(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
2024年4月4日(04.04.2024)

WIPO | PCT

(10) 国際公開番号
WO 2024/069772 A1

(51) 国際特許分類:
G06F 9/455 (2006.01)

(21) 国際出願番号： PCT/JP2022/036020

(22) 国際出願日： 2022年9月27日(27.09.2022)

(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: 日本電信電話株式会社 (**NIPPON TELEGRAPH AND TELEPHONE CORPORATION**) [JP/JP]; 〒1008116 東京都千代田区大手町一丁目５番１号 Tokyo (JP).

(72) 発明者: 川古谷 裕平 (**KAWAKOYA, Yuhei**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP).

岩村 誠(**IWAMURA, Makoto**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP).

(74) 代理人: 弁理士法人酒井国際特許事務所 (**SAKAI INTERNATIONAL PATENT OFFICE**); 〒1000013 東京都千代田区霞が関３丁目８番１号 虎の門三井ビルディング Tokyo (JP).

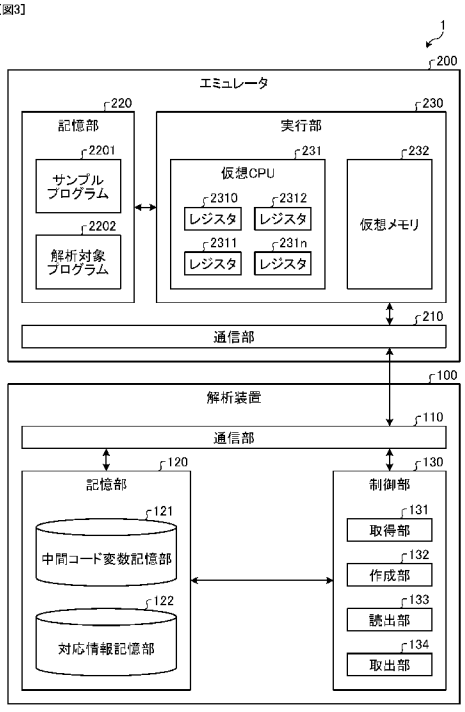
(81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU,

WO 2024/069772 A1

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

(54) 発明の名称： 解析装置、解析方法および解析プログラム

[図3]



100... ANALYSIS DEVICE
110, 210... COMMUNICATION UNIT
120, 220... STORAGE UNIT
121... INTERMEDIATE CODE VARIABLE STORAGE UNIT
122... CORRESPONDENCE INFORMATION STORAGE UNIT
130... CONTROL UNIT
131... ACQUISITION UNIT
132... CREATION UNIT
133... READ UNIT
134... EXTRACTION UNIT
200... EMULATOR
230... EXECUTION UNIT
231... VIRTUAL CPU
232... VIRTUAL MEMORY
2201... SAMPLE PROGRAM
2202... ANALYSIS TARGET PROGRAM
2310, 2311, 2312, 231n... REGISTER

(57) **Abstract:** An analysis device (10): creates a correspondence table on the basis of an argument, which is a value output by executing a known operation included in the same program as or a similar program to an analysis target program, an identifier of the known operation, and an intermediate code variable ID for identifying an intermediate code variable that holds the same value as the argument; acquires the intermediate code variable ID corresponding to a predetermined operation executed by the analysis target program; reads the argument identified by the intermediate code variable ID on

［続葉有］

LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, 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, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類：

一　国際調査報告（条約第21条(3)）

the basis of the correspondence table; and uses the argument read by a read unit (133) to extract information in any one of or both of a register of a virtual CPU on which the analysis target program operates and a memory.

(57) 要約：解析装置（１００）は、解析対象プログラムと同一または類似のプログラムに含まれる既知の動作を実行して出力される値である引数と、該既知の動作の識別子と、該引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成し、解析対象プログラムが実行する所定の動作に対応する中間コード変数ＩＤを取得し、対応表に基づき該中間コード変数ＩＤが識別する引数を読み出し、読出部（１３３）が読み出した引数を用いて、解析対象プログラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報のいずれか１つまたは両方を取り出す。

明　細　書

発明の名称 ： 解析装置、解析方法および解析プログラム

技術分野

[0001] 本発明は、解析装置、解析方法および解析プログラムに関する。

背景技術

[0002] プログラムの解析を行う際、例えば、エミュレータや仮想マシン等を用いて解析対象プログラムを動作させ、解析対象プログラムの命令を逐次実行したり、プログラムの特定箇所の実行をフックしたりし、予め定める条件の成立時にレジスタやメモリ等の状態を出力する挙動監視や挙動解析等を行う方法が知られている。

[0003] このような挙動監視方法に用いられるエミュレータの一例として、ＱＥＭＵが知られている（例えば、非特許文献１）。ＱＥＭＵは、様々なアーキテクチャのＣＰＵやデバイスをエミュレートするエミュレータであり、様々なアーキテクチャ用にコンパイルされたプログラムを、ＱＥＭＵを動作させる実行環境であるホストのアーキテクチャに関係なくエミュレートすることができる。

[0004] 具体的には、ＱＥＭＵは、解析対象プログラムの命令（ゲストコード）を読み取り、中間コード（例えば、「Tiny Code Generator Intermediate Representation, i.e. TCG IR」等)と呼称する、アーキテクチャに依存しないコードに変換する。そして、ＱＥＭＵは、変換された中間コードをホストで実行できる命令列（ホストコード）に変換し、ホスト上で実行する。なお、このホストコードは物理ＣＰＵ上で実行される。

[0005] ＱＥＭＵの場合、解析対象プログラムの挙動を解析するために、ＴＣＧ Ｉ／Ｆ (Tiny Code Generator Interface)といったインタフェースを用いてプラグイン（解析モジュール）を作成する方法が知られている（例えば、非特許文献２）。ＴＣＧ Ｉ／Ｆ は中間コードレイヤで動作し、ゲストコードのエミュレーション処理や、ホストコードを実行する処理に介入する

ことができる。

[0006] ＴＣＧ　Ｉ／Ｆ　を使った解析モジュールのように中間コードで解析を行う利点は、解析対象プラグラムのアーキテクチャに依存することなく挙動監視できる点がある。ＱＥＭＵでは、約３０のアーキテクチャをエミュレートすることができる。そして、ＴＣＧ　Ｉ／Ｆ　を利用して、１つの中間コードレイヤで動作する解析モジュールを作成することで、前述の約３０のアーキテクチャ毎に解析モジュールを作成する必要はなく、様々なアーキテクチャの解析対象プログラムの挙動を解析することが可能となり、効率的に多数のアーキテクチャを解析するモジュールを作成することができる。

先行技術文献

非特許文献

[0007] 非特許文献1：Fabrice Bellard, QEMU, a Fast and Portable Dynamic Translator, 2005 USENIX Annual Technical Conference (USENIX ATC 05), 2005.

非特許文献2：Emilio G. Cota and Luca P. Carloni, Cross-ISA Machine Instrumentation using Fast and Scalable Dynamic Binary Translation, Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2019.

発明の概要

発明が解決しようとする課題

[0008] しかしながら、従来技術では、解析対象プログラムのアーキテクチャにおける仮想ＣＰＵのレジスタに対応する中間コード変数を特定できず、レジスタやメモリの情報等を取り出しが容易ではない、という問題があった。

[0009] 具体的には、中間コードレイヤにおける解析モジュールは、アーキテクチャに依存せずに解析対象プログラムの実行に介入できるが、レジスタやメモリ等の状態のダンプ（値の取得）を行う際、各アーキテクチャ固有のレジスタやポインタを意識する必要が出てくる場合がある。言い換えると、中間コ

ードレイヤにおける解析モジュールは、解析対象プログラムのアーキテクチャに依存することなく各命令の実行に介入することはできるが、システムコール引数のような情報（例えば、レジスタ値やレジスタからポイントされているメモリ値）を得るためには、解析対象プログラムのアーキテクチャを意識して、当該アーキテクチャの仮想ＣＰＵの情報が格納されている場所を特定して、レジスタやメモリの情報等を取り出す必要がある。

課題を解決するための手段

[0010] 上記の課題を解決し目的を達成するために、本発明の解析装置は、解析対象プログラムと同一または類似のプログラムに含まれる既知の動作を実行して出力される値である引数と、該既知の動作の識別子と、該引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成する作成部と、前記解析対象プログラムが実行する所定の動作に対応する前記中間コード変数ＩＤを取得し、前記対応表に基づき該中間コード変数ＩＤが識別する前記引数を読み出す読出部と、前記読出部が読み出した前記引数を用いて、前記解析対象プログラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報のいずれか１つまたは両方を取り出す取出部と、有することを特徴とする。

発明の効果

[0011] 本発明は、解析対象プログラムのアーキテクチャにおける仮想ＣＰＵのレジスタに対応する中間コード変数の特定を可能とし、レジスタやメモリの情報等の取り出しを容易とする、という効果を奏する。

図面の簡単な説明

[0012] [図1]図１は、実施形態１に係るサンプルプログラムの一例を示す図である。

[図2]図２は、実施形態１に係る中間コード変数記憶部が記憶する情報の一例を示すテーブル図である。

[図3]図３は、実施形態１に係る解析システムの装置構成の一例を示す図である。

[図4]図４は、実施形態１に係る解析対象プログラムの一例を示す図である。

[図5]図５は、実施形態１に係る解析方法のフローチャートの一例を示す図である。

[図6]図６は、実施形態２に係る解析方法のフローチャートの一例を示す図である。

[図7]図７は、実施形態３に係る解析方法のフローチャートの一例を示す図である。

[図8]図８は、実施形態に係る解析装置およびエミュレータが実現されるコンピュータの一例を示す図である。

発明を実施するための形態

[0013] 以下、図面を参照しながら、本発明を実施するための形態（以降、「実施形態」）について説明する。なお、各実施形態は、以下に記載する内容に限定されない。

[0014] 〔１．解析方法の概要〕

本実施形態において、エミュレータ２００は、解析対象プログラムを実行する前に、予め用意する解析対象プログラムと同じアーキテクチャのサンプルプログラムに含まれる既知の動作（例えば、システムコールやライブラリコール等）を実行する。次に、解析装置１００は、前述のサンプルプログラムの既知の動作の実行により得られる既知の動作を識別する情報（例えば、システムコール番号やライブラリコールのアドレス等）と、サンプルプログラムの実行により得られる引数と、該引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤを関連付けて記憶し、対応表を作成する。

[0015] そして、解析装置１００は、エミュレータ２００による解析対象プログラムの所定の動作（例えば、システムコールやライブラリコール等）の発生を捕捉した際に、作成した対応表に基づき対応する中間コード変数から引数として渡されている各レジスタ値やメモリの値等を読み出すことで、解析対象プログラムの挙動をより詳細に監視する。

[0016] なお、以降の説明では、ＱＥＭＵやＢｏｃｈｓ等の一般的なエミュレータを利用することを前提に、各実施形態について説明する。ただし、本実施形

態は、ＱＥＭＵやＢｏｃｈｓ等の実装に限定されるものではなく、中間コードを利用する仮想マシン一般に適用できる。また、エミュレーションの形態（システムエミュレーションやプロセスエミュレーション等）についても限定されず、例えば、Ｊａｖａ（登録商標）やＡｎｄｒｏｉｄ（登録商標）等のプロセスレベルの仮想マシンにも適用できる。

[0017] 〔２．実施形態１：システムコール引数に基づく対応表の作成と解析〕

ここから、実施形態１として解析システム１が実現する、システムコール引数に基づく対応表の作成および解析対象プログラムの解析について説明を行う。なお、実施形態１における対応表は、「ＶＡ－Ｍａｐ（Variable Argument Mapping）」と呼称する。また以下の説明では、ＶＡ－Ｍａｐを作成後に解析対象プログラムの解析を実施する、という順番で説明を行う。ただし、解析システム１が行う手順は、前述の順番に限定されず、その他の順番や手順としてよい。例えば、解析装置１００は、ＶＡ－Ｍａｐを作成後に工程を一旦終了し、異なる時系列にて解析対象プログラムの解析を行う等の手順を実施してよい。なお、同じアーキテクチャのプログラムを解析する場合は、過去に作成した対応表を利用することもできる。

[0018] まず、エミュレータ２００は、システムコールを識別するシステムコール番号が既知であるシステムコール（既知の動作）を含むサンプルコードについてコンパイル（例えば、クロスコンパイル等）を実施し、解析対象プログラムと同じアーキテクチャのサンプルプログラム（例えば、バイナリプログラム等）を生成する。

[0019] 例えば、エミュレータ２００は、図１に示すようなサンプルコード２２０１ａに基づいて、サンプルプログラム２２０１を生成する。図１のサンプルコード２２０１ａは、システムコール１（ｓｙｓ＿ｃａｌｌ１）とシステムコール２（ｓｙｓ＿ｃａｌｌ２）を含む。システムコール１（ｓｙｓ＿ｃａｌｌ１）は、３つの引数（「０ｘ１２３４５６７８」、「０ｘａｂｃｄｅｆｇ」、「０ｘｄｅａｄｂｅｅｆ」）を渡して、システムコール番号１のシステムコールを呼び出す。他方、システムコール２（ｓｙｓ＿ｃａｌｌ２）は

、２つの引数（「０ｘ８ｂａｄｆ００ｄ」、「０ｘ８７６５４３２１」）を渡して、システムコール番号２のシステムコールを呼び出す。なお、本実施形態においては、前述したような特定のシステムコールを呼び出すサンプルプログラムを複数（１つ以上）用意する。

[0020] 続けて、エミュレータ２００は、前述のサンプルプログラム２２０１を実行する。具体的には、エミュレータ２００は、サンプルコードに含まれる既知のシステムコールを実行し、識別可能な値を既知のシステムコール引数として出力する。

[0021] 次に、解析装置１００は、エミュレータ２００のシステムコールの実行を捕捉する。なお、本実施形態におけるシステムコールの実行の補足方法は、特に限定されない。例えば、解析装置１００は、システムコールの実行を仮想アドレスに基づいて捉えてもよいし、任意の命令の実行や値の参照を契機として捉えてもよい。

[0022] そして、解析装置１００の取得部１３１は、中間コード変数の値を取得する。中間コード変数は、それぞれを一意に区別できる識別子として中間コード変数ＩＤを有する。例えば、図１のサンプルコード２２０１ａに含まれるシステムコール１（ｓｙｓ＿ｃａｌｌ１）を実行した場合、解析装置１００の中間コード変数記憶部１２１は、図２に示すように「中間コード変数ＩＤ」と、「中間コード変数名」と、「値」と、を記憶する。

[0023] 次に、解析装置１００の作成部１３２は、サンプルプログラム２２０１に記述されたシステムコールのシステムコール番号とサンプルプログラム２２０１の実行により得られる引数を用いて、該引数の値と同じ値を保持する中間コード変数ＩＤを特定する。そして、解析装置１００の作成部１３２は、特定した中間コード変数ＩＤを用いて、システムコール番号と、それに紐づく引数ｎ個に対応する中間コード変数ＩＤとの対応を作成する。

[0024] また、作成部１３２は、特定した中間コード変数ＩＤを用いて、システムコール番号と、それに紐づく引数ｎ個に対応する制約を作成する。具体的には、作成部１３２は、制約を作成する際に同じ引数の値が複数の中間コード

変数に含まれる場合には、それぞれの制約を連結させる。例えば、作成部１３２は、図１に示すサンプルコード２２０１ａに含まれるシステムコール１（ｓｙｓ＿ｃａｌｌ１）を実行した際、「システムコール番号＝ＩＤ０　ａｎｄ　（ａｒｇ０＝ＩＤ１　ｏｒ　ａｒｇ０＝ＩＤ４）　ａｎｄ　ａｒｇ１＝ＩＤ２」という制約を作成する。そして、作成部１３２は、前述の手順を複数のシステムコールで実施し、制約の集合を作成する。

[0025] さらに、作成部１３２は、収集した制約の集合についてＳＭＴ（Satisfiability Modulo Theories）ソルバを用いて解を求めることで割り当てを行い、ＶＡ－Ｍａｐを作成する。例えば、解析装置１００の作成部１３２は、「システムコール番号＝ＩＤ０，ａｒｇ０＝ＩＤ１，ａｒｇ１＝ＩＤ２，・・・」といった形式の第３引数までのＶＡ－Ｍａｐを作成する。なお、ＳＭＴソルバを用いた割当方法は、後述の実施形態１と、実施形態２と、実施形態３とで同様であるため、以降の説明は省略する。

[0026] 次に、エミュレータ２００は、解析対象プログラムを実行する。そして、解析装置１００の読出部１３３は、エミュレータ２００が解析対象プログラムの発行するシステムコールの呼び出しを補足した際、ＶＡ－Ｍａｐから「”システムコール番号が保存されている中間コード変数ＩＤ”、”ａｒｇ０が保存されている中間コード変数ＩＤ”、…、　”ａｒｇＮが保存されている中間コード変数ＩＤ”」を取得し、当該中間コード変数ＩＤが識別する中間コード変数から、対象の引数を読み出す。

[0027] そして、解析装置１００の取出部１３４は、読出部１３３が読み出した引数を用いて、解析対象プログラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報等を取り出す。

[0028] 〔２－１．解析装置とエミュレータの構成〕

ここから、実施形態１における解析システム１の構成について、図３を用いて説明を行う。実施形態１における解析システム１は、解析装置１００とエミュレータ２００を含む装置構成である。以下の項目で、各装置の機能部について詳細な説明を行う。

[0029]（解析装置１００）

まず、解析装置１００の構成について、図３を用いて説明する。図３に示すように、解析装置１００は、通信部１１０と、記憶部１２０と、制御部１３０と、を有する。なお、図示していないが、解析装置１００は、各種操作を受け付ける入力部（例えば、キーボードやマウス等）や、各種情報を表示するための表示部（例えば、ディスプレイ等）を備えてもよい。また、解析装置１００は、デスクトップ型パーソナルコンピュータ（Personal Computer）や、ノート型ＰＣ、仮想ＰＣ、スマートフォンやタブレット、ＰＤＡ（Personal Digital Assistant）等であってよく、情報処理装置としての形態は限定されない。続いて、以下に各部の詳細な機能について記載する。

[0030]（通信部１１０）

通信部１１０は、エミュレータ２００との各種情報のやり取りを行う。なお、本実施形態では、解析装置１００は通信部１１０を介してエミュレータ２００と情報のやり取りを行う前提で説明するが、通信部１１０を介さずに情報のやり取りを行ってもよい。また、通信部１１０は、必要に応じてＬＡＮ（Local Area Network）やインターネット等の電気通信回線を介して通信を制御し、外部の情報処理装置等と双方向に情報の送受信を行ってよい。

[0031]（記憶部１２０）

記憶部１２０は、制御部１３０による各種処理に必要なデータおよびプログラムを格納する。そして、記憶部１２０は、中間コード変数記憶部１２１と、対応情報記憶部１２２と、を有する。なお、記憶部１２０は、ＲＡＭ（Random Access Memory）、フラッシュメモリ（Flash Memory）等の半導体メモリ素子、または、ハードディスク、光ディスク等の記憶装置等で実現する。

[0032]（中間コード変数記憶部１２１）

中間コード変数記憶部１２１は、中間コード変数に関する情報を記憶する。例えば、図２に示すように、「中間コード変数ＩＤ」と、「中間コード変数名」と、「値」といった情報を記憶する。なお、中間コード変数記憶部１

２１は、前述した情報に限定されず、その他の中間コード変数に関する情報を記憶してもよい。

[0033]（対応情報記憶部１２２）

ここから、図３に戻り説明を続ける。対応情報記憶部１２２は、解析装置１００の作成部１３２が作成する対応表を記憶する。具体的には、対応情報記憶部１２２は、ＶＡ－Ｍａｐ（実施形態１および２）とＶＲ－Ｍａｐ（実施形態３）を記憶する。なお、対応情報記憶部１２２は、前述した対応表であるＶＡ－ＭａｐとＶＲ－Ｍａｐに限定されず、その他の対応表を記憶してもよい。さらに、対応情報記憶部１２２は、対応表の形式を表形式に限定せずに記憶してよい。

[0034]（制御部１３０）

制御部１３０は、取得部１３１と、作成部１３２と、読出部１３３と、取出部１３４と、を有する。そして、制御部１３０は、各種の処理手順等を規定したプログラムや処理データを一時的に格納するための内部メモリを有し、ＣＰＵ（Central Processing Unit）やＭＰＵ（Micro Processing Unit）等の電子回路やＡＳＩＣ（Application Specific Integrated Circuit）やＦＰＧＡ（Field Programmable Gate Array）等の集積回路によって実現されてよい。なお、前述したＣＰＵおよび内部メモリは、物理ＣＰＵや物理メモリに限定されず、仮想ＣＰＵや仮想メモリであってよい。

[0035]（取得部１３１）

取得部１３１は、既知の動作としてシステムコールを実行し、出力される引数であるシステムコール引数と、該システムコールを識別する識別子としてシステムコール番号を取得する。具体的には、取得部１３１は、システムコール番号が既知のシステムコールを実行して、既知のシステムコール番号と、既知のシステムコール番号によって識別されるシステムコール引数と、を取得する。

[0036] さらに、解析装置１００の取得部１３１は、既存の動作（システムコールやライブラリコール等）を実施した際に、中間コード変数の値を取得する。

なお、取得部１３１が取得する情報は前述のシステムコール引数に限定されず、その他の引数に関する情報を取得してよい。

[0037]（作成部１３２）

作成部１３２は、解析対象プログラムと同一または類似のプログラムに含まれる既知の動作を実行して出力される引数の値と、該既知の動作の識別子と、該引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成する。具体的には、作成部１３２は、システムコール引数と、システムコール番号と、該システムコール引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表として、ＶＡ－Ｍａｐを作成する。なお、作成部１３２は、前述したＶＡ－Ｍａｐ以外の形式の対応表を作成してもよい。

[0038] そして、作成部１３２は、取得したシステムコール番号およびシステムコール引数に基づき、対象となる（すなわち、取得したシステムコール引数と同じ値を保持する）中間コード変数ＩＤを特定する。

[0039]（読出部１３３）

読出部１３３は、解析対象プログラムが実行する所定の動作（実施形態１では、システムコール）に対応する中間コード変数ＩＤを取得し、対応表（ＶＡ－Ｍａｐ）に基づき該中間コード変数ＩＤが識別する引数を読み出す。なお、読出部１３３は必要に応じてその他の情報を読み出してもよい。

[0040] なお、読出部１３３は、変数の型に応じて、取得する引数のサイズや何バイト先まで辿るかを決定してもよい。例えば、読出部１３３は、第１引数の型が文字列型（引数の型がｃｈａｒ＊）の場合は、第１引数に該当する中間コード変数の値をポインタとして解釈し、その値が指すメモリ領域からヌルバイトが現れるまでのバイト列をその引数の出力としてもよい。さらに、読出部１３３は、必要に応じて、上記で得られた情報をファイルや標準出力やエラー出力に出力してもよい。

[0041]（取出部１３４）

取出部１３４は、読出部１３３が読み出した引数を用いて、解析対象プロ

グラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報のいずれか１つまたは両方を取り出す。なお、取出部１３４は、前述のレジスタもしくはメモリの情報以外の情報を取り出してもよい。

[0042]（エミュレータ２００）

次に、エミュレータ２００について説明する。エミュレータ２００は、仮想ＣＰＵおよび仮想メモリ等によって構成され、本実施形態においては、サンプルプログラムや解析対象プログラム等を実行する機能を有する。なお、エミュレータ２００は、物理ＰＣによって実現され、サンプルプログラムや解析対象プログラム等がこのエミュレータの上で動作する。

[0043] ここから、エミュレータ２００の構成について、引き続き図３を用いて説明する。エミュレータ２００は、図３に示すように、通信部２１０と、記憶部２２０と、実行部２３０と、を有する。

[0044]（通信部２１０）

通信部２１０は、解析装置１００との各種情報のやり取りを行う。なお、本実施形態では、エミュレータ２００は通信部２１０を介して解析装置１００と情報のやり取りを行う前提で説明するが、通信部２１０を介さずに、情報のやり取りを行ってもよい。

[0045]（記憶部２２０）

記憶部２２０は、実行部２３０による各種処理に必要なデータおよびプログラムを格納する。具体的には、記憶部２２０は、サンプルプログラム２２０１と解析対象プログラム２２０２を記憶する。なお、記憶部２２０は、必要に応じて前述のプログラム以外のプログラムを記憶してもよい。

[0046]（サンプルプログラム２２０１）

記憶部２２０が記憶するサンプルプログラム２２０１は、解析対象プログラム２２０２と同じアーキテクチャのプログラムであり、図１に示すようなサンプルコード２２０１ａに基づいて生成される。

[0047]（解析対象プログラム２２０２）

記憶部２２０が記憶する解析対象プログラム２２０２は、解析システム１

が解析の対象とするプログラムである。例えば、解析対象プログラム２２０２は、図４に示すようなゲストコード２２０２ａで表記される。図４は、エミュレータ２００の実行部２３０が、ゲストコード２２０２ａを、中間コードに変換する様子を示している。

[0048] 図４に示されるゲストコード２２０２ａに含まれる「”ｍｏｖ　ｅａｘ，０ｘ３ｆ”」は、「”ｍｏｖ＿ｉ６４　ｔｍｐ０，＄０ｘ３ｆ”，“ｅｘｔ３２ｕ＿ｉ６４　ｒａｘ，ｔｍｐ０”」という、２つの中間コードに変換される。図４に示されるゲストコード２２０２ａの「”ｅａｘ”」は、仮想ＣＰＵにおけるｅａｘレジスタを表す。他方、中間コードに現れる「”ｒａｘ”」や「”ｔｍｐ０”」は、中間コード変数の名称を表す。そして、ゲストコード２２０２ａでは、システムコールを実行する「”ｓｙｓｃａｌｌ１”」命令の実行前に、システムコール番号（ここでは０３ｘｆ）が、仮想ＣＰＵのｒａｘレジスタとは異なる「ｒａｘ」という名称の中間コード変数に格納されていることを表している。

[0049]（実行部２３０）

ここから、図３に戻り説明を続ける。実行部２３０は、記憶部２２０が記憶するプログラムを実行する。実行部２３０は、仮想ＣＰＵ２３１と、仮想メモリ２３２と、を有する。

[0050]（仮想ＣＰＵ２３１）

仮想ＣＰＵ２３１は、ＯＳ（Operating System）やアプリケーションソフト等を動作させる仮想化されたＰＣにおける中央処理装置（ＣＰＵ）の機能を有する。図３に示す通り、仮想ＣＰＵ２３１は、複数のレジスタを有しており、レジスタ２３１０と、レジスタ２３１１と、レジスタ２３１２と、レジスタ２３１ｎとを有している。なお、図３では４つのレジスタが表記されているが、仮想ＣＰＵ２３１が有するレジスタの数は４つに限定されず、それ以外の数のレジスタを有していてよい。

[0051]（仮想メモリ２３２）

仮想メモリ２３２は、ＯＳにより仮想的なアドレスを割り当てられたメモ

リ領域であり、動作するプログラムはＯＳを介して、仮想メモリにアクセスする。

[0052] 〔２－２．処理手順〕

次に、実施形態１に係る解析方法の手順について、図５を用いて説明する。まず、エミュレータ２００は、サンプルコードをクロスコンパイルし、解析対象プログラムと同一または類似のアーキテクチャのサンプルプログラムを生成する（工程Ｓ１００）。続けて、エミュレータ２００は、生成されたサンプルプログラムを実行する（工程Ｓ１０１）。そして、解析装置１００の取得部１３１は、既知のシステムコールの実行を検知する（工程Ｓ１０２）。続けて、解析装置１００の取得部１３１は、システムコール番号およびシステムコール引数を取得する（工程Ｓ１０３）。

[0053] 解析装置１００の作成部１３２は、システムコール番号およびシステムコール引数に基づき、対象となる（すなわち、取得したシステムコール引数と同じ値を保持する）中間コード変数ＩＤを特定する（工程Ｓ１０４）。続けて、解析装置１００の作成部１３２は、特定した中間コード変数ＩＤを用いて、制約を作成する（工程Ｓ１０５）。解析装置１００は、システムコールの実行が残されていると判定すると、工程を戻り処理を継続する（工程Ｓ１０６のＹｅｓ）。

[0054] 解析装置１００は、全てのシステムコールを実行すると、システムコールの実行が残っていないと判定する（工程Ｓ１０６のＮｏ）。その場合は、解析装置１００の作成部１３２は、作成した制約に基づき、ＳＭＴソルバを用いて割り当てを実施して、ＶＡ－Ｍａｐを作成する（工程Ｓ１０７）。

[0055] エミュレータ２００は、解析対象プログラムを実行する（工程Ｓ１０８）。そして、解析装置１００の読出部１３３は、解析対象プログラムによるシステムコールの実行を検知する（工程Ｓ１０９）。続けて、解析装置１００の読出部１３３は、実行されたシステムコールのシステムコール番号を保持する中間コード変数から引数を読み出す（工程Ｓ１１０）。解析装置１００の取出部１３４は、読み出した引数を用いて、レジスタもしくはメモリの情

報等を取り出し（工程Ｓ１１１）、工程が終了する。

[0056] 〔３．実施形態２：ライブラリコール引数に基づく対応表の作成〕

ここから、実施形態２として、解析システム１が実現するライブラリコール引数に基づく対応表の作成および解析対象プログラムの解析について説明を行う。なお、実施形態２における対応表に関しても、実施形態１と同様に「ＶＡ－Ｍａｐ（Variable Argument Mapping）」と呼称する。なお、ＶＡ－Ｍａｐの作成手順は、差異のある部分のみ記載する。

[0057] 実施形態２で用いるサンプルコード２２０１ａは、実施形態１のサンプルコード２２０１ａと類似であるが、システムコールではなくライブラリコールを実施するプログラムである。実施形態２においてエミュレータ２００は、前述のライブラリコールを行うサンプルプログラム２２０１を実行する。そして、解析装置１００の取得部１３１は、ライブラリコールを識別する識別子であるライブラリコールのアドレスと、ライブラリコール引数と、を取得する。

[0058] さらに、実施形態２では、解析装置１００の取得部１３１は、中間コード変数の保持する引数の値が参照可能なメモリ領域を指す場合、該当するメモリ領域のオフセット±ｍバイトのメモリ上の値もオフセットの値と共に取得する。そして、解析装置１００の作成部１３２は、ライブラリコールの各引数の値に基づき、中間コード変数ＩＤとオフセットを含む制約を作成する。

[0059] 次に、解析装置１００の読出部１３３は、エミュレータ２００による解析対象プログラムのライブラリコールの実行を捕捉した場合、ＶＡ－Ｍａｐを読み出す。そして、解析装置１００の読出部１３３は、ＶＭ－ＭＡＰの値にオフセットが含まれる場合は、中間コード変数ＩＤにより識別される中間コード変数が保持する引数の値が指すメモリ領域にオフセットを加えてアドレスを計算し、そのアドレスにある値をその引数の値として出力する。

[0060] 〔３－１．解析装置とエミュレータの構成〕

ここから、実施形態２における解析システム１の構成について、図３に戻り説明を行う。実施形態２における解析システム１は、解析装置１００とエ

ミュレータ２００を含む装置構成である。なお、実施形態２における装置構成は、実施形態１と同様であり、本項目では差異として、取得部１３１と作成部１３２の付加的機能のみ説明し、それ以外の詳細な説明は省略する。

[0061]（取得部１３１）

実施形態２の取得部１３１は、既知の動作としてライブラリコールを実行し、出力される引数であるライブラリコール引数と、該ライブラリコールを識別する識別子であるライブラリコールのアドレスと、所定の条件を満たす場合にはメモリ上の値とそのオフセットと、を取得する。なお、実施形態２の取得部１３１が取得する情報は前述のライブラリコール引数に限定されず、その他の引数に関する情報を取得してよい。

[0062] 加えて、実施形態２の取得部１３１は、中間コード変数の保持する引数の値が参照可能なメモリ領域を指す場合、該当するメモリ領域のオフセット±ｍバイトのメモリ上の値も取得する。

[0063]（作成部１３２）

実施形態２の作成部１３２は、ライブラリコール引数と、ライブラリコールのアドレスと、該ライブラリコール引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、オフセットと、に基づき対応表（実施形態２では、ＶＡ－Ｍａｐ）を作成する。なお、実施形態２の作成部１３２は、前述したＶＡ－Ｍａｐ以外の形式の対応表を作成してもよい。

[0064] さらに、作成部１３２は、取得したライブラリコールのアドレスおよびライブラリコール引数に基づき、対象となる（すなわち、取得したライブラリコール引数と同じ値を保持する）中間コード変数ＩＤと中間コード変数があるメモリ領域をさす場合は、そのオフセットも、特定する。例えば、実施形態２の作成部１３２は、ライブラリコールのある引数に関して、中間コード変数ＩＤ＝Ｘ，　メモリアドレスがＸ＋Ｙ（オフセットが＋Ｙ）の位置に当該引数と同じ値が見つかった場合、「ａｒｇ１＝（Ｘ，＋Ｙ）」といった制約を作成する。

[0065]（読出部１３３）

実施形態２の解析装置１００の読出部１３３は、エミュレータ２００による解析対象プログラムのライブラリコールの実行を捕捉した場合、ＶＡ－Ｍａｐを用いて、対象の第１引数を読み出す。例えば、解析装置１００の読出部１３３は、引数が「ａｒｇ１＝（Ｘ，＋Ｙ）」の場合には、中間コード変数ＩＤ＝Ｘにより識別される中間コード変数が保持する引数の値が指すメモリ領域からオフセット＋Ｙを計算し、そのアドレスにある値を第１引数の値として出力する。

[0066] 〔３－２．処理手順〕

次に、実施形態２に係る解析方法の手順について、図６を用いて説明する。まず、エミュレータ２００は、サンプルコードをクロスコンパイルし、解析対象プログラムと同一または類似のアーキテクチャのサンプルプログラムを生成する（工程Ｓ２００）。続けて、エミュレータ２００は、生成されたサンプルプログラムを実行する（工程Ｓ２０１）。そして、解析装置１００の取得部１３１は、既知のライブラリコールコールの実行を検知する（工程Ｓ２０２）。続けて、解析装置１００の取得部１３１は、ライブラリコール引数と、ライブラリコールのアドレスを取得する（工程Ｓ２０３）。

[0067] 解析装置１００の作成部１３２は、ライブラリコールのアドレスおよびライブラリコール引数に基づき、対象となる（すなわち、取得したライブラリコール引数と同じ値を保持する）中間コード変数ＩＤを特定する（工程Ｓ２０４）。なお、解析装置１００は、中間コード変数に引数の値と合致するものがなかった場合は、中間コード変数の値をポインタとして扱い、メモリ上の±ｍバイトの位置に当該値がないかを探す。続けて、解析装置１００の作成部１３２は、特定した中間コード変数ＩＤと、取得したオフセットを用いて、制約を作成する（工程Ｓ２０５）。そして、解析装置１００は、ライブラリコールの実行が残されていると判定すると、工程を戻り処理を継続する（工程Ｓ２０６のＹｅｓ）。

[0068] 解析装置１００は、全てのライブラリコールを実行すると、ライブラリコールの実行が残っていないと判定する（工程Ｓ２０６のＮｏ）。その場合は

、解析装置１００の作成部１３２は、作成した制約に基づき、ＳＭＴソルバを用いて割り当てを実施して、ＶＡ－Ｍａｐを作成する（工程Ｓ２０７）。

[0069] エミュレータ２００は、解析対象プログラムを実行する（工程Ｓ２０８）。そして、解析装置１００の読出部１３３は、解析対象プログラムによるライブラリコールの実行を検知する（工程Ｓ２０９）。続けて、解析装置１００の読出部１３３は、中間コード変数の値が示すメモリ領域とオフセットからメモリアドレスを計算して、該当アドレス先の引数を読み出す（工程Ｓ２１０）。解析装置１００の取出部１３４は、読み出した引数を用いて、レジスタもしくはメモリの情報等を取り出し（工程Ｓ２１１）、工程が終了する。

[0070] 〔４．実施形態３：仮想ＣＰＵのレジスタ値に基づく対応表の作成〕

ここから、実施形態３として、解析システム１が実現する、仮想ＣＰＵのレジスタ値に基づく対応表の作成および解析対象プログラムの解析について説明を行う。なお、実施形態３における対応表は、「ＶＲ－Ｍａｐ（Variable Register Mapping）」と呼称する。なお、ＶＲ－Ｍａｐの作成手順は、差異のある部分のみ記載する。

[0071] 実施形態３で用いるサンプルコード２２０１ａは実施形態１のサンプルコード２２０１ａと類似であるが、システムコールではなく、解析対象プログラムが稼働するエミュレータのレジスタが保持する情報（以降、「レジスタ値」と表記）を出力する関数を含むプログラムである。そして、解析装置１００の取得部１３１は、エミュレータ２００によって実施形態３で用いるサンプルコード２２０１ａをコンパイルしたサンプルプログラム２２０１を実行した際に出力されるレジスタの値を取得する。続けて、解析装置１００の作成部１３２は、中間コード変数が保持する値から該レジスタ値を持つもの抽出し、その中間コード変数のＩＤを使って制約を作成する。

[0072] そして、解析装置１００の作成部１３２は、サンプルプログラム２２０１から出力されたレジスタ値と同じ値を持つ中間コード変数ＩＤを特定し、ＶＲ－Ｍａｐを作成する。その後、解析装置１００の読出部１３３は、ＶＲ－

Ｍａｐを用いて特定のレジスタの値を対応する中間コード変数から取得する。

[0073] 〔４－１．解析装置とエミュレータの構成〕

ここから、実施形態３における解析システム１の構成について、図３に戻り説明を行う。実施形態３における解析システム１は、解析装置１００とエミュレータ２００を含む装置構成である。なお、実施形態３における装置構成は、基本的には実施形態１と同様であり、本項目では差異として、取得部１３１と作成部１３２の付加的機能のみ説明し、それ以外の詳細な説明は省略する。

[0074] （取得部１３１）

実施形態３の取得部１３１は、既知の動作として、仮想ＣＰＵのレジスタが保持するレジスタ値を出力する解析対象プログラムと同一または類似のプログラムを実行し、該レジスタ値を取得する。なお、実施形態３の取得部１３１が取得する情報は前述のレジスタ値に限定されず、その他の引数に関する情報を取得してよい。

[0075] （作成部１３２）

実施形態３の作成部１３２は、レジスタ値と、該レジスタ値と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表（実施形態３では、ＶＲ－Ｍａｐ）を作成する。なお、実施形態３の作成部１３２は、前述したＶＲ－Ｍａｐ以外の形式の対応表を作成してもよい。

[0076] さらに、実施形態３の作成部１３２は、取得したレジスタ値と同じ値を持つ中間コード変数ＩＤを特定する。また、実施形態３の解析装置１００の作成部１３２は、中間コード変数が保持する引数から該レジスタ値を抽出し、制約を作成する。

[0077] 〔４－２．処理手順〕

次に、実施形態３に係る解析方法の手順について、図７を用いて説明する。まず、エミュレータ２００は、サンプルコードをクロスコンパイルし、解析対象プログラムと同一または類似のアーキテクチャのサンプルプログラム

を生成する（工程Ｓ３００）。続けて、エミュレータ２００は、生成されたサンプルプログラムを実行する（工程Ｓ３０１）。そして、解析装置１００の取得部１３１は、サンプルプログラムの実行を検知する（工程Ｓ３０２）。続けて、解析装置１００の取得部１３１は、サンプルプログラムの実行により得られる仮想ＣＰＵの全てのレジスタ値を取得する（工程Ｓ３０３）。

[0078] 解析装置１００の作成部１３２は、取得したレジスタ値と同じ値を持つ中間コード変数ＩＤを特定する（工程Ｓ３０４）。続けて、解析装置１００の作成部１３２は、中間コード変数が保持する値から、レジスタ値を検索して、制約を作成する（工程Ｓ３０５）。解析装置１００の作成部１３２は、作成した制約に基づき、ＳＭＴソルバを用いて割り当てを実施し、ＶＲ－Ｍａｐを作成する（工程Ｓ３０６）。

[0079] エミュレータ２００は、解析対象プログラムを実行する（工程Ｓ３０７）。そして、解析装置１００の読出部１３３は、解析対象プログラムによるシステムコールもしくはライブラリコールの実行を検知する（工程Ｓ３０８）。解析装置１００の読出部１３３は、対応する中間コード変数から対象のレジスタ値を読み出しする（工程Ｓ３０９）。解析装置１００の取出部１３４は、レジスタもしくはメモリの情報等を取り出し（工程Ｓ３１０）、工程が終了する。

[0080] 〔５．効果〕

従来技術では、ＱＥＭＵ等のエミュレータを利用する場合、中間コードレイヤにおける解析モジュールは、アーキテクチャに依存せずに解析対象プログラムの実行に介入できるが、レジスタやメモリ等の状態のダンプを行う際、各アーキテクチャ固有のレジスタやポインタを意識する必要となる場合があった。具体例として、ｘ８６（Ｉｎｔｅｌ　８０８６およびその後方互換性を持つマイクロプロセッサの命令セットアーキテクチャの総称）の解析対象プログラムが、「ｓｙｓｅｎｔｅｒ」を利用してシステムコールを発行する挙動を解析する場合を基に説明する。

[0081] 例えば、解析対象プラグラムがｓｙｓｅｎｔｅｒを実行する際、システム

コールの種別を表すシステムコール番号が「ｒａｘ」レジスタに保存される。また、このシステムコールに渡される引数は、「ｒｄｉ」、「ｒｓｉ」、「ｒｄｘ」、「ｒｃｘ」、「ｒ８」、「ｒ９」にそれぞれ対応する順番で保存されている。そして、中間コードレイヤにおける解析モジュールがｓｙｓｅｎｔｅｒの実行を捉えた際、このシステムコールの種類を知るためには、ｒａｘレジスタの値を知る必要がある。また、このシステムコールの種類に応じて、渡される引数の数も決定し、その値は「ｒｄｉ」、「ｒｓｉ」、「ｒｄｘ」、「ｒｃｘ」、「ｒ８」、「ｒ９」にそれぞれ保存されている。そのため、より詳しいシステムコールの情報を得るためには、これらレジスタに保存された値や、その値が指す先のメモリの値も知る必要があった。

[0082] また、エミュレータが解析対象プログラムの命令を中間コードに変換する際に、各アーキテクチャへの依存をなくすため、各レジスタの値は中間コードの実行中に定義される変数（中間コード変数）に格納され、利用される。そのため、中間コードレイヤからは、仮想ＣＰＵのｒａｘレジスタやｒｄｉレジスタ等は直接アクセスすることができない。

[0083] つまり、中間コードレイヤにおける解析モジュールは解析対象プログラムのアーキテクチャに依存することなく、各命令の実行に介入することはできるが、システムコール引数のような情報（例えばレジスタの値やレジスタからポイントされているメモリの値）を得るためには、解析対象プログラムのアーキテクチャを意識して、当該アーキテクチャの仮想ＣＰＵの情報が格納されている場所を特定し、レジスタ値を取り出す必要がある。

[0084] 前述の問題をまとめると、中間コードレイヤにおける解析モジュールは中間コード変数にアクセスすることはできるが、各アーキテクチャの仮想ＣＰＵのレジスタと中間コード変数の対応がわからないため、どの中間コード変数に目的のレジスタ値が保持されているかはわからない。

[0085] そこで、本実施形態に係る解析装置１００およびエミュレータ２００は、解析対象プログラムと同一または類似のプログラムに含まれる既知の動作を実行して出力される値である引数と、該既知の動作の識別子と、該引数と同

じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成し、解析対象プログラムが実行する所定の動作に対応する中間コード変数ＩＤを取得し、対応表に基づき該中間コード変数ＩＤが識別する引数を読み出し、読出部１３３が読み出した引数を用いて、解析対象プログラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報のいずれか１つまたは両方を取り出す、ことを特徴とする。したがって、本実施形態によれば、以下のような効果を奏する。

[0086] 解析装置１００は、解析対象プログラムのアーキテクチャにおける仮想ＣＰＵのレジスタ値が保持される中間コード変数の特定を可能とし、レジスタやメモリの情報等の取り出しを容易とする、という効果を奏する。すなわち、解析装置１００は、解析対象プログラムのアーキテクチャに依存しない解析モジュールを作成することができ、より詳細な情報としてレジスタやメモリの情報等も取得することができる、という効果を提供する。

[0087] 〔６．ハードウェア構成〕

図示した各装置の各構成要素は機能概念的なものであり、必ずしも物理的に図示のように構成されていることを要しない。すなわち、各装置の分散・統合の具体的形態は図示のものに限られず、その全部または一部を、各種の負荷や使用状況等に応じて、任意の単位で機能的または物理的に分散・統合して構成することができる。さらに、各装置にて行われる各処理機能は、その全部または任意の一部が、ＣＰＵおよび当該ＣＰＵにて解析実行されるプログラムにて実現され、あるいは、ワイヤードロジックによるハードウェアとして実現され得る。

[0088] また、本実施形態において説明した各処理のうち、自動的に行われるものとして説明した処理の全部または一部を公知の方法で手動的に行うこともできる。この他、図面中で示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

[0089] ［プログラム］

一実施形態として、解析装置１００およびエミュレータ２００を構成する各種の装置は、パッケージソフトウェアやオンラインソフトウェアとして、前述した解析を実行する解析プログラムを、所望のコンピュータにインストールさせることによって実装できる。例えば、上記の解析プログラムを情報処理装置に実行させることにより、解析装置１００およびエミュレータ２００を構成する各種の装置として機能させることができる。ここで言う情報処理装置には、デスクトップ型またはノート型のパーソナルコンピュータが含まれる。また、その他にも、情報処理装置にはスマートフォン、携帯電話機やＰＨＳ（Personal Handyphone System）等の移動体通信端末、さらには、ＰＤＡ（Personal Digital Assistant）等のスレート端末等がその範疇に含まれる。

[0090] 図８は、解析装置１００およびエミュレータ２００を構成する各種の装置が実現されるコンピュータの一例を示す図である。コンピュータ１０００は、例えば、メモリ１０１０、ＣＰＵ１０２０を有する。また、コンピュータ１０００は、ハードディスクドライブインタフェース１０３０、ディスクドライブインタフェース１０４０、シリアルポートインタフェース１０５０、ビデオアダプタ１０６０、ネットワークインタフェース１０７０を有する。これらの各部は、バス１０８０によって接続される。

[0091] メモリ１０１０は、ＲＯＭ（Read Only Memory）１０１１およびＲＡＭ１０１２を含む。ＲＯＭ１０１１は、例えば、ＢＩＯＳ（Basic Input Output System）等のブートプログラムを記憶する。ハードディスクドライブインタフェース１０３０は、ハードディスクドライブ１０９０に接続される。ディスクドライブインタフェース１０４０は、ディスクドライブ１１００に接続される。例えば磁気ディスクや光ディスク等の着脱可能な記憶媒体が、ディスクドライブ１１００に挿入される。シリアルポートインタフェース１０５０は、例えばマウス１１１０、キーボード１１２０に接続される。ビデオアダプタ１０６０は、例えばディスプレイ１１３０に接続される。

[0092] ハードディスクドライブ１０９０は、例えば、ＯＳ１０９１、アプリケー

ションプログラム１０９２、プログラムモジュール１０９３、プログラムデータ１０９４を記憶する。すなわち、解析装置１００およびエミュレータ２００を構成する各種の装置の各処理を規定するプログラムは、コンピュータにより実行可能なコードが記述されたプログラムモジュール１０９３として実装される。プログラムモジュール１０９３は、例えばハードディスクドライブ１０９０に記憶される。例えば、解析装置１００およびエミュレータ２００を構成する各種の装置における機能構成と同様の処理を実行するためのプログラムモジュール１０９３が、ハードディスクドライブ１０９０に記憶される。なお、ハードディスクドライブ１０９０は、ＳＳＤ（Solid State Drive）により代替されてもよい。

[0093] また、前述した実施形態の処理で用いられる設定データは、プログラムデータ１０９４として、例えばメモリ１０１０やハードディスクドライブ１０９０に記憶される。そして、ＣＰＵ１０２０は、メモリ１０１０やハードディスクドライブ１０９０に記憶されたプログラムモジュール１０９３やプログラムデータ１０９４を必要に応じてＲＡＭ１０１２に読み出して、前述した実施形態の処理を実行する。

[0094] なお、プログラムモジュール１０９３やプログラムデータ１０９４は、ハードディスクドライブ１０９０に記憶される場合に限らず、例えば着脱可能な記憶媒体に記憶され、ディスクドライブ１１００等を介してＣＰＵ１０２０によって読み出されてもよい。あるいは、プログラムモジュール１０９３およびプログラムデータ１０９４は、ネットワーク（ＬＡＮ、ＷＡＮ（Wide Area Network）等）を介して接続された他のコンピュータに記憶されてもよい。そして、プログラムモジュール１０９３およびプログラムデータ１０９４は、他のコンピュータから、ネットワークインタフェース１０７０を介してＣＰＵ１０２０によって読み出されてもよい。

[0095] 〔７．その他〕

以上、本実施形態について説明したが、本実施形態は、開示の一部をなす記述および図面により限定されることはない。すなわち、本実施形態に基づ

いて当業者等によりなされる他の実施形態、実施例および運用技術等は全て本実施形態の範疇に含まれる。

符号の説明

[0096]

１	解析システム
１００	解析装置
１１０	通信部
１２０	記憶部
１２１	中間コード変数記憶部
１２２	対応情報記憶部
１３０	制御部
１３１	取得部
１３２	作成部
１３３	読出部
１３４	取出部
２００	エミュレータ
２１０	通信部
２２０	記憶部
２２０１	サンプルプログラム
２２０１ａ	サンプルコード
２２０２	解析対象プログラム
２２０２ａ	ゲストコード
２３０	実行部
２３１	仮想ＣＰＵ
２３１０	レジスタ
２３１１	レジスタ
２３１２	レジスタ
２３１ｎ	レジスタ
２３２	仮想メモリ

１０００　コンピュータ
１０１０　メモリ
１０１１　ＲＯＭ
１０１２　ＲＡＭ
１０２０　ＣＰＵ
１０３０　ハードディスクドライブインタフェース
１０４０　ディスクドライブインタフェース
１０５０　シリアルポートインタフェース
１０６０　ビデオアダプタ
１０７０　ネットワークインタフェース
１０８０　バス
１０９０　ハードディスクドライブ
１０９１　ＯＳ
１０９２　アプリケーションプログラム
１０９３　プログラムモジュール
１０９４　プログラムデータ
１１００　ディスクドライブ
１１１０　マウス
１１２０　キーボード

請求の範囲

[請求項1] 解析対象プログラムと同一または類似のプログラムに含まれる既知の動作を実行して出力される値である引数と、該既知の動作の識別子と、該引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成する作成部と、

前記解析対象プログラムが実行する所定の動作に対応する前記中間コード変数ＩＤを取得し、前記対応表に基づき該中間コード変数ＩＤが識別する前記引数を読み出す読出部と、

前記読出部が読み出した前記引数を用いて、前記解析対象プログラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報のいずれか１つまたは両方を取り出す取出部と、

を有することを特徴とする解析装置。

[請求項2] 前記既知の動作としてシステムコールを実行し、出力される前記引数であるシステムコール引数と、該システムコールを識別する識別子であるシステムコール番号を取得する取得部を更に有し、

前記作成部は、前記システムコール引数と、前記システムコール番号と、該システムコール引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成する、

ことを特徴とする請求項１に記載の解析装置。

[請求項3] 前記既知の動作としてライブラリコールを実行し、出力される前記引数であるライブラリコール引数と、該ライブラリコールを識別する識別子であるライブラリコールのアドレスと、所定の条件を満たす場合にはメモリアドレスと、を取得する取得部を更に有し、

前記作成部は、前記ライブラリコール引数と、前記ライブラリコールのアドレスと、該ライブラリコール引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、前記メモリアドレスと、に基づき対応表を作成する、

ことを特徴とする請求項１に記載の解析装置。

[請求項4] 前記既知の動作として、仮想ＣＰＵのレジスタが保持するレジスタ値を出力する前記解析対象プログラムと同一または類似のプログラムを実行し、該レジスタ値を取得する取得部を更に有し、

前記作成部は、前記レジスタ値と、該レジスタ値と同じ値を保持する前記中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成する、

ことを特徴とする請求項１に記載の解析装置。

[請求項5] 解析装置で実行される解析方法であって、

解析対象プログラムと同一または類似のプログラムに含まれる既知の動作を実行して出力される値である引数と、該既知の動作の識別子と、該引数と同じ値を保持する中間コード変数を識別する中間コード変数ＩＤと、に基づき対応表を作成する工程と、

前記解析対象プログラムが実行する所定の動作に対応する前記中間コード変数ＩＤを取得し、前記対応表に基づき該中間コード変数ＩＤが識別する前記引数を読み出す工程と、

読出部が読み出した前記引数を用いて、前記解析対象プログラムが稼働する仮想ＣＰＵのレジスタもしくはメモリの情報のいずれか１つまたは両方を取り出す工程と、

を含むことを特徴とする解析方法。

[請求項6] コンピュータを請求項１～４に記載の解析装置として機能させるための解析プログラム。

[図1]

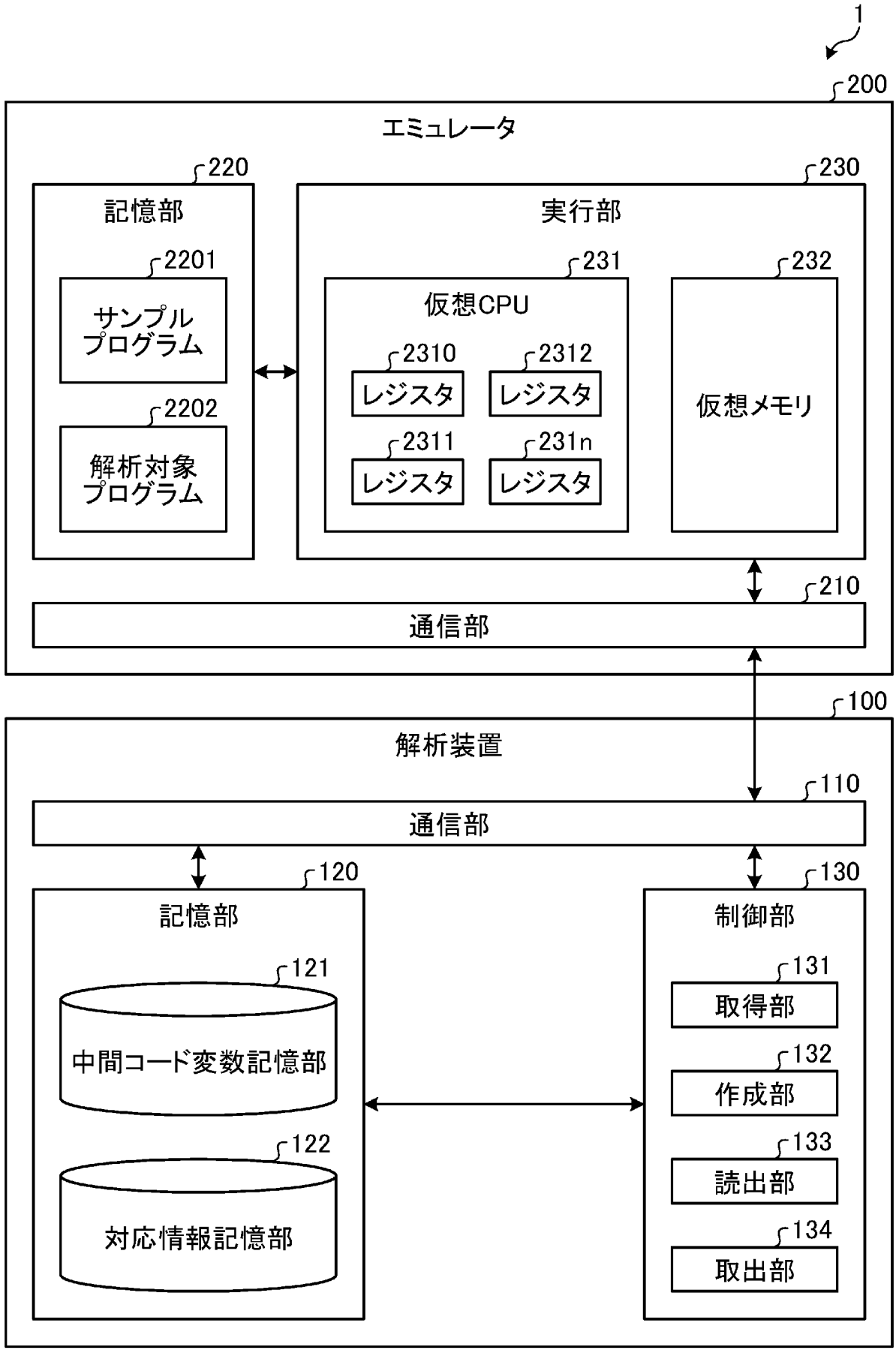
2201a

```
int main(void)
{

    sys_call1(0x12345678, 0xabcdefg, 0xdeadbeef);
    sys_call2(0x8badf00d, 0x87654321);

    return 0;
}
```

[図2]

中間コード変数ID	中間コード変数名	値	…
0	rax	1	…
1	rdi	0x12345678	…
2	rsi	0xabcdefg	…
3	tmp13	0x0	…
4	tmp8	0x12345678	…
…	…	…	…

[図3]

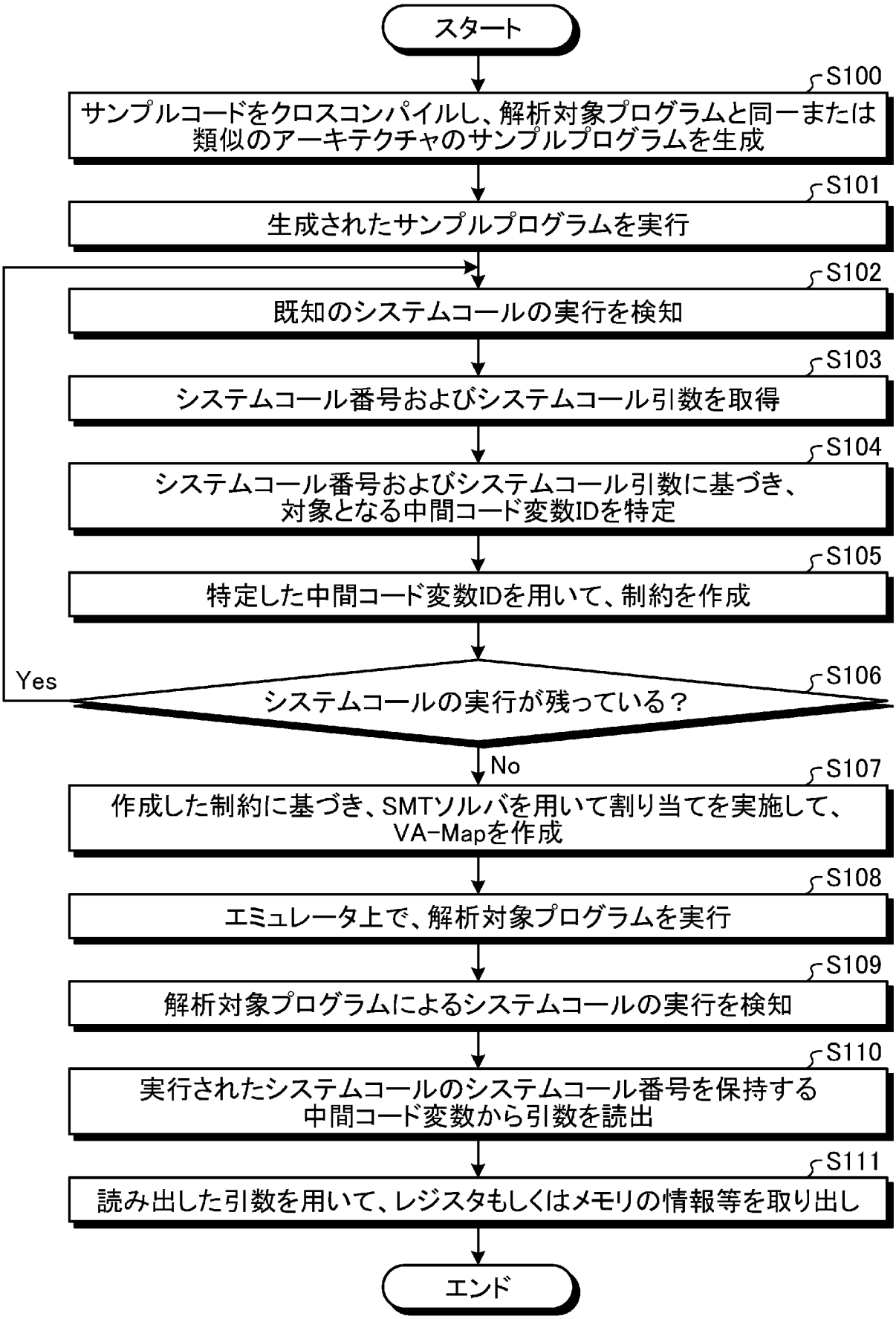

1
200
エミュレータ
220
記憶部
2201
サンプル
プログラム
2202
解析対象
プログラム
230
実行部
231
仮想CPU
2310
レジスタ
2312
レジスタ
2311
レジスタ
231n
レジスタ
232
仮想メモリ
210
通信部
100
解析装置
110
通信部
120
記憶部
121
中間コード変数記憶部
122
対応情報記憶部
130
制御部
131
取得部
132
作成部
133
読出部
134
取出部

[図4]

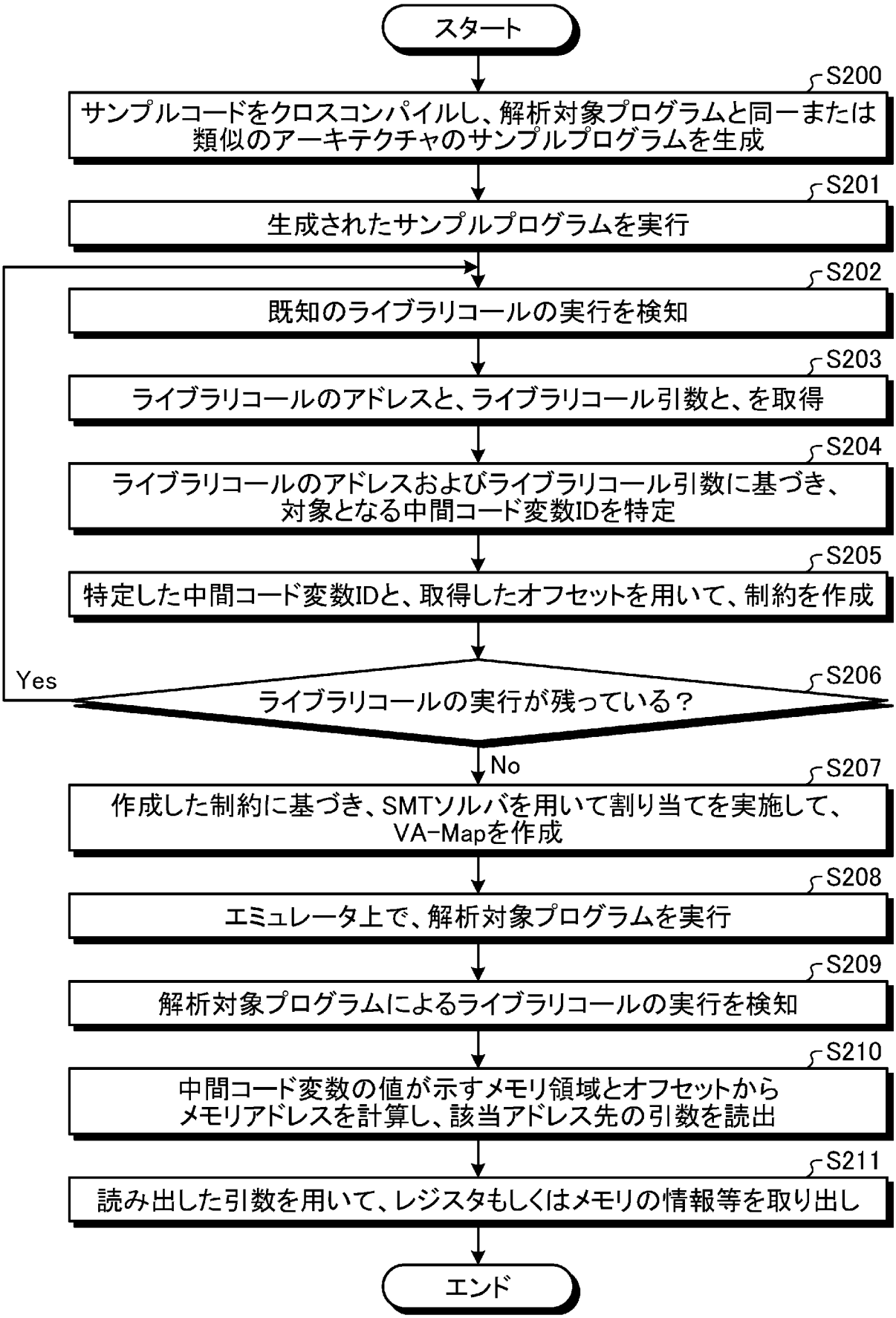
2202a

```
IN:
0x0045e260: b8 3f 00 00 00      mov     eax, 0x3f
0x0045e265: 0f 05               syscall

OP:
...

---- 0000000045e260 0000000000000000
mov_i64 tmp0,$0x3f
ext32u_i64 rax,tmp0

---- 0000000045e265 0000000000000000
mov_i64 tmp3,$0x45e265
st_i64 tmp3,env,$0x80
mov_i32 tmp11,$0x2
call syscall,$0x0,$0,env,tmp11
call rechecking_single_step,$0x0,$0,env
exit_tb $0x0
set_label $L0
exit_tb $0x7fc3cc011cc3
```

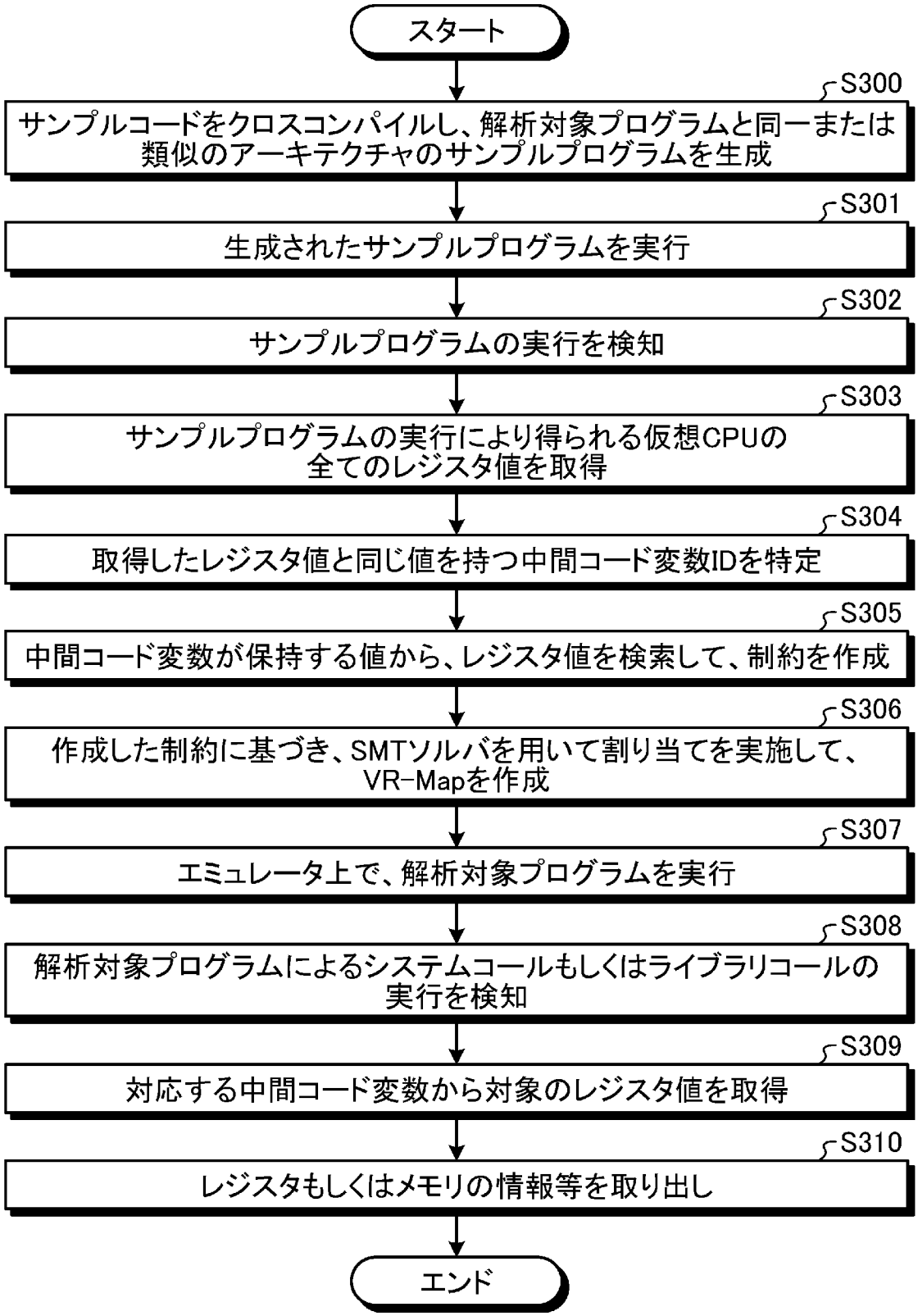
[図5]


スタート
S100
サンプルコードをクロスコンパイルし、解析対象プログラムと同一または類似のアーキテクチャのサンプルプログラムを生成
S101
生成されたサンプルプログラムを実行
S102
既知のシステムコールの実行を検知
S103
システムコール番号およびシステムコール引数を取得
S104
システムコール番号およびシステムコール引数に基づき、対象となる中間コード変数IDを特定
S105
特定した中間コード変数IDを用いて、制約を作成
S106
システムコールの実行が残っている？
Yes
No
S107
作成した制約に基づき、SMTソルバを用いて割り当てを実施して、VA-Mapを作成
S108
エミュレータ上で、解析対象プログラムを実行
S109
解析対象プログラムによるシステムコールの実行を検知
S110
実行されたシステムコールのシステムコール番号を保持する中間コード変数から引数を読出
S111
読み出した引数を用いて、レジスタもしくはメモリの情報等を取り出し
エンド

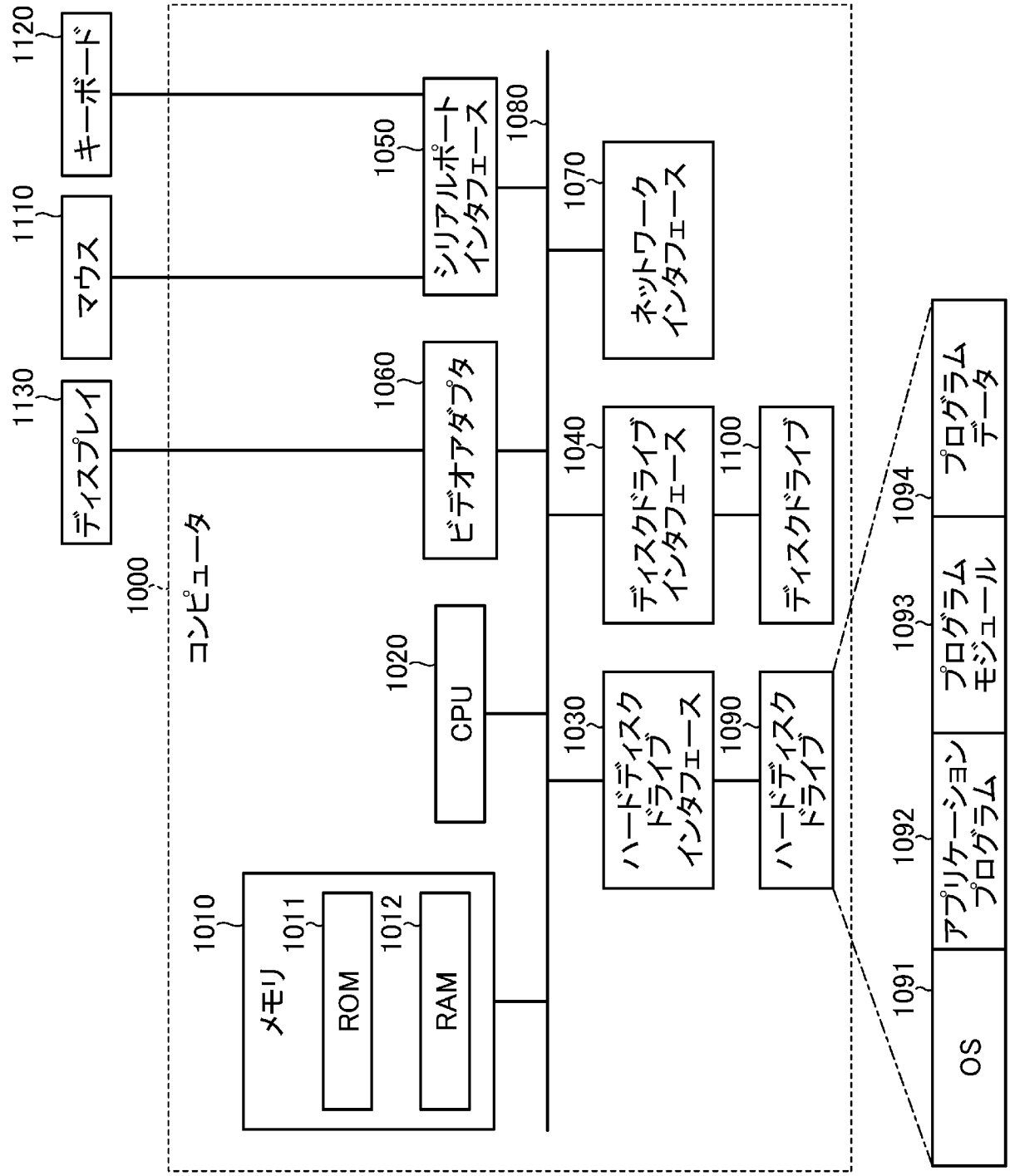
[図6]


スタート
S200
サンプルコードをクロスコンパイルし、解析対象プログラムと同一または類似のアーキテクチャのサンプルプログラムを生成
S201
生成されたサンプルプログラムを実行
S202
既知のライブラリコールの実行を検知
S203
ライブラリコールのアドレスと、ライブラリコール引数と、を取得
S204
ライブラリコールのアドレスおよびライブラリコール引数に基づき、対象となる中間コード変数IDを特定
S205
特定した中間コード変数IDと、取得したオフセットを用いて、制約を作成
S206
ライブラリコールの実行が残っている？
Yes
No
S207
作成した制約に基づき、SMTソルバを用いて割り当てを実施して、VA-Mapを作成
S208
エミュレータ上で、解析対象プログラムを実行
S209
解析対象プログラムによるライブラリコールの実行を検知
S210
中間コード変数の値が示すメモリ領域とオフセットからメモリアドレスを計算し、該当アドレス先の引数を読出
S211
読み出した引数を用いて、レジスタもしくはメモリの情報等を取り出し
エンド

[図7]


スタート
S300
サンプルコードをクロスコンパイルし、解析対象プログラムと同一または類似のアーキテクチャのサンプルプログラムを生成
S301
生成されたサンプルプログラムを実行
S302
サンプルプログラムの実行を検知
S303
サンプルプログラムの実行により得られる仮想CPUの全てのレジスタ値を取得
S304
取得したレジスタ値と同じ値を持つ中間コード変数IDを特定
S305
中間コード変数が保持する値から、レジスタ値を検索して、制約を作成
S306
作成した制約に基づき、SMTソルバを用いて割り当てを実施して、VR-Mapを作成
S307
エミュレータ上で、解析対象プログラムを実行
S308
解析対象プログラムによるシステムコールもしくはライブラリコールの実行を検知
S309
対応する中間コード変数から対象のレジスタ値を取得
S310
レジスタもしくはメモリの情報等を取り出し
エンド

[図8]


1000
コンピュータ
1010
メモリ
1011
ROM
1012
RAM
1020
CPU
1030
ハードディスクドライブインタフェース
1040
ディスクドライブインタフェース
1050
シリアルポートインタフェース
1060
ビデオアダプタ
1070
ネットワークインタフェース
1080
1090
ハードディスクドライブ
1100
ディスクドライブ
1091
OS
1092
アプリケーションプログラム
1093
プログラムモジュール
1094
プログラムデータ
1110
マウス
1120
キーボード
1130
ディスプレイ

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2022/036020

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/455(2006.01)i
FI: G06F9/455

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)

G06F9/44-9/455

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

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2022
Registered utility model specifications of Japan 1996-2022
Published registered utility model applications of Japan 1994-2022

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	DOVGALYUK, P. et al. QEMU-Based Framework for Non-intrusive Virtual Machine Instrumentation and Introspection. Proceedings of the 2017, 11th Joint Meeting on Foundations of Software Engineering. 21 August 2017, pages 944-948, [retrieved on 28 November 2022], Internet: <URL: https://dl.acm.org/doi/pdf/10.1145/3106237.3122817>, <DOI: 10.1145/3106237.3122817> entire text, all drawings	1-6
A	US 2021/0312048 A1 (PALO ALTO NETWORKS, INC.) 07 October 2021 (2021-10-07) entire text, all drawings	1-6
A	JP 2017-517821 A (THE CHARLES STARK DRAPER LABORATORY, INC.) 29 June 2017 (2017-06-29) entire text, all drawings	1-6

☐ 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
28 November 2022	**13 December 2022**
Name and mailing address of the ISA/JP **Japan Patent Office (ISA/JP)** **3-4-3 Kasumigaseki, Chiyoda-ku, Tokyo 100-8915** **Japan**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/JP2022/036020

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2021/0312048 A1	07 October 2021	US 2021/0064753 A1 entire text, all drawings	
JP 2017-517821 A	29 June 2017	US 2015/0363196 A1 entire text, all drawings WO 2015/191746 A1 CN 106537332 A	

国際調査報告

国際出願番号
PCT/JP2022/036020

A. 発明の属する分野の分類（国際特許分類（ＩＰＣ））

G06F 9/455(2006.01)i
FI: G06F9/455

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））

G06F9/44-9/455

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	１９７１－２０２２年
日本国実用新案登録公報	１９９６－２０２２年
日本国登録実用新案公報	１９９４－２０２２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	Dovgalyuk, P., et al., QEMU-Based Framework for Non-intrusive Virtual Machine Instrumentation and Introspection, Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering , 2017.08.21, pages 944-948, [検索日 2022.11.28], インターネット：<URL:https://dl.acm.org/doi/pdf/10.1145/3106237.3122817>, <DOI: 10.1145/3106237.3122817> 全文、全図	1-6
A	US 2021/0312048 A1 (PALO ALTO NETWORKS, INC.) 07.10.2021 (2021-10-07) 全文、全図	1-6
A	JP 2017-517821 A (ザ・チャールズ・スターク・ドレイパー・ラボラトリー・インコーポレイテッド) 29.06.2017 (2017-06-29) 全文、全図	1-6

☐ 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抵触するものではなく、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	国際調査報告の発送日
28.11.2022	13.12.2022

名称及びあて先
日本国特許庁(ISA/JP)
〒100-8915
日本国
東京都千代田区霞が関三丁目４番３号

権限のある職員（特許庁審査官）
北川　純次 5B 3650

電話番号 03-3581-1101 内線 3545

国際調査報告
パテントファミリーに関する情報

国際出願番号
PCT/JP2022/036020

引用文献	公表日	パテントファミリー文献	公表日
US 2021/0312048 A1	07.10.2021	US 2021/0064753 A1 全文、全図	
JP 2017-517821 A	29.06.2017	US 2015/0363196 A1 全文、全図 WO 2015/191746 A1 CN 106537332 A	