(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2013-533545
(P2013-533545A)
(43)公表日　平成25年8月22日(2013.8.22)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/30	(2006.01)	G06F	9/30	350A	5B033
G06F	9/52	(2006.01)	G06F	9/46	472Z	
G06F	9/46	(2006.01)	G06F	9/46	350	

審査請求 未請求　予備審査請求 未請求　（全 54 頁）

(21)出願番号　特願2013-515730 (P2013-515730)
(86)(22)出願日　平成22年11月8日 (2010.11.8)
(85)翻訳文提出日　平成24年11月8日 (2012.11.8)
(86)国際出願番号　PCT/EP2010/067038
(87)国際公開番号　WO2011/160718
(87)国際公開日　平成23年12月29日 (2011.12.29)
(31)優先権主張番号　12/822,886
(32)優先日　平成22年6月24日 (2010.6.24)
(33)優先権主張国　米国 (US)

(71)出願人　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　ＢＵＳＩＮＥＳＳ　ＭＡＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
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
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一

最終頁に続く

(54)【発明の名称】処理を逐次化するための診断命令を実行する方法、システム及びプログラム

(57)【要約】
【課題】複数のプロセッサに同じリソースを更新させる環境において処理を容易化するシステム逐次化機能を提供する。
【解決手段】本システム逐次化機能は、多重処理環境における処理の容易化のために用いられ、ゲストおよびホストがロックを使って逐次化を提供する。本システム逐次化機能は、ホストがロックを取得した後発行される診断命令を含み、ゲストがロックを取得する必要性を排除する。
【選択図】図３

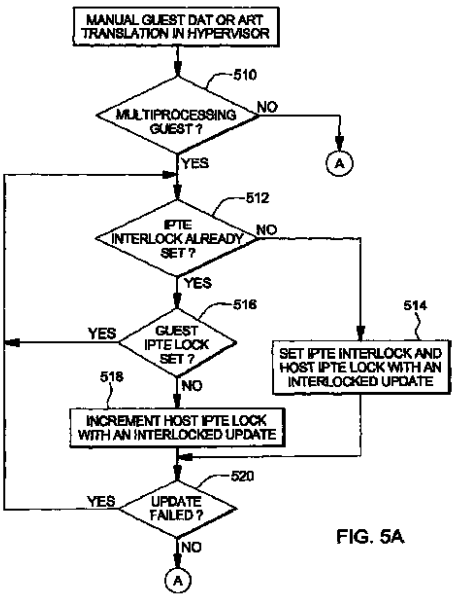


FIG. 5A

【特許請求の範囲】
【請求項１】
コンピューティング環境の処理を逐次化するための診断命令を実行する方法であって、前記方法は、
実行のためマシン命令を取得するステップであって、前記マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、前記マシン命令は、
診断命令を識別するオペコード・フィールド、
前記診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および
第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために前記診断命令が用いられることを識別するのに使われるオペレーション・コード拡張を得るための、前記第二位置を識別するベース・フィールド、
を含む、前記取得するステップと、
前記オペレーション・コード拡張によって示された前記マシン命令を実行するステップであって、
前記サブコードが所定の値であるのに応じて、
前記コンピューティング環境のプロセッサの静止化を開始するステップ、
前記プロセッサが静止されていることを判定するステップ、および
前記プロセッサが静止されるのに応じて、前記診断命令の実行を完了するステップ、
を含む、前記実行するステップと、
を含む前記方法。
【請求項２】
前記マシン命令は、システム・コール領域のアドレスが提供される第三位置を識別する制御領域フィールドをさらに含む、請求項１に記載の方法。
【請求項３】
前記サブコードが前記所定の値であるのに応じて、前記システム・コール領域の前記アドレスは、変換を実施する対象のゲストを識別し、前記方法は、前記ゲストに対する変換が実施されるのに応じて、前記マシン命令を発行するステップをさらに含む、請求項２に記載の方法。
【請求項４】
前記方法は、
前記ゲストが多重処理ゲストかどうかを判定するステップと、
前記ゲストが多重処理ゲストであるのに応じて、インターロックがセットされているかどうかを判定するステップと、
前記インターロックがセットされていないのに応じて、前記インターロックおよびホスト・ロックをセットするステップと、
前記インターロックがセットされているのに応じて、前記ホスト・ロックを更新するステップと、
前記ホスト・ロックが成功裏にセットまたは更新されるのに応じて、前記マシン命令を発行するステップと、
をさらに含む、請求項３に記載の方法。
【請求項５】
前記ホスト・ロックは、ゲスト・ロックにかかわりなくセットされる、請求項４に記載の方法。
【請求項６】
前記方法は、ホスト・ロックをセットするステップと、それに応じて前記マシン命令を発行するステップとをさらに含む、請求項１に記載の方法。
【請求項７】
前記マシン命令の前記実行は、前記ホスト・ロックを、前記コンピューティング環境のプロセッサ群に可視にするステップを含む、請求項６に記載の方法。

【請求項８】
前記診断命令の実行を完了するステップは、前記静止状態を出るステップを含む、請求項１に記載の方法。
【請求項９】
前記方法は、
ゲストによって、ロックを用いる静止型の命令を発行するステップであって、前記静止型命令は、このゲストに対するホストによって遂行される変換処理を使って逐次化されることになる、前記発行するステップと、
ホスト・ロックがセットされているかどうかを判定するステップと、
前記ホスト・ロックがセットされていないのに応じて、ゲスト・ロックの使用なしに、静止型命令を発行するステップと、
をさらに含む、請求項１に記載の方法。
【請求項１０】
前記ホスト・ロックがセットされていて、前記マシン命令の実行が完了されていないのが示されているのに応じて、前記ホストにインターセプトする、請求項９に記載の方法。
【請求項１１】
前記方法は、
第一エンティティと第二エンティティとによって共用されるインターロックをセットするステップと、
前記インターロックのセットに応じて、第一エンティティ・ロックを更新するステップと、
前記第一エンティティ・ロックを成功裏に更新するのに応じて、前記マシン命令を発行するステップであって、前記マシン命令は、第二エンティティ・ロックのセッティングなしで、前記第一エンティティの処理と前記第二エンティティの処理との間で逐次化を遂行する、前記発行するステップと、
をさらに含む、請求項１に記載の方法。
【請求項１２】
コンピューティング環境の処理を逐次化する診断命令を実行するためのコンピュータ・システムであって、前記コンピュータ・システムは
メモリと、
前記メモリと通信するプロセッサと、
実行のためのマシン命令を取得するオブテイナであって、前記マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、前記マシン命令が、
・　診断命令を識別するオペコード・フィールド、
・　前記診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および
・　第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために前記診断命令が用いられることを識別するのに使われる、オペレーション・コード拡張を得るための前記第二位置を識別するベース・フィールド、
を含む、前記オブテイナと、
前記オペレーション・コード拡張によって示された前記マシン命令を実行するためのエグゼキュタであって、前記サブコードが所定の値であるのに応じて、
・　前記コンピューティング環境のプロセッサの静止化を開始するためのイニシエータ、
・　前記プロセッサが静止されているのを判定するためのデターミナ、および
・　前記プロセッサが静止されているのに応じて、前記診断命令の実行を完了するための実行コンプリータ、
を含む手段を包含する、前記エグゼキュタと、
を含む、前記コンピュータ・システム。

【請求項１３】
コンピュータ・システムにロードされ、その上で実行されるとき、前記コンピュータ・システムに請求項１～１１のいずれかに記載の方法の全ステップを遂行させる、コンピュータ可読媒体に格納されたコンピュータ・プログラム・コードを含む、コンピュータ・プログラム。
【請求項１４】
コンピューティング環境の処理を逐次化する診断命令を実行するためのコンピュータ・システムであって、前記コンピュータ・システムは、
メモリと、
前記メモリと通信するプロセッサと、
を含み、
前記コンピュータ・システムは方法を遂行するよう構成され、前記方法は、
実行のためマシン命令を取得するステップであって、前記マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、前記マシン命令が、
診断命令を識別するオペコード・フィールド、
前記診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および
第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために前記診断命令が用いられることを識別するのに使われる、オペレーション・コード拡張を得るための前記第二位置を識別するベース・フィールド、
を含む、前記取得するステップと、
前記オペレーション・コード拡張によって示された前記マシン命令を実行するステップであって、
前記サブコードが所定の値であるのに応じて、
前記コンピューティング環境のプロセッサの静止化を開始するステップ、
前記プロセッサが静止されていることを判定するステップ、および
前記プロセッサが静止されているのに応じて、前記診断命令の実行を完了するステップ、
を含む、前記実行するステップと、
を含む、前記コンピュータ・システム。
【請求項１５】
前記マシン命令は、システム・コール領域のアドレスが提供される第三位置を識別する制御領域フィールドをさらに含む、請求項１４に記載のコンピュータ・システム。
【請求項１６】
前記サブコードが前記所定の値であるのに応じて、前記システム・コール領域の前記アドレスは、変換を実施する対象のゲストを識別し、前記方法は、前記ゲストに対する変換が実施されるのに応じて、前記マシン命令を発行するステップをさらに含む、請求項１５に記載のコンピュータ・システム。
【請求項１７】
前記方法は、
前記ゲストが多重処理ゲストかどうかを判定するステップと、
前記ゲストが多重処理ゲストであるのに応じて、インターロックがセットされているかどうかを判定するステップと、
前記インターロックがセットされていないのに応じて、前記インターロックおよびホスト・ロックをセットするステップと、
前記インターロックがセットされているのに応じて、前記ホスト・ロックを更新するステップと、
前記ホスト・ロックが成功裏にセットまたは更新されるのに応じて、前記マシン命令を発行するステップであって、前記ホスト・ロックはゲスト・ロックにかかわりなくセットされる、前記発行するステップと、

をさらに含む、請求項１４に記載のコンピュータ・システム。
【請求項１８】
前記方法は、ホスト・ロックをセットするステップと、それに応じて前記マシン命令を発行するステップとをさらに含み、前記マシン命令の前記実行は、前記ホスト・ロックを、前記コンピューティング環境のプロセッサ群に可視にするステップを含む、請求項１４に記載のコンピュータ・システム。
【請求項１９】
前記方法は、
ゲストによって、ロックを用いる静止型の命令を発行するステップであって、前記静止型命令は、このゲストに対するホストによって遂行される変換処理を使って逐次化されることになる、前記発行するステップと、
ホスト・ロックがセットされているかどうかを判定するステップと、
前記ホスト・ロックがセットされていないのに応じて、ゲスト・ロックの使用なしに、静止型命令を発行するステップと、
をさらに含む、請求項１４に記載のコンピュータ・システム。
【請求項２０】
前記ホスト・ロックがセットされていて、前記マシン命令の実行が完了されていないのが示されているのに応じて、前記ホストにインターセプトする、請求項１９に記載のコンピュータ・システム。
【請求項２１】
前記方法は、
第一エンティティと第二エンティティとによって共用されるインターロックをセットするステップと、
前記インターロックのセットに応じて、第一エンティティ・ロックを更新するステップと、
前記第一エンティティ・ロックを成功裏に更新するのに応じて、前記マシン命令を発行するステップであって、前記マシン命令は、第二エンティティ・ロックをセットしないで、前記第一エンティティの処理と前記第二エンティティの処理との間で逐次化を遂行する、前記発行するステップと、
をさらに含む、請求項１４に記載のコンピュータ・システム。
【請求項２２】
コンピューティング環境の処理を逐次化する診断命令を実行するためのコンピュータ・プログラム製品であって、前記コンピュータ・プログラム製品は、
処理回路に可読で、請求項１～１１のいずれかに記載の方法を遂行するため前記処理回路が実行する命令を格納する、コンピュータ可読ストレージ媒体を含む、
前記コンピュータ・プログラム製品。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般にコンピューティング環境内での処理に関し、具体的には多重処理コンピューティング環境における処理の逐次化（ｓｅｒｉａｌｉｚｉｎｇ）に関する。
【背景技術】
【０００２】
ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）に基づくマシンにおいては、通常、プログラム（本文脈ではゲストという）がＬＰＡＲ（ｌｏｇｉｃａｌ　ｐａｒｔｉｔｉｏｎ：論理区画モード）またはｚ／ＶＭ（ＩＢＭ社の登録商標）の下で実行されているとき、ゲストの命令はハードウェアまたはファームウェアによって実行される。この命令の実行は、多くの場合、オペランド・アドレスの変換およびそれらアドレスとのデータ・フェッチおよび格納のやり取りを含む。各命令が実行されている間

、ハードウェアは、別のプロセッサから入来し、その命令に関連付けられた変換を無効にしかねない、一切のページ・テーブル・エントリ無効化（ＩＰＴＥ：ｉｎｖａｌｉｄａｔｅ ｐａｇｅ ｔａｂｌｅ ｅｎｔｒｙ）、動的アドレス変換（ＤＡＴ：ｄｙｎａｍｉｃ ａｄｄｒｅｓｓ ｔｒａｎｓｌａｔｉｏｎ）テーブル・エントリ無効化（ＩＤＴＥ：ｉｎｖａｌｉｄａｔｅ ＤＡＴ ｔａｂｌｅ ｅｎｔｒｙ）、または比較／スワップ／パージ（ＣＳＰ／Ｇ：ｃｏｍｐａｒｅ ａｎｄ ｓｗａｐ ａｎｄ ｐｕｒｇｅ）要求が、該命令が完了するまで待つことを保証する。同様に、未処理のＩＰＴＥ、ＩＤＴＥまたはＧＳＰ／Ｇに適用され得るいかなる変換アクセスも格納アクセスも、これらＩＰＴＥ、ＩＤＴＥまたはＧＳＰ／Ｇのオペレーションが完了するまでは実施されない。

【０００３】

ＬＰＡＲまたはｚ／ＶＭ（ＩＢＭ社の登録商標）のハイパーバイザが、ゲストのために、ある命令をエミュレートする必要がある場合がある。これを行うために、ハイパーバイザは、その命令に関連付けられたオペランド・アドレスをマニュアルで変換し（すなわち、ハードウェアが不知の多段階の変換を実施し）、引き続き、それらの変換に基づいて、ストレージ・アクセスを行う必要がある。ゲスト多重処理（ＭＰ：ｍｕｌｔｉ－ｐｒｏｃｅｓｓｉｎｇ）構成中の一つのプロセッサにより発行されたＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇと、その同一構成中の別のプロセッサのためハイパーバイザが遂行中の命令エミュレーションとの間に衝突が生じないことを保証するために、シングル・インターロックが使われる。このロックはＩＰＴＥインターロックといわれ、特定の仮想構成に対する全てのＭＰゲストの間で共用される制御ブロックである、システム制御領域（ＳＣＡ：Ｓｙｓｔｅｍ Ｃｏｎｔｒｏｌ Ａｒｅａ）に配置されている。命令エミュレーションの期間は、ハイパーバイザがロックを保持し、ＩＰＴＥ、ＩＤＴＥ、またはＣＳＰ／Ｇが実行されているときはファームウェアがロックを保持することになる。

【０００４】

パフォーマンスを向上するために、このシングル・ロックは、２部分の共用ロックに分割されている。ＩＰＴＥインターロックの一つの部分は、ハイパーバイザまたはホストによって維持される共用ロックであり、命令のエミュレーションの期間はハイパーバイザによって保持されている。他方の部分は、ゲストのためにファームウェアが維持しており、ＩＰＴＥ、ＩＤＴＥ、またはＣＳＰ／Ｇが実行されている間はファームウェアに保持される。各ロックは、現在ロックを保持しているホストまたはゲスト・プロセッサのカウントである。ロックの整合性を確実にするため、ハイパーバイザまたはファームウェアは、他方の（それぞれ、ゲストまたはホスト）ロックが保持されていない場合にだけ、比較／スワップ（例えば、Ｃｏｍｐａｒｅ ａｎｄ Ｓｗａｐ（ＣＳＧ）命令）を使って、妥当な（それぞれ、ゲストまたはホスト）ロック・カウントをインクリメントする。ホストが、ＩＰＴＥインターロックがゲストによって保持されていると判定した場合、ホストは、ロックが利用可能になるまで、命令エミュレーションが進むのを待つ。ゲストが、ホスト・ロックが保持されていることを検知した場合、ゲストは、（命令インターセプトを使い）インターセプトし、ホストへ戻す。

【０００５】

１９９８年９月１５日発行の特許文献１、Ｂｌａｎｄｙの「Ｐｅｎｄｉｎｇ Ｐａｇｅ Ｒｅｌｅａｓｅ」は、コンピュータ・システムのゲスト・プログラムによってページの解放が要求された後、システムによる実際のページ解放を遅延させ、ページ解放の保留を生じさせることによって、コンピュータ・システム中の仮想ストレージのページを管理するための方法、装置、および製造品を記載している。ページ解放の保留が確定されると、保留されたページ解放がログ・エントリされ、それらページ（群）はゲストに割り当てたままとなる。システムはログ中のページをいつでも発行することができる。ゲストは、保留されたページ解放のキャンセルを要求することができ、その場合、要求されたページがシステムによって解放済みでなければ、それらページはゲストによる再使用のためにログから除去される。鍵設定、ページ消去または初期化、およびページ検証または移動は、随意的サービスであり、これらをキャンセル要求に含めることができる。確定およびキャン

セルは、望ましくは、ゲストによって、インターセプト命令ＤＩＡＧＮＯＳＥ（診断）を用いて要求され、ホスト・オペレーティング・システムのストレージ・マネージャ・コンポーネントによって実施される。保留されたページ解放のキャンセルは、ＤＩＡＧＮＯＳＥ命令を用いて要求されるが、専用のマシン命令で置き換えることもできる。システムは、保留されたページ解放要求のログを処理して、保留ページを定型法で解放する。

【０００６】

２００４年４月１日に発行された特許文献２、Ｆａｒｒｅｌｌらの「Ｍｅｔｈｏｄ ａｎｄ Ａｐｐａｒａｔｕｓ ｆｏｒ Ｃｏｎｔｒｏｌｌｉｎｇ ｔｈｅ Ｅｘｅｃｕｔｉｏｎ ｏｆ ａ Ｂｒｏａｄｃａｓｔ Ｉｎｓｔｒｕｃｔｉｏｎ ｏｎ ａ Ｇｕｅｓｔ Ｐｒｏｃｅｓｓｏｒ ｏｆ ａ Ｇｕｅｓｔ Ｍｕｌｔｉｐｒｏｃｅｓｓｉｎｇ Ｃｏｎｆｉｇｕｒａｔｉｏｎ」は、ゲスト・マシンの他のプロセッサによる対応オペレーションを要求する一斉通信命令の、ゲスト・プロセッサでの実行を管理するための方法および装置を記載している。情報処理システムの複数のプロセッサの各々は、ホスト・マシン上で実行されるホスト・プログラムの制御下にあるホスト・プロセッサ、または、ゲスト・マシン上で実行されるゲスト・プログラムの制御下にあるゲスト・プロセッサのいずれとしても動作可能である。ゲスト・マシンは、ホスト・マシン上で実行されるホスト・プログラムによって定義され、ゲストの多重処理構成を形成するかかる複数のゲスト・プロセッサを包含する。ゲスト・マシンに対するロックが定義され、これは、ロックがホスト・プログラムのホストのロック・ホルダによって保持されているかどうかの表示、およびゲストのロック・ホルダとしてロックを保持しているプロセッサの数のカウントを包含する。ゲスト・プロセッサとして動作しているプロセッサ上で実行される一斉通信命令が復号され次第、ロックが検査され、それがホストのロック・ホルダによって保持されているかどうかが判定される。ロックが、ホストのロック・ホルダによって保持されている場合、命令インターセプトが認識され、その命令の実行は打ち切られる。ロックがホストのロック・ホルダによって保持されていない場合、ロックは更新され、それが共用ロック・ホルダであるゲスト・プロセッサによって保持されていることを表示し、上記命令は実行され、次いで、ロックが再度更新され、それがもはや共用ロック・ホルダとして該ゲスト・プロセッサに保持されていないことを表示する。

【先行技術文献】

【特許文献】

【０００７】

【特許文献１】米国特許第５，８０９，５５１号

【特許文献２】米国特許出願公開第２００４／００６４６１８号（Ａ１）

【発明の概要】

【発明が解決しようとする課題】

【０００８】

大型の単一イメージ・システムでは、比較／スワップを遂行するためのオーバーヘッドが相当なものになり得る。あるプロセッサがＩＰＴＥ、ＩＤＴＥ、またはＣＳＰ／Ｇを実行しているとき、該プロセッサは、一斉通信高速静止（ｆａｓｔ－ｑｕｉｅｓｃｅ）オペレーションを行って、システム中の全プロセッサに無効化を通知する。システム中のどの一つの区画に対しても、一つだけの高速静止化オペレーションが可能である。あるプロセッサが高速静止化オペレーションを発行し、静止化ハードウェアが既に別の要求を処理している場合、その要求は却下される。この大型の単一イメージでは、静止要求の多いワークロードで実行がなされる場合、却下の頻度は高い。適切な実行を確実にするために、高速静止要求を発行する前にＩＰＴＥインターロックが取得され、要求が却下された場合でも、オペレーションの後で解除される。各比較／スワップによって、関連するキャッシュ・ラインが排他的にフェッチされることになり、大きなマルチノードのシングル・イメージ・マシンでは、このためのシステム全体の負担は大きなものになり得る。

【課題を解決するための手段】

【０００９】

診断（ｄｉａｇｎｏｓｅ）命令を実行し、処理を逐次化するためのコンピュータ・プログラム製品の提供を介して、従来技術の欠点を克服し利点を提供する。本コンピュータ・プログラム製品は、方法を遂行するため処理回路が実行する命令を格納する、処理回路による読み取りが可能なコンピュータ可読のストレージ媒体を含む。上記方法は、例えば、実行のためマシン命令を取得するステップであって、マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、該マシン命令は、例えば、診断命令を識別するオペコード・フィールド、診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために診断命令が用いられることを識別するのに使われるオペレーション・コード拡張を得るための、第二位置を識別するベース・フィールドを含む、該取得するステップと、オペレーション・コード拡張によって示されたマシン命令を実行するステップであって、該実行するステップは、サブコードが所定の値であるのに応じて：コンピューティング環境のプロセッサの静止化を開始するステップ、それらのプロセッサが静止されていることを判定するステップ、およびプロセッサが静止されるのに応じて、診断命令の実行を完了するステップを含む、該実行するステップと、を含む。

【００１０】

本発明の一つ以上の態様に関する方法およびシステムも、本明細書に記載され、特許請求される。

【００１１】

本発明の技法を介して、さらなる特質および利点が実現される。本発明の上記以外の実施形態および態様も本明細書に詳細に記載され、これらは請求された発明の一部と見なされる。

【００１２】

本発明の一つ以上の態様が具体的に指摘され、例として、本明細書に添付の請求項において明確に請求される。本発明の前述および他の目的、特質、および利点は、以下の詳細な説明を添付の図面と併せ読めば明らかとなろう。

【図面の簡単な説明】

【００１３】

【図１】本発明の一つ以上の態様を組み込み用いるためのコンピューティング環境の一つの実施形態を示す。

【図２】本発明の一つ以上の態様を組み込み用いるためのコンピューティング環境の別の実施形態を示す。

【図３】本発明の一つ以上の態様を組み込み用いるための疑似的な（ｅｍｕｌａｔｅｄ）コンピューティング環境の一つの実施形態を示す。

【図４】本発明のある態様による、ゲスト変換テーブルの使用を逐次化し、ゲストとホストとの間で更新するために用いられるＩＰＴＥインターロックの一つの実施形態を示す。

【図５】ハイパーバイザの従来式のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。

【図６】ハイパーバイザの従来式のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。

【図７】ゲストの従来式のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。

【図８】ゲストの従来式のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。

【図９】本発明のある態様による、ハイパーバイザの別のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。

【図１０】本発明のある態様による、ハイパーバイザの別のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。

【図１１】本発明のある態様による、ゲストの別のＩＰＴＥインターロック処理に関する

ロジックの一つの実施形態を示す。
【図１２】本発明のある態様による、ゲストの別のＩＰＴＥインターロック処理に関するロジックの一つの実施形態を示す。
【図１３】本発明のある態様によって用いられる診断（ＤＩＡＧ）命令の一つの実施形態を示す。
【図１４】本発明のある態様による、別のＩＰＴＥインターロックの一部として使われるＤＩＡＧ命令に関するロジックの一つの実施形態を示す。
【図１５】本発明のある態様による、別のＩＰＴＥインターロックの一部として使われるＤＩＡＧ命令に関するロジックの一つの実施形態を示す。
【図１６】本発明のある態様による、ＤＩＡＧ命令によって発行された静止化の取扱いに関するロジックの一つの実施形態を示す。
【図１７】図１７Ａは、本発明のある態様によって用いられる高速静止コマンドの一例を示し、図１７Ｂは、本発明のある態様によって用いられる全定型（または全体システム）静止化コマンドの一例を示す。
【図１８】本発明のある態様による、静止要求処理を用いるコンピュータ・システムのプロセッサおよびシステム・コントローラのより詳細な実施形態を示す。
【図１９】本発明の一つ以上の態様を組み込んだコンピュータ・プログラム製品の一つの実施形態を示す。
【図２０】本発明の一つ以上の態様を組み込み使用するためのホスト・コンピュータ・システムの一つの実施形態を示す。
【図２１】本発明の一つ以上の態様を組み込み使用するためのコンピュータ・システムのさらなる例を示す。
【図２２】本発明の一つ以上の態様を組み込み使用するための、コンピュータ・ネットワークを含む、コンピュータ・システムの別の例を示す。
【図２３】本発明の一つ以上の態様を組み込み使用するためのコンピュータ・システムのさまざまなエレメントの一つの実施形態を示す。
【図２４】本発明の一つ以上の態様を組み込み使用するための、図２３のコンピュータ・システムの実行ユニットの一つの実施形態を示す。
【図２５】本発明の一つ以上の態様を組み込み使用するための、図２３のコンピュータ・システムの分岐ユニットの一つの実施形態を示す。
【図２６】本発明の一つ以上の態様を組み込み使用するための、図２３のコンピュータ・システムのロード／格納ユニットの一つの実施形態を示す。
【発明を実施するための形態】
【００１４】
本発明のある態様によって、複数のプロセッサに同じリソース（例えば、アドレス変換テーブル）を更新させる環境において処理を容易化するための、システム逐次化機能が提供される。具体的には、一つの事例において、システム逐次化機能が、ゲストおよびホストがロックを使って逐次化を行う、多重処理環境における処理を容易化するために用いられる。一例として、ホストがロック（例えば、ＩＰＴＥインターロック）を取得した後、システム逐次化機能が用いられ、ゲストがロックを取得する必要性は全く排除される。このことは、ロックの取得に関連してホストから見てオーバーヘッドを増やす可能性があるが、それは比較的まれで、ゲストがＩＰＴＥインターロックを取得するよくあるケースの該ロック取得を完全に排除する。これは、ＩＰＴＥインターロックに対する競合を、完全でないにせよほぼ全て排除する。
【００１５】
本発明の一つ以上の態様を組み込み使用するコンピューティング環境１００の一つの実施形態を、図１を参照しながら説明する。コンピューティング環境１００は、例えば、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）に基づいている。このｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）は、ＩＢＭ（ＩＢＭ社の

登録商標）発行のＩＢＭ発行番号第ＳＡ２２－７８３２－０７号、２００９年2月第8版の題名「ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）　Ｐｒｉｎｃｉｐｌｅｓ　ｏｆ　Ｏｐｅｒａｔｉｏｎ」に説明されている。一つの事例において、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）に基づくコンピューティング環境は、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているＳｙｓｔｅｍ　ｚ（ＩＢＭ社の登録商標）サーバを含む。ＩＢＭ（ＩＢＭ社の登録商標）、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）、Ｚ／ＶＭ（ＩＢＭ社の登録商標）、およびＳｙｓｔｅｍ　ｚ（ＩＢＭ社の登録商標）は、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。本明細書で使用する他の名称が、インターナショナル・ビジネス・マシーンズ・コーポレーションあるいは他の企業の登録商標または商標のことがある。

【００１６】

一つの事例として、コンピューティング環境１００は、システム・コントローラ１１２に連結された中央プロセッサ複合体（ＣＰＣ：ｃｅｎｔｒａｌ　ｐｒｏｃｅｓｓｏｒ　ｃｏｍｐｌｅｘ）１０２を含む。中央プロセッサ複合体１０２は、例えば、一つ以上の区画１０４（例、論理区画ＬＰ１～ＬＰｎ）、一つ以上の中央プロセッサ１０６（例、ＣＰ１～ＣＰｍ）、およびハイパーバイザ１０８（例、論理区画マネージャ）を含んでおり、これらの各々について以下に説明する。

【００１７】

各論理区画１０４は、別々のシステムとして機能する能力を有する。すなわち、各論理区画１０４は独立して、リセットされ、所望の場合最初にオペレーティング・システム１１０を搭載し、相異なるプログラムで作動することができる。ある論理区画１０４で実行されているオペレーティング・システム１１０またはアプリケーション・プログラムは、完全な全システムへのアクセスを有するように見えるが、実際は利用可能であるのは、その一部だけである。ハードウェアとライセンス内部コード（一般にマイクロコードまたはファームウェアといわれる）との組み合わせによって、プログラムを、異なる論理区画１０４のプログラムとの干渉が生じないように、一つの論理区画１０４中に保持する。これにより、いくつかの相異なる論理区画１０４が、単一のまたは複数の物理プロセッサ上で、タイム・スライス方式で動作することが可能になる。この特定の例において、各論理区画１０４は、常駐オペレーティング・システム（ＯＳ：ｏｐｅｒａｔｉｎｇ　ｓｙｓｔｅｍ）１１０を有し、これは一つ以上の論理区画１０４に対して相異なるものとすることができる。一つの実施形態において、オペレーティング・システム１１０は、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているｚ／ＯＳ（ＩＢＭ社の登録商標）オペレーティング・システムである。ｚ／ＯＳ（ＩＢＭ社の登録商標）は、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。

【００１８】

中央プロセッサ１０６は、論理区画１０４に割り当てられた物理プロセッサ・リソースである。例えば、論理区画１０４は、一つ以上の論理プロセッサを含み、該装置の各々は、該区画に割り当てられた物理プロセッサ・リソース１０６の全体または割り当て分を表す。ある特定の区画１０４の論理プロセッサは、ベースになるプロセッサ・リソースをその区画のために保留して該区画の専用とすることもでき、あるいは、別の区画と共用し、別の区画によるベースのプロセッサ・リソースの利用を見込んでおくこともできる。

【００１９】

論理区画１０４は、プロセッサ１０６上で実行されるファームウェアによって実装されたハイパーバイザ１０８によって管理される。論理区画１０４およびハイパーバイザ１０８は、各々、中央プロセッサ１０６に関連付けられた中央ストレージのそれぞれの部分に常駐する一つ以上のプログラムを含む。ハイパーバイザ１０８の一つの例に、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているＰｒｏｃｅｓｓｏｒ　Ｒｅｓｏｕｒｃｅ／Ｓｙｓｔｅｍｓ　Ｍａｎａｇｅｒ（

ＰＲ／ＳＭ）（ＩＢＭ社の商標）がある。

【００２０】

本明細書での使用において、ファームウェアは、例えば、プロセッサのマイクロコード、ファームウェア、もしくはマクロコードまたはこれらの組み合わせを含む。例えば、これは、より高いレベルのマシン・コードの実装に使われるハードウェア・レベルの命令もしくはデータ構造体またはその両方を含む。一つの実施形態において、これは、例えば、ベースのハードウェアに固有の信頼性あるソフトウェアまたはマイクロコードを包含するマイクロコードとして通常配布される、専用コードを含み、オペレーティング・システムのシステム・ハードウェアへのアクセスを制御する。

【００２１】

システム・コントローラ１１２は、中央プロセッサ複合体に連結されており、要求を発行する異なるプロセッサの間を調停する役割を持つ中央ロジックを含む。例えば、システム・コントローラ１１２が静止要求を受信したとき、該コントローラは、要求元が当該要求の開始プロセッサであること、および他のプロセッサが受信プロセッサであることを確定してメッセージを一斉通信し、別途にも要求を処理する。これは後記でさらに詳細に説明する。

【００２２】

本発明の一つ以上の態様を組み込み使用するためのコンピューティング環境の別の例を、図２を参照しながら説明する。この例において、コンピューティング環境２００は、システム・コントローラ２４０に連結された中央プロセッサ複合体（ＣＰＣ）２０２を含む。中央プロセッサ複合体２０２は、例えば、前述のように、一つ以上の区画２０４（例、論理区画ＬＰ１～ＬＰｎ）、一つ以上の中央プロセッサ２３０（例、ＣＰ１～ＣＰｍ）、および第一レベル・ハイパーバイザ１　２０８（例、論理区画マネージャ）を含む。

【００２３】

この特定の事例において、論理区画１（ＬＰ１）２２０は、常駐オペレーティング・システム２２２を有し、論理区画２（ＬＰ２）２２４は、第二レベル・ハイパーバイザ２　２１０を実行し、これが次に仮想マシン（例、ＶＭ１～ＶＭｘ）２１２を生成し、その各々はそれ自体の常駐オペレーティング・システム２１４を実行する。任意の数の論理区画が第二レベル・ハイパーバイザを実行することができる。一つの実施形態において、ハイパーバイザ２　２１０は、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供している、ｚ／ＶＭ（ＩＢＭ社の登録商標）のハイパーバイザである。ｚ／ＶＭはニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。

【００２４】

さらに、さまざまな論理区画で実行されている常駐オペレーティング・システムは相異なるものとすることができ、第二レベル・ハイパーバイザの下で実行される場合、単一の区画内の常駐オペレーティング・システム群も相異ならせることができる。一つの実施形態において、オペレーティング・システム２２２は、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているｚ／ＯＳ（ＩＢＭ社の登録商標）オペレーティング・システムであり、オペレーティング・システム２１４はＬｉｎｕｘ（Ｒ）である。

【００２５】

中央プロセッサ１０６と同様に、中央プロセッサ２３０は、論理区画に割り当てられた物理プロセッサ・リソースである。例えば、論理区画１（ＬＰ１）２２０は一つ以上の論理プロセッサを含み、その各々は、該区画に割り当てられた物理プロセッサ・リソース２３０の全体または割り当て分を表す。ある特定の区画の論理プロセッサは、ベースのプロセッサ・リソースをその区画のため保留して該区画の専用とすることもでき、あるいは、別の区画と共用し、別の区画によるベースのプロセッサ・リソースの利用を見込んでおくこともできる。第二レベル・ハイパーバイザ２　２１０が、ある論理区画（例、ＬＰ２　２２４）中で実行されている場合、これは、ハイパーバイザ１　２０８によって論理区画

２０４に提供されているのと同じリソースの仮想化を、当該区画内の仮想マシン中で実行されるオペレーティング・システム２１４に提供することができる。第一レベルにおいて、各仮想マシンには、複数の仮想プロセッサを含めることができる。本明細書で使われる用語「仮想ＣＰＵ」とは、論理プロセッサまたは仮想プロセッサをいう。

【００２６】

論理区画２０４は、ハイパーバイザ２０８と２１０とによって管理される。論理区画２０４およびハイパーバイザ２０８（ファームウェアによって実装される）は、各々、中央プロセッサに関連付けられた中央ストレージのそれぞれの部分に常駐する一つ以上のプログラムを含む。ハイパーバイザ２０８の一つの例に、ニューヨーク州、アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが提供しているＰｒｏｃｅｓｓｏｒ　Ｒｅｓｏｕｒｃｅ／Ｓｙｓｔｅｍｓ　Ｍａｎａｇｅｒ（ＩＢＭ社の商標）（ＰＲ／ＳＭ）がある。

【００２７】

システム・コントローラ２４０は、中央プロセッサ複合体に連結されており、要求を発行する異なるプロセッサの間を調停する役割を持つ中央ロジックを含む。例えば、システム・コントローラ２４０が静止要求を受信したとき、該コントローラは、要求元が当該要求の開始プロセッサであること、および他のプロセッサが受信プロセッサであることを確定してメッセージを一斉通信し、別途にも要求を処理する。

【００２８】

本発明の一つ以上の態様を組み込むためのコンピューティング環境の別の実施形態を、図３を参照しながら説明する。この例において、ホスト・アーキテクチャのホスト・コンピュータ・システム３０２をエミュレートする、疑似ホスト・コンピュータ・システム３００が提供される。疑似ホスト・コンピュータ・システム３００において、ホスト・プロセッサ（ＣＰＵ）３０４は疑似ホスト・プロセッサ（または仮想ホスト・プロセッサ）であり、ホスト・コンピュータ・システム３０２のプロセッサによって使用されるものとは異なるネイティブ命令セットのアーキテクチャを有するエミュレーションプロセッサ３０６を含む。疑似ホスト・コンピュータ・システム３００は、エミュレーションプロセッサ３０６がアクセスできるメモリ３０８を有する。この例示的な実施形態において、メモリ３０８は、ホスト・コンピュータ・メモリ３１０部分と、エミュレーション・ルーチン・メモリ３１２部分とに区画されている。疑似ホスト・コンピュータ・システム３００のプログラムは、ホスト・コンピュータ・アーキテクチャに従って、ホスト・コンピュータのメモリ３１０を利用することができ、該メモリには、ホストまたはハイパーバイザ３１４と、論理区画群（ＬＰ群）３１６で実行される一つ以上のハイパーバイザとの双方を含めることができる。各論理区画は、オペレーティング・システム３１８を実行することができる。

【００２９】

エミュレーションプロセッサ３０６は、疑似プロセッサ３０４のものとは異なるアーキテクチャで設計された命令セットのネイティブ命令を実行する。ネイティブ命令は、エミュレーション・ルーチン・メモリ３１２から得ることができ、これは、順次アクセス／復号ルーチンで取得した一つ以上の命令（群）を用いて、ホスト・コンピュータ・メモリ３１０中のプログラムの実行のためのホスト命令にアクセスすることができ、該ルーチンは、アクセスされたホスト命令（群）を復号し、アクセスされたホスト命令の機能をエミュレートするためのネイティブ命令の実行ルーチンを決定する。かかるホスト命令の一つを、例えば、解釈実行開始（ＳＩＥ：Ｓｔａｒｔ　Ｉｎｔｅｒｐｒｅｔｉｖｅ　Ｅｘｅｃｕｔｉｏｎ）命令とすることができ、これにより、ホストは、仮想マシンでプログラムを実行しようとする。エミュレーション・ルーチン・メモリ３１２には、この命令に対するサポート、およびこのＳＩＥ命令の定義に従ってゲスト命令のシーケンスを実行するためのサポートを含めることができる。

【００３０】

ホスト・コンピュータ・システム３０２のアーキテクチャのために定義された他の機能

は、例えば、汎用レジスタ、制御レジスタ、動的アドレス変換、およびＩ／Ｏサブシステム・サポートおよびプロセッサ・キャッシュなどの機能を含めて、設計された機能ルーチンによってエミュレートすることができる。また、エミュレーション・ルーチンは、エミュレーションプロセッサ３０４で利用可能な（汎用レジスタおよび仮想アドレスの動的変換などの）機能をうまく活用して、エミュレーション・ルーチンのパフォーマンスを向上することができる。さらに、ホスト・コンピュータ・システム３０２の機能のエミュレートにおいてプロセッサ３０６を助力するために、特別なハードウェアおよびオフロード・エンジンを設けることもできる。

【００３１】

コンピューティング環境において処理を逐次化するため、一つの実施形態では、共用ＩＰＴＥインターロック（本明細書では、共用ロック、共用ＩＰＴＥロック、またはロックともいう）が使われる。このロックは、例えば、ハイパーバイザが、ゲストの動的アドレス変換（ＤＡＴ：Ｄｙｎａｍｉｃ　Ａｄｄｒｅｓｓ　Ｔｒａｎｓｌａｔｉｏｎ）およびアクセス・レジスタ変換（ＡＲＴ：Ａｃｃｅｓｓ　Ｒｅｇｉｓｔｅｒ　Ｔｒａｎｓｌａｔｉｏｎ）テーブルの使用を逐次化するために、さらにファームウェアがゲストのためこれらのテーブルを更新するために用いられる。共用ＩＰＴＥインターロックの一例が図４に示されている。共用ＩＰＴＥインターロック４１０は、ホスト・ハイパーバイザが、ゲストＤＡＴまたはＡＲＴテーブルを使って一つ以上のオペランドを変換している間は、該ハイパーバイザによって保持され、または、ファームウェアがゲストのＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇ要求のために変換テーブルを更新しているときは、該ファームウェアによって、保持される。ハイパーバイザは、それがロックを取得するのに応じ、ホストＩＰＴＥロック・カウント４１４をインクリメントする。本発明の一態様の以前は、同様に、ファームウェアがロックを取得するのに応じ、ゲストＩＰＴＥロック・カウント４１２もインクリメントする。しかしながら、本発明のある態様によれば、ゲストＩＰＴＥロックはもはや使われない。

【００３２】

マニュアルのゲストＤＡＴまたはＡＲＴ変換の間のハイパーバイザによる共用ＩＰＴＥロックの維持のための従来技法の一つの実施形態を、図５～図６を参照しながら説明する。ハイパーバイザは、通常、一つ以上のストレージ・オペランドにアクセスするべくゲストのために命令をエミュレートしているときに、この変換を遂行する。この変換を本明細書ではマニュアルという。というのは、これはハードウェアが不知の多段階の処理だからである。従って、ハードウェアは、ハイパーバイザのために処理を逐次化することはない。

【００３３】

図５に示されるように、実際の変換を始める前に、ハイパーバイザは、アドレス変換が適用されるゲストが多重処理（ＭＰ）ゲスト、すなわち、構成の中の複数の仮想プロセッサであるかどうかを確認する（インクワイアリ５１０）。ゲストが単一プロセッサ・ゲスト、すなわち、構成中の一つだけの仮想プロセッサである場合、ハイパーバイザは、ＩＰＴＥインターロックをセットせずに、マニュアル変換を遂行することができる（図６のステップ５２２）。これは、一つのゲスト・プロセッサが、ハイパーバイザにエミュレートされた命令を実行しており、他のプロセッサが変換テーブルを更新するためのＩＰＴＥ、ＩＤＴＥ、またはＣＳＰ／Ｇを発行した場合を管理するためにＩＰＴＥインターロックが使われることによる。単一プロセッサ・ゲストにおいては、ある命令がエミュレートされている場合に、変換テーブルを更新する命令を発行する他のプロセッサは存在しない。

【００３４】

その後、これが多重処理ゲストであるかどうか、再度判定が行われる（インクワイアリ５２４）。この場合は、多重でないので処理は完了する。

【００３５】

図５に戻ると、適用対象ゲストが、多重処理環境で実行されている場合（インクワイアリ５１０）、ハイパーバイザは、まず、ＩＰＴＥインターロックが既に保持されているか

どうかを判定する（インクワイアリ５１２）。保持されていなければ、ハイパーバイザは該インターロックを取得することができる（ステップ５１４）。これは、例えば、比較／スワップ・オペレーション（例、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）におけるＣｏｍｐａｒｅ ａｎｄ Ｓｗａｐ （ＣＳＧ））などのインターロック更新を使って遂行することができる。この比較／スワップは、当初フェッチされたものからのデータを現在のデータと比較して、データが変化していない場合にだけ更新を行う。ハードウェアは、このチェックおよび更新がアトミックであること、すなわち、この間、他のどのプロセッサもロックを変更していないことを保証する。ハイパーバイザは、ＩＰＴＥインターロックをセットし、ホストＩＰＴＥロック・カウントを、例えば、ゼロから１にインクリメントする。このインターロック更新に失敗した場合（インクワイアリ５２０）、ハイパーバイザはインクワイアリ５１２に戻り、再度インターロックをチェックする。

【００３６】

ＩＰＴＥインターロックが既に保持されている場合（インクワイアリ５１２）、このときハイパーバイザは、それがゲストによって保持されているかどうか、すなわち、ゲスト・カウントが非ゼロかどうかをチェックする（インクワイアリ５１６）。それがゲストによって保持されている場合、ハイパーバイザは戻ってインターロックを再フェッチし、それを再度チェックする（インクワイアリ５１２）。ゲストがインターロックを保持していない場合（インクワイアリ５１６）、これは、ホスト・ハイパーバイザが既にロックを保有していることを意味する。ハイパーバイザは、ホスト・ロック・カウントをインクリメントしようと試み（ステップ５１８）、この試みが失敗した場合（インクワイアリ５２０）、ハイパーバイザは、インクワイアリ５１２に戻って再度インターロックをチェックする。ホストＩＰＴＥインターロックの更新が成功したのに応じ（インクワイアリ５２０）、ハイパーバイザは、このオペレーションの期間どの変換テーブルも修正されることがないのを知った上で、マニュアル変換を進めることができる（図６のステップ５２２）。

【００３７】

ハイパーバイザが変換および関連するストレージの更新を完了する（ステップ５２２）のに引き続いて、ゲストがＭＰゲストの場合（インクワイアリ５２４）、ハイパーバイザは、インターロック更新によってそのＩＰＴＥインターロックの割り当てを解除する（ステップ５２６）。一つの事例において、これは、ホストＩＰＴＥカウントをデクリメントし、新規のカウントがゼロの場合にＩＰＴＥインターロックをリセットすることによって達成される（ステップ５２６）。この更新は成功するまで繰り返される（インクワイアリ５２８）。更新が成功すれば、ハイパーバイザは終了する。

【００３８】

ホストがＩＰＴＥインターロックを使用するのに加えて、ゲストもこれを使用する。ゲストがこれを使用する一つのシナリオは、静止化オペレーションを遂行する命令の実行におけるものである。用語「静止化（“ｑｕｉｅｓｃｅ”）」とは、多重プロセッサ・システム（すなわち、更新されるデータ構造体（例、テーブル）へのアクセスを有するプロセッサ群）中の全てのプロセッサにその動作を、一つのプロセッサがシステム状態に或る変更を加えている間、停止することを強制するために使われるメカニズムをいう。静止化オペレーションに対する一つの一般的遂行には、以下のステップが含まれる：１）全プロセッサが静止状態にされる（すなわち、最も通常的な処理オペレーションが一時停止される）；２）それらプロセッサ上の、更新されているリソースに依存するバッファリングされた一切のエントリが無効にされる；３）静止化イニシエータによって共通リソースが更新される；および４）最後に、静止化が解除され、これらプロセッサはその通常の動作を継続する。この静止化オペレーションは、とりわけて、ゲストが多重処理環境で実行されているときに、ＩＰＴＥ、ＩＤＴＥ、およびほとんどのＣＳＰ命令を実行するために使われる。ＩＰＴＥインターロックがセットされるのは、こういった場合においてである。

【００３９】

図７～図８は、ゲスト多重処理環境における、静止化ＩＰＴＥ、ＩＤＴＥ、およびＣＳＰ／Ｇ命令に関連するＩＰＴＥインターロック更新の従来式ゲスト実行の一つの実施形態

を示す。図７を参照すると、ファームウェアが、まず、ＩＰＴＥインターロックが既に保持されているかどうかを判定する（インクワイアリ６１０）。保持されていなければ、ファームウェアはそのインターロックを取得する（ステップ６１２）。これは、例えば、比較／スワップ・オペレーション（例、Ｃｏｍｐａｒｅ　ａｎｄ　Ｓｗａｐ（ＣＳＧ））を使用するインターロック更新を使って遂行される。この比較／スワップは、当初フェッチされたものからのデータを現在のデータと比較して、データが変化していない場合にだけ更新を行う。ハードウェアは、このチェックおよび更新がアトミックであること、すなわち、この間、他のどのプロセッサもロックを変更していないことを保証する。ファームウェアは、ＩＰＴＥインターロックをセットし、ゲストＩＰＴＥロック・カウントを、例えば、ゼロから１にインクリメントする（ステップ６１８）。このインターロック更新に失敗した場合（インクワイアリ６２０）、ハイパーバイザは、インクワイアリ６１０に戻って再度インターロックをチェックする。

【００４０】

ＩＰＴＥインターロックが既に保持されている場合（インクワイアリ６１０）、このときファームウェアは、それがホストによって保持されているかどうか、すなわち、ホスト・カウントが非ゼロかどうかをチェックする（インクワイアリ６１４）。該インターロックがホストによって保持されている場合、ファームウェアは、インターセプトして、インターセプトの種類および必要な命令の何らかの表示と共に、ハイパーバイザに戻す（ステップ６１６）。通常、この場合、ハイパーバイザは、ゲストを再指名して、ゲスト・コードに、ＩＰＴＥ型（または静止型）命令における実行を再度開始させる。

【００４１】

ホストがインターロックを保持していない場合（インクワイアリ６１４）、これは、ゲスト（ファームウェア）が既にロックを保有していることを意味する。このとき、ファームウェアはゲスト・ロック・カウントをインクリメントしようと試み（ステップ６１８）、この試みが失敗した場合（インクワイアリ６２０）、ファームウェアは、インクワイアリ６１０に戻って再度インターロックをチェックする。ゲストＩＰＴＥインターロックの更新が成功したのに応じ（インクワイアリ６２０）、ファームウェアは、テーブルに依存する変換が、ハイパーバイザによって、関与する命令により更新されていないのを知った上で、高速静止化オペレーションを進める（図８のステップ６２２）。

【００４２】

中央プロセッサ（例えば、図１および図２のそれぞれ１０６または２３０）によって発行される一切の高速静止化オペレーションは、システム・コントローラ（図１および図２のそれぞれ１１２または２４０）によって逐次化されることになる。この逐次化は、全ての異なるプロセッサ上の静止化オペレーションの追跡を含み、そのため、システム中での適切な逐次化を保証するためには、どの一時点でも、区画またはゲスト構成あたり一つの高速静止が許される。高速静止要求が出され、システム・コントローラ中の関連ハードウェアが既に別の要求を処理している場合、その高速静止要求は拒否される。

【００４３】

高速静止要求が拒否された場合（インクワイアリ６２４）、ファームウェアは、そのゲストＩＰＴＥインターロックの割り当てをリセットする（ステップ６２６）。これは、インターロック更新を使い、ゲストＩＰＴＥカウントをデクリメントし、その新しいカウントがゼロの場合に、ＩＰＴＥインターロックをリセットすることによって遂行される（ステップ６２６）。この更新は、成功するまで繰り返される（インクワイアリ６２８）。

【００４４】

このインターロック更新が成功裏に完了したのに応じ、ファームウェアは、ＩＰＴＥ型の命令を取消し、該命令が再度実行され得るようにする（ステップ６３０）。例えば、ファームウェアは、命令ポインタをバック・アップして、ＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇなどにポイントを戻し、それが再度実行されるようにする。（注記：一つの事例において、特定回数の拒否の後、ファームウェアは異なった（従来式）静止化メカニズムを用い、これは高速静止要求が完了されることを保証する。）

【００４５】
インクワイアリ６２４に戻る。高速静止要求が拒否されなかった場合、このとき、ファームウェアは高速静止化オペレーションを完了し（ステップ６３２）、これには、プロセッサ中の関連するバッファ値の妥当性確認、およびストレージ中の適切な変換テーブルのエントリの更新が含まれる。また、ファームウェアは次いでそのＩＰＴＥインターロックの割り当てをリセットして（ステップ６３４）、必要に応じ再試行し（インクワイアリ６３６）、成功した場合、命令（例、ＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇ）を完了する（ステップ６３８）。

【００４６】
大型システムでは、多くの静止要求が発行され、多数の高速静止拒否が生じ得る。当然のことながら、どの所与の高速静止化オペレーションも、それを完了するのに要する全体的時間はそれが受けた拒否の数が増えるほど増加する。ハイパーバイザとの適切な調整を確実にするために、高速静止要求を発行する前にＩＰＴＥインターロックがセットされることになる。各々の拒否に対して、ＩＰＴＥインターロックのセットおよびリセットが必要となり、しかして、ＩＰＴＥインターロックに対する競合が増大する。大規模な単一イメージ・システムでは、システム中の全てのプロセッサが同じインターロックを共用する。システム中のプロセッサの数が増加するにつれ、静止化オペレーションを処理するための相対的時間は、ＣＰＵ時間との対比で増加し、ＩＰＴＥインターロックへのインターロック・アクセスの数が増えるのに応じ拒否の数も増加する。結果として、インターロックを取得しようとして消費される時間量が過度になる。

【００４７】
本発明のある態様によれば、ホストの共用ロック維持の実装において、ゲストＩＰＴＥロックのセッティングが排除される。これを、図９～図１０を参照して説明する。本実装は、特にリカバリに関し、ハイパーバイザの変更を最小化する。

【００４８】
最初に図９を参照すると、ゲスト変換が単一プロセッサ（すなわち、多重プロセッサでない）環境に適用された場合（インクワイアリ７１０）、ＩＰＴＥインターロックと同様に、マニュアルのゲストＤＡＴまたはＡＲＴ変換（群）および関連するストレージの更新が実施され（図１０のステップ７２２）、続いて終了する（ステップ７３０）。

【００４９】
図９のインクワイアリ７１０に戻る。一方、ゲストが多重処理環境で実行されている場合、もはやゲストＩＰＴＥロックを検査する必要はない。ただし、それでもハイパーバイザは、ホスト・ロックをインクリメントする。例えば、ＩＰＴＥインターロックが既にセットされているかどうかの判定が行われる（インクワイアリ７１２）。されていない場合、インターロック更新を使って、該インターロックおよびホストＩＰＴＥロックがセットされる（ステップ７１４）。ＩＰＴＥインターロックがセットされている場合（インクワイアリ７１２）、インターロック更新を使って、ホストＩＰＴＥロックがインクリメントされる。いずれの場合においても、更新に失敗した場合は（インクワイアリ７２０）再試行される。

【００５０】
さらなる実施形態において、別型ＩＰＴＥインターロックの解除が、ハイパーバイザに分かるようにするために、ハイパーバイザは、図５を参照して説明したように、ゲストＩＰＴＥロックの検査を続けることができる。ＩＰＴＥロックのホスト・セッティングは比較的に稀なので、これのパフォーマンスへの影響は最少である。

【００５１】
ホストＩＰＴＥロックのセッティングに応じて、ハイパーバイザは診断（ＤＩＡＧ）命令を発行し、これは、ハードウェア中の静止逐次化をもたらす（図１０のステップ７２１）。この静止逐次化は２つのことを行う。第一に、これは、ホストＩＰＴＥロック更新がシステムの全域に見えることを保証し、第二に、一切の未処理のゲスト静止化オペレーションが完了されることを確実にする。ホストＩＰＴＥロックがセットされたので、新規の

ゲスト静止化オペレーションは開始することができない。この命令については後記でさらに詳しく説明する。

【００５２】

前述したように、ＤＩＡＧの完了の後、ハイパーバイザは、続いて、マニュアルのゲストＡＲＴまたはＤＡＴ変換（群）および関連するストレージの更新を遂行する（ステップ７２２）。次いで、これが多重処理ゲスト環境かどうかの判定が行われる（インクワイアリ７２４）。そうである場合、ホストＩＰＴＥロックはデクリメントされ、ホスト・ロックがゼロである場合、インターロックはリセットされる（ステップ７２６）。これはインターロック更新によって遂行される。更新に失敗した場合（インクワイアリ７２８）、処理はステップ７２６を続ける。成功した場合、またはこれが多重処理環境でない場合、処理は完了する（ステップ７３０）。

【００５３】

静止ＩＰＴＥ、ＩＤＴＥ、およびＣＳＰ／Ｇ命令の別型ＩＰＴＥインターロックのゲスト実装に関するロジックの一つの実施形態を、図１１～図１２を参照しながら説明する。この実施形態において、ファームウェアは、ここでもＩＰＴＥロックのホスト部分を検査するが、もはやゲストＩＰＴＥロックを更新することはない。

【００５４】

図１１を参照すると、ＩＰＴＥインターロックがセットされて（インクワイアリ８１０）ホストＩＰＴＥロックがセットされていることが暗示されている場合（インクワイアリ８１４）、ファームウェアは、静止化命令の命令インターセプトをハイパーバイザに提示する（ステップ８１６）。これは、ハイパーバイザがマニュアル変換を行っている期間、静止化オペレーションが行われるのを阻止する。ハイパーバイザ（ホスト）は、ＤＩＡＧ命令を発行することによって、マニュアル変換の間、静止化が行われないことを既に保証している。これにより、ゲスト・ロックのセットは必要がない。

【００５５】

ホストＩＰＴＥロックがセットされていない場合（インクワイアリ８１４）、ファームウェアは、高速静止一斉通信要求を発行し（図１２のステップ８２２）、高速静止要求が拒否された場合は（インクワイアリ８２４）その命令を取り消す（ステップ８３０）。要求が受け入れられた場合（インクワイアリ８２４）、ファームウェアは高速静止化オペレーションを完了し（ステップ８３２）、命令を完了する（ステップ８３８）。これら２つの場合のいずれにおいても、ロックがセットされないので、ゲストＩＰＴＥロックの更新は不必要である。

【００５６】

図１３～図１５を参照しながら、ＤＩＡＧ命令についてのさらなる詳細を説明する。ＤＩＡＧ命令は、新しく定義されたシステム逐次化を提供する。ハイパーバイザがこの命令を発行し、システム制御領域中でのＩＰＴＥインターロックのセット後、だが、命令インターセプトに関連する一切の変換が開始される前に、ファームウェアによって実行される。

【００５７】

ＤＩＡＧ（診断）命令の一つの実施形態が図１３に示されている。図示のように、診断（本明細書ではＤＩＡＧともいう）命令９００は、例えば、ＤＩＡＧ命令を表すオペコード９０２と、システム制御領域アドレスが所在する場所を指定する第一フィールド９０４（別称、制御領域フィールド）と、ＤＩＡＧの実行に使われるサブコードが備えられている場所を指定する第二フィールド９０６（別称、サブコード・フィールド）と、例えば別型ＩＰＴＥインターロックに対するＤＩＡＧを識別するオペレーション・コード拡張として使われる第四フィールド９１０（別称、変位フィールド）の内容に加算されることになるデータを含む場所を指定する第三フィールド９０８（別称、ベース・フィールド）と、を含む。

【００５８】

サブコードの例には、以下が含まれる。

【００５９】
例えば、サブコード・ゼロが規定されている場合、システム制御領域アドレスは無視され、オペレーションは遂行されない。
【００６０】
例えば、サブコード２が規定されている場合、システム制御領域アドレスは、変換が行われている対象のゲストを識別し、診断命令の実行は、構成の中のＣＰＵの全てが、指定されたゲスト群のインプロセスのＤＡＴ同期（ＤＳ：ＤＡＴ　Ｓｙｎｃｈｒｏｎｉｚｉｎｇ）命令（例、ＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇ）の解釈を完了するまでは、該命令を実行しているＣＰＵ上で完了しない。また、この診断の完了により、ホストがＩＰＴＥインターロックを保持している間は、一切の後続のゲストＤＳ命令も、命令インターセプトを認識することが確実になる。
【００６１】
ＤＩＡＧ命令に関するロジックの一つの実施形態を、図１４～図１５を参照しながら説明する。
【００６２】
図１４を参照すると、ファームウェアに実装されているＤＩＡＧ命令が、ハイパーバイザによって発行される（ステップ９２０）。この命令は、システム中の全プロセッサ（すなわち、更新されるリソースにアクセスを有するプロセッサ群）が、一切のローカルな未処理ストアがシステムに可視となるまで待機することによって、ハイパーバイザによって前にセットされたＩＰＴＥインターロックを順守することを保証する（ステップ９２２）。すなわち、プロセッサは、ローカルな未処理ストア（ＩＰＴＥインターロックのホスト部分を含む）が、全プロセッサに可視となるストレージのレベルに書き込まれるまで、待機する。また、プロセッサは、全定型（ｆｕｌｌ　ｃｏｎｖｅｎｔｉｏｎａｌ）静止メカニズムを発行することによって、システム中で未処理の一切のＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇ命令の完了を待つ（ステップ９２４）。ＤＩＡＧを発行した中央プロセッサ（例えば、図１の１０６）で実行されるファームウェアは、一斉通信Ｓｅｔ　Ｆｕｌｌ　Ｃｏｎｖｅｎｔｉｏｎａｌ　Ｑｕｉｅｓｃｅ（全定型静止を設定する）コマンドをシステム・コントローラ（例えば、図１の１１２）に発行することによって、これを達成する。次いで、システム・コントローラは、このコマンドをシステム中の全プロセッサに一斉通信する。この処理については、図１６を参照しながら後記でさらに説明する。
【００６３】
また、静止させられたのに応じ、ＤＩＡＧのイニシエータは、自分も静止状態に入ったことを表す自己静止宣言（Ｉ－ａｍ－ｑｕｉｅｓｃｅｄ）インジケータの設定も行い（ステップ９２６）、全プロセッサが静止するのを待つ（ステップ９２８）。システム・コントローラは、システム中の全プロセッサが自己静止宣言インジケータを設定したならば、システムが静止していることをあらゆるプロセッサに対し表示する。これは、システム中で未処理になっていた可能性のある全てのＩＰＴＥ、ＩＤＴＥ、およびＣＳＰ／Ｇ命令が完了されたことを保証する。あらゆるプロセッサが自分のローカル・ストアをシステム全体に可視にしているので、これにより、ゲストによって今後実行される一切のＩＰＴＥ、ＩＤＴＥ、およびＣＳＰ／Ｇが、ホストにインターセプトされることも保証する。なぜなら、ＩＰＴＥインターロック・ビットがセットされているからである。これは、ゲストの比較／スワップのオーバーヘッドなく、ＩＰＴＥインターロックによって従来保証されていたのと同じである。
【００６４】
システムが静止されたのに応じ（ステップ９２８）、ＤＩＡＧのイニシエータは定型の静止優先順位を待つ（ステップ９３０）。一時点において、システム中に未処理の複数の定型静止要求がある場合、ハードウェアおよびファームウェアは、各プロセッサが定型の静止優先順位を受けて交代することを確実にする。この一つのプロセッサは、定型静止化（ｃｏｎｖｅｎｔｉｏｎａｌ　ｑｕｉｅｓｅ）を介し、既にそのオペレーションを完了しているので、優先度を有し、Ｒｅｓｅｔ　Ｃｏｎｖｅｎｔｉｏｎａｌ　Ｑｕｉｅｓｃｅ（

定型静止リセット）コマンドを発行する（図１５のステップ９４０）。このリセット・コマンドは、上記と同様にシステム中の全プロセッサに一斉通信され、全ての定型静止イニシエータがそれらの要求をリセットしたならば、未処理の静止要求はなくなることになる。

【００６５】

定型静止要求がリセットされるのに応じ（ステップ９４２）、ＤＩＡＧイニシエータおよびレシーバの両方は、静止状態を出ること、すなわち、自己静止宣言をリセットすることができる（それぞれのステップ９４４）。これで処理は終了する。

【００６６】

静止要求に関するさらなる詳細を、図１６を参照しながら説明する。Ｓｅｔ　Ｆｕｌｌ　Ｃｏｎｖｅｎｔｉｏｎａｌ　Ｑｕｉｅｓｃｅ（全定型静止化の設定）要求の発行に応じ、システム中の各プロセッサは、それが割り込み可能ポイントに達すると、静止割り込みファームウェア・ハンドラに入ることによって、通常の処理を一時停止する（ステップ１０００）。次いで、プロセッサは全てのローカルな未処理ストアがシステムに可視になるのを待つ（ステップ１００２）。これが行われるのに応じ、プロセッサは自己静止宣言インジケータをセットし、これはシステム・コントローラに対し、該プロセッサが静止状態に入ったことを表示し（ステップ１００４）、プロセッサは、システム中の全プロセッサが静止状態に達するのを待つ（ステップ１００６）。ファームウェアは、未処理の定型静止要求がなくなるのを待ち（ステップ１００８）、静止状態を出る（例えば、自己静止宣言をリセットする）（ステップ１０１０）。

【００６７】

以下に、高速静止および全定型（全体システム）静止要求に関するさらなる詳細を説明する。

【００６８】

図１７Ａを参照すると、高速静止（ＦＱｕｉ）要求の一つの実施形態１１００が示されている。一つの事例において、高速静止要求は、

高速静止コマンド（ＦＱｕｉ－高速静止）とプロセッサによって要求される無効化を規定する高速静止要求の種類（例、ＩＰＴＥ、ＩＤＴＥなど）とを規定するコマンド・フィールド１１０２と、

イニシエータの区画ゾーン番号１１０４を示すゾーン・フィールド１１０４と、

要求される無効化（例えば、ＩＰＴＥに対するページ・インデックスなど）をさらに条件付けするアドレス・フィールド１１０６と、

を含む。

【００６９】

図１７Ｂは、全定型静止要求１１５０の一つの実施形態を示す。一つの事例において、全定型静止要求は、

全静止化コマンドを示すコマンド・フィールド１１５２と、

イニシエータのプロセッサＩＤ（ＰＵＩＤ）の表示１１５４と、

コマンドが、全定型要求のＳＥＴあるいはＲＥＳＥＴのいずれであるかの表示１１５６と、

を含む。

【００７０】

図１８は、複数の中央プロセッサ（ＣＰＵ）１２１０に連結されたシステム・コントローラ１２００の例を示し、中央プロセッサは一つだけを図示している。当業者は、複数のプロセッサ１２１０をシステム・コントローラ１２００に連結できることを理解していよう。

【００７１】

図１８を参照すると、システム・コントローラ１２００は、例えば、システム逐次化コントローラ１２０２などを含め、さまざまなコントローラを含む。とりわけ、システム逐次化コントローラ１２０２は、ページ・テーブル・エントリ無効化（ＩＰＴＥ）、ＤＡＴ

テーブル・エントリ無効化（ＩＤＴＥ）、または比較／スワップ／パージ（ＣＳＰ／ＣＳＰＧ）命令によって使用されるものなど逐次化対象のオペレーションが、コンピューティング環境中で、どの一時点においても、どの一つの区画においても、ただ一つのかかる命令が進行するようにして、逐次化されることを確実にするために用いられる。また、該コントローラは、それらオペレーションに対するイベントのシーケンスもモニタする。

【００７２】

システム・コントローラ１２００は、さまざまなインタフェースを介して各中央プロセッサ１２１０に連結される。例えば、コントローラ１２００へのインタフェース１２１４は、システム・オペレーション・コントローラ１２１２からの「制御」コマンドを送信するために、中央プロセッサ中のファームウェアによって使用され、該コマンドは、システム・コントローラ１２００が取るべき処置を、場合によってはシステム逐次化コントローラ１２０２が取るべき処置を、規定する。別のインタフェースとして応答バス１２１６があり、これは、これらのコマンドに関してコントローラ１２００からのステータス情報を返信するために使われる。このステータス情報は、中央プロセッサ１２１０内のシステム・オペレーション・コントローラ１２１２によって、システム・オペレーション要求のステータスを表示するのに使われる状態コード１２１８を設定するために使用される。応答情報は、システム逐次化コントローラ１２０２を含め、コントローラ１２００内の複数のソースから設定することができる。また、中央プロセッサ１２１０は、システム・コントローラ１２００内のシステム逐次化コントローラ１２０２の状態を検知するためにも、このインタフェースを使用することができる。

【００７３】

さらなるインタフェースには、このローカル中央プロセッサ１２１０の定型静止状態（自己静止宣言）１２２０をシステム・コントローラ１２００に提示するインタフェース１２２２が含まれる。システム・コントローラ１２００は、システム中のすべての対象プロセッサからの自己静止宣言状態１２２０のＡＮＤ１２０４を取り、システム静止宣言状態（ｓｙｓｔｅｍ－ｉｓ－ｑｕｉｅｓｃｅｄ　ｓｔａｔｅ）１２０６を表示する。このシステム静止宣言状態１２０６は、インタフェース１２３４を介して各中央プロセッサ１２１０に提示され、各プロセッサではローカル・コピー１２２４が維持され、ファームウェアの問い合わせに対するＱＵＳＹＳ１２２６を設定するのに使われる。

【００７４】

ＩＰＴＥ、ＩＤＴＥ、およびＣＳＰ／Ｇなど静止化オペレーションに対し、要求または所望がある場合に、システム・オペレーション・インタフェース１２１４を介して高速静止化オペレーションが送信される。システム逐次化コントローラ１２０２が、別の高速静止要求を処理作業中の場合、該コントローラは、状態コード１２１８を使い、開始中央プロセッサ１２１０中のシステム・オペレーション・コントローラ１２１２に、これの「拒否」を表示することになる。開始中央プロセッサ中のファームウェアは、所与の一切の高速静止要求１２３０に対する拒否の数のカウントを維持する。このカウントが高速静止拒否限度１２３２に達したとき、当該高速静止が最終的に完了することを保証するために、定型静止シーケンスが使われる。

【００７５】

システム逐次化コントローラ１２０２が別の高速静止化オペレーションの処理作業中でない場合、すなわち、コマンドが拒否されなかった場合、該コントローラは、システムの各中央プロセッサ１２１０内の静止化コントローラ１２３６に対し、インタフェース１２３８を介して高速静止「制御」コマンドを一斉通信することになる。静止化コントローラ１２３６が高速静止（ＦＱｕｉ）要求を受信したとき、該コントローラは、その静止要求を処理するために、当該中央プロセッサ１２１０に割り込みを行うかどうかを判定し、行う場合、この１２４３を静止割り込みコントローラ１２４０に示し、高速静止要求１２４２および割り込み１２４４を待機状態となるようにする。加えて、適切な場合、静止化コントローラ１２３６は、高速静止コマンドを、中央プロセッサ１２１０中の、高速静止コマンドを変換ルックアサイド・バッファ（またはＴＬＢ：Ｔｒａｎｓｌａｔｉｏｎ　Ｌｏ

ｏｋ－ａｓｉｄｅ Ｂｕｆｆｅｒ）１２４６に転送し、該バッファが該要求により要求された任意のＴＬＢエントリを無効化できるようにする。

【００７６】

高速静止拒否が何度もあったために、あるいは高速静止メカニズムが静止を要求する機能に対応していないために、定型静止シーケンスが使われる場合、ファームウェアは、システム・オペレーション・コントローラ１２１２を使い、インタフェース１２１４を介して、全定型静止「制御」コマンドを送信する。いずれの要求に対しても、システム逐次化コントローラ１２０２は、各プロセッサ内の静止化コントローラ１２３６に、定型静止が要求されていることを示すＳＹＳＯＰ（静止化）コマンドを送る。静止化コントローラは、ＳＹＳＯＰコマンド中のイニシエータのプロセッサＩＤ１１５４（図１７）を使い、ＰＵ定型静止優先ベクトル１２５０（図１８）中に対応するビットをセットする。定型静止優先ベクトル１２５０は、とりわけ、当該ローカル・プロセッサが、ローカル・プロセッサのプロセッサＩＤ１２５２に基づいた定型静止優先性を有しているかどうかを判定するのに使われる。ローカルＰＵＩＤ１２５２に対応するビットは優先ベクトル１２５０中の最も左のビットでこれがオンであれば、そのプロセッサは定型静止優先性１２５４を与えられる。

【００７７】

静止優先ベクトル１２５０中のビットの論理ＯＲ１２５６を使って、何らかの未処理の定型静止（ＣＱｕｉ）要求１２５８があるかどうかが表示される。これは、分岐条件としてファームウェアに提示される。さらに、未処理定型静止要求インジケータ１２５８が、静止割り込みコントローラ１２４０に送信される。未処理定型静止要求インジケータ１２５８は、次いで、静止割り込みコントローラに送信され、定型静止割り込みが未処理となっていることを表示する１２６２ために使われる。

【００７８】

静止化についてのさらなる詳細は、以下の特許／特許出願に記載されている：２００９年８月２７日公開の、米国特許出願公開第２００９／０２１６９２８Ａ１号公報、Ｈｅｌｌｅｒらの発明の名称「Ｓｙｓｔｅｍ, Ｍｅｔｈｏｄ ａｎｄ Ｃｏｍｐｕｔｅｒ Ｐｒｏｇｒａｍ Ｐｒｏｄｕｃｔ ｆｏｒ Ｐｒｏｖｉｄｉｎｇ ａ Ｎｅｗ Ｑｕｉｅｓｃｅ Ｓｔａｔｅ,」；２００９年８月２７日公開の、米国特許出願公開第２００９／０２１６９２９Ａ１号公報、Ｈｅｌｌｅｒらの発明の名称「Ｓｙｓｔｅｍ, Ｍｅｔｈｏｄ ａｎｄ Ｃｏｍｐｕｔｅｒ Ｐｒｏｇｒａｍ Ｐｒｏｄｕｃｔ ｆｏｒ Ｐｒｏｖｉｄｉｎｇ ａ Ｐｒｏｇｒａｍｍａｂｌｅ Ｑｕｉｅｓｃｅ Ｆｉｌｔｅｒｉｎｇ Ｒｅｇｉｓｔｅｒ」；２００９年８月２７日公開の、米国特許出願公開第２００９／０２１７２６４Ａ１号公報、Ｈｅｌｌｅｒらの発明の名称「Ｍｅｔｈｏｄ, Ｓｙｓｔｅｍ ａｎｄ Ｃｏｍｐｕｔｅｒ Ｐｒｏｇｒａｍ Ｐｒｏｄｕｃｔ ｆｏｒ Ｐｒｏｖｉｄｉｎｇ Ｆｉｌｔｅｒｉｎｇ ｏｆ Ｇｕｅｓｔ２ Ｑｕｉｅｓｃｅ Ｒｅｑｕｅｓｔｓ」；２００９年８月２７日公開の、米国特許出願公開第２００９／０２１７２６９Ａ１号公報、Ｈｅｌｌｅｒらの発明の名称「Ｓｙｓｔｅｍ, Ｍｅｔｈｏｄ ａｎｄ Ｃｏｍｐｕｔｅｒ Ｐｒｏｇｒａｍ Ｐｒｏｄｕｃｔ ｆｏｒ Ｐｒｏｖｉｄｉｎｇ Ｍｕｌｔｉｐｌｅ Ｑｕｉｅｓｃｅ Ｓｔａｔｅ Ｍａｃｈｉｎｅｓ」；２００６年２月７日発行の米国特許第６，９９６，６９８Ｂ２号、Ｓｌｅｇｅｌらの発明の名称「Ｂｌｏｃｋｅｄ Ｐｒｏｃｅｓｓｉｎｇ Ｒｅｓｔｒｉｃｔｉｏｎｓ Ｂａｓｅｄ Ｏｎ Ａｄｄｒｅｓｓｅｓ,」；２００６年３月２８日発行の米国特許第７，０２０，７６１Ｂ２号、Ｓｌｅｇｅｌらの発明の名称「Ｂｌｏｃｋｉｎｇ Ｐｒｏｃｅｓｓｉｎｇ Ｒｅｓｔｒｉｃｔｉｏｎｓ Ｂａｓｅｄ Ｏｎ Ｐａｇｅ Ｉｎｄｉｃｅｓ」；および２００９年５月５日発行の米国特許第７，５３０，０６７Ｂ２号、Ｓｌｅｇｅｌらの発明の名称「Ｆｉｌｔｅｒｉｎｇ Ｐｒｏｃｅｓｓｏｒ Ｒｅｑｕｅｓｔｓ Ｂａｓｅｄ Ｏｎ Ｉｄｅｎｔｉｆｉｅｒｓ」。

【００７９】

上記で詳細に説明したのは、多重処理環境における処理を容易化する逐次化メカニズムである。本逐次化メカニズムは、逐次化処理に使われるＩＰＴＥインターロックのゲスト

部分を除去することを可能にする。一つの事例において、この逐次化メカニズムは、システム中の全プロセッサが、ハイパーバイザによって前にセットされたＩＰＴＥインターロックを順守することを保証するＤＩＡＧ命令を含む。また、この命令は、一切の未処理のＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇ命令が完了するのを待つ。これは、１）ゲストによって実行される今後の一切のＩＰＴＥ、ＩＤＴＥ、ＣＳＰ／Ｇは、ＩＰＴＥインターロック・ビットがセットされているので、ホストにインターセプトされること、および、２）ハイパーバイザが自分の命令エミュレーションを開始する前に、それ以前の一切のＩＰＴＥ、ＩＤＴＥ、またはＣＳＰ／Ｇが完了されていることを保証する。

【００８０】

ハードウェアの変更は必要がなく、ハイパーバイザによる非常に小さな変更、すなわち、システム逐次化機能をサポートするために、ホストＩＰＴＥインターロックをセットした後でこの機能を発行すること、が必要なだけである。さらに、ＩＰＴＥインターロックに関し、ハイパーバイザによって使われるリカバリ・アルゴリズムに対する変更も必要がない。ハイパーバイザが、旧来のゲスト・ロックのチェックを継続する場合にあっては、これはさらに、ハイパーバイザに見えるようにしてファームウェアが機能を無効化することが可能となり、これによって検査におけるより大きなフレキシビリティ、および新技法によって認識されるパフォーマンス・ゲインを測定する能力が可能となる。

【００８１】

当業者であればよく理解するであろうが、本発明の態様は、システム、方法、またはコンピュータ・プログラム製品として具現することができる。従って、本発明の態様は、全体がハードウェアの実施形態、全体がソフトウェアの実施形態（ファームウェア、常駐ソフトウェア、マイクロコードなどを含む）、あるいはソフトウェアおよびハードウェア態様を組み合わせた実施形態の形を取ることがでる。これらは、どれも本明細書では一般的に全て「回路」、「モジュール」、または「システム」と呼ばれる。さらに、本発明の態様は、内部に具体化されたコンピュータ可読プログラム・コードを有する一つ以上のコンピュータ可読媒体（群）中に具現されたコンピュータ・プログラム製品の形を取ることもできる。

【００８２】

一つ以上のコンピュータ可読媒体（群）の任意の組み合わせを用いることができる。コンピュータ可読媒体はコンピュータ可読記憶媒体とすることができる。コンピュータ可読記憶媒体は、例えば、以下に限らないが、電子的、磁気的、光学的、電磁気的、赤外的、または半導体の、システム、装置、もしくはデバイス、あるいはこれらの任意の適切な組み合わせとすることができる。コンピュータ可読記憶媒体のさらに具体的な例（非包括的リスト）には、一つ以上の配線を有する電気接続、携帯型コンピュータ・ディスケット、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）、消去可能プログラム可能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）、光ファイバ、携帯型コンパクト・ディスク読み取り専用メモリ（ＣＤ－ＲＯＭ：ｃｏｍｐａｃｔ　ｄｉｓｃ　ｒｅａｄ－ｏｎｌｙ　ｍｅｍｏｒｙ）、光記憶デバイス、磁気記憶デバイス、またはこれらの任意の適切な組み合わせが含まれよう。本文書の文脈において、コンピュータ可読記憶媒体は、命令実行システム、装置、またはデバイスによってまたはこれらに関連させて使用するためのプログラムを、包含または格納できる任意の有形媒体とすることができる。

【００８３】

図１９を参照すると、一つの例において、コンピュータ・プログラム製品１３００は、例えば、本発明の一つ以上の態様を提供し支援するためのコンピュータ可読プログラム・コード手段またはロジック１３０４を格納する、一つ以上のコンピュータ可読記憶媒体１３０２を含む。

【００８４】

コンピュータ可読媒体中に具現されたプログラム・コードは、以下に限らないが、無線、有線、光ファイバ・ケーブル、ＲＦなど、またはこれらの任意の適した組み合わせを含め、適切な媒体を用いて送信することができる。

【００８５】

本発明の態様のオペレーションを実行するためのコンピュータ・プログラム・コードは、Ｊａｖａ（Ｒ）、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類似のプログラミング言語などの従来式手続き型プログラミング言語を含め、一つ以上のプログラミング言語の任意の組み合わせで記述することができる。このプログラム・コードは、スタンドアロン・ソフトウェア・パッケージとしてユーザのコンピュータで全体的に実行することも、ユーザのコンピュータで部分的に実行することもでき、一部をユーザのコンピュータで一部を遠隔コンピュータで実行することもでき、あるいは遠隔のコンピュータまたはサーバで全体的に実行することもできる。後者のシナリオでは、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、遠隔コンピュータをユーザのコンピュータに接続することもでき、あるいは（例えばインターネット・サービス・プロバイダを使いインターネットを介し）外部のコンピュータへの接続を行うこともできる。

【００８６】

本明細書では、本発明の実施形態による方法、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もしくはブロック図またはその両方を参照しながら、本発明の態様を説明している。フローチャート図もしくはブロック図またはその両方の各ブロック、および、フローチャート図もしくはブロック図またはその両方中のブロックの組み合わせは、コンピュータ・プログラム命令によって実行可能であることが理解されよう。コンピュータまたは他のプログラム可能データ処理装置のプロセッサを介して実行されるこれらのコンピュータ・プログラム命令が、フローチャートもしくはブロック図またはその両方のブロックまたはブロック群中に規定された機能群／処理群を実施するための手段を生成するように、これらのコンピュータ・プログラム命令を、汎用コンピュータ、特殊用途コンピュータ、またはマシンを形成する他のプログラム可能データ処理装置のプロセッサに供給することができる。

【００８７】

また、これらのコンピュータ・プログラム命令を、コンピュータ、他のプログラム可能データ処理装置、または他のデバイスに対し特定の仕方で機能するよう命令できるコンピュータ可読媒体に格納し、そのコンピュータ可読媒体に格納された命令が、フローチャートもしくはブロック図またはその両方のブロックまたはブロック群中に規定された機能／処理を実施する命令群を包含する製造品を形成するようにすることができる。

【００８８】

同様に、コンピュータ・プログラム命令を、コンピュータ、他のプログラム可能データ処理装置、または他のデバイスにロードして、これらコンピュータ上、他のプログラム可能装置上、またはコンピュータ実行のプロセスを生成する他のデバイス上で一連のオペレーション・ステップを実行させて、これらコンピュータ上または他のプログラム可能装置上で実行されるこれらの命令が、フローチャートもしくはブロック図またはその両方のブロックまたはブロック群中に規定された機能群／処理群を実施するためのプロセスを提供するようにすることもできる。

【００８９】

図中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よる、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、およびオペレーションを例示している。この点に関し、フローチャートまたはブロック図中の各ブロックは、規定の論理機能（群）を実装するための一つ以上の実行可能命令を含む、モジュール、セグメント、またはコードの部分を表し得る。また、一部の別の実装

においては、ブロック中に記載された機能が、図に記載された順序を外れて行われることがあり得ることに留意すべきである。例えば、連続して示された２つのブロックが、実際にはほぼ同時に実行されることがあり、関与する機能によって、時にはこれらのブロックが逆の順序で実行されることもあり得る。さらに、ブロック図もしくはフローチャート図またはその両方の各ブロック、およびブロック図もしくはフローチャート図またはその両方中のブロック群の組み合わせは、特定の機能または処理を実施する、専用ハードウェア・ベースのシステム、または専用ハードウェアとコンピュータ命令との組み合わせによって実装可能なことにも留意すべきである。

【００９０】

上記に加え、顧客環境の管理を提供するサービス・プロバイダによって、本発明の一つ以上の態様を提供し、提案し、展開し、管理し、保守などすることができる。例えば、サービス・プロバイダは、一人以上の顧客に対し、本発明の一つ以上の態様を遂行するコンピュータ・コードもしくはコンピュータ・インフラストラクチャまたはその両方を生成し、維持し、サポートすることなどができる。サービス・プロバイダは、見返りとして、例えば予約申し込みもしくは自由契約またはその両方の下で顧客から支払いを受けることができる。これに加えて、またはこれに換えて、サービス・プロバイダは、一人以上の第三者に対して広告コンテンツを販売することによって支払いを受けることもできる。

【００９１】

本発明の一つの態様において、本発明の一つ以上の態様を実施するためのアプリケーションを展開することができる。一つの事例として、アプリケーションの展開には、本発明の一つ以上の態様を遂行するため動作可能なコンピュータ・インフラストラクチャを提供することが含まれる。

【００９２】

本発明のさらなる態様として、コンピューティング・システム中にコンピュータ可読コードを組み込むことを含めて、コンピューティング・インフラストラクチャを展開することができ、上記コードは、上記コンピューティング・システムと相まって、本発明の一つ以上の態様を遂行する能力を有する。

【００９３】

本発明のまださらなる態様として、コンピュータ可読コードをコンピュータ・システム中に組み込むことを含む、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を提供することができる。上記コンピュータ・システムは、コンピュータ可読媒体を含み、このコンピュータ可読媒体は、本発明の一つ以上の態様を含む。上記コードは、上記コンピュータ・システムと相まって、本発明の一つ以上の態様を遂行する能力を有する。

【００９４】

上記でさまざまな実施形態を説明してきたが、これらは単なる事例である。例えば、他のアーキテクチャのコンピューティング環境も、本発明の一つ以上の態様を組み込み使用することができる。例として、Ｐｏｗｅｒ　Ｓｙｓｔｅｍｓ（ＩＢＭ社の商標）サーバもしくはインターナショナル・ビジネス・マシーンズ・コーポレーションが提供している他のサーバなど、Ｓｙｓｔｅｍ　ｚ（ＩＢＭ社の登録商標）サーバ以外のサーバ、または他の会社のサーバも、本発明の一つ以上の態様を包含し、使用し、もしくはこれから便益を得、またはこれらの組み合わせを実施することができる。さらに、複数のプロセッサを使った単一イメージ・システムは、本発明の一つ以上の態様を組み込み使用することができる。さらに、本逐次化メカニズムは、本明細書に記載した命令以外の命令によって、またはそれらの命令と共に用いることができる。さらに、ＤＩＡＧ命令は、本明細書に記載したものと違えて実装することができ、または、これより多い、少ない、もしくは異なるフィールドを持たせることもでき、あるいはその両方ができる。さらに、逐次化は、ＤＩＡＧ命令以外のメカニズムを使って遂行することもできる。さらに、ＤＡＴおよびＡＲＴ以外の変換スキームを用いることができ、アクセス対象のデータ構造体を変換テーブル以外のものとすることもできる。また、本発明の一つ以上の態様は、ホストおよびゲストを含

む環境以外の環境にも適用可能である。例えば、本逐次化メカニズムを使って、共通のロックを共用する２つのエンティティの処理を逐次化することができる。一つのエンティティは、更新される（例、セット、リセット、またはインクリメントされる）エンティティ・ロックを有し、他方のエンティティは、本逐次化メカニズムによればエンティティ・ロックを必要としない。エンティティ・ロックを有するエンティティは、例えば、エンティティ・ロックの更新回数が少なくなるエンティティである。多くの発展性が存在する。

【００９５】

さらに、他の種類のコンピューティング環境も本発明の一つ以上の態様から便益を得ることができる。一例として、プログラム・コードを格納もしくは実行またはその両方を行うのに適したデータ処理システムが使用可能である。このデータ処理システムは、直接的に、またはシステム・バスを介して間接的にメモリ・エレメントに連結された少なくとも２つのプロセッサを含む。メモリ・エレメントには、例えば、プログラム・コードを実際に実行する過程で用いられるローカル・メモリと、大容量記憶装置と、実行の過程で大容量記憶装置からコードを読み出さなければならない回数を低減するため、少なくとも一部のプログラム・コードを一時的に記憶するキャッシュ・メモリとが含まれる。

【００９６】

入力／出力すなわちＩ／Ｏデバイス（以下に限らないが、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、および他の記憶媒体などを含む）は、直接に、もしくは介在Ｉ／Ｏコントローラを介してのいずれかによってシステムに連結することができる。また、ネットワーク・アダプタをシステムに連結して、データ処理システムが、介在する私的ネットワークまたは公衆ネットワークを介して、他のデータ処理システムあるいは遠隔のプリンタまたは記憶デバイスに連結することを可能にすることもできる。モデム、ケーブル・モデム、およびイーサネット（Ｒ）カードは、利用可能なネットワーク・アダプタのほんの一例である。

【００９７】

図２０を参照すると、本発明の一つ以上の態様を実装するためのホスト・コンピュータ・システム５０００の典型的なコンポーネントが描かれている。典型的なホスト・コンピュータ５０００は、メモリ（すなわち、中央ストレージ）５００２と通信している一つ以上のＣＰＵ５００１を含むほか、ストレージ媒体デバイス５０１１と、他のコンピュータまたはＳＡＮなどと通信するためのネットワーク５０１０とへのＩ／Ｏインタフェースを含む。ＣＰＵ５００１は、設計された命令セットおよび設計された機能を有するアーキテクチャに準拠している。ＣＰＵ５００１は、プログラム・アドレス（仮想アドレス）をメモリの実際のアドレスに変換するための動的アドレス変換（ＤＡＴ）５００３を持つことができる。通常、ＤＡＴは、コンピュータ・メモリ５００２のブロックへの後々のアクセスがアドレス変換の遅れを要さないように、変換をキャッシングするための変換ルックアサイド・バッファ（ＴＬＢ）５００７を含む。通常、キャッシュ５００９は、コンピュータ・メモリ５００２とプロセッサ（ＣＰＵ）５００１との間で用いられる。キャッシュ５００９は、複数のＣＰＵが利用可能な大型のキャッシュ、および大型キャッシュと各ＣＰＵとの間のより小型でより高速の（低位レベルの）キャッシュに階層化することができる。一部の実装において、低位レベルのキャッシュを分割して、命令フェッチ用およびデータ・アクセス用に別個の低位レベル・キャッシュを備えることができる。一つの実施形態において、命令は、命令フェッチ・ユニット５００４によって、メモリ５００２からキャッシュ５００９を経由してフェッチされる。命令は、命令復号ユニット５００６によって復号され、（一部の実施形態では他の命令と共に）命令実行ユニットまたはユニット群５００８に送られる。通常、例えば、算術実行ユニット、浮動小数点実行ユニット、および分岐命令実行ユニットなど、いくつかの実行ユニット５００８が用いられる。命令は、実行ユニットによって実行され、必要に応じ、命令が指定するレジスタまたはメモリのオペランドにアクセスして実行される。メモリ５００２のオペランドがアクセス（ロードまたは格納）される場合は、通常、実行されている命令の制御の下にロード／ストア・ユニット５００５がそのアクセスを処理する。命令は、ハードウェア回路において、もしくは内

部マイクロコード（ファームウェア）において、またはその両方の組み合わせによって実行することができる。

【００９８】

前述のように、コンピュータ・システムは、ローカル（または主）ストレージ内の情報、並びにアドレッシング、保護、参照、および変更の記録を含む。アドレッシングのいくつかの態様は、アドレスのフォーマット、アドレス空間の概念、種々のタイプのアドレス、および一つのタイプのアドレスを別のタイプのアドレスに変換する方法を含む。一部の主ストレージは、永続的に割り当てられたストレージ位置を含む。主ストレージは、直接アドレス指定可能なデータの高速アクセス・ストレージをシステムに提供する。データおよびプログラムの双方は、（入力デバイスから）主ストレージ中にロードされることになり、その後で処理が可能になる。

【００９９】

主ストレージには、時としてキャッシュとも呼ばれる、より小さくより高速アクセスのバッファ・ストレージを含めることができる。通常、キャッシュは、ＣＰＵまたはＩ／Ｏプロセッサに物理的に関連付けられる。物理的構造および個別的なストレージ媒体の使用の影響は、性能に対するものを除き、一般にプログラムにより観察することはできない。

【０１００】

命令およびデータ・オペランドに対し、別個のキャッシュを保持することができる。キャッシュ内の情報は、キャッシュ・ブロックまたはキャッシュ・ライン（または短縮してライン）と呼ばれる整数境界（ｉｎｔｅｇｒａｌ　ｂｏｕｎｄａｒｙ）上の連続したバイト中に維持される。あるモデルでは、キャッシュ・ラインのサイズをバイト数で返す、ＥＸＴＲＡＣＴ　ＣＡＣＨＥ　ＡＴＴＲＩＢＵＴＥ命令を提供することができる。また、あるモデルでは、データまたは命令キャッシュへのストレージのプリフェッチ、あるいは、キャッシュからのデータのリリースに影響を与える、ＰＲＥＦＥＴＣＨ　ＤＡＴＡおよびＰＲＥＦＥＴＣＨ　ＤＡＴＡ　ＲＥＬＡＴＩＶＥ　ＬＯＮＧ命令を提供することができる。

【０１０１】

ストレージは、長い水平方向のビットのストリングと考えられる。ほとんどのオペレーションにおいて、ストレージへのアクセスは、左から右への順序で進む。ビットのストリングは８ビット単位に分割される。８ビットの単位は１バイトと呼ばれ、全ての情報のフォーマットの基本的な構成ブロック（ｂｕｉｌｄｉｎｇ　ｂｌｏｃｋ）である。ストレージ内の各バイトの位置は、負でない一意の整数により識別され、この整数がそのバイト位置のアドレスであり、簡単に言えば、バイト・アドレスである。隣接するバイト位置は、連続するアドレスを有し、左の０で始まり、左から右への順序で進む。アドレスは、符号なしの２進整数であり、２４ビット、３１ビット、又は６４ビットである。

【０１０２】

情報は、ストレージとＣＰＵ又はチャネル・サブシステムとの間で、一度に１バイトずつ、又は１バイト・グループずつ伝送される。別段の指定がなければ、例えばｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）では、ストレージ中の１バイト・グループは、そのグループの左端のバイトによってアドレス指定される。グループ内のバイト数は、遂行対象のオペレーションによって暗黙裡に定められるか、又は明示的に指定される。ＣＰＵオペレーションに使用される場合、バイト・グループはフィールドと呼ばれる。各バイト・グループ内において、例えばｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）では、ビットは、左から右の順序で番号が付される。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）では、左端のビットは「上位（ｈｉｇｈ－ｏｒｄｅｒ）」ビットと呼ばれることがあり、右端ビットは「下位（ｌｏｗ－ｏｒｄｅｒ）」ビットと呼ばれることがある。しかしながら、ビット番号は、ストレージ・アドレスではない。バイトだけが、アドレス指定をすることができる。ストレージ中のあるバイトの個別ビットに対して操作を行うためには、そのバイト全体にアクセスする必要がある。１バイトの中のビットには、（例えば、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）では）左から右に

０から７までの番号が付される。１つのアドレスの中のビットには、２４ビット・アドレスに対しては８～３１または４０～６３の番号を付すことができ、あるいは、３１ビット・アドレスに対しては１～３１または３３～６３の番号を付すこともできる。６４ビット・アドレスに対しては０～６３の番号が付される。複数バイトから成る他の一切の固定長フォーマット内では、そのフォーマットを構成するビットには０から始まる連続番号が付される。エラー検出のため、および望ましくは訂正のために、各バイトまたはバイト・グループと共に、一つ以上の検査ビットが伝送され得る。かかる検査ビットは、マシンにより自動的に生成されるものであり、プログラムが直接制御することはできない。ストレージ容量はバイト数で表わされる。ストレージ・オペランド・フィールドの長さが命令のオペレーション・コードで暗黙的に指定される場合、そのフィールドは固定長を有すると言われ、その固定長は、１バイト、２バイト、４バイト、８バイト、または１６バイトとすることができる。一部の命令では、より長いフィールドが暗黙的に指定されることもある。ストレージ・オペランド・フィールドの長さが暗黙的に指定されず、明示的に記述される場合は、そのフィールドは可変長を有するといわれる。可変長オペランドは、１バイト（あるいは、一部の命令については、２バイトの倍数または他の倍数）のインクリメントにより長さが変化し得る。情報がストレージ内に置かれるとき、ストレージへの物理パスの幅が格納されるフィールドの長さを上回ることがあっても、指定されたフィールド内に含まれるバイト位置の内容のみが置き換えられる。

【０１０３】

特定の情報単位は、ストレージ中の整数境界上にあることになる。そのストレージ・アドレスがバイトでの単位の長さの倍数であるとき、境界は、情報単位に対して整数であるといわれる。整数境界上にある２バイト、４バイト、８バイト、および１６バイトのフィールドには、特別な名称が付与される。ハーフワード（ｈａｌｆｗｏｒｄ）は、２バイト境界上にある２個の連続したバイトのグループであり、これは、命令の基本的な構成ブロックである。ワード（ｗｏｒｄ）は、４バイト境界上にある４個の連続したバイトのグループである。ダブルワード（ｄｏｕｂｌｅｗｏｒｄ）は、８バイト境界上にある８個の連続したバイトのグループである。クワドワード（ｑｕａｄｗｏｒｄ）は、１６バイト境界上にある１６個の連続したバイトのグループである。ストレージ・アドレスが、ハーフワード、ワード、ダブルワード、およびクワドワードを指定する場合、そのアドレスのバイナリ表現は、それぞれ、右端の１個、２個、３個、又は４個のビットが０になる。命令は、２バイトの整数境界上にあることになる。ほとんどの命令のストレージ・オペランドは、境界合わせ（ｂｏｕｎｄａｒｙ　ａｌｉｇｎｍｅｎｔ）要件を持たない。

【０１０４】

命令オペランドとデータ・オペランドとに対して別個のキャッシュを実装するデバイスにおいては、ストアが、後にフェッチされる命令を変更するかどうかに関係なく、後にそこから命令がフェッチされるキャッシュ・ライン中にプログラムが格納される場合に、著しい遅延が生じることがある。

【０１０５】

一つの実施形態において、本発明はソフトウェア（ライセンス内部コード、ファームウェア、マイクロコード、ミリコード、ピココードなどといわれることもあり、これらのいずれも本発明と整合できよう）によって実践することができる。図２０を参照すると、本発明を具現するソフトウェア・プログラム・コードは、通常、ホスト・システム５０００のプロセッサ（ＣＰＵ）５００１によって、ＣＤ－ＲＯＭドライブ、テープ・ドライブ、またはハード・ドライブなど、長期ストレージ媒体５０１１からアクセスされる。該ソフトウェア・プログラム・コードは、ディスケット、ハード・ドライブ、またはＣＤ－ＲＯＭなど、データ処理システムと共に使われるさまざまな周知の媒体のいずれの中にも具現することができる。該コードはかかる媒体で配布することができ、または、他のコンピュータ・システムのユーザが使用するために、一つのコンピュータ・システムのコンピュータ・メモリ５００２またはストレージから、ネットワーク５０１０を介して、該コードをかかる他のシステムのユーザに配送することもできる。

【０１０６】
このソフトウェア・プログラム・コードは、さまざまなコンピュータ・コンポーネントおよび一つ以上のアプリケーション・プログラムの機能および相互作用を制御するオペレーティング・システムを含む。プログラム・コードは、通常、ストレージ媒体デバイス５０１１から、プロセッサ（ＣＰＵ）５００１が処理のため利用可能な、比較的により高速のコンピュータ・ストレージ５００２にページングされる。ソフトウェア・プログラム・コードを、メモリ中、物理媒体上に具現するための、もしくはネットワークを介してソフトウェア・コードを配送するための、またはその両方の技法および方法は周知であり、本明細書ではこれ以上説明しない。プログラム・コードが生成されて、それが有形の媒体（以下に限らないが、電子メモリ・モジュール（ＲＡＭ）、フラッシュ・メモリ、コンパクト・ディスク（ＣＤ：Ｃｏｍｐａｃｔ　Ｄｉｓｃ）、ＤＶＤ、磁気テープなどを含む）に格納されたものは、しばしば「コンピュータ・プログラム製品」と呼ばれる。このコンピュータ・プログラム製品の媒体は、望ましくはコンピュータ・システム中の処理回路によって実行されるために、通常、該処理回路による読み取りが可能となっている。

【０１０７】
図２１は、本発明が実践可能な、典型的ワークステーションまたはサーバ・ハードウェア・システムを示す。図２１のシステム５０２０は、随意の周辺デバイスを含む、パーソナル・コンピュータ、ワークステーション、またはサーバなどの典型的な基本的コンピュータ・システム５０２１を含む。この基本的コンピュータ・システム５０２１は、一つ以上のプロセッサ５０２６、およびプロセッサ（群）５０２６とシステム５０２１の既知の技法による他のコンポーネントとの間の通信を可能にするため用いられるバスを含む。このバスは、プロセッサ５０２６をメモリ５０２５および長期ストレージ５０２７に接続しており、この長期ストレージには、例えば、ハード・ドライブ（例として、磁気媒体、ＣＤ、ＤＶＤ、およびフラッシュ・メモリを含む）またはテープ・ドライブを含めることができる。また、システム５０２１には、マイクロプロセッサ５０２６を、バスを介してキーボード５０２４、マウス５０２３、プリンタ／スキャナ５０３０もしくは他のインタフェース・デバイスまたはこれらの組み合わせなどの一つ以上のインタフェース・デバイスに接続する、ユーザ・インタフェース・アダプタを含めることも可能である。上記他のインタフェース・デバイスは、例えば、タッチ感知スクリーン、デジタル化入力パッドなど任意のユーザ・インタフェース・デバイスとすることができる。また、バスは、ＬＣＤスクリーンまたはモニタなどのディスプレイ・デバイス５０２２を、ディスプレイ・アダプタを介してマイクロプロセッサ５０２６に接続する。

【０１０８】
システム５０２１は、ネットワーク５０２９との通信が可能なネットワーク・アダプタ５０２８を経由して、他のコンピュータまたはコンピュータのネットワークと通信することができる。ネットワーク・アダプタの例には、通信チャネル、トークン・リング、イーサネット（Ｒ）、またはモデムがある。上記に換えて、システム５０２１は、ＣＤＰＤ（ｃｅｌｌｕｌａｒ　ｄｉｇｉｔａｌ　ｐａｃｋｅｔ　ｄａｔａ（セルラー・デジタル・パケット・データ））カードなどの無線インタフェースを使って通信することもできる。システム５０２１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）内で、そのネットワークの他のコンピュータと交信することができ、あるいは、システム５０２１は、別のコンピュータとのクライアント／サーバ・アレンジメントにおけるクライアントにすることなどもできる。こういった構成の全て、並びに適切な通信ハードウェアおよびソフトウェアは、当該技術分野では周知である。

【０１０９】
図２２は、本発明を実践できるデータ処理ネットワーク５０４０を示す。データ処理ネットワーク５０４０には、そのそれぞれが複数の個別ワークステーション５０４１、５０４２、５０４３、５０４４を包含可能な、無線ネットワークおよび有線ネットワークなど、複数の個別ネットワークを含めることができる。さらに、当業者ならよく理解している

ように、一つ以上のＬＡＮを含めることができ、ＬＡＮには、ホスト・プロセッサに連結された複数のインテリジェント・ワークステーションを含めることができる。

【０１１０】

さらに図２２を参照すると、このネットワークには、ゲートウェイ・コンピュータ（クライアント・サーバ５０４６）またはアプリケーション・サーバ（データ・リポジトリにアクセス可能で、またワークステーション５０４５から直接にアクセスを受けることが可能な遠隔サーバ５０４８）などのメインフレーム・コンピュータまたはサーバも含めることができる。ゲートウェイ・コンピュータ５０４６は、各個別ネットワークへのエントリ点としての役目をする。一つのネットワーク・プロトコルを別のプロトコルに接続するとき、ゲートウェイが必要となる。ゲートウェイ５０４６は、望ましくは、通信リンクを使って別のネットワーク（例えば、インターネット５０４７）に連結が可能である。また、ゲートウェイ５０４６は、通信リンクを使って、一つ以上のワークステーション５０４１、５０４２、５０４３、５０４４に直接連結することができる。ゲートウェイ・コンピュータは、インターナショナル・ビジネス・マシーンズ・コーポレーションから入手可能なＩＢＭ　ｅＳｅｒｖｅｒ（ＩＢＭ社の商標）、Ｓｙｓｔｅｍ　ｚ（ＩＢＭ社の登録商標）サーバを用いて実装することができる。

【０１１１】

図２１および図２２を同時に参照すると、本発明を具現可能なソフトウェア・プログラミング・コードには、システム５０２０のプロセッサ５０２６によって、ＣＤ－ＲＯＭドライブまたはハード・ドライブなどの長期ストレージ媒体５０２７からアクセスすることができる。ソフトウェア・プログラミング・コードは、ディスケット、ハード・ドライブ、またはＣＤ－ＲＯＭなどデータ処理システムと共に使われるさまざまな周知の媒体のいずれにも具現することができる。該コードはかかる媒体で配布することができ、または、他のコンピュータ・システムのユーザが使用するために、一つのコンピュータ・システムのメモリまたはストレージから、ネットワークを介して、該コードをこういった他のシステムのユーザ５０５０、５０５１に配送することもできる。

【０１１２】

上記に換えて、プログラミング・コードをメモリ５０２５中に具現し、プロセッサ・バスを使ってプロセッサ５０２６にアクセスさせることもできる。かかるプログラミング・コードは、さまざまなコンピュータ・コンポーネントおよび一つ以上のアプリケーション・プログラム５０３２の機能および相互作用を制御するオペレーティング・システムを含む。プログラム・コードは、通常、ストレージ媒体デバイス５０２７から、プロセッサ５０２６が処理のために利用可能な高速のメモリ５０２５にページングされる。ソフトウェア・プログラミング・コードをメモリ中、物理媒体上に具現するための、もしくはネットワークを介してソフトウェア・コードを配送するための、またはその両方の技法および方法は周知であり、本明細書ではこれ以上説明しない。プログラム・コードが生成されて、それが有形の媒体（以下に限らないが、電子メモリ・モジュール（ＲＡＭ）、フラッシュ・メモリ、コンパクト・ディスク（ＣＤ）、ＤＶＤ、磁気テープなどを含む）に格納されたものは、しばしば「コンピュータ・プログラム製品」と呼ばれる。このコンピュータ・プログラム製品の媒体は、望ましくはコンピュータ・システム中の処理回路によって実行されるために、通常、該処理回路による読み取りが可能となっている。

【０１１３】

プロセッサにとって最も容易に利用可能なキャッシュ（通常、プロセッサの他のキャッシュよりも高速で小型）は、最低位（Ｌ１またはレベル１）キャッシュであり、主ストア（主メモリ）は最高位レベル・キャッシュ（レベルが３つある場合はＬ３）である。最低位キャッシュは、多くの場合、実行されるマシン命令を保持する命令キャッシュ（Ｉ－キャッシュ）と、データ・オペランドを保持するデータ・キャッシュ（Ｄ－キャッシュ）とに分割されている。

【０１１４】

図２３を参照すると、プロセッサ５０２６のための例示的なプロセッサの実施形態が示

されている。通常、プロセッサのパフォーマンスを向上するために、一つ以上のレベルのキャッシュ５０５３が、メモリ・ブロックをバッファするのに用いられる。キャッシュ５０５３は、使用の可能性が高いメモリ・データのキャッシュ・ラインを保持する高速バッファである。典型的なキャッシュ・ラインは、６４バイト、１２８バイト、または２５６バイトのメモリ・データである。多くの場合、データをキャッシングするためというより、命令をキャッシュングするために別個のキャッシュが用いられる。キャッシュ・コヒーレンス（メモリおよびキャッシュ群中のラインのコピーの同期化）は、多くの場合、当該技術分野で周知のさまざまな「スヌープ」アルゴリズムによって提供される。プロセッサ・システムの主メモリ・ストレージ５０２５は、しばしばキャッシュといわれる。４段階レベルのキャッシュ５０５３を有するプロセッサ・システムにおいて、主ストレージ５０２５は、レベル５（Ｌ５）キャッシュと呼ばれることもある。これは、主ストレージ５０２５が、通常、より高速であり、コンピュータ・システムに利用可能な不揮発性ストレージ（ＤＡＳＤ、テープなど）の一部を保持するだけだからである。主ストレージ５０２５は、オペレーティング・システムによって主ストレージ５０２５にページ・インおよびページ・アウトされるデータのページを「キャッシュする」。

【０１１５】

プログラム・カウンタ（命令カウンタ）５０６１は、現在の実行対象の命令のアドレスを記録する。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）のプロセッサ中のプログラム・カウンタは６４ビットであり、従前のアドレッシング限度をサポートするため３１または２４ビットに切りつめることができる。プログラム・カウンタは、通常、コンテキスト切り替えの期間それが持続するように、コンピュータのＰＳＷ（ｐｒｏｇｒａｍ　ｓｔａｔｕｓ　ｗｏｒｄ（プログラム状態ワード））中に具現される。しかして、プログラム・カウンタ値を有する、進行中のプログラムに対し、例えばオペレーティング・システムが割り込むことが可能である（プログラム環境からオペレーティング・システム環境へのコンテキスト切り替え）。プログラムのＰＳＷは、該プログラムがアクティブでない間はプログラム・カウンタ値を維持し、オペレーティング・システムの実行中は、オペレーティング・システムの（ＰＳＷ中の）プログラム・カウンタが使われる。通常、プログラム・カウンタは、現行の命令のバイトの数に等しい量だけインクリメントされる。ＲＩＳＣ（Ｒｅｄｕｃｅｄ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｉｎｇ（縮小命令セット・コンピューティング））命令は、通常、固定長であり、一方、ＣＩＳＣ（Ｃｏｍｐｌｅｘ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｉｎｇ（複合命令セット・コンピューティング））命令は、通常、可変長である。ＩＢＭのｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）の命令は、２バイト、４バイト、および６バイトの長さを有するＣＩＳＣ命令である。プログラム・カウンタ５０６１は、例えば、コンテキスト切り替えオペレーション、もしくは分岐命令の分岐成立オペレーションのいずれかによって変更される。コンテキスト切り替えオペレーションでは、現在のプログラム・カウンタ値は、実行されているプログラムについての他の（状態コードなどの）状態情報とともに、プログラム状態ワード中に保存され、実行されることになる新規のプログラム・モジュールの命令をポイントする新しいプログラム・カウンタがロードされる。プログラムが、判断を下すことまたはプログラム内でループすることを可能にするために、分岐命令の結果をプログラム・カウンタ５０６１の中にロードすることによって分岐成立オペレーションが遂行される。

【０１１６】

一般には、プロセッサ５０２６に代わって命令をフェッチするために、命令フェッチ・ユニット５０５５が用いられる。フェッチ・ユニットは、分岐成立命令のターゲット命令である「次順命令」または、コンテキスト切り替えに続くプログラムの最初の命令のいずれかをフェッチする。近年の命令フェッチ・ユニットの多くは、事前フェッチした命令が使われる可能性の尤度に基づいて、推測的に命令を事前フェッチする事前フェッチ技法を用いている。例えば、フェッチ・ユニットは、次順命令およびさらなる逐次命令の追加バイトを含む１６バイトの命令をフェッチすることができる。

【０１１７】
次いで、フェッチされた命令は、プロセッサ５０２６によって実行される。ある実施形態において、フェッチされた命令（群）はフェッチ・ユニットのタスク指名（ｄｉｓｐａｔｃｈ）ユニット５０５６に渡される。タスク指名ユニットは命令（群）を復号し、復号された命令（群）についての情報を適切なユニット５０５７、５０５８、５０６０に転送する。実行ユニット５０５７は、通常、命令フェッチ・ユニット５０５５から、復号された算術命令についての情報を受信し、該命令のオペコードに従ってオペランド上で算術演算を遂行することになる。オペランドは、望ましくは、メモリ５０２５、設計されたレジスタ５０５９、または実行されている命令の即値フィールドのいずれかから、実行ユニット５０５７に提供される。実行の結果を格納する場合、メモリ５０２５中、レジスタ５０５９中、または（制御レジスタ、ＰＳＷレジスタ、などの）他のマシン・ハードウェア中のいずれかに格納される。

【０１１８】
プロセッサ５０２６は、通常、命令の機能を実行するための一つ以上のユニット５０５７、５０５８、５０６０を有する。図２４を参照すると、実行ユニット５０５７は、インタフェース接続ロジック５０７１を経由して、設計された汎用レジスタ５０５９、復号／タスク指名ユニット５０５６、ロード・ストア・ユニット５０６０、および他のプロセッサ・ユニット５０６５と通信することができる。実行ユニット５０５７は、算術論理ユニット（ＡＬＵ：ａｒｉｔｈｍｅｔｉｃ　ｌｏｇｉｃ　ｕｎｉｔ）５０６６の演算対象となる情報を保持するため、いくつかのレジスタ回路５０６７、５０６８、５０６９を用いる。ＡＬＵは、加算、減算、乗算、および除算などの算術演算、並びにａｎｄ（論理積）、ｏｒ（論理和）、および排他的ｏｒ（ＸＯＲ：ｅｘｃｌｕｓｉｖｅ－ｏｒ（排他的論理和））、ｓｈｉｆｔ（シフト）およびｒｏｔａｔｅ（ローテート）、などの論理関数を遂行する。望ましくは、ＡＬＵは、設計により決まる特別な演算にも対応する。他の回路は、例えば、条件コードおよびリカバリ支援ロジックを含め、他の設計された機能５０７２を提供してもよい。通常、ＡＬＵ演算の結果は、出力レジスタ回路５０７０中に保持され、該レジスタ回路は、該結果をさまざまな他の処理機能に転送することができる。多くのプロセッサ・ユニットのアレンジメントがあり、本説明は、一つの実施形態の代表的な理解を提供することだけを意図している。

【０１１９】
例えばＡＤＤ命令は、算術および論理機能を有する実行ユニット５０５７で実行されることになり、一方、例えば浮動小数点命令は、特殊な浮動小数点機能を有する浮動小数点実行ユニットで実行されることになろう。望ましくは、実行ユニットは、命令により識別されたオペランドを、該オペランド上のオペコードが定義する機能を遂行することによって演算する。例えば、ＡＤＤ命令は、実行ユニット５０５７によって、命令のレジスタ・フィールドにより識別される２つのレジスタ５０５９中にあるオペランドに対して実行することができる。

【０１２０】
実行ユニット５０５７は、２つのオペランドに対する算術加算を遂行し、その結果を第三オペランド中に保存し、この第三オペランドは、第三のレジスタとすることも、または２つのソース・レジスタの一つとすることもできる。実行ユニットは、望ましくは、Ｓｈｉｆｔ、Ｒｏｔａｔｅ、Ａｎｄ、Ｏｒ、およびＸＯＲなどのさまざまな論理関数、並びに、加算、減算、乗算、除算のいずれをも含む、さまざまな代数関数を遂行する能力を有する、算術論理ユニット（ＡＬＵ）５０６６を用いる。一部のＡＬＵ５０６６は、スカラー演算用に設計され、一部は浮動小数点用に設計される。データは、アーキテクチャに応じ、（最小有効バイトが最上位バイト・アドレスに置かれる）ビッグ・エンディアン、または（最小有効バイトが最下位バイト・アドレスに置かれる）リトル・エンディアンとすることができる。ＩＢＭのｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）はビッグ・エンディアンである。符号付きフィールドは、アーキテクチャに応じ、符号および絶対値、１の補数、または２の補数とすることができる。２の補数による数は、２の補数では

、負の値または正の値いずれもＡＬＵ内で加算だけが必要なので、ＡＬＵに減算能力を設計する必要がないという点で有利である。数は、通常、簡略表記で表され、例えば、１２ビットのフィールドは４，０９６バイト・ブロックのアドレスを定義し、通常、４Ｋバイト（キロバイト）ブロックとして表される。

【０１２１】

図２５を参照すると、分岐命令を実行するための分岐命令情報は、通常、分岐ユニット５０５８に送信され、分岐ユニットの多くは、他の条件付き演算が完了する前に、分岐の結果を予測するため、分岐履歴テーブル５０８２などの分岐予測アルゴリズムを用いる。現在の分岐命令のターゲットが、フェッチされ、条件付き演算が完了する前に、予測で実行されることになる。条件付き演算が完了したとき、予測で実行された分岐命令は、該条件付き演算の条件および予測した結果に基づいて、完了されるかもしくは放棄される。典型的分岐命令は、条件コードを検査し、条件コードが分岐命令の分岐要件に合致する場合、ターゲット・アドレスに分岐することができ、ターゲット・アドレスは、例えば、レジスタ・フィールドまたは命令の即値フィールドにある数を含めいくつかの数に基づいて計算することができる。分岐ユニット５０５８は、複数の入力レジスタ回路５０７５、５０７６、５０７７、および一つの出力レジスタ回路５０８０を有するＡＬＵ５０７４を用いることができる。分岐ユニット５０５８は、例えば、汎用レジスタ５０５９、復号／タスク指名ユニット５０５６または他の回路５０７３と通信することが可能である。

【０１２２】

命令のグループの実行は、例えば、オペレーティング・システムによって開始されるコンテキスト切り替え、コンテキスト切り替えを引き起こすプログラム例外またはエラー、コンテキスト切り替えを引き起こすＩ／Ｏ割り込み信号、または（マルチスレッド環境における）複数プログラムのマルチスレッド・アクティビティを含め、さまざまな理由によって中断され得る。望ましくは、コンテキスト切り替え処置では、現在実行されているプログラムについての状態情報を保存し、然る後、起動される別のプログラムについての状態情報をロードする。状態情報は、例えば、ハードウェア・レジスタ中またはメモリ中に保存することができる。状態情報は、望ましくは、次に実行する命令をポイントするプログラム・カウンタ値、条件コード、メモリ変換情報、および設計されたレジスタ内容を含む。コンテキスト切り替えアクティビティは、ハードウェア回路、アプリケーション・プログラム、オペレーティング・システム・プログラム、またはファームウェア・コード（マイクロコード、ピココード、ライセンス内部コード（ＬＩＣ：ｌｉｃｅｎｓｅｄ　ｉｎｔｅｒｎａｌ　ｃｏｄｅ））単独で、あるいはこれらの組み合わせによって行うことができる。

【０１２３】

プロセッサは、命令に規定された方法に従ってオペランドにアクセスする。命令は、命令の一部の値を使った即値オペランドを提示することもでき、汎用レジスタあるいは（例えば浮動小数点レジスタなど）特別用途レジスタのいずれかを明示でポイントする一つ以上のレジスタ・フィールドを提示することもできる。命令は、オペランドとして、オペコード・フィールドによって識別されるインプライド・レジスタを用いることができる。命令は、オペランドに対するメモリ位置を用いることもできる。オペランドのメモリ位置は、レジスタ、即値フィールド、またはｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）の長変位機能（ｌｏｎｇ　ｄｉｓｐｌａｃｅｍｅｎｔ　ｆａｃｉｌｉｔｙ）に例示されるように、レジスタと即値フィールドとの組み合わせによって提供することができ、例えば、命令が、ベース・レジスタ、インデックス・レジスタ、および即値フィールド（変位フィールド）を規定し、これらがともに加算されて、メモリ中のオペランドのアドレスを提供する。上記の位置は、別段の指定がなければ、通常は主メモリ（主ストレージ）の中の位置を意味する。

【０１２４】

図２６を参照すると、プロセッサは、ロード／ストア・ユニット５０６０を使ってストレージにアクセスする。ロード／ストア・ユニット５０６０は、メモリ５０５３中のター

ゲット・オペランドのアドレスを取得し、そのオペランドをレジスタ５０５９または他のメモリ５０５３位置中にロードすることによって、ロード・オペレーションを遂行することができ、あるいは、メモリ５０５３中のターゲット・オペランドのアドレスを取得し、レジスタ５０５９または別のメモリ５０５３位置から取得したデータを、メモリ５０５３中のターゲット・オペランド位置に格納することによってストア・オペレーションを遂行することができる。ロード／ストア・ユニット５０６０は、推測処置を行うことができ、命令シーケンスに対して順序外れのシーケンスでメモリにアクセスすることができるが、ただし、ロード／ストア・ユニット５０６０は、プログラムに対しては、命令が順序通りに実行されたような見掛けを維持する。ロード／ストア・ユニット５０６０は、汎用レジスタ５０５９、復号／タスク指名ユニット５０５６、キャッシュ／メモリ・インタフェース５０５３、または他のエレメント５０８３と通信することができ、ストレージ・アドレスを計算し、オペレーションの順序を保つためのパイプライン・シーケンシングを備えるため、さまざまなレジスタ回路、ＡＬＵ５０８５、およびの制御ロジック５０９０を含む。一部のオペレーションは順序外れとなり得るが、当該技術分野で周知のように、ロード／ストア・ユニットは、その順序外れオペレーションが、プログラムに対し順序通り遂行されたように見せるための機能を備えている。

【０１２５】

望ましくは、アプリケーション・プログラムが「見る」アドレスは、多くの場合仮想アドレスといわれる。仮想アドレスは、「論理アドレス」および「実効アドレス」と呼ばれることもある。これらの仮想アドレスは、それらが、以下に限らないが、単にオフセット値を仮想アドレスの前に付ける仕方、一つ以上の変換テーブルを介して仮想アドレスを変換する仕方などを含め、さまざまな動的アドレス変換（ＤＡＴ）技術の一つによって物理メモリ位置にリダイレクトされるという点で仮想的であって、これら変換テーブルは、望ましくは、少なくともセグメント・テーブルおよびページ・テーブルを単独または組み合わせで含み、セグメント・テーブルは、望ましくは、ページ・テーブルをポイントするエントリを有する。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）では、領域第一テーブル、領域第二テーブル、領域第三テーブル、セグメント・テーブル、および随意のページ・テーブルを含む、変換の階層が設けられている。アドレス変換のパフォーマンスは、多くの場合、仮想アドレスを、関連付けられた物理メモリ・アドレス位置にマッピングするエントリを含む変換ルックアサイド・バッファ（ＴＬＢ）を利用することによって向上する。これらエントリは、ＤＡＴが変換テーブルを使って仮想アドレスを変換するときに生成される。その後の仮想アドレス使用においては、低速な逐次変換テーブルへのアクセスよりはむしろ高速なＴＬＢのエントリを利用することができる。ＴＬＢの内容は、ＬＲＵ（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　ｕｓｅｄ（最低使用頻度法））を含め各種の置き換えアルゴリズムによって管理することが可能である。

【０１２６】

プロセッサが、多重プロセッサ・システムのプロセッサである場合、各プロセッサは、コヒーレンシのため、Ｉ／Ｏ、キャッシュ、ＴＬＢ、およびメモリなどの共用リソースのインターロック状態を保つ責任を有する。通常、キャッシュのコヒーレンシを維持するのに「スヌープ」技術が利用されることになる。スヌープ環境において、各キャッシュ・ラインには、共用を容易化するために、共用状態、専用状態、変更状態、無効状態などのいずれか一つにあることを標識することができる。

【０１２７】

Ｉ／Ｏユニット５０５４（図２３）は、プロセッサに、例えば、テープ、ディスク、プリンタ、ディスプレイ、およびネットワークを含む周辺デバイスに連結するための手段を提供する。Ｉ／Ｏユニットの多くは、ソフトウェア・ドライバによってコンピュータ・プログラムに提供される。ＩＢＭ（ＩＢＭ社の登録商標）社製のＳｙｓｔｅｍ　ｚ（ＩＢＭ社の登録商標）などのメインフレームでは、チャネル・アダプタおよびオープン・システム・アダプタが、オペレーティング・システムと周辺デバイスとの間の通信を提供する、メインフレームのＩ／Ｏユニットである。

【０１２８】
さらに、他の種類のコンピューティング環境も、本発明の一つ以上の態様から便益を得ることができる。一例として、前述のように、環境にはエミュレータ（例、ソフトウェアまたは他のエミュレーション・メカニズム）を含めることができ、これによって、（例えば、アドレス変換および設計されたレジスタなど、命令実行、設計された機能を含む）特定のアーキテクチャ、またはそのサブセットが（例えば、プロセッサおよびメモリを有するネイティブ・コンピュータ・システム上で）エミュレートされる。かかる環境において、エミュレータの一つ以上のエミュレーション機能は、たとえ、エミュレータを実行しているコンピュータがエミュレートされている機能と異なるアーキテクチャを有していたとしても、本発明の一つ以上の態様を実装することができる。一例として、エミュレーション・モードにおいて、エミュレートされる特定の命令またはオペレーションは復号され、その個別の命令またはオペレーションを実装するための適切なエミュレーション機能が構築される。

【０１２９】
あるエミュレーション環境において、ホスト・コンピュータは、例えば、命令およびデータを格納するメモリと、メモリから命令をフェッチし、随意的にそのフェッチされた命令に対しローカル・バッファリングを提供する命令フェッチ・ユニットと、フェッチされた命令を受信し、フェッチされてきた命令の種類を判定する命令復号ユニットと、命令を実行する命令実行ユニットと、を含む。実行には、メモリからレジスタ中にデータをロードするステップ、レジスタからメモリにデータを戻して格納するステップ、あるいは、復号ユニットの判定により、ある種の算術または論理演算を遂行するステップを含めることができる。一つの事例において、各ユニットはソフトウェア中に実装される。例えば、これらユニットによって遂行されるオペレーションは、エミュレータ・ソフトウェア内に一つ以上のサブルーチンとして実装される。

【０１３０】
さらに具体的には、あるメインフレームでは、設計されたマシン命令が、プログラマ（普通、近年では「Ｃ」プログラマ）によって、多くの場合、コンパイラ・アプリケーションを介して使用される。ストレージ媒体中に格納されたこれらの命令は、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）ＩＢＭ（ＩＢＭ社の登録商標）サーバで、あるいは他のアーキテクチャを実行するマシンで、ネイティブに実行することができる。これらの命令は、既存のおよび将来のＩＢＭ（ＩＢＭ社の登録商標）メインフレーム・サーバ中で、並びにＩＢＭ（ＩＢＭ社の登録商標）の他のマシン（例、Ｐｏｗｅｒ　Ｓｙｓｔｅｍｓ（ＩＢＭ社の商標）サーバ、およびＳｙｓｔｅｍ　ｘ（ＩＢＭ社の登録商標）サーバ）上でエミュレートすることができる。これらの命令は、ＩＢＭ（ＩＢＭ社の登録商標）、Ｉｎｔｅｌ（Ｒ）、ＡＭＤ（Ｒ）および他社によって製造されたハードウェアを使った、多種多様なマシン上でＬｉｎｕｘ（Ｒ）を走らせるマシンにおいて実行できる。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＢＭ社の登録商標）の下のハードウェア上での実行に加え、Ｌｉｎｕｘ（Ｒ）、並びに、ＴｕｒｂｏＨｅｒｃｕｌｅｓ（Ｒ）（ｗｗｗ．ｔｕｒｂｏｈｅｒｃｕｌｅｓ．ｃｏｍ／）、Ｈｅｒｃｕｌｅｓ（Ｒ）（ｗｗｗ．ｈｅｒｃｕｌｅｓ－３９０．ｏｒｇ／）またはＦＳＩ（Ｆｕｎｄａｍｅｎｔａｌ　Ｓｏｆｔｗａｒｅ，Ｉｎｃ）（Ｒ）（ｗｗｗ．ｆｕｎｓｏｆｔ．ｃｏｍ／）によるエミュレーションを用いるマシンを使うことができ、一般に、実行はエミュレーション・モードにおいて行われる。エミュレーション・モードでは、エミュレーション・ソフトウェアは、ネイティブ・プロセッサによって実行され、疑似プロセッサのアーキテクチャがエミュレートされる。

【０１３１】
ネイティブ・プロセッサは、通常、疑似プロセッサのエミュレーションを遂行するため、ファームウェアもしくはネイティブ・オペレーティング・システムを含む、エミュレーション・ソフトウェアを実行する。エミュレーション・ソフトウェアは、疑似プロセッサのアーキテクチャの命令をフェッチし実行する役割を有する。エミュレーション・ソフトウェアは、命令境界の経過を追うため疑似プログラムのカウンタを維持する。エミュレー

ション・ソフトウェアは、一度に一つ以上の疑似マシン命令をフェッチし、該一つ以上の疑似マシン命令を、ネイティブ・プロセッサによる実行のため、それらに対応するネイティブ・マシン命令のグループに変換することができる。

【０１３２】

これらの変換された命令は、より速い変換が達成可能なようにキャッシュできる。しかしながら、エミュレーション・ソフトウェアは、疑似プロセッサに対して書かれたオペレーティング・システムおよびアプリケーションが正確に作動することを確実にするため、疑似プロセッサのアーキテクチャのアーキテクチャ・ルールを維持することになる。さらに、エミュレーション・ソフトウェアは、以下に限らないが、制御レジスタ、汎用レジスタ、浮動小数点レジスタ、例えばセグメント・テーブルおよびページ・テーブルを包含する動的アドレス変換機能、割り込みメカニズム、コンテキスト切り替えメカニズム、時刻（ＴＯＤ：Ｔｉｍｅ　ｏｆ　Ｄａｙ）機構、およびＩ／Ｏサブシステムに対し設計されたインタフェースを含め、疑似プロセッサのアーキテクチャによって識別されるリソースを提供し、疑似プロセッサ上で実行するよう設計されたオペレーティング・システムまたはアプリケーション・プログラムが、エミュレーション・ソフトウェアを有するネイティブ・プロセッサ上で実行できるようにする。

【０１３３】

エミュレートされる特定の命令は復号され、個々の命令の機能を遂行するためにサブルーチンが呼び出される。本好適な実施形態の説明を理解すれば当業者には自明なように、疑似プロセッサの機能をエミュレートするエミュレーション・ソフトウェアの機能が、例えば、「Ｃ」サブルーチンまたはドライバに、あるいは特定のハードウェアに対しドライバを提供する他の方法の中に実装される。以下に限らないが、米国特許証第５，５５１，０１３号、Ｂｅａｕｓｏｌｅｉｌらの発明の名称「Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒ　ｆｏｒ　Ｈａｒｄｗａｒｅ　Ｅｍｕｌａｔｉｏｎ」；米国特許証第６，００９，２６１号、Ｓｃａｌｚｉらの発明の名称「Ｐｒｅｐｒｏｃｅｓｓｉｎｇ　ｏｆ　Ｓｔｏｒｅｄ　Ｔａｒｇｅｔ　Ｒｏｕｔｉｎｅｓ　ｆｏｒ　Ｅｍｕｌａｔｉｎｇ　Ｉｎｃｏｍｐａｔｉｂｌｅ　Ｉｎｓｔｒｕｃｔｉｏｎｓ　ｏｎ　ａ　Ｔａｒｇｅｔ　Ｐｒｏｃｅｓｓｏｒ」；、米国特許証第５，５７４，８７３号、Ｄａｖｉｄｉａｎらの発明の名称「Ｄｅｃｏｄｉｎｇ　Ｇｕｅｓｔ　Ｉｎｓｔｒｕｃｔｉｏｎ　ｔｏ　Ｄｉｒｅｃｔｌｙ　Ａｃｃｅｓｓ　Ｅｍｕｌａｔｉｏｎ　Ｒｏｕｔｉｎｅｓ　ｔｈａｔ　Ｅｍｕｌａｔｅ　ｔｈｅ　Ｇｕｅｓｔ　Ｉｎｓｔｒｕｃｔｉｏｎｓ」；米国特許証第６，３０８，２５５号、Ｇｏｒｉｓｈｅｋらの発明の名称「Ｓｙｍｍｅｔｒｉｃａｌ　Ｍｕｌｔｉｐｒｏｃｅｓｓｉｎｇ　Ｂｕｓ　ａｎｄ　Ｃｈｉｐｓｅｔ　Ｕｓｅｄ　ｆｏｒ　Ｃｏｐｒｏｃｅｓｓｏｒ　Ｓｕｐｐｏｒｔ　Ａｌｌｏｗｉｎｇ　Ｎｏｎ－Ｎａｔｉｖｅ　Ｃｏｄｅ　ｔｏ　Ｒｕｎ　ｉｎ　ａ　Ｓｙｓｔｅｍ」；米国特許証第６，４６３，５８２号、Ｌｅｔｈｉｎらの発明の名称「Ｄｙｎａｍｉｃ　Ｏｐｔｉｍｉｚｉｎｇ　Ｏｂｊｅｃｔ　Ｃｏｄｅ　Ｔｒａｎｓｌａｔｏｒ　ｆｏｒ　Ａｒｃｈｉｔｅｃｔｕｒｅ　Ｅｍｕｌａｔｉｏｎ　ａｎｄ　Ｄｙｎａｍｉｃ　Ｏｐｔｉｍｉｚｉｎｇ　Ｏｂｊｅｃｔ　Ｃｏｄｅ　Ｔｒａｎｓｌａｔｉｏｎ　Ｍｅｔｈｏｄ」；米国特許証第５，７９０，８２５号、Ｅｒｉｃ　Ｔｒａｕｔの発明の名称「Ｍｅｔｈｏｄ　ｆｏｒ　Ｅｍｕｌａｔｉｎｇ　Ｇｕｅｓｔ　Ｉｎｓｔｒｕｃｔｉｏｎｓ　ｏｎ　ａ　Ｈｏｓｔ　Ｃｏｍｐｕｔｅｒ　Ｔｈｒｏｕｇｈ　Ｄｙｎａｍｉｃ　Ｒｅｃｏｍｐｉｌａｔｉｏｎ　ｏｆ　Ｈｏｓｔ　Ｉｎｓｔｒｕｃｔｉｏｎｓ」、および他の多くのものを含め、さまざまなソフトウェアおよびハードウェア・エミュレーションの特許が、当業者が利用可能なターゲット・マシンに対する異なるマシンのために設計された命令フォーマットのエミュレーションを実現するための各種の既知の方法を解説している。

【０１３４】

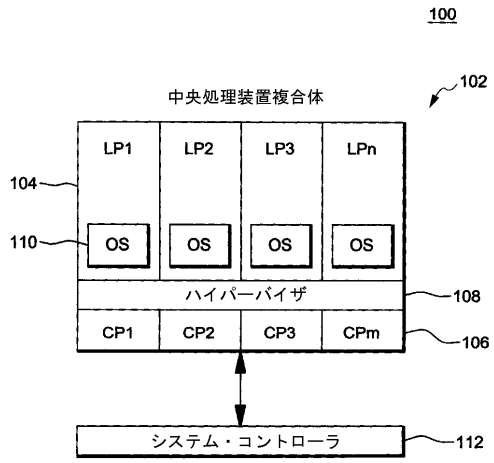
本明細書で使用した用語は、特定の実施形態を説明する目的のためだけのものであり、本発明を限定することは意図されていない。本明細書で用いられる、単数形「ある（ａ、ａｎ）」、および「該（ｔｈｅ）」は、文脈上明確に別途に示されていなければ、複数形も同じように含むことが意図されている。さらに、本明細書で用いられる「含む（ｃｏｍ

ｐｒｉｓｅ）」もしくは「含んでいる（ｃｏｍｐｒｉｓｉｎｇ）」またはその両方は、述べられた機能、完全体、ステップ、オペレーション、エレメント、またはコンポーネント、あるいはこれらの複数の存在を特定するが、一つ以上の他の機能、完全体、ステップ、オペレーション、エレメント、コンポーネント、もしくはこれらの群、または上記の組み合わせの存在あるいは追加を排除するものでないことがさらに理解されよう。

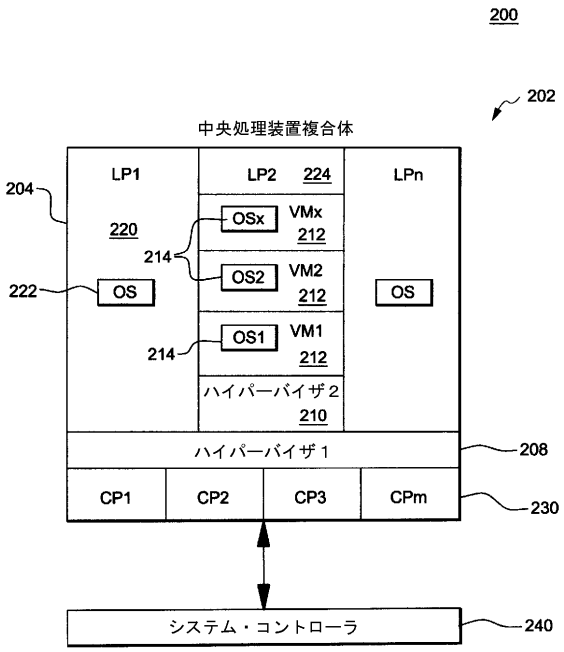
【０１３５】

添付の請求項中のミーンズ・プラス・ファンクションまたはステップ・プラス・ファンクションの要素全ての、対応する構造、材料、動作および均等物は、存在する場合、具体的に請求された他の請求要素と組み合わせて機能を遂行するための、一切の構造、材料または動作を包含することが意図されている。本発明の記述は、例示および説明の目的で提示されたもので、網羅的であることも、または本発明を開示した形態に限定することも意図されていない。当業者には、本発明の範囲および精神から逸脱することのない多くの修改および変形が明白であろう。本実施形態は、本発明の原理および実際的な応用を最善に説明し、他の当業者が、意図する特定の用途に適したさまざまな修改を加えたさまざまな実施形態に関して、本発明を理解できるようにするため、選択し説明されたものである。

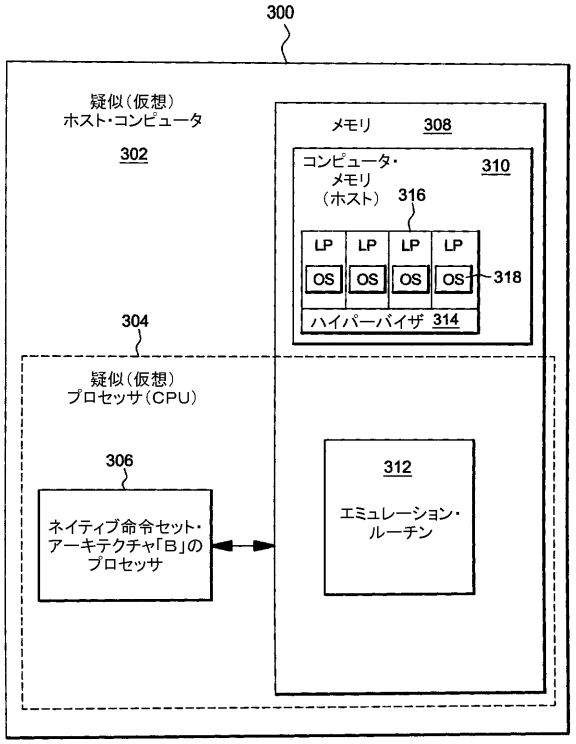
【図１】



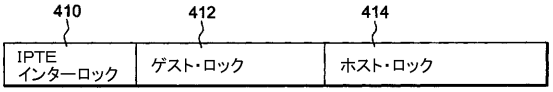
【図２】



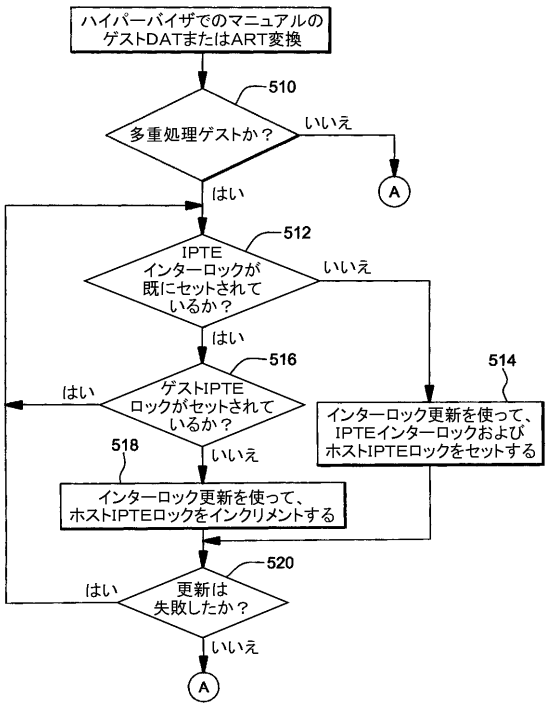
【図３】


300
疑似（仮想）
ホスト・コンピュータ
302
メモリ 308
コンピュータ・
メモリ
（ホスト） 316
310
LP LP LP LP
OS OS OS OS
318
ハイパーバイザ 314
304
疑似（仮想）
プロセッサ（CPU）
306
ネイティブ命令セット・
アーキテクチャ「B」の
プロセッサ
312
エミュレーション・
ルーチン

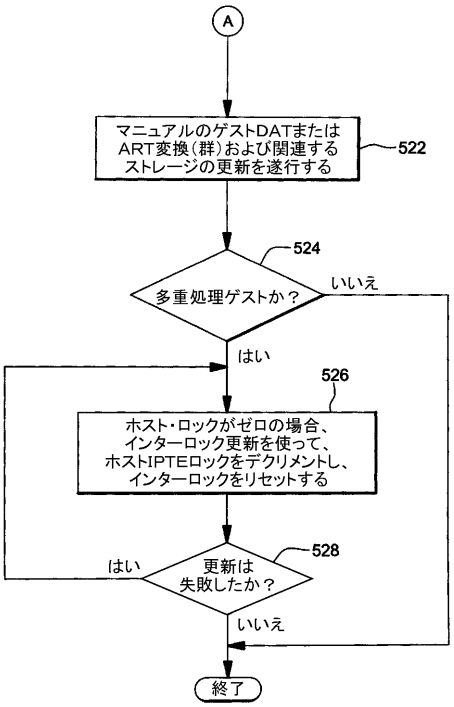
【図４】


410
412
414
IPTE
インターロック
ゲスト・ロック
ホスト・ロック

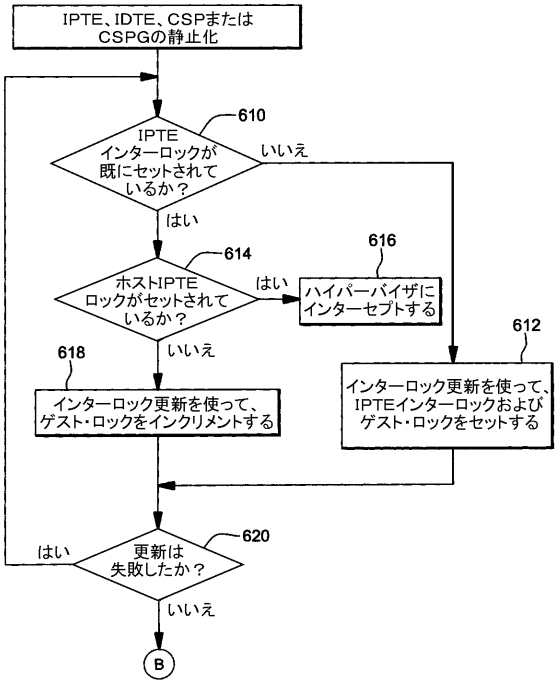
【図５】


ハイパーバイザでのマニュアルの
ゲストDATまたはART変換
510
多重処理ゲストか？
いいえ
はい
A
512
IPTE
インターロックが
既にセットされて
いるか？
いいえ
はい
516
ゲストIPTE
ロックがセットされて
いるか？
はい
514
インターロック更新を使って、
IPTEインターロックおよび
ホストIPTEロックをセットする
518
いいえ
インターロック更新を使って、
ホストIPTEロックをインクリメントする
520
更新は
失敗したか？
はい
いいえ
A

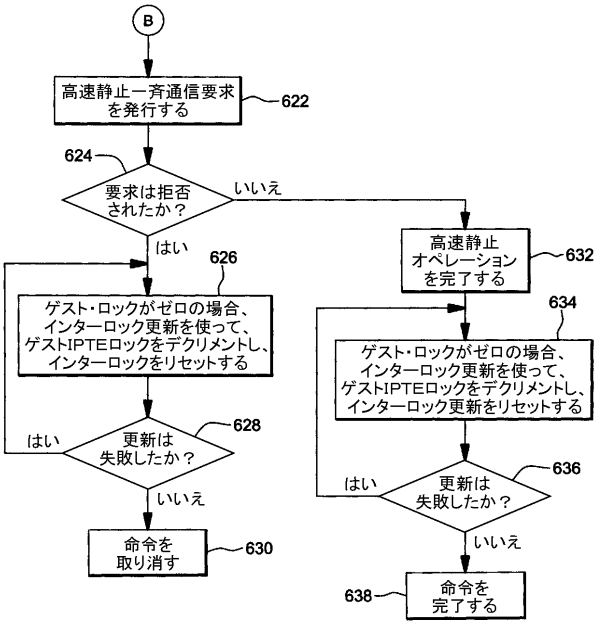
【図６】


A
マニュアルのゲストDATまたは
ART変換（群）および関連する
ストレージの更新を遂行する
522
524
多重処理ゲストか？
いいえ
はい
526
ホスト・ロックがゼロの場合、
インターロック更新を使って、
ホストIPTEロックをデクリメントし、
インターロックをリセットする
528
更新は
失敗したか？
はい
いいえ
終了

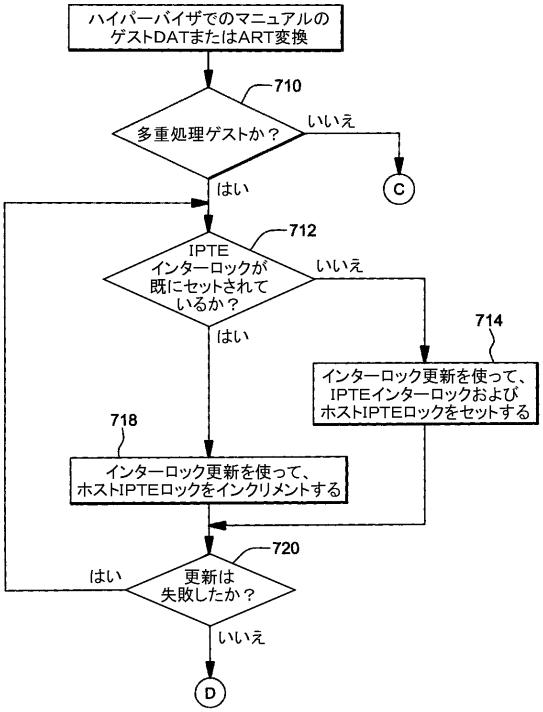
【図７】


IPTE、IDTE、CSPまたは
CSPGの静止化
610
IPTE
インターロックが
既にセットされて
いるか？
いいえ
はい
614
ホストIPTE
ロックがセットされて
いるか？
はい
616
ハイパーバイザに
インターセプトする
612
618
いいえ
インターロック更新を使って、
ゲスト・ロックをインクリメントする
インターロック更新を使って、
IPTEインターロックおよび
ゲスト・ロックをセットする
620
更新は
失敗したか？
はい
いいえ
B

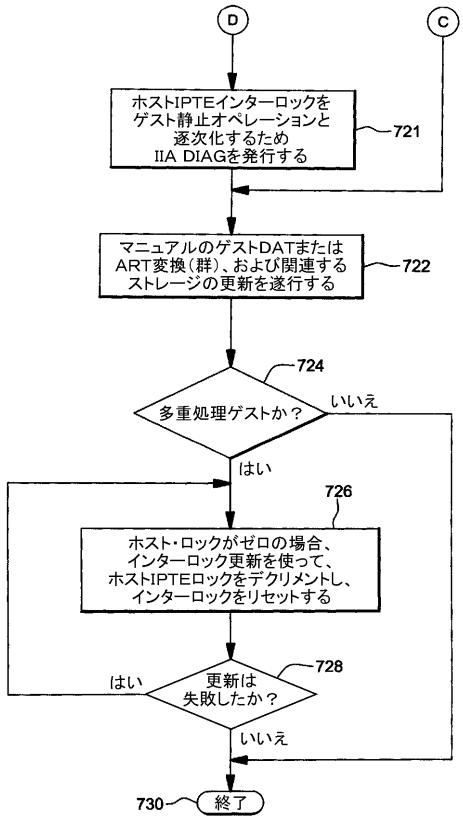
【図８】


B
高速静止一斉通信要求を発行する
622
624
要求は拒否されたか？
いいえ
はい
626
ゲスト・ロックがゼロの場合、インターロック更新を使って、ゲストIPTEロックをデクリメントし、インターロックをリセットする
628
更新は失敗したか？
はい
いいえ
命令を取り消す
630
高速静止オペレーションを完了する
632
634
ゲスト・ロックがゼロの場合、インターロック更新を使って、ゲストIPTEロックをデクリメントし、インターロック更新をリセットする
636
更新は失敗したか？
はい
いいえ
638
命令を完了する

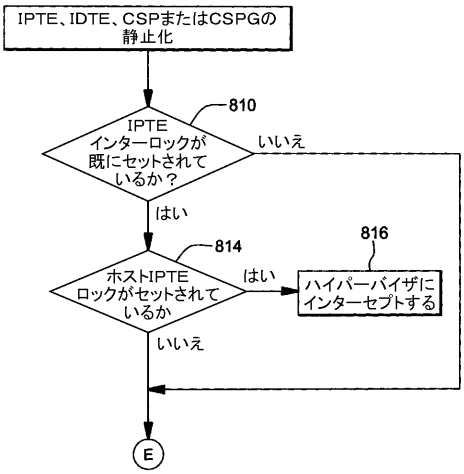
【図９】


ハイパーバイザでのマニュアルのゲストDATまたはART変換
710
多重処理ゲストか？
いいえ
C
はい
712
IPTEインターロックが既にセットされているか？
いいえ
はい
714
インターロック更新を使って、IPTEインターロックおよびホストIPTEロックをセットする
718
インターロック更新を使って、ホストIPTEロックをインクリメントする
720
更新は失敗したか？
はい
いいえ
D

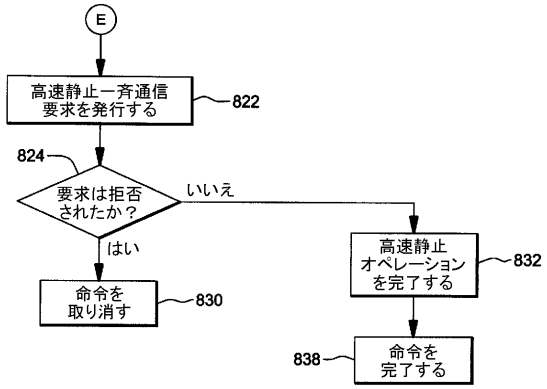
【図１０】


D
C
ホストIPTEインターロックをゲスト静止オペレーションと逐次化するためIIA DIAGを発行する
721
マニュアルのゲストDATまたはART変換（群）、および関連するストレージの更新を逐行する
722
724
多重処理ゲストか？
いいえ
はい
726
ホスト・ロックがゼロの場合、インターロック更新を使って、ホストIPTEロックをデクリメントし、インターロックをリセットする
728
更新は失敗したか？
はい
いいえ
730
終了

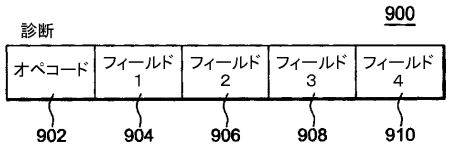
【図１１】


IPTE、IDTE、CSPまたはCSPGの静止化
810
IPTEインターロックが既にセットされているか？
いいえ
はい
814
ホストIPTEロックがセットされているか
はい
816
ハイパーバイザにインターセプトする
いいえ
E

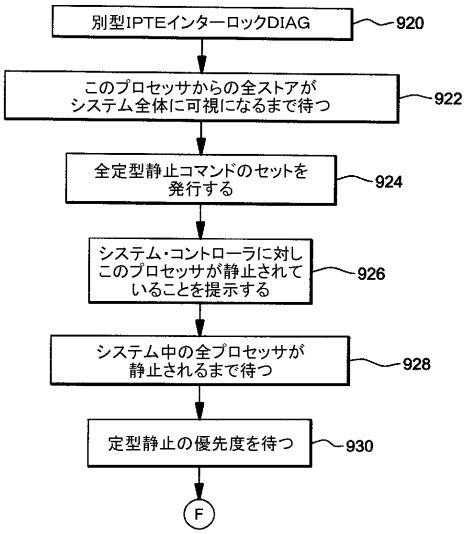
【図１２】



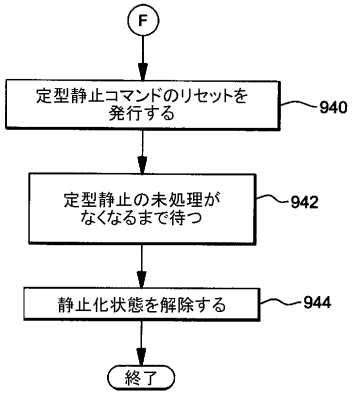
【図１３】



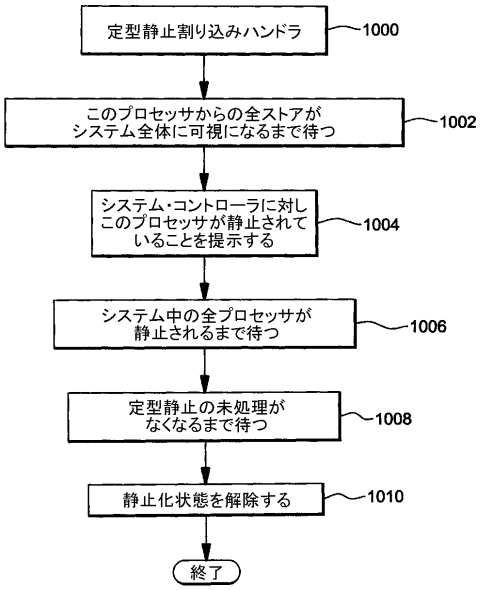
【図１４】



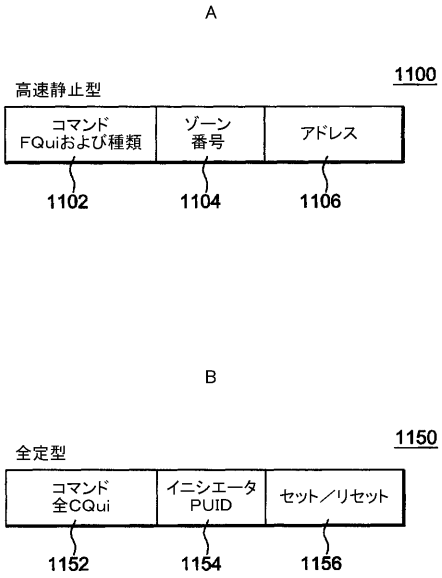
【図１５】



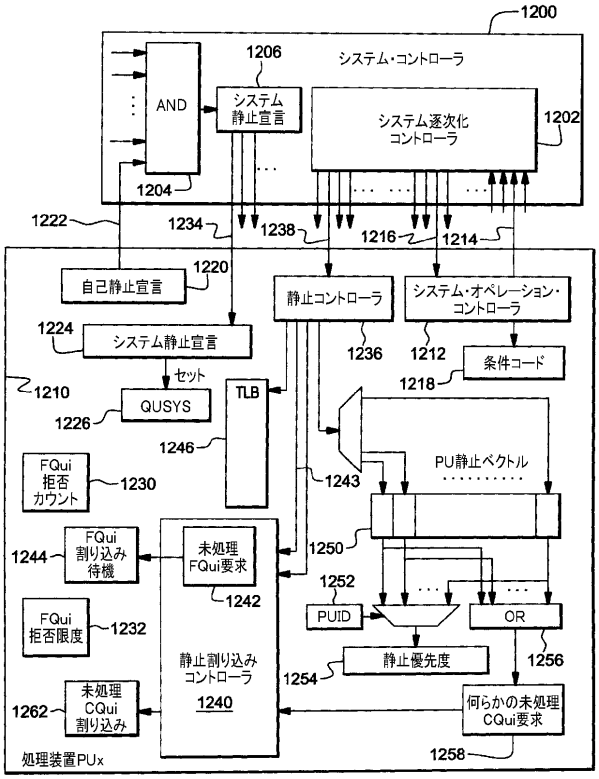
【図１６】



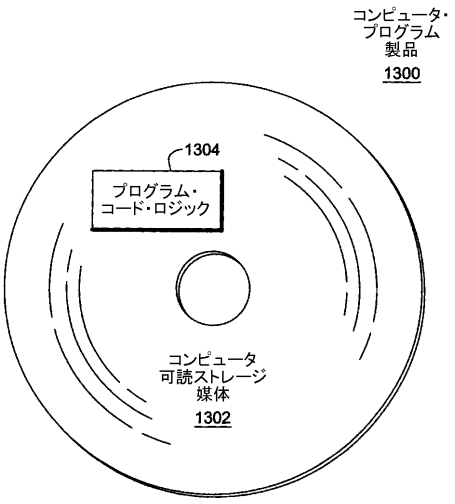
【図１７】



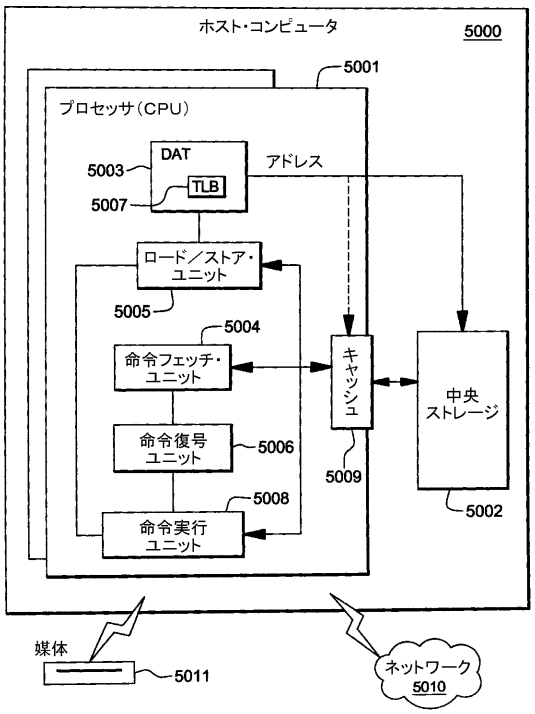
【図１８】



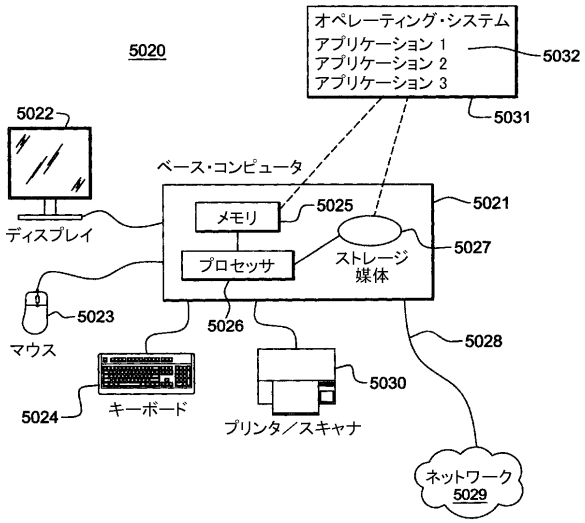
【図１９】



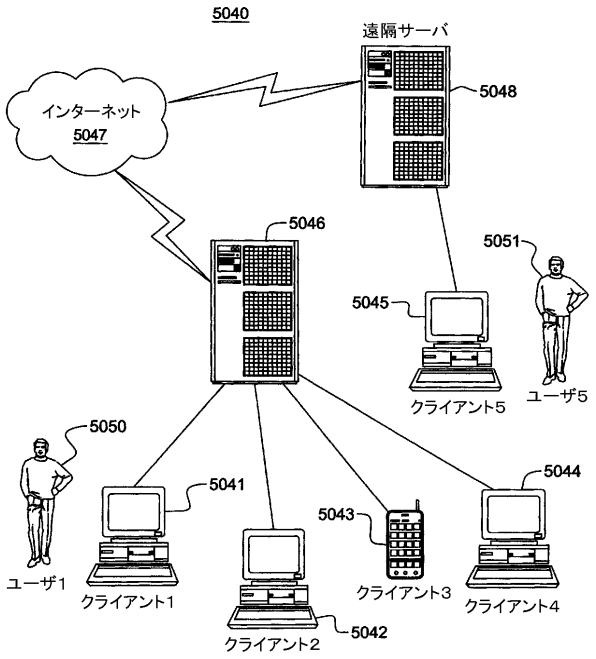
【図２０】



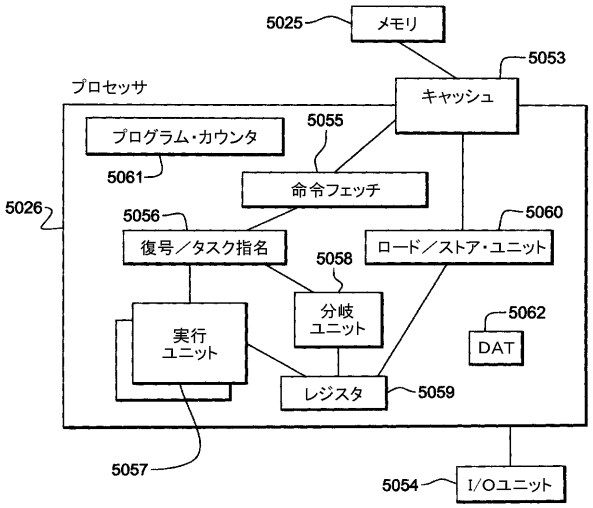
【図２１】



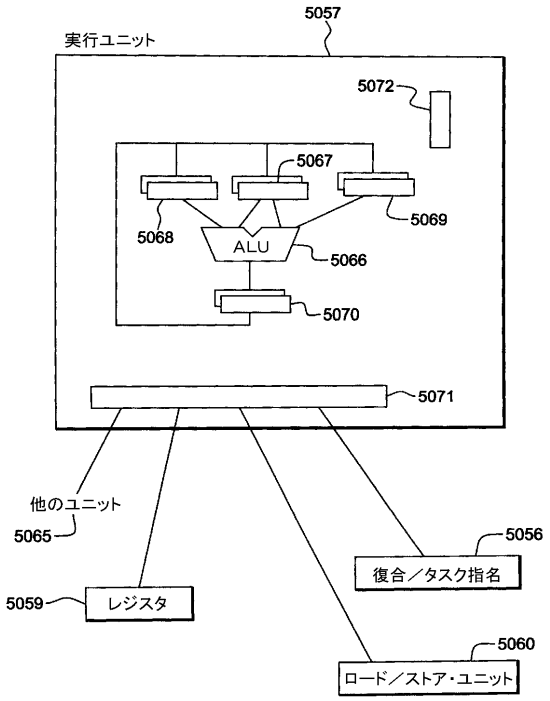
【図２２】



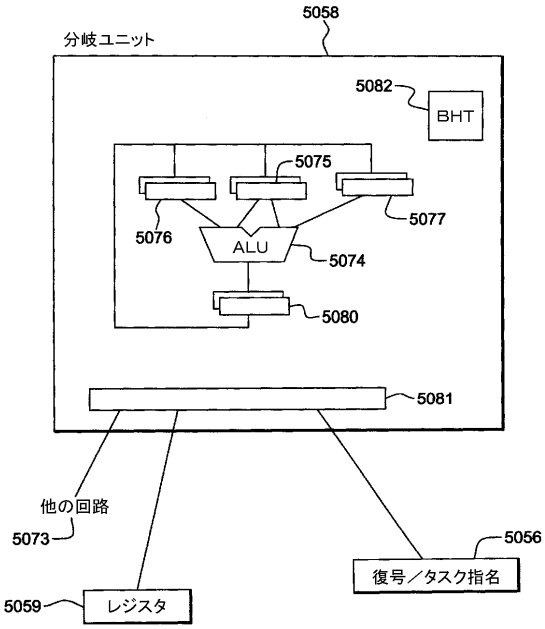
【図２３】



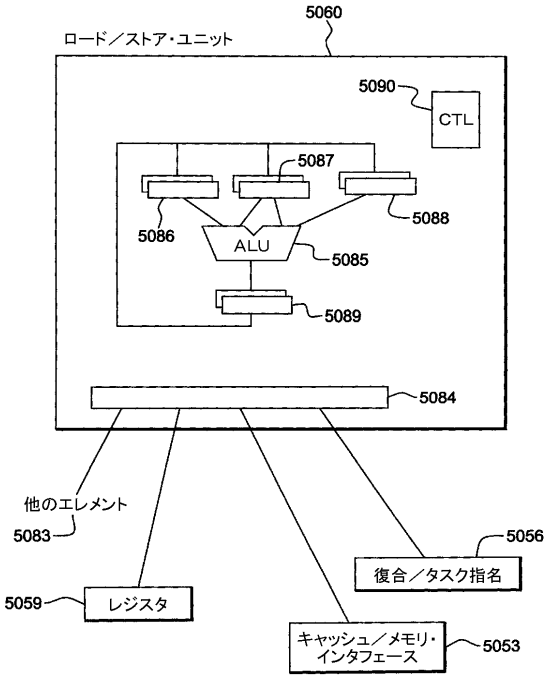
【図２４】



【図２５】



【図２６】



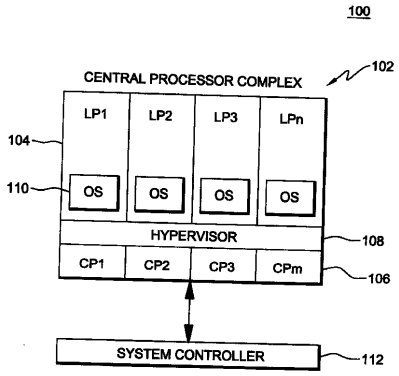


FIG. 1

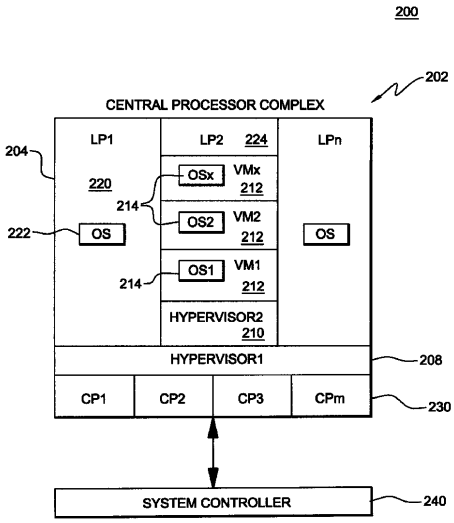


FIG. 2

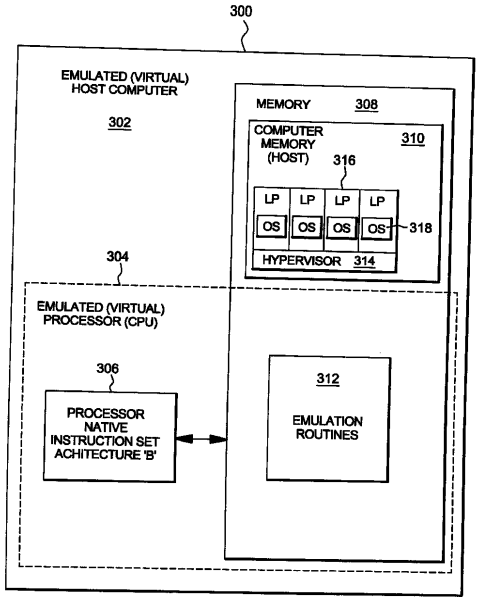


FIG. 3

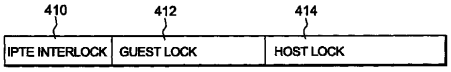


FIG. 4

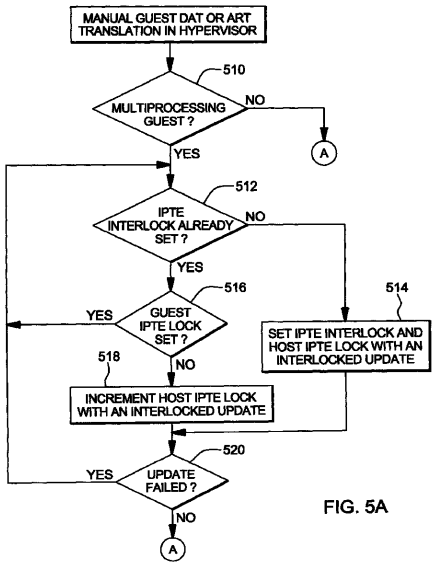


FIG. 5A

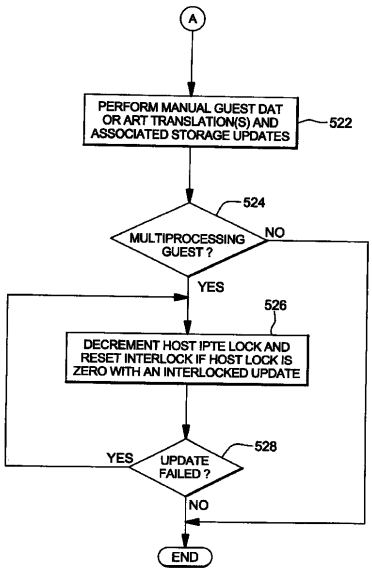


FIG. 5B

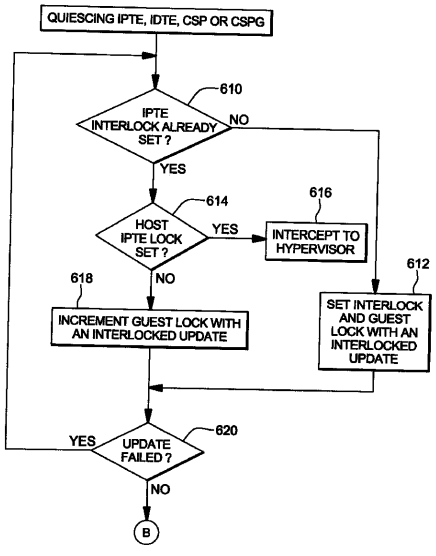


FIG. 6A

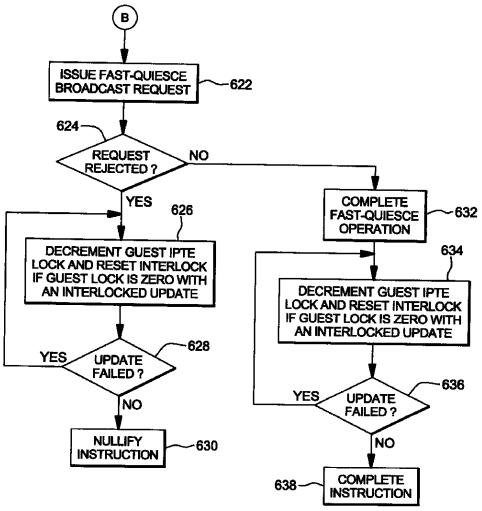


FIG. 6B

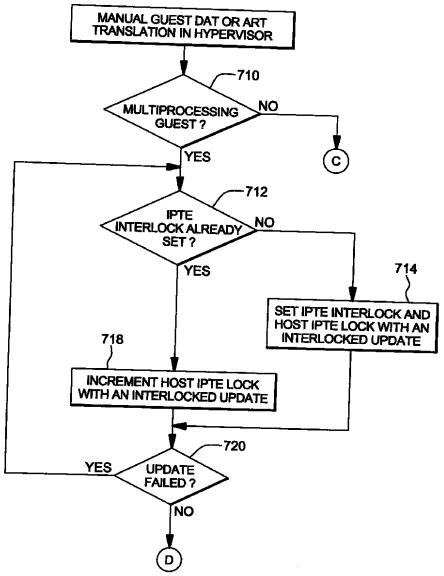


FIG. 7A

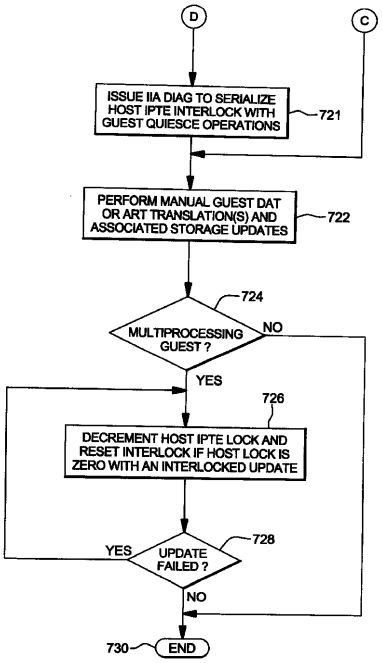


FIG. 7B

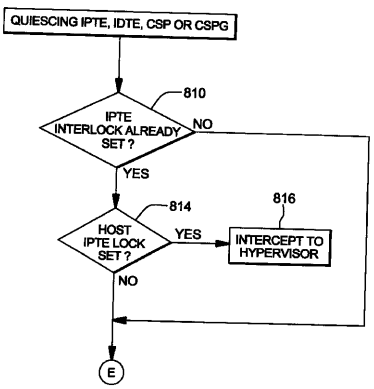


FIG. 8A

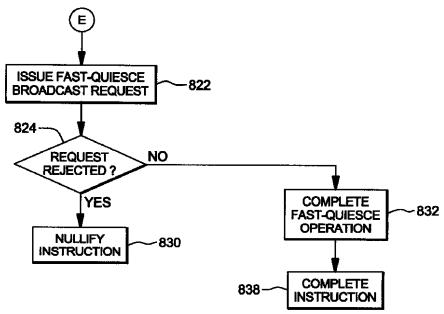


FIG. 8B

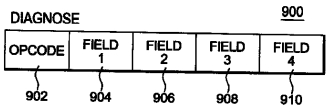


FIG. 9A

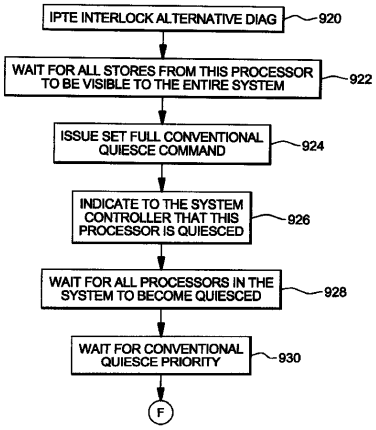


FIG. 9B

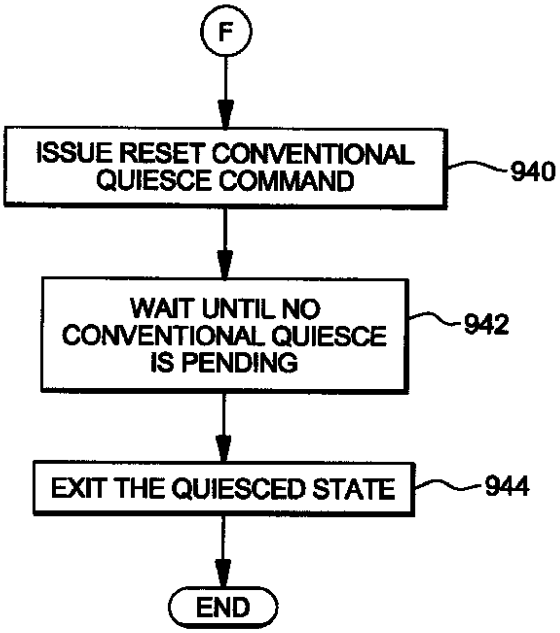


FIG. 9C

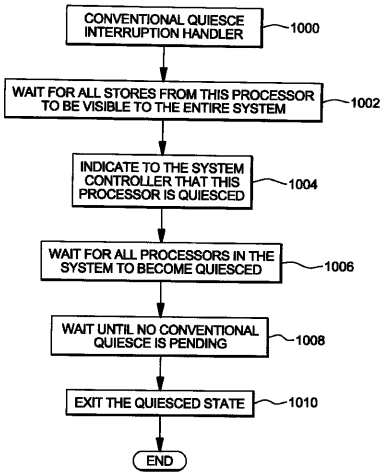


FIG. 10

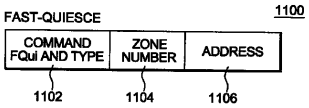


FIG. 11A

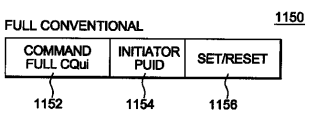


FIG. 11B

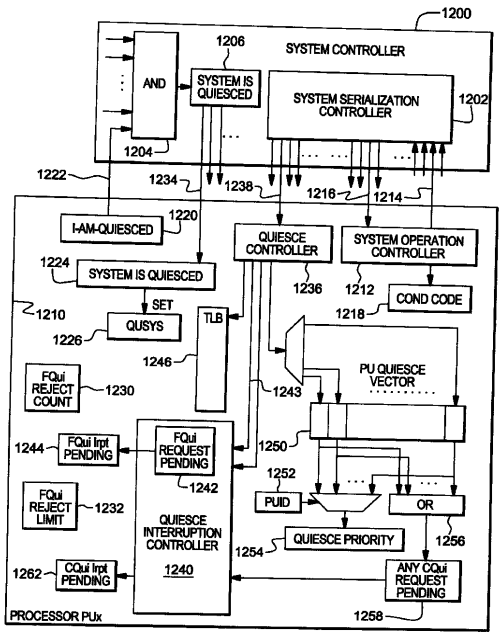


FIG. 12

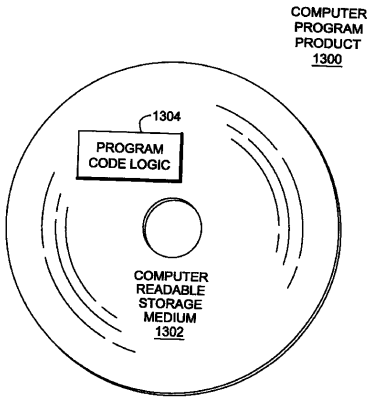


FIG. 13

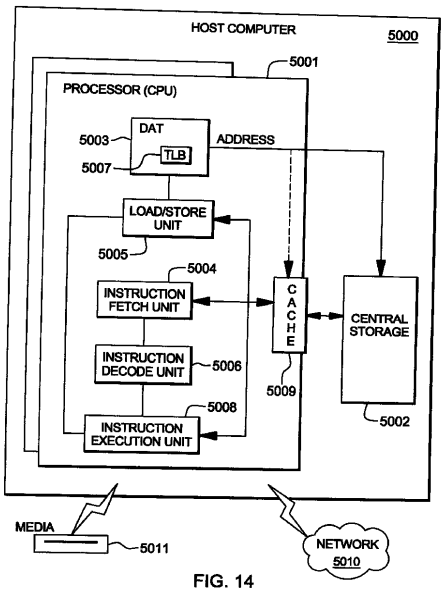


FIG. 14

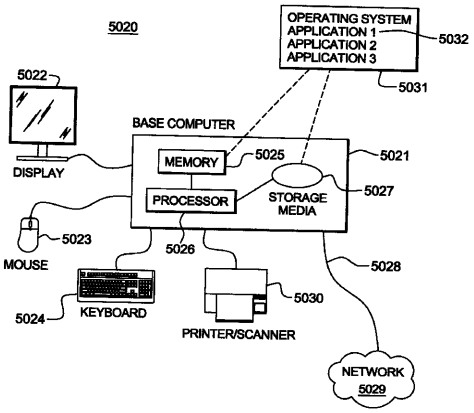


FIG. 15

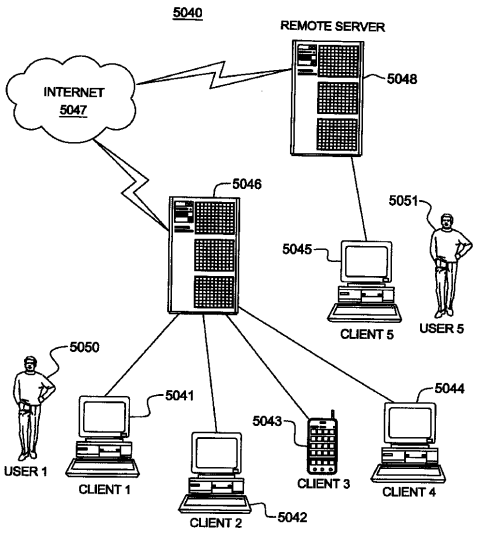


FIG. 16

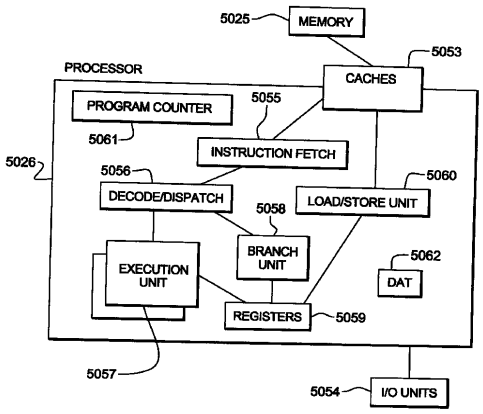


FIG. 17

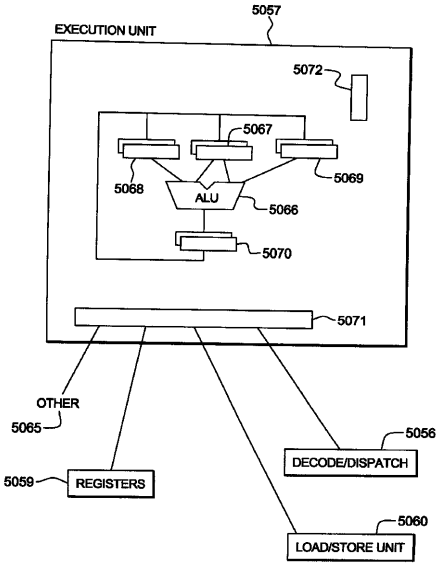


FIG. 18A

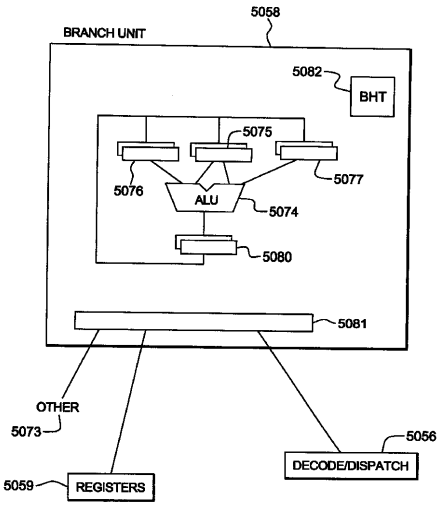


FIG. 18B

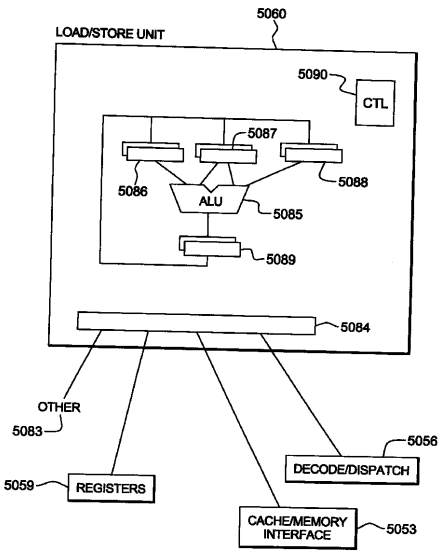


FIG. 18C

【手続補正書】
【提出日】平成24年12月26日(2012.12.26)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
コンピューティング環境の処理を逐次化するための診断命令を実行する方法であって、前記方法は、
実行のためマシン命令を取得するステップであって、前記マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、前記マシン命令は、
診断命令を識別するオペコード・フィールド、
前記診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および
第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために前記診断命令が用いられることを識別するのに使われるオペレーション・コード拡張を得るための、前記第二位置を識別するベース・フィールド、
を含む、前記取得するステップと、
前記オペレーション・コード拡張によって示された前記マシン命令を実行するステップであって、
前記サブコードが所定の値であるのに応じて、
前記コンピューティング環境のプロセッサの静止化を開始するステップ、
前記プロセッサが静止されていることを判定するステップ、および
前記プロセッサが静止されるのに応じて、前記診断命令の実行を完了するステップ、
を含む、前記実行するステップと、
を含む前記方法。
【請求項２】
前記マシン命令は、システム・コール領域のアドレスが提供される第三位置を識別する制御領域フィールドをさらに含む、請求項１に記載の方法。
【請求項３】
前記サブコードが前記所定の値であるのに応じて、前記システム・コール領域の前記アドレスは、変換を実施する対象のゲストを識別し、前記方法は、前記ゲストに対する変換が実施されるのに応じて、前記マシン命令を発行するステップをさらに含む、請求項２に記載の方法。
【請求項４】
前記方法は、
前記ゲストが多重処理ゲストかどうかを判定するステップと、
前記ゲストが多重処理ゲストであるのに応じて、インターロックがセットされているかどうかを判定するステップと、
前記インターロックがセットされていないのに応じて、前記インターロックおよびホスト・ロックをセットするステップと、
前記インターロックがセットされているのに応じて、前記ホスト・ロックを更新するステップと、
前記ホスト・ロックが成功裏にセットまたは更新されるのに応じて、前記マシン命令を発行するステップと、
をさらに含む、請求項３に記載の方法。
【請求項５】
前記ホスト・ロックは、ゲスト・ロックにかかわりなくセットされる、請求項４に記載

の方法。
【請求項６】
前記方法は、ホスト・ロックをセットするステップと、それに応じて前記マシン命令を発行するステップとをさらに含む、請求項１に記載の方法。
【請求項７】
前記マシン命令の前記実行は、前記ホスト・ロックを、前記コンピューティング環境のプロセッサ群に可視にするステップを含む、請求項６に記載の方法。
【請求項８】
前記診断命令の実行を完了するステップは、前記静止状態を出るステップを含む、請求項１に記載の方法。
【請求項９】
前記方法は、
ゲストによって、ロックを用いる静止型の命令を発行するステップであって、前記静止型命令は、このゲストに対するホストによって遂行される変換処理を使って逐次化されることになる、前記発行するステップと、
ホスト・ロックがセットされているかどうかを判定するステップと、
前記ホスト・ロックがセットされていないのに応じて、ゲスト・ロックの使用なしに、静止型命令を発行するステップと、
をさらに含む、請求項１に記載の方法。
【請求項１０】
前記ホスト・ロックがセットされていて、前記マシン命令の実行が完了されていないのが示されているのに応じて、前記ホストにインターセプトする、請求項９に記載の方法。
【請求項１１】
前記方法は、
第一エンティティと第二エンティティとによって共用されるインターロックをセットするステップと、
前記インターロックのセットに応じて、第一エンティティ・ロックを更新するステップと、
前記第一エンティティ・ロックを成功裏に更新するのに応じて、前記マシン命令を発行するステップであって、前記マシン命令は、第二エンティティ・ロックのセッティングなしで、前記第一エンティティの処理と前記第二エンティティの処理との間で逐次化を遂行する、前記発行するステップと、
をさらに含む、請求項１に記載の方法。
【請求項１２】
コンピューティング環境の処理を逐次化する診断命令を実行するためのコンピュータ・システムであって、前記コンピュータ・システムは
メモリと、
前記メモリと通信するプロセッサと、
実行のためのマシン命令を取得するオブテイナであって、前記マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、前記マシン命令が、
診断命令を識別するオペコード・フィールド、
前記診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および
第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために前記診断命令が用いられることを識別するのに使われる、オペレーション・コード拡張を得るための前記第二位置を識別するベース・フィールド、
を含む、前記オブテイナと、
前記オペレーション・コード拡張によって示された前記マシン命令を実行するためのエグゼキュタであって、前記サブコードが所定の値であるのに応じて、

前記コンピューティング環境のプロセッサの静止化を開始するためのイニシエータ、
前記プロセッサが静止されているのを判定するためのデターミナ、および
前記プロセッサが静止されているのに応じて、前記診断命令の実行を完了するための実行コンプリータ、
を含む手段を包含する、前記エグゼキュタと、
を含む、前記コンピュータ・システム。
【請求項１３】
コンピュータ・システムにロードされ、その上で実行されるとき、前記コンピュータ・システムに請求項１～１１のいずれかに記載の方法の全ステップを遂行させる、コンピュータ・プログラム。
【請求項１４】
コンピューティング環境の処理を逐次化する診断命令を実行するためのコンピュータ・システムであって、前記コンピュータ・システムは、
メモリと、
前記メモリと通信するプロセッサと、
を含み、
前記コンピュータ・システムは方法を遂行するよう構成され、前記方法は、
実行のためマシン命令を取得するステップであって、前記マシン命令はコンピュータ・アーキテクチャに従ったコンピュータ実行のために定義されており、前記マシン命令が、
診断命令を識別するオペコード・フィールド、
前記診断命令の実行に使われるサブコードを内容に含む第一位置を識別するサブコード・フィールド、および
第二位置の内容を変位フィールドの内容に加算して、処理を逐次化するために前記診断命令が用いられることを識別するのに使われる、オペレーション・コード拡張を得るための前記第二位置を識別するベース・フィールド、
を含む、前記取得するステップと、
前記オペレーション・コード拡張によって示された前記マシン命令を実行するステップであって、
前記サブコードが所定の値であるのに応じて、
前記コンピューティング環境のプロセッサの静止化を開始するステップ、
前記プロセッサが静止されていることを判定するステップ、および
前記プロセッサが静止されているのに応じて、前記診断命令の実行を完了するステップ、
を含む、前記実行するステップと、
を含む、前記コンピュータ・システム。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/067038

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/455 G06F9/46 G06F9/30 G06F9/318
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2004/064618 A1 (FARRELL MARK S [US] ET AL) 1 April 2004 (2004-04-01) abstract paragraph [0012] - paragraph [0017]; figures 1-4b paragraph [0022] - paragraph [0092]; claims 1-24 -----	1-22
Y	"Z/Architecture Principles of Operation, Chapter 10", INTERNET CITATION, 1 April 2007 (2007-04-01), pages 10-1, XP002523173, Retrieved from the Internet: URL:http://publibz.boulder.ibm.com/epubs/pdf/a2278325.pdf [retrieved on 2009-04-08] pages 10-19 ----- -/--	1-22

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 May 2011	15/06/2011

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Wierzejewski, Piotr

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/067038

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 493 741 B1 (EMER JOEL S [US] ET AL) 10 December 2002 (2002-12-10) abstract column 2, line 35 - column 14, line 19; claims 1-60; figures 1-9 -----	1-22
A	PHILIP M WELLS ET AL: "Serializing instructions in system-intensive workloads: Amdahl's Law strikes again", HIGH PERFORMANCE COMPUTER ARCHITECTURE, 2008. HPCA 2008. IEEE 14TH INTERNATIONAL SYMPOSIUM ON, IEEE, PISCATAWAY, NJ, USA, 16 February 2008 (2008-02-16), pages 264-275, XP031353639, ISBN: 978-1-4244-2070-4 the whole document -----	1-22
A	US 2004/073905 A1 (EMER JOEL S [US] ET AL) 15 April 2004 (2004-04-15) abstract paragraph [0014] - paragraph [0033]; figures 1-9 paragraph [0044] - paragraph [0146]; claims 1-19 -----	1-22

2

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2010/067038

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004064618 A1	01-04-2004	NONE	
US 6493741 B1	10-12-2002	US 2003105944 A1	05-06-2003
US 2004073905 A1	15-04-2004	NONE	

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CZ,DE,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IS,JP,KE,KG,KM,KN,KP,KR,KZ,LA,LC,LK,LR,LS,LT,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PE,PG,PH,PL,PT,RO,RS,RU,SC,SD,SE,SG,SK,SL,SM,ST,SV,SY,TH,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,ZA,ZM,ZW

(72)発明者 エレル、リサ
アメリカ合衆国１２６０１－５４００ ニューヨーク州 ポキプシー サウスロード ２４５５
エムディー エー８５／ピー３１０

Ｆターム(参考) 5B033 BE00