

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号

特許第7249717号

(P7249717)

(45)発行日 令和5年3月31日(2023.3.31)

(24)登録日 令和5年3月23日(2023.3.23)

(51)国際特許分類

F I

G 0 6 F 9/48 (2006.01)

G 0 6 F 9/48 3 5 0 C

G 0 6 F 9/455(2018.01)

G 0 6 F 9/455 1 5 0

請求項の数 12 (全69頁)

(21)出願番号 特願2020-524503(P2020-524503)
 (86)(22)出願日 平成30年11月2日(2018.11.2)
 (65)公表番号 特表2021-503126(P2021-503126 A)
 (43)公表日 令和3年2月4日(2021.2.4)
 (86)国際出願番号 PCT/IB2018/058613
 (87)国際公開番号 WO2019/097342
 (87)国際公開日 令和1年5月23日(2019.5.23)
 審査請求日 令和3年4月23日(2021.4.23)
 (31)優先権主張番号 15/812,034
 (32)優先日 平成29年11月14日(2017.11.14)
 (33)優先権主張国・地域又は機関 米国(US)

(73)特許権者 390009531
 インターナショナル・ビジネス・マシー
 ンズ・コーポレーション
 INTERNATIONAL BUSI
 NESS MACHINES CORPO
 RATION
 アメリカ合衆国10504 ニューヨー
 ク州 アーモンク ニュー オーチャード
 ロード
 New Orchard Road, A
 rmonk, New York 105
 04, United States of
 America
 (74)代理人 100112690
 弁理士 太佐 種一

最終頁に続く

(54)【発明の名称】 メモリ・ポインタを変更することによるコンテキスト切り替えを提供するコンピュータ・プログラム、コンピュータ・システム、およびコンピュータ実装方法

(57)【特許請求の範囲】

【請求項1】

コンピューティング環境内の処理を容易にするコンピュータ実装方法であって、
 第1のコンテキストから第2のコンテキストへのコンテキスト切り替えが実行されることを決定することと、ここで、前記第1のコンテキストのデータが、第1のメモリ・ユニットに少なくとも部分的に格納された1つまたは複数の構成状態レジスタに格納され、前記第2のコンテキストのデータが、第2のメモリ・ユニットに少なくとも部分的に格納された前記1つまたは複数の構成状態レジスタに格納され、前記第2のメモリ・ユニット内の構成状態レジスタのレイアウトが、前記第1のメモリ・ユニット内の構成状態レジスタのレイアウトと異なり、違いの指示がバージョン番号によって提供される、

前記第1のメモリ・ユニットから前記第2のメモリ・ユニットにポインタを変更することによって、前記コンテキスト切り替えを実行することと
 を含む、前記コンピュータ実装方法。

【請求項2】

前記ポインタが前記構成状態レジスタ内のベース・アドレスポインタである、請求項1に記載のコンピュータ実装方法。

【請求項3】

前記第1のメモリ・ユニットがメモリの1つのページであり、前記第2のメモリ・ユニットがメモリの別のページである、請求項1又は2に記載のコンピュータ実装方法。

【請求項4】

前記コンテキスト切り替えを前記実行することが、前記第 1 のコンテキストに関して 1 つまたは複数のプロセッサ内構成状態レジスタを処理することをさらに含み、前記 1 つまたは複数のプロセッサ内構成状態レジスタのデータが、前記 1 つまたは複数のプロセッサ内構成状態レジスタからコピーされてメモリに格納され、前記第 2 のコンテキストに関するデータが、前記 1 つまたは複数のプロセッサ内構成状態レジスタにコピーされる、請求項 1 又は 2 に記載のコンピュータ実装方法。

【請求項 5】

前記第 1 のコンテキストがスーパーバイザまたはアプリケーションを含み、前記第 2 のコンテキストが別の前記スーパーバイザまたは前記アプリケーションを含む、請求項 1 又は 2 に記載のコンピュータ実装方法。

【請求項 6】

前記第 1 のコンテキストが 1 つのスレッドを含み、前記第 2 のコンテキストが別のスレッドを含む、請求項 1 又は 2 に記載のコンピュータ実装方法。

【請求項 7】

複数のスレッドから前記別のスレッドが選択可能であり、前記複数のスレッドが、前記コンテキスト切り替えを実行するプロセッサに読み込まれない 1 つまたは複数のスレッドを含む、請求項 6 に記載のコンピュータ実装方法。

【請求項 8】

前記コンテキスト切り替えが、仮想マシンを 1 つのプロセッサから別のプロセッサに移動することに基づいて実行される、請求項 1 又は 2 に記載のコンピュータ実装方法。

【請求項 9】

前記コンテキスト切り替えを前記実行することが、前記第 1 のメモリ・ユニットに格納された前記 1 つまたは複数の構成状態レジスタに対応するキャッシュされたデータを無効化することを含む、請求項 1 又は 2 に記載のコンピュータ実装方法。

【請求項 10】

前記コンテキスト切り替えを前記実行することが、前記ポインタを変更する前に前記ポインタを格納することをさらに含み、請求項 2 に記載のコンピュータ実装方法。

【請求項 11】

コンピューティング環境内の処理を容易にするためのコンピュータ・システムであって、メモリと、

前記メモリと通信するプロセッサとを備えており、前記コンピュータ・システムが、請求項 1 ～ 10 のいずれか 1 項に記載の方法を実行するように構成されている、前記コンピュータ・システム。

【請求項 12】

コンピューティング環境内の処理を容易にするためのコンピュータ・プログラムであって、

請求項 1 ～ 10 のいずれか 1 項に記載の方法を実行することをコンピュータに実行させる、前記コンピュータ・プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

1 つまたは複数の態様は、一般に、コンピューティング環境内の処理に関連しており、特に、メモリ・ポインタを変更することによるコンテキスト切り替えにより、前記処理を容易にするコンピュータ・プログラム、コンピュータ・システム、およびコンピュータ実装方法に関連している。なお、「コンピュータ・プログラム」は「コンピュータ・プログラム製品」とも呼ぶ。

【背景技術】

【0002】

コンピューティング環境のコンピュータは、コンピュータ内の処理を制御する中央処理装置（CPU：central processing units）またはプロセッサを含んでいる。中央処理装

10

20

30

40

50

置の動作は、制御レジスタによって制御される。制御レジスタは、例えば、割り込み制御、アドレス指定モードの切り替え、ページング制御、またはコプロセッサ制御、あるいはその組合せなどの、特定のタスクを実行するプロセッサのレジスタである。

【0003】

制御レジスタは、プロセッサ・チップの直接上にある半導体素子などの、ラッチとして通常は実装される。一部のコンピュータは、コンピュータのアーキテクチャの実装によって定義されたとおりに、多数の制御レジスタを使用する。したがって、制御レジスタは、チップの面積の増大を表す。

【0004】

さらに、一部のコンピュータは、中央処理装置が複数のプロセスまたはスレッドを同時に実行できるマルチスレッドをサポートする。各スレッドは、制御レジスタの別々のセットを使用し、それによってチップ上の制御レジスタの数が増える。

【0005】

ラッチベースの制御レジスタの数の増加は、性能、チップ面積、または電力消費、あるいはその組合せに影響を与えることがある。例えば、コンテキスト切り替え時に制御レジスタが切り替えられるため、制御レジスタの数の増加によって、コンテキスト切り替えのコストが増える。さらに、ラッチベースの制御レジスタでは、制御に対する更新がプログラム順序で発生し、このことも性能に影響を与えることがある。

【0006】

異なるアーキテクチャは、制御レジスタについて異なる名前を有することがある。例えば、International Business Machines Corporation（ニューヨーク州アーモンク市）によって提供されるPower Architectureでは、制御レジスタは専用レジスタ（SPR：special purpose register）と呼ばれる。その他のアーキテクチャは、他の名前を使用することがある。本明細書では、制御レジスタの使用は、例えばSPRなどの、他の名前の制御レジスタを含む。

【先行技術文献】

【非特許文献】

【0007】

【文献】“z/Architecture Principles of Operation,” IBM Publication No. SA22-7832-10, March 2015

“Power ISA(TM) Version 2.07B,” International Business Machines Corporation, April 9, 2015

【発明の概要】

【発明が解決しようとする課題】

【0008】

本発明は、メモリ・ポインタを変更することによるコンテキスト切り替えにより、コンピューティング環境内の処理を容易にするためのコンピュータ・プログラム、コンピュータ・システム、およびコンピュータ実装方法を提供する。

【課題を解決するための手段】

【0009】

従来技術の欠点が克服され、追加の利点をもたらされるコンピュータ・プログラム製品は、処理回路によって読み取り可能な、方法を実行するための命令を格納しているコンピュータ可読記憶媒体を含む。この方法は、例えば、第1のコンテキストから第2のコンテキストへのコンテキスト切り替えが実行されるということを決断することを含む。第1のコンテキストのデータは、第1のメモリ・ユニットに少なくとも部分的に格納された1つまたは複数の構成状態レジスタに格納され、第2のコンテキストのデータは、第2のメモリ・ユニットに少なくとも部分的に格納された1つまたは複数の構成状態レジスタに格納される。第1のメモリ・ユニットから第2のメモリ・ユニットにポインタを変更することによって、コンテキスト切り替えが実行される。データをコピーするのではなく、ポインタを変更することによって、コンテキスト切り替え時の効率が実現され、それによって処

10

20

30

40

50

理を容易にし、性能を改善する。

【0010】

1つの例として、第1のメモリ・ユニットはメモリの1つのページであり、第2のメモリ・ユニットはメモリの別のページである。

【0011】

1つの実施形態では、コンテキスト切り替えを実行することは、第1のコンテキストに関して1つまたは複数のプロセッサ内構成状態レジスタを処理することを含む。1つまたは複数のプロセッサ内構成状態レジスタのデータが、1つまたは複数のプロセッサ内構成状態レジスタからコピーされてメモリに格納され、第2のコンテキストに関するデータが、1つまたは複数のプロセッサ内構成状態レジスタにコピーされる。

10

【0012】

さらに、1つの例では、コンテキスト切り替えを実行することは、ポインタを変更する前にポインタを格納することを含む。これによって柔軟性を実現し、コンピューティング環境内の処理を容易にする。

【0013】

1つの例として、第1のコンテキストが、スーパーバイザまたはアプリケーションを含み、第2のコンテキストが、別のスーパーバイザまたはアプリケーションを含む。

【0014】

別の例として、第1のコンテキストが1つのスレッドを含み、第2のコンテキストが別のスレッドを含む。1つの態様では、複数のスレッドから他のスレッドが選択可能である。複数のスレッドは、コンテキスト切り替えを実行するプロセッサに読み込まれない1つまたは複数のスレッドを含む。

20

【0015】

1つの実施形態では、コンテキスト切り替えは、仮想マシンを1つのプロセッサから別のプロセッサに移動することに基づいて実行される。

【0016】

さらに、1つの実施形態では、第2のメモリ・ユニット内の構成状態レジスタのレイアウトが、第1のメモリ・ユニット内の構成状態レジスタのレイアウトと異なる。1つの例として、この違いの指示がバージョン番号によって提供される。

【0017】

1つの実施形態では、コンテキスト切り替えを実行することが、第1のメモリ・ユニットに格納された1つまたは複数の構成状態レジスタに対応するキャッシュされたデータを無効化することを含む。

30

【0018】

1つまたは複数の態様に関連するコンピュータ実装方法およびシステムも本明細書に記載され、請求される。さらに、1つまたは複数の態様に関連するサービスも本明細書に記載されており、請求されてよい。

【0019】

その他の特徴および長所が、本明細書に記載された技術によって実現される。その他の実施形態および態様は、本明細書において詳細に説明され、請求される態様の一部と見なされる。

40

【0020】

1つまたは複数の態様は、本明細書の最後にある特許請求の範囲において例として具体的に指摘され、明確に請求される。前述の内容、ならびに1つまたは複数の態様の目的、特徴、および長所は、添付の図面と併せて行われる以下の詳細な説明から明らかになる。

【図面の簡単な説明】

【0021】

【図1】本発明の1つまたは複数の態様を組み込んで使用するためのコンピューティング環境の一例を示す図である。

【図2】本発明の1つまたは複数の態様を組み込んで使用するためのコンピューティング

50

環境の別の例を示す図である。

【図 3】本発明の 1 つまたは複数の態様に従って、図 1 または図 2 のプロセッサの詳細をさらに示す図である。

【図 4】本発明の 1 つまたは複数の態様に従って使用される命令実行パイプラインの一例の詳細をさらに示す図である。

【図 5】本発明の態様に従って、プロセッサの一例の詳細をさらに示す図である。

【図 6】本発明の態様に従って、プロセッサ内構成状態レジスタおよびメモリ内構成状態レジスタの一例を示す図である。

【図 7】本発明の態様に従って、メモリ内構成状態レジスタを使用することに関連付けられたデコード論理の一例を示す図である。

【図 8】本発明の態様に従って、構成状態レジスタ読み込み内部動作の一例を示す図である。

【図 9】本発明の態様に従って、構成状態レジスタ格納内部動作の一例を示す図である。

【図 10】本発明の態様に従って、メモリ内構成状態レジスタを使用する一例を示す図である。

【図 11】本発明の態様に従って、メモリ内構成状態レジスタを使用する別の例を示す図である。

【図 12】本発明の態様に従って、構成状態レジスタの書き込み動作の一例を示す図である。

【図 13】本発明の態様に従って、構成状態レジスタの読み取り動作の一例を示す図である。

【図 14】本発明の態様に従って、構成状態レジスタへの移動または構成状態レジスタからの移動に関連付けられたデコード論理の一実施形態を示す図である。

【図 15】本発明の態様に従って、構成状態レジスタへの移動命令に関連付けられた詳細をさらに示す図である。

【図 16】本発明の態様に従って、構成状態レジスタからの移動命令の詳細をさらに示す図である。

【図 17】本発明の態様に従って、複合構成状態レジスタの読み取り参照に関連付けられた論理の一実施形態を示す図である。

【図 18】本発明の態様に従って、複合構成状態レジスタの書き込み参照に関連付けられた論理の一実施形態を示す図である。

【図 19】本発明の態様に従って、複合構成状態レジスタの一例を示す図である。

【図 20】本発明の態様に従って、構成状態レジスタの線形マッピングの一例を示す図である。

【図 21】本発明の態様に従って、構成状態レジスタの線形マッピングの一例を示す図である。

【図 22】本発明の態様に従って、構成状態レジスタの再マッピング・フロー論理の一例を示す図である。

【図 23】複数の構成状態レジスタの格納動作の一例を示す図である。

【図 24】本発明の態様に従って、構成状態レジスタ一括格納動作の一例を示す図である。

【図 25】本発明の態様に従って、構成状態レジスタ一括読み込み動作の一例を示す図である。

【図 26】本発明の態様に従って、アーキテクチャの構成制御を指定する一例を示す図である。

【図 27】本発明の態様に従って、アーキテクチャの構成制御を指定する別の例を示す図である。

【図 28】本発明の態様に従って、コンテキスト切り替えを実行する一例を示す図である。

【図 29】本発明の態様に従って、コンテキスト切り替えを実行する別の例を示す図である。

【図 30】本発明の態様に従って、構成状態レジスタへの移動動作に関連付けられたアド

10

20

30

40

50

レス変換の一実施形態を示す図である。

【図 3 1】本発明の態様に従って、動的アドレス変換を実行する例を示す図である。

【図 3 2】本発明の態様に従って、動的アドレス変換を実行する例を示す図である。

【図 3 3】本発明の態様に従って、ページ・テーブル・エントリの一例を示す図である。

【図 3 4】本発明の態様に従って、特定のコンテキストに関連付けられている特定の構成状態レジスタの一例を示す図である。

【図 3 5】本発明の態様に従って、固定通知をホスト・システムに提供する一実施形態を示す図である。

【図 3 6】本発明の態様に従って、ページ・テーブル・エントリでの固定動作を指定する一実施形態を示す図である。

【図 3 7】本発明の態様に従って、ページ・テーブル・エントリでの固定解除動作を指定する一実施形態を示す図である。

【図 3 8】本発明の態様に従って、1回のハイパーバイザの呼び出しで固定動作および固定解除動作を結合する一例を示す図である。

【図 3 9】本発明の態様に従って、1回の呼び出しに基づいて固定動作および固定解除動作を実行することに関連付けられた詳細をさらに示す図である。

【図 4 0】本発明の1つまたは複数の態様に従って、データ書き込みのさまざまな例を示す図である。

【図 4 1】本発明の1つまたは複数の態様に従って、データ書き込みのさまざまな例を示す図である。

【図 4 2】本発明の1つまたは複数の態様に従って、データ書き込みのさまざまな例を示す図である。

【図 4 3】本発明の1つまたは複数の態様に従って、データ読み取りのさまざまな例を示す図である。

【図 4 4】本発明の1つまたは複数の態様に従って、データ読み取りのさまざまな例を示す図である。

【図 4 5】本発明の1つまたは複数の態様に従って、データ読み取りのさまざまな例を示す図である。

【図 4 6】本発明の態様に従って、コンピューティング環境内の処理を容易にする一実施形態を示す図である。

【図 4 7】本発明の態様に従って、コンピューティング環境内の処理を容易にする一実施形態を示す図である。

【図 4 8】本発明の1つまたは複数の態様を組み込んで使用するためのコンピューティング環境の別の例を示す図である。

【図 4 9】図 4 8のメモリの詳細をさらに示す図である。

【図 5 0】クラウド・コンピューティング環境の一実施形態を示す図である。

【図 5 1】抽象モデル・レイヤの一例を示す図である。

【発明を実施するための形態】

【0022】

本発明の態様に従って、さまざまな構成状態レジスタが、プロセッサ内ではなくメモリ内で提供される。本明細書において使用されるとき、「構成状態レジスタ」という用語は、制御レジスタ、プログラム状態ワード（PSW：program status word）またはその他のマシン状態レジスタなどのマシン状態レジスタ（MSR：machine state register）、状態レジスタ（例えば、浮動小数点状態制御レジスタ）、専用レジスタ（SPR）、構成レジスタ、または例えば命令の動作を構成するその他のレジスタ、あるいはその組合せを含む。

【0023】

選択された構成状態レジスタ（またはさらに別の態様では、その一部）は、メモリ内で提供され、メモリ内でそれらのレジスタは、システム・メモリにマッピングされ、プロセッサに接続されているがプロセッサから離れているメモリ階層に含まれる。メモリ階層は

10

20

30

40

50

、例えば、読み込み／格納キュー、1つまたは複数のメモリ・キャッシュ、およびシステム・メモリ（本明細書では、メイン・メモリ、中央記憶装置、ストレージ、主記憶装置、メモリとも呼ばれる）を含む。プロセッサ内ではなくメモリ内に存在することにより、レジスタは、メモリ・アドレスを使用することによってアクセスされ、アクセス要求が、再順序付けされるか、または投機的に処理されてよい。これに対して、プロセッサ内に存在する構成状態レジスタに対するアクセス要求は、不規則な順序では処理されず、投機的にも処理されない。プロセッサ内構成状態レジスタは、例えば直接チップ上で、例えば半導体素子（ラッチなど）として実装される。「チップ上」は、単一の集積回路に含まれているか、または特定のデバイスと同じ集積回路に含まれている、回路を示すか、またはそのような回路に関連している。

10

【0024】

構成状態レジスタがシステム・メモリに格納されているということに基づいて、構成状態レジスタへの移動命令（例えば、SPRへの移動（`mt spr : move to SPR`）命令）および構成状態レジスタからの移動命令（例えば、SPRからの移動（`mf spr : move from SPR`）命令）などの、特定の命令が、命令デコード論理による読み込み命令または動作および格納命令または動作に置き換えられる。生成された読み込み命令／動作および格納命令／動作は、格納キューにコミットされ、標準的な読み込み処理および格納処理が実行される。

【0025】

1つの例として、構成状態レジスタを含むための記憶領域が、オペレーティング・システムまたはハイパーバイザあるいはその両方によって定義され、メモリベースのレジスタを格納するために確保される。1つの実施形態では、物理メモリ領域がアーキテクチャによって指定される（例えば、物理メモリの最初または最後のnページ）。

20

【0026】

さらに別の態様では、構成状態レジスタの1つまたは複数の部分がメモリ内に設けられ、一方、構成状態レジスタの1つまたは複数の他の部分がプロセッサ内に設けられる。1つの例では、メモリ内に設けられた部分は、あまり頻繁に使用されない部分である。

【0027】

さらに別の態様では、処理性能を改善するために、通常は一緒に使用される構成状態レジスタ（または少なくともその一部）がメモリ内に一緒に（例えば、単一のキャッシュ・ラインまたは隣接するキャッシュ・ライン内に）配置されるように、構成状態レジスタの再マッピングが提供される。

30

【0028】

さらに別の態様では、複数の構成状態レジスタの一括格納または一括読み込みを実行するための命令または動作が提供される。これは、例えば、コンテキスト切り替えを容易にして、その性能を改善するためである。

【0029】

さらに別の1つの態様では、メモリ内のどこに構成状態レジスタが格納されるかを識別するための制御のセットを定義することによって、処理が容易にされ、性能が改善される。

【0030】

さらに別の態様では、メモリ内構成状態レジスタのメモリ・ポインタを操作することによって、コンテキスト切り替え時の効率が達成される。古い構成データをコピーするのではなく、それらのポインタが操作される。これによって、速度を上げ、コンテキスト切り替え時の複雑さを減らして、コンピューティング環境内の処理を改善する。

40

【0031】

さらに別の態様では、後で命令の処理において発生する可能性のあるページ・フォールトを防ぐために、ベース・アドレスとして使用されるアドレスを読み込む命令を実行することに基づいて、アドレス変換が自動的に実行される。

【0032】

さらに別の態様では、管理の柔軟性を向上させることによって処理を容易にするために

50

、構成状態レジスタがコンテキストまたはグループ（例えば、ハイパーバイザ、オペレーティング・システム、プロセス、スレッド）別に分離される。

【0033】

さらに別の態様として、初期化されたメモリ補助の状態のための自動固定の指示が提供される。

【0034】

さらに別の態様では、準仮想化された固定呼び出しを使用して、メモリ・ページの固定が効率的に管理される。

【0035】

さらに別の1つの態様では、システム・メモリがシングル・イベント・アップセットに対して保護される。

【0036】

本明細書では、さまざまな態様が説明される。さらに、本発明の態様の思想を逸脱することなく、多くの変形が可能である。本明細書に記載された各態様および特徴ならびにその変形を、特に矛盾しない限り、任意の他の態様または特徴と組み合わせることができるということに、注意すべきである。

【0037】

本発明の1つまたは複数の態様を組み込んで使用するためのコンピューティング環境の一実施形態が、図1を参照して説明される。1つの例では、コンピューティング環境は、International Business Machines Corporation（ニューヨーク州アーモンク市）によって提供されるz/Architectureに基づく。z/Architectureの一実施形態は、“z/Architecture Principles of Operation,” IBM Publication No. SA22-7832-10, March 2015に記載されており、この文献は本明細書において参照によってその全体が本明細書に組み込まれている。Z/ARCHITECTUREは、International Business Machines Corporation（米国ニューヨーク州アーモンク市）の登録商標である。

【0038】

別の例では、コンピューティング環境は、International Business Machines Corporation（ニューヨーク州アーモンク市）によって提供されるPower Architectureに基づく。Power Architectureの一実施形態は、“Power ISA(TM) Version 2.07B,” International Business Machines Corporation, April 9, 2015に記載されており、この文献は本明細書において参照によってその全体が本明細書に組み込まれている。POWER ARCHITECTUREは、International Business Machines Corporation（米国ニューヨーク州アーモンク市）の登録商標である。

【0039】

コンピューティング環境は、Intel x86アーキテクチャを含むが、これに限定されない、他のアーキテクチャに基づいてもよい。その他の例も存在する。

【0040】

図1に示されているように、コンピューティング環境100は、例えば、汎用コンピューティング・デバイスの形態で示されるコンピュータ・システム102を含んでいる。コンピュータ・システム102は、1つまたは複数のバスまたはその他の接続110あるいはその両方を介して互いに結合された、1つまたは複数のプロセッサまたはプロセッシング・ユニット104（例えば、中央処理装置（CPU））、メモリ106（例えば、システム・メモリ、メイン・メモリ、主記憶装置、中央記憶装置、またはストレージとも呼ばれる）、および1つまたは複数の入出力（I/O: input/output）インターフェイス108を含んでよいが、これらに限定されない。

【0041】

バス110は、メモリ・バスまたはメモリ・コントローラ、ペリフェラル・バス、アク

10

20

30

40

50

セラレーテッド・グラフィックス・ポート、およびさまざまなバス・アーキテクチャのいずれかを使用するプロセッサまたはローカル・バスを含む、複数の種類のバス構造のいずれかのうちの1つまたは複数を表す。例として、そのようなアーキテクチャは、I S A (Industry Standard Architecture)、M C A (Micro Channel Architecture)、E I S A (Enhanced ISA)、V E S A (Video Electronics Standards Association) ローカル・バス、および P C I (Peripheral Component Interconnects) を含むが、これらに限定されない。

【0042】

メモリ106は、例えば、プロセッサ104のローカル・キャッシュ122に結合されてよい、共有キャッシュなどのキャッシュ120を含んでよい。さらに、メモリ106は、1つまたは複数のプログラムまたはアプリケーション130、オペレーティング・システム132、および1つまたは複数のコンピュータ可読プログラム命令134を含んでよい。コンピュータ可読プログラム命令134は、本発明の態様の実施形態の機能を実行するように構成されてよい。

10

【0043】

コンピュータ・システム102は、例えばI/Oインターフェイス108を介して、1つまたは複数の外部デバイス140、1つまたは複数のネットワーク・インターフェイス142、または1つまたは複数のデータ・ストレージ・デバイス144、あるいはその組合せと通信してもよい。外部デバイスの例としては、ユーザ端末、テープ・ドライブ、ポインティング・デバイス、ディスプレイなどが挙げられる。ネットワーク・インターフェイス142は、コンピュータ・システム102が、ローカル・エリア・ネットワーク(LAN: local area network)、一般的な広域ネットワーク(WAN: wide area network)、またはパブリック・ネットワーク(例えば、インターネット)、あるいはその組合せなどの1つまたは複数のネットワークと通信できるようにし、他のコンピューティング・デバイスまたはシステムとの通信を実現する。

20

【0044】

データ・ストレージ・デバイス144は、1つまたは複数のプログラム146、1つまたは複数のコンピュータ可読プログラム命令148、またはデータ、あるいはその組合せなどを格納してよい。コンピュータ可読プログラム命令は、本発明の態様の実施形態の機能を実行するように構成されてよい。

30

【0045】

コンピュータ・システム102は、取り外し可能／取り外し不可、揮発性／不揮発性のコンピュータ・システム・ストレージ媒体を含むか、またはそのようなコンピュータ・システム・ストレージ媒体に結合されるか、あるいはその両方であってよい。例えば、コンピュータ・システム102は、取り外し不可、不揮発性の磁気媒体(通常は、「ハード・ドライブ」と呼ばれる)、取り外し可能、不揮発性の磁気ディスク(例えば、「フロッピー・ディスク」)に対する読み取りと書き込みを行うための磁気ディスク・ドライブ、あるいはC D-R O M、D V D-R O M、またはその他の光媒体などの取り外し可能、不揮発性の光ディスクに対する読み取りと書き込みを行うための光ディスク・ドライブ、あるいはその組合せを含むか、またはこれらに結合されるか、あるいはその両方であってよい。その他のハードウェア・コンポーネントまたはソフトウェア・コンポーネントあるいはその両方を、コンピュータ・システム102と併用できるということが理解されるべきである。その例として、マイクロコード、デバイス・ドライバ、冗長プロセッシング・ユニット、外部ディスク・ドライブ・アレイ、R A I Dシステム、テープ・ドライブ、およびデータ・アーカイブ・ストレージ・システムなどが挙げられるが、これらに限定されない。

40

【0046】

コンピュータ・システム102は、他の多数の汎用または専用のコンピューティング・システム環境または構成で運用されてよい。コンピュータ・システム102での使用に適した周知のコンピューティング・システム、環境、または構成、あるいはその組合せの例としては、パーソナル・コンピュータ(P C: personal computer)システム、サーバ

50

・コンピュータ・システム、シン・クライアント、シック・クライアント、ハンドヘルドまたはラップトップ・デバイス、マイクロプロセッサ・システム、マイクロプロセッサベース・システム、セット・トップ・ボックス、プログラマブル・コンシューマ・エレクトロニクス、ネットワークPC、マイクロコンピュータ・システム、メインフレーム・コンピュータ・システム、およびこれらのシステムまたはデバイスのいずれかを含む分散クラウド・コンピューティング環境などが挙げられるが、これらに限定されない。

【0047】

別の実施形態では、コンピューティング環境が仮想マシンをサポートする。そのような環境の一例が、図2を参照して説明される。1つの例では、コンピューティング環境161は、仮想マシンをサポートする中央処理装置複合体(CPC: central processor complex)163を含んでいる。CPC163は、1つまたは複数の制御ユニット169を介して1つまたは複数の入出力(I/O)デバイス167に結合される。中央処理装置複合体163は、例えば、1つまたは複数のプロセッサ(中央処理装置(CPU: central processing unit)とも呼ばれる)171に結合されたメモリ165(システム・メモリ、メイン・メモリ、主記憶装置、中央記憶装置、ストレージとも呼ばれる)、および入出力サブシステム173を含んでおり、これらの各々が下で説明される。

【0048】

メモリ165は、例えば、1つまたは複数の仮想マシン175、仮想マシンを管理する仮想マシン・マネージャ(ハイパーバイザ177など)、およびプロセッサ・ファームウェア179を含んでいる。ハイパーバイザ177の1つの例は、International Business Machines Corporation(ニューヨーク州アーモンク市)によって提供されるz/VMである。ハイパーバイザは、ホストと呼ばれることがある。さらに、本明細書において使用されるとき、ファームウェアは、例えば、プロセッサのマイクロコードを含む。ファームウェアは、例えば、上位レベルの機械コードの実装において使用される、ハードウェア・レベルの命令またはデータ構造あるいはその両方を含む。1つの実施形態では、ファームウェアは、例えば、信頼できるソフトウェアを含んでいるマイクロコード、または基盤になるハードウェアに固有のマイクロコードとして通常は提供される、システムのハードウェアへのオペレーティング・システムのアクセスを制御する独自のコードを含む。

【0049】

CPCの仮想マシンのサポートは、多数の仮想マシン175を動作させることができるようにし、各仮想マシン175は、異なるプログラム185で動作し、Linuxなどのゲスト・オペレーティング・システム183を実行することができる。各仮想マシン175は、別々のシステムとして機能することができる。すなわち、各仮想マシンは、独立してリセットされ、ゲスト・オペレーティング・システムを実行し、異なるプログラムで動作することができる。仮想マシン内で実行されるオペレーティング・システムまたはアプリケーション・プログラムは、完全なシステム全体にアクセスできるように見えるが、実際は、その一部のみが利用可能である。

【0050】

メモリ165は、仮想マシンに割り当て可能な物理プロセッサ・リソースである、プロセッサ(例えば、CPU)171に結合される。例えば、仮想マシン175は、1つまたは複数の論理プロセッサを含み、それらの論理プロセッサの各々は、仮想マシンに動的に割り当てることができる物理プロセッサ・リソース171の全部または一部を表す。

【0051】

さらに、メモリ165は、I/Oサブシステム173に結合される。入出力サブシステム173は、入出力制御ユニット169およびデバイス167と主記憶装置165の間の情報の流れを管理する。入出力サブシステム173は中央処理装置複合体に結合され、この結合において、入出力サブシステム173は中央処理装置複合体の一部であるか、または中央処理装置複合体から分離することができる。

【0052】

10

20

30

40

50

さらに、プロセッサ 104（またはプロセッサ 171）などのプロセッサの一例に関する詳細が、図 3 を参照して説明される。プロセッサ 104（またはプロセッサ 171）などのプロセッサは、命令を実行するために使用される複数の機能コンポーネントを含む。これらの機能コンポーネントは、例えば、実行される命令をフェッチするための命令フェッチ・コンポーネント 150 と、フェッチされた命令をデコードするため、およびデコードされた命令のオペランドを取得するための命令デコード・ユニット 152 と、デコードされた命令を実行するための命令実行コンポーネント 154 と、必要な場合に、命令を実行するためにメモリにアクセスするためのメモリ・アクセス・コンポーネント 156 と、実行された命令の結果を提供するための書き戻しコンポーネント 160 とを含む。これらのコンポーネントのうちの 1 つまたは複数は、本発明の態様に従って、メモリベースの構成状態レジスタの処理 166 に関連付けられた 1 つまたは複数の命令または動作あるいはその両方を実行するために使用されてよい。

10

【0053】

プロセッサ 104（またはプロセッサ 171）は、1 つの実施形態では、機能コンポーネントのうちの 1 つまたは複数によって使用される 1 つまたは複数のレジスタ 168 も含む。プロセッサ 104（またはプロセッサ 171）は、本明細書で提供されている例よりも多いコンポーネント、少ないコンポーネント、またはその他のコンポーネント、あるいはその組合せを含んでよい。

【0054】

さらに、プロセッサ 104 またはプロセッサ 171 などのプロセッサの実行パイプラインに関する詳細が、図 4 を参照して説明される。本明細書では、パイプラインのさまざまな処理段階が図に示されて説明されているが、本発明の態様の思想から逸脱することなく、追加の段階、より少ない段階、またはその他の段階、あるいはその組合せが使用されてよいということが理解されるであろう。

20

【0055】

図 4 を参照すると、1 つの実施形態では、命令が命令キューからフェッチされ（170）、命令の分岐予測 172 またはデコーディング 174 あるいはその両方が実行されてよい。デコードされた命令が、一緒に処理される命令のグループ 176 に追加されてよい。グループ化された命令がマッパー 178 に提供され、マッパー 178 が、任意の依存関係を決定し、リソースを割り当て、命令／動作のグループを適切な発行キューにディスパッチする。例えば分岐、読み込み／格納、浮動小数点、固定小数点、ベクトルなどを含む、異なる種類の実行ユニットに対して、1 つまたは複数の発行キューが存在する。発行段階 180 の間に、命令／動作が適切な実行ユニットに発行される。いずれかのレジスタが読み取られて（182）、そのソースを取り出し、実行段階 184 の間に命令／動作が実行される。示されているように、この実行は、分岐の場合、例えば、読み込み（LD）または格納（ST）、固定小数点演算（FX）、浮動小数点演算（FP）、またはベクトル演算（VX）であってよい。書き戻し段階 186 の間に、いずれかの結果が適切なレジスタに書き込まれる。その後、命令が完了する（188）。割り込みまたはフラッシュ 190 が存在する場合、処理が命令フェッチ 170 に戻ってよい。

30

【0056】

さらに、1 つの例では、レジスタの保存／復元において使用されてよいレジスタ名変更ユニット 192 が、デコード・ユニットに結合される。

40

【0057】

プロセッサに関するその他の詳細が、図 5 を参照して説明される。1 つの例では、プロセッサ 104（またはプロセッサ 171）などのプロセッサは、予測ハードウェア、レジスタ、キャッシュ、デコーダ、命令順序付けユニット、および命令実行ユニットを例として含んでよい、パイプライン型プロセッサである。予測ハードウェアは、例えば、ローカル分岐履歴テーブル（BHT: branch history table）105a、グローバル分岐履歴テーブル（BHT）105b、およびグローバル・セクタ 105c を含む。予測ハードウェアは、次の命令フェッチのアドレスを有する命令フェッチ・アドレス・レジスタ（IF

50

A R : instruction fetch address register) 1 0 7 を介してアクセスされる。

【 0 0 5 8 】

同じアドレスが、「フェッチ・グループ」と呼ばれる複数の命令をフェッチしてよい命令キャッシュ 1 0 9 にも提供される。ディレクトリ 1 1 1 が命令キャッシュ 1 0 9 に関連付けられる。

【 0 0 5 9 】

キャッシュおよび予測ハードウェアは、同じアドレスを使用してほぼ同時にアクセスされる。予測ハードウェアが、フェッチ・グループ内の命令に使用できる予測情報を含んでいる場合、その予測が命令順序付けユニット (I S U : instruction sequencing unit) 1 1 3 に転送され、次に、命令順序付けユニット 1 1 3 が、命令を実行するために実行ユニットに発行する。この予測は、分岐ターゲット計算 1 1 5 および分岐ターゲット予測ハードウェア (リンク・レジスタ予測スタック 1 1 7 a およびカウンタ・レジスタ・スタック 1 1 7 b など) と連動して I F A R 1 0 7 を更新するために使用されてよい。予測情報を使用できないが、1 つまたは複数の命令デコーダ 1 1 9 がフェッチ・グループ内で分岐命令を検出した場合、そのフェッチ・グループに関する予測が作成される。予測された分岐が、分岐情報キュー (B I Q : branch information queue) 1 2 5 などの予測ハードウェアに格納され、I S U 1 1 3 に転送される。

【 0 0 6 0 】

I S U 1 1 3 によって分岐実行ユニット (B R U : branch execution unit) 1 2 1 に発行された命令に応答して、分岐実行ユニット 1 2 1 が動作する。B R U 1 2 1 は、条件レジスタ (C R : condition register) ファイル 1 2 3 に対する読み取りアクセス権限を有する。さらに分岐実行ユニット 1 2 1 は、分岐スキャン論理 (branch scan logic) によって分岐情報キュー 1 2 5 に格納された情報にアクセスして、分岐予測の成功を決定することができ、マイクロプロセッサによってサポートされる 1 つまたは複数のスレッドに対応する命令フェッチ・アドレス・レジスタ (I F A R) 1 0 7 に、動作可能なように結合される。少なくとも 1 つの実施形態に従って、B I Q エントリが識別子によって (例えば、分岐タグ (B T A G : branch tag) によって) 関連付けられ、識別される。B I Q エントリに関連付けられた分岐が完了したときに、そのことが B I Q エントリにマーク付けされる。B I Q エントリはキュー内で維持され、完了した分岐に関連付けられた情報を含んでいるとしてキュー・エントリがマーク付けされるときに、最も古いキュー・エントリの割り当てが順次解除される。B R U 1 2 1 は、B R U 1 2 1 が分岐予測ミスを発見したときに予測器の更新を引き起こすように、動作可能なようにさらに結合される。

【 0 0 6 1 】

命令が実行されるときに、B R U 1 2 1 が、予測が誤っているかどうかを検出する。予測が誤っている場合は、予測が更新される。この目的のために、プロセッサは予測器更新論理 1 2 7 も含んでいる。予測器更新論理 1 2 7 は、分岐実行ユニット 1 2 1 からの更新指示に応答して、ローカル B H T 1 0 5 a、グローバル B H T 1 0 5 b、およびグローバル・セクタ 1 0 5 c のうちの 1 つまたは複数に含まれる配列エントリを更新するように構成される。予測器ハードウェア 1 0 5 a、1 0 5 b、および 1 0 5 c は、命令フェッチおよび予測動作によって使用される、読み取りポートと異なる書き込みポートを有してよく、または単一の読み取り／書き込みポートが共有されてよい。予測器更新論理 1 2 7 は、リンク・スタック 1 1 7 a およびカウンタ・レジスタ・スタック 1 1 7 b に動作可能なようにさらに結合されてよい。

【 0 0 6 2 】

ここで条件レジスタ・ファイル (C R F : condition register file) 1 2 3 を参照すると、C R F 1 2 3 は、B R U 1 2 1 によって読み取りアクセス可能であり、固定小数点ユニット (F X U : fixed point unit) 1 4 1、浮動小数点ユニット (F P U : floating point unit) 1 4 3、およびベクトル・マルチメディア拡張ユニット (V M X U : vector multimedia extension unit) 1 4 5 を含むが、これらに限定さない、実行ユニットによって書き込み可能である。条件レジスタ論理 (C R L : condition register logic) 実行

ユニット 147 (CRUとも呼ばれる) および専用レジスタ (SPR: special purpose register) 処理論理 149 は、条件レジスタ・ファイル (CRF) 123 に対する読み取りおよび書き込みアクセス権限を有する。CRU 147 は、CRF ファイル 123 に格納された条件レジスタに対して論理演算を実行する。FXU 141 は、CRF 123 に対して書き込み更新を実行することができる。

【0063】

プロセッサ 104 (またはプロセッサ 171) は、読み込み／格納ユニット 151、ならびにさまざまなマルチプレクサ 153 およびバッファ 155 に加えて、アドレス変換テーブル 157 およびその他の回路をさらに含む。

【0064】

さらに、プロセッサ 104 またはプロセッサ 171 などのプロセッサ 200 によって使用されるさまざまなレジスタに関する詳細が、図 6 を参照して説明される。図に示されているように、プロセッサ 200 は、複数のプロセッサ内構成状態レジスタ (CSR: configuration state registers) 202 を含んでいる。例として、プロセッサ内構成状態レジスタは、リンク・レジスタ (LR: link register)、カウンタ・レジスタ (CTR: counter register)、マシン状態レジスタ (MSR)、浮動小数点状態制御レジスタ (FPSCR: floating point status control register)、次命令アドレス (NIA: next instruction address) レジスタ、および 1 つまたは複数の整数例外レジスタ (XER: exception register) を含んでいる。さらに、本発明の態様に従って、プロセッサ 200 に結合されたシステム・メモリ 206 は、1 つまたは複数のメモリ内構成状態レジスタ 208 を含んでいる。例として、メモリ内構成状態レジスタは、イベントベースの分岐復帰レジスタ (EBRR: event based branch return registers)、イベントベースの分岐 (EBB: event based branch) レジスタ、状態回復レジスタ (SRR: state restoration registers)、整数例外レジスタ (XER)、およびベクトル・レジスタ保存 (VRSAVE: vector register save) レジスタを含んでいる。1 つの例では、メモリ内構成状態レジスタ 208 は、システム・メモリ 206 のメモリ内構成状態レジスタ領域 210 に格納される。

【0065】

頻繁にアクセスされる (例えば、連続的に複数回アクセスされる) 構成状態レジスタは、プロセッサ 200 およびシステム・メモリ 206 に結合されたキャッシュ階層 212 に移動されてよい。

【0066】

1 つの態様に従って、メモリ内に移動または配置されている 1 つまたは複数の構成状態レジスタに基づいて、それらの構成状態レジスタへのプロセッサ内のアクセスが、メモリへのアクセスに置き換えられる。アクセスの種類を決定するデコード論理の一例が、図 7 を参照して説明される。この処理は、例えばプロセッサのデコード・ユニットまたは別のユニットあるいはその両方によって実行される。

【0067】

図 7 を参照すると、最初に命令が受信される (ステップ 300)。命令が SPR への移動 (mtspr) 命令などの構成状態レジスタへの移動命令であるかどうかに関する判定が行われる (照会 302)。命令が構成状態レジスタへの移動命令である場合、命令内で示された構成状態レジスタがメモリ内構成状態レジスタであるかどうかに関する判定がさらに行われる (照会 304)。メモリ内構成状態レジスタでない場合、構成状態レジスタへの移動命令の従来の処理が実行される (ステップ 306)。しかし、構成状態レジスタがメモリ内構成状態レジスタである場合、構成状態レジスタをメモリに格納する (例えば、構成状態レジスタの新しい内容をメモリに格納する) ために、構成状態レジスタ格納内部動作が生成される (ステップ 308)。

【0068】

照会 302 に戻ると、受信された命令が構成状態レジスタへの移動命令でない場合、命令が SPR からの移動 (mf spr) 命令などの構成状態レジスタからの移動命令である

10

20

30

40

50

かに関する判定がさらに行われる（照会 3 1 2）。命令が構成状態レジスタからの移動命令である場合、命令内で示された構成状態レジスタがメモリ内であるかどうかに関する判定が行われる（照会 3 1 4）。メモリ内でない場合、従来の構成状態レジスタからの移動処理が実行される（ステップ 3 1 6）。メモリ内である場合、レジスタの内容をメモリから取得するために、構成状態レジスタ読み込み内部動作が生成される（ステップ 3 1 8）。
【 0 0 6 9 】

照会 3 1 2 に戻ると、受信された命令が構成状態レジスタへの移動命令でも構成状態レジスタからの移動命令でもない場合、受信された命令が、構成状態レジスタを使用する別の命令であるかどうかを判定するための判定がさらに実行されてよい（照会 3 2 2）。受信された命令が構成状態レジスタを使用する別の命令である場合、命令によって実行されている機能に応じて、読み取りまたは書き込みあるいはその両方の内部動作が生成されてよい（ステップ 3 2 4）。受信された命令が構成状態レジスタを使用する別の命令でない場合、処理がステップ 3 3 2 に進み、ステップ 3 3 2 で従来の命令デコード処理が実行される。

【 0 0 7 0 】

本発明の他の態様では、構成状態レジスタを読み込み、構成状態レジスタ値を格納するための内部動作が、命令に対応しないプロセッサの動作を実行することと連動して、例えば、割り込み要求を受信することに応答して例外ハンドラに移行することに応答して、実行される。

【 0 0 7 1 】

構成状態レジスタ読み込み内部動作に関する詳細が、図 8 を参照してさらに説明される。この処理は、プロセッサによって実行される。図 8 を参照すると、最初にメモリ・ベース・アドレス（ベース）が、構成状態レジスタを含んでいるメモリのベース・アドレスであるメモリ・ユニット（例えば、メモリ・ページ）のアドレスを含んでいるレジスタまたは位置（例えば、スレッド制御ベース・レジスタ（TCBR：thread control base register）などのベース・レジスタ）から取得される（ステップ 4 0 0）。さらに、動作において示されるレジスタ番号が取得される（ステップ 4 0 2）。そのレジスタ番号が、メモリ内のオフセットにマッピングされる（ステップ 4 0 4）。例えば、各構成状態レジスタ番号（または別の実施形態では、他の識別情報）が、メモリ内の特定の位置にマッピングされる。その位置は、ベース・アドレスからの特定の数量（例えば、オフセット）である。次に、アドレス（ベース・アドレスにオフセットを加算したアドレス）からの読み込みが実行され（ステップ 4 0 6）、読み込まれた値が返される（ステップ 4 0 8）。

【 0 0 7 2 】

本明細書において使用されるとき、ベースとは、メモリ内構成状態レジスタを含んでいるメモリのベース・アドレスのことを指し、ベース・レジスタとは、ベースを含んでいるレジスタのことを指す。ベース・レジスタの一例はスレッド制御ベース・レジスタ（TCBR）であるが、他の文脈（例えば、オペレーティング・システムなど）では、他のベース・レジスタを使用することがある。

【 0 0 7 3 】

構成状態レジスタ格納内部動作に関する詳細が、図 9 を参照してさらに説明される。この処理は、プロセッサによって実行される。図 9 を参照すると、最初にメモリ・ベース・アドレス（ベース）が、例えばベース・レジスタから（例えば TCBR から）取得され（ステップ 5 0 0）、動作において示されたレジスタ番号も取得される（ステップ 5 0 2）。レジスタ番号がメモリ内のオフセットにマッピングされ（ステップ 5 0 4）、格納のオペランド（例えば、レジスタの内容）が、ベース・アドレスにオフセットを加算することによって指定されたアドレスに格納される（ステップ 5 0 6）。

【 0 0 7 4 】

上で示されたように、構成状態レジスタからの移動命令でも構成状態レジスタへの移動命令でもない命令が構成状態レジスタを使用することがある。したがって、そのような命令のうちの 1 つに関連付けられた処理が、図 1 0 を参照して説明される。この処理は、プ

10

20

30

40

50

ロセッサによって実行される。図10を参照すると、この実施形態では、構成状態レジスタの読み取り参照を含んでいる命令／動作が取得される（ステップ600）。それに基づいて、命令／動作において示された構成状態レジスタのメモリ・ベース・アドレス（ベース）が、例えばベース・レジスタから（例えばTCBRから）取得され（ステップ602）、命令／動作において示されたレジスタ番号も取得される（ステップ604）。レジスタ番号がメモリ内のオフセットにマッピングされ（ステップ606）、ベースにオフセットを加算することによって指定されたアドレスからの内容が、一時的レジスタに読み込まれる（ステップ608）。次に、一時的レジスタが使用される（ステップ610）。

【0075】

図11を参照して説明されているように、構成状態レジスタの書き込み参照に関して、同様の処理が実行される。この処理は、プロセッサによって実行される。図11を参照すると、1つの例では、構成状態レジスタの書き込み参照が取得される（ステップ700）。それに基づいて、メモリ・ベース・アドレス（ベース）が、例えば、命令／動作で示された構成状態レジスタのベース・レジスタから（例えばTCBRから）取得され（ステップ702）、それに加えて、命令／動作で指定されたベース・レジスタ番号が取得される（ステップ704）。レジスタ番号がオフセットにマッピングされ（ステップ706）、書き込み参照に（例えば、一時的レジスタに）含まれている内容が、ベースにオフセットを加算して指定されたアドレスに格納される（ステップ708）。

【0076】

構成状態レジスタの書き込み動作（構成状態レジスタへの移動（例えば、mtspr）など）の動作可能な実装の考え方に関する詳細が、図12を参照してさらに説明される。この処理は、プロセッサによって実行される。図12を参照すると、1つの例では、動作で指定されたレジスタ番号が変換される（ステップ800）。例えば、レジスタ番号（または他の指示）に対応するか、またはマッピングされたメモリ・アドレスが、（例えば、ルックアップ・テーブルを使用することによって、または計算されて）決定される。さらに、格納キュー・エントリが割り当てられ（ステップ802）、対象の構成状態レジスタに対応するアドレスが格納キュー・エントリに格納される（ステップ804）。さらに、対象の構成状態レジスタに書き込まれたデータに対応する内容（例えば、データ値）が、格納キュー・エントリに書き込まれる（ステップ806）。1つの例では、ステップ804および806は、不規則な順序で実行されてよい。

【0077】

格納キュー・エントリは、示されたアドレスに対する読み取り（例えば、格納キューからの回避）に関して監視される（ステップ808）。さらに、投機誤りイベントに基づいて、格納キュー・エントリがフラッシュされてよく、投機誤りイベントは、1つの例では、アーキテクチャ的に正しい順序の位置まで発生する可能性がある（ステップ810）。

【0078】

内容（例えば、データ値）が、メモリ階層（例えば、第1のレベルのキャッシュ）内のアドレスに書き込まれる（ステップ812）。読み取り要求に基づいて、第1のレベルのキャッシュからのデータが提供される（ステップ814）。さらに、キャッシュ置換ポリシーに基づいて、第1のレベルのキャッシュからキャッシュ階層の1つまたは複数の次のレベルへ、データが退避される（ステップ816）。読み取り要求に基づいて、キャッシュ階層の1つまたは複数の次のレベルからのデータが提供される（ステップ818）。キャッシュ置換ポリシーに基づいて、キャッシュの各レベルからのデータがシステム・メモリ（例えば、DRAM）に退避される（ステップ820）。読み取り要求に基づいて、システム・メモリからのデータが提供される（ステップ822）。

【0079】

構成状態レジスタの読み取り動作の動作可能な実装の考え方に関する詳細が、図13を参照してさらに説明される。この処理は、プロセッサによって実行される。図13を参照すると、1つの例では、読み取り動作によって指定されたレジスタ番号が、対応するメモリ・アドレスに変換される（ステップ900）。読み込み要求の追跡に使用される読み込

10

20

30

40

50

みキュー内の位置を示すために使用される読み込みシーケンス番号、および依存関係を追跡する読み込みタグが、取得される（ステップ902）。格納キュー内で読み取られている構成状態レジスタに対応するアドレスにデータが存在する（すなわち、格納キュー内のメモリから読み取られるデータが存在する）かどうかのテストが実行される（ステップ904）。読み取られる構成状態レジスタのデータが格納キュー内で検出された場合（照会906）、格納キューから値が取得され（ステップ908）、処理が完了する。

【0080】

照会906に戻ると、読み取られる構成状態レジスタのデータが格納キュー内で検出されない場合、読み取られる構成状態レジスタのデータが第1のレベルのキャッシュ内で検出されるかどうかに関する判定がさらに行われる（照会910）。データが第1のレベルのキャッシュ内で検出された場合、第1のレベルのキャッシュから値が取得され（ステップ912）、処理が完了する。

10

【0081】

しかし、照会910に戻ると、データが第1のレベルのキャッシュ内で検出されない場合、読み取られる構成状態レジスタのデータが1つまたは複数の次のレベルのキャッシュ内で検出されるかどうかに関する判定がさらに行われる（照会914）。データが1つまたは複数の次のレベルのキャッシュ内で検出された場合、次のレベルのキャッシュからデータが取得され（ステップ916）、処理が完了する。

【0082】

照会914に戻ると、データが1つまたは複数の次のレベルのキャッシュ内にない場合、システム・メモリからデータを取り出すために、読み取り要求が読み込みキューに発行される（ステップ918）。システム・メモリからの読み込みが完了した場合、構成状態レジスタに対応するデータが取得される（ステップ920）。

20

【0083】

本発明の態様に従って、ソフトウェアの互換性を実現するために、メモリ内メモリ・ユニット（例えば、ページ）の割り当てが実行される。例えば、プロセッサがレガシー・ハイパーバイザを実行できるようにする、およびハイパーバイザがレガシー・オペレーティング・システムを実行できるようにする、などのために、ファームウェアによって割り当てが実行される。

【0084】

1つの例では、システムの初期ブート時に、ファームウェアが、ファームウェアが所有するメモリ内のメモリ内ページをメモリ内構成状態レジスタに割り当てる。1つの例では、ハイパーバイザがメモリ内構成状態レジスタを認識しない場合、ベース・レジスタ（例えば、TCBRなど）へのそれ以上のソフトウェアの参照なしで、ファームウェアが所有するページがシステムの実行全体を通じて使用される。

30

【0085】

そのため、ハイパーバイザは、例えば構成状態レジスタからの移動（例えば、mfspr）を使用してコンテキストを読み取り、例えば構成状態レジスタへの移動（例えば、mtspr）を使用してコンテキストを再読み込みすることによって、コンテキスト切り替えを簡単に実行する。これによって、コンピュータ・システム内の大幅な設計の簡略化および性能優位性を実現する。

40

【0086】

さらに、1つの例では、ハイパーバイザが、メモリによって補助されたページを認識する場合、ハイパーバイザは、補助ページのセットを含むように新しい各パーティションを構成してよい。さらに、オペレーティング・システムがメモリ内構成状態レジスタを認識しない場合、例えばベース・レジスタ（例えば、TCBRなど）へのそれ以上のソフトウェアの参照なしで、ハイパーバイザが所有するページがシステムの実行全体を通じて使用される。ハイパーバイザも認識しない場合、オペレーティング・システムが、ファームウェアが所有するページを使用する。

【0087】

50

そのため、オペレーティング・システムは、例えば構成状態レジスタからの移動（例えば、`m f s p r`など）を使用してコンテキストを読み取り、例えば構成状態レジスタへの移動（例えば、`m t s p r`など）を使用してコンテキストを再読み込みすることによって、コンテキスト切り替えを簡単に実行する。これによって、コンピュータ・システム内の大幅な設計の簡略化および性能優位性を実現し、処理を容易にする。

【0088】

本明細書において説明されているように、1つまたは複数の態様に従って、選択された構成状態レジスタがシステム・メモリに格納される。したがって、構成状態レジスタへの移動および構成状態レジスタからの移動が、命令デコード論理によって、読み込み命令および格納命令に置き換えられる。そのように生成された読み込みおよび格納は、格納キューにコミットされ、通常の読み込み処理および格納処理が実行される。1つの例では、常に必要ではない構成状態レジスタ（例えば、プログラム・カウンタ（`PC`: `program counter`）、データおよびアドレス・ブレーク・ポイント・レジスタ、`PSW`、浮動小数点制御などのレジスタ以外のレジスタ）は、メモリに格納される構成状態レジスタである。

【0089】

1つの例として、記憶領域が、オペレーティング・システムおよびハイパーバイザによって定義され、メモリベースのレジスタを格納するために確保される。1つの実施形態では、物理メモリ領域がアーキテクチャによって指定される（例えば、物理メモリの最初または最後の n ページ）。

【0090】

少なくとも1つの実施形態では、メモリ内構成状態レジスタが、通常のキャッシュ可能なメモリにマッピングされる。構成状態レジスタが更新される場合、その構成状態レジスタが格納キューに格納される。格納キューは、キュー・メカニズムであるだけでなく、メモリ・アクセスの投機的実行を可能にするために、格納用の位置の名前を変更する方法を効果的に提供する。アドレスの投機的値の複数のバージョンが（キャッシュ内またはシステム・メモリ内の、アーキテクチャ的に正しい順序の位置での信頼できる設計済み値に加えて）、格納キュー内に存在することができる。キャッシュ・エントリは、割り当てられた後に、不規則な順序で更新されてよい。格納は、エントリを格納キューからフラッシュすることによって取り消されてもよい。

【0091】

したがって、メモリ内構成状態レジスタは、性能を犠牲にすることなく、格納キューを使用することによって更新し、不規則な順序で読み取ることができ、コア内のラッチベースの構成状態レジスタは、プロセッサ内構成状態レジスタの投機的実行のための手段を実装することが、ほとんどの場合、非常に高価であるため、2つのシリアルライズおよび順序通りのアクセスによるコストの不利益を強いる。

【0092】

さらに、値が格納キュー内にない場合、頻繁に使用されるメモリ制御（例えば、メモリ内構成状態レジスタ）が、キャッシュ内で検出され、わずか2〜3サイクル（第1のレベルのキャッシュにアクセスするための時間）で使用可能になることが可能であるため、値の読み取りを、ラッチからよりも効率的に実行することができ、この速度は、プロセッサ内のラッチベースの構成状態レジスタにアクセスするために必要な専用論理よりも非常に高速である。

【0093】

1つの実施形態では、構成状態レジスタを保持するためのページが割り当てられた場合、アーキテクチャは、メモリのオペランドを使用するページへのアクセスを許可しない。これによって、メモリ操作と、構成状態レジスタからの／への移動命令との間の連結を防ぐ。

【0094】

本発明のさらに別の態様に従って、構成状態レジスタの1つまたは複数の部分がメモリ内に設けられ、一方、構成状態レジスタの1つまたは複数の他の部分がプロセッサ内に設

10

20

30

40

50

けられる。例えば、構成状態レジスタは複数の部分（例えば、フィールド）を含んでよく、それらの部分のうちの例えば頻繁にアクセスされる1つまたは複数の、プロセッサ内に残ってよく、例えばあまり使用されない残りの部分が、メモリに移動されてよい。これが、図14～図19を参照してさらに詳細に説明される。

【0095】

最初に図14を参照すると、プロセッサ（または別のコンポーネント）のデコード・ユニットが命令を受信する（ステップ1000）。デコード・ユニット（または別のコンポーネント）によって、命令がSPRへの移動（mtspr）命令などの構成状態レジスタへの移動（mtcsr：move to configuration state register）命令であるかどうかに関する判定が行われる（照会1002）。命令が構成状態レジスタへの移動命令である場合、その命令が下で説明されているように処理される（ステップ1004）。

10

【0096】

照会1002に戻ると、命令が構成状態レジスタへの移動命令でない場合、命令がSPRからの移動（mfspr）命令などの構成状態レジスタからの移動（mfcsr：move from configuration state register）命令であるかに関する判定がさらに行われる（照会1006）。命令が構成状態レジスタからの移動命令である場合、その命令が下で説明されているように処理される（ステップ1008）。照会1006に戻ると、命令が構成状態レジスタへの移動命令でも構成状態レジスタからの移動命令でもない場合、従来の命令デコードが実行される（ステップ1010）。

【0097】

さらに別の実施形態では、命令が構成状態レジスタを使用する別の命令であるかどうかに関する他の照会が行われてよく、そのような命令である場合、それらの命令が適切に処理されてよく、その例が本明細書において説明されている。さらに別の実施形態では、命令に対応していない（例えば、例外処理シーケンスを開始している）プロセッサの動作が同様に実行されてよい。

20

【0098】

構成状態レジスタへの移動命令の処理に関する詳細が、図15を参照してさらに説明される。1つの例では、構成状態レジスタが専用レジスタ（SPR）であってよく、命令がSPRへの移動（mtspr）命令である。ただし、これは1つの例にすぎない。他の構成状態レジスタが同様に処理されてよい。この論理は、例えばプロセッサのデコード・ユニットなどの、プロセッサによって実行される。他の例では、1つまたは複数の他のコンポーネントがこの論理を実行する。

30

【0099】

図15を参照すると、mtspr命令などの構成状態レジスタへの移動命令を取得すること（例えば、受信すること、提供されること、選択することなど）に基づいて、命令によって指定された構成状態レジスタ（CSR）の少なくとも一部がメモリ内にあるかどうかに関する判定が行われる（照会1100）。メモリ内にない場合、構成状態レジスタへの移動命令（例えば、mtspr）の従来の処理が実行される（ステップ1102）。

【0100】

照会1100に戻ると、構成状態レジスタの少なくとも一部がメモリ内にある場合、構成状態レジスタ全体がメモリ内にあるかどうかに関する判定がさらに行われる（照会1104）。構成状態レジスタ全体がメモリ内にある場合、構成状態レジスタ格納内部動作が生成される（ステップ1106）。この内部動作に関連する処理の例が、図9を参照して説明されている。

40

【0101】

照会1104に戻ると、構成状態レジスタの1つまたは複数の部分のみがメモリ内にある場合、1つまたは複数のメモリ内構成状態レジスタの部分に対する1つまたは複数の構成状態レジスタ格納動作が生成される（ステップ1110）。さらに、1つまたは複数のコア内構成状態レジスタの部分に対する更新された内部動作が生成される（ステップ1112）。更新された内部動作は、指定されたコア内の部分のデータを含んでいる1つまた

50

は複数の汎用レジスタの内容をコア内構成状態レジスタの適切な部分にコピーする動作を実行する、1つまたは複数の命令、状態マシンなどであってよい。処理が完了する。

【0102】

構成状態レジスタからの移動命令の処理に関連付けられた処理に関する詳細が、図16を参照してさらに説明される。1つの例では、構成状態レジスタが専用レジスタ（SPR）であってよく、命令がSPRからの移動（mfspr）命令である。ただし、これは1つの例にすぎない。他の構成状態レジスタが同様に処理されてよい。この論理は、例えばプロセッサのデコード・ユニットなどの、プロセッサによって実行される。他の例では、1つまたは複数の他のコンポーネントがこの論理を実行する。

【0103】

図16を参照すると、mfspr命令などの構成状態レジスタからの移動命令を取得すること（例えば、受信すること、提供すること、選択することなど）に基づいて、構成状態レジスタの少なくとも一部がメモリ内にあるかどうかに関する判定が行われる。メモリ内がない場合、構成状態レジスタからの移動命令に対して従来の処理が実行される（ステップ1202）。

【0104】

照会1200に戻ると、構成状態レジスタの少なくとも一部がメモリ内にある場合、構成状態レジスタ全体がメモリ内にあるかどうかに関する判定がさらに行われる（照会1204）。構成状態レジスタ全体がメモリ内にある場合、構成状態レジスタ読み込み内部動作が生成される（ステップ1206）。この動作に関連する処理の例が、図8を参照して説明されている。

【0105】

照会1204に戻ると、構成状態レジスタの1つまたは複数の部分のみがメモリ内にある場合、メモリ内構成状態レジスタの部分のうちの1つまたは複数に対する1つまたは複数の構成状態レジスタ読み込み内部動作が生成される（ステップ1210）。さらに、1つまたは複数のコア内構成状態レジスタの部分に対する1つまたは複数の読み込み内部動作が生成される（ステップ1212）。

【0106】

さらに、1つの実施形態では、メモリ内の部分およびコア内の部分をアーキテクチャによって定義された構成状態レジスタのイメージに結合するために、1つまたは複数の内部動作が生成される（ステップ1214）。これは例えば、下で説明されているように、マスク下挿入命令、あるいはOR、AND、もしくはNOT、あるいはその組合せの論理回路を使用することを含んでよい。

【0107】

1つまたは複数の部分がプロセッサ内にあり、1つまたは複数の部分がメモリ内にある複合構成状態レジスタの使用に関する詳細が、図17を参照してさらに説明され、図17では読み取り参照が説明される。この論理は、例えばプロセッサのデコード・ユニットなどの、プロセッサによって実行される。他の例では、1つまたは複数の他のコンポーネントがこの論理を実行する。

【0108】

図17を参照すると、複合構成状態レジスタの読み取り参照1300に基づいて、アクセスされている特定の部分（コンポーネント（例えばフィールド）とも呼ばれる）がメモリ内にあるかプロセッサ内にあるかに関する判定が行われる（照会1310）。特定の部分がプロセッサ内にある場合、プロセッサ内のコンポーネントがアクセスされ（ステップ1320）、処理が下で説明されている照会1350に進む。しかし、特定のコンポーネントがメモリ内にある場合（照会1310）、図10を参照して説明された処理が実行される。例えば、メモリ・ベース・アドレス（ベース）が取得され（ステップ1330）、複合構成状態レジスタを参照する命令内で示されたレジスタ番号も取得される（ステップ1332）。レジスタ番号がオフセットにマッピングされ（ステップ1334）、アドレス（ベース+オフセット）から一時的レジスタへの読み込みが実行される（ステップ13

10

20

30

40

50

36)。次に、一時的レジスタが使用される（ステップ1338）。その後、またはステップ1320の後に、複合構成レジスタの別のコンポーネントがアクセスされるかどうかに関する判定が行われる（照会1350）。複合構成レジスタの別のコンポーネントがアクセスされる場合は、処理が照会1310を続行する。複合構成レジスタの別のコンポーネントがアクセスされない場合は、処理が完了する。

【0109】

1つまたは複数の部分がプロセッサ内にあり、1つまたは複数の部分がメモリ内にある複合構成状態レジスタの使用に関する詳細が、図18を参照してさらに説明され、図18では書き込み参照が説明される。この論理は、例えばプロセッサのデコード・ユニットなどの、プロセッサによって実行される。他の例では、1つまたは複数の他のコンポーネントがこの論理を実行する。

【0110】

図18を参照すると、複合構成状態レジスタの書き込み参照1360に基づいて、アクセスされている特定の部分（コンポーネント（例えばフィールド）とも呼ばれる）がメモリ内にあるかプロセッサ内にあるかに関する判定が行われる（照会1370）。特定の部分がプロセッサ内にある場合、プロセッサ内のコンポーネントがアクセスされ（ステップ1390）、処理が下で説明されている照会1388に進む。しかし、特定のコンポーネントがメモリ内にある場合（照会1370）、図11を参照して説明された処理が実行される。例えば、メモリ・ベース・アドレス（ベース）が取得され（ステップ1380）、複合構成状態レジスタを参照する命令内で示されたレジスタ番号も取得される（ステップ1382）。レジスタ番号がオフセットにマッピングされ（ステップ1384）、ベース+オフセットによって定義されたアドレスへの格納が実行される（ステップ1386）。その後、またはステップ1390の後に、複合構成レジスタの別のコンポーネントがアクセスされるかどうかに関する判定が行われる（照会1388）。複合構成レジスタの別のコンポーネントがアクセスされる場合は、処理が照会1370を続行する。複合構成レジスタの別のコンポーネントがアクセスされない場合は、処理が完了する。

【0111】

複合構成状態レジスタの1つの例が、図19に示されている。図に示されているように、この例では、複合構成状態レジスタ1400は、整数例外レジスタ（XER）に対応する専用レジスタ（SPR）1である。このレジスタは、複数のフィールド1402を含んでいる。1つの例では、フィールドのうちの1つまたは複数がプロセッサ内のフィールド1404であり、別のフィールド1406がメモリ内のフィールドである。この特定の例では、Xerf0、1、2（すなわち、XERのフィールド0、1、および2）の名前が、プロセッサ内ではSO（要約オーバーフロー）、OV（オーバーフロー）、およびCA（キャリー）に変更され、この例では名前が変更されないXerf3（XERのフィールド3）は、メモリ内のバイト・カウント・フィールドである。この構成では、複合構成状態レジスタに対してmtsprおよびmfsprをそれぞれ実行するために、次のIOPシーケンスが生成されて使用されてよい。

【0112】

mtspr__xer mtxerf2

【0113】

mtxerf0

【0114】

mtxerf1

【0115】

stxerf3

【0116】

上記では、XERレジスタに対するmtsprは、汎用レジスタの内容がXERのフィールド2にコピーされる、XERのフィールド2への移動（mtxerf2）、汎用レジスタの内容がXERのフィールド0にコピーされる、XERのフィールド0への移動（m

10

20

30

40

50

t x e r f 0)、および汎用レジスタの内容が X E R のフィールド 1 にコピーされる、X E R のフィールド 1 への移動 (m t x e r f 1) を含んでいる。m t s p r は、フィールド 3 がメモリ内にあるため、格納動作によって実行される X E R のフィールド 3 への格納 (s t x e r f 3) も含んでいる。

【0117】

m f s p r _ _ x e r m f x e r f 2

【0118】

m f x e r f 0

【0119】

または

【0120】

l d x e r f 3

【0121】

または

【0122】

m f x e r f 1

【0123】

または

【0124】

X E R からの移動の場合、プロセッサ内またはメモリ内のいずれかから、フィールドの各々が読み取られ、それらのフィールドが、例えば O R 演算によって結合される。例えば、フィールド 2 およびフィールド 0 の内容が読み取られ、O R 演算が実行されて結果 1 を提供し、次に、(例えば、フィールド 3 がメモリ内にあるため、x e r f 3 読み込み内部動作 (l d x e r f 3) などの読み込みを使用して) フィールド 3 の内容が読み取られ、結果 1 との O R 演算が実行されて結果 2 を生成する。さらに、フィールド 1 の内容が読み取られ、結果 2 との O R 演算が実行されて最終結果を提供し、この最終結果は、プロセッサ内かメモリ内に関わらず、フィールドを含む X E R のイメージである。

【0125】

本明細書において説明されているように、本発明の態様に従って、構成状態レジスタからの移動命令は、プロセッサ内の部分からの移動のシーケンス、およびメモリに格納された部分の読み取りを生成する。メモリ内の部分またはプロセッサ内の部分の読み取りの内容は、例えば、例えば O R 命令のシーケンスを使用して、照合される。さらに、構成状態レジスタへの移動命令は、プロセッサ内の部分への移動のシーケンス、およびメモリに格納された部分の格納を生成する。

【0126】

1つの態様では、メモリが構成状態レジスタに割り当てられた場合、オフセットがアーキテクチャによって指定される(例えば、定義されて外部から見える)か、またはマイクロアーキテクチャによって指定される(定義されるが、外部からは見えない)。例えば、オフセットは、構成状態レジスタ番号(またはその他の指示)から直接導き出されてよい。

【0127】

1つの例はマッピングであり、各構成状態レジスタが、対応するオフセット(ダブルワード単位)、すなわち、ベース*構成状態レジスタ番号にマッピングされ、構成状態レジスタ 1 が第 1 の位置にあり、構成状態レジスタ 2 が、第 1 の位置に定義されたバイト数(例えば 8)を加算した位置にある、などとなる。

【0128】

しかし、構成状態レジスタ番号は連続しておらず、メモリを浪費し、キャッシュ効率を低下させる。したがって、本発明の態様に従う別の実施形態では、メモリ・ページへのオフセットを直接導き出すために構成状態レジスタ番号が使用されず、機能的類似性に基づいて構成状態レジスタにオフセットが割り当てられる。したがって、キャッシュ局所性を向上させるために、共通の動作において一緒に使用される構成状態レジスタが、同じキャ

10

20

30

40

50

ッシュ・ラインまたは隣接するキャッシュ・ラインに割り当てられる。例えば、EBB処理は、例えばレジスタEBBHR、EBBRR、BESCR、およびTARを使用する。TARは、他のレジスタに隣接していない。しかし、これらのレジスタは、最終的に同じキャッシュ・ライン内または隣接するキャッシュ・ライン内に存在するように、すべてメモリに割り当てられる。

【0129】

線形マッピングの1つの例が、図20～図21に示されている。図に示されているように、1つの例では、線形マッピング1500はスパースである。例えば、1つの例では、1K未満の構成状態レジスタがマッピングされたとしても、8KB（2ページ）が使用される。さらに、EBBHR、EBBRR、BESCR、およびTARなどの共同で使用される構成状態レジスタは、隣接していない。さらに、構成状態レジスタのグループは、同じキャッシュ・ライン（例えば、779 MMCR0、780 SIAR、781 SDA R、782 MMCR1）内にあることを保証するために、アライメント境界上にない。さらに、一部の構成状態レジスタは、同じレジスタ（例えば、異なるアクセス許可、サブフィールドなど）を参照することがある。これは、非効率的なキャッシュの使用である。（各活動で1回のキャッシュ・ミスのみが発生することを保証するための）プリフェッチが欠如しており、大きすぎるキャッシュ・フットプリントが存在する（その結果、ヒット率を減らすワーキング・セットが増加する）。したがって、本発明の態様に従って、構成状態レジスタは、例えばベース+（idx*8）に格納されない。構成状態レジスタは、例えばベース+再マッピング[idx]に格納される。

【0130】

この再マッピングは、キャッシュ・ラインを共有するために、グループが隣接することを保証し、スパース性を除去／低減して、より効率的なキャッシュの使用を実現し、複数の名前を処理する。1つの例として、この再マッピングは静的であり、プロセッサの設計で実行され、テーブルなどのデータ構造において提供されるか、または定義された方程式の計算によって提供される。別の例として、この再マッピングは動的であり、使用によって決定される。例えば、レジスタの追跡が、レジスタのセットのうちの複数のレジスタと一緒に使用されることを示した場合、それらのレジスタがグループ化され、互いに隣接して配置される。その他の可能性が存在する。

【0131】

再マッピングの詳細が、図22を参照してさらに説明される。この処理は、プロセッサによって実行される。1つの例では、構成状態レジスタ番号がプロセッサによって取得される（ステップ1600）。構成状態レジスタ番号に基づいて、メモリ・ユニット（例えば、ページ）へのインデックス位置（オフセットとも呼ばれる）が決定される（ステップ1602）。このインデックス位置は、テーブル検索によって、または計算によって決定されてよい。インデックス位置が要求側（例えば、内部動作）に返される（ステップ1604）。

【0132】

さらに別の例では、マッピング論理を含んでいる回路が使用される。構成状態レジスタ番号がマッピング論理に入力され、出力はページ・インデックスである。

【0133】

前述したように、1つの実施形態では、より効率的なキャッシュを可能にするために、一緒に使用される構成状態レジスタが互いに近接して配置されるように、アーキテクチャで定義された構成状態レジスタ番号が再マッピングされる。これによって、構成状態レジスタに使用されるキャッシュ・ラインの数を減らし、キャッシュの使用に関して、他のプログラムとの競合を少なくする。さらに、これによって、特定の構成状態レジスタと共にキャッシュ・ラインが読み込まれ、その値のキャッシュ・ミスに対してペナルティが支払われた後に、その構成状態レジスタと連動して使用されてもよい他の構成状態レジスタが、キャッシュ内でヒットし、その結果、他のキャッシュ・ミス・ペナルティを伴わない、ということを保証する。

【0134】

さらに別の実施形態では、異なるオフセット割り当てが可能である場合に、`sync__o__csr` 命令がバージョン・スタンプをページ内に書き込む。バージョン・スタンプは、異なるホスト間でパーティションを移行するときにオフセットを調整するため、またはオフセットを（例えば、`sync__i__csr` がパーティションのオフセットのバージョン番号を読み取るときに）ハードウェア内で直接調整するか、もしくはソフトウェア内で調整するため、あるいはその両方のために、使用されてよい。

【0135】

本発明の別の態様に従って、複数の構成状態レジスタを格納するか、または読み込む一括動作を実行するための能力が提供される。構成状態レジスタの個別の読み込みまたは格納は、各読み取りまたは書き込みが順序通りに実行され、次の命令が開始される前に完了するため、高価になる。さらに、プロセッサ内構成状態レジスタの名前変更の欠如がロールバックを許可しないため、正しい例外／エラー・シーケンスが保証される。しかし、本発明の1つまたは複数の態様に従って、メモリ・ユニット（ページなど）からの構成状態レジスタの一括読み込みおよび一括格納が提供される。

【0136】

例えば、構成状態レジスタ格納命令または動作（例えば、`ST__CSR`）が、複数のプロセッサ内構成状態レジスタをメモリに格納する（すなわち、現在のコンテキスト（例えば、アプリケーション、スレッドなど）に関連付けられた構成状態レジスタの内容を、特定の構成状態レジスタに対して定義された、選択されたメモリ位置に格納する）ために使用され、構成状態レジスタ読み込み命令または動作（例えば、`LD__CSR`）が、メモリに格納された構成状態レジスタをプロセッサに再び読み込む（すなわち、現在のコンテキストに関連付けられた構成状態レジスタの内容をメモリからプロセッサに再び読み込む）ために使用される。

【0137】

メモリに格納されてよいプロセッサ内（メモリによって補助されていない、とも呼ばれる）構成状態レジスタには、メモリ内（例えば、補助メモリ・ユニット（例えば、ページ）内）の位置が割り当てられる。1つの例では、メモリ位置またはメモリ・ユニットあるいはその両方は、明確に定義された定義済みの位置／ユニットであり、したがって命令は、位置／ユニットを指定するためのオペランドを含む必要がない。別の実施形態では、特定の位置／ユニットは、命令のオペランドとして指定されてよい。さらに別の実施形態では、各命令が、格納される／読み込まれる特定のレジスタを示すためのオペランドを含んでよい。その他の変形も可能である。

【0138】

さらに、ページは、メモリ・ユニットの1つの例にすぎない。その他のユニットが可能である。また、ページは通常4KBであるが、他の実施形態では他のサイズであってよい。多くの可能性が存在する。

【0139】

前述した`ST__CSR`命令および`LD__CSR`命令に加えて、本発明の態様に従って使用されてよい別の命令は、`mtspr__TCBR`、`next__u->csr__page`である。この命令は、特定のコンテキスト（例えば、プロセッサ、スレッドなど）に関するメモリ内構成状態レジスタをレジスタ（`TCBR`など）に格納するために使用されるメモリ領域のベース・アドレスを読み込むために使用される。その後、このアドレスは、本明細書において説明されているように、ベース・アドレスを採用する処理において使用される。この命令では、`next__u->csr__page`は、命令を発行するコンテキストに関するデータを格納するユーザ・データ構造を参照する。このデータは、メモリ内構成状態レジスタを格納するメモリ・ユニット（例えば、ページ）のベース・アドレスを含む。`mtspr`が指定されているが、他の構成状態レジスタへの移動（`mtcsr`）命令が使用されてよい。また、`TCBR`はベース・レジスタの1つの例にすぎない。他のベース・レジスタが指定されてよい。多くの変形が可能である。

【0140】

上記の命令に加えて、キャッシュをメモリまたはプロセッサ内レジスタと同期させるための2つの同期命令が提供されてよい。例えば、`sync_o_csr`は、キャッシュおよび1つまたは複数のメモリ内構成状態レジスタを同期させるために使用され、`sync_i_csr`は、キャッシュおよび1つまたは複数のプロセッサ内構成状態レジスタを同期させるために使用される。

【0141】

本明細書において説明されているように、1つの態様では、複数の構成状態レジスタが読み込まれるか、または格納される。介在する例外は存在せず、すなわち、例外が回避され、動作が、動作のすべての構成状態レジスタに対して完了するか、またはどの構成状態レジスタに対しても完了しない。さらに、ページ・フォールトが存在しない（例えば、ページが固定され、ページが読み込まれた後でも、同じページへの参照が保証される）。必要に応じて、ハードウェアが、プロセスを再開可能にすることができる（例えば、マイクロコードまたは状態マシン内の読み込みまたは格納シーケンス）。

【0142】

図23～図25を参照して説明されているように、複数の構成状態レジスタを格納する／読み込む命令を使用することによって、順序通りの位置を保証し、命令を順序通りに完了するなどの、特定の高価な動作があまり頻繁に使用されなくなる。図23～図25の処理は、プロセッサによって実行される。

【0143】

図23を参照すると、複数の構成状態レジスタをメモリに格納するために個別の命令（すなわち、一括動作ではない）を使用する1つの例が、図23を参照して説明される。1つの例では、構成状態レジスタをメモリに移動するために、構成状態レジスタからの移動命令および格納命令が使用される。例えば、1つの例では、構成状態レジスタがSPRであり、構成状態レジスタからの移動命令が`mfspr`である。他の構成状態レジスタおよび対応する命令が使用されてよい。

【0144】

この例では、SPRからの移動（`mfspr`）命令などの`mfspr`命令の実行に基づいて、順序通りの位置が保証される（ステップ1700）。その後、命令によって指定された構成状態レジスタ（例えば、SPR）の内容が、構成状態レジスタから、例えば汎用レジスタ（GPR：general purpose register）にコピーされる（ステップ1702）。命令が順序通りに完了する（ステップ1704）。その後、格納命令（STD）を介して、汎用レジスタの内容がメモリに格納される（ステップ1706）。この同じプロセスが、多数の構成状態レジスタであってよいメモリに格納される構成状態レジスタごとに、繰り返される（例えば、ステップ1708～1722）。したがって、メモリに格納される構成状態レジスタごとに、順序通りの位置を保証し、順序通りに命令を完了する動作が必要になる。

【0145】

しかし、本発明の態様に従って、図24を参照して説明されているように、順序通りの位置を保証する単一の動作および順序通りに命令を完了する単一の動作を使用して、複数の構成状態レジスタをメモリに格納する構成状態レジスタ格納（STCSR）命令が提供される。

【0146】

図24を参照すると、構成状態レジスタ格納命令（STCSR）の実行に基づいて、順序通りの位置に達する（ステップ1750）。次に、選択された構成状態レジスタの内容が一時的レジスタに読み込まれる（ステップ1752）。さらに次に、一時的レジスタの内容がメモリ（例えば、メモリ制御ページ（memory control page））に格納される（ステップ1754）。1つまたは複数の追加の構成状態レジスタに対して、読み込み／格納動作が、1回または複数回、繰り返される（1756～1762）。選択された構成状態レジスタ（多数のそのようなレジスタであってよい）をメモリにコピーした後に、命

10

20

30

40

50

令が順序通りに完了する（ステップ1770）。

【0147】

1つの例では、ST_CSR命令が、コピーされるレジスタを指定するためのオペランドを含まず、代わりに、現在のコンテキスト（例えば、プロセス、スレッドなど）のプロセッサ内構成状態レジスタのすべてがコピーされる。別の例では、メモリにコピーされる1つまたは複数の構成状態レジスタを指定するためのオペランドが含まれ、使用されてよい。その他の変形も可能である。

【0148】

さらに別の例では、一括読み込み動作（例えば、LD_CSR）を使用して、複数の構成状態レジスタがメモリからプロセッサ内にコピーされてよい。

10

【0149】

図25を参照すると、構成状態レジスタ読み込み（LD_CSR）命令の実行に基づいて、順序通りの位置が保証される（ステップ1780）。次に、選択された構成状態レジスタの内容がメモリから取得され、一時的レジスタに読み込まれる（ステップ1782）。その後、一時的レジスタの内容が、対応するプロセッサ内構成状態レジスタに格納される（ステップ1784）。1つまたは複数の（場合によっては、多数の）追加の構成状態レジスタに対して、読み込み／格納動作が、1回または複数回、繰り返される（1786～1792）。その後、命令が順序通りに完了する（ステップ1794）。

【0150】

1つの例では、LD_CSR命令が、コピーされるレジスタを指定するためのオペランドを含まず、代わりに、現在のコンテキスト（例えば、プロセス、スレッドなど）の構成状態レジスタのすべてがコピーされる。別の例では、メモリからコピーされる1つまたは複数の構成状態レジスタを指定するためのオペランドが含まれ、使用されてよい。

20

【0151】

1つの態様では、一括動作を実行するための命令が、構成状態レジスタのグループに対して同じ動作（例えば、格納、読み込みなど）を実行してよく、このグループは共通の特徴によって定義される。この共通の特徴は、例えば、レジスタの数値範囲、同じアクセス許可またはコンテキスト（例えば、ユーザ、オペレーティング・システム、ハイパーバイザ）を有していること、同じ機能的目的（例えば、例外処理、時間管理など）を有していること、または同じ実装の特性（例えば、構成状態レジスタのセットがメモリに格納されている）を有していること、などであってよい。

30

【0152】

一括格納または一括読み込みあるいはその両方の動作の使用は、コンピュータ内の処理を改善する。例えば、一括格納動作を使用して複数の構成状態レジスタをメモリに効率的にコピーすることによって、コンテキスト切り替え処理がより高速かつ効率的に実行されてよい。その他の恩恵が実現されてもよい。

【0153】

さらに別の態様では、処理を容易にするために、構成状態レジスタが格納されるメモリ内の位置を示すアーキテクチャの配置制御が提供されてよい。例えば、ハードウェアが、構成状態レジスタが格納されるメモリ内の位置を識別するための制御のセットを定義する。1つの例として、アプリケーションの状態を格納するためのベース・アドレスを指定する少なくとも1つの構成状態レジスタが、提供される。1つの例として、ベース・アドレスがゲスト物理アドレスであり、すなわち、ゲスト・オペレーティング・システムが、それ自身のアドレス空間内のアドレスを指定する。例えば、アドレスが指定される場合、ゲストがホスト物理アドレスを指定できるようにすると、仮想化およびセキュリティを損なうことがあるため、ゲスト・レベルのアドレス（例えば、ゲスト実アドレス、ゲスト物理アドレス、またはゲスト仮想アドレス）が指定される。

40

【0154】

アーキテクチャの構成制御の指定に関する詳細が、図26を参照してさらに説明される。1つの例では、プロセッサがこの論理を実行する。最初に、現在の実行環境（例えば、

50

アプリケーションの状態、スレッドの状態、オペレーティング・システムの状態、ハイパーバイザの状態、特定のゲスト・オペレーティング・システムまたはホスト・オペレーティング・システムのレベルなど）に関するメモリ補助の位置を示すアドレス（すなわち、ベース・アドレス）が受信される（ステップ1800）。例えば、仮想マシン／論理パーティションに関して、ハイパーバイザはハイパーバイザのアドレス（例えば、ホスト仮想アドレスまたはホスト絶対アドレス、ホスト物理アドレス、ホスト実アドレス）を使用し、オペレーティング・システムはゲスト実アドレスまたはゲスト仮想アドレスを使用する。プロセッサは、構成状態レジスタを格納するためのメモリ・ページの位置を示すアドレス（すなわち、ベース・アドレス）を取得する。

【0155】

任意選択的に、ベース・アドレスが物理アドレスに変換される（ステップ1802）。（変換がすでに実行されていることがあり、またはアドレスを変換する必要がない）。1つの実施形態では、この変換がページ・フォールトを引き起こすことがある。さらに別の実施形態では、ページ・フォールトを防ぐために、本手法の1つまたは複数の態様を実行する前に、例えばページを固定することによって、連携するハードウェアおよびソフトウェアが使用される。

【0156】

さらに、1つの例では、変換されたアドレスが捕捉される（ステップ1804）。すなわち、1つの例では、変換されたアドレスがキャッシュされる。別の実施形態では、変換されていないアドレスおよび変換されたアドレスの両方が格納される。

【0157】

図27を参照する別の実施形態では、参照されているメモリ制御（例えば、構成状態レジスタ）に関する捕捉されたアドレスが取得される（ステップ1850）。さらに、このアドレスが物理アドレスに変換され（ステップ1852）、メモリ内の制御（例えば、構成状態レジスタ）がアクセスされる（ステップ1854）。この変換はページ・フォールトを引き起こすことがあるが、連携するハードウェアおよびソフトウェアを使用して、例えば本手法の1つまたは複数の態様を実行する前にページを固定することによって、ページ・フォールトを防ぐことができる。

【0158】

1つの実施形態では、循環する依存関係を防ぐために、ベース・アドレスが、例えばメモリではなく、構成状態レジスタに格納される。その他の例が可能である。

【0159】

さらに、1つの実施形態では、ベース・アドレスが物理アドレス／実アドレスに変換され、別の実施形態では、ベース・アドレスが次のレベルのスーパーバイザのアドレスに変換される（すなわち、オペレーティング・システムがページ・アドレスを設定するときに、ベース・アドレスがスーパーバイザのアドレスに変換される）。その他の例が可能である。

【0160】

1つの例として、変換されていない（物理／実）ベース・アドレスおよび変換された（物理／実）ベース・アドレスの両方が格納される。これによって、すべての構成状態レジスタのアクセスでアドレス変換（例えば、動的アドレス変換（DAT: dynamic address translation））を実行する必要がなくなり、ページ・フォールトを処理する必要がなくなる。

【0161】

1つの実施形態では、変換された（実／物理）ベース・アドレスがプロセッサのレジスタ内で維持され、変換されていないベース・アドレスがメモリ内構成状態レジスタ内で維持される。そのような実施形態では、構成状態レジスタのベース・アドレスの読み出しのソフトウェア要求にตอบสนองして、変換されていないアドレスが再度提供される。変換されたアドレスは、メモリ内の位置からそのようなアドレスにアクセスするために使用されてよい。その他の可能性が存在する。

10

20

30

40

50

【0162】

本明細書において説明されているように、1つまたは複数のメモリ内構成状態レジスタが格納されるメモリ内の位置を指定するベース・アドレスを含んでいる、構成状態レジスタなどの制御が提供される。これらのメモリ内構成状態レジスタは、プロセッサ内レジスタになるようにアーキテクチャによって定義されたレジスタであるが、本発明の1つまたは複数の態様に従って、メモリ内構成状態レジスタに変換されている。少なくとも、メモリに格納される構成値がレジスタではなく、メモリ内に存在するようにアーキテクチャによって定義されているため、これらのメモリ内構成状態レジスタは、メモリに格納される構成値とは異なっている。メモリに格納される構成値は、アーキテクチャによって定義されたプロセッサのレジスタではない。

10

【0163】

さらに別の態様では、プログラム環境と、例えばオペレーティング・システム環境もしくはその他のスーパーバイザ環境との間、または異なるアプリケーションもしくはスレッド間、あるいはその両方などでのコンテキスト切り替え時に、効率が達成される。コンテキスト切り替えが実行されるときに、以前のコンテキストのデータが保存される。1つの例では、コンテキスト・データを保存するために、構成状態レジスタの内容がプロセッサ内レジスタから読み出されて、メモリに保存される。次に、実行を再開するために構成状態レジスタを読み込むことを含めて、次のコンテキストのデータが読み込まれる。これは高価なプロセスである。

【0164】

20

格納キューに基づく投機および不規則な順序での実行がこのプロセスを加速する、メモリ内構成状態レジスタの実行を伴う本発明の態様に従っても、コンテキストの保存および復元に関連する大きなコストが、やはり存在する。

【0165】

したがって、本発明の態様に従って、構成状態レジスタのページ操作によってコンテキスト切り替えが実行される。1つの例では、構成状態レジスタのメモリ内格納の位置が構成可能である。コンテキストを切り替えるときに、古い構成状態レジスタのデータを構成状態レジスタのメモリ・ユニット（またはページ）からコピーし、新しいデータを構成状態レジスタのページにコピーする代わりに、異なる構成状態レジスタのメモリ・ユニット（またはページ）が選択され、それによって、プロセッサから見える構成状態レジスタの値を変更する。

30

【0166】

本発明の態様に従って、構成状態レジスタのアンロードおよび再読み込みが必要となるのを避けるために、1つまたは複数の構成状態レジスタのメモリ内の位置（本明細書ではCSR補助メモリと呼ばれる）を示す、例えばベース構成状態レジスタ（本明細書ではベース・レジスタと呼ばれる）に含まれているベース・ポインタまたはベース・アドレス（本明細書ではベースと呼ばれる）を変更することによって、コンテキスト切り替えが実行される。

【0167】

この態様から恩恵を受けることができるコンテキスト切り替えは、オペレーティング・システムが異なるアプリケーション間で切り替えを実行する、オペレーティング・システムのコンテキスト切り替え、ハイパーバイザまたは仮想マシン・モニタが異なるパーティションまたは仮想マシンを切り替える、ハイパーバイザのコンテキスト切り替え、および異なるハードウェア・スレッド間のハードウェア・スレッドのコンテキスト切り替えを含めて、複数の種類が存在する可能性がある。各コンテキスト切り替えは、異なるレジスタに影響を与える。例えば、オペレーティング・システムによって使用されるコンテキスト切り替えの一部としてアプリケーションを切り替えるときに、そのアプリケーションに対応するいくつかの構成状態レジスタが別の位置に変更されるが、他の構成状態レジスタは変更されない（例えば、オペレーティング・システムの構成状態レジスタは変更されない）。さらに、ハイパーバイザのコンテキスト切り替えでは、切り替えられるさらに多くの

40

50

レジスタが存在することがある。ハードウェア・スレッドのコンテキスト切り替えの場合も同様である。さらに、以下では、ハードウェア・スレッドのコンテキスト切り替えに関する詳細が説明される。

【0168】

1つの実施形態では、ハードウェア・スレッドのコンテキスト切り替えの場合、プロセッサが、スレッド・スケジューラを使用して、プロセッサに読み込まれる複数のスレッドから選択する。しかし、本発明の態様に従って、ハードウェアが、ハードウェアによってスケジューリングされ得る複数のスレッドから選択してよく、この複数のスレッドの数は、プロセッサに読み込まれるハードウェア・スレッドのコンテキストの数を超える。すなわち、本発明の態様に従って、本明細書において説明されているように、コンテキスト切り替えの能力は、ハードウェアが、プロセッサに読み込まれるスレッドよりも多くのスレッドを使用できるようにする。スレッドが選択され、ハードウェアが、そのスレッドのメモリ内構成情報を選択することによって、スレッドをスケジューリングする。1つの実施形態では、実行可能なレジスタの一部が、チップ上のレジスタ・ファイルまたは高速な第2のレベルのストレージに格納される。別の実施形態では、汎用レジスタも、メモリ内の構成ページに格納され、スレッドがスケジューリングされるときに、そこから読み込まれる。これは、必要性が生じたときに（例えば、最初にアクセスされたときに、各レジスタに対して）、または一括して（例えば、スケジューリングされた時間にすべてのレジスタに対して）、実行される。

【0169】

1つの実施形態では、ソフトウェア・エージェント（例えば、オペレーティング・システムまたはハイパーバイザ）に、別の構成状態レジスタのベースへのポインタを変更することを決定させるのではなく、ハードウェアが、ハードウェアの基準に応答して、ベース・ポインタ自体を調整する。ハードウェアにおいて複数のスレッドのうちの1つが選択され、複数のスレッドに基づいて、構成状態レジスタを含んでいるシステム・メモリ・ページへのポインタのうちの1つが選択される。

【0170】

コンテキスト切り替えを実行することに関連する詳細が、図28および図29を参照してさらに説明される。図28は、データがコピーされるコンテキスト切り替えを実行するための1つのプロセスを示しており、図29は、本発明の態様に従ってポインタが変更されるコンテキスト切り替えを実行するための別のプロセスを示している。

【0171】

最初に図28を参照すると、コンテキスト切り替えが実行される場合、コンテキストのコピーアウトが、古いコンテキスト・データをコピーアウトするように初期化される（ステップ1900）。この初期化は、以前のコンテキストのコンテキスト構造（例えば、スーパーバイザ構造）の位置を特定し、読み出されるコンテキストの第1の構成状態レジスタを識別することを含む。

【0172】

例えば、コンテキストは、仮想マシン、論理パーティション、プロセス、スレッドなどのうちの1つであることができる。コピーアウト・プロセスが続行する。

【0173】

格納される構成状態レジスタが選択される（ステップ1902）。この反復では、この構成状態レジスタは、上で識別された構成状態レジスタである。選択された構成状態レジスタが読み取られ（ステップ1904）、コンテキスト切り替えデータを格納するために、構成状態レジスタの内容が、例えばスーパーバイザ・ソフトウェア（またはハードウェア）によって維持されるメモリ内の構造に格納される（ステップ1906）。次に、コンテキストが切り替えられる構成状態レジスタがさらに存在するかどうかに関する判定が行われる（照会1908）。そのような構成状態レジスタが存在する場合は、処理がステップ1902を続行する。

【0174】

以前のコンテキストのデータをコピーアウトした後に、新しいコンテキストに関して、コピーイン・プロセスが実行される。したがって、コンテキストのコピーインが初期化され、次のコンテキストのコンテキスト構造の位置が特定され、コンテキスト書き込み用の第1の構成状態レジスタが識別される（ステップ1910）。メモリに読み込まれる構成状態レジスタが選択され（ステップ1912）、選択された構成状態レジスタの内容がコンテキスト構造（例えば、スーパーバイザ構造）から読み取られる（ステップ1914）。読み取られたコンテキスト・データが、メモリ内の構成状態レジスタに書き込まれる（ステップ1916）。コンテキストが切り替えられる構成状態レジスタがさらに存在するかどうかに関する判定が行われる（照会1920）。そのような構成状態レジスタが存在する場合は、処理がステップ1912を続行する。そのような構成状態レジスタが存在しない場合は、処理が完了する。

10

【0175】

図29を参照すると、本発明の態様に従って、コンテキスト切り替えプロセスの別の例が説明されている。この例では、データをコピーする代わりに、データへのポインタが操作される。最初に、コンテキストのコピーアウトが初期化され、以前のコンテキストのコンテキスト構造（例えば、スーパーバイザ構造またはハードウェア構造）の位置が特定される（ステップ1950）。この場合も、コンテキストは、仮想マシン、論理パーティション、プロセス、スレッドなどのうちの1つであることができる。次に、レジスタの内容がメモリに格納されるように、プロセッサ内構成状態レジスタが処理される（ステップ1952）。1つの実施形態では、この処理は、図28を参照して説明されているように、例えばコピー・ループを使用して実行される。別の実施形態では、この処理は、例えば一括コピー動作（例えば、STCSR）を使用して実行される。

20

【0176】

メモリ内構成状態レジスタのデータ・ユニットのアドレス（すなわち、例えば、このコンテキストの構成状態レジスタを格納するために使用されるメモリ・ページのアドレス（ベース・アドレス））が読み取られ（ステップ1954）、コンテキスト構造に格納されてよい（ステップ1956）。1つの例では、このアドレスが変化せず、そのため、このアドレスを繰り返し読み取ってコンテキスト構造に格納する必要がない。代わりに、このアドレスは、最初に、またはそのページが新しい位置に移動されたときに、格納される。この値は、コンテキスト切り替えデータを格納するために、例えばスーパーバイザ・ソフトウェアまたはハードウェアによって維持されるメモリ内の構造に格納される。

30

【0177】

コピーアウト・プロセスの実行後に、新しいコンテキスト・データを指し示すため、コピーイン・プロセスが利用される。したがって、コンテキストのコピーイン・プロセスが初期化され、次のコンテキストのコンテキスト構造（例えば、スーパーバイザ構造またはハードウェア構造）の位置が特定され、コンテキスト書き込み用の第1の構成状態レジスタが識別される（ステップ1960）。次のコンテキストの構成状態レジスタのデータ・ユニットのアドレスが読み込まれる（ステップ1962）。すなわち、例えば、新しいコンテキストの構成状態レジスタを格納するためのメモリ・ページのアドレス（ベース・アドレス）が取得される。さらに、メモリ内構成状態レジスタのデータ・ページのアドレス（ベース・アドレス）が書き込まれる（ステップ1964）。さらに、プロセッサ内構成状態レジスタが処理される（ステップ1966）。1つの例として、例えばコピー・ループまたは一括読み込み（例えば、LDCSR）を使用して、このコンテキストのプロセッサ内構成状態レジスタがメモリから読み込まれる。

40

【0178】

前述したように、コンテキスト切り替えにおいてポインタ操作が使用されてよい。これは、仮想マシンの移行にも当てはまる。

【0179】

1つの実施形態では、仮想マシン（または論理パーティション）の移行をまたはライブ・マシンの移行あるいはその両方を加速するために、メモリ内レジスタが使用されてよい

50

。1つの例では、従来手法に従ってページが移行されるが、メモリ内構成状態レジスタが移動されない。1つの例として、プロセッサ内構成状態レジスタを捕捉するために `ST_CSR` 命令または動作が使用されるが、構成状態レジスタが移動されない。メモリ内の構成のメモリ・ページは移動される。

【0180】

さらに別の実施形態では、ライブ・マシンの移行の場合、マシンが休止されているときに、メモリ内構成状態レジスタの状態が移動される。複数のコンテキスト（例えば、複数のスレッド／プロセスなど）が存在する場合、この移動が複数のページを含んでよい。

【0181】

1つの実施形態では、ホストの移行形式およびターゲットの移行形式に互換性がない場合（例えば、構成状態レジスタが、アーキテクチャの異なる実装によってメモリ内構成状態レジスタのページ内の異なるオフセットにマッピングされる場合）、移行エージェントによって調整が実行される。1つのそのような実施形態では、構成状態レジスタ形式のアーキテクチャまたはマイクロアーキテクチャのバージョン番号が提供され、受信システムがレイアウトを調整する責任を負う。別の実施形態では、受信システムおよび送信システムが転送形式をネゴシエートする。別の実施形態では、転送形式が、例えば構成状態レジスタの値の線形リストまたは<キー，値>の対として定義され、このキーは構成状態レジスタ番号であり、この値は構成状態レジスタの値である。

【0182】

さらに他の実施形態では、プロセッサが、外部で構成された構成状態レジスタのレイアウト・マップ（例えば、レイアウト識別子を構成状態レジスタに読み込むことによって、ソフトウェアが指定する）に基づいて再マッピング論理を適応させて、複数のバージョンのレイアウトをサポートしてよい。

【0183】

少なくとも一部のプロセッサの実施形態では、設計によって、一部の値を、メモリ内レジスタとして識別されたレジスタのカスタムのプロセッサ内の位置にキャッシュしてよい。したがって、コンテキスト切り替えを実行するときに、キャッシュされたコピー（すなわち、個別の古いキャッシュされた値）が同期される。1つの例では、古いキャッシュされた値を無効化することをプロセッサに示すコンテキスト同期命令（`csync`）が提供される。この命令は、例えば、コンテキスト切り替えが実行されるたびに実行される。1つの例では、コンテキストのすべての構成状態レジスタに対応するキャッシュされた値が無効化される。他の例では、特定のレジスタのキャッシュされた値が無効化される。

【0184】

本明細書において説明されているように、コンテキスト切り替え時に、構成状態レジスタをコピーする代わりに、新しいページが示されてよい。このようにして、構成状態レジスタのコンテキスト・イメージがすでに保存されているため、少なくともメモリ内で維持されている構成状態レジスタに関して、コンテキスト切り替えを減らす。

【0185】

さらに、1つの実施形態では、ページ置き換えを伴うコンテキスト切り替えが有効化される。新しいコンテキストのメモリ・ページを読み込む前に、（少なくとも、メモリ内に存続しているレジスタに関して）メモリ・イメージが同期される。

【0186】

1つの実施形態に従って、構成状態レジスタ同期命令が提供される。具体的には、この命令は、すべてのキャッシュされた値をフラッシュし、切り替えが実行された後にキャッシュを抑制してよく、反対に、値をキャッシュに再び保存できることを示してよい。例示的な命令は次を含む。

【0187】

送信 CSR の同期 `sync__o__CSR`

【0188】

新しい補助ページの読み込み `mtspr TCBR, next_u->spr__pa`

10

20

30

40

50

g e

【0189】

受信SPRの同期 s y n c _ _ i _ _ C S R

【0190】

別の実施形態では、ベース・レジスタ（例えば、TCBR）への構成状態レジスタへの移動命令（例えば、mtspr命令）が、キャッシュされた送信値および受信値の同期を自動的に実行する。

【0191】

さらに別の態様では、1つまたは複数のメモリ内構成状態レジスタを格納するために使用されるメモリ内の位置を示す、ベース構成状態レジスタに格納されてよい、ゲスト・ベース・アドレス（guest base address）を読み込むことに基づいて、ページ・フォールトの可能性を防ぐために、ゲスト・ベース・アドレスから対応するホスト・ベース・アドレスへのアドレス変換が実行される。この変換は、例えば直ちに（すなわち、ベース・アドレスを受信することに基づいて、ベース・アドレスを使用する前に、例えばストレージの参照中に）実行される。例えば、仮想アドレスがベース・レジスタに読み込まれるときに（例えば、ゲスト・ベース・アドレスを、ベース・アドレスとして使用される構成状態ベース・レジスタ（例えば、TCBR）に読み込むことなどの、ベースへの読み込み）、システムが物理メモリ・アドレスへのアドレス変換を自動的に実行し、変換された物理アドレスが仮想アドレスと共に捕捉される。1つのアーキテクチャの実施形態に従って、ベース・レジスタへの読み込みが、変換の実行を引き起こすか、または引き起こす可能性のある命令として識別される。

【0192】

アドレスを変換できない場合、変換フォールトが選択される。例えばページ・テーブル・エントリ（下で説明される）で指定された許可に従って、ページが読み取りおよび書き込みのためにアクセス可能になるか、またはページ・フォールトが引き起こされる。

【0193】

実行されるベース・アドレスの変換を引き起こすか、または引き起こす可能性のある動作の実行に基づいてアドレス変換を自動的に実行することに関する詳細が、図30を参照してさらに説明される。この処理はプロセッサによって実行され、1つの例では、この処理は、構成状態レジスタ・メモリ・ベース・レジスタに対する構成状態レジスタへの移動命令（例えば、mtspr命令）の実行に基づいて実行される。この処理は、その他の命令の実行に基づいて実行されてもよい。

【0194】

この命令の実行に基づいて、現在の実行環境に関するメモリ補助の位置を示すアドレス（例えば、ベース・アドレス）が受信される（ステップ2000）。mtsprの1つの定義に従って、場合によっては動的アドレス変換（DAT）のページ・フォールトを招くように、mtsprが定義される。したがって、本発明の態様に従って、変換が必要とされるということを決定する前であっても、mtsprの一部としてアドレス変換が自動的に実行される。

【0195】

受信されたベース・アドレスが、例えばDATテーブルを使用して、物理ベース・アドレスに変換され（ステップ2002）、その例が下でさらに説明される。DAT変換フォールトが発生したかどうかに関する判定が行われる（照会2004）。DAT変換フォールトが発生した場合、DATページ・フォールトが示される（ステップ2006）。ページ・フォールト・ハンドラ・ソフトウェア・ルーチンに入り（ステップ2008）、ページ・フォールト・ハンドラが実行される（ステップ2010）。1つの例では、このフォールトが許容されるフォールト（例えば、ページ・アウト）である場合、ページ・フォールト・ハンドラに従って、命令が再開される。そうでない場合、コンテキストまたは実行環境（例えば、オペレーティング・システム、ハイパーバイザ、仮想マシン、スレッド、プロセスなど）がエラー指示を受信し、場合によってはコンテキストの終了を引き起こす。

【0196】

照会2004に戻り、DAT変換フォールトが存在しない場合、変換されていないベース・アドレスが捕捉される（例えば、レジスタに格納される）（ステップ2012）。さらに、変換されたベース・アドレスが捕捉される（例えば、キャッシュされる）（ステップ2014）。任意選択的に、対象のページが固定される（ステップ2016）。

【0197】

動的アドレス変換の一例に関する詳細が、図31～図32を参照してさらに説明される。この処理は、プロセッサによって実行される。

【0198】

動的アドレス変換は、仮想アドレスを対応する実（または絶対）アドレスに変換するプロセスである。動的アドレス変換は、CPUによって生成された命令アドレスおよびデータ・アドレスに対して指定されてよい。仮想アドレスは、一次仮想アドレス、二次仮想アドレス、AR（Access Register：アクセス・レジスタ）によって指定された仮想アドレス、またはホーム仮想アドレスであってよい。アドレスは、一次アドレス空間制御要素（ASCE：address space control element）、二次アドレス空間制御要素、ARによって指定されたアドレス空間制御要素、またはホーム・アドレス空間制御要素をそれぞれ用いて変換される。適切なアドレス空間制御要素の選択後の変換プロセスは、4種類の仮想アドレスのすべてで同じである。アドレス空間制御要素は、セグメント・テーブル指定または領域テーブル指定であってよい。セグメント・テーブル指定または領域テーブル指定は、実ストレージまたは絶対ストレージ内でオペレーティング・システムによって確立されたテーブルを用いて実行される変換を引き起こす。

【0199】

変換のプロセスにおいて、セグメント・テーブル指定または領域テーブル指定を使用するときに、情報の3種類の単位（領域、セグメント、およびページ）が認識される。それに応じて、仮想アドレスが4つのフィールドに分割される。1つの例では、64ビット・アドレスの場合、ビット0～32が領域インデックス（RX：region index）と呼ばれ、ビット33～43がセグメント・インデックス（SX：segment index）と呼ばれ、ビット44～51がページ・インデックス（PX：page index）と呼ばれ、ビット52～63がバイト・インデックス（BX：byte index）と呼ばれる。仮想アドレスのRXの部分は、それ自体が3つのフィールドに分割される。1つの実施形態では、ビット0～10が領域の第1のインデックス（RFX：region first index）と呼ばれ、ビット11～21が領域の第2のインデックス（RSX：region second index）と呼ばれ、ビット22～32が領域の第3のインデックス（RTX：region third index）と呼ばれる。

【0200】

仮想アドレスを実アドレスに変換する1つの例が、図31を参照して説明される。このプロセスは、本明細書ではDATウォーク（またはページ・ウォーク）と呼ばれ、DATウォークでは、アドレス変換テーブル内を進んで、あるアドレス（例えば、仮想アドレス）を別のアドレス（例えば、実アドレス）に変換する。この例では、アドレス空間制御要素（ASCE）2100がテーブル原点2102および指定タイプ（DT：designation type）制御2104を含んでおり、指定タイプ制御2104は、変換の開始レベルの指示（すなわち、階層内のどのレベルでアドレス変換が開始するかの指示）である。テーブル原点2102およびDT2104を使用して、特定のテーブルの原点が特定される。次に、このテーブルに基づいて、仮想アドレスのビットが、特定のテーブルのインデックスを参照して次のレベルのテーブルの原点を取得するために使用される。例えば、領域の第1のテーブル（RFT：region first table）2106が選択された場合、仮想アドレスのビット0～10（RFX）2108が、領域の第1のテーブルのインデックスを参照して領域の第2のテーブル2110の原点を取得するために使用される。次に、仮想アドレスのビット11～21（RSX）2112が、領域の第2のテーブル（RST：region second table）2110のインデックスを参照して領域の第3のテーブル2114の原点を取得するために使用される。同様に、仮想アドレスのビット22～32（RTX）211

6 が、領域の第3のテーブル（R T T : region third table）2 1 1 4 のインデックスを参照してセグメント・テーブル2 1 1 8 の原点を取得するために使用される。次に、仮想アドレスのビット3 3 ~ 4 3（S X）2 1 2 0 が、セグメント・テーブル2 1 1 8 のインデックスを参照してページ・テーブル2 1 2 2 の原点を取得するために使用され、仮想アドレスのビット4 4 ~ 5 1（P X）2 1 2 4 が、ページ・テーブル2 1 2 2 のインデックスを参照して、ページ・フレーム実アドレス（P F R A : page frame real address）2 1 2 6 を含んでいるページ・テーブル・エントリ（P T E : page table entry）2 1 2 5 を取得するために使用される。次に、このページ・フレーム実アドレスがオフセット2 1 2 8（ビット5 2 ~ 6 3）と結合されて（例えば、連結されて）、実アドレスを取得する。1つの実施形態では、次に接頭辞の付加が適用されて、対応する絶対アドレスを取得してよい。

10

【0 2 0 1】

アドレス変換の別の例が、図3 2を参照して説明される。この例では、D A T ウォークが実行されて、初期ゲスト仮想アドレスを最終的なホスト実アドレスに変換する。この例では、アドレス空間制御要素（A S C E）2 1 0 0 はゲスト・アドレス空間制御要素であり、A S C E 2 1 0 0 のD T 2 1 0 4 は、ゲスト・アドレス変換構造2 1 6 0 によって決定されたゲスト変換が、テーブル原点2 1 0 2 によって指し示された領域の第1のテーブル2 1 0 6 で開始することを示す。したがって、初期ゲスト仮想アドレス（例えば、R F X 2 1 0 8）の適切なビットが、領域の第1のテーブル2 1 0 6 のインデックスを参照して領域の第1のテーブルのエントリのポインタを取得するために使用される。領域の第1のテーブルのエントリ（R F T E : region first table entry）のアドレスは、ゲスト実アドレスまたはゲスト絶対アドレスである。このゲスト実アドレスまたはゲスト絶対アドレスは、主記憶装置の原点および制限が適用されて、適切な場合に、ホスト仮想アドレスに対応する。次に、ホスト・アドレス変換構造2 1 7 0 を使用して、この中間のホスト仮想アドレスが変換される。具体的には、アドレス空間制御要素（A S C E）2 1 5 0 は、ホスト・アドレス変換構造2 1 7 2 内の変換の開始レベルを示すために使用されるホスト・アドレス空間制御要素である。D T 2 1 5 4 によって示された開始レベル（例えば、領域の第1のテーブル）に基づいて、図3 1を参照して説明されたように、ホスト仮想アドレスの特定のビットが、テーブル原点2 1 5 2 で示されたテーブルのインデックスを参照するために使用され、このテーブルが、ホスト・アドレス変換2 1 7 2 を使用した変換に使用される。ゲストR F T E に対応するホスト仮想アドレスの変換は、ホスト・ページ・フレーム実アドレス（P F R A）2 1 7 4 a が取得されるまで継続する。

20

30

【0 2 0 2】

中間のホスト・ページ・フレーム実アドレスにあるデータは、ゲスト・アドレス変換構造（例えば、この特定の例では、ゲストの領域の第2のテーブル2 1 1 0）の次のレベルへのポインタであり、前述したように変換が継続する。具体的には、ホスト・アドレス変換構造2 1 7 6、2 1 7 8、2 1 8 0、および2 1 8 2 が、ゲストの領域の第2のテーブル2 1 1 0、領域の第3のテーブル2 1 1 4、セグメント・テーブル2 1 1 8、およびページ・テーブル2 1 2 2に関連付けられた中間のホスト仮想アドレスを変換するために使用され、その結果、ホストP F R A 2 1 7 4 b、2 1 7 4 c、2 1 7 4 d、および2 1 7 4 eをそれぞれ得る。ホスト・ページ・フレーム実アドレス2 1 7 4 eは、ゲスト・ページ・テーブル・エントリ2 1 2 5のアドレスを含む。ゲスト・ページ・テーブル・エントリ2 1 2 5は、ゲスト・ページ・フレーム実アドレス2 1 2 6を含み、ゲスト・ページ・フレーム実アドレス2 1 2 6が初期ゲスト仮想アドレスからのオフセットと連結されて、対応するゲスト絶対アドレスを取得する。場合によっては、次に、主記憶装置の原点および制限が適用されて、対応するホスト仮想アドレスを計算し、その後、前述したように、アドレス変換構造2 1 8 4を使用してこのホスト仮想アドレスが変換され、ホスト・ページ・フレーム実アドレス2 1 7 4 fを取得する。次に、ホスト・ページ・フレーム実アドレスがホスト仮想アドレスのオフセット（例えば、ビット5 2 ~ 6 3）と結合されて（例えば、連結されて）、最終的なホスト実アドレスを取得する。これによって、ホスト実ア

40

50

ドレスへのゲスト仮想アドレスの変換が完了する。

【0203】

上記の例では変換が領域の第1のテーブルで開始しているが、これは1つの例にすぎない。変換は、ゲストまたはホストのいずれかの任意のレベルで開始してよい。

【0204】

1つの実施形態では、アドレス変換を改善するために、仮想アドレスから実アドレスまたは絶対アドレスへの変換マッピングが、トランслэшヨン・ルックアサイド・バッファ（TLB：translation look-aside buffer）のエントリに格納される。TLBは、仮想アドレスの変換速度を改善するためにメモリ管理ハードウェアによって使用されるキャッシュである。次に仮想アドレスの変換が要求されたときに、TLBがチェックされ、その仮想アドレスがTLB内にある場合、TLBのヒットが存在し、実アドレスまたは絶対アドレスがTLBから取り出される。その仮想アドレスがTLB内にない場合、前述したようにページ・ウォークが実行される。

【0205】

示されているように、ゲスト変換がTLBに含まれてよい。これらのエントリは、1つまたは複数のホスト変換を暗黙的に含む複合ゲスト／ホスト・エントリであってよい。例えば、ゲスト仮想TLBエントリは、初期ゲスト仮想アドレスから最終的なホスト実アドレスまたはホスト絶対アドレスまでの変換全体をバッファしてよい。この場合、図32で前述したように、ゲストTLBエントリは、すべての中間のホスト変換2172、2176、2178、2180、および2182、ならびに最終的なホスト変換2184を暗黙的に含む。別の例では、階層的TLBが、初期ゲスト仮想アドレスからゲスト・ページ・テーブル2122の関連付けられた原点までの変換をバッファするTLBの第1のレベル内のエントリ、およびゲスト・ページ・テーブル・エントリのアドレスから最終的なホスト実アドレスまたはホスト絶対アドレスまでの変換をバッファするTLBの第2のレベルからの別のエントリを含んでよい。この例では、図32に示されているように、TLBの第1のレベル内のゲスト・エントリが、ゲスト領域およびセグメント・テーブルを補助するホスト変換に対応する中間のホスト変換2172、2176、2178、および2180を暗黙的に含み、第2のレベル内のゲスト・エントリが、ゲスト・ページ・テーブルを補助する中間のホスト変換2182および最終的なホスト変換2184を暗黙的に含む。トランслэшヨン・ルックアサイド・バッファの多くの実装が可能である。

【0206】

上記の例では、ページ・フレーム実アドレスが、ページ・テーブルのページ・テーブル・エントリに含まれる。ページ・テーブルは1つまたは複数のエントリを含み、ページ・テーブル・エントリの詳細が、図33を参照してさらに説明される。

【0207】

1つの例では、ページ・テーブル・エントリ（PTE）2200が、メモリの特定のページに関連付けられ、例えば以下を含む。

【0208】

（a）ページ・フレーム実アドレス（PFRA）2202：このフィールドは、実（例えば、ホスト実）ストレージ・アドレスの左端のビットを提供する。これらのビットが右側の仮想アドレスのバイト・インデックス・フィールドと連結された場合に、実アドレスが取得される。

【0209】

（b）ページ無効インジケータ（I：Indicator）2204：このフィールドは、ページ・テーブル・エントリに関連付けられたページが使用可能であるかどうかを制御する。このインジケータが0である場合、ページ・テーブル・エントリを使用することによって、アドレス変換が進む。インジケータが1である場合、ページ・テーブル・エントリを変換に使用することはできない。

【0210】

（c）ページ保護インジケータ2206：このフィールドは、ページへの格納アクセス

10

20

30

40

50

が許可されるかどうかを制御する。

【0211】

(d) 固定インジケータ2208：このフィールドは、本発明の態様に従って、このページが固定されるかどうかを示すために使用される。1つの例では、1が、このページが固定されることを示し、0が、このページが固定されないことを示す。

【0212】

ページ・テーブル・エントリは、本明細書に記載されたフィールドより多いフィールド、少ないフィールド、または異なるフィールド、あるいはその組合せを含んでよい。例えば、Power Architectureでは、PTEが、メモリの対応するブロックが参照されたかどうかを示す参照インジケータ、またはメモリの対応するブロックが格納されていることを示す変更インジケータ、あるいはその両方を含んでよい。その他の変形が可能である。

【0213】

本発明のさらに別の態様では、管理の柔軟性の向上を可能にするために、例えば、ホストおよびゲストの属性、コンテキスト、または実行環境（例えば、スレッドの状態、アプリケーションの状態、オペレーティング・システムの状態、ハイパーバイザの状態、特定のゲスト・オペレーティング・システムまたはホスト・オペレーティング・システムのレベルなど）、あるいはその組合せに基づいて、構成状態レジスタが分離されて割り当てられる。例えば、構成状態レジスタは、ハイパーバイザ、オペレーティング・システム、アプリケーション、スレッド番号、またはその他の実行環境などによって分離されてよい。

【0214】

特定の例として、ハイパーバイザ特権構成状態レジスタが、ハイパーバイザによって割り当てられたメモリのユニット（例えば、ページ）に格納され、オペレーティング・システム特権構成状態レジスタが、オペレーティング・システムのメモリのユニット（例えば、ページ）に格納される、などとなる。さらに、複数のスレッドがサポートされ、スレッドごとに構成状態レジスタが複製される場合、メモリの別々のユニット（例えば、ページ）がスレッドごとにサポートされてよい。そのような分離の例が、図34に示されている。

【0215】

図34に示されているように、構成状態レジスタの1つのセット2300がスレッドまたはプロセス2302によって使用され、構成状態レジスタの別のセット2304がオペレーティング・システム2306によって使用され、構成状態レジスタのさらに別のセット2308がハイパーバイザ2310によって使用され、構成状態レジスタのさらに別のセット2312がハイパーバイザ2314によって使用される。その他の例も可能である。

【0216】

1つの例では、特定の実行環境の構成レジスタは、その実行環境に対して静的に定義され、その実行環境によって読み取り可能または書き込み可能なレジスタを含む。さらに別の例では、レジスタが使用に基づいて動的に割り当てられる。その他の例も可能である。

【0217】

1つの実施形態では、別々のメモリ領域が（例えば、複数の割り当て可能な変換ユニットとして）、別々に制御可能な各実行環境（例えば、スレッド、プロセス、オペレーティング・システム、ハイパーバイザ）に割り当てられ、したがって、その実行環境に関連付けられた構成状態レジスタのセットが、そのメモリ領域に割り当てられる。1つの例では、一部の構成状態レジスタが複数の実行環境からアクセス可能であってよい（例えば、オペレーティング・システムから読み取りアクセス可能であり、ハイパーバイザから読み取り／書き込み（R／W：read/write）アクセス可能である）ため、論理的な所有権に基づいて構成状態レジスタが対応するメモリ領域に割り当てられる。

【0218】

異なる実行環境が異なる特権レベルを有していることがあるが、1つの態様では、特権レベルのレベルが高いほど、より低いレベルのアクセスを制御することができる。本明細書に記載されたLD_CSR命令およびST_CSR命令ならびに同期動作を使用して、

10

20

30

40

50

特定の特権レベルが指定されてよい。

【0219】

1つの例では、前述したように構成状態レジスタ番号が再マッピングされ、すなわち、各グループ化に関してレジスタを同じ位置に配置するように、構成状態レジスタ番号のインデックスが圧縮される。

【0220】

実行環境ごとにレジスタの特定の設定を提供することによって、コンテキスト切り替えを含む特定の種類の処理が容易になる。前述したように、実行環境に基づいてレジスタの特定の設定を割り当て、別々のメモリ・ユニットをそれらの設定に割り当てることによって、レジスタおよび、それらのレジスタを使用するプロセス（コンテキスト切り替えを含む）の管理を容易にする。

10

【0221】

本明細書において説明されているように、構成状態レジスタの状態をアンロードして再び読み込むのではなく、構成状態ベース・レジスタ内のベース・アドレス・ポインタを変更することによって、コンテキスト切り替えが実行されてよい。これを実現するために、ユーザ・モードの状態が、ユーザのコンテキストを切り替えるように、スーパーバイザの状態から独立して切り替え可能になり、オペレーティング・システムの状態が、仮想マシンを切り替えるように、ハイパーバイザの状態から独立して切り替え可能になり、ハードウェアごとのスレッドまたはサブプロセッサの状態が、それらが他のスレッド／サブプロセッサから独立して切り替えられる場合、独立して切り替え可能になる。別々のメモリ領域および別々に割り当て可能な構成状態レジスタが、これを容易にする。

20

【0222】

本発明の1つの態様に従って、別々の構成状態ベース・レジスタが、メモリ内構成状態レジスタの各グループ化の位置（ベース）を指定する。さらに、少なくとも1つの実施形態では、ベース・レジスタの各々に対するアクセス制御が、適切なアクセス許可を有する。例えば、オペレーティング・システムによって切り替えられるコンテキストの場合、オペレーティング・システム特権が、ベース・レジスタを変更するための最低限の前提条件であり、ハイパーバイザによって切り替えられるコンテキストの場合、ハイパーバイザ特権が、そのようなベース・レジスタを変更するための最低限の前提条件である、などとなる。

30

【0223】

さらに別の態様では、1つまたは複数の構成状態レジスタの格納（すなわち、CSR補助メモリ）を提供するホスト・レベルのソフトウェア（ハイパーバイザまたは仮想マシン・モニタなど）によるメモリ・ユニット（例えば、ページ）の移動を防ぐための能力が、提供される。1つの例では、この能力は、メモリ・ユニットを固定すること、およびCSR補助メモリの自動的固定の指示を提供することを含む。

【0224】

1つまたは複数の構成状態レジスタの格納を提供するメモリのベースを示す構成状態レジスタ（すなわち、TCBRなどのベース・レジスタ）が書き込まれる場合、アーキテクチャの仕様に従って、指示が現在のゲストのホストに提供される。少なくとも1つの実施形態では、この指示は例外に対応する。1つの実施形態では、例外タイプが、構成状態ベース・レジスタに対する書き込みイベントを示す。構成状態ベース・レジスタの変更の指示を受信することに対応して、少なくとも1つのホスト・スーパーバイザ・ソフトウェア（例えば、ハイパーバイザまたは仮想マシン・モニタ）が、ページ固定情報を更新するための動作を実行する。本発明の1つの態様では、固定情報を更新することが、固定されたCSR補助メモリ・ページのアドレスを記録すること、またはページに対応する固定インジケータを設定することを含む。別の態様では、固定情報を更新することが、特定の構成状態ベース・レジスタの以前に記録されたアドレスを、固定されたCSR補助メモリに対応する1つまたは複数の記録されたアドレスのプールから削除すること、または固定インジケータをリセットすることによって、以前に固定されたCSR補助メモリ・ページを固

40

50

定解除することをさらに含む。

【0225】

本発明の1つの態様に従って、これらの更新によって、1つまたは複数のホスト・レベルがCSR補助メモリをページ・アウトまたは移動しないことを保証し、それによって、キャッシュされたアドレス変換を無効化し、そうしなければ、構成状態レジスタに対する更新を引き起こし、ページ変換フォールトをもたらす。本発明の別の態様では、CSR補助メモリの位置の通知を提供し、いずれかのキャッシュされた変換を更新する機会を1つまたは複数のホストに与えることによって、固定情報が、CSR補助メモリの位置を移動するために使用されてもよい。

【0226】

さらに、複数のレベルのゲスト方式をサポートするために、構成状態レジスタを補助するメモリ（CSR補助メモリ）を指し示すベース・レジスタを初期化する命令が、ホストの複数のレベルで例外を他動詞的に引き起こして、適切な固定を保証するように、さらに指定されてよい。1つの実施形態では、ホストの1つのレベルのみが通知され、そのホストが、適切なときに、HCALL（ハイパーバイザ呼び出し）によって固定を引き起こす。

【0227】

構成状態レジスタのメモリ補助ページをメモリ内で固定することによって、構成状態レジスタがアクセスされるときに、ページ・フォールトが回避される。このページ・フォールトは、ソフトウェアによって予期されないことがあり、例えば、ソフトウェアが、どの命令がトラップを引き起こしたかをチェックし、トラップを引き起こした命令が、データ・ページ・フォールトを引き起こしたメモリにアクセスするように定義されていない命令だったことを検出したなどの場合に、一部のソフトウェアにおいてpanic（）を引き起こすことがある。panic（）は、予期されないイベントが発生したときにオペレーティング・システムにおいて実行される呼び出しであり、通常はシステム・クラッシュをもたらす。

【0228】

固定は、例えば、循環する例外を回避するため（例えば、例外に関連する構成状態レジスタに使用されるページが使用可能でない場合、そのページに関するページ・フォールト例外が引き起こされる必要がある、など）、および（例えば、構成状態レジスタの処理を伴う例外およびその他の外部イベントに対する）高速な応答を保証するためにも、使用される。

【0229】

1つの実施形態では、固定はソフトウェアにおいて実行される。例えば、下で説明されているように、固定は、準仮想化された環境内でのソフトウェアのコンテキスト切り替えと連動して、ハイパーバイザ呼び出し（HCALL）を使用して実行されてよい。

【0230】

1つの実施形態では、コンテキスト（例えば、スレッドのコンテキスト、またはプロセスのコンテキスト、または論理パーティションのコンテキスト、または仮想マシンのコンテキスト、またはオペレーティング・システムのコンテキストなど）が、スーパーバイザ・ソフトウェアによって初期化されるときに、スーパーバイザが、初期化されているコンテキストに対応する1つまたは複数の構成状態レジスタの格納を提供するために、メモリを割り当てる。

【0231】

1つの例では、この割り当ては、適切なアライメントおよびサイズのメモリを提供する割り当てルーチンを呼び出すことによって実行されてよい。少なくとも1つの実施形態に従って、返されたアドレスが、コンテキストに対応する情報を格納するメモリ領域に格納される。1つの実施形態では、このメモリ領域は「u領域」と呼ばれ、変数「u」によって示される。少なくとも1つの実施形態では、変数uは、コンテキストに関して記録されるさまざまな属性に対応する複数のメンバーを含む、レコード、構造体、クラス、またはその他の複合データ・タイプである。少なくとも1つの実施形態では、この構造体は、少

10

20

30

40

50

なくとも1つのCSR補助メモリ・ページのアドレスに対応するメンバー（フィールド）を含む。少なくとも1つの例では、このメンバーに、「`csr_page`」という名前が付けられる。

【0232】

`my_csr_page_pointer` = 補助ページの割り当て

【0233】

`u.csr_page` = `my_csr_page_pointer`

【0234】

準仮想化された環境内で、例えばハイパーバイザに対するHCALLを使用して、CSR補助メモリの固定を実行するときに、コンテキスト切り替えのコンテキスト・シーケンス（例えば、図28および図29のうちの1つに従うシーケンス）が、固定HCALLおよび固定解除HCALLを含めて拡張される。本発明のこの態様でのコンテキスト切り替えの1つの実施形態に従って、以下のステップが実行される。

【0235】

（1）既知の手法に従って、汎用レジスタ、浮動小数点レジスタ、ベクトル・レジスタなどを含むが、これらに限定されない、以前のコンテキストの非CSRの状態を保存する。

【0236】

（2）（例えば、図28および図29のうちの1つの手法に基づいて）プロセッサ内構成状態レジスタを保存する。

【0237】

（3）受信CSR補助メモリ・ページ（すなわち、次のコンテキストに切り替える（有効化する）ことの一部として、構成状態レジスタのメモリ・ページとして有効化されているページ）を固定する：`HCALL(PIN, next_u->csr_page)`。`next_u`は、次のコンテキストとして切り替えられて（有効化されて）いるコンテキストのu領域を指し示すポインタである。

【0238】

（4）任意選択的に、少なくとも1つの実施形態では、送信構成状態レジスタを同期する：`sync_o_csr`。

【0239】

（5）有効化されているコンテキストに対応するCSR補助メモリのベース・アドレスを含むベース・レジスタを読み込む（1つの例では、このCSRはTCBRに対応する）：`mtspr TCBR, next_u->csr_page`。

【0240】

（6）任意選択的に、少なくとも1つの実施形態では、受信構成状態レジスタを同期する：`sync_i_csr`。

【0241】

（7）送信CSR補助メモリ・ページ（すなわち、以前のコンテキストを切り替える（無効化する）ことの一部として、CSR補助メモリ・ページとして無効化されているページ）を固定解除する：`HCALL(UNPIN, prev_u->csr_page)`。`prev_u`は、以前のコンテキストとして切り替えられて（無効化されて）いるコンテキストのu領域を指し示すポインタである。

【0242】

（8）既知の手法に従って、汎用レジスタ、浮動小数点レジスタ、ベクトル・レジスタなどを含むが、これらに限定されない、次のコンテキストの他の非CSRの状態を読み込む。

【0243】

（9）制御を新たに有効化されたコンテキストに移す。例えば、（Power ISAと連動する実装では）`rfid`命令を使用して、オペレーティング・システムからアプリケーションのスレッドまたはプロセスのコンテキストに移す。

【0244】

10

20

30

40

50

当業者は、本明細書における上記のステップの順序が変更されてよいということを、理解するであろう。例えば、少なくとも1つの実施形態では、固定動作の前に、固定解除動作が実行されてよい。その他の変形が可能である。

【0245】

別の実施形態では、TCBRなどのベース・レジスタを読み込むことに応答して、固定が実行される。通知イベント（例えば、割り込み）が、スーパーバイザに引き起こされてよい。複数のスーパーバイザ・レベルが存在する場合、1つの実施形態では、通知イベントが各スーパーバイザ・レベルに引き起こされる。通知イベントを受信することに応答して、固定情報が更新される。

【0246】

別の実施形態では、値をベース・レジスタに書き込むことによって、固定されたページを示すページ・テーブル・エントリ指示フラグが対応するPTE内で設定される。ページ・テーブル・エントリの複数のレベルが存在する場合、ページ・テーブル・エントリ指示フラグがレベルごとに設定されてよい。

【0247】

少なくとも別の実施形態では、固定プロセスまたは固定解除プロセスを開始する固定命令および固定解除命令のうちの少なくとも1つが提供される。

【0248】

さらに別の実施形態では、どのページが「固定されている」かを決定するために、システム内で有効なベース・レジスタを調べることによって、ホスト・ソフトウェアによって固定が決定されてよく、すなわち、ベース・レジスタ（例えば、TCBR）の複数の内容が、固定されたページの記録を表す。少なくとも1つの実施形態では、ページを移動またはページ・アウトする前に、スーパーバイザ・レベルのソフトウェアが、ページ・アドレスがベース・レジスタのうちの少なくとも1つに含まれるアドレスに対応するかどうかを判定することによって、ページが固定されたページに対応するかどうかを判定する。

【0249】

さらに別の実施形態では、ホストが、例えば1つの実施形態例では例外として、固定の通知イベントを受信してよい。通知を受信することに基づいて、ホスト・システムが、固定されるアドレスを受信し、将来メモリを管理するときに参照するために、そのアドレスを格納する。1つの実施形態では、通知は、固定解除されている以前のアドレスに関する情報（例えば、ベース構成状態レジスタに格納されている以前の値、または例えば構成状態レジスタを使用してその他の方法で提供される別の値）も含む。

【0250】

固定通知をホストに提供する1つの例に関する詳細が、図35を参照してさらに説明される。この論理は、プロセッサによって実行される。図35を参照すると、メモリ内構成状態レジスタを含んでいるメモリ・ページ（またはその他のメモリのユニット）に関して、新しい構成値（例えば、ゲスト・アドレス）が受信される（ステップ2400）。1つの例では、この新しい構成値がベース・レジスタ（例えば、TCBRなど）に格納される。しかし、他の実施形態では、この値が別の方法で提供されてよい。

【0251】

メモリ・ページのゲスト・アドレス（例えば、ゲスト仮想アドレスまたはゲスト実アドレス）が変換される（ステップ2402）。1つの例では、ゲスト・アドレスが物理的実アドレスに変換され、この変換が将来のアクセスのためにキャッシュされる。変数nが、ゲスト・レベルと同じに設定される（ステップ2404）。次にnが、選択された値（例えば、1）によってデクリメントされる（ステップ2406）。固定されているゲスト・アドレスに対応するホスト・レベルnの仮想アドレスを使用して、ホスト・レベルnに固定イベントが通知される（ステップ2408）。さらに、ホスト・レベルがさらに存在するか（例えば、nが0より大きい）かどうかに関する判定が行われる（照会2410）。ホスト・レベルがさらに存在する場合は、処理がステップ2406を続行する。ホスト・レベルがそれ以上存在しない場合は、処理が完了する。

【0252】

1つの実施形態では、アドレスに対応するページ・テーブル・エントリ内の、ビットなどのインジケータによって、固定が示される。このビットは、ページが固定されており、ゲストによって使用中であるということを示す。この固定インジケータの1つの例が、図33に示されている。

【0253】

固定に関連する詳細が、図36に示されている例示的な変換動作および固定動作を参照して、さらに説明される。1つの例では、プロセッサがこの処理を実行する。最初に、メモリ内構成状態レジスタを含んでいるメモリ・ページに関して、新しい構成値（例えば、ゲスト・アドレス）が受信される（ステップ2500）。ゲスト・アドレスが物理アドレスに変換され、この変換が将来のアクセスのためにキャッシュされる（ステップ2502）。変数nが、ゲスト・レベルと同じに設定され、ADDRESSがゲスト仮想アドレスと同じに設定される（ステップ2504）。その後、nが定義された値（1など）だけデクリメントされ、ADDRESSがtranslate__to__host（ADDRESS, n）と同じに設定される（ステップ2506）。すなわち、ADDRESSが、ホスト・レベル用の変換されたホスト・アドレスに設定される。アドレスのページ・テーブル・エントリ内で、固定インジケータ（例えば、ビット）が設定される（例えば、1に設定される）（ステップ2508）。さらに、ホスト・レベルがさらに存在するか（すなわち、nが0より大きい）かどうかに関する判定が行われる（照会2510）。ホスト・レベルがさらに存在する場合は、処理がステップ2506を続行する。ホスト・レベルがそれ以上存在しない場合は、処理が終了する。この時点で、ADDRESSは、固定されたページの物理アドレスに対応し、アドレス変換との相乗効果に使用され得る。

【0254】

変換およびキャッシングに基づいて、1つの例ではすべてのホスト・レベルで、インジケータ（例えば、ビット）が設定される。1つの実施形態では、ページ・テーブル・ウォーキングおよび固定指示が結合される。これによって、変換が、固定指示に使用されるのと同じページ・テーブル・エントリにアクセスするために、性能が向上する。

【0255】

1つの実施形態では、別の値（例えば、構成状態レジスタに格納されている以前の値（アドレス）、または例えば構成レジスタを使用してその他の方法で提供される別の値）に対して、固定解除が実行される。

【0256】

変換動作および固定解除動作に関連する処理の一例が、図37を参照して説明される。1つの例では、プロセッサがこの処理を実行する。最初に、アドレスの固定解除の要求が受信される（ステップ2600）。この要求は、固定解除されるゲスト仮想アドレスを含む。さらに、nが、ゲスト・レベルと同じに設定され、ADDRESSが、固定解除されるゲスト仮想アドレスと同じに設定される（ステップ2602）。次に、nが定義された値（1など）だけデクリメントされ、ADDRESSがtranslate__to__host（ADDRESS, n）と同じに設定される（ステップ2604）。すなわち、ADDRESSが、ホスト・レベル用の変換されたホスト・アドレスに設定される。アドレスのページ・テーブル・エントリ内の固定インジケータ（例えば、ビット）が、リセットされる（例えば、0に設定される）（ステップ2606）。その後、ホスト・レベルがさらに存在するか（例えば、nが0より大きい）かどうかに関する判定が行われる（ステップ2608）。ホスト・レベルがさらに存在する場合は、処理がステップ2604を続行する。ホスト・レベルがそれ以上存在しない場合は、処理が終了する。この時点で、ADDRESSは、固定解除されたページの物理アドレスに対応する。

【0257】

本明細書において説明されているように、メモリのユニットが固定されるということの決定に基づいて、通知が、例えば自動的に提供される。この通知は、インジケータを設定すること、割り込みを引き起こすこと、例外を提供することなどによるものであってよい

。多くの変形が可能である。

【0258】

さらに別の態様では、準仮想化された固定呼び出しを介して、効率的な固定管理が提供される。ページが導入されるたびに、それらのページの固定および固定解除を行う必要がないということが、望ましい。一方、ホストのページ・キャッシュを不必要に断片化しないように、固定されるページの数制限し、ページ割り当ての自由度を制限することも、望ましい。したがって、ホストによって固定解除されるページをゲストが指定する固定HCALL（ハイパーバイザ呼び出し）が、導入される。ハイパーバイザは、固定解除されるページが固定解除されたかどうかを示すことができ、それによって、ハイパーバイザが使用可能なリソースを有している場合に、すべてのページに対して固定要求を呼び出す必要がないように、ゲストに柔軟性を与える。

10

【0259】

この呼び出しは、1つの実施形態では、CSRメモリ補助ページへのベース・ポインタまたはベース・アドレス（ベース）を更新することを含む。さらに、1つの実施形態では、ゲストは、ページを固定された状態で維持するかどうかを指定する。

【0260】

別の実施形態では、ハイパーバイザは、ハイパーバイザがリソース不足の状況に陥ったときに、オペレーティング・システムへのコールバックによって、以前にオペレーティング・システムに残されている固定されたページを返すことを要求してよい。1つの実施形態では、オペレーティング・システムは、コールバックへの応答として、固定解除する1つまたは複数のページを指定する。

20

【0261】

1つまたは複数の態様に従って、図38を参照して説明されているように、1回のHCALLなどの単一の呼び出しが、例えばプロセッサ上で実行されるホストによって、固定動作および固定解除動作を実行するために使用される。示されているように、1つの例では、1回のハイパーバイザ呼び出し2704に回答して、固定動作2700および固定解除動作2702が実行される。1つの例では、古いページを固定解除するために、第1のアドレス（例えば、第1のベース・アドレス）に対して固定解除動作が実行され（ステップ2710）、新しいページを固定するために、第2のアドレス（例えば、第2のベース・アドレス）に対して固定動作が実行される（ステップ2720）。複数回の呼び出しの代わりに、1回の呼び出しが使用され、処理時間を節約する。固定解除呼び出しおよび固定呼び出しの回数は、結合された固定および固定解除呼び出しを含むこと、固定される新しいページ（例えば、CSR補助メモリ・ページ）を指定すること、および固定解除される以前のページを指定することによって、減らされる。

30

【0262】

1つの実施形態では、オペレーティング・システムは、固定解除されるように呼び出しにおいて指定されたアドレスが固定解除されないことを、ハイパーバイザ・コンソール管理制約（hypervisor console management constraints）が許可する場合に、それを要求することができる。アドレスが固定されるのか、または固定解除されるのかに関する応答が返される。後で、ハイパーバイザは、さらにコールバックを使用して、1つまたは複数の固定されたページを固定解除することを、オペレーティング・システムに要求することができる。

40

【0263】

1つの例では、オペレーティング・システムは、有効なメモリ内構成状態レジスタのページに必要な数より多い数のページを保持する。さらに別の例では、オペレーティング・システムは、有効かどうかに関わらず、メモリ内構成状態レジスタを保持しているすべてのページを固定する。これによって、その後、固定を実行する必要がなくなる。しかしこれは、過剰な数の固定されたページおよびシステムの非効率性につながる可能性がある。したがって、1つの実施形態では、オペレーティング・システムがコールバック関数を提供し、このコールバック関数で、システム内でメモリ内構成状態レジスタに使用するための

50

多すぎるページ（または選択された数より多い数のページ）が固定されている場合に、ハイパーバイザがオペレーティング・システムを呼び出し、固定されたページの割り当てを解除することができる。

【0264】

固定解除／固定がコンテキスト切り替えにおいて実行される下の例を参照して、固定解除／固定に関連する詳細がさらに説明される。具体的には、下の例は、ゲスト（OS）が、固定を切り替えること、およびさらに任意選択的に、固定を維持することをホスト（HV）に要求する、コンテキスト固定の一例を説明する。

【0265】

1つの実施形態では、コンテキスト（例えば、スレッドのコンテキスト、またはプロセスのコンテキスト、または論理パーティションのコンテキスト、または仮想マシンのコンテキスト、またはオペレーティング・システムのコンテキストなど）が、スーパーバイザ・ソフトウェアによって初期化されるときに、スーパーバイザが、初期化されているコンテキストに対応する1つまたは複数の構成状態レジスタの格納を提供するために、メモリを割り当てる。

【0266】

1つの例では、この割り当ては、適切なアライメントおよびサイズのメモリを提供する割り当てルーチンと呼ばれることによって実行されてよい。少なくとも1つの実施形態に従って、返されたアドレスが、コンテキストに対応する情報を格納するメモリ領域に格納される。1つの実施形態では、このメモリ領域は「u領域」と呼ばれ、変数「u」によって示される。少なくとも1つの実施形態では、変数uは、コンテキストに関して記録されるさまざまな属性に対応する複数のメンバーを含む、レコード、構造体、クラス、またはその他の複合データ・タイプである。少なくとも1つの実施形態では、この構造体は、少なくとも1つのCSR補助メモリ・ページのアドレスに対応するメンバー（フィールド）を含む。少なくとも1つの例では、このメンバーに、「csr_page」という名前が付けられる。

【0267】

`my_csr_page_pointer` = 補助ページの割り当て

【0268】

`u.csr_page = my_csr_page_pointer`

【0269】

準仮想化された環境内で、例えばハイパーバイザに対するHCALLを使用して、CSR補助メモリの固定を実行するとき、コンテキスト切り替えのコンテキスト・シーケンス（例えば、図28および図29のうちの1つに従うシーケンス）が、固定HCALLおよび固定解除HCALLを含めて拡張される。本発明のこの態様でのコンテキスト切り替えの1つの実施形態に従って、以下のステップが実行される。

【0270】

(1) 既知の手法に従って、汎用レジスタ、浮動小数点レジスタ、ベクトル・レジスタなどを含むが、これらに限定されない、以前のコンテキストの非CSRの状態を保存する。

【0271】

(2) (例えば、図28および図29のうちの1つの手法に基づいて) プロセッサ内構成状態レジスタを保存する。

【0272】

(3) 任意選択的に、少なくとも1つの実施形態では、送信構成状態レジスタを同期する：`sync_o_csr`。

【0273】

(4) (`next_u->csr_page_pinned`) が偽である場合

【0274】

固定解除するページを検出する `victim = select_TCBR_for_unpin()`

【0275】

維持するのが望ましいか？ `retain=retain_desirable_p(victim)`

【0276】

ページの固定を得るために与える `lost_victim=HCALL(PIN_give_to_get, next_u->csr_page, victim, retain)`

【0277】

`victim`が失われたか？ `(lost_victim) mark_unpinned(victim)` が真である場合、

【0278】

(5) 任意選択的に、少なくとも1つの実施形態では、受信構成状態レジスタを同期する：`sync_i_csr`。

【0279】

(6) 既知の手法に従って、汎用レジスタ、浮動小数点レジスタ、ベクトル・レジスタなどを含むが、これらに限定されない、次のコンテキストの他の非CSRの状態を読み込む。

【0280】

(7) 制御を新たに有効化されたコンテキストに移す。例えば、(Power ISAと連動する実装では) `rfid` 命令を使用して、オペレーティング・システムからアプリケーションのスレッドまたはプロセスのコンテキストに移す。

【0281】

固定／固定解除動作を実行する一例の詳細が、図39を参照してさらに説明される。新しいアドレス (NA: new address) の固定および古いアドレス (OA: old address) の固定解除の要求が、例えば、1回のハイパーバイザ呼び出し、およびOAを固定されたメモリとして維持する要求が行われているかどうかを指定する維持インジケータ (またはそのような指示) を介して、受信される (ステップ2800)。1つの例として、新しいアドレスが、CSRメモリ補助ページを示す。アドレスNAが固定され (ステップ2810)、維持インジケータがアドレスOAをメモリ内で固定された状態で維持する要求を指定しているかどうかに関する判定が行われる (照会2820)。OAの固定を維持する要求が存在する場合、示されたポリシー (例えば、複数の仮想マシンにわたるリソース割り当て) および固定に使用できるメモリに基づいて、固定されたページの維持を許可するかどうかに関する判定が行われる (照会2822)。この要求が許可された場合 (照会2824)、固定解除動作が実行されないという指示が行われる (ステップ2826)。この要求が許可されない場合、アドレスOAが固定解除され (ステップ2830)、固定解除が実行されたことの指示が提供される (ステップ2832)。

【0282】

照会2820に戻り、維持インジケータが、固定が維持されないということを指定する場合、処理がステップ2830を続行する。

【0283】

前述したように、1つのアドレスを固定解除すること、別のアドレスを固定すること、または固定解除の対象のアドレスが実際には固定解除されないことを要求すること、あるいはその組合せを実行するために、1回の呼び出しが使用されてよい。1回の呼び出しを使用することによって、処理が容易になり、性能が改善される。

【0284】

本明細書において説明されているように、選択された構成状態レジスタが、プロセッサ内の代わりに、メモリに格納される。レジスタをメモリに格納することによって、データの破損の検出および修正に関連することを含めて、特定の恩恵および最適化を実現することができる。

【0285】

10

20

30

40

50

1つの例では、回復力を強化するために、およびシングル・イベント・アップセット（SEU：single event upsets）またはソフト・エラーに対処するために、メモリによって補助された状態が使用される。シングル・イベント・アップセットは、電離放射線の影響によって導入される状態の変化である。CMOS（complementary metal-oxide-semiconductor：相補型金属酸化膜半導体）のサイズが縮小するにつれて、ビットごとに蓄積される電荷が少なくなるため、ビットを変更するために使用される電荷量 Q_{CRIT} が縮小する。シングル・イベント・アップセットの影響を修正するために、データ保護が適用される。このデータ保護は、例えば、レジスタのパリティまたはエラー訂正符号（ECC：error correction code）保護を使用して、損傷した状態のレジスタ値をそれぞれ検出および修復することを含む。多くのレジスタにわたる領域全体の設計が償却可能である場合、レジスタ・ファイルの修復が妥当な価格であるため、エラー訂正符号またはパリティ保護が使用される。プロセッサ内構成レジスタの場合、レジスタごとに別々の保護および回復を設計する必要があるため、この費用が負担できないほど高額になることが多い。

10

【0286】

しかし、本発明の態様に従って、構成状態レジスタは、それらが、パリティ・ビットおよびエラー訂正符号（ECC）を含むが、これに限定されない、1つまたは複数の高度な保護メカニズムを使用して保護されている、メモリに格納される。

【0287】

1つの態様では、プロセッサ内構成状態レジスタも、SEUに対する回復力のあるシステム・メモリ階層を使用することによって保護される。1つの実施形態では、プロセッサ内構成状態レジスタは、SEUによって引き起こされる破損を検出するための手法を使用して保護される。本発明の態様と共に、さまざまな検出手法が使用されてよい。1つの例では、破損検出メカニズムは、プロセッサ内構成状態レジスタに対するデータ・パリティ保護の使用に対応する。別の実施形態では、構成レジスタへの書き込みがない場合に、レジスタ値の変化の特徴をテストすることによって、SEUによって引き起こされる破損が検出されてよい。加えて、そのようなプロセッサ内レジスタは、プロセッサ内構成レジスタのECCによって保護されたコピーが使用可能であることを保証するために、メモリにも格納される。1つの実施形態では、そのように保護されたプロセッサ内構成レジスタの更新にตอบสนองして、更新のコピーもメモリ内のコピーに格納される。プロセッサがパリティ・エラーを認識した場合、ECCによって保護されたメモリ内のコピーから値が取り出され、プロセッサ内レジスタが更新される。

20

30

【0288】

さらに、1つの実施形態では、命令アドレス、データ・アドレス、およびコンテンツ・ブレイク・ポイント・レジスタ（content break point registers）などの使用率の高い値は、補助メモリに格納され、シングル・イベント・アップセットが検出されたときに回復され得る。回復は、ハードウェアの再読み込み経路（reload path）を介するか、またはマシン・チェックを実行し、マシン・チェック・ハンドラにそれらのレジスタを再読み込みさせることによる、パリティ保護を含む。したがって、本発明の態様に従って、構成状態レジスタは、構成状態レジスタをシステム・メモリに格納し、例えばECCを使用してシングル・イベント・アップセットから保護することによって、保護される。

40

【0289】

以下では、エラー訂正符号を構成状態レジスタに使用することに関連する詳細がさらに説明される。具体的には、エラー訂正符号をデータ書き込みに使用する例が、図40～図42を参照して説明され、エラー訂正符号をデータ読み取りに使用する例が、図43～図45を参照して説明される。

【0290】

最初に図40を参照すると、この例では、データ書き込みが保護されていない。この例では、プロセッサ内構成状態レジスタの値が受信され（ステップ2900）、その値が、構成状態レジスタを実装するラッチに書き込まれる（ステップ2902）。このデータ書き込みは、シングル・イベント・アップセットおよびその他の種類のエラーに対して保護

50

されていない。

【0291】

これに対して、図41を参照すると、プロセッサ内構成状態レジスタの値が受信され（ステップ2920）、エラー保護またはエラー訂正符号が計算される（ステップ2922）。受信された値が、保護またはエラー訂正符号と共に、構成状態レジスタ（例えば、構成状態レジスタを実装するラッチ）に書き込まれる（ステップ2924）。

【0292】

さらに、図42を参照して、メモリ内構成状態レジスタに対するデータ書き込みの一実施形態が説明される。この例では、メモリ内構成状態レジスタの値が受信され（ステップ2952）、構成状態レジスタが格納されるシステム・メモリ・アドレスに関する決定が行われる（ステップ2954）。次に、この値が、エラー保護から恩恵を受けるメモリの一部であるため保護されている、そのシステム・メモリ・アドレスに格納される（ステップ2956）。

10

【0293】

1つの態様では、この格納することは、例えば、受信されたメモリ内構成状態レジスタ値に対してエラー訂正符号を計算することと、計算されたエラー訂正符号を受信された値と共に格納することを含む。知られているように、エラー訂正符号は、1つまたは複数のパリティ・ビット（例えば、データ・ビットの1つまたは複数のサブセットごとに1つまたは複数のパリティ・ビット）を、値を表すデータ・ビットに追加し、それらのパリティ・ビットを使用してエラーを決定する。データ内にエラーが存在する場合、パリティ・ビットは、エラーが存在するデータ・ビット内の位置を示し、修正を行うことができるようにする（例えば、2進数で表された1つまたは複数のデータ・ビットを別の値に変更する）。

20

【0294】

上記に加えて、プロセッサ内構成状態レジスタおよびメモリ内構成状態レジスタに対してデータ読み取りを実行する一例が、図43～図45を参照して説明される。

【0295】

図43を参照して、保護が提供されていないプロセッサ内構成状態レジスタに対するデータ読み取りの一例が、説明される。この例では、値がプロセッサの論理に提供される（ステップ3000）。

30

【0296】

これに対して、保護が提供されているプロセッサ内構成状態レジスタから値を読み取ることに関連する処理の一例が、図44を参照して説明される。この例では、ラッチから値が受信され（ステップ3020）、訂正符号がチェックされる（ステップ3022）。破損が検出されない場合（照会3024）、値がプロセッサの論理に提供される（ステップ3030）。しかし、照会3024に戻り、破損が検出された場合、修正された値が計算され（ステップ3026）、修正された値が再びラッチに書き込まれる（ステップ3028）。さらに、その値がプロセッサの論理に提供される（ステップ3030）。

【0297】

さらに、メモリ内構成状態レジスタを使用するデータ読み取りが、図45を参照して説明される。この例では、構成状態レジスタ番号がプロセッサによって受信される（ステップ3050）。構成状態レジスタのシステム・メモリ・アドレスに関する決定が行われる（ステップ3052）。保護されたシステム・メモリから値が読み取られ（ステップ3054）、その値がプロセッサの論理に提供される（ステップ3056）。

40

【0298】

1つの態様では、この読み取ることは、エラー訂正符号を使用して、データ（例えば、値）の破損が発生したかどうかを判定することを含む。破損が検出された場合、回復を実行することを含むが、これに限定されない、1つまたは複数のアクションが実行されてよい。回復は、エラー訂正符号を使用して、破損した値に対して、修正された値を計算することを含んでよい。その他の例も存在する。

50

【0299】

メモリ内構成状態レジスタを使用することによって、エラー保護の恩恵が得られ、費用のかかる追加ステップを回避し、およびそのような保護を、プロセッサ内構成状態レジスタを実装するラッチに追加するための待ち時間を防ぐ。

【0300】

例えば、エラーの修正は、メモリ内構成状態レジスタに書き込むことに基づいて提供される。エラー検出および訂正符号が、メモリに格納される構成状態レジスタの値を受信することに基づいて生成されてよい。エラー検出および訂正符号は、構成状態レジスタを格納しているメモリに対する読み取り要求を受信することに基づいて、破損が発生したかどうかを検出するために使用されてよい。メモリ内構成状態レジスタを読み取り、破損が発生したことを検出することに基づいて、破損が修正される。

10

【0301】

本明細書では、メモリ内構成状態レジスタを提供することに関連する態様が、詳細に説明された。構成状態レジスタをメモリ内で提供することによって、処理が容易になり、性能を向上させることができる。特定の改良および最適化が実現されてよい。

【0302】

メモリ内構成状態レジスタは、命令の処理において、およびその他の動作シーケンスにおいて使用されてよい。例えば、外部デバイスから割り込み信号を受信することに基づいて、例外が受信されてよい。例外の処理において、1つまたは複数の構成状態レジスタ（SRROおよびSRR1など）がアクセスされてよい。それらのレジスタがメモリ内構成状態レジスタである場合、割り込み処理シーケンスが、読み込み動作または格納動作あるいはその両方を含むように拡張される。その他の例または変形あるいはその両方が可能である。

20

【0303】

本発明の1つまたは複数の態様は、コンピュータ技術に密接に関係しており、コンピュータ内の処理を容易にし、その性能を改善する。コンピューティング環境内の処理が、本発明の1つまたは複数の態様に関連しているとき、それらを容易にする一実施形態の詳細が、図46～図47を参照してさらに説明される。

【0304】

図46を参照すると、第1のコンテキストから第2のコンテキストへのコンテキスト切り替えが実行されるということの決定が行われる（3100）。第1のコンテキストのデータは、第1のメモリ・ユニットに少なくとも部分的に格納された1つまたは複数の構成状態レジスタに格納され、第2のコンテキストのデータは、第2のメモリ・ユニットに少なくとも部分的に格納された1つまたは複数の構成状態レジスタに格納される（3102）。第1のメモリ・ユニットから第2のメモリ・ユニットにポインタを変更することによって、コンテキスト切り替えが実行される（3104）。1つの例として、第1のメモリ・ユニットはメモリの1つのページであり、第2のメモリ・ユニットはメモリの別のページである（3106）。

30

【0305】

1つの実施形態では、コンテキスト切り替えを実行することは、第1のコンテキストに関して1つまたは複数のプロセッサ内構成状態レジスタを処理することを含む（3108）。1つまたは複数のプロセッサ内構成状態レジスタのデータが、1つまたは複数のプロセッサ内構成状態レジスタからコピーされてメモリに格納され、第2のコンテキストに関するデータが、1つまたは複数のプロセッサ内構成状態レジスタにコピーされる（3110）。

40

【0306】

さらに、1つの例では、コンテキスト切り替えを実行することは、ポインタを変更する前にポインタを格納することを含む（3112）。

【0307】

1つの例として、図47を参照すると、第1のコンテキストが、スーパーバイザまたは

50

アプリケーションを含み、第2のコンテキストが、別のスーパーバイザまたはアプリケーションを含む(3114)。別の例として、第1のコンテキストが1つのスレッドを含み、第2のコンテキストが別のスレッドを含む(3116)。その他の例も可能である。

【0308】

1つの態様では、複数のスレッドから他のスレッドが選択可能である(3118)。複数のスレッドは、コンテキスト切り替えを実行するプロセッサに読み込まれない1つまたは複数のスレッドを含む(3120)。

【0309】

別の例では、コンテキスト切り替えは、仮想マシンを1つのプロセッサから別のプロセッサに移動することに基づいて実行される(3122)。さらに、1つの実施形態では、第2のメモリ・ユニット内の構成状態レジスタのレイアウトが、第1のメモリ・ユニット内の構成状態レジスタのレイアウトと異なる(3124)。1つの例として、この違いの指示がバージョン番号によって提供される(3126)。

【0310】

1つの実施形態では、コンテキスト切り替えを実行することが、第1のメモリ・ユニットに格納された1つまたは複数の構成状態レジスタに対応するキャッシュされたデータを無効化することを含む(3128)。

【0311】

その他の変形および実施形態が可能である。

【0312】

エミュレーション環境を含むが、これに限定されない、その他の種類のコンピューティング環境が、本発明の1つまたは複数の態様を組み込んで使用してもよく、その例が、図48を参照して説明される。この例では、コンピューティング環境20は、例えば、1つまたは複数のバス28またはその他の接続あるいはその両方を介して互いに結合された、ネイティブ中央処理装置(CPU)22、メモリ24、および1つまたは複数の入出力デバイスまたはインターフェイスあるいはその両方26を含む。例えば、コンピューティング環境20は、International Business Machines Corporation(ニューヨーク州アーモンク市)によって提供されるPowerPCプロセッサまたはpSeriesサーバ、あるいはInternational Business Machines Corporation、Intel、またはその他の企業によって提供されるアーキテクチャに基づくその他のマシン、あるいはその組合せを含んでよい。

【0313】

ネイティブ中央処理装置22は、環境内で処理中に使用される1つまたは複数の汎用レジスタまたは1つまたは複数の専用レジスタあるいはその両方などの、1つまたは複数のネイティブ・レジスタ30を含む。これらのレジスタは、いずれかの特定の時点での環境の状態を表す情報を含む。

【0314】

さらに、ネイティブ中央処理装置22は、メモリ24に格納された命令およびコードを実行する。1つの特定の例では、中央処理装置は、メモリ24に格納されたエミュレータ・コード32を実行する。このコードは、あるアーキテクチャで構成されたコンピューティング環境が、別のアーキテクチャをエミュレートできるようにする。例えば、エミュレータ・コード32は、z/Architecture以外のアーキテクチャに基づくマシン(PowerPCプロセッサ、pSeriesサーバ、あるいはその他のサーバまたはプロセッサなど)が、z/Architectureをエミュレートし、z/Architectureに基づいて開発されたソフトウェアおよび命令を実行できるようにする。

【0315】

エミュレータ・コード32に関連する詳細が、図49を参照してさらに説明される。メモリ24に格納されたゲスト命令40は、ネイティブCPU22のアーキテクチャ以外のアーキテクチャにおいて実行されるように開発された(例えば、マシン命令と相互関係が

10

20

30

40

50

ある) ソフトウェア命令を含む。例えば、ゲスト命令 40 は、*z / Architecture* プロセッサ上で実行するように設計されてよいが、代わりに、例えばインテル・プロセッサであってよい、ネイティブ CPU 22 上でエミュレートされる。1 つの例では、エミュレータ・コード 32 は、メモリ 24 から 1 つまたは複数のゲスト命令 40 を取得するため、および取得された命令のローカル・バッファリングを任意選択的に提供するための命令フェッチ・ルーチン 42 を含む。エミュレータ・コード 32 は、取得されたゲスト命令の種類を決定するため、およびゲスト命令を、1 つまたは複数の対応するネイティブ命令 46 に変換するための命令変換ルーチン 44 も含む。この変換は、例えば、ゲスト命令によって実行される機能を識別すること、およびこの機能を実行するためのネイティブ命令を選択することを含む。

10

【0316】

さらに、エミュレータ・コード 32 は、ネイティブ命令の実行を引き起こすためのエミュレーション制御ルーチン 48 を含む。エミュレーション制御ルーチン 48 は、ネイティブ CPU 22 に、1 つまたは複数のすでに取得されたゲスト命令をエミュレートするネイティブ命令のルーチンを実行させ、そのような実行の終了時に、次のゲスト命令またはゲスト命令のグループの取得をエミュレートするために、制御を命令フェッチ・ルーチンに返してよい。ネイティブ命令 46 の実行は、データをメモリ 24 からレジスタに読み込むこと、データをレジスタからメモリに再び格納すること、あるいは変換ルーチンによって決定された何らかの種類の算術演算または論理演算を実行することを含んでよい。

【0317】

20

例えば、各ルーチンは、メモリに格納されてネイティブ中央処理装置 22 によって実行されるソフトウェアにおいて実装される。他の例では、ルーチンまたは動作のうちの 1 つまたは複数のは、ファームウェア、ハードウェア、ソフトウェア、またはこれらの何らかの組合せにおいて実装される。エミュレートされたプロセッサのレジスタは、ネイティブ CPU のレジスタ 30 を使用して、またはメモリ 24 内の位置を使用することによって、エミュレートされてよい。実施形態では、ゲスト命令 40、ネイティブ命令 46、およびエミュレータ・コード 32 は、同じメモリ内に存在してよく、または異なるメモリ・デバイス間で分配されてよい。

【0318】

本明細書において使用されるとき、ファームウェアは、例えば、プロセッサのマイクロコードまたはミリコードを含む。ファームウェアは、例えば、上位レベルの機械コードの実装において使用される、ハードウェア・レベルの命令またはデータ構造あるいはその両方を含む。一実施形態では、ファームウェアは、例えば、信頼できるソフトウェアを含んでいるマイクロコード、または基盤になるハードウェアに固有のマイクロコードとして通常は提供される、システムのハードウェアへのオペレーティング・システムのアクセスを制御する独自のコードを含む。

30

【0319】

取得され、変換されて実行されるゲスト命令 40 は、例えば、本明細書に記載された命令のうちの 1 つであってよい。あるアーキテクチャ（例えば、*z / Architecture*）の命令が、メモリからフェッチされ、変換され、別のアーキテクチャ（例えば、*PowerPC*、*pSeries*、*Intel* など）のネイティブ命令 46 のシーケンスとして表される。その後、これらのネイティブ命令が実行される。

40

【0320】

1 つまたは複数の態様がクラウド・コンピューティングに関連してよい。

【0321】

本開示にはクラウド・コンピューティングの詳細な説明が含まれているが、本明細書において示された内容の実装は、クラウド・コンピューティング環境に限定されないということが、あらかじめ理解される。本発明の実施形態は、現在既知であるか、または今後開発される任意のその他の種類のコンピューティング環境と組み合わせて実装できる。

【0322】

50

クラウド・コンピューティングは、構成可能な計算リソース（例えば、ネットワーク、ネットワーク帯域幅、サーバ、処理、メモリ、ストレージ、アプリケーション、仮想マシン、およびサービス）の共有プールへの便利なオンデマンドのネットワーク・アクセスを可能にするためのサービス提供モデルであり、管理上の手間とサービス・プロバイダとのやりとりを最小限に抑えて、それらのリソースを迅速にプロビジョニングおよび解放することができる。このクラウド・モデルは、少なくとも5つの特徴、少なくとも3つのサービス・モデル、および少なくとも4つのデプロイメント・モデルを含むことができる。

【0323】

特徴は、次のとおりである。

【0324】

オンデマンドのセルフ・サービス：クラウドの利用者は、サーバの時間、ネットワーク・ストレージなどの計算能力を一方的に、サービス・プロバイダとの人間的なやりとりを必要とせず、必要に応じて自動的にプロビジョニングすることができる。

【0325】

幅広いネットワーク・アクセス：クラウドの能力は、ネットワークを経由して利用可能であり、標準的なメカニズムを使用してアクセスできるため、異種のシン・クライアントまたはシック・クライアント・プラットフォーム（例えば、携帯電話、ラップトップ、およびPDA）による利用を促進する。

【0326】

リソース・プール：プロバイダの計算リソースは、プールされ、マルチテナント・モデルを使用して複数の利用者に提供される。さまざまな物理的および仮想的リソースが、要求に従って動的に割り当ておよび再割り当てされる。場所に依存しないという感覚があり、利用者は通常、提供されるリソースの正確な場所に関して管理することも知らないが、さらに高い抽象レベルでは、場所（例えば、国、州、またはデータセンター）を指定できる場合がある。

【0327】

迅速な順応性：クラウドの能力は、迅速かつ柔軟に、場合によっては自動的にプロビジョニングされ、素早くスケールアウトし、迅速に解放されて素早くスケールインすることができる。プロビジョニングに使用できる能力は、利用者には、多くの場合、任意の量をいつでも無制限に購入できるように見える。

【0328】

測定されるサービス：クラウド・システムは、計測機能を活用することによって、サービスの種類（例えば、ストレージ、処理、帯域幅、およびアクティブなユーザのアカウント）に適した抽象レベルで、リソースの使用を自動的に制御および最適化する。リソースの使用量は監視、制御、および報告することができ、利用されるサービスのプロバイダと利用者の両方に透明性が提供される。

【0329】

サービス・モデルは、次のとおりである。

【0330】

SaaS（Software as a Service）：利用者に提供される能力は、クラウド・インフラストラクチャ上で稼働しているプロバイダのアプリケーションの利用である。それらのアプリケーションは、Webブラウザ（例えば、Webベースの電子メール）などのシン・クライアント・インターフェイスを介して、さまざまなクライアント・デバイスからアクセスできる。利用者は、ネットワーク、サーバ、オペレーティング・システム、ストレージ、または個々のアプリケーション機能を含む基盤になるクラウド・インフラストラクチャを、限定的なユーザ固有のアプリケーション構成設定を行う可能性を除き、管理することも制御することもない。

【0331】

PaaS（Platform as a Service）：利用者に提供される能力は、プロバイダによってサポートされるプログラミング言語およびツールを使用して作成された、利用者が作成

10

20

30

40

50

または取得したアプリケーションをクラウド・インフラストラクチャにデプロイすることである。利用者は、ネットワーク、サーバ、オペレーティング・システム、またはストレージを含む基盤になるクラウド・インフラストラクチャを管理することも制御することもないが、デプロイされたアプリケーション、および場合によってはアプリケーション・ホスティング環境の構成を制御することができる。

【0332】

IaaS (Infrastructure as a Service) : 利用者に提供される能力は、処理、ストレージ、ネットワーク、およびその他の基本的な計算リソースのプロビジョニングであり、利用者は、オペレーティング・システムおよびアプリケーションを含むことができる任意のソフトウェアをデプロイして実行できる。利用者は、基盤になるクラウド・インフラストラクチャを管理することも制御することもないが、オペレーティング・システム、ストレージ、およびデプロイされたアプリケーションを制御することができ、場合によっては、選択されたネットワーク・コンポーネント（例えば、ホスト・ファイアウォール）を限定的に制御できる。

【0333】

デプロイメント・モデルは、次のとおりである。

【0334】

プライベート・クラウド：このクラウド・インフラストラクチャは、ある組織のためにのみ運用される。この組織またはサード・パーティによって管理することができ、オンプレミスまたはオフプレミスに存在することができる。

【0335】

コミュニティ・クラウド：このクラウド・インフラストラクチャは、複数の組織によって共有され、関心事（例えば、任務、セキュリティ要件、ポリシー、およびコンプライアンスに関する考慮事項）を共有している特定のコミュニティをサポートする。これらの組織またはサード・パーティによって管理することができ、オンプレミスまたはオフプレミスに存在することができる。

【0336】

パブリック・クラウド：このクラウド・インフラストラクチャは、一般ユーザまたは大規模な業界団体が使用できるようになっており、クラウド・サービスを販売する組織によって所有される。

【0337】

ハイブリッド・クラウド：このクラウド・インフラストラクチャは、データとアプリケーションの移植を可能にする標準化された技術または独自の技術（例えば、クラウド間の負荷バランスを調整するためのクラウド・バースト）によって固有の実体を残したまま互いに結合された2つ以上のクラウド（プライベート、コミュニティ、またはパブリック）の複合である。

【0338】

クラウド・コンピューティング環境は、ステートレス、疎結合、モジュール性、および意味的相互運用性に重点を置いたサービス指向の環境である。クラウド・コンピューティングの中心になるのは、相互接続されたノードのネットワークを備えるインフラストラクチャである。

【0339】

ここで図50を参照すると、例示的なクラウド・コンピューティング環境50が示されている。図示されているように、クラウド・コンピューティング環境50は、クラウドの利用者によって使用されるローカル・コンピューティング・デバイス（例えば、PDA（Personal Digital Assistant）または携帯電話54A、デスクトップ・コンピュータ54B、ラップトップ・コンピュータ54C、または自動車コンピュータ・システム54N、あるいはその組合せなど）が通信できる1つまたは複数のクラウド・コンピューティング・ノード10を備える。ノード10は、互いに通信することができる。ノード10は、1つまたは複数のネットワーク内で、本明細書において前述されたプライベート・クラウド

10

20

30

40

50

、コミュニティ・クラウド、パブリック・クラウド、またはハイブリッド・クラウド、あるいはこれらの組合せなどに、物理的または仮想的にグループ化されてよい（図示されていない）。これによって、クラウド・コンピューティング環境 50 は、クラウドの利用者がローカル・コンピューティング・デバイス上でリソースを維持する必要のないインフラストラクチャ、プラットフォーム、または SaaS、あるいはその組合せを提供できる。図 50 に示されたコンピューティング・デバイス 54A～N の種類は、例示のみが意図されており、コンピューティング・ノード 10 およびクラウド・コンピューティング環境 50 は、任意の種類のネットワークまたはネットワーク・アドレス可能な接続（例えば、Web ブラウザを使用した接続）あるいはその両方を経由して任意の種類のコンピュータ制御デバイスと通信することができると理解される。

10

【0340】

ここで図 51 を参照すると、クラウド・コンピューティング環境 50（図 50）によって提供される機能的抽象レイヤのセットが示されている。図 51 に示されたコンポーネント、レイヤ、および機能は、例示のみが意図されており、本発明の実施形態がこれらに限定されないということが、あらかじめ理解されるべきである。図示されているように、次のレイヤおよび対応する機能が提供される。

【0341】

ハードウェアおよびソフトウェア・レイヤ 60 は、ハードウェア・コンポーネントおよびソフトウェア・コンポーネントを含む。ハードウェア・コンポーネントの例としては、メインフレーム 61、RISC（Reduced Instruction Set Computer）アーキテクチャベースのサーバ 62、サーバ 63、ブレード・サーバ 64、ストレージ・デバイス 65、ならびにネットワークおよびネットワーク・コンポーネント 66 が挙げられる。一部の実施形態では、ソフトウェア・コンポーネントは、ネットワーク・アプリケーション・サーバ・ソフトウェア 67 およびデータベース・ソフトウェア 68 を含む。

20

【0342】

仮想化レイヤ 70 は、仮想サーバ 71、仮想ストレージ 72、仮想プライベート・ネットワークを含む仮想ネットワーク 73、仮想アプリケーションおよびオペレーティング・システム 74、ならびに仮想クライアント 75 などの仮想的実体を提供できる抽象レイヤを備える。

【0343】

一例を挙げると、管理レイヤ 80 は、以下で説明される機能を提供できる。リソース・プロビジョニング 81 は、クラウド・コンピューティング環境内でタスクを実行するために利用される計算リソースおよびその他のリソースの動的調達を行う。計測および価格設定 82 は、クラウド・コンピューティング環境内でリソースが利用される際のコスト追跡、およびそれらのリソースの利用に対する請求書の作成と送付を行う。一例を挙げると、それらのリソースは、アプリケーション・ソフトウェア・ライセンスを含むことができる。セキュリティは、クラウドの利用者およびタスクの ID 検証を行うと共に、データおよびその他のリソースの保護を行う。ユーザ・ポータル 83 は、クラウド・コンピューティング環境へのアクセスを利用者およびシステム管理者に提供する。サービス・レベル管理 84 は、必要なサービス・レベルを満たすように、クラウドの計算リソースの割り当てと管理を行う。サービス水準合意（SLA：Service Level Agreement）計画および実行 85 は、今後の要求が予想されるクラウドの計算リソースの事前準備および調達を、SLA に従って行う。

30

【0344】

ワークロード・レイヤ 90 は、クラウド・コンピューティング環境で利用できる機能の例を示している。このレイヤから提供されてよいワークロードおよび機能の例としては、マッピングおよびナビゲーション 91、ソフトウェア開発およびライフサイクル管理 92、仮想クラスルーム教育の配信 93、データ解析処理 94、トランザクション処理 95、およびメモリベースの構成状態レジスタ処理 96 が挙げられる。

40

【0345】

50

本発明は、任意の可能な統合の技術的詳細レベルで、システム、方法、またはコンピュータ・プログラム製品、あるいはその組合せであってよい。コンピュータ・プログラム製品は、プロセッサに本発明の態様を実行させるためのコンピュータ可読プログラム命令を含んでいるコンピュータ可読記憶媒体を含んでよい。

【0346】

コンピュータ可読記憶媒体は、命令実行デバイスによって使用するための命令を保持および格納できる有形のデバイスであることができる。コンピュータ可読記憶媒体は、例えば、電子ストレージ・デバイス、磁気ストレージ・デバイス、光ストレージ・デバイス、電磁ストレージ・デバイス、半導体ストレージ・デバイス、またはこれらの任意の適切な組合せであってよいが、これらに限定されない。コンピュータ可読記憶媒体のさらに具体的な例の非網羅的リストは、ポータブル・フロッピー・ディスク、ハード・ディスク、ランダム・アクセス・メモリ（RAM：random access memory）、読み取り専用メモリ（ROM：read-only memory）、消去可能プログラマブル読み取り専用メモリ（EPROM：erasable programmable read-only memoryまたはフラッシュ・メモリ）、スタティック・ランダム・アクセス・メモリ（SRAM：static random access memory）、ポータブル・コンパクト・ディスク読み取り専用メモリ（CD-ROM：compact disc read-only memory）、デジタル多用途ディスク（DVD：digital versatile disk）、メモリ・スティック、フロッピー・ディスク、パンチカードまたは命令が記録されている溝の中の隆起構造などの機械的にエンコードされるデバイス、およびこれらの任意の適切な組合せを含む。本明細書において使用されるとき、コンピュータ可読記憶媒体は、それ自体が、電波またはその他の自由に伝搬する電磁波、導波管またはその他の送信媒体を伝搬する電磁波（例えば、光ファイバ・ケーブルを通過する光パルス）、あるいはワイヤを介して送信される電気信号などの一過性の信号であると解釈されるべきではない。

【0347】

本明細書に記載されたコンピュータ可読プログラム命令は、コンピュータ可読記憶媒体から各コンピューティング・デバイス／処理デバイスへ、またはネットワーク（例えば、インターネット、ローカル・エリア・ネットワーク、広域ネットワーク、または無線ネットワーク、あるいはその組合せ）を介して外部コンピュータまたは外部ストレージ・デバイスへダウンロードされ得る。このネットワークは、銅伝送ケーブル、光伝送ファイバ、無線送信、ルータ、ファイアウォール、スイッチ、ゲートウェイ・コンピュータ、またはエッジ・サーバ、あるいはその組合せを備えてよい。各コンピューティング・デバイス／処理デバイス内のネットワーク・アダプタ・カードまたはネットワーク・インターフェイスは、コンピュータ可読プログラム命令をネットワークから受信し、それらのコンピュータ可読プログラム命令を各コンピューティング・デバイス／処理デバイス内のコンピュータ可読記憶媒体に格納するために転送する。

【0348】

本発明の動作を実行するためのコンピュータ可読プログラム命令は、アセンブラ命令、命令セット・アーキテクチャ（ISA：instruction-set-architecture）命令、マシン命令、マシン依存命令、マイクロコード、ファームウェア命令、状態設定データ、集積回路のための構成データ、あるいは、Smalltalk（R）、C++などのオブジェクト指向プログラミング言語、および「C」プログラミング言語または同様のプログラミング言語などの手続き型プログラミング言語を含む1つまたは複数のプログラミング言語の任意の組合せで記述されたソース・コードまたはオブジェクト・コードであってよい。コンピュータ可読プログラム命令は、ユーザのコンピュータ上で全体的に実行すること、ユーザのコンピュータ上でスタンドアロン・ソフトウェア・パッケージとして部分的に実行すること、ユーザのコンピュータ上およびリモート・コンピュータ上でそれぞれ部分的に実行すること、あるいはリモート・コンピュータ上またはサーバ上で全体的に実行することができる。後者のシナリオでは、リモート・コンピュータは、ローカル・エリア・ネットワーク（LAN：local area network）または広域ネットワーク（WAN：wide area network）を含む任意の種類のネットワークを介してユーザのコンピュータに接続されて

よく、または接続は、（例えば、インターネット・サービス・プロバイダを使用してインターネットを介して）外部コンピュータに対して行われてよい。一部の実施形態では、本発明の態様を実行するために、例えばプログラマブル論理回路、フィールドプログラマブル・ゲート・アレイ（FPGA：field-programmable gate arrays）、またはプログラマブル・ロジック・アレイ（PLA：programmable logic arrays）を含む電子回路は、コンピュータ可読プログラム命令の状態情報を利用することによって、電子回路をカスタマイズするためのコンピュータ可読プログラム命令を実行してよい。

【0349】

本発明の態様は、本明細書において、本発明の実施形態に従って、方法、装置（システム）、およびコンピュータ・プログラム製品のフローチャート図またはブロック図あるいはその両方を参照して説明される。フローチャート図またはブロック図あるいはその両方の各ブロック、ならびにフローチャート図またはブロック図あるいはその両方に含まれるブロックの組合せが、コンピュータ可読プログラム命令によって実装され得るということが理解されるであろう。

【0350】

これらのコンピュータ可読プログラム命令は、コンピュータまたはその他のプログラム可能なデータ処理装置のプロセッサを介して実行される命令が、フローチャートまたはブロック図あるいはその両方のブロックに指定される機能／動作を実施する手段を作り出すべく、汎用コンピュータ、専用コンピュータ、または他のプログラム可能なデータ処理装置のプロセッサに提供されてマシンを作り出すものであってよい。これらのコンピュータ可読プログラム命令は、命令が格納されたコンピュータ可読記憶媒体がフローチャートまたはブロック図あるいはその両方のブロックに指定される機能／動作の態様を実施する命令を含んでいる製品を備えるように、コンピュータ可読記憶媒体に格納され、コンピュータ、プログラム可能なデータ処理装置、または他のデバイス、あるいはその組合せに特定の方式で機能するように指示できるものであってもよい。

【0351】

コンピュータ可読プログラム命令は、コンピュータ上、その他のプログラム可能な装置上、またはその他のデバイス上で実行される命令が、フローチャートまたはブロック図あるいはその両方のブロックに指定される機能／動作を実施するように、コンピュータ、その他のプログラム可能なデータ処理装置、またはその他のデバイスに読み込まれてもよく、それによって、一連の動作可能なステップを、コンピュータ上、その他のプログラム可能な装置上、またはコンピュータ実装プロセスを生成するその他のデバイス上で実行させる。

【0352】

図内のフローチャートおよびブロック図は、本発明のさまざまな実施形態に従って、システム、方法、およびコンピュータ・プログラム製品の可能な実装のアーキテクチャ、機能、および動作を示す。これに関連して、フローチャートまたはブロック図内の各ブロックは、規定された論理機能を実装するための1つまたは複数の実行可能な命令を備える、命令のモジュール、セグメント、または部分を表してよい。一部の代替の実装では、ブロックに示された機能は、図に示された順序とは異なる順序で発生してよい。例えば、連続して示された2つのブロックは、実際には、含まれている機能に応じて、実質的に同時に実行されるか、または場合によっては逆の順序で実行されてよい。ブロック図またはフローチャート図あるいはその両方の各ブロック、ならびにブロック図またはフローチャート図あるいはその両方に含まれるブロックの組合せは、規定された機能または動作を実行するか、または専用ハードウェアとコンピュータ命令の組合せを実行する専用ハードウェアベースのシステムによって実装され得るということにも注意する。

【0353】

上記に加えて、顧客の環境の管理を提供するサービス・プロバイダによって、1つまたは複数の態様が提供されること、提示されること、デプロイされること、管理されること、サービス提供されることなどが行われてよい。例えば、サービス・プロバイダは、1人

10

20

30

40

50

または複数人の顧客のために1つまたは複数の態様を実行するコンピュータ・コードまたはコンピュータ・インフラストラクチャあるいはその両方を作成すること、維持すること、サポートすることなどを行うことができる。その見返りとして、サービス・プロバイダは、例えばサブスクリプションまたは料金契約あるいはその両方の下で、顧客から支払いを受け取ってよい。追加または代替として、サービス・プロバイダは、1つまたは複数のサード・パーティへの広告コンテンツの販売から支払いを受け取ってよい。

【0354】

1つの態様では、1つまたは複数の実施形態を実行するために、アプリケーションがデプロイされてよい。1つの例として、アプリケーションのデプロイは、1つまたは複数の実施形態を実行するよう機能するコンピュータ・インフラストラクチャを提供することを含む。

10

【0355】

さらに別の態様として、コンピュータ可読コードをコンピューティング・システムに統合することを含むコンピューティング・インフラストラクチャがデプロイされてよく、このコンピューティング・インフラストラクチャでは、コンピューティング・システムと組み合わせたコードが、1つまたは複数の実施形態を実行できる。

【0356】

さらに別の態様として、コンピュータ可読コードをコンピュータ・システムに統合することを含んでいる、コンピューティング・インフラストラクチャを統合するためのプロセスが提供されてよい。コンピュータ・システムは、コンピュータ可読媒体を備え、このコンピュータ可読媒体では、コンピュータ媒体が1つまたは複数の実施形態を含む。コンピュータ・システムと組み合わせたコードは、1つまたは複数の実施形態を実行できる。

20

【0357】

上ではさまざまな実施形態が説明されたが、それらは単なる例である。例えば、1つまたは複数の実施形態を組み込んで使用するために、他のアーキテクチャのコンピューティング環境が使用され得る。さらに、さまざまな命令または動作が使用されてよい。さらに、さまざまなレジスタが使用されてよく、またはその他の種類の（レジスタ番号以外の）指示が指定されてよく、あるいはその両方が行われてよい。多くの変形が可能である。

【0358】

さらに、他の種類のコンピューティング環境が、恩恵を受けることができ、使用され得る。一例として、プログラム・コードの格納または実行あるいはその両方を行うのに適した、システム・バスを介して直接的または間接的にメモリ素子に結合された少なくとも2つのプロセッサを含んでいる、データ処理システムを使用できる。これらのメモリ素子は、例えば、プログラム・コードの実際の実行時に使用されるローカル・メモリ、バルク・ストレージ、および実行時にバルク・ストレージからコードが取得されなければならない回数を減らすために少なくとも一部のプログラム・コードを一時的に格納するキャッシュ・メモリを含む。

30

【0359】

入出力デバイスまたはI/Oデバイス（キーボード、ディスプレイ、ポインティング・デバイス、DASD、テープ、CD、DVD、サム・ドライブ、およびその他の記憶媒体などを含むが、これらに限定されない）は、直接的に、または介在するI/Oコントローラを通じて、システムに結合できる。ネットワーク・アダプタがシステムに結合され、介在するプライベート・ネットワークまたはパブリック・ネットワークを通じて、データ処理システムを、他のデータ処理システムまたはリモート・プリンタまたはストレージ・デバイスに結合できるようにしてもよい。モデム、ケーブル・モデム、およびイーサネット・カードは、使用可能なネットワーク・アダプタのうちの、ごくわずかの種類にすぎない。

40

【0360】

本明細書で使用される用語は、特定の実施形態を説明することのみを目的としており、制限することを意図していない。本明細書において使用されるとき、単数形「a」、「an」、および「the」は、特に明示的に示されない限り、複数形も含むことが意図され

50

ている。「備える」または「備えている」あるいはその両方の用語は、本明細書で 사용되는場合、記載された機能、整数、ステップ、動作、要素、またはコンポーネント、あるいはその組合せの存在を示すが、1つまたは複数のその他の機能、整数、ステップ、動作、要素、コンポーネント、またはこれらのグループ、あるいはその組合せの存在または追加を除外していないということが、さらに理解されるであろう。

【0361】

下の特許請求の範囲内のすべての手段またはステップおよび機能要素の対応する構造、材料、動作、および等価なものは、もしあれば、具体的に請求されるときに、その他の請求される要素と組み合わせて機能を実行するための任意の構造、材料、または動作を含むことが意図されている。1つまたは複数の実施形態の説明は、例示および説明の目的で提示されているが、網羅的であることは意図されておらず、開示された形態に限定されない。多くの変更および変形が可能であるということは、当業者にとって明らかであろう。さまざまな態様および実質的な適用を最も適切に説明するため、および他の当業者が、企図されている特定の用途に適しているようなさまざまな変更を伴う多様な実施形態を理解できるようにするために、実施形態が選択されて説明された。

10

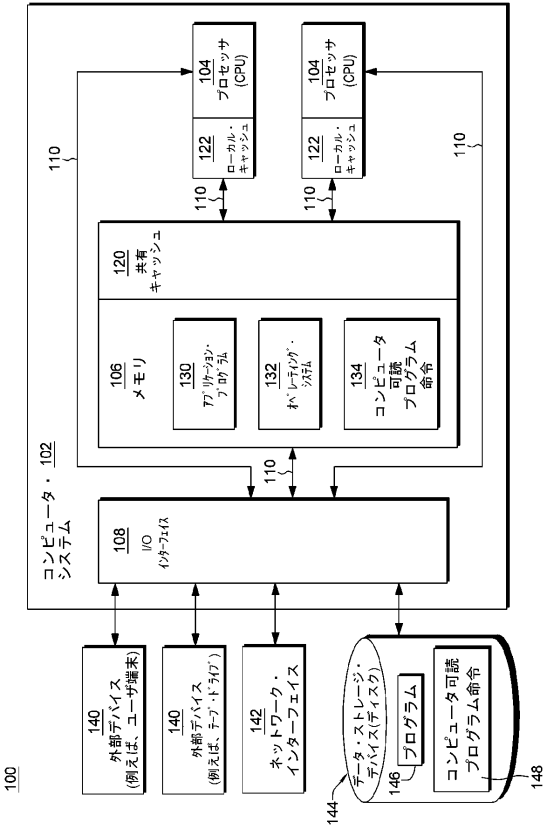
20

30

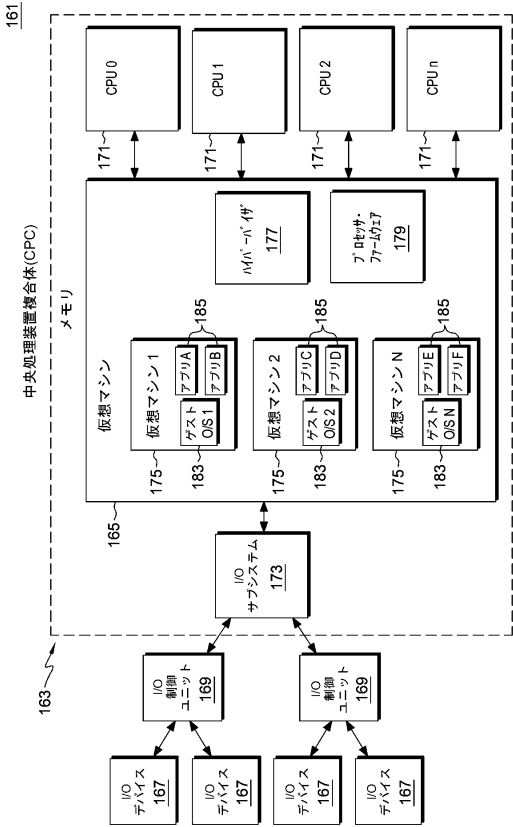
40

50

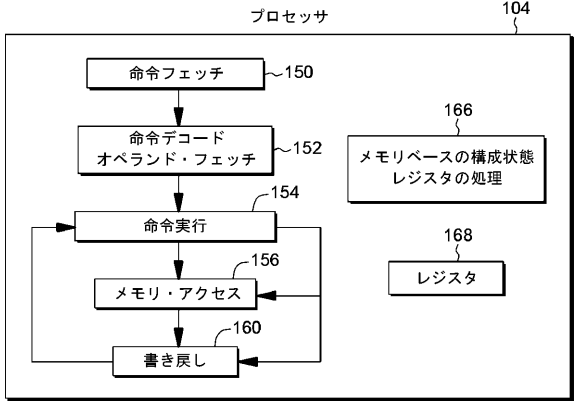
【図面】
【図 1】



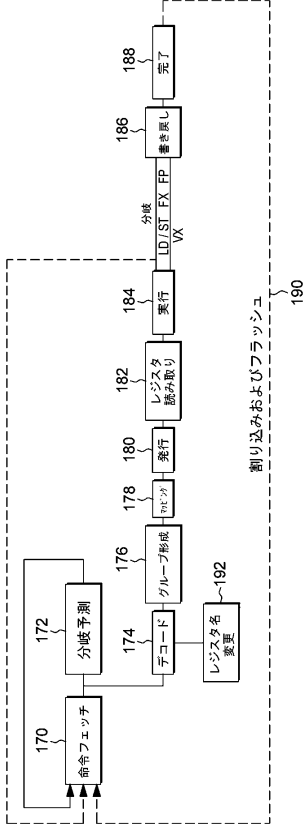
【図 2】



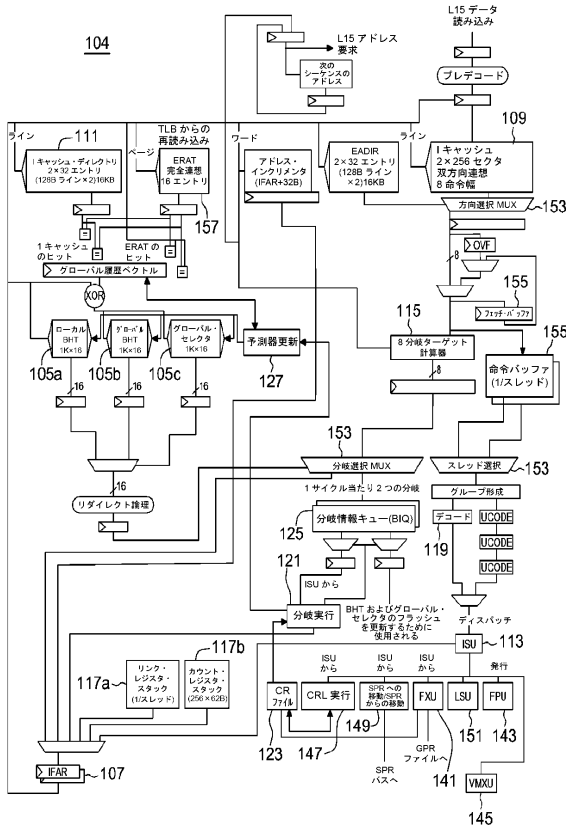
【図 3】



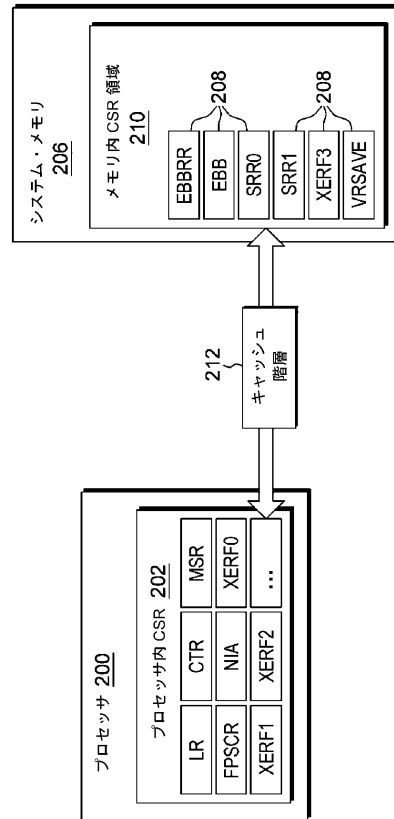
【図 4】



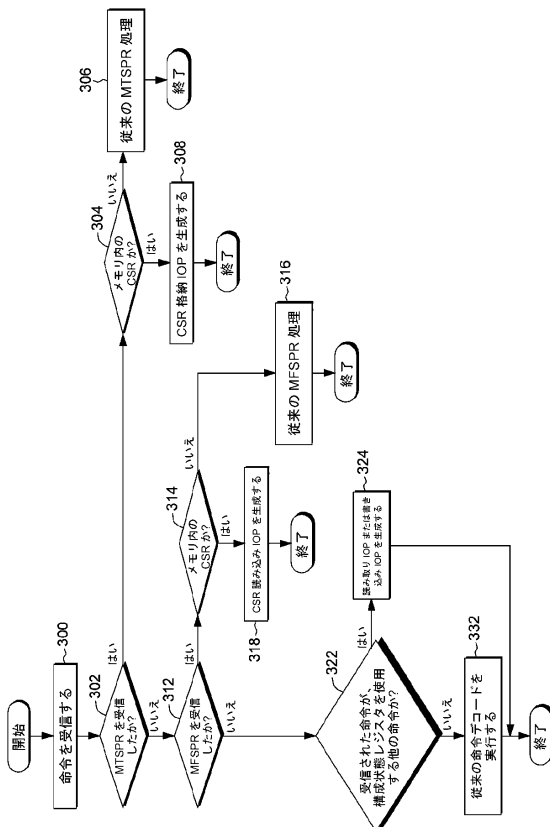
【図 5】



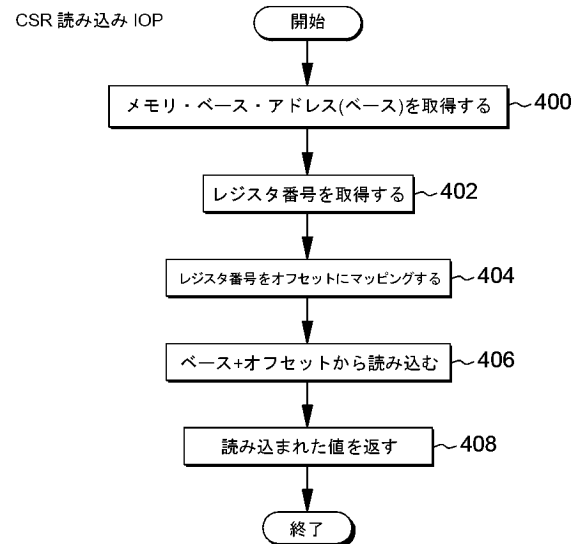
【図 6】



【図 7】



【図 8】



10

20

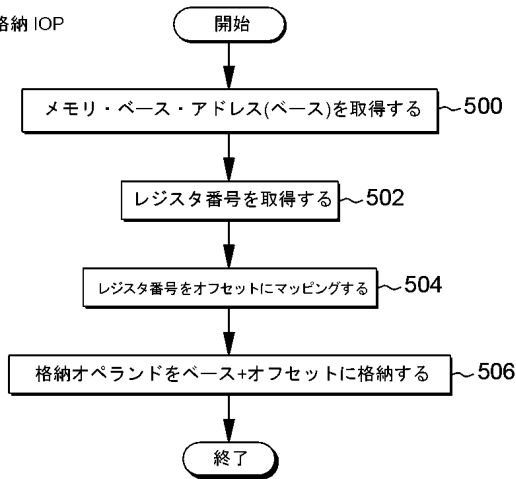
30

40

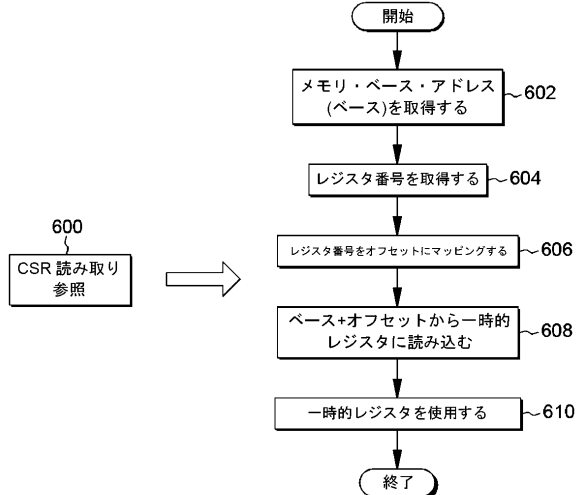
50

【図 9】

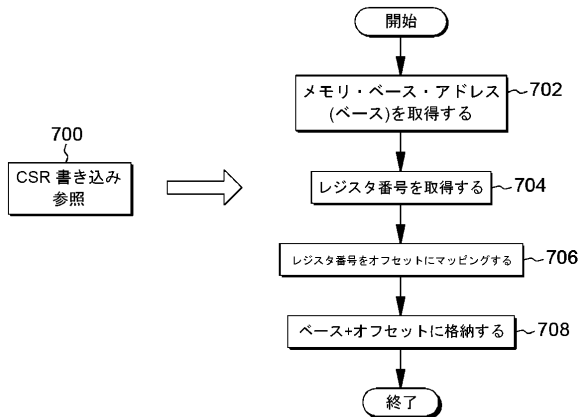
CSR 格納 IOP



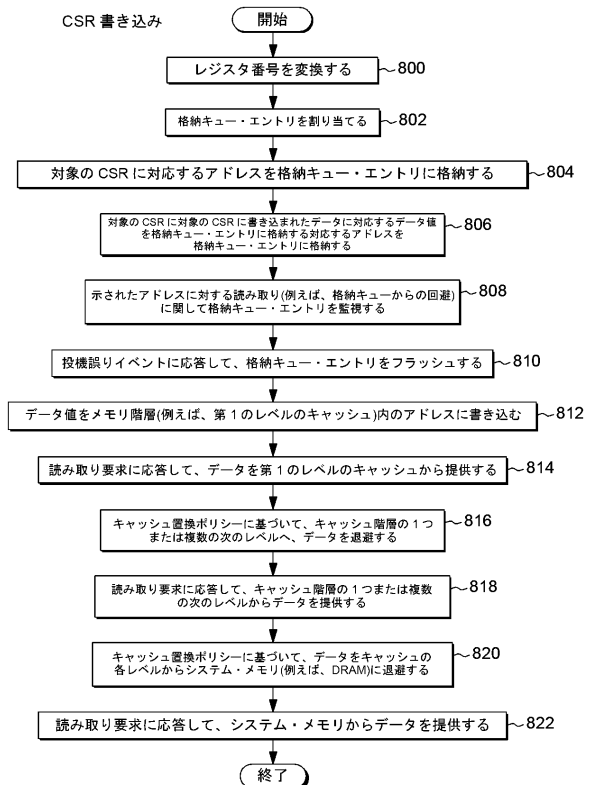
【図 10】



【図 11】



【図 12】



10

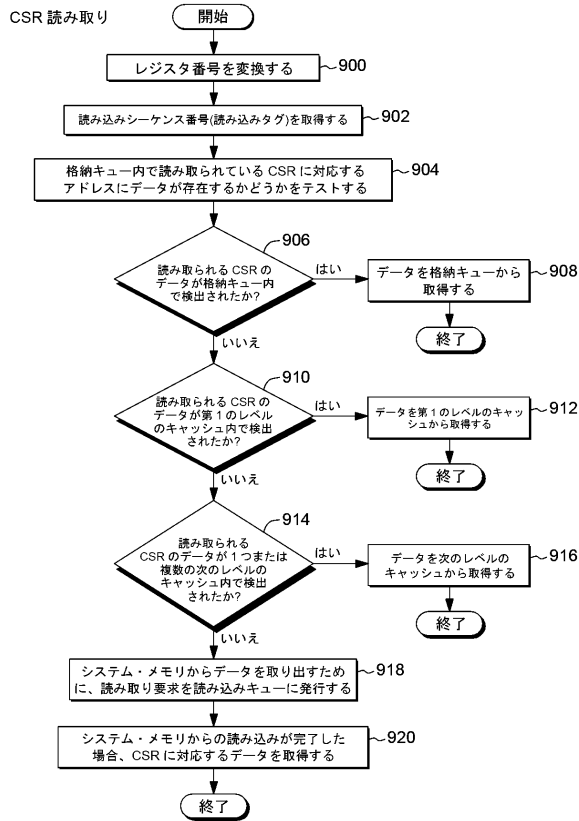
20

30

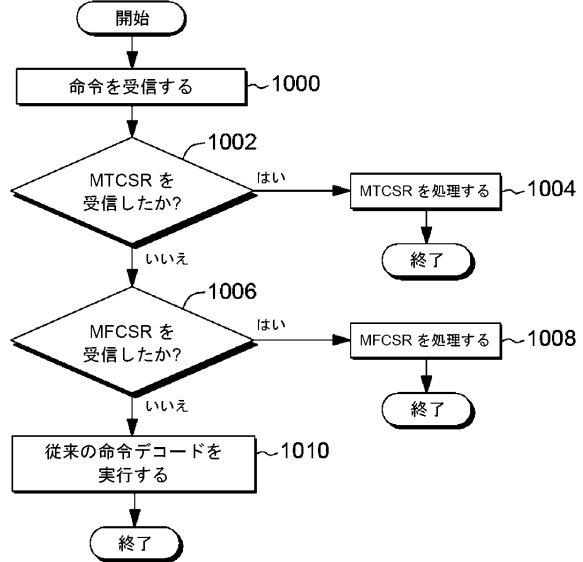
40

50

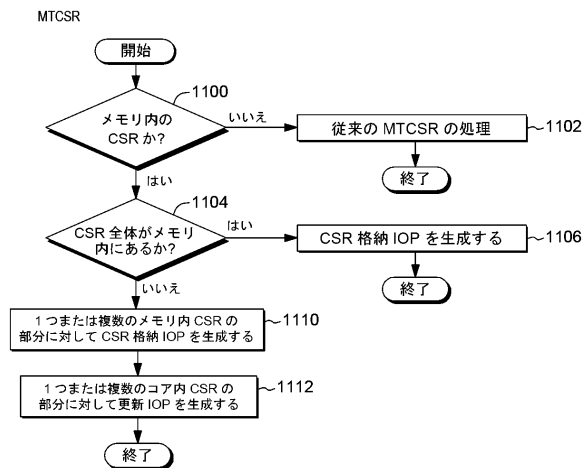
【図 13】



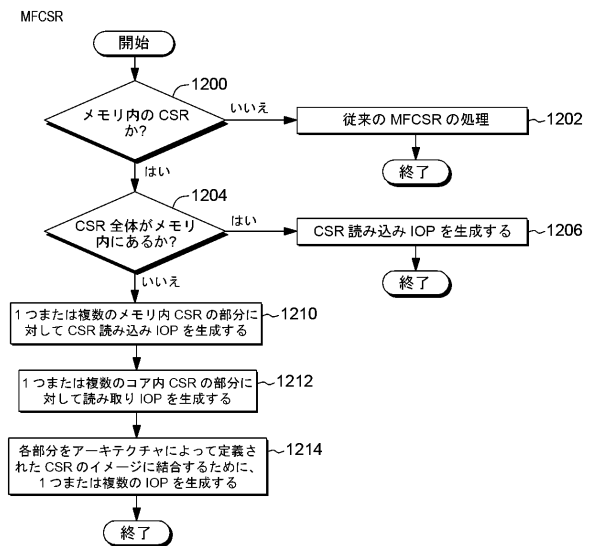
【図 14】



【図 15】



【図 16】



10

20

30

40

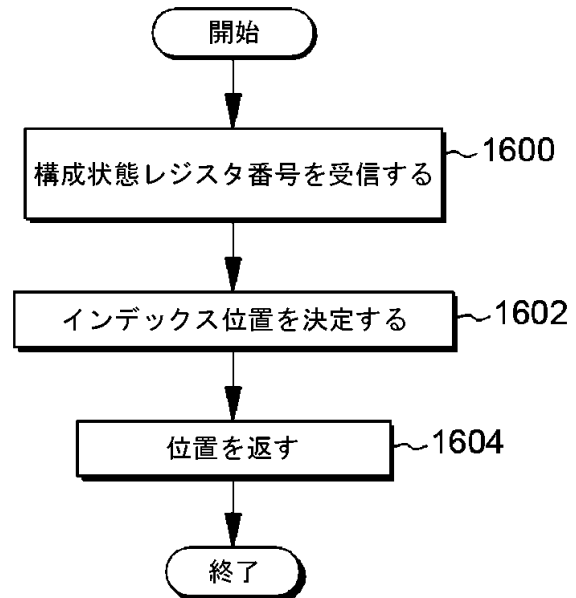
50

【図 2 1】

	-
779	MMCR0
780	SIAR
781	SDAR
782	MMCR1
	-
800	BESCRS
801	BESCRSU
802	BESCRR
803	BESCRRU
804	EBBHR
805	EBBRR
806	BESCR
	-
808	予備 3
809	予備 3
810	予備 3
811	予備 3
	-
813	LMRR
814	LMSER
815	TAR
	-
896	PPR10
	-
898	PPR32

1500

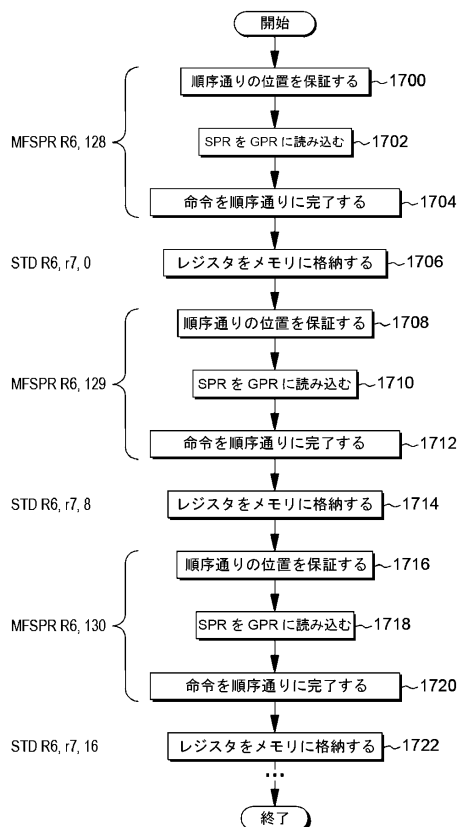
【図 2 2】



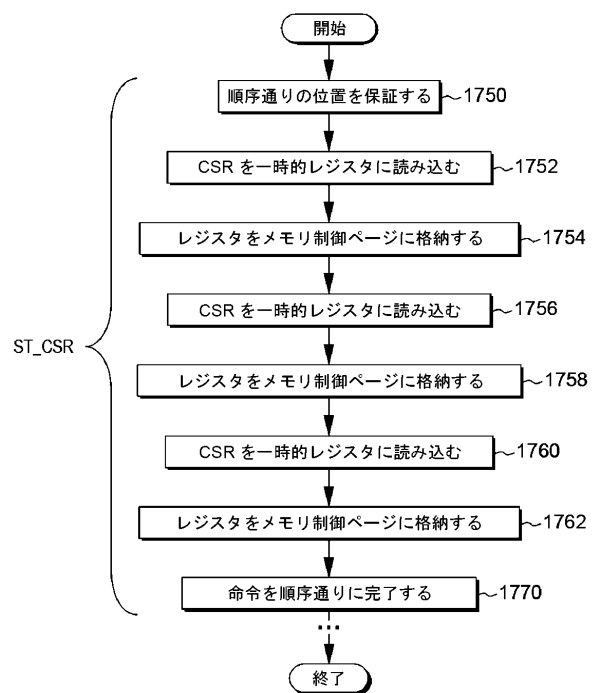
10

20

【図 2 3】



【図 2 4】

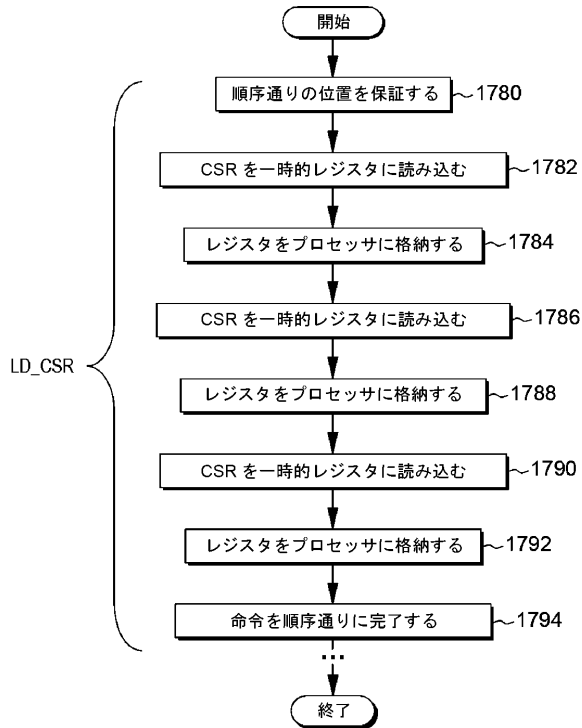


30

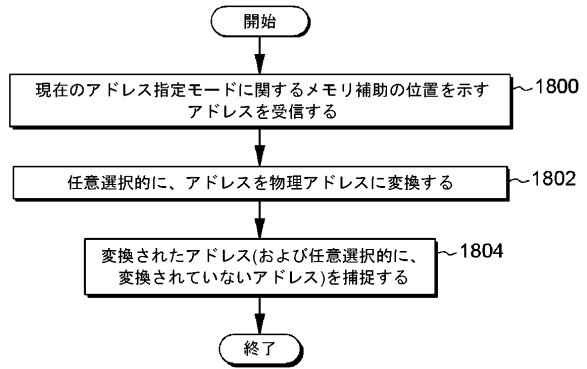
40

50

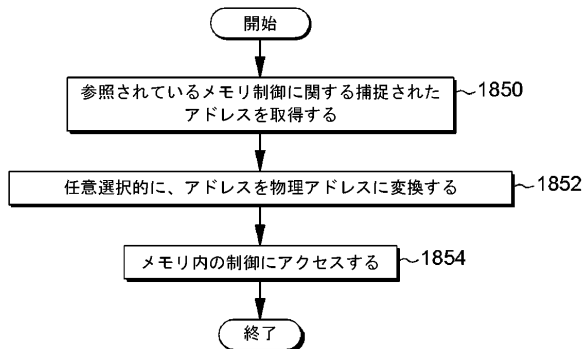
【図 25】



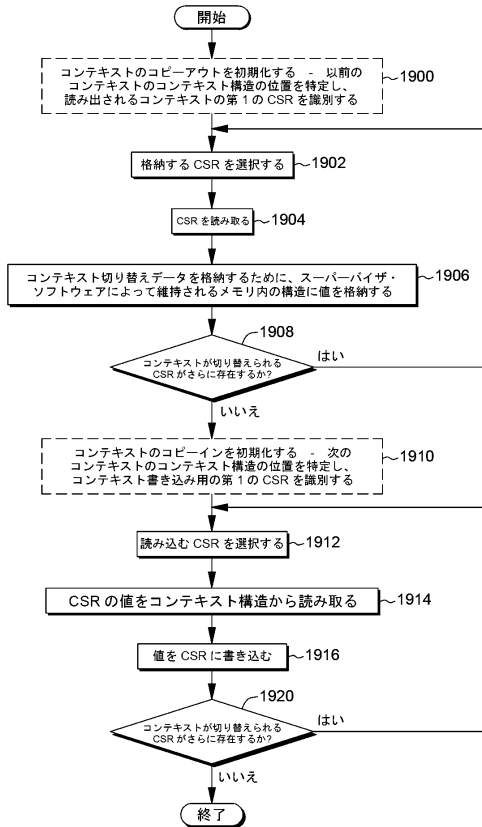
【図 26】



【図 27】



【図 28】



10

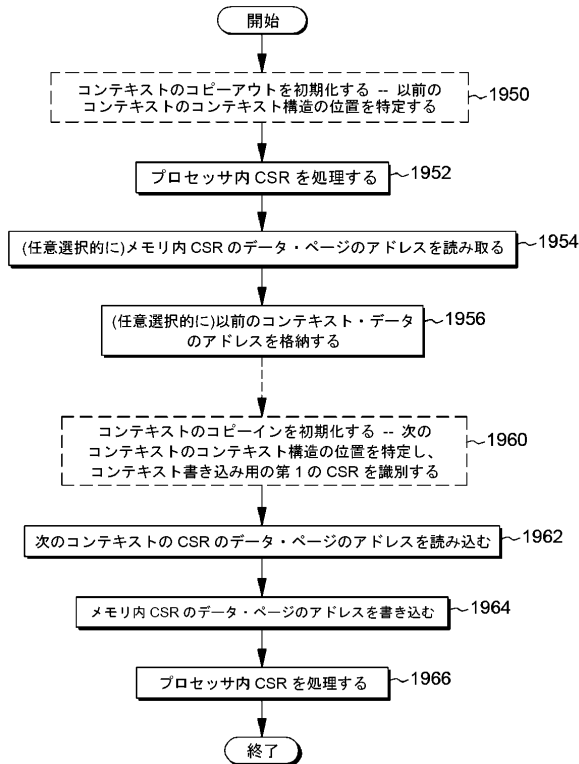
20

30

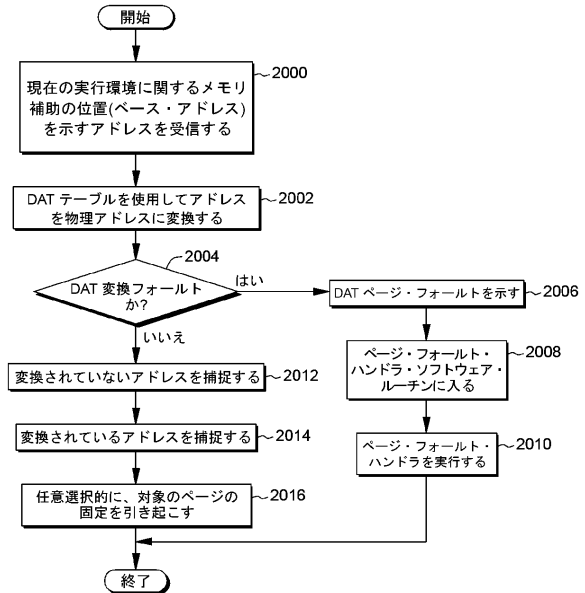
40

50

【図 29】

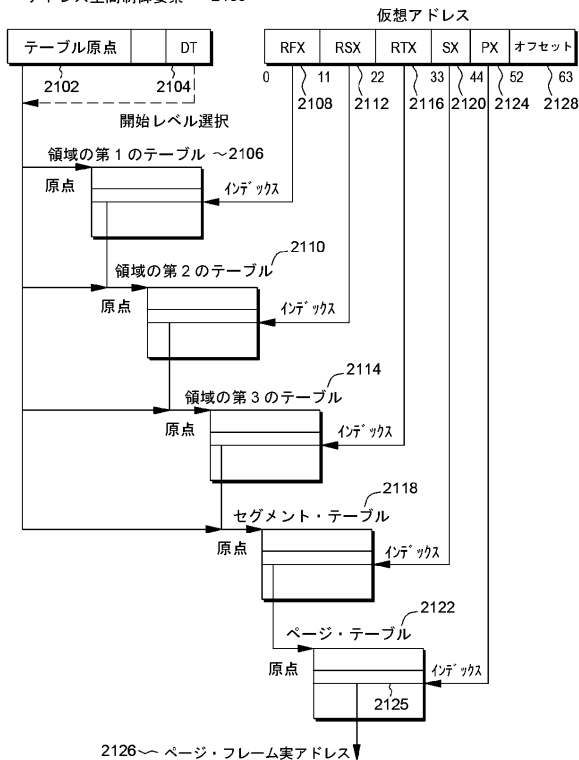


【図 30】

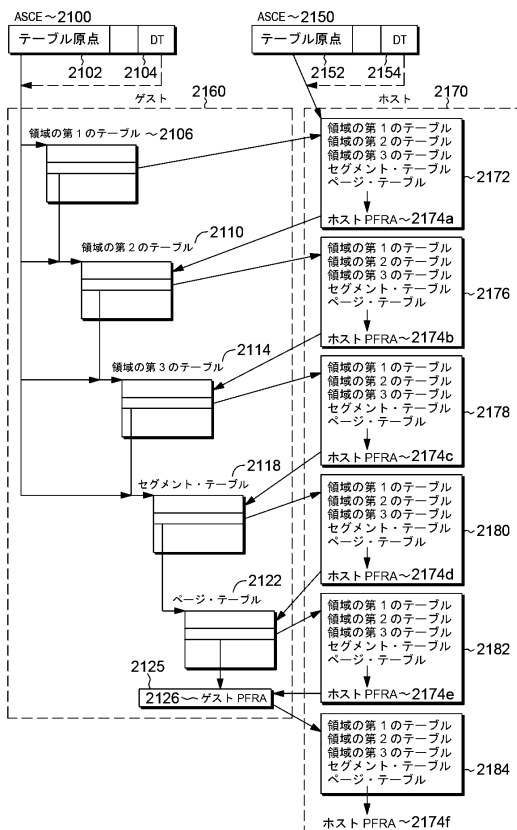


【図 31】

アドレス空間制御要素 ~2100



【図 32】



10

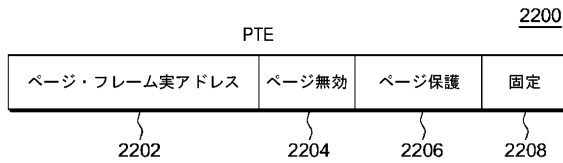
20

30

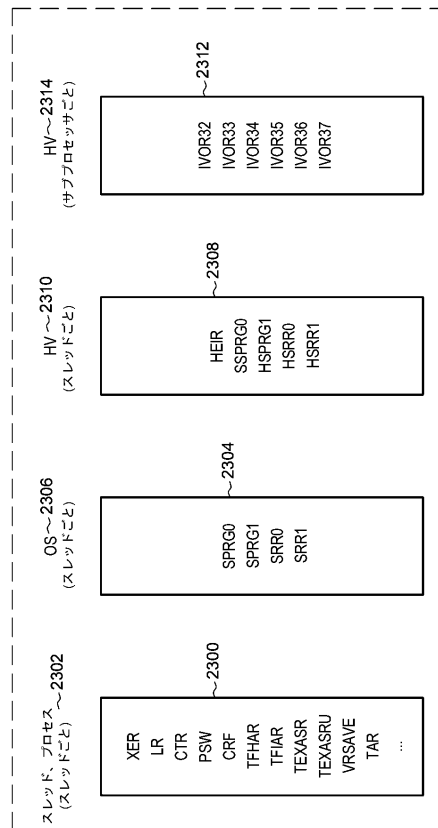
40

50

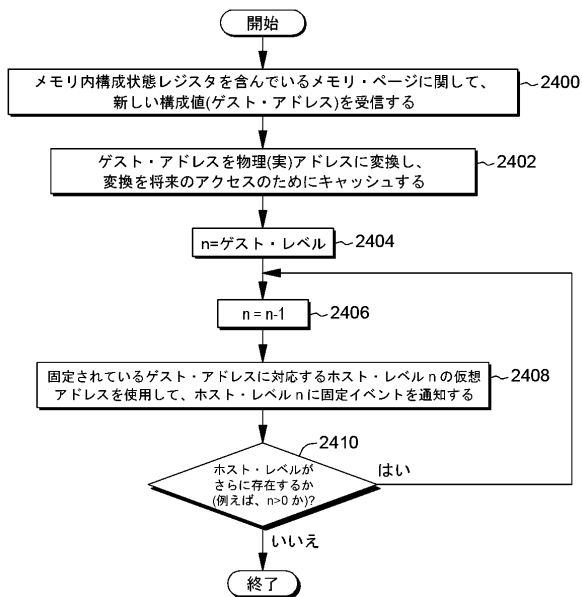
【図 3 3】



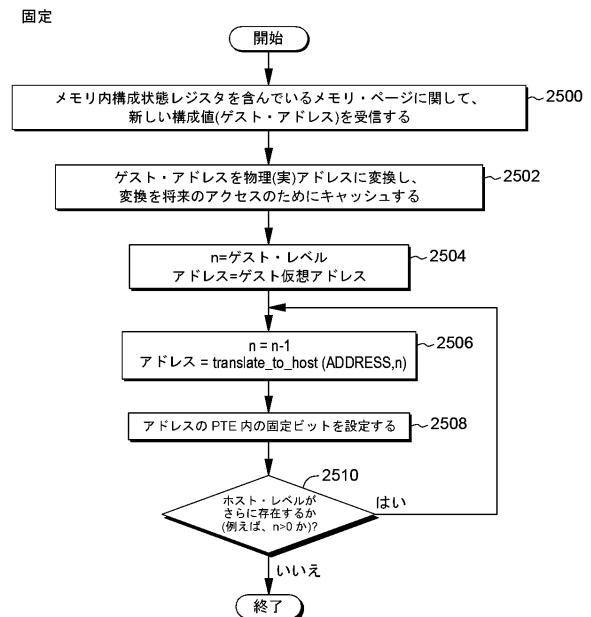
【図 3 4】



【図 3 5】



【図 3 6】



10

20

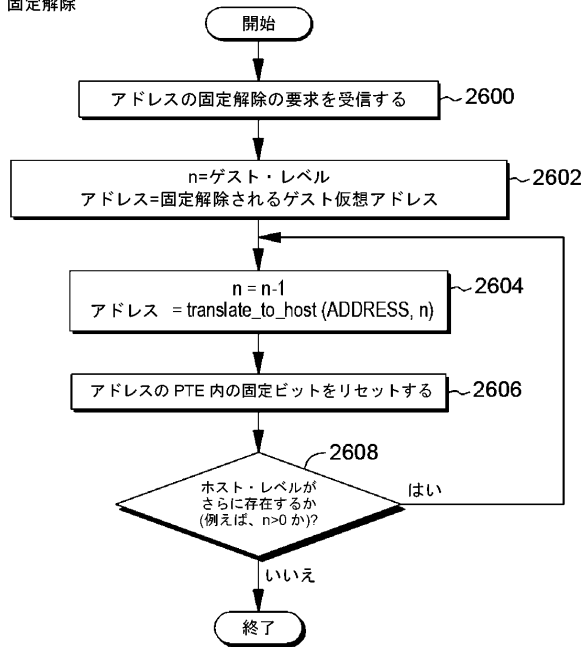
30

40

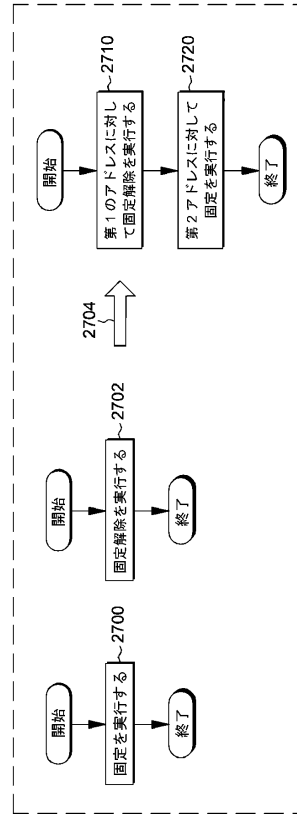
50

【図 37】

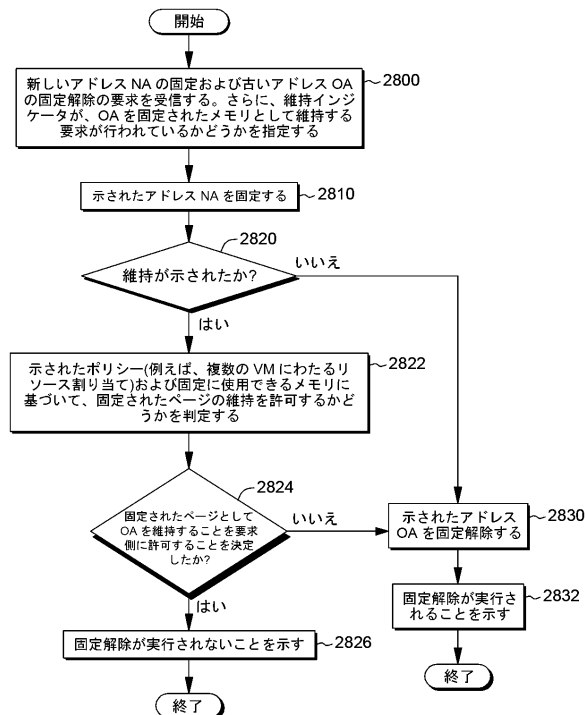
固定解除



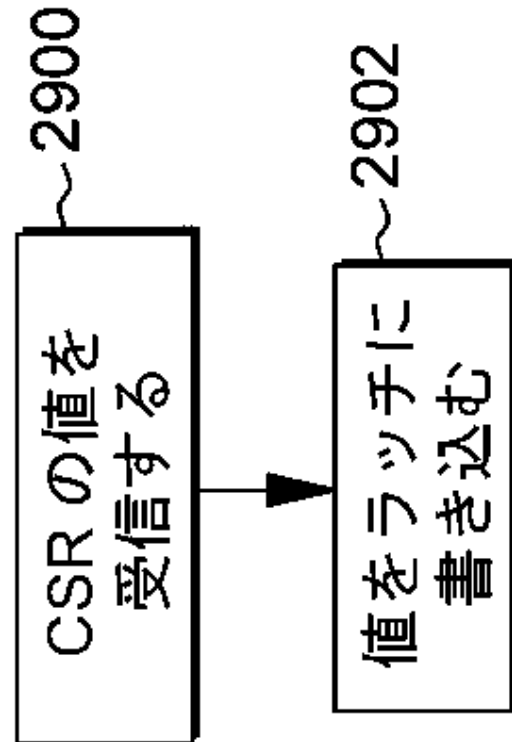
【図 38】



【図 39】



【図 40】



10

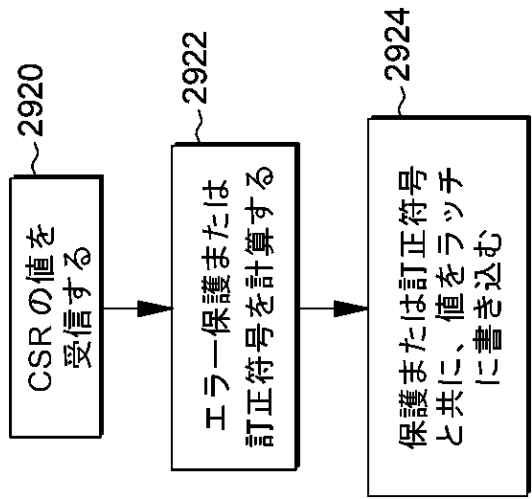
20

30

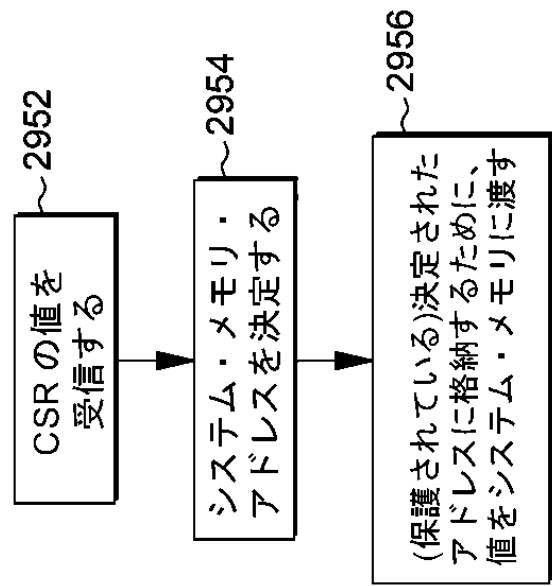
40

50

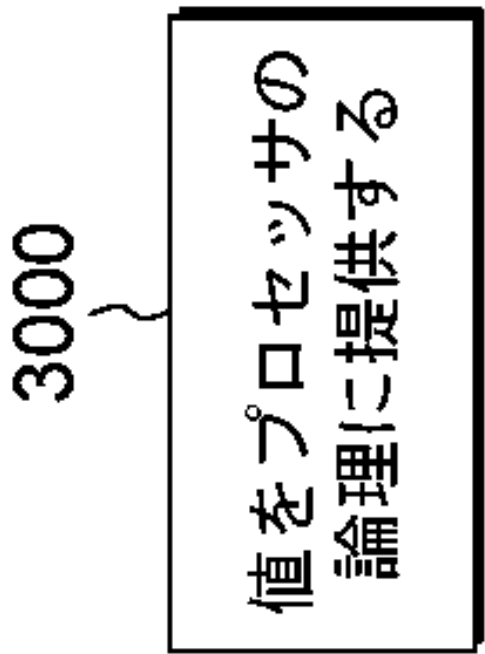
【図 4 1】



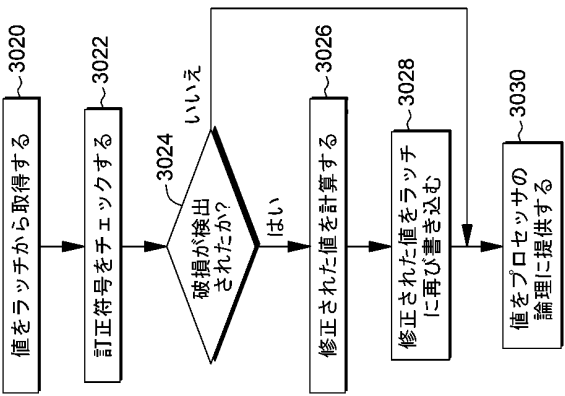
【図 4 2】



【図 4 3】



【図 4 4】



10

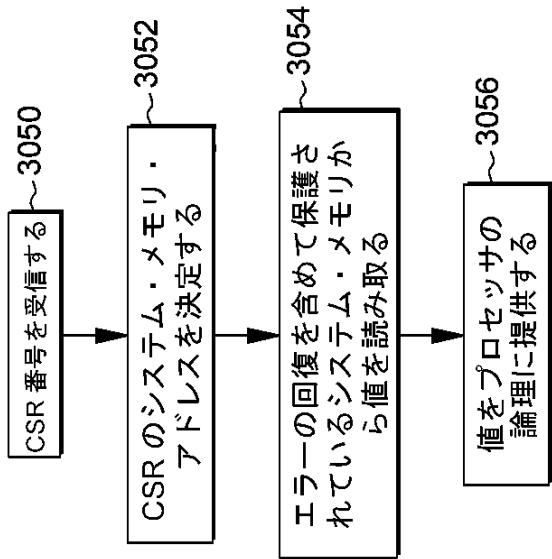
20

30

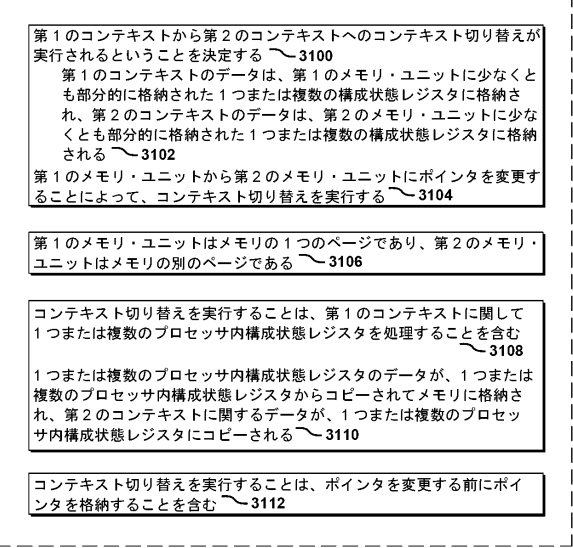
40

50

【図 4 5】

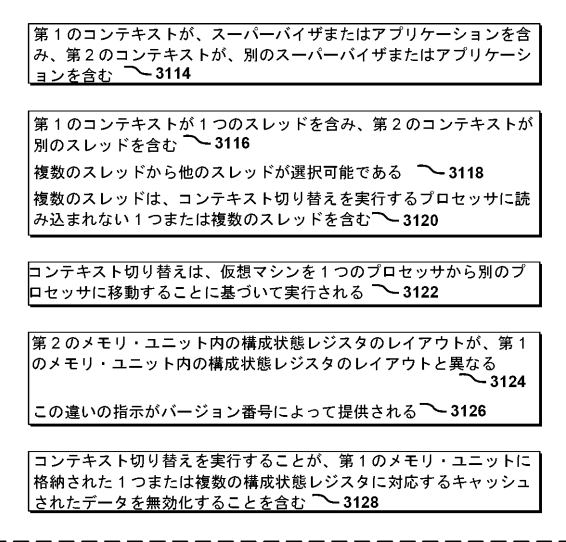


【図 4 6】

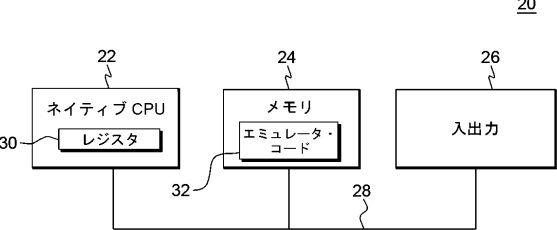


10

【図 4 7】



【図 4 8】



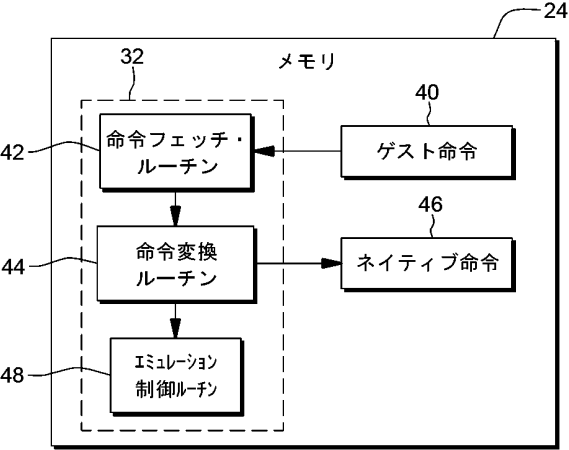
20

30

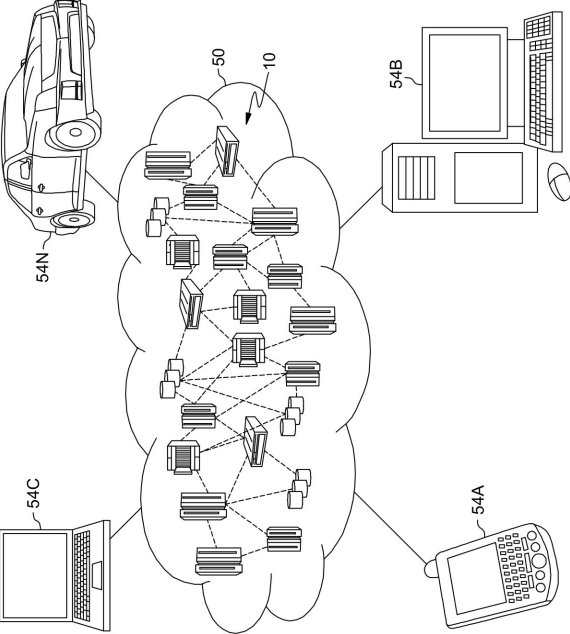
40

50

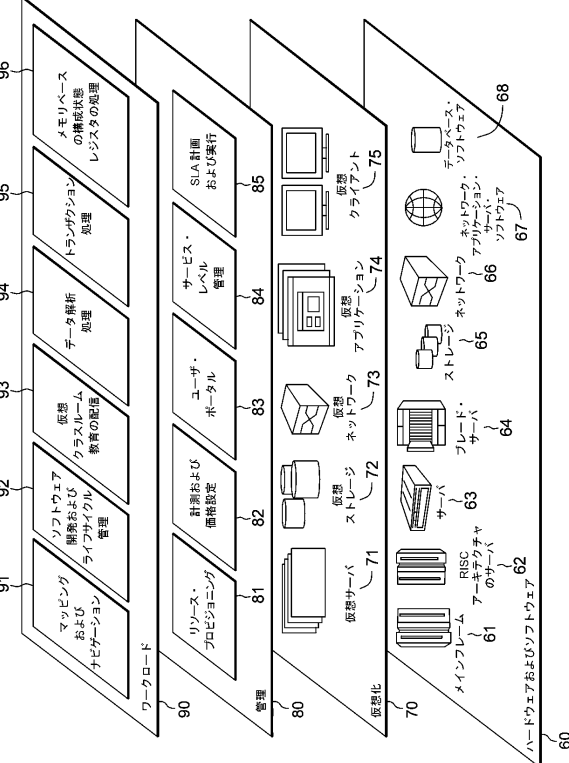
【図 49】



【図 50】



【図 51】



フロントページの続き

- (72)発明者 グシュヴィント、マイケル、カール
 アメリカ合衆国12601 ニューヨーク州ボキプシー サウス・ロード2455 エムエス ピー
 386
- (72)発明者 サラブラ、ヴァレンティナー
 アメリカ合衆国10598 ニューヨーク州ヨークタウン・ハイツ キッチャワン・ロード1101
- 審査官 三坂 敏夫
- (56)参考文献 特表2016-503934 (JP, A)
 特表2015-534188 (JP, A)
 特表2015-503784 (JP, A)
 米国特許出願公開第2014/0095847 (US, A1)
 特開2012-008767 (JP, A)
 特開2012-079357 (JP, A)
 インテル株式会社, ソフトウェア・ディベロッパーズ・マニュアル 下巻: システム・プロ
 グラミング・ガイド, CQ出版株式会社, 1997年, 6-1~6-20頁、7-1頁、15-9~15-10頁
- (58)調査した分野 (Int.Cl., DB名)
 G06F 9/455-9/54