

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4437155号
(P4437155)

(45) 発行日 平成22年3月24日(2010.3.24)

(24) 登録日 平成22年1月8日(2010.1.8)

(51) Int.Cl.

F I

G 0 6 F 9/46 (2006.01)

G 0 6 F 9/46 3 5 0

請求項の数 28 (全 17 頁)

(21) 出願番号	特願2007-515693 (P2007-515693)	(73) 特許権者	591003943
(86) (22) 出願日	平成17年6月9日(2005.6.9)		インテル・コーポレーション
(65) 公表番号	特表2008-501198 (P2008-501198A)		アメリカ合衆国 9 5 0 5 2 カリフォル
(43) 公表日	平成20年1月17日(2008.1.17)		ニア州・サンタクララ・ミッション カレ
(86) 国際出願番号	PCT/US2005/020464		ッジ ブレーバード・2 2 0 0
(87) 国際公開番号	W02006/007361	(74) 代理人	100104156
(87) 国際公開日	平成18年1月19日(2006.1.19)		弁理士 龍華 明裕
審査請求日	平成18年11月28日(2006.11.28)	(72) 発明者	ロジャース、ディオン
(31) 優先権主張番号	10/882, 813		アメリカ合衆国、9 7 1 2 3 オレゴン州
(32) 優先日	平成16年6月30日(2004.6.30)		、ヒルズバロ、ブルックウッド アベニュー 4 5 2
(33) 優先権主張国	米国 (US)	(72) 発明者	アンダーソン、アンドリュウ、ブイ
			アメリカ合衆国、9 7 1 2 3 オレゴン州
			、ヒルズバロ、サウスイースト 6 8 ティーエイチ アベニュー 6 7 7
			最終頁に続く

(54) 【発明の名称】 仮想マシン環境におけるネストフォールトのサポート

(57) 【特許請求の範囲】

【請求項 1】

方法であって、

プロセッサが、仮想マシン (VM) に引渡される第 1 フォールトを検出することと、前記プロセッサが、前記第 1 フォールトを前記 VM に引渡している間に第 2 フォールトを検出することと、前記プロセッサが、前記第 2 フォールトが仮想マシンモニタ (VMM) への制御の移行と対応しているかどうかを判定することと、前記プロセッサが、前記第 2 フォールトが前記 VMM への制御の移行と対応している場合、前記第 1 フォールトに関連する情報を第 1 フィールドに格納し、前記第 2 フォールトに関連する情報を第 2 フィールドに格納し、前記 VMM に制御を移行することと、前記プロセッサが、前記第 2 フォールトが前記 VMM への制御の移行と対応していない場合に前記第 2 フォールトを前記 VM に引渡すことと、

を含む方法。

【請求項 2】

前記第 1 フォールトに関連する前記情報は、前記第 2 フォールトを検出する前に、一時記憶場所または一時記憶メモリに格納される

請求項 1 に記載の方法。

【請求項 3】

前記第 1 フィールドと前記第 2 フィールドは、仮想マシン制御構造 (VMCS) に含ま

10

20

れる

請求項 1 または 2 に記載の方法。

【請求項 4】

前記第 1 フォールトに関連する前記情報は前記第 1 フォールトの識別子を含み、
前記第 2 フォールトに関連する前記情報は前記第 2 フォールトの識別子を含む
請求項 1 から 3 のいずれかに記載の方法。

【請求項 5】

前記第 1 フォールトに関連する前記情報はさらに、前記第 1 フォールトに対応付けられたエラーコードを含み、

前記第 2 フォールトに関連する前記情報はさらに、前記第 2 フォールトに対応付けられたエラーコードを含む

請求項 4 に記載の方法。

【請求項 6】

前記プロセッサが、前記第 2 フォールトが前記 VMM への制御の移行と対応していない場合、現在のネストレベルが最大許容ネストレベルを超えていないかどうか判定することと、

前記プロセッサが、前記現在のネストレベルが前記最大許容ネストレベルを超えている場合、シャットダウン状態への移行が前記 VMM への制御の移行と対応していると判定されれば、前記 VMM へ制御を移行することと

をさらに含む請求項 1 から 5 のいずれかに記載の方法。

【請求項 7】

前記プロセッサが、前記第 1 フォールトが前記 VMM への制御の移行と対応しているかどうかを判定することと、

前記プロセッサが、前記第 1 フォールトが前記 VMM への制御の移行と対応している場合、前記第 1 フォールトを前記 VM に引渡さず、前記第 1 フォールトに関連する情報を前記第 1 フィールドに格納し、前記 VMM に制御を移行することと

をさらに含む請求項 1 から 6 のいずれかに記載の方法。

【請求項 8】

前記プロセッサが、前記第 1 フォールトの前記引渡しが前記 VM への制御の移行の一環でないと判定された時に、前記第 1 フォールトが前記 VMM への制御の移行と対応しているかどうか判定する

請求項 1 から 7 のいずれかに記載の方法。

【請求項 9】

前記プロセッサが、前記第 2 フォールトは前記 VMM への制御の移行と対応していないと判定することと、

前記プロセッサが、前記第 2 フォールトを前記 VM に引渡すことと、

前記プロセッサが、前記 VM に前記第 2 フォールトを引渡している間に第 3 フォールトを検出することと、

前記プロセッサが、前記第 3 フォールトが前記 VMM への制御の移行と対応している場合、前記第 1 フォールトに関連する情報を前記第 1 フィールドに格納し、前記第 2 フォールトに関連する情報を前記第 2 フィールドに格納し、前記第 3 フォールトに関連する情報を第 3 フィールドに格納し、前記 VMM に制御を移行することと

をさらに含む請求項 1 から 8 のいずれかに記載の方法。

【請求項 10】

前記プロセッサが、前記第 2 フォールトは前記 VMM への制御の移行と対応していないと判定することと、

前記プロセッサが、前記第 2 フォールトを前記 VM に引渡すことと、

前記プロセッサが、前記 VM に前記第 2 フォールトを引渡している間に第 3 フォールトを検出することと、

前記プロセッサが、前記第 3 フォールトが前記 VMM への制御の移行と対応している場

10

20

30

40

50

合、前記第 2 フォールトに関連する情報を前記第 1 フィールドに格納し、前記第 3 フォールトに関連する情報を前記第 2 フィールドに格納し、前記 VMM に制御を移行することとをさらに含む請求項 1 から 9 のいずれかに記載の方法。

【請求項 11】

前記プロセッサが、前記第 2 フォールトを前記 VM に引渡す前に、現在のネストレベルが最大許容ネストレベルを超えていないと判定すること

をさらに含む請求項 9 または 10 に記載の方法。

【請求項 12】

前記プロセッサが、前記現在のネストレベルが前記最大許容ネストレベルを超えている場合、前記 VM をシャットダウン状態にすること

をさらに含む請求項 11 に記載の方法。

【請求項 13】

装置であって、

仮想マシンモニタ (VMM) と、

前記 VMM が制御するデータ構造であって、第 1 フィールドおよび第 2 フィールドを持つデータ構造と、

仮想マシン (VM) の動作中に起こる第 1 フォールトに関連する情報を前記第 1 フィールドに格納し、前記第 1 フォールトを前記 VM に引渡している間に検出された第 2 フォールトに関連する情報を前記第 2 フィールドに格納し、前記第 2 フォールトが前記 VMM への制御の移行と対応している場合、前記 VMM に制御を移行し、前記第 2 フォールトが前記 VMM への制御の移行と対応していない場合に前記第 2 フォールトを前記 VM に引渡すフォールト引渡しロジックと

を備える装置。

【請求項 14】

前記データ構造は仮想マシン制御構造 (VMCS) である

請求項 13 に記載の装置。

【請求項 15】

前記第 1 フォールトに関連する前記情報は前記第 1 フォールトの識別子を含み、

前記第 2 フォールトに関連する前記情報は前記第 2 フォールトの識別子を含む

請求項 13 または 14 に記載の装置。

【請求項 16】

前記第 1 フォールトに関連する前記情報はさらに、前記第 1 フォールトに対応付けられたエラーコードを含み、

前記第 2 フォールトに関連する前記情報はさらに、前記第 2 フォールトに対応付けられたエラーコードを含む

請求項 15 に記載の装置。

【請求項 17】

前記フォールト引渡しロジックはさらに、前記第 1 フォールトが前記 VMM への制御の移行に対応しているかどうか判定し、前記第 1 フォールトが前記 VMM への制御の移行と対応している場合は、前記第 1 フォールトを前記 VM に引渡さず、前記第 1 フォールトに関連する情報を前記第 1 フィールドに格納し、前記 VMM に制御を移行する

請求項 13 から 16 のいずれかに記載の装置。

【請求項 18】

前記フォールト引渡しロジックは、前記第 1 フォールトの前記引渡し前記 VM への制御の移行の一環でないと判定した時に、前記第 1 フォールトが前記 VMM への制御の移行と対応しているかどうか判定する

請求項 13 から 17 のいずれかに記載の装置。

【請求項 19】

前記フォールト引渡しロジックはさらに、前記第 2 フォールトは前記 VMM への制御の移行と対応していないと判定し、前記第 2 フォールトを前記 VM に引渡し、前記 VM に前

10

20

30

40

50

記第 2 フォールトを引渡ししている間に第 3 フォールトを検出し、前記第 3 フォールトが前記 VMM への制御の移行と対応している場合、前記第 1 フォールトに関連する情報を前記第 1 フィールドに格納し、前記第 2 フォールトに関連する情報を前記第 2 フィールドに格納し、前記第 3 フォールトに関連する情報を前記データ構造の第 3 フィールドに格納し、前記 VMM に制御を移行する

請求項 13 から 18 のいずれかに記載の装置。

【請求項 20】

前記フォールト引渡しロジックはさらに、前記第 2 フォールトは前記 VMM への制御の移行と対応していないと判定し、前記第 2 フォールトを前記 VM に引渡し、前記 VM に前記第 2 フォールトを引渡ししている間に第 3 フォールトを検出し、前記第 3 フォールトが前記 VMM への制御の移行と対応している場合、前記第 2 フォールトに関連する情報を前記第 1 フィールドに格納し、前記第 3 フォールトに関連する情報を前記第 2 フィールドに格納し、前記 VMM に制御を移行する

10

請求項 13 から 19 のいずれかに記載の装置。

【請求項 21】

前記フォールト引渡しロジックは、前記第 2 フォールトを前記 VM に引渡す前に、現在のネストレベルが最大許容ネストレベルを超えていないか判定する

請求項 19 または 20 に記載の装置。

【請求項 22】

前記フォールト引渡しロジックはさらに、前記現在のネストレベルが前記最大許容ネストレベルを超えている場合、前記 VM をシャットダウン状態にする

20

請求項 21 に記載の装置。

【請求項 23】

システムであって、

ゲストソフトウェアを格納するメモリと、

前記メモリに接合されたプロセッサであって、仮想マシン (VM) の動作中に起こる第 1 フォールトに関連する情報を第 1 フィールドに格納し、前記第 1 フォールトを前記 VM に引き渡ししている間に第 2 フォールトを検出し、前記第 2 フォールトが仮想マシンモニタ (VMM) への制御の移行と対応しているかどうか判定し、前記第 2 フォールトが前記 VMM への制御の移行と対応している場合、前記第 2 フォールトに関連する情報を第 2 フィールドに格納し、前記 VMM に制御を移行し、前記第 2 フォールトが前記 VMM への制御の移行と対応していない場合に前記第 2 フォールトを前記 VM に引渡すプロセッサと

30

を備えるシステム。

【請求項 24】

前記第 1 フォールトに関連する前記情報は前記第 1 フォールトの識別子を含み、

前記第 2 フォールトに関連する前記情報は前記第 2 フォールトの識別子を含む

請求項 23 に記載のシステム。

【請求項 25】

前記第 1 フォールトに関連する前記情報はさらに、前記第 1 フォールトに対応付けられたエラーコードを含み、

40

前記第 2 フォールトに関連する前記情報はさらに、前記第 2 フォールトに対応付けられたエラーコードを含む

請求項 23 または 24 に記載のシステム。

【請求項 26】

プロセッシングシステムにより実行されるプログラムであって、

前記プロセッシングシステムに、

仮想マシン (VM) の動作中に発生する第 1 フォールトに関連する情報を第 1 フィールドに格納することと、

前記第 1 フォールトを前記 VM に引き渡ししている間に第 2 フォールトを検出することと

50

前記第2フォールトが仮想マシンモニタ（VMM）への制御の移行と対応しているかどうか判定することと、

前記第2フォールトが前記VMMへの制御の移行と対応している場合、前記第2フォールトに関連する情報を第2フィールドに格納し、前記VMMに制御を移行することと、

前記第2フォールトが前記VMMへの制御の移行と対応していない場合に前記第2フォールトを前記VMに引渡すことと、

を実行させるプログラム。

【請求項27】

前記第1フォールトに関連する前記情報は前記第1フォールトの識別子を含み、

前記第2フォールトに関連する前記情報は前記第2フォールトの識別子を含む

請求項26に記載のプログラム。

10

【請求項28】

前記第1フォールトに関連する前記情報はさらに、前記第1フォールトに対応付けられたエラーコードを含み、

前記第2フォールトに関連する前記情報はさらに、前記第2フォールトに対応付けられたエラーコードを含む

請求項26または27に記載のプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

20

本発明の実施形態は広く仮想マシンに関する。より具体的には、仮想マシン環境においてネストされたフォールトに対するサポートを提供することに関する。

【背景技術】

【0002】

割り込みもしくは例外が発生すると、特別な内部状態または外部状態に対して処理が必要なことがプロセッサに伝えられる。そのような信号を受信すると、プロセッサは現在行っている処理を一時停止して、特別状態に対応するためのルーチンの実行を開始する。このルーチンをハンドラと呼ぶ。ハンドラのタスクには通常、特別状態の発生源と原因を特定すること、状態に対して処理を行うこと、および一時停止された処理を再開すること（処理が中止されていない場合）などがある。

30

【0003】

ソフトウェアが短時間で割り込みまたは例外の原因を特定できるように、各特別状態にベクトルまたはベクトル数と呼ばれる識別可能な数を割り当てるとしてもよい。このような割り当てはプロセッサの命令セットアーキテクチャ（ISA）の一部である。例えば、Intel（登録商標）Pentium（登録商標）4のISA（本明細書ではIA-32 ISAと呼ぶ）では、例外についてはそれぞれにベクトル数（例えば、ベクトル0は分割エラーに対応し、ベクトル14はページフォールトに対応）を割り当てて、割り込み（例えばネットワークインターフェースハードウェアによる割り込み）に対するベクトルについては割り当てを行わない。

【0004】

40

割り込みおよび例外の処理を迅速に進めるべく、アーキテクチャでは割り込み記述子テーブル（IDT）と呼ばれる構造を定義するとしてもよい。IDTはベクトル毎に1つのエントリを持つ。各エントリには、対応するハンドラの位置を規定した記述子（例えば、IA-32 ISAにおける割り込みゲート、トラップゲートまたはタスクゲート）を含む。このように設計されているので、ソフトウェアはベクトルが割り当てられた割り込みや例外に対して対応するハンドラをインストールすることができる。ISAによっては、すべての例外および／または割り込みについて1つの同じベクトルを使用するものもある。

【0005】

プロセッサは特別状態を検出すると、その現在の状態に対応するIDTのエントリを取

50

り出し、対応するハンドラの最初を示すポインタを特定する。すべての例外および／または割り込みに対して1つの同じベクトルを用いるISAの場合は、IDTの索引を作成する必要はない。対応するハンドラの最初を示すポインタは、この1つのベクトルに基づいて特定される。続いてプロセッサは、現在実行中の処理の状態（例えば該処理の命令ポインタ）を保存し、抽出したポインタを用いてハンドラの最初に飛ぶ。この処理をフォールトの引渡しと呼ぶ。

【0006】

ISAによっては、一部の例外について、ハンドラの最初まで飛ぶ前にエラーコードがスタックにプッシュされるものもある。または、ハードウェアレジスタ、1以上の記憶場所などの手段にエラーコードを保持するとしてもよい。このようなエラーコードにより、ハンドラに対して、この例外的な状況に関連する情報がさらに提供される（例えばフォールト部分用レジスタの識別子）。

10

【0007】

またISAによっては、割り込みまたは例外を引き渡している間に例外が発生することもある。IA-32 ISAを例に挙げて説明すると、プロセッサがハンドラに割り込みもしくは例外を引渡ししている間に、複数の例外が発生することがある。このような例外の例を挙げると、IDTが存在するマークされたページがページテーブルに存在しない場合に発生するページフォールトがある。この場合、プロセッサは引渡し途中の割り込みではなく、ページフォールト例外を引渡す。

20

【0008】

【特許文献1】米国特許第5511217号明細書

【特許文献2】米国特許第4987532号明細書

【図面の簡単な説明】

【0009】

以下の部分では、実施形態を図示した添付図面を参照しつつ本発明を説明する。添付図面の内容を説明する。

【0010】

【図1】本発明の実施形態が動作する、一実施形態に係る仮想マシン環境を示す。

【0011】

【図2】一実施形態に係る、仮想マシン環境においてネストフォールトに対するサポートを提供するための処理を示すフローチャートである。

30

【図3】一実施形態に係る、仮想マシン環境においてネストフォールトに対するサポートを提供するための処理を示すフローチャートである。

【0012】

【図4】一実施形態に係る、仮想マシン環境においてネストフォールトを処理するためのプロセスを示すフローチャートである。

【発明を実施するための最良の形態】

【0013】

仮想マシン環境においてネストされたフォールトに対するサポートを提供する方法および装置を説明する。以下の説明では、説明の便宜上、具体的且つ詳細な内容を記載する。しかし、当業者には明らかであるが、本発明の実施形態は以下の具体的且つ詳細な記載内容を用いず実施することが可能である。

40

【0014】

以下の説明では一部、コンピュータシステムのレジスタまたはメモリ内でデータビットに対して行われる処理のアルゴリズムやそのような処理を表す符号を用いている。データ処理の分野においては、当業者同士間で作業の内容を伝える場合に、最も効率的な手段として、このようなアルゴリズムによる説明を用いる。本明細書内および一般的にアルゴリズムとは、所望の結果を得るために一貫した順序で実施される一連の処理と考えられている。このような一連の処理では、物理量を物理的に操作する必要がある。必ずしもそうではないが多くの場合、そのような物理量とは電気または磁気信号で、記憶、移動、合成、

50

比較といった操作を行うことができる。こういった信号は、利便性を考慮し一般的な用語を使うべく、ビット、値、成分、符号、文字、項、数字などと呼ばれる。

【0015】

しかし、こういった用語はすべて物理量に対応するものであって、対応する物理量に用いられる便利な名称にすぎない。本明細書において明らかであるように、特に明示されていない限り、「処理」「演算」「計算」「決定」等の用語を用いている部分は、コンピュータシステム、または同様の演算電子デバイスによる処理や動作に関する。こういったコンピュータシステムや演算電子デバイスは、システム内のレジスタおよびメモリ内の物理（電子）量として表されたデータを操作および変換して、同システム内のメモリ、レジスタまたは同様の情報記憶／送信／表示装置内に同様に物理量として表される別のデータを
10

【0016】

以下の実施形態の詳細な説明においては、本発明の具体的な実施形態を図示した添付図面に言及する。添付図面においては、ほぼ同じ構成要素については異なる図面にわたって同様の参照番号を用いている。図示した実施形態は当業者が本発明を実施できるよう、十分詳細に説明してある。本発明の範囲を離れることなく、説明した以外の実施形態を実施するとしてもよいし、説明した実施形態の構造、論理的構成および電氣的構成を変更するとしてもよい。また、本発明の実施形態はそれぞれ異なるものの、必ずしも互いに共通点がまったくないわけではない。例えば、一実施形態に関連して説明した特徴や構造が、ほかの実施形態にも含まれるとしてもよい。
20

【0017】

以下に述べる実施形態の例においては、実行ユニットや論理回路に基づいて、仮想マシン環境におけるネストされたフォールトに対するサポートを説明しているが、本発明の別の実施形態はソフトウェアに基づいて実施されるとしてもよい。例えば本発明は、コンピュータ（またはほかの電子デバイス）をプログラムして本発明に係る処理を行わせる命令を格納した機械・コンピュータ可読媒体を備えるコンピュータプログラム製品またはコンピュータプログラムソフトウェアを提供する実施形態を含むとしてもよい。本発明の別の実施形態に係る処理は、該処理を行う論理が組み込まれた特別なハードウェア素子、またはプログラムされたコンピュータ素子およびカスタムハードウェア素子の組み合わせに基づき行われるとしてもよい。
30

【0018】

このように、機械可読媒体は、機械（例えばコンピュータ）が読める形態で情報を格納または送信する機構を備えるとしてもよい。例を挙げると、これらに限定されないが、フロッピー（登録商標）ディスク、光ディスク、CD-ROM（Compact Disc Read Only Memory）、光磁気ディスク、ROM、RAM（Random Access Memory）、EPROM（Erasable Programmable ROM）、EEPROM（登録商標）（Electrically Erasable and Programmable ROM）、磁気／光カード、フラッシュメモリ、インターネット通信、電氣的形態、光学的形態または音響的形態を初めとする形態の伝達信号（例えば搬送波、赤外線信号、デジタル信号など）などがある。
40

【0019】

設計は作成からシミュレーションや製造までさまざまな段階を経て行われるとしてもよい。設計内容を表すデータはさまざまな形態を取るとしてもよい。まず、シミュレーション段階での便宜を考慮して、ハードウェア記述言語またはその他の機能記述言語を用いてハードウェアを表現するとしてもよい。また、ロジックおよび／またはトランジスタゲートを有する回路レベルモデルを設計プロセス中に製造するとしてもよい。さらに、設計のある段階において通常、ハードウェアモデル内でのデバイスの物理的配置を表すデータが必要とされる。従来の半導体製造技術を利用する場合は、ハードウェアモデルを示すデータとは集積回路の製造に用いられるマスク用のさまざまなマスク層がどのような特徴を有するかまたは有さないかを示すデータであるとしてもよい。どのように設計内容を表現す
50

るとしても、データは機械可読媒体に記憶されるとしてよい。このような機械可読媒体の例は、格納された情報を送信するべく変調または生成された光波または電波、メモリ、ディスクなどの磁気ストレージもしくは光ストレージである。例示したこれら媒体は、設計内容またはソフトウェア情報を「運ぶ」または「示す」ものであってよい。そのようなコードもしくは設計内容を示すまたは運ぶ電気搬送波を送信して、電気信号のコピー、バッファもしくは再送を行う場合、新たにコピーが作成される。このため、通信プロバイダまたはネットワークプロバイダは本発明を実施するもの（搬送波）のコピーを作成できる。

【0020】

図1は、本発明を実施する、一実施形態に係る仮想マシン環境100を示す。この実施形態に係るベアプラットフォームハードウェア116は、例えば標準オペレーティングシステム（OS）またはVMM112のような仮想マシンモニタ（VMM）を実行することが出来るコンピューティングプラットフォームから成るとしてもよい。

10

【0021】

VMM112は通常ソフトウェアで実施されるが、ベアマシンインターフェースをより高いレベルのソフトウェアに、エミュレートおよびエクスポートするとしてもよい。このようなより高いレベルのソフトウェアは、標準またはリアルタイムのOSを有するとしてもよいし、OS機能が限定された低機能簡易版の動作環境であってもよいし、従来のOS機能を含まないものとしてもよい。または、VMM112は例えば、別のVMM内もしくは上で実行されるとしてもよい。VMMは例えばハードウェア、ソフトウェア、ファームウェアもしくはさまざまな技術の組み合わせに基づき実施されるとしてもよい。

20

【0022】

プラットフォームハードウェア116は、パーソナルコンピュータ（PC）、メインフレーム、携帯デバイス、携帯コンピュータ、セットトップボックスなどのコンピューティングシステム用のものであってよい。プラットフォームハードウェア116はプロセッサ118およびメモリ120を有する。

【0023】

プロセッサ118は、ソフトウェアを実行することが出来るプロセッサであればどのようなものであってもよく、例を挙げるとマイクロプロセッサ、デジタル信号プロセッサ、マイクロコントローラなどがある。プロセッサ118は、本発明の実施形態に係る方法を実行するためのマイクロコード、プログラム可能なロジックまたはハードコードされたロジックを含む。図1に示したプロセッサはプロセッサ118の1つのみであるが、システム中にはプロセッサが1以上含まれるとしてもよい。

30

【0024】

メモリ120は、ハードディスク、フロッピー（登録商標）ディスク、RAM（Random Access Memory）、ROM（Read Only Memory）、フラッシュメモリ、または上述のデバイスの組み合わせであってよい。または、プロセッサ118が読み取ることが出来る機械可読媒体であればどのようなものであってもよい。メモリ120は本発明の実施形態に係る方法を実行するための命令および／またはデータを格納するとしてもよい。

【0025】

40

VMM112は、ほかのソフトウェア（例えば「ゲスト」ソフトウェア）に対して、1以上の仮想マシン（VM）のアブストラクションを提示する。ここで、それぞれのゲストソフトウェアに対して同一もしくは異なるアブストラクションを提供するとしてもよい。図1には2つのVM（102および114）を示している。各VMで実行中のゲストソフトウェアには、ゲストOS104または106のようなゲストOSおよびさまざまなゲストソフトウェアアプリケーション108および110がある。ゲストOS104および106はそれぞれ、ゲストOS104または106を実行中でほかの機能も持つVM102および114内の物理リソース（例えばプロセッサレジスタ、メモリおよびI/Oデバイス）にアクセスすると考えられる。例えば、VMに示されたプロセッサおよびプラットフォームのアーキテクチャによると、ゲストOSはすべてのレジスタ、キャッシュ、構造、

50

I/Oデバイス、メモリなどに対するアクセスを持つと考えられる。

【0026】

ゲストソフトウェアがアクセス可能なリソースは、「特権状態」もしくは「非特権状態」にあるものに分類できるとしてもよい。特権状態のリソースに関しては、VMM 112はゲストソフトウェアが所望する機能へのアクセスを円滑化する一方、該特権状態リソースに対する最終的な制御は維持する。非特権状態のリソースについては、VMM 112が制御する必要はなく、ゲストソフトウェアがアクセスすることができる。

【0027】

また、ゲストOSはそれぞれフォールトイベントを処理すると考えられる。フォールトイベントの例を挙げると、例外（例えばページフォールト、一般保護違反、トラップ、異常終了など）、割り込み（例えばハードウェア割り込み、ソフトウェア割り込みなど）、プラットフォームイベント（例えば初期化（INIT）やシステム管理割り込み（SMI））などがある。上述のフォールトイベントの中には、ゲストソフトウェアから保護する目的で、VM 102および114が適切に動作するようにVMM 112が処理しなければならないので、「特権状態」となっているものもある。

【0028】

特権フォールトイベントが発生した場合、もしくはゲストソフトウェアが特権リソースへのアクセスを試みた場合、制御がVMM 112に移行することがある。ゲストソフトウェアからVMM 112への制御の移行を本明細書ではVM終了と定義する。リソースへのアクセスを円滑化した後、もしくはイベントを適切な方法で処理した後、VMM 112は制御をゲストソフトウェアに戻すとしてもよい。VMM 112からゲストソフトウェアへ制御の移行することをVM開始と呼ぶ。

【0029】

一実施形態によると、プロセッサ 118は仮想マシン制御構造（VMCS）124に格納されたデータに従ってVM 102および114の動作を制御する。VMCS 124は、ゲストソフトウェアの状態、VMM 112の状態、VMM 112がどのようにゲストソフトウェアの動作を制御したいかを示す実行制御情報、VMM 112とVM間での移行を制御するための情報などを有するとしてもよい。プロセッサ 118はVMCS 124から情報を読み出し、VMの実行環境を決定したり、VMの動作を制限したりする。一実施形態によると、VMCS 124はメモリ 120に格納されるとしてもよい。実施形態によっては、複数のVMを支持するべく複数のVMCSを用いるとしてもよい。

【0030】

VMの動作中にフォールトイベントが発生すると、プロセッサ 118は、このフォールトイベントがVM終了に対応するものかどうか（つまり、このフォールトイベントに対応付けられた実行制御指標が、このフォールトに対してVM終了を生じさせるように、VM終了値に設定されているかどうか）を判定する。フォールトイベントがVM終了に対応付けられている場合、プロセッサ 118はVMM 112に対してVM終了を発生し、フォールトイベントによってVM終了が発生したことを指し示す。VMM 112は続いて、自分でフォールトを処理するのか、それともこのフォールトを適切なVMに引渡すようプロセッサ 118に要求するのかを決定する。

【0031】

一実施形態によると、フォールトを引渡す場合は、転送構造を検索して引渡されるフォールトに対応付けられたエントリを見つけて、このフォールトを処理するように指定されたルーチンの位置を示す記述子をこのエントリから抽出し、抽出された記述子を用いてルーチンの最初まで飛ぶ。割り込みや例外などのフォールトを処理するべく指定されたルーチンはフォールトハンドラと呼ばれる。別の実施形態によると、フォールトを引渡す場合に転送構造を検索する必要はない。このような場合は、フォールトを処理するルーチンの位置を示す記述子は（例えば、ソフトウェアによって構成可能）1つのエントリに含まれるとしてもよいし、（例えば、メモリ内の所定位置に）ハードコードされているとしてもよい。

【0032】

フォールトを引渡す場合は、アドレス変換を1回以上行い、仮想アドレスから物理アドレスへと変換する必要があるとしてもよい。例えば、割り込みテーブルのアドレスまたは対応付けられたハンドラのアドレスは仮想アドレスの場合もある。また、プロセッサはフォールトの引渡し中にさまざまなチェックを行う必要がある場合もある。例えば、プロセッサは一貫性についてチェックを行うとしてもよい。例を挙げると、分割レジスタおよびアクセスアドレスの確認（違反フォールト、セグメント不在フォールト、スタックフォールトなどが抑制される）、および保護違反（例えば一般保護違反など）となる許容レベルチェックなどがある。

【0033】

フォールトを引渡ししている間にアドレス変換やさまざまなチェックを行うと、ページフォールトや一般保護違反などのフォールトが発生する可能性がある。このようなネストフォールトは適切なハンドラに引渡す必要がある。この新しいネストフォールトの引渡しを行うことによってさらにフォールトが発生する可能性がある。ISAによっては、このようなフォールトネスティングが、プロセッサ118がVMをシャットダウン状態にするネストフォールト最大許容レベルに達するまで、続発する場合もある。例としてIA-32 ISAを挙げると、三重にフォールトが発生するとプロセッサによってVMがシャットダウン状態にされる。さまざまなイベントの発生を機に、VMがシャットダウン状態から抜け出すとしてもよい。例としてIA-32 ISAの場合を説明すると、三重にフォールトが発生してシャットダウン状態にある場合、システム初期化メッセージ（INIT）、マスク不可能割り込み（NMI）、マシンチェック例外やシステム管理割り込み（SMI）が発生すると、プロセッサはシャットダウン状態から抜け出す。

【0034】

VMへのフォールトの引渡しは、プロセッサ118によって行われるとしてもよいし、ほかの実体、例えばVMM112が行うとしてもよい。プロセッサ118は、VMに対するVM開始の一環として該VMにフォールトを引渡すための機能をVMM112に提供するとしてもよい。このようにすることによって、VMM112はこの動作をエミュレートする必要がなくなる。

【0035】

一実施形態によると、プロセッサ118は、フォールトネスティングを記録するとともにさまざまなレベルで発生するフォールトに関連する情報を取得するネストフォールトロジック122を備える。このフォールトネスティング情報は、ネストフォールトによってVM終了が発生すると、VMM112によって使用されるとしてもよい。一実施形態によると、VMM112は、このフォールトネスティング情報に基づいてネストフォールトの処理を調整するフォールトコーディネータ126を備えるとしてもよい。例えば、割り込みの引渡し中にVMM112のページフォールトが発生すると（例えば、VMM112が物理メモリをVM間で分割および保護するべく物理メモリ仮想化アルゴリズムを用いたが、物理メモリ内のIDTが存在するページがページテーブルによってマッピングされていない場合）、VMM112に対してVM終了が発生する。続いて、VMM112のフォールトコーディネータ126は、このページフォールトを解決して、VMに制御を戻し、元々の割込みの引渡しを再開する（もしくは引渡しをエミュレートする）。フォールトコーディネータ126は、フォールトネスティング情報に含まれたこの元々の割込みについての情報に基づき、VMへ制御を戻すとともに元々の割込みの引渡しを再開する必要があると判定する。

【0036】

一実施形態によると、ネストフォールトロジック122は、さまざまなネストレベルに対応する一連のフィールドに基づき、ネストされたフォールトについての情報を取得する。一実施形態によると、許容されたネストレベルそれぞれについてフォールト情報を取得する。例えば、ネストレベルが4レベル許容されているとすると、この4レベルそれぞれについてフォールト情報を取得する。一実施形態によると、このフォールト情報は、フォ

ールト識別子およびフォールトタイプ（例えば外部割込み、内部割込み、マスク不可能割込み（NMI）、例外など）といったフォールト識別情報を含む。別の実施形態によると、フォールトはエラーコードと対応付けられ、このエラーコードは、ハンドラの最初まで飛ぶ前にスタックにプッシュされる（または、ハードウェアレジスタまたはそれ以外の手段に提供される）必要がある。このような実施形態の場合、フォールト情報はさらにエラーコードを含む。本実施形態によると、ネストフォールト用の一連のフィールドは各許容ネストレベルに対して2以上のフィールドを含むとしてもよい。この場合、1フィールドはフォールト識別情報用で、残りのフィールドはエラーコードやフォールトに関連するそれ以外の情報用であるとしてもよい。

【0037】

別の実施形態によると、フォールト情報を取得するのは最新の2つのフォールトについてのみである。例えば、許容されているネストレベルが4レベルあって、最初の3レベルではVM終了が発生することなく各許容ネストレベルでネストフォールトが発生した場合、フォールト情報がVMMに伝えられるのはレベル3とレベル4に対応付けられたフォールト（つまり、最新の2つのフォールト）のみである。繰り返しになるが、最新の2つのレベルそれぞれについてのフォールト情報は、フォールト識別子およびフォールトタイプのようなフォールト識別情報を含み、一実施形態によると、さらにフォールトに対応付けられたエラーコードも含む。最新2つより前に発生したフォールトに関するフォールト情報がないので、より多くのタイプのフォールトについてVMMがVMの動作を構成してVM終了を発生させ、ネストフォールト情報がなくならないようにする必要があるとしてもよい。これは、フォールト情報が存在しないとVMの再開が困難もしくは不可能になってしまうためである。

【0038】

フォールト情報を格納するべく1以上の格納場所が利用できる。一実施形態によると、このような格納場所はVMCS124内に存在する。または、プロセッサ118内、メモリ120内、メモリ120およびプロセッサ118の組み合わせ、もしくはこれ以外の格納場所にあってもよい。

【0039】

ある1つのフォールトに関するフォールト情報を格納するために利用できる一連の格納場所を本明細書では、フォールト情報フィールドと呼ぶ。または、簡単にフィールドと呼ぶ（例えば、フォールト識別情報の格納場所とエラーコードの格納場所が別々にあるように格納場所が2以上ある場合もまとめてフィールドと呼ぶ）。

【0040】

図2は、一実施形態に係る、仮想マシン環境においてネストされたフォールトへの支持を提供するためのプロセスを示すフローチャートである。このプロセスは、ハードウェア（例えば回路、専用ロジック、プログラム可能ロジック、マイクロコードなど）、ソフトウェア（例えば、汎用コンピュータシステムまたは専用マシンで実行されるもの）、もしくはハードウェアとソフトウェアの組み合わせから構成される処理ロジックによって行われるとしてもよい。一実施形態によると、このプロセスは図1に示したネストフォールトロジック122によって行われる。

【0041】

図2に示すように、プロセス200は処理ロジックがVMの動作中に最初のフォールト（フォールト1と呼ぶ）を検出することから始まる（処理ブロック202）。VMの動作中に検出されたフォールトは、例外（例えばページフォールト、一般保護違反、トラップ、異常終了など）、割り込み（例えばハードウェア割り込み、ソフトウェア割り込みなど）もしくはプラットフォームイベント（例えばINIT、SMIなど）であってよい。処理ロジックは、フォールトネストレベルを記録するべく変数nを1に初期化する（処理ブロック203）。

処理ブロック204において処理ロジックは、最初のフォールト（つまりフォールト1、n=1）に関連するフォールト情報をフォールト情報フィールド1（つまりフィールド

10

20

30

40

50

n) に格納する。この最初のフォールトもしくはそれ以外の本明細書で説明するフォールトに関連するフォールト情報は、フォールト識別子およびフォールトタイプ（例えば、外部割込み、内部割り込み、NM I、例外など）のように、フォールトを識別する情報を含む。また一実施形態によるとさらに、エラーコードおよび／またはフォールトに対応付けられるこれ以外の情報を含む。別の実施形態によると、処理ロジックは、2番目のフォールトが検出されるまで、最初のフォールトに関連する情報をフォールト情報フィールド1に格納しない構成となっている。または逆に、2番目のフォールトを検出する前に、一時的な格納場所（例えばメモリ、プロセッサレジスタなど）に最初のフォールトに関連する情報を格納する構成となっている。

【0042】

10

判定ブロック206において、処理ロジックはフォールトnがVM終了と対応しているかどうか（つまり、対応付けられた実行制御指標が、フォールトnがVM終了を発生させるようにVM終了値に、設定されているかどうか）を判定する。フォールトnがVM終了と対応する場合、処理ロジックはVM終了を発生させる（処理ブロック210）。この判定は、フォールト識別子（つまり、フォールトベクトル）、フォールトに対応付けられたエラーコード、1以上の指標またはVMCS内のフィールドなどに基づいて行われるとしてもよい。

【0043】

フォールトnがVM終了と対応していない場合、処理ロジックはフォールトnに対応するネストレベル（つまりネストレベルn）が最大許容ネストレベルを超えているかどうか判定する（判定ボックス216）。最大許容ネストレベルを超えている場合、処理ロジックは、シャットダウン状態への移行によってVM終了が発生するように構成されているかどうかを判定する（判定ボックス217）。もしそのように構成されていれば、処理ロジックはVM終了を発生させ、VM終了は保留中のシャットダウン状態への移行によって発生したことをVMMに知らせる（処理ブロック224）。シャットダウン状態への移行によってVM終了が発生するように構成されていない場合、処理ロジックはVMをシャットダウン状態にする（処理ブロック218）。別の実施形態においては、保留中のシャットダウン状態への移行は無条件でVM終了を発生させる。また別の実施形態においては、シャットダウン状態への移行がVM終了を発生させることはない。このような実施形態の場合は、最大許容ネストレベルを超えている場合常にVMがシャットダウン状態となる。

20

30

【0044】

（判定ボックス216で）最大許容ネストレベルを超えていない（と判定された）場合、処理ロジックはフォールトnをVMに引渡すことを試みて（処理ブロック210）、フォールトnの引渡し中に新たにフォールト（フォールトn+1）が発生しないかどうか判定する（判定ボックス212）。

【0045】

フォールトnの引渡し中に新たなフォールトが発生しなかった場合、処理ロジックは制御をフォールトnに対応付けられたハンドラの第1命令へ移行する（処理ブロック214）。一方、フォールトnの引渡し中にフォールトn+1が発生した場合、処理ロジックはフォールトn+1のフォールト情報をフィールドn+1に格納する（処理ブロック220）。また、処理ブロックはnを1増やし（処理ブロック222）、判定ボックス206へと戻る。

40

【0046】

以上より、プロセス200では、すべての許容ネストレベルで発生したネストフォールトについても情報を取得し格納する。発生したフォールトすべてに関連する情報が1以上のフォールト情報フィールドに格納される。

【0047】

図3は、一実施形態に係る、仮想マシン環境においてネストされたフォールトを支持するためのプロセス300を示すフローチャートである。このプロセスは、ハードウェア（例えば回路、専用ロジック、プログラム可能ロジック、マイクロコードなど）、ソフトウ

50

ウェア（例えば、汎用コンピュータシステムまたは専用マシンで実行されるもの）、もしくはハードウェアとソフトウェアの組み合わせから構成される処理ロジックによって行われるとしてもよい。一実施形態によると、このプロセス300は図1に示したネストフォールトロジック122によって行われる。

【0048】

図3に示すように、プロセス300は、処理ロジックがフォールトをVMに引き渡す必要があると判定することから始まる（処理ブロック301）。処理ロジックはレベルカウンタを1に初期化する（処理ブロック302）。続いて処理ロジックは、引渡される現行フォールトについてのフォールト情報を現行フォールト情報フィールドに格納し、先行フォールトは発生していない旨を指し示すべく先行フォールト情報フィールドを初期化する（処理ブロック304）。先行フォールトフィールドの初期化は、例えば、該フォールト情報フィールドが有効であることを示すビットをクリアすることによって行うとしてもよい。または、フォールト情報が該フィールドに含まれていないことを指し示すには、これ以外のエンコード方法を用いるとしてもよい。

【0049】

続いて、処理ロジックは現行フォールトの引渡しをVM開始の一環として行われているかどうか判定する（判定ボックス306）。VM開始の一環として行われていなければ、処理ロジックは現行フォールトがVM終了を発生させるように構成されているかどうかを判定する（判定ボックス308）。一実施形態によると、この判定はVMCSに格納されている1以上の実行制御を参照して行われる。現行フォールトによってVM終了が発生するのであれば、処理ロジックはVM終了を発生させる（処理ブロック314）。一実施形態によると、このVM終了の一環として、処理ロジックは現行フォールト情報フィールドおよび先行フォールト情報フィールドをVMMに提供する。一実施形態によると、これらのフォールト情報フィールドは、VMMがアクセスできるVMCSにある。公知の機構（例えば1以上のレジスタまたは記憶場所）に基づき、VMMがこれらのフォールト情報フィールドを利用できるようにするとしてもよい。

【0050】

現行フォールトによってVM終了が発生するように構成されていなければ、処理ロジックはレベルカウンタが最大許容ネストレベルと同じ値を示しているかどうか判定する（判定ボックス310）。この判定は可変レベル（処理ブロック302で初期化され、処理ブロック316で増やされている）に基づいて行われる。IA-32 ISAを例に挙げると、最大許容フォールトネストレベルは3となっている。最大許容フォールトネストレベルに達していれば、処理ロジックはフォールトシャットダウン状態へ移行するとVM終了が発生するのかどうか判定する（処理ボックス311）。一実施形態によると、この判定は、VMCSにある1以上の実行制御を参照することによって行われるとしてもよい。VM終了が発生するとされていれば、処理ロジックはVMMに対してVM終了を発生させ、保留中のフォールトシャットダウン状態への移行によってVM終了が発生したことを指し示す（処理ブロック326）。VM終了が発生するとされていなければ、処理ロジックはVMをフォールトシャットダウン状態にする（処理ブロック312）。別の実施形態によれば、保留中のシャットダウン状態への移行は無条件でVM終了を発生させる。また別の実施形態においては、シャットダウン状態への移行がVM終了を発生させることはない。このような実施形態の場合は、最大許容ネストレベルを超えている場合にVMが常にシャットダウン状態となる。

【0051】

（処理ボックス310で）最大許容フォールトネストレベルを超えていない（と判定された）場合、処理ロジックは現行フォールトをVMに引渡すことを試みて（処理ブロック318）、現行フォールトの引渡し中に新たにフォールトが発生しないかどうか判定する（判定ボックス320）。新たなフォールトが発生しなかった場合、処理ロジックは制御を現行フォールトに対応付けられたハンドラの第1命令へ移行する（処理ブロック322）。一方、新たなフォールトが発生した場合、処理ロジックはレベルカウンタを1増やし

10

20

30

40

50

(処理ブロック316)、現行フォールト情報フィールドから先行フォールト情報フィールドにデータを移し(処理ブロック324)、判定ボックス308へ進む。

【0052】

上述したように実施形態によっては、VM開始において、VMMがプロセッサに対してVM開始処理の一環としてフォールトをVMに引渡さなければならないと指示するとしてもよい。処理ロジックが判定ボックス306において現行フォールトの引渡しが行われるVM開始の一環として行われていると判定した場合、現行フォールトがVM終了を発生させることはない。処理ロジックは処理ブロック308においてチェックを行わない。さらに、図3に示した実施形態によると、VMMは最初のフォールトを入れるのみであるとしてもよい。このため、処理ブロック306のチェックにより、最初のフォールトの引渡しが行われるVM開始の一環として行われていることがわかれば、処理ブロック310のチェックは行われない。処理ロジックは直接処理ブロック318に進む。

10

【0053】

ISAによっては、幾つかの条件下でネストフォールト情報を保持できないことがある。また、上述した本発明の実施形態で説明した処理よりも複雑な経路でシャットダウン状態に進むISAもあるとしてよい。IA-32 ISAを例に挙げて説明すると、ページフォールトと寄与フォールトの組み合わせによっては二重フォールトが発生する。二重フォールトが発生すると、先行するフォールトの情報がなくなってしまう。二重フォールトの後にフォールトが発生すれば三重フォールトになる可能性があり、先行するすべてのフォールトに関連する情報が失われてしまう。IA-32 ISAでは、良性と分類される例外があり、その場合は二重フォールトや三重フォールトが発生することはない。またさらに、IA-32 ISAでは、マシンチェック例外が連続して発生するとシャットダウン状態への移行が開始されるが、マシンチェックによって二重フォールトや三重フォールトが発生することはない。さらに、IDTを介したフォールトの引渡しによって、タスクスイッチが発生する可能性がある。VMを実行している間にフォールトが発生した場合、一実施形態によると、このタスクスイッチが契機となってVMMに対してVM終了が発生することもある。このようなISA固有の機能は、図2や図3では反映されていない。

20

【0054】

図4は、一実施形態に係る、仮想マシン環境でネストフォールトを処理するためのプロセス400を示すフローチャートである。このプロセスは、ハードウェア(例えば回路、専用ロジック、プログラム可能ロジック、マイクロコードなど)、ソフトウェア(例えば、汎用コンピュータシステムまたは専用マシンで実行されるもの)、もしくはハードウェアとソフトウェアの組み合わせから構成される処理ロジックによって行われるとしてもよい。一実施形態によると、このプロセス400は図1に示したVMM112によって行われる。

30

【0055】

図4に示すように、プロセス400は処理ロジックがVM終了の結果として制御を受信することから始まる(処理ブロック402)。続いて、処理ロジックは、このVM終了がフォールトによって発生したと判定し(処理ブロック404)、このフォールトがネストフォールトであると判定する(つまりこのフォールトは、元々のフォールトを引渡している間に発生したもの)(処理ブロック406)。一実施形態によると、これらの判定は、プロセッサによって提供される、VMCSの終了情報フィールドに格納されたデータを検証することによって行われるとしてもよい。

40

【0056】

さらに、処理ロジックはこのVM終了を発生させたフォールトを処理し(処理ブロック408)、元々のフォールトの引渡しを再開またはVMへの元々のフォールトの引渡しをエミュレートし(処理ブロック412)、制御をVMに戻す(処理ブロック410)。上述したように、元々のフォールトの引渡しの再開は、例えば、VMMがソフトウェアでフォールトの引渡しをエミュレートすることによって行ってもよいし、VMMがゲストに制御を戻す処理の一環としてフォールトの引渡しを行うようにプロセッサに要求することに

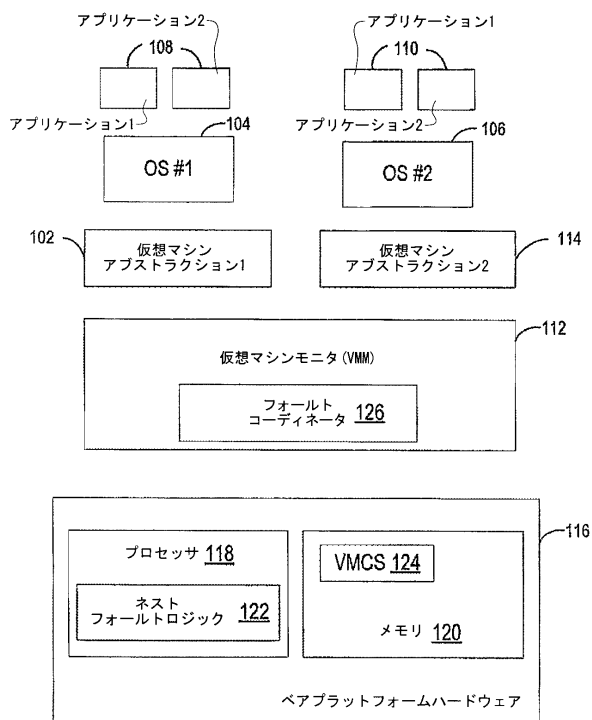
50

よって行うとしてもよい。

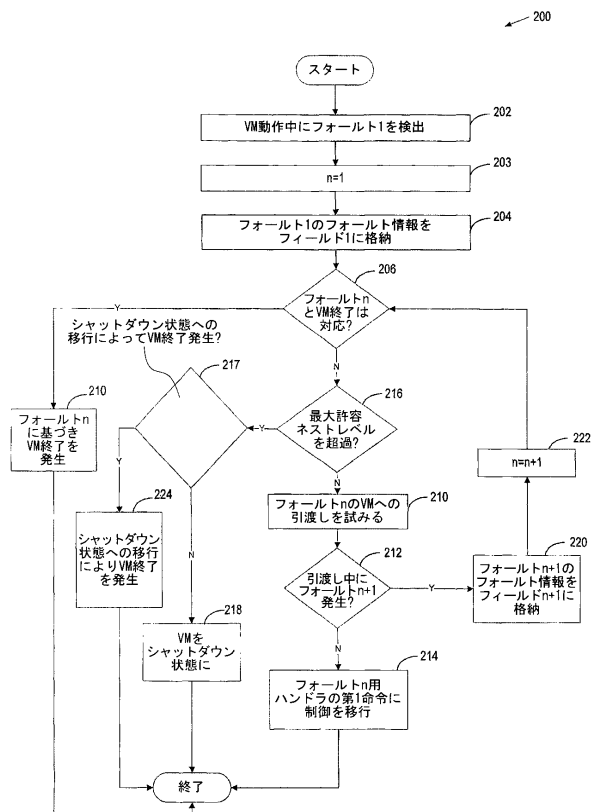
【0057】

以上の部分では、仮想マシン環境においてネストされたフォールトに対する支持を提供するための方法および装置を説明してきた。上述した実施形態は本発明を説明するために挙げられたもので、本発明を限定するものではない。本明細書の内容に基づき、上述の実施形態以外の実施形態を数多く実現できることは当業者には明らかである。このため本発明の範囲は、本願の請求の範囲およびそれに類するものによって決まるものである。

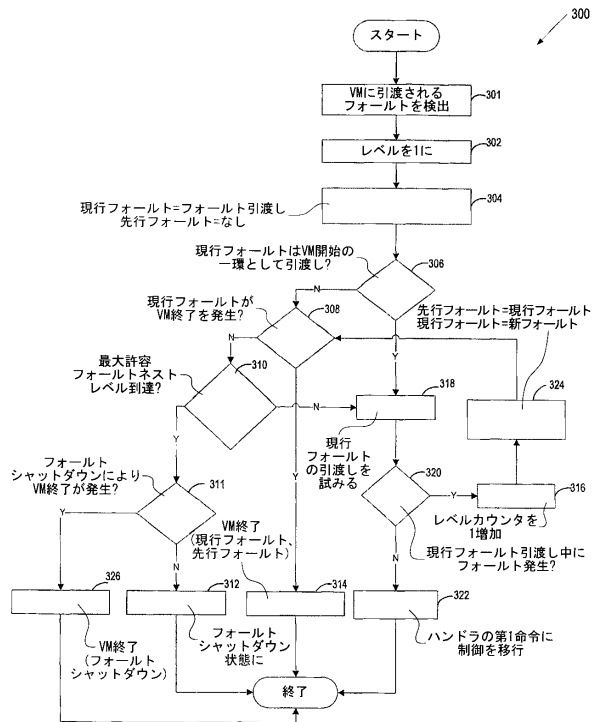
【図1】



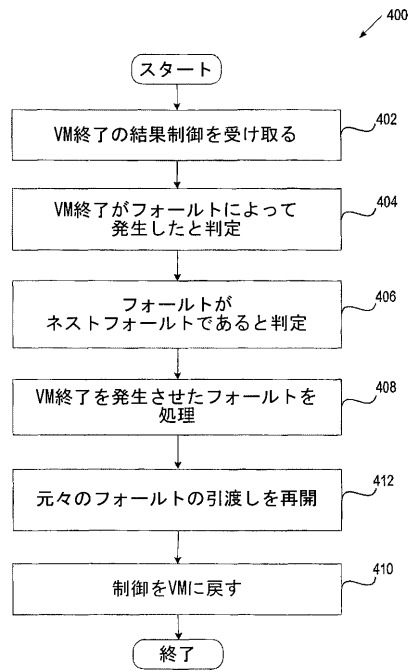
【図2】



【図 3】



【図 4】



フロントページの続き

- (72)発明者 ネイガー、ギルバート
アメリカ合衆国、97212 オレゴン州、ポートランド、ノースイースト 11ティーエイチ
アベニュー 2424
- (72)発明者 コターロブレス、エリック
アメリカ合衆国、97225 オレゴン州、ポートランド、サウスウエスト レーンウッド スト
リート 11790
- (72)発明者 ベネット、スティーブン、エム
アメリカ合衆国、97123 オレゴン州、ヒルズバロ、サウスイースト、シグリッド ストリー
ト 6469
- (72)発明者 ジェヤシング、スターリンセルバラジ
アメリカ合衆国、97006 オレゴン州、ビーバートン、ノースウエスト トゥーソン 168
93
- (72)発明者 ウーリッヒ、リチャード、エー
アメリカ合衆国、97124 オレゴン州、ヒルズバロ、ノースイースト オレンコ ステーショ
ン パークウェイ 1564
- (72)発明者 ブラント、ジェーソン
アメリカ合衆国、78703 テキサス州、オースティン、サミット ビュー プレース 171
3ビー
- (72)発明者 パルタサラティ、ラジェッシュ
アメリカ合衆国、97123 オレゴン州、ヒルズバロ、ノースイースト カーラビー ウェイ
ナンバー 77 1451
- (72)発明者 モンダル、サンジョイ、ケー
アメリカ合衆国、78704 テキサス州、オースティン、サウス ラマー ブールバード ナン
バー 402 3816
- (72)発明者 ケーギ、アライン
アメリカ合衆国、97202 オレゴン州、ポートランド、サウスイースト クリスタル スプリ
ングス ブールバード 3624
- (72)発明者 スミス、ローレンス、オー
アメリカ合衆国、97006 オレゴン州、ビーバートン、ノースウエスト ノーサンブリア 1
4995

審査官 大塚 良平

- (56)参考文献 特開2000-347883 (JP, A)
特開昭61-259343 (JP, A)
特開平10-63513 (JP, A)
特開平6-168146 (JP, A)
特開平1-303529 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
G06F 9/46 - 9/54