

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7369701号
(P7369701)

(45)発行日 令和5年10月26日(2023.10.26)

(24)登録日 令和5年10月18日(2023.10.18)

(51)国際特許分類

G 0 6 F 11/36 (2006.01)

F I

G 0 6 F 11/36 1 2 0

請求項の数 23 (全26頁)

(21)出願番号	特願2020-543981(P2020-543981)	(73)特許権者	500395107
(86)(22)出願日	平成31年1月17日(2019.1.17)		アーム・リミテッド
(65)公表番号	特表2021-515307(P2021-515307 A)		グレート・ブリテン及び北部アイルラン ド連合王国、イングランド、シー・ビー ー1、9エヌ・ジェイ、ケンブリッジ、 フルボーン・ロード 1 1 0
(43)公表日	令和3年6月17日(2021.6.17)	(74)代理人	110000855
(86)国際出願番号	PCT/GB2019/050123		弁理士法人浅村特許事務所
(87)国際公開番号	WO2019/166759	(72)発明者	ウィリアムズ、マイケル ジョン
(87)国際公開日	令和1年9月6日(2019.9.6)		英国、ケンブリッジシャー、ケンブリッ ジ、チェリー ヒントン、フルボーン ロ ード 1 1 0、エイアールエム リミテッ ド 気付
審査請求日	令和4年1月7日(2022.1.7)	(72)発明者	バーンズ、グレイム ピーター
(31)優先権主張番号	1803168.2		英国、ケンブリッジシャー、ケンブリッ 最終頁に続く
(32)優先日	平成30年2月27日(2018.2.27)		
(33)優先権主張国・地域又は機関	英国(GB)		

(54)【発明の名称】 処理回路上で実行されるべきプログラムをデバッグするときにメタデータにアクセスする
方法

(57)【特許請求の範囲】

【請求項1】

処理回路上で実行されるべきプログラムをデバッグするようにデバッガを動作させる方法であって、前記処理回路が、関連するメタデータ項目を有するデータ・グラニュールから形成されたデータに対して動作し、前記方法は、

前記処理回路によって複数回にわたってメタデータ・アクセス動作を引き起こすために指定された数のメタデータ項目にアクセスすることを決めるステップと、

前記処理回路が、前記指定された数のメタデータ項目の少なくともサブセットにアクセスするために前記メタデータ・アクセス動作の複数のインスタンスを実施することを引き起こすために、前記処理回路に少なくとも1つのコマンドを発行するステップであって、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数は、前記デバッガにとって非決定論的であり、前記少なくとも1つのコマンドに従って、前記デバッガが、アクセスされる前記数のメタデータ項目の進捗を決定するために前記メタデータ・アクセス動作の各インスタンスの間に前記処理回路に問い合わせることなしに、前記メタデータ・アクセス動作の前記複数のインスタンスが前記処理回路によって実施される、ステップと

を含む、方法。

【請求項2】

少なくとも1つのアクセス・インジケータが、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数に関する制約情報を示すために提供

され、前記少なくとも1つのコマンドに応答して実施される前記メタデータ・アクセス動作の前記複数のインスタンスが、前記少なくとも1つのアクセス・インジケータを参照して決定される、請求項1に記載の方法。

【請求項3】

前記メタデータ・アクセス動作の前記複数のインスタンスとして、前記処理回路が、前記メタデータ・アクセス動作のインスタンスを、前記指定された数のメタデータ項目のすべてにアクセスするのに十分な数で実施することを、前記デバッガから前記処理回路に発行された前記少なくとも1つのコマンドが引き起こすことを可能にするのに十分な情報を、前記少なくとも1つのアクセス・インジケータが含んでいる、請求項2に記載の方法。

【請求項4】

前記少なくとも1つのアクセス・インジケータが、前記処理回路の実装詳細に固有でない一般的な最小指示を提供し、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最小数を示す、請求項3に記載の方法。

【請求項5】

前記少なくとも1つのアクセス・インジケータが、前記処理回路についての実装最小指示を提供し、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最小数を示す、請求項3に記載の方法。

【請求項6】

前記少なくとも1つのアクセス・インジケータが、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の前記数の正確な指示を提供する、請求項3に記載の方法。

【請求項7】

前記少なくとも1つのアクセス・インジケータが、前記処理回路についての実装最大指示を提供し、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最大数を示す、請求項2に記載の方法。

【請求項8】

前記デバッガによって前記処理回路に発行された前記少なくとも1つのコマンドは、前記処理回路が、前記メタデータ・アクセス動作の前記複数のインスタンスとして、M個のインスタンスを実施することを引き起こし、Mが、メタデータ項目の前記指定された数を、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の前記最大数で除算したものに等しく、

前記処理回路が前記メタデータ・アクセス動作の前記複数のインスタンスを実施したとき、前記デバッガは、前記メタデータ・アクセス動作のさらなるインスタンスが必要とされるかどうかを決定するために、アクセスされる前記数のメタデータ項目の進捗を決定するために、前記処理回路に問い合わせるように構成される、請求項7に記載の方法。

【請求項9】

前記デバッガによって発行された前記少なくとも1つのコマンドは、前記メタデータ・アクセス動作の各インスタンスについてアドレスが前記処理回路によって決定されることを引き起こし、前記アドレスが、前記メタデータ・アクセス動作の関連するインスタンスの実施中にアクセスされる1つ又は複数のメタデータ項目を識別するために使用される、請求項1から8までのいずれか一項に記載の方法。

【請求項10】

前記メタデータ・アクセス動作の各インスタンスについての前記アドレスが、前記少なくとも1つのアクセス・インジケータに従って決定される、請求項2に従属するときの請求項9に記載の方法。

【請求項11】

前記少なくとも1つのコマンドが、前記メタデータ・アクセス動作の各インスタンスについてのコマンド・シーケンスを備え、前記コマンド・シーケンスが、少なくとも、前記メタデータ・アクセス動作の対応するインスタンスと、前記メタデータ・アクセス動作の

10

20

30

40

50

前記対応するインスタンスによって使用されるべき前記アドレスを決定するためのアドレス生成動作とを備える、動作のシーケンスを識別する、請求項 9 又は 10 に記載の方法。

【請求項 12】

アドレス空間領域が、アクセスされることになるメタデータ項目の前記指定された数に基づいて識別され、

前記デバッグによって誘発される前記メタデータ・アクセス動作の現在のインスタンスの実施中に、前記処理回路が、前記メタデータ・アクセス動作の次のインスタンスについての前記アドレスとして使用されるべき更新されたアドレスを生成するように構成され、前記更新されたアドレスは、前記現在のインスタンスによってアクセスされるメタデータ項目の前記数を考慮に入れるために前記現在のインスタンスによって使用される前記アドレスを修正すること、ただし、前記更新されたアドレスが前記アドレス空間領域内に残ることを確実にするために必要に応じて前記アドレス空間領域のある端部から前記アドレス空間領域の別の端部までラッピングするためのラッピング機能を採用することによって生成される、

請求項 9 に記載の方法。

【請求項 13】

前記処理回路の非デバッグ動作モードにおいて、前記メタデータ・アクセス動作の実施は、前記更新されたアドレスを生成したときに前記ラッピング機能が無効にされるように、前記処理回路が前記メタデータ・アクセス動作を実施することを引き起こす、請求項 12 に記載の方法。

【請求項 14】

前記デバッグが、前記ラッピング機能を選択的に無効にするように前記処理回路内の制御記憶要素をセットするように構成される、請求項 12 又は 13 に記載の方法。

【請求項 15】

前記少なくとも 1 つのコマンドが、前記メタデータ・アクセス動作の各インスタンスについての 1 つ又は複数のコマンドを備える、請求項 1 から 14 までのいずれか一項に記載の方法。

【請求項 16】

前記デバッグによって発行された前記少なくとも 1 つのコマンドは、前記処理回路が前記メタデータ・アクセス動作の前記複数のインスタンスを実施するために必要とされる前記動作を作成するために、前記処理回路内の動作生成回路によって処理される、請求項 1 から 10 まで又は 12 から 14 までのいずれか一項に記載の方法。

【請求項 17】

前記処理回路によって前記メタデータ・アクセス動作を前記複数回実施することは、前記指定された数のメタデータ項目が取り出され、選定されたレジスタに記憶されることを引き起こす、請求項 1 から 16 までのいずれか一項に記載の方法。

【請求項 18】

前記処理回路によって前記メタデータ・アクセス動作を前記複数回実施することは、前記指定された数のメタデータ項目の値が、選定されたレジスタに記憶されたメタデータ値を使用して更新されることを引き起こす、請求項 1 から 16 までのいずれか一項に記載の方法。

【請求項 19】

メタデータ項目の前記指定された数が、前記選定されたレジスタに記憶され得るメタデータ項目の前記数によって決定される、請求項 17 又は 18 に記載の方法。

【請求項 20】

各メタデータ項目が 1 つ又は複数のビットを備える、請求項 1 から 19 までのいずれか一項に記載の方法。

【請求項 21】

各メタデータ項目は、関連するデータ・グラニューールが能力を指定するかどうかを識別する能力タグである、請求項 1 から 20 までのいずれか一項に記載の方法。

10

20

30

40

50

【請求項 22】

各メタデータ項目が、前記関連するデータ・グラニュールの割振りポリシーを識別する割振りタグである、請求項 1 から 21 までのいずれか一項に記載の方法。

【請求項 23】

プログラム命令を含むコンピュータ・プログラムであって、前記プログラム命令は、コンピュータ・プロセッサによって実行されたとき、前記コンピュータ・プロセッサが、処理回路上で実行されるべきプログラムをデバッグする方法を実施するためにデバッガとして動作することを引き起こし、前記処理回路が、関連するメタデータ項目を有するデータ・グラニュールから形成されたデータに対して動作し、前記方法は、前記処理回路によって複数回にわたってメタデータ・アクセス動作を引き起こすために指定された数のメタデータ項目にアクセスすることを決めるステップと、

10

前記処理回路が、前記指定された数のメタデータ項目の少なくともサブセットにアクセスするために前記メタデータ・アクセス動作の複数のインスタンスを実施することを引き起こすために、前記処理回路に少なくとも 1 つのコマンドを発行するステップであって、前記メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数は、前記デバッガにとって非決定論的であり、前記少なくとも 1 つのコマンドに従って、前記デバッガが、アクセスされる前記数のメタデータ項目の進捗を決定するために前記メタデータ・アクセス動作の各インスタンスの間に前記処理回路に問い合わせることなしに、前記メタデータ・アクセス動作の前記複数のインスタンスが前記処理回路によって実施される、ステップと

20

を含む、コンピュータ・プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本開示は、処理回路上で実行されるべきプログラムをデバッグするときにメタデータにアクセスするための技法に関する。

【背景技術】

【0002】

特に、処理回路は、関連するメタデータ項目 (metadata item) を有するデータ・グラニュール (data granule) から形成されたデータに対して動作するように構成され得る。データ・グラニュールのサイズは実装に応じて変動し得、実際、各データ・グラニュールに関連するメタデータ項目のサイズも、実装に応じて変動し得る。

30

【0003】

処理回路上で実行されるべきプログラムをデバッグするとき、デバッガは、いくつかの時点において、対応する一連のデータ・グラニュールに関係する一連のメタデータ項目にアクセスすることを望み得る。たとえば、それらのメタデータ項目の各々の値を読み取ることが所望され得る。代替的に、それらのメタデータ項目の各々の現在の値を更新するために書込み動作を実施することが所望され得る。

【0004】

40

処理回路は、メタデータ項目にアクセスするためのメタデータ・アクセス動作の実施をサポートし得、デバッガは、処理回路がそのようなメタデータ・アクセス動作を実施することを引き起こすために、処理回路にコマンドを発行することができる。しかしながら、知られている機構によれば、デバッガは、メタデータ・アクセス動作を実行するときにいくつかのメタデータ項目が処理回路によってアクセスされるかを知らず、したがって、デバッガは、メタデータ・アクセス動作の実施の後に処理回路に問い合わせ、いくつかのメタデータ項目がアクセスされたかを決定し、それに応じて、デバッガがアクセスすることを望むメタデータ項目の総数にアクセスするためにメタデータ・アクセス動作のさらなる反復が必要とされるかどうかを決定することが必要である。

【発明の概要】

50

【発明が解決しようとする課題】

【0005】

しかしながら、各メタデータ・アクセス動作の実施の後に処理回路に問い合わせるプロセスは、性能に著しく影響を及ぼし、したがって、デバッガが指定された数のメタデータ項目にアクセスすることを可能にするためのより効率的な機構を提供することが望ましい。

【課題を解決するための手段】

【0006】

第1の例示的な構成では、処理回路上で実行されるべきプログラムをデバッグするようにデバッガを動作させる方法であって、処理回路が、関連するメタデータ項目を有するデータ・グラニュールから形成されたデータに対して動作し、本方法は、指定された数のメタデータ項目にアクセスすることを決めるステップであって、指定された数は、メタデータ・アクセス動作が、指定された数のメタデータ項目にアクセスするために処理回路によって複数回実施される必要があるようなものである、ステップと、処理回路が、指定された数のメタデータ項目の少なくともサブセットにアクセスするためにメタデータ・アクセス動作の複数のインスタンスを実施することを引き起こすために、処理回路に少なくとも1つのコマンドを発行するステップであって、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数が、メタデータ・アクセス動作からデバッガによって非決定論的 (non-deterministic) であり、少なくとも1つのコマンドは、デバッガが、アクセスされる数のメタデータ項目の進捗を決定するためにメタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせることなしに、メタデータ・アクセス動作の複数のインスタンスが処理回路によって実施されるようなものである、ステップとを含む、方法が提供される。

【0007】

別の例示的な構成では、プログラム命令が記憶された非一時的機械可読記憶媒体を備えるコンピュータ・プログラム製品であって、プログラム命令は、コンピュータ・プロセッサによって実行されたとき、コンピュータ・プロセッサが、処理回路上で実行されるべきプログラムをデバッグする方法を実施するためにデバッガとして動作することを引き起こし、処理回路が、関連するメタデータ項目を有するデータ・グラニュールから形成されたデータに対して動作し、上記方法は、指定された数のメタデータ項目にアクセスすることを決めるステップであって、指定された数は、メタデータ・アクセス動作が、指定された数のメタデータ項目にアクセスするために処理回路によって複数回実施される必要があるようなものである、ステップと、処理回路が、指定された数のメタデータ項目の少なくともサブセットにアクセスするためにメタデータ・アクセス動作の複数のインスタンスを実施することを引き起こすために、処理回路に少なくとも1つのコマンドを発行するステップであって、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数が、メタデータ・アクセス動作からデバッガによって非決定論的であり、少なくとも1つのコマンドは、デバッガが、アクセスされる数のメタデータ項目の進捗を決定するためにメタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせることなしに、メタデータ・アクセス動作の複数のインスタンスが処理回路によって実施されるようなものである、ステップとを含む、コンピュータ・プログラム製品が提供される。

【0008】

本技法は、単に例として、添付の図面に示されているその実例を参照しながらさらに説明される

【図面の簡単な説明】

【0009】

【図1】一実例によるシステムのブロック図である。

【図2】1つの例示的な構成による、メタデータ・アクセス動作の実施を制御するためのデバッガの動作を示す流れ図である。

【図3】異なる形式の制約情報 (constraining information) についてのデバッガの動作を示す流れ図である。

10

20

30

40

50

【図 4】異なる形式の制約情報についてのデバッガの動作を示す流れ図である。

【図 5】異なる形式の制約情報についてのデバッガの動作を示す流れ図である。

【図 6 A】複数のメタデータ項目にアクセスするために非デバッグ・モードでプロセッサによって実行され得る例示的なソフトウェア・コード・シーケンスを示す図である。

【図 6 B】プロセッサがデバッグ動作モードにある間にメタデータ項目にアクセスしようとするときにデバッガによって発行され得る例示的なコマンド・シーケンスを示す図である。

【図 7 A】メタデータ項目が各データ・グラニュールに関連してどのように提供され得るかを示す図である。

【図 7 B】メタデータ項目が各データ・グラニュールに関連してどのように提供され得るかを示す図である。

10

【図 8】1つの例示的な構成における、複数のメタデータ項目に書き込むために使用され得るデバッガ始動コマンド・シーケンス (`debugger initiated command sequence`) を示す図である。

【図 9 A】1つの例示的な構成において実装され得るアドレス・ラッピング機能 (`address wrapping function`) を示す図である。

【図 9 B】1つの例示的な構成において実装され得るアドレス・ラッピング機能を示す図である。

【図 10】デバッガからの少なくとも1つのコマンドに応答するために動作生成回路が処理回路内に組み込まれている、代替の例示的な構成によるシステムを示す図である。

20

【図 11】メタデータ項目が能力タグである例示的な構成を示す図である。

【図 12】メタデータ項目が割振りタグである別の例示的な構成を示す図である。

【発明を実施するための形態】

【0010】

1つの例示的な構成によれば、処理回路上で実行されるべきプログラムをデバッグするようにデバッガを動作させる方法が提供され、特に、デバッガが指定された数のメタデータ項目にアクセスすることを望むことをデバッガが決めたとき、処理回路とのデバッガの対話を制御するための方法が提供され、ここで、指定された数は、メタデータ・アクセス動作が、指定された数のメタデータ項目にアクセスするために処理回路によって複数回実施される必要があるようなものである。メタデータ・アクセス動作の使用は、デバッガがメタデータ項目にアクセスするために利用可能な唯一の機構でないことがあるが、本開示では、指定された数のメタデータ項目にアクセスしようとするためにデバッガによって使用されるものは、メタデータ・アクセス動作手法であることに留意されたい。

30

【0011】

1つの例示的な構成によれば、指定された数のメタデータ項目にアクセスする必要をデバッガが決定すると、デバッガは、処理回路が、指定された数のメタデータ項目の少なくともサブセットにアクセスするためにメタデータ・アクセス動作の複数のインスタンスを実施することを引き起こすために、処理回路に少なくとも1つのコマンドを発行するように構成される。メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数は、メタデータ・アクセス動作からデバッガによって非決定論的であり、したがって、メタデータ・アクセス動作が処理回路によって実施される前に、デバッガは、処理回路がそのメタデータ・アクセス動作を実行するときにいくつのメタデータ項目が処理回路によってアクセスされるかを、メタデータ・アクセス動作自体から決定することができない。

40

【0012】

ただし、説明される機構によれば、少なくとも1つのコマンドは、デバッガが、アクセスされる数のメタデータ項目の進捗を決定するためにメタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせることなしに、メタデータ・アクセス動作の複数のインスタンスが処理回路によって実施されるようなものである。特に、発行される1つ又は複数のコマンドは、アクセスすることが所望される指定された数外のメタデータ項目に

50

処理回路がアクセスする見込みなしに、メタデータ・アクセス動作の複数のインスタンスが実施され得るように構成され得る。デバッガが各インスタンスの間に処理回路に問い合わせる必要なしに、メタデータ・アクセス動作の複数のインスタンスが実施され得るので、これは、デバッガ制御の下で複数のメタデータ項目にアクセスするときの性能を著しく改善することができる。

【0013】

一実例では、少なくとも1つのアクセス・インジケータが、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数に関する制約情報を示すために提供され、少なくとも1つのコマンドに応答して実施されるメタデータ・アクセス動作の複数のインスタンスは、少なくとも1つのアクセス・インジケータを参照して決定される。少なくとも1つのアクセス・インジケータは、様々な形式をとることができる。たとえば、少なくとも1つのアクセス・インジケータは、処理回路の特定の実装詳細とは無関係である何らかの所定の情報であり得る。代替的に、少なくとも1つのアクセス・インジケータは、識別レジスタなど、記憶要素内に指定可能であり得、処理回路のいくつかの実装詳細を考慮に入れる制約情報を示すことができる。1つの例示的な構成では、デバッガ自体が、少なくとも1つのアクセス・インジケータを参照し、実施されるべきメタデータ・アクセス動作の複数のインスタンスを直接識別するコマンドのシーケンスを発行することができる。しかしながら、代替手法では、制約情報にアクセスし、デバッガによって発行されたコマンドに応答して、実施されるべきメタデータ・アクセス動作の複数のインスタンスを決定し、必要とされる動作の実施を制御することができる、処理回路内の動作生成回路があり得る。

【0014】

一実例では、少なくとも1つのアクセス・インジケータは、デバッガによって処理回路に発行された少なくとも1つのコマンドが、処理回路が、メタデータ・アクセス動作の複数のインスタンスとして、指定された数のメタデータ項目のすべてにアクセスするのに十分な数の、メタデータ・アクセス動作のインスタンスを実施することを引き起こすことを可能にするのに十分である。したがって、そのような構成では、メタデータ・アクセス動作の複数のインスタンスが実施される時間までに、必要とされるメタデータ項目のすべてがアクセスされることが確実にされ得る。これは、デバッガが処理回路に問い合わせる必要なしに行われ得るので、その場合、プロセス全体は、進捗を決定するためのデバッガによる処理回路への問合せなしに実施され、それにより、特に効率的な実装形態が提供され得る。

【0015】

そのようなやり方で使用され得るアクセス・インジケータの一実例として、アクセス・インジケータは、処理回路の実装詳細に固有でない一般的な最小指示を提供し得、それは、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最小数を示す。単純な実例として、メタデータ・アクセス動作の各インスタンスを実施するときに単一のメタデータ項目のみがアクセスされると仮定され得る。したがって、その実例では、実施されるメタデータ・アクセス動作の複数のインスタンスは、デバッガがアクセスすることを望むメタデータ項目の指定された数に等しい。後でより詳細に説明されるように、メタデータ・アクセス動作の各インスタンスのために使用されるアドレスの適切な制御によって、メタデータ・アクセス動作のインスタンスが、指定された数のメタデータ項目に関連するアドレス範囲外のメタデータ項目にアクセスしないことが確実にされ得る。

【0016】

メタデータ・アクセス動作の複数のインスタンスを実施することによって、指定された数のメタデータ項目のすべてがアクセスされることを可能にすることができる好適なアクセス・インジケータの別の実例として、アクセス・インジケータは、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最小数を示す、処理回路についての実装最小指示を提供し得る。これは、たとえば、1よりも大きい数を最小数

10

20

30

40

50

として識別し、したがって、実施される必要があるメタデータ・アクセス動作のインスタンスの数を低減し得る。この場合も、メタデータ・アクセス動作によって使用されるアドレス指定機構の適切な制御によって、指定された数に関連するアドレス範囲内のメタデータ項目のみがアクセスされることが確実にされ得る。

【0017】

使用され得るアクセス・インジケータの別の実例として、アクセス・インジケータは、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数の正確な指示を提供し得る。この場合も、そのような形式のアクセス・インジケータは、処理回路に送られた1つ又は複数のコマンドが、デバッガがメタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせる必要なしに、指定された数のメタデータ項目のすべてにアクセスするのに十分である、メタデータ・アクセス動作の複数のインスタンスの実施をトリガすることを可能にすることができる。

10

【0018】

いくつかの例示的な構成では、アクセス・インジケータは、デバッガが処理回路に問い合わせる必要なしに実施され得る、メタデータ・アクセス動作の複数のインスタンスを決定するのを支援し得るが、アクセス・インジケータは、メタデータ・アクセス動作の複数のインスタンスが実施された時間までに、指定された数のメタデータ項目のすべてがアクセスされたかどうかを確実に知られることになるという十分な情報を提供しないことがある。たとえば、少なくとも1つのアクセス・インジケータは、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最大数を示す、処理回路についての実装最大指示を提供し得る。

20

【0019】

そのような形式のアクセス・インジケータの場合、この場合も、デバッガが各インスタンスの間に処理回路に問い合わせる必要なしに実施され得る、メタデータ・アクセス動作の複数のインスタンスが決定され得る。しかしながら、メタデータ・アクセス動作のその複数のインスタンスが実施されると、指定された数のメタデータ項目のすべてがアクセスされたか否かと、したがって、メタデータ・アクセス動作の追加のインスタンスが必要とされるかどうかを決定するために、さらなる検査が必要である。

【0020】

特に、1つの例示的な構成では、デバッガによって処理回路に発行された少なくとも1つのコマンドは、処理回路が、メタデータ・アクセス動作の複数のインスタンスとして、M個のインスタンスを実施することを引き起こし、ここで、Mは、メタデータ項目の指定された数を、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の最大数で除算したものに等しい。処理回路がメタデータ・アクセス動作の複数のインスタンスを実施したとき、デバッガは、次いで、メタデータ・アクセス動作のさらなるインスタンスが必要とされるかどうかを決定するために、アクセスされる数のメタデータ項目の進捗を決定するために、処理回路に問い合わせるように構成され得る。

30

【0021】

1つの例示的な構成では、デバッガによって発行された少なくとも1つのコマンドは、メタデータ・アクセス動作の各インスタンスについてアドレスが処理回路によって決定されることを引き起こし、アドレスは、メタデータ・アクセス動作の関連するインスタンスの実施中にアクセスされる1つ又は複数のメタデータ項目を識別するために使用される。メタデータ・アクセス動作が処理回路によって実施されるとき、これが、アクセスされるメタデータ項目の数を考慮に入れて、処理回路内のアドレス指示が自動的に更新されることを引き起こすことが起こり得る。しかしながら、メタデータ・アクセス動作の複数のインスタンスは、デバッガがメタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせることなしに実施されるべきであるので、そのアドレスは直接使用されず、代わりに、デバッガによって発行された1つ又は複数のコマンドは、メタデータ・アクセス動作の各インスタンスについてアドレスが決定されることを引き起こす。そのような手法によって、メタデータ・アクセス動作の複数のインスタンスが実施されるとき、デバッガ

40

50

がアクセスしようとしている指定された数のメタデータ項目に関連するアドレス範囲外にあるメタデータ項目がアクセスされないことが確実にされ得る。

【0022】

メタデータ・アクセス動作の各インスタンスについてのアドレスが決定され得るいくつかのやり方があるが、一実施例では、アドレスは、少なくとも1つのアクセス・インジケータを顧慮して決定される。たとえば、アドレスは、メタデータ項目に関連するデータ・グラニュールを識別するために使用され得る。アクセス・インジケータに基づいて、L個のメタデータ項目がメタデータ・アクセス動作の各インスタンスによってアクセスされるという仮定が行われた場合、アドレスは、1つの例示的な実装形態では、各反復の間にL×データ・グラニュール・サイズだけ増分され得る。

10

【0023】

アドレスがメタデータ・アクセス動作の各インスタンスについて生成され得るいくつかのやり方がある。しかしながら、一実施例では、少なくとも1つのコマンドは、メタデータ・アクセス動作の各インスタンスについてのコマンド・シーケンスを備え、コマンド・シーケンスは、少なくとも、メタデータ・アクセス動作の対応するインスタンスと、メタデータ・アクセス動作の対応するインスタンスによって使用されるべきアドレスを決定するためのアドレス生成動作とを備える、動作のシーケンスを識別する。したがって、コマンド・シーケンスは、少なくとも、アドレス生成動作と、メタデータ・アクセス動作の対応するインスタンスとを含む、動作のシーケンスを識別する。他の動作も、実施されているアクセスの性質、たとえばメタデータ項目が読み取られているのか書き込まれているのかに応じて、コマンド・シーケンス内に含まれ得る。

20

【0024】

上記の例示的な構成では、コマンド・シーケンスは、アドレスを決定するために実施される特定のアドレス生成動作を識別するが、代替構成では、そのようなアドレス生成動作を省略し、代わりに、メタデータ・アクセス動作のインスタンスが実施されるときに処理回路内で実施されるアドレスの自動更新に依拠することが可能であり得る。特に、1つの例示的な構成では、アドレス空間領域が、アクセスされることになるメタデータ項目の指定された数に基づいて識別される。次いで、デバッグによって誘発されるメタデータ・アクセス動作の現在のインスタンスの実施中に、処理回路は、メタデータ・アクセス動作の次のインスタンスについてのアドレスとして使用されるべき更新されたアドレスを生成するように構成され、更新されたアドレスは、現在のインスタンスによってアクセスされるメタデータ項目の数を考慮に入れるために現在のインスタンスによって使用されるアドレスを修正すること、ただし、更新されたアドレスがアドレス空間領域内に残ることを確実にするために必要に応じてアドレス空間領域のある端部からアドレス空間領域の別の端部までラッピングするためのラッピング機能を採用することによって生成される。そのような状況においてラッピング機能が採用されることを確実にすることによって、メタデータ・アクセス動作の複数のインスタンスが、デバッグが進捗について処理回路に問い合わせることなしに実施されるが、意図されていないメタデータ項目へのアクセスが行われず、特に、メタデータ・アクセス動作の様々なインスタンスが、当該の指定された数のメタデータ項目に関連するアドレス範囲内のメタデータ項目のみにアクセスするように制約されることが確実にされ得る。

30

40

【0025】

しかしながら、そのようなラッピング機能性 (functionality) は、処理回路がプログラムの通常実行中にそのようなメタデータ・アクセス動作を実施しているとき、適切でないことがある。特に、プログラムは、処理回路によって実行されたとき、処理回路がメタデータ・アクセス動作を実施することを引き起こすメタデータ・アクセス命令を含み得る。しかしながら、非デバッグ・モードでは、ラッピング機能制約がアドレス増分プロセスに課されないことが好ましいことがある。したがって、1つの例示的な構成では、処理回路の非デバッグ動作モードにおいて、メタデータ・アクセス動作の実施は、更新されたアドレスを生成したときにラッピング機能が無効にされるように、処理回路が

50

メタデータ・アクセス動作を実施することを引き起こす。

【0026】

1つの例示的な構成では、デバッガは、ラッピング機能を選択的に無効にするように処理回路内の制御記憶要素をセットするように構成され得る。そのような手法によって、ラッピング機能は、デバッガの制御下で、必要に応じてオン及びオフにされ得る。一実施例では、デバッグ・モードを出るとき、デバッガは、ラッピング機能は無効にするように制御記憶要素をセットし得、したがって、処理回路が標準動作モードで動作するとき、そのようなラッピング機能制約はメタデータ・アクセス動作の実行に課されない。

【0027】

1つの例示的な構成では、少なくとも1つのコマンドは、メタデータ・アクセス動作の各インスタンスについての1つ又は複数のコマンドを備える。したがって、デバッガは、関連する1つ又は複数のコマンドを発行することによってメタデータ・アクセス動作の各インスタンスの実施を直接制御する。

10

【0028】

しかしながら、代替構成では、デバッガによって発行された少なくとも1つのコマンドは、処理回路がメタデータ・アクセス動作の複数のインスタンスを実施するために必要とされる動作を作成するために、処理回路内の動作生成回路によって処理される。1つの例示的なシナリオでは、デバッガが動作生成回路に単一のコマンドを発行することが十分であり得、処理回路内の動作生成回路が、次いで、メタデータ・アクセス動作の複数のインスタンスを実施するために処理回路によって実施されるべき必要とされる動作を作成する。

20

【0029】

実施されるアクセス動作のタイプは変動され得る。たとえば、アクセス動作は読取り動作であり得、したがって、処理回路によってメタデータ・アクセス動作を複数回実施することは、指定された数のメタデータ項目が取り出され、選定されたレジスタに記憶されることを引き起こす。その後、デバッガは、メタデータ項目のすべてを取得するために、その選定されたレジスタにアクセスすることができる。

【0030】

代替実施例では、アクセス動作は書込み動作であり得、したがって、処理回路によってメタデータ・アクセス動作を複数回実施することは、指定された数のメタデータ項目の値が、選定されたレジスタに記憶されたメタデータ値を使用して更新されることを引き起こす。そのような場合、選定されたレジスタは、メタデータ・アクセス動作の複数のインスタンスの実施を始動する前に、必要とされる値でポピュレートされ得る。

30

【0031】

アクセスされることになるメタデータ項目の指定された数が決定され得るいくつかのやり方がある。しかしながら、一実施例では、指定された数は、選定されたレジスタに記憶され得るメタデータ項目の数によって決定される。したがって、単に実施例として、選定されたレジスタが64ビット・レジスタであり、各メタデータ項目がシングル・ビットである場合、指定された数は64であり得る。

【0032】

各メタデータ項目は、メタデータの形式に応じて1つ又は複数のビットを備え得る。特定の事例として、各メタデータ項目は、関連するデータ・グラニュールが能力を指定するかどうかを識別する能力タグであり得る。いくつかの能力が所与のプロセスについて定義され、定義された能力外の動作を行おうとする試みがある場合にエラーがトリガされ得る能力ベース・アーキテクチャへの関心が増加している。有界ポインタ (bounded pointer) が能力の一実施例である。ポインタ自体が、たとえば、アクセスされることになるデータ値又は実行されるべき命令のアドレスを指すか、又はそれを決定するために使用され得る。しかしながら、ポインタはまた、ポインタを使用するときのアドレスの許容範囲を示す関連する範囲情報を有し得る。これは、たとえば、挙動のセキュリティ又は機能的正当性を維持するために、ポインタから決定されたアドレスがある限界内に残ることを確実にするために、有用であり得る。そのような能力ベース・システム内では、特

40

50

定のデータ・グラニューールが能力に関係するのか汎用データに関係するのかを知ることは重要であり、能力タグは、この目的で使用され得る。

【0033】

メタデータ項目の別の事例として、メタデータ項目は、関連するデータ・グラニューールの割振りポリシーを識別する割振りタグであり得る。そのような割振りタグは、ガード・タグと呼ばれることもある。いくつかのメモリ使用エラーから保護するための1つの手法は、1つ又は複数のメモリ・ロケーションのブロックに関連してメモリ・システムに記憶されるガード・タグを提供することであり得る（それらの1つ又は複数のメモリ・ロケーション中のデータがデータ・グラニューールである）。タグガード（tag-guard）・メモリ・アクセス動作が、メモリ・システム中の特定のアドレス指定されたロケーションを識別するターゲット・アドレスに基づいて要求されたとき、メモリ・アクセス回路は、ターゲット・アドレスに関連するアドレス・タグを、ターゲット・アドレスによって識別されたアドレス指定されたロケーションを含む1つ又は複数のメモリ・ロケーションのブロックに関連してメモリ・システムに記憶されたガード・タグと比較し得る。メモリ・アクセス回路は、ガード・タグとアドレス・タグとの間の一致が検出されたかどうかの指示を生成し得る。この指示は、メモリ・アクセスが成功することを可能にされるかどうか、又は、後続の動作が成功することができるかどうかを制御するために使用され得るか、或いは、メモリ・アクセスが通常通り続くことを可能にしながら報告され得るにすぎない。

【0034】

本明細書で説明される技法を実装するために使用されるデバグは様々な形態をとることができ、たとえば専用ハードウェアであり得るか、又は代替的に、汎用コンピュータ上で実行しているソフトウェアによって形成され得ることが諒解されよう。したがって、コンピュータによって実行されたとき、コンピュータが、前述されたデバグとして動作することを引き起こすプログラム命令が記憶されたコンピュータ・プログラム製品が提供され得る。

【0035】

次に、図を参照しながら特定の実例が説明される。

【0036】

図1は、1つの例示的な構成によるシステムのブロック図である。処理回路15上で実行されるべきであるプログラム45をデバグするためのデバグ10が提供される。処理回路15は、デバグ・アクセス・ポート20を有し、デバグ・アクセス・ポート20を介して、デバグ10は、処理回路がいくつかの動作を実施することを引き起こすために、処理回路にコマンドを発行することができる。処理回路は様々な形態をとることができるが、1つの例示的な構成では、中央処理ユニット（CPU：central processing unit）などのプロセッサ・コアである。データに対して動作を実施するための実行パイプライン25が提供される。それらの動作の実施中に、実行パイプライン25は、作業レジスタ30へのアクセスを有し、実行パイプライン25は、動作によって処理されるべきデータ値を作業レジスタ30から読み取ることができ、それらの動作の結果を作業レジスタ30に書き込むことができる。実行パイプライン25は、様々なレベルのキャッシュ及びメイン・メモリからなるメモリ・システムへのアクセスを有する。図示の実例では、レベル1キャッシュ35が処理回路15内に提供され、レベル1キャッシュ35は、次いで、1つ又は複数のさらなるレベルのキャッシュ及びメイン・メモリ40に結合される。メモリ35、40からのデータをレジスタ30にロードするために、実行パイプライン25によってロード動作が実施され得、さらに、レジスタ30からのデータをメモリ・システム35、40に記憶するために、実行パイプラインによって記憶動作が実施され得る。

【0037】

通常の非デバグ動作モード中に、実行パイプラインによって実施される動作は、処理回路上で実行されるプログラムを形成する命令によって制御される。しかしながら、デバ

10

20

30

40

50

ッグ動作モードでは、実行パイプラインによって実施される動作は、デバッグ・アクセス・ポート 20 を介してデバッガ 10 から発行されたコマンドによって直接制御され得る。

【0038】

処理回路 15 がそれに対して動作するデータは、関連するメタデータ項目を有するデータ・グラニューールから形成される。データ・グラニューールのサイズは、実装又は何らかの設定可能な挙動に応じて変動することができるが、以下で説明される特定の事例では、データ・グラニューールのサイズが 16 バイト（128 ビット）であると仮定される。さらに、関連するメタデータ項目を形成するビットの数は、実装又は何らかの設定可能な挙動に応じて変動することができるが、メタデータ項目のサイズが 1 ビット又は 4 ビットのいずれかである 2 つの特定の事例が後で説明される。

【0039】

処理回路上で実行されるべきプログラムをデバッグするとき、デバッガが、対応する一連のデータ・グラニューールに関連する一連のメタデータ項目にアクセスすることは、時々有用である。たとえば、それらのメタデータ項目の値を読み取ること、又は、それらのメタデータ項目の値を更新するために書き込み動作を実施することが所望され得る。処理回路 15 は、メタデータ項目にアクセスするためのメタデータ・アクセス動作の実施をサポートするが、デバッガは、メタデータ・アクセス動作が実施されるときにいくつかのメタデータ項目がアクセスされるかを、そのメタデータ・アクセス動作が処理回路によって実施される前に、メタデータ・アクセス動作自体から決定することができない。本明細書で説明される技法は、デバッガが、メタデータ・アクセス動作の複数のインスタンスが処理回路によって実施されることを引き起こすために、処理回路に 1 つ又は複数のコマンドを発行し、特に、アクセスされる数のメタデータ項目に関する進捗についてデバッガが処理回路に問い合わせる必要がある程度を低減するための効率的な機構を提供する。

【0040】

後でより詳細に説明されるように、制約情報 50 は、メタデータ・アクセス動作の各インスタンスによってアクセスされるメタデータ項目の数に関するある情報を提供するために使用され得、その制約情報は、デバッガが、アクセスされる数のメタデータ項目の進捗を決定するためにメタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせる必要なしに、メタデータ・アクセス動作のいくつかのインスタンスがデバッガ制御の下で安全に始動され得るかを決定するときに使用され得る。制約情報は、デバッガによって仮定され得る何らかの所定の情報の形式をとり得るか、又は、代わりに、そのような制約情報が指定され、したがって制約情報が処理回路の特定の实装形態に固有であることを可能にし得る、識別レジスタなどの明示的ストレージが提供され得る。

【0041】

制約情報を使用することによって、デバッガが 1 つ又は複数のコマンドを発行することが可能であり、1 つ又は複数のコマンドは、次いで、当該の指定された数のメタデータ項目外のメタデータ項目に処理回路がアクセスする見込みなしに、メタデータ・アクセス動作の複数のインスタンスが実施されることを引き起こす。1 つの例示的な構成では、これは、メタデータ・アクセス動作の各インスタンスによって使用されるメモリ・アドレスの制御を通して達成される。1 つの特定の事例では、これは、デバッガから発行されたコマンド・シーケンスに基づいて各反復についてアドレスが生成されることによって達成される。しかしながら、代替事例では、アクセスされるメタデータ項目の数に基づいて、ただし、アドレス生成プロセスにラッピング制約を課すことによって、メタデータ・アクセス動作の各インスタンスの実施に続いて処理回路によって直接生成される更新されたアドレスが使用される。これは、メタデータ・アクセス動作の複数のインスタンスが、各インスタンスの実施の間の進捗に関するデバッガによる問合せなしに実施されるとき、当該の指定された数のメタデータ項目内のメタデータ項目のみがアクセスされるように、使用されるアドレスが制約されることを確実にする。後でより詳細に説明されるように、1 つの例示的な構成では、制御記憶要素 55 が提供され得、制御記憶要素 55 の値は、デバッグ・モードの間にラッピング制御機能性を選択的に有効にするように、又は、処理回路がデバ

10

20

30

40

50

ッグ・モードで動作していないときにラッピング制御機能性を選択的に無効にするように、デバッガの制御下でセットされ得る。

【0042】

図2は、処理回路15によるメタデータ・アクセス動作の実施を制御するときのデバッガ10の動作を示す流れ図である。ステップ100において、デバッグ・プロセス中に、デバッガは、指定された数のメタデータ項目にアクセスする必要を決定する。特に、指定された数は、メタデータ・アクセス動作が、その指定された数のメタデータ項目にアクセスするために処理回路によって複数回実施される必要があるようなものである。前述のように、メタデータ・アクセス動作が実施されるたびに処理回路によってアクセスされるメタデータ項目の数は、メタデータ・アクセス動作自体からデバッガ10にとって非決定論的である。しかしながら、ステップ105において示されるように、制約情報50を参照して、デバッガが各インスタンスの実施の間に処理回路に問い合わせる必要なしに実施され得る、メタデータ・アクセス動作の複数のインスタンスが決定され得る。特に、制約情報が与えられれば、当該の指定された数外のメタデータ項目にアクセスする見込みなしに、メタデータ・アクセス動作のその複数のインスタンスが実施されることが安全であると決定され得る。1つの例示的な構成では、これは、メタデータ・アクセス動作の各インスタンスによって使用されるアドレスを識別するために使用されるアドレス生成プロセスの適切な制御によって達成される。

10

【0043】

複数のインスタンスが決定されると、次いで、ステップ110において、メタデータ・アクセス動作の複数のインスタンスが実施されることを引き起こすために、1つ又は複数のコマンドがデバッガによって処理回路15に発行される。上述のように、各インスタンスについてのアドレスは、指定された数のメタデータ項目を含んでいるアドレス範囲内にそのアクセスがとどまることを確実にするように制約される。

20

【0044】

利用可能な制約情報のタイプに応じて、メタデータ・アクセス動作の複数のインスタンスが実施された時間までに、指定された数のメタデータ項目がすべてアクセスされたかどうか知られ得る。したがって、ステップ115において、すべての必要とされるメタデータ項目がアクセスされたことを知るのに制約情報が十分であったかどうか決定される。制約情報が十分であった場合、プロセスはステップ130において終了する。しかしながら、制約情報が十分でなかった場合、プロセスはステップ120に進み、ここで、デバッガが、アクセスされる数のメタデータ項目に関する進捗を決定するために処理回路15に問い合わせるために、デバッグ・アクセス・ポート20を介して1つ又は複数のコマンドを発行する。その結果、次いで、ステップ125において、より多くのメタデータ項目がアクセスされる必要があるか否かが決定される。より多くのメタデータ項目がアクセスされる必要がない場合、プロセスはステップ130において終了する。しかしながら、他の場合、プロセスは、処理回路に問い合わせることなしに実施され得るメタデータ・アクセス動作のいくつかのさらなるインスタンスを決定するために、ステップ105に戻る。図2に示されているプロセスは、次いで、指定された数のメタデータ項目のすべてがアクセスされたことが知られるまで、続くことができる。図2のプロセスを通して最後の反復時に、ステップ105において、メタデータ・アクセス動作の単一のさらなるインスタンスのみが必要とされることが、起こり得ることに留意されたい。単に具体的な実例として、ステップ125において、1つの残りのメタデータ項目がアクセスされる必要があると決定され、少なくとも1つのメタデータ項目がメタデータ・アクセス動作の各インスタンスによってアクセスされることを制約情報が示す場合、その段階において、単一のメタデータ・アクセス動作のみが、ステップ105、110の最後の反復中に実施される必要がある。

30

40

【0045】

制約情報は、様々な形式をとることができる。図3は、制約情報が、メタデータ・アクセス動作が実施されるたびにアクセスされるメタデータ項目の最小数Yを識別する、1つ

50

の例示的なシナリオのためのプロセスを示す。そのような形式の制約情報を使用すると、メタデータ・アクセス動作の複数のインスタンスを構築することが可能であり、メタデータ・アクセス動作の複数のインスタンスは、各インスタンスの実施の間に処理回路に問い合わせることなしに実施され得、それに関して、メタデータ・アクセス動作のその複数のインスタンスが実施された時間までに、指定された数のメタデータ項目のすべてがアクセスされたことが知られている。これは図3に示されており、ここで、ステップ150において、アクセスされているメタデータ項目の指定された数が「X」であると仮定される。最小数「Y」に関して、これは、たとえば、処理回路の実装詳細に固有でない何らかの一般的な最小指示であり得る。単に実例として、たとえば、メタデータ・アクセス動作が実施されるたびに少なくとも1つのメタデータ項目がアクセスされると決定され得る。代替的に、値Yは、問題の処理回路を顧慮した実装最小指示であり得、したがって、潜在的に、一般的な最小指示よりも大きい値であり得る。したがって、単に具体的な実例として、そのインスタンスにおける最小指示は、メタデータ・アクセス動作が実施されるたびに少なくとも2つのメタデータ項目がアクセスされることを識別し得る。

【0046】

ステップ155において、反復カウンタNは1に等しくセットされ、その後、プロセスはステップ160に進み、ここで、処理回路が1つ又は複数の動作を実施することを引き起こすために、少なくとも1つのコマンドがデバッガ10によって処理回路15に発行され、それらの動作はメタデータ・アクセス動作のインスタンスを含む。メタデータ・アクセス動作のインスタンスに加えて、いくつかの他の動作が実施され得る。たとえば、1つの例示的な構成では、アドレス生成動作も、メタデータ・アクセス動作のインスタンスを実施する前に実施され、アドレス生成動作は、次いでメタデータ・アクセス動作によって使用される、アドレスを決定するために使用される。実施されているアクセス動作のタイプに応じて、さらなる動作も必要とされ得る。たとえば、読取り動作が実施されているとき、読み取られたメタデータ項目は、選定されたレジスタ内で照合され得、したがって、メタデータ・アクセス動作の各インスタンスが実施された後に、その特定のインスタンスによって読み取られたメタデータ項目は、次いで、レジスタ内の前に読み取られたメタデータ項目とマージされる必要がある。

【0047】

ステップ165において、様々な動作がステップ160において実施された後に、反復カウンタNがX/Y以上であるかどうか決定される。反復カウンタNがX/Y以上でない場合、反復カウンタNはステップ170において増分され、プロセスは160に戻る。しかしながら、反復カウンタNがX/Y以上であると決定された場合、プロセスはステップ175に進み、ここで、プロセスは終了する。この時点で、X個のメタデータ項目のすべてがアクセスされたことが知られている。したがって、「アクセスされるメタデータ項目の最小数」制約情報への依拠により、厳密に必要とされるよりも多くのアクセス動作が実施されることになり得るが、図3のプロセスが完了する時間までに、必要とされるメタデータ項目のすべてがアクセスされたことが知られる。さらに、これは、デバッガが進捗について処理回路に問い合わせる必要なしに達成される。進捗を決定するために処理回路に問い合わせるプロセスは、性能に著しく影響を及ぼすことがあり、したがって、処理回路に問い合わせる必要を完全に回避することによって、これは、メタデータの複数の項目にアクセスするための極めて効率的な機構につながる可能性がある。

【0048】

デバッガが処理回路に問い合わせることなしに、メタデータ・アクセス動作の複数のインスタンスのすべてが実施されることが安全である理由は、メタデータ・アクセス動作の各インスタンスのためのアドレスが決定されるやり方による。特に、1つの例示的な構成では、ステップ160の各実施の間に、アドレスは、パラメータYに基づいて調整される。代替的に、図9A及び図9Bを参照しながら後で説明されるように、毎回、生成されたアドレスにラッピング制約が課され得る。これは、いくつかのメタデータ項目がメタデータ・アクセス動作の複数のインスタンスの実施中に2回以上アクセスされることを意味し

得るが、当該のメタデータ項目のすべてがアクセスされることが保証される。

【0049】

図4は、図3に対応する流れ図であり、ここで、ステップ200によって示されるように、制約情報は、メタデータ・アクセス動作が実施されるたびにアクセスされるメタデータ項目の正確な数Zを識別する。ステップ205、210は、図3のステップ155、160に対応し、次いで、ステップ215において、反復カウンタNが X/Z 以上であるかどうか決定される。反復カウンタNが X/Z 以上でない場合、反復カウンタはステップ220において増分され、プロセスはステップ210に戻る。しかしながら、ステップ215における条件が満たされると、その時点において、当該のX個のメタデータ項目のすべてがアクセスされたことが知られているので、プロセスはステップ225において終了する。

10

【0050】

1つの例示的な構成では、ステップ210の各実施の間に、アドレスは、パラメータZに基づいて調整される。パラメータZは、アクセスされると知られているメタデータ項目の正確な数を示すので、必要とされる数の反復のみが実施され、したがって、図3の手法に勝るさらなる実施改善が提供される。この場合も、メタデータ・アクセス動作の各インスタンスの間にプロセッサに問い合わせる必要がない。さらに、代わりに、メタデータ・アクセス動作によって作り出された更新されたアドレスが、次の反復によって使用されるアドレスを決定するために使用される場合、必要とされる数のメタデータ項目にアクセスするために最小数のメタデータ・アクセス動作のみが実施されることが知られているので、ラッピング機能性制約がアドレス生成に課される必要がない。

20

【0051】

図5は、別の例示的な構成を示すが、ここで、ボックス250によって示されるように、制約情報は、メタデータ・アクセス動作が実施されるたびにアクセスされるメタデータ項目の最大数Wを識別する。ステップ255、260、265及び270は、概して図3のステップ155、160、165、170と対応するが、ステップ265において、実施される検査は、反復カウンタが X/W 以上であるかどうかである。ステップ260の各実施の間に、アドレスがパラメータWに基づいて調整され得るか、又は、代わりに、アクセスされるアドレスが、当該のメタデータ項目に関連するアドレス範囲内にとどまることを確実にするために、前に説明されたラッピング機能性がエンフォース(enforce)され得る。第3の代替として、メタデータ・アクセス動作によって作り出された更新されたアドレスは、次の反復によって使用されるアドレスを決定するために使用され得る。図5の手法全体が、あまりに多くのアクセスが実施されないことを確実にするように構成され得るので、ラッピング機能性制約がアドレス生成に課される必要がないか又はWに基づくアドレス調整に依拠する必要がないことがある。

30

【0052】

しかしながら、図3及び図4の手法とは対照的に、この実例における制約情報は、反復カウンタNが X/W 以上であるとステップ265において決定された時間までにメタデータの必要とされる項目のすべてがアクセスされたことが保証され得ないようなものである。特に、メタデータ・アクセス動作が実施されたたびに最大数W個のメタデータ項目がアクセスされたかどうかは知られていない。したがって、ステップ275において、デバッグは、メタデータ項目にアクセスすることに関する進捗を決定するために処理回路に問い合わせるように構成される。次いで、必要とされるX個のメタデータ項目のすべてがアクセスされたかどうかステップ280において決定され、必要とされるX個のメタデータ項目のすべてがアクセスされた場合、プロセスはステップ290において終了する。しかしながら、そうでない場合、プロセスはステップ285に進み、ここで、パラメータXが、アクセスされるべき残りのメタデータ項目の数を示すようにセットされ、その後、プロセスはステップ255に戻る。処理回路の何らかの問合せが依然として必要とされるが、メタデータ・アクセス動作の各インスタンスの間に処理回路に問い合わせる必要はなく、したがって、これが依然として有意な性能利益を生じることができることが諒解されよう。

40

50

【0053】

1つの例示的な構成では、処理回路によってサポートされる命令セットは、メタデータ項目の一括アクセスのための命令を含み、その命令の実行は、前述されたメタデータ・アクセス動作が実施されることを引き起こし、その動作は、本明細書では、メタデータ項目の読取りのための `Gettagblock` 動作又はメタデータ項目への書込みのための `Settagblock` 動作とも呼ばれる。図6Aは、ループ内に `Gettagblock` 命令を含み、通常の非デバッグ動作モードにあるときに処理回路によって実行され得る例示的なコード・シーケンスを示す。最初の3つの「`register move (MOV)`」命令は、ベース・アドレスをポインタ値にセットし、レジスタ `Xaddr` をベース・アドレスにセットし、タグ・レジスタ `Xtags` をオール0に初期化するために使用される。ループは、次いで、`Gettagblock` 命令と、「論理OR (`ORR`)」命令と、「テスト (`TST`)」命令とを含む。`Gettagblock` 命令の実行は、`Gettagblock` 動作が、`Xaddr` レジスタ中のアドレスから開始して取得されたいくつかのメタデータ項目をレジスタ `Xtmp` 中に取り出すために、実施されることを引き起こす。感嘆符によって示されるように、`Xaddr` レジスタ中のアドレス値は、次いで、取り出されたメタデータ項目の数に基づいて調整される。次いで、`ORR` 命令は、`Xtmp` レジスタ中の値を `Xtags` レジスタ中にマージするために実行される。経時的に、これは、（本明細書ではタグ値とも呼ばれる）必要とされるメタデータ項目のすべてが `Xtags` レジスタ内に累積されることを可能にする。ループが実施されるたびにポピュレートされる `Xtags` レジスタ内のロケーションは、アドレスの一定数の低位ビットを使用して、処理回路によって決定され得る。

【0054】

次いで、テスト命令は、`Xaddr` レジスタに記憶されたアドレスが最後の予想されるアドレス、すなわち必要とされる数のメタデータ項目のすべてが取り出された後に存在するアドレスであるかどうかを決定する。`Xaddr` レジスタに記憶されたアドレスが最後の予想されるアドレスでない場合、プロセスがループの開始に戻ることを引き起こすために、`branch not equal (BNE)` 命令が実行される。

【0055】

単一の `Xtags` レジスタ内に収容され得るよりも多くのメタデータ項目がアクセスされる必要があることが起こり得る。その場合、シーケンス全体が、さらなる `Xtags` レジスタを用いて1回又は複数回繰り返され得る。

【0056】

上記の説明から、`Gettagblock` 命令がループ内で実行されたとき、必要とされるメタデータ項目のすべてが読み取られるまで、必要とされる回数、ループが実施されることがわかるであろう。

【0057】

図6Bは、デバッグ動作モードで同様のプロセスを実施するために使用され得るデバッグ始動コマンド・シーケンスを示す。ベース・アドレスをポインタ値にセットするために第1の動作が実施される。次いで、コマンドの以下のシーケンスが繰り返される。メタデータ・アクセス動作（`Gettagblock` 動作）の現在の関連するインスタンスのために使用されるべきアドレスを決定するために、アドレス構築コマンド（`construct address command`）が発行される。特に、ベース・アドレス値にオフセット値を加算して、次いで `Xaddr` レジスタに記憶されるアドレスを決定することによって、アドレスが現在の反復について明示的に算出されることがわかり得る。次いで、アドレス構築動作によって算出されたアドレスを使用して、`Gettagblock` 動作が実施される。処理回路は、一般に、命令の実行によって動作を始動されたのと厳密に同じやり方で（すなわち図6Aで説明された様式で）アドレスを更新するが、図6Bに示されている実例では、その更新されたアドレスは使用されない。特に、コマンド・シーケンスが再び実施される場合、アドレス構築動作は、提供されたベース・アドレス及びオフセット値に基づいて、使用されるべきアドレスを再計算するために実施される。

【0058】

最後に、読み取られたメタデータ項目を `X t a g s` レジスタ内に記憶された前のメタデータ項目とマージするために、マージ動作が使用される。図 6 B に示されている 3 つの下部のコマンドは、実施されるべきメタデータ・アクセス動作の各インスタンスについて再発行され得る。制約情報は、処理回路に問い合わせることなしに発行され得るコマンド・シーケンスの複数のインスタンスを決定するために使用される。制約情報は、`G e t t a g b l o c k` 動作の現在のインスタンスについてのアドレスを生成するために使用されるオフセット値を決定するためにも使用される。図 6 A の前の説明と同様に、各反復中に `X t m p` レジスタ中のコンテンツを使用して `X t a g s` レジスタ中のどのロケーションが更新されるかを示すために、アドレスのいくつかの低位ビットが使用され得る。

10

【0059】

図 7 A は、メモリ・アドレス空間 300 と、そのアドレス空間内の様々なデータ・グラニュール 305 及び関連するメタデータ項目 310 の存在とを示す。ただし、1 つの例示的な構成では、メタデータ項目自体は直接アドレス指定可能でなく、したがって、半ば隠れている (*semi-hidden*) と見なされ得る。代わりに、図 7 A 中の下側の図に示されているように、アドレス指定可能なアドレス空間は、データ・グラニュールに関して指定される。この実例では、0 のバイト・アドレスは第 1 のデータ・グラニュールがアクセスされることを引き起こし、16 のバイト・アドレスは第 2 のデータ・グラニュールがアクセスされることを引き起こし、以下同様である。しかしながら、メタデータ項目 310 は各データ・グラニュールに関連する。その場合、メタデータ・アクセス動作が使用

20

【0060】

図 7 B は、メタデータ項目の 2 つの特定の例示的な形式を示す。両方の実例では、データ・グラニュールのサイズが 16 バイトであると仮定されるが、他の実装形態では、データ・グラニュール・サイズは異なり得る。能力実例によれば、16 バイトの各データ・グラニュールは、本明細書では能力タグとも呼ばれる、関連する 1 ビットのメタデータを有し、それは、関連するデータ・グラニュールが能力を識別するのか汎用データを識別するにすぎないのかを識別するために使用される。

30

【0061】

図 7 B 中の第 2 の実例は割振りタグ付けを示し、ここで、図示された特定の事例では、4 ビットのメタデータが各 16 バイト・グラニュールに関連する。これらの 4 ビットのメタデータは、本明細書ではガード・タグとも呼ばれる、割振りタグと呼ばれ得る。メモリ中の特定のロケーションに対するアクセス要求が行われたとき、アドレス・タグが、ターゲット・アドレスに関連して指定され得、そのアドレス・タグは、メモリ中の関連のあるデータ・グラニュールに関連する割振りタグと比較され得る。ガード・タグとアドレス・タグとの間の一致が検出されたかどうかに応じて、異なるアクションがとられ得る。

【0062】

前に説明された図 6 B は、`G e t t a g b l o c k` 動作を使用して、複数のメタデータ項目を読み取るために使用され得るデバグ始動コマンド・シーケンスを示した。図 8 は、`S e t t a g b l o c k` 動作を使用して、複数のメタデータ項目に対する書き込み動作を実施するために使用され得る別の例示的なコマンド・シーケンスを示す。図 8 に示されているように、書き込まれる必要があるメタデータ値を用いて `X t a g s` レジスタをセットするために第 1 の動作が実施され、次いで、ポインタ値に基づいてベース・アドレスがセットされる。その後、2 つのコマンドを含むコマンド・シーケンスが提供され、第 1 のコマンドは、使用されるべきアドレスを識別するためのアドレス構築コマンドであり、次のコマンドは、アドレス構築動作の実施によって決定されたアドレスを使用して `S e t t a g b l o c k` 動作が実施されることを引き起こすために使用される `S e t t a g b l o c k` コマンドである。図 6 B と同様に、アドレスは、現在の反復について明示的に算出され

40

50

ることがわかり得る。Settagblock動作は、Xtagsレジスタ中のいくつかのタグ値が、Xaddrレジスタ中のアドレスを使用して決定された関連するメタデータ項目ロケーションに書き込まれることを引き起こす。図8の実例によれば、アドレスは、書き込まれたメタデータ項目の数を考慮に入れるために、Settagblock動作の実施に基づいて処理回路によって更新され得るが、次の反復時にアドレスはベース・アドレスとオフセット値とに基づいて再び直接決定されるので、その更新されたアドレスは無視される。アドレス構築コマンドとSettagblockコマンドとは、実施されるべきメタデータ・アクセス動作の各インスタンスについて再発行される。制約情報は、処理回路に問い合わせることなしに発行され得るコマンド・シーケンスの複数のインスタンスを決定するために使用され、その制約情報は、各反復についてオフセット値を決定するためにも使用される。図6Bの実例と同様に、アドレスの低位ビットは、各反復について参照されるXtagsレジスタ内のロケーションを決定するために処理回路によって使用され得る。この目的で使用される実際のアドレス・ビットは、データ・グラニュール・サイズに依存し得る。

【0063】

図6B及び図8の、前に説明された実例では、メタデータ・アクセス動作の各インスタンスに関連して明示的アドレス生成動作が実施される。しかしながら、代替の例示的な構成では、図9A及び図9Bを参照しながら説明されるように、そのようなアドレス生成動作の使用を省略し、代わりに、メタデータ・アクセス動作自体の実施によって作り出された自動的に調整されるアドレスを利用することが可能であり得る。図9Aに示されているように、当該のメタデータ項目はアドレス空間領域350内に存在する。メタデータ・アクセス動作が処理回路によって実施されるたびに、処理回路は、アクセスされるメタデータ項目の数に基づいてアドレスを更新し、これは、アドレスが、矢印355、360、365によって概略的に示されているように調整されることを引き起こす。この単純な実例では、毎回1つのメタデータ項目がアクセスされると仮定されるが、実際には、メタデータ・アクセス動作が実施されるたびにより多くのメタデータ項目がアクセスされ得ることが諒解され、実際、アクセスされる正確な数がメタデータ・アクセス動作の異なる反復の間で異なり得ることが起こり得る。図6Aを参照しながら前に説明されたように通常の非デバッグ動作モードでは、アドレスに対するそのような調整は、最終的に、破線矢印375によって示されるように、アドレスが範囲350外で調整されることを引き起こす。ソフトウェア・コード手法では、これは、必要とされるメタデータ項目のすべてがアクセスされたとループ・テストがその時点において決定し、したがって、ループの実施が終了するので、十分である。

【0064】

しかしながら、デバッグ制御の下で同様のプロセスを実施しようとするとき、そのねらいは、デバッグ動作の実施を改善するように、メタデータ・アクセス動作の複数のインスタンスが、デバッガが進捗について処理回路に問い合わせる必要なしに実施されることを可能にすることである。ただし、所望のアドレス範囲350外のメタデータ項目がメタデータ・アクセス動作のそれらの複数のインスタンスの実施中にアクセスされる見込みがないことが確実にされる必要がある。1つの例示的な構成では、これは、図9Aに示されているように、ラッピング機能性をエンフォースすることによって達成される。特に、アドレス範囲の最上部に達したとき、アドレスが範囲外のアドレスに調整されるのではなく、アドレスは、矢印370によって概略的に示されるように、アドレス範囲の開始にラッピングするために調整される。その結果、厳密に必要なよりも多くの、メタデータ・アクセス動作のインスタンスが実施される場合でも、当該のメタデータ項目のみがアクセスされることが確実にされ得ることが確実にされる。処理回路がデバッグ・モード（その間にラッピングがエンフォースされる）で動作されているのか非デバッグ・モード（ラッピングがエンフォースされない）で動作されているのかに依存する、機能性のこの差は、図9Bに関して実例として概略的に示されている。図9Bの上部の2つのコマンドと、図6B中の3つのコマンドとの比較からわかるように、アドレス構築コマンドはもはや必

10

20

30

40

50

要とされず、したがって、メタデータ・アクセス動作の各繰返しに必要なコマンドの数が低減される。第1のアドレス増分シーケンスによって示されているように、アドレス範囲350がメモリの1キロバイト・エリアと16バイト・データ・グラニュールとを指定すると仮定される場合、アドレスが、アクセスされるメタデータ項目の数に基づいて毎回調整されることがわかり得る。この実例では、8つのメタデータ項目がメタデータ・アクセス動作の各実施中にアクセスされると仮定される。しかしながら、ラッピング機能性がエンフォースされた（デバッガ・シナリオ）とき、アドレスは、下側のケースのシナリオ（ソフトウェア・シナリオ）に示されているように、 0×1400 に調整されるのではなく、 0×1000 にラッピングする。

【0065】

図1を参照しながら前に説明されたように、ラッピング機能性を選択的に有効及び無効にするようにデバッガの制御下でセットされ得る制御レジスタ55が提供され得る。

【0066】

図10は、デバッガ10からの1つ又は複数のコマンドに応答して動作のシーケンスが処理回路によって生成されることを可能にするために、動作生成回路390が処理回路15'内に提供される、図1に示されているシステムの代替の例示的なシステムを提供する。したがって、この実例では、デバッガは、それ自体で、メタデータ・アクセス動作の複数のインスタンスを実装するために必要とされるコマンドの特定のシーケンスを指定する必要がない。代わりに、デバッガは、たとえば、デバッグ・アクセス・ポートを介して動作生成回路390に単一のコマンドを送り得、動作生成回路は、次いで、実施されるべきメタデータ・アクセス動作の複数のインスタンスを決定するために制約情報50を参照する。そのような実施例では、動作生成回路は、実施されるべき反復の正しい数を決定し、アドレス生成におけるオーバーランを防ぐことができるので、ラッピング制御機能性の必要がないことがある。

【0067】

図10では、動作生成回路390は制約情報50へのアクセスを有するものとして示されているが、これは要件ではなく、動作生成回路が、実施されるべきメタデータ・アクセス動作のいくつかのインスタンスを決定するように構成され得るいくつかのやり方がある。たとえば、動作生成回路390は、（たとえば、メタデータ・アクセス動作の実施によって生成された更新されたアドレスへのアクセスを有することによって）メタデータ・アクセス動作が実施されるたびにアクセスされるメタデータ項目の数に関する情報への動的アクセスを有し、その情報を、メタデータ・アクセス動作の追加の1つ又は複数のインスタンスが実施される必要があるかどうかを決定するために使用するように構成され得る。

【0068】

メタデータ項目は様々な形式をとることができるが、2つの実例が、図11及び図12を参照しながら以下で説明される。

【0069】

図11は、個々のデータ・ブロックが能力（この場合、有界ポインタ及び関連する制限情報）を表すのか通常のデータを表すのかを識別するために、それらのデータ・ブロックに関連してタグ・ビットがどのように使用されるかを概略的に示す。特に、能力アウェア・システム中のメモリ・アドレス空間410は、一般に、指定されたサイズを有する、一連のデータ・ブロック415を記憶し得る。単に例示のために、この実例では各データ・ブロックが128ビットを備えると仮定される。各データ・ブロック415に関連して、一実例では、タグ・ビットと呼ばれる単一のビット・フィールドであるタグ・フィールド420が提供され、これは、関連するデータ・ブロックが能力を表すことを識別するようにセットされ、関連するデータ・ブロックが通常のデータを表し、したがって、能力として扱われ得ないことを示すようにクリアされる。セット状態又はクリア状態に関連する実際の値が実施例に応じて変動することができるが、単に例として、一実施例では、タグ・ビットが1の値を有する場合、それは、関連するデータ・ブロックが能力であることを示し、タグ・ビットが0の値を有する場合、それは、関連するデータ・ブロックが通常のデ

10

20

30

40

50

ータを含んでいることを示すことが諒解されよう。

【0070】

能力が、図11に示されている能力レジスタ400など、(本明細書では能力レジスタとも呼ばれる)有界ポインタ・レジスタにロードされたとき、タグ・ビットは、能力情報とともに移動する。したがって、能力が能力レジスタ400にロードされたとき、ポインタ402と、範囲情報404と、(パーミッション情報とも呼ばれる)制限情報406とが能力レジスタにロードされる。追加として、その能力レジスタに関連して、又はその内部の特定のビット・フィールドとして、タグ・ビット408は、コンテンツが能力を表すことを識別するようにセットされる。同様に、能力がメモリに戻って記憶されたとき、関連のあるタグ・ビット420は、能力が記憶されたデータ・ブロックに関連してセットされる。そのような手法によって、能力と通常のデータとを区別し、したがって、通常のデータが能力として使用され得ないことを確実にすることが可能である。

【0071】

本技法が適用され得るメタデータ項目の別の実例として、図12は、タグガード・メモリ・アクセス(本明細書では割振りタグとも呼ばれるガード・タグ)の概念を概略的に示す。メモリ・システム内のメモリ・ロケーションを参照するために使用される物理アドレス空間は、各々がある数のアドレス指定可能なロケーションを備えるいくつかのブロック530に、論理的に区分され得る。簡潔のために、図12の実例では、各ブロック530は4つのメモリ・ロケーションを備えるが、他のブロック・サイズが同様に使用され得る。各ブロック530は、対応するガード・タグ532に関連する。ある数のブロック530に関連するガード・タグは、一緒に集められ、物理アドレス空間内の異なるアーキテクチャ上アクセス可能なメモリ・ロケーション534内に、又はアーキテクチャ上アクセス可能でない(同じ物理アドレス空間にマッピングされない)メイン・メモリ中に提供された追加の記憶ロケーション内にのいずれかに記憶され得る。別個のアーキテクチャ上アクセス可能でないストレージの使用は、いくつかの場合には、ガード・タグ値をキャッシュするためのデータ・キャッシュ中の空間を使い尽くすことを回避するために選好され得、これは、通常のコードの性能に影響を及ぼし得、コヒーレンシ管理をより複雑にし得る。タグがメイン・メモリからアクセスされなければならない場合よりも速いアクセスのために、アーキテクチャ上アクセス可能でないストレージからタグ値をキャッシュするための追加のタグ・キャッシュが、処理回路15中に提供され得る。

【0072】

どのタグ記憶ロケーション534が各ブロック530に対応するかの特定のマッピングは、処理回路15のロード／記憶ユニットによって制御され得、ハードワイヤードであり得るか又はプログラマブルであり得る。代替的に、メモリ・システム40は、タグがどこに記憶されるかを決定し得、処理回路は、知る又は気にかける必要がない。図12では、各タグ532が物理アドレスのブロックに関連すると同時に、仮想メモリ・アドレス空間中の仮想メモリ・ロケーションに関連するガード・タグ532を提供することも可能であるが、これは、各メモリ・アクセス時にいくつかの追加のアドレス変換を必要とし得る。したがって、ガード・タグ532を物理メモリ・ロケーションに関連付けることによって、これは、性能を改善することができる。概して、特定の実装形態では、ガード・タグ532が物理アドレス空間の対応するブロック530に厳密にどのように関連付けられるかが選ばれる。概して、必要とされるのは、メモリの所与のブロックに関連するガード・タグ532がアクセスされ、比較され得ることだけである。

【0073】

したがって、タグガード・メモリ・アクセスが必要とされるとき、(アクセスされることになるアドレス指定されたロケーション544を識別するターゲット・アドレス542に関連する)アドレス・タグ540が、アドレス指定されたロケーション544を含むメモリ・ロケーションのブロック530に関連するガード・タグ532と比較される。たとえば、図12では、ターゲット・アドレス542は、図12のアドレス空間中で544で示された、メモリ中のあるロケーションB1を指す。したがって、ロケーションB1を含

むロケーションのブロックBに関連するガード・タグBは、ターゲット・アドレス542に関連するアドレス・タグ540と比較される。図12の上部に示されているように、アドレス・タグ540は、ターゲット・アドレス自体の選択されたビットに応じて決定され得る。特に、アドレス・タグは、アドレス指定されたロケーション544として選択されることになる特定のメモリ・ロケーションを示すための、未使用であるターゲット・アドレスの一部分内のビットから決定され得る。たとえば、いくつかの実装形態では、ターゲット・アドレスのビットの上部分は、常に、符号拡張（オール0又はオール1）など、ある固定値を有し得、したがって、任意のタグ値を用いてこれらの未使用ビットを上書きすることによって、アドレス・タグ540を用いてアドレスがタグ付けされ得る。特定のアドレス・タグ値は、たとえばプログラマー又はコンパイラによって選択され得る。アドレス・タグとガード・タグ532とは、比較的小さいビット数、たとえば4ビットであり得、したがって、メモリ内及びターゲット・アドレス内の多くの空間を占有する必要がない。タグ空間の4ビット、すなわち、タグの16個の可能な値を提供することは、しばしば、多くの共通のタイプのメモリ・アクセス・エラーを検出するのに十分であり得る。

【0074】

したがって、タグガード・メモリ・アクセスが実施されたとき、処理回路15のロード／記憶ユニットは、アドレス・タグ540と、アドレス指定されたロケーション544を含むブロック530に関連するガード・タグ532とを比較し、それらが一致するかどうかを決定する（代替的に、タグ比較は、メモリ・システム40内で実施され得る）。ロード／記憶ユニットは、アドレス・タグ540とガード・タグ532とが一致したかどうかを示す一致指示を生成する。たとえば、この一致指示は、アドレス・タグ540とガード・タグ532との間に不一致がある場合に生成される障害信号、又は、一致があったかどうかを示す、状態レジスタに入れられた指示、又は、エラーが検出されたアドレス及び／若しくはエラーをトリガした命令の命令アドレスを示すためにエラー報告に追加されたエントリであり得る。

【0075】

上記で説明された実例から、説明される技法が、デバッガによるデバッグ動作の実施中に複数のメタデータ項目にアクセスすることの効率の著しい改善を可能にすることが諒解されよう。特に、デバッガが、アクセスされる数のメタデータ項目の進捗を決定するために処理回路に問い合わせる必要がある程度は、著しく低減され、いくつかの事例では完全に回避され、それにより、効率を著しく改善し得る。

【0076】

本出願では、「...ように設定された (configured to...)」という言葉は、装置の要素が、定義された動作を行うことが可能である設定を有することを意味するために使用される。このコンテキストにおいて、「設定」は、ハードウェア又はソフトウェアの相互接続の構成又は様式を意味する。たとえば、装置が、定義された動作を提供する専用ハードウェアを有し得るか、又は、プロセッサ若しくは他の処理デバイスが、機能を実施するようにプログラムされ得る。「ように設定された」は、装置要素が、定義された動作を提供するために、何らかのやり方で変更されることを必要とすることを暗示しない。

【0077】

本発明の例示的な実施例が、添付の図面を参照しながら本明細書では詳細に説明されたが、本発明がそれらの正確な実施例に限定されないこと、並びに、様々な変更、追加及び修正が、添付の特許請求の範囲によって定義される本発明の範囲及び趣旨から逸脱することなく、当業者によってその中で実施され得ることを理解されたい。たとえば、従属請求項の特徴の様々な組合せが、本発明の範囲から逸脱することなく、独立請求項の特徴とともに行われ得る。

10

20

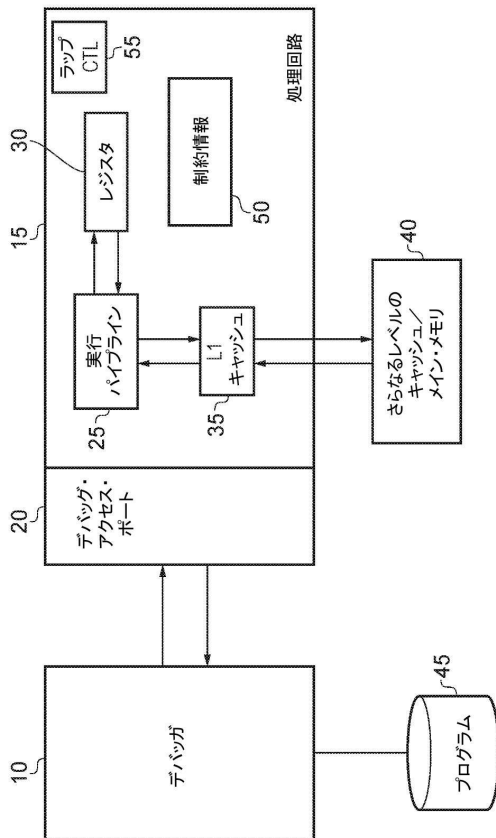
30

40

50

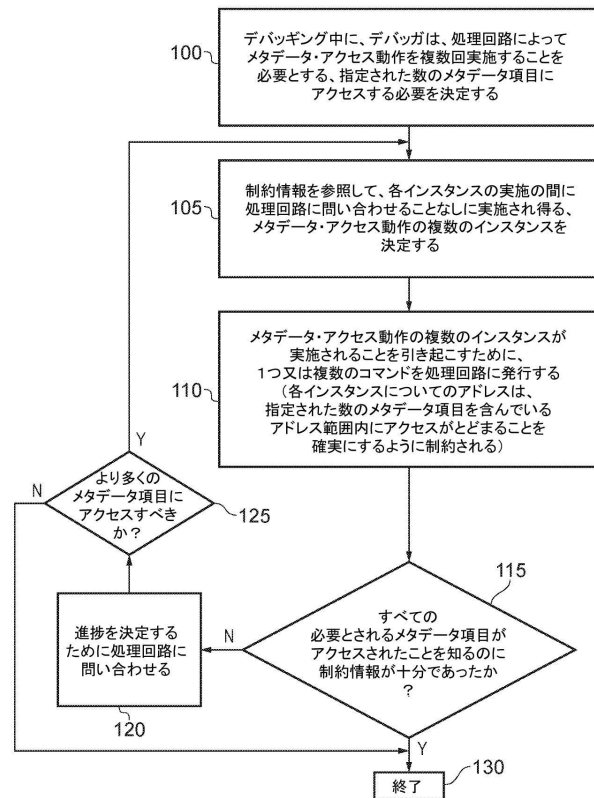
【図面】

【図 1】

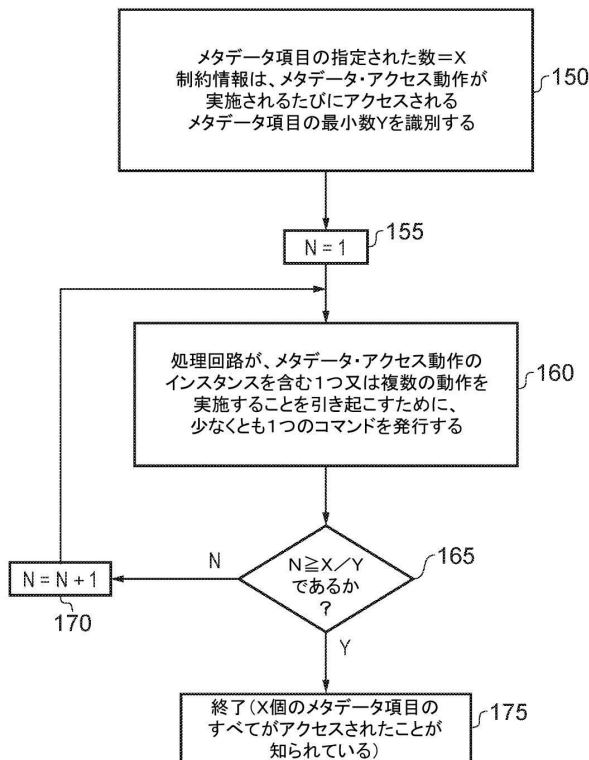


【図 2】

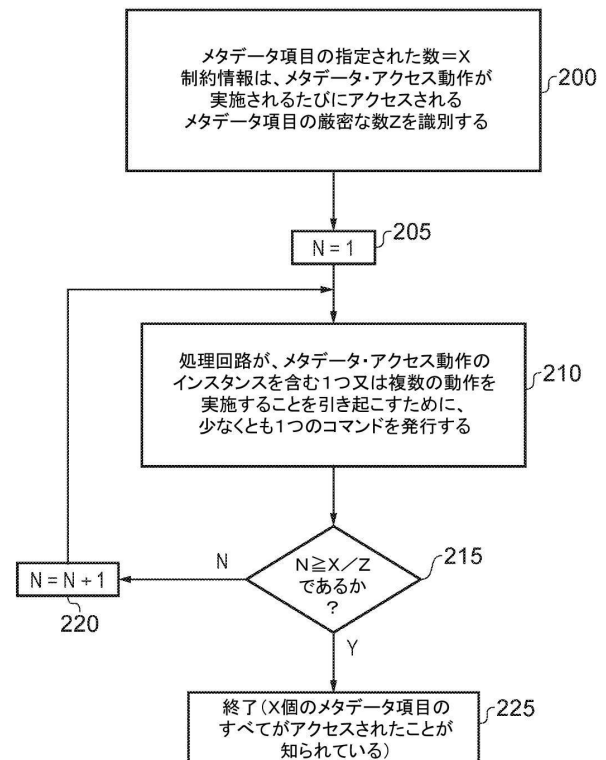
メタデータ・アクセス動作を扱うためのデバッガの動作



【図 3】



【図 4】



10

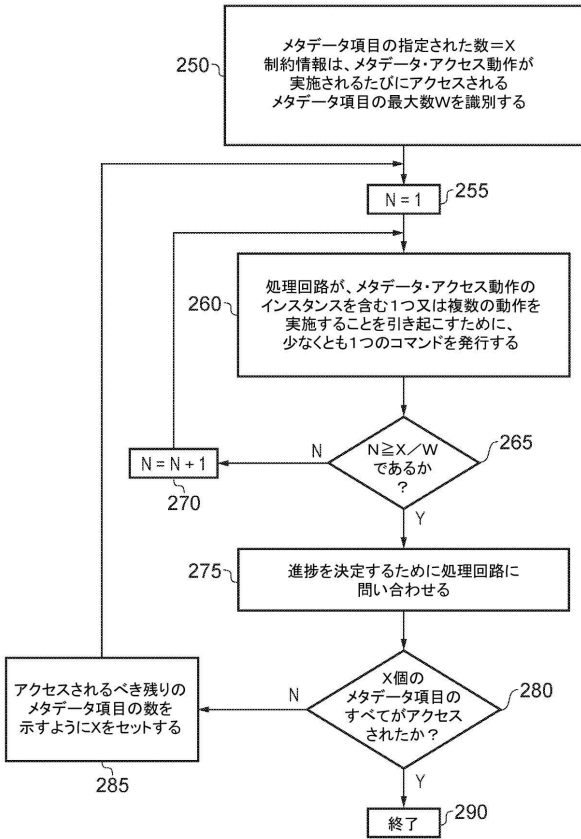
20

30

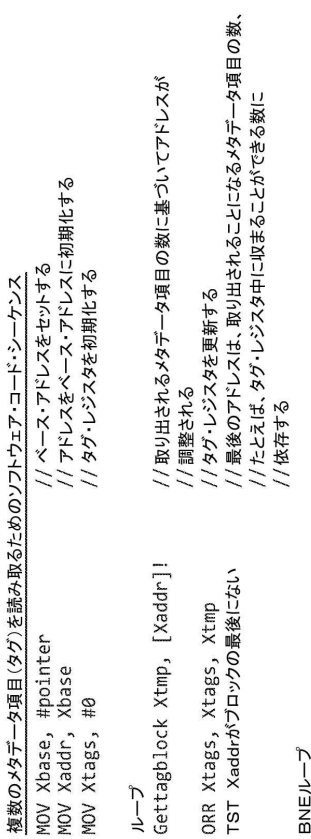
40

50

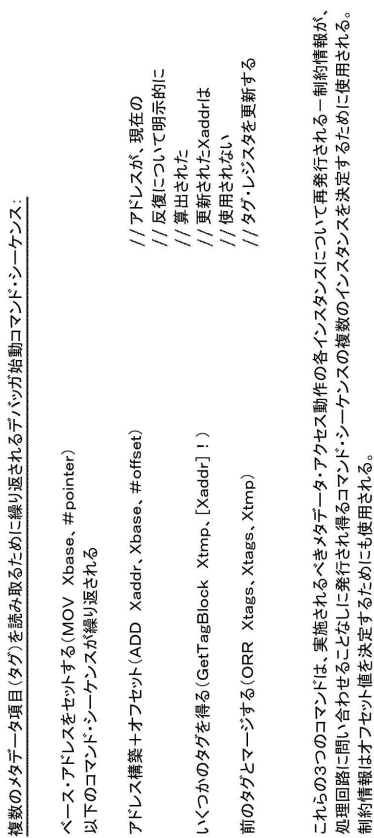
【図 5】



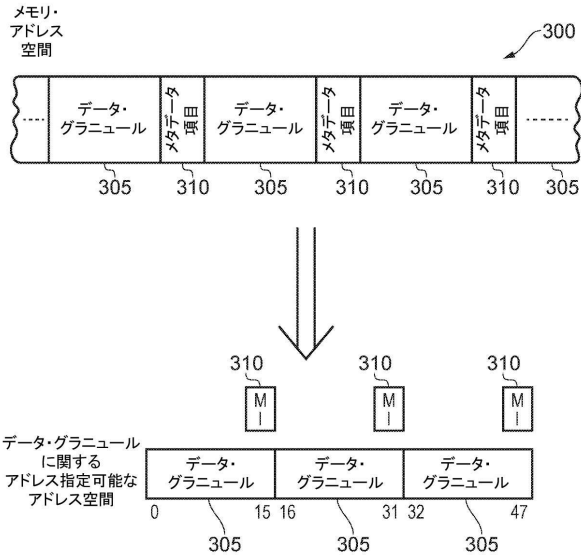
【図 6 A】



【図 6 B】



【図 7 A】



10

20

30

40

50

【図 7 B】

能力(各16バイトのデータについて1ビットのメタデータ)

アドレス	データ
0x1000:	16バイトのデータ、1ビットのメタデータ
0x1010:	16バイトのデータ、1ビットのメタデータ
0x1020:	16バイトのデータ、1ビットのメタデータ

割振りタグ付け(各16バイトのデータについて4ビットのメタデータ)

アドレス	データ
0x1000:	16バイトのデータ、4ビットのメタデータ
0x1010:	16バイトのデータ、4ビットのメタデータ
0x1020:	16バイトのデータ、4ビットのメタデータ

【図 8】

複数のメタデータ項目(タグ)を書き込むためのデバッグ開始コマンド・シーケンス:

いくつかのタグを用いてレジスタをセットする(MRS Xtags, DTR)
ベース・アドレスをセットする(MOV Xbase, #pointer)

以下のコマンド・シーケンスが繰り返される:

アドレス構築+オフセット(ADD Xaddr, Xbase, #offset)

いくつかのタグをセットする(SetTagBlock Xtags, [Xaddr] !)

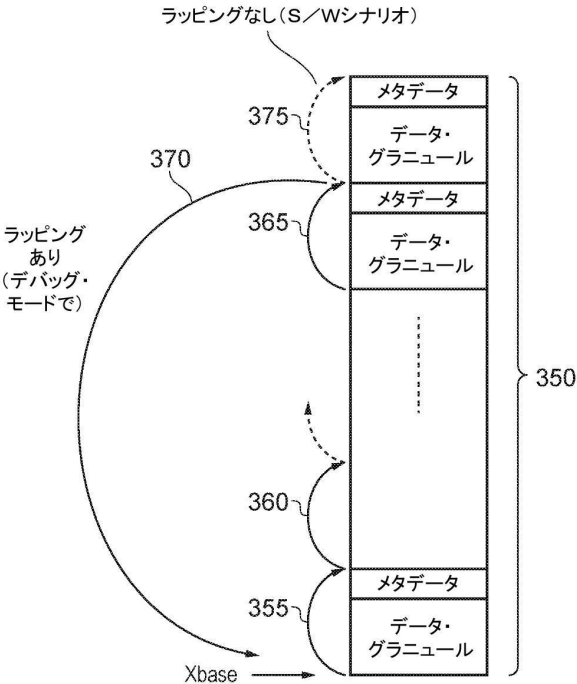
//アドレスが、
//現在の反復について
//明示的に算出された
//更新されたXaddrは
//使用されない

上記の2つのコマンドは、実施されるべきメタデータ・アクセス動作の各インスタンスについて再発行される一制約情報が、
処理回路に問い合わせることなしに発行され得るコマンド・シーケンスの複数のインスタンスを決定するために使用される。
制約情報はオフセット値を決定するためにも使用される。

10

20

【図 9 A】



【図 9 B】

アドレス生成コマンドが必要とされない代替形態

GetTagBlock Xtmp, [Xaddr]!
ORR Xtags, Xtags, Xtmp

これをN回の繰返しについて繰り返す。
しかしながら、インスタンスがこれまでにデータのチャックの上部境界を超えることを試みた場合、
GetTagBlockは、XaddrをXbaseにループバックする。

たとえば、ループバックがある場合(デバッグ・シナリオ)、64個のタグについて1キロバイトのデータ
0x1000
0x1080
0x1100
0x1180
0x1200
0x1280
0x1300
0x1380
0x1000 (0x1000にループバックされる)

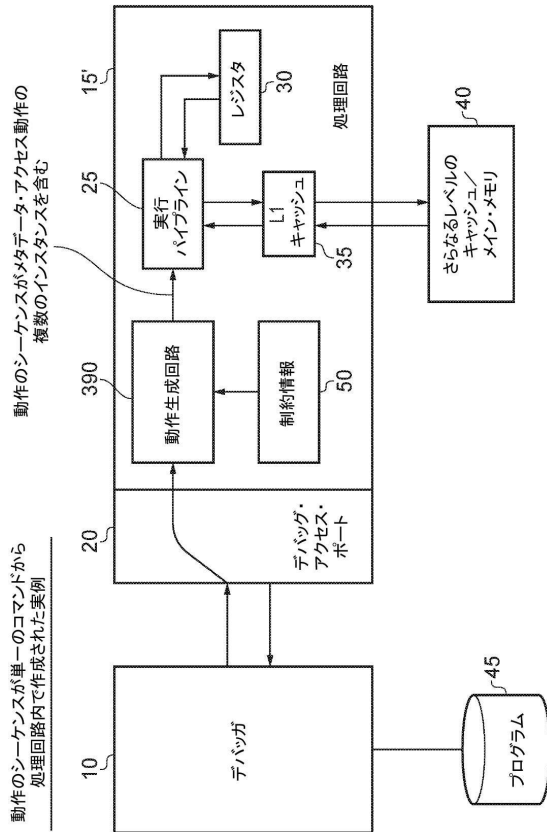
たとえば、ループバックがない場合(ソフトウェア・シナリオ)、64個のタグについて1キロバイトのデータ
0x1000
0x1080
0x1100
0x1180
0x1200
0x1280
0x1300
0x1380
0x1400 (0x1400まで続く)

30

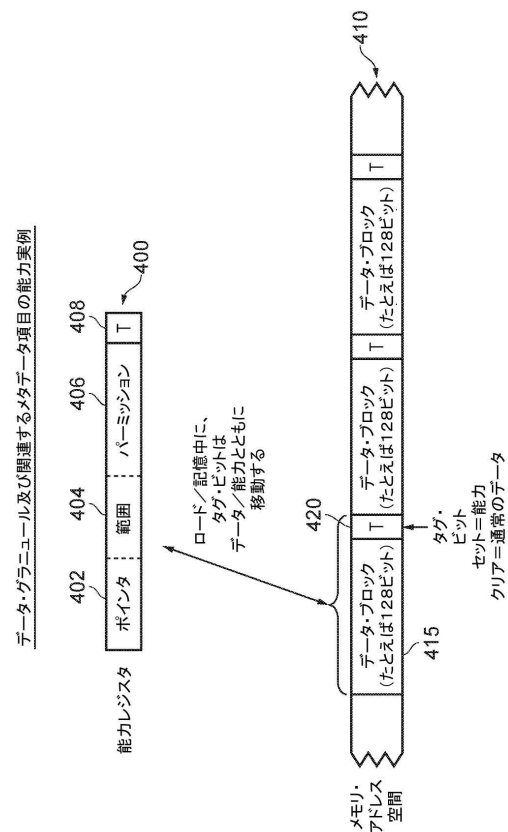
40

50

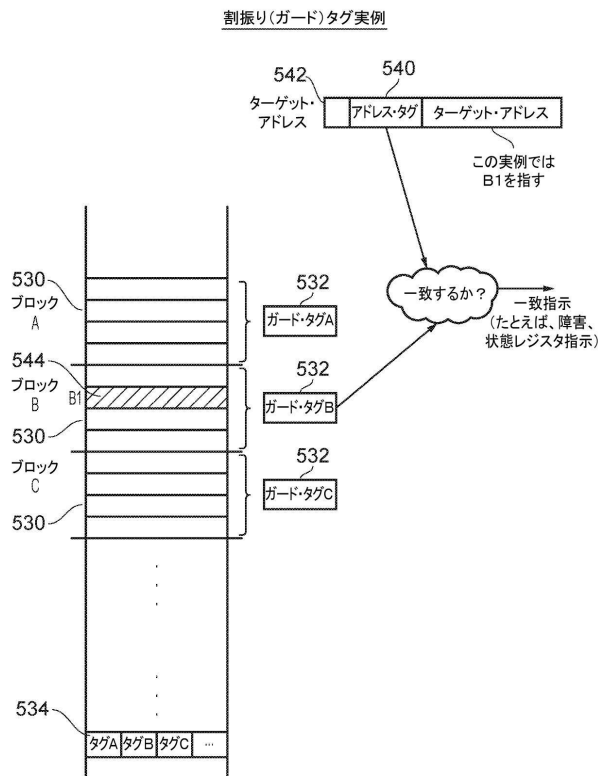
【図 10】



【図 11】



【図 12】



10

20

30

40

50

フロントページの続き

- ジ、チェリー ヒントン、フルボーン ロード 110、エイアールエム リミテッド 気付
- (72)発明者 ホーリー、ジョン マイケル
英国、ケンブリッジシャー、ケンブリッジ、チェリー ヒントン、フルボーン ロード 110、エ
イアールエム リミテッド 気付
- 審査官 岩田 玲彦
- (56)参考文献 米国特許第9053233 (US, B2)
 米国特許出願公開第2009/0254883 (US, A1)
 米国特許出願公開第2011/032439 (US, A1)
 米国特許出願公開第2018/0365130 (US, A1)
- (58)調査した分野 (Int.Cl., DB名)
 G06F 11/36