(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
2012 年 10 月 4 日(04.10.2012)

WIPO | PCT

(10) 国際公開番号
WO 2012/131933 A1

(51) 国際特許分類:
G06F 11/28 (2006.01)　*G06F 11/34* (2006.01)

(21) 国際出願番号: PCT/JP2011/057989

(22) 国際出願日: 2011 年 3 月 30 日(30.03.2011)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 富士通株式会社(FUJITSU LIMITED) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 Kanagawa (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 高原　浩二(TAKAHARA, Koji) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内 Kanagawa (JP).

(74) 代理人: 服部　毅巖(HATTORI, Kiyoshi); 〒1920082 東京都八王子市東町９番８号　八王子東町センタービル　服部特許事務所 Tokyo (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

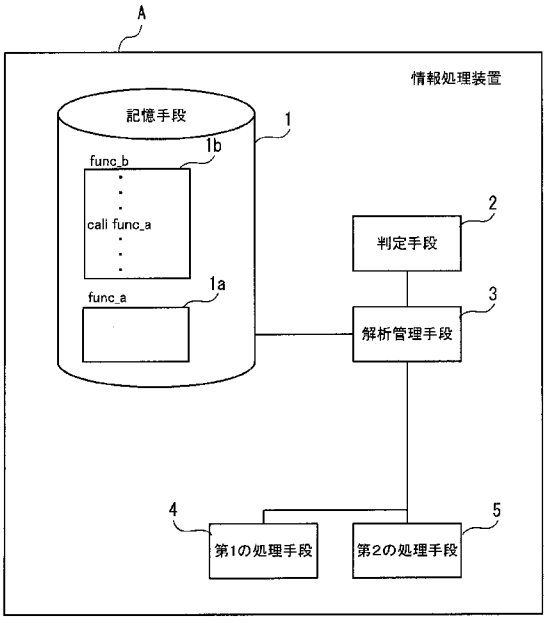
(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT,

[続葉有]

(54) **Title**: INFORMATION PROCESSING DEVICE, PROGRAM, AND ANALYSIS METHOD

(54) 発明の名称 : 情報処理装置、プログラム、および解析方法

[図1]



1 Storing means
2 Determining means
3 Analysis managing means
4 First processing means
5 Second processing means
A Information processing device

(57) **Abstract**: The present invention suppresses a delay in program invoker processing caused by the analysis of information when a program is invoked. A determining means (2) determines which technique will have a shorter response time from among: a first technique for implementing a real-time analysis of all information designated for analysis by an analysis request; and a second technique for implementing a non-real-time analysis of some information of the information designated for analysis, and implementing a real-time analysis of the remaining information. When the first technique has the shorter response time, an analysis managing means (3) causes a first processing means (4) to implement a real-time analysis of all information designated by the analysis request. When the second technique has the shorter response time, the analysis managing means (3) causes the first processing means (4) to implement a real-time analysis of information other than the information to be analyzed in non-real-time, and causes a second processing means (5) to implement a non-real-time analysis of the information to be analyzed in non-real-time.

(57) 要約:

[続葉有]

WO 2012/131933 A1

NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:

— 国際調査報告（条約第 21 条(3)）

プログラムの呼び出し時に情報解析を行うことによる、プログラムの呼び出し元の処理の遅延を抑止する。 判定手段（２）は、解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する第１の手法と、該解析要求で指定された情報のうちの一部の情報の解析を非リアルタイムに実行し、残りの情報の解析をリアルタイムに実行する第２の手法とのうち、応答時間が短い方の手法を判定する。解析管理手段（３）は、第１の手法の方が応答時間が短い場合、解析要求で指定された情報のすべての情報の解析を、第１の処理手段（４）にリアルタイムに実行させる。また解析管理手段（３）は、第２の手法の方が応答時間が短い場合、非リアルタイムに解析する情報以外の情報の解析を、第１の処理手段（４）にリアルタイムに実行させ、非リアルタイムに解析する情報の解析を、第２の処理手段（５）に非リアルタイムに実行させる。

明　細　書

発明の名称 ： 情報処理装置、プログラム、および解析方法

技術分野

[0001] 本発明は、プログラムの呼び出しに関する情報を解析する情報処理装置、プログラム、および解析方法に関する。

背景技術

[0002] プログラムのデバッグに有用な技術の１つとして、関数が呼ばれたときのタイムスタンプなどのデータを出力する技術がある。関数とは、他のプログラムから所定の関数名によって呼び出されることで実行されるプログラムである。例えば、関数が呼び出されたときのタイムスタンプを取得することで、プログラムの実行時に関数や変数がどのような頻度でアクセスされるかなど、プログラム実行時の特性に関する情報を得ることができる。プログラム実行時の特性に関する情報は、例えばメモリ配置の最適化やダウンロードの最適化などに利用できる。このように、関数が呼ばれるたびに所定の情報を解析して出力することを、その関数をトレースすると言い、この機能を「関数呼び出しのトレース機能」と呼ぶ。

[0003] 例えば、プログラムの実行時のデータ参照をトレース関数呼び出しという形で得ることを可能とする技術がある。この技術では、プログラム中のデータ参照命令をトレース関数呼び出し命令に置き換える。そしてプログラムの実行がデータ参照命令に初めて到達すると、データ参照命令の代わりにトレース関数呼び出し命令を実行する。

先行技術文献

特許文献

[0004] 特許文献1：特開２００３－１４０９２１号公報

発明の概要

発明が解決しようとする課題

[0005] しかし、従来の技術では、関数の呼び出しに伴うトレースのための情報解

析を行っている間、関数の呼び出し元の処理が応答待ちの状態となっているため、トレースのための情報解析が、呼び出し元の処理の遅延要因となっていた。例えば、関数の呼び出しに伴うトレースにおいて、詳細な解析処理や大量のコピー処理が必要になった場合、トレースのための情報解析に時間がかかり、処理の呼び出し元の処理が大幅に遅延してしまう。

[0006] なお処理の呼び出しに応じたトレースのための情報解析は、関数と呼ばれるプログラムの呼び出しの場合に限定されず、各種プログラムの呼び出し時に実行できる。関数以外のプログラムの呼び出し時にトレースのための情報解析を行う場合にも、関数の呼び出し時と同様に、プログラムの呼び出し元の処理が遅延するという問題がある。

[0007] １つの側面では、本発明は、プログラムの呼び出し時に情報解析を行うことによる、プログラムの呼び出し元の処理の遅延を抑止した情報処理装置を提供することを目的とする。

課題を解決するための手段

[0008] 上記課題を解決するために、第１のプログラムの呼び出しに関する情報の解析要求に基づき、該解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する第１の手法と、該解析要求で指定された情報のうちの一部の情報の解析を非リアルタイムに実行し、非リアルタイムに解析する情報以外情報の解析をリアルタイムに実行する第２の手法とのうち、第１のプログラムの呼び出しが行われたときの呼び出し元への応答時間が短い方の手法を判定する判定手段と、第１の手法の方が応答時間が短い場合、第１のプログラムの呼び出し命令が記述された第２のプログラムの実行過程における該呼び出し命令の実行時に、解析要求で指定された情報のすべての情報の解析、第１のプログラムの処理の実行、および呼び出しに対する応答を、第１の処理手段にリアルタイムに実行させ、第２の手法の方が応答時間が短い場合、第２のプログラムの実行過程における呼び出し命令の実行時に、解析要求で指定された情報のうちの非リアルタイムに解析する情報以外情報の解析、非リアルタイムに解析する情報の第２の処理手段への解析依頼、第１

のプログラムの処理の実行、および呼び出しに対する応答を、第１の処理手段にリアルタイムに実行させ、非リアルタイムに解析する情報の解析を、第２の処理手段に非リアルタイムに実行させる解析管理手段と、を有することを特徴とする情報処理装置が提供される。

発明の効果

[0009] プログラムの呼び出し時にトレースを実行することによる、プログラムの呼び出し元の処理の遅延を抑止することができる。

本発明の上記および他の目的、特徴および利点は本発明の例として好ましい実施の形態を表す添付の図面と関連した以下の説明により明らかになるであろう。

図面の簡単な説明

[0010] [図1]第１の実施の形態に係る装置の機能構成の一例を示す図である。

[図2]第１の実施の形態の処理手順の一例を示すフローチャートである。

[図3]第１の実施の形態による解析処理の一例を示す図である。

[図4]第２の実施の形態に用いるコンピュータのハードウェアの一構成例を示す図である。

[図5]第２の実施の形態のコンピュータの機能を示すブロック図である。

[図6]パケット情報のデータ構造の一例を示す図である。

[図7]トレース情報を取得しない場合の処理状況を示す第１の図である。

[図8]トレース情報を取得しない場合の処理状況を示す第２の図である。

[図9]第１の手法によりトレース情報を取得する場合の処理状況を示す第１の図である。

[図10]第１の手法によりトレース情報を取得する場合の処理状況を示す第２の図である。

[図11]第２の手法によりトレース情報を取得する場合の処理状況を示す第１の図である。

[図12]第２の手法によりトレース情報を取得する場合の処理状況を示す第２の図である。

[図13]トレース管理処理の手順を示すフローチャートである。

[図14]トレースコマンドと出力情報指定ファイルとのデータ形式の一例を示す図である。

[図15]トレースコマンドと出力情報指定ファイルとに応じた出力結果の第１の例を示す図である。

[図16]トレースコマンドと出力情報指定ファイルとに応じた出力結果の第２の例を示す図である。

[図17]共通域のデータ構造の一例を示す図である。

[図18]デバッグ要求構造体のデータ構造の一例を示す図である。

[図19]デバッグ結果構造体のデータ構造の一例を示す図である。

[図20]情報連鎖構造体のデータ構造の一例を示す図である。

[図21]情報連鎖構造体による連鎖関係の一例を示す図である。

[図22]処理時間情報記憶部のデータ構造の一例を示す図である。

[図23]トレース準備処理の手順を示すフローチャートである。

[図24]非リアルタイム解析情報の解析時間の算出処理の手順の一例を示すフローチャートである。

[図25]トレース管理部への情報引渡時間の算出処理の手順の一例を示すフローチャートである。

[図26]デバッグスレッドへの情報引渡時間の算出処理の手順の一例を示すフローチャートである。

[図27]時間見積もりテーブルに設定された情報の一例を示す図である。

[図28]第１のデバッグ処理の手順の一例を示すフローチャートである。

[図29]第２のデバッグ処理の手順の一例を示すフローチャートである。

[図30]デバッグスレッドへの情報引渡準備処理の手順の一例を示すフローチャートである。

[図31]デバッグスレッドの処理の手順の一例を示す図である。

発明を実施するための形態

[0011] 以下、本実施の形態について図面を参照して説明する。なお各実施の形態

は、矛盾のない範囲で複数の実施の形態を組み合わせて実施することができる。

〔第１の実施の形態〕

図１は、第１の実施の形態に係る装置の機能構成の一例を示す図である。情報処理装置Ａは、記憶手段１、判定手段２、解析管理手段３、第１の処理手段４、および第２の処理手段５を有する。

[0012] 記憶手段１は、解析対象となる第１のプログラム１ａと、第１のプログラム１ａの呼び出し命令が記述された第２のプログラム１ｂとを記憶する。

判定手段２には、第１のプログラム１ａの呼び出しに関する情報の解析手法として、第１の手法と第２の手法とが定義されている。第１の手法は、解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する手法である。第２の手法は、解析要求で指定された情報のうちの一部の情報の解析を非リアルタイムに実行し、非リアルタイムに解析する情報以外情報の解析をリアルタイムに実行する手法である。判定手段２は、第１のプログラム１ａの呼び出しに関する情報の解析要求に基づき、第１の手法と第２の手法とのうち、第１のプログラム１ａの呼び出しが行われたときの呼び出し元への応答時間が短い方の手法を判定する。

[0013] なお第１のプログラム１ａの呼び出しに関する情報には、例えば、呼び出し命令に付加された引数の情報や、引数に設定されたアドレスで指定された記憶領域に格納されている情報が含まれる。

[0014] 解析管理手段３は、第１の処理手段４と第２の処理手段５とを制御し、応答時間が短い方の手法による解析を管理する。第１の処理手段４は処理をリアルタイムに実行し、第２の処理手段５は処理を非リアルタイムに実行する。

[0015] 解析管理手段３は、第１の手法の方が応答時間が短い場合、第１のプログラム１ａの呼び出し命令が記述された第２のプログラム１ｂの実行過程における該呼び出し命令の実行時に、第１の処理手段４に処理を実行させる。例えば解析管理手段３は、解析要求で指定された情報のすべての情報の解析、

第１のプログラム１ａの処理の実行、および呼び出しに対する応答を、第１の処理手段４にリアルタイムに実行させる。

[0016] また解析管理手段３は、第２の手法の方が応答時間が短い場合、第２のプログラム１ｂの実行過程における呼び出し命令の実行時に、第１の処理手段４と第２の処理手段５とに処理を実行させる。例えば解析管理手段３は、次の処理を第１の処理手段４にリアルタイムに実行させる。第１の処理手段４に実行させる処理は、解析要求で指定された情報のうちの非リアルタイムに解析する情報以外情報の解析、非リアルタイムに解析する情報の第２の処理手段５への解析依頼、第１のプログラム１ａの処理の実行、および呼び出しに対する応答である。また解析管理手段３は、非リアルタイムに解析する情報の解析を、第２の処理手段５に非リアルタイムに実行させる。

[0017] なお図１に示した各要素間を接続する線は通信経路の一部を示すものであり、図示した通信経路以外の通信経路も設定可能である。

また第１の処理手段４と第２の処理手段５とは、例えば解析要求に応じて解析管理手段３が生成したプログラムを実行することによって実現できる。この場合、解析管理手段３は、例えば第２のプログラム１ｂにおける第１のプログラム１ａの呼び出し命令の記述を、第１の処理手段４に対応するプログラムの呼び出し命令に書き換える。さらに解析管理手段３は、第１の処理手段４に対応するプログラム内に、第１のプログラム１ａの呼び出し命令を記述することによって、間接的に第１のプログラム１ａが呼び出されるようにする。

[0018] 次に、図１に示した構成の情報処理装置Ａによる処理手順について説明する。

図２は、第１の実施の形態の処理手順の一例を示すフローチャートである。以下、図２に示す処理をステップ番号に沿って説明する。

[0019] ［ステップＳ１］判定手段２は、第１のプログラム１ａの呼び出しに関する情報の解析要求を取得すると、解析要求で指定された情報の一部を非リアルタイムで解析させることにより第１の処理手段４で行わずにすむ処理の処

理時間（短縮時間）を計算する。短縮時間は、例えば非リアルタイムで解析する情報の解析に要する時間と、解析結果を出力する時間とを加算した時間である。

[0020] ［ステップＳ２］判定手段２は、第１のプログラム１ａの呼び出しに関する情報の解析要求を取得すると、解析要求で指定された情報の一部を非リアルタイムで解析させることにより第１の処理手段４で新たに発生する処理の処理時間（増加時間）を計算する。増加時間は、例えば、第１の処理手段４から第２の処理手段５への非リアルタイムに解析する情報の解析の依頼に要する時間である。

[0021] ［ステップＳ３］判定手段２は、短縮時間と増加時間とを比較し、第１の手法と第２の手法とのうち、呼び出し元への応答時間が短い手法を判定する。例えば判定手段２は、第１の処理手段４の処理の短縮時間の方が短ければ、第１の手法の方が応答時間が短いと判断する。また判定手段２は、第１の処理手段４の処理の短縮時間の方が長ければ、第２の手法の方が応答時間が短いと判断する。応答時間が短くなる手法が第１の手法であれば、判定手段２は、処理をステップＳ４に進める。また応答時間が短くなる手法が第２の手法であれば、判定手段２は、処理をステップＳ５に進める。

[0022] ［ステップＳ４］解析管理手段３は、応答時間が短くなると判定された第１の手法により処理を実行する第１の処理手段４の処理内容を記述したプログラムを生成する。例えば、解析管理手段３は、予め用意された第１の手法用の雛形のプログラムに、第１のプログラム１ａの呼び出し命令を追加することで、第１の処理手段４の処理内容を記述したプログラムを生成する。解析管理手段３は、その後、処理をステップＳ６に進める。

[0023] ［ステップＳ５］解析管理手段３は、応答時間が短くなると判定された第２の手法により処理を実行する第１の処理手段４の処理内容を記述したプログラムを生成する。例えば、解析管理手段３は、予め用意された第２の手法用の雛形のプログラムに、第１のプログラム１ａの呼び出し命令を追加することで、第１の処理手段４の処理内容を記述したプログラムを生成する。解

析管理手段３は、その後、処理をステップＳ６に進める。

[0024] ［ステップＳ６］解析管理手段３は、第２のプログラム１ｂを編集し、第２のプログラム１ｂの実行過程における第１のプログラム１ａの呼び出し命令の実行時に、第１の処理手段４に対応するプログラムが呼び出されるようにする。例えば解析管理手段３は、第２のプログラム１ｂにおける第１のプログラム１ａの呼び出し命令を、第１の処理手段４に対応するプログラムの呼び出し命令に書き換える。第２のプログラムが既に主記憶装置に読み込まれている場合、命令の書き換えは、例えば主記憶装置に格納された第２のプログラムに対して行われる。また第２のプログラムが主記憶装置に読み込まれていなければ、命令の書き換えは、例えば二次記憶装置に格納された第２のプログラムに対して行われる。

[0025] ＯＳ（Operating System）により第２のプログラム１ｂが実行され、第１のプログラム１ａの実行タイミングになると、第１の処理手段４に対応するプログラムの呼び出し命令が実行される。するとＯＳにより第１の処理手段４が実行される。第１の手法で解析を行う場合、第１の処理手段４が、解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する。第２の手法で解析を行う場合、第１の処理手段４が、非リアルタイムで解析する情報以外情報の解析をリアルタイムに実行し、第２の処理手段５が、非リアルタイムで解析する情報の解析を非リアルタイムに実行する。

[0026] ［ステップＳ７］解析管理手段３は、第１の処理手段４または第２の処理手段５から解析結果を取得する。

［ステップＳ８］解析管理手段３は、解析結果を出力する。

[0027] このようにして、応答時間が短くなる手法により、第１のプログラム１ａの呼び出しに関する情報の解析が行われる。

図３は、第１の実施の形態による解析処理の一例を示す図である。図３には、判定手段２による判定結果に応じて、解析管理手段３の管理下で行われる解析処理を模式的に示している。

[0028] 第１の手法の方が応答時間が短くなると判断された場合、第２のプログラ

ム１ｂを実行する第３の処理手段６が生成され、第２のプログラム１ｂが実行される。第３の処理手段６は、第１の処理手段４に対応するプログラムの呼び出し命令「call func__dbg1」の実行タイミングになると、そのプログラムの呼び出しを行い、応答待ちの状態となる。するとＯＳにより第１の処理手段４が起動され、解析要求で指定されたすべての情報の解析、解析結果の出力が行われる。その後、第１の処理手段４から、第１のプログラム１ａの呼び出し命令「call func__a」が実行される。するとＯＳにより、第１のプログラム１ａを実行する第４の処理手段７が起動され、第１のプログラム１ａが実行される。第４の処理手段７は、処理が完了すると、処理結果を第１の処理手段４に応答する。第１の処理手段４は、第４の処理手段７の処理結果を第３の処理手段６に応答する。すると、第３の処理手段６により、第１の処理手段４に対応するプログラムの呼び出し命令「call func__dbg1」以降の処理が開始される。

[0029] 第２の手法の方が応答時間が短くなると判断された場合、第２のプログラム１ｂを実行する第３の処理手段６が生成され、第２のプログラム１ｂが実行される。第３の処理手段６は、第１の処理手段４に対応するプログラムの呼び出し命令「call func__dbg2」の実行タイミングになると、そのプログラムの呼び出しを行い、応答待ちの状態となる。するとＯＳにより第１の処理手段４が起動され、解析要求で指定されたすべての情報のうちのリアルタイムに解析する情報の解析、第２の処理手段５への非リアルタイムに解析する情報の解析依頼が行われる。その後、第１の処理手段４から、第１のプログラム１ａの呼び出し命令「call func__a」が実行される。するとＯＳにより、第１のプログラム１ａを実行する第４の処理手段７が起動され、第１のプログラム１ａが実行される。第４の処理手段７は、処理が完了すると、処理結果を第１の処理手段４に応答する。第１の処理手段４は、第４の処理手段７の処理結果を第３の処理手段６に応答する。すると、第３の処理手段６により、第１の処理手段４に対応するプログラムの呼び出し命令「call func__dbg2」以降の処理が開始される。

[0030] 一方、第２の処理手段５では、非リアルタイムに解析する情報の解析を行う。そして、第２の処理手段５は、解析要求で指定されたすべての情報の解析結果をまとめて出力する。

[0031] このように、応答時間が短くなる手法を選択して、プログラムの呼び出しに関する情報の解析を行うようにしたため、第３の処理手段６が応答待ちをする時間が抑止される。その結果、第３の処理手段６による処理に大きな影響を与えずに、解析が可能となる。

[0032] 例えば、システムの運用中に、プログラムの呼び出しに関する情報を解析する場合もある。このような場合、第１の実施の形態によって解析を行えば、運用中のシステムの処理性能の低下を最小限に抑えることができる。

[0033] また第３の処理手段６の応答待ちの時間が長期化すると、正しい解析を行うことができない場合がある。例えば呼び出し元である第２のプログラム１ｂにおいて、第１のプログラム１ａの呼び出し命令実行時の応答時間が所定時間を過ぎると、エラー処理を実行するように記述されている場合がある。このような場合、プログラムの呼び出しに関する情報の解析を実行したことでエラー処理が開始されてしまうと、平常時の動作を正しく解析できなくなってしまう。第１の実施の形態によって解析を行えば、平常時の動作を正しく解析できる可能性が高くなる。

[0034] なお、図１または図３に示した判定手段２、解析管理手段３、第１の処理手段４、第２の処理手段５、第３の処理手段６、第４の処理手段７は、情報処理装置Ａが有するＣＰＵ（Central Processing Unit）により実現することができる。また、記憶手段１は、情報処理装置Ａが有するＲＡＭ（Random Access Memory）やハードディスクドライブ（ＨＤＤ:Hard Disk Drive）などにより実現することができる。

[0035] 〔第２の実施の形態〕

次に第２の実施の形態について説明する。第２の実施の形態は、第１の実施の形態に示した機能に加え、出力する情報の条件を設定し、必要な情報のみを出力することができるようにしたものである。

[0036] 図４は、第２の実施の形態に用いるコンピュータのハードウェアの一構成例を示す図である。ＯＳ１２０は、ＣＰＵ１０１によって装置全体が制御されている。ＣＰＵ１０１には、バス１０８を介してＲＡＭ１０２と複数の周辺機器が接続されている。

[0037] ＲＡＭ１０２は、コンピュータ１００の主記憶装置として使用される。ＲＡＭ１０２には、ＣＰＵ１０１に実行させるＯＳのプログラムやアプリケーションプログラムの少なくとも一部が一時的に格納される。また、ＲＡＭ１０２には、ＣＰＵ１０１による処理に必要な各種データが格納される。

[0038] バス１０８に接続されている周辺機器としては、ＨＤＤ１０３、グラフィック処理装置１０４、入力インタフェース１０５、光学ドライブ装置１０６、および通信インタフェース１０７がある。

[0039] ＨＤＤ１０３は、内蔵したディスクに対して、磁気的にデータの書き込みおよび読み出しを行う。ＨＤＤ１０３は、コンピュータ１００の二次記憶装置として使用される。ＨＤＤ１０３には、ＯＳのプログラム、アプリケーションプログラム、および各種データが格納される。なお、二次記憶装置としては、フラッシュメモリなどの半導体記憶装置を使用することもできる。

[0040] グラフィック処理装置１０４には、モニタ１１が接続されている。グラフィック処理装置１０４は、ＣＰＵ１０１からの命令に従って、画像をモニタ１１の画面に表示させる。モニタ１１としては、ＣＲＴ（Cathode Ray Tube）を用いた表示装置や液晶表示装置などがある。

[0041] 入力インタフェース１０５には、キーボード１２とマウス１３とが接続されている。入力インタフェース１０５は、キーボード１２やマウス１３から送られてくる信号をＣＰＵ１０１に送信する。なお、マウス１３は、ポインティングデバイスの一例であり、他のポインティングデバイスを使用することもできる。他のポインティングデバイスとしては、タッチパネル、タブレット、タッチパッド、トラックボールなどがある。

[0042] 光学ドライブ装置１０６は、レーザ光などを利用して、光ディスク１４に記録されたデータの読み取りを行う。光ディスク１４は、光の反射によって

読み取り可能なようにデータが記録された可搬型の記録媒体である。光ディスク１４には、ＤＶＤ（Digital Versatile Disc）、ＤＶＤ－ＲＡＭ、ＣＤ－ＲＯＭ（Compact Disc Read Only Memory）、ＣＤ－Ｒ（Recordable）／ＲＷ（ReWritable）などがある。

[0043] 通信インタフェース１０７は、ネットワーク１０に接続されている。通信インタフェース１０７は、ネットワーク１０を介して、他のコンピュータまたは通信機器との間でデータの送受信を行う。

[0044] 以上のようなハードウェア構成によって、本実施の形態の処理機能を実現することができる。なお、図１に示した第１の実施の形態の情報処理装置も、図４に示したコンピュータと同様のハードウェアにより実現することができる。

[0045] 図５は、第２の実施の形態のコンピュータの機能を示すブロック図である。ＯＳ１２０は、プログラム記憶部１１０、ＯＳ１２０、およびトレース管理部１３０を有する。

プログラム記憶部１１０は、デバッグ対象となるデバッグ対象関数１１１と、そのプログラムを呼び出す呼び出し元関数１１２とを記憶する。また、トレース管理部１３０内のデバッグ関数生成部１３４によりデバッグプログラムが生成された場合、生成されたデバッグプログラムもプログラム記憶部１１０に格納される。例えばＲＡＭ１０２またはＨＤＤ１０３の記憶領域の一部が、プログラム記憶部１１０として使用される。

[0046] ＯＳ１２０は、プログラムの実行要求に応じて、そのプログラムを実行するプロセスを生成し、生成したプロセスによってプログラムを実行する。プロセスは、ＯＳからＣＰＵ１０１、ＲＡＭ１０２、その他のＩ／Ｏ機器などのハードウェア資源の割り当てを受け、割り当てられたハードウェア資源によりプログラムを実行するものである。

[0047] なおＯＳ１２０は、仮想記憶方式により、仮想メモリ空間をカーネル空間とユーザ空間に分離している。カーネル空間は、ＯＳ１２０のカーネルやデバイスドライバの実行のためのメモリ空間であり、ユーザから直接の操作は

制限される。ユーザ空間は、アプリケーションプログラムなどの実行のためのメモリ空間である。図５の例では、トレース管理部１３０の各機能は、ユーザ空間を利用して実行される。また呼び出し元関数１１２やデバッグ対象関数１１１は、例えばデバイスドライバ用のプログラムであり、カーネル空間を利用してＯＳ１２０により実行されるものとする。

[0048] トレース管理部１３０は、トレースコマンドの実行要求に応じて、デバッグ対象関数１１１実行時の情報をトレースする。トレース管理部１３０は、トレース条件記憶部１３１、処理時間情報記憶部１３２、処理時間解析部１３３、デバッグ関数生成部１３４、呼び出し命令変更部１３５、およびトレース情報出力部１３６を有する。

[0049] トレース条件記憶部１３１は、プログラム実行時のデータをトレースする際の条件を記憶する。例えばトレース条件記憶部１３１には、トレース条件が記述された出力情報指定ファイル１３１ａが格納される。なお、トレース条件記憶部１３１としては、例えばＲＡＭ１０２またはＨＤＤ１０３の記憶領域の一部が使用される。

[0050] 処理時間情報記憶部１３２は、情報のトレースに要する時間の算出の基礎となる情報を記憶する。例えば処理時間情報記憶部１３２は、処理内容ごとの処理時間を記憶する。なお、処理時間情報記憶部１３２としては、例えばＲＡＭ１０２またはＨＤＤ１０３の記憶領域の一部が使用される。

[0051] 処理時間解析部１３３は、トレースコマンドで指定された出力情報指定ファイル１３１ａの内容に基づいて、２つの手法それぞれにおいて、トレース処理に要する時間を計算する。

[0052] 第１の手法は、リアルタイムに解析する情報（リアルタイム解析情報）と、リアルタイムに解析を行わない情報（非リアルタイム解析情報）との解析を、共にリアルタイムにデバッグ処理を実行するプロセスで行う手法である。

[0053] 第２の手法は、リアルタイム解析情報の解析についてはリアルタイムにデバッグ処理を実行するプロセスで行い、非リアルタイム解析情報の解析につ

いては、デバッグ処理を実行するプロセスとは別のデバッグスレッドで、並列的に行う手法である。

[0054] 処理時間解析部１３３は、２つの手法それぞれにおけるデバッグ関数の処理時間を比較し、処理時間が短くなる手法を、適用手法に決定する。処理時間解析部１３３は、決定した適用手法を、デバッグ関数生成部１３４に通知する。

[0055] デバッグ関数生成部１３４は、適用手法を実行するデバッグプログラムを生成する。例えばデバッグ関数生成部１３４は、第１の手法と第２の手法とのそれぞれ用に予め用意された雛形のプログラムを用いて、デバッグプログラムを生成することができる。雛形のプログラムを用いる場合、デバッグ関数生成部１３４は、雛形のプログラムに対して、リアルタイムに解析するトレース処理に関する記述の後に、デバッグ対象関数１１１の呼び出し命令を挿入する。そしてデバッグ関数生成部１３４は、生成したデバッグプログラムをプログラム記憶部１１０に格納する。またデバッグ関数生成部１３４は、デバッグ処理におけるデータのトレースに第２の手法を適用する場合、ＯＳ１２０に対してデバッグスレッドの生成を要求する。

[0056] 呼び出し命令変更部１３５は、呼び出し元関数１１２内のデバッグ対象関数１１１の呼び出し命令を、デバッグ関数生成部１３４が生成したデバッグ関数の呼び出し命令に置き換える。例えば、呼び出し命令変更部１３５は、呼び出し元関数１１２をカーネル空間にロードした後、カーネル空間内で関数の呼び出し命令の置き換えを行うことができる。ここでロードとは、ＨＤＤ１０３などの二次記憶装置に格納されたプログラムを、ＲＡＭ１０２などの主記憶装置にコピーすることである。

[0057] トレース情報出力部１３６は、デバッグ関数を実行中のＯＳ１２０、またはＯＳ１２０がカーネル空間内に生成したデバッグスレッドからトレース情報を取得する。そしてトレース情報出力部１３６は、取得したトレース情報を出力する。例えばトレース情報出力部１３６は、所定のファイルに、解析結果であるトレース情報を出力する。

[0058] このようなコンピュータ１００において、ユーザがトレースコマンドを入力することで、呼び出し元関数１１２の実行時にデバッグ対象関数１１１の呼び出しが行われると、その呼び出し処理に関するトレース情報を取得することができる。

[0059] なお、図５に示した各要素間を接続する線は通信経路の一部を示すものであり、図示した通信経路以外の通信経路も設定可能である。

また、図５に示すプログラム記憶部１１０は、図１に示した第１の実施の形態の記憶手段１の一例である。図５に示す処理時間解析部１３３は、図１に示した第１の実施の形態の判定手段２の一例である。図５に示すデバッグ関数生成部１３４、呼び出し命令変更部１３５、およびトレース情報出力部１３６は、図１に示した第１の実施の形態の解析管理手段３の一例である。

[0060] 次に、リアルタイム解析情報と非タイム解析情報とについて説明する。

リアルタイム解析情報としては、例えばタイムスタンプやスタックがある。

タイムスタンプは、関数呼び出し命令が実行された時刻の情報であり、関数呼び出し命令に実行時にリアルタイムに取得することで、正確な時刻が取得でききる。

[0061] スタックは、その関数がどの関数から呼び出されていたのかを示す。関数Ａが関数Ｂから呼ばれ、関数Ｂは関数Ｃから呼ばれ、関数Ｃは関数Ｄから呼ばれていた場合、関数Ａのスタックは、関数Ａ←関数Ｂ←関数Ｃ←関数Ｄとなる。

[0062] 非リアルタイム解析情報としては、例えばネットワークドライバで受け渡しされるパケット情報がある。パケット情報は、例えば関数呼び出し命令の引数に設定されたアドレスで示される記憶領域に格納されている。

[0063] 図６は、パケット情報のデータ構造の一例を示す図である。ネットワークドライバで受け渡しされるパケット情報は、メッセージブロック構造体２１～２３が連鎖され、各メッセージブロック構造体２１～２３がバッファ２４～２６の先頭アドレス／末尾アドレスを持つデータ構造となっている。メッ

セージブロック構造体２１～２３は、各メッセージブロック構造体２１～２３に設定された、次のメッセージブロック構造体の位置を示すポインタによって、連鎖関係が示される。

[0064] ネットワークドライバの送信関数や受信関数で受け渡されるのは、先頭のメッセージブロック構造体２１の先頭アドレスである。なお、メッセージブロック構造体の個数は、何個でもよい（１個でもよいし１０個でもよい）。

[0065] パケット情報２７は、複数のモジュールで使用されることが前提となっているため、メッセージブロック構造体２１～２３の持つ参照カウンタが、現在いくつのモジュールから参照されているのかを示している。先頭のメッセージブロック構造体２１の持つ参照カウンタの値が「０」にならない限り、連鎖関係にあるメッセージブロック構造体２１～２３およびバッファ２４～２６のメモリが解放されてしまうことはない。

[0066] そのため、ネットワークドライバで受け渡しされるパケット情報２７については、メッセージブロック構造体２１の持つ参照カウンタに１を加えておけば、そのパケット情報２７が格納されたメモリ領域が開放されるのを抑止できる。

[0067] パケット情報２７のように、格納されたメモリ領域の開放を容易に抑止できる情報については、リアルタイムに解析しなくてもよい。つまり、リアルタイムに解析しなくても、情報が格納された領域の解放を抑止しておけば、非リアルタイムにその情報を解析することができる。

[0068] このように、デバッグ時にトレースする情報には、非リアルタイムに解析可能な情報（非リアルタイム解析情報）が含まれることがある。そこで、非リアルタイム解析情報の解析に要する時間が長い場合、デバッグ処理を実行するプロセスとは別個のデバッグスレッドに非リアルタイム解析情報の解析を実行させることで、デバッグ処理を実行することによる応答遅延を低減できる。ただし、非リアルタイム解析情報の解析をデバッグスレッドに依頼するのにも、ある程度の時間がかかる。そのため、デバッグスレッドへの処理の依頼に要する時間よりも、非リアルタイム解析情報の解析をリアルタイム

に実行しないことにより短縮される処理時間の方が短い場合もあり得る。このような場合、デバッグ処理を実行する過程で、非リアルタイム解析情報の解析も含めて、リアルタイムに解析処理を行う方が効率的である。第２の実施の形態では、トレース情報を取得する手法として、第１の手法と第２の手法とのうち、呼び出し元への応答時間が短くなる方の手法により、トレース情報を取得する。

[0069] 以下、図７～図１２を参照し、トレース情報を取得しない場合、第１の手法でトレース情報を取得する場合、および第２の手法でトレース情報を取得する場合のそれぞれについて、呼び出し元関数１１２の実行状況の概要を説明する。

[0070] 図７、図８は、トレース情報を取得しない場合の例を示している。図７、図８の例では、トレースコマンドの実行要求がコンピュータ１００に入力されず、トレース管理部１３０は起動されない。

[0071] 図７は、トレース情報を取得しない場合の処理状況を示す第１の図である。ＯＳ１２０の起動時に、呼び出し元関数１１２とデバッグ対象関数１１１が、メモリのカーネル空間にロードされる。これにより、カーネル空間内に呼び出し元関数３１とデバッグ対象関数３２とが記憶される。またＯＳ１２０の内部では、呼び出し元関数３１の関数名「ｆｕｎｃ＿ｂ」により呼び出し元関数３１を呼び出すコードが生成される。ＯＳ１２０は、各種処理の実行過程で、呼び出し元関数３１の実行タイミングになると、生成されたコードに基づいて呼び出し元関数３１の実行要求（関数呼び出し）を行う。

[0072] なお、ＯＳ１２０の起動後の任意のタイミングで呼び出し元関数１１２とデバッグ対象関数１１１とを、メモリのカーネル空間にロードすることもできる。

カーネル空間にロードされた関数などのプログラムは、ＯＳ１２０の一機能として実行される。そこで、以下の説明では、カーネル空間内でロードしたプログラムは、ＯＳ１２０が実行するものとする。

[0073] 図８は、トレース情報を取得しない場合の処理状況を示す第２の図である

。この状況で、ＯＳ１２０の処理中で呼び出し元関数３１の実行要求（関数呼び出し）が発生したときに、ＯＳ１２０で呼び出し元関数３１が実行される。

[0074] 呼び出し元関数３１の処理が実行され、デバッグ対象関数３２の関数呼び出し命令（ｃａｌｌ　ｆｕｎｃ＿ａ）の実行タイミングになると、デバッグ対象関数３２が呼び出される。するとデバッグ対象関数３２が実行される。

[0075] デバッグ対象関数３２の処理が終了すると、デバッグ対象関数３２の処理結果が、戻り値として呼び出し元関数３１に通知される。そしてデバッグ対象関数３２の処理が終了する。デバッグ対象関数３２から戻り値に基づいて、呼び出し元関数３１における、デバッグ対象関数３２の呼び出し命令以降の命令から処理が続行される。

[0076] このように、トレース情報を取得しない場合、呼び出し元関数３１の実行時に、デバッグ対象関数３２の関数呼び出し行われる。一方、トレース情報を取得する場合、呼び出し元関数３１の実行時には、デバッグ関数が呼び出されるように、呼び出し元関数３１の内容が書き換えられる。

[0077] 図９、図１０は、第１の手法によりトレース情報を取得する場合の例を示している。第１の手法によりトレース情報を取得する場合、ＯＳ１２０起動時の処理は、トレース情報を取得しない場合と同様である。すなわち、図７に示すように、ＯＳ１２０の起動時に、呼び出し元関数１１２とデバッグ対象関数１１１とが、カーネル空間にロードされる。その後、トレースコマンドの実行指示が入力されることで、トレース処理部１３０の処理時間解析部１３３によって、応答時間が短くなるトレース情報の取得手法が判定される。ここで、処理時間解析部１３３により、第１の手法の方が応答時間が短くなると判定されたものとする。

[0078] 図９は、第１の手法によりトレース情報を取得する場合の処理状況を示す第１の図である。第１の手法でトレース情報を取得する場合、デバッグ関数生成部１３４により、第１の手法でトレース情報を取得するための処理が記述された第１のデバッグ関数３３が生成される。生成された第１のデバッグ

関数３３は、カーネル空間にロードされる。

[0079] また第１の手法でトレース情報を取得する場合、呼び出し命令変更部１３５により、呼び出し元関数３１内の関数呼び出し命令の内容が書き換えられる。例えば呼び出し命令変更部１３５は、カーネル空間内にロードされた呼び出し元関数３１内のデバッグ対象関数３２の呼び出し命令（ｃａｌｌ　ｆｕｎｃ＿ａ）を、第１のデバッグ関数３３の呼び出し命令（ｃａｌｌ　ｆｕｎｃ＿ｄｂｇ１）に書き換える。

[0080] 図１０は、第１の手法によりトレース情報を取得する場合の処理状況を示す第２の図である。その後、ＯＳ１２０から呼び出し元関数の実行要求が出されると、コンピュータ１００で呼び出し元関数３１の処理が実行される。

[0081] 呼び出し元関数３１の処理が実行され、呼び出し元関数３１内の第１のデバッグ関数３３の関数呼び出し命令の実行タイミングになると、第１のデバッグ関数３３が呼び出され、コンピュータ１００で実行される。第１のデバッグ関数３３が実行されている間は、呼び出し元関数３１の処理状態は、戻り値の待ち状態となっている。

[0082] ＯＳ１２０は、第１のデバッグ関数３３に記述された命令に従って、処理を実行する。例えば最初にsave命令が実行される。save命令は、ＣＰＵ１０１内のレジスタに格納されたデータのＲＡＭ１０２への退避を一括して行う命令である。save命令を、例えば関数呼び出しが行われたときに先頭で実行することで、呼び出し元関数３１の処理によりレジスタに設定されたデータを、呼び出し先の第１のデバッグ関数３３の処理により書き換えてしまうことを抑止できる。

[0083] またＯＳ１２０は、第１のデバッグ関数３３に記述された命令に従い、リアルタイム解析情報の解析を行う。また第１のデバッグ関数３３の処理では、解析対象のデータが、第１のデバッグ関数３３の実行要求時に指定された限定条件に合致する場合には、非リアルタイム解析情報が解析される。解析結果は、トレース管理部１３０内のトレース情報出力部１３６に引き渡される。限定条件とは、デバッグ対象関数３２の関数呼び出しに関連する情報を

、トレース処理の対象とするための条件である。関数呼び出しに関連する情報には、例えば関数呼び出し命令の引数や、その引数に設定されたアドレスで示される記憶領域内の情報が含まれる。第１のデバッグ関数３３の処理では、限定条件が満たされていなければ、非リアルタイム解析情報の解析や解析結果の出力は行われない。

[0084] また、トレース情報出力部１３６に引き渡される解析結果には、解析対象として指定された情報が含まれる。例えば第１のデバッグ関数３３の処理では、解析結果が、トレース管理部１３０が管理している記憶領域にコピーされる。これにより、解析結果の引き渡しが行われる。

[0085] 解析結果を取得したトレース管理部１３０のトレース情報出力部１３６は、デバッグスレッドから通知された解析結果を、例えば解析結果ファイル３９に書き出す。解析結果ファイル３９は、例えばＨＤＤ１０３に格納される。

[0086] また第１のデバッグ関数３３の処理では、トレース処理終了後、restore命令が実行される。restore命令は、save命令によりレジスタの復元を一括して行う命令である。その後、第１のデバッグ関数３３の処理では、デバッグ対象関数３２の実行要求（関数呼び出し）が行われる。するとコンピュータ１００により、デバッグ対象関数３２が実行される。デバッグ対象関数３２を実行されている間は、第１のデバッグ関数３３の処理状態は、戻り値の待ち状態となっている。

[0087] ＯＳ１２０は、デバッグ対象関数３２の実行が終了すると、デバッグ対象関数３２の処理結果を第１のデバッグ関数３３への戻り値とする。デバッグ対象関数３２から戻り値が得られると、ＯＳ１２０は第１のデバッグ関数３３の処理に復帰し、デバッグ対象関数３２から取得したその戻り値を、呼び出し元関数３１への戻り値とする。そして第１のデバッグ関数３３の処理が終了する。

[0088] ＯＳ１２０は、第１のデバッグ関数３３からの戻り値に基づき、呼び出し元関数３１における関数呼び出し命令の次の命令から処理を続行する。

なお図９、図１０に示した呼び出し元関数３１に基づくＯＳ１２０の処理機能は、図３に示した第１の実施の形態の第３の処理手段６の一例である。また図９、図１０に示した第１のデバッグ関数３３に基づくＯＳ１２０の処理機能は、図３に示した第１の実施の形態の第１の手法適用時の第１の処理手段４の一例である。さらに図９、図１０に示したデバッグ対象関数３２に基づくＯＳ１２０の処理機能は、図３に示した第１の実施の形態の第４の処理手段７の一例である。

[0089] 図１１、図１２は、第２の手法によりトレース情報を取得する場合の処理状況を示す図である。第２の手法でトレース情報を取得する場合、ＯＳ１２０起動時の処理は、トレース情報を取得しない場合と同様である。すなわち、図７に示すように、ＯＳ１２０の起動時に、呼び出し元関数１１２とデバッグ対象関数１１１とが、カーネル空間にロードされる。その後、トレースコマンドの実行指示が入力されることで、トレース処理部１３０の処理時間解析部１３３によって、応答時間が短くなるトレース情報の取得手法が判定される。ここで、処理時間解析部１３３により、第２の手法の方が応答時間が短くなると判定されたものとする。

[0090] 図１１は、第２の手法によりトレース情報を取得する場合の処理状況を示す第１の図である。第２の手法でトレース情報を取得する場合、デバッグ関数生成部１３４により、第２の手法でトレース情報を取得するための処理が記述された第２のデバッグ関数３４が生成される。生成された第２のデバッグ関数３４は、カーネル空間にロードされる。またデバッグ関数生成部１３４は、デバッグスレッド３５のプログラムをカーネル空間内にロードする。この時点では、デバッグスレッド３５はスリープ状態である。デバッグスレッド３５は、呼び出し元関数３１、デバッグ対象関数３２、および第２のデバッグ関数３４の処理と並列的に処理を実行可能な、処理機能である。

[0091] また第２の手法でトレース情報を取得する場合、呼び出し命令変更部１３５により、呼び出し元関数３１内の関数呼び出し命令の内容が書き換えられる。例えば呼び出し命令変更部１３５は、カーネル空間内にロードされた呼

び出し元関数３１内のデバッグ対象関数３２の呼び出し命令（ｃａｌｌ　ｆｕｎｃ＿ａ）を、第２のデバッグ関数３４の呼び出し命令（ｃａｌｌ　ｆｕｎｃ＿ｄｂｇ２）に書き換える。

[0092] 図１２は、第２の手法によりトレース情報を取得する場合の処理状況を示す第２の図である。その後、ＯＳ１２０から呼び出し元関数の実行要求が出されると、コンピュータ１００で呼び出し元関数３１が実行される。

[0093] 呼び出し元関数３１の処理が実行され、呼び出し元関数３１内の第２のデバッグ関数３４の関数呼び出し命令の実行タイミングになると、第２のデバッグ関数３４が呼び出され、コンピュータ１００で実行される。第２のデバッグ関数３４が実行されている間は、呼び出し元関数３１の処理状態は、戻り値の待ち状態となっている。

[0094] ＯＳ１２０は、第２のデバッグ関数３４に記述された命令に従って、処理を実行する。例えば最初にsave命令が実行される。またＯＳ１２０は、第２のデバッグ関数３４に記述された命令に従い、リアルタイム解析情報の解析を行う。

[0095] また第２のデバッグ関数３４の処理では、デバッグスレッド３５への情報の引渡準備を行った後、デバッグスレッド３５のウェイクアップ指示が実行される。スリープ（sleep）状態のデバッグスレッド３５が動作状態に移行される。その後、第２のデバッグ関数３４の処理では、restore命令が実行される。さらに第２のデバッグ関数３４の処理では、デバッグ対象関数３２の実行要求が行われる。するとコンピュータ１００により、デバッグ対象関数３２を実行される。デバッグ対象関数３２を実行されている間は、第２のデバッグ関数３４の処理状態は、戻り値の待ち状態となっている。

[0096] ＯＳ１２０は、デバッグ対象関数３２の実行が終了すると、デバッグ対象関数３２の処理結果を第２のデバッグ関数３４への戻り値とする。デバッグ対象関数３２から戻り値が得られると、ＯＳ１２０は第２のデバッグ関数３４の処理に復帰し、デバッグ対象関数３２から取得したその戻り値を、呼び出し元関数３１への戻り値とする。そして第２のデバッグ関数３４の処理が

終了する。

[0097] ＯＳ１２０は、第２のデバッグ関数３４からの戻り値に基づき、呼び出し元関数３１における関数呼び出し命令の次の命令から処理を続行する。

一方、デバッグスレッド３５は、解析対象が、第２のデバッグ関数３４の実行要求時に指定された限定条件に合致する場合には、非リアルタイム解析情報を解析し、解析結果をトレース管理部１３０内のトレース情報出力部１３６に引き渡す。例えばデバッグスレッド３５は、解析結果を、トレース管理部１３０が管理する記憶領域内にコピーする。なおデバッグスレッド３５は、限定条件が満たされていなければ、非リアルタイム解析情報の解析や解析結果の出力を行わずに、スリープ状態に移行する。

[0098] 解析結果を取得したトレース情報出力部１３６は、デバッグスレッド３５から通知された解析結果を、例えば解析結果ファイル３９に書き出す。解析結果ファイル３９は、例えばＨＤＤ１０３に格納される。

[0099] このように、第２の手法では、非リアルタイム解析情報の解析は、第２のデバッグ関数３４の処理と並列で実行可能なデバッグスレッド３５で実行される。そのため、第２のデバッグ関数３４を実行する第２のデバッグ関数３４は、非リアルタイム解析情報の解析を待たずに、処理を進めることができる。

[0100] なお図１１、図１２に示した第２のデバッグ関数３４に基づくＯＳ１２０の処理機能は、図３に示した第１の実施の形態の第２の手法適用時の第１の処理手段４の一例である。また図１１、図１２に示したデバッグスレッド３５を用いたＯＳ１２０の処理機能は、図３に示した第１の実施の形態の第２の処理手段５の一例である。

[0101] 次に、トレース管理部１３０が実行するトレース管理処理を詳細に説明する。

図１３は、トレース管理処理の手順を示すフローチャートである。以下、図１３に示す処理をステップ番号に沿って説明する。

[0102] ［ステップＳ１１］ＯＳ１２０は、トレースコマンドの入力を受け付ける

。ＯＳ１２０は、トレースコマンドが入力されると、トレース管理部１３０を起動する。

［ステップＳ１２］トレース管理部１３０は、トレース準備処理を行う。トレース準備処理は、デバッグプログラムの生成や、デバッグ対象関数内の関数呼び出し命令の変更などの処理である。トレース準備処理の詳細は後述する（図２３参照）。

[0103] ［ステップＳ１３］トレース情報出力部１３６は、所定時間待機し、処理をステップＳ１４に進める。

［ステップＳ１４］トレース情報出力部１３６は、管理者からの操作などによるトレース終了指示が入力されたか否かを判断する。トレース情報出力部１３６は、トレース終了指示が入力された場合、処理をステップＳ１７に進める。またトレース情報出力部１３６は、トレース終了指示が入力されていなければ、処理をステップＳ１５に進める。

[0104] ［ステップＳ１５］トレース情報出力部１３６は、解析結果が通知されたか否かを判断する。例えばトレース情報出力部１３６は、共通域に解析結果が格納された場合に、解析結果が通知されたと判断する。トレース情報出力部１３６は、解析結果が通知された場合、処理をステップＳ１６に進める。またトレース情報出力部１３６は、解析結果が通知されていなければ、処理をステップＳ１３に進める。

[0105] ［ステップＳ１６］トレース情報出力部１３６は、通知された解析結果を、ファイルなどに出力する。その後、トレース情報出力部１３６は、処理をステップＳ１３に進める。

［ステップＳ１７］呼び出し命令変更部１３５は、トレース準備処理内で行った呼び出し元関数３１に対する関数呼び出し命令の変更を、元の呼び出し命令に戻す。

[0106] ［ステップＳ１８］デバッグ関数生成部１３４は、デバッグスレッド３５の有無を判断する。デバッグ関数生成部１３４は、デバッグスレッド３５が存在していれば、処理をステップＳ１９に進める。またデバッグ関数生成部

１３４は、デバッグスレッド３５が存在していなければ、トレース管理処理を終了する。

[0107] ［ステップＳ１９］デバッグ関数生成部１３４は、ＯＳ１２０に対してデバッグスレッド３５の消去を指示する。その後、トレース管理処理を終了する。

このような手順でトレース情報が取得される。

[0108] ここで、ステップＳ１１で入力されるトレースコマンドに基づくトレース条件の定義例について説明する。

図１４は、トレースコマンドと出力情報指定ファイルとのデータ形式の一例を示す図である。

[0109] トレースコマンド５１は、図１４に示すように「ctrace （出力情報指定ファイル）」という形式で入力される。ここで「ctrace」が、トレースコマンドのコマンド名であり、「出力情報指定ファイル」がトレースコマンド５１に設定する引数である。「出力情報指定ファイル」には、トレース処理に関する限定条件などの定義されたファイル（出力情報指定ファイル５２）の位置（ディレクトリパス）と、ファイル名とが設定される。

[0110] 出力情報指定ファイル５２には、例えば、複数の記述子が設定される。記述子５３には呼び出し元関数、デバッグ対象関数、１つ以上の限定条件、および出力情報が設定される。

[0111] 呼び出し元関数名は、デバッグの対象となるデバッグ対象関数１１１を呼び出す呼び出し元関数の名称である。なお呼び出し元関数は複数指定することもできる。例えば、「func__b1」と「func__b2」と「func__b3」の３個の呼び出し元関数を指定する場合は、「func__b1:func__b2:func__b3」のように設定する。また呼び出し元関数として、カーネル空間で実行されるすべての関数を指定することも可能である。その場合は、例えば、呼び出し元関数名として「kernel」と記述する。

[0112] 一方、第２の実施の形態では、デバッグ対象関数は１個の記述子５３に１個のみ指定するものとする。

記述子５３には、複数の限定条件を設定できる。限定条件５４には、引数番号、引数種別、および任意の数の条件が設定できる。引数番号は、例えば「argN (N=0,1,2,…)」で指定する。これは、引数番号が「Ｎ」の引数に関する条件指定であることを意味する。引数種別は、例えば「plain」、「mblk__t」などのキーワードで指定される。「plain」は、引数に設定されている情報がプレーンデータ（プレーンテキスト）の先頭アドレスであることを示す。「mblk__t」は、引数に設定されている情報がメッセージブロック構造体の先頭アドレスであることを示す。

[0113] 条件は、「Ａ＝Ｂ」や「Ａ＜Ｂ」（Ａ，Ｂは実数）といった形、あるいは、「ftp」などのキーワードで示される。例えば、「Ethertype=0x800」と条件に指定すれば、引数種別が「mblk__t」であった場合に、その引数が示すパケットの「Ethertype」が「0x800」（つまりＩＰ（Internet Protocol）のパケット）である場合だけ、情報出力を行うことを意味する。また「pattern="abc"」と条件に指定すれば、パケット中に"abc"（１６進だと 0x626364）というパターンが含まれる場合のみ情報出力することを意味する。また「LEN<101」と条件に指定すれば、パケット長（ＭＡＣ（Media Access Control）ヘッダを含む）が１０１より小さい場合だけ情報出力を行うことを意味する。キーワード「ftp」を条件に指定すれば、「Ethertype=0x800, protocol=tcp, port=21」を指定したのと同じ意味となる。つまりＦＴＰ（File Transfer Protocol）で送受信されたパケットを意味する。

[0114] また記述子５３における出力情報の形式は、「argN (N=0,1,2,…)」、「timestamp」、「stack」などのキーワード、または、「引数番号：引数種別：範囲」の形式で示される。「argN」は引数番号「Ｎ」の値を出力せよという指定である。「timestamp」は、その関数を通過した時刻を出力せよという指定である。「stack」は、その関数のスタックを出力せよという指定である。「引数番号：引数種別：範囲」の指定は、引数番号で指定された引数に設定されたアドレスの、引数種別で指定された種別の情報から、範囲で指定された部分を出力せよという指定である。引数番号と引数種別の指定方法は、限

定条件の指定方法と同様である。

[0115] 出力情報における範囲は、例えば「Ａ－Ｂ」という形式で指定する。例えば以下のような指定を可能とする。範囲を「０－６４」と指定すると、パケットの０バイト目～６４バイト目までを出力せよという指定となる。範囲を、「０－ＩＰ＿ｈｅａｄｅｒ」と指定すると、パケットの０バイト目～ＩＰヘッダの末尾までを出力せよという指定となる。範囲を「１４－６４」と指定すると、１４バイト目～６４バイト目までを出力せよという指定となる。範囲を「０－ｅｎｄ」と指定すると、パケット全体を出力せよという指定となる。

[0116] 図１５は、トレースコマンドと出力情報指定ファイルとに応じた出力結果の第１の例を示す図である。図１５の例では、トレースコマンド６１において、ファイル名「ｆｉｌｅ＿１」の出力情報指定ファイル６２が指定されている。

[0117] この出力情報指定ファイル６２の例では、関数「ｆｕｎｃ＿ｂ」に対応する呼び出し元関数を実行する呼び出し元関数３１から、関数「ｆｕｎｃ＿ａ」が呼ばれるたびに、トレース情報の取得処理が実行される。トレース情報の取得処理では、関数「ｆｕｎｃ＿ａ」の引数「０」をネットワークで受け渡されるパケット情報であると考えた場合に、「Ethertype」が「0x800」（つまりＩＰパケットである）場合に限って出力が行われる。出力情報は、引数「０」の値、引数「１」の値、タイムスタンプ、スタック、パケットの０バイト目～ＩＰヘッダの末尾までである。解析結果としては、例えば図１５の解析結果ファイル６３に示す内容が出力される。

[0118] 図１６は、トレースコマンドと出力情報指定ファイルとに応じた出力結果の第２の例を示す図である。図１６の例では、トレースコマンド６４において、ファイル名「ｆｉｌｅ＿２」の出力情報指定ファイル６５が指定されている。

[0119] この出力情報指定ファイル６５の例では、関数「ｆｕｎｃ＿ｂ」に対応するプログラムを実行する呼び出し元関数３１から、関数「ｆｕｎｃ＿ａ」が

呼ばれるたびに、トレース情報の取得処理が実行される。トレース情報の取得処理では、関数「ｆｕｎｃ＿ａ」の引数０をネットワークで受け渡されるパケット情報であると考えた場合に、ＦＴＰのパケットで、かつ、「abc」というパターンを含むパケットのみ出力される。出力情報は、引数０の値、引数１の値、タイムスタンプ、スタック、パケットの全体である。解析結果としては、例えば図１６の解析結果ファイル６６に示す内容が出力される。

[0120] このように、予め出力情報指定ファイルを用意しておくことで、出力情報に関する条件を詳細に指定できる。トレース管理部１３０は、出力情報指定ファイルに示された出力情報の条件を、例えば共通域を介して、カーネル空間内の第１のデバッグ関数３３または第２のデバッグ関数３４に渡す。共通域は、カーネル空間内のプロセスとユーザ空間内のプロセスとの両方が使用できる、カーネル空間内のメモリ領域である。

[0121] 図１７は、共通域のデータ構造の一例を示す図である。共通域４０には、例えば３つのフィールド４１～４３が設けられている。

最初のフィールド４１には、情報連鎖構造体のアドレスが設定される。情報連鎖構造体は、複数の呼び出し元から、トレース対象の関数の呼び出しが重複した場合に、複数の関数呼び出しそれぞれの解析結果を連鎖的に関連付ける情報である。共通域４０の情報連鎖構造体のアドレスのフィールド４１には、連鎖関係の先頭にある情報連鎖構造体のアドレスが設定される。

[0122] ２つめのフィールド４２には、デバッグ結果構造体の大きさが設定される。デバッグ結果構造体は、デバッグによる解析結果の格納領域である。

３つめのフィールド４３には、デバッグ要求構造体が設定される。デバッグ要求構造体は、トレース管理部１３０からのデバッグ要求の内容を示す情報である。

[0123] なおデバッグ要求構造体は、トレースコマンドで指定された出力情報指定ファイルに基づいて、デバッグ関数生成部１３４によって生成される。

図１８は、デバッグ要求構造体のデータ構造の一例を示す図である。デバッグ要求構造体７０には、基本情報領域７１と拡張情報領域７２とが含まれ

る。

[0124] 基本情報領域７１には、構造体の大きさ、引数ごとの引数要求フラグ、タイムスタンプ要求フラグ、およびスタック要求フラグの各フィールドが設けられている。

構造体の大きさのフィールドには、デバッグ要求構造体７０のデータサイズが設定される。

[0125] 引数ごとの引数要求フラグのフィールドには、関数呼び出し命令に設定された引数の値を出力するか否かを示すフラグが設定される。図１８の例では、関数呼び出し命令に対して最大で１６個の引数が設定される場合が想定されている。関数呼び出し命令に設定される引数は、例えば、左から順番に０～１５の引数番号が設定される。例えば、引数番号の引数の値を出力しない場合、その引数番号に対応する引数要求フラグのフィールドに「０」が設定される。引数番号の引数の値を出力する場合、その引数番号に対応する引数要求フラグのフィールドに「１」が設定される。

[0126] タイムスタンプ要求フラグのフィールドには、タイムスタンプを出力するか否かを示すフラグが設定される。例えばタイムスタンプを出力しない場合、タイムスタンプ要求フラグのフィールドに「０」が設定される。また呼び出し元関数３１による関数呼び出し命令実行時を示すタイムスタンプを出力する場合、タイムスタンプ要求フラグのフィールドに「１」が設定される。

[0127] スタック要求フラグのフィールドには、スタックの情報を出力するか否かを示すフラグが設定される。例えばスタックの情報を出力しない場合、スタック要求フラグのフィールドに「０」が設定される。またスタックの情報を出力する場合、スタック要求フラグのフィールドに「１」が設定される。

[0128] 拡張情報領域７２には、引数番号、引数種別、限定条件個数、複数の限定条件、および複数の取り出し部分の各フィールドが設けられている。

引数番号のフィールドには、出力する情報が格納された記憶領域のアドレスが設定された引数の番号が設定される。

[0129] 引数種別のフィールドには、出力する情報の種別が設定される。例えば出

力する情報が、ネットワーク送受信データを表すメッセージブロック構造体であれば、引数種別のフィールドに「１」が設定される。また出力する情報が、プレーンデータのファイルであれば、引数種別のフィールドに「２」が設定される。

[0130] 限定条件個数のフィールドには、限定条件の個数が設定される。

限定条件のフィールドには、情報の出力を行うか否かの判断基準となる限定条件が設定される。例えば、引数種別が「１」の場合、限定条件のフィールドに、「Ethertype」、「送信元／送信先ＩＰアドレス」、「パケット長」などの条件が設定される。

[0131] 取り出し部分のフィールドには、出力する情報から解析結果として取り出す部分を指定する情報が設定される。例えば、引数種別が「１」の場合、先頭からＭＡＣヘッダまで、先頭からＩＰヘッダまで、先頭からＴＣＰヘッダまで、ＩＰヘッダのみ、ＴＣＰヘッダのみ、ＴＣＰチェックサム値のみ、xxバイト目～yyバイト目までなどが、取り出し部分として指定できる。

[0132] 図１９は、デバッグ結果構造体のデータ構造の一例を示す図である。デバッグ結果構造体７３には、引数番号ごとの引数要求フラグのフィールドと、引数要求フラグのフィールドそれぞれに対応する引数の値のフィールドが設けられている。引数要求フラグのフィールドには、対応する引数番号で示される引数を取得したか否かを示すフラグが設定される。引数の値のフィールドには、対応する引数番号で示された引数の値が設定される。

[0133] またデバッグ結果構造体７３には、タイムスタンプ要求フラグのフィールドと、タイムスタンプの値のフィールドが設けられている。タイムスタンプ要求フラグのフィールドには、タイムスタンプを取得したか否かを示すフラグが設定される。タイムスタンプの値のフィールドには、取得したタイムスタンプの値が設定される。

[0134] またデバッグ結果構造体７３には、スタック要求フラグのフィールド、スタックの深さのフィールド、およびスタックに示される関数名のフィールドが設けられている。スタック要求フラグのフィールドには、スタックを取得

したか否かを示すフラグが設定される。スタックの深さのフィールドには、何段階の関数呼び出しが行われたか（スタックに示される関数名の数）が設定される。関数名のフィールドには、スタックに含まれる関数名が設定される。例えば関数名のフィールドには順番が設定されており、呼び出し元の関数から順に、その関数の関数名が関数名のフィールドに設定される。

[0135] またデバッグ結果構造体７３には、取り出し部分の大きさのフィールドと、取り出し部分のフィールドとが設けられている。取り出し部分の大きさのフィールドには、引数に設定されたアドレスで示された記憶領域内の情報から取り出した部分データのサイズが設定される。取り出し部分のフィールドには、引数に設定されたアドレスで示された記憶領域内の情報から取り出した部分データが設定される。

[0136] 図２０は、情報連鎖構造体のデータ構造の一例を示す図である。情報連鎖構造体７４には、次の情報連鎖構造体のアドレスのフィールド、デバッグ結果構造体のアドレスのフィールド、非リアルタイム解析情報種別のフィールド、および非リアルタイム解析情報領域のアドレスのフィールドが設けられている。

[0137] 次の情報連鎖構造体のアドレスのフィールドには、次に連鎖された情報連鎖構造体が格納された記憶領域の先頭のアドレスが設定される。なお、次に連鎖された情報連鎖構造体が存在しない場合、次の情報連鎖構造体のアドレスのフィールドには、例えば「０」が設定される。

[0138] デバッグ結果構造体のアドレスのフィールドには、デバッグ結果構造体が格納された記憶領域の先頭のアドレスが設定される。

非リアルタイム解析情報種別のフィールドには、非リアルタイム解析情報が格納された記憶領域のデータ構造の種別が設定される。例えば非リアルタイム解析情報のデータ構造がメッセージブロック構造体であれば、非リアルタイム解析情報種別のフィールドに「１」が設定される。また非リアルタイム解析情報のデータ構造がプレーンデータであれば、非リアルタイム解析情報種別のフィールドに「２」が設定される。

[0139] 非リアルタイム解析情報領域のアドレスのフィールドには、非リアルタイム解析情報が格納された記憶領域の先頭のアドレスが設定される。

図２０に示した情報連鎖構造体７４は、複数の情報連鎖構造体によって連鎖関係を定義できる。

[0140] 図２１は、情報連鎖構造体による連鎖関係の一例を示す図である。図２１の例では、３つの情報連鎖構造体８１～８３が連鎖関係となっている。共通域４０の情報連鎖構造体のアドレスのフィールド４１には、情報連鎖構造体８１が格納された記憶領域の先頭のアドレスが設定されている。情報連鎖構造体８１の次の情報連鎖構造体のアドレスのフィールドには、情報連鎖構造体８２が格納された記憶領域の先頭のアドレスが設定されている。情報連鎖構造体８２の次の情報連鎖構造体のアドレスのフィールドには、情報連鎖構造体８３が格納された記憶領域の先頭のアドレスが設定されている。情報連鎖構造体８３の次の情報連鎖構造体のアドレスのフィールドには、連鎖の最後であることを示す値「０」が設定されている。

[0141] また、情報連鎖構造体８１～８３それぞれに設定されたデバッグ結果構造体のアドレスにより、情報連鎖構造体８１～８３それぞれに、デバッグ結果構造体８４～８５が関連付けられている。さらに情報連鎖構造体８１～８３それぞれに設定された非リアルタイム解析情報領域のアドレスにより、情報連鎖構造体８１～８３それぞれに、非リアルタイム解析情報を含む記憶領域（非リアルタイム解析情報領域）８７～８９が関連付けられている。

[0142] 次に、処理時間情報記憶部１３２のデータ構造について説明する。

図２２は、処理時間情報記憶部のデータ構造の一例を示す図である。処理時間情報記憶部１３２には、時間見積もりテーブル１３２ａが格納されている。時間見積もりテーブル１３２ａには、解析情報、リアルタイム解析の要否、解析時間、情報引渡時間、および依頼準備時間の列が設けられている。

[0143] 解析情報の列には、トレース処理で解析対象となる情報（解析情報）の種別が設定される。解析情報には、出力情報指定ファイルによって出力対象として指定可能な情報（出力情報）と、限定条件との照合対象となる情報とが

含まれる。リアルタイム解析の要否の列には、対応する解析情報が、リアルタイムに解析する対象か否かが設定される。解析時間の列には、解析情報の解析に要する時間の見積もり値が設定される。情報引渡時間の列には、解析結果のトレース情報をトレース管理部１３０に引き渡すのに要する時間の見積もり値が設定される。依頼準備時間の列には、非リアルタイム解析情報の解析をデバッグスレッド３５に依頼するための準備処理に要する時間の見積もり値が設定される。

[0144] このような時間見積もりテーブル１３２ａに基づいて、トレース準備処理（図１３のステップＳ１２）において、トレース取得手法を第１の手法にするのか第２の手法にするのかを、適切に判断することができる。

[0145] 図２３は、トレース準備処理の手順を示すフローチャートである。以下、図２３に示す処理をステップ番号に沿って説明する。

［ステップＳ３１］トレース管理部１３０のデバッグ関数生成部１３４は、カーネル空間内に共通域４０を獲得する。次に、デバッグ関数生成部１３４は、トレースコマンドの引数で指定された出力情報指定ファイルの内容に従って、デバッグ要求構造体を生成する。そしてデバッグ関数生成部１３４は、生成したデバッグ要求構造体を共通域４０に格納する。またデバッグ関数生成部１３４は、デバッグ結果構造体の大きさを計算し、共通域４０に格納する。

[0146] ［ステップＳ３２］処理時間解析部１３３は、非リアルタイム解析情報の解析時間の算出処理を行う。ここで、算出された非リアルタイム解析情報の解析時間を変数「ａ」に設定する。この処理の詳細は後述する（図２４参照）。

[0147] ［ステップＳ３３］処理時間解析部１３３は、トレース管理部１３０への情報引渡時間の算出処理を行う。ここで、算出されたトレース管理部１３０への情報引渡時間を変数「ｂ」に設定する。この処理の詳細は後述する（図２５参照）。

[0148] ［ステップＳ３４］処理時間解析部１３３は、デバッグスレッド３５への

情報引渡時間の算出処理を行う。ここで、算出されたデバッグスレッド３５への情報引渡時間を変数「ｃ」に設定する。この処理の詳細は後述する（図２６参照）。

[0149] ［ステップＳ３５］処理時間解析部１３３は、トレース取得手法として第１の手法が適切か、第２の手法が適切かの判定を行う。例えば処理時間解析部１３３は、非リアルタイム解析情報の解析時間（ａ）とトレース管理部１３０への情報引渡時間（ｂ）との加算結果と、デバッグスレッド３５への情報引渡時間（ｃ）とを比較する。時間が等しいか、またはデバッグスレッド３５への情報引渡時間（ｃ）の方が長ければ、処理時間解析部１３３は、第１の手法が適切であると判定し、処理をステップＳ３６に進める。非リアルタイム解析情報の解析時間（ａ）とトレース管理部１３０への情報引渡時間（ｂ）との加算結果の方が長ければ、処理時間解析部１３３は、第２の手法が適切であると判定し、処理をステップＳ３８に進める。

[0150] ［ステップＳ３６］処理時間解析部１３３において第１の手法が適切であると判定された場合、デバッグ関数生成部１３４は、関数名「ｆｕｎｃ＿ｄｂｇ１」の第１のデバッグ関数３３を生成する。デバッグ関数生成部１３４は、生成した第１のデバッグ関数３３をカーネル空間にロードする。そして、デバッグ関数生成部１３４は、生成した第１のデバッグ関数３３が関数名「ｆｕｎｃ＿ｄｂｇ１」の呼び出しに応じて実行されるように、ＯＳ１２０に対して設定する。

[0151] ［ステップＳ３７］処理時間解析部１３３において第１の手法が適切であると判定された場合、呼び出し命令変更部１３５は、呼び出し元関数３１内の関数呼び出し先を、第１のデバッグ関数３３に変更する。例えば呼び出し命令変更部１３５は、カーネル空間内において、呼び出し元関数３１内のデバッグ対象関数３２の呼び出し命令を、第１のデバッグ関数３３の呼び出し命令に書き換える。その後、トレース準備処理が終了する。

[0152] ［ステップＳ３８］処理時間解析部１３３において第２の手法が適切であると判定された場合、デバッグ関数生成部１３４は、関数名「ｆｕｎｃ＿ｄ

ｂｇ２」の第２のデバッグ関数３４を生成する。デバッグ関数生成部１３４は、生成した第２のデバッグ関数３４をカーネル空間にロードする。そしてデバッグ関数生成部１３４は、生成した第２のデバッグ関数３４が、関数名「ｆｕｎｃ＿ｄｂｇ２」の呼び出しに応じて実行されるように、ＯＳ１２０に対して設定する。

[0153] ［ステップＳ３９］処理時間解析部１３３において第２の手法が適切であると判定された場合、呼び出し命令変更部１３５は、呼び出し元関数３１内の関数呼び出し先を、第２のデバッグ関数に変更する。例えば呼び出し命令変更部１３５は、カーネル空間内において、呼び出し元関数３１内のデバッグ対象関数３２の呼び出し命令を、第２のデバッグ関数３４の呼び出し命令に書き換える。その後、トレース準備処理が終了する。

[0154] 次に、非リアルタイム解析情報の解析時間の算出処理の手順について説明する。

図２４は、非リアルタイム解析情報の解析時間の算出処理の手順の一例を示すフローチャートである。以下、図２４に示す処理をステップ番号に沿って説明する。

[0155] ［ステップＳ４１］処理時間解析部１３３は、時間見積もりテーブル１３２ａの解析情報の列において、トレースコマンドによってユーザが指定した解析情報が設定されたエントリ（行）を選択する。例えば処理時間解析部１３３は、共通域４０に格納されたデバッグ要求構造体７０の基本情報領域７１において情報出力を行うことが示されている（例えばフラグの値が「１」）引数要求フラグに対応する引数の引数番号を判断する。次に処理時間解析部１３３は、トレース対象の関数（ｆｕｎｃ＿ａ）の引数の記述フォーマットに基づいて、情報出力を行う引数に対応する情報の種別を判断する。そして処理時間解析部１３３は、情報出力の対象となる種別の情報の行を、時間見積もりテーブル１３２ａからすべて選択する。また処理時間解析部１３３は、デバッグ要求構造体７０の拡張情報領域７２において、設定されている限定条件の照合対象となる情報、および取り出し部分として指定されている

情報の行を、時間見積もりテーブル１３２ａからすべて選択する。

[0156] ［ステップＳ４２］処理時間解析部１３３は、選択した各行の情報について、何個の引数から解析情報として指定されているのかを計数し、ＲＡＭ１０２などに記録する。

［ステップＳ４３］処理時間解析部１３３は、ステップＳ４１で選択した列の情報のうち、リアルタイム解析が不要な情報をすべて選択する。例えば処理時間解析部１３３は、時間見積もりテーブル１３２ａにおけるステップＳ４１で選択した行のリアルタイム解析の要否の列の値を参照して、リアルタイム解析が不要と設定されている情報の行をすべて選択する。

[0157] ［ステップＳ４４］処理時間解析部１３３は、ステップＳ４３で選択した行の解析時間を合計し、変数「ａ」に設定する。なお処理時間解析部１３３は、複数の引数から解析情報として指定されている情報については、合計する前に、その情報の解析時間に対して、指定を行っている引数の個数を乗算する。

[0158] このようにして、非リアルタイム解析情報の解析時間が算出される。

次に、トレース管理部１３０への情報引渡時間の算出処理の手順について説明する。

図２５は、トレース管理部への情報引渡時間の算出処理の手順の一例を示すフローチャートである。以下、図２５に示す処理をステップ番号に沿って説明する。

[0159] ［ステップＳ５１］処理時間解析部１３３は、時間見積もりテーブル１３２ａの解析情報の列において、トレースコマンドによってユーザが指定した解析情報が設定されたエントリ（行）を選択する。この処理の詳細は、図２４のステップＳ４１と同様である。

[0160] ［ステップＳ５２］処理時間解析部１３３は、選択した各行の情報について、何個の引数から解析情報として指定されているのかを計数し、ＲＡＭ１０２などに記録する。

［ステップＳ５３］処理時間解析部１３３は、ステップＳ５１で選択した

行の情報引渡時間を合計し、変数「ｂ」に設定する。なお処理時間解析部１３３は、複数の引数から解析情報として指定されている情報については、合計する前に、その情報の情報引渡時間に対して、指定を行っている引数の個数を乗算する。

[0161] このようにして、トレース管理部１３０への情報引渡時間が算出される。

次に、デバッグスレッドへの情報引渡時間の算出処理の手順について説明する。

図２６は、デバッグスレッドへの情報引渡時間の算出処理の手順の一例を示すフローチャートである。以下、図２６に示す処理をステップ番号に沿って説明する。

[0162] ［ステップＳ６１］処理時間解析部１３３は、時間見積もりテーブル１３２ａの解析情報の列において、トレースコマンドによってユーザが指定した解析情報が設定されたエントリ（行）を選択する。この処理の詳細は、図２４のステップＳ４１と同様である。

[0163] ［ステップＳ６２］処理時間解析部１３３は、選択した各行の情報について、何個の引数から解析情報として指定されているのかを計数し、ＲＡＭ１０２などに記録する。

［ステップＳ６３］処理時間解析部１３３は、ステップＳ６１で選択した行の依頼準備時間を合計する。なお処理時間解析部１３３は、複数の引数から解析情報として指定されている情報については、合計する前に、その情報の依頼準備時間に対して、指定を行っている引数の個数を乗算する。

[0164] ［ステップＳ６４］処理時間解析部１３３は、ステップＳ６３で計算した合計値に、デバッグスレッドをウェイクアップ（wakeup）させる時間を加算し、変数「ｃ」に設定する。なお、デバッグスレッドをウェイクアップさせる時間は、例えば予め処理時間解析部１３３に設定されている。

[0165] このようにして、デバッグスレッドへの情報引渡時間が算出される。

このようにして算出された非リアルタイム解析情報の解析時間、トレース管理部への情報引渡時間、およびデバッグスレッドへの情報引渡時間に基づ

いて、適用するトレース手法が判定される。以下、具体的な計算例について説明する。

[0166] 図２７は、時間見積もりテーブルに設定された情報の一例を示す図である。図２７の時間見積もりテーブル１３２ａの例では、解析時間、情報引渡時間、依頼準備時間の各時間が、マイクロ秒（μｓ）単位で示されている。また解析情報ごとの解析時間、情報引渡時間、依頼準備時間には、例えば所定の環境で実測した値が設定されている。

[0167] なお、図２７には示していないが、デバッグスレッドをウェイクアップさせるのに要する時間は、「１０μｓ」であるものとする。

図２７に示したような時間見積もりテーブル１３２ａに基づく、トレース手法の２つの判定例を以下に示す。

[0168] ＜第１のトレース手法判定例＞

第１の例では、トレースコマンドで、スタック、タイムスタンプ、引数「０」に設定されたアドレス、引数「１」に設定されたアドレスを出力するように指定されたものとする。この場合の非リアルタイム解析情報の解析時間、トレース管理部への情報引渡時間、およびデバッグスレッドへの情報引渡時間は、以下のようにして算出される。

[0169] 非リアルタイム解析情報の解析時間（変数「ａ」）＝０μｓである。

これは、トレースコマンドで指定された解析情報が、全てリアルタイム解析が必要な情報だからである。

[0170] トレース管理部への情報引渡時間（変数「ｂ」）＝１０μｓ（スタックを渡すのにかかる時間）＋１μｓ（タイムスタンプを渡すのにかかる時間）＋１μｓ×２(引数を渡すのにかかる時間×２個)＝１３μｓである。

[0171] デバッグスレッドへの情報引渡時間（変数「ｃ」）＝１０μｓ（スタックを渡すのにかかる時間）＋１μｓ（タイムスタンプを渡すのにかかる時間）＋１μｓ×２（引数を渡すのにかかる時間×２個）＋１０μｓ（デバッグスレッドのwakeup時間）＝２３μｓである。

[0172] この場合、ａ＋ｂ＝０＋１３＝１３μｓ、ｃ＝２３μｓなので、図２３の

ステップＳ３５の「ａ＋ｂ≦ｃ」の判定は、ＹＥＳとなる。その結果、第１の手法と判定される。

＜第２のトレース手法判定例＞

第２の例では、引数「１」の引数に、メッセージブロック構造体の先頭アドレスが設定されている場合を想定する。このときトレースコマンドで、スタック、タイムスタンプ、引数「０」に設定されたアドレス、引数「１」に設定されたアドレス、引数「１」が示すパケットのEthertypeが0x800のものだけについて、そのパケットの全体を出力するように指定されたものとする。この場合の非リアルタイム解析情報の解析時間、トレース管理部への情報引渡時間、およびデバッグスレッドへの情報引渡時間は、以下のようにして算出される。

[0173] 非リアルタイム解析情報の解析時間（変数「ａ」）＝２μｓ（Ethertype=0x800であることの解析にかかる時間）

トレース管理部への情報引渡時間（変数「ｂ」）＝１０μｓ（スタックを渡すのにかかる時間）＋１μｓ（タイムスタンプを渡すのにかかる時間）＋１μｓ×２（引数を渡すのにかかる時間×２個）＋１００μｓ（パケット全体を渡すのにかかる時間）＝１１３μｓ

デバッグスレッドへの情報引渡時間（変数「ｃ」）＝１０μｓ（スタックを渡すのにかかる時間）＋１μｓ（タイムスタンプを渡すのにかかる時間）＋１μｓ×２（引数を渡すのにかかる時間×２個）＋２μｓ（パケット全体を渡すのにかかる時間）＋１０μｓ（デバッグスレッドのwakeup時間）＝２５μｓ

この場合、ａ＋ｂ＝２＋１１３＝１１５μｓ、ｃ＝２５μｓなので、図２３のステップＳ３５の「ａ＋ｂ≦ｃ」の判定は、ＮＯとなる。その結果、第２の手法と判定される。

[0174] 以上のようにして、第１の手法と第２の手法とのうち、デバッグ処理を行うプロセスから呼び出し元のプロセスへの応答時間が短くなる方の手法によって、トレース処理を含むデバッグ処理が実行される。以下、デバッグ処理

の手順について詳細に説明する。

[0175] まず第１の手法でトレース処理を実行する第１のデバッグ処理について説明する。

図２８は、第１のデバッグ処理の手順の一例を示すフローチャートである。以下、図２８に示す処理をステップ番号に沿って説明する。以下の処理は、第１のデバッグ関数３３に記述された命令に基づいて、コンピュータ１００が実行する。

[0176] ［ステップＳ７１］ＯＳ１２０は、呼び出し元関数３１から第１のデバッグ関数３３呼び出しがあるか否かを判断する。第１のデバッグ関数の呼び出しがある場合、ＯＳ１２０は第１のデバッグ関数３３を起動し、処理をステップＳ７２に進める。第１のデバッグ関数の呼び出しがなければ、ＯＳ１２０は、ステップＳ７１の処理を繰り返す。

[0177] ［ステップＳ７２］ＯＳ１２０は、save命令を実行し、レジスタ内の情報をメモリに退避する。

［ステップＳ７３］コンピュータ１００３３は、デバッグ結果構造体を生成する。ＯＳ１２０は、生成したデバッグ結果構造体をＲＡＭ１０２などのメモリに格納する。

[0178] ［ステップＳ７４］ＯＳ１２０は、リアルタイム解析情報を解析する。例えばＯＳ１２０は、共通域４０に格納されたデバッグ要求構造体７０に基づいて、出力対象の情報を判断する。次にＯＳ１２０は、出力対象の情報のうち、リアルタイムの解析が必要な情報について解析する。なおデバッグ結果構造体に設定する情報のうち、どの情報がリアルタイムに解析する情報なのかは、例えば第１のデバッグ関数３３内に定義しておくことができる。

[0179] ［ステップＳ７５］ＯＳ１２０は、リアルタイム解析情報の解析結果を、デバッグ結果構造体に格納する。

［ステップＳ７６］ＯＳ１２０は、デバッグ対象関数３２の関数呼び出しに関連する情報が限定条件に合致するか否かを判断する。なおＯＳ１２０は、共通域４０に格納されたデバッグ要求構造体７０に基づいて、限定条件を

認識する。第１のデバッグ関数３３は、限定条件に合致する場合、処理をステップＳ７７に進める。またＯＳ１２０は、限定条件に合致しない場合、処理をステップＳ８０に進める。

[0180] ［ステップＳ７７］ＯＳ１２０は、出力対象の情報のうちの非リアルタイム解析情報を解析する。

［ステップＳ７８］ＯＳ１２０は、非リアルタイム解析情報の解析結果を、デバッグ結果構造体に格納する。

[0181] ［ステップＳ７９］ＯＳ１２０は、デバッグ結果構造体の情報を、トレース管理部１３０に引き渡す。例えばＯＳ１２０は、デバッグ結果構造体の先頭のアドレスが設定された情報連鎖構造体を生成し、共通域４０の生成した情報連鎖構造体の先頭のアドレスを設定する。するとトレース管理部１３０のトレース情報出力部１３６は、共通域４０を参照して情報連鎖構造体の先頭アドレスを認識し、その情報連鎖構造体を参照してデバッグ結果構造体の先頭アドレスを認識する。そして、トレース情報出力部１３６は、デバッグ結果構造体の内容を取得する。

[0182] ［ステップＳ８０］ＯＳ１２０は、レジスタの内容をＲＡＭ１０２からレジスタに戻すrestore命令を実行する。

［ステップＳ８１］ＯＳ１２０は、トレース対象の関数（ｆｕｎｃ＿ａ）の呼び出しを行う。

[0183] ［ステップＳ８２］ＯＳ１２０は、デバッグ対象関数３２から戻り値を取得したか否かを判断する。ＯＳ１２０は、戻り値を取得した場合、処理をステップＳ８３に進める。またＯＳ１２０は、戻り値を取得していなければ、ステップＳ８２の処理を繰り返し、デバッグ対象関数３２の処理の実行による戻り値を待つ。

[0184] ［ステップＳ８３］ＯＳ１２０は、復帰命令を実行し、呼び出し元関数３１に処理を復帰させる。この際、ＯＳ１２０は、デバッグ対象関数３２から取得した戻り値を、呼び出し元関数３１への戻り値とする。その後、第１のデバッグ関数３３は、処理をステップＳ７１に進める。

[0185] このようにして、第１の手法によるトレース処理を伴うデバッグ処理が実行される。

次に、第２の手法でトレース処理を実行する第２のデバッグ処理について説明する。

図２９は、第２のデバッグ処理の手順の一例を示すフローチャートである。以下、図２９に示す処理をステップ番号に沿って説明する。図２９に示す処理は、第２のデバッグ関数３４に記述された命令に基づいて、コンピュータ１００が実行する。なお、ステップＳ９１～Ｓ９５、ステップＳ９８～Ｓ１０１の処理は、それぞれ図２８に示したステップＳ７１～Ｓ７５，Ｓ８０～Ｓ８３の処理と同じである。

[0186] ［ステップＳ９６］ＯＳ１２０は、デバッグスレッドへの情報引渡準備処理を行う。この処理の詳細は後述する（図３０参照）。

［ステップＳ９７］ＯＳ１２０は、デバッグスレッド３５のウェイクアップ信号を送信する。ウェイクアップ信号に応じて、デバッグスレッド３５がスリープ状態から起動状態に移行する。

[0187] 図３０は、デバッグスレッドへの情報引渡準備処理の手順の一例を示すフローチャートである。以下、図３０に示す処理をステップ番号に沿って説明する。

［ステップＳ１１１］ＯＳ１２０は、非リアルタイム解析情報が参照カウンタを有する領域に格納されているか否かを判断する。例えば変数に設定されたアドレスで示された領域内の情報が出力情報の対象として指定されており、その情報が図６に示したようなメッセージブロック構造体の場合、参照カウンタを有する領域に格納されていると判断される。ＯＳ１２０は、参照カウンタを有する領域に格納されている場合、処理をステップＳ１１２に進める。またＯＳ１２０は、参照カウンタを有する領域に格納されていない場合、処理をステップＳ１１３に進める。

[0188] ［ステップＳ１１２］ＯＳ１２０は、出力対象として指定されている情報が格納されている領域の参照カウンタに１を加算する。その後、ＯＳ１２０

は、処理をステップＳ１１４に進める。

[0189] ［ステップＳ１１３］ＯＳ１２０は、出力対象の非リアルタイム解析情報を含む情報を、ＲＡＭ１０２内の新たな記憶領域にコピーする。

［ステップＳ１１４］ＯＳ１２０は、情報連鎖構造体を生成する。

[0190] ［ステップＳ１１５］ＯＳ１２０は、情報連鎖構造体に、デバッグ結果構造体のアドレス、非リアルタイム解析情報を含む領域の種別、非リアルタイム解析情報を含む領域の先頭のアドレスを設定する。非リアルタイム解析情報を含む領域の種別は、例えば、メッセージブロック構造体であれば「１」、プレーンデータであれば「２」が設定される。

[0191] ［ステップＳ１１６］ＯＳ１２０は、共通域４０に設定されている情報連鎖構造体アドレスから始まる情報連鎖構造体の連鎖の先端に、ステップＳ１１４で生成した情報連鎖構造体を連鎖させる。例えばＯＳ１２０は、最後に連鎖された情報連鎖構造体の「次の情報連鎖構造体のアドレス」のフィールドに、ステップＳ１１４で生成した情報連鎖構造体の記憶領域の先頭のアドレスを設定する。またＯＳ１２０は、ステップＳ１１４で生成した情報連鎖構造体の「次の情報連鎖構造体のアドレス」のフィールドには「０」を設定する。

[0192] このようにしてデバッグスレッド３５へ情報を引き渡す準備が完了する。この状況でデバッグスレッド３５がwakeupされると、デバッグスレッド３５は共通域４０から情報連鎖構造体のアドレスを順にたどることで、デバッグスレッド３５はすべての情報連鎖構造体の情報を取得可能となる。またデバッグスレッド３５は、各情報連鎖構造体に設定されているデバッグ結果構造体のアドレスを参照することで、リアルタイム解析情報を取得できる。またデバッグスレッド３５は、共通域４０に設定されたデバッグ要求構造体７０を参照することで、非リアルタイムで解析する情報を認識できる。

[0193] 次に、情報の引き渡しを受けたデバッグスレッド３５の処理について説明する。なお、デバッグスレッド３５はＯＳ１２０の機能の一部であるが、第２のデバッグ処理など処理と並行して処理を実行することができる。

[0194] 図３１は、デバッグスレッドの処理の手順の一例を示す図である。以下、図３１の処理をステップ番号に沿って説明する。

［ステップＳ１２１］デバッグスレッド３５は、生成された直後、および共通域４０の情報連鎖構造体のアドレスが「０」となったとき、スリープ状態に移行する。

[0195] ［ステップＳ１２２］デバッグスレッド３５は、第２のデバッグ関数３４の処理によってウェイクアップ信号が入力されたか否かを判断する。デバッグスレッド３５は、ウェイクアップ信号が入力された場合、処理をステップＳ１２３に進める。またデバッグスレッド３５は、ウェイクアップ信号が入力されていなければ、ステップＳ１２２の処理を繰り返し、ウェイクアップ信号の入力を待つ。

[0196] ［ステップＳ１２３］デバッグスレッド３５は、ウェイクアップ信号の入力に応じて、ウェイクアップする（動作状態に移行する）。

［ステップＳ１２４］デバッグスレッド３５は、共通域４０の情報連鎖構造体のアドレスが「０」か否かを判断する。デバッグスレッド３５は、アドレスが「０」の場合、デバッグスレッド３５は、解析する情報連鎖構造体が無いと判断し、処理をステップＳ１２１に進める。またアドレスが「０」以外の場合、デバッグスレッド３５は処理をステップＳ１２５に進める。

[0197] ［ステップＳ１２５］デバッグスレッド３５は、共通域４０の情報連鎖構造体のアドレスのフィールド４１に設定されたアドレスで示される領域に格納された情報連鎖構造体を、解析対象と判断する。次にデバッグスレッド３５は、解析対象の情報連鎖構造体に設定された非リアルタイム解析情報のアドレスを参照する。そしてデバッグスレッド３５は、参照したアドレスが示す領域に格納されている非リアルタイム解析情報が、デバッグ要求構造体に示される限定条件に合致するか否かを判断する。デバッグスレッド３５は、限定条件に合致する場合、処理をステップＳ１２６に進める。またデバッグスレッド３５は、限定条件に合致しない場合、処理をステップＳ１２４に進める。

[0198] ［ステップＳ１２６］デバッグスレッド３５は、非リアルタイム解析情報を解析する。例えばデバッグスレッド３５は、解析対象の情報連鎖構造体に設定された非リアルタイム解析情報のアドレスを参照する。そしてデバッグスレッド３５は、参照したアドレスが示す領域に格納されている非リアルタイム解析情報から、デバッグ要求構造体において指定されている取り出し部分を抽出し、解析結果とする。

[0199] ［ステップＳ１２７］デバッグスレッド３５は、解析対象の情報連鎖構造体に設定されたデバッグ結果構造体のアドレスで示されるデバッグ結果構造体に、解析結果を格納する。

[0200] ［ステップＳ１２８］デバッグスレッド３５は、解析結果をトレース管理部１３０に通知する。例えばデバッグスレッド３５は、解析対象の情報連鎖構造体に設定されたデバッグ結果構造体のアドレスで示される領域のデバッグ結果構造体のコピーを、トレース管理部１３０に通知する。

[0201] ［ステップＳ１２９］デバッグスレッド３５は、解析処理が終了した情報連鎖構造体を連鎖から外す。例えばデバッグスレッド３５は、解析対象の情報連鎖構造体の「次の情報連鎖構造体のアドレス」のフィールドに設定されていたアドレスを、共通域４０の情報連鎖構造体のアドレスのフィールド４１に設定する。

[0202] ［ステップＳ１３０］デバッグスレッド３５は、メモリ解放処理を行う。例えばデバッグスレッド３５は、情報連鎖構造体、デバッグ結果構造体、非リアルタイム解析情報のメモリ解放処理を行う。なおデバッグスレッド３５は、解放した領域が参照カウンタを持つ領域であった場合、参照カウンタから１を引き、参照カウンタが０になった場合のみ、メモリ解放処理を行う。その後、デバッグスレッド３５は、処理をステップＳ１２４に進める。

[0203] 以上のようにして、詳細な解析処理や大量のコピー処理を行っても、呼び出し元の処理を遅延させずにすむ。すなわち、詳細な解析が発生するのは、非リアルタイム解析情報の解析を行うときであると考えられる。また、大量のコピーが発生するのは、トレース管理部１３０へ結果を通知するときであ

ると考えられる。このことから、詳細な解析処理が発生した場合や大量のコピーが発生する場合は、非リアルタイム解析情報の解析処理は別スレッドで行うことで、呼び出し元の処理への応答の遅延を抑止できる。その結果、トレース処理の実施による呼び出し元の処理の遅延が抑止される。

[0204] なお、呼び出し元関数は１つに限定されず、複数のプログラムにすることもできる。例えば、トレース対象となるデバッグ対象関数３２の呼び出し命令を含むすべてのプログラムを、呼び出し元関数とすることもできる。

[0205] また、第２の実施の形態では、呼び出し元関数３１におけるデバッグ対象関数３２の呼び出し命令をデバッグプログラムの呼び出し命令に書き換えることで、デバッグ対象関数３２の呼び出し時のトレース処理の実施を可能としている。これにより、デバッグ対象関数のsave命令が存在していなくても、トレース処理の実施が可能となる。すなわち、デバッグ対象関数の先頭にsave命令があれば、save命令をデバッグ関数の呼び出し命令に書き換えることでトレース処理を実施することも可能である。しかし、このような手法を用いた場合、save命令を有していないデバッグ対象関数３２の呼び出しに関するトレース処理が実施できない。第２の実施の形態に示すように関数呼び出し命令をデバッグプログラムの呼び出し命令に変更することで、save命令を有していないデバッグ対象関数３２の呼び出しに関してもトレース処理が可能となり、トレース処理の汎用性が向上する。

[0206] また第２の実施の形態では、カーネル空間からユーザ空間に通知する情報を、限定条件や出力情報によって制限することができ、処理の効率化が可能である。すなわち、ユーザ空間内のアプリケーションプログラムによって、トレースした情報の一部をユーザ向けに出力することが可能である。しかし、この場合、カーネル空間で動作する機能からユーザ空間で動作する機能へ、トレースした情報のすべてを転送した後、ユーザアプリケーションで情報の要否を判断し、不要な場合破棄することとなる。すると、本来不要な情報の解析や解析結果の転送が発生し、処理効率が悪くなる。一方、第２の実施の形態では、限定条件に合致しない情報については、解析結果のトレース管

理部１３０への転送が抑止されている。これによりデバッグ処理の効率化が図られると共に、カーネル空間からユーザ空間へ転送されるデータ量が削減される。

[0207] また第２の実施の形態では、指定された出力情報のみをカーネル空間内で動作するデバッグ処理部またはデバッグスレッドで抽出し、トレース管理部１３０に通知するようにしている。これにより、解析する情報の低減が可能となり、デバッグ処理の効率化が図られると共に、カーネル空間からユーザ空間へ転送されるデータ量が削減される。

[0208] さらに、第２の実施の形態では、限定条件の指定を、カーネル空間内のデータの表現に依存せずに行うことができる。すなわち、第２の実施の形態では、トレース管理部１３０とカーネル空間内の機能との間でデータ受け渡しのインタフェース（データ構造）が、デバッグ要求構造体７０として規定されている。トレース管理部１３０は、指定された情報を、規定されたインタフェース（データ構造）に従った形式にして、カーネル空間内に共通域４０を獲得し、その領域にデバッグ要求構造体をコピーする。これにより、カーネル空間内でのデータ表現に関しては、カーネル空間内で判断すればよいことになり、トレース管理部１３０から容易に出力情報や限定条件を指定することができる。

[0209] 〔その他の実施の形態〕

なお、上記の各実施の形態に示した処理機能は、コンピュータによって実現することができる。その場合、コンピュータが有する機能の処理内容を記述したプログラムが提供される。そのプログラムをコンピュータで実行することにより、上記処理機能がコンピュータ上で実現される。処理内容を記述したプログラムは、コンピュータで読み取り可能な記録媒体に記録しておくことができる。コンピュータで読み取り可能な記録媒体としては、磁気記憶装置、光ディスク、光磁気記録媒体、半導体メモリなどがある。磁気記憶装置には、ハードディスク装置（ＨＤＤ）、フレキシブルディスク（ＦＤ）、磁気テープなどがある。光ディスクには、ＤＶＤ、ＤＶＤ－ＲＡＭ、ＣＤ－

ＲＯＭ／ＲＷなどがある。光磁気記録媒体には、ＭＯ（Magneto-Optical disc）などがある。

[0210] プログラムを流通させる場合には、例えば、そのプログラムが記録されたＤＶＤ、ＣＤ－ＲＯＭなどの可搬型記録媒体が販売される。また、プログラムをサーバコンピュータの記憶装置に格納しておき、ネットワークを介して、サーバコンピュータから他のコンピュータにそのプログラムを転送することもできる。

[0211] プログラムを実行するコンピュータは、例えば、可搬型記録媒体に記録されたプログラムもしくはサーバコンピュータから転送されたプログラムを、自己の記憶装置に格納する。そして、コンピュータは、自己の記憶装置からプログラムを読み取り、プログラムに従った処理を実行する。なお、コンピュータは、可搬型記録媒体から直接プログラムを読み取り、そのプログラムに従った処理を実行することもできる。また、コンピュータは、サーバコンピュータからプログラムが転送されるごとに、逐次、受け取ったプログラムに従った処理を実行することもできる。

[0212] また、上記の処理機能の少なくとも一部を、ＤＳＰ（Digital Signal Processor）、ＡＳＩＣ（Application Specific Integrated Circuit）、ＰＬＤ（Programmable Logic Device）などの電子回路で実現することもできる。

[0213] 上記については単に本発明の原理を示すものである。さらに、多数の変形、変更が当業者にとって可能であり、本発明は上記に示し、説明した正確な構成および応用例に限定されるものではなく、対応するすべての変形例および均等物は、添付の請求項およびその均等物による本発明の範囲とみなされる。

符号の説明

[0214] １　記憶手段

１ａ　第１のプログラム

１ｂ　第２のプログラム

２　判定手段

３　解析管理手段

４　第１の処理手段

５　第２の処理手段

請求の範囲

[請求項1] 第１のプログラムの呼び出しに関する情報の解析要求に基づき、該解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する第１の手法と、該解析要求で指定された情報のうちの一部の情報の解析を非リアルタイムに実行し、非リアルタイムに解析する情報以外情報の解析をリアルタイムに実行する第２の手法とのうち、前記第１のプログラムの呼び出しが行われたときの呼び出し元への応答時間が短い方の手法を判定する判定手段と、

前記第１の手法の方が応答時間が短い場合、前記第１のプログラムの呼び出し命令が記述された第２のプログラムの実行過程における該呼び出し命令の実行時に、解析要求で指定された情報のすべての情報の解析、前記第１のプログラムの処理の実行、および呼び出しに対する応答を、第１の処理手段にリアルタイムに実行させ、前記第２の手法の方が応答時間が短い場合、前記第２のプログラムの実行過程における前記呼び出し命令の実行時に、解析要求で指定された情報のうちの非リアルタイムに解析する情報以外情報の解析、非リアルタイムに解析する情報の第２の処理手段への解析依頼、前記第１のプログラムの処理の実行、および呼び出しに対する応答を、前記第１の処理手段にリアルタイムに実行させ、非リアルタイムに解析する情報の解析を、前記第２の処理手段に非リアルタイムに実行させる解析管理手段と、

を有することを特徴とする情報処理装置。

[請求項2] 前記判定手段は、解析要求で指定された情報の一部を非リアルタイムで解析させることにより前記第１の処理手段で行わずにすむ処理の処理時間と、解析要求で指定された情報の一部を非リアルタイムで解析させることにより前記第１の処理手段で新たに行うこととなる処理の処理時間とを比較し、新たに行うこととなる処理の処理時間の方が長ければ、前記第１の手法の方が応答時間が短いと判断し、行わずに

すむ処理の処理時間の方が長ければ、前記第２の手法の方が応答時間が短いと判断することを特徴とする請求の範囲第１項記載の情報処理装置。

[請求項3] 前記第１の処理手段で行わずにすむ処理の処理時間は、非リアルタイムに解析する情報の解析に要する時間と、解析結果を出力する時間とを加算した時間であることを特徴とする請求の範囲第２項記載の情報処理装置。

[請求項4] 前記第１の処理手段で新たに行うこととなる処理の処理時間は、前記第１の処理手段から前記第２の処理手段への非リアルタイムに解析する情報の解析の依頼に要する時間であることを特徴とする請求の範囲第２項または第３項のいずれかに記載の情報処理装置。

[請求項5] 前記解析管理手段はユーザ空間内で動作し、解析要求による解析対象の指定内容を、カーネル空間内に確保した共通域を介して、前記第１の処理手段に解析要求の内容を通知することを特徴とする請求の範囲第１項乃至第４項のいずれかに記載の情報処理装置。

[請求項6] 解析要求による解析対象の指定内容には、前記第２のプログラムの実行過程における前記呼び出し命令の実行時に、前記第１のプログラムの呼び出しに関する情報の解析を実行するための条件が指定されており、

前記第１の処理手段は、解析要求で指定された条件が満たされる場合に情報の解析を行うことを特徴とする請求の範囲第１項乃至第５項のいずれかに記載の情報処理装置。

[請求項7] 解析要求には、前記第１のプログラムの呼び出しに関する情報の少なくとも一部が出力対象に指示されており、

前記第１の処理手段と前記第２の処理手段とは、前記第１のプログラムの呼び出しに関する情報から解析要求で出力対象に指定された情報を取得し、該取得した情報を出力することを特徴とする請求の範囲第１項乃至第６項のいずれかに記載の情報処理装置。

[請求項8] 非リアルタイムに実行する情報は、データ構造上、記憶領域の解放を抑止させる設定項目を有する情報であり、

前記第１の処理手段は、非リアルタイムに解析する情報の記憶領域の解放を抑止した後、前記第２の処理手段に非リアルタイムに解析する情報の解析依頼を行うことを特徴とする請求の範囲第１項乃至第７項のいずれかに記載の情報処理装置。

[請求項9] 前記解析管理手段は、前記第１の手法と前記第２の手法のうちの応答時間が短い方の手法の処理が記述された第３のプログラムを生成し、前記第２のプログラム内の前記第１のプログラムの呼び出し命令を、前記第３のプログラムの呼び出し命令に変更することを特徴とする請求の範囲第１項乃至第８項のいずれかに記載の情報処理装置。

[請求項10] コンピュータに、

第１のプログラムの呼び出しに関する情報の解析要求に基づき、該解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する第１の手法と、該解析要求で指定された情報のうちの一部の情報の解析を非リアルタイムに実行し、非リアルタイムに解析する情報以外情報の解析をリアルタイムに実行する第２の手法とのうち、前記第１のプログラムの呼び出しが行われたときの呼び出し元への応答時間が短い方の手法を判定し、

前記第１の手法の方が応答時間が短い場合、前記第１のプログラムの呼び出し命令が記述された第２のプログラムの実行過程における該呼び出し命令の実行時に、解析要求で指定された情報のすべての情報の解析、前記第１のプログラムの処理の実行、および呼び出しに対する応答を、第１の処理手段にリアルタイムに実行させ、前記第２の手法の方が応答時間が短い場合、前記第２のプログラムの実行過程における前記呼び出し命令の実行時に、解析要求で指定された情報のうちの非リアルタイムに解析する情報以外情報の解析、非リアルタイムに解析する情報の第２の処理手段への解析依頼、前記第１のプログラム

の処理の実行、および呼び出しに対する応答を、前記第１の処理手段にリアルタイムに実行させ、非リアルタイムに解析する情報の解析を、前記第２の処理手段に非リアルタイムに実行させる、

処理を実行させることを特徴とするプログラム。

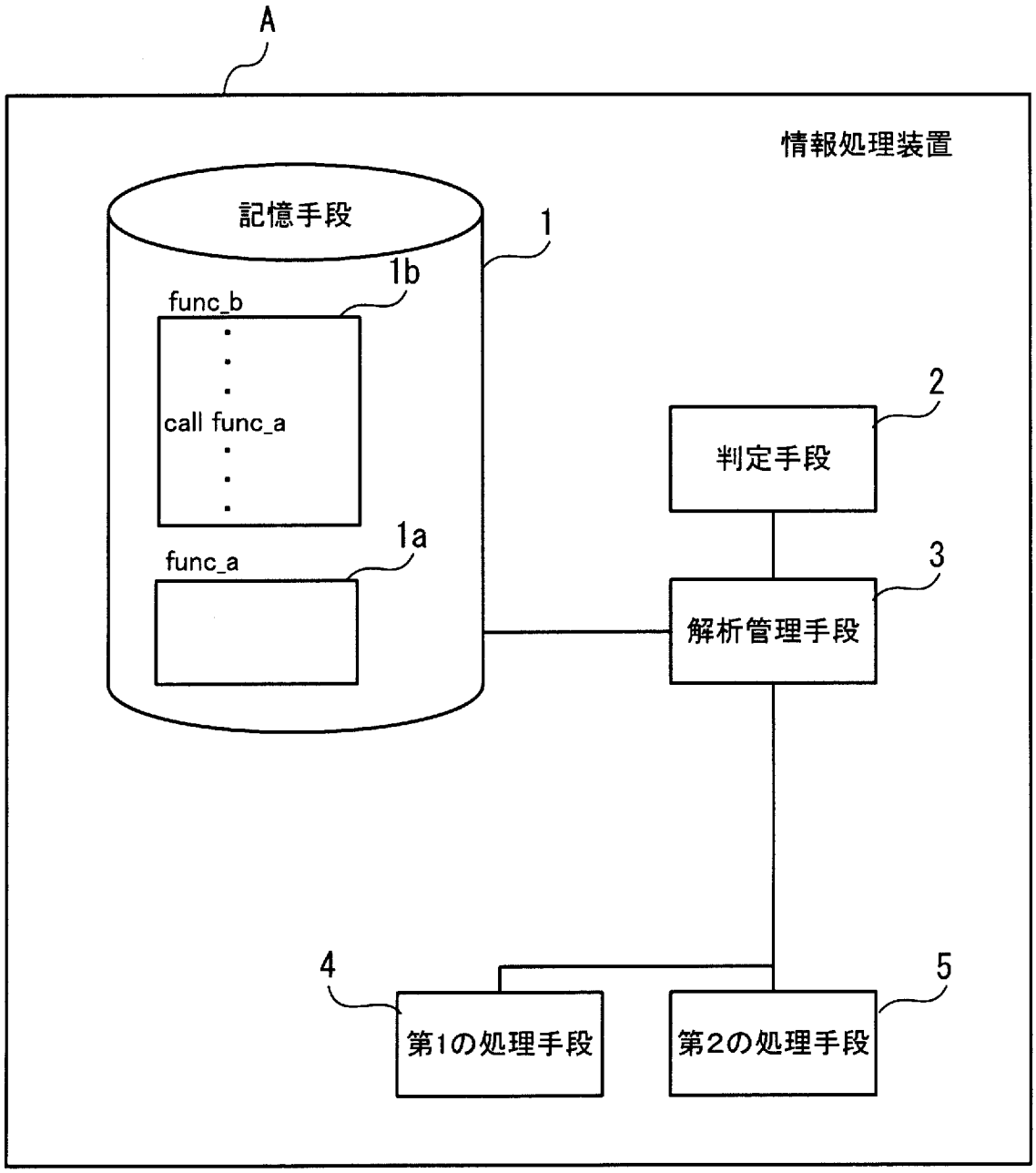
[請求項11] コンピュータが、

第１のプログラムの呼び出しに関する情報の解析要求に基づき、該解析要求で解析対象に指定された情報のすべての解析をリアルタイムに実行する第１の手法と、該解析要求で指定された情報のうちの一部の情報の解析を非リアルタイムに実行し、非リアルタイムに解析する情報以外情報の解析をリアルタイムに実行する第２の手法とのうち、前記第１のプログラムの呼び出しが行われたときの呼び出し元への応答時間が短い方の手法を判定し、

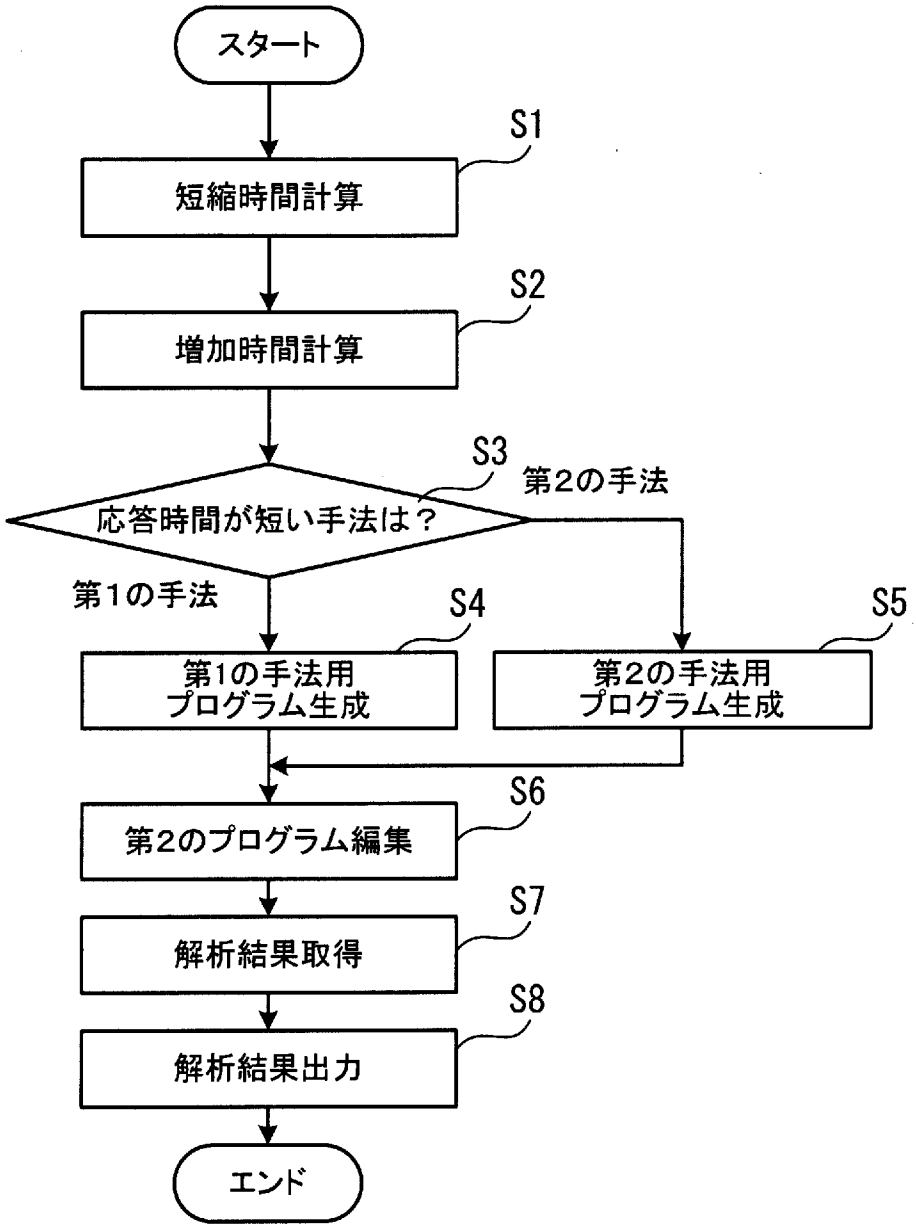
前記第１の手法の方が応答時間が短い場合、前記第１のプログラムの呼び出し命令が記述された第２のプログラムの実行過程における該呼び出し命令の実行時に、解析要求で指定された情報のすべての情報の解析、前記第１のプログラムの処理の実行、および呼び出しに対する応答を、第１の処理手段にリアルタイムに実行させ、前記第２の手法の方が応答時間が短い場合、前記第２のプログラムの実行過程における前記呼び出し命令の実行時に、解析要求で指定された情報のうちの非リアルタイムに解析する情報以外情報の解析、非リアルタイムに解析する情報の第２の処理手段への解析依頼、前記第１のプログラムの処理の実行、および呼び出しに対する応答を、前記第１の処理手段にリアルタイムに実行させ、非リアルタイムに解析する情報の解析を、前記第２の処理手段に非リアルタイムに実行させる、

ことを特徴とする解析方法。

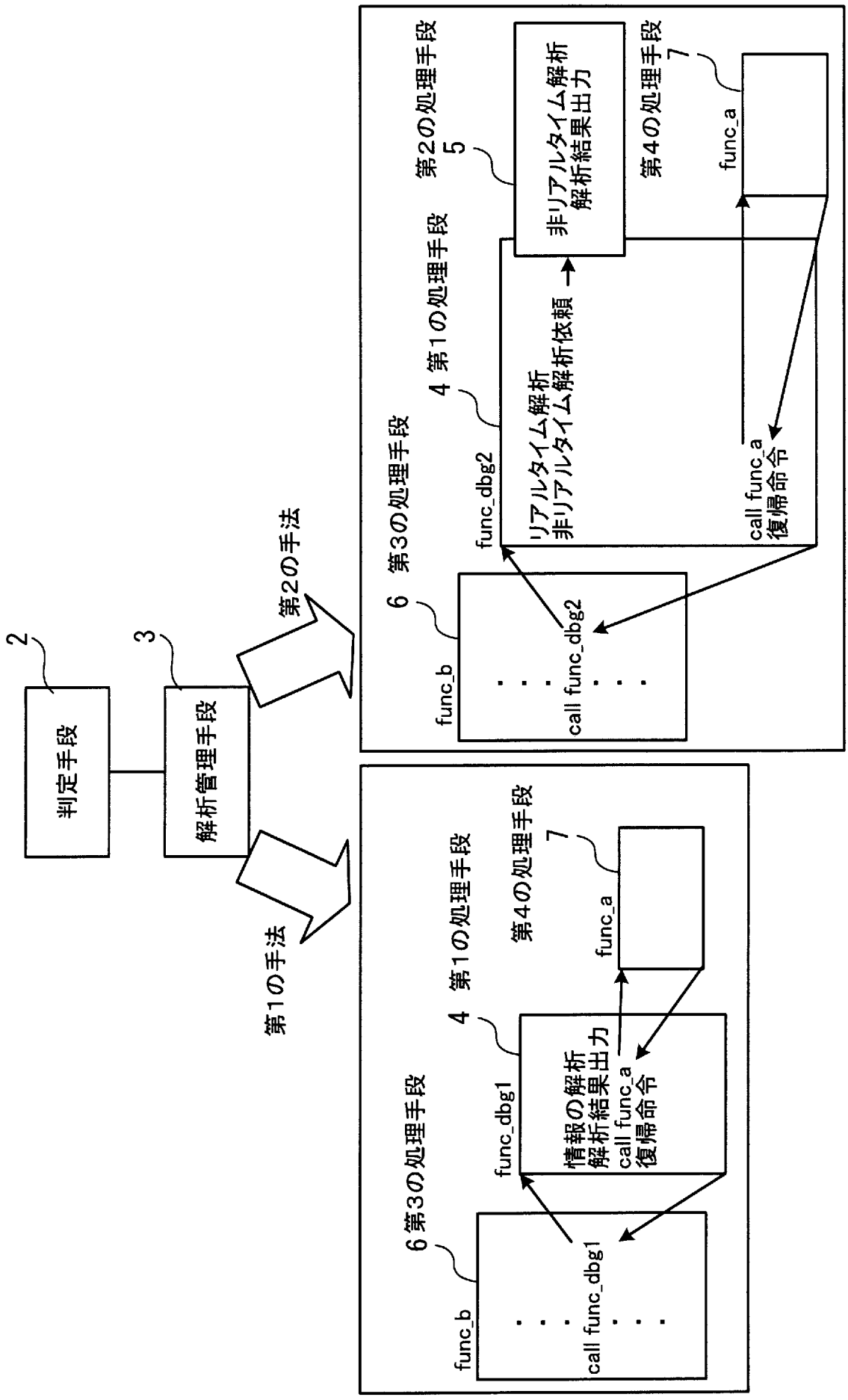
[図1]


A
情報処理装置
記憶手段
1
1b
func_b
call func_a
1a
func_a
2
判定手段
3
解析管理手段
4
第1の処理手段
5
第2の処理手段

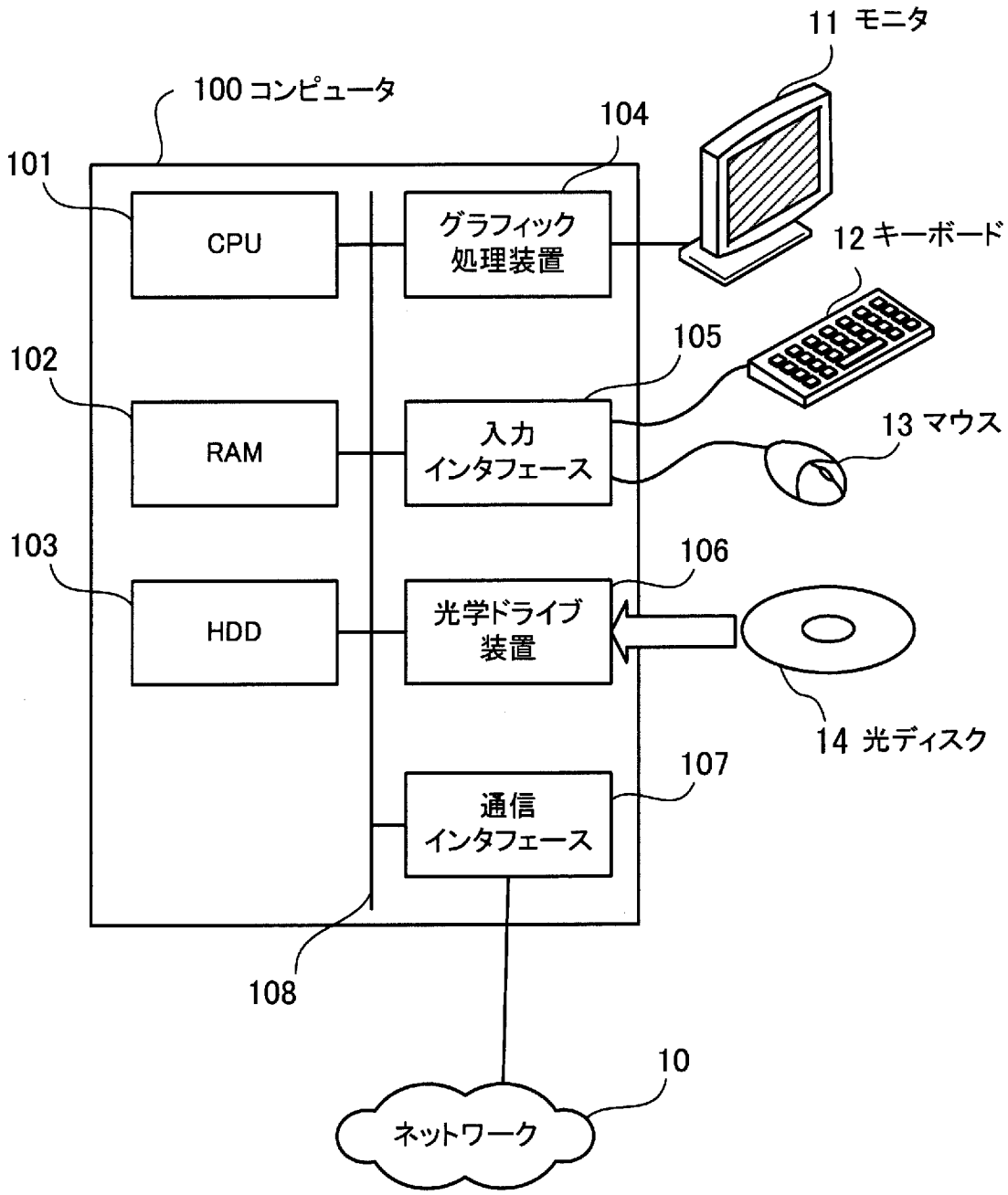
[図2]


スタート
S1
短縮時間計算
S2
増加時間計算
S3
応答時間が短い手法は？
第2の手法
第1の手法
S4
第1の手法用
プログラム生成
S5
第2の手法用
プログラム生成
S6
第2のプログラム編集
S7
解析結果取得
S8
解析結果出力
エンド

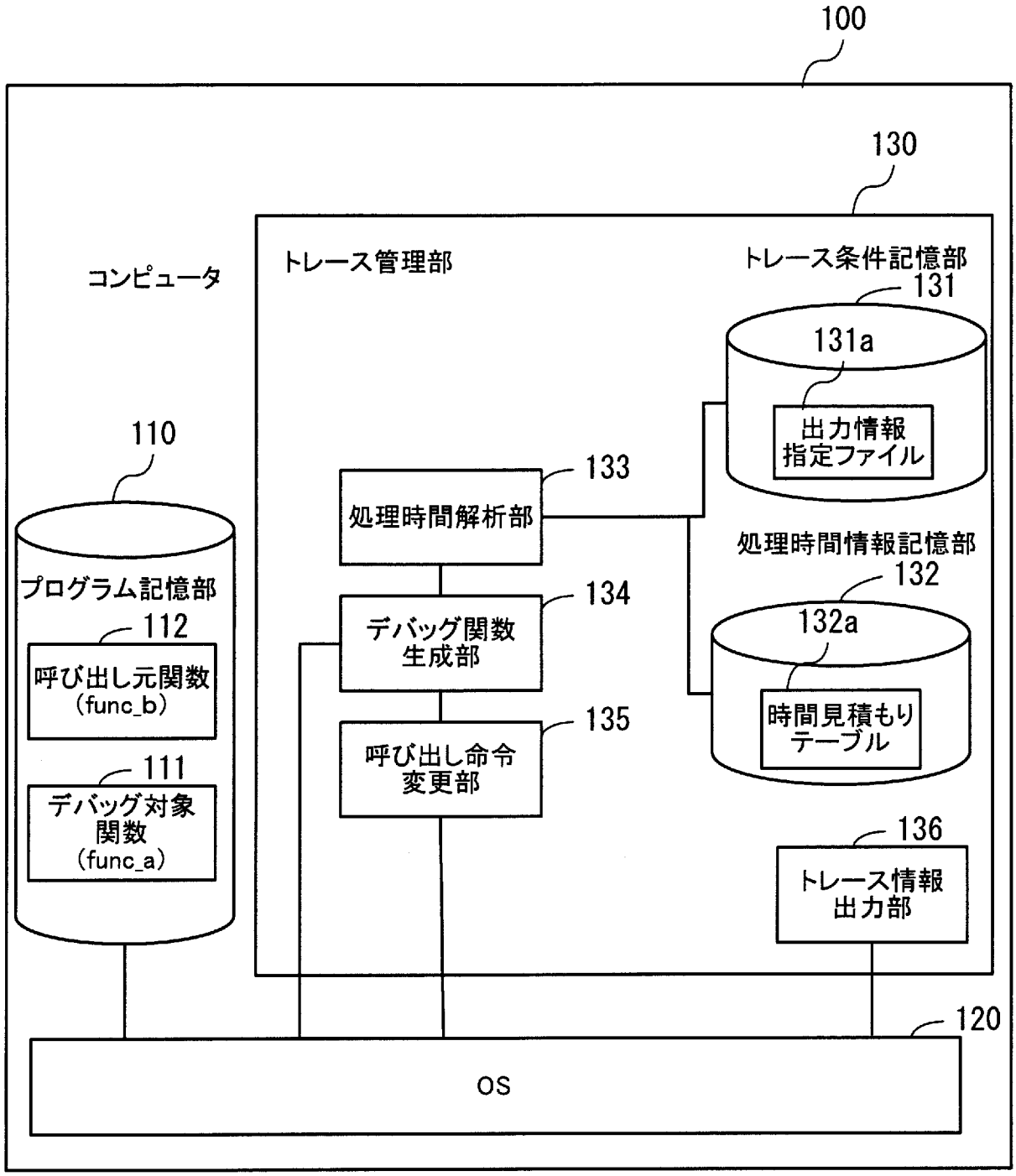
[図3]


2
判定手段
3
解析管理手段
第2の手法
第1の手法
6 第3の処理手段
func_b
call func_dbg2
4 第1の処理手段
func_dbg2
リアルタイム解析
非リアルタイム解析依頼
第2の処理手段
5
非リアルタイム解析
解析結果出力
第4の処理手段
7
func_a
call func_a
復帰命令
6第3の処理手段
func_b
call func_dbg1
4 第1の処理手段
func_dbg1
情報の解析
解析結果出力
call func_a
復帰命令
第4の処理手段
7
func_a

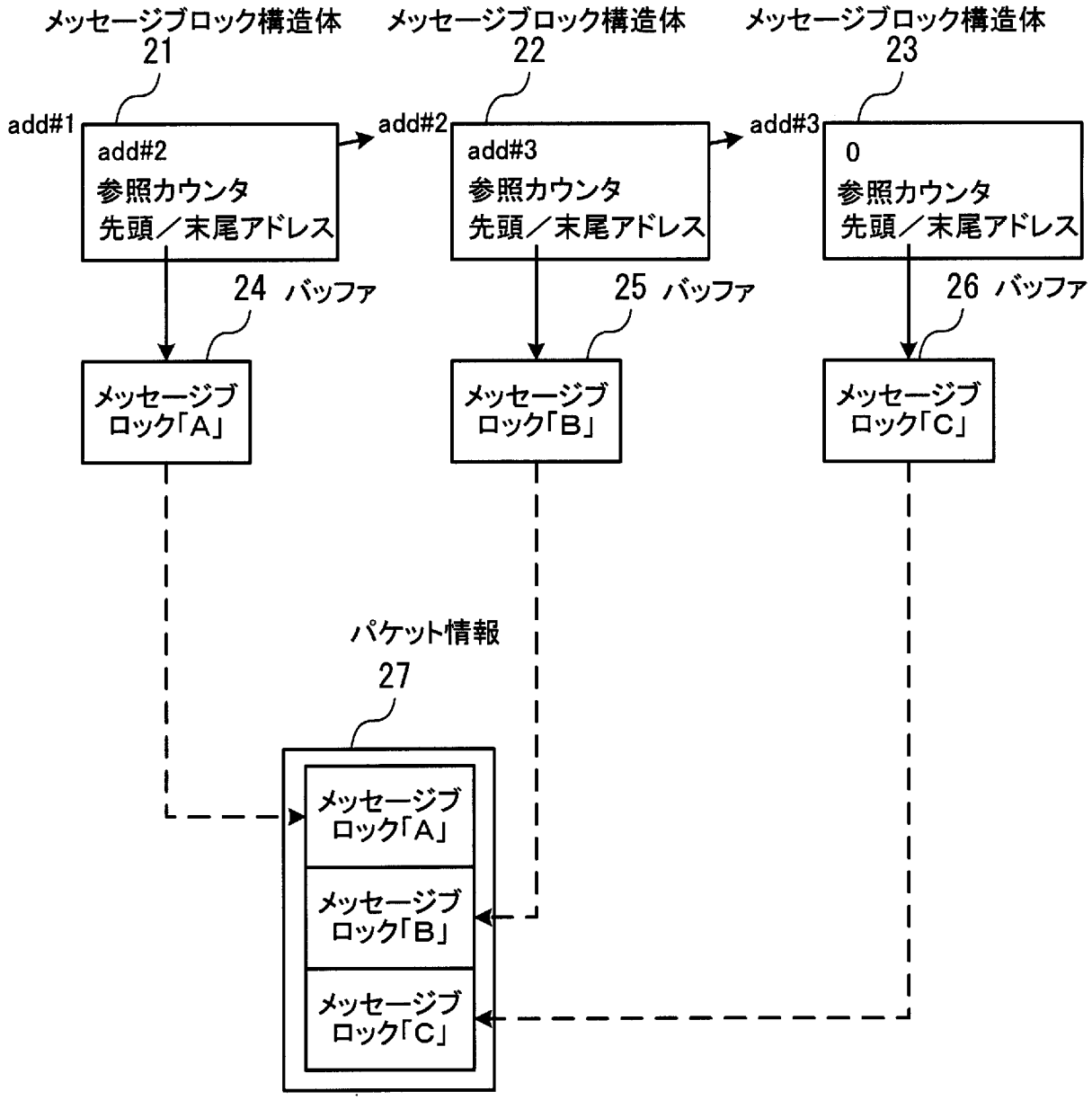
[図4]


11 モニタ
100 コンピュータ
104
101
CPU
グラフィック
処理装置
12 キーボード
105
102
RAM
入力
インタフェース
13 マウス
103
106
HDD
光学ドライブ
装置
14 光ディスク
107
通信
インタフェース
108
10
ネットワーク

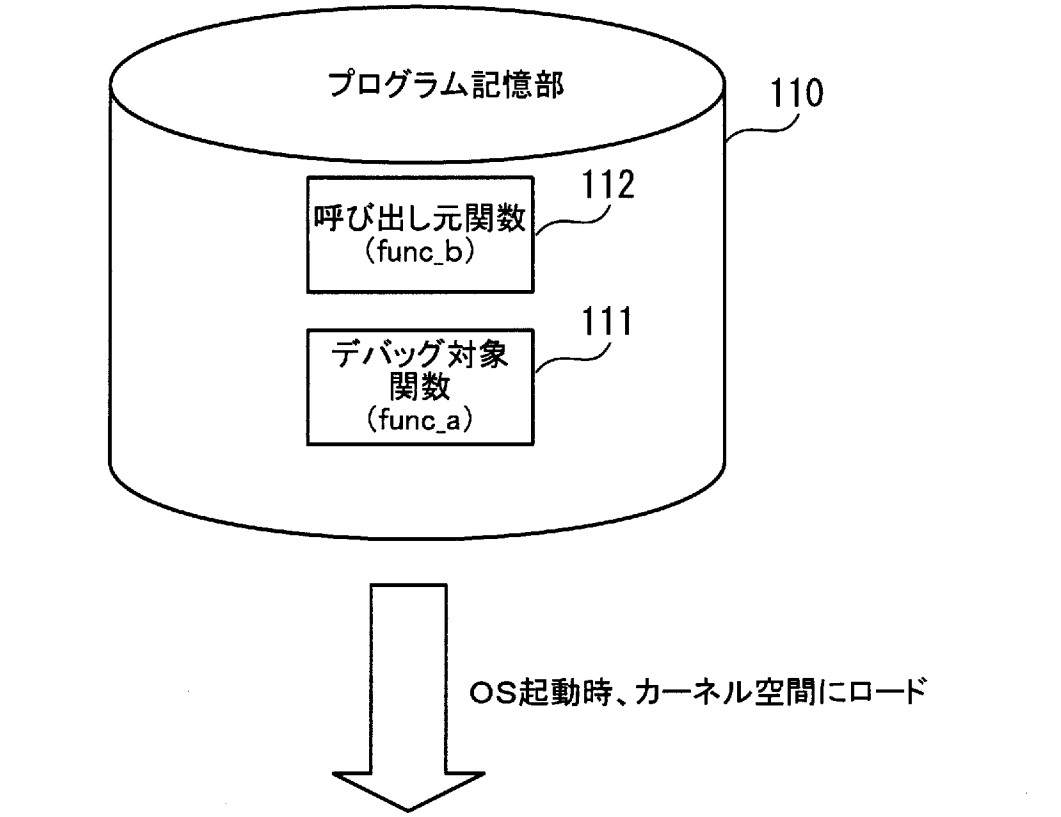
[図5]

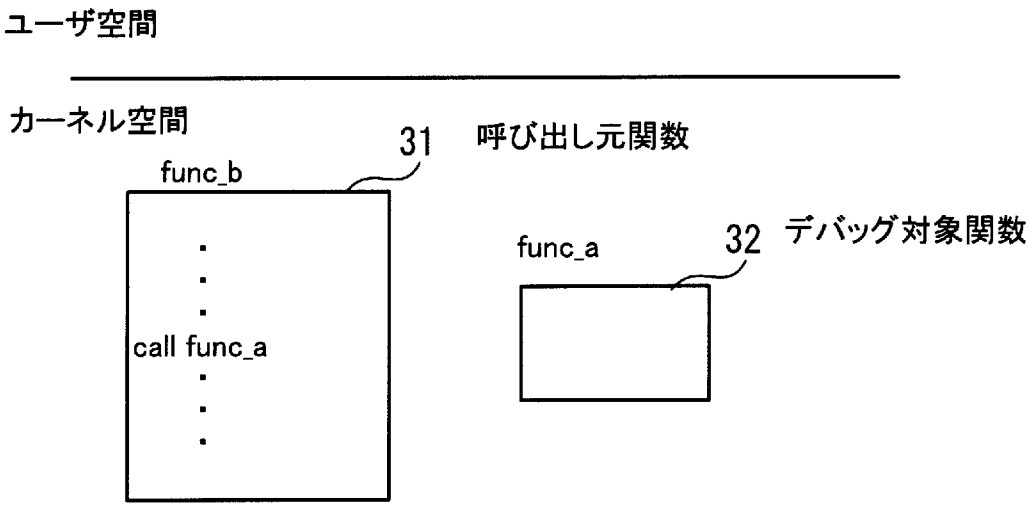

100
130
コンピュータ
トレース管理部
トレース条件記憶部
131
131a
出力情報
指定ファイル
110
133
処理時間解析部
処理時間情報記憶部
132
プログラム記憶部
112
134
132a
デバッグ関数
生成部
呼び出し元関数
(func_b)
135
時間見積もり
テーブル
111
呼び出し命令
変更部
デバッグ対象
関数
(func_a)
136
トレース情報
出力部
120
OS

[図6]

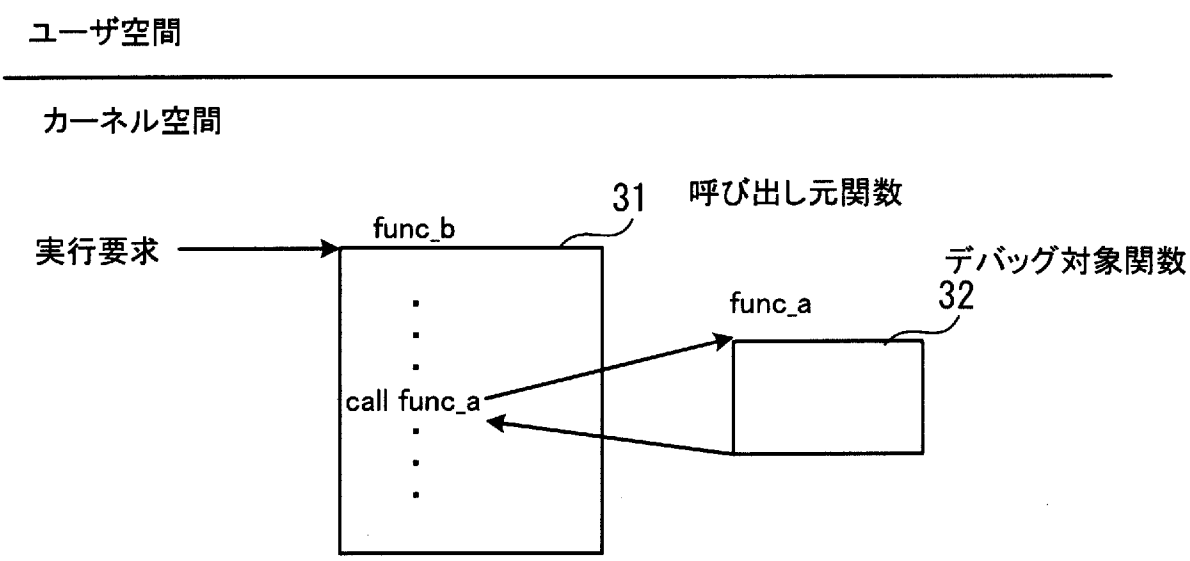

メッセージブロック構造体
21
メッセージブロック構造体
22
メッセージブロック構造体
23
add#1
add#2
参照カウンタ
先頭／末尾アドレス
add#2
add#3
参照カウンタ
先頭／末尾アドレス
add#3
0
参照カウンタ
先頭／末尾アドレス
24 バッファ
25 バッファ
26 バッファ
メッセージブロック「A」
メッセージブロック「B」
メッセージブロック「C」
パケット情報
27
メッセージブロック「A」
メッセージブロック「B」
メッセージブロック「C」

[図7]

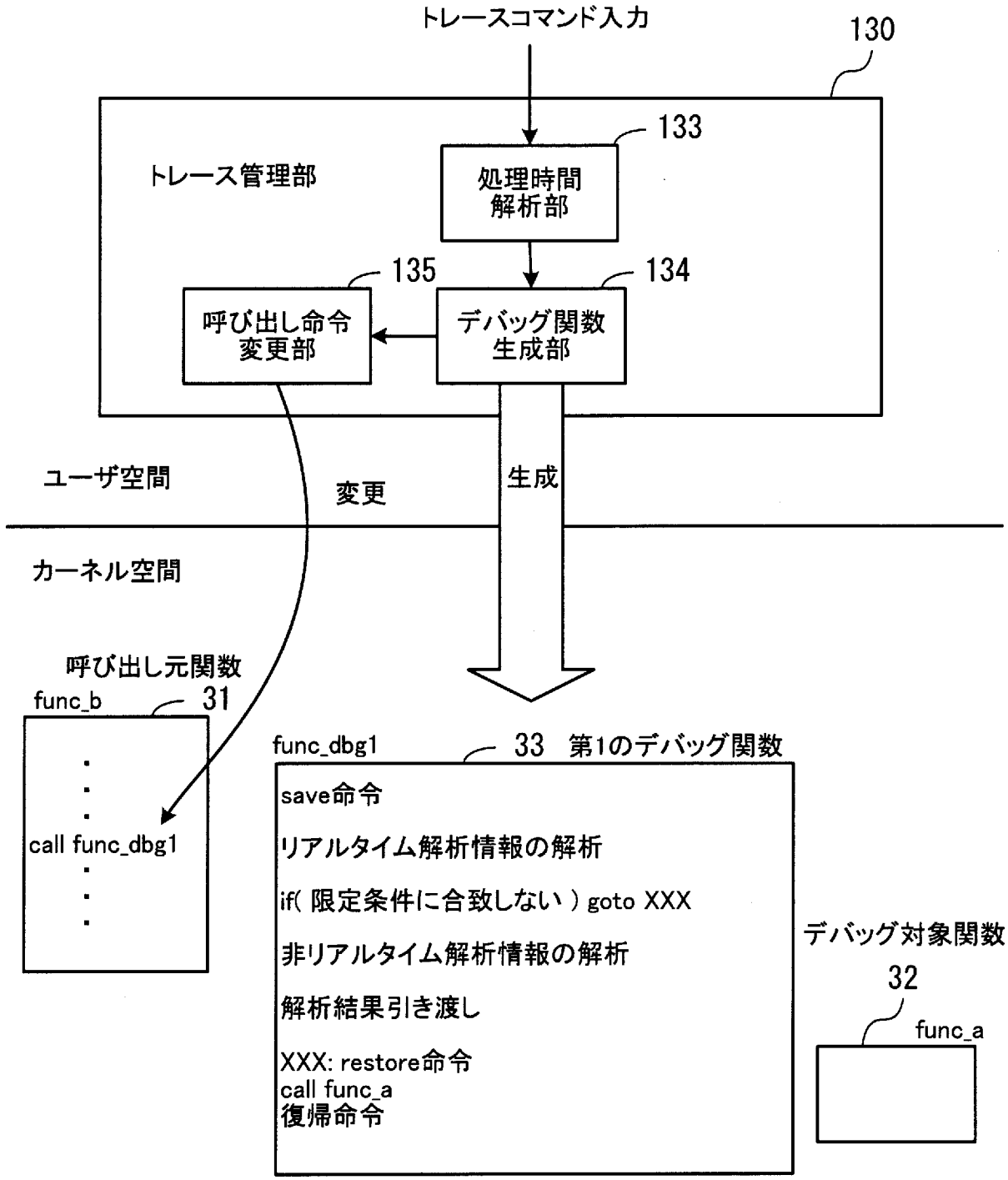

プログラム記憶部
110
呼び出し元関数
(func_b)
112
デバッグ対象
関数
(func_a)
111
OS起動時、カーネル空間にロード


ユーザ空間
カーネル空間
31
呼び出し元関数
func_b
call func_a
func_a
32
デバッグ対象関数

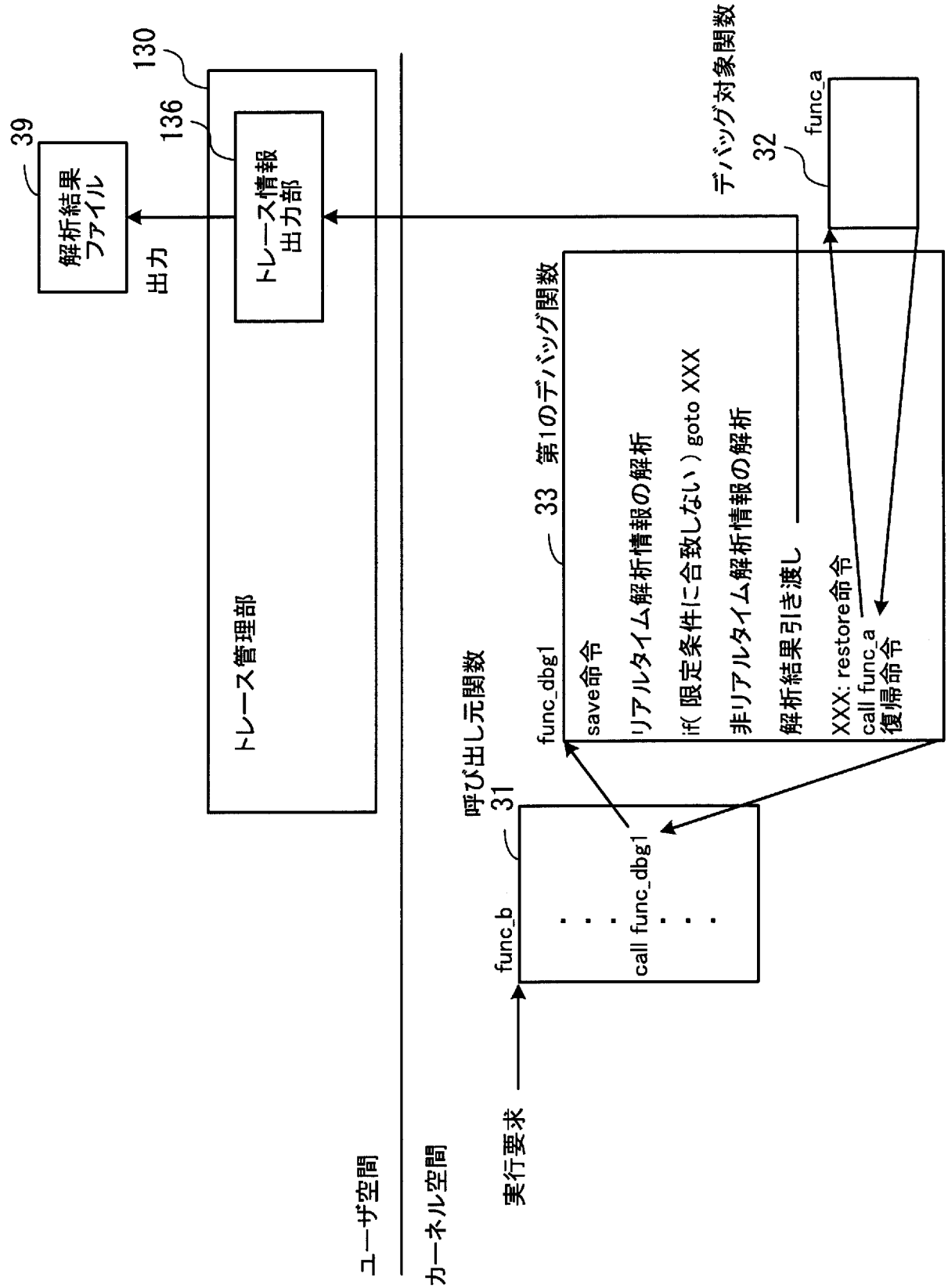
[図8]


ユーザ空間
カーネル空間
実行要求
func_b
31
呼び出し元関数
call func_a
func_a
32
デバッグ対象関数

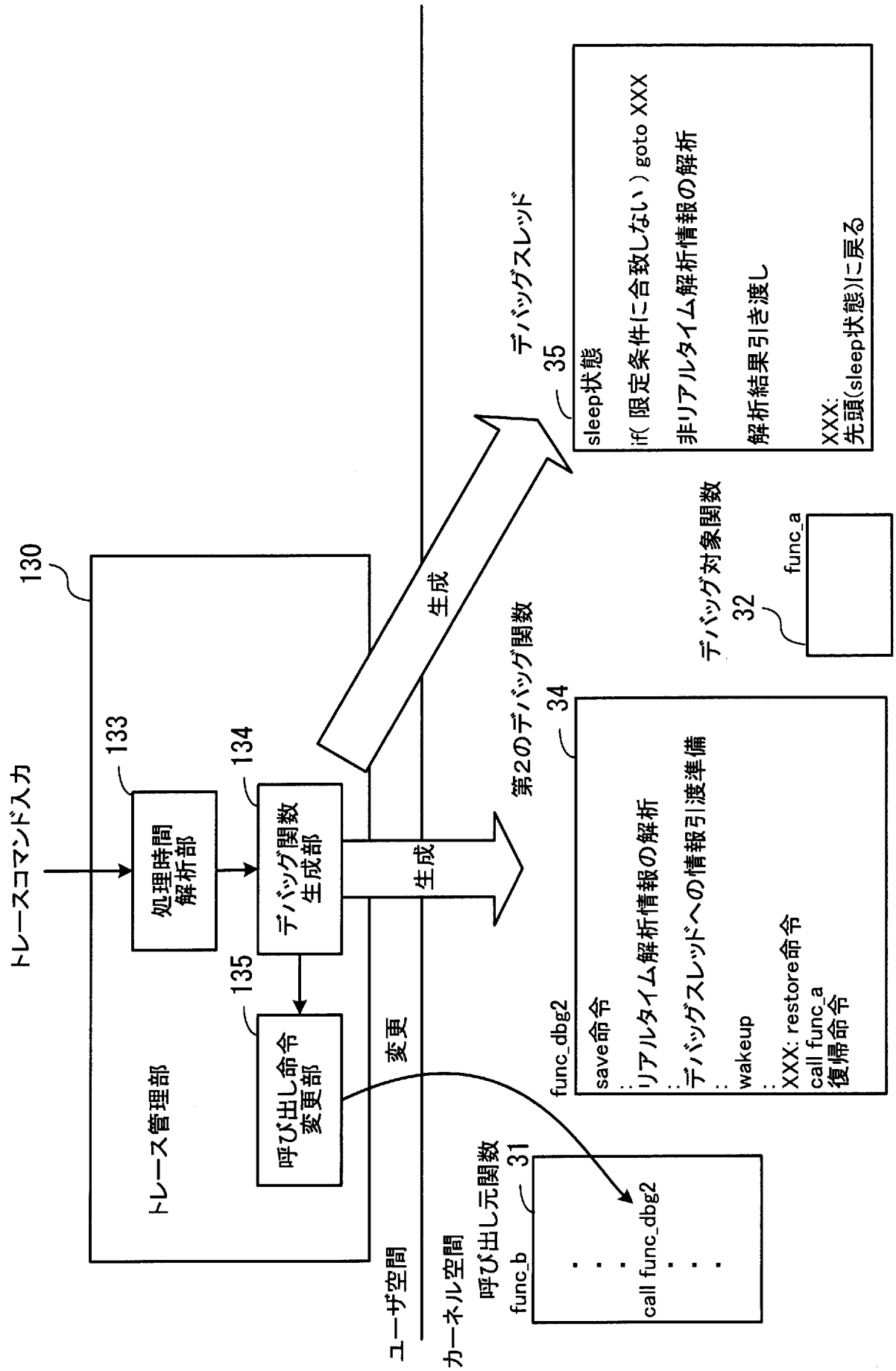
[図9]


トレースコマンド入力
130
トレース管理部
133
処理時間
解析部
135
134
呼び出し命令
変更部
デバッグ関数
生成部
ユーザ空間
変更
生成
カーネル空間
呼び出し元関数
func_b
31
call func_dbg1
func_dbg1
33 第1のデバッグ関数
save命令
リアルタイム解析情報の解析
if(限定条件に合致しない) goto XXX
非リアルタイム解析情報の解析
解析結果引き渡し
XXX: restore命令
call func_a
復帰命令
デバッグ対象関数
32
func_a

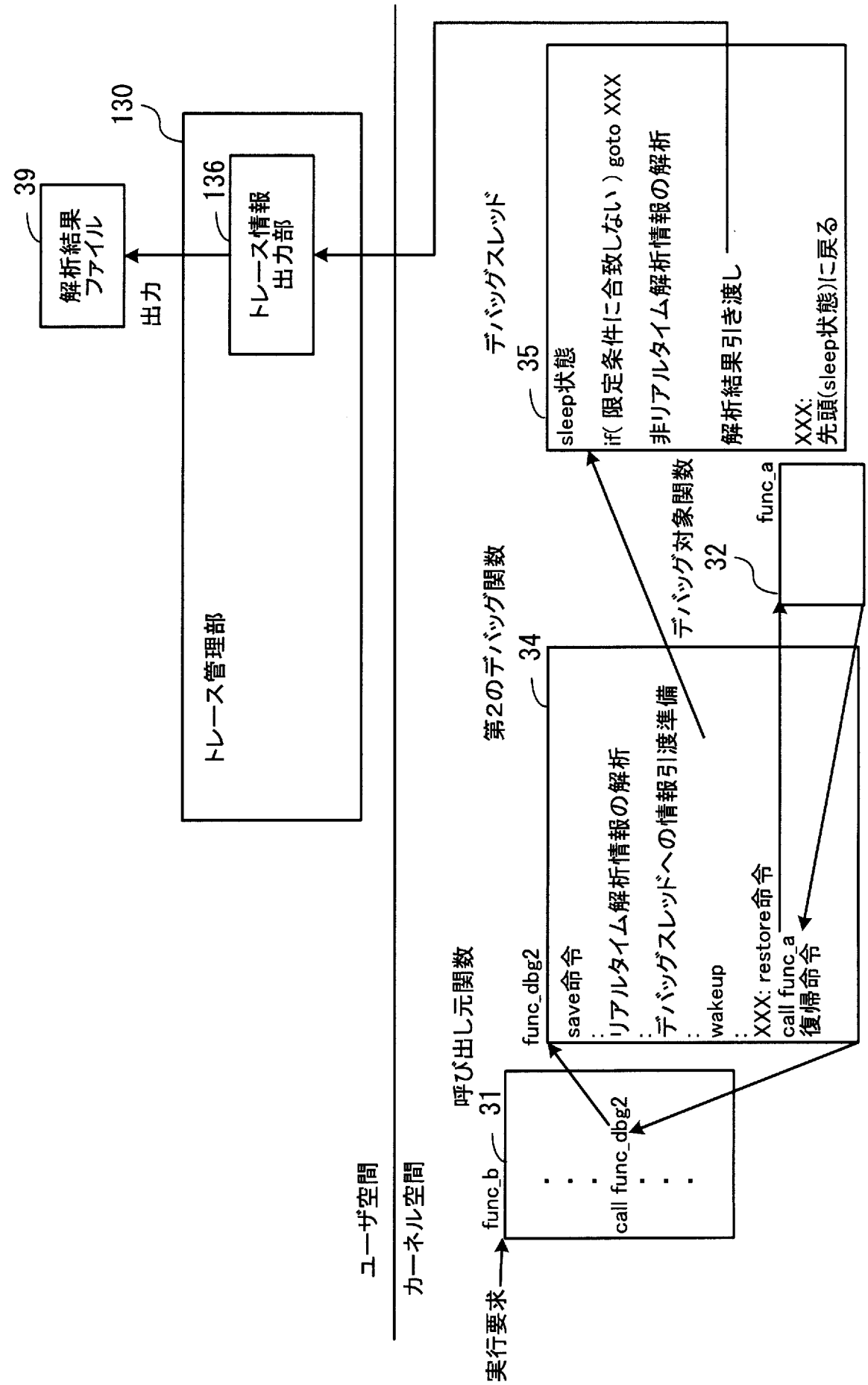
[図10]


130
136
39
解析結果ファイル
出力
トレース情報出力部
トレース管理部
ユーザ空間
カーネル空間
実行要求
呼び出し元関数
31
func_b
call func_dbg1
func_dbg1
33 第1のデバッグ関数
save命令
リアルタイム解析情報の解析
if(限定条件に合致しない) goto XXX
非リアルタイム解析情報の解析
解析結果引き渡し
XXX: restore命令
call func_a
復帰命令
デバッグ対象関数
32
func_a

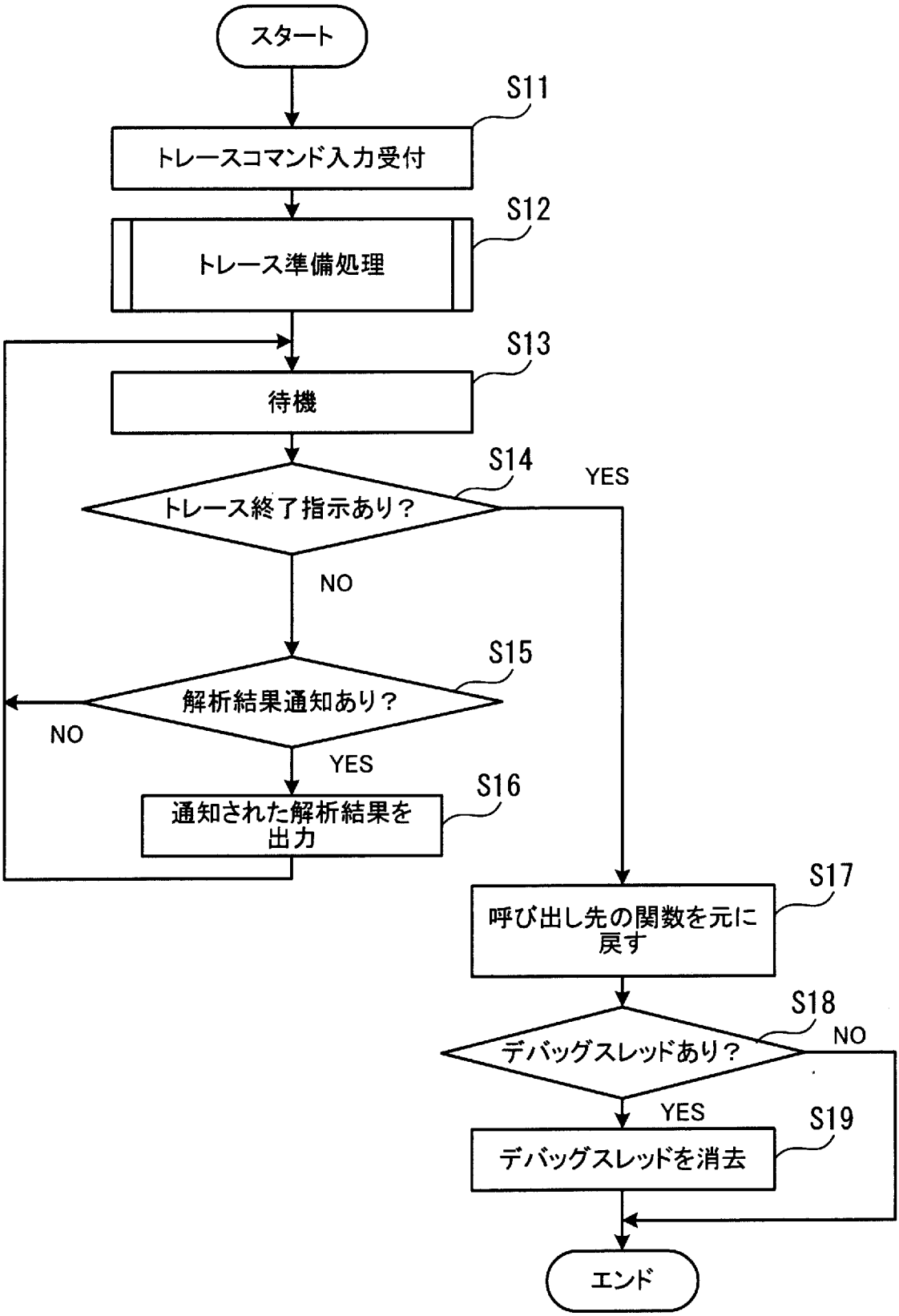
[図11]


130
トレース管理部
処理時間解析部
133
デバッグ関数生成部
134
呼び出し命令変更部
135
トレースコマンド入力
生成
生成
変更
ユーザ空間
カーネル空間
呼び出し元関数
func_b
31
call func_dbg2
第2のデバッグ関数
34
func_dbg2
save命令
リアルタイム解析情報の解析
デバッグスレッドへの情報引渡準備
wakeup
XXX: restore命令
call func_a
復帰命令
デバッグスレッド
35
sleep状態
if(限定条件に合致しない) goto XXX
非リアルタイム解析情報の解析
解析結果引き渡し
XXX:
先頭(sleep状態)に戻る
デバッグ対象関数
32
func_a

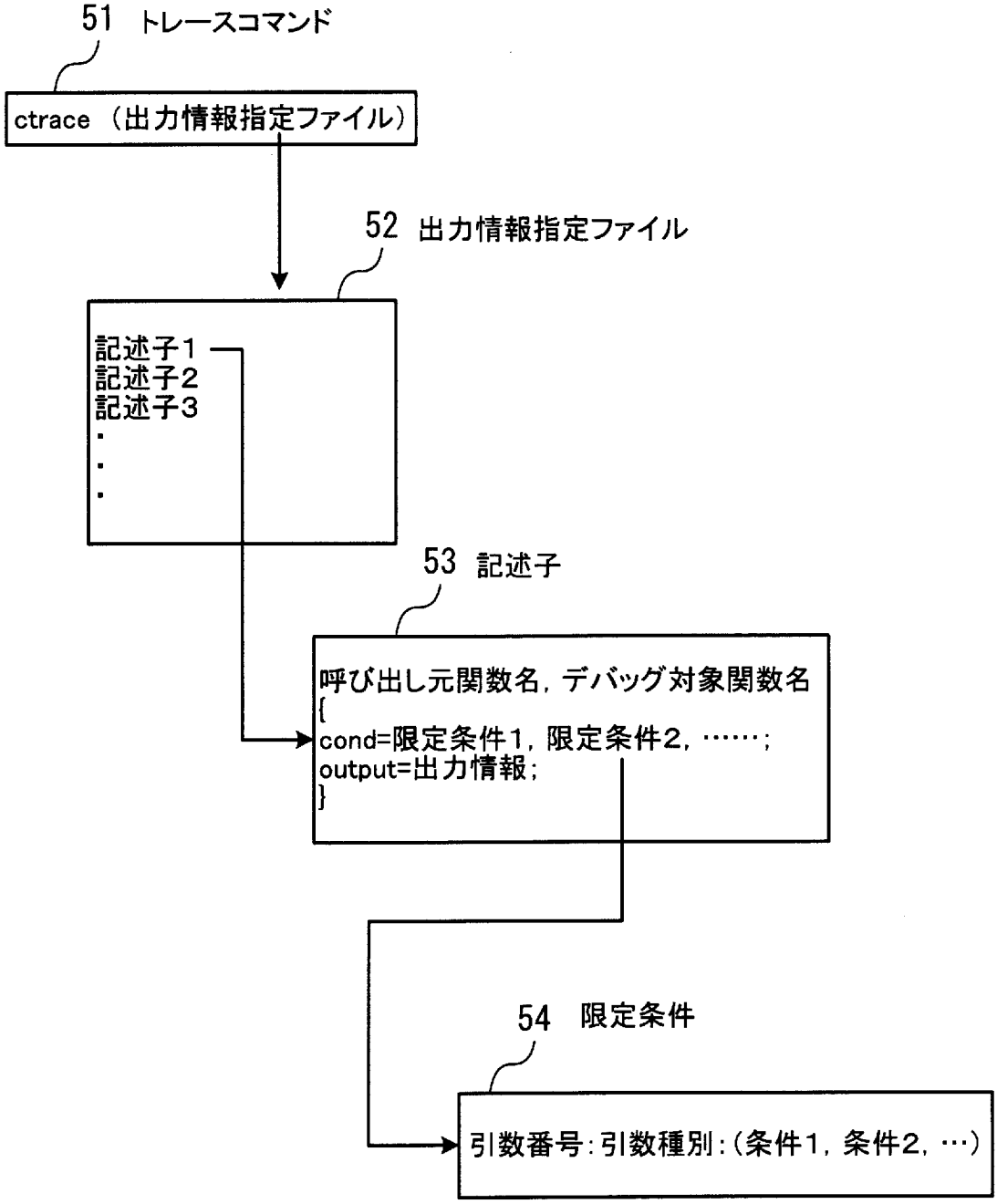
[図12]


130
39
解析結果ファイル
136
トレース情報出力部
出力
トレース管理部
デバッグスレッド
35
sleep状態
if(限定条件に合致しない) goto XXX
非リアルタイム解析情報の解析
解析結果引き渡し
XXX:
先頭(sleep状態)に戻る
第2のデバッグ関数
34
func_dbg2
save命令
:
リアルタイム解析情報の解析
:
デバッグスレッドへの情報引渡準備
:
wakeup
:
XXX: restore命令
call func_a
復帰命令
デバッグ対象関数
32
func_a
呼び出し元関数
31
func_b
call func_dbg2
ユーザ空間
カーネル空間
実行要求

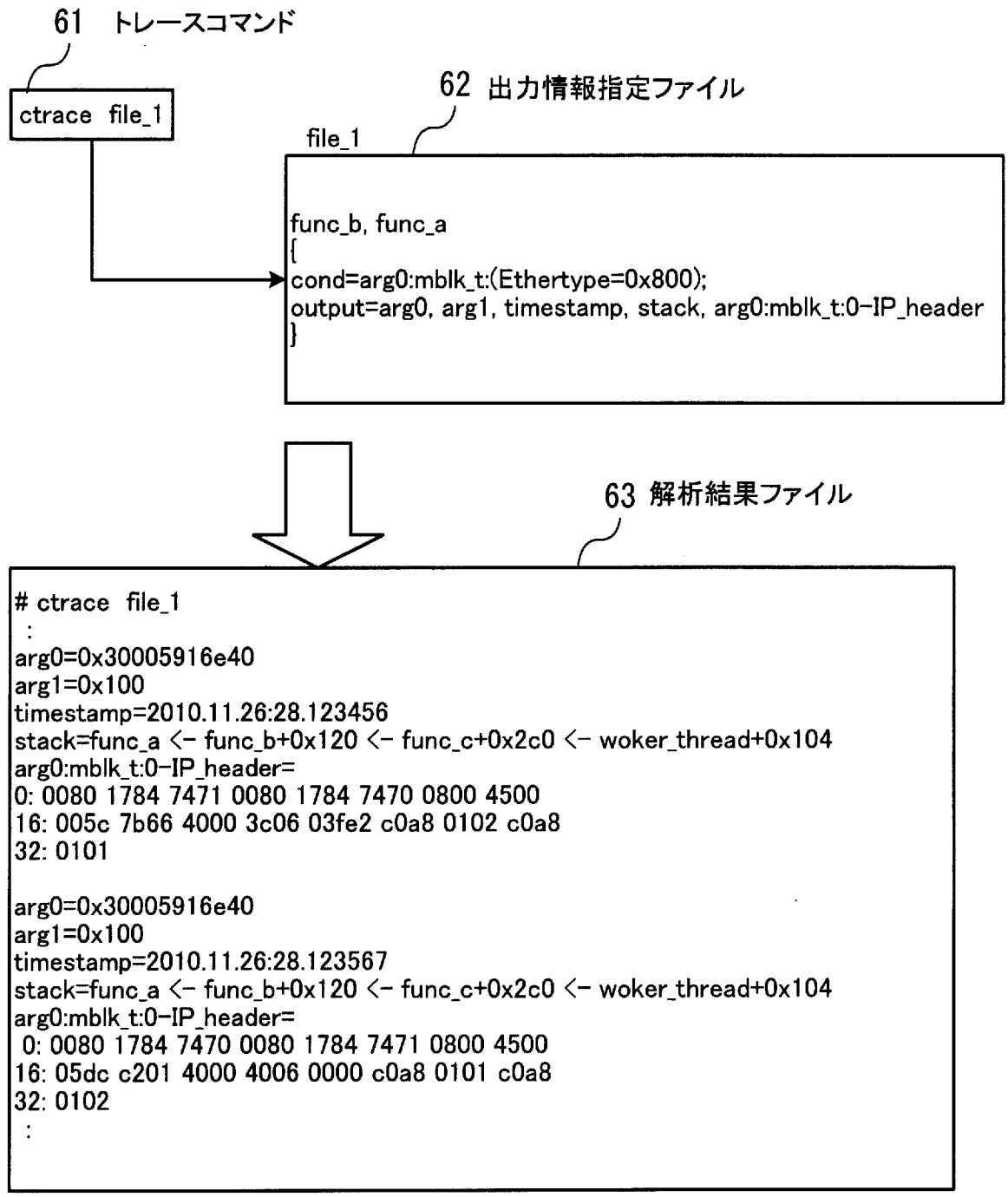
[図13]


スタート
S11
トレースコマンド入力受付
S12
トレース準備処理
S13
待機
S14
トレース終了指示あり？
YES
NO
S15
解析結果通知あり？
NO
YES
S16
通知された解析結果を出力
S17
呼び出し先の関数を元に戻す
S18
デバッグスレッドあり？
NO
YES
S19
デバッグスレッドを消去
エンド

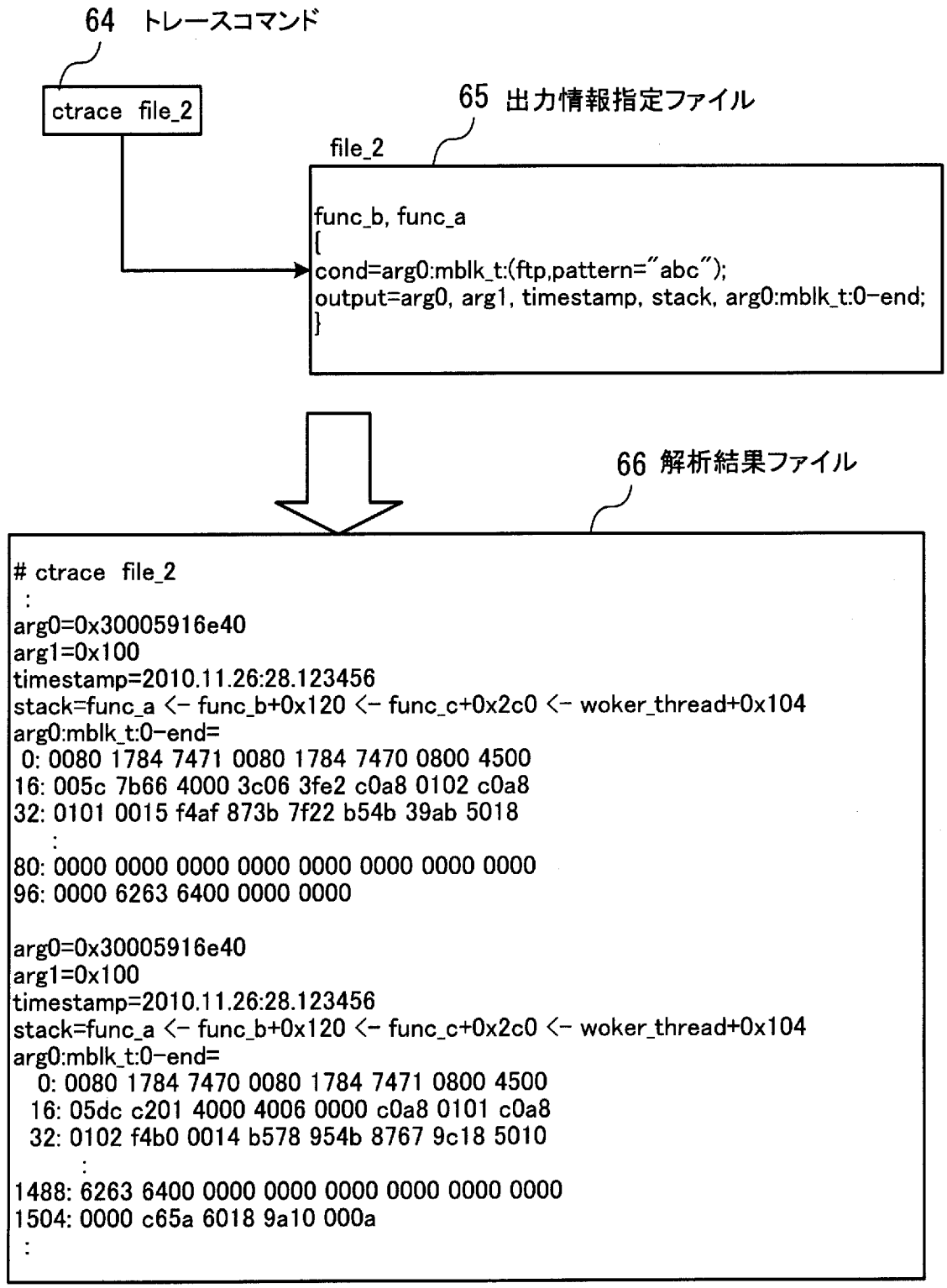
[図14]


51 トレースコマンド
ctrace （出力情報指定ファイル）
52 出力情報指定ファイル
記述子1
記述子2
記述子3
53 記述子
呼び出し元関数名，デバッグ対象関数名
{
cond=限定条件1，限定条件2，……；
output=出力情報；
}
54 限定条件
引数番号：引数種別：（条件1，条件2，…）

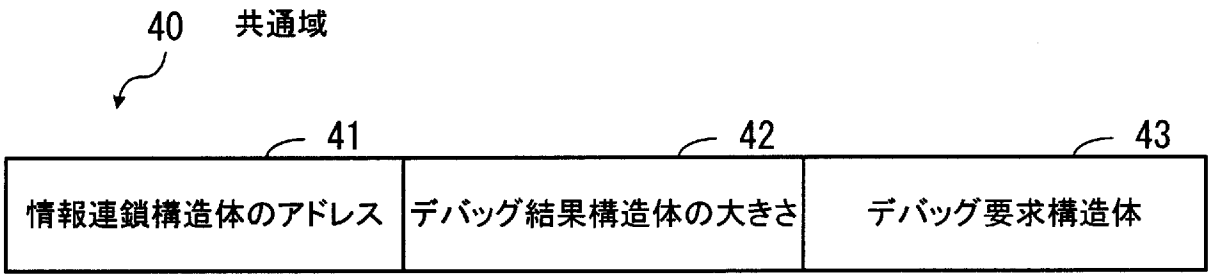
[図15]


61 トレースコマンド
ctrace file_1
62 出力情報指定ファイル
file_1
func_b, func_a
{
cond=arg0:mblk_t:(Ethertype=0x800);
output=arg0, arg1, timestamp, stack, arg0:mblk_t:0-IP_header
}
63 解析結果ファイル
ctrace file_1
:
arg0=0x30005916e40
arg1=0x100
timestamp=2010.11.26:28.123456
stack=func_a <- func_b+0x120 <- func_c+0x2c0 <- woker_thread+0x104
arg0:mblk_t:0-IP_header=
0: 0080 1784 7471 0080 1784 7470 0800 4500
16: 005c 7b66 4000 3c06 03fe2 c0a8 0102 c0a8
32: 0101
arg0=0x30005916e40
arg1=0x100
timestamp=2010.11.26:28.123567
stack=func_a <- func_b+0x120 <- func_c+0x2c0 <- woker_thread+0x104
arg0:mblk_t:0-IP_header=
0: 0080 1784 7470 0080 1784 7471 0800 4500
16: 05dc c201 4000 4006 0000 c0a8 0101 c0a8
32: 0102
:

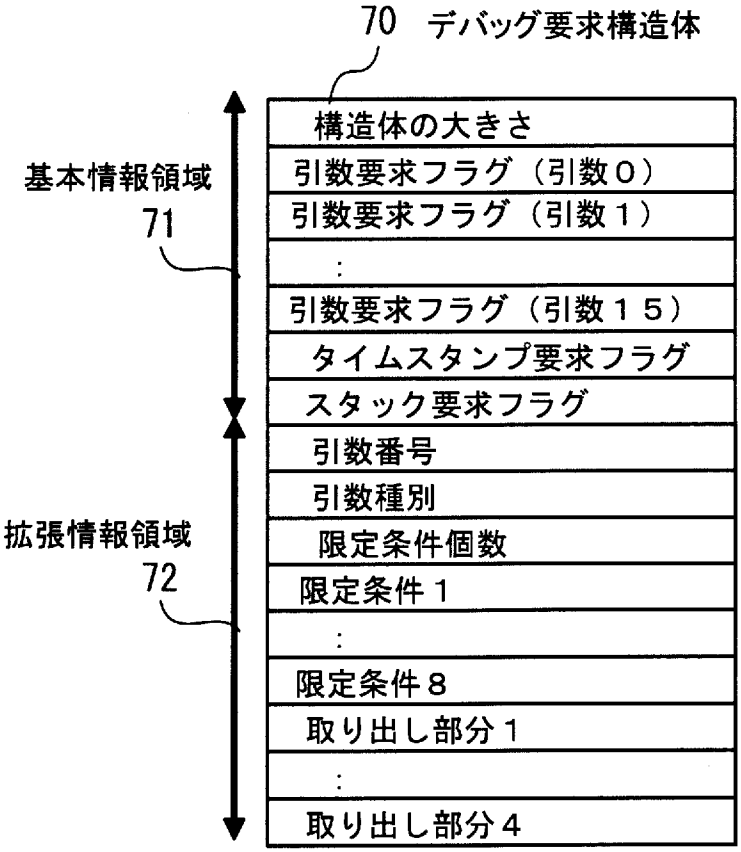
[図16]


64 トレースコマンド
ctrace file_2
65 出力情報指定ファイル
file_2
func_b, func_a
{
cond=arg0:mblk_t:(ftp,pattern="abc");
output=arg0, arg1, timestamp, stack, arg0:mblk_t:0-end;
}
66 解析結果ファイル
ctrace file_2
:
arg0=0x30005916e40
arg1=0x100
timestamp=2010.11.26:28.123456
stack=func_a <- func_b+0x120 <- func_c+0x2c0 <- woker_thread+0x104
arg0:mblk_t:0-end=
0: 0080 1784 7471 0080 1784 7470 0800 4500
16: 005c 7b66 4000 3c06 3fe2 c0a8 0102 c0a8
32: 0101 0015 f4af 873b 7f22 b54b 39ab 5018
:
80: 0000 0000 0000 0000 0000 0000 0000 0000
96: 0000 6263 6400 0000 0000
arg0=0x30005916e40
arg1=0x100
timestamp=2010.11.26:28.123456
stack=func_a <- func_b+0x120 <- func_c+0x2c0 <- woker_thread+0x104
arg0:mblk_t:0-end=
0: 0080 1784 7470 0080 1784 7471 0800 4500
16: 05dc c201 4000 4006 0000 c0a8 0101 c0a8
32: 0102 f4b0 0014 b578 954b 8767 9c18 5010
:
1488: 6263 6400 0000 0000 0000 0000 0000 0000
1504: 0000 c65a 6018 9a10 000a
:

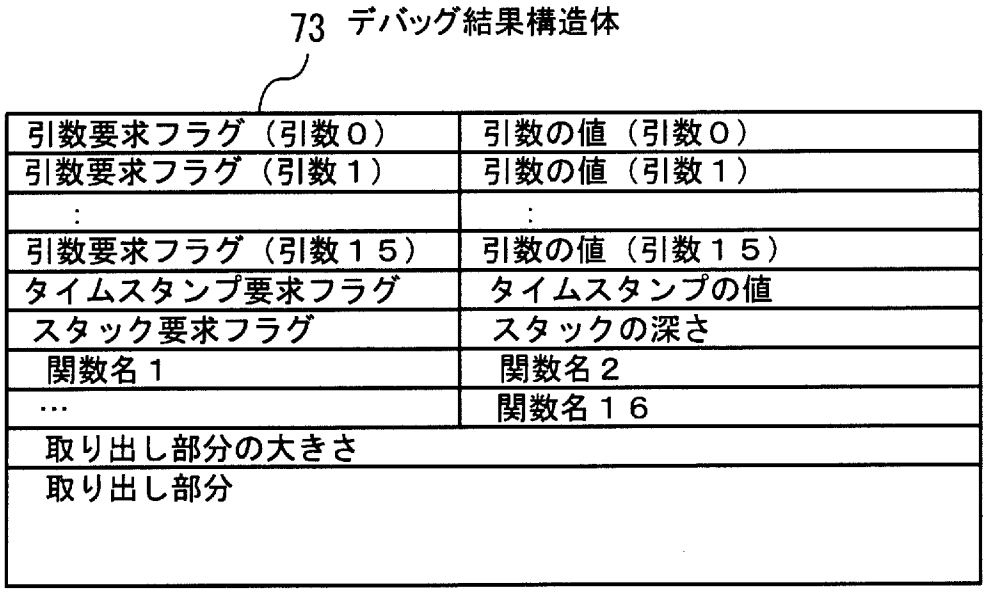
[図17]


40 共通域
41
42
43
情報連鎖構造体のアドレス
デバッグ結果構造体の大きさ
デバッグ要求構造体

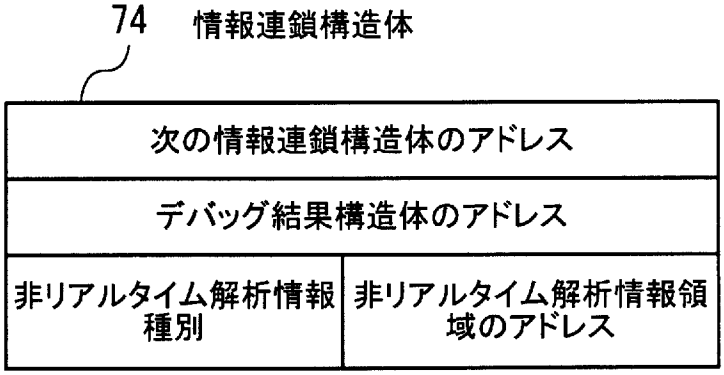
[図18]


70 デバッグ要求構造体
基本情報領域
71
拡張情報領域
72
構造体の大きさ
引数要求フラグ（引数０）
引数要求フラグ（引数１）
：
引数要求フラグ（引数１５）
タイムスタンプ要求フラグ
スタック要求フラグ
引数番号
引数種別
限定条件個数
限定条件１
：
限定条件８
取り出し部分１
：
取り出し部分４

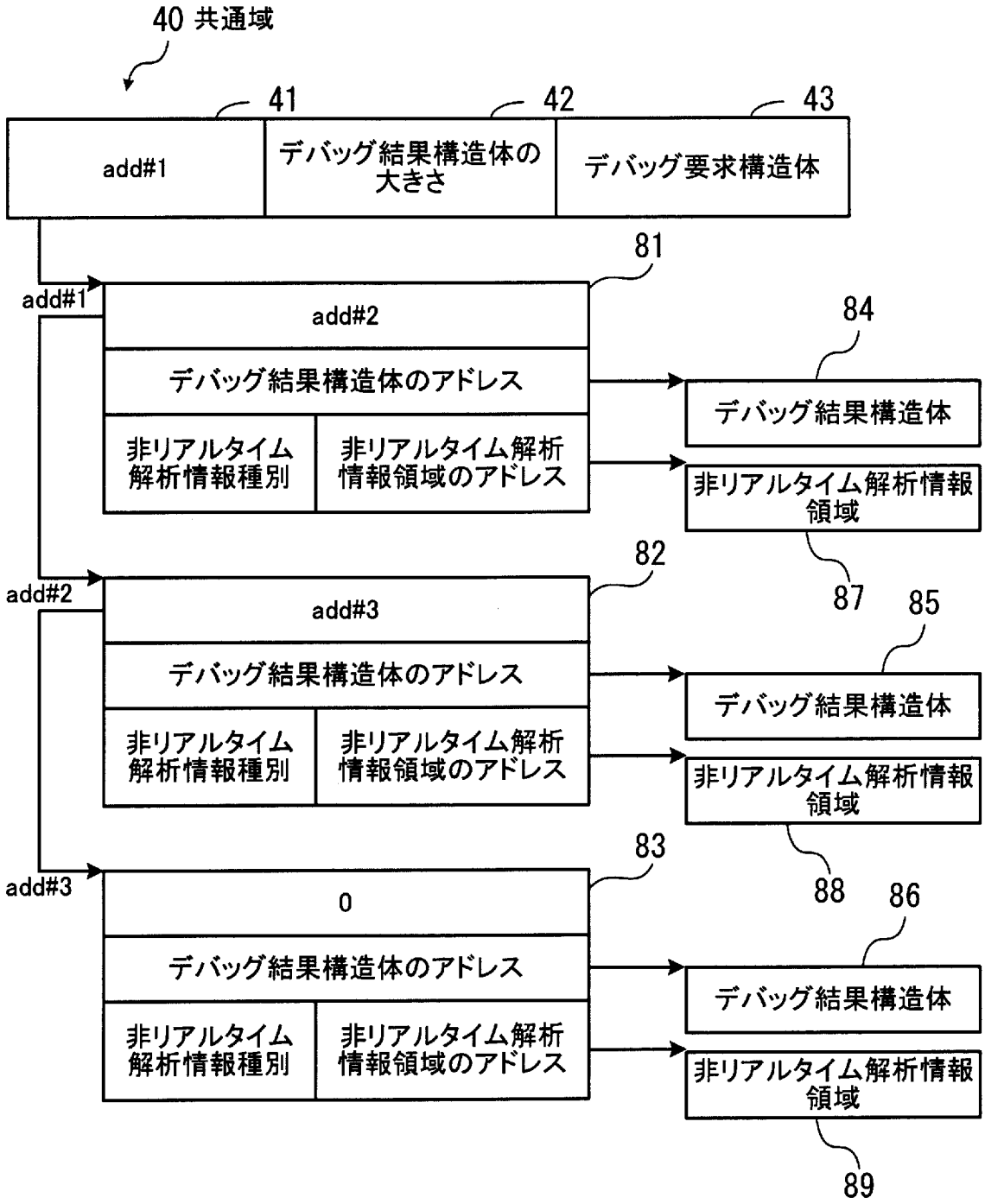
[図19]


73 デバッグ結果構造体
引数要求フラグ（引数０）
引数の値（引数０）
引数要求フラグ（引数１）
引数の値（引数１）
：
：
引数要求フラグ（引数１５）
引数の値（引数１５）
タイムスタンプ要求フラグ
タイムスタンプの値
スタック要求フラグ
スタックの深さ
関数名１
関数名２
…
関数名１６
取り出し部分の大きさ
取り出し部分

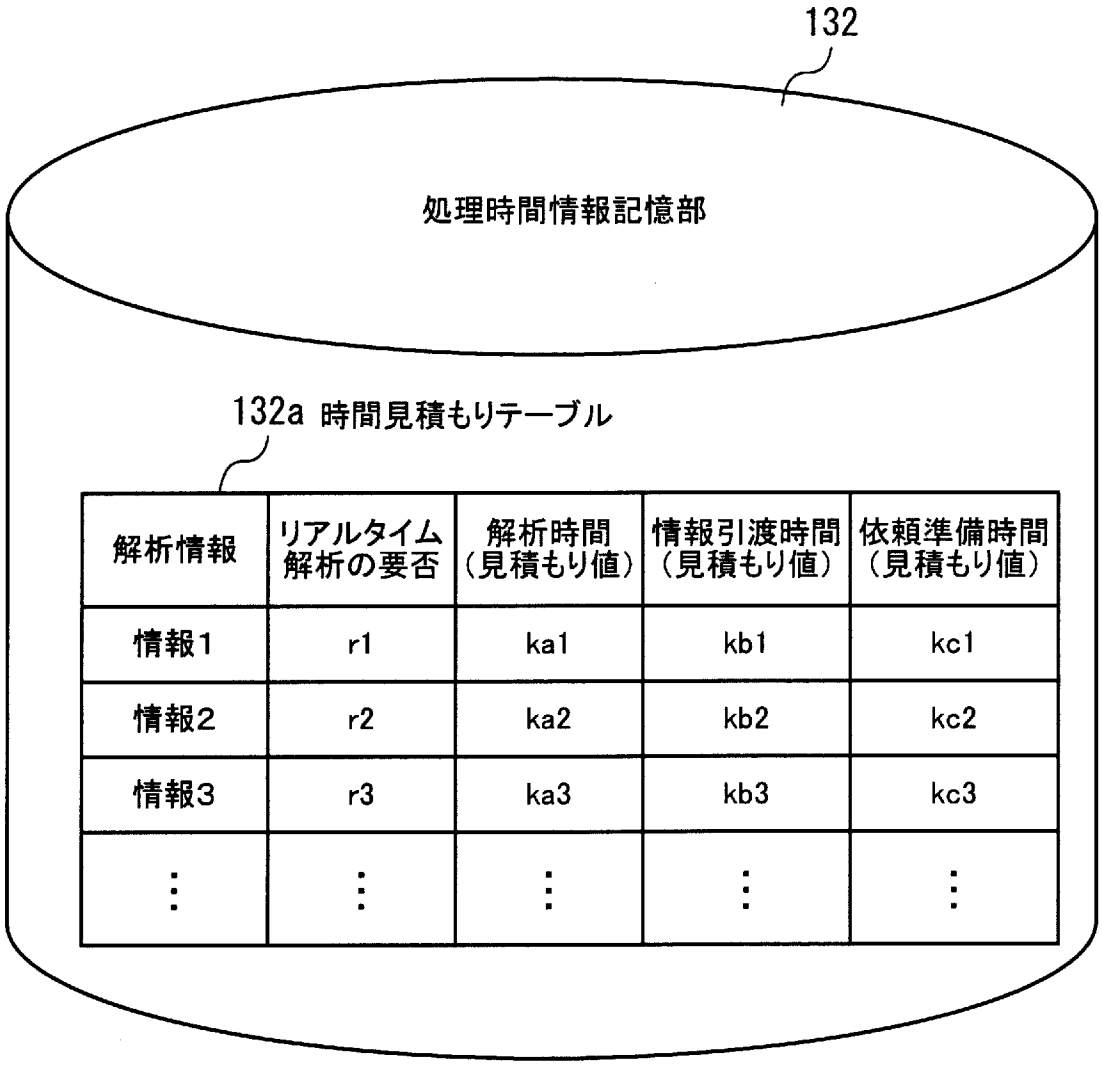
[図20]


74
情報連鎖構造体
次の情報連鎖構造体のアドレス
デバッグ結果構造体のアドレス
非リアルタイム解析情報種別
非リアルタイム解析情報領域のアドレス

[図21]

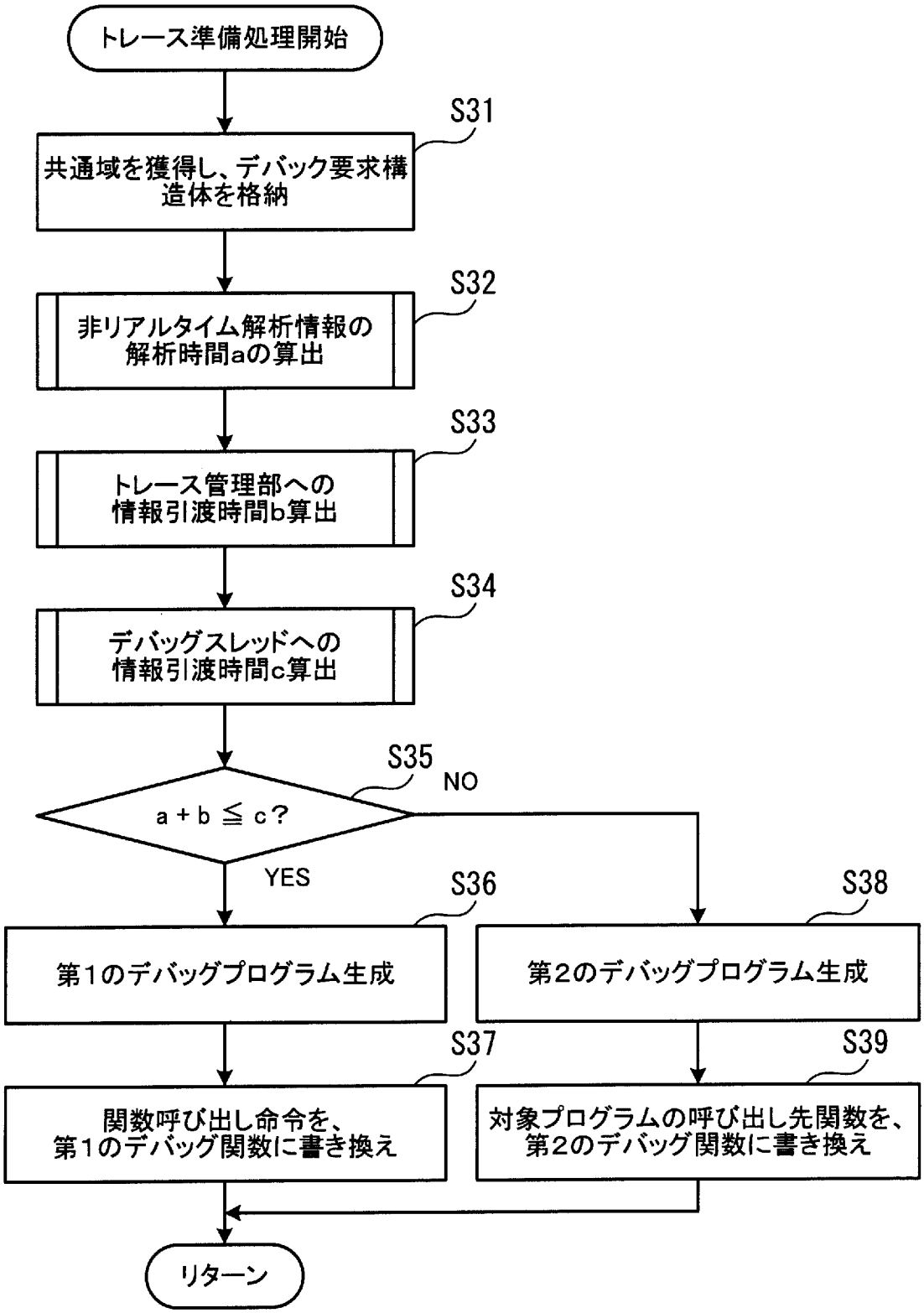

40 共通域
41
42
43
add#1
デバッグ結果構造体の大きさ
デバッグ要求構造体
81
add#1
add#2
デバッグ結果構造体のアドレス
非リアルタイム解析情報種別
非リアルタイム解析情報領域のアドレス
84
デバッグ結果構造体
非リアルタイム解析情報領域
87
82
add#2
add#3
デバッグ結果構造体のアドレス
非リアルタイム解析情報種別
非リアルタイム解析情報領域のアドレス
85
デバッグ結果構造体
非リアルタイム解析情報領域
88
83
add#3
0
デバッグ結果構造体のアドレス
非リアルタイム解析情報種別
非リアルタイム解析情報領域のアドレス
86
デバッグ結果構造体
非リアルタイム解析情報領域
89

[図22]

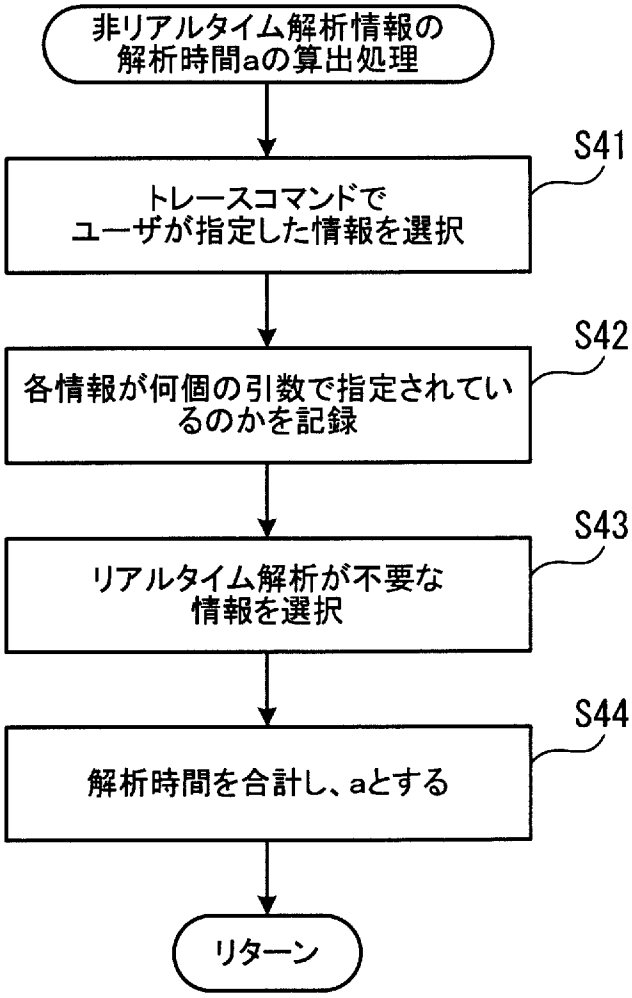


解析情報	リアルタイム解析の要否	解析時間（見積もり値）	情報引渡時間（見積もり値）	依頼準備時間（見積もり値）
情報1	r1	ka1	kb1	kc1
情報2	r2	ka2	kb2	kc2
情報3	r3	ka3	kb3	kc3
⋮	⋮	⋮	⋮	⋮

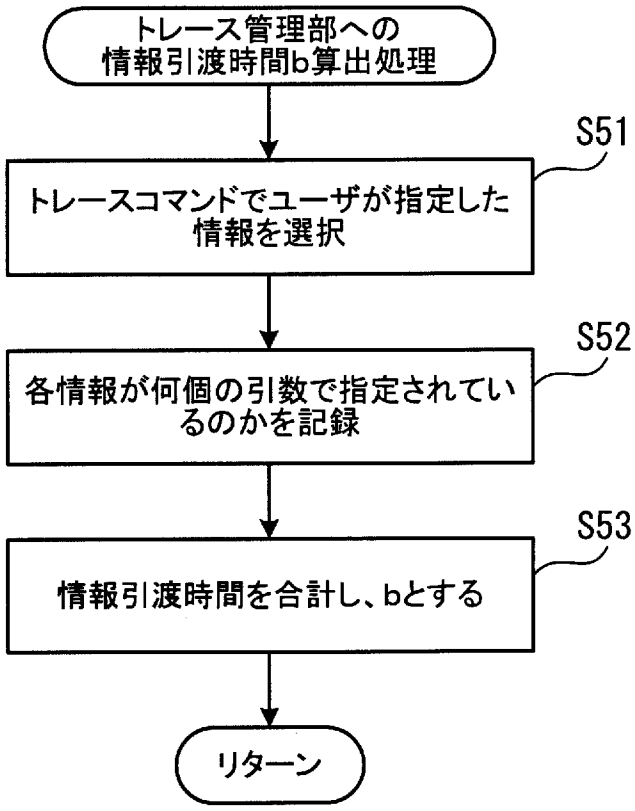
[図23]


トレース準備処理開始
S31
共通域を獲得し、デバック要求構造体を格納
S32
非リアルタイム解析情報の解析時間aの算出
S33
トレース管理部への情報引渡時間b算出
S34
デバッグスレッドへの情報引渡時間c算出
S35
a + b ≦ c?
NO
YES
S36
第1のデバッグプログラム生成
S38
第2のデバッグプログラム生成
S37
関数呼び出し命令を、第1のデバッグ関数に書き換え
S39
対象プログラムの呼び出し先関数を、第2のデバッグ関数に書き換え
リターン

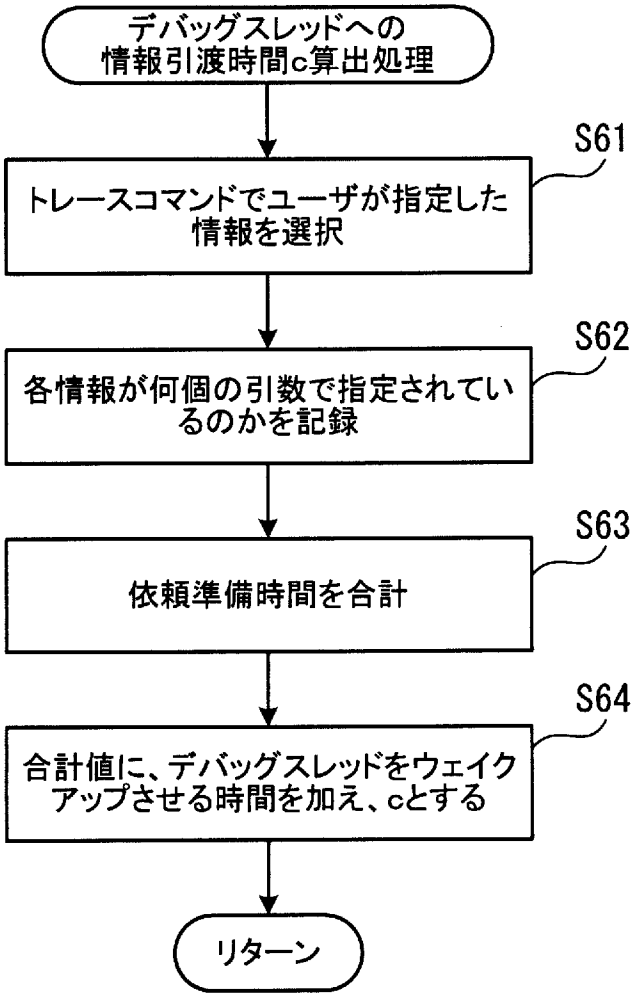
[図24]


非リアルタイム解析情報の
解析時間aの算出処理
S41
トレースコマンドで
ユーザが指定した情報を選択
S42
各情報が何個の引数で指定されている
のかを記録
S43
リアルタイム解析が不要な
情報を選択
S44
解析時間を合計し、aとする
リターン

[図25]


トレース管理部への
情報引渡時間b算出処理
S51
トレースコマンドでユーザが指定した
情報を選択
S52
各情報が何個の引数で指定されている
のかを記録
S53
情報引渡時間を合計し、bとする
リターン

[図26]

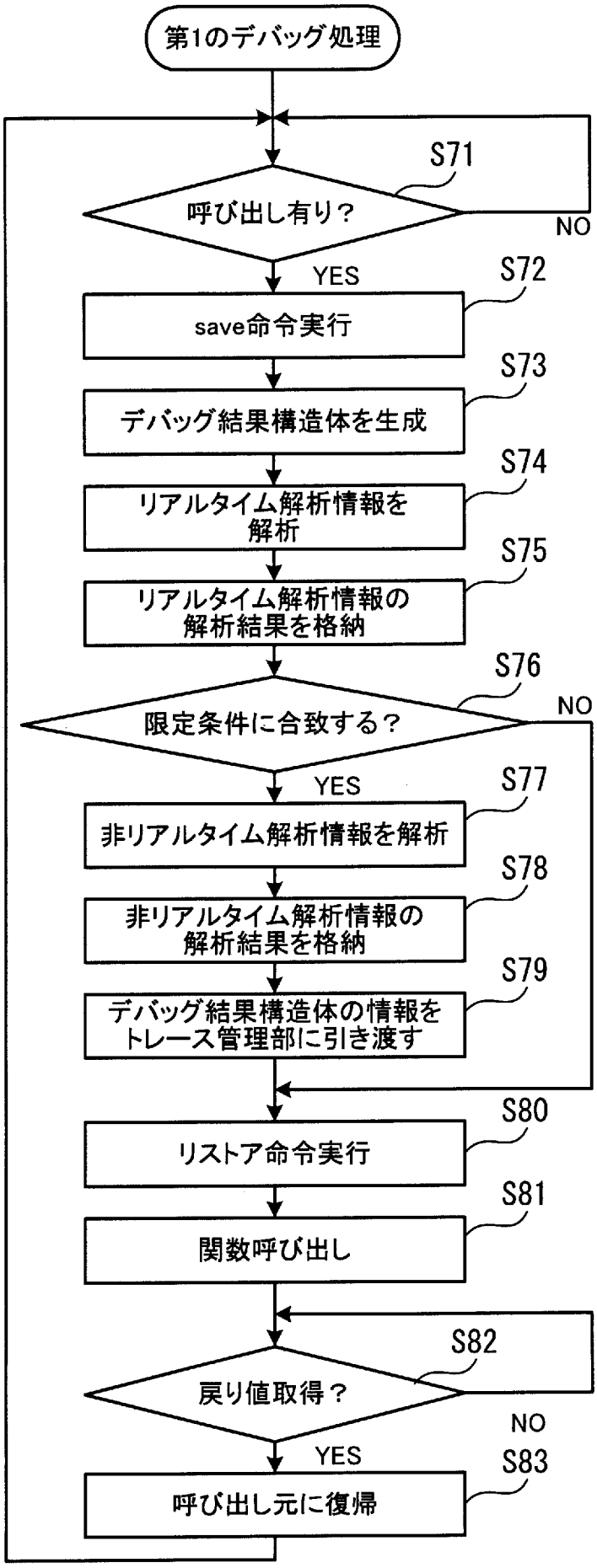

デバッグスレッドへの
情報引渡時間c算出処理
S61
トレースコマンドでユーザが指定した
情報を選択
S62
各情報が何個の引数で指定されてい
るのかを記録
S63
依頼準備時間を合計
S64
合計値に、デバッグスレッドをウェイク
アップさせる時間を加え、cとする
リターン

[図27]

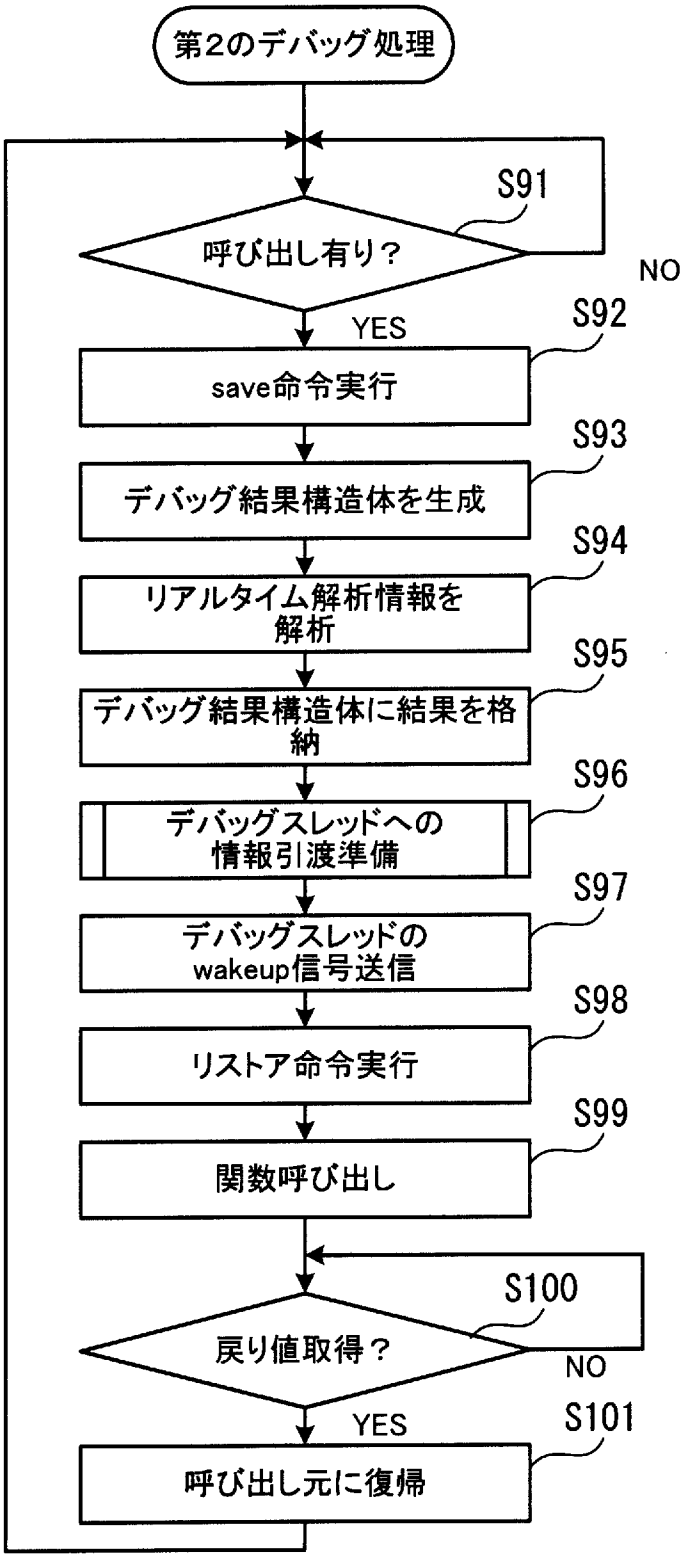
132a 時間見積もりテーブル

解析情報	リアルタイム解析の要否	解析時間（見積もり値）	情報引渡時間（見積もり値）	依頼準備時間（見積もり値）
スタック	必要	20μs	10μs	10μs
タイムスタンプ	必要	1μs	1μs	1μs
引数のアドレス	必要	1μs	1μs	1μs
限定条件(Ethertypeやftpなどのパケット種別)	不要	2μs	0μs	0μs
限定条件(パケット長)	不要	2μs	0μs	0μs
限定条件(パケットの内容のパターンマッチング)	不要	20μs	0μs	0μs
詳細な出力情報：パケット全体	不要	0μs	100μs	2μs
詳細な出力情報：64バイト以下	不要	0μs	10μs	2μs
詳細な出力情報：65-128バイト	不要	0μs	20μs	2μs
詳細な出力情報：129-512バイト	不要	0μs	40μs	2μs
詳細な出力情報：512バイト以上	不要	0μs	100μs	2μs

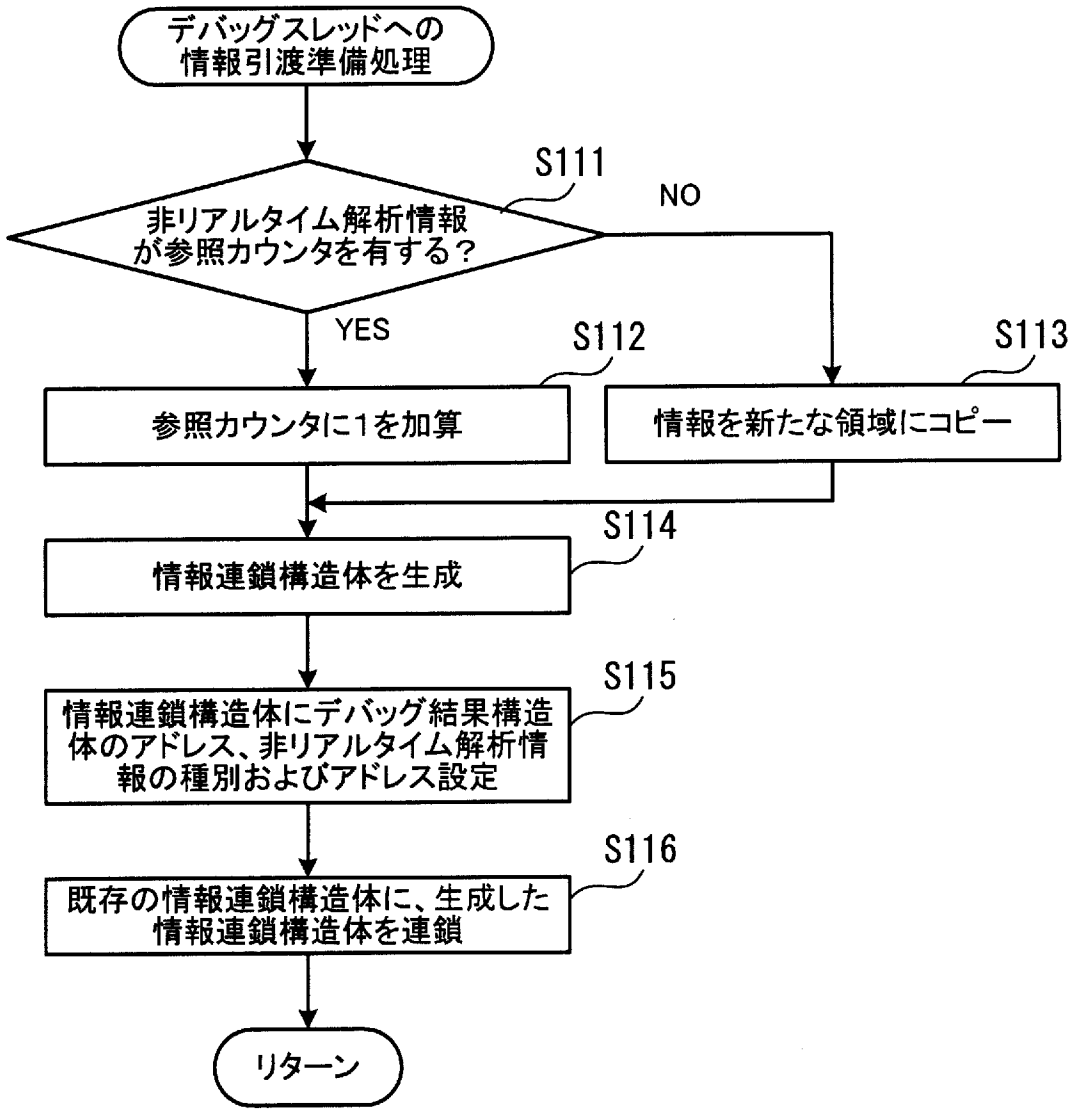
[図28]


第1のデバッグ処理
S71
呼び出し有り？
NO
YES
S72
save命令実行
S73
デバッグ結果構造体を生成
S74
リアルタイム解析情報を解析
S75
リアルタイム解析情報の解析結果を格納
S76
限定条件に合致する？
NO
YES
S77
非リアルタイム解析情報を解析
S78
非リアルタイム解析情報の解析結果を格納
S79
デバッグ結果構造体の情報をトレース管理部に引き渡す
S80
リストア命令実行
S81
関数呼び出し
S82
戻り値取得？
NO
YES
S83
呼び出し元に復帰

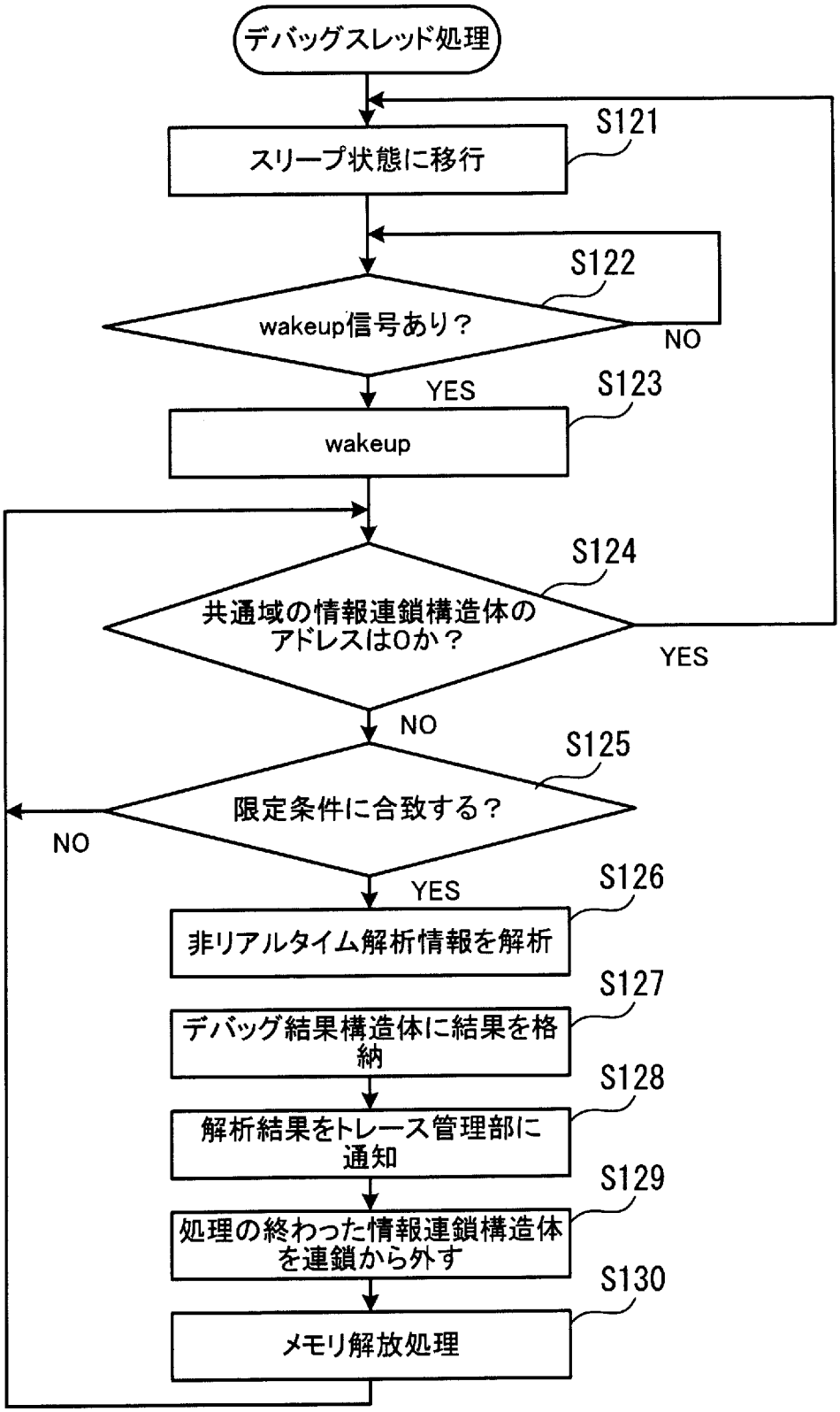
[図29]


第2のデバッグ処理
S91
呼び出し有り？
NO
YES
S92
save命令実行
S93
デバッグ結果構造体を生成
S94
リアルタイム解析情報を解析
S95
デバッグ結果構造体に結果を格納
S96
デバッグスレッドへの情報引渡準備
S97
デバッグスレッドのwakeup信号送信
S98
リストア命令実行
S99
関数呼び出し
S100
戻り値取得？
NO
YES
S101
呼び出し元に復帰

[図30]


デバッグスレッドへの
情報引渡準備処理
S111
非リアルタイム解析情報
が参照カウンタを有する？
NO
YES
S112
参照カウンタに1を加算
S113
情報を新たな領域にコピー
S114
情報連鎖構造体を生成
S115
情報連鎖構造体にデバッグ結果構造
体のアドレス、非リアルタイム解析情
報の種別およびアドレス設定
S116
既存の情報連鎖構造体に、生成した
情報連鎖構造体を連鎖
リターン

[図31]


デバッグスレッド処理
S121
スリープ状態に移行
S122
wakeup信号あり？
NO
YES
S123
wakeup
S124
共通域の情報連鎖構造体の
アドレスは0か？
YES
NO
S125
限定条件に合致する？
NO
YES
S126
非リアルタイム解析情報を解析
S127
デバッグ結果構造体に結果を格納
S128
解析結果をトレース管理部に通知
S129
処理の終わった情報連鎖構造体を連鎖から外す
S130
メモリ解放処理

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2011/057989

A. CLASSIFICATION OF SUBJECT MATTER
G06F11/28(2006.01)i, *G06F11/34*(2006.01)i

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)
G06F11/28, G06F11/34

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	1996-2011
Kokai Jitsuyo Shinan Koho	1971-2011	Toroku Jitsuyo Shinan Koho	1994-2011

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2009-237610 A (NTT Docomo Inc.), 15 October 2009 (15.10.2009), entire text; all drawings (Family: none)	1-11
A	JP 2005-115969 A (Toshiba Corp.), 28 April 2005 (28.04.2005), paragraphs [0058] to [0064] (Family: none)	1-11

☐ 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 application or patent 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
14 July, 2011 (14.07.11)	26 July, 2011 (26.07.11)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１１／０５７９８９

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G06F11/28(2006.01)i, G06F11/34(2006.01)i

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.Cl.　G06F11/28, G06F11/34

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	１９７１－２０１１年
日本国実用新案登録公報	１９９６－２０１１年
日本国登録実用新案公報	１９９４－２０１１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）

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	関連する請求項の番号
A	JP 2009-237610 A (株式会社エヌ・ティ・ティ・ドコモ) 2009.10.15, 全文, 全図 (ファミリーなし)	１－１１
A	JP 2005-115969 A (株式会社東芝) 2005.04.28, 段落[0058]-[0064] (ファミリーなし)	１－１１

☐　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 １４．０７．２０１１	国際調査報告の発送日 ２６．０７．２０１１
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）　５Ｂ　３５６２ 多胡　滋 電話番号　０３－３５８１－１１０１　内線　３５４５