(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5079246号
(P5079246)

(45)発行日　平成24年11月21日(2012.11.21)　(24)登録日　平成24年9月7日(2012.9.7)

(51)Int.Cl.　　F I
G06F 9/46 (2006.01)　　G06F 9/46 350
G06F 13/10 (2006.01)　　G06F 13/10 330C

請求項の数 12　(全 20 頁)

(21)出願番号　特願2006-67701(P2006-67701)
(22)出願日　平成18年3月13日(2006.3.13)
(65)公開番号　特開2006-252565(P2006-252565A)
(43)公開日　平成18年9月21日(2006.9.21)
審査請求日　平成21年2月16日(2009.2.16)
(31)優先権主張番号　11/078,141
(32)優先日　平成17年3月11日(2005.3.11)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　エリック　ピー．トラウト
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】仮想マシン環境におけるマルチレベルインターセプト処理のためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
仮想マシン環境内の区分に対するインターセプトを処理するためのシステムであって、前記仮想マシン環境はバーチャライザおよび少なくとも１つの区分を含み、前記システムは、
区分において発生した特定のイベントをインターセプトして、該インターセプトされたイベントを外部モニタに転送するバーチャライザと、
前記バーチャイラザから前記インターセプトされたイベントを受け取ると、前記区分のオペレーティングシステムによる仮想プロセッサの制御を一時停止させて、前記インターセプトされたイベントを実行し、該実行中のインターセプトされたイベントに対応するように前記区分の仮想プロセッサの状態を修正し、前記インターセプトされたイベントの実行が終了すると前記区分の前記オペレーティングシステムに制御を返す、外部モニタと
を備えたことを特徴とするシステム。
【請求項２】
前記バーチャライザは、前記インターセプトされたイベントを発生させた前記区分の前記オペレーティングシステムの要求に応じて、前記インターセプトされたイベントを前記外部モニタに転送せずに前記バーチャライザ自身で処理することを特徴とする請求項１に記載のシステム。
【請求項３】
前記インターセプトされたイベントは、読取りと、書込みと、ロック付き読取り／修正

／書込み論理演算ＡＮＤ、ＯＲまたはＸＯＲと、ロック付き読取り／修正／書込み算術演算ＡＤＤまたはＳＵＢと、ロック付き比較演算ＸＣＨＧおよびＣＭＰＸＣＨＧと、ＨＬＴ、ＰＡＵＳＥまたはＣＰＵＩＤ命令と、制御レジスタ（ＣＲ）、デバッグレジスタ（ＤＲ）およびモデル固有レジスタ（ＭＳＲ）へのアクセスと、特定の例外と、外部割込みと、Ｉ／Ｏポートへのアクセスと、致命的例外と、のうちのいずれかのイベントであることを特徴とする請求項１に記載のシステム。

【請求項４】

前記インターセプトされたイベントは、プロセッサインターセプトタイプとメモリインターセプトタイプのいずれかであり、前記プロセッサインターセプトタイプは、特定のプロセッサ命令、レジスタ、例外、または割込みに関連付けられるイベントであり、前記メモリインターセプトタイプは、特定のメモリ位置にアクセスするイベントであり、前記インターセプトされたイベントのタイプは、前記バーチャライザから独立に前記区分によって指定されることを特徴とする請求項１に記載のシステム。

【請求項５】

仮想マシン環境内の区分に対するインターセプトを処理する方法であって、前記仮想マシン環境は、バーチャライザおよび少なくとも１つの区分を備え、前記方法は、

前記バーチャライザが、区分において発生した特定のイベントをインターセプトすることと、

前記バーチャライザが、前記インターセプトされたイベントを特定の区分内で稼働している外部モニタに転送することと、

前記外部モニタが、前記区分のオペレーティングシステムによる仮想プロセッサの制御を一時停止させることと、

前記外部モニタが、前記インターセプトされたイベントを実行することと、

前記外部モニタが、実行中の前記インターセプトされたイベントに対応するように前記区分の前記仮想プロセッサの状態を修正することと、

前記外部モニタが、前記インターセプトされたイベントの実行が終了すると前記区分の前記オペレーティングシステムに制御を返すことと

を備えることを特徴とする方法。

【請求項６】

前記バーチャライザは、前記インターセプトされたイベントを発生させた前記区分の前記オペレーティングシステムの要求に応じて、前記インターセプトされたイベントを前記外部モニタに転送せずに前記バーチャライザ自身で処理することを特徴とする請求項５に記載の方法。

【請求項７】

前記インターセプトされたイベントは、読取りと、書込みと、ロック付き読取り／修正／書込み論理演算ＡＮＤ、ＯＲまたはＸＯＲと、ロック付き読取り／修正／書込み算術演算ＡＤＤまたはＳＵＢと、ロック付き比較演算ＸＣＨＧおよびＣＭＰＸＣＨＧと、ＨＬＴ、ＰＡＵＳＥまたはＣＰＵＩＤ命令と、制御レジスタ（ＣＲ）、デバッグレジスタ（ＤＲ）およびモデル固有レジスタ（ＭＳＲ）へのアクセスと、特定の例外と、外部割込みと、Ｉ／Ｏポートへのアクセスと、致命的例外と、のうちのいずれかのイベントであることを特徴とする請求項５に記載の方法。

【請求項８】

前記インターセプトされたイベントは、プロセッサインターセプトタイプとメモリインターセプトタイプのいずれかであり、前記プロセッサインターセプトタイプは、特定のプロセッサ命令、レジスタ、例外、または割込みに関連付けられるイベントであり、前記メモリインターセプトタイプは、特定のメモリ位置にアクセスするイベントであり、前記インターセプトされたイベントのタイプは、前記バーチャライザから独立に前記区分によって指定されることを特徴とする請求項５に記載の方法。

【請求項９】

仮想マシン環境内の区分に対するインターセプトを処理するためのコンピュータ実行可

能な命令を記録したコンピュータ読み取り可能な記録媒体であって、前記仮想マシン環境はバーチャライザおよび少なくとも１つの区分を備え、前記コンピュータ実行可能な命令は、

前記バーチャライザが、区分において発生した特定のイベントをインターセプトするための命令と、

前記バーチャライザが、前記インターセプトされたイベントを特定の区分内で稼働している外部モニタに転送するための命令と、

前記外部モニタが、前記区分のオペレーティングシステムによる仮想プロセッサの制御を一時停止するための命令と、

前記外部モニタが、前記インターセプトされたイベントを実行するための命令と、

前記外部モニタが、実行中の前記インターセプトされたイベントに対応するように前記区分の前記仮想プロセッサの状態を修正するための命令と、

前記外部モニタが、前記インターセプトされたイベントの実行が終了すると前記区分の前記オペレーティングシステムに制御を返すための命令と

を備えることを特徴とするコンピュータ読み取り可能な記録媒体。

【請求項１０】

前記コンピュータ実行可能な命令は、前記バーチャライザが、前記インターセプトされたイベントを発生させた前記区分の前記オペレーティングシステムの要求に応じて、前記インターセプトされたイベントを前記外部モニタに転送せずに、前記インターセプトされたイベントを前記バーチャライザ自身で処理するための命令をさらに備えることを特徴とする請求項９に記載のコンピュータ読み取り可能な記録媒体。

【請求項１１】

前記インターセプトされたイベントは、読取りと、書込みと、ロック付き読取り／修正／書込み論理演算ＡＮＤ、ＯＲまたはＸＯＲと、ロック付き読取り／修正／書込み算術演算ＡＤＤまたはＳＵＢと、ロック付き比較演算ＸＣＨＧおよびＣＭＰＸＣＨＧと、ＨＬＴ、ＰＡＵＳＥまたはＣＰＵＩＤ命令と、制御レジスタ（ＣＲ）、デバッグレジスタ（ＤＲ）およびモデル固有レジスタ（ＭＳＲ）へのアクセスと、特定の例外と、外部割込みと、Ｉ／Ｏポートへのアクセスと、致命的例外と、のうちのいずれかのイベントであることを特徴とする請求項９に記載のコンピュータ読み取り可能な記録媒体。

【請求項１２】

前記インターセプトされたイベントは、プロセッサインターセプトタイプとメモリインターセプトタイプのいずれかであり、前記プロセッサインターセプトタイプは、特定のプロセッサ命令、レジスタ、例外、または割込みに関連付けられるイベントであり、前記メモリインターセプトタイプは、特定のメモリ位置にアクセスするイベントであり、前記インターセプトされたイベントのタイプは、前記バーチャライザから独立に前記区分によって指定されることを特徴とする請求項９に記載のコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は一般に、仮想マシン（「プロセッサ仮想化」とも呼ばれる）の分野に関し、より詳細には、本発明は、マルチレベル外部仮想マシンモニタ（ＶＭＭ：ｖｉｒｔｕａｌ ｍａｃｈｉｎｅ ｍｏｎｉｔｏｒ）を対象とし、いくつかのインターセプト（ｉｎｔｅｒｃｅｐｔ）処理機能は、特定の区分（それぞれは仮想マシンすなわち「ＶＭ」のインスタンス）で稼動する外部モニタによって、これらの区分化された外部モニタを管理するベースレベルＶＭＭと共に実行される。

【背景技術】

【０００２】

コンピュータは、特定のシステム命令セットを実行するように設計された汎用中央処理装置（ＣＰＵ）を備える。類似するアーキテクチャまたは設計仕様を有する一群のプロセッサは、同じプロセッサファミリのメンバと考えることができる。現在のプロセッサファ

ミリの例には、アリゾナ州ＰｈｏｅｎｉｘにあるＭｏｔｏｒｏｌａ，Ｉｎｃ．製のＭｏｔｏｒｏｌａ　６８０Ｘ０プロセッサファミリ、カリフォルニア州ＳｕｎｎｙｖａｌｅにあるＩｎｔｅｌ　Ｃｏｒｐｏｒａｔｉｏｎ製のＩｎｔｅｌ　８０Ｘ８６プロセッサファミリ、および、Ｍｏｔｏｒｏｌａ製であり、カリフォルニア州ＣｕｐｅｒｔｉｎｏにあるＡｐｐｌｅ　Ｃｏｍｐｕｔｅｒ，Ｉｎｃ．製コンピュータで使用されるＰｏｗｅｒＰＣプロセッサファミリが含まれる。一群のプロセッサのアーキテクチャおよび設計に関する考慮事項が類似するためにこれらのプロセッサが同じファミリ中にある場合があっても、プロセッサは、それらのクロック速度およびその他の性能パラメータにより、ファミリ内で大きく異なることがある。

【０００３】

マイクロプロセッサの各ファミリは、そのプロセッサファミリに固有の命令を実行する。プロセッサまたはプロセッサファミリが実行することの可能な命令の総体的なセットは、そのプロセッサの命令セットと呼ばれる。例として、Ｉｎｔｅｌ　８０Ｘ８６プロセッサファミリによって使用される命令セットは、ＰｏｗｅｒＰＣプロセッサファミリによって使用される命令セットと互換性がない。Ｉｎｔｅｌ　８０Ｘ８６命令セットは、ＣＩＳＣ（Ｃｏｍｐｌｅｘ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｅｒ）フォーマットに基づく。Ｍｏｔｏｒｏｌａ　ＰｏｗｅｒＰＣ命令セットは、ＲＩＳＣ（Ｒｅｄｕｃｅｄ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｅｒ）フォーマットに基づく。ＣＩＳＣプロセッサは多数の命令を使用し、これらの命令のいくつかは、いくぶん複雑な機能を実行することが可能であるが、実行には、概して多くのクロックサイクルを必要とする。ＲＩＳＣプロセッサは、より少数の利用可能な命令を使用してより単純な機能セットを実行し、これらの機能は、ずっと高いレートで実行される。

【０００４】

コンピュータシステム間におけるプロセッサファミリの独自性は通常、コンピュータシステムのハードウェアアーキテクチャの、他の要素間でも非互換性をもたらす。Ｉｎｔｅｌ　８０Ｘ８６プロセッサファミリからのプロセッサを備えて製造されたコンピュータシステムは、ＰｏｗｅｒＰＣプロセッサファミリからのプロセッサを備えて製造されたコンピュータシステムのハードウェアアーキテクチャとは異なるハードウェアアーキテクチャを有することになる。プロセッサ命令セットと、コンピュータシステムのハードウェアアーキテクチャとの独自性のため、アプリケーションソフトウェアプログラムは通常、特定のオペレーティングシステムを稼動させる特定のコンピュータシステム上で実行されるように書かれる。

【０００５】

（プロセッサ仮想化）

コンピュータ製造業者は、自社の製品ラインに関連するマイクロプロセッサファミリ上で、少なめよりはむしろ多めのアプリケーションが稼動するようにすることによって、自社の市場シェアを最大限にしたいと望む。コンピュータシステム上で稼動可能なオペレーティングシステムおよびアプリケーションプログラムの数を拡大するために、ある技術分野が発展してきたが、この技術分野では、ある種類のＣＰＵ（ホストと呼ばれる）を有する所与のコンピュータは、関係ない種類のＣＰＵ（ゲストと呼ばれる）の命令をこのホストコンピュータがエミュレートできるようにするエミュレータプログラムを備えることになる。そのため、ホストコンピュータは、所与のゲスト命令に応答して１つまたは複数のホスト命令が呼び出されるようにするアプリケーションを実行することになる。したがって、ホストコンピュータは、それ自体のハードウェアアーキテクチャ用に設計されたソフトウェアと、関係ないハードウェアアーキテクチャを有するコンピュータ用に書かれたソフトウェアとの両方を稼動させることが可能である。より具体的な例として、例えばＡｐｐｌｅ　Ｃｏｍｐｕｔｅｒ製のコンピュータシステムが、ＰＣベースのコンピュータシステム用に書かれたオペレーティングシステムおよびプログラムを稼動させることができる。また、エミュレータプログラムを使用して、単一のＣＰＵ上で複数の非互換オペレーティングシステムを同時に動作させることが可能な場合もある。この構成では、各オペレー

ティングシステムが相互に互換しないにもかかわらず、エミュレータプログラムは、２つのオペレーティングシステムの一方をホストすることが可能であり、普通なら互換性がないはずのこれらのオペレーティングシステムが同じコンピュータシステム上で同時に稼動できるようにする。

【０００６】

ホストコンピュータシステム上でゲストコンピュータシステムがエミュレートされるとき、ゲストコンピュータシステムは「仮想マシン」と言われる。というのは、ゲストコンピュータシステムは、ホストコンピュータシステム中で、ある特定ハードウェアアーキテクチャの動作の純粋なソフトウェア表現として存在するだけだからである。エミュレータ、仮想マシン、およびプロセッサエミュレーションという用語は、コンピュータシステム全体のハードウェアアーキテクチャを模倣またはエミュレートする能力を指すために交換可能に使用されることがある。例として、カリフォルニア州Ｓａｎ　Ｍａｔｅｏにあ
るＣｏｎｎｅｃｔｉｘ　Ｃｏｒｐｏｒａｔｉｏｎによって生み出されたＶｉｒｔｕａｌ　ＰＣソフトウェアは、Ｉｎｔｅｌ　８０Ｘ８６　Ｐｅｎｔｉｕｍ（登録商標）プロセッサと様々なマザーボードコンポーネントおよびカードとを備えたコンピュータ全体をエミュレートする。これらのコンポーネントの動作は、ホストマシン上で稼動している仮想マシン中でエミュレートされる。ＰｏｗｅｒＰＣプロセッサを有するコンピュータシステムなどのホストコンピュータの、オペレーティングシステムソフトウェアおよびハードウェアアーキテクチャ上で実行されているエミュレータプログラムが、ゲストコンピュータシステム全体の動作を模倣する（ｍｉｍｉｃ）。

【０００７】

エミュレータプログラムは、ホストマシンのハードウェアアーキテクチャと、エミュレートされた環境内で稼動するソフトウェアから送られる命令との間のインターチェンジとして働く。このエミュレートされた環境は、仮想マシンモニタ（ＶＭＭ）によって生み出すことができ、ＶＭＭは、ハードウェアのすぐ上で稼動するソフトウェア層であって、ＶＭＭが仮想化しているハードウェアと同じインタフェースを公開することによってマシンのリソースすべてを仮想化するソフトウェア層である（これによりＶＭＭは、ＶＭＭよりも上で稼動するオペレーティングシステム層に認識されずに（ｕｎｎｏｔｉｃｅ）いることができる）。この構成では、ホストオペレーティングシステム（ＨＯＳ）とＶＭＭとが、同じ物理ハードウェア上で並行して稼動することができる。あるいは、エミュレータプログラムは、物理コンピュータハードウェアのすぐ上で稼動して別のハードウェア構成をエミュレートするＨＯＳ自体であってもよい。この実施形態の特定の実装形態では、ＨＯＳソフトウェアは特に、「ハイパーバイザ（ｈｙｐｅｒｖｉｓｏｒ）」の一実施形態を備えることができる。

【０００８】

ハイパーバイザは、ＨＯＳのカーネルレベルの近くに存在する制御プログラムであり、コンピュータシステムの１つまたは複数の物理プロセッサを含めたコンピュータシステムハードウェアを、ＨＯＳ以外の１つまたは複数の２次オペレーティングシステムが使用できるようにするべく動作する。ハイパーバイザは、２次オペレーティングシステムのための動作環境をエミュレートし、それにより２次オペレーティングシステムは、実際には別のハードウェアおよび／またはオペレーティングシステム環境で動作しておりＨＯＳがコンピュータシステムを論理制御しているであろうにもかかわらず、その通例のハードウェアおよび／またはオペレーティングシステム環境で動作しておりそれ自体がコンピュータシステムを論理制御していると認識する（ｂｅｌｉｅｖｅ）。これは重要なことである。というのは、多くのオペレーティングシステムは、コンピュータシステムのハードウェアを排他的に論理制御しているかのように動作しなければならないものとして機能するからである。したがって、複数のオペレーティングシステムが単一のコンピュータシステム上で同時に機能するためには、各オペレーティングシステムのハイパーバイザは、各オペレーティングシステムがコンピュータシステム全体の排他的制御を有するかのごとく機能するように、他のオペレーティングシステムの存在を覆い隠すよう機能しなければならない

。
【０００９】
簡単にするために、プロセッサ仮想化プログラム（限定されないがＶＭＭおよびハイパーバイザを含む）を、本明細書では「バーチャライザ（ｖｉｒｔｕａｌｉｚｅｒ）」と総称する。さらに、ハイパーバイザのコンテキスト（状況、態様）で本明細書に開示される本発明の態様はどれも、ＶＭＭおよびその他のバーチャライザに対しても等しく有効であり開示されるものとし、またその逆でもある。
【００１０】
（インターセプトおよびモニタ）
当業者には知られており理解されるように、バーチャライザ（例えばハイパーバイザやＶＭＭ）の主要な機能の１つは、ソフトウェアが区分（ＶＭの個々のインスタンス）中のゲストオペレーティングシステム上で実行されている間に発生したイベントをインターセプトすることである。このコンテキストで、イベントとは、区分の何らかのコンポーネントと、実際には区分の一部ではない物理的または仮想的な何らかのリソースとの間で発生する対話である。例えば、第１の区分で実行されているプログラムが、ある周辺デバイスにデータを送りたい場合があり、この区分のオペレーティングシステムは、このデバイスに対して排他的制御を有すると認識している。しかしこの場合、このプログラムが、その区分のゲストオペレーティングシステムを介してそのデータを送り、ゲストオペレーティングシステムが周辺デバイスと通信しようとすると、バーチャライザは、同じことをしようとする他の区分による他の試みと共に（すなわちデバイスは実際にはどの特定区分にも専用になっていない）この区分によるこのデバイスへのアクセスを処理するために、この通信をインターセプトする。これらの種類のイベントをインターセプトすることにより、バーチャライザは本質的に、現実には基礎的な物理リソースは実際にいくつかの仮想マシン区分間で共有または区分化されているときに、ゲストＯＳを騙して、（物理ハードウェア上で実行されているＯＳが通常そうするように）マシンのリソースすべてを所有していると思わせる。これに関して、バーチャライザは、このようなイベントをインターセプトすることと、これらが発生したときにインターセプトに応答することとの両方を担う。残念ながら、当業者にとっては容易に明らかな理由でより単純なバーチャライザが望まれるときに、各区分の外で動作する単一のバーチャライザにこの種の機能を集中化させる場合、バーチャライザを非常に複雑にする必要がある。
【発明の開示】
【発明が解決しようとする課題】
【００１１】
したがって、より単純でありながらインターセプト機能をなお提供するバーチャライザ－インターセプトモデルが、当技術分野で必要とされている。本発明は、かかる解決策のひとつを提供する。
【課題を解決するための手段】
【００１２】
本発明の様々な実施形態は、インターセプト関係の機能の大部分を（各区分の外に存在する）ベースレベルのバーチャライザから除去して、その代わりにこの機能の大部分を各区分に直接に組み込むように設計された、マルチレベルバーチャライザを対象とする。いくつかの実施形態では、区分内で稼動し特定のインターセプトイベントに応答する「外部モニタ」によって、いくつかのインターセプト処理機能が実行され、ベースレベルバーチャライザは、これらの外部モニタを各区分内にインストールし、その後、単一区分と区分間との両方のインターセプトイベントについて、外部モニタを管理する。この分散型のインターセプト処理手法により、複雑さのずっと低いバーチャライザが可能になり、インターセプト機能が各区分中にまで引き上げられ、各区分中では、各外部モニタが、その区分中の対応するゲストオペレーティングシステムのリソースを使用してインターセプトイベントを解決する。
【００１３】

前述の概要、ならびに好ましい実施形態に関する後続の詳細な記述は、添付の図面と共に読めばよりよく理解される。本発明を例示するために、図面には本発明の例示的な構造が示してある。しかし、本発明は、開示される特定の方法および手段に限定されない。

【発明を実施するための最良の形態】

【００１４】

本発明の主題について、法定の要件を満たすために具体的に述べる。しかし、この記述自体は、本特許の範囲を限定するものではない。そうではなく、発明者は、特許請求される本主題が他の方法でも実施されて、他の現在または将来の技術に関連する、本明細書に述べるステップに類似する種々のステップまたはステップの組合せを含めることができることを企図している。さらに、用語「ステップ」は、本明細書では、利用される方法の種々の要素を意味するのに使用される場合があるが、この用語は、個々のステップの順序が明示的に述べられていない限り、本明細書に開示される様々な複数のステップ内またはステップ間の任意の特定の順序を暗示するものと解釈されるべきではない。

【００１５】

（コンピュータ環境）

本発明の多くの実施形態は、コンピュータ上で実行することができる。図１および後続の考察は、本発明を実施することのできる適切なコンピューティング環境に関する簡単で一般的な記述を提供するものである。必須ではないが、本発明を、クライアントワークステーションやサーバなどのコンピュータによって実行される、プログラムモジュールなどのコンピュータ実行可能命令の一般的なコンテキストで述べる。一般に、プログラムモジュールは、特定のタスクを実行するか特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを含む。さらに、当業者なら理解するであろうが、本発明は、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベースのまたはプログラム可能な民生用電子機器、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータなどを含めた他の種類のコンピュータシステム構成でも実施することができる。本発明は分散コンピューティング環境で実施することもでき、その場合、通信ネットワークを介してリンクされたリモート処理デバイスによってタスクが実行される。分散コンピューティング環境では、プログラムモジュールは、ローカルとリモートとの両方のメモリ記憶デバイスに位置することができる。

【００１６】

図１に示すように、例示的な汎用コンピューティングシステムは、従来型のパーソナルコンピュータ２０などを備え、コンピュータ２０は、処理ユニット２１と、システムメモリ２２と、システムメモリを含む様々なシステムコンポーネントを処理ユニット２１に結合するシステムバス２３とを備える。システムバス２３は、様々なバスアーキテクチャのいずれかを使用した、メモリバスまたはメモリコントローラ、周辺バスおよびローカルバスを含む、いくつかの種類のバス構造のうちのいずれかとすることができる。システムメモリは、ＲＯＭ（Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）２４およびＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）２５を含む。ＲＯＭ２４には、起動中などにパーソナルコンピュータ２０内の要素間で情報を転送するのを支援する基本ルーチンを含むＢＩＯＳ（ｂａｓｉｃ　ｉｎｐｕｔ／ｏｕｔｐｕｔ　ｓｙｓｔｅｍ）２６を記憶することができる。パーソナルコンピュータ２０はまた、ハードディスク（図示せず）に対して読み書きするためのハードディスクドライブ２７と、取外し可能な磁気ディスク２９に対して読み書きするための磁気ディスクドライブ２８と、ＣＤ　ＲＯＭやその他の光媒体など取外し可能な光ディスク３１に対して読み書きするための光ディスクドライブ３０を備えることもできる。ハードディスクドライブ２７、磁気ディスクドライブ２８、および光ディスクドライブ３０は、それぞれハードディスクドライブインタフェース３２、磁気ディスクドライブインタフェース３３、および光ドライブインタフェース３４によってシステムバス２３に接続される。ドライブおよびそれらに関連するコンピュータ読み取り可能な媒体は、コンピュータ読み取り可能な命令、データ構造、プログラムモジュール、およびその他のデータの不揮発性記憶域をパーソナルコンピュータ２０に提供する。本明細書に述べる例

示的な環境は、ハードディスク、取外し可能な磁気ディスク２９、および取外し可能な光ディスク３１を利用しているが、磁気カセット、フラッシュメモリカード、ディジタルビデオディスク、ベルヌーイカートリッジ、ＲＡＭ、ＲＯＭなど、コンピュータによってアクセス可能なデータを記憶可能な他の種類のコンピュータ読み取り可能な媒体をこの例示的な動作環境で使用してもよいことを、当業者は理解されたい。

【００１７】

ハードディスク、磁気ディスク２９、光ディスク３１、ＲＯＭ２４、またはＲＡＭ２５には、いくつかのプログラムモジュールを記憶することができ、これらのプログラムモジュールにはオペレーティングシステム３５、１つまたは複数のアプリケーションプログラム３６、その他のプログラムモジュール３７、およびプログラムデータ３８が含まれる。ユーザは、キーボード４０やポインティングデバイス４２などの入力デバイスを介して、パーソナルコンピュータ２０にコマンドおよび情報を入力することができる。その他の入力デバイス（図示せず）には、マイクロホン、ジョイスティック、ゲームパッド、衛星受信アンテナ、スキャナなどを含めることができる。これらおよびその他の入力デバイスは、システムバスに結合されたシリアルポートインタフェース４６を介して処理ユニット２１に接続されることが多いが、パラレルポート、ゲームポート、またはＵＳＢ（Ｕｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　Ｂｕｓ）など、他のインタフェースによって接続されてもよい。モニタ４７または他の種類の表示デバイスも、ビデオアダプタ４８などのインタフェースを介してシステムバス２３に接続される。モニタ４７に加えて、パーソナルコンピュータは通常、スピーカやプリンタなど、その他の周辺出力デバイス（図示せず）も備える。図１の例示的なシステムはまた、ホストアダプタ５５、ＳＣＳＩ（Ｓｍａｌｌ　Ｃｏｍｐｕｔｅｒ　Ｓｙｓｔｅｍ　Ｉｎｔｅｒｆａｃｅ）バス５６、および、ＳＣＳＩバス５６に接続された外部記憶デバイス６２も備える。

【００１８】

パーソナルコンピュータ２０は、リモートコンピュータ４９など、１つまたは複数のリモートコンピュータへの論理接続を使用して、ネットワーク化された環境で動作することができる。リモートコンピュータ４９は、別のパーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピアデバイス、またはその他の一般的なネットワークノードであってよく、通常はパーソナルコンピュータ２０に関して上述した要素の多くまたはすべてを備えるが、図１にはメモリ記憶デバイス５０だけが示してある。図１に示す論理接続は、ＬＡＮ（Ｌｏｃａｌ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）５１およびＷＡＮ（Ｗｉｄｅ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）５２を含む。このようなネットワーキング環境は、オフィス、企業規模のコンピュータネットワーク、イントラネット、およびインターネットでよく見られる。

【００１９】

ＬＡＮネットワーキング環境で使用されるときは、パーソナルコンピュータ２０は、ネットワークインタフェースまたはアダプタ５３を介してＬＡＮ５１に接続される。ＷＡＮネットワーキング環境で使用されるときは、パーソナルコンピュータ２０は通常、インターネットなどのワイドエリアネットワーク５２を介した通信を確立するためのモデム５４または他の手段を備える。モデム５４は内蔵でも外付けでもよく、シリアルポートインタフェース４６を介してシステムバス２３に接続される。ネットワーク化された環境では、パーソナルコンピュータ２０に関して示したプログラムモジュールまたはその一部を、リモートのメモリ記憶デバイスに記憶することができる。図示のネットワーク接続は例であり、コンピュータ間で通信リンクを確立するための他の手段を使用してもよいことは理解されるであろう。さらに、本発明の多くの実施形態は、コンピュータ化されたシステムに特によく適合すると考えられるが、本明細書中のいずれのものも、本発明をかかる実施形態に限定することを意図していない。

【００２０】

（仮想マシン）

概念的に見ると、コンピュータシステムは一般に、基礎的なハードウェア層上で稼動す

る１つまたは複数のソフトウェア層を備えている。この階層化は、抽象化の理由で行われる。所与のソフトウェア層のためのインタフェースを定義することにより、この層を、それよりも上にある他の層によって様々に実装することが可能である。適切に設計されたコンピュータシステムでは、各層は、その直下の層だけについて識別している（また、直下の層だけに依拠する）。これにより、ある層または「スタック」（隣接する複数の層）を、この層またはスタックよりも上の層に悪影響を与えずに置き換えることができる。例えば、ソフトウェアアプリケーション（上方の層）は通常、より低いレベルのオペレーティングシステム（下方の層）に依拠して、ファイルを何らかの形の永久記憶装置に書き込むが、これらのアプリケーションは、データをフロッピー（登録商標）ディスクに書き込むか、ハードドライブに書き込むか、ネットワークフォルダに書き込むかの違いを理解する必要はない。この下方の層がファイル書込み用の新しいオペレーティングシステムコンポーネントで置き換えられても、上方の層のソフトウェアアプリケーションの動作は影響を受けない。

【００２１】

階層化されたソフトウェアのフレキシビリティは、仮想マシン（ＶＭ）が、実際には別のソフトウェア層である仮想ハードウェア層を提示することを可能にする。このようにして、ＶＭは、ＶＭよりも上のソフトウェア層に対して、これらのソフトウェア層がそれら自体の個別コンピュータシステム上で稼動しているという錯覚を生み出すことが可能であり、したがってＶＭは、複数の「ゲストシステム」が単一の「ホストシステム」上で同時に稼動できるようにすることが可能である。

【００２２】

図２は、コンピュータシステム中のエミュレートされた動作環境での、ハードウェアおよびソフトウェアアーキテクチャの論理階層化を表す図である。図では、エミュレーションプログラム９４が、物理ハードウェアアーキテクチャ９２の上で直接的または間接的に稼動する。エミュレーションプログラム９４は、（ａ）ホストオペレーティングシステムと並行して稼動する仮想マシンモニタ、（ｂ）ネイティブエミュレーション能力を有する特殊化されたホストオペレーティングシステム、または（ｃ）ハイパーバイザコンポーネントを備えるホストオペレーティングシステムとすることができ、（ｃ）の場合はハイパーバイザコンポーネントがエミュレーションを実行する。エミュレーションプログラム９４は、ゲストハードウェアアーキテクチャ９６をエミュレートする（ゲストハードウェアアーキテクチャ９６は破線で示してあり、それにより、このコンポーネントが「仮想マシン」であること、すなわち実際に存在するのではなくエミュレーションプログラム９４によってエミュレートされたハードウェアであることを示す）。ゲストハードウェアアーキテクチャ９６上では、ゲストオペレーティングシステム９８が実行され、ゲストオペレーティングシステム９８上では、ソフトウェアアプリケーション１００が稼動する。図２のエミュレートされた動作環境では、またエミュレーションプログラム９４の作用により、ソフトウェアアプリケーション１００は、ホストオペレーティングシステムおよびハードウェアアーキテクチャ９２との互換性を概して持たないオペレーティングシステム上で稼動するように設計されているにもかかわらず、コンピュータシステム９０中で稼動することが可能である。

【００２３】

図３Ａに、物理コンピュータハードウェア１０２のすぐ上で稼動するホストオペレーティングシステムソフトウェア層１０４を含む、仮想化されたコンピューティングシステムを示す。ホストオペレーティングシステム（ホストＯＳ）１０４は、ホストＯＳがエミュレートしている（すなわち「仮想化」している）ハードウェアと同じインタフェースを公開することによって、物理コンピュータハードウェア１０２のリソースへのアクセスを提供し、これによりホストＯＳは、ホストＯＳよりも上で稼動するオペレーティングシステム層に認識されずにいることができる。エミュレーションを実行するためには、ホストオペレーティングシステム１０４は、ネイティブエミュレーション能力を備えた特別設計のオペレーティングシステムとすることができ、あるいは、エミュレーションを実行するた

めの組込みハイパーバイザコンポーネントを備えた標準的なオペレーティングシステムとすることもできる。

【００２４】

再び図３Ａを参照すると、ホストＯＳ１０４の上には、２つの仮想マシン（ＶＭ）実装形態ＶＭＡ１０８およびＶＭＢ１１０があり、ＶＭＡ１０８は例えば、仮想化されたＩｎｔｅｌ３８６プロセッサとすることができ、ＶＭＢ１１０は例えば、Ｍｏｔｏｒｏｌａ６８０Ｘ０ファミリのプロセッサのうちの１つの仮想化バージョンとすることができる。各ＶＭ１０８および１１０の上には、それぞれゲストオペレーティングシステム（ゲストＯＳ）Ａ１１２およびＢ１１４がある。ゲストＯＳＡ１１２の上では、２つのアプリケーションすなわちアプリケーションＡ１　１１６およびアプリケーションＡ２　１１８が稼動しており、ゲストＯＳＢ１１４の上では、アプリケーションＢ１　１２０が稼動している。

【００２５】

図３Ａに関して、以下のことに留意するのが重要である。すなわち、（破線で示す）ＶＭＡ１０８およびＶＭＢ１１０は、仮想化されたコンピュータハードウェアの表現であり、これらはソフトウェア構造としてのみ存在し、これらは専用ソフトウェアコードが存在することによって実行可能になり（ｐｏｓｓｉｂｌｅ）、この専用ソフトウェアコードは、ＶＭＡ１０８およびＶＭＢ１１０をそれぞれゲストＯＳＡ１１２およびゲストＯＳＢ１１４に提示するだけでなく、ゲストＯＳＡ１１２およびゲストＯＳＢ１１４が実際の物理コンピュータハードウェア１０２と間接的に対話するのに必要なすべてのソフトウェアの手続きをも実行する。

【００２６】

図３Ｂに、代替の仮想化されたコンピューティングシステムを示す。このコンピューティングシステムでは、ホストオペレーティングシステム１０４”と並行して稼動する仮想マシンモニタ（ＶＭＭ）１０４’によってエミュレーションが実行される。いくつかの実施形態では、ＶＭＭは、ホストオペレーティングシステム１０４の上で稼動し、およびホストオペレーティングシステム１０４を介してのみコンピュータハードウェアと対話する、アプリケーションとすることができる。他の実施形態では、図３Ｂに示すように、ＶＭＭは代わりに、部分的に独立したソフトウェアシステムを構成することができ、そのソフトウェアシステムはあるレベルではホストオペレーティングシステム１０４を介して間接的にコンピュータハードウェア１０２と対話するが、他のレベルでは（ホストオペレーティングシステムがコンピュータハードウェアと直接に対話するのと同様の方式で）コンピュータハードウェア１０２と直接に対話する。他の実施形態では、ＶＭＭは、完全に独立したソフトウェアシステムを構成することができ、このソフトウェアシステムは、（コンピュータハードウェア１０２の使用を調整して競合を回避することなどに関する限りにおいては、やはりホストオペレーティングシステム１０４と対話するが）ホストオペレーティングシステム１０４を利用せずに、（ホストオペレーティングシステムがコンピュータハードウェアと直接に対話するのと同様の方式で）すべてのレベルでコンピュータハードウェア１０２と直接に対話する。

【００２７】

ＶＭＭを実装するためのこれらの変形はすべて、本明細書に述べる本発明の代替実施形態を形成することが予期され、本明細書中のどんなものも、本発明をいずれか特定のＶＭＭ構成に限定するものと解釈されるべきではない。加えて、（おそらくハードウェアエミュレーションのシナリオで、）ＶＭＡ１０８および／またはＶＭＢ１１０をそれぞれ介したアプリケーション１１６、１１８、１２０の間の対話に関するどんな言及も、実際にはアプリケーション１１６、１１８、１２０とＶＭＭとの間の対話であると解釈されるべきである。同様に、（おそらくコンピュータ命令をコンピュータハードウェア１０２に対して直接的または間接的に実行するための）アプリケーションＶＭＡ１０８および／またはＶＭＢ１１０と、ホストオペレーティングシステム１０４および／またはコンピュータハードウェア１０２との間の対話に関するどんな言及も、実際には、ＶＭＭと、適宜ホスト

オペレーティングシステム１０４またはコンピュータハードウェア１０２との間の対話であると解釈されるべきである。

【００２８】

（イベントインターセプトの概観）

本明細書で先に論じたように、バーチャライザは、コンピュータシステム上で稼動する薄いソフトウェア層であり、１つまたは複数の抽象仮想マシン（ＶＭ）インスタンス（それぞれは区分とも呼ばれる）を生み出すことを可能にする。これらの各インスタンスは通常、それ自体のメモリと、１つまたは複数のプロセッサと、Ｉ／Ｏデバイスと（これらは、実際の物理リソースに対応するか、または実際にはどんな物理的意味においても存在しない仮想化されたリソースに対応することができる）を備えた、実際のコンピュータのように振舞う。仮想化を実施するために、ほとんどのバーチャライザは、ゲストソフトウェアが区分中で実行される間に発生する特定のイベントをインターセプトする必要がある。（ゲストソフトウェアは、特定の仮想マシン内で稼動するソフトウェアであり、通常はオペレーティングシステムおよび１つまたは複数のアプリケーションからなる。）特定のイベントをインターセプトすることは、仮想化において重要である。というのは、そうすることによってバーチャライザは、実際には物理マシン上のすべてのリソースは共有または区分化されている（すなわち細分されて特定の仮想マシンに割り当てられている）ときに、ゲストを騙して、これらのリソースすべてを所有していると思わせることができるからである。例えば、４つのプロセッサと、プロセッサの総数を報告するレジスタとを有し、さらに、システム上で稼動する４つの仮想マシン（区分）を有するコンピュータシステムであって、各仮想マシンが単一のプロセッサに割り当てられているコンピュータシステムを考えてみる。この例で、これらの各仮想マシンのゲストソフトウェアが「プロセッサ数」レジスタを読み取るとき、通常は値「４」を読み取ることになる（これは実際には物理プロセッサの総数であり、したがって物理レジスタ中の値である）。しかしこの場合、バーチャライザは、各ゲストＯＳを騙して、１つのプロセッサシステム上で稼動していると認識させる必要がある。これを達成するために、バーチャライザは、ゲストが「プロセッサ数」レジスタから読み取るときに「インターセプト」イベントを生成するように、各仮想マシン（区分）の仮想プロセッサを構成する。このインターセプトは、ＶＭがレジスタを直接に読み取るのを防止し、制御をゲストからバーチャライザに移行させる。次いでバーチャライザは、レジスタ中に実際に存在する正常値「４」を、バーチャライザが報告したい値、この場合なら「１」で、オーバーライド（上書き：ｏｖｅｒｒｉｄｅ）することができる。

【００２９】

当業者には周知であり容易に理解されるように、一般にインターセプトは、プロセッサインターセプトとメモリインターセプトという２つの範疇に分けることが可能である。プロセッサインターセプトは、特定のプロセッサ命令、レジスタ、例外、または割込みに結び付けられるイベントである。メモリインターセプトは、指定のメモリ位置に対する読取りまたは書込みをプロセッサに行わせるアクションに特有であり、明示的なアクセス（例えば、メモリから読み取る「ＭＯＶＥ」命令）、または暗黙的なアクセス（例えば、プロセッサがＴＬＢ（ｔｒａｎｓｌａｔｉｏｎ　ｌｏｏｋ－ａｓｉｄｅ　ｂｕｆｆｅｒ）ミスに応答してページテーブルを検索（ｗａｌｋ）するときにプロセッサによって生成される参照）が含まれる。

【００３０】

従来、（各区分の外に存在する）バーチャライザは、インターセプトを「インストールする」ことと、インターセプトが発生したときにそれに「応答する」こととの両方を担う。インターセプトをインストールする技法は、当該のプロセッサアーキテクチャに大きく依存する。ほとんどのプロセッサは、いくつかのイベント時に「トラップ」する方法を提供する。いくつかの場合では、インターセプトに対する制御は、細粒または粗粒とすることが可能である。後者の場合、バーチャライザは、単一の粗粒インターセプトをインストールし、次いで、インターセプトしたい特定のイベントを「フィルタにかけて選び出し」

、他のインターセプトイベントは無視する必要がある場合がある。プロセッサが特定のインターセプトタイプに対する細粒制御を提供する場合は、バーチャライザは、欲しいインターセプトだけを要求することが可能である。インターセプト報告は一般にかなりのオーバーヘッド処理コストを追加し、これはシステム性能を低下させるので、理想的には、インターセプトは、バーチャライザが標準的なプロセッサ挙動をオーバーライドする必要があるときにだけ行われるべきである。先の４プロセッサ４ＶＭコンピュータシステムの例に再び言及するが、ユーザが４プロセッサの物理システム上で単一の４プロセッサ仮想マシンを構成した場合は、「プロセッサ数」レジスタへのアクセスをインターセプトする必要はなく、したがって、適切に書かれたバーチャライザなら、この場合にインターセプトを回避することになり、それにより、不要なオーバーヘッドコストおよび性能打撃（ｈｉｔ）を回避することになる。

【００３１】

インターセプトの必要を低減するための別の手法は、「シャドウレジスタ（ｓｈａｄｏｗ　ｒｅｇｉｓｔｅｒ）」の一般概念である。シャドウレジスタは、仮想マシン内で動作しているときにアクセスされる、実際のレジスタのコピーである。再び我々の先の例に言及するが、システムがシャドウ「プロセッサ数」レジスタを実装した場合、仮想化された環境の外で動作するプロセス（バーチャライザ自体を含む）は、このレジスタにアクセスして実際の値を読み取ることになるが、仮想化された環境の内で稼動するゲストプロセスの場合は、このレジスタにアクセスするとシャドウ値を読み取ることになり、バーチャライザは、シャドウレジスタが適切なプロセッサ数を報告するようにプログラム済みである。この情報をあらかじめプログラムしておくことにより、バーチャライザは、このレジスタアクセスに対するインターセプトを要求する必要はなく、その代わり、全般的に、各区分に実際の値ではなくシャドウ値を読み取らせる。しかし、この手法は、コストのかかるインターセプトイベントを回避するが、プロセッサがさらに複雑になること（通常は追加のレジスタ）を必要とし、その結果、（任意のアクションを実行可能な汎用インターセプトハンドラと比較して）フレキシビリティがいくらか減少する。

【００３２】

より具体的には、従来のバーチャライザは、本質的にモノリシックであった。すなわち、インターセプトをインストールすることと、インターセプトされるイベントが発生したときにそれに応答することとの両方を担っていた。したがって、これらのバーチャライザは今まで、必然的に複雑で厄介であった。図４は、プロセッサがサポートする手段が何であれ（例えば例外、デフォルト、トラップなど）それを介してインターセプトをインストールすることと、インターセプトされるイベントすべてに応答することとの両方を行う、従来のバーチャライザのモノリシックな性質を示すブロック図である。図では、物理コンピュータハードウェア４０２の上で動作するバーチャライザ４０４が、２つの仮想マシン環境ＶＭＡ４０８およびＶＭＢ４１０を仮想化する。ゲストオペレーティングシステム（ＧＯＳ）Ａ４１２がＶＭＡ４０８中で実行され、ＧＯＳＢ４１４がＶＭＢ４１０中で実行される。ソフトウェアアプリケーションＡ１　４１６およびＡ２　４１８が、ＶＭＡ４０８上のＧＯＳＡ４１２中で実行され、ソフトウェアアプリケーションＢ１　４２０が、ＶＭＢ４１０上のＧＯＳＢ４１４中で実行される。この実施形態では、バーチャライザはハイパーバイザとして示されており（ただし、その他のバーチャライザも予期され使用できる）、ＶＭＡ４０８は「１次区分」である。すなわち、ＶＭＡのゲストオペレーティングシステム（ＧＯＳ）Ａはハイパーバイザによって利用され、普通ならホストオペレーティングシステムから提供されるであろう機能を提供する（当業者には理解されるであろうが、ホストオペレーティングシステムは、この特定の仮想化構成では不要であり存在しない）。

【００３３】

インターセプトに関して、バーチャライザ４０４（やはりこの場合もハイパーバイザ）は、完全なインターセプト機能４５０を備え、これによりバーチャライザ４０４の複雑さは非常に高まる。したがって、この構成では、第１の区分（例えばＶＭＡ４０８）中のイ

ベントがインターセプトを誘因（ｔｒｉｇｇｅｒ）し、プロセッサに、第１の区分で稼動するゲストＯＳ（例えばゲストＯＳＡ４１２）からバーチャライザ４０４に制御を移行させる。次いでバーチャライザ４０４は、インターセプトの原因を決定してそれに応答し、完了時に制御をゲストＯＳＡ４１２に返す。

【００３４】

（マルチレベルインターセプトイベント）

本発明の様々な実施形態は、バーチャライザとインターセプト処理方法とのためのマルチレベル設計を対象とし、このマルチレベル設計では、インターセプトに関係する複雑さの大部分をベースレベルのバーチャライザ（仮想化された環境の外で実行されるバーチャライザコンポーネント）の外に移して、各区分（仮想マシン）内にある外部モニタ中に置くことができる。この構成では、（いくつかの実施形態では汎用ハイパーバイザに統合することができる）ベースレベルのバーチャライザは単純なままであり、インターセプト処理のいくらかまたはすべては、各区分内すなわちゲストレベルで稼動する１つまたは複数の外部モニタによって、または、インターセプトイベントを生成した区分と同じ区分または異なる区分で実行される。

【００３５】

図５は、本発明のいくつかの実施形態に関するマルチレベルインターセプト手法を示すブロック図である。図では、物理コンピュータハードウェア５０２の上で動作するバーチャライザ５０４が、２つの仮想マシン環境ＶＭＡ５０８およびＶＭＢ５１０を仮想化する。ＧＯＳＡ５１２がＶＭＡ５０８中で実行され、ＧＯＳＢ５１４がＶＭＢ５１０中で実行される。ソフトウェアアプリケーションＡ１　５１６およびＡ２　５１８が、ＶＭＡ５０８上のＧＯＳＡ５１２中で実行され、ソフトウェアアプリケーションＢ１　５２０が、ＶＭＢ５１０上のＧＯＳＢ５１４中で実行される。この実施形態でも、バーチャライザはやはりハイパーバイザとして示されており（ただし、その他のバーチャライザも予期され使用できる）、ＶＭＡ５０８は「１次区分」である。すなわち、ＶＭＡのゲストオペレーティングシステム（ＧＯＳ）Ａはハイパーバイザによって利用され、普通ならホストオペレーティングシステムから提供されるであろう機能を提供する。

【００３６】

しかし、インターセプトに関して、バーチャライザ５０４（やはりこの場合もハイパーバイザ）は、最小限のインターセプト機能５５０だけを備え、その代わり、インターセプト機能の大部分は、各区分５０８および５１０中の１つまたは複数の外部モニタ（「ＥＭ」）５６２および５６４に組み込まれている（ここでは各区分のＶＭ仮想化に組み込まれているのが示されているが、代替の実施形態では、区分中のＶＭとゲストＯＳとアプリケーションとに対する相対的なＥＭの位置は、異なってもよい）。

【００３７】

この構成では、図５と６の両方を参照すると（後者の図は、マルチレベル方法がインターセプトイベントを処理する方法を示すプロセスフロー図である）、ステップ６０２で、第１の区分（例えばＶＭＡ５０８）中のイベントがインターセプトを誘因し、ステップ６０４で、プロセッサに、第１の区分中で稼動するゲストＯＳ（例えばゲストＯＳＡ５１２）からバーチャライザ５０４に制御を移行させる。ステップ６０６で、次いでバーチャライザ５０４は、それ自体のデフォルトの（かつ単純な）処理機構を使用してインターセプトを処理するか、あるいはインターセプトを「外部モニタ」（例えばＥＭ５６２）に渡すかを決定する。例えばバーチャライザは、インターセプトのソースがデフォルト処理を要求した場合に、（ステップ６０８でインターセプトをそれ自体で処理する）前者を選択することができ、あるいは反対に、例えばデフォルト挙動を使用しないようインターセプトのソースが要求した場合に、（ステップ６１０を介してインターセプトをＥＭに転送する）後者を選択することができる。バーチャライザがインターセプトを処理するために外部モニタに転送しようとする場合、バーチャライザはまず、ステップ６１０で、インターセプトを処理することになる特定の外部モニタに信号を送り、ステップ６１２で、このＥＭが２つの方法のうちの一方でインターセプトに応答できるようにする。２つの方法は次の

とおりである。（ａ）（ＥＭが特定のインターセプトイベントを処理することができないときなど）ＥＭは、ステップ６１４で、インターセプトをバーチャライザに返し、インターセプトを通常のデフォルト方式で処理するようバーチャライザに要求する。あるいは（ｂ）ＥＭは、ステップ６１６でインターセプトを処理する（かつ、いくつかの実施形態では、バーチャライザによるデフォルトアクションが不要であることをバーチャライザに通知する）。インターセプト処理が完了すると、ステップ６１８で、制御はインターセプトのソース（例えばゲストＯＳＡ４１２）に返される。

【００３８】

実施の際、外部モニタを各区分ごと、各仮想プロセッサごと、各インターセプトクラスごとにバーチャライザに登録することが可能であり、あるいは、登録の代わりに、インターセプトを処理する必要があるときにバーチャライザがＥＭをポーリングすることができる（すなわち、１つの外部モニタがインターセプトを処理すると決定するまで、順番に呼び出すことができる）。本発明のいくつかの実施形態では、外部モニタは区分ごとにのみ登録することができ、それにより、所与の区分に関するすべての要求されたインターセプトは、指定の外部モニタに向けられることになる。しかし、他の実施形態では、複数の区分が、様々な区分で稼動する代替の外部モニタを有することができ、これにより、各外部モニタが各区分に対してわずかに異なる挙動（例えば異なるバージョンの外部モニタ）を有することができる。

【００３９】

いくつかの実施形態では、循環依存およびデッドロック状況を防止するために、区分はそれ自体のインターセプトを処理することが許されないものとすることができる。したがって、これらの実施形態では、各区分の外部モニタは別の区分内で稼動する必要がある。１次区分を有するハイパーバイザベースのシステムでは、当業者には理解されるように、このことは、少なくとも１つの区分（１次区分）は外部モニタを有することが不可能であることを意味し、したがって、このような実施形態の１次区分は、バーチャライザ（ハイパーバイザ）によって提供されるデフォルトのインターセプト処理だけに依拠することが可能である。他方、（ホストオペレーティングシステムを有する）ホストされるシステムに関するいくつかの代替の実施形態では、循環依存およびデッドロック状況に関する同様の懸念は実際のホストオペレーティングシステムには当てはまらないので、外部モニタはホストＯＳおよびすべての区分内で稼動することが可能である。

【００４０】

（代替の実施形態）

本明細書で先に述べたように、インターセプトは、メモリインターセプトとプロセッサインターセプトという２つのクラスに分けることが可能である。とりわけ、メモリインターセプトは、メモリマップレジスタ、ＲＯＭ、プレーナＶＲＡＭなど、通常のＲＡＭのように働かないメモリの領域を仮想化するのに使用され、限定はされないが次のことのために使用することができる。すなわち、読取り（１から１６バイト）と、書込み（１から１６バイト）と、ロック付き読取り／修正／書込み論理演算ＡＮＤ、ＯＲ、ＸＯＲ（１、２、４、または８バイト）と、ロック付き読取り／修正／書込み算術演算ＡＤＤ、ＳＵＢ（１、２、４、または８バイト）と、ロック付き比較演算ＸＣＨＧ、ＣＭＰＸＣＨＧ（１、２、４、または８バイト）である。対照的に、とりわけ、プロセッサインターセプトは、次のことのために使用することができる。すなわち、ＨＬＴ、ＰＡＵＳＥ、およびＣＰＵＩＤ命令と、制御レジスタ（ＣＲ）、デバッグレジスタ（ＤＲ）、モデル固有レジスタ（ＭＳＲ）へのアクセスと、特定の例外と、外部割込みと、Ｉ／Ｏポートへのアクセス（ＩＮ／ＯＵＴ命令）と、致命的例外（例えばトリプルフォールト）である。

【００４１】

加えて、本発明のいくつかの実施形態では、ゲスト命令の途中でインターセプトを外部モニタに報告することができる。例えば、単一の命令が、関連するインターセプトを有する２つのメモリアドレスにアクセスする場合がある。この場合、ゲスト命令を完了することができる前に、両方のインターセプトを処理する必要がある。これらのインターセプト

は、連続的に検出および処理され、いくつかの実施形態では、バーチャライザは命令完了を実施することを担う。例えば、ＣＰＵＩＤは、レジスタＥＡＸ、ＥＤＸ、ＥＢＸ、ＥＣＸ中の、プロセッサに関する情報を返し、バーチャライザがＣＰＵＩＤインターセプトを外部モニタに報告するために選択できる方法は２つある。すなわち、（ａ）外部モニタは、ＣＰＵＩＤ命令を完了し、逆に適切なレジスタに値を書き込み、ＣＰＵＩＤ命令を通過してＥＩＰ（命令ポインタ）をインクリメントするか、あるいは（ｂ）外部モニタは、ＣＰＵＩＤ命令から提供して欲しい値を返す。前者の場合では、バーチャライザは、命令間のインターセプトを配信している。外部モニタから見ると、仮想化されたプロセッサはまだＣＰＵＩＤ命令を実行しておらず、外部モニタがインターセプトの処理を終えたとき、ＣＰＵＩＤ命令の実行は完了することになる。他方、後者の場合では、バーチャライザは、逆にこれらの値を適切なレジスタに書き込み、ＣＰＵＩＤ命令の最後を通過してＥＩＰをインクリメントすることを担う。この方法は、バーチャライザが命令中インターセプトを配信しているとき、より単純な外部モニタおよびよりよい性能をもたらし、外部モニタから見ると、仮想化されたプロセッサはＣＰＵＩＤ命令を実行し始めている。外部モニタがインターセプトの処理を終えたとき、バーチャライザはＣＰＵＩＤ命令を完了することになる。加えて、いくつかの実施形態は、可能な場合は常に命令中インターセプトをサポートする。というのは、より単純な外部モニタインタフェースがもたらされるとともに、例外に対するインターセプトや、命令境界で必然的に発生する割込みなど、命令間インターセプトも可能だからである。

【００４２】

さらに、前に論じたように、ベースレベルモニタは、各インターセプトタイプにつき、「デフォルトの」（通常は単純な）処理を定義する。例えば、ＣＰＵＩＤ命令はデフォルトで、物理プロセッサが通常返すことになるのと同じＣＰＵ情報を返し、致命的例外（例えばｘ８６アーキテクチャに対するトリプルフォールト）はデフォルトで、区分を終了させる。しかし、いくつかの実施形態では、外部モニタは、基礎をなすバーチャライザを呼び出して、どのインターセプトをオーバーライドしたいかを指定することにより、特定のインターセプトクラスのデフォルトの挙動をオーバーライドすることができ、オーバーライドされないインターセプトは、引き続きデフォルト方式で処理される。

【００４３】

ほとんどのバーチャライザはまた、明確に定義された機構を介してゲストソフトウェアがバーチャライザと通信できるようにもする。本発明のいくつかの実施形態では、これは、総合ＭＳＲ（モデル固有レジスタ）を使用することによって行うことができる。外部モニタがこれらのＭＳＲアクセスの挙動をオーバーライドできるようにすることによって、外部モニタは、バーチャライザが存在しないように見せることが可能である。バーチャライザはまた、異なるバージョンのハイパーバイザまたはＶＭＭの存在をシミュレートすることも可能である。これは、再帰的な仮想化を可能にする。すなわち、ハイパーバイザまたはＶＭＭは、別のハイパーバイザまたはＶＭＭの最上部で稼動する区分内で稼動することができる。これは、ハイパーバイザ／ＶＭＭの新バージョンのプロトタイプを作成すること、古い方のバージョンとの後方互換性を提供すること、または第三者の実装との互換性を提供することのために有用となることが可能である。

【００４４】

ベースレベルのバーチャライザと外部モニタとの両方を含む設計では、インターセプトが発生したときに外部モニタに信号を送り、任意選択でインターセプト関係のパラメータを渡す方法が必要である。外部モニタは、今は他のコードの実行に使用されているかもしれない別の区分で稼動しているので、いくつかの実施形態では、この信号を配信するための非同期機構が必要である。一実装形態では、この信号は、総合割込みとして配信することができる（本明細書で先に挙げた相互参照の特許出願を参照されたい）。従来のプロセッサ割込み機構は、特定の割込みソースに応答して割込みハンドラを実行することを必要とするが、総合割込みコントローラは、この機構を拡張して、割込みハンドラにパラメータを渡せるようにする。インターセプトの場合、これらのパラメータは、保留中のインタ

ーセプトに関する情報を示し、外部モニタは、この情報を使用して効率的にインターセプトを処理することが可能である。

【００４５】

（結び）

本明細書に述べた様々なシステム、方法、および技法は、ハードウェアまたはソフトウェア、あるいは適切ならこの両方の組合せで実施することができる。したがって、本発明の方法および装置、またはそのいくつかの態様または部分は、フロッピー（登録商標）ディスケット、ＣＤ－ＲＯＭ、ハードドライブ、またはその他の任意のマシン読み取り可能な記憶媒体など、有形媒体に組み入れられたプログラムコード（すなわち命令）の形をとることができ、プログラムコードがコンピュータなどのマシンにロードされマシンによって実行されると、このマシンは本発明を実施するための装置になる。プログラムコードがプログラム可能なコンピュータ上で実行される場合、コンピュータは一般に、プロセッサと、プロセッサによって読取り可能な記憶媒体（揮発性と不揮発性のメモリおよび／または記憶素子を含む）と、少なくとも１つの入力デバイスと、少なくとも１つの出力デバイスとを備えることになる。１つまたは複数のプログラムは、コンピュータシステムと通信するために高レベルのプロシージャ指向またはオブジェクト指向プログラミング言語で実現されることが好ましい。しかし、１つまたは複数のプログラムは、望むならアセンブリ言語またはマシン言語で実現することが可能である。いずれの場合でも、言語はコンパイルまたは解釈される言語とすることができ、ハードウェア実装形態と組み合わせることができる。

【００４６】

本発明の方法および装置はまた、電気ワイヤリングまたはケーブリング、光ファイバ、あるいはその他の任意の形の伝送など、何らかの伝送媒体を介して伝送されるプログラムコードの形で具体化することもでき、プログラムコードが、ＥＰＲＯＭ、ゲートアレイ、プログラマブルロジックデバイス（ＰＬＤ）、クライアントコンピュータ、ビデオレコーダなどのマシンによって受け取られてロードおよび実行されると、このマシンは本発明を実施するための装置になる。汎用プロセッサ上で実施されるとき、プログラムコードは、プロセッサと組み合わさって、本発明の索引付け機能を実施するように動作する独特の装置を提供する。

【００４７】

様々な図の好ましい実施形態に関して本発明を述べたが、本発明を逸脱することなく、その他の類似の実施形態を使用して、または述べた実施形態に修正および追加を施して、本発明と同じ機能を実施することもできることを理解されたい。例えば、本発明の例示的な実施形態は、パーソナルコンピュータの機能をエミュレートするディジタルデバイスのコンテキストで述べたが、本発明は、本明細書に述べたようなディジタルデバイスに限定されず、ゲーム用コンソール、ハンドヘルドコンピュータ、ポータブルコンピュータなど、有線であれ無線であれ既存のまたは新たに出現する任意の数のコンピューティングデバイスまたは環境にも適用することができ、通信ネットワークを介して接続されネットワークを介して対話する任意の数のこのようなコンピューティングデバイスに適用することができることは、当業者なら理解するであろう。さらに、特に無線ネットワーク化デバイスの数が急増し続けているので、本明細書では、ハンドヘルドデバイスオペレーティングシステムやその他の特定用途向けハードウェア／ソフトウェアインタフェースシステムを含めて、様々なコンピュータプラットフォームが企図されることを強調しておくべきである。したがって、本発明は、いずれか単一の実施形態に限定されるべきではなく、添付の特許請求の範囲による幅および範囲で解釈されるべきである。

【００４８】

最後に、本明細書に記載の開示した実施形態は、他のプロセッサアーキテクチャ、コンピュータベースのシステム、またはシステム仮想化で使用されるように適合させることもでき、そのような実施形態も、本明細書で行った開示によって明白に予期され、したがって本発明は、本明細書に述べた特定の実施形態に限定されるべきではなく、最も広範に解

釈されるべきである。同様に、プロセッサ仮想化以外の目的で総合命令を使用することも、本明細書で行った開示によって明白に予期され、プロセッサ仮想化以外のコンテキストにおけるこのようなどんな総合命令利用も、本明細書で行った開示の中に最も広範に読み込まれるべきである。

【図面の簡単な説明】

【００４９】

【図１】本発明の態様を組み込むことのできるコンピュータシステムを表すブロック図である。

【図２】コンピュータシステム中のエミュレートされた動作環境での、ハードウェアおよびソフトウェアアーキテクチャの論理階層化を示す図である。

【図３Ａ】エミュレーションがホストオペレーティングシステムによって（直接にまたはハイパーバイザを介して）実行される、仮想化されたコンピューティングシステムを示す図である。

【図３Ｂ】ホストオペレーティングシステムと並行して稼動する仮想マシンモニタによってエミュレーションが実行される、代替の仮想化されたコンピューティングシステムを示す図である。

【図４】プロセッサがサポートする手段が何であれ（例えば例外、デフォルト、トラップなど）それを介してインターセプトをインストールすることと、インターセプトされたイベントすべてに応答することとの両方を行う、従来のバーチャライザのモノリシックな性質を示すブロック図である。

【図５】本発明のいくつかの実施形態に関するマルチレベルインターセプト手法を示すブロック図である。

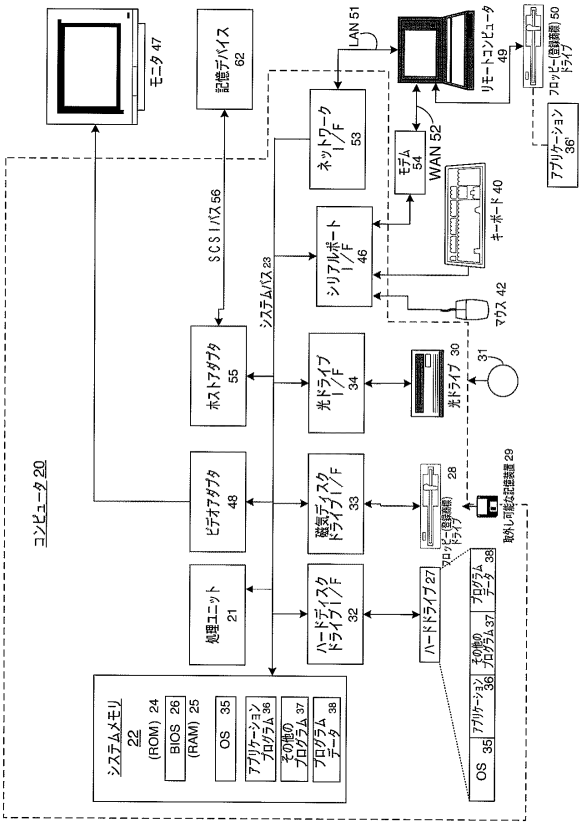
【図６】マルチレベル方法がインターセプトイベントを処理する方法を示すプロセスフロー図である。

【符号の説明】

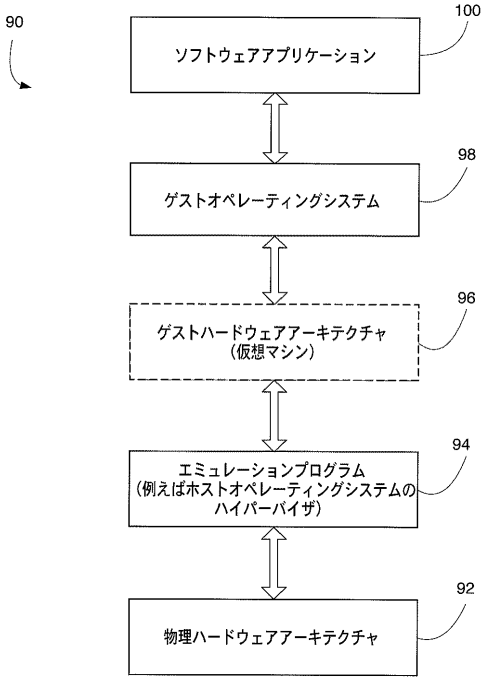
【００５０】

９２　物理ハードウェアアーキテクチャ
９４　エミュレーションプログラム
９６　ゲストハードウェアアーキテクチャ（仮想マシン）
９８　ゲストオペレーティングシステム
１００　ソフトウェアアプリケーション

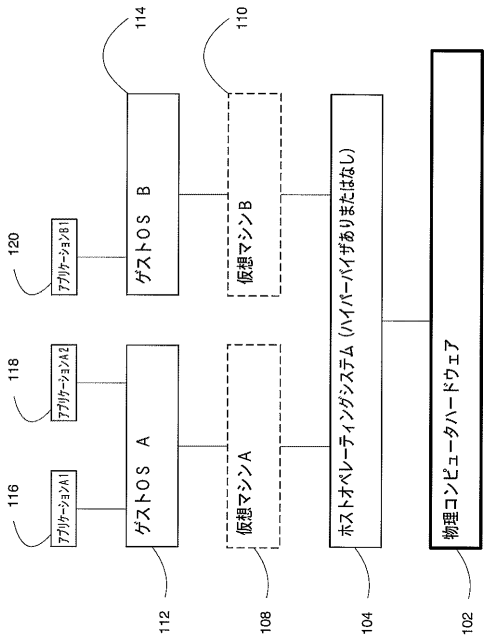
【図１】


モニタ 47
記憶デバイス 62
LAN 51
リモートコンピュータ 49
プロッター(筆記装置) 50
ドライブ
アプリケーション 39
ネットワーク I/F 53
モデム 54
WAN 52
キーボード 40
SCSIバス 56
シリアルポート I/F 46
システムバス 23
マウス 42
ホストアダプタ 55
光ドライブ I/F 34
光ドライブ 30
31
ビデオアダプタ 48
磁気ディスク ドライブ I/F 33
28
取外し可能な記憶装置 29
処理ユニット 21
ハードディスク I/F 32
ハードドライブ 27
プログラム データ 38
その他の プログラム 37
アプリケーション 36
OS 35
システムメモリ 22
(ROM) 24
BIOS 26
(RAM) 25
OS 35
アプリケーション プログラム 36
その他の プログラム 37
プログラム データ 38
コンピュータ 20

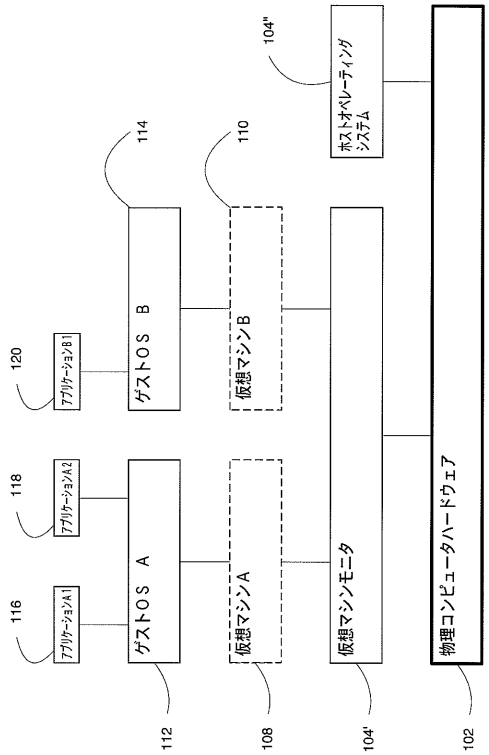
【図２】


90
100
ソフトウェアアプリケーション
98
ゲストオペレーティングシステム
96
ゲストハードウェアアーキテクチャ
（仮想マシン）
94
エミュレーションプログラム
（例えばホストオペレーティングシステムの
ハイパーバイザ）
92
物理ハードウェアアーキテクチャ

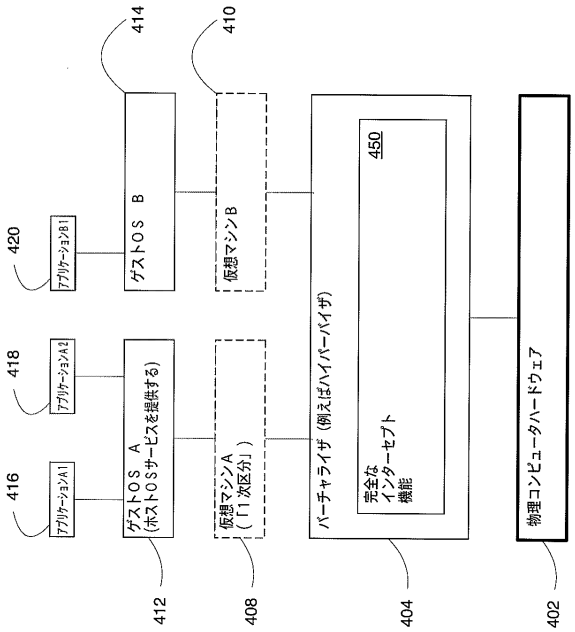
【図３Ａ】


114
110
120
アプリケーションB1
ゲストOS B
仮想マシンB
118
アプリケーションA2
116
アプリケーションA1
ゲストOS A
仮想マシンA
ホストオペレーティングシステム（ハイパーバイザありまたはなし）
物理コンピュータハードウェア
112
108
104
102

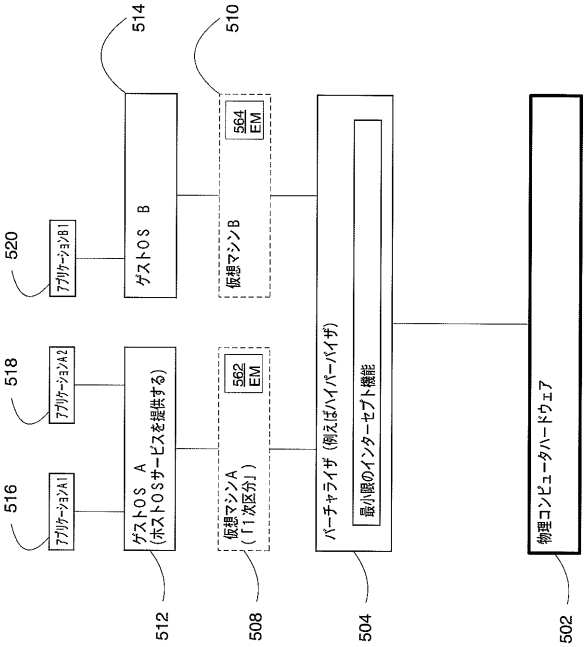
【図３Ｂ】


104"
ホストオペレーティング システム
114
110
120
アプリケーションB1
ゲストOS B
仮想マシンB
118
アプリケーションA2
116
アプリケーションA1
ゲストOS A
仮想マシンA
仮想マシンモニタ
物理コンピュータハードウェア
112
108
104'
102

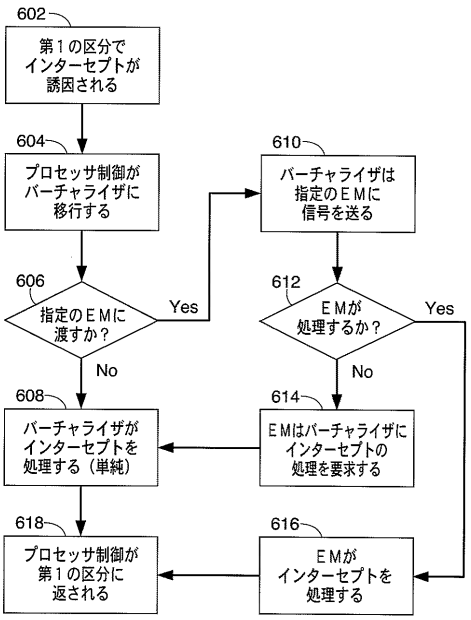
【図４】


414
410
420
アプリケーションB1
ゲストOS B
仮想マシンB
450
完全な
インターセプト
機能
バーチャライザ（例えばハイパーバイザ）
418
アプリケーションA2
ゲストOS A
（ホストOSサービスを提供する）
仮想マシンA
（「1次区分」）
416
アプリケーションA1
物理コンピュータハードウェア
412
408
404
402

【図５】


514
510
520
アプリケーションB1
ゲストOS B
564
EM
仮想マシンB
518
アプリケーションA2
ゲストOS A
（ホストOSサービスを提供する）
562
EM
仮想マシンA
（「1次区分」）
516
アプリケーションA1
バーチャライザ（例えばハイパーバイザ）
最小限のインターセプト機能
物理コンピュータハードウェア
512
508
504
502

【図６】


602
第1の区分で
インターセプトが
誘因される
604
プロセッサ制御が
バーチャライザに
移行する
606
指定のEMに
渡すか？
Yes
No
608
バーチャライザが
インターセプトを
処理する（単純）
618
プロセッサ制御が
第1の区分に
返される
610
バーチャライザは
指定のEMに
信号を送る
612
EMが
処理するか？
Yes
No
614
EMはバーチャライザに
インターセプトの
処理を要求する
616
EMが
インターセプトを
処理する

フロントページの続き

(72)発明者　レネ　アントニオ　ベガ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　ジョイ　ガングリー
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

審査官　久保　光宏

(56)参考文献　米国特許出願公開第２００４／０１１７５３２（ＵＳ，Ａ１）
特開平２－１９９３７（ＪＰ，Ａ）
国際公開第２００４／０９５２７２（WO，Ａ１）
特表２００８－５０８５９８（ＪＰ，Ａ）
特開平２－２３９３３４（ＪＰ，Ａ）
特開平３－１００８３３（ＪＰ，Ａ）
特開２００１－３１８７９７（ＪＰ，Ａ）
特表２００７－５０５４０２（ＪＰ，Ａ）
特開昭６２－２２１０４１（ＪＰ，Ａ）
特開平９－２８２１９６（ＪＰ，Ａ）
特開平９－８１４０１（ＪＰ，Ａ）
Barham, P., et.al., "Xen and the art of Virtualization", Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03), ２００３年１２月, p.164-177, ISBN:1-58113-757-5
梅野英典（外４名），「仮想計算機システムの高性能化方式」，情報処理，日本，社団法人情報処理学会，１９９０年１２月１５日，Vol.31，No.12，第１６６５～１６８０頁，ISSN:0447-8053
梅野英典（外２名），「仮想計算機システムにおける論理プロセッサをスケジュールする新方式の開発と評価」，情報処理学会論文誌，日本，社団法人情報処理学会，２００３年　3月１５日，Vol.44，No.3，第８６８～８８２頁，ISSN:0387-5806
「性能強化が進み本格的実用期に入った仮想計算機システム」，日経コンピュータ，日本，日経マグロウヒル社，１９８４年　7月２３日，1984年7月23日号（No.74），第６３～８２頁，ISSN:0285-4619

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４６－９／５４，
Ｇ０６Ｆ１３／１０－１３／１４，
ＣＳＤＢ（日本国特許庁）