(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4302493号
(P4302493)
(45)発行日　平成21年7月29日(2009.7.29)　(24)登録日　平成21年5月1日(2009.5.1)

(51)Int.Cl.　F I
G06F 12/14 (2006.01)　G06F 12/14 510A

請求項の数 18　外国語出願　(全 95 頁)

(21)出願番号　特願2003-386040(P2003-386040)
(22)出願日　平成15年11月17日(2003.11.17)
(65)公開番号　特開2004-171563(P2004-171563A)
(43)公開日　平成16年6月17日(2004.6.17)
審査請求日　平成17年12月8日(2005.12.8)
(31)優先権主張番号　0226886.0
(32)優先日　平成14年11月18日(2002.11.18)
(33)優先権主張国　英国(GB)
(31)優先権主張番号　0303446.9
(32)優先日　平成15年2月14日(2003.2.14)
(33)優先権主張国　英国(GB)

(73)特許権者　594154428
エイアールエム　リミテッド
イギリス国　シービー１　９エヌジェイ　ケンブリッジ，チェリー　ヒントン，フルバーン　ロード　１１０
(74)代理人　100066692
弁理士　浅村　皓
(74)代理人　100072040
弁理士　浅村　肇
(74)代理人　100094673
弁理士　林　鉐三
(74)代理人　100091339
弁理士　清水　邦明

最終頁に続く

(54)【発明の名称】データ処理装置内のメモリへアクセスするための技術

(57)【特許請求の範囲】
【請求項１】
安全ドメインおよび非安全ドメインを有し、前記非安全ドメインでアクセスできない安全データに前記安全ドメインで複数デバイスがアクセスするデータ処理装置において、
デバイスバスと、
前記デバイスバスに結合され、
それぞれが前記非安全ドメインにおける１つのモードである少なくとも１つの非安全モードと、前記安全ドメインにおける１つのモードである少なくとも１つの安全モードとを含む複数のモードで少なくとも１つの前記デバイスが作動できるようになっている、前記安全ドメインまたは前記非安全ドメインのいずれかに関するメモリアクセスリクエストを発生するようになっている複数のデバイスと、
前記デバイスバスに結合されており、前記複数のデバイスが必要とするデータを記憶するようになっているメモリとを備え、該メモリが安全データを記憶するための安全メモリと非安全データを記憶するための非安全メモリとを備え、
前記メモリ内のデータのあるアイテムへのアクセスが必要となったときに、それぞれのデバイスが前記デバイスバスにメモリアクセスリクエストを発生するようになっており、前記メモリアクセスリクエストが前記安全ドメインに関係するか、または前記非安全ドメインに関係するかのいずれかを識別するドメイン信号を、前記デバイスによって発生されるメモリアクセスリクエストが含み、そして前記ドメイン信号が前記メモリアクセスリクエストで定義された前記アクセスを始めてよいかを決定するために使用される、データ処

理装置。
【請求項２】
前記少なくとも１つの前記デバイスに対して、前記複数のモードが前記安全ドメインと非安全ドメインで反復される請求項１記載のデータ処理装置。
【請求項３】
前記複数のデバイスが前記ドメイン信号を前記デバイスバスに出力するための所定のピンを有する、請求項１記載のデータ処理装置。
【請求項４】
前記非安全ドメインでは、前記少なくとも１つの前記デバイスが非安全オペレーティングシステムの制御により作動でき、前記安全ドメインでは前記少なくとも１つの前記デバイスが安全オペレーティングシステムの制御により作動できる、請求項１記載のデータ処理装置。
【請求項５】
前記デバイスバスに結合されたパーティションチェックロジックを更に備え、前記デバイスの１つが発生した前記メモリアクセスリクエストが前記非安全ドメインに関するものであるときは、前記メモリアクセスリクエストが前記安全メモリへのアクセスを求めているかどうかを検出するように、前記パーティションチェックロジックが作動でき、かかる検出がされたときに、前記メモリアクセスリクエストが指定するアクセスを防止するようになっている、請求項１記載のデータ処理装置。
【請求項６】
前記パーティションチェックロジックが前記安全ドメインにおいて所定の安全モードで作動するときに、前記デバイスの１つによって管理される、請求項５記載のデータ処理装置。
【請求項７】
前記デバイスバスに発生されるメモリアクセスリクエストの間を仲裁するように、前記デバイスバスに結合されたアービッタ内に前記パーティションチェックロジックが設けられている、請求項５記載のデータ処理装置。
【請求項８】
前記少なくとも１つの前記デバイスがプロセッサを内蔵するチップであり、このチップが更にメモリ管理ユニットを含み、前記プロセッサがメモリアクセスリクエストを発生すると、前記メモリ管理ユニットが前記デバイスバスへの前記メモリアクセスリクエストの発生を制御するための１つ以上の所定のアクセス制御機能を実行するようになっている、請求項１記載のデータ処理装置。
【請求項９】
前記チップが、
システムバスを介して前記プロセッサに結合された別のメモリを更に含み、前記プロセッサが必要とするデータを前記別のメモリが記憶するように作動でき、この別のメモリが安全データを記憶するための別の安全メモリと、非安全データを記憶するための別の非安全メモリとを備え、更に前記デバイスバスに結合された別のパーティションチェックロジックを更に備え、前記デバイスが発生した前記メモリアクセスリクエストが前記非安全ドメインに関するものであるときはいつも、前記メモリアクセスリクエストが前記安全メモリまたは前記別の安全メモリへのアクセスを求めていることを検出するよう前記別のパーティションチェックロジックが作動でき、かかる検出がされたときに、前記メモリアクセスリクエストが指定するアクセスを防止するようになっている、請求項８記載のデータ処理装置。
【請求項１０】
安全ドメインおよび非安全ドメインを有し、前記非安全ドメインでアクセスできない安全データに前記安全ドメインで複数のデバイスアクセスするデータ処理装置において、
デバイスバスと、前記デバイスバスに結合され、
それぞれが前記非安全ドメインにおける１つのモードである少なくとも１つの非安全モ

ードと、前記安全ドメインにおける１つのモードである少なくとも１つの安全モードとを含む複数のモードで少なくとも１つの前記デバイスが作動できるようになっている、前記安全ドメインまたは前記非安全ドメインのいずれかに関するメモリアクセスリクエストを発生するようになっている複数のデバイスと、

前記デバイスバスに結合されており、前記複数のデバイスが必要とするデータを記憶するようになっているメモリとを備え、該メモリが安全データを記憶するための安全メモリと非安全データを記憶するための非安全メモリとを備えた、データ処理装置におけるメモリにアクセスする方法において、

（ｉ）前記メモリ内のデータのアイテムへのアクセスが必要となったときに、任意の前記デバイスから前記デバイスバスに前記メモリアクセスリクエストを発生する工程と、

（ｉｉ）前記メモリアクセスリクエストが前記安全ドメインに関するものか、または非安全ドメインに関するものかのいずれかを識別するドメイン信号を前記メモリアクセスリクエスト内に含ませる工程と、そして

（ｉｉｉ）前記ドメイン信号が前記メモリアクセスリクエストで定義された前記アクセスを始めてよいかを決定するために使用される工程とを備えたデータ処理装置内のメモリへアクセスする方法。

【請求項１１】

前記少なくとも１つの前記デバイスに対して、前記複数のモードが前記安全ドメインと非安全ドメインで反復される、請求項１０記載の方法。

【請求項１２】

前記複数のデバイスが前記ドメイン信号を前記デバイスバスに出力するための所定のピンを有する、請求項１０記載の方法。

【請求項１３】

前記非安全ドメインでは、前記少なくとも１つの前記デバイスが非安全オペレーティングシステムの制御により作動でき、前記安全ドメインでは前記少なくとも１つの前記デバイスが安全オペレーティングシステムの制御により作動できる、請求項１０記載の方法。

【請求項１４】

（ｉｉｉ）１つの前記デバイスによって発生された前記メモリアクセスリクエストが前記非安全ドメインに関するものであるときはいつも、前記メモリアクセスリクエストが前記安全メモリへのアクセスを求めているかどうかを検出するように、デバイスバスに結合されたパーティションチェックロジックを使用する工程と、

（ｉｖ）かかる検出時に、前記メモリアクセスリクエストが指定するアクセスを防止する工程とを更に備えた、請求項１０記載の方法。

【請求項１５】

前記パーティションチェックロジックが前記安全ドメインにおいて所定の安全モードで作動するときに、１つの前記デバイスによって管理される、請求項１４記載の方法。

【請求項１６】

前記デバイスバスに発生されるメモリアクセスリクエストの間を仲裁するように、前記デバイスバスに結合されたアービッタ内に前記パーティションチェックロジックが設けられている、請求項１４記載の方法。

【請求項１７】

前記少なくとも１つの前記デバイスがプロセッサを内蔵するチップであり、該チップが更にメモリ管理ユニットを含む、請求項１０記載の方法において、前記プロセッサが前記メモリアクセスリクエストを発生する際に、

前記デバイスバスへの前記メモリアクセスリクエストの発生を制御するための１つ以上の所定のアクセス制御機能を実行するためのメモリ管理ユニットを使用する工程を含む方法。

【請求項１８】

前記チップがシステムバスを介して前記プロセッサに結合された別のメモリを更に含み、前記別のメモリが前記プロセッサが必要とするデータを記憶するようになっており、前

記別のメモリが安全データを記憶するための別の安全メモリと、非安全データを記憶するための別の非安全メモリとを含み、前記チップが更にシステムバスに結合された別のパーティションチェックロジックを含み、

前記プロセッサが前記非安全ドメインにおいて非安全モードで作動する際に、前記メモリアクセスリクエストが前記プロセッサによって発生されるときはいつも、前記メモリアクセスリクエストが安全メモリまたは別の安全メモリへのアクセスを求めていることを検出するよう、前記別のパーティションチェックロジックを使用する工程と、

かかる検出時に前記メモリアクセスリクエストが指定するアクセスを防止する工程とを備えた、請求項１７記載の方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明はデータ処理装置内のメモリにアクセスするための技術に関する。

【背景技術】

【０００２】

データ処理装置は、このデータ処理装置にロードされたアプリケーションを実行するためのプロセッサを一般に含む。このプロセッサはオペレーティングシステムの制御により作動する。データ処理装置のメモリには特定のアプリケーションを実行するのに必要なデータが一般に記憶される。このデータはアプリケーション内に含まれる命令および／またはプロセッサでのこれら命令の実行中に使用される実際のデータ値から構成できることが理解できよう。

アプリケーションのうちの少なくとも１つによって使用されるデータが、プロセッサで実行される他のアプリケーションがアクセス不可能な秘密データとなっている場合が多く生じる。一例として、データ処理装置がスマートカードであり、アプリケーションのうちの１つが有効化、認証、暗号解読などを実行するために、秘密データ、例えば安全キーを使用するセキュリティアプリケーションである場合が挙げられる。かかる状況では、データ処理装置にロードできる他のアプリケーション、例えば安全データにアクセスしようとする目的でデータ処理装置にロードされたハッキング用アプリケーションにより、データにアクセスできないように、かかる秘密データを安全な状態に維持することを保証することは明らかに重要である。

【発明の開示】

【発明が解決しようとする課題】

【０００３】

公知のシステムでは、オペレーティングシステムが十分なセキュリティを提供し、オペレーティングシステムの制御により実行中の他のアプリケーションによって、あるアプリケーションの安全データにアクセスできないように保証することが、オペレーティングシステム開発者の作業であった。しかしながら、システムがより複雑になるにつれ、一般的な動向としてオペレーティングシステムが次第に巨大になり、より複雑となっている。かかる状況では、オペレーティングシステム内で十分なセキュリティを保証することが次第に困難となっている。

米国特許出願第US2002/0007456A1号および米国特許第US6,282,657B号および同第US6,292,874B号には、秘密データの安全な記憶を行い、有害なプログラムコードから保護しようとするシステム例が記載されている。

従って、データ処理装置のメモリ内に含まれるかかる安全データのセキュリティを保持せんとするための改良された技術を提供することが望ましい。

【０００４】

第１の特徴によれば、本発明は安全ドメインおよび非安全ドメインを有し、前記非安全ドメインでアクセスできない安全データに前記安全ドメインでアクセスするデータ処理装置において、

デバイスバスと、

前記デバイスバスに結合され、前記安全ドメインまたは前記非安全ドメインのいずれかに関するメモリアクセスリクエストを発生するようになっているデバイスと、

前記デバイスバスに結合されており、前記デバイスが必要とするデータを記憶するようになっているメモリとを備え、該メモリが安全データを記憶するための安全メモリと非安全データを記憶するための非安全メモリとを備え、

前記メモリ内のデータのあるアイテムへのアクセスが必要となったときに、前記デバイスが前記デバイスバスにメモリアクセスリクエストを発生するようになっており、前記メモリアクセスリクエストが前記安全ドメインに関係するか、または前記非安全ドメインに関係するかのいずれかを識別するドメイン信号を、前記デバイスによって発生されるメモリアクセスリクエストが含む、データ処理装置を提供するものである。

【０００５】

本発明によれば、データ処理装置はデバイスバスを介してメモリに結合されたデバイスを備え、このデバイスは安全ドメインまたは非安全ドメインのいずれかに関するメモリアクセスリクエストを発生するようになっている。メモリはデバイスが必要とするデータを記憶するようになっており、安全データを記憶するための安全メモリと、非安全データを記憶するための非安全メモリとを含む。

メモリ内のデータのあるアイテムへのアクセスが必要とされると、デバイスはデバイスバスにメモリアクセスリクエストを発生する。本発明によれば、デバイスが発生するメモリアクセスリクエストはこのメモリアクセスリクエストが安全ドメインに関するものであるか、または非安全ドメインに関するものであるかのいずれかを識別するドメイン信号を含む。

アプリケーションがデバイスにロードされた場合、このアプリケーションは非安全ドメイン内でデフォルトにより実行される。これと対照的に、安全ドメインは所定の特定の安全機能に対してしか使用されないので、所定の安全アプリケーションを作動させるのにしか使用されない。従って、安全ドメインと非安全ドメインとが存在することによって実際にデバイス内に２つの異なる世界が構成される。すなわち所定のキーとなる安全機能が実行される安全世界と他のすべてのアプリケーションが実行される非安全世界とが構成される。

【０００６】

メモリアクセスリクエストに関連してデバイスが発生するドメイン信号は、メモリアクセスリクエストがいずれのドメインに関係するものであるかを指定し、このことはメモリアクセスリクエストが定めるアクセスの進行を認めるべきかどうかを判断するのに使用できる。デバイスバスに結合されたメモリは種々の形態を取り得る。従って、例えばランダムアクセスメモリ（ＲＡＭ）、リードオンリーメモリ（ＲＯＭ）、ハードディスク、所定の周辺デバイス内に設けられたレジスタなどから構成できる。従って、メモリアクセスリクエストが向けられる先の実際のメモリデバイスは、そのメモリのどの部分が安全メモリであるかを知っていると仮定した場合、このメモリデバイスはドメイン信号を使用し、デバイスが発生するメモリアクセスリクエストが非安全ドメインに関するものであるかどうかを識別し、よってかかる状況において、メモリの安全部分へのどのアクセスも防止できる。

【０００７】

一実施例では、デバイスは非安全ドメインにおける１つのモードである少なくとも１つの非安全モードと、安全ドメインにおける１つのモードである少なくとも１つの安全モードとを含む複数のモードで作動できる。

メモリアクセスリクエスト内にドメイン信号を組み込むことができるようにする方法が多数あることが理解できよう。ドメイン信号はハードウェアのレベルでアサートされるので、安全ドメイン内で実行できるとデバイス自身が証明したアプリケーションによってしかアサートできないことが好ましい。従って、デバイスで実行中の不正アプリケーションがドメイン信号の設定を不正に操作することはできなくなる。より詳細に説明すれば、好ましい実施例ではデバイスはデバイスバスにドメイン信号を出力するための所定のピンを

有し、ドメイン信号は、デバイスが非安全モードで作動していることを表示するようにデフォルトにより設定される。従って、一実施例では、デバイスが安全ドメインで安全アプリケーションを実行している場合にしか、デバイスが安全ドメインで作動していることを表示するためのドメイン信号が設定されない。ドメイン信号が設定された場合にしか、パーティションチェックロジックはメモリ内の安全データへのアクセスを許可しない。

好ましい実施例では、前記非安全ドメインではデバイスは非安全オペレーティングシステムの制御により作動でき、前記安全ドメインではデバイスは安全オペレーティングシステムの制御により作動できる。安全オペレーティングシステムは一般に非安全オペレーティングシステムよりもかなり小さく、所定の安全機能を制御するのに設けられた安全カーネルとして見ることができる。このようなアプローチにより、安全ドメインは制御された環境で所定の秘密演算を実行できるようにする安全な世界を構成すると見なすことができる。他のアプリケーションは非安全ドメインに留まる。この非安全ドメインは非安全な世界と見なすことができる。

【０００８】

上記のように、一実施例ではメモリアクセスリクエストが発生された先の個々のメモリユニット自身は、非安全モードで作動しているときのデバイスにより、そのユニット内の安全メモリにアクセスされないことを保証するように、メモリアクセスリクエストを規制する役割を果たす。このことは、メモリアクセスリクエストの一部として発生されたドメイン信号を参照して行われる。しかしながら、一実施例ではデータ処理装置はデバイスバスに結合されたパーティションチェックロジックを更に含み、このパーティションチェックロジックはデバイスが発生したメモリアクセスリクエストが前記非安全ドメインに関するものであるときはいつも、メモリアクセスリクエストが安全メモリへのアクセスをすることを望んでいることを検出するようになっており、かかる検出時にそのメモリアクセスリクエストが指定したアクセスを防止するようになっている。従って、かかる実施例ではデバイスバスに結合されたパーティションチェックロジックにより、このような規制機能が中心で実行される。

【０００９】

パーティションチェックロジックは安全メモリと非安全メモリとの間のパーティションに関する情報へアクセスする。安全メモリと非安全メモリとの間の物理パーティションを変更できない実施例では、このパーティション情報をハードウェアに実装できることが理解できよう。しかしながら、好ましい実施例ではデバイスが所定の安全モードで作動しているときは、安全メモリと非安全メモリとの間のパーティションをデバイスによって構成できる。かかる実施例ではパーティションチェックロジックはその所定の安全モードで作動しているときのデバイスによって管理される。従って、安全データにアクセスしようとする目的でデバイスに不正アプリケーションがインストールされた場合、そのアプリケーションを安全ドメインで実行することはできないので、このアプリケーションはパーティション情報を変更できない。従って、そのアプリケーションが安全メモリ内のロケーションにアクセスしようと望むメモリアクセスリクエストを出力できた場合でも、デバイスの非安全モードで実行中のアプリケーションが安全メモリロケーションへのアクセスをしようとしていることをパーティションチェックロジックが検出し、このようなアクセスが行われるのを防止する。

【００１０】

好ましい実施例では、デバイスバスに発生されたメモリアクセスリクエストの間を仲裁するように、デバイスバスに結合されたアービッタ内にパーティションチェックロジックが設けられる。このパーティションチェックロジックとアービッタとを関連させることは好ましいことがこれまでに判っている。この場合、アービッタは競合するメモリアクセスリクエストの間の優先度を判断し、パーティションチェックロジックはアービッタが付与するメモリアクセスリクエストの発生を認めることができるかどうかを判断する。

デバイスは種々の形態をとり得ることが理解できよう。バスに結合されたかかるデバイスは多数存在し得る。バスに多数のデバイスが結合されている場合、安全モードおよび非

安全モードの双方で作動できる各々のデバイスは、安全モードで作動しているときにパーティションチェックロジックを独立して制御でき、かかる状況ではパーティションチェックロジックは別個の各デバイス固有のパーティション情報にアクセスする。しかしながら、これらデバイスのうちの１つがパーティションチェックロジックを管理する役割を果たすことが好ましい。

【００１１】

好ましい実施例では、少なくとも１つのデバイスはプロセッサを内蔵するチップであり、このチップは更にメモリ管理ユニットを含み、プロセッサがメモリアクセスリクエストを発生すると、このメモリ管理ユニットはデバイスバスへのメモリアクセスリクエストの発生を制御するための１つ以上の所定のアクセス制御機能を実行するようになっている。このデータ処理装置の１つ以上の他の部品は同じチップに設けてもよいし、チップ外に設けてもよい。

デバイスバスに結合されたメモリは種々の形態、例えばランダムアクセスメモリ（ＲＡＭ）、リードオンリーメモリ（ＲＯＭ）、ハードディスク、所定の周辺デバイス内に設けられたレジスタなどの形態をとり得る。デバイスがプロセッサに関連するシステムバスを備えたプロセッサを内蔵するチップであるときには、かかるメモリの他にシステムバスに接続された所定のメモリ、例えばキャッシュメモリ、密結合メモリ（ＴＣＭ）などが設けられていてもよい。

【００１２】

従って、所定の実施例では、前記チップはシステムバスを介して前記プロセッサに結合された別のメモリを含み、該別のメモリは前記プロセッサが必要とするデータを記憶するように作動でき、この別のメモリは安全データを記憶するための別の安全メモリと、非安全データを記憶するための別の非安全メモリとを備え、更に前記デバイスバスに結合された別のパーティションチェックロジックを更に備え、前記デバイスが発生した前記メモリアクセスリクエストが前記非安全ドメインに関するものであるときはいつも、前記メモリアクセスリクエストが前記安全メモリまたは前記別の安全メモリへのアクセスを求めていることを検出するよう前記別のパーティションチェックロジックが作動でき、かかる検出がされたときに、前記メモリアクセスリクエストは指定するアクセスを防止するようになっている。

デバイスに結合されたパーティションチェックロジックはシステムバスに接続されたメモリに関するパーティションチェック機能を実行できないので、プロセッサが非安全モードで作動しているときに安全メモリシステムの一部はデバイスバスに結合された安全メモリの一部であるか、または安全データを含む別のメモリの一部であるかどうかに係わらず、安全メモリシステムのどの部分にもアクセスできないように保証するために、かかる実施例では別のパーティションチェックロジックが設けられている。

【００１３】

第２の特徴によれば、本発明は安全ドメインおよび非安全ドメインを有し、前記非安全ドメインでアクセスできない安全データに前記安全ドメインでアクセスするデータ処理装置において、

デバイスバスと、前記デバイスバスに結合され、前記安全ドメインまたは前記非安全ドメインのいずれかに関するメモリアクセスリクエストを発生するようになっているデバイスと、前記デバイスバスに結合されており、前記デバイスが必要とするデータを記憶するようになっているメモリとを備え、該メモリが安全データを記憶するための安全メモリと非安全データを記憶するための非安全メモリとを備えた、データ処理装置におけるメモリにアクセスする方法において、

（ｉ）前記メモリ内のデータのアイテムへのアクセスが必要となったときに、前記デバイスから前記デバイスバスに前記メモリアクセスリクエストを発生する工程と、

（ｉｉ）前記メモリアクセスリクエストが前記安全ドメインに関するものか、または非安全ドメインに関するものかのいずれかを識別するドメイン信号を前記メモリアクセスリクエスト内に含ませる工程とを備えたデータ処理装置内のメモリへアクセスする方法を提

供するものである。
別の特徴によれば、本発明はデバイスバスと、
前記デバイスバスに結合されており、複数のモードおよび安全ドメインまたは非安全ドメインで作動できるデバイスとを備え、前記複数のモードが非安全ドメインにおける１つのモードである少なくとも１つの非安全モードと、安全ドメインにおける１つのモードである少なくとも１つの安全モードとを含み、
前記デバイスバスに結合されており、前記デバイスが必要とするデータを記憶するように作動できるメモリを備え、該メモリが安全データを記憶するための安全メモリと非安全データを記憶するための非安全メモリとを含み、
前記メモリにおけるデータのアイテムへのアクセスが必要となったときに、前記デバイスが前記デバイスバスにメモリアクセスリクエストを発生するようになっており、前記デバイスが前記少なくとも１つの安全モードで作動しているか、または前記少なくとも１つの非安全モードで作動しているか否かを識別するドメイン信号を、前記デバイスによって発生される前記メモリアクセスリクエストが含む、データ処理装置を提供するものである。
更に別の特徴によれば、本発明はデバイスバスと、前記デバイスバスに結合されており、複数のモードおよび安全ドメインまたは非安全ドメインで作動できるデバイスとを備え、前記複数のモードが非安全ドメインにおける１つのモードである少なくとも１つの非安全モードと、安全ドメインにおける１つのモードである少なくとも１つの安全モードとを含み、前記デバイスバスに結合されており、前記デバイスが必要とするデータを記憶するように作動できるメモリを備え、該メモリが安全データを記憶するための安全メモリと非安全データを記憶するための非安全メモリとを含む、データ処理装置におけるメモリにアクセスする方法において、
（ｉ）前記メモリ内のデータのアイテムへのアクセスが必要とされたときに、前記デバイスから前記デバイスバスに前記メモリアクセスリクエストを発生する工程と、
（ｉｉ）前記メモリアクセスリクエストが前記安全ドメインに関するものか、または非安全ドメインに関するものかのいずれかを識別するドメイン信号を前記メモリアクセスリクエスト内に含ませる工程とを備えたデータ処理装置内のメモリへアクセスする方法を提供するものである。
【実施例】
【００１４】
図１は本発明の好ましい実施例に係わるデータ処理装置を示すブロック図である。このデータ処理装置はプロセッサコア１０を含み、このコア１０内には命令のシーケンスを実行するようになっている代数論理ユニット（ＡＬＵ）１６が設けられている。このＡＬＵ１６が必要とするデータはレジスタバンク１４内に記憶されており、このコア１０にはプロセッサコアの作動を表示する診断データを捕捉できるようにする種々のモニタ機能が設けられている。一例として、どの作動をトレースすべきかを定める埋め込みトレースモジュール（ＥＴＭ）２２内の所定の制御レジスタ２６の内容に応じてプロセッサコアの所定の作動のリアルタイムのトレースを発生するための埋め込みトレースモジュール（ＥＴＭ）２２が設けられている。これらトレース信号は一般にトレースバッファへ出力され、このトレースバッファから、これらトレース信号を後で分析できるようになっている。種々の周辺機器（図示されず）によって立ち上げられる複数の割り込みサービスを管理するためのベクトル化されたインターラプトコントローラ２１が設けられている。
【００１５】
更に図１に示されるように、コア１０内に設けることができる別のモニタ機能としてデバッグ機能がある。１つ以上のスキャンチェーン１２に結合されているジョイントテストアクセスグループ（ＪＴＡＧ）コントローラ１８を介してデータ処理装置の外部のデバッグアプリケーションがコア１０と通信できるようになっている。スキャンチェーン１２およびＪＴＡＧコントローラ１８を介して外部デバッグアプリケーションへプロセッサコア１０の種々の部分のステータスに関する情報を出力することができる。デバッグ機能を開

始すべきか停止すべきかを識別する条件をレジスタ２４内に記憶するのに、インサーキットエミュレータ（ＩＣＥ）２０が使用され、従って、このエミュレータは例えばブレークポイント、ウォッチポイントなどを記憶するのに使用される。

【００１６】

コア１０はメモリ管理ロジック３０を介してシステムバス４０に結合されており、メモリ管理ロジック３０はデータ処理装置のメモリ内のロケーションへアクセスするためにコア１０が発生するメモリアクセスリクエストを管理するようになっている。システムバス４０に直接接続されるメモリユニット、例えば密結合メモリ（ＴＣＭ）３６および図１に示されるキャッシュ３８によって、メモリの所定部分を具現化できる。かかるメモリ、例えばダイレクトメモリアクセス（ＤＭＡ）コントローラ３２へアクセスするのに別のデバイスを設けてもよい。一般に、チップの種々の素子の所定の制御パラメータを定めるための種々の制御レジスタ３４が設けられる。本明細書ではこれら制御レジスタをコプロセッサ１５（ＣＰ１５）のレジスタとも称す。

【００１７】

外部バスインターフェース４２を介し、外部バス７０（例えばＡＲＭリミティッド社によって開発された「高度マイクロコントローラバスアーキテクチャ」（ＡＭＢＡ）仕様に従って作動するバス）へコア１０を含むチップを結合できる。これらデバイスはマスターデバイス、例えばデジタル信号プロセッサ（ＤＳＰ）５０またはダイレクトメモリアクセス（ＤＭＡ）コントローラ５２だけでなく、種々のスレーブデバイス、例えばブートＲＯＭ４７、スクリーンドライバ４６、外部メモリ５６、入出力（Ｉ／Ｏ）インターフェース６０またはキー記憶ユニット６４も含むことができる。図１に示されたこれら種々のスレーブデバイスをいずれもデータ処理装置の全メモリの内蔵部品と見なすことができる。例えばブートＲＯＭ４４は外部メモリ５６のようにデータ処理装置のアドレス指定可能なメモリの一部を形成する。更に、スクリーンドライバ４６、Ｉ／Ｏインターフェース６０およびキー記憶ユニット６４のようなデバイスはいずれも内部記憶素子、例えばレジスタまたはバッファ４８、６２、６６をそれぞれ含み、これらバッファはいずれもデータ処理装置の全メモリの一部として別々にアドレス指定可能である。後により詳細に説明するように、メモリの一部、例えば外部メモリ５６の一部を使ってメモリアクセスの制御に対応する情報を構成する１つ以上のページテーブル６８を記憶する。

【００１８】

当業者であれば理解できるように、外部バス７０には一般にアービッターおよびデコーダロジック５４が設けられる。アービッターは多数のマスターデバイス、例えばコア１０、ＤＭＡ３２、ＤＳＰ５０、ＤＭＡ５２などが発生する多数のメモリアクセスリクエストの間の仲裁をするのに使用され、一方、デコーダは特定のメモリアクセスリクエストを外部バス上のどのスレーブデバイスが取り扱うかを決定するのに使用される。

【００１９】

一部の実施例では、コア１０を含むチップの外部に外部バスを設けることができるが、別の実施例では、外部バスはコア１０と共にオンチップ状態で設けられる。このようにした場合、外部バス上の安全データは外部バスをオフチップにしたときよりもセキュリティを維持するのが容易となるという利点がある。外部バスがオフチップとされた場合、安全データのセキュリティを高めるのにデータ暗号化技術を使用できる。

【００２０】

図２は安全ドメインと非安全ドメインとを有する処理システムで作動する種々のプログラムを略図で示す。このシステムには少なくとも部分的にモニタモードで実行されるモニタプログラム７２が設けられる。この実施例ではセキュリティステータスフラグはモニタモードでしか書き込みアクセスできず、このフラグはモニタプログラム７２によって書き込みできる。モニタプログラム７２は安全ドメインと非安全ドメインとの間のすべての変更を両方向に管理する役割を果たす。コアを外から見ると、モニタモードは常に安全であり、モニタプログラムは安全メモリ内にある。

【００２１】

非安全ドメイン内には非安全オペレーティングシステム７４および複数の非安全アプリケーションプログラム７６、７８が設けられ、これらプログラムは非安全オペレーティングシステム７４と共に協働して実行される。安全ドメインには、安全カーネルプログラム８０が設けられている。この安全カーネルプログラム８０は安全なオペレーティングシステムを形成すると見なすことができる。一般に、かかる安全なカーネルプログラム８０は処理活動に不可欠な機能しか実行しないようになっており、安全カーネルプログラム８０をできるだけ小さく、かつ簡単にするように、安全ドメイン内でこれら処理活動を行わなければならない。その理由は、このようにカーネルプログラムを小さく、かつ簡単にすることにより、プログラムをより安全にできるからである。安全カーネルプログラム８０と組み合わせて複数の安全アプリケーション８２、８４が実行されるように示されている。

【００２２】

図３は、異なるセキュリティドメインに関連した処理モードのマトリックスを示す。この特定の例では、処理モードはセキュリティドメインに対して対称的であるので、モード１とモード２とは安全形式および非安全形式の双方で存在する。

【００２３】

モニタモードはシステム内のセキュリティアクセスの最高レベルを有し、この実施例では非安全ドメインと安全ドメインとの間にてシステムを両方向に切り替える権利が与えられた唯一のモードである。従って、すべてのドメインの切り替えはモニタモードへの切り替えおよびモニタモードでのモニタプログラム７２の実行により行われる。

【００２４】

図４は、非安全ドメイン処理モード１、２、３、４および安全ドメイン処理モードａ、ｂ、ｃの別の組を略図で示す。図３の対称的な配置と対照的に、図４はセキュリティドメインのうちの一方または他方に処理モードの一部は存在できないことを示す。モニタモード８６は非安全ドメインと安全ドメインとをまたがるように、再び示されており、このモニタモード８６は安全処理モードと見なすことができる。このモードで安全ステータスフラグを変更することができ、モニタモードにあるモニタプログラム７２は自らセキュリティステータスフラグをセットできる能力を有するので、このモニタモードは効果的にシステム全体のセキュリティを究極レベルにすることができる。

【００２５】

図５は、セキュリティドメインに関する処理モードの別の配置を略図で示す。この配置では、安全ドメインと非安全ドメインとの双方が識別されているだけでなく、別のドメインも識別されている。この別のドメインは、図示された安全ドメインまたは非安全ドメインのいずれとも相互に対話する必要がなく、また、それが属する部分のいずれに対する事項も問題とならないよう、システムの他の部分からアイソレートされたものにすることができる。

【００２６】

後に理解できるように、処理システム、例えばマイクロプロセッサには通常、レジスタバンク８８が設けられ、このバンクにはオペランドの値を記憶できる。図６はレジスタバンクの一例のプログラマーのモデル図を示し、処理モードのうちの所定モードにある所定のレジスタ番号に対し、専用レジスタが設けられている。より詳細には、図６の例は（英国ケンブリッジのＡＲＭリミティッド社のＡＲＭ７プロセッサに設けられているような）公知のＡＲＭレジスタバンクの拡張例であり、このＡＲＭレジスタバンクには専用のセーブされたプログラムステータスレジスタ、専用スタックポインタレジスタおよび各処理モードのための専用リンクレジスタＲ１４が設けられている、しかし、このケースではモニタモードのプロビジョンによって拡張されている。図６に示されるように、高速割り込みモードは、この高速割り込みモードとなったときに他のモードからのレジスタの内容をセーブし、またリストアしなくてもよいように設けられた追加専用レジスタを有する。別の実施例では、このモニタモードに高速割り込みモードと同じように別の専用レジスタを設けることもでき、このようにしてセキュリティドメインの切り替えの処理速度をスピードアップし、かかる切り替えに関連するシステムの待ち時間を小さくすることができる。

【００２７】
図７は、別の実施例を略図で示しており、この実施例では安全ドメインおよび非安全ドメインでそれぞれ使用される２つの完全な別個のレジスタバンクとしてレジスタバンク８８が設けられている。このようなレジスタバンクの配置方法は、非安全ドメインへの切り替えを行うときに安全ドメイン内で作動するレジスタに記憶された安全データにアクセスできるようになることを防止できる１つの方法である。しかしながら、このような配置は非安全ドメインと安全ドメインの双方においてアクセスできるレジスタ内にデータを入れる高速かつ効率的な機構を使用することによって可能となり、かつそのようにすることが望ましい、非安全ドメインから安全ドメインへデータが移動する可能性を阻害する。
【００２８】
安全レジスタバンクを設ける重要な利点は、ある世界から別の世界へ切り替える前にレジスタの内容をフラッシュしなくてもよいことである。待ち時間が重要な問題ではない場合、安全ドメインの世界のために二重のレジスタを有しない、より簡単なハードウェアシステムを、例えば図６により使用できる。モニタモードはあるドメインから他のドメインへ切り替える役割を果たす。少なくともモニタモードで実行中にはモニタプログラムにより内容のリストア、前のコンテクストのセーブだけでなく、レジスタのフラッシュが行われる。従って、このシステムは仮想化モデルのように作動する。このタイプの実施例については後に更に説明する。例えば、本明細書に説明するセキュリティの特徴を組み込んだＡＲＭ７のプログラマーのモデルを参照すべきである。
【００２９】
プロセッサモード
安全な世界におけるコピーモードの代わりに、同じモードが安全ドメインと非安全ドメインの双方をサポートしている（図８参照）。モニタモードは（例えばコプロセッサコンフィギュレーションレジスタから記憶されているＳビットが読み出されるように）コアのその時のステータスが安全ステータスであるのか、または非安全ステータスであるのかを知る。
【００３０】
図８において、ＳＭＩ（ソフトウェアモニタ割り込み命令）が生じるときはいつも、コアはある世界から別の世界へ正しく切り替えるためのモニタモードに入る。
ユーザーモードからＳＭＩが許可される図９を参照する：
１．スケジューラはスレッド１を起動する。
２．スレッド１は安全な機能（ファンクション）、＝＞ＳＭＩ安全コールを実行しなければならず、コアはモニタモードに入る。ハードウェアの制御によりＲ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎ（モニタモードのためのセーブされたプロセッサステータスレジスタ）に現在のＰＣおよびＳＰＳＲ（現在のプロセッサステータスレジスタ）が記憶され、ＩＲＱ／ＦＩＱ割り込みがディスエーブルされる。
３．モニタプログラムは次のタスクを実行する。
－Ｓビットをセットする（安全ステータスフラグ）
－安全アプリケーションの実行中に例外が生じても、非安全コンテクストが失われないよう、スタック内に少なくともＲ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎをセーブする。
－起動すべき新しいスレッド：安全スレッド１があるかどうかをチェックする。安全な世界においてスレッド１がアクティブであることを（一部の実施例におけるスレッドＩＤテーブルを介した）機構が示す。
－ＩＲＱ／ＦＩＱ割り込みを再イネーブルする。次に安全ユーザーモードで安全アプリケーションがスタートできる。
４．スレッド１が終了するまでこれを実行し、モニタプログラムモードの「安全からのリータン」機能へのブランチ（ＳＭＩ）を実行する。（コアがモニタモードに入ったときに、ＩＲＱ／ＦＩＱ割り込みをディスエーブルする。
５．「安全からのリータン」機能は次のタスクを実行する。

－スレッド１が終了したことを表示する（例えばスレッドＩＤテーブルの場合、テーブルからスレッド１を除く）。

－スタックから非安全コンテクストをリストア（復元）し、必要なレジスタをフラッシュし、よって一旦非安全ドメインへのリターンが行われると、安全データを読み出すことができなくなる。

－ＳＵＢＳ命令により非安全ドメインへ戻るように分岐し（これによってプログラムカウンタをカレントポイントにリストアし、ステータスフラグを更新する）、（リストアされたＲ１４＿ｍｏｎからの）ＰＣおよび（ＳＰＳＲ＿ｍｏｎらの）ＳＰＳＲをリストアする。よって非安全ドメインにおけるリターンポイントはスレッド１における、前に実行されたＳＭＩに続く命令となる。

６．終了するまでスレッド１が実行され、スケジューラへハンドを戻す。

特定の実施例によってはモニタプログラムと安全オペレーティングシステムとの間で上記機能の一部を分割してもよい。

別の実施例では、ユーザーモードでＳＭＩが生じないようにすることが望ましい。

【００３１】

安全な世界への入口

リセット

ハードウェアのリセットが生じると、ＭＭＵはディスエーブルされ、ＡＲＭコア（プロセッサ）はＳビットの組により安全スーパーバイザーモードへ分岐する。安全ブートが一旦終了すると、モニタモードに進むＳＭＩを実行することができ、モニタは必要な場合に非安全な世界内のＯＳ（非安全ｓｖｃモード）へ切り替わることができる。これまでのＯＳを使用することが望ましい場合、このＯＳを使用することによってスーパーバイザーモードで簡単にブートすることができ、安全ステートを無視できる。

【００３２】

ＳＭＩ命令

非安全ドメインにおける非安全モード（上記のようにこのモードはＳＭＩを特権モードに制限することが望ましい）からこの命令（モード切り替えソフトウェア割り込み命令）を呼び出すことができるが、関連するベクトルによって決定される目標となる入口ポイントは常に固定されており、モニタモード内にある。これは実行すべき（例えば命令によってパスされたオペランドによって制御される）適当な安全な機能へ分岐するためのＳＭＩハンドラーまでである。

【００３３】

図６のタイプのレジスタバンク内のレジスタバンクの共用レジスタを使って非安全な世界から安全な世界までパラメータの移動を行うことができる。非安全な世界でＳＭＩが生じると、ＡＲＭコアはハードウェア内で次のアクションを行うことができる。

－モニタモードに入るＳＭＩベクトルへのブランチ（モニタモードに入るので安全メモリではアクセスが認められる）。

－Ｒ１４＿ｍｏｎにＰＣをセーブし、ＳＰＳＲ＿ｍｏｎへＳＰＳＲをセーブする。

－モニタプログラムを使ってＳビットをセットする。

－モニタモードで安全な例外ハンドラーの実行をスタートする（マルチスレッドの場合、コンテクストをリストア／セーブ）。

－安全ユーザーモード（またはｓｖｃモードのような別のモード）への分岐をし、適当な機能を実行する。

－コアがモニタモードにある間、ＩＲＱおよびＦＩＱをディスエーブルする（待ち時間が増加する）。

【００３４】

安全な世界からの退出

安全な世界から出るのに２つの可能性がある。

－安全な機能を終了し、この機能を呼び出しした前の非安全モードに戻る場合。

－非安全な例外（例えばＩＲＱ－ＦＩＱ－ＳＭＩ）により安全な機能に割り込む。

【００３５】
安全な機能の正常終了
安全な機能が正常に終了すると、ＳＭＩの直後の命令で非安全な世界内のアプリケーションを再開しなければならない。安全ユーザーモードでは「安全な世界からのリターン」ルーチンに対応する適当なパラメータを有するモニタモードへリターンするＳＭＩ命令を実行する。この段階では、非安全な世界と安全な世界との間のデータの漏れを防止するようにレジスタをフラッシュし、次に非安全コンテクスト用の汎用レジスタをリストアし、非安全な世界内に持っていた値で非安全バンクレジスタを更新する。従って、Ｒ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎに適当な値が与えられ、「ＭＯＶＳ　ＰＣ、Ｒ１４」命令を実行することによってＳＭＩの後で非安全アプリケーションを再開する。

【００３６】
非安全例外に起因する安全な機能からの退出
このケースでは、安全な機能を終了せず、非安全例外ハンドラーに進む前に安全コンテクストをセーブしなければならない、どの割り込みもこの要求が処理される。

【００３７】
安全な割り込み
安全な割り込みに対し、いくつかの可能性がある。
次の場合に応じた２つの可能な解決案が提案される。
－どんな種類の割り込みであるか（安全または非安全）
－ＩＲＱが生じたときにコアがどのモードとなっているか（安全な世界内か非安全な世界内か）

【００３８】
解決案１
この解決案では安全および非安全な割り込みをサポートするのに２つの別個のピンが必要とされる。
非安全な世界内にある間に、
－ＩＲＱが生じた場合、コアはＡＲＭコア、例えばＡＲＭ７内のように、この割り込みを処理するのにコアはＩＲＱモードとなる。
－ＳＩＲＱが生じた場合、コアはモニタモードに進み、非安全コンテクストをセーブし、次に安全なＩＲＱハンドラーに進み、安全な割り込みを取り扱う。
安全な世界内にある間に、
－ＳＩＲＱが生じた場合、コアは安全ＩＲＱハンドラーに進む。コアは安全な世界を離れない。
－ＩＲＱが生じた場合、コアは安全コンテクストがセーブされるモニタモードに進み、次に非安全ＩＲＱハンドラーに進み、この非安全割り込みを扱う。
換言すれば、現在の世界に属さない割り込みが生じたとき、コアは直接モニタモードとなり、そうでない場合、そのときの世界に留まる（図１０参照）。

【００３９】
安全な世界で生じるＩＲＱ
図１１Ａ参照：
１．スケジューラはスレッド１を起動する。
２．スレッド１は安全な機能、＝＞ＳＭＩ安全コールを実行しなければならず、コアはモニタモードに入る。Ｒ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎにそのときのＰＣおよびＣＰＳＲが記憶され、ＩＲＱ／ＦＲＱがディスエーブルされる。
３．モニタハンドラー（プログラム）は次のタスクを実行する：
－Ｓビットをセットする。
－安全アプリケーションの実行中に例外が生じた場合に、非安全コンテクストが失われることがないよう、スタック（更に可能な場合には他のレジスタもプッシュする）内に少なくともＲ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎをセーブする。
－起動する新しいスレッド、例えば安全なスレッド１があるかどうかをチェックす

る。一つの機構が安全な世界内でスレッド１がアクティブであることを（スレッドＩＤテーブルを介し）表示する。

－安全ユーザーモードで安全アプリケーションがスタートできる。次にＩＲＱ／ＦＩＱをイネーブルし直す。

４．安全なスレッド１が実行されている間にＩＲＱが生じる。コアはモニタモード（特定ベクトル）に直接ジャンプし、モニタモードでＲ１４＿ｍｏｎにそのときのＰＣを記憶し、ＳＰＳＲ＿ｍｏｎにＣＰＳＲを記憶し（ＩＲＱ／ＦＩＱを次にディスエーブルする）。

５．安全コンテクストはセーブしなければならず、前の非安全コンテクストをリストアする。Ｒ１４＿ＩＲＱ／ＳＰＳＲ＿ＩＲＱを適当な値で更新するように、モニタハンドラーはＩＲＱモードとなり、次に制御信号を非安全なＩＲＱハンドラーへ送る。

６．ＩＲＱハンドラーはＩＲＱにサービスし、次に非安全な世界内で制御信号をスレッド１へ与える。ＳＰＳＲ＿ＩＲＱおよびＲ１４＿ＩＲＱをＳＰＳＲおよびＰＣにリストアすることにより、スレッド１は割り込まれたＳＭＩ命令を指示する。

７．ＳＭＩ命令を再実行する（２．と同じ命令）。

８．モニタハンドラーはそのスレッドが以前割り込まれたものであるかどうかを見て、スレッド１のコンテクストをリストアする。次にユーザーモードで安全スレッド１に分岐し、割り込まれた命令をポイントする。

９．安全スレッド１は終了するまで作動し、次にモニタモード（専用ＳＭＩ）で「安全からのリターン」機能に分岐する。

１０．「安全からのリターン」機能は次のタスクを実行する。

－安全なスレッド１が終了したことを表示するタスク（例えばスレッドＩＤテーブルの場合、このテーブルからスレッド１を除く）。

－スタックから非安全コンテクストをリストアし、必要なレジスタをフラッシュし、よって非安全な世界に一旦リターンすると、非安全データを読み出すことができるようにするタスク。

－ＳＵＢＳ命令で非安全な世界に戻るよう分岐し、（リストアされたＲ１４＿ｍｏｎから）ＰＣをリストアし、（ＳＰＳＲ＿ｍｏｎから）ＣＰＳＲをリストアするタスク。従って、非安全な世界内のリターンポイントはスレッド１で前に実行されたＳＭＩに続く命令でなければならない。

１１．スレッド１は終了まで実行され、次にスレッド１は制御信号をスケジューラに戻す。

【００４０】

非安全な世界で生じるＳＩＲＱ

図１１Ｂ参照：

１．スケジューラはスレッド１を起動する。

２．安全なスレッド１が実行されている間にＳＩＲＱが生じる。コアはモニタモード（特定ベクトル）に直接ジャンプし、モニタモードにおいてＲ１４＿ｍｏｎにそのときのＰＣを記憶し、ＳＰＳＲ＿ｍｏｎにＣＰＳＲを記憶し、ＩＲＱ／ＦＩＱをディスエーブルする。

３．非安全コンテクストをセーブしなければならず、コアは安全なＩＲＱハンドラーに進む。

４．ＩＲＱハンドラーはＳＩＲＱにサービスし、次に、適当なパラメータを有するＳＭＩを使ってモニタモードハンドラーへ制御を戻す。

５．ＳＵＢＳ命令がコアを非安全な世界にリターンし、割り込まれたスレッド１を再開させるように、モニタハンドラーは非安全コンテクストをリストアする。

６．終了点までスレッド１が実行され、ハンドをスケジューラに戻す。

図１１Ａの機構は安全な世界に入るための決定方法を与えることができるという利点を有する。しかしながら、割り込み優先権に関連したある問題が生じる。すなわち安全割り込みハンドラー内でＳＩＲＱが実行されている間、より高い優先権を有する非安全なＩＲ

Ｑが生じることがある。一旦非安全なＩＲＱが終了すると、コアが安全な割り込みを再開できるように、ＳＩＲＱイベントを再形成する必要が生じる。

【００４１】

解決案２

この機構（図１２参照）では、２つの別個のピンまたは１つのピンだけで、安全な割り込みおよび非安全な割り込みをサポートできる。２つのピンを設けたことにより、割り込み待ち時間が下がる。

非安全な世界にある間に、

－ＩＲＱが生じた場合、コアはＩＲＱモードに進み、ＡＲＭ７システム内と同じようにこの割り込みを処理する。

－ＳＩＲＱが生じた場合、コアはＩＲＱハンドラーに進み、このハンドラーでＳＭＩ命令がコアをモニタモードに分岐させ、非安全コンテクストをセーブし、次に安全なＩＲＱハンドラーに分岐し、安全な割り込みを取り扱う。

安全な世界にある間に、

－ＳＩＲＱが生じた場合、コアは安全なＩＲＱハンドラーに進む。コアは安全な世界から離れない。

－ＩＲＱが生じた場合、コアは安全なＩＲＱハンドラーに進み、ここでＳＭＩ命令がコアをモニタモードに分岐させ（このモードで安全コンテクストがセーブされる）、次に非安全なＩＲＱハンドラーに分岐し、この非安全な割り込みを取り扱う。

【００４２】

安全な世界で生じるＩＲＱ

図１３Ａ参照：

１．スケジューラがスレッド１を起動する。

２．スレッド１は安全機能、＝＞ＳＭＩ安全コールを実行しなければならず、コアはモニタモードに入る。Ｒ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎにそのときのＰＣおよびＣＰＳＲが記憶され、ＩＲＱ／ＦＲＱがディスエーブルされる。

３．モニタハンドラーは次のタスクを実行する：

－Ｓビットをセットする。

－安全なアプリケーションの実行中に例外が生じた場合に、スタック（最終的には他のレジスタ）内に少なくともＲ１４＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎをセーブする。

－起動する新しいスレッド、例えば安全なスレッド１があるかどうかをチェックする。一つの機構が安全な世界内でスレッド１がアクティブであることを（スレッドＩＤテーブルを介し）表示する。

－安全なユーザーモードで安全なアプリケーションがスタートできる。次にＩＲＱ／ＦＩＱをイネーブルし直す。

４．安全なスレッド１が実行されている間にＩＲＱが生じる。コアは安全なＩＲＱモードに直接ジャンプする。

５．コアはＲ１４＿ｉｒｑにそのときのＰＣを記憶し、ＳＰＳＲ＿ｉｒｑにＣＰＳＲを記憶する。ＩＲＱハンドラーはこれが非安全な割り込みであることを検出し、適当なパラメータを有するモニタモードに入るためのＳＭＩを実行する。

６．安全コンテクストはセーブしなければならず、前の非安全コンテクストをリストアする。モニタハンドラーはＣＰＳＲを読み出すことにより、ＳＭＩがどこから来たかを知る。モニタハンドラーはＩＲＱモードに進み、Ｒ１４＿ｉｒｑ／ＳＰＳＲ＿ｉｒｑを読み出し、適当な安全なコンテクストをセーブすることもできる。更にＩＲＱルーチンが一旦終了されると、リストアしなければならない非安全コンテクストをこれら同じレジスタにセーブすることもできる。

７．ＩＲＱハンドラーはＩＲＱにサービスし、次に非安全な世界内で制御信号をスレッド１へ与える。ＳＰＳＲ＿ＩＲＱおよびＲ１４＿ＩＲＱをＳＰＳＲおよびＰＣにリストアすることにより、スレッド１は割り込まれたＳＭＩ命令をポイントする。

８．ＳＭＩ命令を再実行する（２．と同じ命令）。

９．モニタハンドラーはそのスレッドが以前割り込まれたものであるかどうかを見て、スレッド１のコンテクストをリストアする。次にユーザーモードで安全なスレッド１に分岐し、割り込まれた命令をポイントする。

１０．安全なスレッド１は終了するまで作動し、次にモニタモード（専用ＳＭＩ）で「安全からのリターン」機能に分岐する。

１１．「安全からのリターン」機能は次のタスクを実行する。

－安全なスレッド１が終了したことを表示するタスク（例えばスレッドＩＤテーブルの場合、このテーブルからスレッド１を除く）。

－スタックから非安全コンテクストをリストアし、必要なレジスタをフラッシュし、よって非安全な世界に一旦リターンすると、非安全データを読み出すことができるようにするタスク。

－ＳＵＢＳ命令で非安全な世界に戻るよう分岐し、（リストアされたＲ１４＿ｍｏｎから）ＰＣをリストアし、（ＳＰＳＲ＿ｍｏｎから）ＣＰＳＲをリストアするタスク。従って、非安全な世界内のリターンポイントはスレッド１で前に実行されたＳＭＩに続く命令でなければならない。

１２．スレッド１は終了まで実行され、次にスレッド１は制御信号をスケジューラに戻す。

【００４３】

非安全な世界で生じるＳＩＲＱ

図１３Ｂ参照：

１．スケジューラはスレッド１を起動する。

２．安全なスレッド１が実行されている間にＳＩＲＱが生じる。

３．コアはＩＲＱモードに直接ジャンプし、Ｒ１４＿ｉｒｑにそのときのＰＣを記憶し、ＳＰＳＲ＿ｉｒｑにＣＰＳＲを記憶する。次にＩＲＱをディスエーブルする。ＩＲＱハンドラーはこれがＳＩＲＱであることを検出し、適当なパラメータを有するＳＭＩ命令を実行する。

４．一旦、モニタモードとなると、非安全コンテクストをセーブしなければならず、次にコアは安全なＩＲＱハンドラーへ進む。

５．安全なＩＲＱハンドラーはＳＩＲＱのサービスルーチンをサービスし、次に、適当なパラメータを有するＳＭＩを使って制御をモニタへ戻す。

６．ＳＵＢＳ命令がコアを非安全な世界にリターンさせ、割り込まれたＩＲＱハンドラーを再開させるように、モニタハンドラーは非安全コンテクストをリストアする。

７．ＩＲＱハンドラーはＳＵＢＳを実行することにより、非安全なスレッドに戻ることができる。

８．終了点までスレッド１が実行され、ハンドをスケジューラに戻す。

図１２の機構を用いると、入れ子式割り込みの場合にＳＩＲＱイベントを再形成する必要がなくなるが、安全な割り込みが実行される保証はない。

【００４４】

例外ベクトル

（仮想アドレスの見地から、物理ベクトルのテーブルは１つのベクトルテーブルとして見えるが）少なくとも２つの物理ベクトルのテーブルが維持され、一方のテーブルは非安全メモリ内の非安全な世界用のものであり、他方のテーブルは（非安全な世界からアクセスできない）安全メモリ内の安全な世界用のものである。安全な世界および非安全な世界で使用される仮想対物理メモリの異なるマッピングによって、同じ仮想メモリアドレスが物理メモリ内に記憶された異なるベクトルテーブルに効果的にアクセスできるようになる。モニタモードは物理メモリ内に第３のベクトルテーブルを提供するように、常にフラットなメモリマッピングを使用することができる。

【００４５】

割り込みが図１２の機構に従う場合、各テーブルに対し、図１４に示されるような次のベクトルが生じる。このベクトルの組は安全メモリと非安全メモリの双方で附勢される。

【００４６】

【表１】

表１

例外	ベクトルオフセット	対応するモード
リセット	０ｘ００	スーパーバイザーモード（Sビットがセット）
Undef	０ｘ０４	モニタモード／Undefモード
SWI	０ｘ０８	スーパーバイザーモード／モニタモード
プリフェッチアボート	０ｘ０Ｃ	アボートモード／モニタモード
データアボート	０ｘ１０	アボートモード／モニタモード
IRQ/SIRQ	０ｘ１８	IRQモード
FIQ	０ｘ１Ｘ	FIQモード
SMI	０ｘ２０	Undefモード／モニタモード

【００４７】

注。リセットエントリーは安全ベクトルテーブル内にしかない。非安全な世界内でリセットを実行するときは、安全メモリ内でリセットベクトルにアクセスできるように、コアのハードウェアはスーパーバイザーモードに入ること、およびＳビットの設定を強制する。

【００４８】

図１５は安全モード、非安全モードおよびモニタモードにそれぞれ適用できる３つの例外ベクトルのテーブルを示す。安全および非安全オペレーティングシステムの条件および特徴にマッチするように、これら例外ベクトルのテーブルに例外ベクトルをプログラムすることができる。例外ベクトルテーブルの各々はメモリ内にそのテーブルをポイントするベースアドレスを記憶するＣＰ１５内に関連するベクトルテーブルのベースアドレスレジスタを有することができる。例外が生じたときにハードウェアはシステムのそのときのステートに対応するベクトルテーブルのベクトルアドレスレジスタを参照し、使用すべきベクトルテーブルのベースアドレスを決定する。これとは異なり、異なるモードで適用される異なる仮想－物理メモリのマッピングを使用し、異なる物理メモリアドレスに記憶された３つの異なるベクトルテーブルを分離することができる。図１６に示されるように、プロセッサコアに関連するシステム（コンフィギュレーションを制御する）コプロセッサ（ＣＰＵ１５）内に例外トラップマスクレジスタが設けられる。この例外トラップマスクレジスタはそれぞれの例外のタイプに関連するフラグを提供する。これらフラグはハードウェアがそのときのドメイン内に関連する例外用ベクトルに処理を向けるように作動するのか、または（安全モードのあるタイプである）モニタモードへの切り替えを強制し、モニタモードベクトルテーブル内のテーブルに従うのかのいずれかを表示する。例外トラップマスクレジスタ（例外制御レジスタ）はモニタモードからしか書き込みできない。非安全モードにあるときには例外トラップマスクレジスタに対する読み出しアクセスも防止するようにすることもできる。システムが安全なブートおよび後方へのコンパチビリティを保証するように、安全ベクトルテーブル内で指定された安全スーパーバイザーモードにおいてリセットベクトルへレジスタを常にジャンプさせるようになっている場合、図１６の例外トラップマスクレジスタはリセットベクトルのためのフラグを含まないことが理解できよう。図１５では、完全にするためにリセットベクトルは安全スーパーバイザーモードの安全ベクトルテーブル以外のベクトルテーブル内に示されていることが理解できよう。

【００４９】

図１６は、例えば安全なブート中のモニタプログラムにより例外トラップマスクレジスタ内の異なる例外タイプに対するフラグがプログラム可能であることも示している。これとは異なり、ある実現例では、フラグの一部またはすべてを物理入力信号によって提供してもよい。例えば安全な割り込み信号を受信したときに、安全な割り込みフラグＳＩＲＱがモニタモードへのエントリーおよび対応するモニタモード安全割り込みリクエストベクトルの実行を常に強制するように、ハードによる配線をすることができる。図１６は非安全ドメイン例外に関係する例外トラップレジスタの部分しか示しておらず、安全ドメイン例外に対してはプログラム可能なビットの同様な組が提供される。

【００５０】

１つのレベルではハードウェアは、例外制御レジスタのフラグに応じて現在のドメイン例外ハンドラーまたはモニタモード例外ハンドラーにより割り込みに対するサービスを調整するように働くことが上記記載から理解できようが、これは適用される第１レベルの制御にすぎない。例として、安全モードで例外が生じることができ、この安全モードの例外ベクトルは安全モード例外ハンドラーに従うことができるが、次にこの安全モード例外ハンドラーはその例外が非安全例外ハンドラーによって良好に取り扱いできる性質である旨を決定し、従ってＳＭＩ命令を使って非安全モードへの切り替えをし、非安全例外ハンドラーを呼び出す。ハードウェアが非安全例外ハンドラーを開始するように働くことができる場合、この逆も可能であるが、その場合、処理を安全例外ハンドラーまたはモニタモード例外ハンドラーに向ける命令を実行する。

【００５１】

図１７は、新しいタイプの例外に関連した別の可能性のあるタイプの切り替えリクエストをサポートするためのシステムの作動を略図で示すフローチャートである。ステップ９８では、ハードウェアは現在プログラムステータスレジスタ（ＣＰＳＲ）における表示としてのモニタモードを変えようと試みる命令を検出する。かかる試みが検出されると、新しいタイプの例外がトリガーされる。この例外を本明細書ではＣＰＳＲ違反例外と称す。ステップ１００におけるこのようなＣＰＳＲ違反例外が発生する結果、モニタモードにおける適当な例外ベクトルを参照し、ステップ１０２でモニタプログラムが実行され、ＣＰＳＲ違反例外を取り扱う。

【００５２】

上記ＳＭＩ命令のためのサポートに加え、図１７に関連して説明した安全ドメインと非安全ドメインとの切り替えを開始するための機構を設けてもよいことが理解できよう。ＳＭＩ命令によりすべての正当な試みを行うべきであるので、モードを切り替えようとする不正な試みに応答するように、このような例外機構を設けてもよい。これとは異なり、かかる機構を、安全ドメインと非安全ドメインとの切り替えを行うための正当な方法としてもよいし、現在のコードとの後方コンパティビリティを与えるようにかかる機構を設けてもよい。現在のコードは、例えば安全ドメインと非安全ドメインとを切り替えようとする不正な試みを真に行わない場合でも、正常な動作の一部として処理ステータスレジスタをクリアしようとする場合がある。

【００５３】

上記のように、プロセッサがモニタモードで作動中には一般に割り込みがディスエーブルされる。このようなディスエーブルはシステムの安全性を高めるために行われる。割り込みが発生すると、そのときのプロセッサのステータスが割り込み例外レジスタに記憶され、割り込み機能の完了時に割り込みポイントにて割り込まれた機能の処理を再開できる。このプロセスがモニタモードで許可された場合、このことはモニタモードのセキュリティを低下させ、よって安全なデータのリーク路を発生し得る。この理由から、モニタモードでは一般に割り込みはディスエーブルされる。しかしながら、モニタモード中の割り込みのディスエーブルの１つの結果として、割り込みの待ち時間が増す。

【００５４】

機能を実行するプロセッサのステートが記憶されていない場合、モニタモードで割り込みを認めることが可能となる。割り込みの後で機能が再開されない場合に限り、このこと

を行うことができる。従って、安全に再開できる機能のモニタモードに限り割り込みを認めることによって、モニタモードにおける割り込み待ち時間の問題は解決できる。この場合、モニタモードにおけるインターラプトの後では機能の処理に関連するデータは捨てられ記憶されていないが、割り込みが終了してしまった時、最初からその機能の処理を開始することがプロセッサに命令される。上記例では、プロセッサはモニタモードに切り替わったポイントに単に戻るだけであるので、このことは簡単なことである。機能を再スタートすることは、再スタートできることが確実な機能に対してしか可能でなく、このことは、繰り返し可能な結果を生じさせることに留意すべきである。機能を再スタートしたときに異なる結果が生じるように、プロセッサのステータスを機能が変更した場合、機能を再スタートすることはよい考えではない。この理由から、モニタモードでは安全に再スタートできる機能しか割り込みできず、他の機能に対しては割り込みがディスエーブルされる。

【００５５】

図１８は本発明の実施例に従ってモニタモードで生じる割り込みの取り扱い方法を示す。非安全モードにおいて、タスクＡの処理中にＳＭＩが生じると、これによってプロセッサはモニタモードに切り替えられる。ＳＭＩ命令は専用の非安全ＳＭＩベクトルを通してコアをモニタモードにする。ＰＣのそのときのステートがセーブされ、Ｓビットがセットされ、割り込みがディスエーブルされる。一般に非安全モードのＰＣおよびＣＰＳＲをセーブするのにＬＲ＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎが使用される。

【００５６】

次にモニタモードにおいて、機能Ｃが開始される。まず機能Ｃが行うことは割り込みをイネーブルすることであり、次に機能Ｃが処理される。機能Ｃの処理中に割り込みが生じた場合、他の割り込みがディスエーブルされ、その割り込みは取り込まれ、実行される。しかしながら、モニタモードインジケータは割り込みの後で機能を再開すべきでなく、むしろ再スタートすべきであることをプロセッサに示す。これとは異なり、別個の制御パラメータによりこのことをプロセッサに表示してもよい。従って、割り込みの後でＬＲ＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎの値により割り込み例外ベクトルが更新され、プロセッサのその時のステートは記憶されない。

【００５７】

図８に示されるように、割り込みタスク、すなわちタスクＢの完了後にプロセッサは割り込みレジスタにコピーされたＳＭＩ命令のアドレスを読み出し、ＳＭＩを実行し、機能Ｃの処理を再びスタートする。

【００５８】

上記プロセスは機能Ｃが再スタート可能な場合にしか、すなわち再スタートプロセスＣが繰り返し可能な処理ステップを生じる場合にしか働かない。これは、機能Ｃがプロセッサのステータス、例えば将来の処理に影響を与え得るスタックポインタのいずれかを変更したケースではない。このように、繰り返し可能な機能はベキ等性（ｉｄｅｍｐｏｔｅｎｃｅ）を有すると言われている。べき等性を有しない機能の問題を扱う１つの方法は、コードの最初の部分がべき等性を有するように機能を定めるコードを再配置することであり、べき等性を有するようにコードを配置することが不可能となってしまうと、割り込みはディスエーブルされる。例えばコードＣがスタックへの書き込みに関係している場合、少なくとも最初にスタックポインタを更新することなくこれを行うことは可能である。コードを安全に再スタートできると予想できないと判断された場合、機能Ｃのためのコードは割り込みをディスエーブルすることをプロセッサに命令でき、コードはスタックポインタを正しい位置へ更新できる。このことは図１８に示されており、図では機能Ｃの処理によるある方法によって割り込みをディスエーブルしている。

【００５９】

図１９は若干異なる例を示す。この例ではタスクＣの処理による方法であり、追加の制御パラメータがセットされる。このことは、固定ルーチンが最初に実行されると、厳密にはベキ等性ではないタスクＣのその後の部分が安全に再スタートできることを示している

。この固定ルーチンはプロセッサの状態をタスクＣのスタート時の状態にリストアするように働くので、タスクＣを安全に再スタートし、タスクＣは割り込みされなかった場合と同じ、タスクの終了時のプロセッサ状態を発生できる。一部の実施例では別の制御パラメータをセットした時点で、例えばスタックポインタが更新されるようなプロセッサのステートが変更される、短期間の間割り込みをディスエーブルする。これによってプロセッサを後でべき等性状態にリストアすることが可能となる。

【００６０】

追加の制御パラメータがセットされた後に割り込みが生じると、２つの態様が進行し得る。（Ｆ１にて即座に固定ルーチンを実行し、割り込みを処理することもできるし、または即座に割り込みを処理し、割り込みが完了した後にＳＭＩを実行し、次にタスクＣの再スタートをする前に（Ｆ２にて）固定ルーチンを実行する。理解できるように、これら実施例のいずれにおいても、モニタモードで固定ルーチンを実行するので、安全ドメインまたはモニタモードを認識していない非安全ドメインにおける実行は影響を受けない。

【００６１】

図９から判るように、コードＣの最初の部分はべき等性を有し、割り込みの後で再スタートできる。固定ルーチンが最初に実行されていることを条件に、第２部分は再スタート可能であり、このことは追加の制御パラメータをセットすることによって表示される。コードの最終部分は再スタートできないので、このコードを処理する前に割り込みがディスエーブルされる。

【００６２】

図２０は別の実施例と異なる別の例を示し、この場合、モニタモード中に割り込みがイネーブルされる。モニタモードで実行中の機能はこれら機能を安全に再スタートできなくなると直ちに、これら割り込みをディスエーブルするように働く。このことはモニタモードで割り込みされたすべての機能が再開されるのではなく、再スタートされる場合に限り可能である。

【００６３】

割り込み時に、所定のモードで実行中のすべての機能を再開するのではなく再スタートすることを保証できるようにするいくつかの方法がある。１つの方法として割り込みが割り込まれた命令のアドレスではなく、命令シーケンスのスタートのアドレスをセーブする新しいプロセッサ状態を加えることが挙げられる。この場合、モニタモードは常にこのステートで実行される。別の方法は、各機能のスタート時に機能のスタートのアドレスを割り込み例外レジスタへあらかじめロードし、割り込みの後のプロセッサのステートの割り込み例外レジスタへの書き込みをディスエーブルすることである。

【００６４】

図２０に示された実施例では、割り込み機能の終了直後に機能の再スタートを行ってもよいし、また機能を安全に再スタートできるようにすることが求められている場合には、固定ルーチンの後で再スタートを行ってもよい。

【００６５】

以上で安全および非安全ドメイン、ならびにモニタモードを有するシステムを参照して、割り込み待ち時間を取り扱う上記方法について説明したが、特定の理由により再開すべきでない機能を有するシステムにも明らかに適用できる。一般に、かかる機能は割り込み待ち時間を増すような割り込みをディスエーブルすることによって作動する。再スタート可能なように機能を変え、割り込みの後でこれら機能を再スタートするようにプロセッサを制御することによって、機能処理の少なくとも一部の間で割り込みをイネーブルすることが可能となり、かつ割り込みの待ち時間を小さくすることに役立つ。例えば作動システムの正常なコンテクストの切り替えに役立つ。

【００６６】

安全および非安全メモリへのアクセス

図１を参照して説明するように、データ処理装置はメモリを有する。このメモリは特にＴＣＭ３６、キャッシュ３８、ＲＯＭ４４、スレーブデバイスのメモリおよび外部メモリ

５６を含む。例として図３７を参照して説明するように、メモリは安全メモリと非安全なメモリとに区分されている。一般には製造時にはメモリの安全メモリ領域と非安全なメモリ領域との物理的な区別はないが、これら領域は安全ドメインで作動する際のデータ処理装置の安全なオペレーティングシステムによって定義されることが理解できよう。従って、メモリデバイスの任意の物理的部分を安全メモリとして割り当て、別の物理的部分を非安全メモリとして割り当てることができる。

【００６７】

図２～５を参照して説明するように、処理システムは安全ドメインと非安全ドメインとを有する。安全ドメインには安全なカーネルプログラム８０が設けられ、このプログラムは安全モードで実行される。安全ドメインと非安全ドメインとをまたがるモニタプログラム７２も設けられ、このプログラムは少なくとも部分的にモニタモードで実行される。本発明の実施例では、モニタプログラムは一部はモニタモードで実行され、一部は安全モードで実行される。例えば図１２０に示されるように、特にスーパーバイザーモードＳＶＣを含む複数の安全モードがある。

【００６８】

モニタプログラム７２は安全ドメインと非安全ドメイン間のいずれの方向への変更を管理する役割を果たす。「プロセッサモード」の章において、図８および９を参照して、これら機能の一部について説明する。モニタプログラムは前記非安全モードから前記安全モードへの切り替えを開始するために、非安全モードで発生されるモード切り替えリクエストＳＭＩおよび前記安全モードから前記非安全モードへの切り替えを開始するために、安全モードで発生されるモード切り替えリクエストＳＭＩに対する責任がある。「世界間の切り替え」という章で説明するようにモニタモードでは安全および非安全ドメインの１つから別のドメインへのレジスタの少なくとも一部の切り替えが行われる。この切り替えでは一方のドメイン内に存在するレジスタのステートがセーブされ、他方のドメインにあるレジスタへ新しいステートの書き込み（またはレジスタ内に前にセーブされたステートのリストア）が行われる。本明細書で説明したように、かかる切り替えの実行時にあるレジスタへのアクセスをディスエーブルしてもよい。モニタモードではすべての割り込みをディスエーブルすることが望ましい。

【００６９】

モニタプログラムの実行が安全ドメインと非安全ドメインにまたがっているモニタモードでは、モニタプログラムが検証可能に安全であること、すなわちモニタプログラムが実現しようとしている機能だけを実現することが重要である。従って、モニタプログラムをできるだけ簡単にすることが有利となる。安全モードは安全ドメインでしか処理を実行できないようにできる。本発明の本実施例では、特権安全モード（単数または複数）およびモニタモードは同じ安全メモリおよび非安全メモリへのアクセスを認めている。特権安全モードが同じ安全メモリおよび非安全メモリを「見る」ことを保証することによって、そうしない場合にモニタモードでしか実現できない機能を安全モードに移し、モニタプログラムを簡略化できる。更に、これによって特権安全モードで作動するプロセスを直接モニタモードに切り替え、この逆への切り替えを行うことができる。特権安全モードからモニタモードへの切り替えが認められ、モニタモードでは非安全ドメインへの切り替えを行うことができる。非特権安全モードはＳＭＩを使ってモニタモードに入らなければならない。システムはリセットの後で特権安全モードに入る。ドメイン間を移動する際にステートのセーブを容易にするためにモニタモードと特権安全モードとの切り替えが行われる。

【００７０】

別の実施例では、安全特権モード内からもモニタモード内からもＳフラグへのアクセスが認められる。プログラムフローの制御を維持する間、安全特権モードがプロセッサをモニタモードに切り替えることが認められる場合、かかる安全特権モードは既にＳフラグ（ビット）を変える能力を有効に有している。従って、Ｓフラグをモニタモード内でしか変更できないことによる別の複雑性は正当化されない。その代わりに１つ以上の安全特権モードによって変更できる別のコンフィギュレーションフラグと同じようにＳフラグを記憶

できる。現在の技術には１つ以上の安全特権モード内でＳフラグを変更できるかかる実施例が含まれる。

【００７１】

前に説明した実施例に戻ると、本装置はモードを定め、モードの特権レベル、すなわちモードが可能にする機能の組を定めるプロセッサコア１０を含む。従って、プロセッサコア１０は安全モードおよびモニタモードによる安全および非安全メモリへのアクセス、およびモニタモードがアクセスを認めるすべてのメモリへの安全モードによるアクセスを許可し、かつ特権安全モードで作動中のプロセスが直接モニタモードに切り替わり、かつこの逆の切り替えを許可するように、公知の態様で配置されている。プロセッサコア１０は次のことを認めるようになっていることが好ましい。

【００７２】

装置の一例では、メモリは安全メモリと非安全メモリとに区分され、安全メモリと非安全メモリとの双方はモニタモードおよび安全モードでしかアクセスできない。非安全メモリはモニタモード、安全モードおよび非安全モードでアクセスできることが好ましい。

【００７３】

本装置の別の例では、モニタモードおよび１つ以上の安全モードにおいて、安全モードに対して非安全メモリへのアクセスが否定され、非安全モードにおいて、安全モードおよびモニタモードに対し非安全メモリへのアクセスが否定される。従って、安全メモリはモニタモードおよび安全モードでしかアクセスされず、非安全メモリは非安全モードでしかアクセスされないので、セキュリティが高まる。

【００７４】

本装置の例では、安全モードである特権モードよりも特権の大きいモードとして見なすことができるモニタモードでは、装置のリセットまたはブートを行うことができる。しかしながら、本装置の多くの例は、安全モードとモニタモードとの間で認められる直接切り替えにより可能な安全モードにおいてリセットまたはブートをするようになっている。

【００７５】

図２を参照して説明したように、安全ドメインおよび安全モードでは、安全カーネル８０（またはオペレーティングシステム）が機能し、安全カーネル８０のもとで１つ以上の安全アプリケーションプログラム８２、８４を実行できる。安全モードで作動する安全カーネルおよび／または安全アプリケーションプログラム、または他のプログラムコードは、安全メモリおよび非安全メモリの双方へのアクセスが認められる。

【００７６】

以上で、プロセッサを有する装置を参照して本発明の例について説明したが、適当なプロセッサで実行される際に、この章で説明したように作動するようプロセッサを構成するコンピュータプログラムによって本発明を実施してもよい。

【００７７】

プログラマーのモデル図から検討した本技術の別の実施例について、次のように図２１～２３を参照して説明する。

【００７８】

次の説明では、英国ケンブリッジのＡＲＭリミティッド社によって設計されたＡＲＭプロセッサに関連して理解しなければならない次の用語を使用することにする。

－Ｓビット：　専用ＣＰ１５レジスタ内に含まれる安全ステートビット

－「安全／非安全ステート」：　このステートはＳビットの値によって定められ、コアが安全な世界にアクセスできる（安全ステートのときにはＳ＝１）かどうかを表示するか、または非安全な世界だけに制限される（Ｓ＝０）。モニタモード（更に参照）はＳビットのステータスを無効にする。

－「非安全な世界」：　セキュリティを必要としない非安全アプリケーションへアクセスできるすべてのハードウェア／ソフトウェアのグループのことである。

－「安全な世界」：　安全コードを実行しているときにしかアクセスできないすべてのハードウェア／ソフトウェア（コア、メモリ．．．）のグループのことである。

－モニタモード：　安全ステートと非安全ステートとの間でコアを切り替える役割を果たす新しいモードである。

【００７９】

簡単な要約

－コアは常に非安全な世界にアクセスできる。

－コアは安全な世界が安全ステートまたはモニタモードとなっているときにしか、安全な世界にアクセスできない。

－ＳＭＩ：　ソフトウェアモニタ割り込みのことであり、専用ＳＭＩ例外ベクトルを通してコアをモニタモードにさせる新しい命令である。

－「スレッドＩＤ」：　（ＯＳによって制御される）各スレッドに関連する識別子である。ＯＳが非安全な世界で実行されるあるタイプのＯＳに対しては、安全機能が呼び出される度に、パラメータとして現在のスレッドＩＤをパスし、呼び出している非安全アプリケーションに安全機能をリンクさせなければならない。従って、安全な世界は多数のスレッドをサポートできる。

－安全割り込みは安全周辺機器が発生する割り込みを定める。

【００８０】

プログラマーのモデル

カーボンコアの概観

本技術を使用するプロセッサのために本明細書で使用する用語である、カーボンアーキテクチャなる概念は、安全な世界と非安全な世界の２つに分離することから成る。安全な世界はどんなデータも非安全な世界にリークしてはならないようになっている。

【００８１】

提案する解決案では、安全ステートと非安全ステートは同じ（現行の）レジスタバンクを共用する。結果としてＡＲＭコアに存在する現在のすべてのモード（Abort、Undef、Irq、User、．．．．）は各ステートで存在することになる。

【００８２】

コアは専用ＣＰ１５レジスタで示される新しいステートビット、すなわちＳ（安全）ビットにより安全ステートで作動するのか非安全ステートで作動するのかを知る。

【００８３】

Ｓビットを変えること、すなわちあるステートから別のステートへ変更することをどの命令またはイベントに許可するかを制御することは、このシステムのセキュリティのキーとなる特徴である。現在の解決案は新しいモード、すなわち２つのステートの切り替えをスーパーバイズするモニタモードを加えることを提案するものである。適当なＣＰ１５レジスタに書き込むことにより、モニタモードはＳビットが変更することが認められる唯一のモードとなる。

【００８４】

最後に、例外ハンドリングに対するある程度のフレキシビリティを加えることを提案する。リセットとは別にして、すべての例外はそれが生じたステートにおいて取り扱われるか、またはモニタモードに向けられる。このことは専用ＣＰ１５レジスタを構成可能な状態にしておくことを必要とする。

【００８５】

次のパラグラフで、この解決案の詳細について検討する。

プロセッサのステートおよびモード

カーボンアーキテクチャの新しい特徴

安全または非安全ステート（Ｓビット）

カーボンコアの主な特徴は、コアが安全ステート（Ｓ＝１）または非安全ステート（Ｓ＝０）にあるかどうかを表示するＳビットが存在していることである。安全ステートにあるとき、コアは安全な世界または非安全な世界内のどのデータにもアクセスできる。非安全なステートにあるとき、コアは非安全な世界へのアクセスだけに限定される。

【００８６】

この規則への唯一の例外は、Ｓビット情報を無効にするモニタモードに関連している。Ｓ＝０であってもコアがモニタモードにあると、コアは安全特権アクセスを実行する。更に情報を望む場合には次のパラグラフであるモニタモードを参照されたい。

【００８７】

モニタモードでのみＳビットを読み出し／書き込みできる。Ｓビットの値がどんな値であっても、他のモードがこれにアクセスしようとする場合、これを無視するか、または結果としてUndef（未定義）例外が生じる。

【００８８】

リセットを除くすべての例外は安全ステートビットに影響しない、リセット時にＳビットがセットされ、スーパーバイザーモードでコアがスタートする。更に詳細な情報を望む場合にはブートの章を参照されたい。

【００８９】

安全／非安全ステートは分離しており、ＡＲＭ／Ｔｈｕｍｂ／Ｊａｖａ（Ｒ）ステートと独立に作動する。

【００９０】

モニタモード

カーボンシステムの別の重要な特徴は、新しいモード、すなわちモニタモードを形成することである。このモードは安全ステートと非安全ステートとの間のコアの切り替えを制御するのに使用される。このモードは常に安全モードと見なされる。すなわちＳビットの値がどんな値であっても、コアがモニタモードにあると、外部世界への安全特権アクセスを常に実行する。

【００９１】

任意の安全特権モード（すなわちＳ＝１のときの特権モード）はＣＰＳＲモードビット（ＭＳＲ、ＭＯＶＳまたは等価的命令）を書き込むだけで、モニタモードに切り替わることができる。しかしながら、このことは非安全モードまたは安全ユーザーモードでは禁止される。このことが起きた場合、命令を無視するか、または例外を生じさせる。

【００９２】

専用ＣＰＳＲ違反例外に対するニーズがあり得る。非安全モードまたは安全ユーザーモードからＣＰＳＲを直接書き込むことにより、モニタモードへ切り替える試みによってこの例外が生じる。

【００９３】

モニタモードがアクティブであるときにリセットを除くすべての例外は実際にはディスエーブルされる。

・すべての割り込みをマスクする。

・すべてのメモリ例外は無視されるか、または致命的例外を生じさせる。

・無定義／ＳＷＩ／ＳＭＩは無視されるか、または致命的例外を生じさせる。

【００９４】

モニタモードに入ると、システムモニタの作動中に他のタイプの例外が生じないよう、割り込みは自動的にディスエーブルされ、システムモニタが書かれる。

【００９５】

モニタモードはプライベートレジスタを有しなければならない。この解決案は、レジスタの最初の組、すなわちＲ１３（ｓｐ＿ｍｏｎ）、Ｒ１４（ｌｒ＿ｍｏｎ）およびＳＰＳＲ（ｓｐｓｒ＿ｍｏｎ）しかコピーしないことを提案するものである。

【００９６】

モニタモードでは、ＭＰＵまたはパーティションチェッカー（モニタモードは常に安全特権外部アクセスを実行する）と同様、ＭＭＵもディスエーブルする（フラットアドレスマップ）。しかしながら、特別にプログラムされたＭＰＵ領域の属性（キャッシュ可能性、．．．）はアクティブなままとなる。モニタモードが使用できる別の例として、安全ドメインによってどんなマッピングも使用できる。

【００９７】

新しい命令

この提案は現在のＡＲＭ命令セットに１つの新しい命令を加えることを求める。

モニタモードに入れ、固定ＳＭＩ例ベクトルで分岐するＳＭＩ（ソフトウェアモニタ割り込み）命令が使用される。この命令は主に非安全ステートと安全ステートとの間のスワップをするためのモニタに対する表示をするのに使用される。

【００９８】

別の例として（またはそれに加えて）、モニタモードがモニタスタックからの／モニタスタックへの他のモードのステートのセーブ／リストアを可能にし、コンテクストの切り替え性能を改善するための新しい命令を加えることができる。

【００９９】

プロセッサモード

前のパラグラフで説明したように、コアには新しい１つのモード、すなわちモニタモードしか追加されない。現在のすべてのモードは利用できる状態のままであり、安全ステートおよび非安全ステートの双方で存在する。

実際にはカーボンユーザーは図２１に示される構造を見ることになる。

【０１００】

プロセッサのレジスタ

この実施例は、安全な世界と非安全な世界とが同じレジスタバンクを共用することを提案するものである。このことは、ある世界からモニタモードを通して別の世界に切り替わるときに、システムモニタは第１の世界のコンテクストをセーブし、第２の世界のコンテクストを発生（またはリストア）する必要があることを意味する。

【０１０１】

パラメータをパスすることは容易なタスクとなる。システムがＳビットを一旦切り替えると、第１の世界におけるレジスタに含まれるデータは第２の世界内の同じレジスタによって利用できるようになる。

【０１０２】

しかしながら、厳密に制御する必要がある、パラメータをパスさせるための専用の限られた数のレジスタとは別に、安全ステートから非安全ステートへパスする際に他のすべてのレジスタをフラッシュさせ、安全データがリークする危険性を防止する必要がある。このことは、モニタカーネルによって保証しなければならない。

【０１０３】

安全ステートから非安全ステートへの切り替え時にレジスタを直接フラッシュさせるハードウェア機構または新しい命令を実現する可能性も１つの可能性である。

【０１０４】

提案する別の解決案は安全ステートと非安全ステートとの間で２つの物理的に分離されたレジスタバンクを有する現在のレジスタバンクのすべて（またはほとんど）を複製することである。この解決案はレジスタに含まれる安全データと非安全データとを明瞭に分離できるという主な利点を有する。更に安全ステートと非安ステートとの間で高速コンテクスト切り替えも可能にできる。しかしながら、欠点として、非安全レジスタにアクセスできるようにする幾つかの専用の命令を作成しなければ、パラメータがレジスタを通過するようにさせることが困難となることが挙げられる。

【０１０５】

図２２はプロセッサモードに応じた利用可能なレジスタを示す。ここでプロセッサのステートがこの主題には影響しないことに留意されたい。

例外

安全な割り込み

現在の解決案

現在のコア内に、例えばＩＲＱおよびＦＩＱと同じ割り込みピンを持たせることが現在提案されている。（本明細書で後に記載する）例外トラップマスクレジスタに関連し、異なる種類の割り込みを実現し、取り扱うシステムのために、フレキシビリティを十分にす

る必要がある。

【０１０６】

ＶＩＣ増強

次のように、ＶＩＣ（ベクトル化された割り込みコントローラ）を増強することができる。このＶＩＣはベクトル化された各アドレスに関連した１つの安全情報ビットを含むことができる。このビットはモニタまたは安全特権モードでしかプログラムできない。このことは、検討する割り込みを安全として処理し、よって安全サイドで取り扱うべきか否かを示している。

【０１０７】

２つの新しいベクトルアドレスレジスタも増設し得る。一方のレジスタは非安全ステートで生じるすべての安全割り込みのためのものであり、他方は安全ステートで生じるすべての非安全割り込みに対するものである。

ＣＰ１５に含まれるＳビット情報は新しいＶＩＣ入力としてＶＣＩにも利用できる。

【０１０８】

次の表は入進割り込みのステータス（各割り込みラインに関連する、Ｓビットにより表示される安全ステートまたは非安全ステート）およびコアのステート（ＣＰ１５内のＳビット＝ＶＣＩでのＳ入力信号）に応じた異なる可能なシナリオを要約したものである。

【０１０９】

【表２】

	安全状態にあるコア（ＣＰ１５－Ｓ＝１）	非安全状態にあるコア（ＣＰ１５－Ｓ＝０）
安全な割り込み	世界を切り替える必要はない。ＶＩＣは割り込みラインに関連した安全なアドレスを直接コアに示す。コアは関連するＩＳＲを発見するであろうこのアドレスで分岐するだけでよい。	ＶＩＣは非安全ドメイン内のこの割り込みに関連したベクトルを有しない。従ってＶＩＣは非安全な世界で生じるすべての安全割り込みに対して専用のベクトルアドレスレジスタに含まれるアドレスをコアに示す。次にまだ非安全な世界内にあるコアはこのアドレスに分岐し、このアドレスで安全な世界へ切り替えるためのＳＭＩ命令を発見することとなる。一旦安全な世界内に入れば、正しいＩＳＲにアクセスできる。
非安全な割り込み	ＶＩＣは安全ドメイン内にこの割り込みに関連したベクトルを有しない。従って、ＶＩＣは安全な世界内で生じるすべての非安全な割り込みに対して専用のベクトルアドレスレジスタに含まれるアドレスをコアに示す。次に、まだ安全な世界にあるコアはこのアドレスに分岐し、ここでＳＭＩ命令が見い出され非安全な世界に切り替えられる。一旦非安全な世界内に入ると、正しいＩＳＲにアクセスできる。	世界を切り替える必要はない。ＶＩＣは割り込みラインに関連した非安全アドレスをコアに直接示す。コアはこのアドレスに分岐し関連する非安全ＩＳＲを見い出す。

【０１１０】

例外取り扱いコンフィギュラビリティ

カーボンフレキシビリティを改善するためにＣＰ１５では新しいレジスタ、例外トラップマスクが追加される。このレジスタは次のビットを含むことになる。

- ビット０：Ｕｎｄｅｆ（未定義）例外　（非安全ステート）
- ビット１：ＳＷＩ例外　（非安全ステート）
- ビット２：プリフェッチアボート例外　（非安全ステート）
- ビット３：データアボート例外　（非安全ステート）
- ビット４：ＩＲＱ例外　（非安全ステート）
- ビット５：ＦＩＱ例外　（非安全ステート）
- ビット６：ＳＭＩ例外　（非安全ステートと安全ステートの双方）
- ビット１６：未定義例外　（安全ステート）

－　ビット１７：ＳＷＩ例外　（安全ステート）
－　ビット１８：プリフェッチアボート例　（安全ステート）
－　ビット１９：データアボート例外　（安全ステート）
－　ビット２０：ＩＲＱ例外　（安全ステート）
－　ビット２１：ＦＩＱ例外　（安全ステート）

【０１１１】

リセット例外はこのレジスタ内に対応するビットを有しない。リセットによって常にコアはその専用ベクトルを通して安全スーパーバイザーモードに入る。

ビットがセットされると、対応する例外によってコアはモニタモードとなる。そうでない場合、例外が生じた世界内の対応するハンドラー内で例外が処理される。

レジスタはモニタモードでしか見ることができない。その他のモードではこのレジスタにアクセスしようとする命令は無視されることになる。

システムがモニタをサポートしているか否かに応じて、このレジスタはシステム固有の値に初期化しなければならない。この機能はＶＩＣによって制御できる。

【０１１２】

例外ベクトルテーブル

別個の安全な世界と非安全な世界とが設けられるので、別個の安全例外ベクトルの表と非安全例外ベクトルのテーブルを必要とする。

更にモニタがある例外もトラップできるので、我々はモニタ専用の第３の例外ベクトルテーブルも必要とする。次のテーブルは３つの異なる例外ベクトルのテーブルを要約するものである。

【０１１３】

【表３】

非安全メモリにおいて：

ｱﾄﾞﾚｽ	例外	モード	自動的にアクセスされる時
0x0	-	-	
0x04	Undef（未定義）	Undef（未定義）	コアが非安全ステートであり、例外トラップマスクレジスタ［非安全Undef］=0のときに、未定義命令が実行された
0x08	SWI	スーパーバイザー	コアが非安全ステートであり、例外トラップマスクレジスタ［非安全SWI］=0のときに、SWI命令が実行された
0x0C	プリフェッチアボート	アボート	コアが非安全ステートで例外トラップマスクレジスタ［非安全PAbort］=0のときに、命令がアボートされた
0x10	データアボート	アボート	コアが非安全ステートで例外トラップマスクレジスタ［非安全DAbort］=0のときに、データがアボートされた
0x14	保留		
0x18	IRQ	IRQ	コアが非安全ステートで例外トラップマスクレジスタ［非安全IRQ］=0のときに、IRQピンがアサートされた
0x1C	FIQ	FIQ	コアが非安全ステートで例外トラップマスクレジスタ［非安全FIQ］=0のときに、FIQピンがアサートされた

【０１１４】

【表４】

安全メモリにおいて：

ｱﾄﾞﾚｽ	例外	モード	自動的にアクセスされる時
0x00	Reset*	スーパーバイザー	リセットピンがアサートされた
0x04	Undef（未定義）	Undef（未定義）	コアが安全ステートであり、例外トラップマスクレジスタ［安全Undef］=0のときに、未定義命令が実行された
0x08	SWI	スーパーバイザー	コアが安全ステートであり、例外トラップマスクレジスタ［安全SWI］=0のときに、SWI命令が実行された
0x0C	プリフェッチアボート	アボート	コアが安全ステートであり、例外トラップマスジスタ［安全PAbort］=0のときに、命令がアボートされた
0x10	データアボート	アボート	コアが安全ステートで例外トラップマスクレジスタ［安全DAbort］=0のときに、データがアボートされた
0x14	保留		
0x18	IRQ	IRQ	コアが安全ステートで例外トラップマスクレジスタ［安全IRQ］=0のときに、IRQピンがアサートされた
0x1C	FIQ	FIQ	コアが安全ステートで例外トラップマスクレジスタ［安全FIQ］=0のときに、FIQピンがアサートされた

＊　リセット機構に関する更なる説明は、「ブート」の章を参照。

【０１１５】

【表５】

モニタメモリ（フラットマッピング）：

ｱﾄﾞﾚｽ	例外	モード	自動的にアクセスされる時
0x00	–	–	–
0x04	未定義	モニター	コアが安全ステートであり、例外トラップマスクレジスタ［安全Undef］=1のとき、およびコアが非安全ステートであり、例外トラップマスクレジスタ［非安全Undef］=1のときに、未定義命令が実行された
0x08	SWI	モニター	コアが安全ステートであり、例外トラップマスクレジスタ［安全SWI］=1のとき、およびコアが非安全ステートであり、例外トラップマスクレジスタ［非安全SWI］=1のときに、SWI命令が実行された
0x0C	プリフェッチアボート	モニター	コアが安全ステートであり、例外トラップマスクレジスタ［安全IAbort］=1のとき、およびコアが非安全ステートであり、例外トラップマスクレジスタ［非安全IAbort］=1のときに、命令がアボートされた
0x10	データアボート	モニター	コアが安全ステートであり、例外トラップマスクレジスタ［安全PAbort］=1のとき、およびコアが非安全ステートであり、例外トラップマスクレジスタ［非安全PAbort］=1のときに、データがアボートされた
0x14	SMI	モニター	
0x18	IRQ	モニター	コアが安全ステートであり、例外トラップマスクレジスタ［安全IRQ］=1のとき、およびコアが非安全ステートであり、例外トラップマスクレジスタ［非安全IRQ］=1のときに、IRQピンがアサートされた
0x1C	FIQ	モニター	コアが安全ステートであり、例外トラップマスクレジスタ［安全FIQ］=1のとき、およびコアが非安全ステートであり、例外トラップマスクレジスタ［非安全FIQ］=1のときに、FIQピンがアサートされた

【０１１６】

モニタモードでは各例外が２つの異なる関連ベクトルを有するように、例外ベクトルを複製してもよい。

－　非安全ステートで生じる例外用のベクトル

－　安全ステートで生じる例外用のベクトル

モニタカーネルは例外が生じた元のステートをそれ以上検出する必要はないので、このことは例外の時間待ちを少なくする上で有効である。

この特徴は数個の例外に制限されることに注意する必要がある。ＳＭＩは安全ステートと非安全ステートとの切り替えを採用する最も適当な候補の１つである。

【０１１７】

世界間の切り替え

ステートを切り替える時、モニタモードはそのモニタスタックに最初のステートのコン

テクストをセーブし、モニタスタックから第２のステートのコンテクストをリストアしなければならない。
従って、モニタモードはプライベートレジスタ（ｒ１４ＳＰＳＲ、．．．．）を含む他のモードのレジスタにアクセスしなければならない。
これを処理するために、提案される解決案はＳＰＳＲを書き込むだけで直接モニタモードに切り替える権利を安全ステートにある特権モードに与えることである。
かかるシステムでは、次のように世界間の切り替えが実行される。
－　モニタモードに入る。
－　Ｓビットをセットする。
－　スーパーモードに切り替え－モニタスタックにスーパーバイザーレジスタをセーブする（当然ながらスーパーバイザーモードはモニタスタックポインタにアクセスしなければならないが、これは例えば共通レジスタ（Ｒ０～Ｒ８）を使用することにより容易に実行できる）。
－　システムモードへ切り替え－モニタスタックに（ユーザーモードと同じ）レジスタをセーブする。
－　すべてのモードに対し、モニタスタックにＩＲＱレジスタをセーブする。
－　すべてのモードのすべてのプライベートレジスタがセーブされると、簡単なＭＳＲ命令（＝ＣＰＳＲモードフィールドにモニタの値を単に書き込む）によりモニタモードへ戻る。
次のその他の解決案を検討する。
－　モニタが自身のスタックに他のモードのプライベートレジスタをセーブできるようにする新しい命令を追加する。
－　例えば（適当なアクセス権を有するために）モニタステートとなり、ＩＲＱ（または他の）プライベートレジスタを見るためにＩＲＱ（または他の）モードとなることができる新しい「ステート」としてモニタを実現する。
【０１１８】
基本的なシナリオ（図２３）
１．スレッド１が非安全な世界（Ｓビット＝０）で作動中である。
このスレッドは安全な機能、＝＞ＳＭＩ命令を実行しなければならない。
２．このＳＭＩ命令は非安全ＳＭＩベクトルによりコアをモニタモードにする。
ＬＲ＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎを使用して非安全モードのＰＣおよびＣＰＳＲをセーブする。
システムは安全ステートであるが、この段階ではＳビットは変化しないままである。
モニタカーネルがモニタに非安全コンテクストをセーブする。モニタはＬＲ＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎもプッシュする。
次にモニタカーネルはＣＰ１５レジスタに書き込みをすることにより、「Ｓ」ビットを変える。
この実施例では、モニタカーネルは（例えばスレッドＩＤテーブルを更新することにより）安全な世界で「安全スレッド１」をスタートさせる道を維持する。
最後に、モニタモードから出て安全スーパーバイザーモードに切り替わる（ＬＲ＿ｍｏｎおよびＳＰＳＲ＿ｍｏｎを更新した後のＭＯＶＳ命令）。
３．安全カーネルが正しい安全メモリのロケーションへアプリケーションを送り（例えばＭＯＶＳを使って）、ユーザーモードに切り替わる。
４．安全ユーザーモードで安全機能が実行される。一旦終了すると安全機能は適当なＳＷＩを実行することにより「退出」機能を呼び出しする。
５．ＳＷＩ命令は専用ＳＷＩベクトルによりコアを安全ｓｖｃモードにし、次にコアは「退出」機能を実行する。この「退出」機能はモニタモードに戻すよう切り替えるための「ＳＭＩ」により終了する。
６．ＳＭＩ命令は専用安全ＳＭＩベクトルによりコアをモニタモードにする。
安全ＳＶＣモードのＰＣおよびＣＰＳＲをセーブするのにＬＲ＿ｍｏｎおよびＳＰＳＲ

＿ｍｏｎを使用する。
Ｓビットは変わらないままである（例えば安全ステート）。
安全なスレッド１が終了した事実をモニタカーネルが登録する（スレッドＩＤテーブルから安全なスレッド１のＩＤを除く）。
モニタカーネルがＣＰ１５レジスタに書き込むことにより「Ｓ」ビットを変え、非安全ステートに戻る。
モニタカーネルがモニタスタックから非安全コンテクストをリストアする。モニタカーネルはステップ２で前にセーブされたＬＲ＿ｍｏｎおよびＣＰＳＲ＿ｍｏｎもロードする。
最後に、モニタカーネルはＳＵＢＳによりモニタモードから出る。このＳＵＢＳは命令時に非安全ユーザーモードにコアを戻す。
７．スレッド１を正常に再開できる。

【０１１９】
図６を参照する。安全ドメインと非安全ドメインの間ですべてのレジスタを共用する。モニタモードでは安全および非安全ドメインの一方から他方のドメインへレジスタを切り替える切り替えが行われる。この切り替えでは、上記「世界間の切り替え」の章でも説明したように、あるドメインで存在するレジスタのステートをセーブし、他のドメインにあるレジスタに新しいステートを書き込む（またはレジスタに前にセーブされたステートをリストアする）。

【０１２０】
この切り替えを実行するのにかかる時間を短縮するのが好ましい。切り替えにかかる時間を短縮するために安全ドメインと非安全ドメインとの間の切り替え時に共用レジスタをディスエーブルし、内部に記憶されているデータの値を変えないように維持する。例えば非安全ドメインから安全ドメインへの切り替えを検討する。例えば図６に示されたＦＩＱレジスタは安全な世界では必要でないと仮定する。従って、これらレジスタはディスエーブルされ、これらレジスタを安全ドメインに切り替え、これらレジスタの内容をセーブする必要はない。

【０１２１】
レジスタのディスエーブル化はいくつかの方法で行うことができる。１つの方法はこれらレジスタを使用するモードをロックすることである。この方法は、モードのディスエーブルを指示するＣＰ１５レジスタ内の制御ビットを書き込むことによって行われる。
これとは異なり、再びＣＰ１５レジスタ内の制御ビットを書き込むことにより、命令に基づいて命令によるレジスタへのアクセスをディスエーブルしてもよい。ＣＰ１５レジスタに書き込まれるビットはモードではなくレジスタに明らかに関係しているので、モードはディスエーブルされず、このモードにおけるレジスタへのアクセスがディスエーブルされる。

【０１２２】
ＦＩＱレジスタは高速割り込みに関連するデータを記憶する。ＦＩＱレジスタ（単数または複数）がディスエーブルされ、高速割り込みが生じた場合、プロセッサはモニタ内で例外の信号を出す。この例外に応答し、モニタモードは、あるドメインに関連し、前記ディスエーブルされたレジスタに記憶されているデータ値をセーブし、かつ他のドメインに関連する新しいデータ値をそのレジスタにロードし、ＦＩＱモードレジスタを再イネーブルするようになっている。
モニタモードにおいてプロセッサがドメインを切り替える際にすべてのバンク状レジスタがディスエーブルされるようにプロセッサを構成してもよい。これとは異なり、ドメインを切り替える際に共用レジスタの所定の一部がディスエーブルされ、他のレジスタをプログラマーの選択でディスエーブルできるように、レジスタのディスエーブル化を選択的にしてもよい。

【０１２３】
モニタモードでドメインを切り替える際に共用レジスタの１つ以上をディスエーブルし

、共用レジスタの１つ以上の他のレジスタはあるドメインから出る際にセーブされたデータを有し、他のドメインにロードされる新しいデータを有するようにプロセッサを構成してもよい。この新しいデータをヌルデータとしてもよい。

【０１２４】

図２４は従来のＡＲＭコアに安全処理オプションを追加する原理を略図で示すものである。この図は現在のコアに安全処理オプションを追加することによって、安全処理オプションを含むプロセッサをどのように形成できるかを略図で示している。システムを既存の従来オペレーティングシステムと後方コンパーチブルにすべき場合、プロセッサの従来の非安全部分で作動する従来システムを考えることが直感的である。しかしながら、図の下方部分に略図に示され、後に詳細にされるように、実際には従来システムはシステムの安全部分で作動する。

【０１２５】

図２５は安全および非安全ドメインを有し、リセットを示すプロセッサを示し、図２５は図２に類似している。図２は安全ドメインにおける処理を制御する安全ＯＳシステムおよび非安全ドメインにおける処理を制御する非安全ＯＳシステムによるセキュリティに応答するタイプのオペレーションを実行するようになっているプロセッサを示す。しかしながら、このプロセッサは伝統的な従来オペレーティングシステムと後方コンパーチブルでもあり、従ってプロセッサは従来オペレーティングシステムを使用してセキュリティに影響されないように作動できる。

【０１２６】

図２５に示されるように、リセットは安全ドメイン内であり、Ｓビット即ちセキュリティステータスフラグのセットと共にどんなタイプのオペレーションリセットも生じる。セキュリティに影響されないタイプのオペレーションの場合、安全ドメイン内でリセットが生じ、安全ドメイン内で処理が進む。しかしながら、処理を制御する従来のオペレーティングシステムはシステムのセキュリティの特徴については知らない。

図２５に示されるように、処理をセキュリティに影響されるようにすべきか、または実際にセキュリティに影響されないようにすべきかに応じて、安全スーパーバイザーモード内の処理をスタートするアドレスを設定するようにリセットが実行される。一旦リセットが実行された場合、ブート機構またはリブート機構に存在する別のタスクが実行される。このブート機構について以下に説明する。

【０１２７】

ブート機構

ブート機構は次の特徴を考慮しなければならない。

－　従来のＯＳとのコンパチビリティを維持すること

－　ほとんどの特権モードにおいてシステムのセキュリティを保証するようにブートすること。

結果として安全スーパーバイザーモードでカーボンコアがブートする。

別のシステムは次のようになる。

－　従来のＯＳを実行するシステムに対してはＳビットは考慮されず、コアはスーパーバイザーモードでブートするように見せる。

－　カーボンの特徴を利用するシステムに対しては、コアは（潜在的にはモニタモードにスワップした後に）システム内のすべての安全保護を構成できなければならない安全特権モードでブートする。

【０１２８】

上記ブート機構の詳細について、本発明の実施例のプロセッサはいずれのケースにおいても安全スーパーバイザーモードにおける処理をスタートさせるようにプロセッサをリセットする。セキュリティに影響されないタイプのオペレーションの場合、Ｓビットがセットされる（しかしながらオペレーティングシステムはこのことは知らない）ので、セキュリティはここでは問題とならないが、オペレーティングシステムは実際には安全ドメインで作動する。このことには、非安全ドメインからアクセスできないメモリ部分にこの状況

でアクセスできるという利点がある。

【０１２９】

すべてのケースにおける安全スーパーバイザーモードでのブートは、システムのセキュリティを保証するのに役立つので、セキュリティに影響されるシステムでは有利である。安全に影響されるシステムでは、安全スーパーバイザーモードでブートプログラムが記憶される場所をブート時に提供されるアドレスがポイントするので、このアドレスはシステムを安全システムとして構成し、モニタモードへの切り替えを可能にする。安全スーパーバイザーモードからモニタモードへの切り替えは一般に許可され、これによって適当な時間における安全システムがモニタモードでの処理を開始するのを可能にし、モニタモードのコンフィギュレーションを初期化できる。

【０１３０】

図２６は、ステップ１で、非安全オペレーティングシステムにより非安全スレッドＮＳＡが実行されることを示している。ステップ２において、非安全スレッドＮＳＡはステップ３でモニタモードプログラムを実行するモニタモードを通して安全ドメインを呼び出す。モニタモードプログラムはドメインを切り替えるためにＳビットを変え、ステップ５で安全オペレーティングシステムに移る前に必要なコンテクストのセーブおよびコンテクストのリストアを実行する。次に、ステップ６で割り込みＩＲＱを受ける前に対応する安全スレッドＳＡが実行される。この割り込み取り扱いハードウェアはステップ７におけるモニタモードへのリターンをトリガーし、このステップ７では割り込みを安全オペレーティングシステムによって取り扱うのか、非安全オペレーティングシステムによって取り扱うのかが判断される。この場合、割り込みはステップ９でスタートする非安全オペレーティングシステムによって取り扱われる。この割り込みが非安全オペレーティングシステムによって取り扱われた場合、ステップ１１にて正常なスレッド切り替え動作の前に非安全スレッドＮＳＡは非安全オペレーティングシステムにおける現在のタスクとして再開される。このスレッドの切り替えはタイミングイベントまたは同様なことの結果となり得る。ステップ１２において、非安全オペレーティングシステムにより非安全ドメインで異なるスレッドＮＳＡが実行され、次にステップ１４にてモニタドメイン／プログラムにより安全ドメインへの呼び出しがなされる。ステップ７は安全スレッドが実行を終了したため、または通常のリクエストが残されたことに起因して残されるというよりも、割り込みの結果として安全オペレーティングシステムが最後にサスペンドされた旨を表示するための、別の機構で使用されたフラグを記憶する。従って、安全オペレーティングシステムは割り込みによってサスペンドされるので、ステップ１０におけるモニタプログラムはリターンスレッドＩＤ（例えば非安全スレッドＮＳＢだけでなく他のパラメータデータによってリクエストされる安全オペレーティングシステムによってスタートすべきスレッドの識別子）を指定する、ソフトウェア疑似割り込みを使って安全オペレーティングシステムを再入力する。ソフトウェア疑似割り込みのこれらパラメータはレジスタの値としてパスできる。

【０１３１】

ステップ１５において、ソフトウェア疑似割り込みはステップ１５における安全オペレーティングシステムのリターン割り込みハンドラールーチンをトリガーする。このリターン割り込みハンドラールーチンはソフトウェア疑似割り込みのリターンスレッドＩＤを検査し、このＩＤがサスペンションの前に安全オペレーティングシステムが実行された最後のときに割り込まれた安全スレッドＳＡのスレッドＩＤに一致するかどうかを判断する。この場合、一致がないので、安全スレッドＳＡのコンテクストがセーブされた後で非安全スレッドＮＳＢによって指定されたリターンスレッドへのスレッドの切り替えを実行するようステップ１６で安全オペレーティングシステムがトリガーされる。この安全スレッドＳＡは、必要な場合に割り込みがなされたポイントから後で再スタートできる。

【０１３２】

図２７は図２６に示されたタイプの挙動の別の例を略図で示す。この例ではｉｒｑを取り扱うために非安全オペレーティングシステムの制御により処理が進行するが、非安全スレッドの切り替えはないので、安全オペレーティングシステムのリターン割り込みハンド

ラーによってソフトウェア疑似割り込みが受信されると、リターン割り込みハンドラーはスレッド切り替えが不要であると判断し、ステップ１５で安全スレッドＳＡを単に再開するだけである。

【０１３３】

図２８は、リターンスレッドハンドラーによって実行される処理を略図で示すフローチャートである。ステップ４００２において、リターンスレッドハンドラーがスタートされる。ステップ４００４において、ソフトウェア疑似割り込みからのリターンスレッド識別子が検査され、安全オペレーティングシステムがサスペンドされたときに現在実行中の安全スレッドと比較される。これらが一致した場合、処理はステップ４００６に進み、このステップで安全スレッドが再開される。ステップ４００４における規格が一致しなければ、処理はステップ４００８に進み、ステップ４０１０で新しい安全スレッドへの切り替えが行われる前に（その後の再開のために）ステップ４００８にて旧い安全スレッドのコンテクストがセーブされる。この新しいスレッドは既に途中となり得るので、ステップ４０１０は再開となる。

【０１３４】

図２９は、スレーブ安全オペレーティングシステムがマスター非安全オペレーティングシステムにより実行されるタスク切り替えに従うことができるようにする処理を略図で示す。マスター非安全オペレーティングシステムは他のオペレーティングシステムにその活動を伝え、コーディネートするための機構を有しない従来のオペレーティングシステムでよいので、マスターとしてしか作動しない。図２９における最初の入力ポイントのように、非安全オペレーティングシステムは非安全スレッドＮＳＡを実行する。この非安全スレッドＮＳＡはソフトウェアの割り込みを使用する安全オペレーティングシステムにより実行すべき安全スレッドへの呼び出し、ＳＭＩ呼び出しを行う。このＳＭＩ呼び出しはステップ２でモニタモードで実行されるモニタプログラムに進み、ステップ４にて安全オペレーティングシステムに呼び出しを送る前にステップ２にてモニタプログラムは必要なコンテクストセービングおよび切り替えを実行する。次に安全オペレーティングシステムは対応する安全スレッドＳＡをスタートさせる。この安全スレッドはタイマーイベントなどの結果として、モニタモードを介して非安全オペレーティングシステムへ制御をリターンしてもよい。ステップ９にて非安全スレッドＮＳＡが再び制御を安全オペレーティングシステムへ送る際に、このＮＳＡは元のオリジナル割り込みを再発行することによってこれを行う。ソフトウェア割り込みがＮＳＡを識別する非安全スレッドＩＤ、起動すべき目標安全スレッドのスレッドＩＤ、すなわち安全スレッドＳＡを識別するスレッドＩＤだけでなく、他のパラメータも含む。

【０１３５】

ステップ９で発生された呼び出しがモニタプログラムによってパスされ、安全オペレーティングシステムにより安全ドメイン内でステップ１２で受信されると、オペレーティングシステムによりコンテクスト切り替えがあったかどうかを判断するように、非安全スレッドＩＤを検査できる。目標スレッドの安全スレッドＩＤも検査し、安全オペレーティングシステム下の正しいスレッドが新しいスレッドとして再スタートまたはスタートされたかどうかを見ることもできる。図２９の例では、安全ドメインで安全オペレーティングシステムによりスレッド切り替えは必要とされない。

【０１３６】

図３０は図２９に類似するが、非安全オペレーティングシステムの制御により非安全ドメイン内でステップ９でスレッドの切り替えが行われる点で異なっている。従って、ステップ１１で安全オペレーティングシステムにソフトウェア割り込み呼び出しを行うのは別の非安全スレッドＮＳＢである。ステップ１４において、オペレーティングシステムは非安全スレッドＮＳＢの別のスレッドＩＤを認識し、従って、安全スレッドＳＡのコンテクストをセーブし、安全スレッドＳＢをスタートさせるタスク切り替えを実行する。

【０１３７】

図３１は安全オペレーティングシステムのスレッドをスタートさせるか、または再開さ

せるための呼び出しとして、ソフトウェア割り込みを受信したときに安全オペレーティングシステムが実行する処理を略図で示すフローチャートである。ステップ４０１０にて呼び出しが受信される。ステップ４０１４において、パラメータが安全オペレーティングシステムでの現在アクティブな安全スレッドと一致しているかどうかを判断するために呼び出しのパラメータが検査される。一致していれば、ステップ４０１６にてこの安全スレッドが再スタートされる。一致していなければ、処理はステップ４０１８へ進み、ここで新しくリクエストされたスレッドが利用できるかどうかの判断をされる。新しくリクエストされたスレッドが安全オペレーティングシステムで実行されている他のあるスレッドによって既に使用されている排他的リソースであるか、またはこのような排他的リソースを必要とするような理由により、利用できない場合がある。かかるケースでは、ステップ４０２０にて呼び出しが拒否され、非安全オペレーティングシステムへ適当なメッセージがリターンされる。ステップ４０１８において、新しいスレッドが利用できると判断された場合、処理はステップ４０２２へ進み、このステップにて旧い安全スレッドのコンテクストが後に起こり得る再開のためにセーブされる。

ステップ４０２４にて、安全オペレーティングシステムに対して行われたソフトウェア割り込み呼び出しで指定された新しい安全スレッドへの切り替えが行われる。

【０１３８】

図３２は異なる割り込みが異なるオペレーティングシステムにより処理される多数のオペレーティングシステムを有するシステム内で割り込みを取り扱う際に、優先権の反転ができるようにする作動を略図で示す。

【０１３９】

安全オペレーティングシステムが安全スレッドＳＡを実行する状態で処理がスタートする。この処理は第１の割り込みＩｎｔ１によって割り込みがされる。これによりモニタモード内のモニタプログラムがトリガーされ、割り込みを安全ドメインまたは非安全ドメインで処理するのか否かが判断される。この場合、割り込み安全ドメイン内で処理すべきであり、処理は安全オペレーティングシステムに戻り、割り込みＩｎｔ１のための割り込み取り扱いルーチンがスタートされる。Ｉｎｔ１のための割り込み取り扱いルーチンの実行の途中において、より高い優先権を有する別の割り込みＩｎｔ２が受信される。従って、Ｉｎｔ１のための割り込みハンドラーが停止され、割り込みＩｎｔ２を取り扱うかどうかの判断をするために、モニタモードのモニタプログラムが使用される。この場合、割り込みＩｎｔ２は非安全オペレーティングシステムにより処理すべきであるので、制御は非安全オペレーティングシステムに移り、Ｉｎｔ２のための割り込みハンドラーがスタートされる。割り込みＩｎｔ２のためのこの割り込みハンドラーが完了したときに、非安全オペレーティングシステムは安全ドメイン内でサービスがサスペンドされたペンディング中の割り込みＩｎｔ１があることを示す情報を有していない。従って、非安全オペレーティングシステムはある別の処理、例えば別の非安全スレッドＮＳＢのタスクの切り替えまたはスタートを実行できるが、元の割り込みＩｎｔ１はサービスを受けないままである。

【０１４０】

図３３は図３２の動作に関連した問題を解消できるようにした技術を示す。割り込みＩｎｔ１が生じると、モニタプログラムはこの割り込みを非安全ドメインへ送り、このドメインでスタブ割り込みハンドラーが開始される。このスタブ割り込みハンドラーは比較的小さく、モニタモードを介して安全ドメインへ迅速に処理を戻し、安全ドメイン内で割り込みＩｎｔ１のための割り込みハンドラーをトリガーする。この割り込みＩｎｔ１は主に安全ドメイン内で処理され、安全ドメイン内で割り込みがペンディング中であることを非安全ドメインに示すための、あるタイプのプレイスホルダーとして、非安全ドメインにおけるスタブ割り込みハンドラーの開始を見なすことができる。

【０１４１】

割り込みＩｎｔ１のための安全ドメイン内の割り込みハンドラーは再び高い優先権のＩｎｔ２を受ける。これによって前と同じように、非安全ドメインにおいてＩｎｔ２のための割り込みハンドラーの実行がトリガーされる。しかしながらこの場合、Ｉｎｔ２のため

の割り込みが終了したときに、非安全オペレーティングシステムは割り込みＩｎｔ２のためのスタブ割り込みハンドラーがまだ未処理であることを示すデータを有しているので、このスタブ割り込みハンドラーを再開する。このスタブ割り込みハンドラーは安全ドメインへの呼び出しバックが行われたポイントで、あたかもサスペンドされているように見えるので、この呼び出しが再実行され、従って、安全ドメインへの切り替えが行われる。一旦安全ドメインへ戻ると、安全ドメイン自身は割り込みのＩｎｔ１がサスペンドされたポイントで割り込みＩｎｔ１のための割り込みハンドラーを見ずから再スタートできる。安全ドメイン内で割り込みＩｎｔ１のための割り込みハンドラーが完了すると、非安全ドメインへの呼び出しバックが行われ、最初に実行中の安全スレッドＳＡが再開される前に、非安全ドメインにおいてスタブ割り込みハンドラーをクローズする。

【０１４２】

図３４は関連する優先権を有する異なるタイプの割り込みおよびこれら割り込みをどのように取り扱うことができるかを略図で示している。純粋に安全なドメインの割り込みハンドラーを使って高い優先権の割り込みを処理できるが、これを行うには非安全ドメインによって取り扱われるそれよりも高い優先権の割り込みがないことが条件となる。非安全ドメインで取り扱われ、その後の割り込みよりも高い優先権を有する割り込みがあると、それよりも低いすべての割り込みは純粋に非安全ドメイン内で取り扱うか、または図３３に示されるスタブ割り込みハンドラー技術を利用し、安全ドメイン内で主要取り扱いが生じても非安全ドメインがこれら割り込みを追跡できるようにしなければならない。

【０１４３】

前に述べたように、安全ドメインと非安全ドメインとの切り替えを実行するのにモニタモードが使用される。２つの異なるドメインの間でレジスタを共用する実施例では、このような共用を行うにはレジスタ内のステートをメモリにセーブし、メモリからそれらレジスタに宛て先ドメインのための新しいステートをロードする。２つのドメイン間で共用されないレジスタに対してはステートをセーブする必要はない。その理由は、これらレジスタは他のドメインによりアクセスされることがなく、ステート間の切り替えは安全ドメインと非安全ドメインとの切り替えの直接の結果として実現される（すなわちＣＰ１５レジスタのうちの１つに記憶されているＳビットの値が、共用されないレジスタのどれを使用するかを判断する）からである。

【０１４４】

モニタモード中に切り替える必要のあるステートの部分は、プロセッサによるメモリへのアクセスを制御するプロセッサコンフィギュレーションデータである。各ドメイン内には異なる表示のメモリがあるので、例えば安全データを記憶するための安全メモリにアクセスする安全ドメインがあり、非安全ドメインからこの安全メモリへはアクセスできないので、ドメインの切り替えを行う際にはプロセッサコンフィギュレーションデータを変える必要が生じることが明らかである。

【０１４５】

図３５に示されるように、ＣＰ１５レジスタ３５内にはこのプロセッサコンフィギュレーションデータが記憶されており、一実施例ではドメイン間でこれらレジスタが共用される。従って、安全ドメインと非安全ドメインとの間でモニタモードが切り替えられるとき、そのときにＣＰ１５レジスタ３４内にあるプロセッサコンフィギュレーションデータはＣＰ１５レジスタからメモリへ切り替える必要があり、宛て先ドメインに関連するプロセッサコンフィギュレーションデータをＣＰ１５レジスタ３４にロードする必要がある。

【０１４６】

ＣＰ１５レジスタ内にあるプロセッサコンフィギュレーションデータは一般に、システム内のメモリへのアクセスに直接的な効果を有するので、モニタモードで作動している間にこれら設定をプロセッサが更新する際に、これら設定は即座に有効となることは明らかである。しかしながら、このことは望ましいことではない。その理由は、モニタモード中にメモリのアクセスを制御するプロセッサコンフィギュレーションデータのスタティックな組をモニタモードが有することが望ましいからである。

【０１４７】
従って図３５に示されるように、本発明の一実施例では、プロセッサがモニタモードで作動している間にＣＰ１５レジスタ３４内のプロセッサコンフィギュレーションデータを無効にするのに使用できる、モニタモード固有のプロセッサコンフィギュレーションデータ２０００が設けられている。図３５に示された実施例では、このことはマルチプレクサ２０１０を設けることによって達成され、マルチプレクサはＣＰ１５レジスタ内に記憶されたプロセッサコンフィギュレーションデータとモニタモード固有のプロセッサコンフィギュレーションデータ２０００の双方を入力端で受信する。更にこのマルチプレクサ２０１０はプロセッサがそのときにモニタモードで作動しているのか否かを表示する制御信号をパス２０１５として受信する。プロセッサがモニタモードで作動していない場合、ＣＰ１５レジスタ３４内のプロセッサコンフィギュレーションデータがシステムに出力されるが、プロセッサがモニタモードで作動している場合、その代わりにマルチプレクサ２０１０はモニタモード固有のプロセッサコンフィギュレーションデータ２０００を出力し、プロセッサがモニタモードで作動している間、一貫した組のプロセッサコンフィギュレーションデータが加えられることを保証する。
【０１４８】
このモニタモード固有のプロセッサコンフィギュレーションデータをシステム内でハードコピーし、よってデータを処理できないようにすることができる。しかしながら、プロセッサが安全特権モードで作動しているときにモニタモード固有のプロセッサコンフィギュレーションデータは、プロセッサによってしか変えられないことを条件にセキュリティと妥協することなく、モニタモード固有のプロセッサコンフィギュレーションデータ２０００をプログラム可能にすることができる。このようにすることによってモニタモード固有のプロセッサコンフィギュレーションデータの設定に関し、若干のフレキシビリティが認められる。モニタモード固有のプロセッサコンフィギュレーションデータがプログラムできるようになっている場合、このコンフィギュレーションデータをシステム内の適当な場所、例えばＣＰ１５レジスタ３４内のレジスタの別個の組内に記憶することができる。
【０１４９】
一般に、このモニタモード固有のプロセッサコンフィギュレーションデータはモニタモードでのプロセッサの作動に対し、極めて安全な環境を提供するように設定される。従って、上記実施例ではモニタモード固有のプロセッサコンフィギュレーションデータはプロセッサがモニタモードで作動中にメモリ管理ユニット３０がディスエーブルすることを指定し、よって仮想アドレス－物理アドレス変換をディスエーブルする。ディスエーブルしなかった場合、この変換はメモリ管理ユニットにより実施される。かかる状況ではプロセッサはメモリアクセスリクエストを発生する際に常に物理アドレスを直接発生するようになっている。すなわちフラットなマッピングが使用される。これによって仮想アドレス－物理アドレスマッピングに対して不正な操作が行われたかどうかにかかわらず、プロセッサはモニタモードの作動中に確実にメモリにアクセスできる。
【０１５０】
モニタモード固有のプロセッサコンフィギュレーションデータはプロセッサがモニタモードで作動中にプロセッサが安全データにアクセスできるようにすることも一般に指定する。このことは、ドメインステータスビット状をしたメモリ許可データによって指定することが好ましく、このドメインステータスビットは完全プロセッサコンフィギュレーションデータ内の対応するドメインステータスビット（Ｓビット）に対して指定されるのと同じ値を有する。従ってＣＰ１５レジスタ内に記憶されるドメインステータスビットの実際の値にかかわらず、この値はポインタモードが安全データにアクセスするのを保証するよう、ポインタモード固有のプロセッサコンフィギュレーションデータによって指定されるドメインステータスビットによって無効状態となる。
【０１５１】
モニタモード固有のプロセッサコンフィギュレーションデータはメモリの部品へのアクセスを制御するのに使用される他のデータも指定できる。例えばモニタモード固有のプロ

セッサコンフィギュレーションデータはプロセッサがモニタモードで作動中にデータにアクセスするのにキャッシュ３８を使用すべきでない旨を指定できる。

【０１５２】

上記実施例ではプロセッサコンフィギュレーションデータを含むＣＰ１５レジスタのすべてが、ドメインの間で共用されると仮定されている。しかしながら別の実施例では、例えばプロセッサコンフィギュレーションデータの特定のアイテムを記憶するのに２つのレジスタが設けられるように、多数のＣＰ１５レジスタがバンク状となっている。すなわち１つのレジスタが非安全ドメインでアクセス可能であり、非安全ドメインのためのプロセッサコンフィギュレーションデータのそのアイテムの値を含み、他方のレジスタが安全ドメインでアクセス可能であり、安全ドメインのためのプロセッサコンフィギュレーションデータの値を含むようになっている。

【０１５３】

バンク状ではない１つのＣＰ１５レジスタはＳビットを含むレジスタであるが、基本的には所望すれば他方のＣＰ１５レジスタのいずれもバンク状にすることができる。かかる実施例ではモニタモードによるプロセッサコンフィギュレーションデータの切り替えは共用ＣＰ１５レジスタがその共用されているレジスタに現在記憶されているプロセッサコンフィギュレーションデータをメモリに切り替え、宛て先ドメインに関連するプロセッサコンフィギュレーションデータを共用されているＣＰ１５レジスタにロードすることを行う。バンク状レジスタに対し、プロセッサコンフィギュレーションデータはメモリに記憶させる必要はなく、むしろ対応する共用されているＣＰ１５レジスタに記憶されているＳビットの値を変えた結果として自動的に切り替えが行われる。

【０１５４】

前に述べたように、モニタモードプロセッサコンフィギュレーションデータは対応するＣＰ１５レジスタに記憶されたデータを無効にするドメインステータスビットを含むが、安全ドメインで指定されたドメインステータスビットに対して使用された値と同じ値（すなわち上記実施例では１のＳビット値）を有する。多数のＣＰ１５レジスタがバンク状となっているとき、このことは図３５内のモニタモード固有のプロセッサコンフィギュレーションデータ２０００の少なくとも一部をバンク状レジスタ内に記憶されている安全プロセッサコンフィギュレーションデータから発生できることを意味する。その理由は、このレジスタの内容は切り替え中にメモリへ書き込まれることがないからである。

【０１５５】

従って、一例としてモニタモード固有のプロセッサコンフィギュレーションデータは無効にすべきドメインステータスビットを指定し、無効にしない場合、このビットはモニタモードではないときに使用され、一実施例ではこのビットは安全ドメインで使用された値と同じ値を有するので、このことはバンク状ＣＰ１５レジスタのうちのいずれがアクセス可能であるかを選択する論理回路により、安全なバンク状ＣＰ１５レジスタにアクセスできるようになることを意味する。モニタモード固有のプロセッサコンフィギュレーションデータの対応する部分として、モニタモードがこの安全プロセッサコンフィギュレーションデータを使用できることにより、リソースでのセーブを実現できる。その理由は、モニタモード固有のプロセッサコンフィギュレーションデータのアイテムに対してレジスタの別個の組を設ける必要がないからである。

【０１５６】

図３６は一方のドメインと他方のドメインとの間の切り替えが必要なときの、プロセッサコンフィギュレーションデータを切り替えるのに実行されるステップを示すフローチャートである。前に述べたように、ドメイン間の切り替えを誘導するのに、ＳＭＩ命令が発生される。従って、ステップ２０２０ではＳＭＩ命令の発生が待たれる。ＳＭＩ命令が受信されると、プロセッサはステップ２０３０に進み、このステップでプロセッサはモニタモードでのモニタプログラムの実行を開始する。これによってモニタモード固有のプロセッサコンフィギュレーションデータはマルチプレクサ２０１０へのパス２０１５上の制御回路の結果として使用され、マルチプレクサはそのモニタモード固有のプロセッサコンフ

ィギュレーションデータを切り替える。前に述べたように、このデータはデータのうちの自ら含まれる組でもよいし、またデータの所定の部分はバンク状レジスタに記憶された安全プロセッサコンフィギュレーションデータから発生されたものでもよい。

【０１５７】

その後、ステップ２０４０にてメモリへＳＭＩ命令を発生するドメインからそのときのステートがセーブされる。このセーブはそのドメインに対応するプロセッサコンフィギュレーションデータのステートを共用されているＣＰ１５レジスタからセーブすることを含む。一般に、かかるステートを記憶するのに、別にメモリの組の一部が設けられる。次に、ステップ２０５０にて、ステートポインタは宛て先ドメインのための対応するステートを含むメモリ部分へのポイントへ切り替えられる。従って、一般にステート情報を記憶するのにメモリの２つの部分が割り当てられる。すなわち非安全ドメインのためのステートを記憶するのに１つの部分が割り当てられ、安全ドメインに対するステートを記憶するのに別の部分が割り当てられる。

【０１５８】

ステップ２０５０にて、ステートポインタが一旦切り替えられると、ステートポインタによってポイントされていたステートがステップ２０６０で対応する共用ＣＰ１５レジスタへロードされる。このローディングでは宛て先ドメインのための対応するプロセッサコンフィギュレーションデータのロードが行われる。その後、ステップ２０７０にてモニタモードのようにモニタプログラムが附勢され、次にプロセッサは宛て先ドメインにおける必要なモードへ切り替わる。

【０１５９】

図３７は、本発明の一実施例のメモリ管理ロジック３０の作動をより詳細に示す。このメモリ管理ロジックはメモリ管理ユニット（ＭＭＵ）２００と、メモリ保護ユニット（ＭＰＵ）２２０とから成る。仮想アドレスを指定するコア１０が発生するメモリアクセスリクエストはパス２３４を通してＭＭＵ２００へ送られる。このＭＭＵ２００は所定のアクセス制御機能、特に仮想アドレスに対応する物理アドレスの決定およびアクセス許可権を分析したり、領域の属性を決定したりする役割を果たす。

【０１６０】

データ処理装置のメモリシステムは安全メモリと非安全メモリとから成り、安全メモリはコア１０または他のデバイスが安全作動モードで作動し、従って安全ドメインで作動しているときに、コア１０または１つ以上のマスターデバイスしかアクセスできないようになっている安全データを記憶するのに使用される。

【０１６１】

図３７に示された本発明の実施例では、非安全モードでコア１０上で実行中のアプリケーションにより、安全メモリ内の安全データへアクセスしようとする試みの規制がＭＰＵ２２０内のパーティションチェッカー２２２によって実行され、このＭＰＵ２２０は本明細書で安全カーネルとも称される安全オペレーティングシステムによって管理されている。

【０１６２】

本発明の好ましい実施例によれば、非安全メモリ、例えば外部メモリ５６の非安全メモリ部分内に非安全ページテーブル５８が設けられ、このテーブルはそのページテーブル内に定められた多数の非安全メモリ領域の各々に対して、対応するデスクリプタを記憶するのに使用される。このデスクリプタは情報を含み、この情報からＭＭＵ２００はこのＭＭＵが所定のアクセス制御機能を実行できるようにするのに必要なアクセス制御情報を発生でき、従って、図３７を参照して説明した実施例では、仮想アドレス－物理アドレスマッピング、アクセス許可権および任意の領域属性に関する情報を提供する。

【０１６３】

更に本発明の好ましい実施例によれば、メモリ領域のうちの安全メモリ内、例えば外部メモリ５６の安全部分内に少なくとも１つの安全ページテーブル５８が設けられ、このテーブルはこのテーブル内に定められた多数のメモリ領域に対して再び関連するデスクリプ

タを提供する。プロセッサが非安全モードで作動しているときに、メモリアクセスを管理するのに使用するための対応するデスクリプタを得るために、非安全ページテーブルが参照されるが、プロセッサが安全モードで作動しているときには安全ページテーブルからのデスクリプタが使用されることになる。

【０１６４】

対応するページテーブルからＭＭＵへのデスクリプタの検索は次のように進む。コア１０が発生したメモリアクセスが仮想アドレスを指定した場合、多数の仮想アドレス部分のうちの１つに対し、対応するページテーブルから得られた対応する物理アドレス部分を記憶するマイクロＴＬＢ２０６内でルックアップが実行される。従って、マイクロＴＬＢ２０６は仮想アドレスの所定部分とマイクロＴＬＢ内に記憶された対応する仮想アドレス部分とを比較し、一致があるかどうかを判断する。比較された部分は一般に仮想アドレスの所定の数の最大桁ビットとなる。このビット数はページテーブル５８内のページの粒度に応じて決まる。マイクロＴＬＢ２０６は比較的少数のエントリー、例えば８個のエントリーしか含まないので、マイクロＴＬＢ２０６内で実行されるルックアップは一般に比較的短時間である。

【０１６５】

マイクロＴＬＢ２０６内で一致が見いだされない場合、ページテーブルから得られた多数のデスクリプタを含む主ＴＬＢ２０８へパス２４２を通してメモリアクセスリクエストが送られる。後により詳細に説明するように、主ＴＬＢ２０８内には非安全ページテーブルと安全ページテーブルの双方からのデスクリプタが共存することができ、主ＴＬＢ内の各エントリーはそのエントリー内の対応するデスクリプタが安全ページテーブルから得られたものか、または非安全ページテーブルから得られたものかどうかを表示するように設定できる対応するフラグ（本明細書ではドメインフラグと称す）を有する。すべての安全作動モードがメモリアクセスリクエスト内に直接物理アドレスを指定する実施例では、主ＴＬＢが非安全デスクリプタしか記憶しないので、主ＴＬＢにはかかるフラグに対する必要性がないことが理解できよう。

【０１６６】

主ＴＬＢ２０８内では、メモリアクセスリクエスト内で発生された仮想アドレスの対応する部分が特定の作動モードに対応する主ＴＬＢ２０８内のデスクリプタに関連する仮想アドレス部分のいずれかに対応するかどうかを判断するのに、同様なルックアッププロセスが実行される。従って、コア１０が非安全モードで作動している場合、非安全ページテーブルから得られた主ＴＬＢ２０８内のデスクリプタしかチェックされないが、一方、コア１０が安全モードで作動している場合、安全ページテーブルから得られた主ＴＬＢ内のデスクリプタしかチェックされない。

【０１６７】

前記チェックプロセスの結果として、主ＴＬＢ内で一致があった場合、対応するデスクリプタからアクセス制御情報が抽出され、パス２４２を通して送り戻される。特にデスクリプタの仮想アドレス部分および対応する物理アドレス部分がマイクロＴＬＢのエントリー内に記憶できるように、パス２４２を通してマイクロＴＬＢ２０６へルーティングされ、アクセス許可ロジック２０２へアクセス許可権がロードされ、領域属性ロジック２０４へ領域属性がロードされる。このアクセス許可ロード２０２および領域属性ロジック２０４はマイクロＴＬＢと別個のものでもよいし、またマイクロＴＬＢ内に内蔵されていてもよい。

【０１６８】

このポイントでは、マイクロＴＬＢ２０６内で一致が生じるので、ＭＭＵ２００はメモリアクセスリクエストを処理できる。従って、マイクロＴＬＢ２０６は物理アドレスを発生し、この物理アドレスはパス２０８を通して対応するメモリへルーティングするためのシステムバス４０へ出力できる。このメモリはオンチップメモリ、例えばＴＣＭ３６、キャッシュ３８などでもよいし、また外部バスインターフェース４２を介してアクセスできる外部メモリユニットの１つのいずれかである。同時に、アクセス許可ロジック２０２が

メモリアクセスを許可するかどうかを判断し、そのときの作動モードで特定のメモリロケーションへコアがアクセスするのを許可しないと判断した場合、パス２３０を通してコア１０へアボート信号を発生する。例えばスーパーバイザーモードで作動しているコアによってしかアクセスできないものとして、安全メモリ内または非安全メモリ内のメモリの所定部分を指定してもよい。従って、コア１０が、例えばユーザーモードのときにかかるメモリロケーションにアクセスしようとしている場合、アクセス許可ロジック２０２はコア１０がそのときに適当なアクセス権を有しないことを検出し、パス２３０を通してアボート信号を発生する。これによってメモリアクセスがアボートされる。最後に、領域属性ロジック２０４は特定のメモリアクセスに対する領域属性、例えばアクセスがキャッシュ可能であるか、バッファ化できるかどうかを判断し、パス２３２を通してかかる信号を発生し、ここでかかる信号を使ってメモリアクセスリクエストの主題であるデータを例えばキャッシュ３８内でキャッシュできるのか、書き込みアクセスの場合、書き込みデータをバッファ化できるのかどうかを判断する。

【０１６９】

主ＴＬＢ２０８内に一致がなかった場合、対応するページテーブル５８にアクセスするのに変換テーブルウォークロジック２１０が使用され、パス２４８を通して必要なデスクリプタを検索し、次にパス２４６を通して主ＴＬＢ２０８へデスクリプタを送り、ＴＬＢの内部に記憶できるようにする。ＣＰ１５　３４のレジスタには非安全ページテーブルと安全ページテーブルの双方のためのベースアドレスが記憶され、プロセッサコア１０が作動しているそのときのドメイン、すなわち安全ドメインまたは非安全ドメインもＣＰ１５のレジスタ内にセットされる。このドメインステータスレジスタは非安全ドメインと安全ドメインとの間で切り替えが行われたときにモニタモードでセットされる。ドメインステータスレジスタの内容を本明細書ではドメインビットと称する。従って、変換テーブルウォークプロセスを実行しなければならない場合、変換テーブルウォークロジック２１０はコア１０が実行しているのはどのドメインであるかについて知り、従って、対応するテーブルにアクセスするのにどのベースアドレスを使用するかについて知る。次に、適当なベーステーブル内の適当なエントリーにアクセスし、必要なデスクリプタを得るのに、ベースアドレスに対するオフセットとして仮想アドレスが使用される。

【０１７０】

変換テーブルウォークロジック２１０によって一旦デスクリプタが検索され、主ＴＬＢ２０８内に入れられると、主ＴＬＢ内で一致が得られ、アクセス制御情報を検索するのに、前に述べたプロセスが呼び出され、この情報がマイクロＴＬＢ２０６、アクセス許可ロジック２０２および領域属性ロジック２０４内に記憶される。次に、ＭＭＵ２００によってメモリアクセスを行うことができる。

【０１７１】

前に述べたように、好ましい実施例では主ＴＬＢ２０８は安全ページテーブルおよび非安全ページテーブルの双方からのデスクリプタを記憶できるが、マイクロＴＬＢ２０６内に一旦対応する情報が記憶されると、メモリアクセスリクエストはＭＭＵ２００によってしか処理されない。好ましい実施例では、主ＴＬＢ２０８とマイクロＴＬＢ２０６との間の情報の転送はＭＰＵ２２０内に設けられたパーティションチェッカー２２２によってモニタされ、コア１０が非安全モードで作動中の場合であって、主ＴＬＢ内のデスクリプタからマイクロＴＬＢ２０６へアクセス制御情報が転送された場合に、安全メモリ内にある物理アドレスが発生される場合、このようなアクセス制御情報が転送されないことを保証している。

【０１７２】

メモリ保護ユニットは安全オペレーティングシステムによって管理されており、このオペレーティングシステムは安全メモリと非安全メモリとの間のパーティションを定めるパーティション情報をＣＰ１５　３４のレジスタ内に設定できるようになっている。このパーティションチェッカー２２２はパーティション情報を参照し、非安全モードでコア１０による安全メモリへのアクセスを認めるマイクロＴＬＢ２０６へアクセス制御情報が転送

されているかどうかを判断することができる。より詳細には、好ましい実施例では、ＣＰ１５のドメインステータスレジスタ内でモニタモードによって設定されたドメインビットが表示するように、コア１０が非安全作動モードで作動しているときに、パーティションチェッカー２２２は主ＴＬＢ２０８からマイクロＴＬＢ２０６へ検索すべき物理アドレス部分を、パス２４２を通してモニタし、更にその物理アドレスに基づき、仮想アドレスに対して発生される物理アドレスが安全メモリ内にあるかどうかを判断するようになっている。かかる状況では、パーティションチェッカー２２２はパス２３０を通してコア１０に対してアボート信号を発生し、メモリアドレスが発生するのを防止する。

【０１７３】

更にパーティションチェッカー２２２がマイクロＴＬＢ２０６内に物理アドレス部分の記憶されるのを実際に防止するようにしてもよいし、これとは異なり、マイクロＴＬＢ２０６内に物理アドレス部分が記憶されるが、アボートプロセスの一部が例えばマイクロＴＬＢ２０６をフラッシングすることによってマイクロＴＬＢ２０６から正しくない物理アドレス部分を除くようにできることが理解できよう。

【０１７４】

コア１０がモニタモードを介して非安全モードと安全作動モードとの間で変化するときはいつも、モニタモードはプロセッサの動作を変えようとするドメインを表示するための、ＣＰ１５ドメインステータスレジスタ内のドメインビットの値を変える。ドメイン間の変換プロセスの一部としてマイクロＴＬＢ２０６はフラッシングされ、従って、安全ドメインと非安全ドメインとの間の切り替え後の最初のメモリアクセスはマイクロＴＬＢ２０６内で不一致を発生させ、主ＴＬＢ２０８から直接に、または対応するページテーブルからの対応するデスクリプタの検索により検索すべきアクセス情報を必要とする。

【０１７５】

上記方法により、コアが非安全ドメインで作動しているときに、安全メモリへのアクセスを可能にするマイクロＴＬＢ２０６のアクセス制御情報への検索の試みがなされた場合、メモリアクセスのアボートが達成されることが理解できよう。

【０１７６】

プロセッサコア１０の作動モードではメモリアクセスリクエストは直接物理アドレスを指定するようになっており、その作動モードではＭＭＵ２００がディスエーブルされ、パス２３６を通してＭＰＵ２２０へ物理アドレスが送られる。安全作動モードではアクセス許可ロジック２２４および領域属性ロジック２２６がＣＰ１５　３４内のパーティション情報レジスタ内の対応する領域に対して識別された領域属性およびアクセス許可権利に基づき、必要なアクセス許可および領域属性分析を実行する。アクセスしたい安全メモリロケーションが所定の作動モード、例えば安全特権モードでしかアクセスできない安全メモリ部分内にある場合、別の作動モード、例えば安全ユーザーモードにあるコアによるアクセスの試みがなされると、ＭＭＵのアクセス許可ロジック２０２がかかる状況でアボート信号を発生したのと同じように、アクセス許可ロジック２２４がパス２３０を通してコアへアボート信号を発生する。同様に、領域属性ロジック２２６は、ＭＭＵの領域属性ロジック２０４が仮想アドレスで指定されたメモリアクセスリクエストに対してかかる信号を発生したのと同じように、キャッシュ可能およびバッファ可能な信号を発生する。アクセスが認められていると仮定した場合、アクセスリクエストがパス２４０を通ってシステムバス４０に進み、このバスからリクエストは適当なメモリユニットへルーティングされる。

【０１７７】

アクセスリクエストが物理アドレスを指定している場合の非安全アクセスに対し、アクセスリクエストはパス２３６を介してパーティションチェッカー２２２へルーティングされ、パーティションチェッカー２２２はＣＰ１５レジスタ３４内のパーティション情報を参照してパーティションチェックを実行し、物理アドレスが安全メモリ内のロケーションを指定しているかを判断し、指定している場合に再びパス２３０を通してアボート信号が発生される。

【０１７８】
次に図３９および４０のフローチャートを参照し、メモリ管理ロジックの上記処理についてより詳細に説明する。図３９はステップ３００が示すように、コア１０で実行中のプログラムが仮想アドレスを発生する状況を示している。モニタモードによってセットされるＣＰ１５のドメインステータスレジスタ３４内の対応するドメインビットは、コアがそのときに安全ドメインで作動中か、または非安全ドメインで作動中かを表示する。コアが安全ドメインで作動している場合、プロセスはステップ３０２へ分岐し、このステップで仮想アドレスの対応する部分がマイクロＴＬＢ内の仮想アドレス部分の１つに一致するかどうかを見るために、マイクロＴＬＢ２０６内でルックアップを実行する。ステップ３０２で一致している場合、プロセスは直接ステップ３１２へ分岐し、このステップでアクセス許可ロジック２０２が必要なアクセス許可分析を実行する。ステップ３１４では、アクセス許可の違反があるかどうかが判断され、違反がある場合、プロセスはステップ３１６に進み、ここでアクセス許可ロジック２０２がパス２３０を通してアボート信号を発生する。一致がなければ、すなわちかかるアクセス許可違反がなければ、プロセスはステップ３１４からステップ３１８へ進み、このステップ３１８でメモリアクセスが進む。特に領域属性ロジック２０４はパス２３２を通して必要なキャッシュ可能かつバッファ可能な属性を出力し、マイクロＴＬＢ２０６は先に述べたようにパス２３８を通して物理アドレスを発生する。

【０１７９】
ステップ３０２においてマイクロＴＬＢにおいて不一致があった場合、必要な安全デスクリプタが主ＴＬＢ内にあるかどうかの判断をするために、ステップ３０４で主ＴＬＢ内でルックアッププロセスが実行される。デスクリプタがなければ、ステップ３０６でページテーブルウォークプロセスが実行され、よって変換テーブルウォークロジック２１０は図３７を参照して前に説明したように、安全ページテーブルから必要なデスクリプタを得る。次にこのプロセスはステップ３０８に進むか、または既に主ＴＬＢ２０８内に安全デスクリプタがあった場合、ステップ３０から直接ステップ３０８に進む。

【０１８０】
ステップ３０８にて主ＴＬＢがタグのついた有効な安全デスクリプタを含むと判断されると、プロセスはステップ３１０に進み、このステップで物理アドレス部分を含むデスクリプタのサブセクションがマイクロＴＬＢにロードされる。コア１０はそのときに安全モードで作動しているのでパーティションチェッカー２２２がパーティションチェック機能を実行する必要はない。

【０１８１】
次にプロセスはステップ３１２に進み、ここで前に述べたようにメモリアクセスの残りの部分が進行する。
非安全メモリアクセスの場合、ステップ３００からステップ３２０にプロセスが進み、ステップ３２０にて非安全デスクリプタからの対応する物理アドレス部分が存在するかどうかの判断をするために、マイクロＴＬＢ２０６内でルックアッププロセスが実行される。存在する場合、プロセスは直接ステップ３３６に進み、このステップでアクセス許可ロジック２０２によりアクセス許可権がチェックされる。この時点では、対応する物理アドレス部分がマイクロＴＬＢ内にある場合、セキュリティの違反がないと見なすと考えるのが重要である。その理由は情報がマイクロＴＬＢ内に記憶される前にパーティションチェッカー２２２が情報を有効に規制し、よって状況がマイクロＴＬＢ内にある場合、この情報は適当な非安全情報であると考えられるからである。ステップ３３６でアクセス許可がチェックされると、ステップ３３８へ進み、違反があるかどうかの判断がされ、そこでアクセス許可が違反となると、ステップ３１６でアボート信号が発生される。違反がなければステップ３１８に進み、ここで前に述べたようにメモリアクセスの残りの部分が実行される。

【０１８２】
ステップ３２０にてマイクロＴＬＢ内に一致がない場合、プロセスはステップ３２２へ

進み、ここで対応する非安全デスクリプタがあるかどうかの判断をするために、主ＴＬＢ２０８内でルックアッププロセスが実行される。デスクリプタがない場合、変換テーブルウォークロジック２１０によりステップ３２４でページテーブルウォークプロセスが実行され、非安全ページテーブルからの必要な非安全デスクリプタに関し、主ＴＬＢ２０８内の検索が行われる。次に、プロセスはステップ３２６に進むか、またはステップ３２２で主ＴＬＢ２０８内の一致が生じた場合、ステップ３２２からステップ３２６に直接進む。ステップ３２６にて、ＴＬＢが現在当該仮想アドレスに対する、タグの付いた有効な非安全デスクリプタを含むと判断され、次にステップ３２８にて（デスクリプタ内の物理アドレス部分とすると）メモリアクセスリクエストの仮想アドレスから発生された物理アドレスが、非安全メモリ内のロケーションをポイントするかどうかをパーティションチェッカー２２２がチェックする。ポイントしていない場合、すなわち物理アドレスが安全メモリ内のロケーションをポイントしている場合、ステップ３３０にてセキュリティの違反があったと判断され、プロセスはステップ３３２へ進み、パーティションチェッカー２２２により安全／非安全フォールトアボート信号が発生される。

【０１８３】

しかしながら、セキュリティの違反がないとパーティションチェッカー２２２が判断した場合、プロセスはステップ３３４に進み、このステップで物理アドレス部分を含む対応するデスクリプタのサブセクションがマイクロＴＬＢへロードされ、その後、ステップ３３６にて前に述べたようにメモリアクセスが処理される。

【０１８４】

次に図４０を参照し、物理アドレスを直接発生するメモリアクセスリクエストの取り扱いについて説明する。前に述べたように、このシナリオではＭＭＵ２００が除勢される。このことは、ＣＰ１５レジスタの対応するレジスタ内にＭＭＵイネーブルビットをセットすることによって行うことが好ましく、このセットプロセスはモニタモードで実行される。従って、ステップ３５０ではコアは物理アドレスを発生し、このアドレスはパス２３６を通ってＭＰＵ２２０へ送られる。次にステップ３５２にて、ＭＰＵはそのときの作動モード、例えばユーザー、スーパーバイザーなどのモードを仮定した場合、リクエストされたメモリアクセスが進行できることを証明するための許可をチェックする。更にコアが非安全モードで作動している場合、パーティションチェッカー２２２は非安全メモリ内に物理アドレスがあるかどうかもステップ３５２でチェックする。次にステップ３５４にて、違反があるかどうかの判断がされる。すなわちアクセス許可処理が違反を明らかにしたかどうか、または非安全モードにおいてパーティションチェックプロセスが違反を識別したかどうかを判断する。これら違反のいずれかが生じた場合、プロセスはステップ３５６に進み、ここでＭＰＵ２２０によりアクセス許可フォールトアボート信号が発生される。所定の実施例では、２つのタイプのアボートには区別はないが、別の実施例ではアボート信号はアクセス許可フォールトまたはセキュリティのフォールトに関係しているかどうかを表示できる。

【０１８５】

ステップ３５４にて違反が検出されない場合、プロセスはステップ３５８に進み、ここで物理アドレスによって識別されたロケーションへのメモリアクセスが行われる。

【０１８６】

好ましい実施例では、モニタモードしか物理アドレスを直接発生しないようになっているので、他のすべてのケースではＭＭＵ２００がアクティブとなり、前に述べたように、メモリアクセスリクエストの仮想アドレスからの物理アドレスの発生が生じる。

【０１８７】

図３８は、すべてのメモリアクセスリクエストが仮想アドレスを指定し、従って、いかなる作動モードでも直接物理アドレスが発生されない状況におけるメモリ管理ロジックの別の実施例も示す。このシナリオでは、別のＭＰＵ２２０は必要ではなく、その代わりにＭＭＵ２００内にパーティションチェッカー２２２を組み込むことができることが理解できよう。この変化を別にすれば、図３７および３９を参照して前に説明したのと全く同じ

ように処理が進む。

【０１８８】

他の種々のオプションも可能であることが理解できよう。例えば安全モードおよび非安全モードの双方により、仮想アドレスを指定するメモリアクセスリクエストを発生でき、２つのＭＭＵ（１つは安全アクセスリクエスト用であり、他方は非安全アクセスリクエスト用である）を設けることができると仮定した場合、図３７のＭＰＵ２２０を完全なＭＭＵに置き換えできる。かかるケースでは、一方のＭＭＵがその主ＴＬＢに非安全デスクリプタを記憶し、他方のＭＭＵがその主ＴＬＢに安全デスクリプタを記憶するので、デスクリプタが安全であるか非安全であるかを定めるための２つのフラグを、各ＭＭＵの主ＴＬＢと共に使用することは不要となる。当然ながら、コアが非安全ドメイン内にある間、安全メモリへのアクセスが試みられているかをチェックするのにパーティションチェッカーがまだ必要である。

【０１８９】

これとは異なり、すべてのメモリアクセスリクエストが直接物理アドレスを指定した場合、２つのＭＰＵ（一方は安全リクエスト用であり、他方は非安全リクエスト用である）を使用することを、別の実現例としてもよい。非安全アクセスリクエスト用に使用されるＭＰＵは安全メモリへのアクセスが非安全モードでは認められないようにすることを保証するように、パーティションチェッカーによって規制されたアクセスリクエストを有する。

【０１９０】

図３７または図３８の配置を設けることができる別の特徴として、パーティションチェッカー２２２があるパーティションチェックを実行し、変換テーブルウォークロジック２１０の活動を規制するようにできる。特にコアがそのときに非安全ドメインで作動している場合、変換テーブルウォークロジック２１０がページテーブルへのアクセスを求めているときはいつも、このロジックが安全ページテーブルではなく、非安全ページテーブルにアクセスしていることを、パーティションチェッカー２２２がチェックするようにできる。違反が検出された場合、アボート信号を発生することが望ましい。変換テーブルウォークロジック２１０はページテーブルベースアドレスとメモリアクセスリクエストが発生した仮想アドレスの所定ビットとを組み合わせることによって、一般にページテーブルルックアップを実行するので、このようなパーティションチェックは例えば屁何テーブルウォークロジック２１０が安全ページテーブルのベースアドレスではなく、非安全ページテーブルのベースアドレスを使用していることをチェックすることを必要とし得る。

【０１９１】

図４１は、コア１０が非安全モードで作動しているときにパーティションチェッカー２２２によって実行されるプロセスを略図で示す。通常の動作では、非安全ページテーブルから得られるデスクリプタは非安全メモリにマッピングされたページしか記述しないことが理解できよう。しかしながら、ソフトウェアによる攻撃があった場合、このデスクリプタがメモリの非安全領域を安全領域の双方を含むセクションを記述するように、デスクリプタを不正操作することができる。従って、図４１の例を検討すると、不正操作された非安全デスクリプタは非安全領域３７０、３７２、３７４、および安全領域３７６、３７８、３８０を含むページをカバーし得る。メモリアクセスリクエストの一部として発行された仮想アドレスは安全メモリ領域、例えば図４１に示されるような安全メモリ領域３７６内の物理アドレスに対応する場合、パーティションチェッカー２２２はアクセスが生じないようにするためのアボート信号を発生するようになっている。従って、安全メモリへアクセスしようとする試みにおいて、非安全デスクリプタが不正に書き換えられた場合でも、パーティションチェッカー２２２はアクセスが行われるのを防止できる。これと対照的に、このデスクリプタを使って誘導された物理アドレスが非安全メモリ領域、例えば図４１に示されるような領域３７４に対応する場合、マイクロＴＬＢ２０６にロードされたアクセス制御情報は単にこの非安全領域３７４を識別するだけである。従って、非安全メモリ領域３７４内のアクセスが生じることができるが、安全領域３７６、３７８または３８

０のいずれへのアクセスも生じることはできない。従って、主ＴＬＢ２０８が不正に書き換えされた非安全ページテーブルからのデスクリプタを含み得る場合であっても、マイクロＴＬＢは非安全メモリ領域にアクセスできることになる物理アドレス部分しか含まないことが理解できよう。

【０１９２】

前に述べたように、非安全モードと安全モードの双方が仮想アドレスを指定するメモリアクセスリクエストを発生できる実施例では、メモリは非安全メモリ内の非安全ページテーブルと安全メモリ内の安全ページテーブルとの双方を含むことが望ましい。非安全モードでは、変換テーブルウォークロジック２１０により非安全ページテーブルが参照されるが、他方、安全モードでは変換テーブルウォークロジック２１０により安全ページテーブルが参照される。図４２は、これら２つのページテーブルを示す。図４２に示されるように、例えば図１の外部メモリ５６内に設けることができる非安全メモリ３９０は、ベースアドレス３９４を参照することにより、ＣＰ１５レジスタ３４内に指定された非安全ページテーブル３９５を内部に含む。同じように、図１の外部メモリ５６内に設けることができる非安全メモリ４００内には、安全ページテーブルのベースアドレス４０７により複製ＣＰ１５レジスタ３４内で指定される対応する安全ページテーブル４０５が設けられる。非安全ページテーブル３９５内の各デスクリプタは、非安全メモリ３９０内の対応する非安全ページを指定するが、安全ページテーブル４０５内の各デスクリプタは安全メモリ４００内の対応する安全ページを定める。更に、後により詳細に説明するように、メモリの所定領域を共用メモリ領域４１０とし、これら領域に非安全モードと安全モードの双方によりアクセスできるようにすることができる。

【０１９３】

図４３は好ましい実施例に従い、主ＴＬＢ２０８内で実行されるルックアップ方法をより詳細に示す。前に述べたように、主ＴＬＢ２０８は対応するデスクリプタが安全ページテーブルからのものか、または非安全ページテーブルからのものかを識別するドメインフラグ４２５を含む。これによりルックアッププロセスを実行したときに、コア１０が作動中の特定のドメインに対応するデスクリプタしかチェックされないように保証できる。図４３は、安全な世界とも称される安全ドメインでコアが作動中の例を示す。図４３から理解できるように、主ＴＬＢ２０８のルックアップが実行されているときに、この結果、デスクリプタ４４０が無視され、デスクリプタ４４５だけがルックアッププロセスのための候補として識別される。

【０１９４】

好ましい実施例によれば、プロセス固有のページテーブルからのデスクリプタを識別するための、本明細書においてＡＳＩＤフラグとも称される別のプロセスＩＤフラグ４３０が設けられている。従って、プロセスＰ１、Ｐ２およびＰ３の各々はメモリ内に設けられた対応するページテーブルを有することができ、更に非安全動作および安全動作のために異なるページテーブルを有することができる。更に安全ドメインにおけるプロセスＰ１、Ｐ２、Ｐ３を非安全ドメインにおけるプロセスＰ１、Ｐ２、Ｐ３と完全に別個にすることができることが理解できよう。従って、図４３に示されるように、主ＴＬＢのルックアップ２０８が必要なときに、ドメインをチェックする外に、ＡＳＩＤフラグもチェックする。

【０１９５】

従って、安全ドメインにおいてプロセスＰ１が実行されている図４３における例では、ルックアッププロセスは主ＴＬＢ２０８内の２つのエントリー４５０を識別し、２つのデスクリプタ内の仮想アドレス部分がメモリアクセスリクエストによって発生された仮想アドレスの対応する部分と一致するかどうかに応じて一致信号または不一致信号が発生される。一致している場合、対応するアクセス制御情報が抽出され、マイクロＴＬＢ２０６、アクセス許可ロジック２０２および領域属性ロジック２０４へ送られる。一致していない場合、不一致信号が発生され、変換テーブルウォークロジック２１０が使用され、安全プロセスＰ１用に設けられたページテーブルから必要なデスクリプタを探すように、主ＴＬ

Ｂ２０８内の検索が行われる。当業者であれば理解できるように、ＴＬＢの内容を管理する技術は多数あるので、ＴＬＢ２０８に記憶するのに新しいデスクリプタが検索され、主ＴＬＢが既に満杯となっているときには主ＴＬＢから取り出されるデスクリプタを決定し、新しいデスクリプタ、例えば最後に利用されたアプローチなどのための余裕を設けるために、公知の多数の技術のうちの１つを使用できる。

【０１９６】

安全作動モードで使用される安全カーネルを非安全オペレーティングシステムと完全に別個に開発できることが理解できよう。しかしながら、あるケースでは安全カーネルと非安全オペレーティングシステムの開発とは密にリンクしていることがあり、かかる状況では安全アプリケーションは非安全デスクリプタを使用できるようにすることが適当である。これによって仮想アドレスを知るだけで安全アプリケーションが（表４のために）非安全データに直接アクセスできるようになる。当然ながらこのことは、特定のＡＳＩＤに対して安全仮想マッピングと非安全仮想マッピングが排他的であると仮定している。かかるシナリオでは、安全デスクリプタと非安全デスクリプタとを区別するのに、前に導入されたタグ（すなわちドメインフラグ）は不要となる。その代わりに、利用できるデスクリプタのすべてと共にＴＬＢ内のルックアップが実行される。

【０１９７】

好ましい実施例では、別個の安全デスクリプタと非安全デスクリプタによる主ＴＬＢのこのようなコンフィギュレーションと前に説明したコンフィギュレーションの選択をＣＰ１５制御レジスタ内に設けられた特定のビットによってセットできる。好ましい実施例では、このビットは安全カーネルによってしかセットされない。

【０１９８】

安全アプリケーションが非安全仮想アドレスを利用することが直接認められている実施例では、非安全スタックポインタを安全ドメインから利用できるようにすることが可能である。このことは、非安全スタックポインタをＣＰ１５レジスタ３４内の専用レジスタ内に識別させる非安全レジスタの値をコピーすることによって行うことができる。これによって安全アプリケーションによって理解される方式に従い、非安全アプリケーションがスタックを介してパラメータを送ることができるようになる。

【０１９９】

前に述べたように、メモリを非安全部分と安全部分とに区分することができ、このような区分はパーティションチェッカー２２２に対して専用のＣＰ１５レジスタ３４を使用する安全カーネルによって制御される。基本的な許可方法は、代表的なＭＰＵデバイス内で定義できるような領域アクセスパーミッションに基づく。従って、メモリは複数の領域に分割され、各領域はそのベースアドレス、サイズ、メモリ属性およびアクセスの許可によって定義することが望ましい。更にオーバーラップ領域をプログラムする際に上方領域の属性が最高の優先権を有する。更に本発明の好ましい実施例によれば、対応する領域が安全メモリにあるのか、または非安全メモリ内にあるのかを定めるのに、新しい領域属性が提供される。この新しい領域属性は安全メモリとして保護すべきメモリ部分を定めるように、安全カーネルによって使用される。

【０２００】

ブートステージにおいて、図４４に示されるように第１パーティションが実行される。この最初のパーティションは非安全な世界、非安全オペレーティングシステムおよび非安全アプリケーションに割り当てられたメモリ４６０の数量を決定する。この数量はパーティションに定められた非安全領域に対応する。この情報はメモリ管理のために非安全オペレーティングシステムによって利用される。安全として定められたメモリ４６２、４６４の他の部分は非安全オペレーティングシステムによって知られることはない。非安全な世界における保全性を保護するために、非安全メモリに安全特権モードだけのためのアクセス許可をプログラムすることができる。従って、安全アプリケーションは非安全アプリケーションの内容を破壊しない。図４４から判るようにこのブートステージパーティションの後でメモリ４６０は非安全オペレーティングシステムによって使用できるようになり、

メモリ４６２は安全カーネルによって使用できるようになり、メモリ４６４は安全アプリケーションによって使用できるようになる。

【０２０１】

ブートステージパーティションが一旦実行されると、非安全メモリ４６０のメモリマッピングはＭＭＵ２００を使用する非安全オペレーティングシステムにより取り扱われ、従って、通常の態様で一連の非安全ページを定義できる。これについては図４５に示されている。

【０２０２】

安全アプリケーションが非安全アプリケーションと共にメモリを共用する必要がある場合、安全カーネルはあるドメインから別のドメインへデータを人為的に転送するために、メモリの一部の権利を変えることができる。従って、図４６に示されるように、安全カーネルは非安全ページの保全性をチェックした後に、そのページの権利を変え、よってそのページが共用メモリとしてアクセスできる安全ページ４６６となる。

【０２０３】

メモリのパーティションを変えるときには、マイクロＴＬＢ２０６をフラッシングする必要がある。従って、このシナリオではその後に非安全アクセスが生じると、マイクロＴＬＢ２０６で不一致が生じるので、主ＴＬＢ２０８から新しいデスクリプタがロードされる。この新しいデスクリプタは、マイクロＴＬＢへの検索が試みられる場合のように、ＭＰＵのパーティションチェッカー２２２によってその後チェックされ、よってメモリの新しいパーティションと一致した状態となる。

【０２０４】

好ましい実施例では、キャッシュ３８には仮想インデックスがつけられ、物理タグがつけられる。従って、キャッシュ３８でアクセスが実行される際に、まずマイクロＴＬＢ２０６でルックアップが実行されるので、アクセス許可、特に安全許可および非安全許可がチェックされる。従って、非安全アプリケーションによりキャッシュ３８に安全データを記憶することはできない。キャッシュ３８へのアクセスはパーティションチェッカー２２２により実行されるパーティションチェックの制御下にあるので、安全データへのアクセスは非安全モードでは実行できない。

【０２０５】

しかしながら、生じ得る１つの問題は、非安全ドメインにおいてアプリケーションがキャッシュ動作レジスタを使用してキャッシュを無効にしたり、クリーンまたはフラッシングできることである。かかる動作がシステムのセキュリティに影響できないように保証する必要がある。例えば非安全オペレーティングシステムがキャッシュ３８をクリーンにすることなくキャッシュ３８を無効にしようとする場合、データを置換する前にこの外部メモリに安全な汚れたデータを書き込まなければならない。キャッシュにおいて、安全データにタグを付け、よって所望する場合、異なるように取り扱いできるようにすることが望ましい。

【０２０６】

好ましい実施例において、アドレスによるラインを無効化する動作が非安全プログラムによって実行される場合、物理アドレスはパーティションチェッカー２２２によりチェックされ、キャッシュラインが安全キャッシュラインである場合、動作はクリーンおよび無効化動作となり、システムのセキュリティを維持することを保証できる。更に好ましい実施例では、非安全プログラムによって実行されるインデックスによるライン無効化動作のすべてはインデックスによるクリーンおよび無効化動作となる。同様に、非安全プログラムによって実行されるすべてを無効化する動作のすべては、すべてをクリーンかつ無効化する動作となる。

【０２０７】

更に図１を参照すると、ＤＭＡ３２によるＴＣＭ３６へのアクセスは、マイクロＴＬＢ２０６によって制御される。従って、ＤＭＡ３２が仮想アドレスを物理アドレスに変換するために、ＴＬＢ内でルックアップを実行するときに、主ＴＬＢにおいて追加された、前

に説明したフラグにより、必要なセキュリティチェックをあたかもアクセスリクエストがコア１０によって発生されたかのように実行できるようになる。更に後述するように、好ましくはアービッター／デコーダブロック５４内に位置する外部バス７０にレプリカパーティションチェッカーが送られるので、ＤＭＡ３２が外部バスインターフェース４２を介し、外部バス７０に結合されたメモリに直接アクセスする場合、外部バスへ接続されたレプリカパーティションチェッカーがアクセスの有効性をチェックする。更に所定の好ましい実施例では、ＤＭＡコントローラ３２を非安全ドメイン内で使用できるかどうかを定めるのにＣＰＵ１５レジスタ３４にビットを追加することができる。このビットは特権モードで作動しているときの安全カーネルによってしかセットできない。

【０２０８】

ＴＣＭ３６を検討する際に、このＴＣＭ３６に安全データを入れなければならない場合、注意深くこれを取り扱わなければならない。例えば非安全オペレーティングシステムがＴＣＭメモリ３６のための物理アドレスレンジをプログラムし、このアドレスレンジが外部安全メモリ部分とオーバーラップするようなシナリオを想像することができる。作動モードが安全モードに変わった場合、安全カーネルはデータをそのオーバーラップした部分に記憶させることができ、一般にＴＣＭ３６は外部メモリよりも高い優先権を有するので、ＴＣＭ３６内に一般にデータが記憶される。非安全オペレーティングシステムがＴＣＭ３６のための物理アドレス空間の設定を変え、よってメモリの非安全物理領域内に前の安全領域がマッピングされる場合に、非安全オペレーティングシステムは安全データにアクセスできることが理解できよう。その理由は、パーティションチェッカーがその領域を非安全領域と見てアボート信号をアサートしないからである。従って、要約すれば、ＴＣＭが通常のローカルＲＡＭとして働き、スマートキャッシュとしては働かないようになっている場合、非安全オペレーティングシステムがＴＣＭベースレジスタの非安全物理アドレスへ移動できる場合に、安全な世界のデータを読み出すことが可能である。

【０２０９】

この種のシナリオを防止するために、好ましい実施例では安全特権作動モードでしかアクセスできないＣＰ１５レジスタ３４内に制御ビットが設けられ、２つの可能性のあるコンフィギュレーションが提供される。第１コンフィギュレーションではこの制御ビットは「１」に制御され、この場合、ＴＣＭは安全特権モードによってしか制御できない。従って、ＣＰ１５レジスタ３４内のＴＣＭ制御レジスタに対して試みられる非安全アクセスにより、未定義命令例外が入力される。従って、この第１コンフィギュレーションでは安全モードと非安全モードの双方がＴＣＭを利用できるが、ＴＣＭは安全特権モードによってしか制御されない。第２コンフィギュレーションでは制御ビットは０にセットされ、この場合、ＴＣＭは非安全オペレーティングシステムによって制御できる。この場合、ＴＣＭは非安全アプリケーションによってしか使用されない。ＴＣＭから安全データを記憶したりロードしたりすることはできない。従って、安全アクセスが実行されるときに、アドレスがＴＣＭアドレスレンジに一致しているかどうかを見るために、ＴＣＭではルックアップは実行されない。

【０２１０】

このシナリオにおいて、非安全オペレーティングシステムを変える必要がないように、デフォルトによりＴＣＭを非安全オペレーティングシステムによってしか使用されないようにすることができる。

【０２１１】

前に述べたように、ＭＰＵ２２０内にパーティションチェッカー２２２を設ける他に、本発明の好ましい実施例は外部バス７０に結合された類似のパーティションチェックブロックも設けている。この追加パーティションチェッカーは他のマスターデバイス、例えばデジタル信号プロセッサ（ＤＳＰ）５０、外部バスに直接結合されたＤＭＡコントローラ５２、外部バスインターフェース４２などを介して外部バスに接続できるＤＭＡコントローラ３２によるメモリへのアクセスを規制するのに使用される。外部（またはデバイス）バスにパーティションチェックブロックを結合することしかできず、パーティションチェ

ッカーをメモリ管理ロジック３０として設けることはできない。かかる実施例では、オプションとしてメモリ管理ロジック３０の一部としてパーティションチェッカーを設けることができる。かかる状況では、このパーティションチェッカーはデバイスバスに結合されたチェッカーの他に設けられた別のパーティションチェッカーと見なすことができる。

【０２１２】

上記のように、全メモリシステムをいくつかのメモリユニットから構成でき、外部バス７０上にこれら種々のメモリユニット、例えば外部メモリ５６、ブートＲＯＭ４４または周辺デバイス、例えばスクリーンドライバー４６内に設けられたバッファまたはレジスタ４８、６２、６６、Ｉ／Ｏインターフェース６０、キー記憶ユニット６４などが存在し得る。更に、安全メモリとしてメモリシステムの異なる部分を構成する必要がある場合がある。例えばキー記憶ユニット６４内のキーバッファ６６を安全メモリとして取り扱わなければならないことが好ましい場合もある。外部バスに結合されたデバイスにより、かかる安全メモリへのアクセスを試みる場合、コア１０を含むチップ内に設けられた前記メモリ管理ロジック３０はかかるアクセスを規制できなくなる。

【０２１３】

図４７は、本明細書でデバイスバスとしても称す外部バスに結合された増設パーティションチェッカー４９２をどのように使用するかを示している。外部バスは、例えばデバイス４７０、４７２により外部バスへメモリアクセスリクエストが発生されるときはいつも、メモリアクセスリクエストは作動モード、例えば特権作動モード、ユーザー作動モードなどを定める所定の信号も外部バスに含む。本発明の好ましい実施例によれば、メモリアクセスリクエストはデバイスが安全モードで作動しているか、または非安全モードで作動しているかを識別するためのドメイン信号を外部バスに発生することにも関与する。このドメイン信号はハードウェアレベルで発生することが好ましく、好ましい実施例では、安全ドメインまたは非安全ドメインで作動できるデバイスは外部バス内のパス４９０にドメイン信号を出力するための所定のピンを含む。図解のために、このパス４９０は外部バス上の他の信号パス４８８と別個に示されている。

【０２１４】

本明細書でＳビットとも称するこのドメイン信号はメモリアクセスリクエストを発生するデバイスが安全ドメインで作動しているのか、または非安全ドメインで作動しているのかを識別する。この情報は外部バスに結合されているパーティションチェッカー４９２により受信される。パーティションチェッカー４９２はメモリのどの領域が安全であるか、または安全でないかを識別するパーティション情報にもアクセスを行うので、安全作動モードを識別するのにＳビットがアサートされた場合にデバイスがメモリの安全部分にアクセスできるだけでよいように構成できる。

【０２１５】

デフォルトによりＳビットがアンアサートされ、従って、例えば図４７に示されるデバイス４７２の様な既に存在する非安全デバイスがアサートされたＳビットを出力せず、従ってメモリの安全部分がスクリーンドライバー４８０、Ｉ／Ｏインターフェース４８４のレジスタまたはバッファ４８２、４８６内にあるのか、または外部メモリ４７８内にあるのかにかかわらず、パーティションチェッカー４９２によるこのメモリの安全部分へのアクセス権は、このデバイスには付与されない。

【０２１６】

図解のため、マスターデバイス、例えばデバイス４７０、４７２が発生するメモリアクセスリクエスト間の仲裁のために使用されるアービッターブロック４７６は、メモリアクセスリクエストのサービスをするのに適当なメモリデバイスを決定するのに使用されるデコーダ４７８およびパーティションチェッカー４９２と別個に示されている。しかしながら、所望すれば同じユニット内に１つ以上のこれら部品を組み込んでもよいことが理解できよう。

【０２１７】

図４８はパーティションチェッカー４９２が設けられておらず、代わりに各メモリデバ

イス４７４、４８０、４８４がＳビットの値に応じて自己のメモリアクセスを規制するようになっている別の実施例を示す。従って、デバイス４７０が安全メモリとしてマークされたスクリーンドライバー４８０内のレジスタ４８２に対する非安全モードのメモリアクセスリクエストをアサートした場合、スクリーンドライバー４８０はＳビットがアサートされていないと決定し、メモリアクセスリクエストを処理しない。従って、種々のメモリデバイスを適当に設計することにより、パーティションチェッカー４９２を外部バスと別個に設けなくてもよいようにできる。

【０２１８】

図４７および４８の上記説明では、メモリアクセスリクエストを発生するデバイスが安全ドメインで作動しているのか、または非安全ドメインで作動しているのかを識別するのにＳビットが使用される。別の見方をすれば、このＳビットはメモリアクセスリクエストが安全ドメインに関係するのか、または非安全ドメインに関係するのかを表示すると見なすこともできる。

【０２１９】

図３７および３８を参照して説明した実施例では、仮想アドレス－物理アドレス変換を実行するのに、ページテーブルの１つの組と共に単一のＭＭＵが使用された。かかる方法では、物理アドレス空間は図４９に示されるように、簡単な態様で一般に非安全メモリと安全メモリとにセグメント化される。ここで物理アドレス空間２１００はメモリシステム内のメモリユニットのうちの１つ、例えば外部メモリ５６に対してアドレスゼロでスタートし、アドレスＹまで続くアドレス空間を含む。各メモリユニットに対し、アドレス指定可能なメモリは一般に２つの部分、すなわち非安全メモリとして割り当てられる第１部分２１１０と、安全メモリとして割り当てられる第２部分２１２０とに区分される。

【０２２０】

このような方法を使用すると、特定のドメイン（単数または複数）にアクセスできない所定のアドレスが生じ、これらギャップはドメインで使用されるオペレーティングシステムに対して明らかとなることが理解できよう。安全ドメインで使用されるオペレーティングシステムは非安全ドメインの知識を有し、従ってこれと関係することはないが、非安全ドメイン内のオペレーティングシステムは理想的には安全ドメインが存在するとの知識を有しなくてもよいようにすべきであり、むしろ安全ドメインがそこに存在していなかったかのように作動すべきである。

【０２２１】

別の問題として、非安全オペレーティングシステムは外部メモリに対するアドレス空間をアドレスゼロでスタートし、アドレスＸまで続くものと見て、非安全オペレーティングシステムは安全カーネル、特にアドレスＸ＋１からアドレスＹまで続く安全メモリの存在について何も知る必要はないことが理解できよう。これと対照的に、安全カーネルはアドレスゼロで始まるアドレス空間を見ない。このことはオペレーティングシステムが一般に期待することではない。

【０２２２】

図５１には、物理アドレス空間の非安全オペレーティングシステムの視界から安全メモリ領域を完全に隠すことができるようにし、かつ安全ドメインにおける安全カーネルおよび非安全ドメインにおける非安全オペレーティングシステムが外部メモリのためのアドレス空間をアドレスゼロで開始するものと見ることができるようにすることによって、上記問題を解消した一実施例が略図で示されている。ここでは、物理アドレス空間２２００をページレベルで安全セグメントまたは非安全セグメントのいずれかにセグメント化することができ、図５１に示された例では、外部メモリ用のアドレス空間は２つの安全メモリ領域と２つの非安全メモリ領域から成る４つのセクション２２１０、２２２０、２２３０、２２４０にセグメント化された状態に示されている。

【０２２３】

単一ページテーブル変換による仮想アドレス空間と物理アドレス空間との切り替えではなく、第１ページテーブルと第２ページテーブルとを参照して、アドレスの２つの別個の

層の変換を実行し、プロセッサが安全ドメイン内にあるのか、または非安全ドメイン内にあるのかに応じて別々に構成できる中間アドレス空間の概念を導入できる。より詳細に説明すれば、図５１に示されるように、物理アドレス空間内の２つの安全メモリ領域２２１０および２２３０をページテーブル２２５０の組内の安全ページテーブル内で設けられたデスクリプタを使用することにより、安全ドメインのための中間アドレス空間における単一領域２２６５にマッピングできる。プロセッサ上で作動しているオペレーティングシステムに関しては、オペレーティングシステムは中間アドレス空間を物理アドレス空間として見て、ＭＭＵを使って仮想アドレスを中間アドレス空間内の中間アドレスに変換する。

【０２２４】

同様に非安全ドメインに対して中間アドレス空間２２７０を構成でき、この空間では物理アドレス空間内の２つの非安全メモリ領域２２２０と２２４０が、ページテーブル２２５０の組内の非安全ページテーブル内の対応するデスクリプタを介し、非安全ドメインのための中間アドレス空間内の非安全領域２２７５にマッピングされる。

【０２２５】

一実施例では、図５０Ａに示されるように、２つの別個のＭＭＵを使って中間アドレスを介した仮想アドレスの物理アドレスへの変換が取り扱われる。図５０Ａ内のＭＭＵ２１５０と２１７０の各々は、図３７に示されたＭＭＵ２００へ同じように構成されたものと見なすことができるが、図解を容易にするために、図５０Ａでは所定の細部は省略されている。

【０２２６】

第１ＭＭＵ２１５０はマイクロＴＬＢ２１５５と、主ＴＬＢ２１６０と、変換テーブルウォークロジック２１６５とを含むが、同じように第２ＭＭＵ２１７０はマイクロＴＬＢ２１７５と、主ＴＬＢ２１８０と、変換テーブルウォークロジック２１８５とを含む。第１ＭＭＵはプロセッサが非安全ドメイン内で作動しているときには非安全オペレーティングシステムにより制御でき、またプロセッサが安全ドメイン内で作動しているときには安全カーネルにより制御できる。しかしながら、好ましい実施例では第２ＭＭＵは安全カーネルまたはモニタプログラムによってしか制御できない。

【０２２７】

プロセッサコア１０がメモリアクセスリクエストを発生する際に、このコアはパス２１５３を通して仮想アドレスをマイクロＴＬＢ２１５５へ発生する。このマイクロＴＬＢ２１５５は主ＴＬＢ２１６０内に記憶されていたデスクリプタから検索された対応する中間アドレス部分を多数の仮想アドレス部分に対して記憶する。主ＴＬＢ２１６０に記憶されていたデスクリプタは第１ＭＭＵ２１５０に関連していたページアドレスの第１の組内のページテーブルから検索されたものである。マイクロＴＬＢ２１５５内で一致が検出された（ヒットした）場合、マイクロＴＬＢ２１５５はパス２１５３を通して受信された仮想アドレスに対応する中間アドレスをパス２１５７を通して発生できる。マイクロＴＬＢ２１５５内に一致がない場合、主ＴＬＢ内で一致が検出されたかどうかを見るために、主ＴＬＢ２１６０が参照される。一致が発見された場合、対応する仮想アドレス部分および対応する中間アドレス部分がマイクロＴＬＢ２１５５内に検索され、その後、パス２１５７を通して中間アドレスを発生できる。

【０２２８】

マイクロＴＬＢ２１５５および主ＴＬＢ２１６０内に一致がない場合、第１ＭＭＵ２１５０によってアクセスできるページテーブルの第１の組内の所定のページテーブルから必要なデスクリプタのためのリクエストを発生するのに変換テーブルウォークロジック２１６５が使用される。一般に、安全ドメインと非安全ドメインの双方のための個々のプロセスに関連するページテーブルがあり、例えばＣＰ１５レジスタ３４内の適当なレジスタから変換テーブルウォークロジック２１６５によってページテーブルのための中間ベースアドレスにアクセスできる。従って、変換テーブルウォークロジック２１６５は適当なページテーブルからデスクリプタをリクエストするために、パス２１６７を通して中間アドレスを発生できる。

【０２２９】
第２ＭＭＵ２１７０はパス２１５７を通してマイクロＴＬＢ２１５５によりパス２１６７を通って変換テーブルウォークロジック２１６５により中間アドレス出力を受信するようになっており、マイクロＴＬＢ２１７５内で一致が検出された場合、マイクロＴＬＢはパス２１９２を通してメモリへ必要な物理アドレスを発生し、データバス２１９０を通して必要なデータを検索することができる。中間アドレスがパス２１５７を通して発生された場合、これによって必要なデータがコア１０に戻され、一方、パス２１６７を通して発生された中間アドレスに対し、この発生によって必要なデスクリプタが第１ＭＭＵ２１５０へ戻され、主ＴＬＢ２１６０内に記憶できる。

【０２３０】
マイクロＴＬＢ２１７５内で不一致が生じた場合、主ＴＬＢ２１８０が参照され、主ＴＬＢ内で一致が発見された場合、必要な中間アドレス部分および対応する物理アドレス部分がマイクロＴＬＢ２１７５へ戻され、マイクロＴＬＢ２１７５がパス２１９２を通して必要な物理アドレスを発生できるようにする。しかしながら、マイクロＴＬＢ２１７５でも主ＴＬＢ２１８０でも一致がない場合、変換テーブルウォークロジック２１８５は、第２ＭＭＵ２１７０に関連するページテーブルの第２の組内の対応するページテーブルから必要なデスクリプタに対し、パス２１９４を通してリクエストを出力する。ページテーブルのこの第２の組は中間アドレス部分と物理アドレス部分とを関連づけるデスクリプタを含み、一般に安全ドメインに対して少なくとも１つのページテーブルおよび非安全ドメインに対して１つのページテーブルが設けられる。パス２１９４を通してリクエストが発生されると、この結果、ページテーブルの第２の組から対応するデスクリプタが第２のＭＭＵ２１７０に戻され、主ＴＬＢ２１８０内に記憶される。

【０２３１】
次に、下記の特定の実施例により、図５０Ａに示された実施例の作動について説明する。この例では、略語ＶＡは仮想アドレスを示し、ＩＡは中間アドレスを示し、ＰＡは物理アドレスを示す。

１）コアがＶＡ＝３０００を発生する　［ＩＡ＝５０００、ＰＡ＝７０００］
２）ＭＭＵ１のマイクロＴＬＢで不一致（失敗）
３）ＭＭＵ１の主ＴＬＢで不一致
ページテーブル１のベースアドレス＝８０００ＩＡ　　［ＰＡ＝１００００］
４）ＭＭＵ１内の変換テーブルウォークロジックがページテーブルルックアップを実行し、ＩＡ＝８００３を発生する
５）ＭＭＵ２のマイクロＴＬＢ内で不一致
６）ＭＭＵ２の主ＴＬＢ内で不一致
ページテーブル２のベースアドレス＝１２０００ＰＡ
７）ＭＭＵ２内の変換テーブルウォークロジックがページテーブルルックアップを実行し、ＰＡ＝１２００８を発生する
　　ページテーブルデータとして戻された「８０００ＩＡ＝１００００ＰＡ」
８）ＭＭＵ２の主ＴＬＢに記憶される
９）ＭＭＵ２のマイクロＴＬＢに記憶される
１０）ＭＭＵ２内のマイクロＴＬＢで一致（ヒット）し、ＰＡ＝１０００３を発生
　　ページテーブルデータとして戻された「３０００ＶＡ＝５０００ＩＡ」
１１）ＭＭＵ１の主ＴＬＢに記憶される
１２）ＭＭＵ１のマイクロＴＬＢに記憶される
１３）ＭＭＵ１内のマイクロＴＬＢで一致し、
　　データアクセスを実行するのにＩＡ＝５０００を発生する
１４）ＭＭＵ２のマイクロＴＬＢ内で不一致
１５）ＭＭＵ２の主ＴＬＢ内で不一致
１６）ＭＭＵ２内の変換テーブルウォークロジックがページテーブルルックアップを実行し、ＰＡ＝１２００５を発生する

ページテーブルデータとして戻された「５０００ＩＡ＝７０００ＰＡ」
１７）ＭＭＵ２の主ＴＬＢ内に記憶される
１８）ＭＭＵ２のマイクロＴＬＢ内に記憶される
１９）ＭＭＵ２内のマイクロＴＬＢで一致し、
データアクセスを実行するのにＰＡ＝７０００を発生する
２０）物理アドレス７０００にあるデータがコアに戻される

【０２３２】

次のときにコアがメモリアクセスリクエスト（すなわちＶＡ３００１．．．）を発生する。
１）コアがＶＡ＝３００１を発生する
２）ＭＭＵ１のマイクロＴＬＢ内で一致すると、リクエストＩＡ５００１がＭＭＵ２に発生される
３）ＭＭＵ２のマイクロＴＬＢで一致すると、ＰＡ７００１に対するリクエストがメモリに対して発生される
４）ＰＡ７００１におけるデータがコアに戻される
上記例では双方のＭＭＵのマイクロＴＬＢおよび主ＴＬＢの双方で不一致となるので、この例は最悪ケースのシナリオを示している。一般にマイクロＴＬＢまたは主ＴＬＢの少なくとも一方で一致が発見されることが予想されるので、データを検索するのにかかる時間を大幅に短縮できる。

【０２３３】

次に図５１を参照すると、物理アドレス空間の所定領域、好ましい実施例では安全領域内にページテーブル２２５０の第２の組が一般に設けられる。ページテーブルの第１の組は２つのタイプ、すなわち安全ページテーブルと非安全ページテーブルとに分割できる。非安全中間アドレス空間２２７５内の非安全ページテーブルと同じように、中間アドレス空間２２６５内では安全ページテーブルが次々に現れることが好ましい。しかしながら、物理アドレス空間内ではこれらページテーブルは次々に生じる必要はないので、例えばページテーブルの第１の組に対する安全ページテーブルが安全領域２２１０、２２３０内にわたって拡散されていてもよく、同じように非安全ページテーブルも非安全メモリ領域２２２０および２２４０にわたって拡散していてもよい。

【０２３４】

上記のように、ページテーブルの２つの組の２レベルの方法を使用する主な利点の１つは、安全ドメインのオペレーティングシステムの非安全ドメインのオペレーティングシステムの双方に対し、物理アドレス空間をゼロからスタートするように配置できることであり、このことは、一般にオペレーティングシステムが期待することである。更に、安全メモリ領域を物理アドレス空間の非安全オペレーティングシステムの視界から完全に隠すことができる。この理由は、オペレーティングシステムは中間アドレスの連続シーケンスを有するように配置できる中間アドレス空間を物理アドレス空間として見るからである。

【０２３５】

更にかかる解決方法を使用することによって、非安全メモリと安全メモリとの間でメモリの領域をスワップするプロセスをかなり簡潔にできる。このことは図５２を参照して略図で示されている。図５２から判るように、例えばメモリの単一ページとすることができるメモリ２３００の領域が、非安全メモリ領域２２２０内に存在することができ、同様に安全メモリ領域２２１０内にはメモリ領域２３１０が存在し得る。しかしながらページテーブルの第２の組内で対応するデスクリプタを変えるだけで、これら２つのメモリ領域２３００と２３１０とを容易にスワップすることができるので、領域２３００は安全ドメインの中間アドレス空間内の領域２３０５にマッピングされた安全領域となり、次に領域２３１０は非安全ドメインの中間アドレス空間内の領域２３１５にマッピングされた非安全領域となる。このことは安全ドメインと非安全ドメインの双方においてオペレーティングシステムに対して完全にトランスペアレントに生じ得る。その理由は、物理アドレス空間の視界は実際には安全ドメインまたは非安全ドメインの中間アドレス空間であるからであ

る。従って、この方法によって各オペレーティングシステム内で物理アドレス空間を再定義しなくてもよいようになる。

【０２３６】

次に図５０Ｂを参照し、図５０Ａの配置と異なる配置の２つのＭＭＵも使用する、本発明の別の実施例について説明する。図５０Ｂと図５０Ａを比較することから判るように、配置はほとんど同じであるが、この実施例では第１ＭＭＵ２１５０が仮想アドレス－物理アドレス変換を実行するようになっており、第２ＭＭＵが中間アドレス－物理アドレス変換を実行するようになっている。従って、図５０Ａの実施例で使用される、第１ＭＭＵ２１５０内のマイクロＴＬＢ２１５５から第２ＭＭＵ２１７０内のマイクロＴＬＢ２１７５までのパス２１５７の代わりに、第１ＭＭＵ内のマイクロＴＬＢが図５０Ｂに示されるようにパス２１９２を通して直接物理アドレスを出力するようになっている。次に、下記の特定の例により、図５０Ｂに示された実施例の作動について説明する。この図は図５０Ａの実施例に対して前に説明したように、同じコアメモリのアクセスリクエストの処理を詳細に示している。

【０２３７】

１）コアがＶＡ＝３０００を発生する　［ＩＡ＝５０００、ＰＡ＝７０００］

２）ＭＭＵ１のマイクロＴＬＢおよび主ＴＬＢ内で不一致

ページテーブル１のベースアドレス＝８０００ＩＡ［ＰＡ＝１００００］

３）ＭＭＵ１内の変換テーブルウォークロジックがページテーブルルックアップを実行し、

ＩＡ＝８００３を発生する

４）ＭＭＵ２のマイクロＴＬＢおよび主ＴＬＢ内でＩＡ８００３が不一致

ページテーブル２のベースアドレス＝１２０００ＰＡ

５）ＭＭＵ２内の変換テーブルウォークロジックがページテーブルルックアップを実行し、

ＰＡ＝１２００８を発生する

ページテーブルデータとして戻された［８０００ＩＡ＝＝１００００ＰＡ］

６）ＭＭＵ２の主およびマイクロＴＬＢに「８０００ＩＡ＝１００００ＰＡ」がマッピング記憶される

７）ＭＭＵ２内のマイクロＴＬＢはステップ（３）からＰＡ１０００３へリクエストを変換でき、フェッチ信号を発生する

ページテーブルとして戻された「３０００ＶＡ＝５０００ＩＡ」

注：　この変換はＭＭＵ１によって一時記憶装置内に保持されるが、ＴＬＢ内には直接記憶されない

８）次にＭＭＵ１の変換テーブルウォークロジックはＩＡ＝５０００に対し、ＭＭＵ２にリクエストを発生する

９）ＭＭＵ２のマイクロＴＬＢおよび主ＴＬＢ内でＩＡ＝５０００が不一致

１０）ＭＭＵ２内の変換テーブルウォークロジックがページテーブルルックアップを実行し、

ＰＡ＝１２００５を発生する

ページテーブルデータとして戻された「５０００ＩＡ＝７０００ＰＡ」

１１）ＭＭＵ２がマイクロＴＬＢおよび主ＴＬＢに「５０００ＩＡ＝７０００ＰＡ」を記憶する。この変換もＭＭＵ１に伝えられる

１２ａ）ＭＭＵ２がメモリへＰＡ＝７０００のアクセスを発生する

１２ｂ）ＭＭＵ２が「３０００ＶＡ＝５０００ＩＡ」デスクリプタと、

「５０００ＩＡ＝７０００ＰＡ」デスクリプタとを組み合わせ、「３０００ＶＡ＝７０００ＰＡ」デスクリプタを発生し、このデスクリプタはＭＭＵ１の主ＴＬＢおよびマイクロＴＬＢに記憶される

１３）ＰＡ＝７０００にあるデータがコアに戻される

【０２３８】

次にコアがメモリアクセスリクエスト（例えばＶＡ＝３００１．．．）を発生する。
１）コアがＶＡ＝３００１を発生する
２）ＭＭＵ１、ＭＭＵ２のマイクロＴＬＢ内での一致がＰＡ＝７００１に対するリクエストを発生する
３）ＰＡ＝７００１におけるデータがコアに戻される
上記例と図５０Ａの例との比較から判るように、主な違いはＭＭＵ１が第１テーブルデスクリプタを直接記憶しないステップ７、およびＭＭＵ１もＩＡ→ＰＡ変換を受信し、組み合わせを行わず、組み合わされたデスクリプタをそのＴＬＢに記憶するステップ１２Ｂ（ステップ１２Ａと１２Ｂとは同時に生じ得る）にある。
従って、この別の実施例は仮想アドレスから物理アドレスへの変換をするのにまだページテーブルの２つの組を使用しているが、マイクロＴＬＢ２１５５と主ＴＬＢ２１６０が直接仮想アドレス－物理アドレス変換を記憶することにより、マイクロＴＬＢ２１５５または主ＴＬＢ２１６０のいずれかで一致が生じたときに、双方のＭＭＵでルックアップを実行する必要がなくなることが理解できよう。かかるケースでは、第１ＭＭＵは第２ＭＭＵを参照することなく、コアからのリクエストを直接取り扱うことができる。

【０２３９】

第２ＭＭＵ２１７０がマイクロＴＬＢ２１７５および主ＴＬＢ２１８０を含まないようにすることができると理解できよう。かかる場合、第２ＭＭＵによる取り扱い必要とするリクエストごとにページテーブルウォークロジック２１８５が使用される。これによって第２ＭＭＵの複雑性およびコストを減らすことができ、このことは第２ＭＭＵが比較的頻繁には必要とされないと仮定された場合に許容できる。リクエストごとに第１ＭＭＵを使用しなければならないので、第１ＭＭＵの作動速度を改善するのに第１ＭＭＵ２１５０内にマイクロＴＬＢ２１５５および主ＴＬＢ２１６０を設けることが一般に好ましい。

【０２４０】

ページテーブル内のページはサイズが変わることがあり、従って変換のうちの２つの半分に対するデスクリプタが異なるサイズのページに関係するようにできることに留意すべきである。一般にＭＭＵ１のページはＭＭＵ２のページよりも少ないが、必ずしもこのようなケースとはならない。例えば、
テーブル１は０ｘ４０００３０００における４Ｋｂを０ｘ０００８１０００にマッピングする。
テーブル２は０ｘ００００００００における１Ｍｂを０ｘ０２０００００にマッピングする。
ここで組み合わせ変換に対しては２つのサイズのうちの小さい方のサイズを使用しなければならないので、組み合わせデスクリプタは０ｘ４０００３０００における４Ｋｂを０ｘ０２０８１０００へマッピングする。
しかしながら、（図５２を参照して前に説明したように）世界間でデータがスワップされる場合、例えば反対も真となり得る。
テーブル１は０ｘｃ００００００から０ｘ００００００への１Ｍｂをマッピングする。
テーブル１は０ｘ０００４２０００から０ｘ０２０４２０００への４Ｋｂマッピングする。次に、コアからのアドレス０ｘｃ００４２０１０におけるルックアップによって、次のマッピング、すなわち０ｘｃ００４２０００から０ｘ０２０４２０００への４Ｋｂのマッピングを生じさせる。
すなわち組み合わせマッピングに対して２つのサイズのうちの小さい方のサイズが常に使用される。

【０２４１】

第２のケースでは、このプロセスはあまり効率的ではなくなることに留意されたい。その理由は、テーブル１における（１Ｍｂ）のデスクリプタは繰り返しルックアップされ、異なる４Ｋｂの領域に対するアクセスがされるときに廃棄されるからである。しかしながら、一般的なシステムではテーブル２のデスクリプタのほうがほとんどの時間で（第１の

例と同じように）大きくなる。このほうがより効率的である（１Ａの空間の適当な部分をポイントする別の４Ｋｂのページに対して１Ｍｂのマッピングを繰り返すことができる）。

【０２４２】

図５０Ａおよび５０Ｂに示されるような２つの別個のＭＭＵを使用する別の例として、図５３に示されるような１つのＭＭＵを使用することもできる。この場合、主ＴＬＢ２４２０で一致が生じると、ＭＭＵにより例外信号が発生され、この信号はコア１０内でソフトウェアを作動させ、ページテーブルの２つの異なる組からのデスクリプタの組み合わせに基づき、仮想－物理アドレス変換を生じさせる。より詳細には、図５３に示されるように、マイクロＴＬＢ２４１０および主ＴＬＢ２４２０を含むＭＭＵ２４００にコア１０が結合されている。コア１０がメモリアクセスリクエストを発生すると、パス２４３０を通して仮想アドレスが与えられ、マイクロＴＬＢ内で一致が発見された場合、対応する物理アドレスがパス２４４０を通して直接出力され、パス２４５０を通してコア１０へデータが戻される。しかしながら、マイクロＴＬＢ２４１０内に一致がある場合、主ＴＬＢ２４２０が参照され、主ＴＬＢ内に対応するデスクリプタが含まれる場合、関連する仮想アドレス部分および対応する物理アドレス部分がマイクロＴＬＢ２４１０内へ検索され、その後、パス２４４０を通して物理アドレスを発生できる。しかしながら主ＴＬＢでも一致を生じた場合、パス２４２２を通してコアへ例外が発生される。次に、図５４を参照し、かかる例外の受信からコア内で実行されるプロセスについて更に説明する。

【０２４３】

図５４に示されるように、ステップ２５００でコアによりＴＬＢの一致例外が検出された場合、コアはステップ２５１０でその例外に対する所定ベクトルでモニタモードとなる。これによって、図５４に示されるステップの残りを実行するように、ページテーブルマージコードが作動させられる。

【０２４４】

より詳細には、ステップ２５２０にてパス２４３０を通して発生され、マイクロＴＬＢ２４１０および主ＴＬＢ２４２０の双方で不一致を生じさせた仮想アドレス（以下、フォールト仮想アドレスと称す）を検索し、その後、ステップ２５３０にてテーブルの第１の組内の適当なテーブルに対する中間ベースアドレスに応じて必要な第１デスクリプタに対する中間アドレスを決定する。（一般に仮想アドレスと中間ベースアドレスとの所定の組み合わせにより）一旦その中間アドレスが決定されると、次に第１デスクリプタに対する対応する物理アドレスを得るために、テーブルの第２の組内の対応するテーブルが参照される。その後、ステップ２５５０にて、メモリから第１デスクリプタをフェッチし、フォールト仮想アドレスに対する中間アドレスを決定することができる。

【０２４５】

次にステップ２５６０にてフォールト仮想アドレスの中間アドレスに対する物理アドレスを生じさせる第２デスクリプタを探すために再び第２テーブルを参照する。その後、ステップ２５７０でフォールト仮想アドレスに対する物理アドレスを得るために第２デスクリプタをフェッチする。

【０２４６】

一旦上記情報が得られると、プログラムは第１デスクリプタと第２デスクリプタとをマージし、必要な仮想アドレス－物理アドレス変換をする新しいデスクリプタを発生する。この操作はステップ２５８０で実行される。図５０Ｂを参照して前に説明したのと同じように、ソフトウェアにより実行されるマージは再び組み合わせ変換に対する最小ページテーブルサイズを使用する。その後、ステップ２５９０にて主ＴＬＢ２４２０内に新しいデスクリプタを記憶し、その後プロセスはステップ２５９０にて例外から戻る。

【０２４７】

その後、コア１０はパス２４３０を通してメモリアクセスリクエストに対して仮想アドレスを再発行するようになっており、この結果、マイクロＴＬＢ２４１０では不一致が生じるが、主ＴＬＢ２４２０では一致が生じる。従って、仮想アドレス部分と対応する物理

アドレス部分とをマイクロＴＬＢ２４１０へ検索することができ、その後、マイクロＴＬＢ２４１０がパス２４４０を通して物理アドレスを発生することができ、その結果、必要なデータがパス２４５０を通してコア１０へ戻される。

【０２４８】

図５０Ａおよび５０Ｂを参照して前に説明した実施例とは別の実施例として、図５３および５４を参照してこれまで説明した原理を使用するソフトウェアにより、これら実施例における一方または双方のＭＭＵを管理できる。

【０２４９】

図５０Ａまたは５０Ｂに示されるように、２つのＭＭＵを使用するか、図５３に示されるように１つのＭＭＵを使用するかにかかわらず、モニタモード（または特権安全モード）で作動する際に、プロセッサによりページテーブルの第２の組が管理されることによって、これらページテーブルを安全にすることができる。この結果、プロセッサが非安全ドメインにあるときには、プロセッサは非安全メモリしか見ることができない。その理由は、非安全ドメイン内にあるときにプロセッサが見ることができるのは、ページテーブルの第２の組により非安全ドメインに対して発生される中間アドレス空間だけであるからである。この結果、図１に示されるメモリ管理ロジック３０の一部として、パーティションチェッカーを設ける必要はない。しかしながら、システム内の他のバスマスターにより行われるアクセスをモニタするのに、外部バスにパーティションチェッカーが設けられる。

【０２５０】

図３７および図３８を参照して前に説明した実施例では、ＭＭＵ２００に関連してパーティションチェッカー２２２が設けられていた。従って、キャッシュ３８内でアクセスを実行すべきときに、マイクロＴＬＢ２０６内でまずルックアップが実行され、従って、アクセスパーミション、特に安全および非安全許可がチェックされていたはずである。従って、かかる実施例では非安全アプリケーションによりキャッシュ３８内に安全データを記憶することはできない。キャッシュ３８へのアクセスはパーティションチェッカー２２２により実行されるパーティションチェックによって制御されているので、非安全モードでは安全データへのアクセスを実行することはできない。

【０２５１】

しかしながら、本発明の別の実施例では、システムバス４０を通して行われるアクセスをモニタするためのパーティションチェッカー２２２は設けられず、その代わりに、データ処理装置が外部バス７０に接続されたメモリユニットへのアクセスをモニタするよう、この外部バス７０に結合された単一パーティションチェッカーを有するにすぎない。かかる実施例では、このことはプロセッサコア２０が外部パーティションチェッカーによるアクセスの規制を行うことなく、システムバス４０、例えばＴＣＭ３６およびキャッシュ３８に直接結合されたメモリユニットにアクセスでき、従って、非安全モードで作動中にプロセッサコア１０がキャッシュ３８またはＴＣＭ３６内の安全データにアクセスできないように保証するのに、ある機構が必要であることを意味している。

【０２５２】

図５５は、本発明の一実施例に係わるデータ処理装置を示し、この実施例ではキャッシュ３８および／またはＴＣＭ３６がＭＭＵ２００に関連してパーティションチェックロジックを設ける必要なく、これらへのアクセスを制御できるようにする機構が設けられている。図５５に示されるように、コア１０はＭＭＵ２００を介してシステムバス４０に結合されており、システムバス４０にはキャッシュ３８およびＴＣＭ３６も結合されている。コア１０、キャッシュ３８およびＴＣＭ３６は外部バスインターフェース４２を介して外部バス７０に結合されており、外部バス７０は図５５に示されるようにアドレスバス２６２０、制御バス２６３０およびデータバス２６４０から成る。

【０２５３】

コア１０、ＭＭＵ２００、キャッシュ３８、ＴＣＭ３６および外部バスインターフェース４２はこれらが外部バス７０に接続された単一のデバイス（デバイスバスとも称される）を構成すると見なすことができ、このデバイスには他のデバイス、例えば安全周辺デバ

イス４７０または非安全周辺デバイス４７２も結合することができる。デバイスバス７０には１つ以上のメモリユニット、例えば外部メモリ５６も接続されている。更に、デバイスバス７０にはバス制御ユニット２６５０が接続されており、この制御ユニットは一般にアービッタ２６５２と、デコーダ２６５４と、パーティションチェッカー２６５６とを含む。デバイスバスに接続された部品の作動を総合的に説明するために、前に説明した図４７を参照しなければならない。前に説明した図４７では、アービッタとデコーダとパーティションチェッカーとが別個のブロックとして示されていたが、これらを単一の制御部ロック２６５０内に設けたときでも、これら要素は同じように作動する。

【０２５４】

図５６には図５５のＭＭＵ２００がより詳細に示されている。図５６と図３７とを比較することにより、ＭＭＵ２００を図３７のＭＭＵと全く同じように構成できるが、主ＴＬＢ２０８とマイクロＴＬＢ２０６との間でパス２４２を通して送られるデータをモニタするためのパーティションチェッカー２２２が設けられている点が異なっているにすぎない。プロセッサコア１０が仮想アドレスを指定するメモリアクセスリクエストを発生する場合、このメモリアクセスリクエストはＭＭＵ２００を通してルーティングされ、図３７を参照して前に説明したように処理され、その結果、マイクロＴＬＢ２６０からパス２３８を通してシステムバス４０に物理アドレスが出力される。これと対照的に、メモリアクセスリクエストが直接物理アドレスを指定する場合、このアクセスリクエストはＭＭＵ２００をバイパスし、その代わりに直接パス２３６を介してシステムバス４０へルーティングされる。一実施例では、プロセッサはモニタモードで作動しているときにしか、物理アドレスを直接指定するメモリアクセスリクエストを発生しない。

【０２５５】

ＭＭＵ２００の前の説明から、特に図４３の説明から思い出すことができるように、主ＴＬＢ２０８は多数のデスクリプタ４３５を含み、各デスクリプタに対して対応するデスクリプタが安全ページテーブルからのものか、または非安全ページテーブルからのものかを識別するためのドメインフラグ４２５が設けられている。図５５のＭＭＵ２００内にはこれらデスクリプタ４３５および関連するドメインフラグ４２５が略図で示されている。

【０２５６】

コア１０がメモリアクセスリクエストを発生すると、この結果、メモリアクセスリクエストのための物理アドレスがシステムバス４０に出力され、そのアドレスによって指定されたデータアイテムがキャッシュ内に記憶されているかどうかを判断するためのルックアッププロセスを一般にキャッシュ３８が実行する。キャッシュ内で一致が生じた場合、すなわちアクセスリクエストを受けたデータアイテムがキャッシュ内に記憶されていないと判断された場合にはいつも、キャッシュによりラインフィル手順が開始され、メモリアクセスリクエストを受けたデータアイテムを含むデータのラインを外部メモリ５６から検索する。特にキャッシュはＥＢＩ４２を介し、デバイスバス７０の制御バス２６３０にラインフィルリクエストを出力する。この場合、アドレスバス２６２０にはスタートアドレスが出力される。更に、パス２６３２を通して制御バス２６３０にＨＰＲＯＴ信号が出力される。この信号はメモリアクセスリクエストが発生されたときにコアの作動モードを指定するドメイン信号を含む。従って、ラインプロセスをキャッシュ３８による外部バス上の元のメモリアクセスリクエストの伝搬と見なすことができる。

【０２５７】

このＨＰＲＯＴ信号はパーティションチェッカー２６５６により受信され、従ってこの信号は外部メモリ５６からの指定データをリクエストするデバイス（この場合、コア１０およびキャッシュ３８を内蔵するデバイス）がメモリアクセスリクエストの発生されたときに安全ドメインで作動中か、または非安全ドメインで作動中かをパーティションチェッカーに対して識別する。このパーティションチェッカー２６５０はメモリのどの領域が安全であるのか、または非安全であるのかを識別するパーティション情報にもアクセスし、従ってデバイスがリクエスト中のデータにアクセスできるかどうかを判断できる。従って、ＨＰＲＯＴ信号内のドメイン信号（本明細書ではＳビットとも称す）が安全モードで作

動中のデバイスにより、このデータへのアクセスがリクエストされたかどうかを識別するようにアサートされた場合にしか、デバイスがメモリの安全部分へアクセスできないようにパーティションチェッカーを構成できる。

【０２５８】

例えばコアが非安全作動モードで作動しているとＨＰＲＯＴ信号が識別したために、コア１０がリクエストされたデータへアクセスできるとパーティションチェッカーが判断し、メモリの安全領域内にあるデータを外部メモリから検索することをラインフィルリクエストが求めている場合、パーティションチェッカー２６５６は制御バス２６３０へアボート信号を発生する。このアボート信号はパス２６３６を通してＥＢＩ４２へ送り戻され、このＥＢＩ４２からキャッシュ３８へ戻されるので、その結果、アボート信号はパス２６７０を通してコア１０へ発生される。しかしながら、アクセスが認められていないとパーティションチェッカー２６５６が判断した場合、パーティションチェッカーは外部メモリから検索されたデータが安全データであるか、または非安全データであるかを識別するＳタグ信号を出力し、このＳタグ信号はパス２６３４を介してＥＢＩ４２へ送り戻され、次にこのＥＢＩからキャッシュ３８へ送られ、ラインフィルプロセスの主題であるキャッシュライン２６００に関連する２６０２のセットを可能にする。

【０２５９】

これと同時に、制御信号２６５０は、外部メモリがリクエストされたラインフィルデータを出力すること認可する。このデータはＥＢＩ４２を介してパス２６８０を通ってキャッシュ３８へ送り戻され、対応するキャッシュライン２６００に記憶される。従って、このプロセスの結果としてキャッシュ３８内の選択されたキャッシュラインは外部メモリ５６からのデータアイテムで満たされる。これらデータアイテムはコア１０からの元のメモリアクセスリクエストの主題であったデータアイテムを含む。コアからのメモリアクセスリクエストの主題であるデータアイテムは次にキャッシュ３８からコアへ戻すことができるし、またはこれとは異なり、直接ＥＢＩ４２からパス２６６０を通してコア１０へ与えることができる。

【０２６０】

好ましい実施例では、上記ラインフィルプロセスの結果としてキャッシュラインへのデータの最初の記憶が行われるので、このキャッシュラインに関連したフラグ２６０６がパーティションチェッカー２６５６によって与えられる値に基づきセットされ、そのフラグがキャッシュ３８に使用され、そのキャッシュライン２６００内のデータアイテムへのその後のアクセスを直接制御できる。従って、コア１０がキャッシュ３８の特定のキャッシュライン２６００内の一致を生じさせるメモリアクセスリクエストをその後発生する場合、キャッシュ３８は関連するフラグ２６０２の値を検討し、その値とコア１０のそのときの作動モードとを比較する。好ましい実施例では、ＣＰ１５のドメインステータスレジスタ内のモニタモードによってセットされるドメインビットにより、コア１０のそのときの作動モードが表示される。従って、プロセッサコア１０が安全モードで作動しているときに、対応するフラグ２６０２が安全データであると識別するキャッシュラインのデータアイテムをプロセッサコア１０によってしかアクセスできないように、キャッシュ３８を構成できる。コアが非安全モードで作動中に、キャッシュ３８内の安全データにコアがアクセスしようとする試みの結果、キャッシュ３８はパス２６７０を通してアボート信号を発生する。

【０２６１】

種々の方法でＴＣＭ３６を設定できる。一実施例では、このＴＣＭはキャッシュのように働くように設定でき、この実施例では複数のライン２６１０を含むようになっており、このラインの各々はキャッシュ３８と同じようにラインに関連するフラグ２６１２を有する。次に、ＴＣＭ３６へのアクセスはキャッシュ３８を参照して前に説明したのと全く同じように管理され、ＴＣＭの不一致の結果、ラインフィルプロセスが実行され、この結果、特定ライン２６１０にデータが検索され、パーティションチェッカー２６５６がライン２６１０に関連するフラグ２６１２に記憶するための必要なＳタグの値を発生する。

【０２６２】
別の実施例では、ＴＣＭ３６を外部メモリ５６の延長として設定し、プロセッサによって頻繁に使用されるデータを記憶するように使用できる。その理由は、システムバスを介したＴＣＭへのアクセスは外部メモリへのアクセスよりもかなり高速であるからである。かかる実施例では、ＴＣＭ３６はフラグ２６１２を使用せず、代わりにＴＣＭへのアクセスを制御するのに別の機構が使用される。特に前に説明したように、かかる実施例では特権安全モードで実行中の場合にしかプロセッサによって密結合メモリを制御できないか、または少なくとも非安全モードで実行中のプロセッサによって制御できるかどうかを示すために、特権安全モードで作動中のプロセッサによってセットできる制御フラグを発生することができる。この制御フラグは安全オペレーティングシステムによってセットされ、実際にＴＣＭを特権安全モードで制御するのか、または非安全モードで制御するのかを定める。従って、定めることができる１つのコンフィギュレーションはプロセッサが特権安全作動モードでしかＴＣＭを制御しないようにすることである。かかる実施例では、ＴＣＭ制御レジスタへ試みられる非安全アクセスによって、未定義命令例外が入力される。
【０２６３】
別のコンフィギュレーションでは、プロセッサが非安全作動モードで作動中にこのプロセッサによってＴＣＭを制御できる。かかる実施例ではＴＣＭは非安全アプリケーションによってしか使用されない。ＴＣＭから安全データを記憶したり、ロードすることはできない。従って、安全アクセスの実行中にアドレスがＴＣＭアドレスレンジと一致しているかどうかを見るためのルックアップはＴＣＭ内では実行されない。
【０２６４】
図５７はプロセッサコア１０で作動中の非安全プログラムが仮想アドレス（ステップ２７００）を発生するときの、図５５の装置によって実行される処理を示すフローチャートである。まずステップ２７０５でマイクロＴＬＢ２０６内でルックアップが実行され、この結果、一致が生じた場合、マイクロＴＬＢはステップ２７３０でアクセス許可をチェックする。図５６を参照すると、このプロセスはアクセス許可ロジック２０２によって実行されるものと見なすことができる。
【０２６５】
ステップ２７０５において、マイクロＴＬＢのルックアップで一致が生じた場合、内部に記憶されている非安全デスクリプタの間でもルックアップが主ＴＬＢ２０８で実行される（ステップ２７１０）。この結果一致が生じると、ステップ２７１５でページテーブルウォークプロセスが実行される（このプロセスは図３７を参照して前に詳細に説明した）。その後、ステップ２７２０にて主ＴＬＢがタグの付いた有効な非安全デスクリプタを含むかどうかの判断がされる。ステップ２７１０におけるルックアップが一致を発生した場合、プロセスは直接ステップ２７２０に進む。
【０２６６】
その後、ステップ２７２５において、マイクロＴＬＢに物理アドレスを含むデスクリプタの部分がロードされ、その後、ステップ２７３０でマイクロＴＬＢがアクセス許可をチェックする。
【０２６７】
ステップ２７３０でアクセス許可の違反がないと判断された場合、プロセスはステップ２７４０に進み、ここでパス２３０を通して（図５５に示されているパス２６７０に類似する）プロセッサコアにアボート信号が発生される。しかしながら、違反が検出されないと仮定した場合、ステップ２７４５でアクセスがキャッシュ可能なデータアイテムに関連しているかどうかの判断がされる。関連していなければ、ステップ２７９０で外部アクセスが開始され、外部メモリ５６からのデータアイテムの検索が望まれる。ステップ２７９５にて、パーティションチェッカー２６５６は安全パーティション違反があるかどうか、すなわちプロセッサコア１０が非安全モードで作動中に安全メモリ内のデータアイテムにアクセスすることを望んでいるかどうかを判断する。違反が検出された場合、パーティションチェッカー２６５６はステップ２７７５にてアボート信号を発生する。しかしながら

、安全パーティション違反がない場合、プロセスはステップ２７８５に進み、ここでデータのアクセスが行われる。

【０２６８】

ステップ２７４５にて、リクエストされているデータアイテムがキャッシュ可能であると判断された場合、キャッシュ内でステップ２７５０にてキャッシュルックアップが実行され、一致が検出された場合、キャッシュはステップ２７５５にて安全ラインタグ違反があるかどうかの判断をする。従って、この段階ではキャッシュはデータアイテムを含むキャッシュラインに関連するフラグ２６０２の値を検討し、そのフラグの値とコア１０の作動モードとを比較し、リクエストされているデータアイテムにアクセスする権限がコアにあるかどうかの判断をする。安全ラインタグ違反が検出された場合、プロセスはステップ２７６０に進み、このステップで安全違反フォールトアボート信号がキャッシュ３８により発生され、パス２６７０を通してコア１０へ発生される。しかしながら、ステップ２７５５で非安全ラインタグ違反が検出されない場合、ステップ２７８５でデータアクセスが行われる。

【０２６９】

ステップ２７５０でキャッシュルックアップが実行されるときにキャッシュの一致が生じた場合、ステップ２７６５でキャッシュラインフィルが開始される。ステップ２７７０でパーティションチェッカー２６５６は安全パーティション違反があるかどうかを検出し、違反が検出された場合、ステップ２７７５でアボート信号を発生する。しかしながら、安全パーティション違反が検出されなかった場合、キャッシュラインフィルはステップ２７８０に進み、この結果、進み２７８５でデータアクセスが完了する。

【０２７０】

図５７に示されるように、ステップ２７０５、２７１０、２７１５、２７２０、２７２５、２７３０および２７３５はＭＭＵ内で実行され、ステップ２７４５、２７５０、２７５５、２７６５、２７８０および２７９０はキャッシュにより実行され、ステップ２７７０および２７９５はパーティションチェッカーにより実行される。

【０２７１】

図５８は、コア上で実行中の安全プログラムが仮想アドレスを発生する場合（ステップ２８００）に実行される同様なプロセスを示すフローチャートである。図５８と図５７とを比較すると、ＭＭＵ内で実行されるステップ２８０５～２８３５は図５７を参照して前に説明したステップ２７０５～２７３５に類似することが理解できよう。唯一の差はステップ２８１０にて主ＴＬＢ内で実行されるルックアップが主ＴＬＢ内に記憶された安全デスクリプタに関連して行われ、この結果、ステップ２８２０にて主ＴＬＢがタグの付いた有効な安全デスクリプタを含むことである。

【０２７２】

キャッシュ内でキャッシュは安全ラインタグ違反を待たなくてよい。図５８を参照して説明した実施例では、安全プログラムは安全データおよび非安全データの双方にアクセスできると仮定されている。従って、ステップ２８５０にてキャッシュルックアップ中に一致が生じた場合、プロセスはステップ２８８５にてデータアクセスに直接進む。

【０２７３】

同様に、外部メモリへの外部アクセスが求められた場合（すなわちステップ２８６５または２８９０にて）、パーティションチェッカーはパーティションチェックを実行する必要はない。安全プログラムは安全データまたは非安全データのいずれかにアクセスできると考えられるからである。

【０２７４】

キャッシュ内で実行されるステップ２８４５、２８５０、２８６５、２８８０および２８９０は、図５７を参照して前に説明したステップ２７４５、２７５０、２７６５、２７８０および２７９０に類似している。

【０２７５】

図５９はプロセッサで実行される別のモードおよびアプリケーションを示す。点線は本

発明の一実施例に係わるプロセッサのモニタ中に、異なるモードおよび／またはアプリケーションをどのように分離し、互いにアイソレートできるかを示している。

【０２７６】

生じ得る障害を見つけ、どうして期待するようにアプリケーションが作動しないかを発見するためにプロセッサをモニタする能力は極めて有効であり、多くのプロセッサはかかる機能を提供できる。デバッグおよびトレース機能を含む種々の方法でモニタリングを実行することができる。

【０２７７】

本技術に係わるプロセッサでは、停止デバッグモードおよびモニタデバッグモードを含むいくつかのモードでデバッグを作動させることができる。これらモードは割り込み的であり、そのときに作動しているプログラムをストップさせる。停止デバッグモードにおいて、ブレークポイントまたはウォッチポイントが生じると、コアを停止し、システムの他の部分からアイソレートし、コアはデバッグ状態に入る。入力時にコアは停止され、パイプラインはフラッシングされ、命令はプリフェッチされない。ＰＣはフリーズされ、どの割り込み（ＩＲＱおよびＦＩＱ）も無視される。（ＪＴＡＧシリアルインターフェースを介し）コアの内部ステートだけでなくメモリシステムのステートも検査できる。このステートはそのときのモードを変更したり、レジスタの内容を変えたりすることができるように、プログラムの実行に対して侵入的である。一旦デバッグが終了されると、コアはデバッグＴＡＰ（テストアクセスポート）を通した再スタート命令においてスキャニングをすることによりデバッグステートから脱出する。次にプログラムは実行を再開する。

【０２７８】

モニタデバッグモードでは、ブレークポイントまたはウォッチポイントはコアをアボートモードにし、それぞれプリフェッチベクトルまたはデータアボートベクトルを取り込む。この場合、コアはまだ機能モードにあり、停止デバッグモードと同じように停止されない。アボートハンドラーはデバッガーアプリケーションと通信し、プロセッサおよびコプロセッサのステートまたはダンプメモリにアクセスする。デバッグハードウェアをソフトウェアデバッガーの間をデバッグモニタプログラムがインターフェースするようになっている。デバッグステートおよび制御レジスタＤＳＣＲのビット１１がセットされた場合（後の記載を参照）、割り込み（ＦＩＱおよびＩＲＱ）を禁止できる。モニタデバッグモードでは、ベクトルのキャッチがデータアボートおよびプリフェッチアボートでディスエーブルされ、モニタデバッグモードに対して発生されたアボートの結果としてプロセッサが回復不能なステートになるのを防止できる。モニタデバッグモードはあるタイプのデバッグモードであり、安全な世界と非安全な世界との間の切り替えを監督するモードであるプロセッサのモニタモードには関連していないことに留意すべきである。

【０２７９】

デバッグはある時間におけるプロセッサのステートのスナップショットを提供できる。デバッグはデバッグの開始リクエストが受信されたときに種々のレジスタにおける値を注目することによりこれを行う。これら値はスキャンチェーン（図６７の５４１、５４４）に記録され、これら値はＪＴＡＧコントローラ（図１の１８）を使ってシリアルに出力される。

【０２８０】

コアをモニタする別の方法はトレースによる方法である。このトレースは割り込み的ではなく、コアが作動し続ける際のその後のステートを記録する。このトレースハう１の埋め込みトレースマクロセル（ＥＴＭ）２２、２６で作動する。ＥＴＭはトレースポートを有し、このトレースポートを通してトレース情報が出力され、トレースポートは外部トレースポートアナライザによって分析される。

【０２８１】

本技術の実施例のプロセッサは２つの別個のドメインで作動し、上記実施例ではこれらドメインは安全ドメインと非安全ドメインとを含む。しかしながら、モニタ機能の目的のために、当業者にはこれらドメインは任意の２つのドメインとすることができ、これらド

メインの間ではデータをリークしてはならないことが明らかとなろう。本技術の実施例は２つのドメインの間でのデータのリークを防止することに関係しており、従来システム全体へのアクセスが認められるモニタ機能、例えばデバッグおよびトレースは、ドメイン間でデータがリークされる潜在的な原因となっている。

【０２８２】

安全ドメインまたは世界および非安全ドメインまたは世界の上記例では、安全データは非安全な世界にとって利用できるものであってはならない。更に、デバッグが許可された場合、安全な世界では安全な世界内のデータの一部を制限するかまたは隠すことが有利である。図５７におけるハッシュラインはデータアクセスをセグメント化し、異なるレベルの粒度を提供する可能な方法のいくつかの例を示している。図５９では、ブロック５００によりモニタモードが示されており、このモードはすべてのモードのうちで最も安全なものであり、安全な世界と非安全な世界の間の切り替えを制御するようになっている。モニタモード５００の下にはスーパーバイザーモードがあり、このモードは安全スーパーバイザーモード５１０と非安全スーパーバイザーモード５２０とを含む。次に、アプリケーション５２２および５２４を有する非安全ユーザーモードおよびアプリケーション５１２、５１４および５１６を有する安全ユーザーモードがある。モニタモード（デバッグおよびトレース）は、（ハッシュライン５０１の左に対する）非安全モードをモニタするためにしか制御できない。これとは異なり、非安全ドメインまたは世界および安全ユーザーモードをモニタすること（５０２の下方にある５０１の左および右部分）が認められている。別の実施例では、非安全な世界および安全ユーザードメインで作動する所定のアプリケーションが認められており、この場合、ハッシュライン５０３による更なるセグメント化が行われる。かかる分割は異なるアプリケーションを作動し得る異なるユーザー間での安全データの漏れを防止することに役立つ。ある制御されるケースでは、システム全体をモニタすることが認められる。コアの次の部分は必要とされる粒度に従ってモニタ機能中にアクセスを制御する必要がある。

【０２８３】

デバッグイベントでセットできる４つのレジスタがある。すなわち命令フォールトステータスレジスタ（ＩＦＳＲ）、データフォールトステータスレジスタ（ＤＦＳＲ）、フォールトアドレスレジスタ（ＦＡＲ）および命令フォールトアドレスレジスタ（ＩＦＡＲ）がある。これらレジスタは一部の実施例では安全な世界から非安全な世界に入ったときに、データのリークを防止するためにフラッシングしなければならない。

【０２８４】

ＰＣサンプルレジスタ：　デバッグＴＡＰはスキャンチェーン７を通してＰＣにアクセスできる。安全な世界内でデバッグを行う際に、安全な世界で選択されたデバッグの粒度に応じてその値をマスクすることができる。非安全な世界または非安全な世界プラス安全ユーザーアプリケーションは、安全な世界でコアが実行されている間、ＰＣの値を入手できない。

【０２８５】

ＴＬＢエントリー：　ＣＰ１０を使い、マイクロＴＬＢエントリーを読み出し、主ＴＬＢエントリーを読み出しかつ書き込みできる。主ＴＬＢおよびマイクロＴＬＢのローディングおよびマッチングを制御することもできる。特に安全スレッドを知っているデバッグがＭＭＵ／ＭＰＵの補助を必要とする場合、この種の作動を厳密に制御しなければならない。

【０２８６】

性能モニタ制御レジスタ：　キャッシュの不一致、マイクロＴＬＢの不一致、外部メモリリクエスト、実行される分岐命令などに関する情報を与える。デバッグステートでも非安全な世界はこのデータにアクセスしてはならない。安全な世界でデバッグがディスエーブルされた場合でも、安全な世界でカウンタは作動できるようになっていなければならない。

【０２８７】

キャッシュシステムにおけるデバッギング：　キャッシュシステムではデバッグは非割り込み的でなければならない。重要なことは、キャッシュと外部メモリとの間のコヒーレンスを維持することである。ＣＰ１５を使ってキャッシュを無効にすることができるし、またはキャッシュをすべての領域で強制的に書き込みスルーとすることができる。いずれのケースにおいてもデバッグにおけるキャッシュの挙動の変更を認めることはセキュリティが脆弱になり得ることであり、これは制御しなければならない。

【０２８８】

エンディアン性：　デバッグにアクセスできる非安全な世界または安全ユーザーアプリケーションがエンディアン性を変更できるように認めるべきではない。エンディアン性を変更することは安全カーネルを誤作動させる可能性があり、エンディアン性のアクセスは粒度に従ってデバッグでは禁止される。

【０２８９】

モニタ機能の開始時にコアの一部に対するモニタ機能のアクセスを制御できる。デバッグおよびトレースは種々の方法で初期化される。本技術の実施例は所定の条件下での初期化を認めるだけでコアの所定安全部分に対するモニタ機能のアクセスを制御するようになっている。

【０２９０】

本技術の実施例は次の粒度によるモニタ機能への進入を制限しようとしている。

割り込み的なデバッグと観測可能な（トレース）デバッグとを別々に制御することにより；

安全ユーザーモードだけ、または全安全な世界におけるデバッグの進入を認めることにより；

安全ユーザーモードだけのデバッグを認め、更にスレッドＩＤ（アプリケーションの実行）を考慮することにより、モニタ機能への進入を制限する。

【０２９１】

モニタ機能の開始を制御するためには、機能をどのように開始できるかについて知ることが重要である。図６０はモニタ機能を開始する可能な方法、開始されるモニタ機能のタイプおよびかかる開始命令をプログラム化できるようにする方法を示す表である。

【０２９２】

一般にソフトウェアまたはハードウェアを介し、例えばＪＴＡＧコントローラを介し、これらモニタ命令を入力できる。モニタ機能の開始を制御するために制御値が使用される。これら値は条件に応じたイネーブルビットを含むので、特定の条件が存在した場合、イネーブルビットがセットされた場合にしかモニタは開始が認められない。これらビットはＩＣＥ５３０（図６７参照）にある安全レジスタＣＰ１４（デバッグおよびステータス制御レジスタ、ＤＳＣＲ）に記憶される。

【０２９３】

好ましい実施例では割り込み的および観測可能なデバッグをイネーブル／ディスエーブルするビットが４つ設けられ、これらビットは安全デバッグイネーブルビット、安全トレースイネーブルビット、安全ユーザーモードイネーブルビットおよび安全スレッドアウェアイネーブルビットを含む。これら制御値はモニタ機能のためのある程度の制御可能な粒度を提供するように働き、例えば特定のドメインからのデータの漏れをストップするように働くことができる。図６１は、これらビットの概要およびこれらビットにどのようにアクセスできるかを示している。

【０２９４】

これら制御ビットは安全ドメイン内のレジスタに記憶され、このレジスタへのアクセスは３つの可能性に限定される。ＡＲＭコプロセッサのＭＲＣ／ＭＣＲ命令を介してソフトウェアのアクセスが行われ、これら命令は安全スーパーバイザーモードからしか認められない。これとは異なり、認証コードを使用することにより他のモードからのソフトウェアへのアクセスを行うことができる。更に別の方法は、更にハードウェアのアクセスに関連し、入力ポートを介したＪＴＡＧへの命令の書き込みに関係する。更にモニタ機能の利用

性に関連する制御値を入力するのに使用される他に、この入力ポートはプロセッサの他の機能に関連する制御値を入力するのに使用できる。

【０２９５】

次に、スキャンチェーンおよびＪＴＡＧに関連する別の詳細について説明する。

レジスタロジックセル

どの集積回路（ＩＣ）も２つの種類のロジックから成る。

・組み合わせロジックセル；ＡＮＤ、ＯＲ、ＩＮＶゲートに類似する。かかるゲートまたはかかるゲートの組み合わせを使用し、１つまたは複数の入力信号に従い、ブール演算式を計算するのに使用される。

・レジスタロジックセル；ラッチ、フリップフロップに類似する。かかるセルは信号の値を記憶するのに使用される。図６２は正のエッジでトリガーされるフリップフロップの図である。

クロック信号（ＣＫ）で正のエッジのイベントが生じると、出力（Ｑ）は入力（Ｄ）の値を受信し、そうでない場合、出力（Ｑ）はメモリにその状態を維持する。

【０２９６】

スキャンチェーンセル

テストまたはデバッグのためには、レジスタロジックセルの機能アクセスをバイパスさせ、直接レジスタロジックセルの内容にアクセスすることが必要である。従って、レジスタセルは図６３に示されるようにスキャンチェーンセル内に集積化されている。

機能モードではＳＥ（スキャンイネーブル）がクリアされ、レジスターセルは単一のレジスターセルとして働く。テストまたはデバッグモードではＳＥがセットされ、Ｄ入力の代わりにＳＩ入力（スキャン入力）から入力データが出ることができる。

【０２９７】

スキャンチェーン

図６４に示されるように、スキャンチェーンにおいてすべてのスキャンチェーンセルがチェーン化される。

機能モードではＳＥがクリアされ、すべてのレジスターセルに正常にアクセスすることができ、回路の他のロジックと相互作用することができる。テストまたはデバッグモードではＳＥがセットされ、スキャンチェーン内ですべてのレジスタが互いにチェーン化される。第１スキャンチェーンセルからデータが出ることができ、各クロックサイクルのカデンスにおいて他のスキャンチェーンシェルを通過するようにシフトできる。レジスタの内容を見るためにデータをシフトして出すこともできる。

【０２９８】

ＴＡＰコントローラ

いくつかのスキャンチェーンを取り扱うのにデバッグＴＡＰコントローラが使用される。ＴＡＰコントローラは特定のスキャンチェーンを選択できる。例えば「スキャンイン」および「スキャンアウト」信号を特定のスキャンチェーンへ送り、データはチェーン内にスキャンし、シフトし、またはスキャンアウトできる。このＴＡＰコントローラはＪＴＡＧポートインターフェースにより外部から制御される。図６５はＴＡＰコントローラを略図で示す。

【０２９９】

ＪＴＡＧの選択可能なディスエーブルスキャンチェーンセル

セキュリティ上の理由から、デバッグモードまたはテストモードでもスキャンチェーンによって一部のレジスタにはアクセスできないようになっている。ＪＡＤＩ（ＪＴＡＧアクセスディスエーブル）と称される新しい入力信号は集積回路内のスキャンチェーン構造を変えることなく、スキャンチェーン全体からスキャンチェーンセルをダイナミックまたはスタティックに除去することを認めることができる。図６６Ａおよび６６Ｂはこの入力を略図で示す。

【０３００】

機能モードまたはテストモード、またはデバッグモードのいずれかのときにＪＡＤＩが

アクティブでない（ＪＡＤＩ＝０）場合、スキャンチェーンは通常のように働く。ＪＡＤＩがアクティブ（ＪＡＤＩ＝１）であり、テストまたはデバッグモードとなっている場合（設計者によって選択された）一部のスキャンチェーンセルをスキャンチェーン構造から「除く」ことができる。同じ数のスキャンチェーンセルを維持するにはＪＴＡＧ選択ディスエーブルスキャンチェーンセルはバイパスレジスタを使用する。スキャンアウト（ＳＯ）とスキャンチェーンセル出力（Ｑ）とは異なることに留意されたい。

【０３０１】

図６７はＪＴＡＧの部品の一部を含むプロセッサを略図で示す。正常な動作では命令メモリ５５０はコアと通信し、所定の状況でレジスタＣＰ１４とも通信し、制御値をリセットする。このことは一般に安全スーパーバイザーモードからしか認められない。

【０３０２】

デバッグが開始されると、デバッグＴＡＰ５８０を介して命令が入力され、この命令がコアを制御する。デバッグ内のコアはステップごとのモードで作動する。デバッグＴＡＰは（ＪＡＤＩピンとして示されたＪＳＤＡＥＮピンに入力されるアクセス制御信号、すなわち図４５におけるＪＴＡＧ　ＡＣＣＥＳＳ　ＤＩＳＡＢＬＥ　ＩＮＰＵＴに応じて）コアを介し、ＣＰ１４にアクセスし、このように制御値もリセットできる。

【０３０３】

アクセス制御信号ＪＳＤＡＥＮによりデバッグＴＡＰ５８０を介したＣＰ１４スペースレジスタへのアクセスが制御される。この制御はアクセス、特に書き込みアクセスを認めるためにＪＳＤＡＥＮをハイレベルに設計しなければならないようになっている。プロセッサ全体が証明中のボード段階でＪＳＤＡＥＮはハイレベルにセットされ、システム全体でデバッグがイネーブルされる。一旦システムがチェックされると、ＪＳＤＡＥＮピンをアースに接続できる。このことは、安全モードのデバッグをイネーブルする制御値へのアクセスは、そのときにデバッグＴＡＰ５８０を介して利用できないことを意味する。一般に製造モードにおけるプロセッサはアースに接続されたＪＳＤＡＥＮを有する。従って、制御値へのアクセスは命令メモリ５５０を介してソフトウェアルートによってしか利用できない。認証コードが当てられることを条件に、このルートを介したアクセスは安全スーパーバイザーモードまたは別のモードに制限される（図６８参照）。

【０３０４】

デフォルトによるデバッグ（割り込み的または観測可能なトレース）は非安全な世界においてしか利用できないことに留意すべきである。安全な世界でデバッグを利用できるようにするには、制御値イネーブルビットをセットする必要がある。

【０３０５】

この利点は非安全な世界において作動するように常にユーザーによってデバッグを開始できることである。従って、一般にデバッグの際に安全な世界へのアクセスはユーザーには利用できないが、このことは多くの場合には問題とならない。その理由は、この安全な世界へのアクセスは限定されており、利用できるようになる前にボード段階で安全な世界が完全に証明されているからである。従って、多くの場合、安全な世界のデバッグは不要となることが予測される。必要な場合、安全スーパーバイザーモードはＣＰ１４を書き込むソフトウェアルートによりデバッグを開始できる。

【０３０６】

図６８はデバッグ初期化の制御を略図で示す。この図では、コア６００の一部は記憶素子６０１（この素子は前に説明したようにＣＰ１５レジスタでよい）を含み、この素子内にシステムが安全な世界であるか否かを表示する安全ステータスビットＳが記憶されている。コア６００はプロセッサが作動しているときのモード、例えばユーザーモードを表示するビットを含むレジスタ６０２と、現在コアで作動中のアプリケーションまたはスレッドを識別するコンテクスト識別子を発生するレジスタ６０３も含む。

【０３０７】

ブレークポイントに達すると、レジスタ６１１に記憶されたブレークポイントとレジスタ６１２に記憶されたコアのアドレスとを比較するコンパレータ６１０が制御ロジック６

２０へ信号を送る。制御ロジック６２０は安全ステートＳ、モード６０２およびスレッド（コンテクスト識別子）６０３を見て、これらと制御値およびレジスタＣＰ１４上に記憶された条件インジケータとを比較する。システムが安全な世界内で作動していない場合、「デバッグ進入」信号が６３０で出力される。しかしながら、システムが安全な世界で作動する場合、制御ロジック６２０はモード６０２を見る。このモードがユーザーモードであれば、ユーザーモードイネーブルビットおよびデバッグイネーブルビットがセットされているかどうかを見るためにチェックする。そうである場合、スレッドアウェアビットが初期化されていないことを条件に、デバッグが初期化される。これまでの説明は制御値の階層的性質を示すものである。

【０３０８】

図６８にはレジスタＣＰ１４に記憶された制御値を、どのように安全スーパーバイザーモードだけから変更できるのか（この実施例ではプロセッサは製造段階にあり、ＪＳＤＡＥＮはアースに接続されている）と共に、モニタ制御のスレッドアウェア部分も略図で示されている。安全ユーザーモードから認証コードを使って安全スーパーバイザーモードを入力し、次にＣＰ１４に制御値をセットできる。

【０３０９】

スレッドに対してデバッグが可能なことをスレッドコンパレータ６４０が示すことを条件に、ブレークポイントに達したことをアドレスコンパレータ６１０が表示したときに、制御ロジック６２０は「デバッグ進入」信号を出力する。このことは、ＣＰ１４にスレッドアウェア初期化ビットがセットされることを仮定している。ブレークポイントの後でスレッドアウェア初期化ビットがセットされあ場合、アドレスおよびコンテクスト識別子がブレークポイントおよび許容できるスレッドインジケータ内に表示された識別子に一致する場合にしか、デバッグまたはトレースに入ることはできない。モニタ機能の開始後にコンテクスト識別子がコンパレータ６４０により許容されたスレッドとして検出される間にしか、診断データの捕捉が続かない。アプリケーションの作動が認められない作動であることをコンテクスト識別子が示すと、診断データの捕捉が抑制される。

【０３１０】

好ましい実施例では、粒度内にある程度の階層性があることに留意すべきである。実際に安全デバッグまたはトレースイネーブルビットが頂部にあり、その後、安全ユーザーモードイネーブルビットが続き、最後に安全スレッドアウェアイネーブルビットがある。これについては図６９Ａおよび６９Ｂ（以下参照）に示されている。

【０３１１】

「デバッグおよびステータス」レジスタ（ＣＰ１４）内に保持された制御値はドメイン、モードおよび実行スレッドに従って安全デバッグ粒度を制御する。この値はスーパーバイザーモードの頂部にある。「デバッグおよびステータス制御」レジスタＣＰ１４が構成されると、このレジスタは安全スーパーバイザーモードまでなり、対応するブレークポイント、ウォッチポイントなどをプログラムし、コアをデバッグステートにする。

【０３１２】

図６９Ａは割り込み的デバッグのための安全デバッグ粒度の概要を示す。リセット時のデフォルト値は灰色で示される。

【０３１３】

観測可能なデバッグに関するデバッグ粒度についても同じである。図６９Ｂは、この場合における安全デバッグ粒度の概要を示し、リセット時のデフォルト時も灰色で示される。

【０３１４】

安全ユーザーモードデバッグイネーブルビットおよび安全スレッドアウェアデバッグイネーブルビットは割り込み的なデバッグと観測可能なデバッグの双方に対して共通に使用される。

【０３１５】

レジスタＣＰ１４にはスレッドアウェア初期化ビットが記憶され、このビットはアプリ

ケーションによる粒度が必要であるかどうかを表示する。スレッドアウェアビットが初期化されている場合、制御ロジックはアプリケーション識別子またはスレッド６０３がスレッドアウェア制御値に表示されたものであるかどうかを更にチェックする。そうである場合、デバッグが初期化される。ユーザーモードビットまたはデバッグイネーブルビットのいずれかがセットされないか、またはスレッドアウェアビットがセットされ、実行中のアプリケーションがスレッドアウェア制御値で表示されたアプリケーションでない場合、ブレークポイントが無視され、コアはこれまで行われてきたことを続け、デバッグは初期化されない。

【０３１６】

モニタ機能の初期化を制御する他に、モニタ機能中の診断データの捕捉も同じように制御できる。これを行うために、コアは制御値の双方、すなわちレジスタＣＰ１４に記憶されているイネーブルビットおよびモニタ機能の作動中に関係する条件を検討し続けなければならない。

【０３１７】

図７０は実行中のモニタ機能の粒度を略図で示す。この場合、領域Ａは診断データを捕捉することが許可された領域に関係し、領域Ｂは診断データを捕捉できないことをＣＰ１４内に記憶された制御値が表示する領域に関係する。

【０３１８】

従って、デバッグが実行中であり、プログラムが領域Ａ内で作動しているときに、デバッグ中に診断データがステップごとに出力される。作動が領域Ｂに切り替わり、ここで診断データの捕捉が認められていない場合、デバッグはステップごとに進行するのではなく、自動的に進行し、データは捕捉されない。このことは、プログラムの作動が領域Ａに再進入するまで続き、再進入後、診断データの捕捉が再びスタートし、デバッグがステップごとに続けられる。

【０３１９】

上記実施例では、安全ドメインがイネーブルされた場合、ＳＭＩ命令は常に原子事象として見られ、診断データの捕捉が抑制される。

【０３２０】

更に、スレッドアウェア初期化ビットがセットされた場合、アプリケーションに対する作動中のモニタ機能の粒度も生じる。

【０３２１】

観測可能なデバッグまたはトレースに関し、このことはＥＴＭによって行われ、完全にデバッグから独立している。トレースがイネーブルされるとＥＴＭが通常のように作動し、ディスエーブルされると、ＥＴＭは安全な世界内にトレースを隠すか、または粒度に依存する安全な世界部分が選択される。イネーブルされていないときの安全ドメイン内のＥＴＭ捕捉およびトレース診断データを回避する１つの方法は、Ｓビットがハイレベルのときにトレースをストールすることである。このことは、ＳビットとＥＴＭＰＷＲＤＯＷＮ信号とを組み合わせることによって行うことができるので、コアが安全な世界に入るときの最終の値にＥＴＭの値が保持される。従って、ＥＴＭはＳＭＩ命令をトレースしなければならず、コアが非安全な世界に戻るまでストールしなければならない。従って、ＥＴＭは非安全活動を見るにすぎない。

【０３２２】

異なるモニタ機能の一部およびそれらの粒度の概要について下記に示す。

ボード段階における割り込み的デバッグ

ボード段階においてＪＳＤＡＥＮピンが結ばれていないときに、ブートセッションをスタートする前のどこでも、デバッグをイネーブルする能力がある。同様に、安全スーパーバイザーモードにある場合、同様な権利を有する。

【０３２３】

フォールトデバッグモードでデバッグを初期化する場合、すべてのレジスタ（非安全レジスタバンクおよび安全レジスタバンク）にアクセスでき、デバッグを制御するための専

用のビットを除き、メモリ全体をダンプできる。どのモードからもどのドメインからもデバッグフォールトモードに入ることができる。安全メモリまたは非安全メモリにブレークポイントおよびウォッチポイントをセットできる。デバッグステートでは、ＭＣＲ命令を介し、Ｓビットを変えるだけで安全な世界に入ることができる。

【０３２４】

安全例外が生じたときにデバッグモードに入ることができるので、

ＳＭＩベクトルトラッピングイネーブル

安全データアボートベクトルトラッピングイネーブル

安全プリフェッチアボートベクトルトラッピングイネーブル

安全未定義ベクトルトラッピングイネーブル

である新しいビットによりベクトルトラップレジスタが拡張される。

【０３２５】

モニタデバッグモードでは非安全な世界でＳＭＩがコールされたときでもデバッグをどこでも認める場合、ステップごとのデバッグで安全な世界に入ることができる。安全ドメインでブレークポイントが生じると、安全アボートハンドラーは安全レジスタバンクおよび安全メモリをダンプするように作動できる。

安全な世界および非安全な世界における２つのアボートハンドラーは（ＰＣを制御する関連するデバッグでの）デバッガーウィンドーが、安全な世界および非安全な世界の双方におけるレジスタのステートを示すことができるように、それらの情報をデバッガーアプリケーションへ与える。

【０３２６】

図７１Ａは、コアがモニタデバッグモードに構成され、デバッグが安全な世界内でイネーブルされたときに何が生じるかを示している。図７１Ｂはコアがモニタデバッグモードで構成され、デバッグが安全な世界でディスエーブルされたときに何が生じるかを示している。次に後者のプロセスについて説明する。

【０３２７】

製造段階における割り込み的デバッグ

ＪＳＤＡＥＮが結ばれ、デバッグが非安全な世界に制限される製造ステージでは、安全スーパーバイザーが判断をしない場合、図７１Ｂに示されたテーブルは、何が生じるかを示す。この場合、ＳＭＩは常に原子命令と見なすべきであるので、デバッグステートに入る前に常に安全機能が終了する。

【０３２８】

デバッグ停止モードに入るには次の制限を受ける。

非安全な世界に限り、外部デバッグリクエストまたは内部デバッグリクエストが考慮される。安全な世界にある間、ＥＤＢＧＲＱ（外部デバッグリクエスト）がアサートされた場合、一旦安全機能が終了すれば、コアはデバッグ停止モードに入り、非安全な世界でコアが戻される。

【０３２９】

安全メモリにブレークポイントまたはウォッチポイントをプログラムすることは全く効果がなく、プログラムされたアドレスが一致すると、コアが停止される。

ベクトルトラップレジスタ（この詳細を下記に示す）は非安全例外だけに関係する。前に説明したすべての拡張されたトラッピングイネーブルビットは効果がない。

【０３３０】

一旦デバッグ停止モードに入ると次の制限が適用される。

安全デバッグをイネーブルしなければ、安全な世界への進入を強制するのにＳビットを変更できない。

安全スーパーバイザーモードだけでしかデバッグが許可されていない場合、モードビットを変えることはできない。

安全デバッグを制御する専用ビットを変えることはできない。

（システムスピードアクセスにより）ＳＭＩをロードし、実行する場合、安全機能が完

全に実行された場合にしかコアはデバッグステートに再進入しない。

【０３３１】

モニタデバッグモードでは安全な世界でモニタを行うことができないので、安全アボートハンドラーはデバッグモニタプログラムをサポートする必要はない。非安全な世界ではステップバイステップが可能であるが、ＳＭＩが実行されるときは常に安全機能が完全に実行される。換言すれば、ＸＷＳＩの「ステップオーバー」しか認められないが、他のすべての命令では「ステップイン」および「ステップオーバー」が可能である。こうしてＸＷＳＩは原子命令と見なされる。

【０３３２】

安全デバッグが一旦ディスエーブルされた場合、次の制限がある。

モニタモードに入る前：

非安全な世界ではブレークポイントおよびウォッチポイントしか考慮しない。ビットＳがセットされた場合、ブレークポイントおよびウォッチポイントがバイパスされる。安全メモリではブレークポイント／ウォッチポイントは効果を有しないので、セキュリティの問題ではないＭＣＲ／ＭＲＣ（ＣＰ１４）によってもウォッチポイントユニットにアクセス可能であることに留意されたい。

【０３３３】

ＢＫＰＴは通常、ブレークポイントをセットした命令を変えるのに使用される。これには非安全モードでしか可能にならないＢＫＰＴ命令によりメモリ内でこの命令をオーバーライトすると仮定する。

ベクトルトラップレジスタは非安全例外にしか関係しない。前に説明したすべての拡張トラッピングイネーブルビットは効果を有しない。データアボートおよびプリフェッチアボートイネーブルビットはプロセッサが強制的に回復不能なステートになるのを防止するためにディスエーブルしなければならない。

ＪＴＡＧを介し、停止モードに対するのと同じ制限を有する（Ｓビットは変更できない、など）。

【０３３４】

一旦モニタモード（非安全アボートモード）となる。

非安全アボートハンドラーは非安全な世界をダンプでき、安全バンクレジスタだけでなく安全メモリにも視覚性を有しない。

原子ＳＭＩ命令により安全機能を実行。

安全な世界に入ることを強制するためにＳビットを変えることはできない。

安全スーパーバイザーモードでしかデバッグが許可されない場合、モードビットを変えることはできない。

【０３３５】

外部デバッグリクエスト（ＥＤＢＧＲＱ）が生じた場合、次のことに留意されたい。

非安全な世界ではコアはそのときの命令を終了し、即座に（フォールトモードで）デバッグステートに入る。

安全な世界ではコアはそのときの機能を終了し、非安全な世界に戻ったときにデバッグステートとなる。

【０３３６】

新しいデバッグ条件はコアハードウェアを若干変更することを意味する。Ｓビットは注意深く制御しなければならず、セキュリティの理由からスキャンチェーンに安全ビットを挿入してはならない。

【０３３７】

要約すれば、デバッグでは安全スーパーバイザーモードにおいてデバッグをイネーブルするときしかモードビットを変更できない。システムを変更する（ＴＢＬエントリーなどを変更する）ことにより、すべての安全な世界にアクセスするために安全ドメイン内のデバッグにだれもがアクセスしないようにする。このように、各スレッドは自己のコードしかデバッグできない。安全カーネルは安全状態に維持しなければならない。従って、コア

が非安全な世界で作動中にデバッグに入るときには、以前と同じようにモードビットしか変更できない。

【０３３８】

この技術の実施例は新しいベクトルトラップレジスタを使用するものである。このレジスタ内のビットの１つはハイレベルにセットされ、対応するベクトルがトリガーされた場合、プロセッサはあたかもブレークポイントが対応する例外ベクトルからフェッチされた命令でセットされたかのように、デバッグステートに入る。デバッグ制御レジスタにおける「安全世界イネーブル内デバッグ」ビットの値に従い、これらビットのふるまいは異なっていてもよい。

【０３３９】

新しいベクトルトラップレジスタは次のビット、すなわちＤ＿ｓ＿アボート、Ｐ＿ｓ＿アボート、Ｓ＿ｕｎｄｅｆ、ＳＭＩ、ＦＩＱ、ＩＲＱ、Ｕｎａｌｉｇｎｅｄ、Ｄ＿アボート、Ｐ＿アボート、ＳＷＩおよびＵｎｄｅｆを含む。

・Ｄ＿ｓ＿アボートビット：　安全な世界でデバッグがイネーブルされるとき、およびデバッグがデバッグフォールトモードに構成されたときにしか、このビットをセットしてはならない。モニタデバッグモードではこのビットは決してセットしてはならない。安全な世界内のデバッグがディスエーブルされた場合、このビットはどんな値にも影響しない。

・Ｐ＿ｓアボートビット：　Ｄ＿ｓ＿アボートビットと同じである。

・Ｓ＿Ｕｎｄｅｆビット：　安全な世界でデバッグがイネーブルされるときにしか、このビットをセットしてはならない。安全な世界におけるデバッグがディスエーブルされた場合、このビットは値がどんな値でも影響を与えない。

・ＳＭＩビット：　安全な世界でデバッグがイネーブルされたときにしか、これをセットしてはならない。安全な世界におけるデバッグがディスエーブルされた場合、このビットはどんな値であっても影響を与えない。

・ＦＩＱ、ＩＲＱ、Ｄ＿アボート、Ｐ＿アボート、ＳＷＩ、Ｕｎｄｅｆビット：　これらビットは非安全例外に対応するので、安全な世界におけるデバッグがディスエーブルされた場合でも有効である。Ｄ＿アボートおよびＰ＿アボートはモニタモードでハイレベルにアサートしてはならない。

・リセットビット：　リセットが生じたときに安全な世界に入る場合、安全な世界におけるデバッグがイネーブルされたときにしかこのビットは有効とならず、有効でない場合、影響を与えない。

【０３４０】

以上で本明細書に本発明の特定の実施例について説明したが、本発明はこれら実施例だけに限定されるものでなく、本発明の範囲内で大きな変形および追加を行うことができることは明らかである。例えば本発明の範囲から逸脱することなく、従属請求項の特徴事項と独立請求項の特徴事項とを種々に組み合わせることができる。

【図面の簡単な説明】

【０３４１】

【図１】本発明の好ましい実施例に係わるデータ処理装置を略図で示すブロック図である。

【図２】非安全ドメインおよび安全ドメインで作動する異なるプログラムを略図で示す。

【図３】異なるセキュリティドメインに関連する処理モードのマトリックスを略図で示す。

【図４】処理モードとセキュリティドメインとの間の異なる関係を略図で示す。

【図５】処理モードとセキュリティドメインとの間の異なる関係を略図で示す。

【図６】処理モードに応じたプロセッサのレジスタバンクのプログラマーのモデルを示す。

【図７】安全ドメインおよび非安全ドメインのための別個のレジスタバンクを提供する一例を示す。

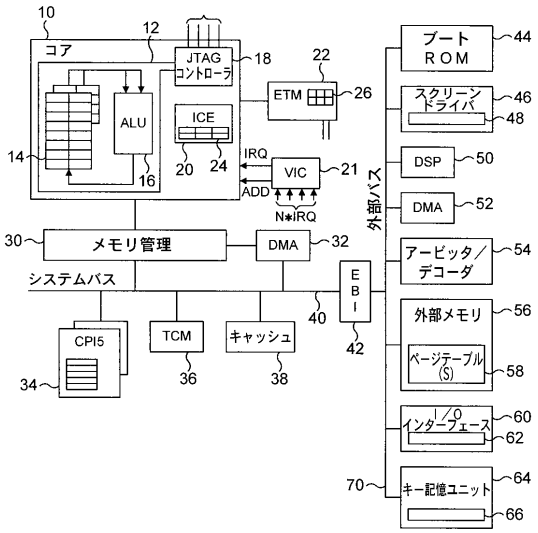
【図８】別個のモニタモードを介して行われるセキュリティドメイン間の切り替えを有する複数の処理モードを略図で示す。
【図９】モード切り替えソフトウェア割り込み命令を使ったセキュリティドメインの切り替えのためのシナリオを略図で示す。
【図１０】非安全割り込みリクエストおよび安全割り込みリクエストをどのようにシステムで処理できるかの一例を略図で示す。
【図１１Ａ】図１０に従った非安全割り込みリクエスト処理の一例を略図で示す。
【図１１Ｂ】図１０に従った安全割り込み略図で処理の一例を略図で示す。
【図１２】図１０に示された信号と比較された非安全割り込みリクエスト信号および安全割り込みリクエスト信号の取り扱いを行うための別の方式を示す。
【図１３Ａ】図１２に示された方式に係わる非安全割り込みリクエストを取り扱うためのシナリオ例を示す。
【図１３Ｂ】図１２に示された方式に従う、安全割り込みリクエストを取り扱うためのシナリオ例を示す。
【図１４】ベクトル割り込みテーブルの一例である。
【図１５】異なるセキュリティドメインに関連する多数のベクトル割り込みテーブルを略図で示す。
【図１６】例外制御レジスタを略図で示す。
【図１７】セキュリティドメインの設定を変えるように、処理ステータスレジスタを変えようとする命令がどのように別個のモード変更例外を発生し、次にこの例外がモニタモードおよびモニタプログラムの実行へ入ることをトリガーするかを示すフローチャートである。
【図１８】モニタモードのタスクに割り込みが行われる複数のモードで作動するプロセッサの制御のスレッドを略図で示す。
【図１９】複数のモードで作動するプロセッサの制御の異なるスレッドを略図で示す。
【図２０】モニタモードで割り込みがイネーブルされる複数のモードで作動するプロセッサの制御の別のスレッドを略図で示す。
【図２１】別の実施例に係わる安全ドメインと非安全ドメインとの切り替えをするためのシナリオおよび異なる処理モードの図を示す。
【図２２】別の実施例に係わる安全ドメインと非安全ドメインとの切り替えをするためのシナリオおよび異なる処理モードの図を示す。
【図２３】別の実施例に係わる安全ドメインと非安全ドメインとの切り替えをするためのシナリオおよび異なる処理モードの図を示す。
【図２４】従来のＡＲＭコアに安全処理オプションを追加する概念を略図で示す。
【図２５】安全ドメインおよび非安全ドメインおよびリセットを有するプロセッサを略図で示す。
【図２６】ソフトウェア疑似割り込みを使用するサスペンドされたオペレーティングシステムへ処理リクエストを送ることを略図で示す。
【図２７】ソフトウェア疑似割り込みを介したサスペンドされたオペレーティングシステムへの処理リクエストの送りの別の例を略図で示す。
【図２８】図２６および２７で発生されたタイプのソフトウェア疑似割り込みの受信時に実行される処理を略図で示すフローチャートである。
【図２９】非安全オペレーティングシステムにより行われる、可能なタスクの切り替えをトラッキングするために、安全オペレーティングシステムが従うタスクを略図で示す。
【図３０】非安全オペレーティングシステムにより行われる、可能なタスクの切り替えをトラッキングするために、安全オペレーティングシステムが従うタスクを略図で示す。
【図３１】図２９および３０の安全オペレーティングシステムにおけるコールの受信時に実行される処理を略図で示すフローチャートである。
【図３２】異なる割り込みを別のオペレーティングシステムで取り扱うことができる、多数のオペレーティングシステムを有するシステムで生じ得る、割り込み優先権の反転の問

題を略して示す図である。
【図３３】図３２に示された問題を解消するためにスタブ割り込みハンドラーを使用することを略して示す図である。
【図３４】異なるオペレーティングシステムを使ってサービスされる割り込みによって割り込みを行うことができるかどうかに応じて、異なるタイプおよび優先権の割り込みをどのように取り扱うことができるかを略図で示す。
【図３５】プロセッサがモニタモードで作動しているときに、モニタモード固有のプロセッサコンフィギュレーションデータにより、どのようにプロセッサコンフィギュレーションデータが無効とされるかを示す。
【図３６】本発明の一実施例に係わる安全ドメインと非安全ドメインとの切り替え時にプロセッサコンフィギュレーションデータがどのように切り替えられるかを示すフローチャートである。
【図３７】メモリへのアクセスを制御するために本発明の一実施例で使用されるメモリ管理ロジックを示す図である。
【図３８】メモリへのアクセスを制御するのに使用される本発明の第２実施例のメモリ管理ロジックを示すブロック図である。
【図３９】仮想アドレスを指定するメモリアクセスリクエストを処理するために、メモリ管理ロジック内で本発明の一実施例で実行されるプロセスを示すフローチャートである。
【図４０】物理アドレスを指定するメモリアクセスリクエストを処理するために、メモリ管理ロジック内で本発明の一実施例で実行されるプロセスを示すフローチャートである。
【図４１】メモリアクセスリクエストが発生するデバイスが非安全モードで作動するときに、好ましい実施例のパーティションチェッカーが安全メモリ内の物理アドレスへのアクセスを防止するのにどのように作動するかを略図で示す。
【図４２】本発明の好ましい実施例における、非安全ページテーブルおよび安全ページテーブルの双方の使用を示す図である。
【図４３】好ましい実施例の主変換ルックアサイドバッファ（ＴＬＢ）内で使用されるフラグの２つのフォームを示す図である。
【図４４】本発明の一実施例におけるブートステージ後に、メモリをどのように分割できるかを示す。
【図４５】本発明の一実施例に係わるブートパーティションの性能に従う、メモリ管理ユニットによる非安全メモリのマッピングを示す。
【図４６】本発明の一実施例に係わる非安全アプリケーションと共に、安全アプリケーションがメモリを共用できるように、メモリの一部の権利をどのように変更できるかを示す。
【図４７】本発明の一実施例に係わるデータ処理装置の外部バスにデバイスをどのように接続できるかを示す。
【図４８】本発明の第２実施例に従ってデバイスをどのように外部バスに結合できるかを示すブロック図である。
【図４９】１つの組のページテーブルを使用する実施例における物理メモリの配置を示す。
【図５０Ａ】中間アドレスを介した仮想アドレスから物理アドレスへの変換を実行するのに２つのＭＭＵを使用する装置を示す。
【図５０Ｂ】中間アドレスを介した仮想アドレスから物理アドレスへの変換を実行するのに２つのＭＭＵを使用する別の装置を示す。
【図５１】安全ドメインと非安全ドメインの双方に対する物理アドレス空間と中間アドレス空間の間の対応を例として示す。
【図５２】第２ＭＭＵに関連するページテーブルの操作による安全ドメインと非安全ドメインとの間のメモリ領域のスワップを示す。
【図５３】仮想アドレスから物理アドレスへの変換を判断するのに、主ＴＬＢにおける不一致が例外を呼び出す、単一のＭＭＵを使用した実現例を示す一実施例である。

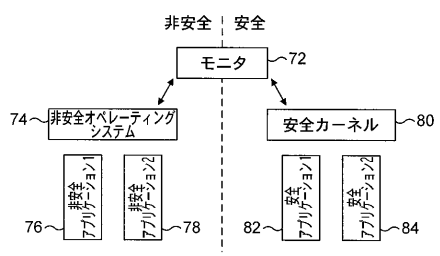
【図５４】図５３のＭＭＵの主ＴＬＢで不一致が生じた場合に発生される例外を扱うために、プロセッサコアによって実行されるプロセスを示すフローチャートである。
【図５５】個々のキャッシュラインに記憶されたデータが安全データであるか、または非安全データであるかに関する情報がキャッシュに提供される一実施例のデータ処理装置内に設けられた部品を示すブロック図である。
【図５６】図５５に示されたメモリ管理ユニットの構造を示す。
【図５７】非安全メモリアクセスリクエストを処理するために、図５５のデータ処理装置内で実行される処理を示すフローチャートである。
【図５８】安全メモリアクセスリクエストを処理するために、図５５のデータ処理装置内で実行される処理を示すフローチャートである。
【図５９】プロセッサで実行されるアプリケーションおよび異なるモードのためのモニタ機能の生じ得る粒度を略図で示す。
【図６０】異なるモニタ機能を開始する可能な方法を示す。
【図６１】異なるモニタ機能の利用可能性を制御するための制御値のテーブルを示す。
【図６２】正のエッジでトリガーされるフリップフロップの図を示す。
【図６３】スキャンチェーンセルを示す。
【図６４】スキャンチェーンにおける複数のスキャンチェーンセルを示す。
【図６５】デバッグＴＡＰコントローラを示す。
【図６６Ａ】ＪＡＤＩ入力を有するデバッグＴＡＰコントローラを示す。
【図６６Ｂ】バイパスレジスタを有するスキャンチェーンセルを示す。
【図６７】コア、スキャンチェーンおよびデバッグステータスおよび制御レジスタを含むプロセッサを略図で示す。
【図６８】デバッグまたはトレース初期化を制御する要素を略図で示す。
【図６９Ａ】デバッグ粒度の概要を示す。
【図６９Ｂ】デバッグ粒度の概要を示す。
【図７０】実行中のデバッグの粒度を略図で示す。
【図７１Ａ】デバッグが安全な世界でイネーブルされているときのモニタデバッグを示す。
【図７１Ｂ】デバッグがイネーブルされていないときのモニタデバッグを示す。
【符号の説明】
【０３４２】
１０　プロセッサコア
１４　レジスタバンク
１６　演算論理ユニット
１８　ＪＴＡＧコントローラ
２０　回路内エミュレータ
２１　割り込みコントローラ
２２　埋め込みトレースモジュール
２４　レジスタ
２６　制御レジスタ
３０　メモリ管理ロジック
３２　ダイレクトメモリアクセスコントローラ
３４　制御レジスタ
３６　ＴＣＭ
３８　キャッシュ
４０　システムバス
４２　外部バスインターフェース
４４　ブートＲＯＭ
４６　スクリーンドライバ
５０　デジタル信号プロセッサ

５２　ダイレクトメモリアクセスコントローラ
５４　アービッタ／デコーダ
５６　外部メモリ
６０　Ｉ／Ｏインターフェース
６４　キー記憶ユニット

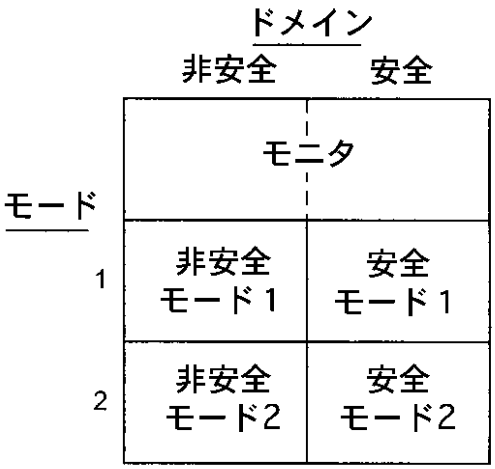
【図１】



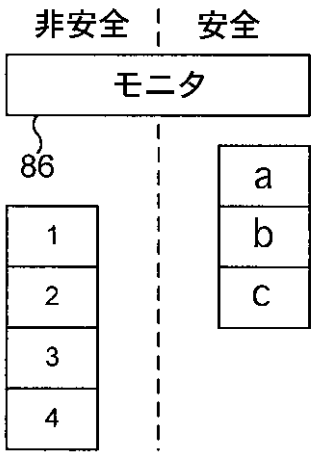
【図２】



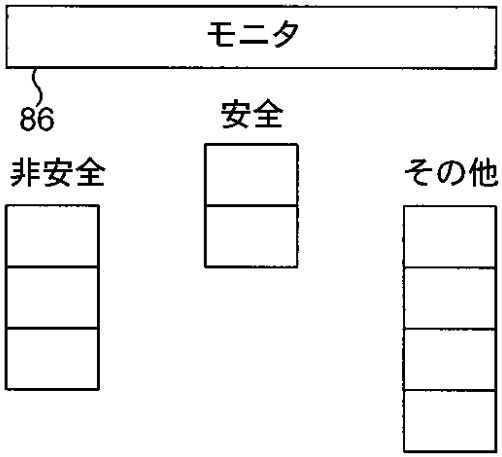
【図３】



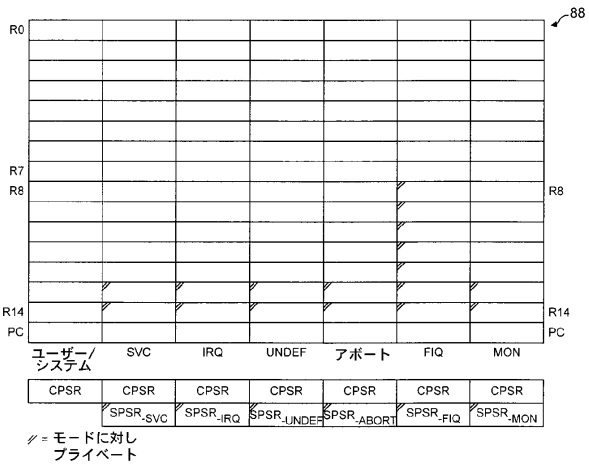
【図４】


非安全
安全
モニタ
86
1
2
3
4
a
b
c

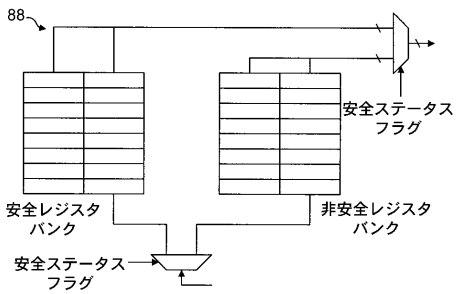
【図５】


モニタ
86
安全
非安全
その他

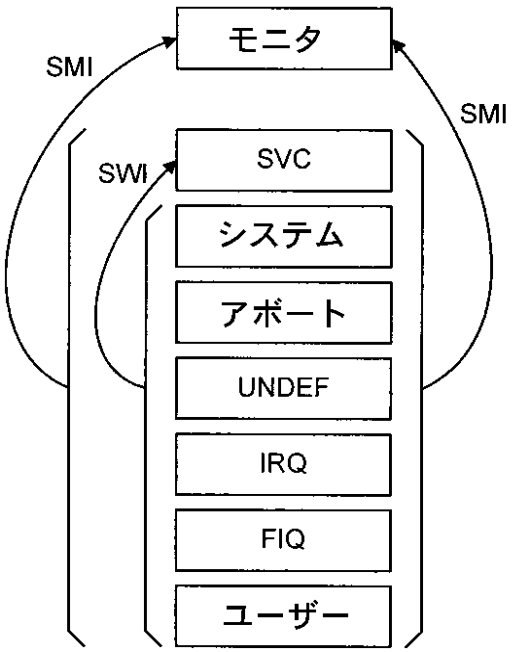
【図６】


88
R0
R7
R8
R14
PC
ユーザー/システム
SVC
IRQ
UNDEF
アボート
FIQ
MON
CPSR
SPSR_SVC
SPSR_IRQ
SPSR_UNDEF
SPSR_ABORT
SPSR_FIQ
SPSR_MON
= モードに対しプライベート

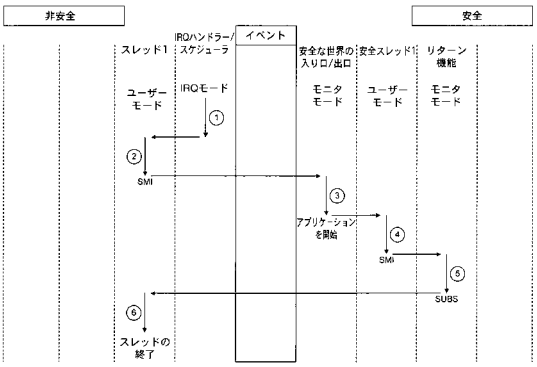
【図７】


88
安全ステータスフラグ
安全レジスタバンク
非安全レジスタバンク
安全ステータスフラグ

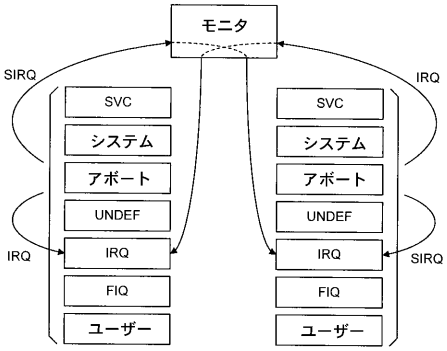
【図８】


モニタ
SMI
SMI
SWI
SVC
システム
アボート
UNDEF
IRQ
FIQ
ユーザー

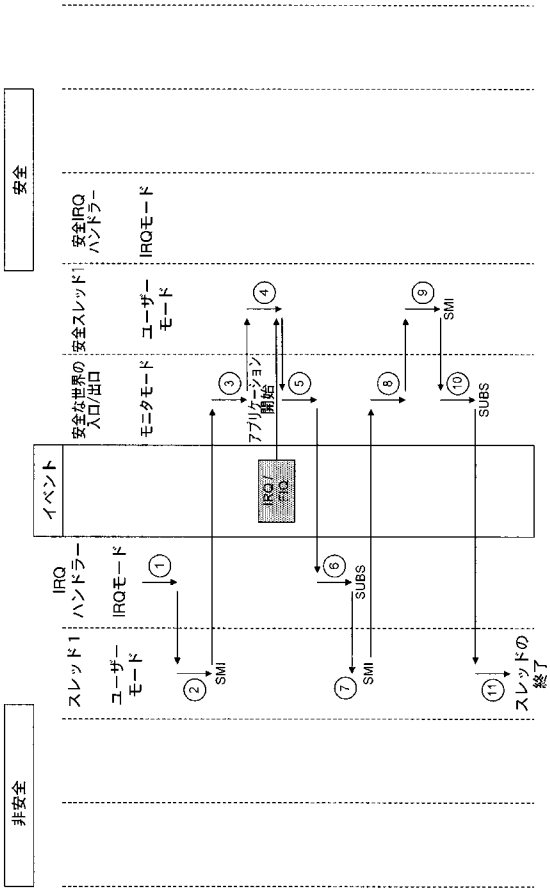
【図９】



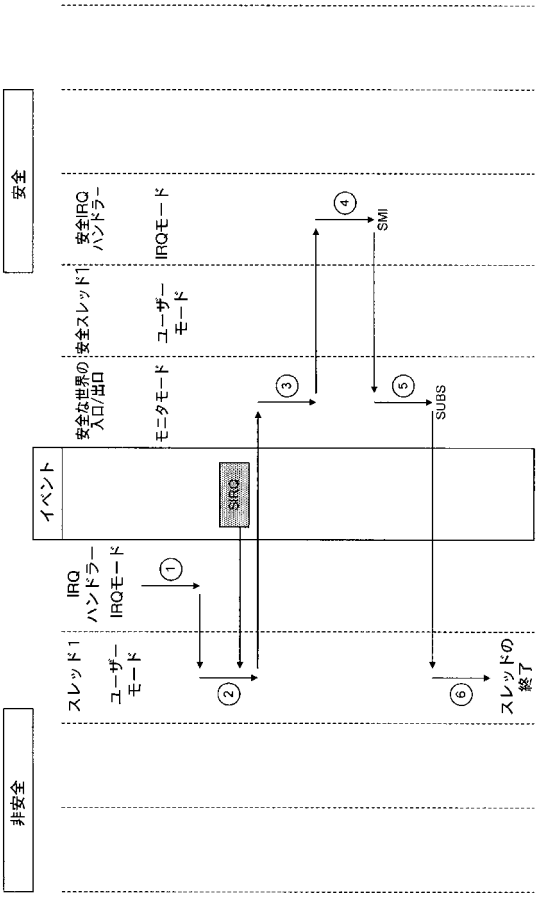
【図１０】



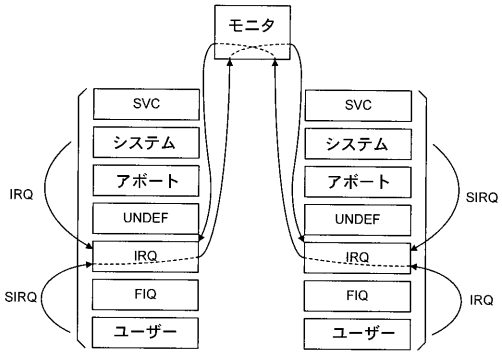
【図１１Ａ】



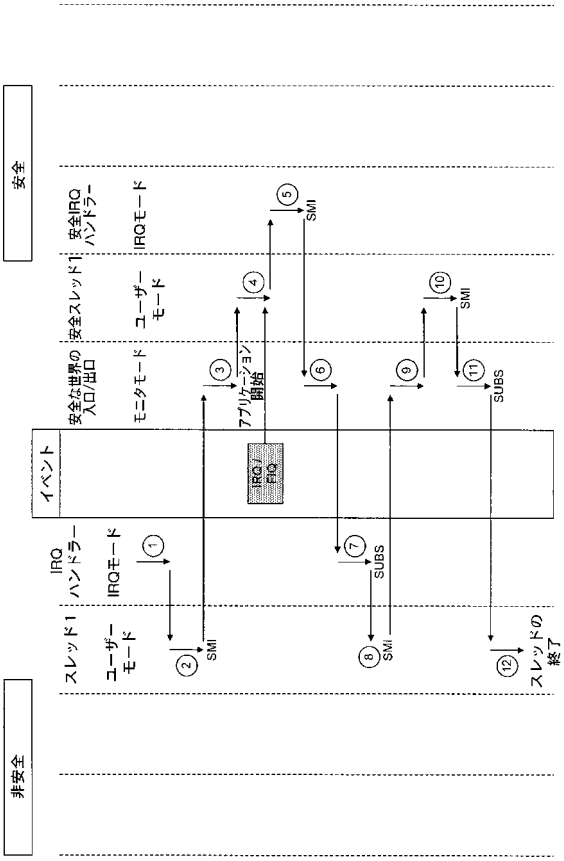
【図１１Ｂ】



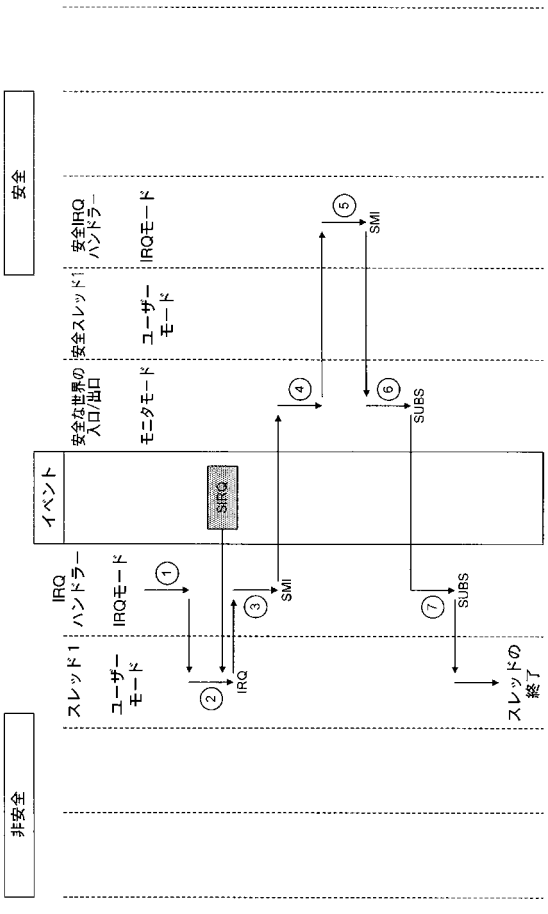
【図１２】



【図１３Ａ】



【図１３Ｂ】



【図１４】

例外	ベクトル オフセット	対応モード
リセット	0x00	スーパーバイザーモード
UNDEF	0x04	モニタモード/UNDEF モード
SWI	0x08	スーパーバイザーモード/モニタモード
プリフェッチ アボート	0x0C	アボートモード/モニタモード
データ アボート	0x10	アボートモード/モニタモード
IRQ / SIRQ	0x18	IRQモード/モニタモード
FIQ	0x1C	FIQモード/モニタモード
SMI	0x20	UNDEF モード / モニタモード

【図１５】

モニタ

リセット	VM0
UNDEF	VM1
SWI	VM2
プリフェッチ アボート	VM3
データ アボート	VM4
IRQ / SIRQ	VM5
FIQ	VM6
SMI	VM7

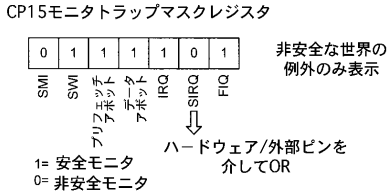
安全

リセット	VS0
UNDEF	VS1
SWI	VS2
プリフェッチ アボート	VS3
データ アボート	VS4
IRQ / SIRQ	VS5
FIQ	VS6
SMI	VS7

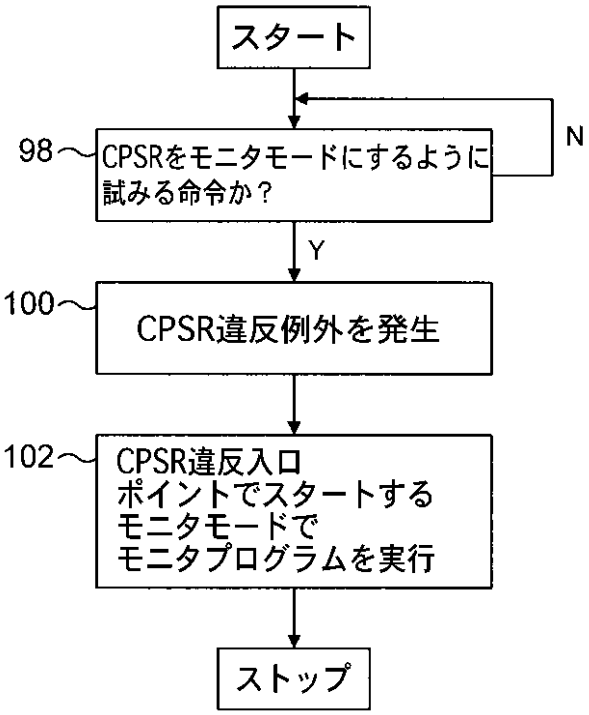
非安全

リセット	VNS0
UNDEF	VNS1
SWI	VNS2
プリフェッチ アボート	VNS3
データ アボート	VNS4
IRQ / SIRQ	VNS5
FIQ	VNS6
SMI	VNS7

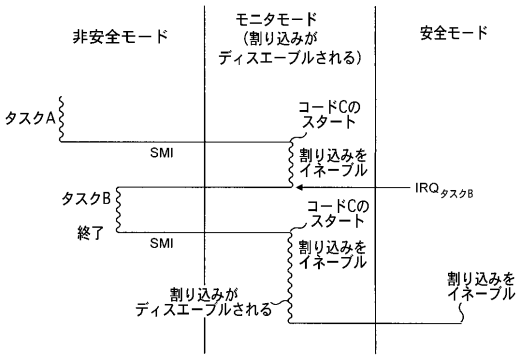
【図１６】



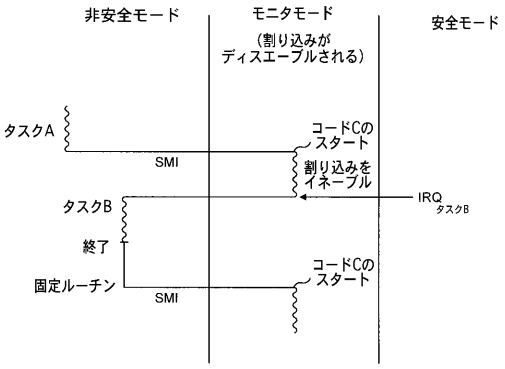
【図１７】



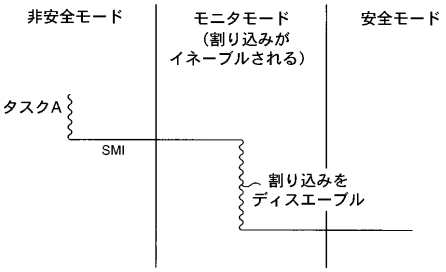
【図１８】



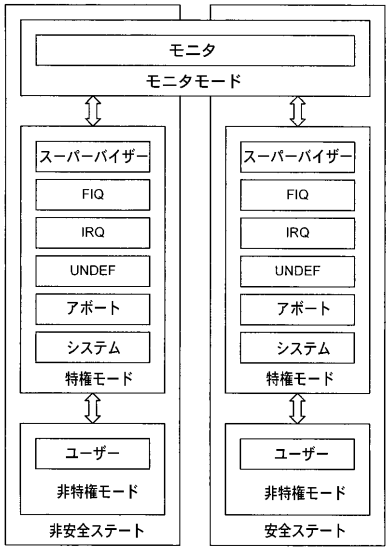
【図１９】



【図２０】



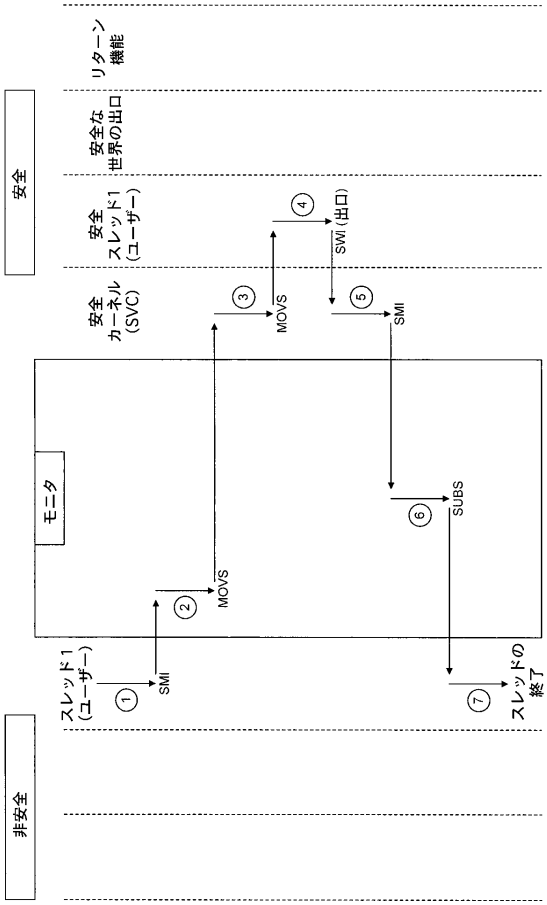
【図２１】



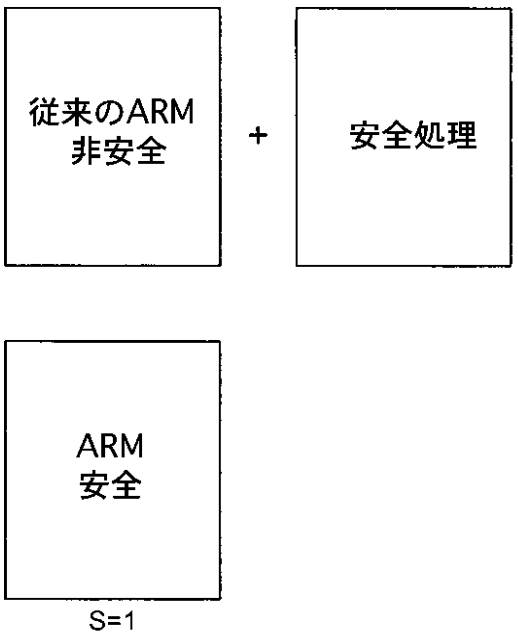
【図２２】

ユーザー	システム	スーパーバイザー	アボート	無定義	割り込み	高速割り込み	モニタ
R0	R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_FIQ	R8
R9	R9	R9	R9	R9	R9	R9_FIQ	R9
R10	R10	R10	R10	R10	R10	R10_FIQ	R10
R11	R11	R11	R11	R11	R11	R11_FIQ	R11
R12	R12	R12	R12	R12	R12	R12_FIQ	R12
R13	R13	R13_SVC	R13_ABT	R13_UND	R13_IRQ	R13_FIQ	R13_MON
R14	R14	R14_SVC	R14_ABT	R14_UND	R14_IRQ	R14_FIQ	R14_MON
PC	PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_SVC	SPSR_ABT	SPSR_UND	SPSR_IRQ	SPSR_FIQ	SPSR_MON

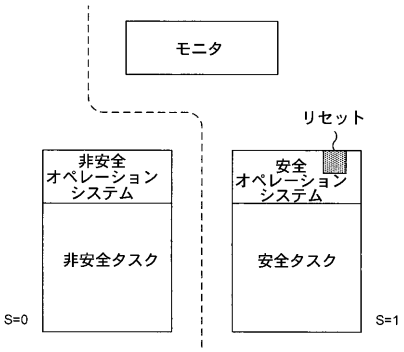
【図２３】



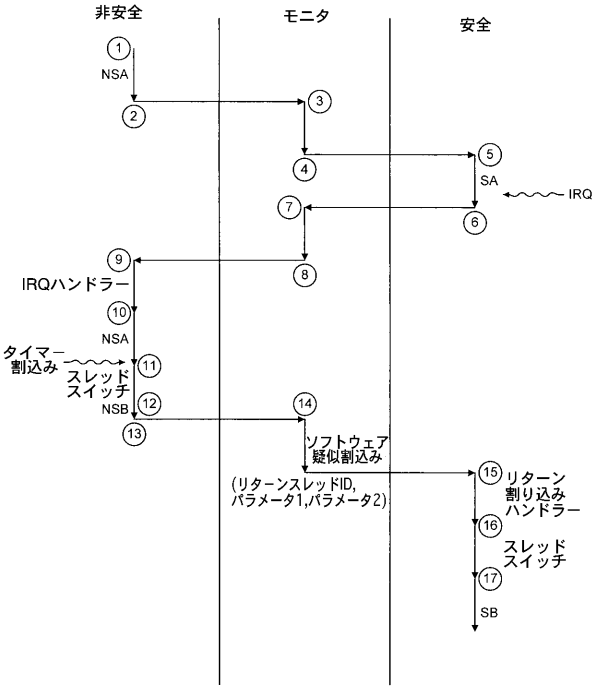
【図２４】



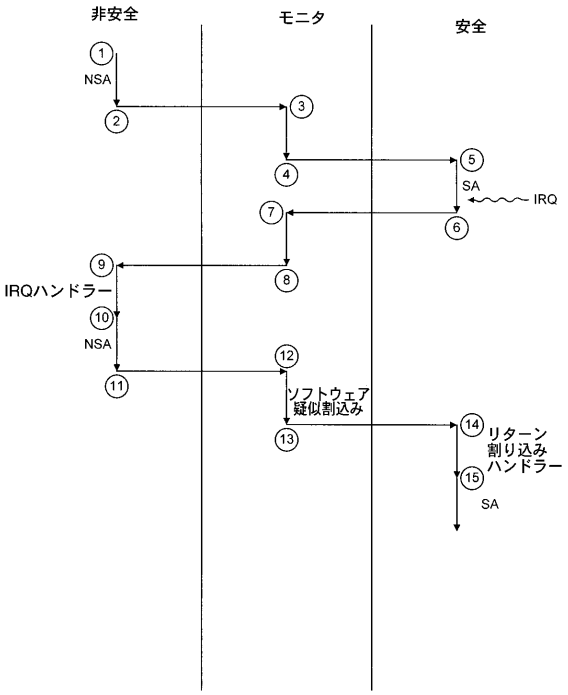
【図２５】



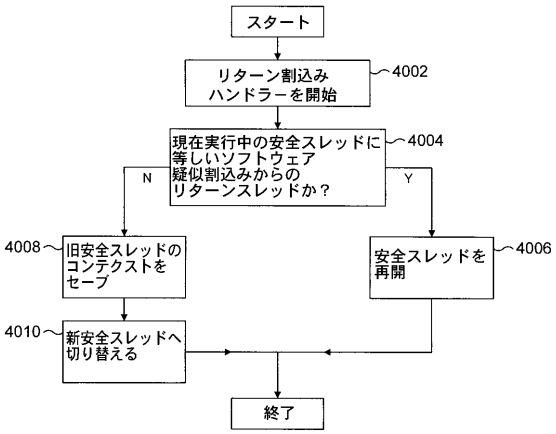
【図２６】



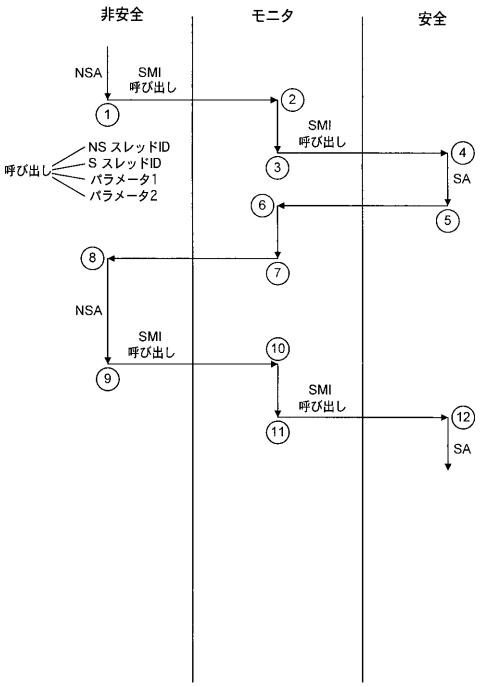
【図２７】



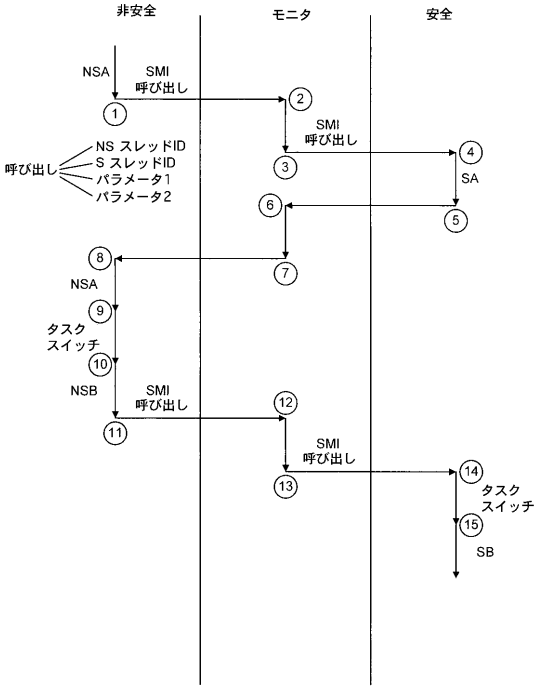
【図２８】



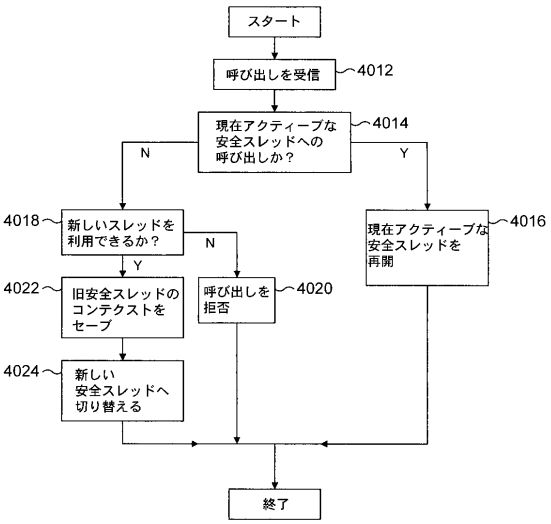
【図２９】



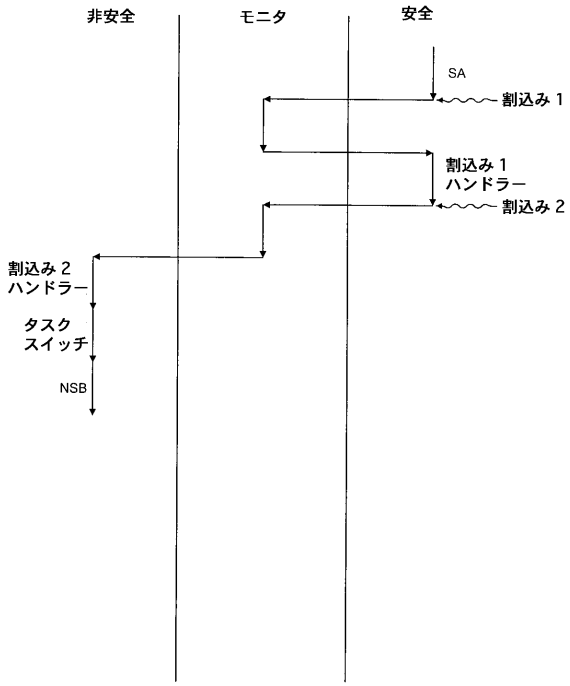
【図３０】



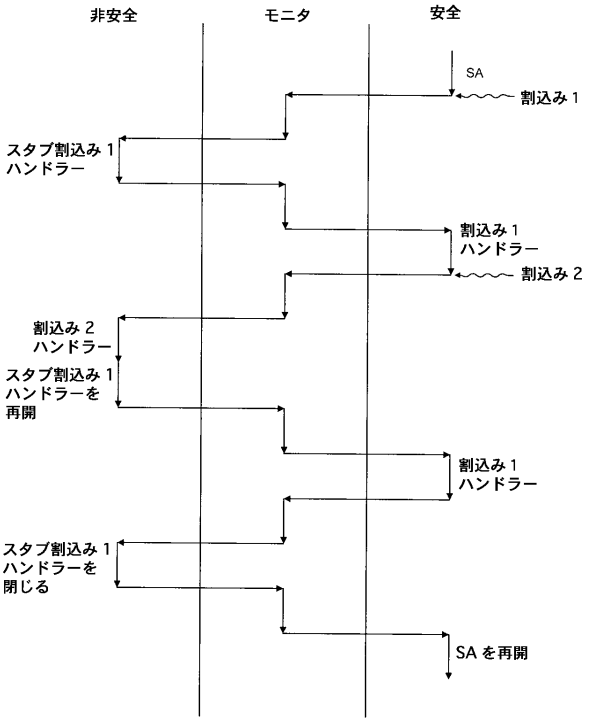
【図３１】



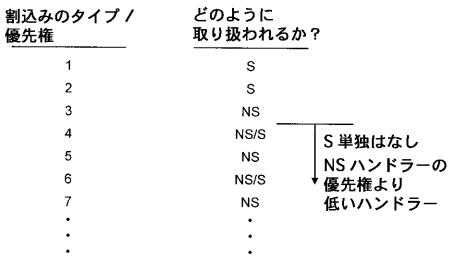
【図３２】



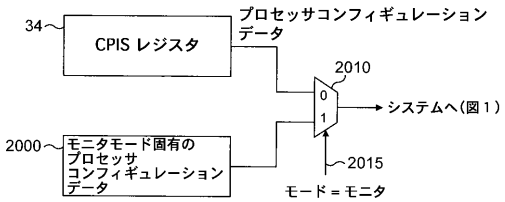
【図３３】



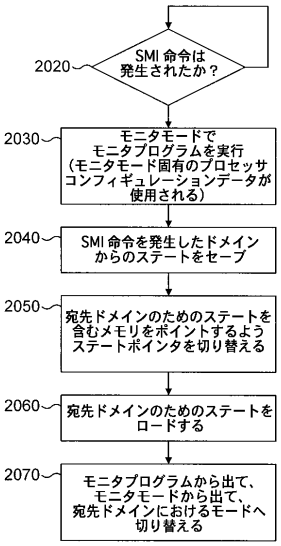
【図３４】



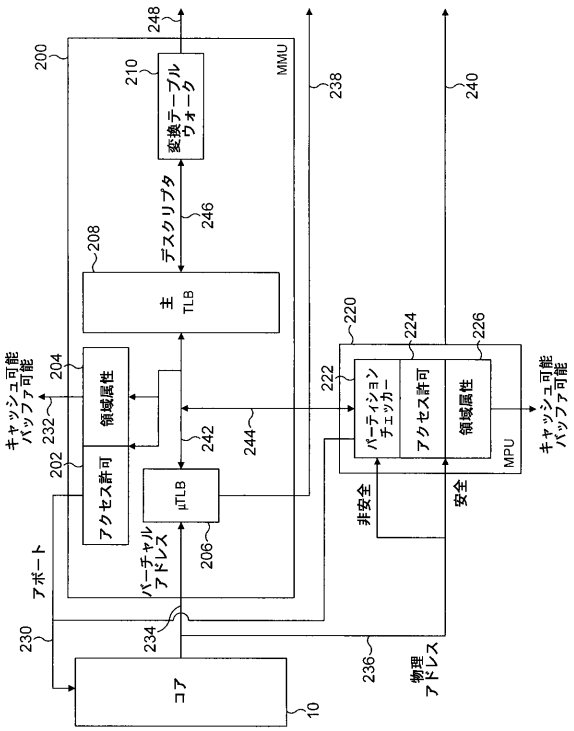
【図３５】



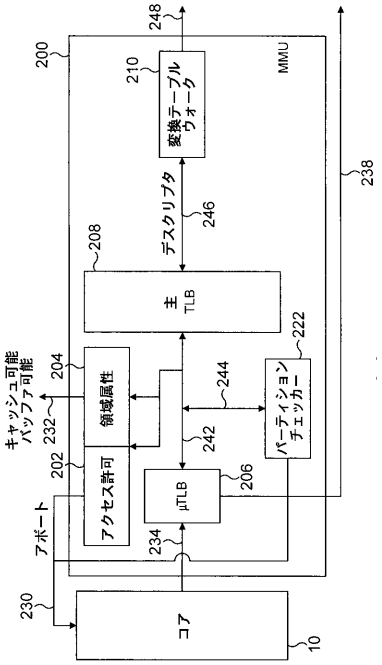
【図３６】



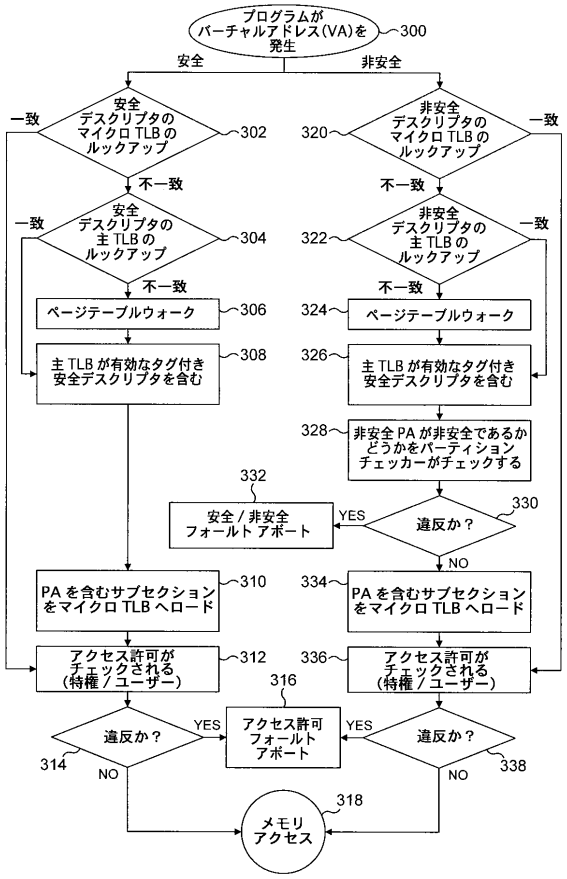
【図３７】



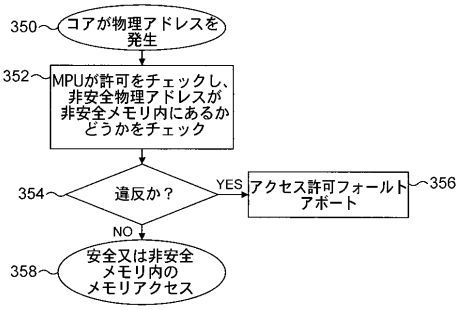
【図３８】



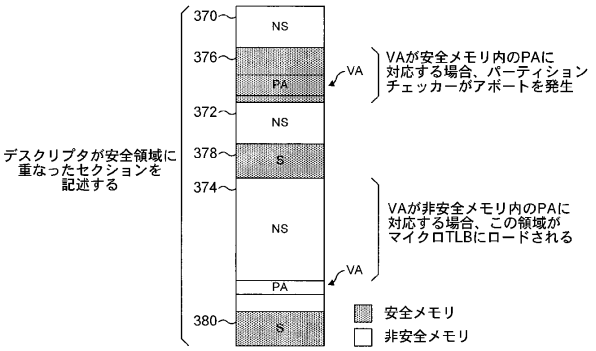
【図３９】



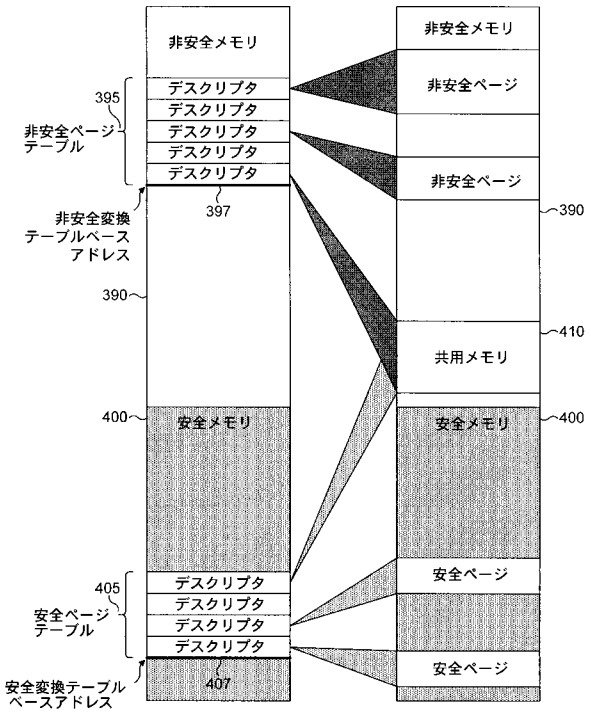
【図４０】



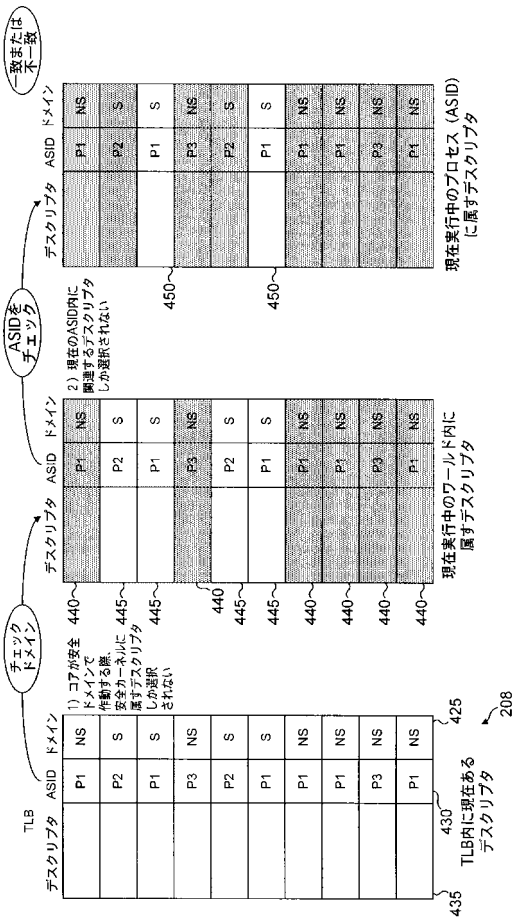
【図４１】



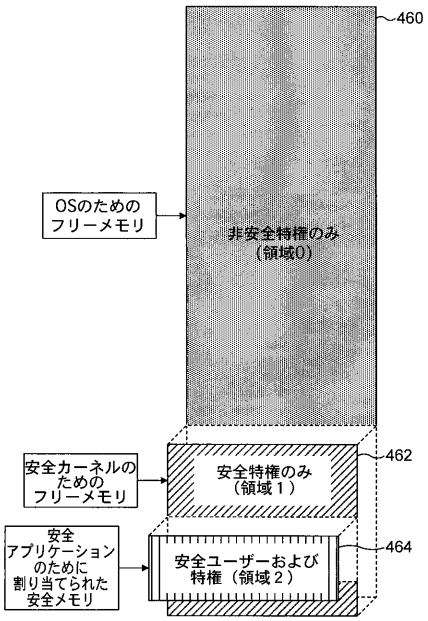
【図４２】



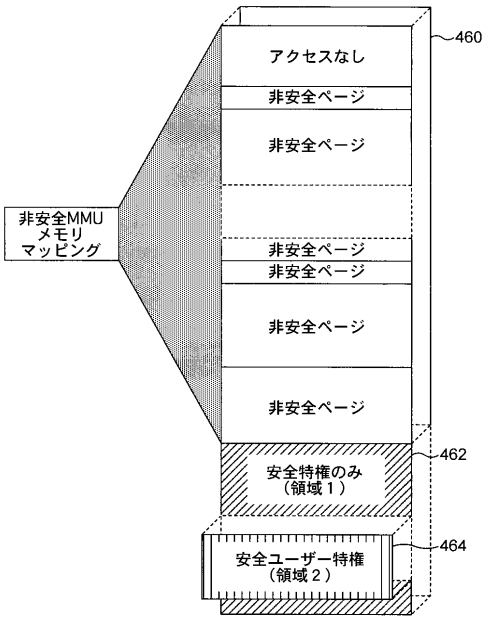
【図４３】



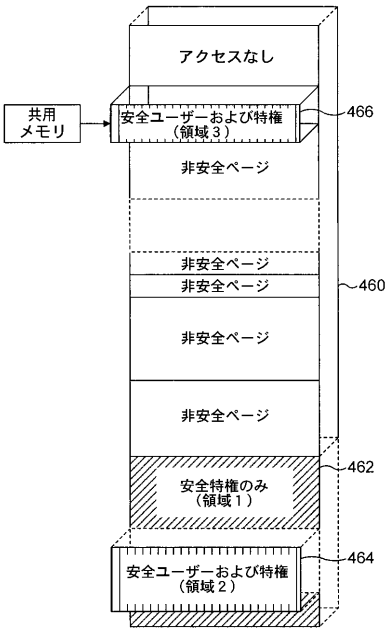
【図４４】



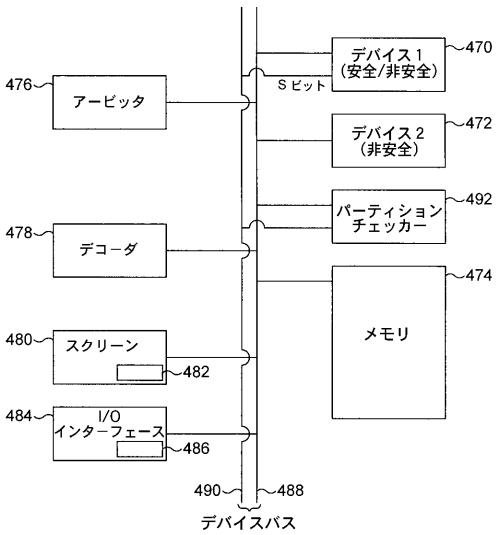
【図４５】



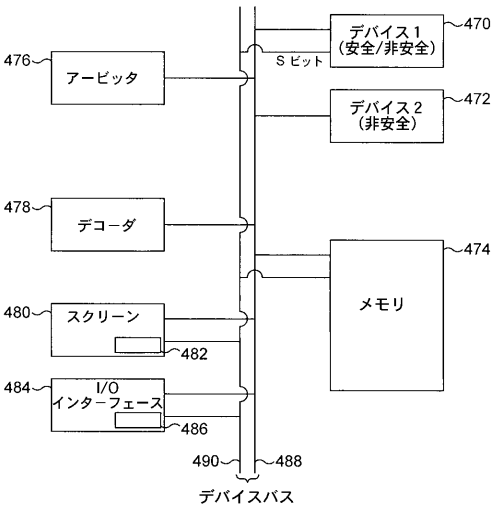
【図４６】



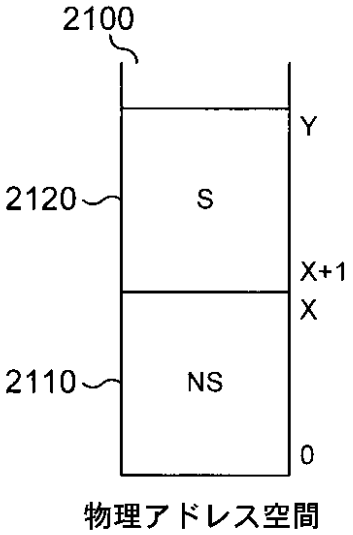
【図４７】



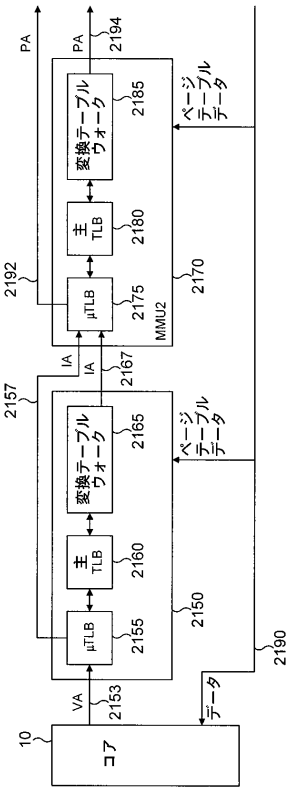
【図４８】



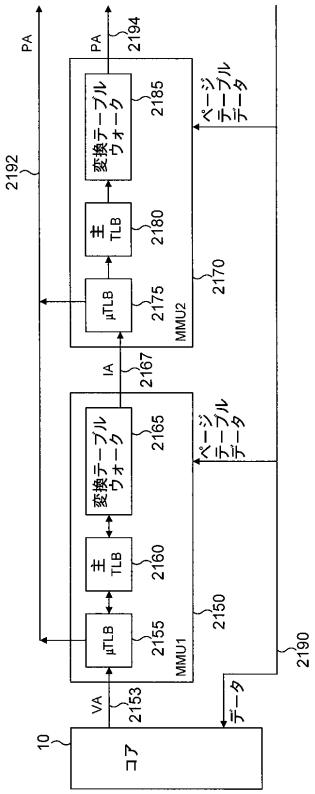
【図４９】



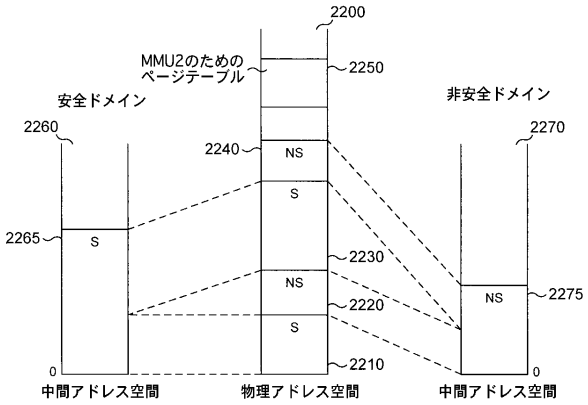
【図５０Ａ】



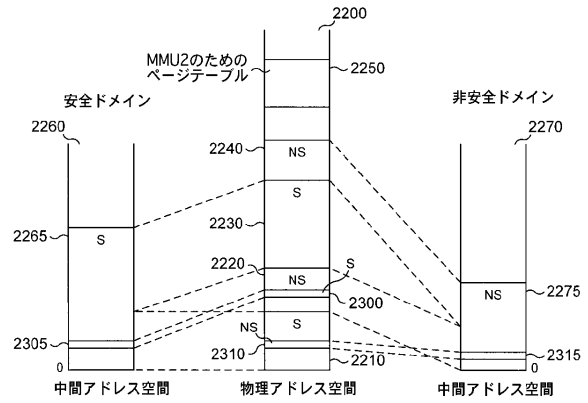
【図５０Ｂ】



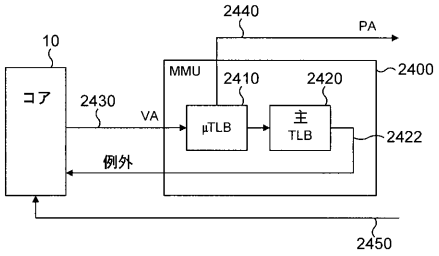
【図５１】



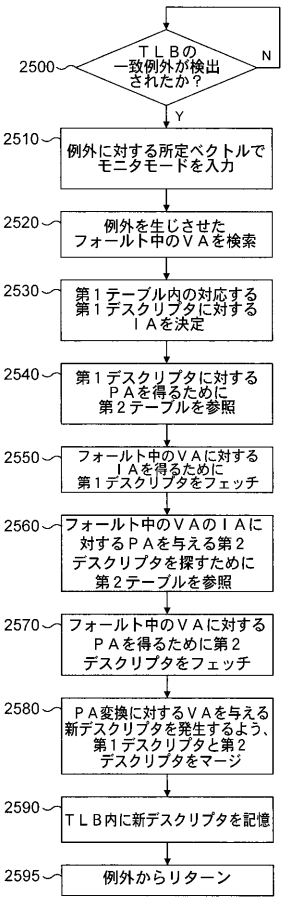
【図５２】



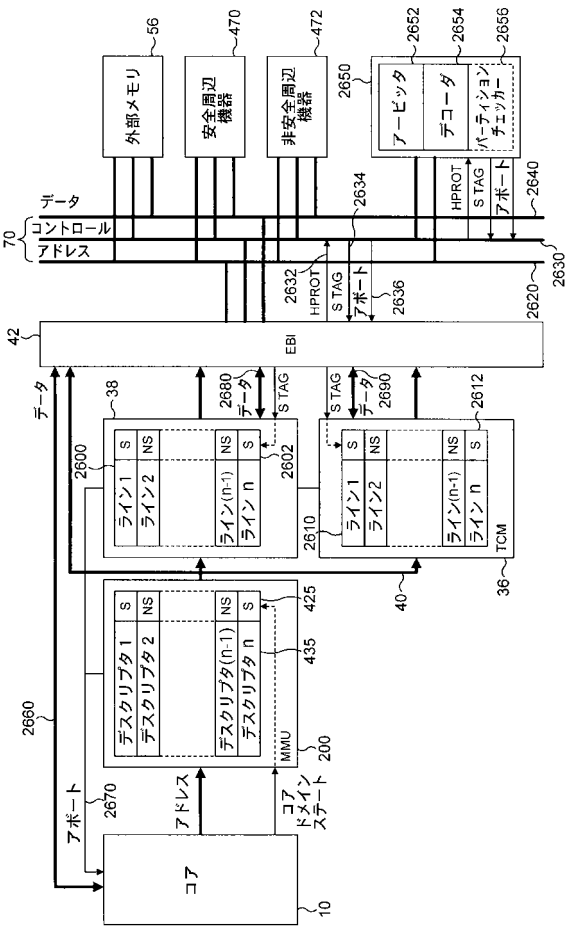
【図５３】



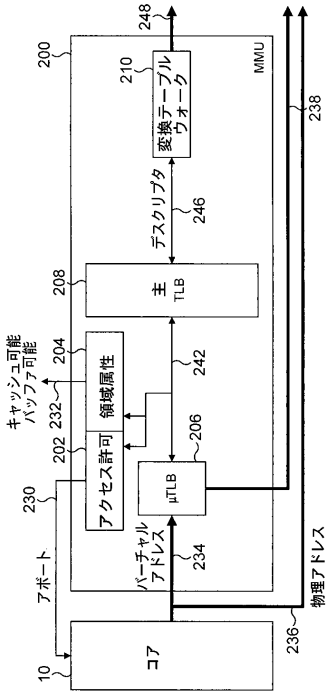
【図５４】



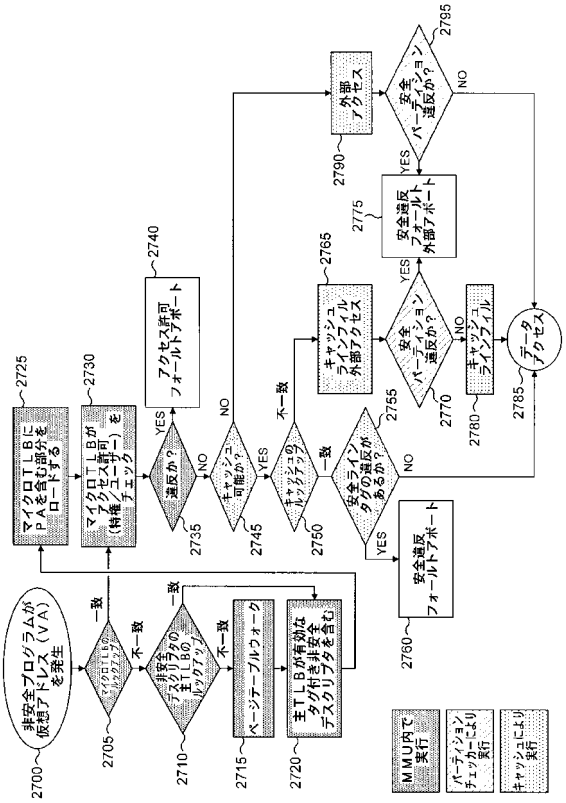
【図５５】



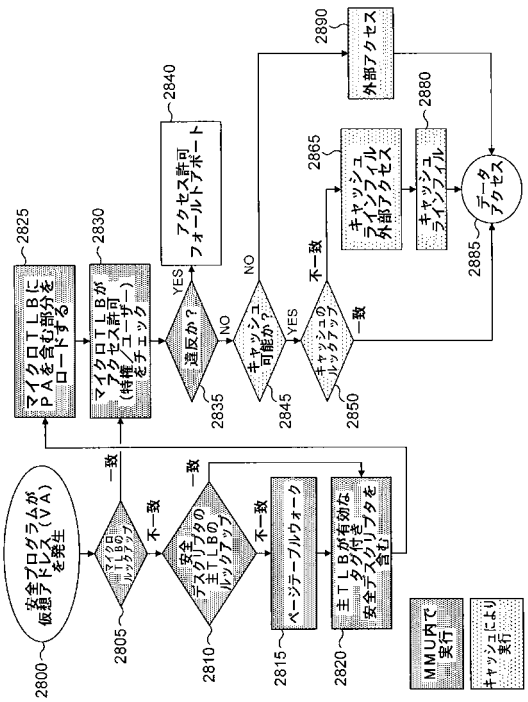
【図５６】



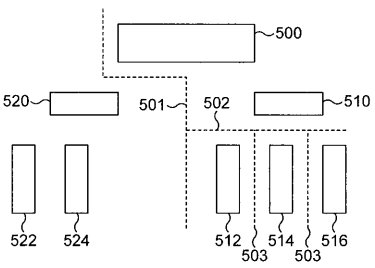
【図５７】



【図５８】



【図５９】



【図６０】

入力法	どのようにプログラムするか？	どのように入力するか？	入力モード
ブレークポイントの一致	ＴＡＰまたはソフトウェア（ＣＰ１４）をデバッグ	ブレークポイントレジスタおよび／またはコンテクストＩＤレジスタをプログラムし、命令アドレスおよび／またはＣＰ１５コンテクストＩＤとの比較に成功（２）	停止／モニタ（１）
ソフトウェアブレークポイント命令	デバッグＴＡＰを介してスキャンチェーン４（命令転送レジスタ）内に、ＢＫＰＴ命令を入力するか、またはコード内で直接ＢＫＰＴ命令を使用	ＢＫＰＴ命令は実行段階に達しなければならない	停止／モニタ
ベクトルトラップブレークポイント	ＴＡＰをデバッグ	ベクトルトラップレジスタをプログラムし、アドレスが一致する	停止／モニタ
ウォッチポイントの一致	ＴＡＰまたはソフトウェア（ＣＰ１４）をデバッグ	ウォッチポイントレジスタおよび／またはコンテクストＩＤレジスタをプログラムし、命令アドレスおよび／またはＣＰ１５コンテクストＩＤとの比較に成功（２）	停止／モニタ（１）
内部デバッグリクエスト	ＴＡＰをデバッグ	停止命令はスキャンインされる	停止
外部デバッグリクエスト	適用不可	ＥＤＢＧＲＱ入力ピンがアサートされる	停止

（１）：モニタモードではブレークポイントおよびウォッチポイントはデータに依存できない
（２）：コアは特定のスレッドで安全デバッグをイネーブルするためにスレッドアウェアブレークポイントおよびウォッチポイントのためのサポートを有する

【図６１】

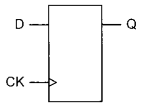
名称	意味	リセット値	アクセス	テスト用のスキャンチェーンに挿入
モニタモードイネーブルビット	0：停止モード 1：モニタモード	1	ＪＴＡＧ（スキャン１）によりＩＣＥをプログラムすることによってＲ／W ・ＭＲＣ／ＭＣＲ命令（ＣＰ１４）を使用することによりＲ／W	YES
安全デバッグイネーブルビット	0：非安全な世界でのみデバッグ 1：安全な世界および非安全な世界でデバッグ	0	機能モードまたはデバッグモニタモードでは：（安全スーパーバイザーモードでのみ）ＭＲＣ／ＭＣＲ命令（ＣＰ１４）によりＲ／W デバッグ停止モードでは：アクセスなし－ＭＣＲ／ＭＲＣ命令は効果がある （ＪＳＤＡＥＮ＝１の場合、ＪＴＡＧ（スキャン１）によりＩＣＥをプログラムすることによってＲ／W）	NO
安全トレースイネーブルビット	0：非安全な世界でのみＥＴＭをイネーブル 1：安全な世界および非安全な世界でＥＴＭをイネーブル	0	機能モードまたはデバッグモニタモードでは：（安全スーパーバイザーモードでのみ）ＭＲＣ／ＭＣＲ命令（ＣＰ１４）によりＲ／W デバッグ停止モードでは：アクセスなし－ＭＣＲ／ＭＲＣ命令は効果がある （ＪＳＤＡＥＮ＝１の場合、ＪＴＡＧ（スキャン１）によりＩＣＥをプログラムすることによってＲ／W）	NO
安全ユーザーモードイネーブルビット	0：安全ユーザーモードではデバッグは可能ではない 1：安全ユーザーモードではデバッグが可能	1	機能モードまたはデバッグモニタモードでは：（安全スーパーバイザーモードでのみ）ＭＲＣ／ＭＣＲ命令（ＣＰ１４）によりＲ／W デバッグ停止モードでは：アクセスなし－ＭＣＲ／ＭＲＣ命令は効果がある （ＪＳＤＡＥＮ＝１の場合、ＪＴＡＧ（スキャン１）によりＩＣＥをプログラムすることによってＲ／W）	NO
安全スレッドアウェアイネーブルビット	0：特定スレッドに対し、デバッグは可能ではない 1：特定スレッドに対し、デバッグが可能	0	機能モードまたはデバッグモニタモードでは：（安全スーパーバイザーモードでのみ）ＭＲＣ／ＭＣＲ命令（ＣＰ１４）によりＲ／W デバッグ停止モードでは：アクセスなし－ＭＣＲ／ＭＲＣ命令は効果がある （ＪＳＤＡＥＮ＝１の場合、ＪＴＡＧ（スキャン１）によりＩＣＥをプログラムすることによってＲ／W）	NO

【図６２】

機能テーブル

D	CK	Q[n+1]
0	↑	0
1	↑	1
X	↑	Q[n]

ロジックシンボル

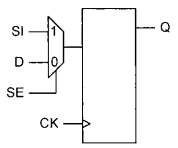


【図６３】

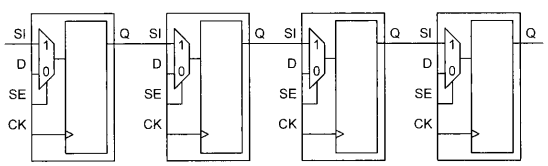
機能テーブル

D	SI	SE	CK	Q[n+1]
0	X	0	↑	0
1	X	0	↑	1
X	X	X	↑	Q[n]
X	0	1	↑	0
X	1	1	↑	1

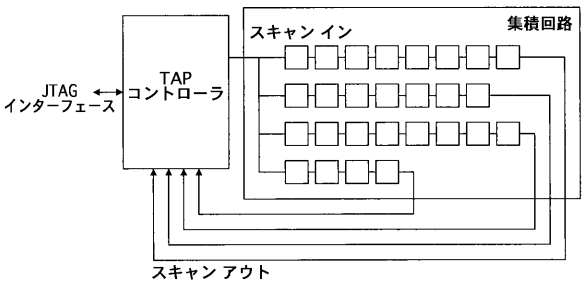
ロジックシンボル



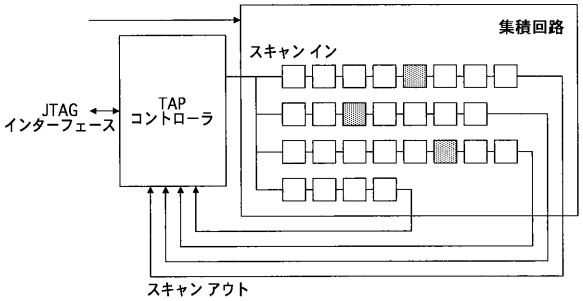
【図６４】



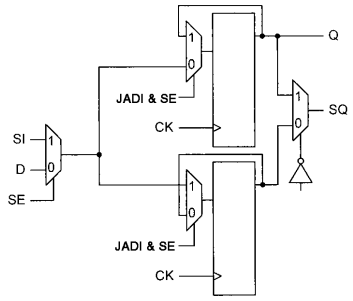
【図６５】



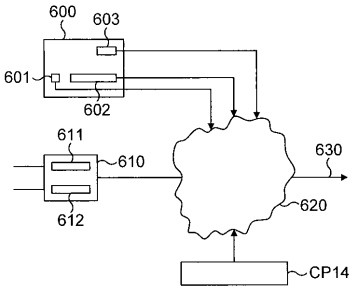
【図６６Ａ】



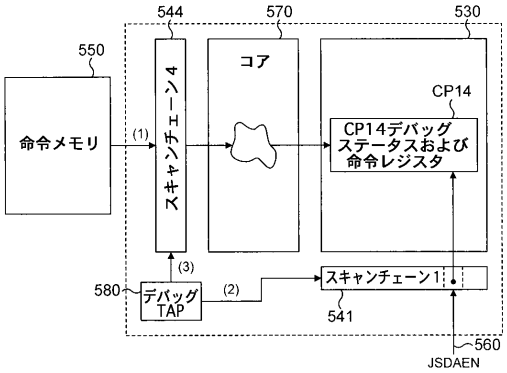
【図６６Ｂ】



【図６８】



【図６７】



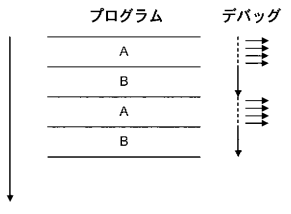
【図６９Ａ】

デバッグおよびステータス制御レジスタにおけるＣＰ１４のビット			意味
安全デバッグイネーブルビット	安全ユーザーモードデバッグイネーブルビット	安全スレッドアウェアデバッグイネーブルビット	
0	X	X	安全な世界全体での割り込み的デバッグは不可能である。安全な世界全体ではデバッグステートに入るためのデバッグリクエスト、ブレークポイント、ウォッチポイントおよびその他の機構は無視される
1	0	X	安全な世界全体でのデバッグは可能である
1	1	0	安全ユーザーモードだけのデバッグ。ユーザーモードだけでデバッグステートに入るためのデバッグリクエスト、ブレークポイント、ウォッチポイントおよびその他の機構が考慮される。（スレッドＩＤにリンクしているか、またはリンクしていないブレークポイントおよびウォッチポイントを考慮）。デバッグにおけるアクセスは安全ユーザーがアクセスできることに限定される
1	1	1	特定のスレッドに限りデバッグが可能である。この場合、デバッグステートに入るためにスレッドＩＤにリンクしたスレッドアウェアブレークポイントおよびウォッチポイントが考慮される。更に、各スレッドは自らのコードをデバッグすることができ、自らのコードだけをデバッグすることもできる

【図６９Ｂ】

デバッグおよびステータス制御レジスタにおけるＣＰ１４のビット			意味
安全デバッグイネーブルビット	安全ユーザーモードデバッグイネーブルビット	安全スレッドアウェアデバッグイネーブルビット	
0	X	X	安全な世界全体での観測可能なデバッグはできない。トレースモジュール（ＥＴＭ）は内部コアの活動をトレースしてはならない
1	0	X	安全な世界全体でのトレースは可能である
1	1	0	コアが安全ユーザーモードにしかないときには、トレースは可能である
1	1	1	コアが安全ユーザーモードで特定のスレッドを実行しているときにしかトレースはできない。このために特定のハードウェアを専用とするか、またはブレークポイントレジスタペアを再使用しなければならない。コンテクストＩＤの一致はデバッグステートに入る代わりにトレースをイネーブルしなければならない

【図７０】



【図７１Ａ】

入力方法	非安全な世界の内にあるときの入力	安全な世界の内にあるときの入力
ブレークポイントの一致	非安全プリフェッチアボートハンドラー	安全プリフェッチアボートハンドラー
ソフトウェアブレークポイント命令	非安全プリフェッチアボートハンドラー	安全プリフェッチアボートハンドラー
ベクトルトラップブレークポイント	非安全データアボートおよび非安全プリフェッチアボート割り込みに対してディスエーブル。その他の非安全例外に対してプリフェッチアボート	安全データアボートおよび安全プリフェッチアボート例外に対してディスエーブル（１）。その他の例外に対して安全プリフェッチアボート
ウォッチポイントの一致	非安全データアボートハンドラー	安全データアボートハンドラー
内部デバッグリクエスト	停止モードでステートをデバッグ	停止モードでステートをデバッグ
外部デバッグリクエスト	停止モードでステートをデバッグ	停止モードでステートをデバッグ

（１）ベクトルトラップレジスタに関する情報を参照
（２）外部または内部デバッグリクエストがアサートされたときに、コアは停止モードに入り、非モニタモードには入らない

【図７１Ｂ】

入力方法	非安全な世界の内にあるときの入力	安全な世界のでの入力
ブレークポイントの一致	非安全プリフェッチアボートハンドラー	ブレークポイントを無視
ソフトウェアブレークポイント命令	非安全プリフェッチアボートハンドラー	命令を無視（１）
ベクトルトラップブレークポイント	非安全データアボートおよび非安全プリフェッチアボート割り込みに対してディスエーブル。その他の割り込みに対しては非安全プリフェッチアボート	ブレークポイントを無視
ウォッチポイントの一致	非安全データアボートハンドラー	ウォッチポイントを無視
内部デバッグリクエスト	停止モードでステートをデバッグ	リクエストを無視
外部デバッグリクエスト	停止モードでステートをデバッグ	リクエストを無視
システムスピードアクセスからの再入力をデバッグ	適用不可	適用不可

（１）非安全な世界からの安全な世界の内でのＢＫＰＴ命令の置換は可能ではないので、非安全アボートは違反を取り扱わなければならない

フロントページの続き

(72)発明者　デイヴィッド　ヘナー　マンセル
イギリス国　ウスク、ランデニー、カエ　ノヴィル
(72)発明者　リヨネル　ベルネ
フランス国　ヴィルヌーブ　－　ローベ、アブニュ　デュ　ロジ　デ　ボノー　２６１、ル　ロイヤル　パルク
(72)発明者　サイモン　チャールズ　ワット
イギリス国　ケンブリッジ、バーウェル、マンデヴィル、ザ　タン　ハウス

審査官　塚田　肇

(56)参考文献　特開昭６２－０７８６４２（ＪＰ，Ａ）
特開平０２－２３９３４９（ＪＰ，Ａ）
特開平０５－０１２０４５（ＪＰ，Ａ）
特開２００１－１７５４８６（ＪＰ，Ａ）
特開２００２－０７３３５８（ＪＰ，Ａ）
特開２０００－０７６０８７（ＪＰ，Ａ）
特表２００３－５１８２８７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／１４
Ｇ０６Ｆ　２１／２４