関数記録領域を再利用する本発明の関数実行方法による呼び出し深さと処理時間との関係を示し、○印は呼出関数と被呼出関数とが同じである自己末尾呼出関数を扱う場合の呼び出し深さと処理時間との関係を示している。
【００６０】
図７に示すように従来の関数実行方法と比較して、本発明の関数実行方法は処理時間が短く、特に自己末尾呼出関数ではその傾向が顕著である。
また本発明の実施の形態としては、ＪＩＴ(Just In Time Compiler)と呼ばれるＪＶＭの実装技術を用いてもよく、ＪＩＴ技術により、ＪＶＭのバイトコードを機械語に変換して、ＣＰＵ１１にて実行することで、実行処理の速度を向上させることが可能である。
【００６１】
そして前記実施の形態では、呼出関数が末尾呼出又は自己末尾呼出である場合に、代替関数を用いる形態を示したが、本発明はこれに限らず、関数実行時に都度再帰判定を行い、末尾再帰であると判断した場合に、末尾再帰処理を行うようにしてもよい。
【図面の簡単な説明】
【００６２】
【図１】プログラムの実行に必要な関数を記録する関数記録領域を示す概念図である。
【図２】本発明の関数実行装置を示すブロック図である。
【図３】本発明の関数実行方法を概念的に示す説明図である。
【図４】本発明の関数実行方法における解析処理を示すフローチャートである。
【図５】本発明の関数実行方法における実行処理を示すフローチャートである。
【図６】本発明の関数実行方法における実行処理を示すフローチャートである。
【図７】本発明の関数実行方法及び従来の関数実行方法の処理速度を示すグラフである。
【符号の説明】
【００６３】

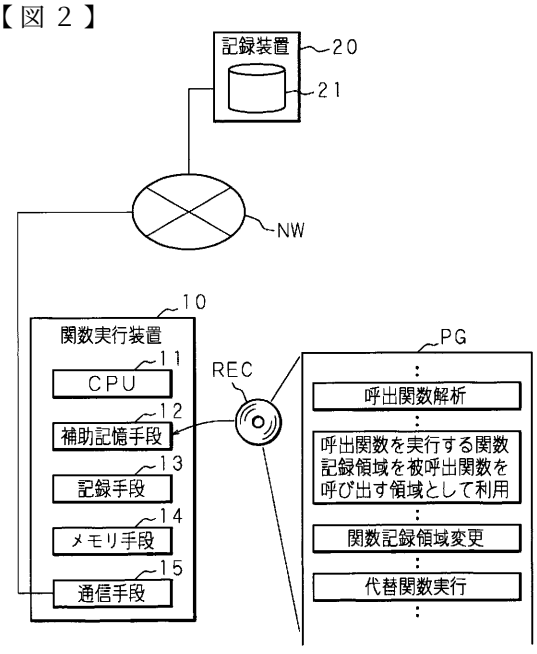
１０　関数実行装置
ＰＧ　コンピュータプログラム
ＲＥＣ　記録媒体

【図１】



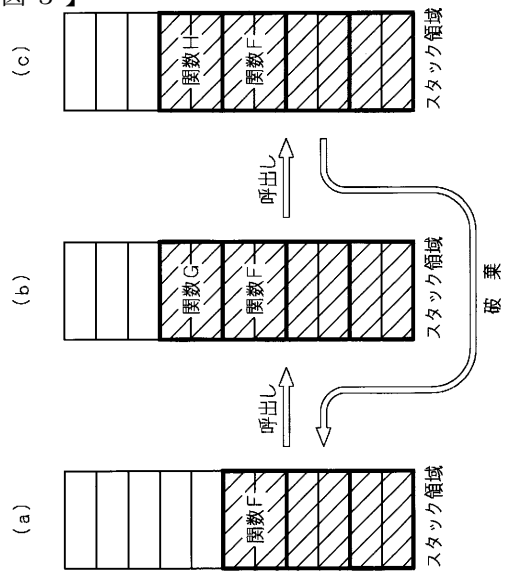
第１図

【図２】



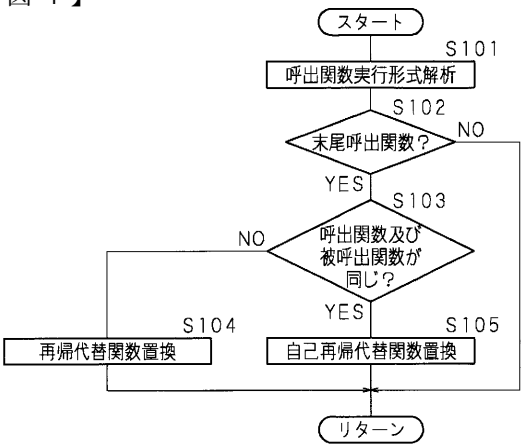
第２図

【図３】



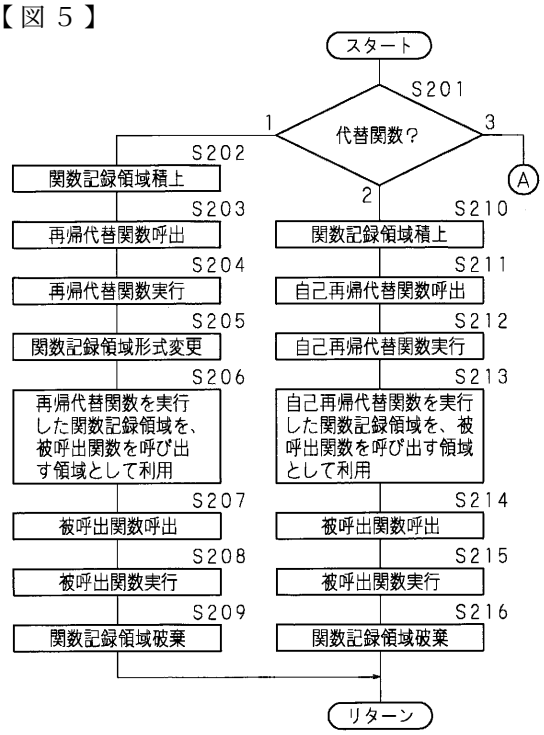
第３図

【図４】



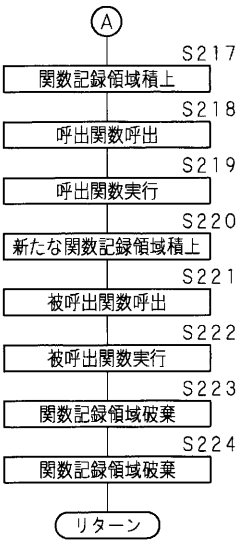
第４図

【図５】



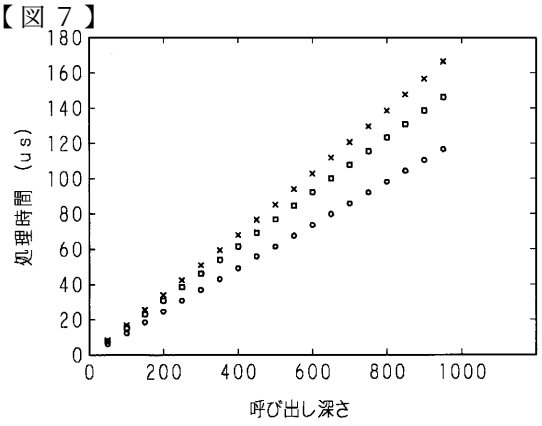
第５図

【図６】



第６図

【図７】


処理時間（us）
呼び出し深さ
0
20
40
60
80
100
120
140
160
180
0
200
400
600
800
1000

第７図

フロントページの続き

(56)参考文献 特開昭６０－８９４４（ＪＰ，Ａ）
特開平４－２４５５４３（ＪＰ，Ａ）
河合英治 外2名，”高速かつ移植性の高いＪａｖａ Ｖｉｒｔｕａｌ Ｍａｃｈｉｎｅの設計と実装”，情報処理学会研究報告（９８－ＯＳ－７７ ９８－ＤＰＳ－８７），１９９８年 ２月２６日，第９８巻，第１５号，ｐ．２５－３０
石崎一明 外8名，”Ｊａｖａ Ｊｕｓｔ－Ｉｎ－Ｔｉｍｅコンパイラにおける最適化とその評価”，電子情報通信学会技術研究報告（ＣＰＳＹ９９－６２～７２），１９９９年 8月 5日，第９９巻，第２５２号，ｐ．１７－２４

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/42 - 9/44