(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6079868号
(P6079868)
(45)発行日　平成29年2月15日(2017.2.15)　(24)登録日　平成29年1月27日(2017.1.27)

(51)Int.Cl.　F I
G06F 11/07 (2006.01)　G06F 11/07 178
G06F 11/07 190
G06F 11/07 140N

請求項の数 6　(全 24 頁)

(21)出願番号　特願2015-506440 (P2015-506440)
(86)(22)出願日　平成25年3月19日 (2013.3.19)
(86)国際出願番号　PCT/JP2013/057798
(87)国際公開番号　WO2014/147742
(87)国際公開日　平成26年9月25日 (2014.9.25)
審査請求日　平成27年5月14日 (2015.5.14)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100092152
弁理士　服部　毅巖
(72)発明者　大平　良行
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

審査官　多胡　滋

最終頁に続く

(54)【発明の名称】エラー分析方法、情報処理装置およびエラー分析プログラム

(57)【特許請求の範囲】
【請求項１】
複数のプログラムモジュールを実行するコンピュータが行うエラー分析方法であって、
前記複数のプログラムモジュールが実行される間、前記コンピュータのプロセッサが備える複数のレジスタの少なくとも一部について、当該少なくとも一部のレジスタへの値の書き込みを検出し、検出した書き込みを示す履歴情報であって、前記コンピュータが備えるメモリのうち書き込まれた値を記憶していた記憶領域を示す情報を含む履歴情報を記録し、
前記複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、前記少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定し、
前記参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、前記参照されたレジスタに書き込まれた値を記憶していた前記メモリの記憶領域を特定し、特定した記憶領域に基づいて、前記複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する、
エラー分析方法。
【請求項２】
複数のプログラムモジュールを実行するコンピュータが行うエラー分析方法であって、
前記複数のプログラムモジュールが実行される間、前記コンピュータのプロセッサが備える複数のレジスタの少なくとも一部について、当該少なくとも一部のレジスタへの値の書き込みを検出し、検出した書き込みを示す履歴情報を記録し、

前記複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、前記少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定し、
前記参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、前記参照されたレジスタに書き込まれた値が正常か判定し、
正常と判定された場合、前記参照されたレジスタに値を書き込むときに参照された他のレジスタを特定し、前記参照された他のレジスタへの値の書き込みを示す履歴情報に基づいて、前記複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する、
エラー分析方法。
【請求項３】
書き込みを検出する前記少なくとも一部のレジスタを、前記プロセッサが備える前記複数のレジスタのうちベースアドレスを記憶するレジスタに限定する、
請求項１または２記載のエラー分析方法。
【請求項４】
前記少なくとも一部のレジスタへの値の書き込みの検出は、前記プロセッサが備えるエミュレータを用いて行い、
前記エラーの原因となったプログラムモジュールの判定は、前記プロセッサに実行させる他のプログラムモジュールを用いて行う、
請求項１乃至３の何れか一項に記載のエラー分析方法。
【請求項５】
複数のプログラムモジュールを実行する情報処理装置であって、
複数のレジスタを備えるプロセッサと、
メモリと、を有し、
前記プロセッサは、
前記複数のプログラムモジュールが実行される間、前記複数のレジスタの少なくとも一部について、当該少なくとも一部のレジスタへの値の書き込みを検出し、検出した書き込みを示す履歴情報であって、前記メモリのうち書き込まれた値を記憶していた記憶領域を示す情報を含む履歴情報を記録し、
前記複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、前記少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定し、前記参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、前記参照されたレジスタに書き込まれた値を記憶していた前記メモリの記憶領域を特定し、特定した記憶領域に基づいて、前記複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する、
情報処理装置。
【請求項６】
複数のプログラムモジュールを実行するコンピュータに、
前記コンピュータのプロセッサが備える複数のレジスタの少なくとも一部について、前記複数のプログラムモジュールが実行される間に行われた、当該少なくとも一部のレジスタへの値の書き込みを示す履歴情報であって、前記コンピュータが備えるメモリのうち書き込まれた値を記憶していた記憶領域を示す情報を含む履歴情報を記録し、
前記複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、前記少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定し、
前記参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、前記参照されたレジスタに書き込まれた値を記憶していた前記メモリの記憶領域を特定し、特定した記憶領域に基づいて、前記複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する、
処理を実行させるエラー分析プログラム。

【発明の詳細な説明】

【技術分野】
【０００１】
本発明はエラー分析方法、情報処理装置およびエラー分析プログラムに関する。
【背景技術】
【０００２】
コンピュータがプログラムを実行しているとき、プログラムの不備によって意図しないエラーが発生することがある。エラーの例としては、存在しないメモリ領域を示すアドレスやアクセスが許可されていない保護されたメモリ領域のアドレスを指定してメモリにアクセスしようとしたときに発生するメモリアクセスエラーが挙げられる。エラーが検出されると、例えば、プロセッサに対して割り込み信号が入力され、現在実行中の処理が中断されてエラーの種類に応じた復旧処理（例外処理と言うことがある）が行われる。
【０００３】
複数のプログラムモジュールが実行されるコンピュータでは、復旧処理の中で、エラーの原因となった不備のあるプログラムモジュールを強制的に停止させて実行されないようにすることが考えられる。しかし、エラーが検出された時点で実行されていたプログラムモジュールが、エラーの原因を作った不備のあるものとは限らない。エラー原因の判定を誤ると、不備のあるプログラムモジュールが停止されずに不備のないプログラムモジュールが停止され、コンピュータの動作が意図せず不安定になるおそれがある。そこで、エラーの原因となったプログラムモジュールを分析する方法が提案されている。
【０００４】
例えば、現在実行しているプログラムがユーザプログラムとシステムプログラムとの間で遷移するコンピュータにおいて、システム制御管理方法が提案されている。このシステム制御管理方法では、ユーザプログラムからシステムプログラムへの遷移、および、システムプログラムからユーザプログラムへの遷移が発生したときに、プロセスの属性情報を時間情報と対応付けてトレース用バッファに記憶する。プログラム実行中にエラーが発生すると、トレース用バッファに記憶された属性情報を時系列に辿ることで、エラー原因がユーザプログラムにあるかシステムプログラムにあるかを判定する。
【０００５】
なお、プログラム開発におけるデバッグ支援に関して、次のようなものが提案されている。例えば、レジスタまたはメモリへの書き込み命令が実行される直前にトレースデータを出力するようなデバッグ用プログラムを生成し、ユーザによるデバッグ作業を容易にするデバッグシステムが提案されている。このデバッグシステムは、デバッグ用プログラムが異常停止したとき、トレースデータに基づいて命令を１つずつ逆順に実行していく。
【０００６】
また、メモリ上のスタック領域へのアクセスが適切であるか確認できるようにするトレース装置が提案されている。このトレース装置は、プログラムの中から、スタック領域内の位置を指し示すスタックポインタを変更する命令を検出し、当該命令が実行されたときにトレースデータを出力する。また、使用するメモリの記憶領域を変更しながらプログラムを複数回テストし、参照アドレスやメモリアクセス回数や命令の実行順序が全てのテストを通じて等しいかを確認するデバッグ方法が提案されている。
【先行技術文献】
【特許文献】
【０００７】
【特許文献１】特開平７－２９５８６２号公報
【特許文献２】特開平９－６６４７号公報
【特許文献３】特開平１１－２０３１６６号公報
【特許文献４】国際公開第２００９／０７５１１６号
【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、上記の特許文献１に記載されたシステム制御管理方法のように、プログラムモジュールの間の呼び出し関係をトレースするだけの方法では、エラーの原因となったプログラムモジュールの判定精度に改善の余地がある。例えば、あるプログラムモジュールにおいて、受け取った値（例えば、アドレス）が不適切であるために実行中にエラーが発生する場合が考えられる。この場合に、呼び出し関係をトレースするだけでは、エラーが検出されたプログラムモジュールと呼び出し関係にある他のプログラムモジュールの中から、不適切な値を用意したものを精度よく判定することは容易でなかった。また、呼び出し関係にあるプログラムモジュールの中に不適切な値を用意したものがあるとは限らず、エラーの原因となったプログラムモジュールを発見できない可能性もあった。

【０００９】

１つの側面では、本発明は、エラーの原因となったプログラムモジュールの判定精度を向上させることができるエラー分析方法、情報処理装置およびエラー分析プログラム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】

【００１０】

１つの態様では、複数のプログラムモジュールを実行するコンピュータが行うエラー分析方法が提供される。エラー分析方法では、複数のプログラムモジュールが実行される間、コンピュータのプロセッサが備える複数のレジスタの少なくとも一部について、当該少なくとも一部のレジスタへの値の書き込みを検出し、検出した書き込みを示す履歴情報を記録する。複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定する。参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する。

【００１１】

また、１つの態様では、複数のプログラムモジュールを実行する情報処理装置が提供される。情報処理装置は、複数のレジスタを備えるプロセッサと、複数のレジスタの少なくとも一部について、複数のプログラムモジュールが実行される間に行われた、当該少なくとも一部のレジスタへの値の書き込みを示す履歴情報を記憶する記憶部と、を有する。プロセッサは、複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定し、参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する。

【００１２】

また、１つの態様では、複数のプログラムモジュールを実行するコンピュータに、以下の処理を実行させるエラー分析プログラムが提供される。コンピュータのプロセッサが備える複数のレジスタの少なくとも一部について、複数のプログラムモジュールが実行される間に行われた、当該少なくとも一部のレジスタへの値の書き込みを示す履歴情報を記録する。複数のプログラムモジュールの中に含まれる一の命令でエラーが発生すると、少なくとも一部のレジスタのうち当該命令で参照されたレジスタを特定する。参照されたレジスタへの値の書き込みを示す履歴情報に基づいて、複数のプログラムモジュールのうちエラーの原因となったプログラムモジュールを判定する。

【発明の効果】

【００１３】

１つの側面では、エラーの原因となったプログラムモジュールの判定精度が向上する。

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】

【００１４】

【図１】第１の実施の形態の情報処理装置の例を示す図である。

【図２】情報処理装置のハードウェア例を示すブロック図である。

【図３】プロセッサのハードウェア例を示す図である。
【図４】複数のプログラムの間の呼び出し例を示す図である。
【図５】メモリアクセスエラーの発生例を示す図である。
【図６】情報処理装置の機能例を示すブロック図である。
【図７】管理情報の全体構造の例を示す図である。
【図８】システム情報の構造例を示す図である。
【図９】呼び出し関係情報の構造例を示す図である。
【図１０】プロセス情報の構造例を示す図である。
【図１１】実行ログの構造例を示す図である。
【図１２】ログ採取の手順例を示すフローチャートである。
【図１３】エラー分析の手順例を示すフローチャートである。
【図１４】エラー分析の手順例を示すフローチャート（続き）である。
【図１５】ジャンプ先での実行エラーの発生例を示す図である。
【図１６】他の実行ログの構造例を示す図である。
【図１７】他のログ採取の手順例を示すフローチャートである。
【発明を実施するための形態】
【００１５】
以下、本実施の形態を図面を参照して説明する。
［第１の実施の形態］
図１は、第１の実施の形態の情報処理装置の例を示す図である。
【００１６】
第１の実施の形態に係る情報処理装置１０は、プログラムモジュール１３－１～１３－４を含む複数のプログラムモジュールを実行する。情報処理装置１０は、プロセッサ１１および記憶部１２を有する。プロセッサ１１は、レジスタＲａ，Ｒｂ，Ｒｃを含む複数のレジスタを備える。プロセッサ１１は、ＣＰＵ（Central Processing Unit）やＭＰＵ（Micro Processing Unit）と呼ばれるものでもよい。プロセッサ１１は、レジスタを用いてプログラムモジュール１３－１～１３－４を実行する。記憶部１２は、プログラムモジュール１３－１～１３－４が実行されている間に生成される履歴情報１４を記憶する。記憶部１２は、ＲＡＭ（Random Access Memory）などの揮発性メモリでもよいし、ＨＤＤ（Hard Disk Drive）やフラッシュメモリなどの不揮発性の記憶装置であってもよい。
【００１７】
プログラムモジュール１３－１～１３－４は、プロセッサ１１が実行可能な形式の命令の集合である。命令には、メモリからレジスタへのロード命令、レジスタからメモリへのストア命令、算術演算命令や論理演算命令などの演算命令、分岐命令などの制御命令が含まれ得る。プログラムモジュール１３－１～１３－４は、オペレーティングシステム（ＯＳ：Operating System）プログラムでもよいし、ＯＳプログラム以外のユーザプログラムでもよい。また、プログラムモジュール１３－１～１３－４は、呼び出し関係によって直列に関連付けられていてもよいし、互いに呼び出し関係になくてもよい。
【００１８】
一例として、プログラムモジュール１３－１～１３－４に次のような動作が定義されているとする。プログラムモジュール１３－１により、メモリ上にメモリ領域＃１が確保され、メモリ領域＃１に値１が記憶される。プログラムモジュール１３－２により、メモリ領域＃１に記憶されている値１がレジスタＲｂに書き込まれる。プログラムモジュール１３－２は、プログラムモジュール１３－１から呼び出されるものでなくてもよい。
【００１９】
また、プログラムモジュール１３－２からプログラムモジュール１３－３が呼び出される。プログラムモジュール１３－３により、メモリ領域＃２に記憶された値２がレジスタＲａに書き込まれ、メモリ領域＃３に記憶された値３がレジスタＲｃに書き込まれる。また、プログラムモジュール１３－３からプログラムモジュール１３－４が呼び出される。プログラムモジュール１３－４により、レジスタＲｂを参照した処理が行われる。

【００２０】
履歴情報１４は、プログラムモジュール１３－１～１３－４が実行される間に行われたレジスタＲａ，Ｒｂ，Ｒｃへの値の書き込みを示す。例えば、プロセッサ１１が、レジスタＲａ，Ｒｂ，Ｒｃへの値の書き込みを検出し、履歴情報１４を生成して記憶部１２に格納する。プロセッサ１１による書き込みの検出は、プロセッサ１１が備えるエミュレータなどのハードウェアを用いて実現してもよいし、ソフトウェアを用いて実現してもよい。書き込みを検出するレジスタＲａ，Ｒｂ，Ｒｃは、プロセッサ１１が備える複数のレジスタのうち所定の性質を備えるものに限定してもよい。例えば、プロセッサ１１がメモリにアクセスするときに参照するベースアドレスを記憶するレジスタに限定してもよい。
【００２１】
一例として、履歴情報１４は、メモリ領域＃１からレジスタＲｂに値１がロードされたことを示す情報を含む。この情報は、プログラムモジュール１３－２が実行されたときに生成されて記録される。また、履歴情報１４は、メモリ領域＃２からレジスタＲａに値２がロードされたことを示す情報、および、メモリ領域＃３からレジスタＲｃに値３がロードされたことを示す情報を含む。この情報は、プログラムモジュール１３－３が実行されたときに生成されて記録される。レジスタＲａ，Ｒｂ，Ｒｃへの書き込みを示すこれらの情報は、好ましくは、実行順序が明確になるように記録される。
【００２２】
ここで、プログラムモジュール１３－１～１３－４の中に含まれる一の命令を実行したときにエラーが発生したとする。例えば、プロセッサ１１が一のレジスタの値に応じたメモリ領域にアクセスする命令を実行したときに、メモリアクセスエラーが発生した場合が考えられる。すると、プロセッサ１１は、その命令で参照されたレジスタを特定し、記憶部１２に記憶された履歴情報１４の中から、当該参照されたレジスタへの値の書き込みを示す情報を検索する。そして、プロセッサ１１は、検索された情報に基づいて、プログラムモジュール１３－１～１３－４のうちエラーの原因となったものを判定する。
【００２３】
一例として、プログラムモジュール１３－４に含まれるレジスタＲｂを参照する命令が実行されたときにエラーが発生したとする。すると、履歴情報１４の中から、メモリ領域＃１からレジスタＲｂに値１がロードされたことを示す情報が検索される。この値１が不適切であるために、レジスタＲｂを参照する命令でエラーが発生したと推定される。そこで、例えば、プロセッサ１１は、ＯＳがもつメモリ管理情報を参照して、メモリ領域＃１を使用しているプログラムモジュール１３－１を特定し、プログラムモジュール１３－１がエラーの原因となったプログラムモジュールであると判定する。
【００２４】
ただし、プロセッサ１１は、値１をレジスタＲｂに書き込んだプログラムモジュール１３－２を、エラーの原因となったプログラムモジュールであると判定してもよい。その場合、履歴情報１４には、レジスタＲａ，Ｒｂ，Ｒｃに値を書き込む命令またはその命令を含むプログラムモジュールを識別するための情報が含まれることが好ましい。
【００２５】
第１の実施の形態によれば、プログラムモジュール１３－１～１３－４が実行される間、レジスタＲａ，Ｒｂ，Ｒｃへの値の書き込みが検出されて履歴情報１４が記録される。そして、エラー原因を分析するとき、エラーが発生した命令で参照されているレジスタへの書き込みを示す情報が履歴情報１４から検索される。これにより、レジスタに記憶された不適切な値を用意したプログラムモジュールを特定でき、エラーの原因となったプログラムモジュールの判定精度を向上させることができる。また、エラーが検出されたプログラムモジュールとエラーの原因となったプログラムモジュールとの間に呼び出し関係がないときであっても、エラー原因を特定できる可能性が高くなる。
【００２６】
［第２の実施の形態］
第２の実施の形態を説明する。第２の実施の形態に係る情報処理装置１００は、ＯＳプ

ログラムおよび複数のユーザプログラムを実行する。情報処理装置１００は、ユーザが操作する端末装置（クライアントコンピュータと呼んでもよい）でもよいし、端末装置からアクセスされるサーバ装置（サーバコンピュータと呼んでもよい）であってもよい。

【００２７】

この情報処理装置１００は、プログラムの実行中にエラーが発生すると、エラーの原因を分析する。エラーの原因が一のユーザプログラムにあると判定した場合、情報処理装置１００はそのユーザプログラムを強制的に停止させる。この場合、他のユーザプログラムやＯＳプログラムの実行は継続し得る。一方、エラーの原因がＯＳプログラムにあると判定した場合、情報処理装置１００はＯＳを停止または再起動させる。この場合、情報処理装置１００の全てのプログラムの実行が停止する可能性がある。

【００２８】

なお、第２の実施の形態の「ユーザプログラム」は、アプリケーションプログラムやユーザライブラリなど、ユーザ空間で実行されるプログラムのモジュールである。また、第２の実施の形態の「ＯＳプログラム」は、ＯＳ空間で実行されるプログラムのモジュールである。第２の実施の形態の「ユーザプログラム」および「ＯＳプログラム」は、第１の実施の形態のプログラムモジュール１３－１～１３－４の一例である。

【００２９】

図２は、情報処理装置のハードウェア例を示すブロック図である。

情報処理装置１００は、ＣＰＵ１０１、ＲＡＭ１０２、ＨＤＤ１０３、画像信号処理部１０４、入力信号処理部１０５、媒体リーダ１０６および通信インタフェース１０７を有する。ＣＰＵ１０１は、第１の実施の形態のプロセッサ１１の一例であり、ＲＡＭ１０２やＨＤＤ１０３は、第１の実施の形態の記憶部１２の一例である。

【００３０】

ＣＰＵ１０１は、プログラムの命令を実行する演算回路を含むプロセッサである。ＣＰＵ１０１は、ＨＤＤ１０３に記憶されているプログラムやデータの少なくとも一部をＲＡＭ１０２にロードし、プログラムを実行する。なお、ＣＰＵ１０１は複数のプロセッサコアを備えてもよく、情報処理装置１００は複数のプロセッサを備えてもよく、以下で説明する処理を複数のプロセッサまたはプロセッサコアを用いて並列実行してもよい。また、複数のプロセッサの集合（マルチプロセッサ）を「プロセッサ」と呼んでもよい。

【００３１】

ＲＡＭ１０２は、ＣＰＵ１０１が実行するプログラムやＣＰＵ１０１が演算に用いるデータを一時的に記憶する揮発性メモリである。なお、情報処理装置１００は、ＲＡＭ以外の種類のメモリを備えてもよく、複数個のメモリを備えてもよい。

【００３２】

ＨＤＤ１０３は、ＯＳやファームウェアやアプリケーションソフトウェアなどのソフトウェアのプログラム、および、データを記憶する不揮発性の記憶装置である。なお、情報処理装置１００は、フラッシュメモリやＳＳＤ（Solid State Drive）などの他の種類の記憶装置を備えてもよく、複数の不揮発性の記憶装置を備えてもよい。

【００３３】

画像信号処理部１０４は、ＣＰＵ１０１からの命令に従って、情報処理装置１００に接続されたディスプレイ２１に画像を出力する。ディスプレイ２１としては、ＣＲＴ（Cathode Ray Tube）ディスプレイ、液晶ディスプレイ（ＬＣＤ：Liquid Crystal Display）、プラズマディスプレイ（ＰＤＰ：Plasma Display Panel）、有機ＥＬ（ＯＥＬＤ：Organic Electro-Luminescence）ディスプレイなどを用いることができる。

【００３４】

入力信号処理部１０５は、情報処理装置１００に接続された入力デバイス２２から入力信号を取得し、ＣＰＵ１０１に出力する。入力デバイス２２としては、マウスやタッチパネルやタッチパッドやトラックボールなどのポインティングデバイス、キーボード、リモートコントローラ、ボタンスイッチなどを用いることができる。また、情報処理装置１００に、複数の種類の入力デバイスが接続されてもよい。

【００３５】
媒体リーダ１０６は、記録媒体２３に記録されたプログラムやデータを読み取る駆動装置である。記録媒体２３として、例えば、フレキシブルディスク（ＦＤ：Flexible Disk）やＨＤＤなどの磁気ディスク、ＣＤ（Compact Disc）やＤＶＤ（Digital Versatile Disc）などの光ディスク、光磁気ディスク（ＭＯ：Magneto-Optical disk）、半導体メモリなどを使用できる。媒体リーダ１０６は、例えば、記録媒体２３から読み取ったプログラムやデータをＲＡＭ１０２またはＨＤＤ１０３に格納する。
【００３６】
通信インタフェース１０７は、ネットワークを介して他の情報処理装置と通信を行うインタフェースである。通信インタフェース１０７は、ケーブルにより通信装置に接続する有線インタフェースでもよいし、無線で基地局に接続する無線インタフェースでもよい。
【００３７】
ただし、情報処理装置１００は、媒体リーダ１０６を備えていなくてもよい。また、端末装置からネットワーク経由で情報処理装置１００を制御できる場合には、情報処理装置１００は、画像信号処理部１０４や入力信号処理部１０５を備えなくてもよい。
【００３８】
図３は、プロセッサのハードウェア例を示す図である。
ＣＰＵ１０１は、レジスタファイル１１１、演算回路１１２および命令エミュレータ１１３を有する。レジスタファイル１１１は、複数のレジスタを有する。この複数のレジスタには、ベースレジスタとしてのレジスタＢＲ１，ＢＲ２，ＢＲ３，・・・と、汎用レジスタとしてのレジスタＲ１，Ｒ２，・・・が含まれる。レジスタＢＲ１，ＢＲ２，ＢＲ３は、第１の実施の形態のレジスタＲａ，Ｒｂ，Ｒｃの一例である。
【００３９】
レジスタＢＲ１，ＢＲ２，ＢＲ３は、ベースアドレスを記憶する。ベースアドレスは、ＲＡＭ１０２上の基準となる位置を指し示すメモリアドレスである。プログラムの命令は、アクセスしたい位置を特定するために、絶対アドレスではなくベースアドレスからの差分を指定することができる。ベースアドレスは、例えば、ＲＡＭ１０２にテーブル構造のデータが記憶されているときに用いられる。その場合、テーブルの先頭アドレスがレジスタＢＲ１，ＢＲ２，ＢＲ３の何れかにベースアドレスとして記憶され、テーブルの先頭からアクセスしたいデータが格納されている位置までの差分が命令によって指定される。
【００４０】
レジスタＲ１，Ｒ２は、記憶するデータの種類が限定されておらず、演算回路１１２が多目的に使用できるレジスタである。レジスタＲ１，Ｒ２に記憶され得るデータには、算術演算や論理演算のオペランドや、プログラムの中のジャンプ先の命令を示す命令アドレスなどが含まれる。なお、汎用レジスタの一部または汎用レジスタ以外のレジスタの一部は、相対的なアドレス差（インデックス）を記憶するインデックスレジスタとして用いられることがある。また、ベースレジスタは予め固定されていてもよいし、レジスタファイル１１１が有するレジスタの中からＣＰＵ１０１が選択するようにしてもよい。
【００４１】
演算回路１１２は、命令エミュレータ１１３から取得した命令を実行する。例えば、演算回路１１２は、ロード命令に従い、ＲＡＭ１０２からレジスタＢＲ１，ＢＲ２，ＢＲ３，Ｒ１，Ｒ２にデータを読み込む。また、演算回路１１２は、算術演算命令や論理演算命令に従い、レジスタＲ１，Ｒ２に記憶されたデータを書き換える。また、演算回路１１２は、ストア命令に従い、レジスタＲ１，Ｒ２からＲＡＭ１０２にデータを書き戻す。
【００４２】
命令エミュレータ１１３は、ＲＡＭ１０２からプログラムの命令を読み込み、演算回路１１２に命令を出力する。ただし、ＲＡＭ１０２に記憶されたプログラムが、演算回路１１２が解釈できる命令セットと異なる命令セットを用いて記述されている場合がある。その場合、命令エミュレータ１１３は、ＲＡＭ１０２から読み込んだ命令を、演算回路１１２が解釈可能な命令に変換して演算回路１１２に出力する。

【００４３】
また、命令エミュレータ１１３は、レジスタＢＲ１，ＢＲ２，ＢＲ３へのベースアドレスの書き込みを伴う命令（例えば、ＲＡＭ１０２からレジスタＢＲ１，ＢＲ２，ＢＲ３へのロード命令）が実行されることを検出する。すると、命令エミュレータ１１３は、ベースアドレスの書き込みを示す実行ログをＲＡＭ１０２に記録する。なお、上記の命令変換機能およびログ記録機能は、ハードウェアとして実現することもできるし、命令エミュレータ１１３に記憶したファームウェアを用いて実現することもできる。
【００４４】
次に、ユーザプログラムおよびＯＳプログラムの実行の流れの例を説明する。
図４は、複数のプログラムの間の呼び出し例を示す図である。
１つのプログラムからは、並列に実行される複数のプロセスを起動することが可能である。例えば、情報処理装置１００は、あるプログラムに従ってデータＡを処理するプロセスＡと、同じプログラムに従ってデータＢを処理するプロセスＢとを、並行して実行することができる。また、１つのプロセスの中では、複数のプログラムが起動されることがある。例えば、プログラムＡがプログラムＢを呼び出したとき、呼び出されたプログラムＢが呼び出し元のプログラムＡと同じプロセスの中で実行されることがある。すなわち、プログラムとプロセスは多対多に対応付けられる。図４は、４つのユーザプログラムと１つのＯＳプログラムとが同一のプロセスの中で実行される例を示している。
【００４５】
例えば、ユーザプログラムＡが起動されると、ＣＰＵ１０１はユーザプログラムＡの命令に従って、ＲＡＭ１０２にユーザプログラムＡのためのメモリ領域を確保し、データを生成して確保したメモリ領域に格納する。そして、ユーザプログラムＡからユーザプログラムＢが呼び出される。ユーザプログラムＢが呼び出されると、ＣＰＵ１０１はユーザプログラムＢの命令に従って、ユーザプログラムＡのデータをＲＡＭ１０２から読み出して処理を行う。そして、ユーザプログラムＢからユーザプログラムＣが呼び出される。
【００４６】
ユーザプログラムＣが呼び出されると、ＣＰＵ１０１はユーザプログラムＣの命令に従って、ＲＡＭ１０２にユーザプログラムＣのためのメモリ領域を確保し、データを生成して確保したメモリ領域に格納する。そして、ユーザプログラムＣからユーザプログラムＤが呼び出される。ユーザプログラムＤが呼び出されると、ＣＰＵ１０１はユーザプログラムＤの命令に従って処理を行う。そして、ユーザプログラムＤからＯＳプログラムが呼び出される。ユーザプログラムからＯＳプログラムの呼び出しは、例えば、スーパーバイザーコール（システムコールとも呼ばれ得る）によって行われる。
【００４７】
ＯＳプログラムが呼び出されると、ＣＰＵ１０１はＯＳプログラムの命令に従って、ユーザプログラムＣのデータをＲＡＭ１０２から読み出して処理を行う。ＯＳプログラムの処理が正常に終了すると、呼び出しとは逆方向にユーザプログラムＤ、ユーザプログラムＣ、ユーザプログラムＢ、ユーザプログラムＡの順に制御が戻される。なお、ＯＳプログラムおよび各ユーザプログラムに割り当てられたメモリ領域（使用中のメモリ領域）は、ＯＳによって管理されており、ＯＳがもつメモリ管理情報に登録されている。
【００４８】
このように、あるプログラムによって生成されてＲＡＭ１０２に格納されたデータが、他のプログラムによって読み出されて参照されることがある。なお、データを生成するプログラムとそのデータを参照するプログラムとは、直列の呼び出し関係で関連付けられている（呼び出す側と呼び出される側の関係にある）とは限らない。例えば、ある親プログラムから２つの子プログラムが呼び出されるとき、一方の子プログラムによって生成されたデータが他方の子プログラムによって参照される場合が考えられる。
【００４９】
図５は、メモリアクセスエラーの発生例を示す図である。
ここでは、プログラム＃１からプログラム＃２が呼び出される場合を考える。プログラ

ム＃１，＃２は、ユーザプログラムでもよいしＯＳプログラムでもよい。

【００５０】

プログラム＃１には、ＲＡＭ１０２のメモリ領域＃１からレジスタＢＲ２にデータを読み込むロード命令が含まれる。メモリ領域＃１にはベースアドレス＃１が格納されていることを想定している。プログラム＃２には、ＲＡＭ１０２のメモリ領域＃２からレジスタＢＲ１にデータを読み込むロード命令が含まれる。このロード命令は、レジスタＢＲ２に格納されたベースアドレス＃１を基点として、メモリ領域＃２の位置を指定している。すなわち、このロード命令は、参照するベースレジスタ（レジスタＢＲ２）、および、ベースアドレス＃１とメモリ領域＃２の先頭アドレスとの差分を指定している。メモリ領域＃２にはベースアドレス＃２が格納されていることを想定している。

【００５１】

また、プログラム＃２には、ＲＡＭ１０２のメモリ領域＃３からレジスタＢＲ３にデータを読み込むロード命令が含まれる。このロード命令は、ベースアドレス＃１を基点として、メモリ領域＃３の位置を指定する。メモリ領域＃３にはベースアドレス＃３が格納されていることを想定している。また、プログラム＃２には、ＲＡＭ１０２からレジスタＲ１にデータを読み込むロード命令が含まれる。このロード命令は、ベースアドレス＃１を基点として、データが格納されたメモリ領域の位置を指定する。

【００５２】

ここで、ＣＰＵ１０１がレジスタＲ１へのロード命令を実行するときに、メモリアクセスエラーが発生したとする。メモリアクセスエラーが発生する場合として、例えば、ＲＡＭ１０２にアクセスするときに指定するアドレスが、ＲＡＭ１０２に存在しないアドレスであった場合が考えられる。また、他の場合として、例えば、アクセスしようとしたメモリ領域が、プログラム＃２からのアクセスが許可されていない場合が考えられる。

【００５３】

メモリアクセスエラーが発生したこのロード命令は、レジスタＢＲ２を参照している。よって、メモリアクセスエラーの原因として、レジスタＢＲ２に格納されたベースアドレス＃１が異常なアドレスであることが考えられる。そこで、ベースアドレス＃１が異常である可能性があるとき、このベースアドレス＃１を用意したプログラムをメモリアクセスエラーの原因となったプログラムと推定できる。第２の実施の形態では、情報処理装置１００は、ベースアドレス＃１が格納されていたＲＡＭ１０２のメモリ領域＃１を使用するプログラムを、メモリアクセスエラーの原因となったプログラムと判定する。

【００５４】

なお、ベースアドレス＃１を用意したプログラムは、ベースアドレス＃１をレジスタＢＲ２に読み込んだプログラム＃１自身であることもある。また、ベースアドレス＃１を用意したプログラムは、プログラム＃１を呼び出したプログラムであることもあり、プログラム＃１と呼び出し関係にない他のプログラムであることもある。

【００５５】

図６は、情報処理装置の機能例を示すブロック図である。

情報処理装置１００は、ログ記録部１２１、命令変換部１２２、記憶部１２３、割り込み検出部１２４、エラー分析部１２５、ユーザプログラム停止部１２６およびシステム停止部１２７を有する。ログ記録部１２１および命令変換部１２２は、例えば、命令エミュレータ１１３が備えるハードウェアを用いて実現される。記憶部１２３は、例えば、ＲＡＭ１０２またはＨＤＤ１０３に確保した記憶領域として実現される。割り込み検出部１２４、エラー分析部１２５、ユーザプログラム停止部１２６およびシステム停止部１２７は、例えば、ＯＳに含まれるソフトウェアのモジュールとして実現される。

【００５６】

ログ記録部１２１は、レジスタＢＲ１，ＢＲ２，ＢＲ３へのベースアドレスの書き込みを検出し、ベースアドレスの書き込みを示す実行ログを記憶部１２３に格納する。ログ記録部１２１は、ＲＡＭ１０２からＣＰＵ１０１にプログラムの命令が読み込まれたとき、その命令が何れかのベースレジスタを更新する命令（例えば、レジスタＢＲ１，ＢＲ２，

ＢＲ３の何れかを指定したロード命令）であるか判断する。ベースレジスタを更新する命令の場合、ログ記録部１２１はその命令に対応する実行ログを生成し、ベースレジスタを更新する命令でない場合、ログ記録部１２１は実行ログを生成しない。

【００５７】

命令変換部１２２は、ＲＡＭ１０２に格納されているプログラムが、演算回路１１２が解釈可能な命令セットと異なる命令セットによって記述されているとき、命令の変換を行う。命令変換部１２２は、ＲＡＭ１０２からＣＰＵ１０１にプログラムの命令が読み込まれると、その命令を演算回路１１２が解釈可能な命令に変換して演算回路１１２に渡す。

【００５８】

記憶部１２３は、エラー分析に用いられる管理情報を記憶する。管理情報は、ログ記録部１２１によって生成された実行ログを含む。記憶部１２３に記憶される実行ログには、ＯＳプログラムの中で行われたベースレジスタへの書き込みを示すＯＳプログラムの実行ログと、ユーザプログラムの中で行われたベースレジスタへの書き込みを示すユーザプログラムの実行ログとが含まれる。また、管理情報は、ログ採取の設定を示すシステム情報、プログラムの間の呼び出し関係を示す呼び出し関係情報、情報処理装置１００で実行中のプロセスを示すプロセス情報などを含む。システム情報の少なくとも一部は、予めユーザによって記述される。呼び出し関係情報およびプロセス情報は、ＯＳによって生成されて記憶部１２３に格納される。管理情報の構造については後述する。

【００５９】

割り込み検出部１２４は、プログラムの実行中にエラーが発生すると、割り込み信号を受け付ける。割り込み信号が入力されると、エラー発生直前にＣＰＵ１０１で実行されていたプログラムの処理が中断されて、割り込み検出部１２４の処理が開始される。割り込み検出部１２４は、割り込み信号を受け付けると割り込みの種類を判定し、メモリアクセスエラーによる割り込みである場合はエラー分析部１２５を呼び出す。

【００６０】

エラー分析部１２５は、メモリアクセスエラーが発生したとき、記憶部１２３に記憶された実行ログに基づいてエラーの原因となったプログラムを判定する。前述のように、第２の実施の形態では、エラーが検出されたプログラムではなく、異常なベースアドレスを用意したプログラムをエラーの原因と判定する。ベースアドレスが正常か判定する方法は後述する。エラーの原因となったプログラムがユーザプログラムのとき、エラー分析部１２５はユーザプログラム停止部１２６を呼び出す。エラーの原因となったプログラムがＯＳプログラムのとき、エラー分析部１２５はシステム停止部１２７を呼び出す。

【００６１】

ユーザプログラム停止部１２６は、あるユーザプログラムがエラーの原因であるとエラー分析部１２５において判定されると、そのユーザプログラムの実行を強制的に停止させる（ユーザプログラムをシャットダウンする）。この場合、シャットダウンされたユーザプログラムと並行して実行されているＯＳプログラムや他のユーザプログラムは、原則として、影響を受けずに継続して実行し続けることができる。

【００６２】

システム停止部１２７は、ＯＳプログラムがエラーの原因であるとエラー分析部１２５において判定されると、ＯＳを強制的に停止させる（ＯＳをシャットダウンする）。ＯＳがシャットダウンされると、原則として、情報処理装置１００で実行されている全てのプログラムが停止する。ただし、情報処理装置１００が複数のプロセッサを備え、プロセッサ毎にＯＳを実行している場合、エラーの原因のＯＳプログラムと異なるプロセッサで実行されるプログラムは、影響を受けずに継続して実行し続けることができる場合がある。なお、システム停止部１２７は、ＯＳを再起動するようにしてもよい。

【００６３】

次に、実行ログを含む管理情報のデータ構造の例を説明する。

図７は、管理情報の全体構造の例を示す図である。

記憶部１２３に記憶される管理情報は、システム情報１３１、ジョブ管理情報１３２、

呼び出し関係情報１３３、プロセス情報の集合、ＯＳプログラムの実行ログの集合およびユーザプログラムの実行ログの集合を含む。各プロセス情報は、１つのプロセスとユーザプログラムの組に対応する。各ＯＳプログラムの実行ログは、ＯＳプログラムの中で行われた１回のベースレジスタへの書き込みに対応する。各ユーザプログラムの実行ログは、ユーザプログラムの中で行われた１回のベースレジスタへの書き込みに対応する。

【００６４】

システム情報１３１は、実行ログを採取する条件を示す設定情報を含む。システム情報１３１は、情報処理装置１００が備えるＣＰＵの数分だけ、ＯＳによって生成されて管理される。設定情報は、情報処理装置１００のユーザからの指示に応じて生成される。また、システム情報１３１は、ＯＳプログラムの実行ログのうち先頭の実行ログへのポインタを含む。ジョブ管理情報１３２は、呼び出し関係情報１３３へのポインタと、各プロセス情報へのポインタとを含む。呼び出し関係情報１３３は、プログラムの間の呼び出し関係を示す。呼び出し関係情報１３３は、ＯＳによって生成されて管理される。ジョブ管理情報１３２からポインタを辿ることで呼び出し関係情報１３３にアクセスすることができる。

【００６５】

各プロセス情報は、１つのプロセスの中で１つのユーザプログラムが起動されたときにＯＳによって生成される。各プロセス情報は、プロセスおよびユーザプログラムを示す情報を含む。また、各プロセス情報は、プロセスの中でそのユーザプログラムが実行されたときの実行ログ（プロセスとユーザプログラムの組に対応する実行ログ）のうち、先頭の実行ログへのポインタを含む。ジョブ管理情報１３２からポインタを辿ることで、所望のプロセスおよびプログラムの組に対応するプロセス情報にアクセスすることができる。

【００６６】

複数のＯＳプログラムの実行ログは、書き込みの時系列順に並べられてリスト構造として保存されている。また、同じプロセスとユーザプログラムの組についての複数の実行ログは、書き込みの時系列順に並べられてリスト構造として保存されている。実行ログの間のポインタを辿ることで、所望の条件を満たす実行ログを検索することができる。なお、図７に示したように、プロセス情報の集合には後述するプロセス情報１３４が含まれ、ＯＳプログラムの実行ログの集合には後述する実行ログ１３５が含まれ、ユーザプログラムの実行ログの集合には実行ログ１３６が含まれているものとする。

【００６７】

図８は、システム情報の構造例を示す図である。

システム情報１３１は、ＯＳプログラムの実行ログのうち、先頭の実行ログのアドレスと末尾の実行ログのアドレスを含む。新しいＯＳプログラムの実行ログはリストの末尾に追加される。すなわち、末尾の実行ログが最新の実行ログになる。また、システム情報１３１は、制御フラグの集合とログを採取するユーザプログラムの識別情報を含む。制御フラグの集合には、ＯＳ適用フラグおよびプログラム限定フラグが含まれる。

【００６８】

ＯＳ適用フラグは、ＯＳプログラムを実行ログ採取の対象とするか否かを示すフラグである。例えば、ＯＳ適用フラグ＝１のときＯＳプログラムを実行ログ採取の対象とし、ＯＳ適用フラグ＝０のときＯＳプログラムを実行ログ採取の対象外とする。ＯＳプログラムを実行ログ採取の対象外とした場合、命令エミュレータ１１３は、ベースレジスタへの書き込みを示す命令がＯＳプログラムに含まれていても実行ログを生成しない。

【００６９】

プログラム限定フラグは、実行ログ採取の対象とするユーザプログラムを限定するか否かを示すフラグである。例えば、プログラム限定フラグ＝０のとき全てのユーザプログラムを実行ログ採取の対象とし、プログラム限定フラグ＝１のとき一部のユーザプログラムのみを実行ログ採取の対象とする。ユーザプログラムを限定する場合、実行ログ採取の対象とするユーザプログラムがシステム情報１３１に登録される。その場合、命令エミュレータ１１３は、システム情報１３１に登録されていないユーザプログラムにベースレジス

タへの書き込みを示す命令が含まれていても、実行ログを生成しない。
【００７０】
図９は、呼び出し関係情報の構造例を示す図である。
呼び出し関係情報１３３は、複数のプログラムの間の呼び出し関係を示す。呼び出し関係情報１３３は、プログラム毎に、プログラムの識別情報と、そのプログラムから呼び出される子プログラムを指し示す呼出ポインタと、そのプログラムを呼び出した親プログラムを指し示す復帰ポインタとを含む。呼出ポインタおよび復帰ポインタを順に辿ることで、あるプログラムと直列の呼び出し関係にある他のプログラムを検索することができる。例えば、あるプログラムでエラーが発生したとき、復帰ポインタを辿ることで、エラーが発生したプログラムにデータを渡した可能性のあるプログラムを検索できる。
【００７１】
図１０は、プロセス情報の構造例を示す図である。
プロセス情報１３４は、プロセスの識別情報とプログラムの識別情報を含む。また、プロセス情報１３４は、そのプロセスとプログラムの組に対応する実行ログのうち、先頭の実行ログのアドレスと末尾の実行ログのアドレスを含む。新しいユーザプログラムの実行ログはリストの末尾に追加される。すなわち、末尾の実行ログが最新の実行ログになる。なお、他のプロセス情報もプロセス情報１３４と同様のデータ構造を有する。
【００７２】
図１１は、実行ログの構造例を示す図である。
実行ログ１３５は、実行ログのリストにおける次の実行ログのアドレスと前の実行ログのアドレスを含む。次の実行ログは実行ログ１３５より１つ後に生成されたものであり、前の実行ログは実行ログ１３５より１つ前に生成されたものである。
【００７３】
また、実行ログ１３５は、命令タイプ、レジスタ番号、命令アドレス、メモリアドレス、ロード前のレジスタ値およびロード後のレジスタ値を含む。命令タイプは、ロード命令などの命令の種類である。レジスタ番号は、更新するベースレジスタ（例えば、レジスタＢＲ１，ＢＲ２，ＢＲ３の何れか）の番号である。命令アドレスは、実行された命令をプログラムの中で識別するためのアドレスである。メモリアドレスは、ベースアドレスが格納されていたＲＡＭ１０２のメモリ領域を示すアドレスである。ロード前のレジスタ値は、命令を実行する前にベースレジスタに格納されていたベースアドレスである。ロード後のレジスタ値は、命令を実行することで書き込まれたベースアドレスである。
【００７４】
また、実行ログ１３５は、エラー分析に役立つ他のレジスタ値を含んでもよい。他のレジスタ値としては、例えば、命令が実行されたときのインデックスレジスタの値（インデックス）が挙げられる。なお、ログ記録部１２１は、演算回路１１２によって命令が実行される前に実行ログ１３５を生成してもよいし、命令が実行された後に実行ログ１３５を生成してもよい。ただし、ロード後のレジスタ値は、命令が実行された後に確定する。
【００７５】
上記ではＯＳプログラムの実行ログである実行ログ１３５のデータ構造を説明したが、ユーザプログラムの実行ログである実行ログ１３６も、実行ログ１３５と同様のデータ構造を有する。また、他の実行ログも、実行ログ１３５と同様のデータ構造を有する。
【００７６】
次に、ログ採取およびエラー分析の手順例を説明する。
図１２は、ログ採取の手順例を示すフローチャートである。
（Ｓ１０）ログ記録部１２１は、ＲＡＭ１０２から読み込まれた命令を取得する。
【００７７】
（Ｓ１１）ログ記録部１２１は、記憶部１２３に記憶されたシステム情報１３１を参照して、取得した命令がログ採取の対象となるプログラムの命令であるか判断する。ログ採取の対象となるプログラムの命令である場合、処理をステップＳ１２に進める。それ以外の場合、ログ記録部１２１は実行ログを生成せずに処理を終了する。

【００７８】
具体的には、ログ記録部１２１は、ＯＳプログラムの命令を取得したとき、ＯＳ適用フラグを確認する。例えば、ＯＳ適用フラグ＝１の場合はログ採取の対象と判断し、ＯＳ適用フラグ＝０の場合はログ採取の対象外と判断する。また、ログ記録部１２１は、ユーザプログラムの命令を取得したとき、プログラム限定フラグを確認する。例えば、プログラム限定フラグ＝０の場合、ログ採取の対象と判断する。また、プログラム限定フラグ＝１で、かつ、実行中のユーザプログラムがシステム情報１３１に登録されている場合、ログ採取の対象と判断する。プログラム限定フラグ＝１で、かつ、実行中のユーザプログラムがシステム情報１３１に登録されていない場合、ログ採取の対象外と判断する。
【００７９】
（Ｓ１２）ログ記録部１２１は、取得した命令の命令タイプを確認し、ベースレジスタであるレジスタＢＲ１，ＢＲ２，ＢＲ３の何れかを更新する命令であるか判断する。ベースレジスタを更新する命令としては、例えば、ベースレジスタを指定したロード命令が挙げられる。ベースレジスタを更新する命令である場合、処理をステップＳ１３に進める。それ以外の命令である場合、ログ記録部１２１は実行ログを生成せずに処理を終了する。
【００８０】
（Ｓ１３）ログ記録部１２１は、記憶部１２３に実行ログの記憶領域を確保する。具体的には、ＯＳプログラムの命令が読み込まれたとき、ログ記録部１２１は、ＯＳプログラムの実行ログのリストの末尾に、新たな実行ログのための記憶領域を確保する。ユーザプログラムの命令が読み込まれたとき、ログ記録部１２１は、実行中のプロセスおよびプログラムの組に対応するプロセス情報からユーザプログラムの実行ログのリストにアクセスし、その末尾に新たな実行ログのための記憶領域を確保する。
【００８１】
（Ｓ１４）ログ記録部１２１は、記憶部１２３に確保した記憶領域に対して実行ログを書き込む。命令タイプおよびレジスタ番号は、例えば、読み込まれた命令から特定できる。命令アドレスは、例えば、ＣＰＵ１０１が備えるプログラムカウンタを参照して特定できる。メモリアドレスは、例えば、読み込まれた命令とその命令が参照するレジスタの内容から特定できる。ロード前のレジスタ値は命令実行前のレジスタの内容から特定でき、ロード後のレジスタ値は命令実行後のレジスタの内容から特定できる。また、ログ記録部１２１は、システム情報１３１またはプロセス情報に含まれる「末尾の実行ログのアドレス」と、１つ前の実行ログに含まれる「次の実行ログのアドレス」を更新する。
【００８２】
図１３は、エラー分析の手順例を示すフローチャートである。
（Ｓ２０）割り込み検出部１２４は、割り込み信号に基づいてメモリアクセスエラーを検出すると、エラー分析部１２５を呼び出す。エラー分析部１２５は、エラーが発生したときの命令アドレス（例えば、プログラムカウンタの値）からエラーを生じさせた命令を特定し、ＯＳがもつ情報に基づいてその命令を含むプログラムを特定する。
【００８３】
（Ｓ２１）エラー分析部１２５は、ＯＳプログラムに含まれる命令でエラーが発生したか判断する。ＯＳプログラムに含まれる命令である場合は処理をステップＳ２３に進め、ユーザプログラムに含まれる命令である場合は処理をステップＳ２２に進める。
【００８４】
（Ｓ２２）エラー分析部１２５は、エラーが発生したときに実行していたプロセスおよびプログラムを特定し、記憶部１２３からそのプロセスおよびプログラムの組に対応するプロセス情報を検索する。そして、エラー分析部１２５は、検索したプロセス情報から辿ることができる実行ログのリストを選択する。その後、処理をステップＳ２５に進める。なお、エラー分析部１２５は、以下のステップＳ２７～Ｓ３１を通じて、選択したリストに含まれる実行ログを末尾から先頭に向かって（新しい順に）１つずつ確認していく。
【００８５】
（Ｓ２３）エラー分析部１２５は、記憶部１２３に記憶されたシステム情報１３１のＯ

Ｓ適用フラグを参照して、ＯＳプログラムがログ採取の対象であるか判断する。ＯＳプログラムがログ採取の対象である場合、処理をステップＳ２４に進める。ＯＳプログラムがログ採取の対象外である場合、処理をステップＳ２６に進める。

【００８６】

（Ｓ２４）エラー分析部１２５は、記憶部１２３に記憶されたシステム情報１３１から辿ることができる実行ログのリスト（ＯＳプログラムの実行ログのリスト）を選択する。なお、エラー分析部１２５は、以下のステップＳ２７～Ｓ３１を通じて、選択したリストに含まれる実行ログを末尾から先頭に向かって（新しい順に）１つずつ確認していく。

【００８７】

（Ｓ２５）エラー分析部１２５は、ステップＳ２１で特定したエラーが発生した命令から、参照されたベースレジスタを特定する。ここでは、レジスタＢＲ１，ＢＲ２，ＢＲ３の何れかが特定される。そして、処理をステップＳ２７に進める。

【００８８】

（Ｓ２６）エラー分析部１２５は、ＯＳプログラムがエラー原因である可能性がある一方でＯＳプログラムの実行履歴を分析できないため、ＯＳプログラムをエラーの原因となったプログラムとみなす。そして、処理を後述するステップＳ３６に進める。

【００８９】

（Ｓ２７）エラー分析部１２５は、ステップＳ２２またはＳ２４で選択したリストに含まれる実行ログの探索が終了したか判断する。探索が終了した場合は処理を後述するステップＳ３３に進め、探索が終了していない場合は処理をステップＳ２８に進める。

【００９０】

（Ｓ２８）エラー分析部１２５は、現在着目している実行ログに含まれる「レジスタ番号」が、参照されたベースレジスタのレジスタ番号と一致するか判断する。すなわち、エラー分析部１２５は、現在着目している実行ログが、参照されたベースレジスタへの書き込みを示す実行ログであるか判断する。レジスタ番号が一致した場合、処理をステップＳ２９に進める。レジスタ番号が一致しない場合、「前の実行ログのアドレス」に基づいて１つ前の実行ログが存在するか確認し、処理をステップＳ２７に進める。

【００９１】

（Ｓ２９）エラー分析部１２５は、レジスタ番号が一致した実行ログから、参照されたベースレジスタに書き込まれた値（ベースアドレス）を確認する。

（Ｓ３０）エラー分析部１２５は、ベースアドレスが正常であるか判断する。ベースアドレスが正常であると判断した場合、処理をステップＳ３１に進める。ベースアドレスが正常でないと判断した場合、処理をステップＳ３２に進める。ベースアドレスが正常か否かは、例えば、以下の方法（ａ）または方法（ｂ）によって判断できる。

【００９２】

（ａ）エラー分析部１２５は、ＯＳがもつメモリ管理情報を参照して、書き込まれたベースアドレスが示すメモリ領域が、何れかのプログラムによって使用されているか判断する。そのメモリ領域を使用するプログラムが存在する場合、エラー分析部１２５は、ベースアドレスが正常であると判断する。そのメモリ領域を使用するプログラムが存在しない場合、エラー分析部１２５は、ベースアドレスが異常であると判断する。

【００９３】

（ｂ）エラー分析部１２５は、書き込まれたベースアドレスと、プログラムに割り当てられるメモリ領域の単位であるブロックの先頭アドレスとを比較する。そして、エラー分析部１２５は、以下の条件が満たされる場合にベースアドレスが異常であると判断し、満たされない場合にベースアドレスが正常であると判断する。

【００９４】

条件：書き込まれたベースアドレス＜ブロックの先頭アドレスであり、かつ、書き込まれたベースアドレス＋閾値≧ブロックの先頭アドレスである。ここで、ブロックは、例えば、４ｋバイトや８ｋバイトなどの固定長のメモリ領域である。閾値は、情報処理装置１００のハードウェアやＯＳなどの実行環境に応じて変わり得る。

【００９５】
（Ｓ３１）エラー分析部１２５は、現在着目している実行ログから、ベースレジスタにベースアドレスを書き込んだときに参照された他のベースレジスタを特定する。そして、「参照されたベースレジスタ」を当該他のベースレジスタに置き換えて、処理をステップＳ２７に進める。すなわち、エラー分析部１２５は、現在着目している実行ログより前の実行ログの中から、当該他のベースレジスタへの書き込みを示す実行ログを探索する。これは、読み込まれた値が外見上ベースアドレスとして正常に見えても、上位のベースアドレスが異常であるために意図した値が読み込まれなかった可能性があるためである。
【００９６】
（Ｓ３２）エラー分析部１２５は、現在着目している実行ログに含まれる「メモリアドレス」とＯＳがもつメモリ管理情報とに基づいて、ベースアドレスが格納されていたメモリ領域を確保したプログラムを特定する。特定されるプログラムは、ＯＳプログラムであることもあるし、ユーザプログラムであることもある。そして、エラー分析部１２５は、特定したプログラムをエラーの原因と判定し、処理を後述するステップＳ３６に進める。
【００９７】
図１４は、エラー分析の手順例を示すフローチャート（続き）である。
（Ｓ３３）エラー分析部１２５は、記憶部１２３に記憶された呼び出し関係情報１３３を参照して、同一プロセス内で現在着目しているプログラムを呼び出した他のプログラムが存在するか判断する。呼び出し元のプログラムが存在する場合は処理をステップＳ３４に進め、存在しない場合は処理をステップＳ３５に進める。
【００９８】
（Ｓ３４）エラー分析部１２５は、エラーが発生したときに実行していたプロセスおよびステップＳ３３で特定したプログラムの組に対応するプロセス情報を、記憶部１２３から検索する。そして、エラー分析部１２５は、検索したプロセス情報から辿ることができる実行ログのリストを選択する。その後、処理をステップＳ２７に進める。
【００９９】
（Ｓ３５）エラー分析部１２５は、実行ログを分析しても異常なベースアドレスを用意したプログラムを発見できなかったため、ステップＳ２０で特定したエラーが発生したプログラムを、エラーの原因となったプログラムであると判定する。
【０１００】
（Ｓ３６）エラー分析部１２５は、エラーの原因となった異常なプログラムは、ＯＳプログラムであるか判断する。エラーの原因がＯＳプログラムである場合は処理をステップＳ３８に進め、ユーザプログラムである場合は処理をステップＳ３７に進める。
【０１０１】
（Ｓ３７）ユーザプログラム停止部１２６は、エラーの原因と判定されたユーザプログラムを強制的に停止させる。このとき、ＯＳプログラムは停止させなくてもよいため、他のユーザプログラムやＯＳプログラムの実行は継続され得る。
【０１０２】
（Ｓ３８）システム停止部１２７は、ＯＳをシャットダウンする。なお、システム停止部１２７は、その後にＯＳを再起動するようにしてもよい。
第２の実施の形態によれば、メモリアクセスエラーが発生したとき、ベースレジスタへの書き込みを示す実行ログに基づいて、異常なベースアドレスを用意したプログラムを特定することができる。よって、エラーの原因となったプログラムの判定精度を向上させることができる。また、ＯＳプログラムでエラーが発生してもエラー原因がユーザプログラムにあると判定された場合は、ＯＳをシャットダウンしなくてもよい。よって、情報処理装置１００が突然停止することを抑制でき、情報処理装置１００の安定性が向上する。また、エラーが発生したプログラムとエラーの原因となったプログラムとが直列の呼び出し関係にない場合であっても、エラー原因を特定できる。
【０１０３】
なお、上記の第２の実施の形態ではベースレジスタへの書き込みに限定して実行ログを

採取したが、他の種類のレジスタへの書き込みを示す実行ログを採取してもよい。また、上記の第２の実施の形態ではメモリアクセスエラーの原因を分析したが、レジスタへの書き込みの実行ログに基づいて他の種類のエラーを分析できるようにしてもよい。

【０１０４】

［第３の実施の形態］

第３の実施の形態を説明する。前述の第２の実施の形態との違いを中心に説明し、第２の実施の形態と同様の事項については説明を省略することがある。

【０１０５】

第３の実施の形態に係る情報処理装置は、レジスタの値に応じた命令にジャンプするジャンプレジスタ命令の実行ログを更に採取する。第３の実施の形態の情報処理装置は、図２，３，６に示した第２の実施の形態の情報処理装置１００と同様の構成によって実現できる。以下、図２，３，６と同じ符号を用いて第３の実施の形態を説明する。

【０１０６】

図１５は、ジャンプ先での実行エラーの発生例を示す図である。

ここでは、命令の実行シーケンスが、プログラム＃１の途中からプログラム＃２にジャンプし、プログラム＃２の途中からプログラム＃３にジャンプし、プログラム＃３の途中から更に他のプログラムにジャンプする場合を考える。プログラム＃１，＃２，＃３は、ユーザプログラムでもよいしＯＳプログラムであってもよい。

【０１０７】

プログラム＃１には、ＲＡＭ１０２のメモリ領域＃１からレジスタＲ１にデータを読み込むロード命令が含まれる。メモリ領域＃１には、プログラム＃２に含まれる命令を示す命令アドレス＃１が格納されていることを想定している。また、プログラム＃１には、レジスタＲ１の値が示す命令にジャンプするジャンプレジスタ命令が含まれる。

【０１０８】

プログラム＃２には、ＲＡＭ１０２のメモリ領域＃２からレジスタＲ１にデータを読み込むロード命令が含まれる。メモリ領域＃２には、命令アドレス＃２が格納されていることを想定している。このロード命令が実行されると、命令アドレス＃１が消えて命令アドレス＃２がレジスタＲ１に上書きされる。また、プログラム＃２には、プログラム＃２に含まれる所定の命令にジャンプするジャンプ命令が含まれる。ジャンプレジスタ命令ではジャンプ先が可変であるのに対し、ジャンプ命令ではジャンプ先が固定である。なお、ジャンプ命令およびジャンプレジスタ命令は、分岐命令と呼ばれることがある。

【０１０９】

また、プログラム＃２には、ＲＡＭ１０２のメモリ領域＃３からレジスタＲ２にデータを読み込むロード命令が含まれる。メモリ領域＃３には、プログラム＃３に含まれる命令を示す命令アドレス＃３が格納されていることを想定している。また、プログラム＃２には、レジスタＲ２の値が示す命令にジャンプするジャンプレジスタ命令が含まれる。

【０１１０】

プログラム＃３には、レジスタＲ１の値が示す命令にジャンプするジャンプレジスタ命令が含まれる。ここで、レジスタＲ１に書き込まれた命令アドレス＃２が、存在しない命令を指し示す異常な命令アドレスであったとする。すると、ジャンプ後にＣＰＵ１０１が、ＲＡＭ１０２上のプログラムが格納されていない非プログラム領域から命令を読み込もうとする可能性がある。この場合、ＣＰＵ１０１がＲＡＭ１０２から実行可能な命令を取得することができず、プログラム実行エラーが発生し得る。

【０１１１】

非プログラム領域でプログラム実行エラーが発生したとき、情報処理装置１００は、非プログラム領域にジャンプしてしまった経緯の分析を支援できるようにすることが好ましい。そこで、第３の実施の形態では、情報処理装置１００は、プログラム＃１，＃２，＃３が実行されている間にジャンプレジスタ命令の実行ログを採取する。

【０１１２】

ジャンプレジスタ命令の実行ログは、ジャンプによる複数のプログラムの間の関連を分

析するために用いることができる。例えば、情報処理装置１００は、プログラム実行エラーが発生したとき、ジャンプレジスタ命令の実行ログに基づいて、当該プログラム実行エラーが発生するまでに実行されたプログラムとその実行の順序を特定できる。ジャンプによるプログラム間の関連の分析は、第２の実施の形態のエラー分析と併せて行うことができる。その関連の分析結果は、第２の実施の形態においてエラーの原因となったプログラムを判定するときに呼び出し関係情報として参照されてもよい。

【０１１３】

また、ジャンプレジスタ命令の実行ログは、プログラム実行エラーの原因を作ったプログラムを判定するために用いることもできる。例えば、情報処理装置１００は、第２の実施の形態のログ採取と同様の方法で、レジスタＲ１，Ｒ２を含む汎用レジスタへの書き込みを示す実行ログを採取する。そして、情報処理装置１００は、プログラム実行エラーが発生したとき、ジャンプレジスタ命令の実行ログに基づいてジャンプ元のプログラムを辿り、異常なジャンプ先の命令アドレスを用意したプログラムを判定する。

【０１１４】

例えば、図１５の場合では、ジャンプレジスタ命令の実行ログに基づいて、プログラム＃３に含まれるジャンプレジスタ命令から非プログラム領域にジャンプしたことが特定される。このジャンプレジスタ命令は、レジスタＲ１を参照している。また、ジャンプレジスタ命令の実行ログに基づいて、プログラム＃２からプログラム＃３にジャンプしたことが特定される。そして、レジスタ書き込みの実行ログに基づいて、プログラム＃２においてメモリ領域＃２からレジスタＲ１にデータが読み込まれたことが特定される。そこで、情報処理装置１００は、例えば、メモリ領域＃２を使用するプログラムを、プログラム実行エラーの原因となったプログラムであると判定する。情報処理装置１００は、第２の実施の形態と同様に、判定したプログラムを強制的に停止させてもよい。

【０１１５】

図１６は、他の実行ログの構造例を示す図である。

実行ログ１３７は、ＣＰＵ１０１がユーザプログラムやＯＳプログラムなどのプログラムを実行している間に生成されて記憶部１２３に記憶される。実行ログ１３７は、実行ログのリストにおける次の実行ログのアドレスと前の実行ログのアドレスを含む。

【０１１６】

また、実行ログ１３７は、命令タイプ、命令アドレスおよびジャンプ先アドレスを含む。命令タイプは、命令の種類がジャンプレジスタ命令であることを示す。命令アドレスは、そのジャンプレジスタ命令をプログラムの中で識別するためのアドレスである。ジャンプ先アドレスは、ジャンプレジスタ命令が参照したレジスタの値であり、ジャンプ先の命令を示す命令アドレスである。また、実行ログ１３７は、エラー分析に役立つ他のレジスタ値を含んでもよい。他のレジスタ値としては、例えば、ジャンプレジスタ命令が実行されたときのインデックスレジスタの値（インデックス）が挙げられる。

【０１１７】

図１７は、他のログ採取の手順例を示すフローチャートである。

（Ｓ４０）ログ記録部１２１は、ＲＡＭ１０２から読み込まれた命令を取得する。

（Ｓ４１）ログ記録部１２１は、記憶部１２３に記憶されたシステム情報１３１を参照して、取得した命令がログ採取の対象となるプログラムの命令であるか判断する。ログ採取の対象となるプログラムの命令である場合、処理をステップＳ４２に進める。それ以外の場合、ログ記録部１２１は実行ログを生成せずに処理を終了する。

【０１１８】

（Ｓ４２）ログ記録部１２１は、取得した命令の命令タイプがジャンプレジスタ命令であるか判断する。ジャンプレジスタ命令である場合、処理をステップＳ４３に進める。それ以外の命令である場合、ログ記録部１２１は実行ログを生成せずに処理を終了する。

【０１１９】

（Ｓ４３）ログ記録部１２１は、記憶部１２３に実行ログの記憶領域を確保する。

（Ｓ４４）ログ記録部１２１は、記憶部１２３に確保した記憶領域に対して実行ログを

書き込む。命令タイプは、読み込まれた命令から特定できる。命令アドレスは、例えば、ＣＰＵ１０１が備えるプログラムカウンタを参照して特定できる。ジャンプ先アドレスは、ジャンプレジスタ命令が指定するレジスタの内容から特定できる。

【０１２０】

第３の実施の形態によれば、非プログラム領域でエラーが発生したときであっても、ジャンプレジスタ命令の実行ログに基づいて、どのプログラムから非プログラム領域にジャンプしたかを特定することができる。また、エラーが発生するまでの経緯、すなわち、どのプログラムがどの順序で実行されたかを特定することが容易となる。また、ジャンプレジスタ命令が参照し得るレジスタへの書き込みの実行ログを併せて採取することで、非プログラム領域へのジャンプの原因を作ったプログラムを判定することができる。

【０１２１】

なお、前述のように、第１の実施の形態の情報処理は、情報処理装置１０にプログラムを実行させることで実現することができる。また、第２の実施の形態の情報処理は、情報処理装置１００にプログラムを実行させることで実現することができる。

【０１２２】

プログラムは、コンピュータ読み取り可能な記録媒体（例えば、記録媒体２３）に記録しておくことができる。記録媒体としては、例えば、磁気ディスク、光ディスク、光磁気ディスク、半導体メモリなどを使用できる。磁気ディスクには、ＦＤおよびＨＤＤが含まれる。光ディスクには、ＣＤ、ＣＤ－Ｒ（Recordable）／ＲＷ（Rewritable）、ＤＶＤおよびＤＶＤ－Ｒ／ＲＷが含まれる。プログラムは、可搬型の記録媒体に記録されて配布されることがある。その場合、可搬型の記録媒体からＨＤＤなどの他の記録媒体（例えば、ＨＤＤ１０３）にプログラムを複製して（インストールして）実行してもよい。

【０１２３】

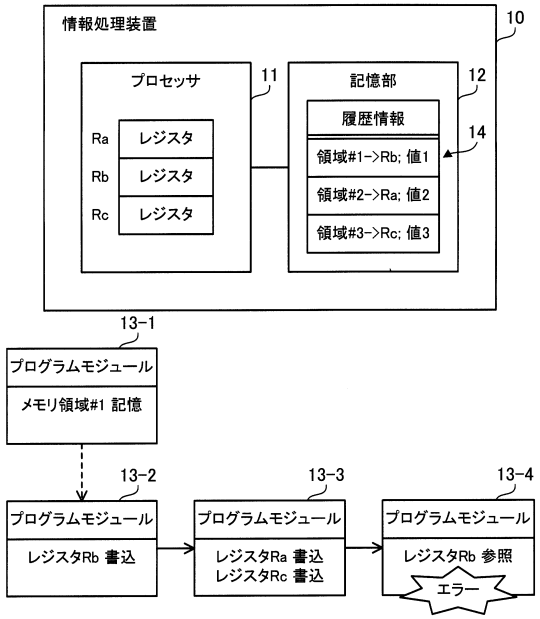
上記については単に本発明の原理を示すものである。更に、多数の変形や変更が当業者にとって可能であり、本発明は上記に示し、説明した正確な構成および応用例に限定されるものではなく、対応する全ての変形例および均等物は、添付の請求項およびその均等物による本発明の範囲とみなされる。

【符号の説明】

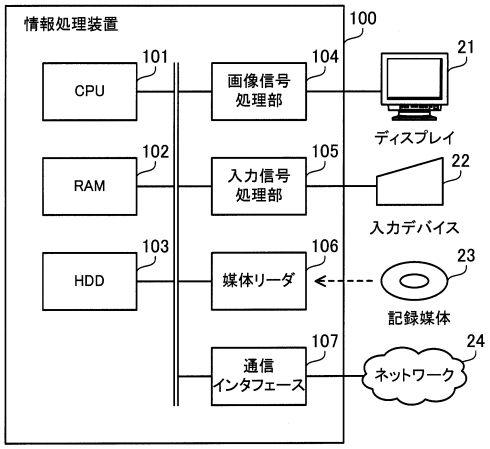
【０１２４】

１０　情報処理装置
１１　プロセッサ
１２　記憶部
１３－１～１３－４　プログラムモジュール
１４　履歴情報
Ｒａ，Ｒｂ，Ｒｃ　レジスタ

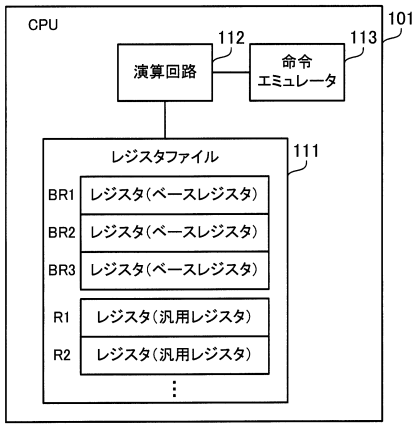
【図１】


情報処理装置
10
プロセッサ
11
Ra
レジスタ
Rb
レジスタ
Rc
レジスタ
記憶部
12
履歴情報
14
領域#1->Rb; 値1
領域#2->Ra; 値2
領域#3->Rc; 値3
13-1
プログラムモジュール
メモリ領域#1 記憶
13-2
プログラムモジュール
レジスタRb 書込
13-3
プログラムモジュール
レジスタRa 書込
レジスタRc 書込
13-4
プログラムモジュール
レジスタRb 参照
エラー

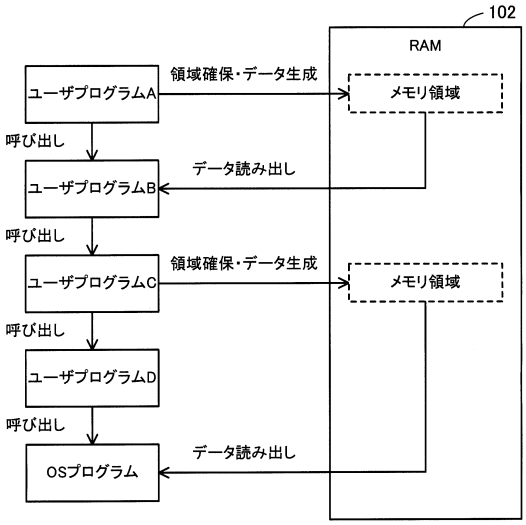
【図２】


情報処理装置
100
CPU
101
RAM
102
HDD
103
画像信号処理部
104
入力信号処理部
105
媒体リーダ
106
通信インタフェース
107
21
ディスプレイ
22
入力デバイス
23
記録媒体
24
ネットワーク

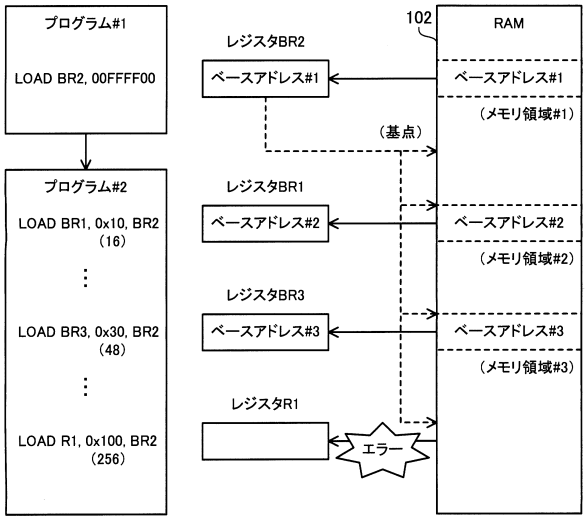
【図３】


CPU
101
演算回路
112
命令エミュレータ
113
レジスタファイル
111
BR1
レジスタ(ベースレジスタ)
BR2
レジスタ(ベースレジスタ)
BR3
レジスタ(ベースレジスタ)
R1
レジスタ(汎用レジスタ)
R2
レジスタ(汎用レジスタ)

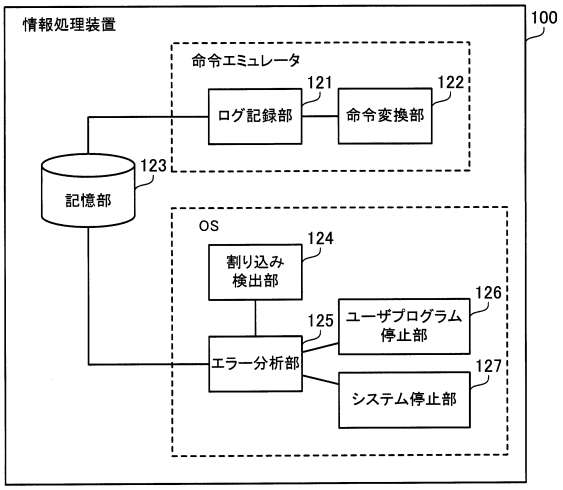
【図４】


102
RAM
ユーザプログラムA
領域確保・データ生成
メモリ領域
呼び出し
ユーザプログラムB
データ読み出し
呼び出し
ユーザプログラムC
領域確保・データ生成
メモリ領域
呼び出し
ユーザプログラムD
呼び出し
OSプログラム
データ読み出し

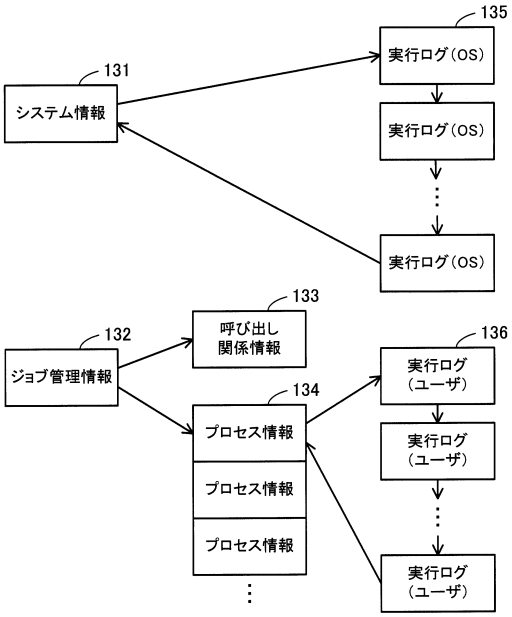
【図５】


プログラム#1
LOAD BR2, 00FFFF00
プログラム#2
LOAD BR1, 0x10, BR2
(16)
LOAD BR3, 0x30, BR2
(48)
LOAD R1, 0x100, BR2
(256)
102
RAM
レジスタBR2
ベースアドレス#1
ベースアドレス#1
(メモリ領域#1)
(基点)
レジスタBR1
ベースアドレス#2
ベースアドレス#2
(メモリ領域#2)
レジスタBR3
ベースアドレス#3
ベースアドレス#3
(メモリ領域#3)
レジスタR1
エラー

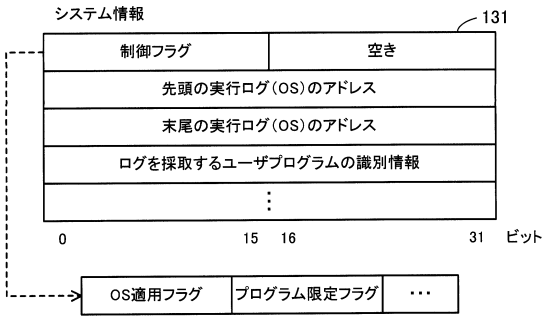
【図６】


情報処理装置
100
命令エミュレータ
121
ログ記録部
122
命令変換部
123
記憶部
OS
124
割り込み
検出部
125
エラー分析部
126
ユーザプログラム
停止部
127
システム停止部

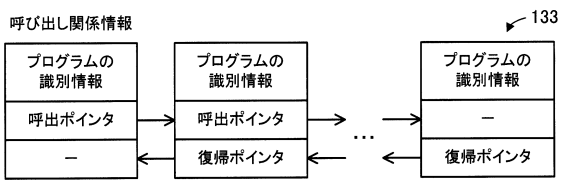
【図７】


135
実行ログ(OS)
131
システム情報
実行ログ(OS)
実行ログ(OS)
133
呼び出し
関係情報
132
ジョブ管理情報
136
実行ログ
(ユーザ)
134
プロセス情報
実行ログ
(ユーザ)
プロセス情報
プロセス情報
実行ログ
(ユーザ)

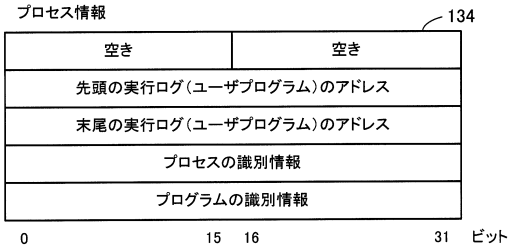
【図８】


システム情報
131
制御フラグ
空き
先頭の実行ログ(OS)のアドレス
末尾の実行ログ(OS)のアドレス
ログを採取するユーザプログラムの識別情報
0
15
16
31
ビット
OS適用フラグ
プログラム限定フラグ
…

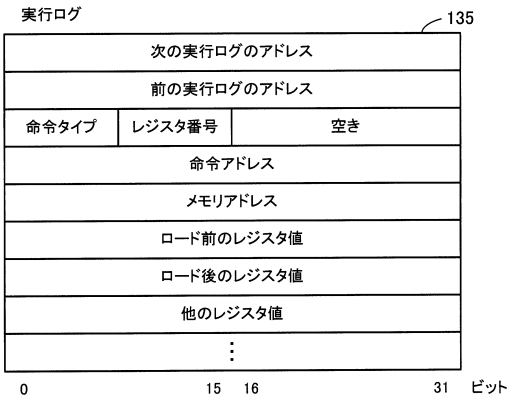
【図９】


呼び出し関係情報
133
プログラムの
識別情報
呼出ポインタ
−
プログラムの
識別情報
呼出ポインタ
復帰ポインタ
…
プログラムの
識別情報
−
復帰ポインタ

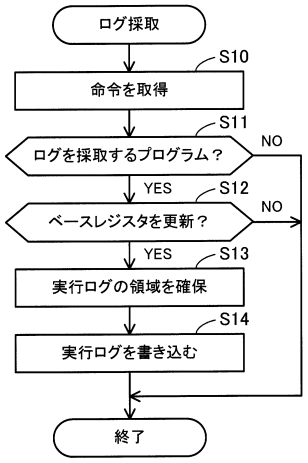
【図１０】



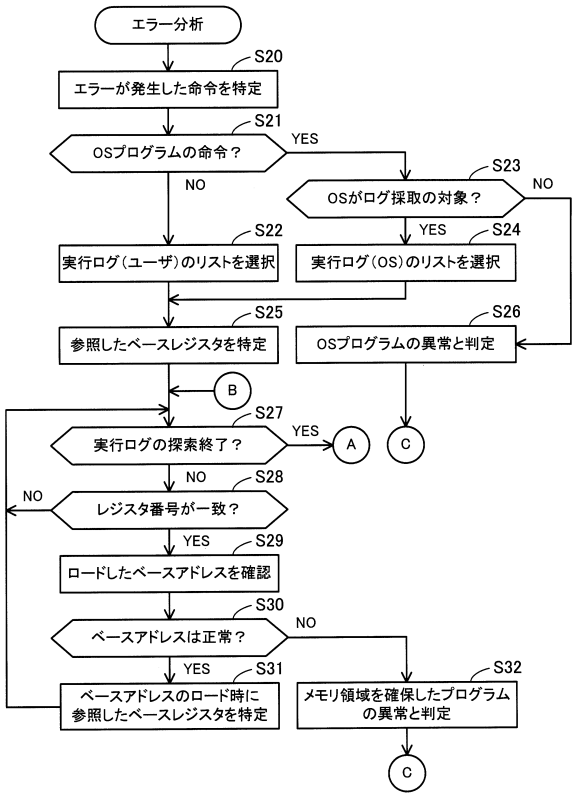
【図１１】



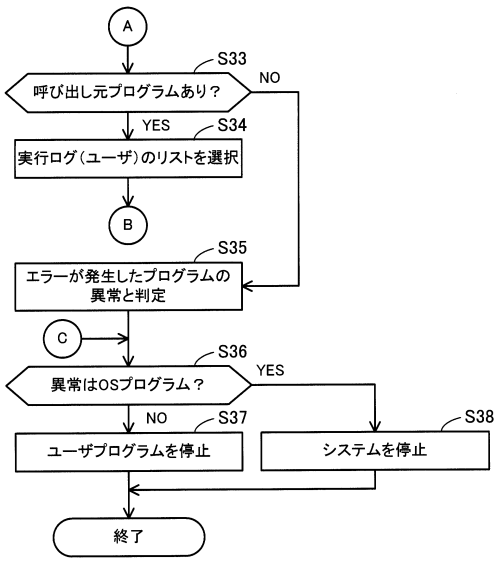
【図１２】



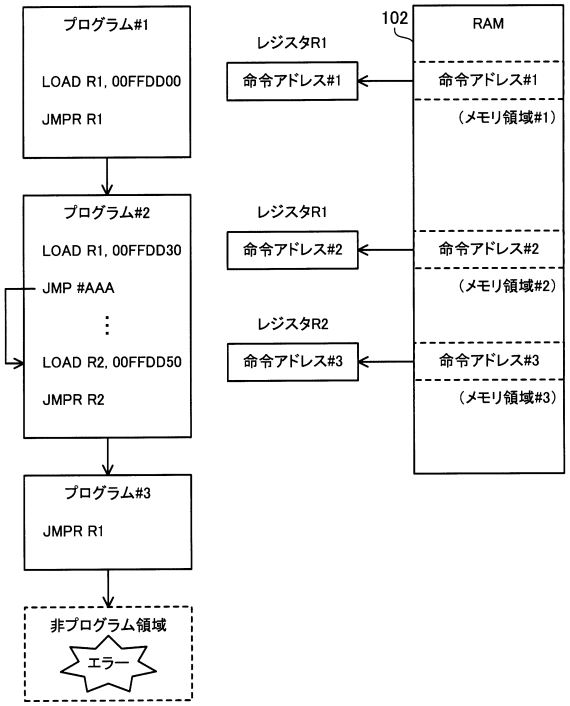
【図１３】



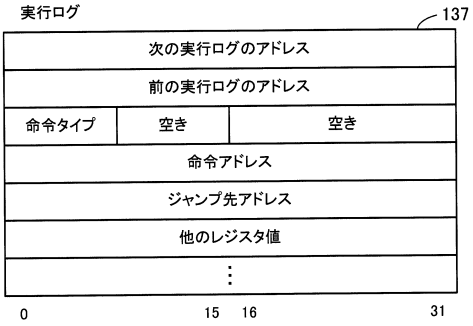
【図１４】



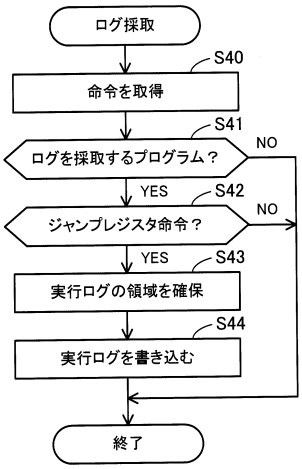
【図１５】


プログラム#1
LOAD R1, 00FFDD00
JMPR R1
プログラム#2
LOAD R1, 00FFDD30
JMP #AAA
LOAD R2, 00FFDD50
JMPR R2
プログラム#3
JMPR R1
非プログラム領域
エラー
102
RAM
レジスタR1
命令アドレス#1
命令アドレス#1
（メモリ領域#1）
レジスタR1
命令アドレス#2
命令アドレス#2
（メモリ領域#2）
レジスタR2
命令アドレス#3
命令アドレス#3
（メモリ領域#3）

【図１６】


実行ログ
137
次の実行ログのアドレス
前の実行ログのアドレス
命令タイプ
空き
空き
命令アドレス
ジャンプ先アドレス
他のレジスタ値
0
15
16
31

【図１７】


ログ採取
S40
命令を取得
S41
ログを採取するプログラム？
NO
YES
S42
ジャンプレジスタ命令？
NO
YES
S43
実行ログの領域を確保
S44
実行ログを書き込む
終了

フロントページの続き

(56)参考文献　特開平１０－３２０２３２（ＪＰ，Ａ）
特開平０６－３４８５４０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１１／０７