(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6496401号
(P6496401)
(45)発行日　平成31年4月3日(2019.4.3)　(24)登録日　平成31年3月15日(2019.3.15)

(51)Int.Cl.　F I
G06F 21/56 (2013.01)　G06F 21/56
G06F 9/455 (2006.01)　G06F 9/455 １５０

請求項の数 21　(全 24 頁)

(21)出願番号　特願2017-509643 (P2017-509643)
(86)(22)出願日　平成27年8月11日 (2015.8.11)
(65)公表番号　特表2017-526071 (P2017-526071A)
(43)公表日　平成29年9月7日 (2017.9.7)
(86)国際出願番号　PCT/RO2015/050009
(87)国際公開番号　WO2016/118033
(87)国際公開日　平成28年7月28日 (2016.7.28)
審査請求日　平成30年2月26日 (2018.2.26)
(31)優先権主張番号　62/038,476
(32)優先日　平成26年8月18日 (2014.8.18)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　14/489,820
(32)優先日　平成26年9月18日 (2014.9.18)
(33)優先権主張国　米国 (US)

(73)特許権者　312016539
ビットディフェンダー　アイピーアール　マネジメント　リミテッド
キプロス共和国、ニコシア、クレオントス、１２　シーワイ－１０７６
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100118902
弁理士　山本　修
(74)代理人　100106208
弁理士　宮前　徹
(74)代理人　100120112
弁理士　中西　基晴
(74)代理人　100196508
弁理士　松尾　淳一

最終頁に続く

(54)【発明の名称】仮想マシンを終了する際に現在のプロセッサ命令の結果を公開するためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
仮想マシンとコンピュータセキュリティプログラムとを実行するように構成された少なくとも１つのハードウェアプロセッサを備えたホストシステムであって、前記少なくとも１つのハードウェアプロセッサは、
実行のためのゲスト命令を受け取るのに応答して、前記ゲスト命令の前記仮想マシン内での実行によりメモリアクセス許可の違反が生じるかどうかを判定することであって、前記ゲスト命令は、前記少なくとも１つのハードウェアプロセッサに、オペレータのオペランドへの適用の結果を求めるように命令し、前記ゲスト命令の前記違反のない実行は、前記結果の第１の場所への書き込みを生じさせる、判定することと、
前記ゲスト命令の実行により前記違反が生じるかどうかの判定に応答して、前記ゲスト命令の実行により前記違反が生じる場合に、前記結果を前記第１の場所とは異なる第２の場所に書き込むことであって、前記第２の場所は前記コンピュータセキュリティプログラムからアクセス可能である、書き込むことと、
前記結果の前記第２の場所への書き込みに応答して、前記ゲスト命令の実行から前記コンピュータセキュリティプログラムの実行に切り換えることであって、前記コンピュータセキュリティプログラムは、前記違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成された、切り換えることと
を行うようにさらに構成された、ホストシステム。
【請求項２】

請求項１に記載のホストシステムであって、前記第１の場所は前記少なくとも１つのハードウェアプロセッサの汎用レジスタである、ホストシステム。
【請求項３】
請求項１に記載のホストシステムであって、前記第１の場所は前記少なくとも１つのハードウェアプロセッサの制御レジスタである、ホストシステム。
【請求項４】
請求項１に記載のホストシステムであって、前記第１の場所は前記ホストシステムのメモリのセクションを含む、ホストシステム。
【請求項５】
請求項１に記載のホストシステムであって、前記結果はメモリアドレスを含む、ホストシステム。
【請求項６】
請求項１に記載のホストシステムであって、前記結果は前記オペランドを含む、ホストシステム。
【請求項７】
請求項１に記載のホストシステムであって、前記コンピュータセキュリティプログラムは前記仮想マシン内で実行される、ホストシステム。
【請求項８】
請求項１に記載のホストシステムであって、前記コンピュータセキュリティプログラムは前記仮想マシン外で実行される、ホストシステム。
【請求項９】
請求項１に記載のホストシステムであって、前記第２の場所は、前記少なくとも１つのハードウェアプロセッサの所定のレジスタを含む、ホストシステム。
【請求項１０】
請求項１に記載のホストシステムであって、前記第２の場所は、前記ホストシステムのメモリの所定のセクションを含む、ホストシステム。
【請求項１１】
請求項１に記載のホストシステムであって、前記第２の場所は、前記仮想マシンの現在状態を示すデータ構造を含む、ホストシステム。
【請求項１２】
請求項１に記載のホストシステムであって、前記少なくとも１つのハードウェアプロセッサは、前記コンピュータセキュリティプログラムの実行に切り換えるのに応答して、前記結果を前記第２の場所から読み出すようにさらに構成された、ホストシステム。
【請求項１３】
請求項１２に記載のホストシステムであって、前記少なくとも１つのハードウェアプロセッサは、前記結果を読み出すのに応答して、前記結果を、前記ゲスト命令に従って定まる宛先に書き込むようにさらに構成された、ホストシステム。
【請求項１４】
請求項１に記載のホストシステムであって、前記コンピュータセキュリティプログラムは、前記少なくとも１つのハードウェアプロセッサにより実行されると、前記少なくとも１つのハードウェアプロセッサに、前記結果を前記第２の場所から読み出し、前記結果を、前記ゲスト命令に従って定まる宛先に書き込むことをさせる命令を含む、ホストシステム。
【請求項１５】
請求項１に記載のホストシステムであって、前記少なくとも１つのハードウェアプロセッサは、前記コンピュータセキュリティプログラムの実行に切り換えるのに応答して、
前記ゲスト命令がアトミック命令であるかどうかを判定し、
それに応答して、前記ゲスト命令がアトミック命令である場合に、前記ゲスト命令を前記仮想マシン内で実行する
ようにさらに構成された、ホストシステム。

【請求項１６】
ホストシステムをコンピュータセキュリティの脅威から保護する方法であって、
実行のためのゲスト命令を受け取るのに応答して、前記ホストシステムの少なくとも１つのハードウェアプロセッサが、前記ゲスト命令の実行によりメモリアクセス許可の違反が生じるかどうかを判定するステップであって、前記ゲスト命令は、前記ホストシステムによって公開されるゲスト仮想マシン内で実行され、前記ゲスト命令は、前記少なくとも１つのハードウェアプロセッサに、オペレータのオペランドへの適用の結果を求めるように命令し、前記ゲスト命令の前記違反のない実行は、前記結果の第１の場所への書き込みを生じさせる、ステップと、
前記ゲスト命令の実行により前記違反が生じるかどうかの判定に応答して、前記ゲスト命令の実行により前記違反が生じる場合に、前記少なくとも１つのハードウェアプロセッサが、前記結果を前記第１の場所とは異なる第２の場所に書き込むステップであって、前記第２の場所はコンピュータセキュリティプログラムからアクセス可能である、ステップと、
前記結果の前記第２の場所への書き込みに応答して、前記少なくとも１つのハードウェアプロセッサが、前記ゲスト命令の実行から前記コンピュータセキュリティプログラムの実行に切り換えるステップであって、前記コンピュータセキュリティプログラムは、前記違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成された、ステップと
を含む方法。
【請求項１７】
請求項１６に記載の方法であって、前記コンピュータセキュリティプログラムの実行に切り換えるのに応答して、前記少なくとも１つのハードウェアプロセッサが、前記結果を前記第２の場所から読み出すステップをさらに含む方法。
【請求項１８】
請求項１７に記載の方法であって、前記結果を読み出すのに応答して、前記少なくとも１つのハードウェアプロセッサが、前記結果を、前記ゲスト命令に従い定まる宛先に書き込むステップをさらに含む方法。
【請求項１９】
ホストシステムの少なくとも１つのハードウェアプロセッサであって、
実行のためのゲスト命令を受け取るのに応答して、前記ゲスト命令の実行によりメモリアクセス許可の違反が生じるかどうかを判定することであって、前記ゲスト命令は、前記ホストシステムによって公開されるゲスト仮想マシン内で実行され、前記ゲスト命令は、前記ホストシステムによって公開されるゲスト仮想マシン内で実行され、前記ゲスト命令は、前記少なくとも１つのハードウェアプロセッサに、オペレータのオペランドへの適用の結果を求めるように命令し、前記ゲスト命令の前記違反のない実行は、前記結果の第１の場所への書き込みを生じさせる、判定することと
前記ゲスト命令の実行により前記違反が生じるかどうかの判定に応答して、前記ゲスト命令の実行により前記違反が生じる場合に、前記結果を前記第１の場所とは異なる第２の場所に書き込むことであって、前記第２の場所はコンピュータセキュリティプログラムからアクセス可能である、書き込むことと、
前記結果の前記第２の場所への書き込みに応答して、前記ゲスト命令の実行からコンピュータセキュリティプログラムの実行に切り換えることであって、前記コンピュータセキュリティプログラムは、前記違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成された、切り換えることと
を行うように構成された少なくとも１つのハードウェアプロセッサ。
【請求項２０】
ホストシステムをコンピュータセキュリティの脅威から保護するためのプログラムであって、前記プログラムは、前記ホストシステムの少なくとも１つのハードウェアプロセッサに、

実行のためのゲスト命令を受け取るのに応答して、前記ゲスト命令の実行によりメモリアクセス許可の違反が生じるかどうかを判定することであって、前記ゲスト命令は、前記ホストシステムによって公開されるゲスト仮想マシン内で実行され、前記ゲスト命令は、前記少なくとも１つのハードウェアプロセッサに、オペレータのオペランドへの適用の結果を求めるように命令し、前記ゲスト命令の前記違反のない実行は、前記結果の第１の場所への書き込みを生じさせる、判定することと、

前記ゲスト命令の実行により前記違反が生じるかどうかの判定に応答して、前記ゲスト命令の実行により前記違反が生じる場合に、前記結果を前記第１の場所とは異なる第２の場所に書き込むことであって、前記第２の場所はコンピュータセキュリティプログラムからアクセス可能である、書き込むことと、

前記結果の前記第２の場所への書き込みに応答して、前記ゲスト命令の実行から前記コンピュータセキュリティプログラムの実行に切り換えることであって、前記コンピュータセキュリティプログラムは、前記違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成された、切り替えることと

を行わせる、プログラム。

【請求項２１】

請求項２０に記載のプログラムを格納したコンピュータ可読記憶媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

関連出願

[0001]本出願は、２０１４年8月１８日に出願された、「Ｓｙｓｔｅｍｓ　Ａｎｄ　Ｍｅｔｈｏｄｓ　ｆｏｒ　Ｅｘｐｏｓｉｎｇ　Ａ　Ｃｕｒｒｅｎｔ　Ｐｒｏｃｅｓｓｏｒ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｕｐｏｎ　Ｅｘｉｔｉｎｇ　Ａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ（仮想マシンを終了する際に現在のプロセッサ命令を公開するためのシステムおよび方法）」という名称の米国仮特許出願第６２／０３８，４７６号の出願日の利益を主張するものであり、そのすべての内容は、参照により本明細書に組み込まれている。

【０００２】

[0002]本発明はコンピュータセキュリティに関し、詳細にはハードウェア仮想化構成でのコンピュータセキュリティ動作の実施に関する。

【背景技術】

【０００３】

[0003]マルウェアとしても知られている悪意のあるソフトウェアは、世界中で非常に多くのコンピュータシステムに影響を及ぼしている。マルウェアは、コンピュータウイルス、ワーム、ルートキットおよびスパイウェアなどの多くの形態で何百万ものコンピュータユーザを重大な危険にさらし、したがってコンピュータユーザは、とりわけ、データおよび機密情報の損失、個人情報の盗難、および生産性の損失の攻撃にさらされている。

【０００４】

[0004]現代の計算アプリケーションは、しばしばハードウェア仮想化技術を使用して、仮想マシン（ＶＭ）として知られている、物理コンピュータシステムと同じように様々な方式で振る舞う模擬的なコンピュータ環境を生成している。サーバ統合およびサービスとしてのインフラストラクチャ（Ｉｎｆｒａｓｔｒｕｃｔｕｒｅ－ａｓ－ａ－ｓｅｒｖｉｃｅ）などの用途では、いくつかの仮想マシンが同じコンピュータシステム上で同時に走り、それらの間でハードウェア資源を共有することができ、したがって投下資本および運転コストを低減することができる。個々の仮想マシンは、その独自のオペレーティングシステムおよび／またはソフトウェアを他の仮想マシンとは別に走らせることができる。マルウェアおよびスパイウェアなどのコンピュータセキュリティの脅威の止まることのない拡散のため、個々のこのような仮想マシンは保護を必要とする可能性がある。

【０００５】

[0005]いくつかのセキュリティ解決法は、保護されたＶＭ内で実行されているゲストプ

ロセスがメモリにアクセスする方式を監視して、潜在的な悪意のあるアクティビティを識別することによって仮想マシンを保護している。一例では、コンピュータセキュリティプログラムは、メモリの特定の領域、例えばゲストプロセスによって使用されるメモリの領域への書込みが試行され、またはその領域からのコードの実行が試行される際に、内部イベント（例えば例外イベントまたはＶＭ終了イベント）を生成するためのプロセッサを構成することができる。このようなプロセッサイベントは、典型的には、現在のスレッドの実行を中断し、また、プロセッサを、コンピュータセキュリティプログラムの一部を形成することができるイベントハンドラルーチンの実行に切り換える。コンピュータセキュリティプログラムは、そのようにして、マルウェアであることが疑われる方式によるメモリアクセスの試行を検出することができる。イベントを分析した後、コンピュータセキュリティプログラムは、イベントが生じた時点で実行されていたプロセッサ命令をエミュレートすることができ、また、実行を元のスレッドに戻すことができる。このような方法は、当分野ではトラップアンドエミュレートとして総称的に知られている。

【発明の概要】

【発明が解決しようとする課題】

【０００６】

[0006]従来のトラップアンドエミュレート方法は、相当な計算負荷をホストコンピュータシステムに課すことがあり、潜在的にユーザ体験および生産性に影響を及ぼす可能性がある。したがって仮想化環境に適した有効なコンピュータセキュリティシステムおよび方法を開発することは大いなる関心事である。

【課題を解決するための手段】

【０００７】

[0007]一態様によれば、ホストシステムは、仮想マシンおよびコンピュータセキュリティプログラムを実行するように構成された少なくとも１つのハードウェアプロセッサを備える。少なくとも１つのハードウェアプロセッサは、実行のためのゲスト命令を受け取るのに応答して、ゲスト命令の仮想マシン内での実行によりメモリアクセス許可の違反が生じるかどうかを判定するようにさらに構成される。少なくとも１つのハードウェアプロセッサは、ゲスト命令の実行により違反が生じるかどうかの判定に応答して、ゲスト命令の実行により違反が生じる場合に、ゲスト命令のオペレータをゲスト命令のオペランドへの適用の結果を求め、その結果を、コンピュータセキュリティプログラムからアクセス可能な所定の場所に書き込み、ゲスト命令の実行を中断するようにさらに構成される。少なくとも１つのハードウェアプロセッサは、ゲスト命令の実行の中断に応答して、コンピュータセキュリティプログラムの実行に切り換えるようにさらに構成され、コンピュータセキュリティプログラムは、違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成される。

【０００８】

[0008]別の態様によれば、ホストシステムをコンピュータセキュリティの脅威から保護する方法であって、実行のためのゲスト命令を受け取るのに応答して、ゲスト命令を実行することによりメモリアクセス許可の違反が生じるかどうかを、ホストシステムの少なくとも１つのハードウェアプロセッサを使用して判定するステップを含み、ゲスト命令は、ホストシステムによって公開される仮想マシン内で実行される。方法は、ゲスト命令が違反を生じさせるかどうかの決定に応答して、ゲスト命令を実行することにより違反が生じる場合に、少なくとも１つのハードウェアプロセッサを使用して、ゲスト命令のオペレータのゲスト命令のオペランドへの適用の結果を求めるステップと、少なくとも１つのハードウェアプロセッサを使用して、その結果を、コンピュータセキュリティプログラムからアクセス可能な所定の場所に書き込むステップと、ゲスト命令の実行を中断するステップとをさらに含む。方法は、ゲスト命令の実行の中断に応答して、コンピュータセキュリティプログラムの実行に切り換えるステップをさらに含み、コンピュータセキュリティプログラムは、違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成される。

【０００９】
[0009]別の態様によれば、ホストシステムの少なくとも１つのハードウェアプロセッサは、実行のためのゲスト命令を受け取るのに応答して、ゲスト命令を実行することによりメモリアクセス許可の違反が生じるかどうかを決定するように構成することができ、ゲスト命令は、ホストシステムによって公開されるゲスト仮想マシン内で実行される。少なくとも１つのハードウェアプロセッサは、ゲスト命令が違反を生じさせるかどうかの判定に応答して、ゲスト命令を実行することにより違反が生じる場合に、ゲスト命令のオペレータのゲスト命令のオペランドへの適用の結果を求め、その結果を、コンピュータセキュリティプログラムからアクセス可能な所定の場所に書き込み、ゲスト命令の実行を中断するようにさらに構成することができる。少なくとも１つのハードウェアプロセッサは、ゲスト命令の実行の中断に応答して、コンピュータセキュリティプログラムの実行に切り換えるようにさらに構成することができ、コンピュータセキュリティプログラムは、違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成される。
【００１０】
[0010]別の態様によれば、非一時的コンピュータ可読媒体は、ホストシステムの少なくとも１つのハードウェアプロセッサによって実行されると、ホストシステムに、メモリアクセス許可の違反がコンピュータセキュリティの脅威を示すかどうかを判定するように構成されたコンピュータセキュリティプログラムを形成させる命令を記憶する。少なくとも１つのハードウェアプロセッサは、実行のためのゲスト命令を受け取るのに応答して、ゲスト命令を実行することにより違反が生じるかどうかを判定するように構成することができ、ゲスト命令は、ホストシステムによって公開される仮想マシン内で実行される。少なくとも１つのハードウェアプロセッサは、ゲスト命令が違反を生じさせるかどうかの判定に応答して、ゲスト命令を実行することにより違反が生じる場合に、ゲスト命令のオペレータのゲスト命令のオペランドへの適用の結果を求め、その結果を、コンピュータセキュリティプログラムからアクセス可能な所定の場所に書き込み、ゲスト命令の実行を中断するようにさらに構成することができる。少なくとも１つのハードウェアプロセッサは、ゲスト命令の実行の中断に応答して、コンピュータセキュリティプログラムの実行に切り換えるようにさらに構成することができる。
【００１１】
[0011]本発明の上記態様および利点は、以下の詳細な説明を読み、また、図面を参照することにより、より良好に理解されるであろう。
【図面の簡単な説明】
【００１２】
【図１】[0012]本発明のいくつかの実施形態によるホストコンピュータシステムの例示的ハードウェア構成を示す図である。
【図２－Ａ】[0013]本発明のいくつかの実施形態による、ホストシステム上で実行するハイパーバイザによって公開される仮想マシンの例示的セット、および仮想マシンのセットを保護するコンピュータセキュリティモジュール（ＣＳＭ）を示す図である。
【図２－Ｂ】[0014]ＣＳＭが仮想マシンの下で実行され、また、例外ハンドラが保護された仮想マシン内で実行される、本発明の代替実施形態を示す図である。
【図２－Ｃ】[0015]ＣＳＭおよび例外ハンドラの両方が保護された仮想マシン内で実行する、本発明のさらに別の実施形態を示す図である。
【図３】[0016]本発明のいくつかの実施形態による、ゲスト仮想マシンとして公開された仮想化ハードウェアの例示的構成を示す図である。
【図４】[0017]本発明のいくつかの実施形態による、図２－Ａに示されるハードウェア仮想化構成における一組の例示的メモリアドレス変換を示す図である。
【図５】[0018]本発明のいくつかの実施形態によるプロセッサの例示的構成要素を示す図である。
【図６】[0019]本発明のいくつかの実施形態によるプロセッサの例示的中断イベントレジスタを示す図である。

【図７】[0020]ｘ８６命令セットの例示的プロセッサ命令のアセンブリ言語表現、およびその対応するマシンコード表現を示す図である。
【図８】[0021]本発明のいくつかの実施形態による、プロセッサ命令を実行するためにプロセッサによって実施されるステップの例示的シーケンスを示す図である。
【図９】[0022]本発明のいくつかの実施形態による、ゲスト仮想マシンを保護するためにコンピュータセキュリティモジュールによって実施されるステップの例示的シーケンスを示す図である。
【発明を実施するための形態】
【００１３】
[0023]以下の説明では、構造間の詳説されているすべての接続は、直接の動作接続であっても、あるいは中間構造を介した間接的な動作接続であってもよいことを理解されたい。一組の要素は、１つまたは複数の要素を含む。要素の詳説は、すべて、少なくとも１つの要素を意味することを理解されたい。複数の要素は、少なくとも２つの要素を含む。特に要求されない限り、説明されているすべての方法ステップは、必ずしも示されている特定の順序で実施される必要はない。第２の要素から導出される第１の要素（例えばデータ）は、第２の要素に等しい第１の要素、ならびに第２の要素を処理することによって生成される第１の要素、および任意選択で他のデータを包含する。パラメータに従って決定する（ｍａｋｉｎｇ ａ ｄｅｔｅｒｍｉｎｅ ｏｒ ａ ｄｅｃｉｓｉｏｎ；判定する、求める）ことは、パラメータに従って、また、任意選択で他のデータに従って決定することを包含する。特に明記されていない限り、何らかの量／データのインジケータは、その量／データ自体であっても、あるいはその量／データ自体とは異なるインジケータであってもよい。コンピュータプログラムは、タスクを実行する一連のプロセッサ命令である。本発明のいくつかの実施形態で説明されるコンピュータプログラムは、他のコンピュータプログラムの独立型ソフトウェアエンティティまたはサブエンティティ（例えばサブルーチン、ライブラリ）であってもよい。特に明記されていない限り、コンピュータセキュリティプログラムは、意図されないまたは許可されていないアクセス、修正または破壊から機器およびデータを保護するコンピュータプログラムである。特に明記されていない限り、プロセスは、アプリケーションまたはオペレーティングシステムの一部などのコンピュータプログラムのインスタンスであり、また、少なくとも実行スレッドおよび実行スレッドに割り当てられた仮想メモリ空間を有することを特徴とし、それぞれの仮想メモリ空間の内容は実行可能コードを含む。特に明記されていない限り、ページは、ホストシステムの物理メモリに個々にマッピングされ得る仮想メモリの最小単位を表している。「論理」という用語は、固定された機能または再構成可能な機能（例えばフィールドプログラマブルゲートアレイ回路）を有するハードウェア回路機構を包含するが、このような機能を汎用コンピュータ上でエミュレートするソフトウェアは包含しない。特に明記されていない限り、レジスタは、プロセッサと一体化された、すなわちプロセッサの一部を形成している記憶構成要素を表しており、また、ランダムアクセスメモリ（ＲＡＭ）とは全く異なる。コンピュータ可読媒体は、磁気記憶媒体、光記憶媒体および半導体記憶媒体などの非一時的媒体（例えばハードドライブ、光ディスク、フラッシュメモリ、ＤＲＡＭ）、ならびに導電ケーブルなどの通信リンクおよび光ファイバリンクを包含している。いくつかの実施形態によれば、本発明は、とりわけ、本明細書において説明される方法を実施するようにプログラムされたハードウェア（例えば１つまたは複数のプロセッサ）、ならびに本明細書において説明される方法を実施するための命令をエンコードするコンピュータ可読媒体を備えるコンピュータシステムを提供する。
【００１４】
[0024]以下の説明は、一例として本発明の実施形態を示したものであり、必ずしも制限するためのものではない。
[0025]図１は、本発明のいくつかの実施形態によるホストシステム１０の例示的ハードウェア構成を示したものである。ホストシステム１０は、企業サーバなどの法人組織の計算デバイス、またはパーソナルコンピュータ、タブレットコンピュータもしくはスマート

フォンなどのエンドユーザデバイスを表すことができる。他の例示的ホストシステムは、ＴＶ、ゲームコンソール、着用可能計算デバイス、またはメモリおよびプロセッサを有する任意の他の電子デバイスを含む。ホストシステム１０を使用して、とりわけ、ブラウザ、文書処理アプリケーションおよび電子通信（例えば電子メール、インスタントメッセージング）アプリケーションなどの一組のソフトウェアアプリケーションが実行され得る。いくつかの実施形態では、ホストシステム１０は、ハードウェアの仮想化を支援し、また、以下で示されるように一組の仮想マシンを公開するように構成される。

【００１５】

[0026]図１は、コンピュータシステムを示したもので、スマートフォンおよびタブレットコンピュータなどの他のホストシステムのハードウェア構成は異なる場合がある。システム１０は、プロセッサ１２、メモリユニット１４、一組の入力デバイス１６、一組の出力デバイス１８、一組の記憶デバイス２０、および一組のネットワークアダプタ２２を含む一組の物理デバイスを備えており、すべてがコントローラハブ２４によって接続されている。いくつかの実施形態では、プロセッサ１２は、一組の信号および／またはデータを使用して計算および／または論理演算を実行するように構成された物理デバイス（例えば半導体基板の上に形成されたマルチコア集積回路）を備えている。いくつかの実施形態では、このような論理演算は、一連のプロセッサ命令（例えばマシンコードまたは他のタイプのソフトウェア）の形態でプロセッサ１２に引き渡される。本発明のいくつかの実施形態は、従来のプロセッサの構造および機能に変更を導入し、それぞれの変更により、ハードウェア仮想化構成におけるプロセッサ１２のより有効な動作が可能になる。

【００１６】

[0027]メモリユニット１４は、命令を実行する過程でプロセッサ１２によってアクセスまたは生成されるデータ／信号を記憶する揮発性コンピュータ可読媒体（例えばＲＡＭ）を備えることができる。入力デバイス１６は、とりわけ、コンピュータキーボード、マウスおよびマイクロホンを含むことができ、これらは、ホストシステム１０へのデータおよび／または命令のユーザによる導入を可能にするそれぞれのハードウェアインタフェースおよび／またはアダプタを含む。出力デバイス１８は、とりわけ、モニタなどの表示デバイスおよびスピーカを含むことができ、また、ホストシステム１０によるユーザへのデータ通信を可能にする、グラフィックカードなどのハードウェアインタフェース／アダプタを含むことができる。いくつかの実施形態では、入力デバイス１６および出力デバイス１８は、タッチスクリーンデバイスの場合と同様、１つの共通のハードウェアを共有することができる。記憶デバイス２０は、不揮発性記憶装置によるプロセッサ命令および／またはデータの読出しおよび書込みを可能にするコンピュータ可読媒体を含む。例示的記憶デバイス２０は、磁気ディスク、光ディスクおよびフラッシュメモリデバイス、ならびにＣＤおよび／またはＤＶＤディスクおよびドライバなどの取外し可能媒体を含む。一組のネットワークアダプタ２２は、コンピュータネットワークおよび／または他のデバイス／コンピュータシステムへのホストシステム１０の接続を可能にしている。コントローラハブ２４は、複数のシステム、周辺装置および／もしくはチップセットバス、ならびに／またはプロセッサ１２とデバイス１４、１６、１８、２０、および２２との間の通信を可能にするあらゆる他の回路機構を総称的に表している。例えばコントローラハブ２４は、とりわけ、メモリ管理ユニット（ＭＭＵ）、入力／出力（Ｉ／Ｏ）コントローラ、および割込みコントローラを含むことができる。別の例では、コントローラハブ２４は、プロセッサ１２をメモリ１４に接続するノースブリッジ、ならびに／またはプロセッサ１２をデバイス１６、１８、２０、および２２に接続するサウスブリッジを備えることができる。いくつかの実施形態では、コントローラハブ（ＭＭＵなどの）の部分はプロセッサ１２と統合されることが可能であり、すなわち共通基板をプロセッサ１２と共有することができる。

【００１７】

[0028]図２－Ａは、本発明のいくつかの実施形態による例示的機能構成を示したもので、ホストシステム１０は、ハードウェア仮想化技術を使用して、ハイパーバイザ５０によって公開される一組のゲスト仮想マシン５２ａ～ｂを動作させる。このような構成は、と

りわけ、クラウドコンピューティングおよびサーバ統合などの用途で一般的である。仮想マシン（ＶＭ）は、当分野では、実際の物理マシン／コンピュータシステムの抽象化、例えばソフトウェアエミュレーションとして知られており、ＶＭは、オペレーティングシステムおよび他のソフトウェアを走らせることができる。いくつかの実施形態では、ハイパーバイザ５０は、仮想プロセッサおよび仮想コントローラハブなどの複数の仮想化デバイスを生成または可能にするように構成され、また、ホストシステム１０の現実の物理デバイスの代わりにこのような仮想化デバイスをソフトウェアに提供するように構成されたソフトウェアを含む。ハイパーバイザ５０のこのような動作は、当分野では、仮想マシンの公開として広く知られている。いくつかの実施形態では、ハイパーバイザ５０は、複数の仮想マシンによるホストシステム１０のハードウェア資源の多重化（共有）を可能にする。ハイパーバイザ５０は、さらに、個々のゲストＶＭ５２ａ～ｂが独立して動作し、また、ホストシステム１０上で同時に実行している他のＶＭ実行を認識しないように、このような多重化を管理することができる。一般的なハイパーバイザの例は、とりわけ、ＶＭｗａｒｅ社のＶＭｗａｒｅ　ｖＳｐｈｅｒｅ（商標）、およびオープンソースのＸｅｎハイパーバイザを含む。

【００１８】

[0029]個々のＶＭ５２ａ～ｂは、それぞれゲストオペレーティングシステム（ＯＳ）５４ａ～ｂを実行することができる。一組の例示的アプリケーション５６ａ～ｄは、とりわけ、文書処理、画像処理、メディアプレーヤ、データベース、カレンダ、個人連絡先管理、ブラウザ、ゲーミング、音声通信、データ通信、およびマルウェア防止アプリケーションなどの任意のソフトウェアアプリケーションを総称的に表している。オペレーティングシステム５４ａ～ｂは、とりわけ、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ（登録商標）、ＭａｃＯＳ（登録商標）、Ｌｉｎｕｘ（登録商標）、ｉＯＳ（登録商標）、またはＡｎｄｒｏｉｄ（商標）などの広く利用可能な任意のオペレーティングシステムを含むことができる。個々のＯＳ５４ａ～ｂは、それぞれのＶＭおよびそれぞれのＶＭの仮想化ハードウェアデバイス内で実行するアプリケーション間のインタフェースを提供する。以下の説明では、仮想マシンの仮想プロセッサ上で実行するソフトウェアは、それぞれの仮想マシン内で実行する、と言われる。例えば図２－Ａの例では、アプリケーション５６ａ～ｂはゲストＶＭ５２ａ内で実行する、と言われ、一方、アプリケーション５６ｃ～ｄはゲストＶＭ５２ｂ内で実行する、と言われる。それとは対照的に、ハイパーバイザ５０はゲストＶＭ５２ａ～ｂの外部、すなわち下で実行する、と言われる。

【００１９】

[0030]図３は、ハイパーバイザ５０によって公開される仮想マシン５２の例示的構成を示したものである。ＶＭ５２は、図２－Ａの任意のＶＭ５２ａ～ｂを表すことができる。ＶＭ５２は、仮想化プロセッサ１１２、仮想化メモリユニット１１４、仮想化入力デバイス１１６、仮想化出力デバイス１１８、仮想化記憶装置１２０、仮想化ネットワークアダプタ１２２、および仮想化コントローラハブ１２４を含む。仮想化プロセッサ１１２は、プロセッサ１２の機能のうちの少なくともいくつかのエミュレーションを含み、また、オペレーティングシステムおよび他のアプリケーションなどのソフトウェアの一部を形成するプロセッサ命令を受け取って実行するように構成される。プロセッサ１１２を使用して実行するソフトウェアは、仮想マシン５２内で実行すると見なされる。いくつかの実施形態では、仮想化メモリユニット１１４は、仮想化プロセッサ１１２によって使用されるデータを記憶し、かつ、取り出すためのアドレス指定可能空間を備えている。他の仮想化デバイス（例えば仮想化入力、出力、記憶装置、等々）は、ホストシステム１０のそれぞれの物理デバイスの機能のうちの少なくともいくつかをエミュレートする。仮想化プロセッサ１１２は、あたかも対応する物理デバイスと対話するかのように、このような仮想化デバイスと対話するように構成され得る。例えばＶＭ５２内で実行するソフトウェアは、仮想化ネットワークアダプタ１２２を介してネットワークトラフィックを送信および／または受信することができる。いくつかの実施形態では、ハイパーバイザ５０は、仮想化デバイスのサブセットのみ（例えば仮想化プロセッサ１１２、仮想化メモリ１１４、およびハ

ブ１２４の部分のみ）をＶＭ５２に公開することができる。また、ハイパーバイザ５０は、選択されたＶＭに、ホストシステム１０のいくつかのハードウェアデバイスの排他的な使用を与えることも可能である。１つのそのような例では、ＶＭ５２ａ（図２－Ａ）は、入力デバイス１６および出力デバイス１８の排他的な使用を有することができるが、仮想化ネットワークアダプタは持たない。一方、ＶＭ５２ｂは、ネットワークアダプタ２２の排他的な使用を有することができる。このような構成は、例えばＩｎｔｅｌ（登録商標）のＶＴ－ｄ（登録商標）技術を使用して実現され得る。

【００２０】

[0031]現代のプロセッサは、当分野では保護リングとしても知られている、プロセッサ特権レベルの階層を実現している。個々のこのようなリングすなわちレベルは、それぞれのリング内で実行するソフトウェアがその実行を許可される一組のアクションおよび／またはプロセッサ命令を特徴とする。例示的な特権レベル／リングは、ユーザモード（リング３）およびカーネルモード（リング０）を含む。ハードウェアの仮想化を支援するように構成されるいくつかのホストシステムは、プロセッサ特権が最も高い追加リングを含むことができる（例えばリング－１、ルートモードまたはＩｎｔｅｌ（登録商標）プラットフォーム上のＶＭＸｒｏｏｔ）。いくつかの実施形態では、ハイパーバイザ５０は、最も特権の高いレベル（リング－１）でプロセッサ１２の制御を獲得し、したがってホストシステム１０上で実行する他のソフトウェアに仮想マシンとして公開されるハードウェア仮想化プラットフォームを生成する。図２－ＡのゲストＯＳ５４ａなどのオペレーティングシステムは、典型的にはプロセッサ特権がハイパーバイザ５０より劣るそれぞれのＶＭの仮想環境内（例えばリング０すなわちカーネルモード）で実行する。５６ａ～ｂなどの共通ユーザアプリケーションは、典型的には、ＯＳ３４ａより劣るプロセッサ特権で（例えばリング３すなわちユーザモードで）実行する。アプリケーション５６ａ～ｂのいくつかの部分は、カーネル特権レベルで実行することができ、一方、ＯＳ３４ａのいくつかの部分は、ユーザモード（リング３）で実行することができる。ソフトウェアオブジェクトが、その割り当てられた保護リングによって許容される特権より高いプロセッサ特権を必要とするアクションまたは命令の実行を試行する場合、その試行は、典型的には、それぞれのアクションを実行するのに十分な特権を有するリング内で実行するエンティティ（例えばハンドラルーチン）へプロセッサ１２の制御を移行する、例外または障害などのプロセッサイベントを生成する。

【００２１】

[0032]詳細には、ゲストＶＭの中から特定のアクションを実施する試行、または特定の命令を実行する試行は、本明細書において総称的にＶＭ中断イベントと呼ばれる特定のカテゴリのプロセッサイベントをトリガすることができる。いくつかの実施形態では、ＶＭ中断イベントは、ゲストＶＭ内の現在のスレッドの実行を中断し、かつ、プロセッサ１２をハンドラルーチンの実行に切り換える。例示的ＶＭ中断イベントは、とりわけ、ＶＭ終了イベント（例えばＩｎｔｅｌ（登録商標）プラットフォーム上のＶＭＥｘｉｔ）および仮想化例外（例えばＩｎｔｅｌ（登録商標）プラットフォーム上の＃ＶＥ）を含む。ＶＭ終了イベントは、プロセッサ１２を、典型的にはハイパーバイザ５０のレベルであるそれぞれのゲストＶＭの外部のハンドラルーチンの実行に切り換える。仮想化例外は、それぞれのＶＭを終了する代わりに、プロセッサ１２をそれぞれのゲストＶＭ内のハンドラルーチンの実行に切り換えることができる。

【００２２】

[0033]ＶＭ中断イベントをトリガする例示的命令は、Ｉｎｔｅｌ（登録商標）プラットフォームのＶＭＣＡＬＬを含む。また、ＶＭ中断イベントは、メモリアクセス違反などの他のイベントによってもトリガされ得る。１つのそのような例では、ＶＭ内で実行しているソフトウェアオブジェクトが、非書込み可能とマークされたメモリのセクションへの書込みを試行するか、または非実行可能とマークされたメモリのセクションからのコードの実行を試行すると、プロセッサ１２は、ＶＭ終了イベントを生成することができる。このようなＶＭ切換え機構は、例えばコンピュータセキュリティプログラムによる、それぞれ

のＶＭの外部からの仮想マシンの保護を可能にする。コンピュータセキュリティプログラムは、ＶＭ内で走っているソフトウェアによって実施される、セキュリティの脅威を示し得る特定のアクションに応答して、生じているＶＭ終了イベントを傍受することができる。コンピュータセキュリティプログラムは、次に、このようなアクションを阻止し、および／または、可能性としてはＶＭ内ソフトウェアの知識を必要とすることなく、このようなアクションをさらに分析することができる。このような構成は、コンピュータセキュリティを実質的に強化することができる。

【００２３】

[0034]いくつかの実施形態（例えば図２－Ａ）では、ハイパーバイザ５０は、とりわけ、このようなコンピュータセキュリティ動作を実施するように構成されたコンピュータセキュリティモジュール（ＣＳＭ）６０を含む。モジュール６０は、ハイパーバイザ５０に組み込まれ得るか（例えばライブラリとして）、またはハイパーバイザ５０とは異なりハイパーバイザ５０から独立しているが、ハイパーバイザ５０の特権レベルで実行するコンピュータプログラムとして引き渡され得る。単一のモジュール６０が、ホストシステム１０上で実行する複数のゲストＶＭを保護するように構成され得る。モジュール６０によって実行されるセキュリティ動作は、ゲストＶＭ内で実行するプロセスによって実施されるアクション（例えばＯＳの特定の機能の呼出し、ＯＳのレジストリのアクセス、遠隔の場所からのファイルのダウンロード、ファイルへのデータの書込み、等々）の検出を含むことができる。モジュール６０の他のセキュリティ動作は、ゲストＶＭ内で実行するソフトウェアオブジェクトの一部を含むメモリセクションのアドレスの決定、それぞれのメモリセクションへのアクセス、およびそれぞれのメモリセクションに記憶されている内容の分析を含むことができる。セキュリティ動作の他の例は、このようなメモリセクションへのアクセスの傍受および／または制限、例えば保護されたプロセスに属するコードまたはデータの重ね書きの防止、および特定のメモリページに記憶されているコードの実行の防止を含む。いくつかの実施形態では、ＣＳＭ６０は、ゲストＶＭ５２ａ～ｂ内で生じるＶＭ終了イベントを傍受するように構成されたＶＭ終了イベントハンドラ６１を含む。代替実施形態では、ハンドラ６１は、ＶＭ終了イベントを傍受し、かつ、生じた個々のＶＭ終了の理由および／またはタイプを決定した後に、制御をＣＳＭ６０に選択的に移行する、ＣＳＭ６０とは別のハイパーバイザ５０の全く異なるモジュール（例えばライブラリ）であってもよい。

【００２４】

[0035]図２－Ｂは、コンピュータセキュリティモジュール６０がそれぞれのＶＭの外部からゲストＶＭ５２ｃを保護する代替実施形態を示したものである。このような実施形態では、プロセッサ１２は、メモリアクセス違反が生じると仮想化例外を生成するように構成され得る（図２－Ａに関連して上で説明したＶＭ終了イベントの代わりに）。図２－Ｂの例示的実施形態では、仮想化例外ハンドラ６３は、ＶＭ５２ｃ内で、例えばオペレーティングシステム５４ｃの特権レベルで実行し、また、仮想化例外を傍受し、ＣＳＭ６０とインタフェースするように構成される。

【００２５】

[0036]ハンドラ６３とＣＳＭ６０の間の通信は、当分野で知られている任意のプロセス間通信方法に従って進行することができる。データを保護されたＶＭ内からハイパーバイザ５０のレベルへ伝送するために、ハンドラ６３のいくつかの実施形態は、専用の命令（例えばＩｎｔｅｌ（登録商標）プラットフォームのＶＭＣＡＬＬ）を使用して、プロセッサ１２の制御をそれぞれのＶＭからハイパーバイザ５０へ移行する。伝送されるデータは、例外ハンドラ６３によって、ＣＳＭ６０と共有されるメモリの所定のセクションに置かれ得る。データをハンドラ６３に伝送するために、ＣＳＭ６０のいくつかの実施形態は、ハンドラ６３によって処理される割込みをＶＭ５２ｃにかけることができる。それぞれのデータは、上で説明した共有メモリセクションを介して再度伝送され得る。

【００２６】

[0037]図２－Ｃに示されているさらに別の実施形態では、ＣＳＭ６０およびハンドラ６

３の両方が、保護されたＶＭ内で、例えばカーネルモード（リング０）で実行する。このような実施形態は、仮想化例外を使用してメモリアクセス違反を検出することも可能である。構成２－Ａ－Ｂ－Ｃ間の決定は、性能とセキュリティの間のトレードオフを評価することを含み得る。ＶＭ終了イベントは、計算の点で比較的コストがかかり、典型的には、個々の終了および再開始サイクルでのメモリへの／メモリからの大型データ構造のロードおよび／またはアンロードが必要である。したがって２－Ａなどの構成は、２－Ｂ－Ｃなどの構成よりも、イベントを傍受するためにより多くの計算を必要とし得る。一方、ＣＳＭ６０およびハンドラ６１～６３などの重要なセキュリティ構成要素を保護されたＶＭの外部に維持することは（例２－Ａ－Ｂの場合のように）、それぞれのＶＭ内で実行するマルウェアがこのような構成要素の動作を妨害することはより困難であり得るため、セキュリティを強化し得る。

【００２７】

[0038]図２－Ａ－Ｂに示されている構成で（すなわちそれぞれのＶＭの外部から）ゲストＶＭを保護することができるようにするために、ＣＳＭ６０のいくつかの実施形態は、プロセッサ１２のアドレス変換データ構造および／またはアドレス変換機構を使用する。仮想マシンは、典型的には、当分野でゲスト物理メモリとして知られている仮想化物理メモリを使用して動作する（例えば図３のメモリ１１４を参照されたい）。仮想化物理メモリは、例えばアドレスの連続空間として実際の物理メモリ１４の抽象表現を含み、これは一般にゲスト物理アドレス（ＧＰＡ）と呼ばれる。個々のこのようなアドレス空間は、ゲストＶＭに固有に結び付けられ、前記アドレス空間の部分は、物理メモリ１４および／または物理記憶デバイス２０のセクションにマッピングされる。仮想化を支援するように構成されるシステムでは、このようなマッピングは、典型的には、プロセッサ１２によって制御される、第２のレベルのアドレス変換（ＳＬＡＴ）として知られている、ハードウェアで高速化された専用データ構造および機構を使用して達成される。一般的なＳＬＡＴ実施態様は、Ｉｎｔｅｌ（登録商標）プラットフォーム上の拡張ページテーブル（ＥＰＴ）、およびＡＭＤ（登録商標）プラットフォーム上の高速仮想化指標付け（ＲＶＩ）／入れ子ページテーブル（ＮＰＴ）を含む。このようなシステムでは、仮想化物理メモリは、当分野でページとして知られている単位で分割されることが可能であり、ページは、ＥＰＴ／ＮＰＴなどの機構を介して物理メモリに個々にマッピングされる仮想化物理メモリの最小単位を表し、すなわち物理メモリと仮想化物理メモリの間のマッピングは、ページ単位の粒度で実施される。すべてのページは、典型的には、所定のサイズ、例えば４キロバイト、２メガバイト、等々を有している。仮想化物理メモリのページへの分割は、通常、ハイパーバイザ５０によって構成される。いくつかの実施形態では、ハイパーバイザ５０もＳＬＡＴ構造を構成し、したがって物理メモリと仮想化物理メモリの間のアドレス変換を構成する。このようなアドレス変換は、当分野ではゲスト物理－ホスト物理（ＧＰＡ－ｔｏ－ＨＰＡ）変換として知られている。

【００２８】

[0039]いくつかの実施形態では、ＶＭ内で実行するオペレーティングシステムは、それぞれのＶＭ内で実行するプロセス毎に仮想メモリ空間をセットアップし、前記仮想メモリ空間は、物理メモリの抽象化を表している。プロセス仮想メモリは、典型的には、当分野で一般にゲスト仮想アドレス（ＧＶＡ）またはゲスト線形アドレス（ＧＬＡ）として知られている、アドレスの連続空間を含む。いくつかの実施形態では、プロセス仮想メモリ空間も同じくページに分割され、このようなページは、ＯＳによってそれぞれのＶＭの仮想化物理メモリに個々にマッピングされる仮想メモリの最小単位を表し、すなわち仮想化物理メモリへの仮想メモリのマッピングは、ページ単位の粒度で実施される。ＯＳは、ゲスト仮想－ゲスト物理、すなわちＧＶＡ－ＧＰＡアドレス変換を実施するためにそれぞれのＶＭの仮想化プロセッサによって使用される、ページテーブルなどの専用データ構造を構成することができる。

【００２９】

[0040]図４は、図２－Ａの実施形態における例示的メモリアドレス変換を示したもので

ある。ハイパーバイザ５０による公開後、ゲストＶＭ５２ａは、仮想化物理メモリ空間１１４ａをその独自の物理メモリ空間として見る。ゲストＶＭ５２ａ内で実行するプロセスには、ゲストＯＳ５４ａによって仮想メモリ空間２１４ａが割り当てられる。ゲスト仮想アドレス６２へのメモリアクセスをプロセスが試行すると、ＧＶＡ６２は、ゲストＶＭ５２ａの（仮想化）ＭＭＵによって仮想化物理メモリ空間１１４ａ内のゲスト物理アドレス６４に変換される。ＧＶＡ－ＧＰＡ変換７０ａは、例えばゲストＯＳ３４ａによって構成され、かつ、制御されるページテーブルに従って進行することができる。ＧＰＡ６４は、ＭＭＵによってホストシステム１０の物理メモリ１４内のホスト物理アドレス（ＨＰＡ）６６にさらにマッピングされる。ＧＰＡ－ＨＰＡ変換７０ｂは、例えばハイパーバイザ５０によって構成されるＳＬＡＴ構造に従って進行することができる。

【００３０】

[0041]ゲストＶＭ５２ａ～ｂの下で実行する個々のプロセスには、典型的には、当分野でホスト仮想アドレス（ＨＶＡ）として知られるものを介してアドレス指定できる仮想メモリ空間が割り当てられる。図４の例では、ハイパーバイザ５０は、コンピュータセキュリティモジュール６０のための仮想メモリ空間２１４ｂをセットアップしている。すると、ＣＳＭ６０は、ＨＶＡ６８を介してＨＰＡ６６を参照することができる。モジュール６０が例えばライブラリとしてハイパーバイザ５０内に統合される場合、メモリ空間２１４ｂは、ハイパーバイザ５０の仮想メモリ空間と一致し得る。このような空間を管理するために、ハイパーバイザ５０は、変換７０ｃなどのＨＶＡ－ＨＰＡ変換を実施するためにＭＭＵによって使用される専用データ構造および機構（例えばページテーブル）を構成することができる。

【００３１】

[0042]いくつかの実施形態では、ハイパーバイザ５０および／またはＣＳＭ６０は、物理メモリページのサブセットの各々に対してアクセス許可を設定することができる。このようなメモリページは、例えば、ＯＳおよび／またはマルウェア防止ルーチンのプロセスなど、保護されたＶＭ内で実行する特定の重要なゲストプロセスによって使用され得る。アクセス許可は、例えばそれぞれのページがそのページから読み出され、そのページに書き込まれ得るかどうか、また、ソフトウェアがそれぞれのページからのコードの実行を許可されるかどうかを示す。アクセス許可は、例えばそれぞれのメモリページを表すＳＬＡＴエントリの一部として示され得る。いくつかのホストシステムは、サブページ単位の粒度でアクセス許可の設定を許可することができる。

【００３２】

[0043]ハイパーバイザ５０および／またはＣＳＭ６０は、さらに、ゲストＶＭ内で実行しているソフトウェアがアクセス許可を違反するようなメモリアクセス（例えば非書込み可能とマークされているメモリページへの書込み）を試行すると、ＶＭ中断イベントを生成するようにプロセッサ１２を構成することができる。このような試行は、本明細書においてはメモリアクセス違反と呼ばれる。それぞれのＶＭ中断イベントは、図２－Ａなどの構成におけるＶＭ終了イベント、および図２－Ｂ－Ｃなどの構成における仮想化例外であってもよい。

【００３３】

[0044]本発明のいくつかの実施形態は、ハードウェア仮想化構成においてプロセッサがより有効に機能することができるように、従来のハードウェアプロセッサの構造および機能に変更を導入している。図５は、本発明のいくつかの実施形態によるプロセッサ１２の例示的ハードウェア構成要素を示したものである。示されている構成要素は、説明されている機能を実施する一般化デバイスを意味しており、構造的詳細は、実施態様間で大きく変更することが可能である。例えば示されている個々の構成要素は、相互接続された複数のサブシステムを備えることができ、これらは、必ずしも互いに物理的に近接している必要はない。示されている構成要素は、すべてを網羅したものではなく、プロセッサ１２は、多くの他の構成要素（例えばスケジューラ、割込みコントローラ、様々なキャッシュ、等々）を含むことができ、それらは、描写を簡潔にするために図５から省略されている。

【００３４】
[0045]プロセッサ１２は、プロセッサパイプラインの様々なステージを実行するように構成された論理／回路機構を含むことができる。例えば命令デコーダ３０は命令デコード操作を実施し、この操作は、一組の基本プロセッサ動作および／またはマイクロオペレーションへの個々のプロセッサ命令の変換を含むことができる。デコーダ３０に接続された一組の実行ユニット３６は、パイプラインの実行ステージを実施することができる。例示的実行ユニット３６は、とりわけ、演算論理ユニット（ＡＬＵ）および浮動小数点ユニット（ＦＰＵ）を含む。
【００３５】
[0046]いくつかの実施形態では、命令のためのパイプラインの実行ステージは、それぞれの命令のオペレータをそれぞれの命令のオペランドに適用した結果を決定することを含む。このような結果は、とりわけ、メモリアドレス、メモリアドレス、またはプロセッサレジスタ（例えばＡＸなどの汎用レジスタ、モデル特化レジスタ－ＭＳＲ、ＥＦＬＡＧＳなどの制御レジスタ、または同じく記述子キャッシュとして知られているｘ８６セグメントレジスタの隠れた部分などの隠れレジスタ）にコミットされる値、命令ポインタ（例えばＲＩＰ）の値、およびスタックポインタ（例えばＲＳＰ）の値を含むことができる。
【００３６】
[0047]命令のオペランドは、命令文の中で明示的であっても、あるいは暗示的であってもよい。オペランドが暗示的である例示的ｘ８６命令は、プロセッサのＥＦＬＡＧＳ制御レジスタのキャリーフラグを１に設定するＳＴＣ命令である。いくつかの実施形態では、レジスタＥＦＬＡＧＳおよび値１は、（暗示的な）オペランドとして解釈されるが、それらはＳＴＣ命令文には明示的に出現しない。
【００３７】
[0048]コミットユニット３８は、パイプラインのコミットステージを実施することができ、すなわち実行ユニット３６の出力をメモリ１４に記憶することができ、および／または特定のプロセッサレジスタの内容を更新して、実行ステージによって生成される変化／結果を反映することができる。コミットユニット３８は、当分野では退却ユニットとして知られている論理モジュールを備えることができる。
【００３８】
[0049]デコーダ３０および実行ユニット３６に接続されたメモリアクセスモジュール３４は、例えばメモリから命令をフェッチするため、メモリからデータをロードするため、およびプロセッサ命令の実行の結果をメモリに記憶するために、メモリ１４とインタフェースするように構成された論理を含む。いくつかの実施形態では、メモリアクセスモジュールは、メモリアクセスに必要な仮想－物理アドレス変換を実施するように構成されたＭＭＵを備えている。
【００３９】
[0050]現代のプロセッサは、典型的には、プロセッサ命令のｏｕｔ－ｏｆ－ｏｒｄｅｒ実行および／または投機的実行をサポートする。このようなシステムでは、同じ実行ユニット３６によって複数の命令が同時にフェッチされ、デコードされ、実行される。このような実行の結果は、次に、それぞれのコンピュータプログラムの意図されたフローを保存するために出現順に（ｉｎ－ｏｒｄｅｒ）コミットされる。このような構成は、例えばプロセッサ１２の性能を改善するために分岐予測アルゴリズムと共に使用される。ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行のために構成されるいくつかの実施形態では、プロセッサ１２は、デコーダ３０および実行ユニット３６に結合されたディスパッチャユニット３２、および実行ユニット３６およびコミットユニット３８に結合されたレジスタファイル４０をさらに備えることができる。ディスパッチャユニット３６は、実行のために個々のマイクロオペレーションをスケジュールすることができ、また、実行およびコミットの順序を制御するために、そのそれぞれの命令を使用して、個々のマイクロオペレーションに関連するマッピングを維持することができる。レジスタファイル４０は、例えばリオーダバッファとして編成された内部プロセッサレジスタのアレイを備えている。レジスタファイル４０は

、ディスパッチャユニット３６による、スケジュールされた個々のマイクロオペレーションへのファイル４０のレジスタの行の結合を可能にする論理をさらに備えることができ、これは当分野でレジスタリネーミングとして知られている動作である。このような構成では、レジスタの個々のこのような行は、例えばプロセッサ１２の汎用レジスタおよび／または状態レジスタの値を保持することができ、前記値は、特定のプロセッサ命令の実行の中間ステージに対応する。

【００４０】

[0051]プロセッサ１２は、仮想マシンの状態データを管理するように構成された仮想マシン制御ユニット３８をさらに含むことができる。いくつかの実施形態では、仮想マシン状態オブジェクト（ＶＭＳＯ）は、ホストシステム１０上に公開された個々の仮想化プロセッサの現在の状態を示すためにプロセッサ１２によって内部で使用されるデータ構造を備えている。例示的ＶＭＳＯは、Ｉｎｔｅｌ（登録商標）プラットフォームの仮想マシン制御構造（ＶＭＣＳ）、およびＡＭＤ（登録商標）プラットフォームの仮想マシン制御ブロック（ＶＭＣＢ）を含む。ＶＭＳＯは、典型的には、ハイパーバイザ５０によって、個々の仮想マシンを公開する部分としてセットアップされる。いくつかの実施形態では、プロセッサ１２は、ソフトウェアがメモリアドレスまたはポインタ（例えばＩｎｔｅｌ（登録商標）プラットフォーム上のＶＭＣＳポインタ）を使用して特定のＶＭＳＯを参照することができるように、メモリ内のある領域を個々のＶＭＳＯと結合する。

【００４１】

[0052]個々のＶＭＳＯは、ゲスト状態領域およびホスト状態領域を備えることができ、ゲスト状態領域は、それぞれのゲストＶＭのＣＰＵ状態を保持し、ホスト状態領域は、ハイパーバイザ５０の現在の状態を記憶する。いくつかの実施形態では、ＶＭＳＯのゲスト状態領域は、とりわけ、制御レジスタ（例えばＣＲ０、ＣＲ３、等々）、命令ポインタ（例えばＲＩＰ）、汎用レジスタ（例えばＥＡＸ、ＥＣＸ、等々）、およびそれぞれのゲストＶＭの仮想プロセッサの状態レジスタ（例えばＥＦＬＡＧＳ）の内容を含む。ＶＭＳＯのホスト状態領域は、それぞれのゲストＶＭのためのＧＰＡ－ＨＰＡアドレス変換のために構成されたＳＬＡＴデータ構造に対するポインタ（例えばＩｎｔｅｌ（登録商標）プラットフォーム上のＥＰＴポインタ）を含むことができる。

【００４２】

[0053]いくつかの実施形態では、プロセッサ１２は、ＶＭＳＯの一部を専用内部レジスタ／キャッシュに記憶することができ、一方、それぞれのＶＭＳＯの他の部分は、メモリに存在させることができる。任意の所与の時に、最大でも１つのＶＭＳＯ（本明細書においては現在のＶＭＳＯと呼ばれる）がプロセッサにロードされ得、プロセッサ１２の制御を現在有する仮想マシンを識別する。現代のプロセッサは、通例、マルチスレッドのために構成されている。このような構成では、物理プロセッサ１２は複数のコアを動作させることができ、個々のコアは複数の論理プロセッサをさらに備え、個々の論理プロセッサは、他の論理プロセッサとは独立して、かつ、他の論理プロセッサと同時に実行スレッドを処理することができる。複数の論理プロセッサは、いくつかのハードウェア資源、例えば共通ＭＭＵを共有することができる。マルチスレッド実施形態では、全く異なるＶＭＳＯが個々の全く異なる論理プロセッサ上にロードされ得る。

【００４３】

[0054]プロセッサ１２がそれぞれのＶＭの実行からハイパーバイザ５０の実行に切り換える場合（例えばＶＭ終了に応じて）、プロセッサ１２は、それぞれのＶＭの状態を現在のＶＭＳＯのゲスト状態領域に保存することができる。プロセッサ１２が第１のＶＭの実行から第２のＶＭの実行に切り換えると、第１のＶＭに結合されたＶＭＳＯがアンロードされ、また、第２のＶＭに結合されたＶＭＳＯがプロセッサにロードされ、第２のＶＭＳＯが現在のＶＭＳＯになる。いくつかの実施形態では、このようなＶＭＳＯデータのプロセッサ１２へのロード／プロセッサ１２からのアンロードは、仮想マシン制御モジュール３８によって実施される。モジュール３８は、メモリ１４からのＶＭＳＯデータの取り出しおよび／またはメモリ１４へのＶＭＳＯデータの保存をさらに実行することができる。

【００４４】
[0055]いくつかの実施形態では、プロセッサ１２は、実行ユニット３６および／またはコミットユニット３８に接続され、かつ、ゲスト命令と結合した命令特化データを記憶するように構成された中断イベントレジスタ４４をさらに備えており、前記ゲスト命令を実行するとＶＭ中断イベントが生じる（例えばＶＭ終了または仮想化例外）。いくつかの実施形態では、中断イベントレジスタ４４は公開されたレジスタであり、ホストシステム１０上で実行しているソフトウェアからアクセスすることができ、すなわちレジスタ４４に記憶されているデータは、セキュリティモジュール６０などのソフトウェアによって読み出され得る。１つのこのような例では、中断イベントレジスタ４４は、プロセッサ１２のモデル特化レジスタ（ＭＳＲ）を含む。いくつかの実施形態は、レジスタ４４へのアクセスを、プロセッサ特権（例えばリング１またはルートモードのみ）またはオブジェクトタイプ（例えばドライバのみ）などの基準に従って選択されたソフトウェアオブジェクトのサブセットに制限することができる。いくつかの実施形態は、レジスタ４４へのソフトウェアアクセスを操作のサブセットのみに制限することができる（例えば読出しのみ）。
【００４５】
[0056]図６は、本発明のいくつかの実施形態による中断イベントレジスタ４４のフィールドの例示的セットを示したものである。レジスタ４４は、逆アセンブリフィールド４６ａおよび実行結果フィールド４６ｂを含むことができる。逆アセンブリフィールド４６ａは、それぞれのゲスト命令を逆アセンブルすることによって得られるデータを記憶することができる。
【００４６】
[0057]図７は、例示的なＩｎｔｅｌ（登録商標）ｘ８６プロセッサ命令のアセンブリ言語表現４５を示したものである。示されている命令は、メモリの（仮想）アドレスＥＢＸ＋４＊ＥＣＸ＋０ｘ０２に記憶されている内容を、レジスタＡＸに現在記憶されている値だけ増分するようにプロセッサに命令する。それぞれの命令は、マシンコード表現４７としてメモリの中に表現されており、表現４５と４７の間の変換は、典型的にはコンパイラまたはアセンブラによって実施される。ｘ８６命令のマシンコード表現は、一連のエンコードフィールド４９を含む一般化形態４８を有しており、このようなエンコードフィールドは、とりわけ、Ｐｒｅｆｉｘ、Ｏｐｃｏｄｅ、およびＤｉｓｐｌａｃｅｍｅｎｔフィールドを含む（他のＩＳＡの命令の表現は異なりうる）。図７は、所与の例示的ｘ８６命令についての個々のエンコードフィールドの例を示したものである。いくつかのＩＳＡでは、マシンコード表現は可変長を有することができ、例えばいくつかのエンコードフィールドは、特定の命令のマシンコード表現に出現し得るが、他の命令の表現には出現し得ない。
【００４７】
[0058]いくつかの実施形態では、命令の逆アセンブルは、命令のマシンコード表現を構文解析して、一組の意味要素を識別および／または計算することを含む。このような意味要素は、とりわけ、命令のオペレータ（例えばＭＯＶ、ＡＤＤ、等々）およびオペランド（例えばＡＸ、［ＥＢＸ＋４＊ＥＣＸ＋０ｘ０２］）を含むことができる。いくつかの実施形態において、命令の意味要素は、個々の命令エンコードフィールド（例えば、ｘ８６ＩＳＡの場合、Ｐｒｅｆｉｘ、Ｏｐｃｏｄｅ、ｍｏｄＲ／Ｍ、ＳＩＢ、Ｄｉｓｐｌａｃｅｍｅｎｔ、およびＩｍｍｅｄｉａｔｅ）を含む。そのような逆アセンブリは、少なくとも部分的に、命令デコーダ３０によって、および／または実行ユニット３６によって実行され得る。
【００４８】
[0059]図７の例では、命令の逆アセンブルは、マシンコード表現４７と一般化形態４８との間の対応を示す矢印によって示されているように、個々のエンコードフィールドの内容を決定することを含みうる。いくつかの実施形態では、示されている命令の逆アセンブルは、ＡＤＤオペレータを識別すること、および／またはマシンコード４７に従って、それぞれの命令が２つのオペランドを有し、オペランドのうちの１つがＡＸレジスタの内容

であり、第２のオペランドがメモリの内容であると決定すること、並びに、それぞれのメモリアドレスの式（例えばＥＢＸ＋４＊ＥＣＸ＋０ｘ０２）を決定することを含むことができる。いくつかの実施形態では、命令の逆アセンブルは、それぞれの命令のオペランドによって示されるメモリアドレス（例えば、図７の例における式ＥＢＸ＋４＊ＥＣＸ＋０ｘ０２の値）を計算することをさらに含む。逆アセンブルされる命令が相対ジャンプ命令（例えばマシンコードの中に０ｘＥＢ　０ｘ０８として表現された、ｘ８６プラットフォーム上のＪＭＰ　＄＋１０）である別の例では、命令の逆アセンブルは、命令のアドレス、ゲスト命令の長さ、および／または相対ジャンプのサイズに従って、行先の絶対メモリアドレスを計算することを含むことができる。

【００４９】

[0060]いくつかの実施形態では、レジスタ４４の逆アセンブリフィールド４６ａ（図６）は、それぞれの命令の命令エンコードフィールドの内容を含む（図７を参照）。フィールド４６ａの他の例示的内容は、それぞれの命令のオペレータを示すオペレータ識別子、およびそれぞれの命令のオペランドのインジケータを含む。オペランドインジケータは、プロセッサレジスタ（例えばＡＸ）の識別子、および例えばそれぞれのオペランドがレジスタの内容であるか、またはメモリの内容であるかどうかを示すフラグをさらに含むことができる。逆アセンブリフィールド４６ａは、オペランドによって示されるメモリアドレス（例えばＧＶＡ、ＧＰＡおよび／またはＨＰＡ）をさらに含むことができる。逆アセンブルフィールド４６ａの構造は、プラットフォーム固有でありうる。例えば、インテル（登録商標）プラットフォームでは、逆アセンブリフィールド４６ａは、現在のゲスト命令のｐｒｅｆｉｘ、ｏｐｃｏｄｅ、ｍｏｄＲ／Ｍ、ＳＩＢ、ｄｉｓｐｌａｃｅｍｅｎｔ、およびｉｍｍｅｄｉａｔｅのエンコードフィールドの内容を含むことができる。他のプラットフォームでは、フィールド４６ａは、それぞれのプラットフォームの命令セットアーキテクチャ（ＩＳＡ）に従う他の値を記憶することができる。

【００５０】

[0061]いくつかの実施形態では、中断イベントレジスタ４４の実行結果フィールド４６ｂは、それぞれのプロセッサ命令を実行した結果を示すデータを記憶することができる。このような結果は、それぞれの命令を実行することによって得られる、状態レジスタ（例えばＦＬＡＧＳ）の値、命令ポインタ（例えばＲＩＰ）の値、および汎用レジスタ（例えばＥＡＸ）の値を含むことができる。フィールド４６ｂは、それぞれの命令を実行した結果としてメモリにコミットされる値、それぞれの値のサイズ（例えばバイト、語、等々）、および／またはそれぞれの値がコミットされるメモリアドレスをさらに含むことができる。

【００５１】

[0062]いくつかの実施形態では、実行ユニット３６および／またはコミットユニット３８は、ゲスト命令の実行がＶＭプロセッサイベント（仮想化例外のＶＭ終了などの）を生じさせるかどうかを決定し、生じさせる場合、それぞれのイベントを生成する前に、命令逆アセンブリデータを中断イベントレジスタ４４に保存するように構成され得る。プロセッサ１２は、それぞれのゲスト命令の実行ステージが完了するまでプロセッサイベントの生成を遅延し、かつ、それぞれの命令を実行した結果をメモリおよび／またはプロセッサ１２の汎用レジスタにコミットする代わりに、そのような結果をイベントレジスタ４４に保存するようにさらに構成され得る。そのような命令の結果をコミットするのを回避するために、プロセッサ１２は、それぞれの命令のためのパイプラインのコミットステージの前に、ＶＭプロセッサイベントを生成するように構成され得る。このような機能については、以下でさらに詳細に説明される。

【００５２】

[0063]図８は、本発明のいくつかの実施形態による、ゲスト命令を実行するためにプロセッサ１２によって実施されるステップの詳細な例示的シーケンスを示したものである。図８は、プロセッサ１２がメモリアクセス違反に応答してＶＭ終了イベントを生成するように構成される一実施形態を示している。当業者は、この説明が、ＶＭ終了イベントの代

わりに他のＶＭ中断イベント（仮想化例外など）を生成する実施形態を包含するべく容易に修正され得ることを認識するであろう。「ゲスト命令」は、本明細書においては、図２－ＡのＶＭ５２ａ～ｂなどのゲストＶＭ内で実行するコンピュータプログラムの一部を形成しているプロセッサ命令を表すために使用される用語である。

【００５３】

[0064]ステップ３０２は、ゲスト命令のフェッチを試行する。フェッチの試行に失敗すると、ステップ３０３は、失敗がメモリアクセス違反によるものかどうかを決定することができる（例えばゲストＶＭのＳＬＡＴ構造内の非実行可能とマークされたメモリページにゲスト命令が存在する場合）。メモリアクセス違反によるものではない場合、ステップ３０６でプロセッサ１２は、ＶＭ終了イベントを生成し、かつ、実行を図２－Ａのハンドラ６１などのイベントハンドラに移行する。ゲスト命令のフェッチの失敗がメモリアクセス違反によるものである場合、そのような失敗は、セキュリティプログラム（例えばマルウェア防止モジュール）がそれぞれのメモリページの内容を保護しようとしていることを示し得る。このようにして典型的に実行から保護された例示的メモリセクションは、ゲストプロセスの実行スタックを記憶する。スタックを非実行可能としてマークすることにより、例えばスタックの活用からゲストプロセスを保護することができる。このような状況では、いくつかの実施形態は、ゲスト命令のフェッチを再試行し、それぞれのメモリアクセス許可を無視することができる（ステップ３０５）。ステップ３０７で、フェッチされたゲスト命令に、それぞれの命令が「強制的にフェッチされた」こと、すなわちメモリアクセス許可を無効化している間にフェッチされたことを示すために専用フラグの印が付けられる。プロセッサ１２は、次にステップ３０８へ進行することができる。

【００５４】

[0065]フェッチステージに続いて、ステップ３０８は、ゲスト命令をデコードし、ディスパッチする。ステップ３１０でゲスト命令が実行される。ゲスト命令の実行がメモリアクセスには無関係のＶＭ終了基準を満たすと、プロセッサ１２は、以下で詳細に説明されるステップ３２２に進行する。このようなＶＭ終了は、様々な状況でトリガされ得る。例えばゲスト命令は、ゲストＶＭ内から呼び出されると、ＶＭ終了イベントを自動的にトリガするＶＭＣＡＬＬなどの専用命令であってもよい。メモリアクセスには無関係である、ＶＭ終了の別の例示的理由は、ゲスト命令を実行している間のハードウェアイベント（例えば割込み）の発生である。

【００５５】

[0066]ゲスト命令の実行がメモリアクセス違反を生じさせる場合（例えばゲスト命令が非書込み可能とマークされたメモリページに結果を書き込むようにプロセッサに命令する場合）、従来のプロセッサは、典型的にはゲスト命令の実行を中断し、プロセッサパイプラインをフラッシュして、ＶＭ中断イベント（例えばＶＭＥｘｉｔ）を生成する。一方、本発明のいくつかの実施形態では、ゲスト命令の実行は中断されない。その代わりにステップ３１８で、ゲスト命令のためのパイプラインの実行ステージが終了するまでＶＭ終了イベントが遅延される。しかしながら、いくつかの実施形態では、従来のシステムで発生した場合と同様、完了した実行ステージの結果はコミットされない。その代わりにステップ３２０で、プロセッサ１２が、ゲスト命令の完了した実行ステージの結果を中断イベントレジスタ４４に記憶するようにコミットユニット３８に命令することができる。このような機能は、例えば、メモリアクセス違反が生じると、結果をメモリおよび／またはプロセッサ１２の汎用レジスタにコミットすることから、結果をレジスタ４４に記憶することへコミットユニット３８を切り換えるための起動信号を使用して達成され得る。制御信号は、ゲスト命令の実行がメモリアクセス違反を生じさせたかどうかを示すことができる。コミットユニット３８は、例えばＭＭＵからメモリアクセスモジュール３４を介してこのような信号を受け取ることができる。いくつかの実施形態では、ステップ３２０は、コミットユニット３８が、ゲスト命令を実行した結果をレジスタファイル４０から取り出すことを含む。

【００５６】

[0067]一代替実施形態では、ゲスト命令の実行結果をレジスタ４４に保存する代わりに、ステップ３２０は、それぞれのゲストＶＭのＶＭＳＯのゲスト状態領域などの専用メモリ領域にこのような結果を保存することができる。さらに別の実施形態では、プロセッサ１２は、ＶＭ終了を実行すると（ステップ３０６）、このような結果をＶＭ終了ハンドラ６１に伝送することができる。

【００５７】

[0068]ステップ３２２で、プロセッサ１２は、ゲスト命令の逆アセンブルの結果を中断イベントレジスタ４４に（および／または上で説明したようにメモリに）記憶することができる。あるいは、命令逆アセンブリデータは、現在実行しているゲストＶＭのＶＭＳＯの専用エリアに記憶することができる。命令逆アセンブリデータは、ゲスト命令をデコードおよび／または実行するプロセスにおいて、命令デコーダ３０および／または実行ユニット３６によって生成されることが可能であり、ステップ３２２は、このようなデータをそれぞれのプロセッサモジュールから取り出すことを含むことができる。ゲスト命令に対する実行結果および／または逆アセンブリデータを記憶すると、プロセッサ１２は、ＶＭ終了イベントを生成することができる（ステップ３０６）。

【００５８】

[0069]メモリアクセス違反を生じさせることなく（ステップ３１４）、また、ＶＭ終了に対する非メモリ関連の理由なく（ステップ３１２）、現在のゲスト命令の実行が進行すると、ステップ３１５は、現在のゲスト命令が強制的にフェッチされたものであるかどうかを決定することができる（上記ステップ３０５～３０７を参照されたい）。強制的にフェッチされたものではない場合、ステップ３１６は、実行の結果をメモリおよび／または汎用プロセッサレジスタにコミットする。現在のゲスト命令が強制的にフェッチされたものである場合、いくつかの実施形態は、それぞれの命令を、メモリアクセス違反を生じさせる命令として処理することができ、すなわちそれぞれの命令がパイプラインの実行ステージを完了するのを待機することにより、ＶＭ終了イベントを生成する前に結果および／または命令逆アセンブリデータをレジスタ４４に記憶することができる（上記ステップ３１８～３２０～３２２～３０６を参照されたい）。

【００５９】

[0070]図９は、コンピュータセキュリティに関連する本発明のいくつかの実施形態による、ゲストＶＭおよび／またはコンピュータセキュリティモジュール６０（図２－Ａ－Ｂ）によって実施されるステップの例示的シーケンスを示したものである。アプリケーション（例えば図２－Ａの５６ａ）などのゲストプロセス、またはオペレーティングシステム（例えば図２－ＡのゲストＯＳ５４ａ）のプロセスは、ゲストＶＭ内で実行し、一連のゲスト命令を介してステップ方式で進行することができる（ステップ３３２）。ゲストプロセスの実行は、例えば図８に関連して上で説明したシナリオに従ってＶＭ終了が生成されるまで継続する。当業者は、この説明が、プロセッサ１２がＶＭ終了イベントの代わりに仮想化例外を生成し、また、ゲストＶＭ内で実行する例外ハンドラ（例えば図２－Ｂのハンドラ６３）が、それぞれの例外を傍受するように構成されるシステムに如何に適合され得るかを認識することができる。

【００６０】

[0071]ステップ３３６で、ハンドラ６１がＶＭ終了イベントを傍受し、このＶＭ終了イベントがセキュリティの脅威の証拠であることが分析される。イベントがセキュリティの脅威（例えば悪意のある意図で実行された操作）を示す場合、ステップ３４０でＣＳＭ６０は、ゲストプロセスに対する保護アクション、および／またはゲストＶＭに対する保護アクションを取ることができる。このようなアクションは、とりわけ、ゲストプロセスの実行を阻止すること、エラーメッセージまたは一組の疑似結果をゲストプロセスに返すこと、およびホストシステムの管理者に警告することを含むことができる。

【００６１】

[0072]ＶＭ終了イベントがセキュリティの脅威を示さない場合、ステップ３４２は、ゲスト命令を実行した結果が利用可能である（プロセッサ１２のイベントレジスタ４４また

はメモリのいずれかにある）かどうかを決定する。利用可能でない場合、ＣＳＭ６０は、以下で詳細に説明されるステップ３４８に進行する。利用可能である場合、ステップ３４４は、それぞれの結果をレジスタ４４および／またはメモリ（例えばそれぞれのゲストＶＭのＶＭＳＯのゲスト状態領域）から取り出す。ステップ３４６でＣＳＭ６０は、現在のゲスト命令を実行した結果を適用することができる。いくつかの実施形態では、ステップ３４６は、コミットステージにおいて従来のシステムの中で実行される一組の操作を含む。例えばステップ３４６は、それぞれのゲストＶＭの仮想化プロセッサの汎用、制御、および状態プロセッサレジスタの値を更新することを含むことができる。いくつかの実施形態では、このようなレジスタは、それぞれのゲストＶＭのＶＭＳＯのゲスト状態領域内でアクセスすることができる。ステップ３４６は、現在のゲスト命令のオペランドによって示されるメモリアドレスにいくつかの結果を保存することをさらに含むことができる。ステップ３４６は、現在のゲスト命令の実行が完了したことを示すために、命令ポインタ（例えばｘ８６プラットフォーム内のＲＩＰ）を増分することをさらに含むことができる。

【００６２】

[0073]本発明のいくつかの実施形態は、プロセッサ１２の現在の命令セットアーキテクチャ（ＩＳＡ）に専用命令を追加し、この新しい命令は、ゲスト命令を実行した結果を、それぞれのゲスト命令を実行しているゲストＶＭの下から直接適用するようにプロセッサ１２に命令する。この新しい命令（例示的ニーモニックはＶＭＡＰＰＬＹである）は、ステップ３４６（図９）の操作を実行することができ、例えば内容を中断イベントレジスタ４４からそれぞれのゲストＶＭの仮想化プロセッサの仮想レジスタおよび／またはメモリにコピーすることができる。

【００６３】

[0074]いくつかの実施形態では、ステップ３４６は、さらに、現在のゲスト命令がアトミック命令（例えばＬＯＣＫプレフィックスで示される）であるかどうかを検証することができる。アトミック命令である場合、結果をゲストのレジスタおよび／またはメモリに直接適用する代わりに、ステップ３４６は、ゲストＶＭへ戻すと、現在のゲスト命令の再実行を強制することができる（以下のステップ３５６を参照されたい）。

【００６４】

[0075]現在のゲスト命令の実行結果を利用することができない場合（例えば現在のＶＭ終了がＶＭＣＡＬＬなどの特権命令によって生じたものである場合）、ステップ３４８でコンピュータセキュリティモジュール６０は、逆アセンブリデータが現在のゲスト命令のために利用可能であるかどうかを決定する。利用可能である場合、ステップ３５０でＣＳＭ６０は、そのデータを例えばレジスタ４４の逆アセンブリフィールド４６ａ（例えば図６を参照されたい）から取り出すことができる。ＣＳＭ６０は、次に、取り出された逆アセンブリデータに応じた現在のゲスト命令のエミュレートへ進行することができる（ステップ３５４）。

【００６５】

[0076]逆アセンブリデータを利用することができない場合、ステップ３５２は、エミュレーションを進める前に、現在のゲスト命令を逆アセンブルすることができる。ステップ３５６でＣＳＭ６０は、それぞれのゲストＶＭを再起動することができる（例えばＩｎｔｅｌ（登録商標）プラットフォームのＶＭＲＥＳＵＭＥ命令を発行することによって）。ステップ３４６が命令ポインタの修正を含むいくつかの実施形態では、ゲストプロセスの実行は、現在のゲスト命令の直後のプロセッサ命令で開始するか、または現在のゲスト命令によって示されるプロセッサ命令で開始することができる（例えばＪＭＰ、ＣＡＬＬ、等々などの制御フロー変更命令の場合）。

【００６６】

[0077]上で説明した例示的システムおよび方法は、コンピュータまたはスマートフォンなどのホストシステムによる、ハードウェア仮想化構成で動作している場合のコンピュータセキュリティタスクの有効な実行を可能にする。セキュリティタスクは、とりわけ、コンピュータウイルスおよびスパイウェアなどのマルウェアに対するホストシステムの保護

を含むことができる。いくつかの実施形態では、ホストシステムは、オペレーティングシステムおよび一組のソフトウェアアプリケーションを仮想マシン内で実行するように構成される。セキュリティモジュールは、それぞれの仮想マシンの外部で、例えばハイパーバイザのレベルで実行することができ、また、それぞれの仮想マシンをマルウェアから保護することができる。

【００６７】

[0078]いくつかの実施形態では、セキュリティモジュールは、保護されたＶＭのセキュリティには極めて重要なコードおよび／またはデータを含むメモリのセクション（例えば一組のメモリページ）を識別し、かつ、メモリのそれぞれのセクションに対するアクセス許可を構成する。このようなアクセス許可は、例えばメモリのそれぞれのセクションが非書込み可能および／または非実行可能であることを示すことができる。セキュリティモジュールは、さらに、例えば保護されたＶＭ内で実行しているソフトウェアが、非書込み可能とマークされたメモリのセクションへの書込みを試行し、または非実行可能とマークされたメモリのセクションからのコードの実行を試行すると、メモリアクセス違反に応答してＶＭ中断イベント（ＶＭ終了または仮想化例外など）を生成するようにホストシステムのプロセッサを構成することができる。セキュリティモジュールは、次に、イベントハンドラを介してこのようなプロセッサイベントを傍受することができ、また、このようなイベントがコンピュータセキュリティの脅威を示すかどうかを決定することができる。セキュリティモジュールが保護されたＶＭの外部で実行する構成では、セキュリティモジュールのアクティビティは、マルウェアを含む、保護されたＶＭ内で実行するソフトウェアには見ることができない可能性がある。

【００６８】

[0079]従来のシステムでは、ＶＭ中断イベントの傍受は、当分野で総称的に「トラップアンドエミュレート」として知られる方法に従って進行する。従来の技法の一例では、それぞれのイベント（例えばＶＭ終了）を生じさせた命令を決定した後、実行を保護されたＶＭに戻す前に、また、それぞれの命令が既に実行されていることを示すために命令ポインタを修正する前に、マルウェア防止プログラムがそれぞれの命令をエミュレートする。エミュレーションステップがない場合、実行を保護されたＶＭに戻すと、典型的にはＶＭ終了を再トリガすることになり、したがって無限ループを生成することになる。

【００６９】

[0080]したがって、従来のトラップアンドエミュレートシステムおよび方法には、場合によっては、命令逆アセンブラおよび／または命令エミュレータを含むためのマルウェア防止プログラムが必要である。このような構成要素は、場合によっては開発および維持が複雑であり、また、例えば１つのプロセッサから別のプロセッサに移植できないことがある。さらに、従来のシステムでは、逆アセンブリおよび／またはエミュレーションステップは、典型的にはすべてのＶＭ中断イベントに対して実行され、相当な計算負荷をホストシステムに課す。一方、本発明のいくつかの実施形態は、逆アセンブラおよび／またはエミュレータの必要性をなくし、コンピュータセキュリティ動作を実質的に加速する。

【００７０】

[0081]本発明のいくつかの実施形態は、従来のプロセッサの構成および動作に変更を導入し、ハードウェア仮想化構成におけるこのようなプロセッサのより有効な動作を可能にする。いくつかの実施形態において、プロセッサは、ＶＭ中断イベントの生成を、現在の命令のためのパイプラインの実行フェーズが完了するまで、少なくともある状況（例えば、ＶＭ中断イベントがメモリアクセス違反によりトリガされるとき）では遅らせるように構成される。プロセッサは、現在の命令の実行ステージの結果を、特別なプロセッサレジスタ（プロセッサの汎用レジスタ若しくは制御レジスタとは異なる）または特別なメモリ領域（例えば、それぞれのゲストＶＭのゲスト状態領域）に保存するようにさらに構成することができる。

【００７１】

[0082]このような改善は、コンピュータセキュリティアプリケーションにはとりわけ有

利であり、外部、例えばそれぞれのＶＭを公開するハイパーバイザのレベルからの仮想マシンの有効な保護を可能にすることができる。従来のコンピュータセキュリティ解決法と比較すると、本発明のいくつかの実施形態は、ＶＭ中断イベントを傍受し、かつ、分析するように構成されるセキュリティソフトウェアの操作から逆アセンブリおよびエミュレーションのステージを除去することにより、計算の実質的な低減を可能にする。それぞれのイベントを生成した命令のエミュレートに代えて、いくつかの実施形態は、セキュリティソフトウェアが、それぞれの結果をプロセッサレジスタまたはメモリの場所から読み出し、そのような結果を直接適用することを可能にする。

【００７２】

[0083]従来のシステムとは対照的に、本発明のいくつかの実施形態は、ＶＭ中断イベントを、現在のゲスト命令がパイプラインの実行ステージを完了した後にのみ生成する。ハードウェアコンポーネントの機能に対するそのような変更は、それぞれの命令の実行ステージを実行するのに必要な余分なクロックチックによる遅延を導入することがある。しかしながら、そのような性能のペナルティは、従来のコンピュータセキュリティソフトウェアには必要な（潜在的には、各ＶＭ中断イベントに対して数百の追加命令に達する）命令エミュレーションおよび／または逆アセンブリの操作を除去することによって実質的に相殺される。

【００７３】

[0084]上記実施形態は、本発明の範囲を逸脱することなく、多くの方法で変更され得ることは当業者には明らかであろう。したがって本発明の範囲は、以下の特許請求の範囲およびそれらの合法等価物によって決定されるものとする。

【図１】

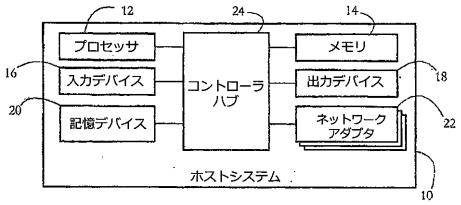

FIG. 1

【図２－Ａ】

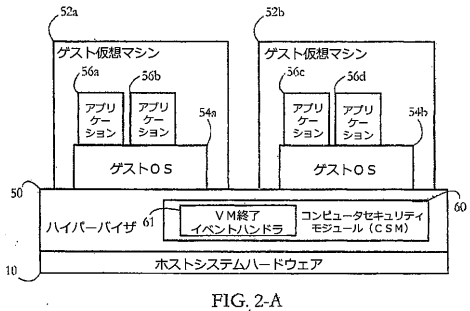

FIG. 2-A

【図２－Ｂ】

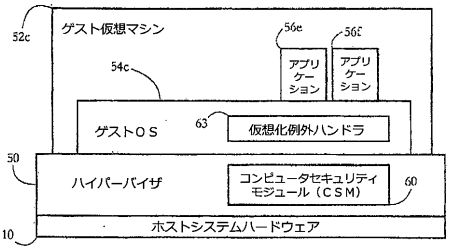

FIG. 2-B

【図２－Ｃ】

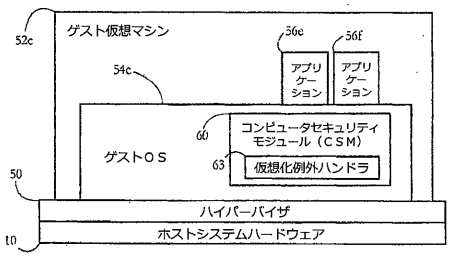

FIG. 2-C

【図３】

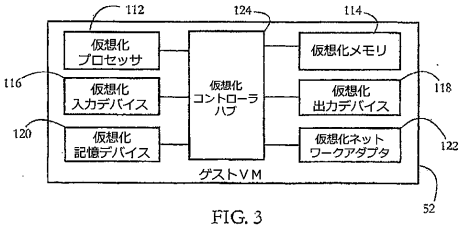

FIG. 3

【図６】

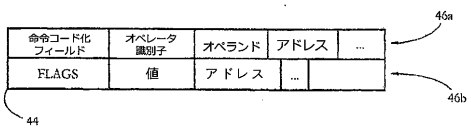

FIG. 6

【図７】

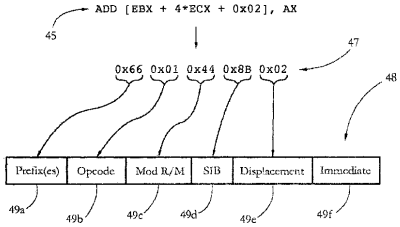

FIG. 7

【図８】

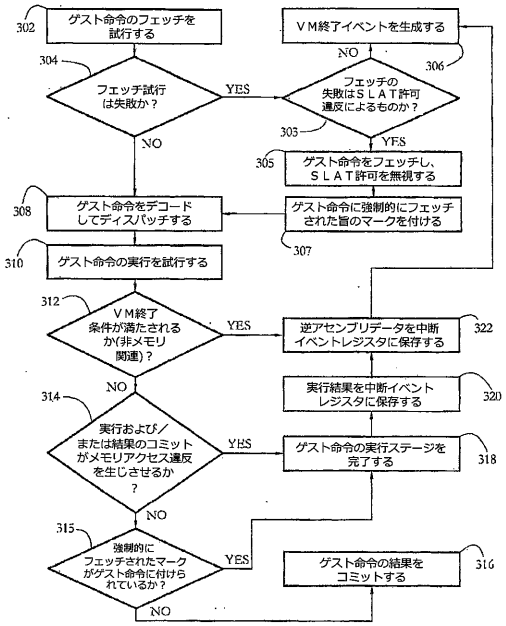

FIG. 8

【図９】

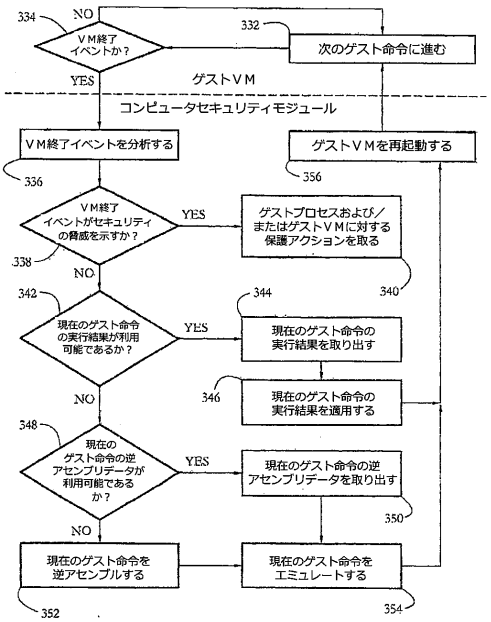

FIG. 9

フロントページの続き

(72)発明者　ルカクス，サンドル
ルーマニア国ジュデツ・クルジュ，コム．フロレシチ，サト・フロレシチ，ビルド．チェタテア・フェテイ　ブロック　８，エタジュル　３
(72)発明者　ルツァス，アンドレイ－ヴラド
ルーマニア国ジュデツ・サトゥ・マーレ，サトゥ・マーレ，ビルド．クロシカ　ヌマルル　１１１

審査官　上島　拓也

(56)参考文献　特表２００７－５２８０８４（ＪＰ，Ａ）
特開２０１１－２２７９３９（ＪＰ，Ａ）
特開２００９－９２３２（ＪＰ，Ａ）
特表２００５－５２８６９０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／５６
Ｇ０６Ｆ　９／４５５