(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6069719号
(P6069719)
(45)発行日　平成29年2月1日(2017.2.1)　(24)登録日　平成29年1月13日(2017.1.13)

(51)Int.Cl.　F I
G06F 11/36 (2006.01)　G06F 11/36 112
G06F 11/36 132
G06F 11/36 136

請求項の数 9　(全 41 頁)

(21)出願番号　特願2015-529270(P2015-529270)
(86)(22)出願日　平成25年7月31日(2013.7.31)
(86)国際出願番号　PCT/JP2013/070734
(87)国際公開番号　WO2015/015591
(87)国際公開日　平成27年2月5日(2015.2.5)
審査請求日　平成27年11月11日(2015.11.11)

(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】ソフトウェアデバッグ方法、情報処理装置およびプログラム

(57)【特許請求の範囲】
【請求項１】
複数のプロセッサを含むコンピュータが実行するソフトウェアデバッグ方法であって、
第１および第２のプロセッサそれぞれが、命令の実行を制御する制御ソフトウェアを用いて、デバッグするソフトウェアに含まれる命令を実行し、
前記第１のプロセッサが、メモリに対するデータ書込またはデータ参照を示す第１の命令を実行するとき、前記第１の命令に応じた第１の履歴情報を生成し、
前記第２のプロセッサが、前記メモリに対するデータ書込を示す第２の命令を実行するとき、前記第２の命令に応じた第２の履歴情報を生成し、
前記第１のプロセッサが、前記第２の履歴情報を取得し、前記第１および第２の履歴情報に基づいて、前記メモリ上の領域に対する排他制御の不備を判定する、
ソフトウェアデバッグ方法。
【請求項２】
前記第１のプロセッサが、前記デバッグするソフトウェアの実行時間を、ロック獲得またはロック解放を行う第３の命令の実行タイミングに応じて複数の区間に分割し、
前記排他制御の不備の判定は、区間毎に当該区間で生成された前記第１の履歴情報と前記第２の履歴情報とを比較することによって行う、
請求項１記載のソフトウェアデバッグ方法。
【請求項３】
前記第１のプロセッサが、前記デバッグするソフトウェアの実行時間を、ソフトウェア

モジュールの切替または動作モードの切替を行う第３の命令の実行タイミングに応じて複数の区間に分割し、
前記排他制御の不備の判定は、区間毎に当該区間で生成された前記第１の履歴情報と前記第２の履歴情報とを比較することによって行う、
請求項１記載のソフトウェアデバッグ方法。
【請求項４】
前記第１および第２の履歴情報は、前記メモリのアドレスを示す情報を含み、
前記第１のプロセッサが、ロック獲得またはロック解放を行う第３の命令を実行するとき、更新されるロック制御情報のアドレスを含む第３の履歴情報を生成し、
前記排他制御の不備の判定は、前記第１および第２の履歴情報が示すアドレスから前記第３の履歴情報が示すアドレスを除外することによって行う、
請求項１記載のソフトウェアデバッグ方法。
【請求項５】
前記第１の命令は、前記メモリ上の複数のブロックに跨がるデータ参照を行い、
前記排他制御の不備の判定は、前記第１の命令が読み込まれてから前記第１の命令の実行が完了するまでの間に生成された第２の履歴情報を用いることによって行う、
請求項１記載のソフトウェアデバッグ方法。
【請求項６】
前記制御ソフトウェアは、前記デバッグするソフトウェアの命令セットを前記第１および第２のプロセッサに応じた他の命令セットに変換するエミュレータである、
請求項１記載のソフトウェアデバッグ方法。
【請求項７】
前記第１の履歴情報は、前記制御ソフトウェアを用いて、前記メモリ上の前記第１のプロセッサに対応する第１の領域に保存され、
前記第２の履歴情報は、前記制御ソフトウェアを用いて、前記第１の領域および前記メモリ上の前記第２のプロセッサに対応する第２の領域の少なくとも一方に保存される、
請求項１記載のソフトウェアデバッグ方法。
【請求項８】
命令の実行を制御する制御ソフトウェアを用いて、デバッグするソフトウェアに含まれる命令を実行する第１および第２のプロセッサと、
前記第１および第２のプロセッサからアクセスされるメモリと、を有し、
前記第１のプロセッサは、前記メモリに対するデータ書込またはデータ参照を示す第１の命令を実行するとき、前記第１の命令に応じた第１の履歴情報を生成し、
前記第２のプロセッサは、前記メモリに対するデータ書込を示す第２の命令を実行するとき、前記第２の命令に応じた第２の履歴情報を生成し、
前記第１のプロセッサは、前記第２の履歴情報を取得し、前記第１および第２の履歴情報に基づいて、前記メモリ上の領域に対する排他制御の不備を判定する、
情報処理装置。
【請求項９】
複数のプロセッサを含むコンピュータに、
第１および第２のプロセッサそれぞれにおいて、デバッグするソフトウェアに含まれる命令が実行されるように制御し、
前記第１のプロセッサを用いて、メモリに対するデータ書込またはデータ参照を示す第１の命令が実行されるとき、前記第１の命令に応じた第１の履歴情報を生成し、
前記第２のプロセッサを用いて、前記メモリに対するデータ書込を示す第２の命令が実行されるとき、前記第２の命令に応じた第２の履歴情報を生成し、
前記第１のプロセッサを用いて、前記第２の履歴情報を取得し、前記第１および第２の履歴情報に基づいて、前記メモリ上の領域に対する排他制御の不備を判定する、
処理を実行させるプログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明はソフトウェアデバッグ方法、情報処理装置およびプログラムに関する。
【背景技術】
【０００２】
コンピュータで実行されるソフトウェアが途中で異常停止・異常終了するなど、ソフトウェアの動作に仕様と異なる不具合（バグ）が含まれていることがあり、その不具合を検出し修正するためにデバッグ作業が行われる。デバッグ作業では、不具合の発生状況を示す情報（例えば、エラーログやメモリダンプなど）を収集し、収集した情報を分析してその不具合の原因を特定し、ソフトウェアの修正（例えば、ソースコードの書き換え）を行う。
【０００３】
ここで、複数のプロセッサ（プロセッサコアと呼ばれるものを含む）でソフトウェアを並列に動作させる場合、１つのプロセッサのみでソフトウェアを動作させる場合とは異なる環境条件によって不具合が発生し得る。そのような不具合の原因の一例として、メモリ上の領域に対する排他制御の不備が挙げられる。メモリ上の領域を使用するとき適切にロックが獲得されないと、２つ以上のプロセッサが同時期に同じメモリアドレスの領域にアクセスし、あるプロセッサの使用するデータが他のプロセッサによって意図せず書き換えられてしまうことがある。
【０００４】
例えば、あるプロセッサが、ある変数に一時的に値を代入してからその値を読み出すとする。変数への値の代入と変数からの値の読み出しの間に、排他制御が適切に行われずに他のプロセッサによって当該変数の値が書き換えられてしまうと、当該プロセッサの処理で異常が発生することがある。このような排他制御の不備に基づく不具合は、各プロセッサにおける命令の実行タイミングにも依存するため再現率が低くなることがあり、デバッグ作業においてその原因を特定することが容易でないことが多い。
【０００５】
そこで、例えば、ソフトウェアとしてのシミュレータ上に、仮想的なプロセッサや仮想的なメモリなどのハードウェアモデルを構築し、シミュレータ上で並列プログラムを実行するシミュレーション方法が提案されている。このシミュレーション方法では、シミュレータ上で並列プログラムを実行する間、仮想的なメモリモニタが、各プロセッサからメモリへのアクセスについての情報を収集する。そして、メモリモニタが、収集した情報に基づいて、複数のプロセッサが重複してアクセスしたメモリ領域を特定する。
【０００６】
また、例えば、プロセッサおよびメモリが接続されたバスに、ハードウェアとしてのメモリ監視回路を設けたデータ処理装置が提案されている。このメモリ監視回路は、ユーザから指定された所定のアドレス範囲の領域について、メモリへのアクセスを監視する。メモリ監視回路は、データ書込用のカウンタとデータ読込用のカウンタとを有し、所定のアドレス範囲の領域に対するデータ書込およびデータ読込の回数をそれぞれカウントする。カウンタの値は、メモリ監視回路から分析ロジックに出力される。
【先行技術文献】
【特許文献】
【０００７】
【特許文献１】特開２００９－３２１９７号公報
【特許文献２】特開２０１０－２０７６７号公報
【発明の概要】
【発明が解決しようとする課題】
【０００８】
しかし、メモリアクセスの情報を収集して排他制御の不備を判定するために、上記のよ

うな特殊なシミュレーション環境または特殊なハードウェアを使用することは、デバッグ作業の負担が大きくなることがあるという問題がある。デバッグ作業時の環境によっては、デバッグ対象のソフトウェアが動作する本番システムまたは本番環境と同等のハードウェア構成のシステムのもとで、デバッグ作業を実施したいことがある。

【０００９】

１つの側面では、本発明は、メモリに対する排他制御の不備の判定を容易にするソフトウェアデバッグ方法、情報処理装置および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、複数のプロセッサを含むコンピュータが実行するソフトウェアデバッグ方法が提供される。第１および第２のプロセッサそれぞれが、命令の実行を制御する制御ソフトウェアを用いて、デバッグするソフトウェアに含まれる命令を実行する。第１のプロセッサが、メモリに対するデータ書込またはデータ参照を示す第１の命令を実行するとき、第１の命令に応じた第１の履歴情報を生成する。第２のプロセッサが、メモリに対するデータ書込を示す第２の命令を実行するとき、第２の命令に応じた第２の履歴情報を生成する。第１のプロセッサが、第２の履歴情報を取得し、第１および第２の履歴情報に基づいて、メモリ上の領域に対する排他制御の不備を判定する。

【００１１】

また、１つの態様では、命令の実行を制御する制御ソフトウェアを用いて、デバッグするソフトウェアに含まれる命令を実行する第１および第２のプロセッサと、第１および第２のプロセッサからアクセスされるメモリと、を有する情報処理装置が提供される。第１のプロセッサは、メモリに対するデータ書込またはデータ参照を示す第１の命令を実行するとき、第１の命令に応じた第１の履歴情報を生成する。第２のプロセッサは、メモリに対するデータ書込を示す第２の命令を実行するとき、第２の命令に応じた第２の履歴情報を生成する。第１のプロセッサは、第２の履歴情報を取得し、第１および第２の履歴情報に基づいて、メモリ上の領域に対する排他制御の不備を判定する。

【００１２】

また、１つの態様では、複数のプロセッサを含むコンピュータに、次の処理を実行させるプログラムが提供される。第１および第２のプロセッサそれぞれにおいて、デバッグするソフトウェアに含まれる命令が実行されるように制御する。第１のプロセッサを用いて、メモリに対するデータ書込またはデータ参照を示す第１の命令が実行されるとき、第１の命令に応じた第１の履歴情報を生成する。第２のプロセッサを用いて、メモリに対するデータ書込を示す第２の命令が実行されるとき、第２の命令に応じた第２の履歴情報を生成する。第１のプロセッサを用いて、第２の履歴情報を取得し、第１および第２の履歴情報に基づいて、メモリ上の領域に対する排他制御の不備を判定する。

【発明の効果】

【００１３】

１つの側面では、メモリに対する排他制御の不備の判定が容易になる。

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】ソフトウェアの階層例を示す図である。

【図４】データ書込時の排他制御不備の例を示す図である。

【図５】データ書込時の排他制御不備の検出例を示すシーケンス図である。

【図６】排他制御不備の有無の判定例を示す図である。

【図７】情報処理装置の機能例を示すブロック図である。

【図８】第２の実施の形態の制御情報の全体構造例を示す図である。

【図９】第２の実施の形態の管理情報のデータ構造例を示す図である。
【図１０】第１のポインタテーブルのデータ構造例を示す図である。
【図１１】収集領域のデータ構造例を示す図である。
【図１２】収集領域ヘッダのデータ構造例を示す図である。
【図１３】集計領域のデータ構造例を示す図である。
【図１４】書込ビットマップのデータ構造例を示す図である。
【図１５】追記領域のデータ構造例を示す図である。
【図１６】第２のポインタテーブルのデータ構造例を示す図である。
【図１７】ログ管理情報のデータ構造例を示す図である。
【図１８】第２の実施の形態のログ情報のデータ構造例を示す図である。
【図１９】起動処理および停止処理の手順例を示すフローチャートである。
【図２０】データ書込時のデバッグ支援の手順例を示すフローチャートである。
【図２１】アドレス情報収集の手順例を示すフローチャートである。
【図２２】ロック獲得記録の手順例を示すフローチャートである。
【図２３】ロック解放判定の手順例を示すフローチャートである。
【図２４】排他制御不備検出の手順例を示すフローチャートである。
【図２５】排他制御不備検出の手順例を示すフローチャート（続き）である。
【図２６】書込衝突判定の手順例を示すフローチャートである。
【図２７】データ参照時の排他制御不備の例を示す図である。
【図２８】データ参照時の排他制御不備の検出例を示すシーケンス図である。
【図２９】第３の実施の形態の制御情報の全体構造例を示す図である。
【図３０】第３の実施の形態の管理情報のデータ構造例を示す図である。
【図３１】第３のポインタテーブルのデータ構造例を示す図である。
【図３２】収集依頼情報のデータ構造例を示す図である。
【図３３】比較命令情報のデータ構造例を示す図である。
【図３４】第４のポインタテーブルのデータ構造例を示す図である。
【図３５】衝突情報のデータ構造例を示す図である。
【図３６】第３の実施の形態のログ情報のデータ構造例を示す図である。
【図３７】データ参照時のデバッグ支援の手順例を示すフローチャートである。
【図３８】衝突情報収集の手順例を示すフローチャートである。
【図３９】読み書き衝突判定の手順例を示すフローチャートである。
【図４０】データ参照時の排他制御不備検出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を示す図である。
【００１６】
第１の実施の形態の情報処理装置１０は、デバッグ対象のソフトウェアとして、複数のプロセッサを使用して並列処理を行うソフトウェアを実行できる。このソフトウェアは、ＯＳ（Operating System）やデバイスドライバなどのシステムプログラムを含んでもよいし、システムプログラム上で動作するユーザプログラムを含んでもよい。
【００１７】
情報処理装置１０は、プロセッサ１１，１２およびメモリ１３を有する。プロセッサ１１，１２は、メモリ１３を用いて、算術演算命令・論理演算命令・比較命令・分岐命令・データ転送命令などの各種の命令を実行する。プロセッサ１１，１２それぞれは、ＣＰＵ（Central Processing Unit）などのプロセッサパッケージでもよいし、プロセッサパッケージ内のプロセッサコアであってもよい。メモリ１３は、プロセッサ１１，１２からアクセスされる共有メモリであり、例えば、ＲＡＭ（Random Access Memory）である。
【００１８】

ここで、プロセッサ１１，１２それぞれは、命令の実行を制御する制御ソフトウェア１４を用いて、デバッグ対象のソフトウェアに含まれる命令を実行する。制御ソフトウェア１４は、例えば、エミュレータと呼ばれるものである。エミュレータは、デバッグ対象のソフトウェアが使用する命令セットとプロセッサ１１，１２が解釈可能な命令セットとが異なる場合に、前者の命令セットを後者の命令セットに変換する。エミュレータは、ハードウェアとソフトウェアの間の命令セットの違いを吸収していると言うこともできる。

【００１９】

第１の実施の形態では、制御ソフトウェア１４には、デバッグ作業を支援するため、メモリ１３上の領域に対する排他制御の不備を判定する機能が実装される。メモリ１３上のある領域を使用するときに適切にロックが獲得されないと、プロセッサ１１，１２が同時期に同じメモリアドレスの領域にアクセスし、プロセッサ１１で使用されているデータがプロセッサ１２によって意図せず破壊されてしまうことがある。排他制御の不備によって不具合が発生する場合として、例えば、プロセッサ１１とプロセッサ１２が同じ領域に対して同時期にデータを書き込む場合が挙げられる。また、不具合が発生する他の場合として、例えば、プロセッサ１１がメモリ１３上の複数のブロックに跨がるデータを参照している間に、プロセッサ１２がそのうちの１つのブロックのデータを書き換えてしまう場合が挙げられる。

【００２０】

そこで、プロセッサ１１，１２は、デバッグ対象のソフトウェアを実行する間、制御ソフトウェア１４の制御に従って次のような処理を行う。プロセッサ１１は、デバッグ対象のソフトウェアに含まれる命令のうち、メモリ１３に対するデータ書込またはデータ参照を伴う命令１５を実行するとき、命令１５に応じた履歴情報１７を生成する。データ書込を伴う命令の一例としては、ストア命令が挙げられ、データ参照を伴う命令の一例としては、メモリ１３上のデータを利用する比較命令が挙げられる。履歴情報１７は、アクセスされるメモリ１３の領域を示すアドレス情報を含んでもよい。また、履歴情報１７は、命令１５が属するプログラムモジュールの識別情報を含んでもよい。生成された履歴情報１７は、例えば、メモリ１３上に予め設けられたプロセッサ１１用の領域に格納される。

【００２１】

また、プロセッサ１２は、デバッグ対象のソフトウェアに含まれる命令のうち、メモリ１３に対するデータ書込を伴う命令１６を実行するとき、命令１６に応じた履歴情報１８を生成する。履歴情報１８は、アクセスされるメモリ１３の領域を示すアドレス情報を含んでもよい。また、履歴情報１８は、命令１６が属するプログラムモジュールの識別情報を含んでもよい。生成された履歴情報１８は、例えば、メモリ１３上に予め設けられたプロセッサ１１用の領域およびプロセッサ１２用の領域の少なくとも一方に格納される。

【００２２】

そして、プロセッサ１１は、プロセッサ１２が生成した履歴情報１８を収集し、生成した履歴情報１７と収集した履歴情報１８とに基づいて、デバッグ対象のソフトウェアに排他制御の不備があるか判定する。履歴情報１７，１８は、例えば、メモリ１３上に予め設けられたプロセッサ１１用の領域に集められる。プロセッサ１１は、例えば、履歴情報１７が示すアドレスと履歴情報１８が示すアドレスとを比較し、履歴情報１７，１８が同一または近いアドレスを指し示しているとき、排他制御の不備があると判定する。

【００２３】

排他制御の不備があると判定したとき、プロセッサ１１は、判定結果を示すログ情報を生成してもよい。このログ情報は、命令１５が属するプログラムモジュールの識別情報や命令１６が属するプログラムモジュールの識別情報含んでもよい。これにより、デバッグ作業において、不具合の原因となるプログラムモジュール中の位置を特定することが容易となる。ログ情報は、例えば、制御ソフトウェア１４に従ってメモリ１３に格納される。ログ情報は、メモリ１３とは異なる不揮発性の記憶装置に書き出されてもよい。

【００２４】

なお、プロセッサ１１は、デバッグ対象のソフトウェアが実行される時間を複数の区間

に分割し、同じ区間に属する履歴情報同士を比較し、異なる区間に属する履歴情報同士は比較しないようにしてもよい。この区間は、経過時間や命令の実行数に基づいて決定される周期的な区間であってもよい。また、プロセッサ１１は、ロック獲得命令またはロック解放命令が実行されたとき、実行するプログラムモジュールが切り替えられたとき、または、実行中のプログラムモジュールの動作モードが変わったときに、区間が変わったと判定してもよい。これにより、不具合の原因となるプログラムモジュールの判定結果の精度を向上させることができる。

【００２５】

第１の実施の形態の情報処理装置１０によれば、プロセッサ１１，１２が、制御ソフトウェア１４を用いてデバッグ対象のソフトウェアの命令を実行し、制御ソフトウェア１４の制御のもとで、実行した命令に関する履歴情報を収集し排他制御の不備を判定する。これにより、特殊なハードウェアやシミュレーション環境を用意しなくてもよく、デバッグ対象のソフトウェアが動作する本番システムまたは本番環境と同等のハードウェア構成のシステムを用いて、デバッグ対象のソフトウェアの排他制御の不備を判定することが可能となる。

【００２６】

［第２の実施の形態］

次に、第２の実施の形態を説明する。第２の実施の形態の情報処理装置は、排他制御の不備によって、２つ以上のＣＰＵからＲＡＭへのデータ書込が衝突したことを検出する。

【００２７】

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、ＣＰＵ１１１～１１４およびＲＡＭ１１５を有する。ＣＰＵ１１１～１１４とＲＡＭ１１５とは、システムバス１１６に接続されている。また、情報処理装置１００は、ＨＤＤ（Hard Disk Drive）１２１、画像信号処理部１２２、入力信号処理部１２３、媒体リーダ１２４および通信インタフェース１２５を有する。ＨＤＤ１２１、画像信号処理部１２２、入力信号処理部１２３、媒体リーダ１２４および通信インタフェース１２５は、入出力バス１２６に接続されている。システムバス１１６と入出力バス１２６とは、例えば、ブリッジ（図示せず）で接続されている。

【００２８】

ＣＰＵ１１１～１１４は、プログラムを実行するプロセッサである。ＣＰＵ１１１～１１４は、ＨＤＤ１２１からプログラムやデータの少なくとも一部をＲＡＭ１１５にロードし、ＲＡＭ１１５を利用してプログラムを実行する。ＣＰＵ１１１～１１４は、物理的に並列にプログラムを実行することができる。ＣＰＵ１１１～１１４それぞれは、複数のプロセッサコアを有していてもよく、１または２以上のキャッシュメモリを有してもよい。なお、第２の実施の形態では、複数のＣＰＵが並列にプログラムを実行する場合を説明するが、複数のプロセッサコアが並列にプログラムを実行する場合にも応用可能である。

【００２９】

ＲＡＭ１１５は、プログラムやデータを一時的に記憶する揮発性メモリである。ＲＡＭ１１５は、ＣＰＵ１１１～１１４から高速なシステムバス１１６を介してアクセスされる共有メモリであると言うことができる。なお、情報処理装置１００は、ＲＡＭ１１５以外の種類の揮発性メモリを備えていてもよく、複数個のメモリを備えていてもよい。

【００３０】

ＨＤＤ１２１は、ＯＳプログラム、ドライバプログラム、ユーザプログラム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いてもよく、複数の不揮発性の記憶装置を備えてもよい。

【００３１】

画像信号処理部１２２は、何れかのＣＰＵ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１０１に画像を出力する。ディスプレイ１０１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 Displ

ay）、プラズマディスプレイ（ＰＤＰ：Plasma Display Panel）、有機ＥＬ（ＯＥＬ：Organic Electro-Luminescence）ディスプレイなどを用いることができる。

【００３２】

入力信号処理部１２３は、情報処理装置１００に接続された入力デバイス１０２から入力信号を取得し、何れかのＣＰＵに出力する。入力デバイス１０２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、入力信号処理部１２３には、複数の種類の入力デバイスが接続されていてもよい。

【００３３】

媒体リーダ１２４は、記録媒体１０３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１０３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１２４は、例えば、記録媒体１０３から読み取ったプログラムやデータをＲＡＭ１１５またはＨＤＤ１２１に格納する。

【００３４】

通信インタフェース１２５は、ネットワーク１０４に接続され、ネットワーク１０４を介して他の情報処理装置と通信するインタフェースである。通信インタフェース１２５は、ケーブルでスイッチやルータなどの通信装置と接続される有線通信インタフェースでもよいし、無線で基地局と接続される無線通信インタフェースであってもよい。

【００３５】

なお、情報処理装置１００は、クライアントコンピュータであってもよいし、サーバコンピュータであってもよい。情報処理装置１００は、媒体リーダ１２４を備えていなくてもよい。また、ユーザが操作する端末装置からネットワーク１０４経由で情報処理装置１００を制御できる場合、情報処理装置１００は、画像信号処理部１２２や入力信号処理部１２３を備えていなくてもよい。また、ディスプレイ１０１や入力デバイス１０２が、情報処理装置１００の筐体と一体に形成されてもよい。なお、ＣＰＵ１１１，１１２は、前述の第１の実施の形態におけるプロセッサ１１，１２の一例である。また、ＲＡＭ１１５は、前述の第１の実施の形態におけるメモリ１３の一例である。

【００３６】

図３は、ソフトウェアの階層例を示す図である。

情報処理装置１００で実行されるソフトウェアには、ユーザプログラム、システムプログラムおよび命令エミュレータが含まれる。ユーザプログラムは、アプリケーションプログラムなど、システムプログラムの機能を利用するものである。システムプログラムは、ＯＳプログラムやドライバプログラムなど、１または２以上のユーザプログラムの実行を制御するものである。命令エミュレータは、システムプログラムの使用する命令セットがＣＰＵ１１１～１１４の命令セットと異なるとき、命令の変換によって、当該システムプログラムをＣＰＵ１１１～１１４上で実行可能にするものである。命令エミュレータは、例えば、ＯＳプログラムと合わせて情報処理装置１００にインストールされる。

【００３７】

ユーザプログラムは、スーパーバイザコール（システムコール）によってＯＳカーネルの機能を呼び出すことがある。ＯＳカーネルの機能が呼び出されると、ＣＰＵの動作モードが変更され、制御がユーザプログラムからＯＳに移る。要求されたＯＳ処理が完了すると、制御がＯＳからユーザプログラムに戻される。また、ユーザプログラムとシステムプログラムの間だけでなく、ユーザプログラム間においても切替が発生し得る。ユーザプログラムの切替は、例えば、他のユーザプログラムの呼び出し、呼び出したユーザプログラムからの復帰、タイマ割り込みなどによって発生する。

【００３８】

プログラム（ユーザプログラムやシステムプログラムを含む）の命令が実行されるとき、命令エミュレータが、プログラムの命令をＣＰＵが解釈可能な命令セットの命令に変換

する。これにより、例えば、新しいＣＰＵを用いて古いＯＳを実行することが可能となる。命令エミュレータは、プログラムの命令を読み込み、読み込んだ命令を変換してＣＰＵに実行させる。ＣＰＵ１１１～１１４は、プログラムや命令エミュレータを並列に実行できる。第２の実施の形態では、後述するように、排他制御の不備によってＲＡＭ１１５へのデータ書込が衝突したことを検出する機能を命令エミュレータの中に実装する。

【００３９】

次に、排他制御の不備によって２つ以上のデータ書込が衝突する例およびＣＰＵ１１１～１１４がこのような排他制御の不備を判定する流れについて説明する。

図４は、データ書込時の排他制御不備の例を示す図である。

【００４０】

プログラムＡ，Ｂ，Ｃ，Ｄの４つを考える。例えば、プログラムＡ，Ｂはユーザプログラムであり、プログラムＣはＯＳプログラムである。プログラムＡ，Ｂ，ＣはＣＰＵ１１１で実行され、プログラムＤはＣＰＵ１１２で実行される。プログラムＡは、呼出命令によってプログラムＢを呼び出す。これにより、実行中のプログラムがプログラムＡからプログラムＢに切り替わる。プログラムＢは、スーパーバイザコール命令（ＳＶＣ命令）によってプログラムＣを呼び出す。これにより、ＣＰＵ１１１の動作モードが変更され、実行中のプログラムがプログラムＢからプログラムＣに切り替わる（ＳＴ１０）。

【００４１】

その後に、プログラムＣ，ＤがＲＡＭ１１５のブロック＃２にユーザデータを書き込む場合を考える。排他制御が正常に行われるとき、プログラムＣは、テストアンドセット命令（ＴＳ命令）によって、ＲＡＭ１１５のブロック＃１に記憶されたロックワードを更新する。ロックワードは、ＲＡＭ１１５上の領域のロックが獲得されているか否かを示すフラグの集合である。ブロック＃２に対応するフラグをＯＦＦからＯＮに更新することで、プログラムＣはブロック＃２のロックを獲得したことになる（ＳＴ１１）。

【００４２】

プログラムＤがＴＳ命令によってブロック＃２のロックを獲得しようとしたとき、プログラムＣによって既にロックが獲得されていれば（フラグがＯＮになっていれば）、プログラムＤのＴＳ命令は失敗する（ＳＴ１２）。そして、プログラムＤは、継続的に（例えば、定期的に）ブロック＃１に記憶されたロックワードを確認し、ブロック＃２のロックが解放されるまで（ブロック＃２に対応するフラグがＯＦＦになるまで）待つことになる。このような排他制御の方法は、スピンロックと呼ばれることがある。

【００４３】

ブロック＃２のロックを獲得したプログラムＣは、ストア命令（ＳＴＲ命令）によってブロック＃２にデータを書き込む（ＳＴ１３）。データ書込が成功すると、プログラムＣは、ＳＴＲ命令によってブロック＃１に記憶されたロックワードを更新する。ブロック＃２に対応するフラグをＯＮからＯＦＦに更新することで、プログラムＣはブロック＃２のロックを解放したことになる（ＳＴ１４）。

【００４４】

ブロック＃２のロックが解放されると、プログラムＤは、ＴＳ命令によってブロック＃１に記憶されたロックワードを更新し、ブロック＃２のロックを獲得する（ＳＴ１５）。ブロック＃２のロックを獲得したプログラムＤは、プログラムＣの場合と同様に、ブロック＃２にデータを書き込む（ＳＴ１６）。ブロック＃２へのデータ書込が成功すると、プログラムＤは、ＳＴＲ命令によってブロック＃１に記憶されたロックワードを更新し、ブロック＃２のロックを解放する（ＳＴ１７）。同一の領域に対するデータ書込が連続しても、排他制御が正常に行われる限り、それらデータ書込は正常であると言える。

【００４５】

その後に、プログラムＣ，ＤがＲＡＭ１１５のブロック＃３にデータを書き込む場合を考える。排他制御が正常に行われないとき、プログラムＣは、ブロック＃３のロックを獲得せずに（ブロック＃３に対応するフラグをＯＮにせずに）、ＳＴＲ命令によってブロック＃３にデータを書き込む（ＳＴ１８）。また、プログラムＤも、ブロック＃３にデータ

を書き込む（ＳＴ１９）。排他制御なしにプログラムＣによるデータ書込とプログラムＤによるデータ書込とが短い時間内に行われた場合、このデータ書込の衝突によって、プログラムＣのデータが意図せず破壊されることになり得る。その結果、プログラムＣの処理が異常停止するなどの不具合が発生する可能性がある。

【００４６】

図５は、データ書込時の排他制御不備の検出例を示すシーケンス図である。

ここでは、ＣＰＵ１１１がＣＰＵ１１２との間でデータ書込が衝突していないか確認する場合を考える。実装時はＣＰＵ１１１は、ＣＰＵ１１２と同様に他ＣＰＵ（ＣＰＵ１１３，１１４）との間でも、データ書込が衝突していないか確認する。また、ＣＰＵ１１１以外のＣＰＵも、ＣＰＵ１１１と同様にデータ書込が衝突していないか確認する。

【００４７】

ＲＡＭ１１５には、各ＣＰＵに対応する制御領域が用意される。ＣＰＵ１１１用の制御領域には、ＣＰＵ１１１がデータを書き込んだ領域を示すアドレス情報やＣＰＵ１１２～１１４から収集したアドレス情報が格納される。また、ＣＰＵ１１１用の制御領域には、アドレス情報の収集をＣＰＵ１１２～１１４に依頼するか否かを示す依頼フラグが格納される。ＣＰＵ１１２用の制御領域にも、同様の情報が格納される。

【００４８】

ＣＰＵ１１１は、以下の制御を開始するとき、ＣＰＵ１１１用の制御領域に格納された依頼フラグをＯＮにする（ＳＴ２０）。命令エミュレータを用いてＣＰＵ１１１でＳＴＲ命令が実行されるとき、ＣＰＵ１１１は、アクセス先の領域を示すアドレス情報を生成して、ＣＰＵ１１１用の制御領域に格納する（ＳＴ２１）。また、命令エミュレータを用いてＣＰＵ１１２でＳＴＲ命令が実行されるとき、ＣＰＵ１１２は、アクセス先の領域を示すアドレス情報を生成して、ＣＰＵ１１２用の制御領域に格納する（ＳＴ２２）。このとき、ＣＰＵ１１２は、ＣＰＵ１１１の依頼フラグを確認し、依頼フラグがＯＮであるときは、生成したアドレス情報をＣＰＵ１１１用の制御領域にコピーする（ＳＴ２３）。

【００４９】

同様にして、ＣＰＵ１１１は、ＣＰＵ１１１でＳＴＲ命令が実行されるとき、アドレス情報を生成してＣＰＵ１１１用の制御領域に格納する（ＳＴ２４）。ＣＰＵ１１２は、ＣＰＵ１１２でＳＴＲ命令が実行されるとき、アドレス情報を生成してＣＰＵ１１２用の制御領域に格納する（ＳＴ２５）。また、ＣＰＵ１１２は、依頼フラグがＯＮであることを確認し、アドレス情報をＣＰＵ１１１用の制御領域にコピーする（ＳＴ２６）。

【００５０】

後述する所定の条件が満たされると、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に格納された依頼フラグをＯＦＦにし、ＣＰＵ１１２からのアドレス情報の収集を停止する。そして、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に収集されているアドレス情報を用いて、直近の１区間内でＣＰＵ１１１とＣＰＵ１１２とが同一または近い領域に対してデータ書込を行ったか（データ書込の衝突があったか）確認する。すなわち、ＣＰＵ１１１は、ＣＰＵ１１１のアドレス情報が示すアドレスとＣＰＵ１１２のアドレス情報が示すアドレスとが、重複または近接しているか確認する（ＳＴ２７）。

【００５１】

データ書込の衝突が検出された場合、ＣＰＵ１１１は、ＣＰＵ１１１で実行中のプログラムおよびＣＰＵ１１２で実行中のプログラムの少なくとも一方に排他制御の不備があると判定する。一方、データ書込の衝突が検出されない場合、ＣＰＵ１１１は、直近の１区間内で実行された命令については排他制御の不備がないと判定する。図５の排他制御不備なしの例では、直近の１区間について排他制御の不備がないと判定している。なお、直近の１区間とは、依頼フラグを前回ＯＮにしてから、依頼フラグをＯＦＦにするまで（後述する所定の条件が満たされるまで）の区間である。ＣＰＵ１１１は、排他制御の不備の有無を判定した後は、収集したアドレス情報を破棄してよい。

【００５２】

その後、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に格納された依頼フラグをＯＮに

し、ＣＰＵ１１２からのアドレス情報の収集を再開する（ＳＴ２８）。ステップＳＴ２１～ＳＴ２６と同様に、ＣＰＵ１１１は、アドレス情報を適宜生成してＣＰＵ１１１用の制御領域に格納する。また、ＣＰＵ１１２は、アドレス情報を適宜生成してＣＰＵ１１２用の制御領域に格納すると共に、ＣＰＵ１１１用の制御領域にコピーする。

【００５３】

所定の条件が満たされると、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に格納された依頼フラグをＯＦＦにし、ＰＵ１１２からのアドレス情報の収集を停止する。そして、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に収集されているアドレス情報を用いて、排他制御の不備の有無を判定する。図５の排他制御不備ありの例では、ここで排他制御の不備があると判定している（ＳＴ２９）。排他制御の不備があると判定した場合、ＣＰＵ１１１は、ＣＰＵ１１１，１１２で実行されていたプログラムを特定するためのプログラム情報などを含むログ情報を生成し、ログファイルとしてＨＤＤ１２１に出力する。

【００５４】

このように、ＣＰＵ１１１は、１つの区間内で収集されたアドレス情報同士を比較することで、排他制御の不備を判定する。排他制御の不備を判定するタイミング、すなわち、区間を区切るタイミングは、次の６つの条件の何れかが満たされたときとする。

【００５５】

第１の条件は、他のユーザプログラムの呼び出し、呼び出したユーザプログラムからの復帰、タイマ割り込みなどによってユーザプログラム間の切替が発生したことである。第２の条件は、ＳＶＣ命令などによってユーザプログラムとシステムプログラム間の切替が発生したことである。第１および第２の条件に応じて区間を区切ることで、排他制御の不備があるプログラムの範囲が限定できるので、不備の特定が容易になる。第３の条件は、ＲＡＭ１１５の領域に対してロックが獲得されたことである。第４の条件は、ロックが解放されたことである。第３および第４の条件に応じて区間を区切ることで、正常な排他制御のもとで行われるデータ書込が排他制御の不備であると誤判定されるのを抑制できる。

【００５６】

第５の条件は、実行中のプログラムの動作モードの切り替えが発生したことである。動作モードには、入出力装置からの割り込みやタイマ割り込みを禁止するモードと禁止しないモードとがある。第６の条件は、排他制御の不備が前回判定されてから（依頼フラグがＯＮになってから）、所定数の命令が実行されたことである。

【００５７】

図６は、排他制御不備の有無の判定例を示す図である。

排他制御の不備を判定するにあたり、第２の実施の形態では、ＲＡＭ１１５の各ブロックに対するデータ書込の有無を示す書込ビットマップが用いられる。書込ビットマップでは、直近の区間においてデータ書込があったブロック（ブロックについては、後述する図１４に関する説明を参照）に対応するビットが１に設定され、データ書込がなかったブロックに対応するビットが０に設定されている。

【００５８】

ＣＰＵ１１１は、ＣＰＵ１１１で生成された書込ビットマップとＣＰＵ１１２で生成された書込ビットマップの論理積（ＡＮＤ）を求める。論理積としてのビットマップでは、ＣＰＵ１１１，１１２の両方からデータ書込のあったブロックに対応するビットが１になり、それ以外のブロックに対応するビットが０になる。論理積の全ビットが０であれば、ＣＰＵ１１１，１１２は同じブロックに対してデータ書込を行っていないことになる。

【００５９】

ただし、論理積としてのビットマップの中に１のビットがあっても、そのビットに対応するブロックはロックワードが記憶されているブロックである可能性がある。ＣＰＵ１１１，１１２の両方がロックワードを更新することは正常な排他制御であるため、ロックワードへのアクセスを除外して排他制御の不備を判定することが好ましい。

【００６０】

そこで、ＣＰＵ１１１は、ロック獲得命令としてのＴＳ命令によってアクセスされたブ

ロックを示す書込ビットマップを生成する。ＣＰＵ１１１は、上記の論理積としてのビットマップと排他制御用の書込ビットマップの排他的論理和（ＸＯＲ）を求める。排他的論理和としてのビットマップでは、ロックワードでないデータがＣＰＵ１１１，１１２の両方から書き込まれたブロックに対応するビットが１になり、それ以外のブロックに対応するビットが０になる。論理積としてのビットマップの中に１のビットがあっても、排他的論理和の全ビットが０であれば、ＣＰＵ１１１，１１２は排他制御の不備に該当するような同一ブロックへのデータ書込を行っていないことになる。

【００６１】

以上では、同一ブロックに対するデータ書込の有無を確認したが、ＣＰＵ１１１は、同様にして近接するブロックに対するデータ書込の有無についても確認する。例えば、ＣＰＵ１１１は、ＣＰＵ１１１の書込ビットマップで１に設定されているビットの前後所定数のビットも１に書き換えて、ＣＰＵ１１２の書込ビットマップとの論理積を求める。この論理積としてのビットマップの中に１のビットがある場合、ＣＰＵ１１１は、この論理積としてのビットマップと排他制御用の書込ビットマップとの排他的論理和を求める。

【００６２】

このようにして、書換前のＣＰＵ１１１の書込ビットマップを用いた論理積または排他的論理和の全ビットが０になれば、同一ブロックに対する実質的なデータ書込の衝突がないと判断できる。また、書換後のＣＰＵ１１１の書込ビットマップを用いた論理積または排他的論理和の全ビットが０になれば、近接するブロックに対する実質的なデータ書込の衝突がないと判断できる。ＣＰＵ１１１は、同一または近接するブロックに対する実質的なデータ書込の衝突がなければ、直近の区間について排他制御の不備がないと判定する。一方、ＣＰＵ１１１は、それ以外の場合は排他制御の不備があると判定する。

【００６３】

次に、情報処理装置１００の機能およびデータ構造について説明する。

図７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、命令判別部１３１、命令変換部１３２、制御情報記憶部１３３、ログファイル記憶部１３４、起動部１３５およびデバッグ支援部１３６を有する。

【００６４】

命令判別部１３１、命令変換部１３２、起動部１３５およびデバッグ支援部１３６は、命令エミュレータの中のソフトウェアモジュールとして実現できる。命令判別部１３１、命令変換部１３２およびデバッグ支援部１３６の処理は、複数のＣＰＵ（ＣＰＵ１１１～１１４）で並列に実行される。ここでは主に、命令判別部１３１、命令変換部１３２、起動部１３５およびデバッグ支援部１３６がＣＰＵ１１１で動作する場合を説明する。制御情報記憶部１３３は、ＲＡＭ１１５に確保した領域として実現できる。ログファイル記憶部１３４は、ＨＤＤ１２１に確保した記憶領域として実現できる。

【００６５】

命令判別部１３１は、命令（例えば、システムプログラムの命令）が読み込まれると、読み込まれた命令の種類を判別する。命令の種類が所定の種類である場合、命令判別部１３１は、命令を命令変換部１３２とデバッグ支援部１３６の両方に渡す。命令の種類が所定の種類でない場合、命令判別部１３１は、命令を命令変換部１３２に渡す。第２の実施の形態では、所定の種類の命令として主に、データの書込またはロック解放に用いるＳＴＲ命令、ロック獲得に用いるＴＳ命令、ユーザプログラムの呼出に用いる呼出命令、および、システムプログラムの呼出に用いるＳＶＣ命令の４種類を考える。

【００６６】

命令変換部１３２は、命令判別部１３１から渡された命令をＣＰＵ１１１が解釈可能な命令に変換し、ＣＰＵ１１１に実行させる。命令の実行が完了すると、命令変換部１３２は、命令の実行結果をデバッグ支援部１３６に通知する。例えば、命令変換部１３２は、ＴＳ命令が実行されたときにはロック獲得の成否をデバッグ支援部１３６に通知する。

【００６７】

制御情報記憶部１３３は、排他制御の不備の判定に用いられる制御情報を記憶する。制

御情報記憶部１３３の領域は、デバッグ支援部１３６が起動するときに起動部１３５によってＲＡＭ１１５上に確保される。制御情報記憶部１３３は、ＣＰＵ１１１～１１４から共通にアクセス可能である。ただし、アドレス情報やプログラム情報を収集する領域については、制御情報記憶部１３３内にＣＰＵ毎に区分して設けられている。

【００６８】

ログファイル記憶部１３４は、ログファイルを記憶する。ログファイルには、排他制御の不備に該当するデータ書込の衝突が検出されたときに、問題となったデータ書込についての情報（アドレス情報やプログラム情報など）がログ情報として書き込まれる。情報処理装置１００のユーザは、デバッグ作業において、ログファイルに記載されたログ情報を分析することで、排他制御の不備のあるプログラムを特定することが容易となり、不具合が発生しないようにプログラムを修正することが容易となる。

【００６９】

起動部１３５は、ユーザからデバッグ開始のコマンドやデバッグ終了のコマンドを受け付ける。デバッグ開始のコマンドが入力されると、起動部１３５は、ＲＡＭ１１５に制御情報記憶部１３３の領域を確保し、デバッグ支援部１３６を起動して排他制御の不備を判定できるようにする。デバッグ終了のコマンドが入力されると、起動部１３５は、デバッグ支援部１３６を停止させ、制御情報記憶部１３３の領域を解放する。

【００７０】

デバッグ支援部１３６は、命令判別部１３１によって読み込まれる命令を監視し、ＣＰＵ１１１～１１４で実行されるプログラムの排他制御の不備を判定する。デバッグ支援部１３６は、情報収集部１３７、情報提供部１３８および検出部１３９を有する。

【００７１】

情報収集部１３７は、ＣＰＵ１１１が排他制御の不備の判定に用いる情報を収集する。情報収集部１３７は、ＣＰＵ１１１でＳＴＲ命令が実行されるとき、当該ＳＴＲ命令に関するアドレス情報やプログラム情報などを生成して制御情報記憶部１３３に格納する。また、情報収集部１３７は、ＣＰＵ１１１でＴＳ命令が実行されるとき、当該ＴＳ命令に関するアドレス情報などを生成して制御情報記憶部１３３に格納する。また、情報収集部１３７は、ＣＰＵ１１１の依頼フラグをＯＮにすることで、ＣＰＵ１１２～１１４からＳＴＲ命令に関するアドレス情報やプログラム情報などを収集する。

【００７２】

情報提供部１３８は、ＣＰＵ１１２～１１４が排他制御の不備の判定に用いる情報を、ＣＰＵ１１２～１１４に提供する。情報提供部１３８は、ＣＰＵ１１１でＳＴＲ命令が実行されるとき、ＣＰＵ１１２～１１４の依頼フラグを確認する。依頼フラグがＯＮであれば、情報提供部１３８は、情報収集部１３７が生成した当該ＳＴＲ命令に関するアドレス情報やプログラム情報などをＣＰＵ１１２～１１４に提供する。具体的には、情報提供部１３８は、生成されたアドレス情報やプログラム情報などを、制御情報記憶部１３３の中のＣＰＵ１１２～１１４に対応する領域にコピーする。

【００７３】

検出部１３９は、前述のような所定の条件が満たされたタイミングで、制御情報記憶部１３３の中のＣＰＵ１１１に対応する領域に収集された情報を用いて、排他制御の不備を判定する。所定の条件は、例えば、ＣＰＵ１１１で呼出命令またはＳＶＣ命令などが実行されてプログラムの切り替えが発生すること、ＴＳ命令によりロック獲得が成功したこと、ロックが解放されること、または、前回判定を行ってからＣＰＵ１１１で所定数の命令が実行されたことである。

【００７４】

なお、情報収集部１３７は、命令の実行が完了する前に当該命令に関する情報を生成してもよいし、命令の実行が完了してから当該命令に関する情報を生成してもよい。生成された情報を情報提供部１３８がコピーするタイミングについても同様である。また、検出部１３９は、呼出命令、ＳＶＣ命令またはロック解放を示すＳＴＲ命令に応じて排他制御の不備を判定するときは、その命令の実行が完了する前に判定を開始してもよいし、命令

の実行が完了してから判定を開始してもよい。また、検出部１３９は、命令実行数に応じて排他制御の不備を判定するときは、最後の命令の実行が完了する前に判定を開始してもよいし、最後の命令の実行が完了してから判定を開始してもよい。

【００７５】

図８は、第２の実施の形態の制御情報の全体構造例を示す図である。

制御情報記憶部１３３には、図８に示すような構造をもつ制御情報が格納される。制御情報は、他ＣＰＵ情報を管理するために、管理情報１４１、ポインタテーブル１４２を含む複数の第１のポインタテーブル、および、収集領域１４３を含む複数の収集領域を有する。また、制御情報は、自ＣＰＵ情報を管理するために、ポインタテーブル１５１（第２のポインタテーブル）、一般記録領域１５２を含む複数の一般記録領域、排他記録領域１５３を含む複数の排他記録領域、排他管理情報１５４を含む複数の排他管理情報、および、作業領域１５５を含む複数の作業域を有する。また、制御情報は、ログ情報を管理するために、ログ管理情報１５６およびログ記録領域１５７を有する。

【００７６】

管理情報１４１は、情報処理装置１００全体の状態やＣＰＵ１１１～１１４それぞれの状態を示す。第１のポインタテーブルは、ＣＰＵ１１１～１１４に対応して設けられる。１つの第１のポインタテーブルは、１つの「自ＣＰＵ」を示す。管理情報１４１から第１のポインタテーブルを辿ることができる。収集領域には、他ＣＰＵから収集された情報が格納される。収集領域は、「自ＣＰＵ」と「他ＣＰＵ」の組み合わせ毎に設けられる。１つの「自ＣＰＵ」に対応する第１のポインタテーブルから、「他ＣＰＵ」に対応する３つの収集領域を辿ることができる。例えば、ＣＰＵ１１１に対応する第１のポインタテーブルから、ＣＰＵ１１２～１１４に対応する収集領域を辿ることができる。

【００７７】

ポインタテーブル１５１は、ＣＰＵ１１１～１１４それぞれのロック獲得状況を示す情報を含む。一般記録領域には、自ＣＰＵで実行されるＳＴＲ命令に関するアドレス情報やプログラム情報などが格納される。排他記録領域には、自ＣＰＵで実行されるＴＳ命令に関するアドレス情報などが格納される。排他管理情報は、ロック獲得時に更新されたロックワードのアドレスを含む。ロックワードのアドレスは、ＳＴＲ命令がロック解放を示すものであるか判断するために用いられる。作業領域は、自ＣＰＵで生成された情報を一時的に記憶する。一般記録領域、排他記録領域、排他管理情報および作業領域は、ＣＰＵ１１１～１１４に対応して設けられ、ポインタテーブル１５１から辿ることができる。

【００７８】

ログ管理情報１５６は、ログ記録領域１５７を管理するための情報を含む。ログ記録領域１５７には、排他制御の不備があると判定されたときに、検出部１３９によって生成されたログ情報が格納される。ログ記録領域１５７に格納されたログ情報は、ログファイル記憶部１３４に記憶された（ＨＤＤ１２１上の）ログファイルに書き出される。

【００７９】

図９は、第２の実施の形態の管理情報のデータ構造例を示す図である。

管理情報１４１は、ヘッダ内に、システム状態、ロックワードおよび対象プログラムの項目を有する。また、管理情報１４１は、ＣＰＵ１１１～１１４それぞれについて、ＣＰＵ状態およびポインタテーブルのアドレスの項目を有する。

【００８０】

システム状態の項目は、デバッグ開始フラグを含む。デバッグ開始フラグは、デバッグ支援部１３６が動作中であるか否かを示す。デバッグ開始フラグは、１ビットで表現できる。例えば、デバッグ開始フラグ＝１（ＯＮ）は、デバッグ支援部１３６が動作中であり排他制御の不備が判定されることを示す。デバッグ開始フラグ＝０（ＯＦＦ）は、デバッグ支援部１３６が停止中であり排他制御の不備が判定されないことを示す。ロックワードの項目は、管理情報１４１へのアクセスの排他制御に用いられるロックワードを含む。対象プログラムの項目は、デバッグ対象のプログラム（排他制御の不備を判定するプログラム）の識別情報を含む。対象プログラムは、例えば、ユーザが指定することもできる。対

象プログラムを限定することで、情報処理装置１００の負荷を軽減できる。
【００８１】
ＣＰＵ状態の項目は、依頼フラグを含む。依頼フラグは、あるＣＰＵが他ＣＰＵに対して情報収集を依頼しているか否かを示す。依頼フラグは、１ビットで表現できる。例えば、依頼フラグ＝１（ＯＮ）は、情報収集を依頼中であることを示す。依頼フラグ＝０（ＯＦＦ）は、情報収集を依頼していないことを示す。ポインタテーブルのアドレスの項目は、第１のポインタテーブルが存在するＲＡＭ１１５の領域の先頭アドレスである。
【００８２】
図１０は、第１のポインタテーブルのデータ構造例を示す図である。
ポインタテーブル１４２は、「他ＣＰＵ」それぞれについて（例えば、ＣＰＵ１１２～１１４それぞれについて）、領域状態、ロックワードおよび収集領域のアドレスの項目を有する。領域状態の項目は、使用フラグを含む。使用フラグは、収集領域が使用されているか否か、すなわち、他ＣＰＵから収集された情報が存在するか否かを示す。使用フラグは、１ビットで表現できる。例えば、使用フラグ＝１（ＯＮ）は、収集された情報が存在することを示す。使用フラグ＝０（ＯＦＦ）は、収集された情報が存在しないことを示す。ロックワードの項目は、ポインタテーブル１４２へのアクセスの排他制御に用いられる。収集領域のアドレスの項目は、ＲＡＭ１１５上の収集領域の先頭アドレスである。
【００８３】
図１１は、収集領域のデータ構造例を示す図である。
収集領域１４３は、収集領域ヘッダ１４４、集計領域１４５を含む複数の集計領域、および、追記領域１４７を含む複数の追記領域を有する。
【００８４】
収集領域ヘッダ１４４は、集計領域の集合の状態を示す。各集計領域は、ＲＡＭ１１５上のデータ書込が行われたブロックを示す書込ビットマップを含む。１つの集計領域は、ＲＡＭ１１５の１つのアドレス範囲に対応する。１つの集計領域が担当するアドレス範囲の大きさは、予め設定されている。排他制御の不備を判定する１つの区間内で、異なるアドレス範囲に属する領域に対してデータ書込が行われた場合、複数の集計領域が使用されることになる。複数の集計領域は、連結リストとして形成されている。収集領域ヘッダ１４４からは、先頭の集計領域と末尾の集計領域とを辿ることができる。
【００８５】
１つの集計領域に対して、複数の追記領域が設けられる。各追記領域は、書込ビットマップ以外のアドレス情報やプログラム情報や時刻情報などを含む。１つの追記領域は、１回のデータ書込に対応する。１つの集計領域に対応する複数の追記領域は、連結リストとして形成されている。集計領域からは、先頭の追記領域を辿ることができる。
【００８６】
図１２は、収集領域ヘッダのデータ構造例を示す図である。
収集領域ヘッダ１４４は、領域状態、先頭の集計領域のアドレスおよび末尾の集計領域のアドレスの項目を有する。領域状態の項目は、使用フラグを含む。使用フラグは、少なくとも１つの集計領域が使用されているか否かを示す。使用フラグは、１ビットで表現できる。例えば、使用フラグ＝１（ＯＮ）は、少なくとも１つの集計領域が使用されていることを示す。使用フラグ＝０（ＯＦＦ）は、何れの集計領域も使用されていないことを示す。先頭の集計領域のアドレスの項目は、ＲＡＭ１１５上の先頭の集計領域の先頭アドレスである。末尾の集計領域のアドレスの項目は、末尾の集計領域の先頭アドレスである。
【００８７】
図１３は、集計領域のデータ構造例を示す図である。
集計領域１４５は、ヘッダ内に、次の集計領域のアドレス、領域状態、書込ビットマップに対応するメモリアドレス、先頭の追記領域のアドレスおよび末尾の追記領域のアドレスの項目を有する。また、集計領域１４５は、書込ビットマップ１４６を有する。
【００８８】
次の集計領域のアドレスの項目は、ＲＡＭ１１５上の次の集計領域の先頭アドレスを含

む。領域状態の項目は、末尾フラグおよび使用フラグを含む。末尾フラグは、集計領域１４５が連結リストの末尾であるか否かを示す。末尾フラグは、１ビットで表現できる。例えば、末尾フラグ＝１（ＯＮ）は、集計領域１４５が連結リストの末尾であることを示す。末尾フラグ＝０（ＯＦＦ）は、集計領域１４５の後方に他の集計領域があることを示す。使用フラグは、集計領域１４５が使用されているか否かを示す。使用フラグは、１ビットで表現できる。例えば、使用フラグ＝１（ＯＮ）は、集計領域１４５が使用中であることを示す。使用フラグ＝０（ＯＦＦ）は、集計領域１４５が未使用であることを示す。

【００８９】

書込ビットマップに対応するメモリアドレスの項目は、書込ビットマップ１４６が担当するＲＡＭ１１５のアドレス範囲の先頭アドレスを含む。先頭の追記領域のアドレスの項目は、集計領域１４５に対応するＲＡＭ１１５上の複数の追記領域のうち、先頭の追記領域の先頭アドレスを含む。末尾の追記領域のアドレスの項目は、集計領域１４５に対応するＲＡＭ１１５上の複数の追記領域のうち、末尾の追記領域の先頭アドレスである。

【００９０】

図１４は、書込ビットマップのデータ構造例を示す図である。

書込ビットマップ１４６は、３２列×４行＝１２８個のビットを含む。１つのビットは、ＲＡＭ１１５上の１つのブロックに対応する。ブロックは、ＣＰＵ１１１～１１４がアクセスする領域の最小単位であり、例えば、４バイトの領域である。この場合、書込ビットマップ１４６は、１２８ブロック×４バイト＝５１２バイトの領域をカバーすることができる。書込ビットマップ１４６に含まれる各ビットは、当該ビットに対応するブロックに対してデータ書込が行われたか否かを示す。ビット＝１（ＯＮ）は、データが書き込まれたことを示し、ビット＝０（ＯＦＦ）は、データ書き込まれていないことを示す。書込ビットマップ１４６の初期化時には、全てのビットが０に設定される。

【００９１】

図１５は、追記領域のデータ構造例を示す図である。

追記領域１４７は、領域状態、次の追記領域のアドレス、書込が行われたメモリアドレス、ベースアドレス、プログラム情報、書込時刻およびＣＰＵ番号の項目を有する。

【００９２】

領域状態の項目は、末尾フラグおよび使用フラグを含む。末尾フラグは、１つの集計領域に対する追記領域の連結リストの中で、追記領域１４７が末尾であるか否かを示す。末尾フラグは、１ビットで表現できる。使用フラグは、追記領域１４７が使用されているか否かを示す。使用フラグは、１ビットで表現できる。この末尾フラグおよび使用フラグのＯＮ・ＯＦＦの意味は、前述の集計領域１４５のものと同様である。

【００９３】

次の追記領域のアドレスの項目は、ＲＡＭ１１５上の次の追記領域の先頭アドレスである。書込が行われたメモリアドレスの項目は、データ書込が行われた領域の先頭アドレスである。ベースアドレスの項目は、データ書込が行われた時点のベースアドレス、すなわち、アクセス先の領域を特定するための起点となるアドレスを含む。ベースアドレスは、各ＣＰＵが備えるベースレジスタに格納されている。ベースレジスタとしては、各ＣＰＵが備える複数のレジスタのうちの１または２以上のレジスタが使用される。ベースアドレスは、アクセス先の領域が不適切になった原因を分析するのに有用な情報である。

【００９４】

プログラム情報の項目は、データ書込を行ったＣＰＵにおいてデータ書込の時点で実行されていたプログラムを特定するための情報を含む。プログラム情報としては、例えば、プログラム状態ワード（ＰＳＷ：Program Status Word）を用いることができる。ＰＳＷは、次に実行する命令を示す命令アドレスや、ＣＰＵによる命令実行を制御するための各種の制御フラグを含む。書込時刻の項目は、データ書込が行われた時刻を含む。ＣＰＵ番号は、データ書込を行ったＣＰＵを識別するための識別番号を含む。なお、ＣＰＵ１１１～１１４それぞれには、予め識別番号が付与されているものとする。

【００９５】

図１６は、第２のポインタテーブルのデータ構造例を示す図である。
ポインタテーブル１５１は、ＣＰＵ１１１～１１４それぞれについて、一般記録領域のアドレス、排他記録領域のアドレス、ロック状態、ロック獲得数、命令カウンタ、排他管理情報のアドレスおよび作業領域のアドレスの項目を有する。
【００９６】
一般記録領域のアドレスの項目は、当該ＣＰＵ用の一般記録領域（例えば、一般記録領域１５２）の先頭アドレスを含む。排他記録領域のアドレスの項目は、当該ＣＰＵ用の排他記録領域（例えば、排他記録領域１５３）の先頭アドレスを含む。
【００９７】
ロック状態の項目は、ロック獲得フラグを含む。ロック獲得フラグは、当該ＣＰＵがＲＡＭ１１５の少なくとも１つの領域に対して、ロックを現在獲得しているか否かを示す。ロック獲得フラグは、１ビットで表現できる。例えば、ロック獲得フラグ＝１（ＯＮ）は、ロックを獲得していることを示す。ロック獲得フラグ＝０（ＯＦＦ）は、ロックを獲得していないことを示す。ロック獲得数の項目は、当該ＣＰＵが獲得しているロックの数を含む。ロック獲得フラグがＯＮであれば、ロック獲得数は１以上になる。
【００９８】
命令カウンタの項目は、排他制御の不備を前回判定してから当該ＣＰＵが実行した命令の数を含む。命令カウンタは、当該ＣＰＵが命令を実行する毎にカウントアップされ、当該ＣＰＵが排他制御の不備を判定する毎に０にリセットされる。排他管理情報のアドレスの項目は、当該ＣＰＵ用の排他管理情報（例えば、排他管理情報１５４）が記憶されているＲＡＭ１１５の領域の先頭アドレスを含む。作業領域のアドレスの項目は、当該ＣＰＵ用の作業領域（例えば、作業領域１５５）の先頭アドレスを含む。
【００９９】
図１７は、ログ管理情報のデータ構造例を示す図である。
ログ管理情報１５６は、ロックワード、ログ情報のサイズ、ログ記録領域の全体サイズ、先頭のログ情報のアドレスおよび末尾のログ情報のアドレスの項目を含む。
【０１００】
ロックワードの項目は、ログ管理情報１５６へのアクセスの排他制御に用いられるロックワードを含む。ログ情報のサイズの項目は、ログ記録領域１５７に現在記憶されているログ情報の量（例えば、バイト数）を示す。ログ記録領域の全体サイズの項目は、使用領域と空き領域の両方を含むログ記録領域１５７の大きさ（例えば、バイト数）を示す。先頭のログ情報のアドレスは、ログ記録領域１５７に記憶された先頭のログ情報（最初に追加されたもの）の先頭アドレスを含む。末尾のログ情報のアドレスは、ログ記録領域１５７に記憶された末尾のログ情報（最後に追加されたもの）の先頭アドレスを含む。
【０１０１】
図１８は、第２の実施の形態のログ情報のデータ構造例を示す図である。
ログ記録領域１５７は、図１８に示すようなログ情報１５８を１単位として、１単位または２単位以上のログ情報を記憶することができる。２単位以上のログ情報は、例えば、連結リストとして記憶される。ログ情報１５８は、検出時刻の項目を有する。また、ログ情報１５８は、「他ＣＰＵ」に関して、書込が行われたメモリアドレス、ベースアドレス、プログラム情報および書込時刻の項目を有する。また、「自ＣＰＵ」に関して、書込が行われたメモリアドレス、ベースアドレスおよびプログラム情報の項目を有する。
【０１０２】
検出時刻の項目は、自ＣＰＵが排他制御の不備を検出した時刻を含む。
他ＣＰＵに関して、他ＣＰＵ番号の項目は、自ＣＰＵと衝突するデータ書込を行った他ＣＰＵの識別番号を含む。書込が行われたメモリアドレスの項目は、他ＣＰＵがデータ書込を行ったＲＡＭ１１５の領域の先頭アドレスを含む。ベースアドレスの項目は、他ＣＰＵがデータ書込を行った時点の他ＣＰＵのベースレジスタの値を含む。プログラム情報の項目は、他ＣＰＵがデータ書込を行った時点において他ＣＰＵが実行していたプログラムを特定するための情報（例えば、ＰＳＷ）を含む。書込時刻の項目は、他ＣＰＵがデータ

書込を行った時刻を含む。他ＣＰＵに関する情報としては、図１５に示すように、当該他ＣＰＵに対応する収集領域の中の追記領域に記憶された情報を用いることができる。

【０１０３】

自ＣＰＵに関して、自ＣＰＵ番号の項目は、自ＣＰＵの識別番号を含む。書込が行われたメモリアドレスの項目は、自ＣＰＵがデータ書込を行ったＲＡＭ１１５の領域の先頭アドレスを含む。ベースアドレスの項目は、自ＣＰＵがデータ書込を行った時点の自ＣＰＵのベースレジスタの値を含む。プログラム情報の項目は、自ＣＰＵがデータ書込を行った時点において自ＣＰＵが実行していたプログラムを特定するための情報（例えば、ＰＳＷ）を含む。これらの自ＣＰＵに関する情報としては、当該自ＣＰＵに対応する一般記録領域の中の追記領域に記憶された情報を用いることができる。

【０１０４】

なお、一般記録領域１５２は、収集領域１４３と同様のデータ構造によって実現できる。すなわち、一般記録領域１５２は、収集領域ヘッダ、複数の集計領域および複数の追記領域を有する。排他記録領域１５３は、収集領域１４３と同様に、収集領域ヘッダおよび複数の集計領域を有する。ただし、排他記録領域１５３は、追記領域を有しなくてよい。排他管理情報１５４は、現在獲得しているロックについてのロックワード（当該ロックの獲得時に更新されたロックワード）のアドレスを含む。ロック獲得数が２以上である場合、例えば、複数のロックワードのアドレスが連結リストとして記憶される。

【０１０５】

次に、情報処理装置１００が実行する処理について説明する。

図１９は、起動処理および停止処理の手順例を示すフローチャートである。

（Ｓ１１０）起動部１３５は、デバッグ開始のコマンドを受け付ける。デバッグ開始のコマンドは、デバッグ支援部１３６を起動することによって、情報処理装置１００をデバッグモード（排他制御の不備のログ情報が出力されるモード）で動作させることを示す。デバッグ開始のコマンドは、例えば、ユーザが入力デバイス１０２を用いて入力する。

【０１０６】

（Ｓ１１１）起動部１３５は、情報処理装置１００が既にデバッグモードで動作しているか、すなわち、デバッグ支援部１３６が起動中か判断する。既にデバッグモードである場合は処理が終了し、デバッグモードでない場合はステップＳ１１２に処理が進む。

【０１０７】

（Ｓ１１２）起動部１３５は、ＲＡＭ１１５に制御情報記憶部１３３のための領域を確保し、図８に示した構造の制御情報が記憶されるように当該領域を初期化する。

（Ｓ１１３）情報収集部１３７は、管理情報１４１に含まれるＣＰＵ１１１～１１４の依頼フラグをＯＦＦからＯＮに変更する。これにより、ＣＰＵ１１４～１１４それぞれは、他ＣＰＵで実行されたＳＴＲ命令に関する情報の収集を開始することになる。そして、デバッグ開始のコマンドに応じた起動部１３５のプロセスは終了する。

【０１０８】

（Ｓ１１４）起動部１３５は、デバッグ終了のコマンドを受け付ける。デバッグ終了のコマンドは、デバッグ支援部１３６を停止することによって情報処理装置１００を非デバッグモード（排他制御の不備のログ情報が出力されないモード）で動作させることを示す。デバッグ終了のコマンドは、例えば、ユーザが入力デバイス１０２を用いて入力する。

【０１０９】

（Ｓ１１５）起動部１３５は、制御情報記憶部１３３のための領域を解放する。そして、デバッグ終了のコマンドに応じた起動部１３５のプロセスは終了する。

なお、起動部１３５は、ＣＰＵ１１１～１１４の何れか１つ（例えば、ユーザ入力を受け付ける所定のＣＰＵ）で動作すればよい。一方、命令判別部１３１、命令変換部１３２およびデバッグ支援部１３６は、ＣＰＵ１１１～１１４それぞれで動作する。以下では、「自ＣＰＵ」をＣＰＵ１１１として、デバッグ支援について説明する。

【０１１０】

図２０は、データ書込時のデバッグ支援の手順例を示すフローチャートである。

（Ｓ１２０）命令判別部１３１は、命令（例えば、システムプログラムの命令）を１つ読み込む。読み込まれた命令は、命令変換部１３２によってＣＰＵ１１１が解釈可能な命令に変換されて実行される。以下のステップＳ１２１以降の処理は、命令の実行が完了する前に開始してもよいし、命令の実行が完了してから開始してもよい。

【０１１１】

（Ｓ１２１）命令判別部１３１は、読み込まれた命令がＳＴＲ命令（ストア命令）であるか判断する。読み込まれた命令がＳＴＲ命令である場合はステップＳ１２２に処理が進み、ＳＴＲ命令でない場合はステップＳ１２５に処理が進む。

【０１１２】

（Ｓ１２２）情報収集部１３７は、ＣＰＵ１１１のデータ書込の情報として、ＳＴＲ命令に関するアドレス情報などを収集する。また、情報提供部１３８は、ＣＰＵ１１１の情報をＣＰＵ１１２～１１４に提供する。アドレス情報収集の詳細は後述する。

【０１１３】

（Ｓ１２３）情報収集部１３７は、読み込まれたＳＴＲ命令が、獲得済みのロックを解放するための命令であるか判定する。ＳＴＲ命令がロック解放を意図したものか否かは、アクセス先のＲＡＭ１１５のアドレスが、ロック獲得時に更新されたロックワードを指し示しているか否かによって判定できる。ロック解放判定の詳細は後述する。

【０１１４】

（Ｓ１２４）読み込まれたＳＴＲ命令がロック解放を意図したものと判定された場合、ステップＳ１２９に処理が進む。それ以外の場合、ステップＳ１２８に処理が進む。

（Ｓ１２５）命令判別部１３１は、読み込まれた命令がＴＳ命令（テストアンドセット命令）であるか判断する。読み込まれた命令がＴＳ命令である場合はステップＳ１２６に処理が進み、ＴＳ命令でない場合はステップＳ１２７に処理が進む。

【０１１５】

（Ｓ１２６）情報収集部１３７は、ＴＳ命令が実行されてロック獲得が成功した場合、ＴＳ命令によって更新されたロックワードの位置を示すアドレス情報を記録しておく。ロックワードのアドレス情報は、ステップＳ１２３においてＳＴＲ命令がロック解放を意図したものか判定するときに用いられる。そして、ステップＳ１２９に処理が進む。

【０１１６】

（Ｓ１２７）命令判別部１３１は、読み込まれた命令が呼出命令やＳＶＣ命令（スーパーバイザコール命令）など、プログラムの切替を示す命令であるか判断する。読み込まれた命令がプログラムの切替を示す命令である場合はステップＳ１２９に処理が進み、プログラムの切替を示す命令でない場合はステップＳ１２８に処理が進む。

【０１１７】

（Ｓ１２８）情報収集部１３７は、ポインタテーブル１５１に含まれるＣＰＵ１１１の命令カウンタの値が、閾値を超えたか判断する。すなわち、情報収集部１３７は、情報処理装置１００がデバッグモードになって以降、または、排他制御の不備を前回判定して以降、ＣＰＵ１１１で実行された命令の数が閾値を超えたか判断する。命令カウンタの値が閾値を超えた場合はステップＳ１２９に処理が進み、それ以外の場合は処理が終了する。

【０１１８】

（Ｓ１２９）検出部１３９は、ＣＰＵ１１１に対応する一般記録領域、排他記録領域および収集領域に収集されたアドレス情報に基づいて、ＣＰＵ１１１とＣＰＵ１１２～１１４との間におけるデータ書込の衝突を検出する。ＣＰＵ１１１と何れかの他ＣＰＵとの間でデータ書込の衝突があると、検出部１３９は、ＣＰＵ１１１が実行したプログラムおよび当該他ＣＰＵが実行したプログラムの少なくとも一方に排他制御の不備があると判定し、プログラム情報などを含むログ情報を出力する。排他制御不備検出の詳細は後述する。

【０１１９】

このように、デバッグ支援部１３６は、ＣＰＵ１１１～１１４が行ったデータ書込についてのアドレス情報やプログラム情報を収集する。そして、デバッグ支援部１３６は、ロック獲得、ロック解放、ユーザプログラム間の切替、ユーザプログラムとシステムプログ

ラム間の切替、一定数の命令の実行の何れかがＣＰＵ１１１において発生した時点で、排他制御の不備を判定する。排他制御の不備が検出されると、デバッグ支援部１３６は、排他制御の不備があるプログラムを特定するのに有用な情報を含むログ情報を出力する。

【０１２０】

図２１は、アドレス情報収集の手順例を示すフローチャートである。

このアドレス情報収集は、前述のステップＳ１２２において実行される。

（Ｓ１３０）情報収集部１３７は、管理情報１４１を参照して、ＣＰＵ１１１で実行されているプログラムがデバッグ対象であるか判断する。プログラムがデバッグ対象である場合はステップＳ１３１に処理が進み、デバッグ対象でない場合は処理が終了する。

【０１２１】

（Ｓ１３１）情報収集部１３７は、命令判別部１３１から受け取ったＳＴＲ命令に基づいて、データ書込が行われるＲＡＭ１１５のアドレス（絶対アドレス）を算出する。

（Ｓ１３２）情報収集部１３７は、ＣＰＵ１１１に対応する作業領域に、ステップＳ１３１で算出したアドレスを格納する。また、情報収集部１３７は、ＣＰＵ１１１が備えるベースレジスタの現在値（現在のベースアドレス）やＣＰＵ１１１で実行されているプログラムを示すプログラム情報（例えば、現在のＰＳＷ）を確認し、作業領域に格納する。

【０１２２】

（Ｓ１３３）情報収集部１３７は、図１４に示したような書込ビットマップを生成し、ＣＰＵ１１１に対応する作業領域に格納する。このとき、情報収集部１３７は、ステップＳ１３１で算出されたアドレスに対応するビットを１とし、他のビットを０にする。

【０１２３】

（Ｓ１３４）情報収集部１３７は、ＣＰＵ１１１に対応する一般記録領域の中から、ステップＳ１３３で生成した書込ビットマップと担当するアドレス範囲が一致する集計領域を探す。情報収集部１３７は、該当する集計領域があれば当該集計領域を使用し、なければ未使用の集計領域を１つ獲得する。そして、情報収集部１３７は、論理和（ＯＲ）を求めることで、作業領域に格納された書込ビットマップを、使用する集計領域の書込ビットマップに合成する。また、情報収集部１３７は、上記の集計領域に対応する未使用の追記領域を１つ獲得し、ステップＳ１３２で生成した書込先のアドレス、ベースアドレスおよびプログラム情報を、作業領域から獲得した追記領域にコピーする。

【０１２４】

（Ｓ１３５）情報提供部１３８は、ＣＰＵ１１２～１１４の１つ（１つの他ＣＰＵ）を特定する。情報提供部１３８は、管理情報１４１を参照し、特定した他ＣＰＵの依頼フラグがＯＮであるか、すなわち、当該他ＣＰＵから情報収集が依頼されているか判断する。依頼フラグがＯＮである場合はステップＳ１３６に処理が進み、依頼フラグがＯＦＦである場合はステップＳ１３８に処理が進む。

【０１２５】

（Ｓ１３６）情報提供部１３８は、特定した他ＣＰＵに対応する複数の収集領域のうち、ＣＰＵ１１１の情報を収集するための収集領域を検出する。

（Ｓ１３７）情報提供部１３８は、ステップＳ１３４と同様の方法で、ＣＰＵ１１１の作業領域にある情報をステップＳ１３６で検出した収集領域にコピーする。すなわち、情報提供部１３８は、検出した収集領域の中から、ステップＳ１３３で生成した書込ビットマップと担当するアドレス範囲が一致する集計領域を探して集計領域を１つ使用する。情報提供部１３８は、論理和（ＯＲ）を求めることで、作業領域に格納された書込ビットマップを、使用する集計領域の書込ビットマップに合成する。また、情報提供部１３８は、上記の集計領域に対応する未使用の追記領域を１つ獲得し、作業領域に格納された書込先のアドレス、ベースアドレスおよびプログラム情報を追記領域にコピーする。

【０１２６】

（Ｓ１３８）情報提供部１３８は、全ての他ＣＰＵ（ＣＰＵ１１２～１１４）を確認したか判断する。全ての他ＣＰＵを確認し終えた場合は処理が終了し、未確認の他ＣＰＵがある場合はステップＳ１３５に処理が進む。

【０１２７】
図２２は、ロック獲得記録の手順例を示すフローチャートである。
このロック獲得記録は、前述のステップＳ１２６において実行される。
（Ｓ１４０）情報収集部１３７は、ＴＳ命令の完了を検出する。
【０１２８】
（Ｓ１４１）情報収集部１３７は、管理情報１４１を参照して、ＣＰＵ１１１で実行されているプログラムがデバッグ対象であるか判断する。プログラムがデバッグ対象である場合はステップＳ１４２に処理が進み、デバッグ対象でない場合は処理が終了する。
【０１２９】
（Ｓ１４２）情報収集部１３７は、ＴＳ命令によりロック獲得が成功したか判断する。ロック獲得が成功した場合は、ステップＳ１４３に処理が進む。ロック獲得が失敗した場合（ロック解放を待ち、後でＴＳ命令が再実行される場合）は、処理が終了する。
【０１３０】
（Ｓ１４３）情報収集部１３７は、実行されたＴＳ命令に基づいて、ロック獲得のためにアクセスされたＲＡＭ１１５の領域を示すアドレス、すなわち、更新されたロックワードが記憶されている領域のアドレス（絶対アドレス）を算出する。
【０１３１】
（Ｓ１４４）情報収集部１３７は、ＣＰＵ１１１に対応する排他管理情報の中に、ステップＳ１４３で算出したアドレスを追加する。また、情報収集部１３７は、ポインタテーブル１５１の中のＣＰＵ１１１に対応する情報を更新する。すなわち、情報収集部１３７は、ロック獲得フラグがＯＦＦであるときはロック獲得フラグをＯＮに変更する。また、情報収集部１３７は、ロック獲得数をインクリメント（１だけ加算）する。
【０１３２】
（Ｓ１４５）情報収集部１３７は、図１４に示したような書込ビットマップを生成し、ＣＰＵ１１１に対応する作業領域に格納する。このとき、情報収集部１３７は、ステップＳ１４３で算出されたアドレスに対応するビットを１とし、他のビットを０とする。
【０１３３】
（Ｓ１４６）情報収集部１３７は、ＣＰＵ１１１に対応する排他記録領域の中から、生成した書込ビットマップと担当するアドレス範囲が一致する集計領域を探す。情報収集部１３７は、該当する集計領域があれば当該集計領域を使用、なければ未使用の集計領域を１つ獲得する。そして、情報収集部１３７は、論理和（ＯＲ）を求めることで、生成した書込ビットマップを使用する集計領域の書込ビットマップに合成する。
【０１３４】
図２３は、ロック解放判定の手順例を示すフローチャートである。
このロック解放判定は、前述のステップＳ１２３において実行される。
（Ｓ１５０）情報収集部１３７は、管理情報１４１を参照して、ＣＰＵ１１１で実行されているプログラムがデバッグ対象であるか判断する。プログラムがデバッグ対象である場合はステップＳ１５１に処理が進み、デバッグ対象でない場合は処理が終了する。
【０１３５】
（Ｓ１５１）情報収集部１３７は、ＣＰＵ１１１に対応する排他管理情報の中から、読み込まれたＳＴＲ命令に基づいてデータ書込が行われるＲＡＭ１１５上の領域のアドレス（前述のステップＳ１３１で算出されたアドレス）を検索する。
【０１３６】
（Ｓ１５２）情報収集部１３７は、ステップＳ１５１において排他管理情報の中から該当するアドレスが検索されたか判断する。読み込まれたＳＴＲ命令が、ロック解放のためにロックワードを更新するものである場合、排他管理情報の中に該当するアドレスが存在する。排他管理情報の中から該当するアドレスが検索された場合はステップＳ１５３に処理が進み、該当するアドレスが検索されなかった場合は処理が終了する。
【０１３７】
（Ｓ１５３）情報収集部１３７は、ＣＰＵ１１１に対応する排他管理情報から、前述の

ステップＳ１３１で算出されたアドレスを削除する。また、情報収集部１３７は、ポインタテーブル１５１の中のＣＰＵ１１１に対応する情報を更新する。すなわち、情報収集部１３７は、ロック獲得数をデクリメント（１だけ減算）する。ロック獲得数が０になった場合、情報収集部１３７は、ロック獲得フラグをＯＮからＯＦＦに変更する。

【０１３８】

図２４は、排他制御不備検出の手順例を示すフローチャートである。

この排他制御不備検出は、前述のステップＳ１２９において実行される。

（Ｓ１６０）検出部１３９は、管理情報１４１を参照して、ＣＰＵ１１１で現在実行されているプログラムがデバッグ対象であるか判断する。デバッグ対象である場合はステップＳ１６２に処理が進み、デバッグ対象でない場合はステップＳ１６１に処理が進む。

【０１３９】

（Ｓ１６１）検出部１３９は、ＣＰＵ１１１において直近の所定時間以内にプログラムの切替が発生しており、かつ、現在実行されているプログラムの１つ前に実行されていたプログラムがデバッグ対象であるか判断する。上記の条件に該当する場合はステップＳ１６２に処理が進み、上記の条件に該当しない場合は処理が終了する。

【０１４０】

（Ｓ１６２）検出部１３９は、管理情報１４１の中のＣＰＵ１１１に対応する依頼フラグをＯＮからＯＦＦに変更する。これにより、他ＣＰＵからの情報収集が中断される。

（Ｓ１６３）検出部１３９は、ＣＰＵ１１１に対応する一般記録領域に、ＣＰＵ１１１で行われたデータ書込についての情報が存在するか判断する。ＣＰＵ１１１が直近の１区間内でＳＴＲ命令を１つ以上実行した場合、一般記録領域にデータ書込の情報が存在することになる。ＣＰＵ１１１のデータ書込の情報が存在する場合はステップＳ１６６に処理が進み、存在しない場合はステップＳ１６４に処理が進む。

【０１４１】

（Ｓ１６４）検出部１３９は、ＣＰＵ１１１の収集領域にある情報を破棄する。

（Ｓ１６５）検出部１３９は、管理情報１４１の中のＣＰＵ１１１に対応する依頼フラグをＯＦＦからＯＮに変更する。これにより、他ＣＰＵからの情報収集が再開される。

【０１４２】

図２５は、排他制御不備検出の手順例を示すフローチャート（続き）である。

（Ｓ１６６）検出部１３９は、ＣＰＵ１１２～１１４の１つ（１つの他ＣＰＵ）を特定する。検出部１３９は、ＣＰＵ１１１が特定した他ＣＰＵから情報を収集するための収集領域の中に、情報が存在するか判断する。特定した他ＣＰＵが直近の１区間内でＳＴＲ命令を１つ以上実行した場合、当該収集領域にデータ書込の情報が存在することになる。特定した他ＣＰＵのデータ書込の情報が存在する場合はステップＳ１６７に処理が進み、存在しない場合はステップＳ１７２に処理が進む。

【０１４３】

（Ｓ１６７）検出部１３９は、ＣＰＵ１１１に対応する一般記録領域の中から、使用されている集計領域（使用フラグがＯＮであるもの）を１つ特定する。検出部１３９は、特定した他ＣＰＵに対応する収集領域の中から、ＣＰＵ１１１の集計領域と担当するアドレス範囲が一致する他ＣＰＵの集計領域を検索する。該当する他ＣＰＵの集計領域がある場合はステップＳ１６８に処理が進み、ない場合はステップＳ１７１に処理が進む。

【０１４４】

（Ｓ１６８）検出部１３９は、特定したＣＰＵ１１１の集計領域に含まれる書込ビットマップと、これに対応する他ＣＰＵの集計領域に含まれる書込ビットマップとを用いて、データ書込の衝突を判定する。書込衝突判定の詳細は後述する。

【０１４５】

（Ｓ１６９）検出部１３９は、ステップＳ１６８でデータ書込の衝突が検出されたか判断する。データ書込の衝突が検出された場合、ステップＳ１７０に処理が進む。データ書込の衝突が検出されなかった場合、ステップＳ１７１に処理が進む。

【０１４６】

（Ｓ１７０）検出部１３９は、衝突したデータ書込についての情報を含むログ情報を生成し、ログ記録領域１５７に格納する。具体的には、検出部１３９は、特定したＣＰＵ１１１の集計領域と関連付けられている複数の追記領域の中から、データ書込先のアドレスに基づいて、衝突したデータ書込に対応する追記領域を探す。また、検出部１３９は、ステップＳ１６７で検索された他ＣＰＵの集計領域と関連付けられている複数の追記領域の中から、データ書込先のアドレスに基づいて、衝突したデータ書込に対応する追記領域を探す。そして、検出部１３９は、検索されたＣＰＵ１１１の追記領域と他ＣＰＵの追記領域からプログラム情報などを抽出し、ログ情報に挿入する。

【０１４７】

（Ｓ１７１）検出部１３９は、使用されているＣＰＵ１１１の集計領域を全て確認したか判断する。全ての集計領域を確認し終えた場合はステップＳ１７２に処理が進み、未確認の集計領域がある場合はステップＳ１６７に処理が進む。

【０１４８】

（Ｓ１７２）検出部１３９は、全ての他ＣＰＵ（ＣＰＵ１１２～１１４）を確認したか判断する。全ての他ＣＰＵを確認し終えた場合はステップＳ１７３に処理が進み、未確認の他ＣＰＵがある場合はステップＳ１６６に処理が進む。

【０１４９】

（Ｓ１７３）検出部１３９は、ログ記録領域１５７に格納されているログ情報を、ログファイル記憶部１３４に記憶されるログファイルに書き出す。このとき、検出部１３９は、ログファイルに書き出したログ情報をログ記録領域１５７から削除してよい。

【０１５０】

（Ｓ１７４）検出部１３９は、ＣＰＵ１１１に対応する一般記録領域、排他記録領域および作業領域にある情報を破棄する。そして、ステップＳ１６４に処理が進む。

図２６は、書込衝突判定の手順例を示すフローチャートである。

【０１５１】

この書込衝突判定は、前述のステップＳ１７０において実行される。

（Ｓ１８０）検出部１３９は、ＣＰＵ１１１の一般記録領域にある書込ビットマップ（一般用ビットマップ）と他ＣＰＵの書込ビットマップとの論理積（ＡＮＤ）を求める。

【０１５２】

（Ｓ１８１）検出部１３９は、ステップＳ１８０で算出された論理積の全ビットが０であるか判断する。論理積の全ビットが０である場合はステップＳ１８４に処理が進み、少なくとも１つのビットが１である場合はステップＳ１８２に処理が進む。

【０１５３】

（Ｓ１８２）検出部１３９は、ＣＰＵ１１１に対応する排他記録領域の中から、上記の一般用ビットマップと担当するアドレス範囲が一致する集計領域の書込ビットマップ（排他用ビットマップ）を探す。そして、検出部１３９は、ステップＳ１８０で算出された論理積としてのビットマップと排他用ビットマップとの排他的論理和（ＸＯＲ）を求める。

【０１５４】

（Ｓ１８３）検出部１３９は、ステップＳ１８２で算出された排他的論理和の全ビットが０であるか判断する。排他的論理和の全ビットが０である場合はステップＳ１８４に処理が進み、少なくとも１つのビットが１である場合はステップＳ１９０に処理が進む。

【０１５５】

（Ｓ１８４）検出部１３９は、一般用ビットマップの中で値が１になっているビットの周辺ビット（例えば、前後所定数のビット）を１に変更する。

（Ｓ１８５）検出部１３９は、ステップＳ１８４で修正した一般用ビットマップと他ＣＰＵの書込ビットマップとの論理積（ＡＮＤ）を求める。

【０１５６】

（Ｓ１８６）検出部１３９は、ステップＳ１８５で算出された論理積の全ビットが０であるか判断する。論理積の全ビットが０である場合はステップＳ１８９に処理が進み、少なくとも１つのビットが１である場合はステップＳ１８７に処理が進む。

【０１５７】
（Ｓ１８７）検出部１３９は、ステップＳ１８５で算出された論理積としてのビットマップと排他用ビットマップとの排他的論理和（ＸＯＲ）を求める。
（Ｓ１８８）検出部１３９は、ステップＳ１８７で算出された排他的論理和の全ビットが０であるか判断する。排他的論理和の全ビットが０である場合はステップＳ１８９に処理が進み、少なくとも１つのビットが１である場合はステップＳ１９０に処理が進む。
【０１５８】
（Ｓ１８９）検出部１３９は、データ書込の衝突がないと判定する。
（Ｓ１９０）検出部１３９は、データ書込の衝突があると判定する。
このようにして、検出部１３９は、ＣＰＵ１１１と他ＣＰＵの両方から同じブロックに対してデータ書込が行われたか判断する。同じブロックに対してデータ書込が行われた場合、検出部１３９は、原則として実行されたプログラムに排他制御の不備があると判定する。ただし、データ書込先がロックワードの記憶されているブロックである場合、そのデータ書込は正常な排他制御に基づくものであるため、排他制御の不備としては扱わない。
【０１５９】
上記で排他制御の不備があると判定しなかった場合、検出部１３９は次に、ＣＰＵ１１１と他ＣＰＵから、アドレスの近いブロックに対してデータ書込が行われたか判断する。近いブロックに対してデータ書込が行われた場合、検出部１３９は、原則として実行されたプログラムに排他制御の不備があると判定する。ただし、他ＣＰＵのデータ書込先がロックワードの記憶されているブロックである場合、そのデータ書込は正常な排他制御に基づくものであるため、排他制御の不備としては扱わない。
【０１６０】
以上、第２の実施の形態によれば、ＣＰＵ１１１～１１４それぞれが命令エミュレータを用いて、自ＣＰＵで行われたデータ書込および他ＣＰＵで行われたデータ書込についてのアドレス情報やプログラム情報を収集する。そして、ＣＰＵ１１１～１１４それぞれが命令エミュレータを用いて、直近の１区間で収集されたアドレス情報に基づいて排他制御の不備を判定し、プログラム情報などを含むログ情報を出力する。これにより、特殊なハードウェアやシミュレーション環境を用意しなくても、ＲＡＭ１１５へのアクセスの排他制御に不備があることを効率的に検出することができる。特に、第２の実施の形態では、スピンロック方式の排他制御について、その不備を効率的に検出できる。
【０１６１】
また、ユーザは、ログ情報に含まれるプログラム情報に基づいて、排他制御の不備のあるプログラムを特定することができ、デバッグ作業が容易となる。また、ロック獲得やロック解放を契機として情報収集区間を区切ることで、正常な排他制御によるアクセスを排他制御の不備と誤判定するのを抑制できる。また、ロックワードへのアクセスを除外してアドレス情報を比較することで、排他制御の不備の判定精度を向上させることができる。また、実行するプログラムの切替を形式として情報収集区間を区切ることで、排他制御の不備があるプログラムの判定精度を向上させることができる。
【０１６２】
［第３の実施の形態］
次に、第３の実施の形態を説明する。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。第３の実施の形態の情報処理装置は、排他制御の不備によって、あるＣＰＵからＲＡＭへのデータ参照と他の１以上のＣＰＵからＲＡＭへのデータ書込とが衝突したことを検出する。
【０１６３】
第３の実施の形態の情報処理装置は、図２に示した第２の実施の形態と同様のハードウェア構成によって実現できる。また、第３の実施の形態の情報処理装置は、図７に示した第２の実施の形態と同様のソフトウェア構成によって実現できる。以下、図２，７で用いたものと同様の符号を用いて、第３の実施の形態の情報処理装置を説明する。
【０１６４】

まず、排他制御の不備によってデータ参照とデータ書込が衝突する例およびＣＰＵ１１１～１１４がこのような排他制御の不備を判定する流れについて説明する。
図２７は、データ参照時の排他制御不備の例を示す図である。
【０１６５】
互いに異なるＣＰＵで実行されるプログラムＥ，Ｆを考える。例えば、プログラムＥはＣＰＵ１１１で実行され、プログラムＦはＣＰＵ１１２で実行される。
プログラムＥは、ＲＡＭ１１５に対するデータ参照を伴う命令として、複数のブロックに跨がるデータ同士を比較する比較命令（ＣＭＰ命令）を含む。プログラムＥは、このＣＭＰ命令により、ブロック＃１，＃２に跨がって格納されているデータＸとブロック＃３，＃４に跨がって格納されているデータＹとを比較する。前述のように、ブロックはＲＡＭ１１５へのアクセスの最小単位である。そこで、プログラムＥは、ブロック＃１のデータとブロック＃３のデータを読み込んで比較する（ＳＴ３０）。また、プログラムＥは、ブロック＃２のデータとブロック＃４のデータを読み込んで比較する（ＳＴ３２）。
【０１６６】
一方、プログラムＦは、ブロック＃３にデータを書き込むストア命令（ＳＴＲ命令）を含む。ここで、プログラムＥとプログラムＦの間で排他制御が正常に行われないとする。その場合、プログラムＥのＣＭＰ命令が実行されている途中で、プログラムＦのＳＴＲ命令が実行される可能性がある。例えば、プログラムＥがブロック＃３のデータを参照した後、ＣＭＰ命令が完了する前に、プログラムＦがブロック＃３のデータを更新する（ＳＴ３１）。その結果、ＣＭＰ命令が完了した時点のデータＸ，Ｙの同一性とＣＭＰ命令の結果とが整合しない場合があるなど、ＣＭＰ命令の結果の正しさが保証されなくなる。
【０１６７】
なお、図２７の例では、比較されるデータＸ，Ｙの両方がＲＡＭ１１５に格納されているとしたが、データＸ，Ｙの一方がＣＰＵ１１１のレジスタに格納されていてもよい。例えば、プログラムＥは、データＸをレジスタにロードしておき、ＣＭＰ命令を実行するときにブロック＃３，＃４に格納されているデータＹを参照してもよい。
【０１６８】
図２８は、データ参照時の排他制御不備の検出例を示すシーケンス図である。
ここでは、ＣＰＵ１１１が、ＣＰＵ１１１によるデータ参照とＣＰＵ１１２によるデータ書込とが衝突していないか確認する場合を考える。ＣＰＵ１１１は、ＣＰＵ１１２と同様にＣＰＵ１１３，１１４によるデータ書込についても衝突の有無を確認する。また、ＣＰＵ１１２～１１４も、ＣＰＵ１１１と同様に衝突の有無を確認する。
【０１６９】
ＲＡＭ１１５には、ＣＰＵ１１１～１１４それぞれに対応する制御領域が用意される。ＣＰＵ１１１用の制御領域には、ＣＰＵ１１１が参照するＲＡＭ１１５の領域を示すアドレス情報が格納される。また、ＣＰＵ１１１用の制御領域には、データ書込の監視をＣＰＵ１１２～１１４に依頼するか否かを示す依頼フラグや、ＣＰＵ１１１が参照する領域へのデータ書込の発生（データ参照とデータ書込の衝突）を示す衝突情報が格納される。ＣＰＵ１１２～１１４用の制御領域にも、同様の情報が格納される。
【０１７０】
命令エミュレータを用いてＣＰＵ１１１でＣＭＰ命令が開始されるとき、ＣＰＵ１１１は、比較されるデータが格納されている１つまたは２つの領域を示すアドレス情報を生成して、ＣＰＵ１１１用の制御領域に格納する。そして、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に格納された依頼フラグをＯＮにする（ＳＴ４０）。
【０１７１】
一方、命令エミュレータを用いてＣＰＵ１１２でＳＴＲ命令が実行されるとき、ＣＰＵ１１２は、ＣＰＵ１１１の依頼フラグを確認し、依頼フラグがＯＮであればＣＰＵ１１１用の制御領域からアドレス情報を読み込む。そして、ＣＰＵ１１２は、ＣＰＵ１１２によるデータ書込がＣＰＵ１１１によるデータ参照と衝突するか、すなわち、データ書込先がアドレス情報の示す領域に含まれるか判断する（ＳＴ４１）。データ参照とデータ書込の

衝突が検出された場合、ＣＰＵ１１２は、衝突発生を示す衝突情報を生成してＣＰＵ１１１用の制御領域に格納する。図２８の排他制御不備なしの例では、この時点で衝突は検出されない。

【０１７２】

ＣＭＰ命令が完了すると、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に格納された依頼フラグをＯＦＦにし、ＣＰＵ１１２からの衝突情報の収集を停止する。そして、ＣＰＵ１１１は、ＣＰＵ１１１用の制御領域に衝突情報があるか確認する（ＳＴ４２）。衝突情報がある場合、ＣＰＵ１１１は、ＣＰＵ１１１で実行中のプログラムおよびＣＰＵ１１２で実行中のプログラムの少なくとも一方に排他制御の不備があると判定する。一方、衝突情報がない場合、ＣＰＵ１１１は、ＣＭＰ命令が実行された区間について排他制御の不備がないと判定する。図２８の排他制御不備なしの例では、排他制御の不備がないと判定されている。

【０１７３】

その後、ステップＳＴ４０～ＳＴ４２と同様にして、ＣＰＵ１１１は、ＣＭＰ命令が開始されるとき、アドレス情報を生成してＣＰＵ１１１用の制御領域に格納し、依頼フラグをＯＮにする（ＳＴ４３）。ＣＰＵ１１２は、ＳＴＲが実行されるとき、依頼フラグがＯＮであることを確認し、データ書込先がアドレス情報の示す領域に含まれるか判断する（ＳＴ４４）。図２８の排他制御不備ありの例では、この時点で衝突が検出されている。そこで、ＣＰＵ１１２は、衝突情報を生成してＣＰＵ１１１用の制御領域に格納する（ＳＴ４５）。

【０１７４】

ＣＭＰ命令が完了すると、ＣＰＵ１１１は、依頼フラグをＯＦＦにし、ＣＰＵ１１１用の制御領域に衝突情報があるか確認する（ＳＴ４６）。ＣＰＵ１１１は、衝突情報を検出すると、ＣＰＵ１１１で実行中のプログラムおよびＣＰＵ１１２で実行中のプログラムの少なくとも一方に排他制御の不備があると判定する。そして、ＣＰＵ１１１は、プログラム情報などを含むログ情報を生成し、ログファイルとしてＨＤＤ１２１に出力する。

【０１７５】

図２９は、第３の実施の形態の制御情報の全体構造例を示す図である。

制御情報記憶部１３３には、図２９に示すような構造をもつ制御情報が格納される。制御情報は、自ＣＰＵ情報を管理するために、管理情報１６１、ポインタテーブル１６２（第３のポインタテーブル）、収集依頼情報１６３を含む複数の収集依頼情報、および、比較記録領域１６４を含む複数の比較記録領域を有する。また、制御情報は、他ＣＰＵ情報を管理するために、ポインタテーブル１７１を含む複数の第４のポインタテーブル、および、収集領域１７２を含む複数の収集領域を有する。また、制御情報は、ログ情報を管理するために、ログ管理情報１７４およびログ記録領域１７５を有する。

【０１７６】

管理情報１６１は、情報処理装置１００全体の状態を示す情報を含み、ポインタテーブル１６２を指し示す。ポインタテーブル１６２は、ＣＰＵ１１１～１１４の依頼フラグを含み、ＣＰＵ１１１～１１４それぞれに対応する収集依頼情報、比較記録領域および第４のポインタテーブルを指し示す。収集依頼情報には、ＣＭＰ命令によって比較されるデータが格納されている領域を示すアドレスが格納される。比較記録領域には、「自ＣＰＵ」で実行されるＣＭＰ命令に関するアドレスやプログラム情報などが格納される。収集依頼情報および比較記録領域は、ＣＰＵ１１１～１１４に対応して設けられる。

【０１７７】

第４のポインタテーブルは、ＣＰＵ１１１～１１４に対応して設けられる。１つの第４のポインタテーブルは、１つの「自ＣＰＵ」を示す。第４のポインタテーブルは、複数の「他ＣＰＵ」に対応する収集領域を指し示す。収集領域は、「自ＣＰＵ」と「他ＣＰＵ」の組み合わせ毎に設けられる。収集領域には、他ＣＰＵから収集された衝突情報が格納される。衝突情報は、ＳＴＲ命令によってデータ書込が行われた領域を示すアドレス情報やプログラム情報を含む。ただし、衝突が発生していないとき衝突情報は格納されない。

【０１７８】
ログ管理情報１７４は、ログ記録領域１７５を管理するための情報を含む。ログ記録領域１７５には、排他制御の不備があると判定されたときに、収集された情報に基づいて生成されたログ情報が格納される。ログ記録領域１７５に格納されたログ情報は、ログファイル記憶部１３４に記憶された（ＨＤＤ１２１上の）ログファイルに書き出される。
【０１７９】
図３０は、第３の実施の形態の管理情報のデータ構造例を示す図である。
管理情報１６１は、システム状態、ロックワード、対象プログラムおよびポインタテーブルのアドレスの項目を有する。システム状態の項目は、デバッグ開始フラグをもつ。デバッグ開始フラグは、デバッグ支援部１３６が動作中であるか否かを示す。デバッグ開始フラグは、１ビットで表現できる。ロックワードの項目は、管理情報１６１へのアクセスの排他制御に用いられるロックワードを含む。対象プログラムの項目は、デバッグ対象のプログラムの識別情報をもつ。ポインタテーブルのアドレスの項目は、ポインタテーブル１６２が存在するＲＡＭ１１５の領域の先頭アドレスである。
【０１８０】
図３１は、第３のポインタテーブルのデータ構造例を示す図である。
ポインタテーブル１６２は、ＣＰＵ１１１～１１４に対応して、依頼状態、収集依頼情報のアドレス、比較記録領域のアドレスおよび他ＣＰＵ用ポインタテーブルのアドレスの項目を有する。依頼状態の項目は、依頼フラグとしてデータ１依頼フラグおよびデータ２依頼フラグをもつ。データ１依頼フラグは、後述する収集依頼情報が示す１番目のアドレス範囲について、データ書込の監視を他ＣＰＵへ依頼しているか否かを示す。データ２依頼フラグは、後述する収集依頼情報が示す２番目のアドレス範囲について、データ書込の監視を他ＣＰＵへ依頼しているか否かを示す。各依頼フラグは、１ビットで表現できる。例えば、依頼フラグ＝１（ＯＮ）は、監視を依頼中であることを示す。依頼フラグ＝０（ＯＦＦ）は、監視を依頼していないことを示す。
【０１８１】
収集依頼情報のアドレスの項目は、「自ＣＰＵ」に対応する収集依頼情報が存在するＲＡＭ１１５の領域の先頭アドレスである。比較記録領域のアドレスの項目は、「自ＣＰＵ」に対応する比較記録領域が存在するＲＡＭ１１５の領域の先頭アドレスである。他ＣＰＵ用ポインタテーブルのアドレスの項目は、「自ＣＰＵ」に対応する第４のポインタテーブルが存在するＲＡＭ１１５の領域の先頭アドレスである
図３２は、収集依頼情報のデータ構造例を示す図である。
【０１８２】
収集依頼情報１６３は、比較データ１の先頭メモリアドレスおよび末尾メモリアドレスと、比較データ２の先頭メモリアドレスおよび末尾メモリアドレスの項目を有する。
比較データ１の先頭メモリアドレスおよび末尾メモリアドレスは、ＣＭＰ命令の１番目の引数に相当するデータが格納されているＲＡＭ１１５の領域の先頭アドレスおよび末尾アドレスである。なお、ＣＭＰ命令の１番目の引数がレジスタを参照している場合（ＣＭＰ命令を実行するときＲＡＭ１１５上にある比較データ１を参照しなくてよい場合）は、比較データ１の先頭メモリアドレスおよび末尾メモリアドレスの項目は空でよい。
【０１８３】
比較データ２の先頭メモリアドレスおよび末尾メモリアドレスは、ＣＭＰ命令の２番目の引数に相当するデータが格納されているＲＡＭ１１５の領域の先頭アドレスおよび末尾アドレスである。なお、ＣＭＰ命令の２番目の引数がレジスタを参照している場合（ＣＭＰ命令を実行するときＲＡＭ１１５上にある比較データ２を参照しなくてよい場合）は、比較データ２の先頭メモリアドレスおよび末尾メモリアドレスの項目は空でよい。比較データ１，２の少なくとも一方は、ＲＡＭ１１５にあるものがアクセスされるとする。
【０１８４】
図３３は、比較命令情報のデータ構造例を示す図である。
比較記録領域１６４は、図３３に示すような比較命令情報１６５を記憶する。比較命令

情報１６５は、領域状態、プログラム情報および比較時刻の項目を有する。また、比較命令情報１６５は、比較データ１について、参照されたメモリアドレス、ベースアドレスおよびサイズの項目を有する。同様に、比較命令情報１６５は、比較データ２について、参照されたメモリアドレス、ベースアドレスおよびサイズの項目を有する。

【０１８５】

領域状態の項目は、データ１フラグおよびデータ２フラグを含む。データ１フラグは、比較命令情報１６５の比較データ１の項目が使用されているか否かを示す。データ２フラグは、比較命令情報１６５の比較データ２の項目が使用されているか否かを示す。データ１フラグおよびデータ２フラグは、それぞれ１ビットで表現できる。例えば、データ１フラグ＝１（ＯＮ）は、比較データ１の項目が使用されている（有効なデータが記憶されている）ことを示す。データ１フラグ＝０（ＯＦＦ）は、比較データ１の項目が使用されていない（有効なデータがない）ことを示す。

【０１８６】

比較データ１について、参照されたメモリアドレスの項目は、比較データ１が記憶されているＲＡＭ１１５の領域の先頭アドレスである。ベースアドレスの項目は、ＣＭＰ命令が開始された時点のベースレジスタの値であって、比較データ１が記憶された領域を特定するための起点となるアドレスである。サイズの項目は、参照する比較データ１の大きさを示す情報（例えば、バイト数）を含む。比較データ２について、参照されたメモリアドレス、ベースアドレスおよびサイズの項目は、比較データ１と同様の情報を含む。

【０１８７】

プログラム情報の項目は、「自ＣＰＵ」で実行されたＣＭＰ命令を含むプログラムを特定するための情報である。プログラム情報としては、例えば、プログラム状態ワード（ＰＳＷ）を用いることができる。比較時刻の項目は、ＣＭＰ命令が実行された時刻を含む。

【０１８８】

図３４は、第４のポインタテーブルのデータ構造例を示す図である。

ポインタテーブル１７１は、「他ＣＰＵ」（例えば、ＣＰＵ１１２～１１４）に対応して、領域状態、ＣＰＵ番号、ロックワードおよび収集領域のアドレスの項目を有する。領域状態の項目は、使用フラグをもつ。使用フラグは、収集領域が使用されているか否か、すなわち、他ＣＰＵから収集された衝突情報が存在するか否かを示す。使用フラグは、１ビットで表現できる。ＣＰＵ番号の項目には、「他ＣＰＵ」を識別するための識別番号が格納される。ロックワードの項目は、ポインタテーブル１７１へのアクセスの排他制御に用いられる。収集領域のアドレスの項目は、ＲＡＭ１１５上の当該収集領域の先頭アドレスである。

【０１８９】

図３５は、衝突情報のデータ構造例を示す図である。

収集領域１７２は、図３５に示すような衝突情報１７３を１単位として、１単位または２単位以上の衝突情報を記憶する。２単位以上の衝突情報は、連結リストとして記憶される。衝突情報１７３は、次の衝突情報のアドレス、領域状態、書込が行われたメモリアドレス、ベースアドレス、プログラム情報および書込時刻の項目を有する。

【０１９０】

次の衝突情報のアドレスの項目は、連結リストにおける次の衝突情報が記憶されたＲＡＭ１１５の領域の先頭アドレスである。領域状態の項目は、末尾フラグおよび使用フラグを含む。末尾フラグは、衝突情報１７３が連結リストの末尾であるか否かを示す。使用フラグは、他ＣＰＵから情報が収集されたか否かを示す。末尾フラグおよび使用フラグは、それぞれ１ビットで表現できる。

【０１９１】

書込が行われたメモリアドレスの項目は、他ＣＰＵによってデータ書込が行われた領域の先頭アドレスを含む。ベースアドレスの項目は、データ書込が行われた時点の他ＣＰＵのベースアドレス、すなわち、他ＣＰＵがアクセスした領域を特定するための起点となるアドレスを含む。プログラム情報の項目は、データ書込の時点で実行されていたプログラ

ムを特定するための情報である。プログラム情報としては、例えば、ＰＳＷを用いることができる。書込時刻の項目は、データ書込が行われた時刻である。

【０１９２】

ログ管理情報１７４は、例えば、図１７に示した第１の実施の形態のログ管理情報１５６と同様のデータ構造によって実現することができる。

図３６は、第３の実施の形態のログ情報のデータ構造例を示す図である。

【０１９３】

ログ記録領域１７５は、図３６に示すようなログ情報１７６を１単位として、１単位または２単位以上のログ情報を記憶する。２単位以上のログ情報は、連結リストとして記憶される。ログ情報１７６は、次のログ情報のアドレス、領域状態およびログサイズの項目を有する。また、ログ情報１７６は、ＣＭＰ命令を実行したＣＰＵ（自ＣＰＵ）について、ＣＰＵ番号、比較時刻、比較データ１，２の参照されたメモリアドレス、比較データ１，２のベースアドレスおよびプログラム情報の項目を有する。また、ログ情報１７６は、データ書込を行ったＣＰＵ（他ＣＰＵ）について、他ＣＰＵ番号、書込時刻、書込が行われたメモリアドレス、ベースアドレスおよびプログラム情報の項目を有する。

【０１９４】

次のログ情報のアドレスの項目は、連結リストにおける次のログ情報が記憶されたＲＡＭ１１５の領域の先頭アドレスである。領域状態の項目は、データ１フラグおよびデータ２フラグを含む。データ１フラグは、ログ情報１７６の比較データ１の項目が使用されているか否かを示す。データ２フラグは、ログ情報１７６の比較データ２の項目が使用されているか否かを示す。データ１フラグおよびデータ２フラグは、それぞれ１ビットで表現できる。ログサイズの項目は、ログ情報１７６の大きさを示す。

【０１９５】

ＣＰＵ番号の項目は、ＣＭＰ命令を実行したＣＰＵを識別するための識別番号である。ＣＭＰ命令を実行したＣＰＵの情報としては、例えば、比較記録領域１６４に記憶された比較命令情報１６５に含まれるものが用いられる。他ＣＰＵ番号の項目は、ＣＭＰ命令と衝突するストア命令を実行したＣＰＵを識別するための識別番号である。データ書込を行ったＣＰＵの情報としては、例えば、収集領域１７２に記憶された衝突情報１７３に含まれるものが用いられる。

【０１９６】

次に、第３の実施の形態で情報処理装置１００が実行する処理について説明する。起動処理の手順は、図１９に示した第２の実施の形態のものと同様であるため説明を省略する。以下では、「自ＣＰＵ」をＣＰＵ１１１として、デバッグ支援について説明する。

【０１９７】

図３７は、データ参照時のデバッグ支援の手順例を示すフローチャートである。

（Ｓ２１０）命令判別部１３１は、命令を１つ読み込む。

（Ｓ２１１）命令判別部１３１は、読み込まれた命令がＣＭＰ命令（比較命令）であるか判断する。読み込まれた命令がＣＭＰ命令である場合はステップＳ２１２に処理が進み、ＣＭＰ命令でない場合はステップＳ２１３に処理が進む。

【０１９８】

（Ｓ２１２）情報収集部１３７は、読み込まれたＣＭＰ命令の開始時に、ＣＭＰ命令によって参照されるＲＡＭ１１５の領域に対してデータ書込が行われていないか監視するよう他ＣＰＵに依頼する。また、情報収集部１３７は、ＣＭＰ命令に関するアドレス情報やプログラム情報を生成する。そして、ステップＳ２１５に処理が進む。この処理（衝突情報収集）の詳細は後述する。

【０１９９】

（Ｓ２１３）命令判別部１３１は、読み込まれた命令がＳＴＲ命令（ストア命令）であるか判断する。読み込まれた命令がＳＴＲ命令である場合はステップＳ２１４に処理が進み、ＳＴＲ命令でない場合は処理が終了する。なお、読み込まれた命令は、命令変換部１３２によってＣＰＵ１１１が解釈可能な命令に変換されて実行される。

【０２００】
（Ｓ２１４）情報提供部１３８は、ＣＰＵ１１２～１１４によるデータ参照とＣＰＵ１１１によるデータ書込とが衝突したか、すなわち、ＣＭＰ命令で比較されたデータが記憶されているＲＡＭ１１５の領域に対して、データ書込が行われるか判定する。データ参照とデータ書込の衝突を検出した場合、情報提供部１３８は、ＳＴＲ命令に関するアドレス情報やプログラム情報を含む衝突情報を生成する。そして、読み込まれた命令の処理が終了する。この処理（読み書き衝突判定）の詳細は後述する。
【０２０１】
（Ｓ２１５）検出部１３９は、ＣＭＰ命令が開始されてから完了するまでの間に、ＣＰＵ１１１に対応する収集領域に衝突情報が収集されたか確認する。衝突情報が収集されていることで、検出部１３９は、ＣＰＵ１１１によるデータ参照とＣＰＵ１１２～１１４の少なくとも１つのＣＰＵによるデータ書込とが衝突したことを検出する。検出部１３９は、衝突を検出すると、ＣＰＵ１１１で実行されたプログラムおよびデータ書込を行った他ＣＰＵで実行されたプログラムの少なくとも一方に排他制御の不備があると判定し、プログラム情報などを含むログ情報を出力する。排他制御不備検出の詳細は後述する。
【０２０２】
図３８は、衝突情報収集の手順例を示すフローチャートである。
この衝突情報収集は、前述のステップＳ２１２において実行される。
（Ｓ２２０）情報収集部１３７は、管理情報１６１を参照して、ＣＰＵ１１１で実行されているプログラムがデバッグ対象であるか判断する。プログラムがデバッグ対象である場合はステップＳ２２１に処理が進み、デバッグ対象でない場合は処理が終了する。
【０２０３】
（Ｓ２２１）情報収集部１３７は、実行しようとするＣＭＰ命令が所定の条件を満たす命令であるか判断する。第３の実施の形態では、所定の条件は、比較する２つのデータの少なくとも一方がＲＡＭ１１５へアクセスすることで参照され、かつ、ＲＡＭ１１５上の当該データが複数のブロックに跨がって記憶されていることである。ブロックは、１回にアクセスされるＲＡＭ１１５の領域の単位である。ＣＭＰ命令が所定の条件を満たす場合はステップＳ２２２に処理が進み、所定の条件を満たさない場合は処理が終了する。
【０２０４】
（Ｓ２２２）情報収集部１３７は、命令判別部１３１から受け取ったＣＭＰ命令に基づいて、参照されるＲＡＭ１１５の領域を示すアドレス範囲（絶対アドレス）を算出する。ここでは、ＣＭＰ命令の引数に応じて１つまたは２つのアドレス範囲が算出される。ＣＭＰ命令の２つの引数の一方がレジスタを指している場合はアドレス範囲が１つ算出され、２つの引数が共にＲＡＭ１１５を指している場合はアドレス範囲が２つ算出される。
【０２０５】
（Ｓ２２３）情報収集部１３７は、ＣＰＵ１１１に対応する収集依頼情報に、ステップＳ２２２で算出した１つまたは２つのアドレス範囲を示す情報を格納する。
（Ｓ２２４）情報収集部１３７は、ＣＰＵ１１１が備えるベースレジスタの現在値（現在のベースアドレス）や、ＣＰＵ１１１で実行されているプログラムを示すプログラム情報（例えば、現在のＰＳＷ）を収集する。そして、情報収集部１３７は、アドレス範囲を示す情報、ベースアドレス、プログラム情報などを含む比較命令情報を生成し、ＣＰＵ１１１に対応する比較記録領域に比較命令情報を格納する。
【０２０６】
（Ｓ２２５）情報収集部１３７は、ポインタテーブル１６２に含まれるＣＰＵ１１１に対応する依頼フラグをＯＮに変更する。このとき、ＣＭＰ命令の引数に応じて、データ１依頼フラグとデータ２依頼フラグが設定される。データ１依頼フラグとデータ２依頼フラグの少なくとも一方がＯＮになる。
【０２０７】
図３９は、読み書き衝突判定の手順例を示すフローチャートである。
この読み書き衝突判定は、前述のステップＳ２１４において実行される。

（Ｓ２３０）情報提供部１３８は、ＣＰＵ１１２～１１４の１つ（１つの他ＣＰＵ）を特定する。情報提供部１３８は、ポインタテーブル１６２を参照して、特定した他ＣＰＵに対応する２つの依頼フラグ（データ１依頼フラグおよびデータ２依頼フラグ）の少なくとも一方がＯＮであるか判断する。少なくとも一方の依頼フラグがＯＮであることは、当該他ＣＰＵからデータ書込の監視が依頼されていることを示す。少なくとも一方の依頼フラグがＯＮである場合はステップＳ２３１に処理が進み、２つの依頼フラグが共にＯＦＦである場合はステップＳ２３５に処理が進む。

【０２０８】

（Ｓ２３１）情報提供部１３８は、命令判別部１３１から受け取ったＳＴＲ命令に基づいて、データ書込が行われるＲＡＭ１１５のアドレス（絶対アドレス）を算出する。

（Ｓ２３２）情報提供部１３８は、特定した他ＣＰＵに対応する収集依頼情報を参照して、当該他ＣＰＵから指定された１つまたは２つのアドレス範囲（ＯＮの依頼フラグに対応するアドレス範囲）を確認する。そして、情報提供部１３８は、ステップＳ２３１で算出されたデータ書込先のアドレスが、指定された何れかのアドレス範囲に含まれるか判断する。データ書込先が指定されたアドレス範囲に属する場合はステップＳ２３３に処理が進み、属さない場合はステップＳ２３５に処理が進む。

【０２０９】

（Ｓ２３３）情報提供部１３８は、特定した他ＣＰＵに対応する複数の収集領域のうち、ＣＰＵ１１１の衝突情報を収集するための収集領域を検出する。

（Ｓ２３４）情報提供部１３８は、ＣＰＵ１１１のベースレジスタの値（ベースアドレス）やプログラム情報（例えば、ＰＳＷ）を収集し、データ書込先のアドレス、ベースアドレスおよびプログラム情報を含む衝突情報を生成する。そして、情報提供部１３８は、ステップＳ２３３で検出した収集領域に衝突情報を格納する。

【０２１０】

（Ｓ２３５）情報提供部１３８は、全ての他ＣＰＵ（ＣＰＵ１１２～１１４）を確認したか判断する。全ての他ＣＰＵを確認し終えた場合は処理が終了し、未確認の他ＣＰＵがある場合はステップＳ２３０に処理が進む。

【０２１１】

図４０は、データ参照時の排他制御不備検出の手順例を示すフローチャートである。

この排他制御不備検出は、前述のステップＳ２１５において実行される。

（Ｓ２４０）検出部１３９は、ＣＭＰ命令の完了を検出する。

【０２１２】

（Ｓ２４１）検出部１３９は、管理情報１６１を参照して、ＣＰＵ１１１で実行されているプログラムがデバッグ対象であるか判断する。プログラムがデバッグ対象である場合はステップＳ２４２に処理が進み、デバッグ対象でない場合は処理が終了する。

【０２１３】

（Ｓ２４２）検出部１３９は、ＣＭＰ命令が所定の条件を満たす命令であるか判断する。所定の条件は前述の図３９におけるステップＳ２２１と同様である。ＣＭＰ命令が所定の条件を満たす場合はステップＳ２４３に処理が進み、所定の条件を満たさない場合は処理が終了する。

【０２１４】

（Ｓ２４３）検出部１３９は、ポインタテーブル１６２に含まれるＣＰＵ１１１に対応するデータ１依頼フラグおよびデータ２依頼フラグをＯＦＦにする。

（Ｓ２４４）検出部１３９は、ＣＰＵ１１２～１１４の１つ（１つの他ＣＰＵ）を特定する。検出部１３９は、ＣＰＵ１１１と特定した他ＣＰＵの組に対応する収集領域に、当該他ＣＰＵの衝突情報が存在するか判断する。衝突情報がある場合はステップＳ２４５に処理が進み、ない場合はステップＳ２４６に処理が進む。

【０２１５】

（Ｓ２４５）検出部１３９は、衝突情報に含まれるアドレス情報やプログラム情報などを、ログ情報の一部としてログ記録領域１７５にコピーする。

（Ｓ２４６）検出部１３９は、全ての他ＣＰＵ（ＣＰＵ１１２～１１４）を確認したか判断する。全ての他ＣＰＵを確認し終えた場合はステップＳ２４７に処理が進み、未確認の他ＣＰＵがある場合はステップＳ２４４に処理が進む。

【０２１６】

（Ｓ２４７）検出部１３９は、少なくとも１つの他ＣＰＵで衝突情報が収集されていたか、すなわち、ステップＳ２４５が実行されたか判断する。衝突情報が収集された場合はステップＳ２４８に処理が進み、収集されていない場合は処理が終了する。

【０２１７】

（Ｓ２４８）検出部１３９は、ＣＰＵ１１１に対応する比較記録領域から比較命令情報を取得する。検出部１３９は、比較命令情報に含まれるアドレス情報やプログラム情報などを、ログ情報の一部としてログ記録領域１７５にコピーする。

【０２１８】

（Ｓ２４９）検出部１３９は、ログ記録領域１７５に格納されているログ情報を、ログファイル記憶部１３４に記憶されるログファイルに書き出す。このとき、検出部１３９は、ログファイルに書き出したログ情報をログ記録領域１７５から削除してよい。

【０２１９】

以上、第３の実施の形態によれば、ＣＰＵ１１１～１１４それぞれが命令エミュレータを用いて、自ＣＰＵで行われたデータ参照および他ＣＰＵで行われたデータ書込についてのアドレス情報やプログラム情報などを収集する。そして、ＣＰＵ１１１～１１４それぞれが命令エミュレータを用いて、データ参照とデータ書込の間の排他制御の不備を判定してログ情報を出力する。これにより、特殊なハードウェアやシミュレーション環境を用意しなくても、ＲＡＭ１１５へのアクセスの排他制御に不備があることを効率的に検出できる。特に、第３の実施の形態では、スピンロック方式の排他制御について、その不備を効率的に判定できる。また、ユーザは、ログ情報に含まれるプログラム情報に基づいて、排他制御の不備のあるプログラムを特定することができ、デバッグ作業が容易となる。

【０２２０】

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および第３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。

【０２２１】

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１０３）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、ＨＤＤ１２１）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。

【０２２２】

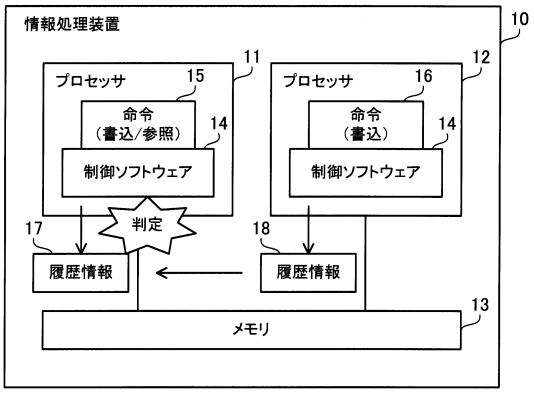
上記については単に本発明の原理を示すものである。更に、多数の変形や変更が当業者にとって可能であり、本発明は上記に示し、説明した正確な構成および応用例に限定されるものではなく、対応する全ての変形例および均等物は、添付の請求項およびその均等物による本発明の範囲とみなされる。

【符号の説明】

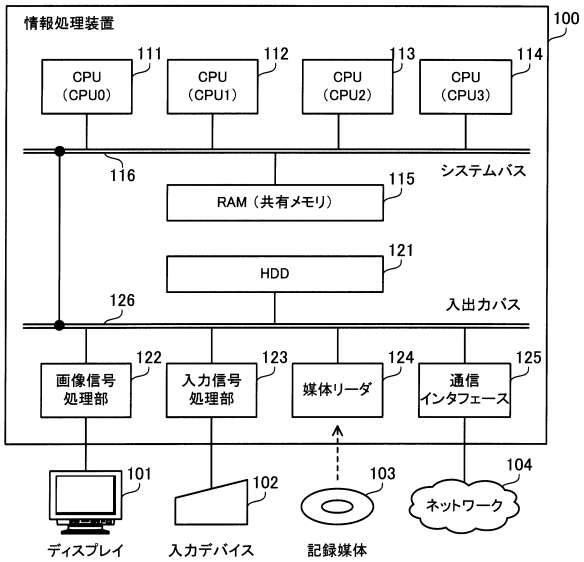
【０２２３】

１０　情報処理装置
１１，１２　プロセッサ
１３　メモリ
１４　制御ソフトウェア
１５，１６　命令
１７，１８　履歴情報

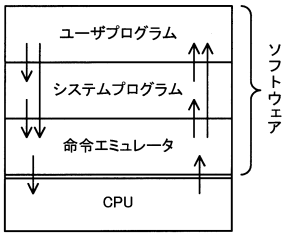
【図１】


情報処理装置
10
プロセッサ
11
15
命令
（書込/参照）
14
制御ソフトウェア
プロセッサ
12
16
命令
（書込）
14
制御ソフトウェア
判定
17
履歴情報
18
履歴情報
メモリ
13

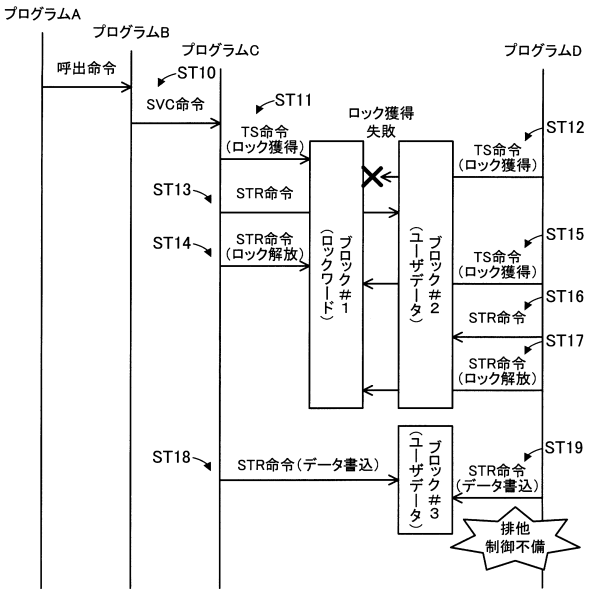
【図２】


情報処理装置
100
CPU
（CPU0）
111
CPU
（CPU1）
112
CPU
（CPU2）
113
CPU
（CPU3）
114
116
システムバス
RAM（共有メモリ）
115
HDD
121
126
入出力バス
画像信号
処理部
122
入力信号
処理部
123
媒体リーダ
124
通信
インタフェース
125
101
102
103
104
ネットワーク
ディスプレイ
入力デバイス
記録媒体

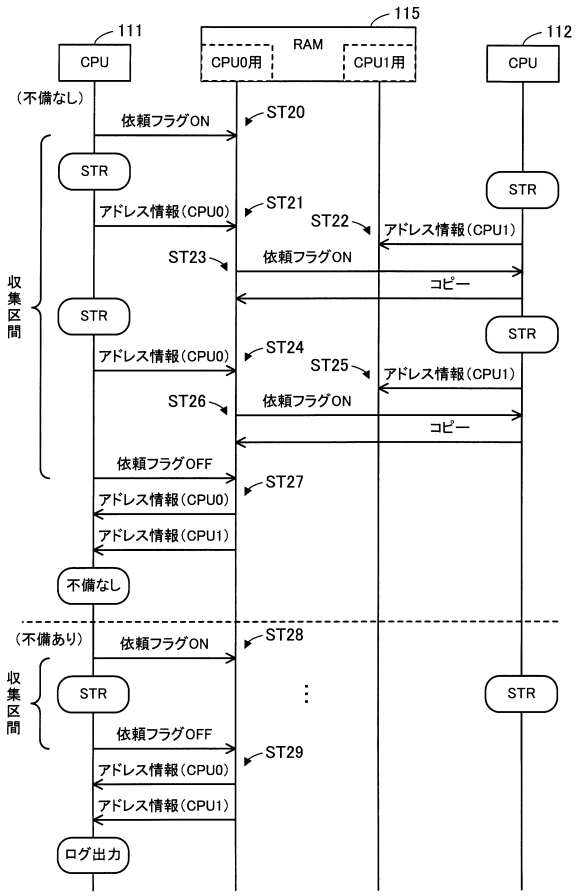
【図３】


ユーザプログラム
システムプログラム
命令エミュレータ
CPU
ソフトウェア

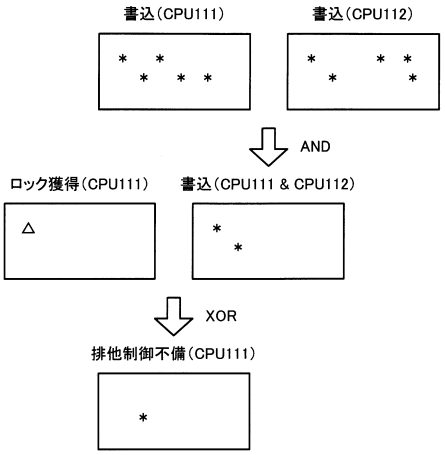
【図４】


プログラムA
プログラムB
プログラムC
プログラムD
呼出命令
ST10
SVC命令
ST11
TS命令
（ロック獲得）
ロック獲得
失敗
ST12
TS命令
（ロック獲得）
ST13
STR命令
ST14
STR命令
（ロック解放）
ブロック#1
（ロックワード）
ブロック#2
（ユーザデータ）
ST15
TS命令
（ロック獲得）
ST16
STR命令
ST17
STR命令
（ロック解放）
ST18
STR命令（データ書込）
ブロック#3
（ユーザデータ）
ST19
STR命令
（データ書込）
排他
制御不備

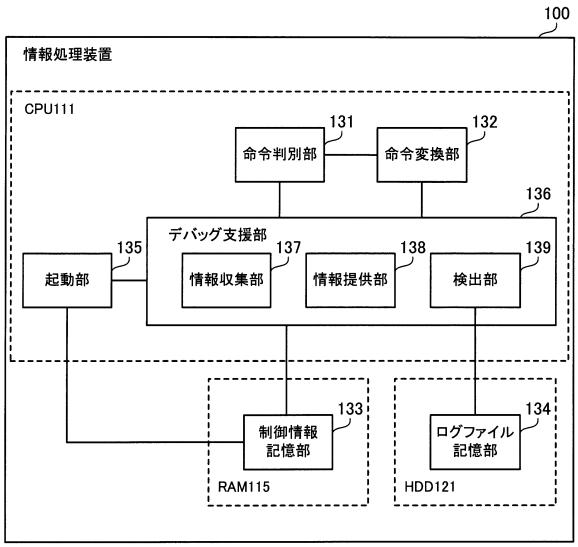
【図５】


115
111
CPU
RAM
CPU0用
CPU1用
112
CPU
（不備なし）
依頼フラグON
ST20
STR
STR
アドレス情報（CPU0）
ST21
ST22
アドレス情報（CPU1）
ST23
依頼フラグON
コピー
収集区間
STR
STR
アドレス情報（CPU0）
ST24
ST25
アドレス情報（CPU1）
ST26
依頼フラグON
コピー
依頼フラグOFF
ST27
アドレス情報（CPU0）
アドレス情報（CPU1）
不備なし
（不備あり）
依頼フラグON
ST28
収集区間
STR
STR
依頼フラグOFF
ST29
アドレス情報（CPU0）
アドレス情報（CPU1）
ログ出力

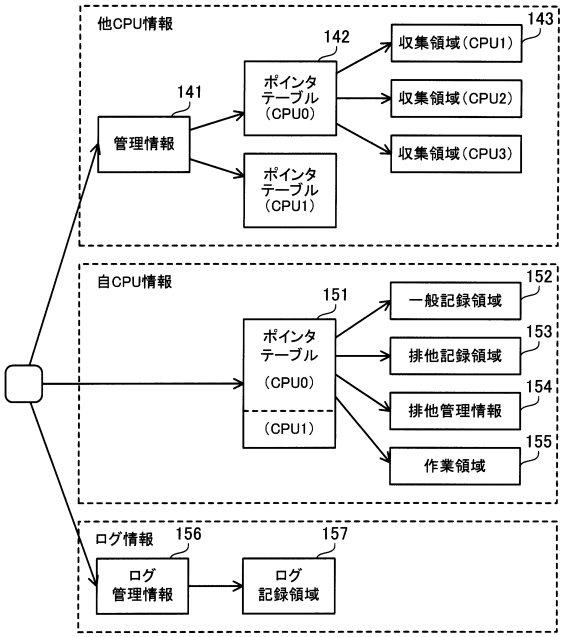
【図６】



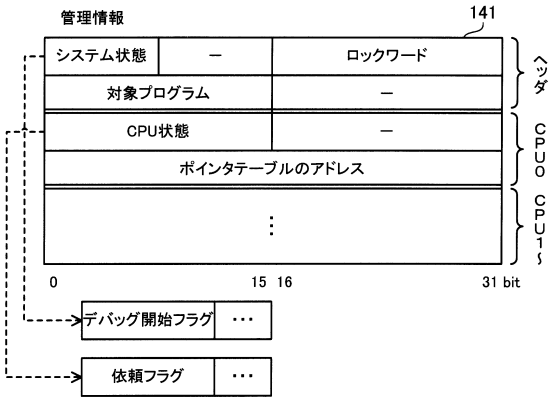
【図７】



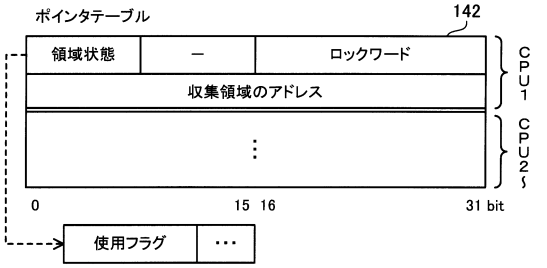
【図８】



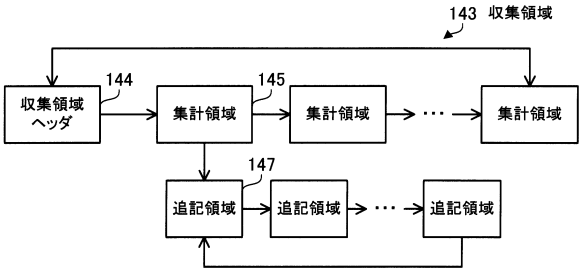
【図９】



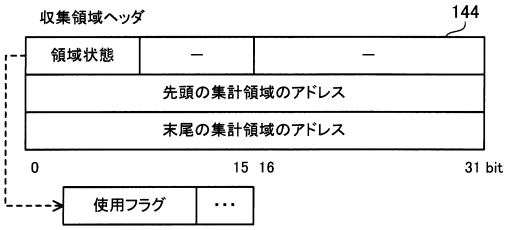
【図１０】



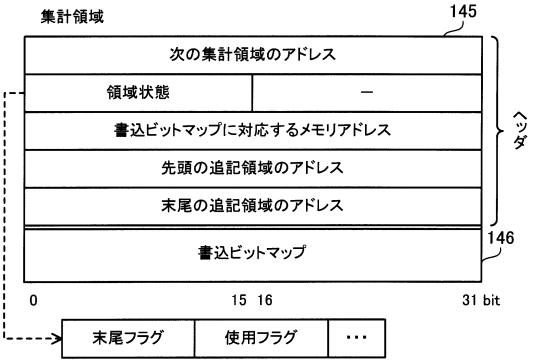
【図１１】



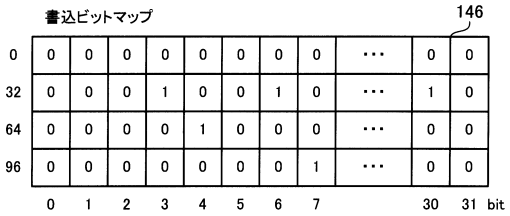
【図１２】



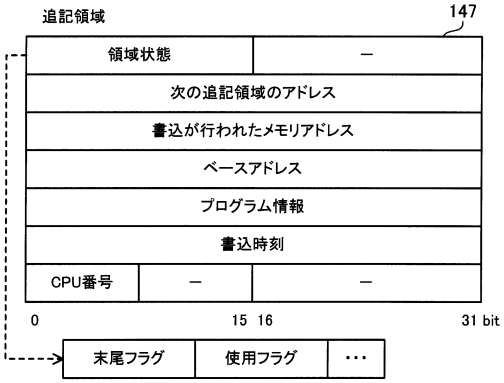
【図１３】



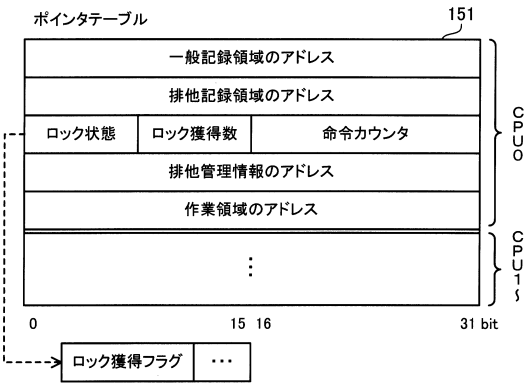
【図１４】



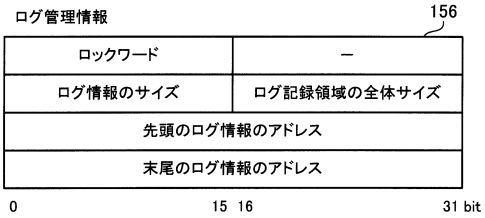
【図１５】



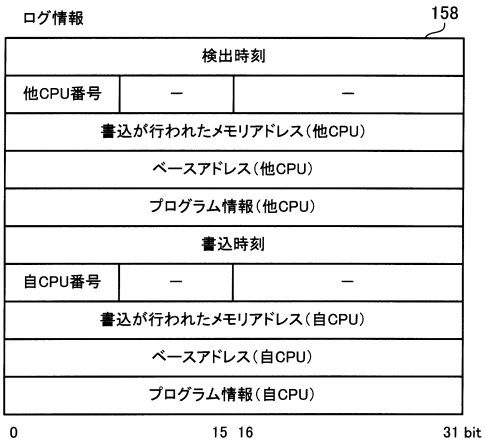
【図１６】



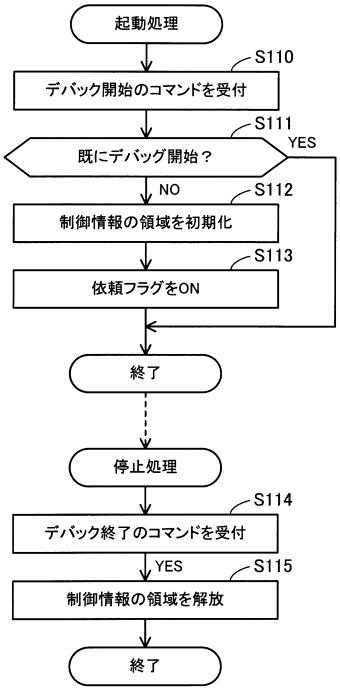
【図１７】



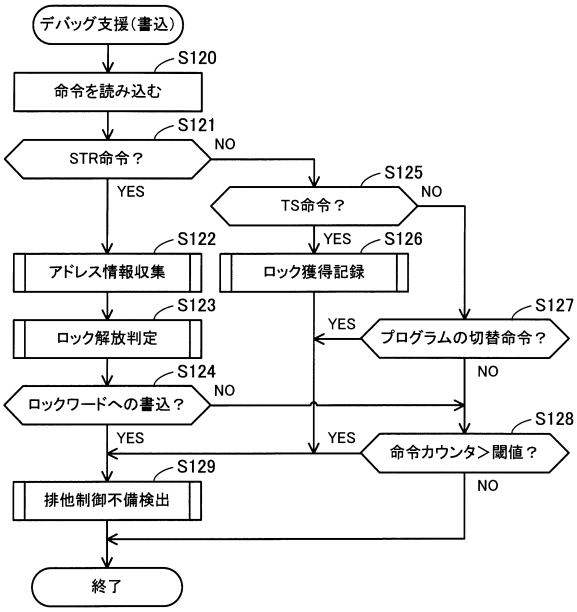
【図１８】



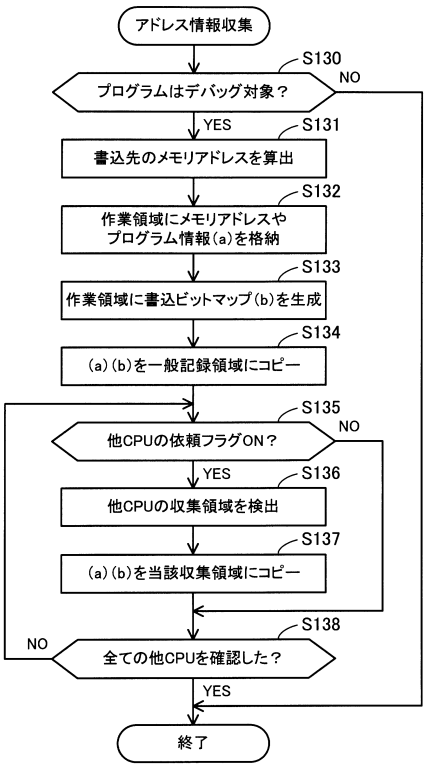
【図１９】



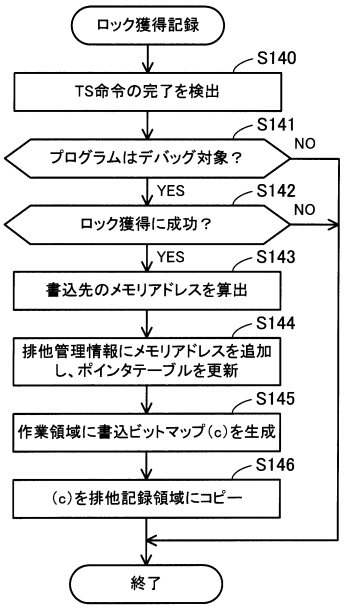
【図２０】



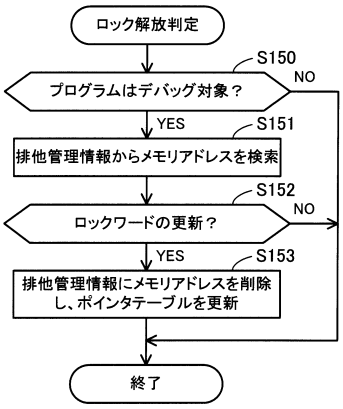
【図２１】



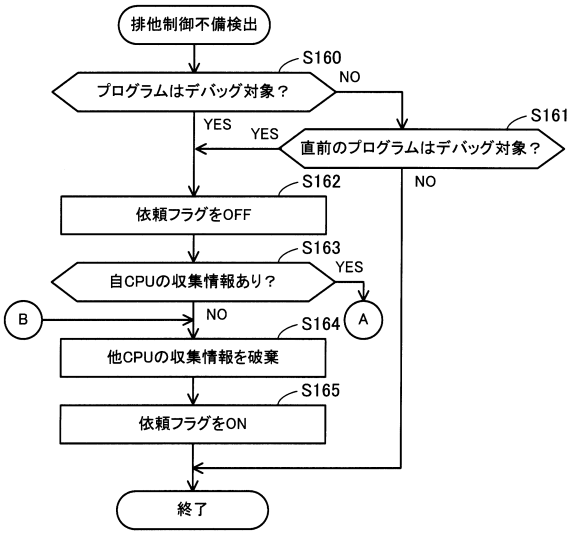
【図２２】



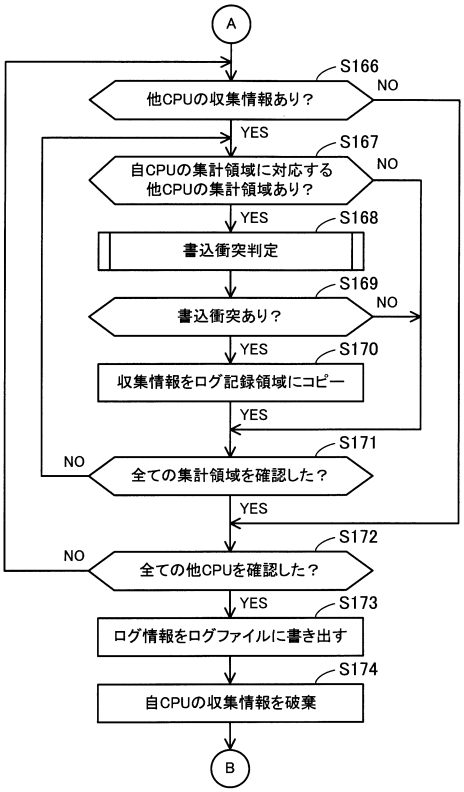
【図２３】



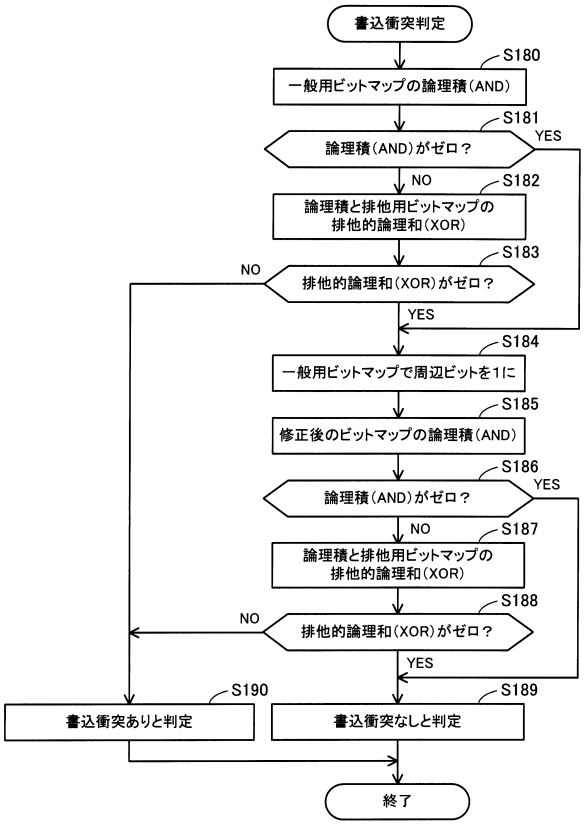
【図２４】



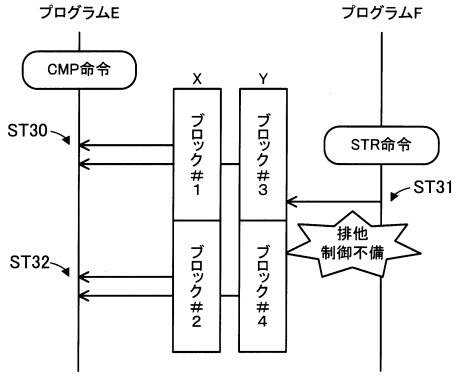
【図２５】



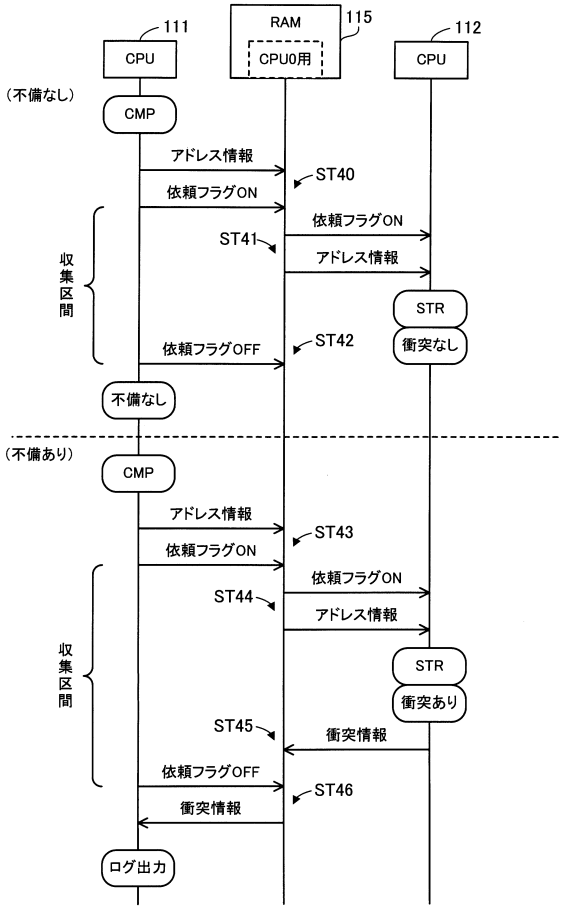
【図２６】



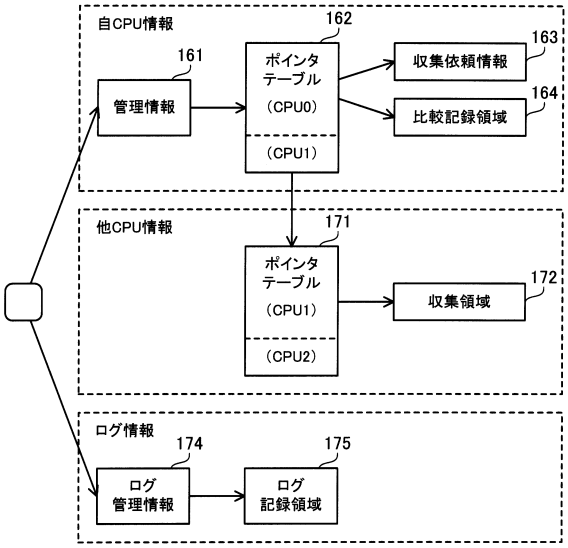
【図２７】



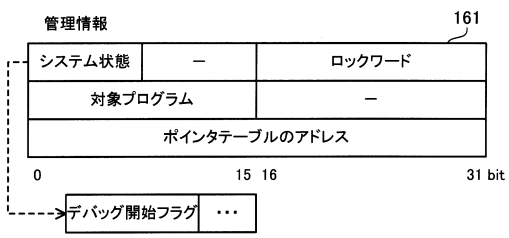
【図２８】



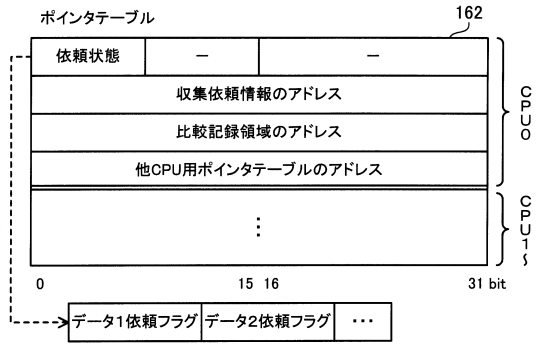
【図２９】



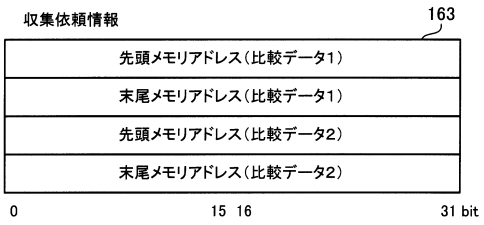
【図３０】



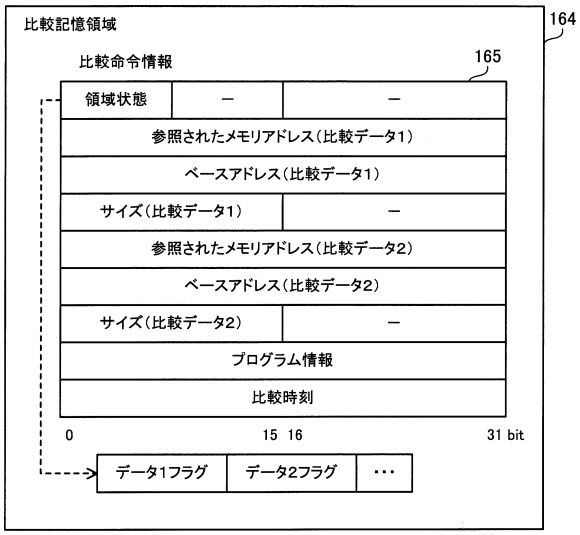
【図３１】



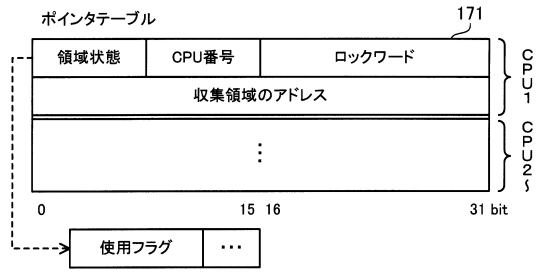
【図３２】



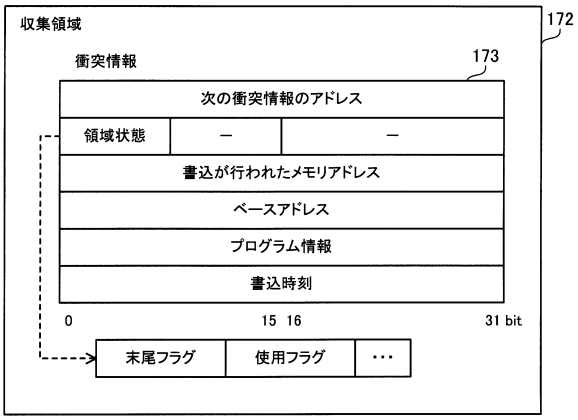
【図３３】



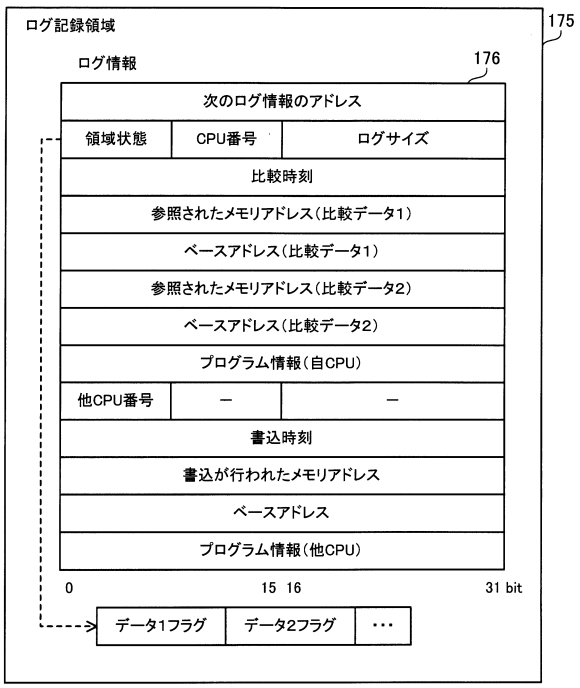
【図３４】



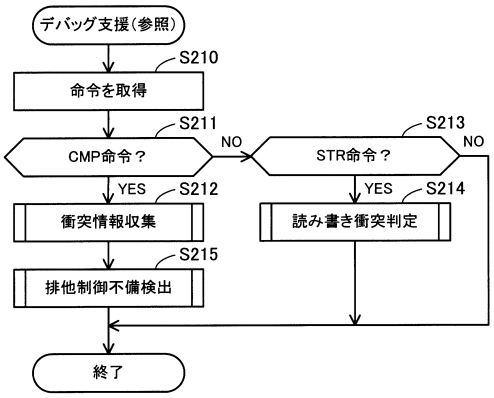
【図３５】



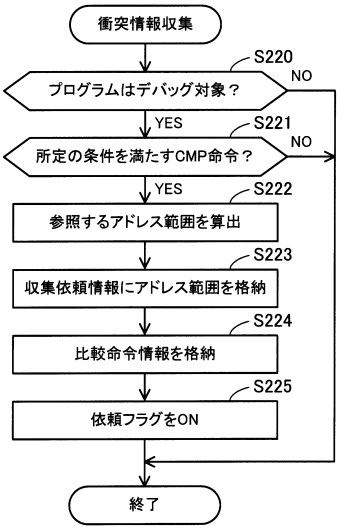
【図３６】



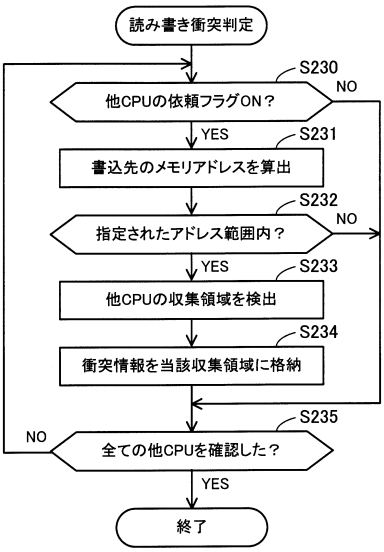
【図３７】



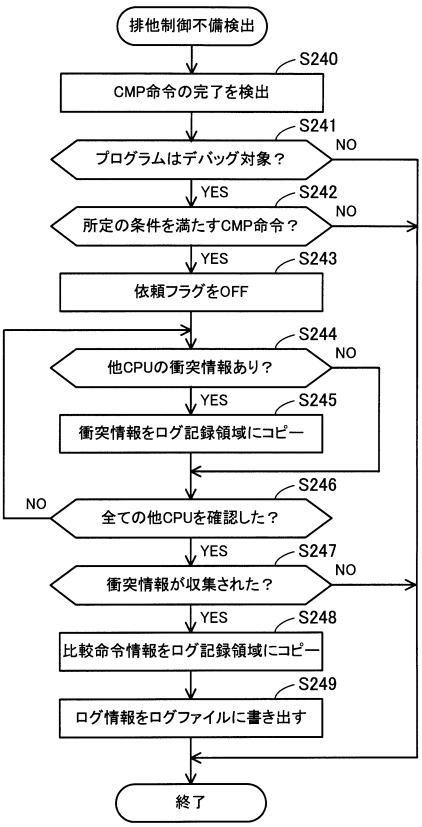
【図３８】


衝突情報収集
S220
プログラムはデバッグ対象？
NO
YES
S221
所定の条件を満たすCMP命令？
NO
YES
S222
参照するアドレス範囲を算出
S223
収集依頼情報にアドレス範囲を格納
S224
比較命令情報を格納
S225
依頼フラグをON
終了

【図３９】


読み書き衝突判定
S230
他CPUの依頼フラグON？
NO
YES
S231
書込先のメモリアドレスを算出
S232
指定されたアドレス範囲内？
NO
YES
S233
他CPUの収集領域を検出
S234
衝突情報を当該収集領域に格納
S235
NO
全ての他CPUを確認した？
YES
終了

【図４０】


排他制御不備検出
S240
CMP命令の完了を検出
S241
プログラムはデバッグ対象？
NO
YES
S242
所定の条件を満たすCMP命令？
NO
YES
S243
依頼フラグをOFF
S244
他CPUの衝突情報あり？
NO
YES
S245
衝突情報をログ記録領域にコピー
S246
NO
全ての他CPUを確認した？
YES
S247
衝突情報が収集された？
NO
YES
S248
比較命令情報をログ記録領域にコピー
S249
ログ情報をログファイルに書き出す
終了

フロントページの続き

(56)参考文献　特開２０１０－１６０７０４（ＪＰ，Ａ）
特開２００６－３１８４１２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１１／３６