(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4486288号
(P4486288)
(45)発行日　平成22年6月23日(2010.6.23)　(24)登録日　平成22年4月2日(2010.4.2)

(51)Int.Cl.　　F I
G06F 9/445 (2006.01)　　G06F 9/06 610K
G06F 1/24 (2006.01)　　G06F 1/00 350C

請求項の数 8　(全 29 頁)

(21)出願番号　特願2001-357612(P2001-357612)
(22)出願日　平成13年11月22日(2001.11.22)
(65)公開番号　特開2002-287978(P2002-287978A)
(43)公開日　平成14年10月4日(2002.10.4)
審査請求日　平成16年11月22日(2004.11.22)
審判番号　不服2007-14636(P2007-14636/J1)
審判請求日　平成19年5月21日(2007.5.21)
(31)優先権主張番号　09/721,398
(32)優先日　平成12年11月22日(2000.11.22)
(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
(74)復代理人　100129171
弁理士　柿沼　健一
(72)発明者　ポール　イングランド
アメリカ合衆国　９８００８　ワシントン州　ベルビュー　ノースラップ　ウェイ　１６６５９

最終頁に続く

(54)【発明の名称】コンピュータにおいてトラステッドコア初期化プロセスを安全に実行するためのプログラム、方法、メモリコントローラ、装置及びコンピュータ

(57)【特許請求の範囲】
【請求項１】
１つまたは複数の中央処理装置、１つまたは複数のバスマスタ、メモリ及びメモリコントローラを備えたコンピュータにおいて、トラステッドコア初期化プロセスを実行するためのプログラムであって、
前記コンピュータが、必ずしも安全ではないコードに基づいて動作している間に、前記必ずしも安全ではないコードの制御下で、前記メモリコントローラが、トラステッドコアをメモリにロードするステップと、
前記メモリコントローラが、前記トラステッドコアが正常に前記メモリにロードされた後に、前記コンピュータにおける前記１つまたは複数の中央処理装置の各々および前記１つまたは複数のバスマスタの各々が前記メモリにアクセスするのを防止するステップと、
前記メモリへのアクセスが防止された後に、前記１つまたは複数の中央処理装置のうちの１つをリセットするステップであって、
前記メモリコントローラが、中央処理装置リセットベクトルを、初期化ベクトルにマップするステップと、
前記メモリコントローラが、前記中央処理装置リセットベクトルに対応する読取り要求を、前記１つまたは複数の中央処理装置のうちの１つから受信するステップと、
前記メモリコントローラが、前記読取り要求に応答して、前記初期化ベクトルを前記１つまたは複数の中央処理装置のうちの１つに返すステップと、
前記メモリコントローラが、前記１つまたは複数の中央処理装置のうちの１つが、前

記初期化ベクトルで始まる前記メモリにアクセスできるようにするステップと
を含む、ステップと、
前記１つまたは複数の中央処理装置のうちの１つがリセットされた後に、前記メモリコントローラが、前記１つまたは複数の中央処理装置のうちの１つが前記メモリにアクセスして前記トラステッドコアに含まれるトラステッドコア初期化コードを読み出すことができるようにするステップであって、前記１つまたは複数の中央処理装置のうちの１つが前記トラステッドコア初期化コードを実行することにより、前記トラステッドコアを初期化することができるようにする、ステップと、
前記トラステッドコアが初期化された後、前記メモリコントローラが、前記コンピュータにおける他のいかなる中央処理装置およびいかなるバスマスタも前記メモリにアクセスできるようにするステップと
を順に実行させるためのプログラム。
【請求項２】
コンピュータに、
前記初期化ベクトルを前記１つの中央処理装置に返した後に、前記メモリコントローラが、前記中央処理装置リセットベクトルを追加の中央処理装置開始ベクトルに再マップするステップと、
前記メモリコントローラが、別の中央処理装置からの、前記中央処理装置リセットベクトルに対応する、他のいかなる読取り要求にも応答して、前記追加の中央処理装置開始ベクトルを返すステップとをさらに実行させることを特徴とする請求項１に記載のプログラム。
【請求項３】
コンピュータに、
前記中央処理装置をリセットした後に、マイクロコードを、前記メモリにおけるトラステッドコアから前記１つの中央処理装置へロードするステップをさらに実行させるための請求項１に記載のプログラム。
【請求項４】
１つまたは複数の中央処理装置、１つまたは複数のバスマスタ、メモリ及びメモリコントローラを備えたコンピュータにおいて、トラステッドコア初期化プロセスを実行するための方法であって、
前記コンピュータが、必ずしも安全ではないコードに基づいて動作している間に、前記必ずしも安全ではないコードの制御下で、前記メモリコントローラが、トラステッドコアをメモリにロードするステップと、
前記メモリコントローラが、前記トラステッドコアが正常に前記メモリにロードされた後に、前記コンピュータにおける前記１つまたは複数の中央処理装置の各々および前記１つまたは複数のバスマスタの各々が前記メモリにアクセスするのを防止するステップと、
前記メモリへのアクセスが防止された後に、前記１つまたは複数の中央処理装置のうちの１つをリセットするステップであって、
前記メモリコントローラが、中央処理装置リセットベクトルを、初期化ベクトルにマップするステップと、
前記メモリコントローラが、前記中央処理装置リセットベクトルに対応する読取り要求を、前記１つまたは複数の中央処理装置のうちの１つから受信するステップと、
前記メモリコントローラが、前記読取り要求に応答して、前記初期化ベクトルを前記１つまたは複数の中央処理装置のうちの１つに返すステップと、
前記メモリコントローラが、前記１つまたは複数の中央処理装置のうちの１つが、前記初期化ベクトルで始まる前記メモリにアクセスできるようにするステップと
を含む、ステップと、
前記１つまたは複数の中央処理装置のうちの１つがリセットされた後に、前記メモリコントローラが、前記１つまたは複数の中央処理装置のうちの１つが前記メモリにアクセスして前記トラステッドコアに含まれるトラステッドコア初期化コードを読み出すことがで

きるようにするステップであって、前記１つまたは複数の中央処理装置のうちの１つが前記トラステッドコア初期化コードを実行することにより、前記トラステッドコアを初期化することができるようにする、ステップと、
前記トラステッドコアが初期化された後、前記メモリコントローラが、前記コンピュータにおける他のいかなる中央処理装置およびいかなるバスマスタも前記メモリにアクセスできるようにするステップと
を有することを特徴とする方法。
【請求項５】
前記１つまたは複数の中央処理装置が、前記トラステッドコアの実行を終了するステップと、
前記１つまたは複数の中央処理装置が、前記トラステッドコアの実行を、前記コンピュータをリブートすることなく、再び開始するステップと、
をさらに含むことを特徴とする請求項４に記載の方法。
【請求項６】
前記１つまたは複数の中央処理装置が、前記トラステッドコアの実行を終了するステップと、
前記メモリコントローラが、別のトラステッドコアをメモリにロードするステップと、
前記メモリコントローラが、前記トラステッドコアの実行を開始した手順と同様の手順で、前記別のトラステッドコアの実行を開始するステップと、
をさらに含むことを特徴とする請求項４に記載の方法。
【請求項７】
前記メモリコントローラが、前記初期化ベクトルを前記１つの中央処理装置に返した後に、前記中央処理装置リセットベクトルを追加の中央処理装置開始ベクトルに再マップするステップと、
前記メモリコントローラが、前記１つまたは複数の中央処理装置の別の中央処理装置からの、前記中央処理装置リセットベクトルに対応する、他のいかなる読取り要求にも応答して、前記追加の中央処理装置開始ベクトルを返すステップと、
をさらに含むことを特徴とする請求項４に記載の方法。
【請求項８】
請求項１乃至３のいずれかに記載のプログラムを記録したコンピュータ読取可能な記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、安全なコード実行に関し、より詳細には、コンピュータにおいてトラステッドコア初期化プロセスを安全に実行するためのプログラム、方法、メモリコントローラ、装置及びコンピュータに関する。
【０００２】
【従来の技術】
人々がコンピュータを信用できるようにすることは、ますます重要なゴールとなってきている。この信用は一般に、コンピュータが格納あるいは受信する情報を正しく使用することを信用できることに集中している。厳密にこの信用が何を内含するかは、環境に基づいて変わる可能性がある。たとえば、マルチメディアコンテンツプロバイダは、コンピュータが不適切にそれらのコンテンツをコピーしないことを信用できることを望む。もう１つの例として、ユーザは自分のコンピュータが内密の金融情報（たとえば、銀行口座番号）を適切な宛先のみに転送すること（たとえば、この情報を自分の銀行に渡すことはできるが、それ以外のところには渡すことができないようにすること）を信用できることを望む。残念ながら、大抵のコンピュータの一般に開かれた特性が与えられると、幅広い範囲のアプリケーションを大抵の現在のコンピュータ上で、ユーザが知ることなく実行させることができ、これらのアプリケーションがこの信用を損なう（たとえば、ユーザの金融情報

を、ある他の宛先へ不当な使用のために転送する）おそれがある。

【０００３】

これらの信用の問題に対処するため、コンピュータまたはその一部を信用できるようにする、異なる機構が提案されている（また、新しい機構が開発されている）。一般に、これらの機構は、ある種類の認証手順を内含しており、コンピュータが、少なくともその一部（たとえば、メモリのある領域、あるアプリケーションなど）が少なくともそれらがそれら自体を提示するものと同じように信用できること（たとえば、コンピュータまたはアプリケーションが実際に、それが主張するものであること）を認証あるいは証明することができる。すなわち、これらの機構は、不当なアプリケーションが別のアプリケーションのふりをすること（あるいは、コンピュータが別のコンピュータのふりをできるようにすること）を防止する。このような機構を確立することができるようになった後、ユーザまたは他者（たとえば、コンテンツプロバイダ）が、特定のアプリケーションを信用できるものとして受け入れるかどうかについての判断を行うことができる（たとえば、コンピュータが、特定のアプリケーションがそれ自体の主張するアプリケーションであるという、コンテンツプロバイダの満足を証明することができるようになった後、マルチメディアコンテンツプロバイダが、特定のアプリケーションを信用できるものとして受け入れることができる）。しかし、このような機構をコンピュータ上に設置することは困難である可能性があり、これは、この機構に干渉する不当なアプリケーション（たとえば、信用されている機構のふりをする不当なアプリケーション）に対する保護が必要となるからである。

【０００４】

１つの解決策は、コンピュータをブートするための信用できる機構を含むコンピュータを構築することである。しかし、コンピュータをブートすることは、典型的には、様々なコード片を使用することを含み、これはしばしば基本入出力システム（ＢＩＯＳ）と呼ばれ、また、潜在的に多数のオプションの読取り専用メモリ（ＲＯＭ）またはＢＩＯＳ拡張機能を使用することを含み、これはオペレーティングシステムをある他の記憶デバイス（典型的にはハードディスクドライブ）からロードするように動作する。したがって、コンピュータをブートするための信用できる機構は、ＢＩＯＳが信用できるものであり、オプションの各ＲＯＭが信用できるものであり、各ＢＩＯＳ拡張機能が信用できるものであることを、信用できるオペレーティングシステムをコンピュータにロードできるようになる前に必要とする。これらの各構成要素を信用できるものにすることが困難であるだけでなく、ＢＩＯＳ、オプションのＲＯＭ、およびＢＩＯＳ拡張機能が頻繁に、コンピュータにおいて再書き込みすることができるデバイス（たとえば、フラッシュメモリ）上に格納され、したがって、これらの保全性が損なわれる。したがって、コンピュータをブートするための信用できる機構を含むコンピュータを構築することは、問題がある可能性がある。

【０００５】

【発明が解決しようとする課題】

よって本発明の目的は、これらの欠点に対処し、コンピュータにおいてトラステッドコア初期化プロセスを安全に実行するためのプログラム、方法、メモリコントローラ、装置及びコンピュータを提供することにある。

【０００６】

【課題を解決するための手段】

コンピュータにおいてトラステッドコア初期化プロセスを安全に実行するためのプログラム、方法、メモリコントローラ、装置及びコンピュータが本明細書に記載されている。

【０００７】

本発明の一形態によれば、メモリコントローラが、ＣＰＵおよび他のＩ／Ｏバスマスタがコード（たとえば、ＯＳ、マイクロカーネル、または他のトラステッドコア）初期化プロセス中にメモリにアクセスするのを防止する。メモリコントローラは、ＣＰＵをコンピュータにおいてリセットし、ＣＰＵがメモリの特定の場所（ＣＰＵに対してメモリコントローラによって識別される）でアクセスを開始できるようにする。初期化プロセスがそのＣＰＵによって実行された後、コードが動作状態であり、他のいかなるＣＰＵも（リセット

された後に）メモリにアクセスすることができ、（開始されたコードによって課せられたいかなる制御も受ける）他のいかなるバスマスタもメモリにアクセスすることができる。

【０００８】

請求項１に係る本発明は、コンピュータに、暴走する可能性のある信用されていないコードに基づいて前記コンピュータの動作が正常に開始された場合、前記信用されていないコードの制御下で、トラステッドコアをメモリにロードするステップと、前記トラステッドコアが正常に前記メモリにロードされた場合、前記コンピュータにおける１つまたは複数の中央処理装置の各々および１つまたは複数のバスマスタの各々が前記メモリにアクセスするのを防止するステップと、前記メモリへのアクセスが防止された後に、前記１つまたは複数の中央処理装置の各々をリセットするステップと、前記リセットがなされた後に、１つの中央処理装置が前記メモリにアクセスして、前記トラステッドコアに含まれるトラステッドコア初期化コードを実行することにより、前記トラステッドコアを初期化するステップと、前記トラステッドコアが初期化された後、前記コンピュータにおける他のいかなる中央処理装置およびいかなるバスマスタも前記メモリにアクセスできるようにするステップとを順に実行させるための１又は複数の実行制御用のプログラムである。

請求項２に係る本発明は、コンピュータに、中央処理装置リセットベクトルを、初期化ベクトルにマップするステップと、前記中央処理装置リセットベクトルに対応する読取り要求を、前記１つの中央処理装置から受信するステップと、前記読取り要求に応答して、前記初期化ベクトルを前記１つの中央処理装置に返すステップと、前記１つの中央処理装置が、前記初期化ベクトルで開始して、前記メモリにアクセスできるようにするステップをさらに実行させるための請求項１に記載のプログラムである。

請求項３に係る本発明は、コンピュータに、前記初期化ベクトルを前記１つの中央処理装置に返した後に、前記中央処理装置リセットベクトルを追加の中央処理装置開始ベクトルに再マップするステップと、別の中央処理装置からの、前記中央処理装置リセットベクトルに対応する、他のいかなる読取り要求にも応答して、前記追加の中央処理装置開始ベクトルを返すステップとをさらに実行させるための請求項２に記載のプログラムである。

請求項４に係る本発明は、コンピュータに、前記中央処理装置をリセットした後に、マイクロコードを、前記メモリにおけるトラステッドコアから前記１つの中央処理装置へロードするステップをさらに実行させるための請求項１に記載のプログラムである。

請求項５に係る本発明は、メモリコントローラ、メモリ、トラステッドコア、中央処理装置及びバスマスタを備えたコンピュータが行う安全な初期化方法において、前記中央処理装置が、暴走する可能性のある信用されていないコードに基づいて、前記コンピュータをブートするステップと、前記ブートが正常に行われた場合、前記メモリコントローラが、トラステッドコアをメモリにロードするステップと、前記トラステッドコアが正常に前記メモリにロードされた場合、前記トラステッドコアの安全な実行を開始するにあたり、前記メモリコントローラが、前記コンピュータにおける１または複数の中央処理装置の各々が前記メモリにアクセスするのを防止するステップと、前記メモリコントローラが、前記コンピュータにおける１または複数のバスマスタの各々が前記メモリにアクセスするのを防止するステップと、前記メモリコントローラが、前記１または複数の中央処理装置の各々が前記メモリにアクセスするのを防止された後であって、かつ、前記１または複数のバスマスタの各々が前記メモリにアクセスするのを防止された後に、前記１または複数の中央処理装置の各々をリセットするステップと、前記メモリコントローラが、前記リセットの後に、前記１または複数の中央処理装置の各々が前記メモリにアクセスして前記トラステッドコアを読み込み、該トラステッドコアに含まれるトラステッドコア初期化コードを実行するトラステッドコア初期化プロセスを実行できるようにするステップと、前記トラステッドコア初期化プロセスが実行された後、前記メモリコントローラが、他のいかなる中央処理装置およびいかなる前記１または複数のバスマスタも前記メモリにアクセスできるようにするステップと、を含むことを特徴とする方法である。

請求項６に係る本発明は、前記トラステッドコアが、前記トラステッドコアの実行を終了するステップと、前記トラステッドコアが、前記トラステッドコアの実行を、前記コン

ピュータをリブートすることなく、再び開始するステップと、をさらに含むことを特徴とする請求項５に記載の方法である。

請求項７に係る本発明は、前記トラステッドコアが、前記トラステッドコアの実行を終了するステップと、前記メモリコントローラが、別のトラステッドコアをメモリにロードするステップと、前記メモリコントローラが、前記トラステッドコアの実行を開始した手順と同様の手順で、前記別のトラステッドコアの実行を開始するステップと、をさらに含むことを特徴とする請求項５に記載の方法である。

請求項８に係る本発明は、中央処理装置が、暴走する可能性のある信用できないコードに基づいて、コンピュータをブートするステップと、正常に前記ブートが行われた場合、メモリコントローラが、トラステッドコアをメモリにロードするステップと、前記トラステッドコアが正常に前記メモリにロードされた場合、前記メモリコントローラが、前記トラステッドコアの安全な実行を開始するにあたり、中央処理装置リセットベクトルを、初期化ベクトルにマップするステップと、前記メモリコントローラが、前記コンピュータにおける１または複数の中央処理装置の各々をリセットするステップと、前記メモリコントローラが、前記マップおよび前記リセットの後に、前記中央処理装置リセットベクトルに対応する読取り要求を、前記１つまたは複数の中央処理装置のいずれか１つから受信するステップと、前記メモリコントローラが、前記読取り要求に応答して、前記初期化ベクトルを前記１つの中央処理装置に返すステップと、前記メモリコントローラが、前記１つの中央処理装置が前記初期化ベクトルで開始して前記メモリにアクセスすることを許可するステップと、を含むことを特徴とする方法である。

請求項９に係る本発明は、前記メモリコントローラが、前記初期化ベクトルを前記１つの中央処理装置に返した後に、前記中央処理装置リセットベクトルを追加の中央処理装置開始ベクトルに再マップするステップと、前記メモリコントローラが、前記１つまたは複数の中央処理装置の別の中央処理装置からの、前記中央処理装置リセットベクトルに対応する、他のいかなる読取り要求にも応答して、前記追加の中央処理装置開始ベクトルを返すステップと、をさらに含むことを特徴とする請求項８に記載の方法である。

請求項１０に係る本発明は、コンピュータに、暴走する可能性のある信用されていないコードに基づいて前記コンピュータが正常にブートされた場合、トラステッドコアをメモリにロードするステップと、前記トラステッドコアが正常に前記メモリにロードされた場合、前記トラステッドコアの安全な実行を開始するにあたり、前記コンピュータにおける１または複数の中央処理装置の各々が前記メモリにアクセスするのを防止するステップと、前記コンピュータにおける１または複数のバスマスタの各々が前記メモリにアクセスするのを防止するステップと、前記１または複数の中央処理装置の各々が前記メモリにアクセスするのを防止された後であって、かつ、前記１または複数のバスマスタの各々が前記メモリにアクセスするのを防止された後に、前記１または複数の中央処理装置の各々をリセットするステップと、前記リセットの後に、前記１または複数の中央処理装置の各々が前記メモリにアクセスして前記トラステッドコアを読み込み、該トラステッドコアに含まれるトラステッドコア初期化コードを実行するトラステッドコア初期化プロセスを実行できるようにするステップと、前記トラステッドコア初期化プロセスが実行された後、他のいかなる中央処理装置およびいかなる前記１または複数のバスマスタも前記メモリにアクセスできるようにするステップと、を順に実行させるための１又は複数の実行制御用のコンピュータプログラムを記録したコンピュータ読取可能な記録媒体である。

請求項１１に係る本発明は、暴走する可能性のある信用できないコードに基づいて、動作を開始するステップと、前記動作が正常に開始された場合、前記コンピュータが、前記信用できないコードの制御下で、任意の種類のコードをメモリにロードするステップと、前記トラステッドコアが正常にメモリにロードされた場合、前記コンピュータが、前記任意の種類のコードの実行を、請求項５に記載されたトラステッドコア初期化プロセスの実行と同様の手順で開始するステップと、を含むことを特徴とする方法である。

請求項１２に係る本発明は、前記コンピュータが、中央処理装置リセットベクトルを、初期化ベクトルにマップするステップと、前記コンピュータが備えるメモリコントローラ

が、前記中央処理装置リセットベクトルに対応する読取り要求を、前記コンピュータが備える１つの中央処理装置から受信するステップと、前記メモリコントローラが、前記読取り要求に応答して、前記初期化ベクトルを前記１つの中央処理装置に返すステップと、

前記メモリコントローラが、前記１つの中央処理装置が前記初期化ベクトルで開始して前記メモリにアクセスすることを許可するステップと、をさらに含むことを特徴とする請求項１１に記載の方法である。

請求項１３に係る本発明は、前記メモリコントローラが、前記初期化ベクトルを前記１つの中央処理装置に返した後に、前記中央処理装置リセットベクトルを追加の中央処理装置開始ベクトルに再マップするステップと、前記メモリコントローラが、別の中央処理装置からの、前記中央処理装置リセットベクトルに対応する、他のいかなる読取り要求にも応答して、前記追加の中央処理装置開始ベクトルを返すステップと、をさらに含むことを特徴とする請求項１２に記載の方法である。

請求項１４に係る本発明は、前記メモリコントローラが、中央処理装置がリセットされた後に実行を開始するアドレスに現れるように、前記任意の種類のコードを再マップするステップをさらに含む、ことを特徴とする請求項１１に記載の方法である。

請求項１５に係る本発明は、前記メモリコントローラが、中央処理装置から、中央処理装置リセットベクトルに対応する読取り要求を受信するステップと、前記メモリコントローラが、前記トラステッドコアの開始場所へジャンプさせるための命令により、前記中央処理装置からの前記読取り要求に応答するステップと、をさらに含むことを特徴とする請求項１１に記載の方法である。

請求項１６に係る本発明は、コンピュータに、暴走する可能性のある信用できないコードに基づいて前記コンピュータの動作が正常に開始された場合、前記信用できないコードの制御下で、任意の種類のコードをメモリにロードするステップと、前記トラステッドコアが正常に前記メモリにロードされた場合、前記任意の種類のコードの実行を、請求項５に記載されたトラステッドコア初期化プロセスの実行と同様の手順で開始するステップと、を順に実行させるための１又は複数の実行制御用のコンピュータプログラムを記録したコンピュータ読取可能な記録媒体である。

請求項１７に係る本発明は、プロセッサとの通信を可能にするための第１のインターフェースと、システムメモリとの通信を可能にするための第２のインターフェースと、前記第１のインターフェースおよび前記第２のインターフェースに結合されたコントローラであって、前記プロセッサをリセットし、他のいかなるプロセッサによる前記システムメモリへのアクセスもが防止されている間に前記プロセッサにコード初期化プロセスを実行させ、プロセッサリセットベクトルを初期化ベクトルにマップし、前記プロセッサリセットベクトルに対応する読取り要求を前記プロセッサから受信し、前記読取り要求に応答して、前記プロセッサに前記初期化ベクトルを返し、前記プロセッサが前記初期化ベクトルで開始して前記メモリにアクセスできるようにし、前記プロセッサに前記初期化ベクトルを返した後に、前記プロセッサリセットベクトルを追加のプロセッサ開始ベクトルに再マップし、別のプロセッサからの、前記プロセッサリセットベクトルに対応する他のいかなる読取り要求にも応答して、前記追加のプロセッサ開始ベクトルを返すコントローラとを含むことを特徴とするメモリコントローラである。

請求項１８に係る本発明は、前記コントローラはさらに、前記プロセッサが前記コード初期化プロセスを実行できるようにし、いかなるバスマスタもが前記システムメモリにアクセスするのを防止するためのものであることを特徴とする請求項１７に記載のメモリコントローラである。

請求項１９に係る本発明は、前記コントローラはさらに、前記プロセッサが前記コード初期化プロセスを実行できるようにする前に、前記メモリコントローラに結合された、他のいかなるプロセッサをもリセットし、前記１つの処理が前記コード初期化プロセスを実行するまで、前記メモリコントローラに結合された、他のいかなるプロセッサおよびいかなるバスマスタもが、前記システムメモリにアクセスするのを防止し、前記コード初期化プロセスの実行後、前記メモリコントローラに結合された、他のいかなる中央処理装置、

および、前記メモリコントローラに結合されたいかなるバスマスタも前記メモリにアクセスできるようにするためのものであることを特徴とする請求項１７に記載のメモリコントローラである。

請求項２０に係る本発明は、プロセッサへのリセット信号をアサートするプロセッサリセット部と、前記プロセッサが、トラステッドコア初期化プロセスの実行を完了するまではいかなるバスマスタもメモリにアクセスするのを防止し、かつ、前記プロセッサが前記トラステッドコア初期化プロセスの実行を完了した後にはいかなる前記バスマスタも前記メモリにアクセスすることを許すメモリ保護部と、プロセッサリセットベクトルを初期化ベクトルにマップし、前記プロセッサリセットベクトルに対応する読取り要求を前記プロセッサから受信し、前記読取り要求に応答して、前記プロセッサに前記初期化ベクトルを返し、前記プロセッサが前記初期化ベクトルで開始して前記メモリにアクセスできるようにし、前記プロセッサに前記初期化ベクトルを返した後に、前記プロセッサリセットベクトルを追加のプロセッサ開始ベクトルに再マップし、別のプロセッサからの、前記プロセッサリセットベクトルに対応する他のいかなる読取り要求にも応答して、前記追加のプロセッサ開始ベクトルを返すコントローラとを含むことを特徴とする装置である。

請求項２１に係る本発明は、前記コントローラはさらに、前記プロセッサが前記トラステッドコア初期化プロセスの実行を完了するまで、別のプロセッサからのメモリアクセス要求も前記メモリに出力しないことにより前記別のプロセッサがメモリにアクセスするのを防止することを特徴とする請求項２０に記載の装置である。

請求項２２に係る本発明は、プロセッサへのリセット信号をアサートするプロセッサリセット部と、前記プロセッサが、トラステッドコア初期化プロセスの実行を完了するまではいかなるバスマスタもメモリにアクセスするのを防止するメモリ保護部と、前記メモリ保護部に結合されたコントローラであって、プロセッサリセットベクトルを初期化ベクトルにマップし、前記プロセッサリセットベクトルに対応した読取り要求を、前記プロセッサから受信し、前記読取り要求に応答して、前記初期化ベクトルを前記プロセッサに返し、前記プロセッサが、前記初期化ベクトルで開始して、前記メモリにアクセスできるようにし、前記プロセッサに前記初期化ベクトルを返した後に、前記プロセッサリセットベクトルを追加のプロセッサ開始ベクトルに再マップし、別のプロセッサからの、前記プロセッサリセットベクトルに対応する別の読取り要求に応答して、前記追加のプロセッサ開始ベクトルを返すコントローラと、を含む、ことを特徴とする装置である。

請求項２３に係る本発明は、前記コントローラはさらに、前記初期化ベクトルを前記プロセッサに返した後に、前記プロセッサリセットベクトルを、追加のプロセッサ開始ベクトルに再マップし、別のプロセッサからの、前記プロセッサリセットベクトルに対応した、別の読取り要求に応答して、前記追加のプロセッサ開始ベクトルを返す、ことを特徴とする請求項２２に記載の装置である。

請求項２４に係る本発明は、前記トラステッドコアの一部が格納される記憶部をさらに含む、ことを特徴とする請求項２０に記載の装置である。

請求項２５に係る本発明は、プロセッサと、バスマスタと、システムメモリと、前記プロセッサ、前記バスマスタおよび前記システムメモリに結合されたメモリコントローラとを含み、前記メモリコントローラは、暴走する可能性のある信用できないコードに基づいて正常に動作した場合、前記プロセッサおよび前記バスマスタからの前記システムメモリへのアクセスを正常に中継し、前記プロセッサを、トラステッドコア初期化プロセスを開始するためにリセットし、前記バスマスタが前記システムメモリにアクセスすることを、前記トラステッドコア初期化プロセスが完了される後まで防止するように構成される、ことを特徴とするコンピュータである。

請求項２６に係る本発明は、複数の追加のプロセッサをさらに含み、前記複数の追加のプロセッサが前記システムメモリにアクセスすることを、前記トラステッドコア初期化プロセスが完了される後まで防止する、ことを特徴とする請求項２５に記載のコンピュータである。

請求項２７に係る本発明は、コンピュータにおける異なるトラステッドコアの実行を、

前記コンピュータをリブートする必要なく、逐次開始するステップを含む、ことを特徴とする、前記コンピュータが実行する方法である。

請求項２８に係る本発明は、前記開始するステップは、前記異なるトラステッドコアの実行を、任意の回数で開始するステップをさらに含む、ことを特徴とする請求項２７に記載の前記コンピュータが実行する方法である。

請求項２９に係る本発明は、前記メモリコントローラは、さらに加えて、前記プロセッサの状態をクリアすることにより、前記プロセッサをリセットするように構成されている、ことを特徴とする請求項２５に記載のコンピュータである。

請求項３０に係る本発明は、前記メモリコントローラは、さらに加えて、前記プロセッサへのリセット信号をプロセッサバス上にてアサートすることにより、前記プロセッサをリセットするように構成されている、ことを特徴とする請求項２５に記載のコンピュータである。

請求項３１に係る本発明は、前記メモリコントローラは、さらに加えて、前記プロセッサへのＲＥＳＥＴ＃信号をアサートすることにより、前記プロセッサをリセットするように構成されている、ことを特徴とする請求項２５に記載のコンピュータである。

請求項３２に係る本発明は、中央処理装置が、コンピュータの動作を、暴走する可能性のある信用されていないコードに基づいて開始させるステップと、前記コンピュータの動作が正常に開始された場合、メモリコントローラが、前記信用されていないコードの制御下で、前記コンピュータのトラステッドコアをメモリにロードするステップと、前記トラステッドコアが正常に前記メモリにロードされた場合、前記メモリコントローラが、前記コンピュータにおける１つまたは複数の中央処理装置の各々および１つまたは複数のバスマスタの各々が前記メモリにアクセスするのを防止するステップと、前記メモリコントローラが、前記１つまたは複数の中央処理装置の各々における状態をクリアするステップと、１つの中央処理装置が、前記トラステッドコアを初期化するために、前記メモリにアクセスし前記トラステッドコアに含まれるトラステッドコア初期化コードを実行するステップと、前記メモリコントローラが、前記トラステッドコアが初期化された後、前記コンピュータにおける他のいかなる中央処理装置およびいかなるバスマスタも前記メモリにアクセスできるようにするステップと、を具備したことを特徴とする方法である。

【００７８】

【発明の実施の形態】

以下、図面を参照して、本発明の実施の形態を詳細に説明する。なお、図面全体で同じ番号を使用して、同様の構成要素および／または特徴を表す。

【００７９】

コンピュータにおけるコード（たとえば、ソフトウェア命令）の安全な初期化を可能にするための方法およびシステムが、本明細書に記載されている。この安全なコード初期化プロセス(secure code initialization process)は、主として、コンピュータにおいてトラステッドコア(trusted core)を初期化することに関連して記載してある。しかし、安全なコード初期化プロセスを使用して、幅広い種類のタイプのコードのいずれをも、コードがどの程度信用できるかに関わらず、安全に初期化することができる。本明細書で記載された機構(mechanism)は、特定の関心のあるものであり、これは、コアＣＰＵ(core-CPU)に対する変更が必要とされず、ロジックをサポートするＣＰＵに対する変更がほとんど必要とされないからである。

【００８０】

本明細書で使用されるように、「トラステッド（信用されている:trusted）」コードは、特性(nature)において不変であり、識別(identity)において不変であるコードを指す。トラステッドコードは、コンピュータの他の部分（たとえば、コード）により不正に変更されることを免れており、信頼性を有して、かつ曖昧でなく識別することができる。すなわち、「このコードは誰か」と尋ねる他のいかなるエンティティ(entity)または構成要素(component)にも、「これはコードｘｙｚである」と伝えることができ、このコードが（ある詐称者ではなく）真にコードｘｙｚであること、および、コードｘｙｚが本物であるこ

とを保証することができる。信用(trust)は、コードのいかなる質または有用性の面をも扱わず、特性の不変性および識別の不変性のみを扱う。本明細書に記載された方法およびシステムを使用して、トラステッドコード(trusted code)が、それが順序正しい状態で実行を開始したことを知ることができるようにし、初期化の不変性(immutability of initialization)を提供することができる。

【００８１】

図１は、本発明のある実施形態による、例示的コンピュータ１００を例示するブロック図である。コンピュータ１００は、幅広い種類のコンピューティングデバイスを表すことを意図するものであり、これらは、パーソナルコンピュータ（ＰＣ）、ハンドヘルドまたはポケットデバイス、携帯情報端末（ＰＤＡ）、ゲーム用コンソール、インターネット器具、マルチプロセッサまたはシングルプロセッサシステム、マイクロプロセッサベースまたはプログラマブル消費者電化製品、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、埋め込み型システムなどである。

【００８２】

コンピュータ１００は、１つまたは複数の中央処理装置（ＣＰＵ）またはプロセッサ１０２，１０４、メモリコントローラ１０６、Ｉ／Ｏコントローラ１０８、およびシステムメモリ１１０を含む。ＣＰＵ１０２および１０４は、幅広い種類のプロセッサのいずれかを表し、これらは、ｘ８６アーキテクチャ（または互換性のある）プロセッサ、ＰｏｗｅｒＰＣアーキテクチャ（または互換性のある）プロセッサなどである。多数（ｎ）のＣＰＵ１０２、１０４が図１において例示されるが、コンピュータ１００は、他の実施形態として、単一のＣＰＵのみと共に実施することができる。ＣＰＵ１０２，１０４は共に、メモリコントローラ１０６へ、プロセッサバス１１２を介して結合される。

【００８３】

メモリコントローラ１０６は、分離した構成要素として例示されるが、他の実施形態として、割り込みコントローラ、ＡＧＰ(accelerated graphics port)などを含むブリッジなど、別の構成要素またはデバイスに組み込むことができる。メモリコントローラ１０６は、ＣＰＵ１０２，１０４、Ｉ／Ｏコントローラ１０８、およびメモリ１１０と通信する。メモリ１１０は、ランダムアクセスメモリ（ＲＡＭ）など、システムメモリを表す。Ｉ／Ｏコントローラ１０８は、Ｉ／Ｏバス１１４、および、コンピュータ１００内の他の構成要素に結合された、様々なＩ／Ｏデバイスの間のブリッジまたはインターフェースとして動作する。幅広い種類の従来のＩ／Ｏデバイスのいずれも、Ｉ／Ｏバス１１４に結合することができる。例示された例では、大容量記憶デバイスコントローラ１１６がバス１１４および大容量記憶デバイス（たとえば、ハードディスク）１１８に結合され、ＲＯＭ１２０（ＢＩＯＳ１２２、および、オプションのＲＯＭおよび／またはＢＩＯＳ拡張機能１２４を含む）がバス１１４に結合され、アダプタコントローラ１２６がバス１１４およびネットワークアダプタ１２８（たとえば、ネットワークインターフェースカード（ＮＩＣ）またはモデム）に結合される。

【００８４】

メモリコントローラ１０６は、プロセッサバスインターフェース１３０、メモリインターフェース１３２、Ｉ／Ｏインターフェース１３４、制御ロジック（プロセッサまたはコントローラとも呼ばれる）１３６、ダイジェスト１３８、２つのリセットベクトル１４０，１４２、および初期化パラメータ(initialization parameter)１５３を含む。プロセッサバスインターフェース１３０は、メモリコントローラ１０６とプロセッサバス１１２の間のインターフェースとして動作し、メモリコントローラ１０６がコマンドおよび要求をＣＰＵ１０２，１０４から受信し、さらに、応答およびコマンドをＣＰＵ１０２，１０４に送ることができるようにする。メモリインターフェース１３２は、システムメモリ１１０へのインターフェースとして動作し、コントローラ１０６が、メモリ１１０に格納されたデータを読み取り、データをメモリ１１０に書き込むことができるようにする。Ｉ／Ｏインターフェース１３４は、Ｉ／Ｏコントローラ１０８へのインターフェースとして動作し、Ｉ／Ｏコントローラ１０８を介して、Ｉ／Ｏ構成要素１１８、１２０および１２８がメ

モリコントローラ１０６と通信し、メモリコントローラ１０６がＩ／Ｏ構成要素１１８、１２０および１２８と通信することができるようにする。例示された例では、コントローラ１０６の内部の構成要素（たとえば、制御ロジック１３６、リセットベクトル１４０、１４２など）へ、かつ／または、それからの通信が、適切なインターフェースを介して渡される。

【００８５】

Ｉ／Ｏバス１１４に結合されたＩ／Ｏ構成要素のうち選択されたものが、バスマスタとして動作することができる。バスマスタは、アドレスバス信号および他のバス制御信号をバス上で駆動することができるデバイス（典型的には、ＣＰＵ以外）を指す。バスマスタの例は、大容量記憶デバイスコントローラ１１６およびネットワークアダプタコントローラ１２６である。バスマスタは、メモリ１１０と、Ｉ／Ｏコントローラ１０８および１０６を介して通信して、メモリ１１０へ、かつ／または、それからの直接メモリアクセス（ＤＭＡ）転送を実行することなどができ、それにより、ＣＰＵ１０２，１０４がこのような転送を直接管理しなければならないことを軽減する。

【００８６】

図１の例示された例では、Ｉ／Ｏコントローラ１０８がメモリコントローラ１０６と直接通信する。他の実施形態として、これらの通信が、メモリコントローラ１０６を通過せず、プロセッサを介してＩ／Ｏブリッジ（図示せず）を通ることができる。加えて、Ｉ／Ｏブリッジへのプロセッサも、Ｉ／Ｏコントローラ１０８とＣＰＵ１０２，１０４の間の、プロセッサバス１１２上の通信を可能にする。一実施形態では、メモリコントローラ１０６が「チップセット」の一部として含まれ、これもＩ／Ｏブリッジへのプロセッサなどを含む。チップセットは、コンピュータのサブシステムの間のインターフェースを提供する１つまたは複数のチップのセットを指し、ＣＰＵ，メモリ，Ｉ／Ｏデバイスなどが対話することができるようにするためのバスおよび電子機器を含む。

【００８７】

制御ロジック１３６は、トラステッドコアをコンピュータ１００において安全にロードできることを保証(ensure)するように動作し、メモリ１１０へのアクセスを特定の環境下でのみ可能とし、これについては以下でより詳細に論じる。制御ロジック１３６は、マイクロコード（マイクロコントローラのみがアクセス可能である不揮発性の安全な場所（たとえば、コントローラ１０６内）に格納されたもの）を実行するマイクロコントローラ，プログラマブルロジックデバイス，または他の特定用途向け集積回路（ＡＳＩＣ）など、様々な方法のいずれにおいても実施することができる。

【００８８】

コンピュータ１００の起動（たとえば、電源を入れるか、あるいはリセットすること）において、メモリ１１０の状態は一般的には保証されない。メモリ１１０は一般的にはダイナミックＲＡＭ（ＤＲＡＭ）であり、これは、その状態を維持するために定期的なリフレッシュを必要とし、そのため、コンピュータ１００の電源が入っていなかったときにいかなる状態も失われているようになる。メモリ１１０は、電源を入れたときに特定の状態（たとえば、すべてのゼロまたはすべての１を格納する）となるように構成することができ、あるいは、他の実施形態として、メモリ１１０がランダム値を格納することができる（たとえば、電源が入っている間に、個々のメモリセルがどのような値に安定されるかに関わらず）。それにも関わらず、メモリ１１０は、信頼性のあるデータまたは命令を、電源を入れるときに格納しない。コンピュータシステム全体または単にメモリの起動時に、メモリの内容が幾分ランダムであることに留意されたい。しかし、ＣＰＵの電源を入れること、および／または、リセットすることは、通常、メモリの状態を変えない（そのため、メモリの内容は、ＣＰＵを単にリセットした後に必ずしもランダムではない）。

【００８９】

コンピュータ１００の電源が入れられた後、ＣＰＵ１０２，１０４のうちの１つが、最終的に命令の実行を開始する。ＣＰＵ１０２，１０４のうちの１つのみが最初に命令の実行を開始することができるか、あるいは他の実施形態として、ＣＰＵ１０２，１０４のうち

の１つを、コンピュータのブートを開始するために選択することができる。幅広い種類のアービトレーション機構(arbitration mechanism)のいずれかを使用して、多数のＣＰＵのどれがコンピュータのブートを開始するためのものであるかを決定することができ、これらは、事前に構成された(pre-configured)デフォルトプロセッサ，分散アービトレーション方式(distributed arbitration scheme)，プロセッサバス１１２に結合されたアービトレーションデバイスまたは構成要素(component)（図示せず）などである。

【００９０】

ＣＰＵの数、または、どのＣＰＵが命令の実行を開始するかに関わらず、少なくとも１つのＣＰＵが命令の実行を開始する。コンピュータ１００の電源がちょうど入れられたとき、命令はＣＰＵのキャッシュメモリに（あるいは、コンピュータ１００において存在することができる、他のいかなるキャッシュメモリにも）格納されていない。ＣＰＵは命令をＢＩＯＳ１２２から得て、これらの命令の実行を開始する。一実施形態では、ＣＰＵが、最初に特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０$_{16}$）のための読取り要求を、プロセッサバス１１２上に配置するように構成される。メモリコントローラ１０６は、ＣＰＵが命令の実行を開始するべきである、ＢＩＯＳブートブロックのアドレス（ＢＩＯＳ１２２におけるアドレス）により応答する。メモリコントローラ１０６によって返されたこのアドレスは、リセットベクトル(reset vector)と呼ばれ、この（一般的な）例では、外部ロジックによって提供される。代替実施では、ＣＰＵが単に命令を特定の場所（たとえば、０００００００００$_{16}$）から直接取り出すことを開始し、そのため、この場合、リセットベクトルが固定される。

【００９１】

次いで、ＣＰＵが、このリセットベクトルで開始して、命令のロードおよび実行を開始する。ロードされた命令はＲＯＭ１２０からのものであり、ＢＩＯＳ１２２、ならびに１つまたは複数のオプションのＲＯＭおよび／またはＢＩＯＳ拡張機能１２４からの命令を含むことができる。ＲＯＭ１２０からのいくつかの命令は、命令がメモリ１１０に（ＲＯＭ１２０、または、大容量記憶デバイス１１８など、別のデバイスから）ロードされる結果となり、これらが、次いで、ＣＰＵによってロードされ、実行される。

【００９２】

したがって、オペレーティングシステムの様々な構成要素１４４がメモリ１１０へ、ＲＯＭ１２０および／または他のデバイス（たとえば、記憶デバイス１１８）からロードされる。構成要素(component)１４４は、ＢＩＯＳ１２２、ならびにオプションのＲＯＭおよびＢＩＯＳ拡張機能(extension)１２４の一部を含むことができる。ＯＳ構成要素１４４は、ワシントン州レドモンドのＭｉｃｒｏｓｏｆｔ　Ｃｏｒｐ．から入手可能なオペレーティングシステムのＷｉｎｄｏｗｓ（登録商標）ファミリのいずれかなど、汎用オペレーティングシステムの選択された構成要素も含むことができる。

【００９３】

最終的に、トラステッドコア(trusted core)がメモリ１１０にロードされる。これは、たとえば、ＢＩＯＳ１２２、オプションのＲＯＭまたはＢＩＯＳ拡張機能１２４、ＯＳ構成要素１４４または別のアプリケーション（図示せず）における命令に応答して行うことができる。トラステッドコアは、信用できるコード（たとえば、ソフトウェア命令）を指し、これは、コンピュータ１００の動作の少なくとも一部を制御し、それ自体をコンピュータ１００の特定のハードウェア上で（少なくともある範囲まで）保護する方法を知っているものである。それ自体を保護することにおいて、トラステッドコアは、トラステッドコアの制御下で動作中の他のアプリケーションも保護することができる。トラステッドコアによって提供された保護の量、ならびに、このような保護がトラステッドコアによって提供される方法は、コンピュータ，コンピュータ（またはＣＰＵ）のアーキテクチャ、およびトラステッドコアの間で変わる可能性がある。トラステッドコアは、一般的なオペレーティングシステムの機能性のすべて、または、他の実施形態として、選択された機能性のみを含むことができる。トラステッドコアは一般的には、ある方法において暗号により証明され、トラステッドコアの保全性を後に検証できるようにする。たとえば、トラステッ

ドコアの暗号化措置(cryptographic measure)を、トラステッドコアから抽出(extract)し、トラステッドコアと共に大容量記憶デバイス１１８上（あるいは、あるリモートデバイス上）に格納することができる。トラステッドコアが後にメモリ１１０にロードされたとき、コンピュータ１００の構成要素が、メモリ１１０におけるトラステッドコアの暗号化措置を、格納された暗号化措置と比較するために抽出することができ、２つの措置(measure)が合致した場合、トラステッドコアがデバイス１１８上あるいはメモリ１１０内にある間に（たとえば、不当なユーザまたはプログラムによって）改変されていないことを安全に保証することができる。

【００９４】

例示された例では、抽出された暗号化措置(cryptographic measure)がダイジェスト１３８に格納される。ダイジェスト１３８は、トラステッドコアがロードされたときにその暗号化措置を格納するために使用される１つまたは複数のレジスタ（または、他の記憶機構）を指す。１つのそのような手段が「暗号化ダイジェスト(cryptographic digest)」であり、これは、ＳＨＡ－１(Secure Hash Algorithm 1)など、一方向ハッシュ演算を使用して計算される。暗号化ダイジェストは、ダイジェストされたときに同じハッシュ値(hash value)を生成する第２のプレイメージ(pre-image)（この場合、第２のトラステッドコア）を見つけることが極度に困難であるという特性を有する。よって、ダイジェストレジスタ(digest-register)は、使用中のトラステッドコアを一意に表すものと見なすことができる値を含む。暗号手法の基本的な紹介については、Ｂｒｕｃｅ Ｓｃｈｎｅｉｅｒ著のテキスト「Ａｐｐｌｉｅｄ Ｃｒｙｐｔｏｇｒａｐｈｙ： Ｐｒｏｔｏｃｏｌｓ，Ａｌｇｏｒｉｔｈｍｓ，ａｎｄ Ｓｏｕｒｃｅ Ｃｏｄｅ ｉｎ Ｃ」，Ｊｏｈｎ Ｗｉｌｅｙ ＆ Ｓｏｎｓ，著作権１９９４年（または、著作年１９９６年の第２版）を参照されたい。

【００９５】

他の実施形態として、ダイジェスト以外の手段を使用することができる。たとえば、ロジックが、トラステッドコアにデジタル証明書が添付されることを必要とする可能性がある。チップセット(chipset)は、証明書がトラステッドコアイメージに合致することを保証することができ、次いで、証明書、証明書から得られた公開鍵、またはレジスタにおける証明書のダイジェストを保存することができる。

【００９６】

最終的に、コンピュータは、動作中のコンピュータのための「信用の根元(root of trust)」を提供する構成要素をロードかつ初期化するための準備ができる。以前のコンピュータにおいて直面した著しい問題は、ＣＰＵまたは他のデバイスが不正を働くようにプログラムすることによって、事前に実行したすべてのコードが、トラステッドコアの実行をくつがえす可能性があることである。たとえば、悪質なＯＳローダは、バスマスタリングコントローラが、短い遅延の後、新しい構成要素を、信用されていないコードをその上に書き込むことによって、くつがえすべきであるように構成することができる。本明細書に記載された本発明は、事前に実行するすべてのコードを信用のあるチェーン(trust chain)から除去する。

【００９７】

トラステッドコア(trusted core)は、幅広い種類の方法のいずれにおいても、それ自体を信用できるようにするために動作することができる。トラステッドコアが動作する方法は変わる可能性があり、同じように、トラステッドコアによって提供された安全性または信用可能性の量も変わる可能性がある。図２および図３は、トラステッドコアを実施することができる例示的方法を例示する。しかし、図２および図３は、トラステッドコアを実施することができる方法の例のみを例示し、トラステッドコアを実際には異なる方法において実施することができる。一般に、トラステッドコアは、リング(ring)，カーテニング(curtaining)など、コンピュータ構成要素によって提供されるどのような機構も使用して、実行できるようになった後にそれ自体を保護することができる。

【００９８】

本明細書に記載された本発明は、コードの特性を、「信用の根元」を含むものであると限定しないことに留意されたい。このコードが、それ自体を、それが動作する可能性のあるコード、またはそれがプログラムする可能性のあるデバイスから、保護するための手段（ＣＰＵメモリコントローラのプログラミングなど）を取ることが予想される。しかし、「信用の根元」を、完全なＯＳ、マイクロカーネル、ハイパーバイザ、または、特定の安全性のサービスを提供する、ある小型の構成要素にすることができる。以下で、このような構成要素を「トラステッドコア」と称する。

【００９９】

図２では、トラステッドコアが、図１のＣＰＵ１０２，１０４の異なる特権レベル(privilege level)（たとえば、ｘ８６アーキテクチャプロセッサにおけるリング）を利用することによって実施される。例示された例では、これらの特権レベルがリングと呼ばれるが、異なるプロセッサアーキテクチャを使用する代替実施は、異なる名称を使用することができる。多数のリングが、ソフトウェアが実行することができる、優先順序が付けられたレベルのセットを提供し、これはしばしば４レベル（リング０，１，２および３）を含む。リング０は通常、もっとも特権のあるリング(most privileged ring)と呼ばれる。リング０において実行するソフトウェア処理は通常、より特権の少ないリングにおいて実行する処理より多い機能（たとえば、命令）にアクセスすることができる。さらに、特定のリングにおいて実行するプロセッサは、より高い優先順位のリングにおけるコードまたはデータを改変することができない。例示された例では、トラステッドコア１６０がリング０において実行し、オペレーティングシステム１６２がリング１において実行し、アプリケーションがリング３において実行する。したがって、トラステッドコア１６０が、より特権の多いレベルで動作し、オペレーティングシステム１６２の実行を、このレベルから制御することができる。加えて、トラステッドコア１６０（リング０において実行）のコードおよび／またはデータを、オペレーティングシステム１６２（リング１において実行）またはアプリケーション１６４（リング３において実行）によって直接改変することができない。むしろ、このようないかなる改変も、オペレーティングシステム１６２またはアプリケーション１６４が、トラステッドコア１６０に改変を行うように要求すること（たとえば、メッセージを、トラステッドコア１６０に送信すること、トラステッドコア１６０の機能を呼び出すこと、などによる）によって行われなければならない。

【０１００】

他の実施形態として、オペレーティングシステムを、トラステッドコア１６０として動作するメモリ管理構成要素と、ＯＳ１６２として動作するオペレーティングシステムの残りに分離することができる。次いで、トラステッドコア１６０が、すべてのページマップ(page map)を制御し、したがって、リング３において実行するトラステッドエージェント(trusted agent)を他の構成要素（ＯＳ１６２を含む）から遮蔽(shield)することができる。この代替方法では、追加の制御を（たとえば、図１のメモリコントローラ１０６において）追加して、トラステッドコアを、リング特権に従わない他のバスマスタから保護する必要もある。

【０１０１】

図３では、トラステッドコアが、コンピュータ１００内で２つの分離した「空間(space)」を確立することによって実施される。すなわち、トラステッド空間(trusted space)１６６（保護された並列領域(parallel area)、またはカーテニングされたメモリ(curtained memory)とも呼ばれる）、およびノーマル（信用されていない:untrusted）空間１６８である。これらの空間を、たとえば、コンピュータ１００内の１つまたは複数のアドレス範囲にすることができる。トラステッド空間１６６もノーマル空間１６８も、ユーザ空間およびカーネル空間を含み、トラステッドコア１７０は、トラステッド空間１６６のカーネル空間において実施される。様々なトラステッドアプレット(trusted applet)，アプリケーション、および／またはエージェントが、トラステッド空間１６６のユーザ空間内で、トラステッドコア１７０の制御下で実行することができる。しかし、ノーマル空間１６８において実行するいかなるアプリケーション１７４，オペレーティングシステム１７６

、またはデバイスドライバ１７８も、トラステッドコア１７０によって、トラステッド空間１６６にアクセスするのを防止される。したがって、トラステッド空間１６６におけるアプリケーションまたはデータに改変を行うことは、トラステッドコア１７０によって承認されない限りできない。

【０１０２】

図１に戻ると、トラステッドコア１４６が３つの部分において例示されている。すなわち、プラットフォームトラステッドコア部(platform trusted core portion)１４８、コモントラステッドコア部(common trusted core portion)１５０、および、トラステッドコアデータ部(trusted core data portion)１５２である。これらの部分１４８，１５０および１５２は、メモリ１１０において隣接して配置されるように例示されるが、これらの部分をそのように配置する必要はない。たとえば、異なる部分を、異なる、隣接していない領域に格納することができ、各部の異なるセクションを、異なる、隣接していない領域に格納することができる。他の実施形態として、トラステッドコア１４６が実行時に動作する方法に応じて、これらの部分を、メモリ１１０内で物理的に隣接した場所に配置する必要のある可能性がある（つまり、実際には、ＲＡＭの物理的に隣接した場所に格納され、仮想メモリによる別のデバイスにページアウトされない）。

【０１０３】

トラステッドコアデータ部１５２は、メモリ１１０の、トラステッドコアによって使用されたデータを格納するために使用される領域である。プラットフォームトラステッドコア部１４８は、コンピュータ１００において使用された特定のハードウェアに固有であるコードを含む（たとえば、特定のタイプのＣＰＵ、そのタイプのチップセットなど）。コモントラステッドコア部は、トラステッドコアの動作および機能性を実施するが、コンピュータ１００において使用された特定のハードウェアに固有ではないコードを含む。２つの部分１４８および１５０を分離することによって、コンピュータ１００のハードウェアとの対話方法の細部（たとえば、タイマを設定する方法）を、部分１４８によって部分１５０に抽象化することができ、そのため、同じコモントラステッドコア部を異なるコンピュータ上で使用することができる。たとえば、コモントラステッドコアコード１５０は、すべてのタイマが同じ方法で設定されたかのように動作することができ、プラットフォームトラステッドコアコード１４８は、コード１５０からの命令を、その下にあるコンピュータ１００のハードウェアによって実際に理解されるコマンドに変換するように動作する。

【０１０４】

トラステッドコア１４６をメモリ１１０に、同じソース、または他の実施形態として、異なるソースからロードすることができる。例として、プラットフォームトラステッドコア部１４８をメモリ１１０に、コンピュータ１００のチップセット（チップセットは、バスおよび電子機器を含む、コンピュータのサブシステムの間のインターフェースを提供して、ＣＰＵ，メモリ，Ｉ／Ｏデバイスなどが対話することができるようにする、１つまたは複数のチップのセットを指し、これは、メモリコントローラ１０６，Ｉ／Ｏコントローラ１０８、および、バス１１２および１１４を含むことができる）からロードすることができ、コモントラステッドコア部１５０をメモリ１１０に、大容量記憶デバイス１１８からロードすることができる。もう１つの例として、１つの部分１４８，１５０または１５２をメモリ１１０に、ネットワークアダプタ１２８から（たとえば、リモートサーバから）ロードすることができ、あるいは、これらの部分をすべてネットワークアダプタ１２８からロードすることができるが異なるソース（たとえば、異なるリモートサーバ）からはロードできない。加えて、１つまたは複数の部分１４８，１５０および１５２の異なる部分を、異なるソースからロードすることができる（たとえば、部分１５０の一部をメモリ１１０に、大容量記憶デバイス１１８からロードすることができ、部分１５０の別の一部をリモートサーバから、ネットワークアダプタ１２８を介してロードすることができる）。

【０１０５】

加えて、各部を、１つまたは複数のソースからの様々なオーバーレイされた部分を結合することによって、生成することができる。たとえば、第１部をあるソース（たとえば、ロ

ーカルのディスク）から読み取ることができ、第２部を別のソース（たとえば、ネットワークを介したリモートサーバ）から読み取ることができ、次いで、２つの部分を結合して（たとえば、各部分のビットの排他的論理和を取る）この部分を生成することができる。

【０１０６】

トラステッドコア１４６を安全な場所に格納すること、および／または、安全な方法においてメモリ１１０にロードすることが必要ないことに留意されたい。トラステッドコア１４６を、公にアクセス可能な（または、半ば公にアクセス可能な）いかなる領域にも格納することができ、これは、大容量記憶デバイス１１８、ネットワークアダプタ１２８を介してアクセスされるリモートサーバなどである。むしろ、トラステッドコア１４６の暗号化措置(cryptographic measure)を、メモリ１１０にロードした後に（かつ、これが、信用されていないコードによるさらなる修正が可能でないように保護された後に）検証することができる。暗号化措置は、幅広い種類の従来の方法のいずれにおいても生成することができ、トラステッドコアにおけるいかなる変更も暗号化措置において反映されるように設計される。たとえば、不当なユーザが、トラステッドコアの安全性をくつがえすための試みにおいて、命令を削除あるいは修正すること、または、新しい命令を追加することによって、トラステッドコアを改変しようと試みた場合、この改変が暗号化措置において反映される（そのため、後に生成されたいかなる暗号化措置も、先に生成された暗号化措置に合致しないようになり、これは命令における変更による）。一実施態様では、暗号化措置は、トラステッドコアのダイジェスト値を、よく知られているＳＨＡ，ＳＨＡ－１，ＭＤ５（メッセージダイジェスト５）などのアルゴリズムに従うなどして生成することによって得られる。

【０１０７】

この暗号化措置は、本明細書に記載された安全な初期化プロセス中に生成され、後に挑戦者(challenger)（たとえば、銀行，ＩＴ管理者，スマートカードなど）により使用可能となる。ダイジェストの解釈は、いくつかの方法において行うことができる。これは、企業がそのトラステッドコアの予想値をそのｗｅｂサイト上で公開するなど、「よく知られた(well known)」値にすることができる。他の実施形態として、企業が、それが書き込む、各トラステッドコアのための証明書を生成し、これを、トラステッドコアと共に出荷する（あるいは、ｗｅｂサイト上で使用可能にする）ことができる。これにより、挑戦者がダイジェストを見て、ダイジェストを「命名する(name)」証明書を求める（あるいは、探す）ことができる。証明書がダイジェストを命名するので、この証明書をコア自体に入れるべきではないことに留意されたい。

【０１０８】

本明細書に記載された初期化プロセス(initialization process)は、ユーザが「トラステッドコア」として動作するように試みることができるものを限定するものではなく、ユーザが、実質的にいかなる命令のセットも、トラステッドコアとして動作するように試みることができることに留意されたい。しかし、トラステッドコアが不正に変更された場合、「トラステッドコア」として動作しようと試みている命令（トラステッドコアを不正に変更したもの）は、他の関係者から信用されなくなり、これは、トラステッドコアを不正に変更したものが、異なる暗号化措置を有するようになり、トラステッドコアを不正に変更しないものにアクセス可能である鍵にアクセスできなくなるからである。

【０１０９】

本明細書に記載された初期化プロセスにより、当然かつ容易に、コンピュータが、異なるトラステッドコアを逐次、任意の回数で、システムをリブート(re-boot)することなく動作できることにも留意されたい。たとえば、ユーザは、トラステッドコアｘの実行を開始し、次いで、トラステッドコアｙ（これを、トラステッドコアｘの異なるバージョン、または、まったく異なるトラステッドコアにすることができる）の実行に切り替え、次いで、トラステッドコアｚに切り替える（あるいは、トラステッドコアｘに戻る）ことなどができる。

【０１１０】

メモリコントローラ１０６は、トラステッドコア１４６がメモリ１１０にロードされた後、これを、コンピュータ１００のいかなる先の状態もが初期化プロセスの保全性を損なうことのないような方法において、初期化し、実行を開始できることを保証するように動作する。メモリコントローラ１０６が、トラステッドコアを初期化することができる安全な環境を提供するように動作することに留意されたい。しかし、初期化プロセスが完了した後、トラステッドコア１４６が、それ自体の安全性の責任を負う。加えて、メモリコントローラ１０６は、トラステッドコア１４６によって提供された信用可能性または安全性に関するいかなる保証または確信も生成しない。

【０１１１】

トラステッドコア１４６の初期化は、トラステッドコア１４６の実行を開始することを指す。この初期化は一般的には、トラステッドコア１４６が全体としてメモリ１１０にロードされた後に起こるが、代替実施態様では、コード１４６のいくつかの部分を、初期化プロセスが起こった後にメモリにロードできるようにすることができる。初期化プロセスは、トラステッドコア初期化命令のセットによって実行され、これは、トラステッドコア１４６の一部である（また、場合によっては、１つまたは複数の部分１４８、１５０および１５２の一部である）。初期化プロセスによって実行される厳密な動作は、トラステッドコアが動作する特定の方法、ならびに、コンピュータ１００の特定のハードウェア（たとえば、ＣＰＵアーキテクチャ）に基づいて変わる可能性があるが、一般に、トラステッドコアによって提供された保護を確立すること（たとえば、トラステッド空間をセットアップすること）、および、ＣＰＵを初期化してこの保護を使用するようにすることを含む。「初期化コード(initialization code)」が行わなければならないことの１つが「リセットベクトル(reset vector)」を提供することであり、これは適所に残され、常に、リセット，開始(initiate)，ホットプラグ追加(hot-plug add)などが行われるいかなるプロセッサをも処理する準備ができている。これを行うことにより、トラステッドコアが、その継続された保全性を、マシンのＣＰＵ構成における深い変更（たとえば、新しいものがオンザフライで追加される）に対して保証する。

【０１１２】

トラステッドコア１４６の初期化は、ＣＰＵ１０２，１０４のうちの１つによって発行された「トラステッドコア初期化(trusted core initialization)」コマンドに応答して開始する。トラステッドコア初期化コマンドは一般に、コンピュータ１００のブート中に発行されるが、コンピュータ１００が完全にブートされている間あるいはその後のいかなるときにも起こる可能性がある。トラステッドコア初期化コマンドは、トラステッドコア初期化をトリガするように設計されており、幅広い種類の方法のいずれにおいても実施することができる。たとえば、このコマンドを、特定のポートへの書き込み、特定のアドレスへの書き込みなどにすることができるが、ポートまたはアドレスは、トラステッドコアが、コアが制御を獲得した後にハードウェアが提供する保護手段がどんなものであれ、それを使用してポートまたはアドレスを保護することができるという特性を有するべきである。これは、トラステッドコアが動作中になった後、信用されていないいかなるコードもこれをメモリから除去したり置き換えることができる、単純なサービス拒絶攻撃に対して、脆弱ではないことを保証するものである。他の実施形態として、このような保護が提供されなかった（たとえば、トラステッドコア初期化コマンドを、ある保護不可能なポートによって呼び出さなければならない）場合、これを「一回実行(run once)」の動作にすることができ、これは、システムがリブートされるまで、その機能を再度実行することを拒絶するものである。しかし、このようなマシンは、トラステッドコアを停止かつ開始するか、あるいは、ブートにつき複数のトラステッドコアを使用するための能力を失う。

【０１１３】

一実施態様では、トラステッドコア初期化コマンドが３つのパラメータ、すなわち、開始、コード長およびメモリ長を含む。開始パラメータは、メモリ１１０においてトラステッドコア１４６が開始する場所を指す（たとえば、プラットフォームトラステッドコア部１４８の最初の命令のメモリアドレス）。他の実施形態として、特に、トラステッドコア１

４６が物理的に隣接したメモリに位置しない実施形態では、メモリ記述子リスト（ＭＤＬ）を開始パラメータとして識別することができ、メモリコントローラ１０６に対してどのメモリページがトラステッドコアを含むかを識別する。コード長パラメータは、トラステッドコア１４６のコードの長さ（たとえば、バイト数）を指す（たとえば、部分１４８および１５０の結合であるが、トラステッドコアデータ部１５２を除く）。メモリ長パラメータは、トラステッドコアの長さ（たとえば、バイト数）を指す（たとえば、すべての３つの部分１４８，１５０および１５２の結合）。これらのパラメータが、トラステッドコア初期化コマンドの前に（あるいは、その一部として）、メモリコントローラ１０６に渡される。一実施態様では、３つのパラメータが、ＣＰＵ１０２，１０４のうちの１つによって、トラステッドコア初期化コマンドの発行の前に、メモリコントローラ１０６のパラメータレジスタ１５３にロードされる。

【０１１４】

トラステッドコア初期化コマンドを受信すると、メモリコントローラ１０６がメモリ１１０を、すべてのＣＰＵ１０２，１０４およびコンピュータ１００における他のいかなるバスマスタ(bus master)からも保護する。ＣＰＵ１０２，１０４、ならびに、コンピュータ１００における他のいかなるバスマスタも、メモリ１１０に直接アクセスできず、むしろメモリコントローラ１０６を介してメモリ１１０にアクセスしなければならないことを考えると、メモリコントローラ１０６が容易にメモリ１１０をＣＰＵおよびバスマスタから保護することができる。メモリ１１０を保護することによって、メモリコントローラ１０６は、他のコード（たとえば、ＣＰＵによって実行中であるか、あるいは別のバスマスタを制御しているもの）が、トラステッドコア初期化プロセスを干渉しないことを保証する。メモリ１１０の保護において、メモリコントローラ１０６は最初に、メモリ１１０へのアクセスを、メモリリフレッシュコントローラ（図示せず）からのメモリリフレッシュ信号のみにすることができ、これは、ＤＲＡＭをリフレッシュさせ、その状態を維持することができる。

【０１１５】

要素化されたメモリ(factored memory)、つまり、多数のメモリコントローラを有する多数のメモリのブロックを有するマシンでは、多数の異なる手法を使用することができる。１つの手法では、多数のメモリコントローラが互いに調整して、トラステッドコア初期化オペレータをすべてのメモリに渡って適用する。もう１つの手法では、トラステッドコア初期化を行うために選択されたメモリコントローラ／メモリが、そのすべてを行い（そのため、トラステッドコア全体が１つの「バンク」にロードされる）、他のメモリコントローラがこれを無視する。すべてのトラステッドコアが単一の保護メモリバンクにあるので、他の動作は重要ではない。選択されたメモリコントローラがなお、すべてのＣＰＵの挙動を強制し、そのため、プラットフォームのハードウェアが、これを行うことができるようにセットアップされる。

【０１１６】

メモリコントローラ１０６における制御ロジック１３６も、トラステッドコア初期化ビット１５４を、メモリコントローラ１０６の安全部分１５６内に設定する。この安全部分１５６は、制御ロジック１３６のみが安全部分１５６内のレジスタおよび／またはメモリにアクセスできることにおいて安全である。安全部分１５６は、メモリコントローラ１０６の外部のコンピュータ１００における他の構成要素に対して可視にすることはできず、あるいは、他の実施形態として、制御ロジック１３６が単に、安全部分１５６内のいかなるレジスタまたはメモリにもアクセスしようと試みる、メモリコントローラ１０６で受信されたいかなるコマンドも、無視（あるいは、拒絶）することができる。トラステッドコア初期化ビット１５４により、制御ロジック１３６は、トラステッドコア初期化プロセスが進行中であるかどうかを知ることができ、ビット１５４の使用については以下でより詳細に論じる。

【０１１７】

メモリコントローラ１０６は、様々な異なる方法においてメモリ１１０を保護することが

できる。一実施態様では、メモリコントローラ１０６は、ＣＰＵ１０２，１０４が読み取りまたは書き込み要求をプロセッサバス１１２上に発行するのを防止し、それにより、ＣＰＵ１０２，１０４がメモリ１１０にアクセスするのを防止する。ＣＰＵ１０２，１０４をバスから離しておく方法は、ＣＰＵ１０２，１０４およびプロセッサバス１１２のアーキテクチャに基づいて変わる（たとえば、あるＩｎｔｅｌアーキテクチャの実施では、ＣＰＵが読み取りおよび書き込み要求をバス上に発行するのを防止することができ、これは、ＢＮＲ#(Block Next Request)信号をプロセッサバス１１２上にアサートすることによって、あるいは、各ＣＰＵがリセットされるまでその動作を停止する停止（たとえば、ＨＬＴ）コマンドをＣＰＵに発行することによって行われる）。ＣＰＵ１０２，１０４は最終的に、再度読み取りおよび書き込み要求をプロセッサバス１１２上に発行することができるようになるので、制御ロジック１３６も、メモリコントローラ１０６がＣＰＵ１０２，１０４からメモリ１１０におけるトラステッドコア１４６のアドレス範囲に受信するすべての要求を比較し、メモリコントローラ１０６がＣＰＵ１０２，１０４からこのトラステッドコア初期化プロセス中に受信する（たとえば、ビット１５４セットによる）、このアドレス範囲内であるいかなる要求も、従来の方法において受け入れられ、応答されるが、このアドレス範囲内でないか、あるいはＣＰＵ１０２，１０４以外の構成要素からのいかなる要求も拒絶される（たとえば、メモリコントローラ１０６が、要求を確認することを拒絶することができる、要求を否定することができる、など）。他の実施形態として、このアドレス範囲内でなく受信された要求を許可することができるが、このようなメモリの領域が保護されていないことを理解することを伴う。

【０１１８】

メモリコントローラ１０６は、Ｉ／Ｏバス１１４上の他のバスマスタがメモリ１１０にアクセスすることも防止する。加えて、メモリコントローラ１０６は、Ｉ／Ｏバス１１４上の他のバスマスタが（たとえば、ＣＰＵ内あるいはバス１１２上の）キャッシュメモリに影響を与えることを、このようないかなるトランザクションをもバス１１２上に発行しないことなどによって、防止することができる。たとえば、メモリコントローラ１０６が、Ｉ／Ｏコントローラ１０８に信号をＩ／Ｏバス１１４上にアサートさせることができ、これにより、いかなるバスマスタもが読み取りおよび書き込み要求をバス１１４上に発行するのを防止する。もう１つの例として、メモリコントローラ１０６が単に、Ｉ／Ｏコントローラ１０８から受信するいかなるメモリアクセス要求も無視することができ、これは、Ｉ／Ｏインターフェース１３４にこのような要求を否定させ、このような要求の受け入れまたは確認を拒絶させることなどによって行う。メモリ１１０へのアクセスを防止することは、ＣＰＵのアクセスおよびＩ／Ｏバスマスタのアクセスのための類似の様式において実施することができるが、初期化プロセス中に、ＣＰＵのアクセスが可能とされても、Ｉ／Ｏバスマスタのアクセスが可能とされないことに留意されたい。

【０１１９】

メモリコントローラ１０６は、トラステッドコア初期化コマンドを受信すると、即時にメモリ１１０を保護することができ、あるいは、他の実施形態として、メモリ１１０を保護する前に、ある期間、あるいは、ある他の動作が発生するまで待機することができる。たとえば、特定のバスマスタが、トラステッドコア初期化コマンドが受信されたときに、長いＤＭＡ転送の処理中である可能性があり、メモリコントローラ１０６が、ＤＭＡ転送を完了させることを選択することができる。メモリ１１０の保護を即時にすることはできないが、トラステッドコア初期化プロセスはメモリ１１０が保護されるまで進行しない。

【０１２０】

メモリ１１０が保護された後、制御ロジック１３６が、トラステッドコア１４６に基づいてダイジェスト値を生成する。他の実施形態として、専用の暗号化プロセッサなど、もう１つの構成要素（図示せず）が、ダイジェスト値を生成するために一時的にメモリ１１０にアクセスできるようにすることができる（あるいは、ダイジェストを暗号化プロセッサにおいて保持することができ、これは、暗号化プロセッサが私的にメモリコントローラと共に作業して、ダイジェスト操作を、ＣＰＵが初期化プロセスを実行中である（かつ、暗

号化プロセッサのこの要求を行う、信用されていないソフトウェアによってだますことができない）ときにのみ実行する限りである）。しかし、このアクセスは、暗号化プロセッサがメモリ１１０からしか読み取ることができないようにするなど、限定することができる。このダイジェストは、上述したような同じ暗号化機構を使用して生成される。次いで、制御ロジック１３６が、生成されたダイジェスト値をダイジェスト１３８として、メモリコントローラ１０６の安全部分１５６に格納する。次いで、ダイジェスト１３８を、トラステッドコア１４６による後続の証明および／または認証のために使用することができる。

【０１２１】

次いで、制御ロジック１３６が、リセットベクトルを新しいリセットベクトルにマップし、これは初期化ベクトルと呼ばれ、メモリコントローラ１０６がプロセッサバス１１２上の特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０_{16}）のための次の読み取り要求に、初期化ベクトルにより応答するようにする。ＣＰＵが常に、（リセットベクトル取り出し手順によって供給された間接化技法を使用するのではなく）分かっている場所から実行を開始する状況では、制御ロジック１３６が命令を、メモリサブシステムから取り出された命令の最初のものまたは少数のものとして発行するように構成し、これにより、ＣＰＵに、トラステッドコア（たとえば、ＪＵＭＰ動作）の開始時に命令の実行を開始させるか、あるいは、他の実施形態として、トラステッドコアを再マップさせて、固定プロセッサリセットベクトルに現れるようにする。この初期化ベクトルは、トラステッドコア１４６の最初のアドレスである。一実施態様では、このマッピングが、制御ロジック１３６によって実施され、トラステッドコア１４６の開始場所（たとえば、トラステッドコア初期化コマンドのパラメータとして受信されたもの）をリセットベクトル１４０として書き込む。メモリコントローラ１０６が、プロセッサバス１１２上の特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０_{16}）のための読み取り要求を受信するときは、制御ロジック１３６が安全部分１５６のビット１５４をチェックする。このビットが設定された場合、マップされたリセットベクトル１４０が、要求側のＣＰＵに、リセットベクトルとして返される。しかし、このビットが設定されていなかった場合、最初のＢＩＯＳブートブロックアドレスが要求側のＣＰＵに返される。

【０１２２】

次いで、制御ロジック１３６が、リセット信号をプロセッサバス１１２上のＣＰＵにアサートする。多数のＣＰＵがプロセッサバス１１２上にある状況では、リセット信号がすべてのＣＰＵにアサートされる（他の実施形態として、リセット信号を単一のＣＰＵに向けて送ることができる）。リセット信号がアサートされる方法（および、リセット信号自体の特性）は、ＣＰＵ１０２，１０４およびプロセッサバス１１２のアーキテクチャに基づいて変わる可能性がある（たとえば、あるＩｎｔｅｌアーキテクチャの実施では、プロセッサバス１１２上のすべてのＣＰＵを、ＲＥＳＥＴ＃信号をプロセッサバス１１２上にアサートすることによってリセットすることができる）。

【０１２３】

他の実施形態として、他の機構を使用して、プロセッサバス１１２上のＣＰＵをリセットすることができる。ＣＰＵをリセットする目的は、ＣＰＵのその状態（たとえば、ＣＰＵのレジスタ、キャッシュ、バッファなどの内に存在する可能性のある、すべての命令およびデータ）をクリアして、その以前の状態がトラステッドコア初期化プロセスの実行保全性を損なう可能性のないようにすることである。たとえば、ＣＰＵのキャッシュに格納されている不当なアプリケーションの命令は、トラステッドコア初期化プロセスの保全性を損なうことができない。

【０１２４】

プロセッサバス１１２上のＣＰＵをリセットすることに加えて、プロセッサバス１１２上でメモリを含む、他のいかなるキャッシュメモリまたは他のデバイスもリセットされる。たとえば、追加のＬ３キャッシュ（図示せず）もプロセッサバス１１２に結合することができ、この場合、制御ロジック１３６がＬ３キャッシュならびにＣＰＵをリセットする。

プロセッサバス１１２上のこれらの追加のキャッシュメモリまたは他のデバイスは、様々な方法のいずれにおいてもリセットすることができ、これらは、ランダム値をメモリに書き込むこと、ゼロまたは１の文字列をメモリに書き込むこと、電力をメモリから除去すること（たとえば、一時的にＤＲＡＭをリフレッシュしないこと）、特定のバストランザクション(specific bus transaction)を発行してキャッシュ全体または特定のキャッシュ線エントリ(specific cache-line entry)を無効にすることなどである。

【０１２５】

ＣＰＵ１０２，１０４（および、プロセッサバス１１２上の他のいかなるキャッシュまたはデバイス（メモリコントローラ１０６を除く））をリセットした後、メモリコントローラ１０６により、プロセッサバス１１２上のＣＰＵの１つがメモリ１１０にアクセスできるようにする。例示された例では、ＣＰＵ１０２，１０４のいずれもが、最初にメモリ１１０にアクセスできるようにすることができる（が、ただ１つのみが可能とされる）。メモリコントローラ１０６は、多数のプロセッサのうちのどれがメモリ１１０にアクセスできるようにするかを、様々な方法のうちのいずれにおいても決定することができ、これらは、ある所定の設定、ランダム決定、ある外部の（あるいは、ＣＰＵ内部の）バスアービトレーション機構が決定できるようにすることなどである。

【０１２６】

考察のため、メモリコントローラ１０６が、ＣＰＵ１０２がメモリ１１０にアクセスできるようにすることを決定すると仮定する。次いで、メモリコントローラ１０６は、ＣＰＵ１０２がプロセッサバス１１２にアクセスできるようにする（他のすべてのＣＰＵ１０４はなお、プロセッサバス１１２にアクセスするのを防止される）。ＣＰＵがリセットされた後、ＣＰＵは、そのいかなるキャッシュまたはバッファにおいても命令を有していない（あるいは、少なくとも、ＣＰＵが読み取る命令またはデータがない）ようになる。したがって、ＣＰＵ１０２は、実行するための命令を、プロセッサバス１１２上で特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０$_{16}$）をアサートすることによって要求する。メモリコントローラ１０６がこの要求を受信し、制御ロジック１３６が、トラステッドコア初期化プロセスビット１５４が設定されていることを決定し、そのため、メモリコントローラ１０６がリセットベクトル１４０をＣＰＵ１０２に返す。特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０$_{16}$）を受信すると、制御ロジックが追加のプロセッサ毎のベクトルビット１５８を設定する。このビット１５８により、制御ロジック１３６が、別の特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０$_{16}$）が受信された場合に、リセットベクトル１４０がすでに別のプロセッサに返されていることを知ることができる。ビット１５８の使用については以下でより詳細に論じる。

【０１２７】

リセットベクトル１４０を受信すると、ＣＰＵ１０２は、プロセッサバス１１２上に、リセットベクトル１４０におけるメモリアドレスのための読み取り要求を発行し、これはトラステッドコア１４６における最初の命令である（また、トラステッドコア初期化プロセスにおける最初の命令である）。したがって、ＣＰＵ１０２が、トラステッドコアコード１４６の実行を開始する。ＣＰＵ１０２がリセットされたので、先にＣＰＵ１０２にロードされた命令（たとえば、不当なコード）が、トラステッドコア１４６のその実行に影響を与えることはできない。加えて、他のすべてのバスマスタがメモリ１１０にアクセスすることが防止されるので、これらが、ＣＰＵ１０２によってメモリ１１０から取り込みかつ実行中のコード１４６を改変することはできない。

【０１２８】

ＣＰＵ１０２は、トラステッドコアコード１４６の取り込みおよび実行を継続し、これにより、コンピュータ１００においてトラステッドコアを初期化する。上述のように、トラステッドコアが初期化される厳密な方法は、コンピュータ１００のアーキテクチャ、ＣＰＵ１０２，１０４、実施中のトラステッドコアのタイプなどに基づいて変わる可能性がある。この初期化プロセスの一部として、トラステッドコア１４６が多数のプロセッサをサポートする場合、追加のＣＰＵ開始ベクトルがリセットベクトル１４２としてロードされ

る。この追加のＣＰＵ開始ベクトルは、コード１４６において、追加のプロセッサが命令の実行を開始した場合に追加の命令が取り込まれるメモリアドレスを識別する。この第２のリセットベクトルを使用して、たとえば、追加のプロセッサに、トラステッドコアにおいて、トラステッドコア１４６の開始ではない他のところ（たとえば、コード１４６のトラステッドコア初期化部分の開始以外）で命令の実行を開始させることができる。しかし、他のＣＰＵは、「初期化の終了(end of initialization)」まで動作することができないので、１４２のベクトルは「初期化の終了」後まで使用されない。

【０１２９】

最終的に、トラステッドコア初期化プロセスが完了し、「初期化の終了」コマンドがＣＰＵ１０２によって発行される。初期化の終了コマンドを、上述のトラステッドコア初期化コマンドに類似した、様々な方法のいずれにおいてもアサートすることができる。初期化の終了コマンドを受信すると、メモリコントローラ１０６により、プロセッサバス１１２上のすべての残りのＣＰＵ１０４がメモリ１１０にアクセスできるようになる。これらの各ＣＰＵ１０４はリセットされており、それらのキャッシュまたはバッファのいずれにおいても命令を有していない（あるいは、少なくとも、有効な命令がない）ようになる。したがって、各ＣＰＵ１０４は、命令を実行するように、プロセッサバス１１２上で特定のアドレス（たとえば、ＦＦＦＦＦＦＦ０$_{16}$）をアサートすることによって要求する。メモリコントローラ１０６がこれらの要求を受信し、制御ロジック１３６が、追加のプロセッサベクトルビット１５８が設定されていることを決定し、そのため、メモリコントローラ１０６がリセットベクトル１４２を各ＣＰＵ１０４に返す。

【０１３０】

次いで、制御ロジック１３６は、トラステッドコア初期化プロセスが完了したときトラステッドコア初期化ビット１５４をクリアする。メモリコントローラ１０６は、他のバスマスタがメモリ１１０にアクセスするのを防止することもやめる。したがって、別のバスマスタからの不当なコードに対するいかなる追加の保護も、メモリコントローラ１０６ではなくトラステッドコア１４６の実行によって防止されるものである。

【０１３１】

各ＣＰＵ１０４は、リセットベクトル１４２を受信すると、プロセッサバス１１２上に、リセットベクトル１４２におけるメモリアドレスのための読み取り要求を発行し、これは、トラステッドコア１４６における命令である。したがって、各ＣＰＵ１０４は、トラステッドコアコード１４６における命令の実行を開始する。各ＣＰＵ１０４がリセットされると、先にＣＰＵ１０４にロードされた命令（たとえば、不当なコード）は、トラステッドコア１４６のその実行に影響を与えることはできない。

【０１３２】

一実施態様では、１つまたは複数のＣＰＵ１０２，１０４が、正しく機能するために、マイクロコードがそれにロードされることを必要とする可能性がある。このマイクロコードは、たとえば、ＣＰＵにおけるエラーまたは他のバグを修正し、ＣＰＵの固有の命令セットとは異なる命令セットを露出することなどができる。このマイクロコードを、トラステッドコア１４６に（たとえば、トラステッドコアデータ部１５２において）含めることができ、命令を、トラステッドコア初期化プロセスに含めて、マイクロコードを適切なＣＰＵにロードすることができる。加えて、マイクロコードが本物であることを検証することができ、これは、ダイジェストを計算し、これをマイクロコード内で証明されたダイジェスト（たとえば、マイクロコードの作成者または配布者によってそこに配置されたもの）に対して比較して、このマイクロコードが、マイクロコードの配布者によって署名された以後に改変されていないことを保証することなどによる。

【０１３３】

一実施態様では、メモリコントローラ１０６は、トラステッドコア初期化コマンドに応答して、以下の保証を提供する。

・メモリに対するすべてのＩ／Ｏアクセスが停止する。リフレッシュコントローラおよびＣＰＵ（リセットされた後）を除き、どの構成要素もメモリにアクセスできない。

・トラステッドコアの暗号化措置は、メモリに対するＩ／Ｏアクセスが停止した後に計算される。
・どのＣＰＵも、リセットされた後まで再起動できず、メモリにアクセスできない。
・すべてのＣＰＵは、トラステッドコアコードにおける２つの場所のうちの一方で命令の実行を開始し、これらの場所が、トラステッドコアコードから設定される。
・ＣＰＵのうちの１つのみが、２つの場所のうちの一方で命令の実行を開始し、他のものはすべて、トラステッドコア初期化プロセスの終了まで待機し、２つの場所のうちの他方で命令の実行を開始する。
・メモリに対するＩ／Ｏアクセスは、トラステッドコア初期化プロセスが完了するまで可能にされない。

【０１３４】
加えて、メモリコントローラ１０６は、いかなる割り込みもインターセプトかつ禁ずるように動作することもできる。たとえば、メモリコントローラ１０６は、コンピュータ１００のための割り込みコントローラを含むか、あるいは他の実施形態としてそれと通信することができる。トラステッドコア初期化プロセス中に受信されたいかなる割り込みも禁ずることができる。他の実施形態として、メモリコントローラ１０６が割り込みに関与することができない（たとえば、ＣＰＵ１０２，１０４がリセットされ、メモリコントローラ１０６およびトラステッドコア１４６によって、命令を得るためのそれらの能力が制限されるため）。

【０１３５】
加えて、ある実施形態では、トラステッドコア初期化プロセスが、トラステッドコアの保護あるいはカーテニングされた空間内である、メモリのある領域（実および／または仮想領域のいずれか）を確立する。トラステッドコアの動作中に、トラステッドコアは、その保護された空間内で実行中のトラステッドアプリケーションのみが、その保護された空間内の他のメモリアドレスにアクセスできることを保証するように動作する。これらの実施形態では、メモリコントローラ１０６のすべての制御（たとえば、パラメータレジスタ１５３など、コントローラ１０６のアドレス指定可能な部分）は、保護された空間内に含まれる。加えて、トラステッドコア初期化コマンドが発行される機構（たとえば、特定のレジスタアドレス、特定のポートなど）も保護された空間に含まれ、信用できないコードがトラステッドコア初期化コマンドを再発行するのを防止する。

【０１３６】
ある実施態様では、ＣＰＵ１０２，１０４が２つの動作モード、すなわち、実モード(real mode)および保護モード(protected mode)をサポートする。実モードは、前のＣＰＵ（たとえば、Ｉｎｔｅｌ　８０８６プロセッサ）のプログラミング環境を提供することを指し、保護モードは、ＣＰＵのすべての命令およびアーキテクチャの特徴を使用可能にする。これらの実施態様では、ＣＰＵ１０２，１０４が、電源が入れられたときに（かつ、ＲＥＳＥＴ＃信号に応答して）、実モードに初期化される。トラステッドコア初期化プロセスが開始し、これは、各ＣＰＵについて、そのＣＰＵを保護モードに移行し、次いでメモリページングをオンにすることによって行う（メモリページングは、仮想メモリを指し、メモリの「ページ」をＣＰＵによって参照することができ、メモリ１１０と別の記憶デバイス（たとえば、デバイス１１８）の間で必要とされたときにスワップすることができる）。したがって、トラステッドコア初期化プロセスにおいて、リセットされ、メモリへのアクセスが許可される追加のＣＰＵについて（最初のＣＰＵの後）、これらが実モードにおいて、保護モードおよびオンにされたページングに移行されるために十分長く動作することができることに配慮するべきである。これは、たとえば、追加のプロセッサ開始ベクトル（リセットベクトル１４２）の選択における配慮によって実施することができる。

【０１３７】
また、トラステッドコアの外部のコードが、強制のプロセッサリセットを構成した場合（これはしばしばプログラムにより可能であり、一般的にはＰＣクラスのマシンで可能である）、プロセッサは、動作中のコアを潜在的にくつがえす可能性のあるコードではなく、

トラステッドコアを実行することによって起動することにも留意されたい。この場合、そのコアは、これを例外条件として処理することを選択し、すべての既存の内部状態を破棄する（メモリをクリアする）ことができ、あるいは、実行を継続するための動作を取ることができる（たとえば、プロセッサのリセット前に実施されていたＣＰＵ保護プロファイルを再ロードする）。

【０１３８】

トラステッドコア初期化プロセスには、プロセッサバス１１２においていかなる追加のバストランザクションも必要ではなく、ＣＰＵ１０２，１０４における追加のピン、または、ＣＰＵ１０２，１０４への他のこのような修正も必要ではないことに留意されたい。むしろ、初期化プロセスがメモリコントローラにおいて（たとえば、チップセットにおいて）実施され、それによりＣＰＵへのいかなる修正の必要性も防止される。

【０１３９】

本明細書では、主としてコンピュータのブート中に実行されるものとして論じたが、トラステッドコア初期化プロセスを異なる時点で実行することができる。たとえば、トラステッドコアが動作することを必要とするアプリケーションが実行される（たとえば、ユーザが、保護される必要のあるマルチメディアコンテンツをダウンロードしようとする）まで、トラステッドコアをメモリ１１０にロードすることができず、トラステッドコア初期化コマンドを発行することができない。加えて、トラステッドコア初期化プロセスを、多くの回数繰り返すことができる。例として、トラステッドコアコードは、トラステッドコアの実行を終了する「トラステッドコアを終了させる(terminate trusted core)」コマンドを有している可能性がある。しかし、トラステッドコアを、後の時点で再度メモリ１１０にロードすることができ（それが実際に実行終了の一部として除去されたかのように）、トラステッドコア初期化プロセスを、トラステッドコアコードが再度実行を開始するために繰り返すことができる。また、多数の異なるトラステッドコアを、一度に１つずつ、交互に動作することができる。

【０１４０】

加えて、トラステッドコア初期化プロセスが完了し、他のＣＰＵがメモリにアクセスできるようにされた後、これらの追加のＣＰＵが、トラステッドコア初期化プロセスが開始するときにコンピュータにおいて存在している必要がないことに留意されたい。例として、コンピュータは、ＣＰＵを「ホットプラグ(hot-plug)」する（コンピュータの動作中にＣＰＵをコンピュータに追加する）ための能力をサポートすることができる。このような新たに追加されたいかなるＣＰＵもメモリにアクセスすることができ、命令の実行を追加のプロセッサリセットベクトルで開始する。

【０１４１】

図１は、単一のメモリコントローラ１０６がメモリ１１０へのアクセスを制御する例示的アーキテクチャを例示する。他の実施形態として、トラステッドコア初期化プロセスを、多数のメモリコントローラがあるアーキテクチャにおいて実施することができる。

【０１４２】

図４は、本発明のある実施形態による、分散メモリコントローラを使用した例示的コンピュータアーキテクチャを例示する。コンピュータ２００は、プロセッサバス２０６に結合された多数（ｍ）のＣＰＵ２０２，２０４を含むように例示される。ＣＰＵ２０２，２０４は、様々なＩ／Ｏ構成要素（図示せず）と、ブリッジ２０８を介して通信する。加えて、各ＣＰＵ２０２，２０４はメモリコントローラ２１０を含む。個別のメモリコントローラ２１０は、分散メモリコントローラアーキテクチャを実施し、これは、図１のメモリコントローラ１０６に類似した機能を実行する。ただし、個別のコントローラ２１０は多数のＣＰＵに渡って分散している。分散メモリコントローラアーキテクチャは、幅広い種類の従来の方法のいずれにおいても実施することができ、追加の結合部（図示せず）を個別のメモリコントローラ２１０の間に含めて、通信をメモリコントローラ２１０の間で渡すことができるようにすることができる（たとえば、個別のメモリコントローラ２１０の同期化のため）。

【０１４３】
図４の例では、各ＣＰＵは、それ自体の直接接続されたメモリを含む。コンピュータ２００に含まれたバスロジックおよびプロトコルは、各ＣＰＵによる全体のメモリサブシステムのビューがコヒーレントの状態であることを保証する。このようなメモリアーキテクチャはしばしば、キャッシュコヒーレント不均等メモリアクセスシステム（ＣＣ－ＮＵＭＡ）と呼ばれる。他の実施形態として、それ自体の接続されたメモリを有する各ＣＰＵではなく、メモリコントローラを多数のＣＰＵの間に分散させて、単一の共有システムメモリにアクセスすることができる。

【０１４４】
図５は、本発明のある実施形態による、コード初期化プロセスのための例示的処理を例示する流れ図である。図５の処理は、図１のコンピュータ１００または図４のコンピュータ２００など、コンピュータによって実施され、ソフトウェアにおいて実施することができる。図５の処理を、トラステッドコア初期化プロセス、または、他の実施形態として、他のコードのための初期化プロセスのために使用することができる。

【０１４５】
最初に、コンピュータのための通常のブート処理が、必ずしも安全ではないＢＩＯＳを使用して開始される（動作２５２）。最終的に、コードがメモリにロードされ（動作２５４）、コード初期化コマンドが受信される（動作２５６）。コード初期化コマンドに応答して、メモリがすべてのプロセッサおよびバスマスタから保護される（動作２５８）。次いで、メモリにおけるコードの暗号化措置が抽出され、安全な場所に格納される（動作２６０）。次いで、プロセッサリセットベクトルが、メモリにおけるコードの開始など、初期化ベクトルにマップされる（動作２６２）。

【０１４６】
プロセッサリセットベクトルが初期化ベクトルにマップされた後、すべてのプロセッサはリセットされ（動作２６４）、プロセッサのうちの１つがメモリにアクセスできるようになる（動作２６６）。次いで、処理は、メモリにアクセスできるようにされたプロセッサがメモリからコード初期化プロセスを実行するまで待機する（動作２６８）。コード初期化プロセスの終了が信号により通知され、これは、たとえば、コード初期化プロセスを実行するプロセッサから発行されたコマンドによる。コード初期化プロセスが終了した後、プロセッサリセットベクトルは追加のプロセッサ開始ベクトルに再マップされる（動作２７０）。次いで、コンピュータにおける他のいかなるプロセッサもメモリにアクセスできるようになり（動作２７２）、コンピュータにおける他のいかなるバスマスタもメモリにアクセスできるようになる（動作２７４）。

【０１４７】
本明細書の考察では、本発明の実施形態を、１つまたは複数の従来のプロセッサまたはコントローラ（たとえば、図１の制御ロジック１３６）によって実行されるプログラムモジュールまたはコードなど、一般にプロセッサ実行可能命令に関して記載してある。一般に、プログラムモジュールまたはコードは、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み、これらが特定のタスクを実行し、あるいは、特定の抽象データ型を実施する。分散環境(distributed environment)では、プログラムモジュールまたはコードは、多数のメモリ記憶デバイスに位置することができる。

【０１４８】
他の実施形態として、本発明の実施形態を、ハードウェア、または、ハードウェア、ソフトウェアおよび／またはファームウェアの組み合わせにおいて実施することができる。たとえば、本発明のすべてまたは一部を、１つまたは複数の特定用途向け集積回路（ＡＳＩＣ）またはプログラマブルロジックデバイス（ＰＬＤ）において実施することができる。

【０１４９】
コンピューティングデバイス（図１のデバイス１００または図４のデバイス２００など）は、一般的には、少なくともある形式のコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピューティングデバイスによってアクセスすることができる、いかなる使用

可能な媒体にすることもできる。例として、限定ではなく、コンピュータ可読媒体は、コンピュータ記憶媒体および通信媒体を含むことができる。コンピュータ記憶媒体は、揮発性および不揮発性，取外し可能および取外し不能な媒体であって、情報の格納のためのいずれかの方法または技術において実施されたものを含む。これら情報は、コンピュータ可読命令，データ構造，プログラムモジュールまたは他のデータなどである。コンピュータ記憶媒体には、ＲＡＭ，ＲＯＭ，ＥＥＰＲＯＭ，フラッシュメモリまたは他のメモリ技術，ＣＤ－ＲＯＭ，デジタル多用途ディスク（ＤＶＤ）または他の光学記憶装置，磁気カセット，磁気テープ，磁気ディスク記憶装置または他の磁気記憶デバイス、または、所望の情報を格納するために使用することができ、コンピューティングデバイスによってアクセスすることができる、他のいかなる媒体もが含まれるが、これらに限定されるものではない。通信媒体は、一般的には、コンピュータ可読命令、データ構造、プログラムモジュールまたは他のデータを、搬送波などの変調データ信号または他の移送機構において実施し、いかなる情報送達媒体をも含む。「変調データ信号(modulate data signal)」という用語は、１つまたは複数のその特性セットを有するか、あるいは、信号において情報を符号化するような方法において変更された信号を意味する。例として、限定ではなく、通信媒体には、有線ネットワークまたは直接有線接続などの有線媒体、および、音波，ＲＦ，赤外線および他の無線媒体などの無線媒体が含まれる。上記のいかなるものの組み合わせもコンピュータ可読媒体の範囲内に含まれるべきである。

【０１５０】

例示のため、オペレーティングシステムなど、プログラムおよび他の実行可能プログラム構成要素は、本明細書において離散的ブロック(discrete block)として例示したが、このようなプログラムおよび構成要素が様々な時点でコンピュータの異なる記憶構成要素において存在し、コンピュータのデータプロセッサによって実行されることを理解されたい。

【０１５１】

（むすび）

上記の説明は、構造上の特徴および／または方法上の動作に特有である言語を使用したが、付属の特許請求の範囲において定義された本発明については、記載された特定の特徴または動作に限定されないことを理解されたい。むしろ、特定の特徴および動作は、本発明を実施する例示的形式として開示したものである。

【図面の簡単な説明】

【図１】本発明の一実施形態によるコンピュータを示すブロック図である。

【図２】トラステッドコアを実現する手法を示す図である。

【図３】トラステッドコアを実現する手法を示す図である。

【図４】本発明の一実施形態による、分散メモリコントローラを使用したコンピュータアーキテクチャを示す図である。

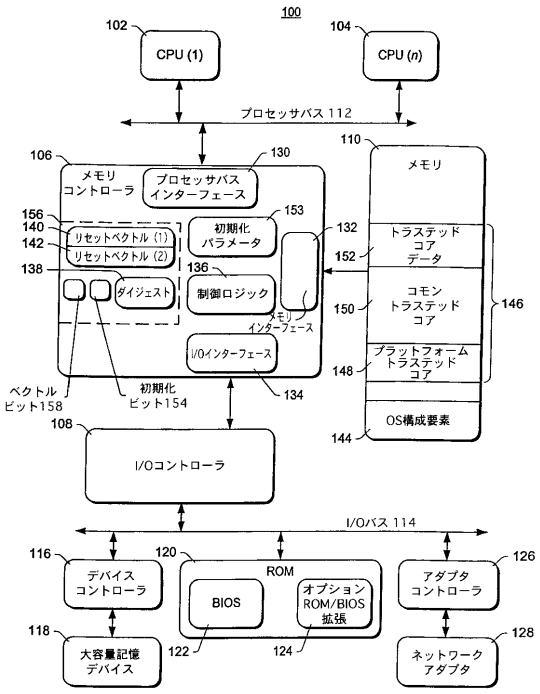
【図５】本発明の一実施形態による、コード初期化プロセスを示す流れ図である。

【符号の説明】

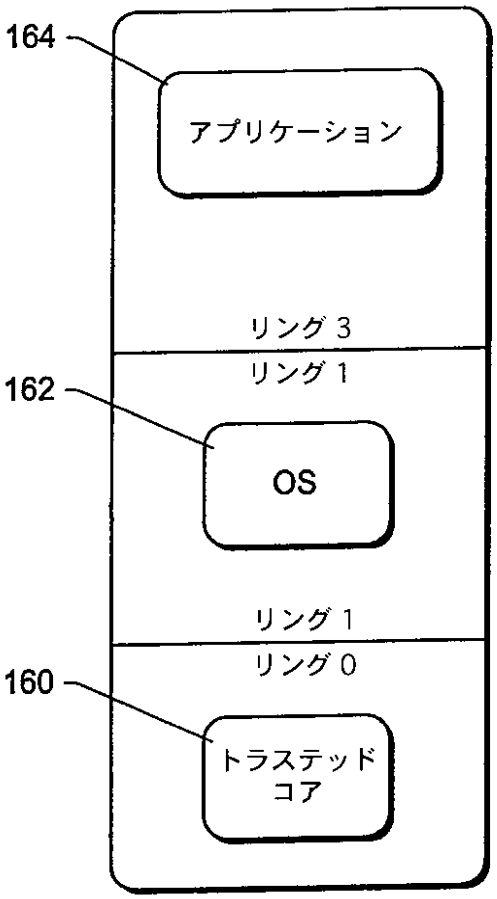
１０２，１０４，２０２，２０４　ＣＰＵ
１０６，２１０　メモリコントローラ
１１０　メモリ
１１２，２０６　プロセッサバス
１１４　Ｉ／Ｏバス
１１８　大容量記憶デバイス
１２０　ＲＯＭ
１２２　ＢＩＯＳ
１２４　オプションのＲＯＭ／ＢＩＯＳ拡張
１３０　プロセッサバスインターフェース
１３２　メモリインターフェース
１３６　制御ロジック
１３８　ダイジェスト

１４０，１４２　　リセットベクトル
１４５　　初期化ビット
１４６，１６０，１７０　　トラステッドコア
１５８　　ベクトルビット

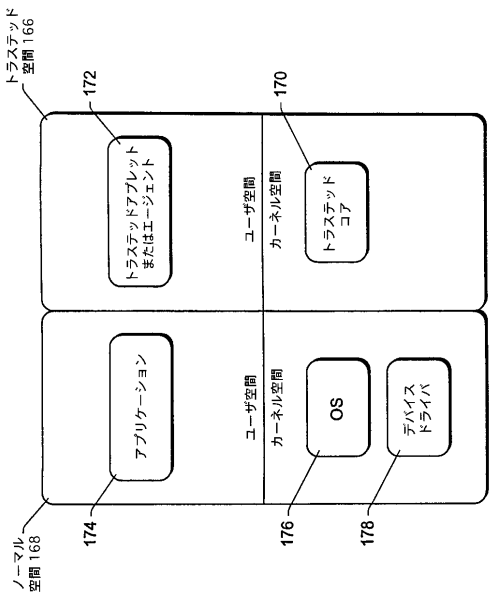
【図１】



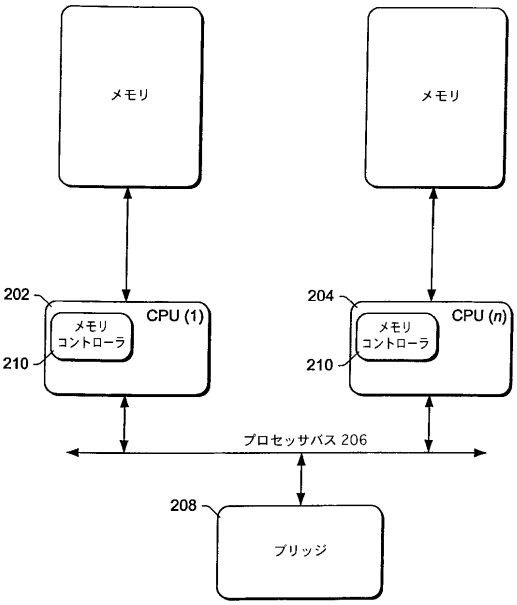
【図２】



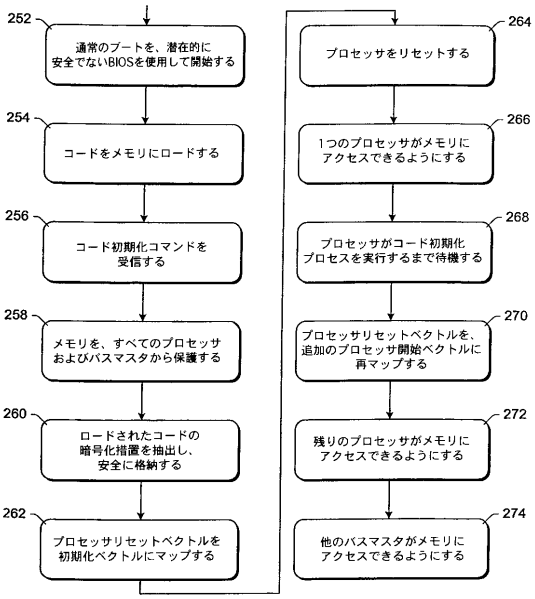
【図３】


トラステッド空間 166
172
170
トラステッドアプレットまたはエージェント
ユーザ空間
カーネル空間
トラステッドコア
アプリケーション
ユーザ空間
カーネル空間
OS
デバイスドライバ
ノーマル空間 168
174
176
178

【図４】


メモリ
メモリ
202
CPU (1)
メモリコントローラ
210
204
CPU (n)
メモリコントローラ
210
プロセッサバス 206
208
ブリッジ

【図５】


252
通常のブートを、潜在的に安全でないBIOSを使用して開始する
254
コードをメモリにロードする
256
コード初期化コマンドを受信する
258
メモリを、すべてのプロセッサおよびバスマスタから保護する
260
ロードされたコードの暗号化指置を抽出し、安全に格納する
262
プロセッサリセットベクトルを初期化ベクトルにマップする
264
プロセッサをリセットする
266
1つのプロセッサがメモリにアクセスできるようにする
268
プロセッサがコード初期化プロセスを実行するまで待機する
270
プロセッサリセットベクトルを、追加のプロセッサ開始ベクトルに再マップする
272
残りのプロセッサがメモリにアクセスできるようにする
274
他のバスマスタがメモリにアクセスできるようにする

フロントページの続き

(72)発明者　ブライアン　ウィルマン
アメリカ合衆国　９８０３４　ワシントン州　カークランド　８０　プレイス　ノースイースト
１０９０１

合議体
審判長　赤川　誠一
審判官　宮司　卓佳
審判官　鈴木　匡明