(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2002 年10 月3 日 (03.10.2002)

PCT

(10) 国際公開番号
WO 02/077802 A1

(51) 国際特許分類[7]: G06F 9/44

(21) 国際出願番号: PCT/JP02/02888

(22) 国際出願日: 2002 年3 月25 日 (25.03.2002)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願2001-88850 2001 年3 月26 日 (26.03.2001) JP

(71) 出願人 (米国を除く全ての指定国について): 関西ティー・エル・オー株式会社 (KANSAI TECHNOLOGY LICENSING ORGANIZATION CO.,LTD.) [JP/JP]; 〒600-8815 京都府 京都市 下京区中堂寺粟田町1番地 Kyoto (JP).

(72) 発明者; および
(75) 発明者/出願人 (米国についてのみ): 山本 晃成 (YAMAMOTO,Akishige) [JP/JP]; 〒160-0022 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 Tokyo (JP). 湯淺 太一 (YUASA,Taiichi) [JP/JP]; 〒606-8501 京都府 京都市 左京区吉田本町 京都大学内 Kyoto (JP).

(74) 代理人: 河野 登夫 (KOHNO,Takao); 〒540-0035 大阪府 大阪市中央区 釣鐘町二丁目４番３号 河野特許事務所 Osaka (JP).

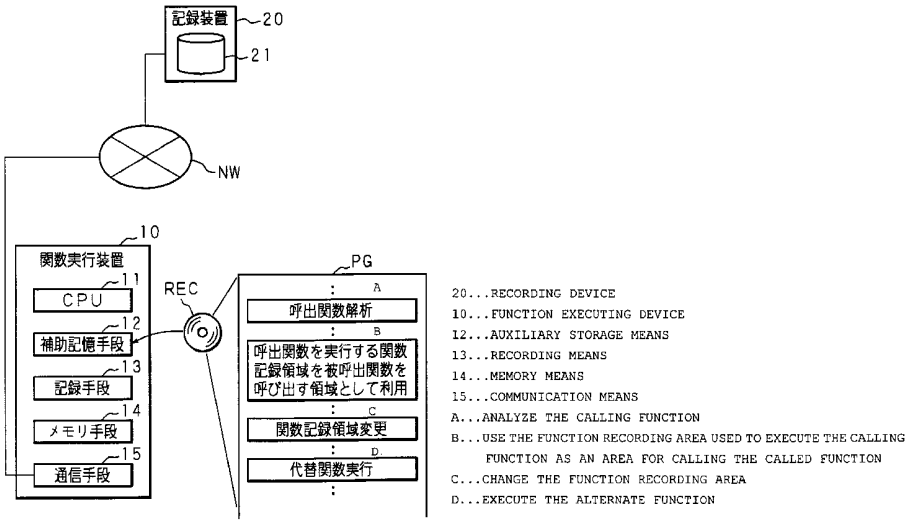
(81) 指定国 (国内): JP, US.

(84) 指定国 (広域): ヨーロッパ特許 (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

添付公開書類:
— 国際調査報告書

(54) Title: FUNCTION EXECUTING METHOD, FUNCTION EXECUTING DEVICE, COMPUTER PROGRAM AND RECORDING MEDIUM

(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) Abstract: There are provided a function executing method for executing a called function, a function executing device, a computer program and a recording function. In the method, to execute a program for stacking function recording areas conforming to the type of a called function called by a calling function including a call of another function in a stack area of a memory, calling a called function, executing the called function, and discarding the function recording areas; the calling function defined by a byte code such as of the JVM is analyzed, a predetermined alternate function is substituted for the calling function if the calling function is judged to be a tail calling function, and the substituted alternate function is executed. Thus, the called function is called by the calling function, which is the alternate function, by making use of the function recording areas used for executing the calling function and the called function is executed.

[続葉有]

WO 02/077802 A1

(57) 要約:

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て、被呼出関数を呼び出し、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プログラムの実行において、ＪＶＭ等のバイトコードによる呼出関数を解析し、呼出関数が末尾呼出関数であると判断したときに、呼出関数を所定の代替関数に置換し、置換した代替関数を実行することで、代替関数である呼出関数から被呼出関数を呼び出す場合に、呼出関数の実行に利用した関数記録領域を利用して被呼出関数を呼び出し、呼び出した被呼出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。

明　細　書

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

技術分野

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言語等のオブジェクト指向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

Ｊａｖａ言語等のオブジェクト指向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。

第１図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。即ち第１図（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。

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、第１図（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、第１図（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、呼び出し前の第１図（ａ）の状態となる。

なお第１図（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、第１図（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、呼び出し前の第１図（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。

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。

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。

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。

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。

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。

発明の開示

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、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、関数記録領域で実行される呼出関数を解析し、該解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、被呼出関数を呼び出す領域として利用する。

第１発明の関数実行方法では、呼出関数が所定の条件を満足した場合に、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して、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。

第２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、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の呼出関数が所定の条件を満足す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。

第２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び出しである等の所定の条件を満足す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の呼出関数を実行する際に行われるので、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。

第３発明の関数実行方法は、第１発明又は第２発明において、前記所定の条件は、被呼出関数の実行結果が呼出関数の実行結果となることである。

第３発明の関数実行方法では、呼出関数が所定の条件を満足した場合に、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

利用して、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。

第４発明の関数実行方法は、第１発明乃至第３発明のいずれかにおいて、前記呼出関数及び被呼出関数が異なるときに、被呼出関数の形式に基づいて、前記関数記録領域を変更する。

第４発明の関数実行方法では、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。

第５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から呼び出された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、関数記録領域で実行される呼出関数を解析する手段と、該解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、被呼出関数を呼び出す領域として利用する手段とを備える。

第５発明の関数実行装置では、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等の所定の条件を満足すると判断した場合に、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。

第６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、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の呼出関数が所定の条件を満足す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手段とを備える。

第６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び出しである等の所定の条件を満足す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の呼出関数を実行する際に行われるので、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。

第７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、関数記録領域で実行される呼出関数を解析させる手順と、コンピュータに、解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、被呼出関数を呼び出す領域として利用させる手順とを含む。

第７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、呼出関数が所定の条件を満足した場合に、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。

第８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、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の呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、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を含む。

第８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び出しである等の所定の条件を満足す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

呼出関数を実行する際に行われるので、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。

第９発明のコンピュータプログラムは、第７発明又は第８発明において、前記所定の条件は、被呼出関数の実行結果が呼出関数の実行結果となることである。

第９発明のコンピュータプログラムでは、呼出関数による呼び出しが末尾呼び出し、即ち被呼出関数の実行結果が呼出関数の実行結果となり、被呼出関数の実行完了後、被呼出関数の関数記録領域に続いて、呼出関数の関数記録領域も破棄される呼び出しであることを条件とすることにより、呼出関数の関数記録領域を被呼出関数の関数記録領域として問題を発生することなく再利用することが可能である。

第１０発明のコンピュータプログラムは、第７発明乃至第９発明のいずれかにおいて、前記呼出関数及び被呼出関数が異なるときに、被呼出関数の形式に基づいて、前記関数記録領域を変更する。

第１０発明のコンピュータプログラムでは、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。

第１１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、関数記録領域で実行される呼出関数を解析させる手順と、コンピュータに、解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、被呼出関数を呼び出す領域として利用させる手順と、を含むコンピュータプログラムを記録してある。

第１１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、呼出関数が所定の条件を満足した場合に、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。

第１２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、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の呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、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を含むコンピュータプログラムを記録してある。

第１２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び出しである等の所定の条件を満足す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の呼出関数を実行する際に行われるので、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。

図面の簡単な説明

第１図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、第２図は本発明の関数実行装置を示すブロック図、第３図は本発明の関数実行方法を概念的に示す説明図、第４図は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、第５図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、第６図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、第７図は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。

発明を実施するための最良の形態

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。

第２図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。

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。

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。

次に本発明の関数実行方法の処理内容を説明する。

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。

第３図は本発明の関数実行方法を概念的に示す説明図である。

第３図（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、第３図（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。

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、第３図（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。

次に本発明の関数実行方法における解析処理を第４図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実行するにあたり、先ず新たに関数記録領域を確保し、そして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）。

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。

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）。

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。

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。

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。

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。

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）

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）。

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２１０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）。

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）。

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２１７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）。

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）。

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第７図に示すグラフを用いて説明する。

第７図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。

第７図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。

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。

産業上の利用可能性

以上詳述した如く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にて実行する場合で、メモリ中のスタック領域に、他の関数を呼び出す処理を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等の所定の条件を満足す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。

また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等、優れた効果を奏する。

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。

請　求　の　範　囲

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、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、

関数記録領域で実行される呼出関数を解析し、

該解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、被呼出関数を呼び出す領域として利用する

ことを特徴とする関数実行方法。

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方法において、

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の呼出関数が所定の条件を満足す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

ことを特徴とする関数実行方法。

3．前記所定の条件は、被呼出関数の実行結果が呼出関数の実行結果となることであることを特徴とする請求項1又は請求項2に記載の関数実行方法。

4．前記呼出関数及び被呼出関数が異なるときに、被呼出関数の

形式に基づいて、前記関数記録領域を変更することを特徴とする請求項１乃至請求項３のいずれかに記載の関数実行方法。

５．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た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、

該解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、被呼出関数を呼び出す領域として利用する手段と

を備えることを特徴とする関数実行装置。

６．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、

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の呼出関数が所定の条件を満足す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手段と

を備えることを特徴とする関数実行装置。

７．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、関数記録領域で実行される呼出関数を解析させる手順と、

コンピュータに、解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、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。

８．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、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の呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、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

を含むことを特徴とするコンピュータプログラム。

９．前記所定の条件は、被呼出関数の実行結果が呼出関数の実行結果となることであることを特徴とする請求項７又は請求項８に記載のコンピュータプログラム。

１０．前記呼出関数及び被呼出関数が異なるときに、被呼出関数の形式に基づいて、前記関数記録領域を変更することを特徴とする請求項７乃至請求項９のいずれかに記載のコンピュータプログラム。

１１．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、関数記録領域で実行される呼出関数を解析させる手順と、

コンピュータに、解析により呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、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。

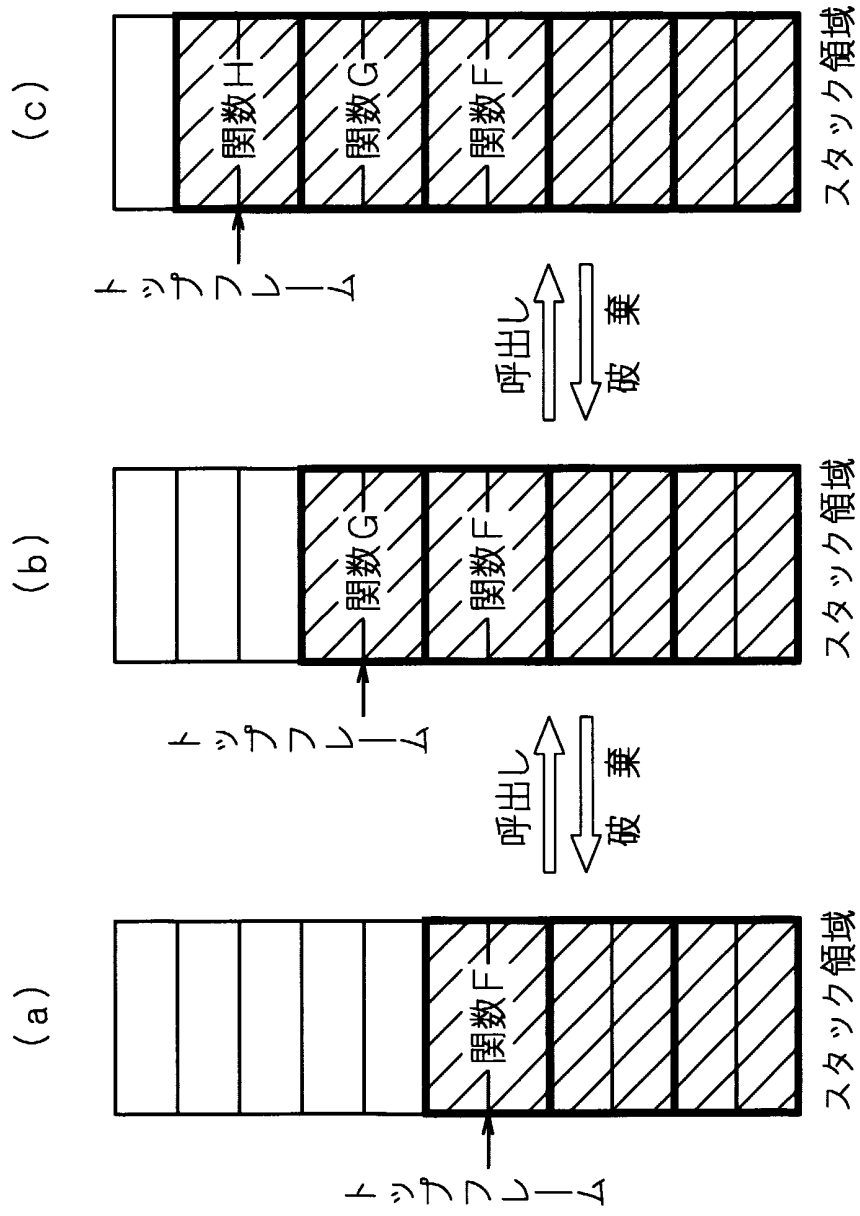
１２．コンピュータに、メモリ中のスタック領域に、他の関数を呼び出す処理を含む呼出関数を実行することにより呼び出される被呼出関数の形式に基づく関数記録領域を積み上げさせ、積み上げた関数記録領域を利用して被呼出関数を呼び出させ、呼び出された被呼出関数の実行後、積み上げた関数記録領域を破棄させるコンピュータプログラムを記録してある、コンピュータでの読み取りが可能な記録媒体において、

コンピュータに、関数記録領域で実行される第１の呼出関数の実

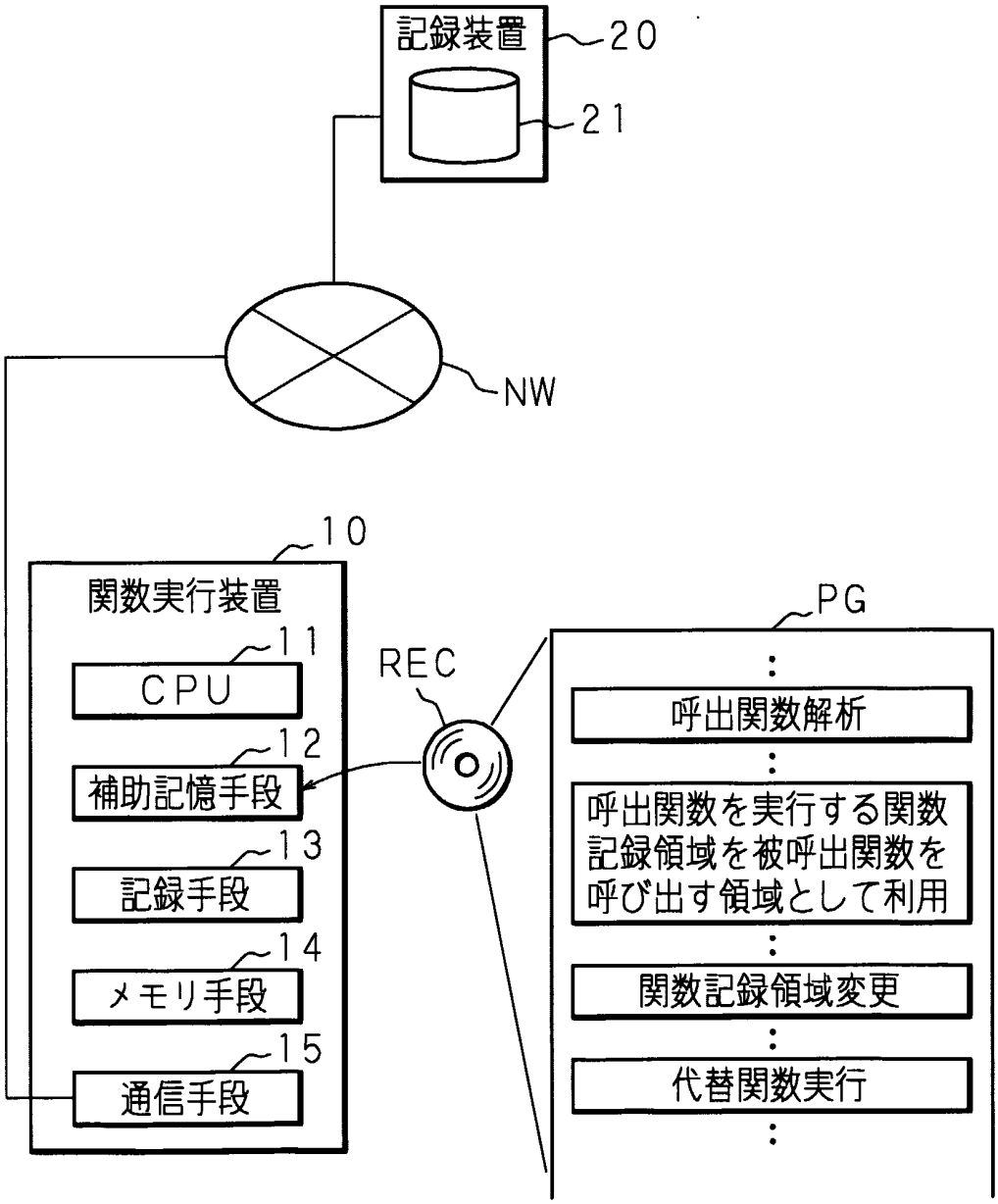
行形式を解析させる手順と、

コンピュータに、解析により第１の呼出関数が所定の条件を満足すると判断した場合に、呼出関数を実行する前記関数記録領域を、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

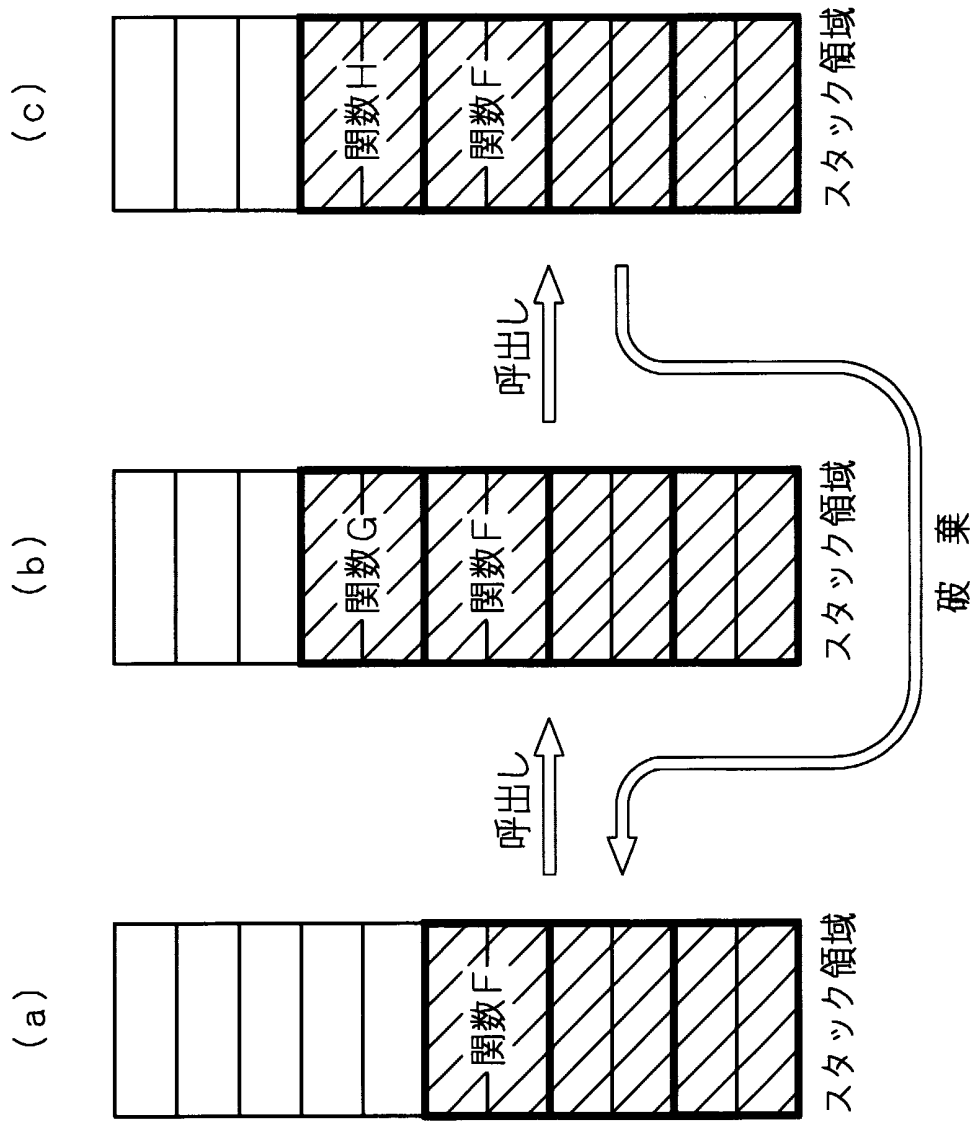
を含むコンピュータプログラムを記録してあることを特徴とするコンピュータでの読み取りが可能な記録媒体。


(c)
関数H
関数G
関数F
スタック領域
トップフレーム
呼出し
破棄
(b)
関数G
関数F
スタック領域
トップフレーム
呼出し
破棄
(a)
関数F
スタック領域
トップフレーム

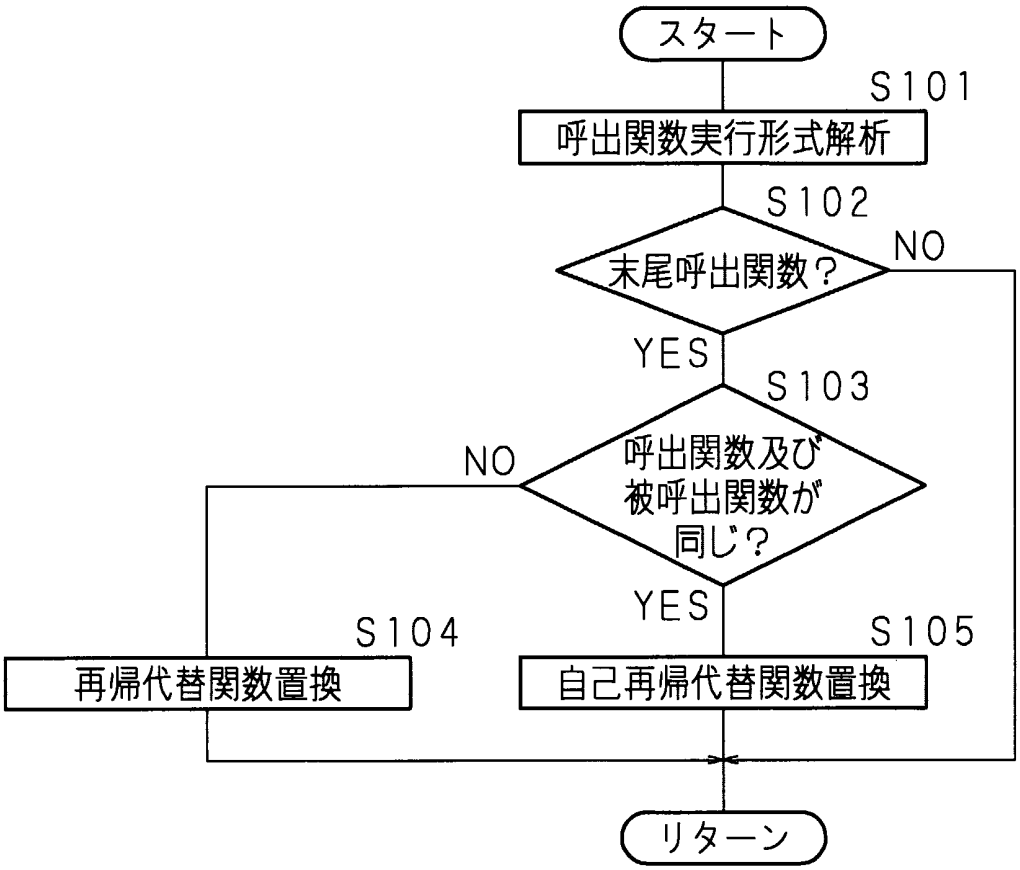
第 1 図


記録装置
20
21
NW
10
関数実行装置
11
CPU
12
補助記憶手段
13
記録手段
14
メモリ手段
15
通信手段
REC
PG
呼出関数解析
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
関数記録領域変更
代替関数実行

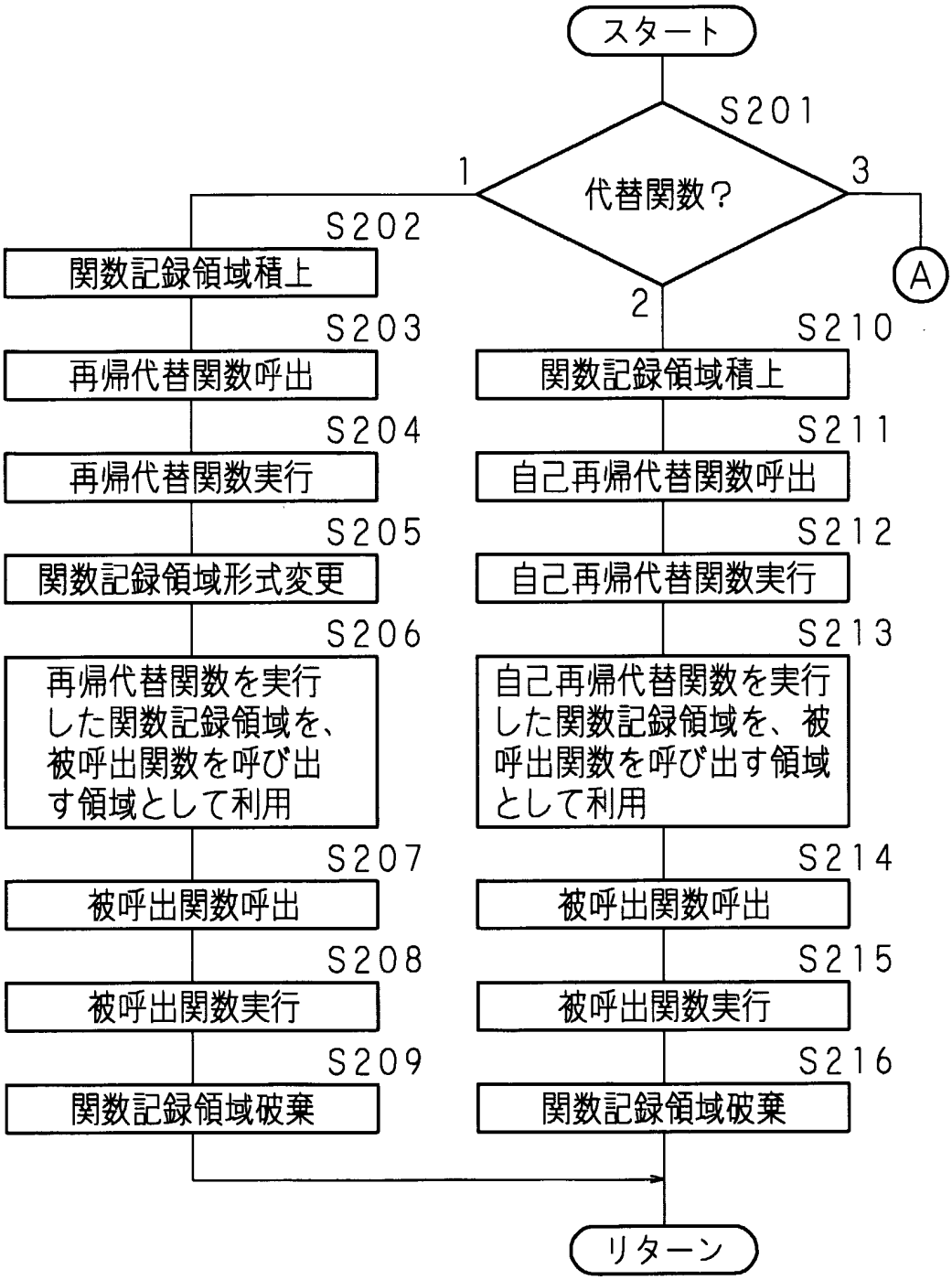
第 2 図



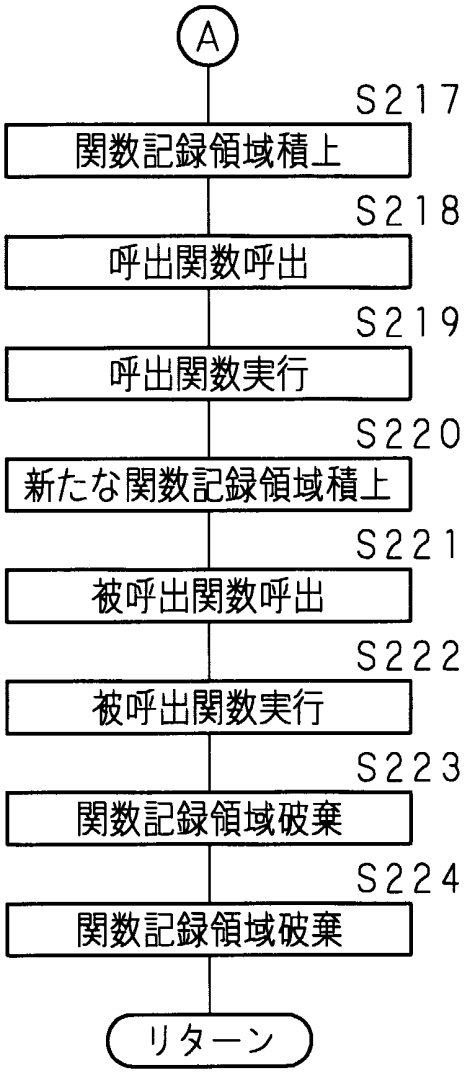
第 3 図


スタート
S101
呼出関数実行形式解析
S102
末尾呼出関数？
NO
YES
S103
呼出関数及び
被呼出関数が
同じ？
NO
YES
S104
再帰代替関数置換
S105
自己再帰代替関数置換
リターン

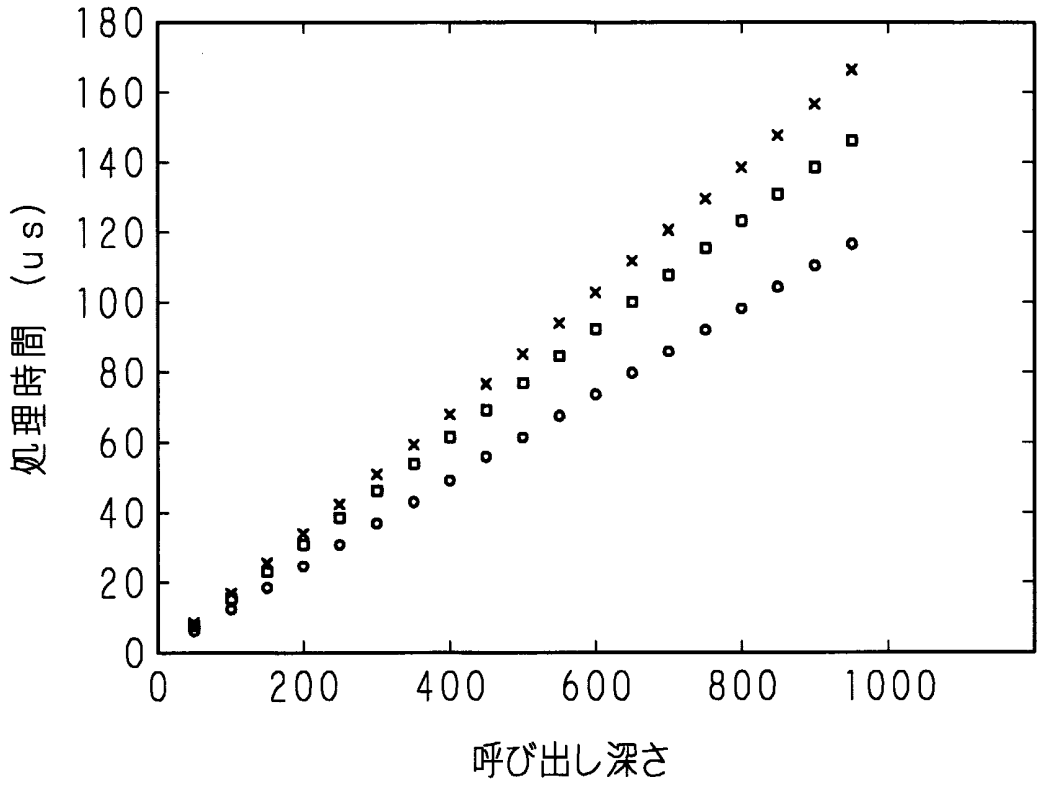
第 4 図


スタート
S201
代替関数？
1
2
3
A
S202
関数記録領域積上
S203
再帰代替関数呼出
S204
再帰代替関数実行
S205
関数記録領域形式変更
S206
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
S207
被呼出関数呼出
S208
被呼出関数実行
S209
関数記録領域破棄
S210
関数記録領域積上
S211
自己再帰代替関数呼出
S212
自己再帰代替関数実行
S213
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
S214
被呼出関数呼出
S215
被呼出関数実行
S216
関数記録領域破棄
リターン

第 5 図


A
S217
関数記録領域積上
S218
呼出関数呼出
S219
呼出関数実行
S220
新たな関数記録領域積上
S221
被呼出関数呼出
S222
被呼出関数実行
S223
関数記録領域破棄
S224
関数記録領域破棄
リターン

第 6 図


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

第 7 図

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP02/02888

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2002
Kokai Jitsuyo Shinan Koho	1971-2001	Toroku Jitsuyo Shinan Koho	1994-2002

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
CSDB(NIHONKOKU TOKKYOCHO) (in Japanese)
JOIS(JST FILE)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JP 60-8944 A (Fujitsu Ltd.), 17 January, 1985 (17.01.85), Page 3, upper left column, lines 10 to 19 (Family: none)	1,3,5,7, 9,11
Y	The same as the above	2,4,6,8, 10,12
Y	MAEDA, SOWA, "Asai Sokubaku ni yoru Doteki Scope Hensu ga Sonzai suru Tokino Matsubi Saiki Yobidashi", Transactions of Information Processing Society of Japan 15 June, 2000 (15.06.00), Vol.41, No.SIG4 (PRO07), pages 1 to 10	1,3,5,7, 9,11
A	The same as the above	2,4,6,8, 10,12

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 April, 2002 (17.04.02)	30 April, 2002 (30.04.02)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP02/02888

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	KOMIYA, YUASA, "Museigen no Jumyo o Motsu Tan'itsu Yobidashi Keizoku", Transactions of Information Processing Society of Japan, 15 January, 1996 (15.01.96), Vol.37, No.1, pages 92 to 100	1,3,5,7, 9,11
A	The same as the above	2,4,6,8, 10,12
Y	WATANABE, ITO, "Call/cc o mochiita Kurikaeshi teki Scheme Program to CPS Henkan", Information Processing Society of Japan, 16 September, 1994 (16.09.94), Vol.94, No.79 (94-SYM-75), pages 7 to 14	1,3,5,7, 9,11
A	The same as the above	2,4,6,8, 10,12
A	ISHIZAKI et al., "Java Just-In-Time Compiler ni okeru Saitekika to sono Hyoka", The Institute of Electronics, Information and Communication Engineers Gijutsu Kenkyu Hokoku, 05 August, 1999 (05.08.99), Vol.99, No.252 (CPSY99-64), pages 17 to 24	1-12
A	MATSUOKA, "Open JIT- Jiko Han'ei Keisan ni Motoduita Doteki ni Henko Kano na Java JIT Compiler", Computer Today, 01 November, 1998 (01.11.98), Vol.15, No.6, pages 2 to 11	1-12

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ０２／０２８８８

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Ｉｎｔ．Ｃｌ⁷　Ｇ０６Ｆ９／４４

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Ｉｎｔ．Ｃｌ⁷　Ｇ０６Ｆ９／４４

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２００１年
日本国実用新案登録公報	１９９６－２００２年
日本国登録実用新案公報	１９９４－２００２年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

ＣＳＤＢ（日本国特許庁）

ＪＯＩＳ（ＪＳＴファイル）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ｘ	ＪＰ　６０－８９４４　Ａ（富士通株式会社） １９８５．０１．１７，第３頁左上欄第１０－１９行 （ファミリーなし）	１，３，５，７，９，１１
Ｙ	同上	２，４，６，８，１０，１２

☒ Ｃ欄の続きにも文献が列挙されている。　　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー

「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの

「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの

「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）

「Ｏ」口頭による開示、使用、展示等に言及する文献

「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献

「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「＆」同一パテントファミリー文献

国際調査を完了した日 １７．０４．０２	国際調査報告の発送日　30.04.02
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員） 林　毅　印　５Ｂ　９１９３ 電話番号　０３－３５８１－１１０１　内線　３５４６

様式ＰＣＴ／ＩＳＡ／２１０（第２ページ）（１９９８年７月）

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ｙ	前田・曽和，”浅い束縛による動的スコープ変数が存在する時の末尾再帰呼び出し”、情報処理学会論文誌，２０００．０６．１５，第４１巻，第ＳＩＧ４（ＰＲＯ０７）号，ｐ．１－１０	１，３，５，７，９，１１
Ａ	同上	２，４，６，８，１０，１２
Ｙ	小宮・湯淺，”無制限の寿命を持つ単一呼出継続”，情報処理学会論文誌，１９９６．０１．１５，第３７巻，第１号，ｐ．９２－１００	１，３，５，７，９，１１
Ａ	同上	２，４，６，８，１０，１２
Ｙ	渡辺・伊藤，”call/ccを用いた繰返し的Schemeプログラムと ＣＰＳ変換”，情報処理学会研究報告，１９９４．０９．１６，第９４巻，第７９号（９４－ＳＹＭ－７５），　ｐ．７－１４	１，３，５，７，９，１１
Ａ	同上	２，４，６，８，１０，１２
Ａ	石崎・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，１９９９．０８．０５，第９９巻，第２５２号（ＣＰＳＹ９９－６４），ｐ．１７－２４	１－１２
Ａ	松岡，”OpenJIT－自己反映計算に基づいた動的に変更可能な Java JIT コンパイラ”，Ｃｏｍｐｕｔｅｒ　Ｔｏｄａｙ，１９９８．１１．０１，第１５巻，第６号，ｐ．２－１１	１－１２