

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6867082号
(P6867082)

(45) 発行日 令和3年4月28日 (2021.4.28)

(24) 登録日 令和3年4月12日 (2021.4.12)

(51) Int. Cl.	F I
G06F 9/38 (2006.01)	G06F 9/38 370A
G06F 9/46 (2006.01)	G06F 9/38 310F
G06F 9/30 (2018.01)	G06F 9/38 350A
G06F 8/41 (2018.01)	G06F 9/46 430
G06F 17/16 (2006.01)	G06F 9/30 350A

請求項の数 26 (全 68 頁) 最終頁に続く

(21) 出願番号	特願2017-526678 (P2017-526678)	(73) 特許権者	591003943
(86) (22) 出願日	平成27年11月24日 (2015.11.24)		インテル・コーポレーション
(65) 公表番号	特表2017-539008 (P2017-539008A)		アメリカ合衆国 95054 カリフォルニア州・サンタクララ・ミッション カレッジ ブレーバード・2200
(43) 公表日	平成29年12月28日 (2017.12.28)		
(86) 国際出願番号	PCT/US2015/062249	(74) 代理人	110000877
(87) 国際公開番号	W02016/105786		龍華国際特許業務法人
(87) 国際公開日	平成28年6月30日 (2016.6.30)	(72) 発明者	オウルド・アハムド・ヴァル、エルモウスタファ
審査請求日	平成30年11月19日 (2018.11.19)		アメリカ合衆国 95054 カリフォルニア州・サンタクララ・ミッション カレッジ ブレーバード・2200 インテル・コーポレーション内
(31) 優先権主張番号	14/582,717		
(32) 優先日	平成26年12月24日 (2014.12.24)		
(33) 優先権主張国・地域又は機関	米国 (US)		

最終頁に続く

(54) 【発明の名称】 装置、方法、プログラム、およびコンピュータ可読記録媒体

(57) 【特許請求の範囲】

【請求項 1】

命令をデコードするためのハードウェアデコーダであって、前記命令は、オペコードおよびフォールバックアドレスの少なくとも一部を格納するためのオペランドを示す、ハードウェアデコーダと、

前記デコードされた命令を実行するための実行ハードウェアであって、前記デコードされた命令に応じて、投機的メモリアクセスを追跡し且つデータ投機的実行領域（DSX領域）における順序違反を検出するためのDSX追跡ハードウェアがアクティブ化され、且つ、前記フォールバックアドレスが格納されることによって、前記DSX領域における命令実行を開始する、実行ハードウェアと、を備え、

前記DSX領域において、ストアはバッファリングされ、ロードはバッファリングされない

装置。

【請求項 2】

前記フォールバックアドレスの前記少なくとも一部は、前記実行ハードウェアによって、前記デコードされた命令の直後にある命令の命令ポインタの値に対して加算されることにより前記フォールバックアドレスを得るための変位値である、請求項 1 に記載の装置。

【請求項 3】

前記フォールバックアドレスの前記少なくとも一部は、完全アドレスである、請求項 1 に記載の装置。

【請求項 4】

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、即値である、請求項 1 から 3 のいずれか一項に記載の装置。

【請求項 5】

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、レジスタである、請求項 1 から 3 のいずれか一項に記載の装置。

【請求項 6】

前記実行ハードウェアはさらに、RTM (Restricted Transactional Memory) トランザクションが発生していることを判断し且つ前記 RTM トランザクションを処理するためのものである、請求項 1 から 5 のいずれか一項に記載の装置。

10

【請求項 7】

対応する DSX 領域の終了を備えていない DSX 領域の開始の数に対応する値を格納するための DSX ネストカウンタをさらに備える、請求項 1 から 6 のいずれか一項に記載の装置。

【請求項 8】

ハードウェアデコーダを使用して、命令をデコードする段階であって、前記命令はオペコードおよびフォールバックアドレスの少なくとも一部を格納するためのオペランドを示す、デコードする段階と、

前記デコードされた命令を実行する段階であって、前記デコードされた命令に応じて、投機的メモリアクセスを追跡し且つデータ投機的実行領域 (DSX 領域) における順序違反を検出するための DSX 追跡ハードウェアがアクティブ化され且つ前記フォールバックアドレスが格納されることによって、前記 DSX 領域における命令実行を開始する、実行する段階と、を備え、

20

前記 DSX 領域において、ストアはバッファリングされ、ロードはバッファリングされない

方法。

【請求項 9】

前記フォールバックアドレスの前記少なくとも一部は、前記実行する段階によって前記デコードされた命令の直後にある命令の命令ポインタの値に対して加算されることにより前記フォールバックアドレスを得るための変位値である、請求項 8 に記載の方法。

30

【請求項 10】

前記フォールバックアドレスの前記少なくとも一部は、完全アドレスである、請求項 8 に記載の方法。

【請求項 11】

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、即値である、請求項 8 から 10 のいずれか一項に記載の方法。

【請求項 12】

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、レジスタである、請求項 8 から 10 のいずれか一項に記載の方法。

40

【請求項 13】

前記実行する段階は、

RTM (Restricted Transactional Memory) トランザクションが発生していることを判断する段階および前記 RTM トランザクションを処理する段階をさらに含む、請求項 8 から 12 のいずれか一項に記載の方法。

【請求項 14】

対応する DSX 領域の終了を備えていない DSX 領域の開始の数に対応する値を格納する段階をさらに備える、請求項 8 から 13 のいずれか一項に記載の方法。

【請求項 15】

機械に、回路を生成する手順を実行させるためのプログラムであって、前記回路は、

50

命令をデコードするためのハードウェアデコーダであって、前記命令は、オペコードおよびフォールバックアドレスの少なくとも一部を格納するためのオペランドを示す、ハードウェアデコーダと、

前記デコードされた命令を実行するための実行ハードウェアであって、前記デコードされた命令に応じて、投機的メモリアクセスを追跡し且つデータ投機的実行領域（D S X 領域）における順序違反を検出するための D S X 追跡ハードウェアがアクティブ化され且つ前記フォールバックアドレスが格納されることによって、前記 D S X 領域における命令実行を開始する、実行ハードウェアと、を含み、

前記 D S X 領域において、ストアはバッファリングされ、ロードはバッファリングされない

10

プログラム。

【請求項 1 6】

前記フォールバックアドレスの前記少なくとも一部は、前記実行ハードウェアによって、前記デコードされた命令の直後にある命令の命令ポインタの値に対して加算されることにより前記フォールバックアドレスを得るための変位値である、請求項 1 5 に記載のプログラム。

【請求項 1 7】

前記フォールバックアドレスの前記少なくとも一部は、完全アドレスである、請求項 1 5 に記載のプログラム。

【請求項 1 8】

20

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、即値である、請求項 1 5 から 1 7 のいずれか一項に記載のプログラム。

【請求項 1 9】

命令をデコードするためのハードウェアデコーダ手段であって、前記命令は、オペコードおよびフォールバックアドレスの少なくとも一部を格納するためのオペランドを示す、ハードウェアデコーダ手段と、

前記デコードされた命令を実行するための実行手段であって、前記デコードされた命令に応じて、投機的メモリアクセスを追跡し且つデータ投機的実行領域（D S X 領域）における順序違反を検出するための D S X 追跡ハードウェアがアクティブ化され、且つ、前記フォールバックアドレスが格納されることによって、前記 D S X 領域における命令実行を開始する、実行手段と、を備え、

30

前記 D S X 領域において、ストアはバッファリングされ、ロードはバッファリングされない

装置。

【請求項 2 0】

前記フォールバックアドレスの前記少なくとも一部は、前記実行手段によって、前記デコードされた命令の直後にある命令の命令ポインタの値に対して加算されることにより前記フォールバックアドレスを得るための変位値である、請求項 1 9 に記載の装置。

【請求項 2 1】

前記フォールバックアドレスの前記少なくとも一部は、完全アドレスである、請求項 1 9 に記載の装置。

40

【請求項 2 2】

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、即値である、請求項 1 9 から 2 1 のいずれか一項に記載の装置。

【請求項 2 3】

前記フォールバックアドレスの前記少なくとも一部を格納するための前記オペランドは、レジスタである、請求項 1 9 から 2 1 のいずれか一項に記載の装置。

【請求項 2 4】

前記実行手段はさらに、R T M（R e s t r i c t e d T r a n s a c t i o n a l M e m o r y）トランザクションが発生していることを判断し且つ前記 R T M トランザ

50

クッションを処理するためのものである、請求項 19 から 23 のいずれか一項に記載の装置。

【請求項 25】

対応する DSX 領域の終了を備えていない DSX 領域の開始の数に対応する値を格納するための DSX ネストカウンタ手段をさらに備える、請求項 19 から 24 のいずれか一項に記載の装置。

【請求項 26】

請求項 15 から 18 のいずれか一項に記載のプログラムを記録するコンピュータ可読記録媒体。

【発明の詳細な説明】

10

【技術分野】

【0001】

本発明の技術分野は概して、コンピュータプロセッサアーキテクチャに関し、より具体的には投機的実行に関する。

【背景技術】

【0002】

イタレーション間の依存関係を含む可能性のあるループのベクトル化は、難しいことで有名である。以下に、このタイプの例示的なループを示す。

【数 1】

20

```
for (i = 0; i < N; i++) {
    A[i] = B[C[i]];
}
```

【0003】

以下に、このループの単純な（および正しくない）ベクトル化を示す。

【数 2】

30

```
for (i = 0; i < N; i += SIMD_WIDTH) {
    zmm0 = vmovdqu32 &C[i]
    k1 = kxnor k1, k1
    zmm1 = vgatherdd B, zmm0, k1
    vmovdqu &A[i], zmm1
}
```

40

【0004】

しかしながら、ループのベクトル化されたバージョンを生成するコンパイラが、A、B および C のアドレスまたは配置に関する事前知識を有していない場合、上記のベクトル化は安全ではない。

【図面の簡単な説明】

【0005】

本発明は、添付図面中に、限定ではなく例示として示されており、同様の参照符号は類似の要素を示す。

50

【0006】

【図1】データ投機拡張（DSX）をハードウェアで実行可能なプロセッサコアの一実施形態の例示的なブロック図である。

【0007】

【図2】一実施形態による、投機的命令実行の一例を示す。

【0008】

【図3】DSX追跡ハードウェアの具体的な実施形態を示す。

【0009】

【図4】DSX追跡ハードウェアによって実行されるDSX投機ミス検出の例示的な方法を示す。

10

【0010】

【図5A】DSX追跡ハードウェアによって実行されるDSX投機ミス検出の例示的な方法を示す。

【図5B】DSX追跡ハードウェアによって実行されるDSX投機ミス検出の例示的な方法を示す。

【0011】

【図6】DSXを開始する命令の実行に係る一実施形態を示す。

【0012】

【図7】YBEGIN命令フォーマットのいくつかの例示的な実施形態を示す。

【0013】

【図8】YBEGIN命令等の命令の実行に係る詳細な実施形態を示す。

20

【0014】

【図9】YBEGIN命令等の命令の実行を示す擬似コードの例を示す。

【0015】

【図10】DSXを開始する命令の実行に係る一実施形態を示す。

【0016】

【図11】YBEGIN WITH STRIDE命令フォーマットのいくつかの例示的な実施形態を示す。

【0017】

【図12】YBEGIN WITH STRIDE命令等の命令の実行に係る詳細な実施形態を示す。

30

【0018】

【図13】DSXを終了させずに、DSXを続行する命令の実行に係る一実施形態を示す。

【0019】

【図14】YCONTINUE命令フォーマットのいくつかの例示的な実施形態を示す。

【0020】

【図15】YCONTINUE命令等の命令の実行に係る詳細な実施形態を示す。

【0021】

【図16】YCONTINUE命令等の命令の実行を示す擬似コードの例を示す。

40

【0022】

【図17】DSXをアボートする命令の実行に係る一実施形態を示す。

【0023】

【図18】YABORT命令フォーマットのいくつかの例示的な実施形態を示す。

【0024】

【図19】YABORT命令等の命令の実行に係る詳細な実施形態を示す。

【0025】

【図20】YABORT命令等の命令の実行を示す擬似コードの例を示す。

【0026】

【図21】DSXの状態をテストする命令の実行に係る一実施形態を示す。

50

【0027】

【図22】 Y T E S T 命令フォーマットのいくつかの例示的な実施形態を示す。

【0028】

【図23】 Y T E S T 命令等の命令の実行を示す擬似コードの例を示す。

【0029】

【図24】 D S X を終了させる命令の実行に係る一実施形態を示す。

【0030】

【図25】 Y E N D 命令フォーマットのいくつかの例示的な実施形態を示す。

【0031】

【図26】 Y E N D 命令等の命令の実行に係る詳細な実施形態を示す。

10

【0032】

【図27】 Y E N D 命令等の命令の実行を示す擬似コードの例を示す。

【0033】

【図28A】 本発明の実施形態による、汎用ベクトル向け命令フォーマットおよびその命令テンプレートを示すブロック図である。

【図28B】 本発明の実施形態による、汎用ベクトル向け命令フォーマットおよびその命令テンプレートを示すブロック図である。

【0034】

【図29A】 特定ベクトル向け命令フォーマット2900を示す。特定ベクトル向け命令フォーマット2900は、フィールドの場所、サイズ、解釈および順序に加え、これらのフィールドの一部の値を指定するという意味において特定のである。

20

【図29B】 特定ベクトル向け命令フォーマット2900を示す。特定ベクトル向け命令フォーマット2900は、フィールドの場所、サイズ、解釈および順序に加え、これらのフィールドの一部の値を指定するという意味において特定のである。

【図29C】 特定ベクトル向け命令フォーマット2900を示す。特定ベクトル向け命令フォーマット2900は、フィールドの場所、サイズ、解釈および順序に加え、これらのフィールドの一部の値を指定するという意味において特定のである。

【図29D】 特定ベクトル向け命令フォーマット2900を示す。特定ベクトル向け命令フォーマット2900は、フィールドの場所、サイズ、解釈および順序に加え、これらのフィールドの一部の値を指定するという意味において特定のである。

30

【0035】

【図30】 本発明の一実施形態によるレジスタアーキテクチャのブロック図である。

【0036】

【図31A】 本発明の実施形態による、例示的なインオーダーパイプラインおよび例示的なレジスタリネーミング、アウトオブオーダー発行／実行パイプラインの両方を示すブロック図である。

【0037】

【図31B】 本発明の実施形態による、プロセッサに含まれる、インオーダーアーキテクチャコアに係る例示的な実施形態および例示的なレジスタリネーミング、アウトオブオーダー発行／実行アーキテクチャコアの両方を示すブロック図である。

40

【0038】

【図32A】 より具体的な例示的インオーダーコアアーキテクチャのブロック図を示しており、コアは、チップ内のいくつかの論理ブロック（同一タイプおよび／または異なるタイプの他のコアを含む）のうちの1つである。

【図32B】 より具体的な例示的インオーダーコアアーキテクチャのブロック図を示しており、コアは、チップ内のいくつかの論理ブロック（同一タイプおよび／または異なるタイプの他のコアを含む）のうちの1つである。

【0039】

【図33】 本発明の実施形態による、プロセッサのブロック図であり、当該プロセッサは、2以上のコアを有してよく、統合メモリコントローラを有してよく、統合グラフィック

50

を有してよい。

【0040】

【図34】本発明の一実施形態によるシステムのブロック図を示す。

【0041】

【図35】本発明の一実施形態による第1のより具体的な例示的システムのブロック図を示す。

【0042】

【図36】本発明の一実施形態による第2のより具体的な例示的システムのブロック図を示す。

【0043】

【図37】本発明の一実施形態によるS o Cのブロック図を示す。

【0044】

【図38】本発明の実施形態による、ソース命令セット内のバイナリ命令をターゲット命令セット内のバイナリ命令に変換するためのソフトウェア命令コンバータの使用を対比するブロック図である。

【発明を実施するための形態】

【0045】

以下の詳細な説明において、多数の具体的な詳細が記載される。しかしながら、本発明の実施形態は、これらの具体的な詳細を省いても実施可能であることを理解されたい。他の例においては、本説明の理解を曖昧にしないように、周知の回路、構造および技術は詳細に示されていない。

【0046】

本明細書における「一実施形態」、「実施形態」、「例示的な実施形態」等の言及は、記載される実施形態は特定の特徴、構造または特性を含み得るが、すべての実施形態が必ずしも、当該特定の特徴、構造または特性を含まなくてもよいことを示している。さらに、このような文言が必ずしも同一の実施形態を指しているわけではない。さらに、特定の特徴、構造または特性がある実施形態に関し記載されている場合、明示の記載のあるなしに関わらず、このような特徴、構造または特性を他の実施形態に関し作用させることは当業者の知識の範囲内に属するものである。

【0047】

この詳細な説明を通して、データ投機拡張(D S X)と称される投機的実行に関する技術について記載する。この詳細な説明には、D S XハードウェアおよびD S Xをサポートする新しい命令が含まれている。

【0048】

D S Xは本質的には、制限付きトランザクション型メモリ(R T M)の実装に類似しているが、より簡易的なものである。例えば、D S X領域は、暗示フェンスを必要としない。むしろ、通常のロード／ストア順序ルールが維持される。さらに、D S X領域は、ロードについてアトミックな動作を強制するプロセッサの構成を設定しないのに対し、R T Mでは、トランザクションのロードおよびストアはアトミックに扱われる(トランザクションの完了時、コミットされる)。また、ロードは、R T Mにおける場合のように、バッファリングされない。しかしながら、投機実行が不要になると、直ちにストアはバッファリングおよびコミットされる。実施形態に応じて、これらのストアは、専用の投機的実行ストレージ内または共有レジスタ若しくはメモリ位置にバッファリングされてよい。いくつかの実施形態において、投機的ベクトル化は単一スレッドのみで発生し、それは、他のスレッドからの干渉に対する保護の必要がないことを意味する。

【0049】

上記のベクトル化されたループでは、安全性のために動的チェックが必要となるだろう。例えば、特定のベクトルイタレーションにおけるAへの書き込みが、スカラーループ内でのイタレーションで読み取られるBまたはCの要素と重複しないことを保証することが挙げられる。以下の実施形態は、投機実行の使用を通し、ベクトル化事例の処理について

10

20

30

40

50

詳述する。投機的バージョンは、各ループのイタレーションは投機的に実行されるべきであること（例えば、後述の命令を使用して）、およびハードウェアはアドレスチェックの実行に寄与すべきであるということを示す。アドレスチェック（非常に高価なハードウェアを必要とする）を専門に担うハードウェアに依存する代わりに、説明するアプローチはソフトウェアを使用して、ハードウェアを支援する情報を提供し、実行時間に影響を与えずに、またはプログラマ若しくはコンパイラに過度の負荷をかけずに、はるかに安価なハードウェアソリューションを可能にする。

【0050】

残念なことに、ベクトル化には、順序違反が存在する可能性がある。上記のスカルループの例を再度見てみることにする。

10

【数3】

```
for (i = 0; i < N; i++) {
    A[i] = B[C[i]];
}
```

【0051】

このループの最初の4回のイタレーション中、以下のメモリ操作が以下の順序で発生する。

20

```
Read C[0]
Read B[C[0]]
Write A[0]
Read C[1]
Read B[C[1]]
Write A[1]
Read C[2]
Read B[C[2]]
Write A[2]
Read C[3]
Read B[C[3]]
Write A[3]
```

30

【0052】

同一アレイへの複数のアクセス間の距離（命令個数における）は3であり、これはまた、ひとたびループがベクトル化（SIMDに生成）された場合のループ内の投機的メモリ命令の数でもある。この距離は「ストライド」と呼ばれる。これはまた、ひとたびループがベクトル化された場合のループ内のメモリ命令の数でもあり、当該メモリ命令は、自身に実行されるアドレスチェックを有することになる。いくつかの実施形態において、このストライドは、ループの開始時に、特別な命令を介して、アドレス追跡ハードウェアに伝達される（これについては後述する）。いくつかの実施形態において、この命令はまた、アドレス追跡ハードウェアをクリアする。

40

【0053】

本明細書には、ベクトル化されたループ実行等の場合に、DSXで使用される新しい命令（DSXメモリ命令）について記載する。各DSXメモリ命令（ロード、ストア、ギャザーおよびスキャッター等）は、DSX中に使用されるオペランドを含み、オペランドは、DSX実行内の位置（例えば、実行されるループ内の位置）を示す。いくつかの実施形態において、オペランドは、即値にエンコードされた順序の数値を持つ即値（例えば、8ビット即値）である。他の実施形態において、オペランドは、エンコードされた順序の数値を格納するレジスタまたはメモリ位置である。

50

【0054】

また、いくつかの実施形態において、これらの命令は、通常の命令とは異なるオペコードを有する。これらの命令は、スカラまたはスーパースカラ（例えば、SIMDまたはMIMD）であってよい。これらの命令のいくつかの例をいかに示す。ここでは、オペコードのニーモニックが、それが投機的バージョンであることを示す「S」（以下で下線付き）を含み、imm8は実行の位置（例えば、実行されるループ内の位置）を示すために使用される即値オペランドである。

【数4】

VMOVSDQA32 zmm1 {k1}{z}, mV, imm8 // 投機的SIMDロード

10

VMOVS xmm1, m32, imm8 // 投機的スカラロード

VSCATTERSDPS vm32z {k1}, zmm1, imm8 // 投機的スキッター

【0055】

もちろん、他の命令は、説明されたオペランド並びに論理（AND、OR、XOR等）およびデータ操作（加算、減算等）命令等のオペコードニーモニック（および下線のオペコード）の変形版を利用してもよい。

【0056】

20

上記スカラの例のベクトル化されたバージョン（4つのパックされたデータ要素のSIMD幅を想定）では、メモリ操作の順序は以下の通りである。

Read C[0], C[1], C[2], C[3]

Read B[C[0]], B[C[1]], B[C[2]], B[C[3]]

Write A[0], A[1], A[2], A[3]

【0057】

この順序は、例えば、B[C[1]]がA[0]と重複する場合、誤った実行をもたらす可能性がある。元のスカラ順序では、B[C[1]]の読み取りは、A[0]への書き込みの後に生じるが、ベクトル化された実行では、B[C[1]]の読み取りはA[0]への書き込みの前に生じる。

30

【0058】

誤った実行をもたらす可能性のあるループ内の演算に対し、投機的メモリ命令を使用すると、この問題の対処に役立つ。後述の通り、各投機的メモリ命令が、DSX追跡ハードウェア（後述）に対し、ループ本体内部におけるその位置を通知する。

【数5】

for (i = 0; i < N; i += SIMD_WIDTH) {

zmm0 = vmovsdqu32 &C[i], 0 // アドレストラッカーに、これは命令0であると通知

k1 = kxnor k1, k1

40

zmm1 = vgatherddd B, zmm0, k1, 1 // アドレストラッカーに、これは命令1であると通知

vmovsdqu &A[i], zmm1, 2 // アドレストラッカーに、これは命令2であると通知

}

【0059】

各投機的メモリ操作によって提供されるループ位置情報がストライドと組み合わせられ、スカラメモリ操作を再構築可能である。投機的メモリ命令が実行されると、各要素につい

50

て識別子（ID）がDSXハードウェアトラッカーによって計算される（ID＝シーケンス番号＋ストライド×SIMD演算内の要素数）。ハードウェアトラッカーは、シーケンス番号、計算されたIDおよび各パックされたデータ要素のアドレスおよびサイズを使用して、順序違反（すなわち、データ要素が別のデータ要素と重複しているか、および間違っただ順序で読み取りまたは書き込みされたか）が存在したかどうかを判断する。

【0060】

各ベクトルメモリ命令を有する個々のメモリ操作をアンロールし、各アンロールのストライドを蓄積し、結果の数字を「ID」として割り当てると、以下のようになる。

```
Read C[0] // ID = 0
Read C[1] // ID = 3
Read C[2] // ID = 6
Read C[3] // ID = 9
Read B[C[0]] // ID = 1
Read B[C[1]] // ID = 4
Read B[C[2]] // ID = 7
Read B[C[3]] // ID = 10
Write A[0] // ID = 2
Write A[1] // ID = 5
Write A[2] // ID = 8
Write A[3] // ID = 11
```

10

20

【0061】

上記の個々のメモリ操作をIDでソートすると、元のスカラメモリの順序が再構築される。

【0062】

図1は、データ投機拡張（DSX）をハードウェアで実行可能なプロセッサコアの一実施形態の例示的なブロック図である。

【0063】

プロセッサコア106は、コア106が実行する命令をフェッチするためのフェッチユニット102を含んでよい。例えば、命令は、L1キャッシュまたはメモリからフェッチされてよい。コア106はまた、後述のものを含むフェッチされた命令をデコードするためのデコードユニット104を含んでもよい。例えば、デコードユニット104は、フェッチされた命令を複数のマイクロオペレーション（マイクロop）へとデコードしてよい。

30

【0064】

また、コア106は、スケジューリングユニット107を含んでよい。スケジューリングユニット107は、命令がディスパッチの準備が整うまで、例えば、デコードされた命令のオペランドのすべてのソース値が利用可能になるまで、デコードされた命令（例えば、デコードユニット104から受信された）の格納に関連する様々な操作を実行してよい。一実施形態において、スケジューリングユニット107は、デコードされた命令をスケジューリングし、および／または実行のために1または複数の実行ユニット108に発行（またはディスパッチ）してよい。実行ユニット108は、メモリ実行ユニット、整数実行ユニット、浮動小数点実行ユニットまたは他の実行ユニットを含んでよい。リタイアメントユニット110は、命令がコミットされた後、実行された命令をリタイアしてよい。ある実施形態において、実行された命令のリタイアメントによって、命令の実行からコミットされるプロセッサの状態、命令によって使用された物理レジスタの割り当て解除等がもたらされてよい。

40

【0065】

メモリ順序バッファ（MOB）118は、ロードバッファ、ストアバッファ、およびメインメモリに読み込みまたはライトバックされていない保留中のメモリ操作を格納するためのロジックを含んでよい。いくつかの実施形態において、MOB 118またはMOB

50

118に類似する回路が、DSX領域の投機的ストア（書き込み）を格納する。様々な実施形態において、コアは、例えば、1または複数のキャッシュライン124（例えば、キャッシュライン0からW）を含んでよく且つキャッシュ回路139によって管理されるキャッシュ116等のプライベートキャッシュのようなローカルキャッシュを含んでよい。一実施形態において、キャッシュ116の各ラインは、コア106で実行される各スレッドについて、DSX読み取りビット126および／またはDSX書き込みビット128を含んでよい。ビット126およびビット128は、DSXメモリアクセスリクエストによる対応するキャッシュラインへの（ロードおよび／またはストア）アクセスを示すために、設定またはクリアされてよい。図1の実施形態では、各キャッシュライン124は、それぞれのビット126および128を有するように図示されているが、他の構成も可能であることに留意されたい。例えば、DSX読み取りビット126（またはDSX書き込みビット128）は、キャッシュブロックまたはキャッシュ116の他の部分等、キャッシュ116の選択された部分に対応してよい。また、ビット126および／または128は、キャッシュ116以外の場所に格納されてもよい。

10

【0066】

DSX操作の実行に役立つべく、コア106は、合致するDSX終了がない状態に遭遇したDSX開始の数に対応する値を格納するためのDSXネストカウンタ130を含んでよい。カウンタ130は、ハードウェアレジスタ等の任意のタイプのストレージデバイスまたはメモリ（例えば、システムメモリまたはキャッシュ116）内に格納された変数として実装されてよい。コア106はまた、カウンタ130に格納された値を更新するためのDSXネストカウンタ回路132を含んでもよい。コア106は、コア106の様々なコンポーネントの状態をチェックポイント（または格納）するためのDSXチェックポイント回路134および例えば、特定のDSXのアポート時に、フォールバックアドレスを使用して、コア106の様々なコンポーネントの状態を復元するためのDSX復元回路136を含んでよい。フォールバックアドレスは、DSX復元回路136に格納される、またはレジスタ140等の別の場所に格納される。また、コア106は、DSX状態および制御レジスタ（DSXSR）等、様々なDSXメモリアクセスリクエストに対応する1または複数の追加のレジスタ140を含んでよい。DSXSRは、DSXがアクティブかどうかの指標、DSX命令ポインタ（DSXXIP）（例えば、DSX命令ポインタは、対応するDSXの開始（または直前）位置における命令への命令ポインタであってよい）、および／またはDSXスタックポインタ（DSXSP）（例えば、DSXスタックポインタは、コア106の1または複数のコンポーネントの様々な状態を格納するスタックの先頭へのスタックポインタであってよい）を格納する。これらのレジスタはまた、MSR150であってよい。

20

30

【0067】

DSXアドレス追跡ハードウェア152（場合によっては、単にDSX追跡ハードウェアと呼ばれる）は、投機的メモリアクセスを追跡し、DSXの順序違反を検出する。特に、この追跡ハードウェア152は、元のスカラメモリ順序を再構築するための情報を取得した後、元のスカラメモリ順序を強制するアドレストラッカーを含む。通常、入力は、ループ本体内の追跡が必要な投機的メモリ命令の数であり、それらの命令の各々についての情報の一部は（1）シーケンス番号、（2）命令アクセスのアドレス、および（3）命令がメモリへの読み取りまたは書き込みのどちらを生じさせるか、といったものである。2つの投機的メモリ命令がメモリの重複部分にアクセスする場合、ハードウェアトラッカー152はこの情報を使用して、メモリ操作の元のスカラ順序が変更されたかどうかを判断する。変更されており、且ついずれかの操作が書き込みである場合、ハードウェアは投機ミスをトリガする。図1は、独自のDSX追跡ハードウェア152を図示するものの、いくつかの実施形態においては、このハードウェアは他のコアコンポーネントの一部である。

40

【0068】

図2は、一実施形態による、投機的命令実行の一例を示す。201において、投機的命

50

令がフェッチされる。例えば、上記のような投機的メモリ命令がフェッチされる。いくつかの実施形態において、この命令は、その投機的性質を示すオペコードおよびDSX内での順序付けを示すオペランドを含む。順序付けオペランドは、即値またはレジスタ／メモリ位置であってよい。

【0069】

203において、フェッチされた投機的命令がデコードされる。

【0070】

205において、デコードされた投機的命令がDSXの一部であるかどうかの判断がなされる。例えば、一実施形態により、DSXが上記のDSX状態および制御レジスタ(DSXSR)に示されているか?といったものである。DSXがアクティブでない場合、207において、命令は、処理なし(nop)になるか、または通常の命令である非投機的命令として実行される。

【0071】

DSXがアクティブである場合、209において、投機的命令が投機的に実行され(例えば、コミットされない)、且つDSX追跡ハードウェアが更新される。

【0072】

図3は、DSXアドレス追跡ハードウェアの詳細な実施形態を示す。このハードウェアは、投機的メモリインスタンスを追跡する。通常、DSX追跡ハードウェアによって解析される要素(例えば、SIMD要素)は、サイズが「B」バイト以下のチャンクと呼ばれる部分に細分化される。

【0073】

シフト回路301は、チャンクのアドレス(開始アドレス等)をシフトする。多くの実施形態において、シフト回路301は、右シフトを実行する。通常、右シフトは、log₂B分である。シフトされたアドレスは、ハッシュ関数ユニット回路303によって実行されるハッシュ関数が適用される。

【0074】

ハッシュ関数の出力は、ハッシュテーブル305へのインデックスである。図示の通り、ハッシュテーブル305は、複数のバケット307を含む。いくつかの実施形態において、ハッシュテーブル305はブルームフィルタである。ハッシュテーブル305は、投機ミスの検出および投機的にアクセスされるデータのアドレス、アクセスタイプ、シーケンス番号およびID番号の記録に使用される。ハッシュテーブル305は、N個の「セット」を含んでおり、各セットはM個のエントリ309を含む。各エントリ309は、前に実行された投機的メモリ命令の要素の有効なビット、シーケンス番号、ID番号およびアクセスタイプを保持する。いくつかの実施形態において、各エントリ309は、対応するアドレス(図中、破線ボックスで図示)も含む。DSXが命令(例えば、後述のYBEGINおよびバリエーション)を開始すると、すべての有効なビットはクリアされ、「投機実行アクティブ」フラグが設定され、命令がDSXを終了する際、投機アクティブフラグがクリアされる。

【0075】

競合チェック回路311は、テスト対象要素315(または要素のチャンク)に対する競合チェックをエントリ309ごとに行う。いくつかの実施形態において、エントリ309が有効であり且つ次のうちの少なくとも1つの場合に、競合が存在する。すなわち、i)エントリ309内のアクセスタイプが書き込みである、またはii)テスト中のアクセスタイプが書き込みである、場合に、i)エントリ309内のシーケンス番号がテスト対象要素315のシーケンス番号より小さく且つエントリ309内のID番号がテスト対象要素315のID番号より大きい、またはii)エントリ309内のシーケンス番号がテスト対象要素315のシーケンス番号より大きく且つエントリ309内のID番号がテスト対象要素315のID番号より小さい、のうちのいずれかが組み合わされたとき、である。

【0076】

換言すると、以下の場合に競合が存在する。(エントリが有効である) AND ((エントリ内のアクセスタイプ==書き込み) OR (テスト対象のアクセスタイプ==書き込み) AND ((エントリ内のシーケンス番号<テスト対象のシーケンス番号) AND (エントリ内のID番号>テスト対象のID番号)) OR ((エントリ内のシーケンス番号>テスト対象のシーケンス番号) AND (エントリ内のID番号<テスト対象のID番号)))).

【0077】

多くの実施形態においては、アドレス重複のテストは存在しないことに留意されたい。この重複は、ハッシュテーブル内のエントリにヒットすることから暗示される。ヒットは、アドレス重複が存在しない場合に発生する可能性があり、これはハッシュ関数のエイリアシングから、および／または、チェックの粒度が粗すぎる(すなわち、Bが大きすぎる)ことからのエイリアシングに起因する。しかしながら、アドレス重複が存在する場合は、ヒットが存在するであろう。よって、正確性は保証されるが、誤検知が存在する可能性がある(すなわち、ハードウェアはアドレス重複がない場合に投機ミスを検出する可能性がある)。一実施形態において、チャンクアドレスは各エントリ309に格納され、投機ミスをテストするための追加の条件が適用される(すなわち、これは、エントリ309内のアドレスがテスト対象要素315のアドレスと等しいという上記条件と論理AND演算される)。

【0078】

ORゲート313(または同等物)は、競合チェックの結果を論理OR演算する。OR演算の結果が1である場合、投機ミスが発生した可能性があり、ORゲート313はその出力と共にそれを示す。

【0079】

この実施形態の合計ストレージは、 $M \times N$ エントリである。このことは、それが、最大 $M \times N$ の投機的にアクセスされたデータ要素を追跡し得ることを意味する。しかしながら、実際、ループは、N個のセットのうち、一部のセットへのアクセスを他のセットへのアクセスより多く有する可能性がある。任意のセット内の空間を使い果たした場合、いくつかの実施形態においては、正確性を保証するために投機ミスがトリガされる。Mを増加させると、この問題が緩和するが、競合チェックハードウェアのより多くのコピーが存在することを強制し得る。すべてのM個の競合チェックを同時に実行するためには(いくつかの実施形態においてなされるように)、競合チェックロジックのM個のコピーが存在する。

【0080】

B、N、Mおよびハッシュ関数を特定の方法で選択することで、L1データキャッシュと非常に類似する方法で構造が編成されることを可能にする。特に、Bがキャッシュラインサイズ、NがL1データキャッシュ内のセット数、MがL1データキャッシュの結合性、およびハッシュ関数がアドレスの最下位ビット(右シフト後)となるようにする。この構造は、L1データキャッシュと同数のエントリおよび編成を有することになり、その実装を簡易にできる。

【0081】

最後に、代替的な実施形態は、アクセスタイプ情報を格納する必要性を回避し、および競合チェック中に、アクセスタイプをチェックする必要性を回避するために、読み取りおよび書き込みについて、別個のブルームフィルタを使用することに留意されたい。代わりに、読み取りについては、実施形態は「書き込み」フィルタに対してのみ競合チェックを実行し、投機ミスが存在しない場合、要素を「読み取り」フィルタに挿入する。同様に、書き込みについては、実施形態は「読み取り」および「書き込み」フィルタの両方に対して競合チェックを実行し、投機ミスが存在しない場合、要素を「書き込み」フィルタに挿入する。

【0082】

図4は、DSX追跡ハードウェアによって実行されるDSX投機ミス検出の例示的な方

10

20

30

40

50

法を示す。401において、DSXが開始される、または前の投機的イタレーションがコミットされる。例えば、YBEGIN命令が実行される。この命令の実行により、エントリ309内の有効なビットをクリアし、状態レジスタ（上記のDSX状態レジスタ等）内の投機アクティブフラグを設定する（既に設定されていない場合）。DSXが開始された後、投機的メモリ命令が実行され、テスト対象のデータ要素を提供する。

【0083】

403において、投機的メモリ命令からのテスト対象のデータ要素は、Bバイト以下のチャンクに細分化される。ハッシュテーブルは、Bバイトの粒度でアクセスされる（すなわち、アドレスの低ビットは破棄される）。要素が十分大きく、および／または、アライメントされていない場合、要素はBバイト境界を越えてよく、その場合、要素は複数のチャンクに細分化される。

10

【0084】

チャンクごとに、以下（405～421）が実行される。チャンクの開始アドレスは、 $\log_2 B$ 分右シフトされる。407において、シフトされたアドレスはハッシュされ、インデックス値を生成する。

【0085】

409において、インデックス値を使用して、ハッシュテーブルの対応するセットのルックアップが行われ、411において、当該セットのすべてのエントリが読み出される。

【0086】

413において、読み出された各エントリについて、テスト対象の要素に対し競合チェック（上記のような）が実行される。415において、すべての競合チェックに対するOR演算が実行される。417において、いずれかのチェックが競合を示す（結果、ORは1である）場合、419において投機ミスの指標が形成される。この時点で、DSXは通常アボートされる。投機ミスがない場合、421において、セット内の無効なエントリが検出され、無効なエントリはテスト対象要素の情報で埋められ、有効とマークされる。無効なエントリが存在しない場合、投機ミスがトリガされる。

20

【0087】

図5A～図5Bは、DSX追跡ハードウェアによって実行されるDSX投機ミス検出の例示的な方法を示す。501において、DSXが開始される、または前の投機的イタレーションがコミットされる。例えば、YBEGIN命令が実行される。

30

【0088】

この命令の実行により、503において、エントリ309内の有効なビットをクリアすることによって、追跡ハードウェアをリセットし且つ状態レジスタ（上記のDSX状態レジスタ等）内の投機アクティブフラグを設定する（既に設定されていない場合）。

【0089】

505において、投機的メモリ命令が実行される。これらの命令の例は、上記されている。507において、投機的命令からのテスト対象要素の数であるカウンタ（e）がゼロに設定され、509において、IDが計算される（ $ID = \text{シーケンス番号} + \text{ストライド} \times e$ ）。

【0090】

40

511において、いずれかの前の書き込みがカウンタ値eと重複するかどうかの判断がなされる。これは、前の格納（書き込み）に対する依存性チェックとして動作する。重複する書き込みがあれば、513において、競合チェックが実行される。いくつかの実施形態において、この競合チェックは次のものの確認を試みる。すなわち、i) エントリ309内のシーケンス番号がテスト対象要素315のシーケンス番号より小さく且つエントリ309内のID番号がテスト対象要素315のID番号より大きいかどうか、またはii) エントリ309内のシーケンス番号がテスト対象要素315のシーケンス番号より大きく且つエントリ309内のID番号がテスト対象要素315のID番号より小さいかどうかである。

【0091】

50

競合が存在する場合、515において、投機ミスがトリガされる。競合が存在しない場合、または重複する前の書き込みが存在しなかった場合、517において、投機的メモリ命令は書き込みであるかどうかの判断がなされる。

【0092】

はいの場合、519において、前のいずれかの読み取りがカウンタ値eと重複するかどうかの判断がなされる。これは、前のロード（読み取り）に対する依存性チェックとして動作する。いずれかの重複する読み取りがあれば、521において、競合チェックが実行される。いくつかの実施形態において、この競合チェックは次のものの確認を試みる。すなわち、i) エントリ309内のシーケンス番号がテスト対象要素315のシーケンス番号より小さく且つエントリ309内のID番号がテスト対象要素315のID番号より大きいかどうか、またはii) エントリ309内のシーケンス番号がテスト対象要素315のシーケンス番号より大きく且つエントリ309内のID番号がテスト対象要素315のID番号より小さいかどうかである。

10

【0093】

競合が存在する場合、523において、投機ミスがトリガされる。競合が存在しない場合、または重複する前の読み取りが存在しなかった場合、525において、カウンタeがインクリメントされる。

【0094】

526において、カウンタeが投機的メモリ命令内の要素数に等しいかどうかの判断がなされる。換言すると、すべての要素が評価済みかどうか？で、いいえの場合、509において、別のIDが計算される。はいの場合、527において、ハードウェアは別の命令の実行を待機する。次の命令が別の投機的メモリ命令である場合、507においてカウンタがリセットされる。次の命令がYBEGINの場合、503においてハードウェアがリセットされるといった具合である。次の命令がYENDの場合、529においてDSXは無効にされる。

20

【0095】

[YBEGIN命令]

【0096】

図6は、DSXを開始する命令の実行に係る一実施形態を示す。本明細書で説明する通り、この命令は「YBEGIN」と称され、DSX領域の開始をシグナリングするために使用される。もちろん、当該命令は別の名前で称されてもよい。いくつかの実施形態において、この実行は、中央処理装置（CPU）、グラフィック処理ユニット（GPU）、アクセラレーテッド処理ユニット（APU）、デジタル信号プロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

30

【0097】

601において、YBEGIN命令が受信／フェッチされる。例えば、命令はメモリから命令キャッシュへフェッチされ、または命令キャッシュからフェッチされる。フェッチされた命令は、後述のいくつかの形式のうちの1つを取ってよい。

【0098】

図7は、YBEGIN命令フォーマットのいくつかの例示的な実施形態を示す。701に図示の通り、一実施形態において、YBEGIN命令は、オペコード（YBEGIN）およびフォールバックアドレスの変位を提供するための単一のオペランドを含み、これは、プログラム実行が投機ミス进行处理する必要がある場合である。要するに、変位値は、フォールバックアドレスの一部である。いくつかの実施形態において、この変位値は即値オペランドとして提供される。他の実施形態において、この変位値はレジスタまたはメモリ位置オペランド内に格納される。YBEGIN実装に応じ、DSX状態レジスタ、ネストカウントレジスタおよび／またはRTM状態レジスタに対する暗示的オペランドが使用される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等

40

50

）といった具体である。

【0099】

703に図示の通り、別の実施形態においては、YBEGIN命令はオペコードおよび変位オペランドのみでなく、DSX状態レジスタ等のDSX状態に対する明示的オペランドも含む。YBEGIN実装に応じ、ネストカウントレジスタおよび／またはRTM状態レジスタに対する暗示的オペランドが使用される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

【0100】

705に図示の通り、別の実施形態においては、YBEGIN命令はオペコードおよび変位オペランドのみでなく、DSXネストカウントレジスタ等のDSXネストカウントに対する明示的オペランドも含む。上記の通り、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）といった具体である。YBEGIN実装に応じ、DSX状態レジスタおよび／またはRTM状態レジスタに対する暗示的オペランドが使用される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

10

【0101】

707に図示の通り、別の実施形態においては、YBEGIN命令はオペコードおよび変位オペランドのみでなく、DSX状態レジスタ等のDSX状態およびDSXネストカウントレジスタ等のDSXネストカウントに対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタ等のような全体的な状態レジスタ等）。そして、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）。YBEGIN実装に応じ、RTM状態レジスタに対する暗示的オペランドが使用される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具合である。

20

【0102】

709に図示の通り、別の実施形態において、YBEGIN命令は、オペコードおよび変位オペランドのみでなく、DSX状態レジスタ等のDSX状態、DSXネストカウントレジスタ等のDSXネストカウントおよびRTM状態に対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタ等のような全体的な状態レジスタ等）。そして、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）。

30

【0103】

もちろん、YBEGINの他の変形例も考え得る。例えば、変位値を提供する代わりに、命令はフォールバックアドレス自体を即値、レジスタ、またはメモリ位置のいずれかに含む。

40

【0104】

図6の参照に戻ると、603において、フェッチ／受信されたYBEGIN命令がデコードされる。いくつかの実施形態において、命令は、後述のようなハードウェアデコーダによってデコードされる。いくつかの実施形態において、命令はマイクロオペレーション（マイクロop）へとデコードされる。例えば、一部のCISCベースの機械は通常、マクロ命令から派生したマイクロオペレーションを使用する。他の実施形態において、デコーディングは、ジャストインタイムコンパイル等のソフトウェアルーチンの一部である。

【0105】

605において、デコードされた命令に関連付けられた任意のオペランドが取得される

50

。例えば、DSXレジスタ、DSXネストカウントレジスタ、および／またはRTM状態レジスタのうちの1または複数からデータが取得される。

【0106】

607において、デコードされたYBEGIN命令が実行される。命令がマイクロopへとデコードされる実施形態においては、これらのマイクロopが実行される。デコードされた命令の実行により、ハードウェアに対し、以下の実行されるべき動作のうちの1または複数を実行させる。すなわち、1) RTMトランザクションがアクティブであることを判断し且つそのトランザクションを続行する、2) YBEGIN命令の命令ポインタに追加された変位値を使用し、フォールバックアドレスを計算する、3) DSXネストカウンタをインクリメントする、4) アボートする、5) DSX状態をアクティブに設定する、および／または6) DSX追跡ハードウェアをリセットする。

10

【0107】

通常、YBEGIN命令のインスタンスにおいて、アクティブなRTMトランザクションが存在しない場合、DSX状態はアクティブに設定され、DSXネストカウンタはインクリメントされ（カウンタが最大値より小さい場合）、DSX追跡ハードウェアはリセットされ（例えば、上記のように）、およびDSX領域を開始すべく、変位値を使用してフォールバックアドレスが計算される。上記の通り、DSXの状態は通常、図1に関し記載したDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。DSX追跡ハードウェアのリセットについても説明済みである。上記の通り、DSXの状態は通常、図1に関し記載したDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェックされ、DSXが実際に発生していたかどうか判断されてよい。

20

【0108】

DSXを開始できない何らかの理由が存在した場合、他の潜在的な動作のうちの1または複数が発生する。例えば、RTMをサポートするプロセッサのいくつかの実施形態においては、RTMトランザクションがアクティブであった場合、もともとDSXアクティブが生じるべきではなかったため、RTMが遂行される。もともとDSXの設定について何らかの間違ひがある（ネストカウンタが正しくない）場合、アボートが発生することになる。また、いくつかの実施形態において、DSXが生じなかった場合、エラーが生成され、処理が実行されない（NOP）。どの動作が実行されるかに関わらず、多くの実施形態において、その動作の後、保留中のDSXが存在しないことを示すために、DSX状態がリセット（設定されていた場合）される。

30

【0109】

図8は、YBEGIN命令等の命令の実行に係る詳細な実施形態を示す。例えば、いくつかの実施形態において、このフローは図6の607のボックスである。いくつかの実施形態において、この実行は、中央処理装置（CPU）、グラフィック処理ユニット（GPU）、アクセラレーテッド処理ユニット（APU）、デジタル信号プロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

40

【0110】

いくつかの実施形態において、例えば、RTMトランザクションをサポートするプロセッサにおいては、801において、RTMトランザクションが発生しているかどうかの判断がなされる。例えば、RTMをサポートするプロセッサのいくつかの実施形態において、RTMトランザクションがアクティブであった場合、もともとDSXアクティブが生じるべきではなかった。この例では、RTMトランザクションにおいて何らかの問題が発生し、その終了手順がアクティブ化されるべきである。通常、RTMトランザクションの状態はRTM制御および状態レジスタ等のレジスタに格納される。プロセッサのハードウェア

50

アは、このレジスタの内容を評価し、R T Mトランザクションが発生しているかどうかを判断する。R T Mトランザクションが発生している場合、8 0 3において、R T Mトランザクションは処理を続行する。

【0 1 1 1】

R T Mトランザクションが発生していない場合、またはR T Mがサポートされていない場合、8 0 5において、現在のD S Xネストカウントが最大ネストカウントより小さいかどうかの判断がなされる。いくつかの実施形態において、現在のネストカウントを格納するためのネストカウントレジスタが、Y B E G I N命令によってオペランドとして提供される。代替的に、現在のネストカウントを格納するために使用される専用のネストカウントレジスタがハードウェアに存在してよい。最大ネストカウントは、対応するD S X終了（例えば、Y E N D命令命令を介する）がない状態で発生し得るD S X開始（例えば、Y B E G I N命令を介する）の最大数である。

10

【0 1 1 2】

現在のD S Xネストカウントが最大値より大きい場合、8 0 7において、アボートが発生する。いくつかの実施形態において、アボートは、D S X復元回路1 3 5等の復元回路を使用して、ロールバックをトリガする。他の実施形態においては、Y A B O R T命令は後述の通り実行される。そこでは、フォールバックアドレスへのロールバックを実行するだけでなく、投機的に格納された書き込みを破棄し、現在のネストカウントをリセットし、D S X状態を非アクティブに設定する。上記の通り、D S X状態は通常、図1中に示されるD S X状態および制御レジスタ（D S X S R）等の制御レジスタに格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のD S X状態フラグ等の他の手段が利用されてもよい。

20

【0 1 1 3】

現在のネストカウントが最大値より大きくない場合、8 0 9において、現在のD S Xネストカウントがインクリメントされる。

【0 1 1 4】

8 1 1において、現在のD S Xネストカウントが1に等しいかどうかの判断がなされる。現在のD S Xネストカウントが1に等しい場合、いくつかの実施形態においては、8 1 3で、Y B E G I N命令によって提供される変位値をY B E G I N命令の後にある命令のアドレスに追加することによって、フォールバックアドレスが計算される。Y B E G I N命令がフォールバックアドレスを提供した実施形態においては、この計算は必要ではない。

30

【0 1 1 5】

8 1 5において、D S X状態がアクティブに設定され（必要な場合）、D S X追跡ハードウェアがリセットされる（例えば、上記の通りに）。例えば、上記の通り、D S Xの状態は通常、図1に関し記載されたD S X状態および制御レジスタ（D S X S R）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のD S X状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェックされ、D S Xが実際に発生していたかどうか判断されてよい。

40

【0 1 1 6】

図9は、Y B E G I N命令等の命令の実行を示す擬似コードの例を示す。

【0 1 1 7】

[Y B E G I N W I T H S T R I D E 命令]

【0 1 1 8】

図10は、D S Xを開始する命令の実行に係る一実施形態を示す。本明細書で説明する通り、この命令は「Y B E G I N W I T H S T R I D E」と称され、D S X領域の開始をシグナリングするために使用される。もちろん、当該命令は別の名前で称されてもよい。いくつかの実施形態において、この実行は、中央処理装置（C P U）、グラフィック処理ユニット（G P U）、アクセラレーテッド処理ユニット（A P U）、デジタル信号プ

50

ロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

【0119】

1001において、YBEGIN WITH STRIDE命令が受信／フェッチされる。例えば、命令はメモリから命令キャッシュへフェッチされ、または命令キャッシュからフェッチされる。フェッチされた命令は、後述のいくつかの形式のうちの1つを取ってよい。

【0120】

図11はYBEGIN WITH STRIDE命令フォーマットのいくつかの例示的な実施形態を示す。1101に図示の通り、一実施形態において、YBEGIN WITH STRIDE命令は、オペコード（YBEGIN WITH STRIDE）およびフォールバックアドレスの変位を提供するためのオペランドを含み、これは、プログラム実行が投機ミスおよびストライド値オペランドを処理する必要がある場合である。要するに、変位値は、フォールバックアドレスの一部である。いくつかの実施形態において、変位は即値オペランドとして提供される。他の実施形態において、変位値はレジスタまたはメモリ位置オペランド内に格納される。いくつかの実施形態において、ストライドは即値オペランドとして提供される。他の実施形態において、ストライドはレジスタまたはメモリ位置オペランド内に格納される。YBEGIN WITH STRIDE実装に応じ、DSX状態レジスタ、ネストカウントレジスタおよび／またはRTM状態レジスタに対する暗示的オペランドが使用される。

【0121】

1103に図示の通り、別の実施形態においては、YBEGIN WITH STRIDE命令はオペコードおよび変位オペランドおよびストライド値オペランドのみでなく、DSX状態レジスタ等のDSX状態に対する明示的オペランドも含む。いくつかの実施形態において、変位は即値オペランドとして提供される。他の実施形態において、変位値はレジスタまたはメモリ位置オペランド内に格納される。いくつかの実施形態において、ストライドは即値オペランドとして提供される。他の実施形態において、ストライドはレジスタまたはメモリ位置オペランド内に格納される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。YBEGIN WITH STRIDE実装に応じ、ネストカウントレジスタおよび／またはRTM状態レジスタに対する暗示的オペランドが使用される。

【0122】

1105に図示の通り、別の実施形態においては、YBEGIN WITH STRIDE命令はオペコード、変位オペランドおよびストライド値オペランドのみでなく、DSXネストカウントレジスタ等のDSXネストカウントに対する明示的オペランドも含む。いくつかの実施形態において、変位は即値オペランドとして提供される。他の実施形態において、変位値はレジスタまたはメモリ位置オペランド内に格納される。いくつかの実施形態において、ストライドは即値オペランドとして提供される。他の実施形態において、ストライドはレジスタまたはメモリ位置オペランド内に格納される。上記の通り、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）。YBEGIN WITH STRIDE実装に応じ、DSX状態レジスタおよび／またはRTM状態レジスタに対する暗示的オペランドが使用される。

【0123】

1107に図示の通り、別の実施形態においては、YBEGIN WITH STRIDE命令はオペコード、変位オペランドおよびストライド値オペランドのみでなく、DSX状態レジスタ等のDSX状態およびDSXネストカウントレジスタ等のDSXネストカウントに対する明示的オペランドも含む。いくつかの実施形態において、変位は即値オペランドとして提供される。他の実施形態において、変位値はレジスタまたはメモリ位置オ

ペランド内に格納される。いくつかの実施形態において、ストライドは即値オペラントとして提供される。他の実施形態において、ストライドはレジスタまたはメモリ位置オペラント内に格納される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタ等のような全体的な状態レジスタ等）。そして、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）。YBEGIN WITH STRIDE実装に応じ、RTM状態レジスタに対する暗示的オペラントが使用される。

【0124】

1109に図示の通り、別の実施形態においては、YBEGIN WITH STRIDE命令は、オペコード、変位オペラントおよびストライド値オペラントのみでなく、DSX状態レジスタ等のDSX状態、DSXネストカウントレジスタ等のDSXネストカウントおよびRTM状態レジスタに対する明示的オペラントも含む。いくつかの実施形態において、変位は即値オペラントとして提供される。他の実施形態において、変位値はレジスタまたはメモリ位置オペラント内に格納される。いくつかの実施形態において、ストライドは即値オペラントとして提供される。他の実施形態において、ストライドはレジスタまたはメモリ位置オペラント内に格納される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタ等のような全体的な状態レジスタ等）。そして、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）。

【0125】

もちろん、YBEGIN WITH STRIDEの他の変形例も考え得る。例えば、変位値を提供する代わりに、命令はフォールバックアドレス自体を即値、レジスタ、またはメモリ位置のいずれかに含む。

【0126】

図10の参照に戻ると、1003において、フェッチ／受信されたYBEGIN WITH STRIDE命令がデコードされる。いくつかの実施形態において、命令は、後述のようなハードウェアデコーダによってデコードされる。いくつかの実施形態において、命令はマイクロオペレーション（マイクロop）へとデコードされる。例えば、一部のCISCベースの機械は通常、マクロ命令から派生したマイクロオペレーションを使用する。他の実施形態において、デコーディングは、ジャストインタイムコンパイル等のソフトウェアルーチンの一部である。

【0127】

1005において、デコードされたYBEGIN WITH STRIDE命令に関連付けられた任意のオペラントが取得される。例えば、DSXレジスタ、DSXネストカウントレジスタおよび／またはRTM状態レジスタのうちの1または複数からデータが取得される。

【0128】

1007において、デコードされたYBEGIN WITH STRIDE命令が実行される。命令がマイクロopへとデコードされる実施形態においては、これらのマイクロopが実行される。デコードされた命令の実行により、ハードウェアに対し、以下の実行されるべき動作のうちの1または複数を実行させる。すなわち、1) RTMトランザクションがアクティブであることを判断し且つそのトランザクションを開始する、2) YBEGIN WITH STRIDE命令の命令ポインタに追加された変位値を使用し、フォールバックアドレスを計算する、3) DSXネストカウントをインクリメントする、4) アポートする、5) DSX状態をアクティブに設定する、6) DSX追跡ハードウェアをリセットする、および／または7) ストライド値をDSXハードウェアトラッカーに提供する。

【0129】

通常、YBEGIN WITH STRIDE命令の第1のインスタンスにおいて、アクティブなRTMトランザクションが存在しない場合、DSX状態がアクティブに設定され、DSX追跡ハードウェアがリセットされ（例えば、上記の通り、提供されたストライド値を使用して）、DSX領域を開始するために、変位値を使用してフォールバックアドレスが計算される。上記の通り、DSXの状態は通常、図1に関し記載したDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。DSX追跡ハードウェアのリセットについても説明済みである。

【0130】

通常、YBEGIN WITH STRIDE命令のインスタンスにおいて、アクティブなRTMトランザクションが存在しない場合、DSX状態はアクティブに設定され、DSXネストカウントはインクリメントされ（カウントが最大値より小さい場合）、DSX追跡ハードウェアはリセットされ（例えば、上記の通り、提供されたストライドを使用して）、DSX領域を開始するために、変位値を使用してフォールバックアドレスが計算される。上記の通り、DSXの状態は通常、図1に関し記載したDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。DSX追跡ハードウェアのリセットについても説明済みである。上記の通り、DSXの状態は通常、図1に関し記載したDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェックされ、DSXが実際に発生していたかどうか判断されてよい。

【0131】

DSXが開始できない何らかの理由が存在した場合、他の潜在的な動作のうちの1または複数が発生する。例えば、RTMをサポートするプロセッサのいくつかの実施形態においては、RTMトランザクションがアクティブであった場合、もともとDSXアクティブが生じるべきではなかったので、RTMが遂行される。もともとDSXの設定について何らかの間違ひがある（ネストカウントが正しくない）場合、アボートが発生することになる。また、いくつかの実施形態において、DSXが存在しなかった場合、エラーが生成され、処理が実行されない（NOP）。どの動作が実行されるかに関わらず、多くの実施形態において、その動作の後、保留中のDSXが存在しないことを示すために、DSX状態がリセット（設定されていた場合）される。

【0132】

図12は、YBEGIN WITH STRIDE命令等の命令の実行に係る詳細な実施形態を示す。例えば、いくつかの実施形態において、このフローは図10の1007のボックスである。いくつかの実施形態において、この実行は、中央処理装置（CPU）、グラフィック処理ユニット（GPU）、アクセラレーテッド処理ユニット（APU）、デジタル信号プロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

【0133】

いくつかの実施形態において、例えば、RTMトランザクションをサポートするプロセッサにおいては、1201において、RTMトランザクションが発生しているかどうかの判断がなされる。例えば、RTMをサポートするプロセッサのいくつかの実施形態において、RTMトランザクションがアクティブであった場合、もともとDSXアクティブが生じるべきではなかった。この例では、RTMトランザクションにおいて何らかの問題が発生し、その終了手順がアクティブ化されるべきである。通常、RTMトランザクションの状態はRTM制御および状態レジスタ等のレジスタに格納される。プロセッサのハードウ

エアは、このレジスタの内容を評価し、RTMトランザクションが発生しているかどうかを判断する。RTMトランザクションが発生している場合、1203において、RTMトランザクションは処理を続行する。

【0134】

RTMトランザクションが発生していない場合、またはRTMがサポートされていない場合、1205において、現在のDSXネストカウントが最大ネストカウントより小さいかどうかの判断がなされる。いくつかの実施形態において、現在のネストカウントを格納するためのネストカウントレジスタが、YBEGIN WITH STRIDE命令によってオペランドとして提供される。代替的に、現在のネストカウントを格納するために使用される専用のネストカウントレジスタがハードウェアに存在してよい。最大ネストカウ
10
ントは、対応するDSX終了（例えば、YEND命令を介する）がない状態で発生し得るDSX開始（例えば、YBEGIN命令を介する）の最大数である。

【0135】

現在のネストカウントが最大値より大きい場合、1207において、アボートが発生する。いくつかの実施形態において、アボートはロールバックをトリガする。他の実施形態においては、YABORT命令は後述の通り実行される。そこでは、フォールバックアドレスへのロールバックを実行するだけでなく、また投機的に格納された書き込みを破棄し、現在のネストカウントをリセットし、DSX状態を非アクティブに設定する。上記の通り、DSX状態は通常、図1中に示されるDSX状態および制御レジスタ（DSXSR）等の制御レジスタに格納される。しかしながら、専用ではない制御／状態レジスタ（フラ
20
グレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。

【0136】

現在のネストカウントが最大値より大きくない場合、1209において、現在のDSXネストカウントがインクリメントされる。

【0137】

1211において、現在のDSXネストカウントが1に等しいかどうかの判断がなされる。現在のDSXネストカウントが1に等しい場合、いくつかの実施形態においては、1213において、YBEGIN WITH STRIDE命令によって提供される変位値をYBEGIN WITH STRIDE命令の後にある命令のアドレスに追加することによって、フォールバックアドレスが計算される。YBEGIN WITH STRID
30
E命令がフォールバックアドレスを提供した実施形態においては、この計算は必要ではない。

【0138】

1215において、DSX状態がアクティブに設定され（必要な場合）、DSX追跡ハードウェアがリセットされる（例えば、上記の通り、提供されたスライド値を使用して）。例えば、上記の通り、DSXの状態は通常、図1に関し記載されたDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェ
40
ックされ、DSXが実際に発生していたかどうか判断されてよい。

【0139】

[YCONTINUE命令]

【0140】

DSXが問題なく終了すると（例えば、ループのイタレーションが最後まで実行された）、いくつかの実施形態において、後述の通り、投機的領域の終了を示すための命令（YEND）が実行される。つまり、この命令の実行は、後述の通り、現在の投機的状態のコミットメント（書き込まれていないすべての書き込み）および現在の投機的領域からの終了を発生させる。その後、ループの別のイタレーションが、別のYBEGINを呼び出すことによって開始されてよい。

【0141】

10

20

30

40

50

しかしながら、いくつかの実施形態において、Y B E G I N、Y E N D、Y B E G I N等のこのサイクルに対する最適化が、投機実行がこれ以上必要でない場合（例えば、ストア間に競合がない場合）に、現在のループイタレーションをコミットするためのC o n t i n u e命令の使用を通して利用可能である。C o n t i n u e命令はまた、Y B E G I Nを呼び出す必要なしに、新しい投機的ループのイタレーションを開始する。

【0142】

図13は、D S Xを終了させずに、D S Xを続行する命令の実行に係る一実施形態を示す。本明細書で説明する通り、この命令は「Y C O N T I N U E」と称され、トランザクションの終了をシグナリングするために使用される。もちろん、当該命令は別の名前で称されてもよい。

【0143】

いくつかの実施形態において、この実行は、中央処理装置（C P U）、グラフィック処理ユニット（G P U）、アクセラレーテッド処理ユニット（A P U）、デジタル信号プロセッサ（D S P）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

【0144】

1301において、Y C O N T I N U E命令が受信／フェッチされる。例えば、命令はメモリから命令キャッシュへフェッチされ、または命令キャッシュからフェッチされる。フェッチされた命令は、後述のいくつかの形式のうちの1つを取ってよい。

【0145】

図14は、Y C O N T I N U E命令フォーマットのいくつかの例示的な実施形態を示す。1401に図示の通り、一実施形態において、Y C O N T I N U E命令は、オペコード（Y C O N T I N U E）を含むが、明示的オペランドは含まない。Y C O N T I N U E実装に応じ、D S X状態レジスタおよびネストカウントレジスタに対する暗示的オペランドが使用される。上記の通り、D S Xネストカウントは専用レジスタであってよく、レジスタ内のフラグはD S Xネストカウントに専用でなくてもよい（全体的な状態レジスタ等）といった具合である。また、D S X状態レジスタは専用レジスタであってよく、レジスタ内のフラグはD S X状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

【0146】

1403に図示の通り、別の実施形態において、Y C O N T I N U E命令はオペコードのみでなく、D S X状態レジスタ等のD S X状態に対する明示的オペランドも含む。Y C O N T I N U E実装に応じ、ネストカウントレジスタに対する暗示的オペランドが使用される。上記の通り、D S Xネストカウントは専用レジスタであってよく、レジスタ内のフラグはD S Xネストカウントに専用でなくてもよい（全体的な状態レジスタ等）といった具体である。また、D S X状態レジスタは専用レジスタであってよく、レジスタ内のフラグはD S X状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

【0147】

1405に図示の通り、別の実施形態においては、Y C O N T I N U E命令はオペコードのみでなく、D S Xネストカウントレジスタ等のD S Xネストカウントに対する明示的オペランドも含む。Y C O N T I N U E実装に応じ、D S X状態レジスタに対する暗示的オペランドが使用される。上記の通り、D S Xネストカウントは専用レジスタであってよく、レジスタ内のフラグはD S Xネストカウントに専用でなくてもよい（全体的な状態レジスタ等）といった具合である。また、D S X状態レジスタは専用レジスタであってよく、レジスタ内のフラグはD S X状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

【0148】

1407に図示の通り、別の実施形態においては、Y C O N T I N U E命令はオペコードのみでなく、D S X状態レジスタ等のD S X状態およびD S Xネストカウントレジスタ

10

20

30

40

50

等のDSXネストカウントに対する明示的オペランドも含む。上記の通り、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい（全体的な状態レジスタ等）といった具合である。また、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

【0149】

図13の参照に戻ると、1303において、フェッチ／受信されたYCONTINUE命令がデコードされる。いくつかの実施形態において、命令は、後述のようなハードウェアデコーダによってデコードされる。いくつかの実施形態において、命令はマイクロオペレーション（マイクロop）へとデコードされる。例えば、一部のCISCベースの機械は通常、マクロ命令から派生したマイクロオペレーションを使用する。他の実施形態において、デコーディングは、ジャストインタイムコンパイル等のソフトウェアルーチンの一部である。

10

【0150】

1305において、デコードされたYCONTINUE命令に関連付けられた任意のオペランドが取得される。例えば、DSXレジスタおよびDSXネストカウントレジスタのうちの1または複数からデータが取得される。

【0151】

1307において、デコードされたYCONTINUE命令が実行される。命令がマイクロopへとデコードされる実施形態においては、これらのマイクロopが実行される。デコードされた命令の実行は、ハードウェアに対し、実行されるべき次の動作のうちの1または複数を実行させる。すなわち、1）投機実行がこれ以上必要ではないとき、DSXに関連付けられた投機的書き込みがコミットされるべきであることを判断し且つそれらをコミットし且つ新しい投機的ループのイタレーション（新しいDSX領域等）を開始する、および／または2）処理を実行しない。

20

【0152】

これらの動作のうちの第1のもの（投機的書き込みを確定し、新しい投機的ループのイタレーションを開始する）は、上記のDSXチェックハードウェアによって実行されてよい。この動作において、DSXのループのイタレーションに関連付けられた投機的書き込みのすべてがコミットされる（それらはDSX外部でアクセス可能であるように格納される）が、YEND命令の場合とは異なり、DSXが存在しないことを示すためのDSX状態は設定されない。例えば、DSXに関連付けられたすべての書き込み（キャッシュ、レジスタまたはメモリ内に格納されたもの）がコミットされ、その結果、それらは確定され且つDSXの外部から可視であるようにされる。通常、DSXコミットは、DSXネストカウントが1である場合を除き、発生しない。それ以外の場合、いくつかの実施形態においては、何も処理が実行されない。

30

【0153】

いくつかの実施形態において、DSXがアクティブでない場合、何も処理が実行されなくてよい。

【0154】

図15は、YCONTINUE命令等の命令の実行に係る詳細な実施形態を示す。例えば、いくつかの実施形態において、このフローは図13のボックス1307である。いくつかの実施形態において、この実行は、中央処理装置（CPU）、グラフィック処理ユニット（GPU）、アクセラレーテッド処理ユニット（APU）、デジタル信号プロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

40

【0155】

1501において、DSXがアクティブであるかどうかの判断がなされる。上記の通り、DSX状態は通常、図1中に示されるDSX状態および制御レジスタ（DSXSR）等の制御レジスタに格納される。しかしながら、専用ではない制御／状態レジスタ（フラグ

50

レジスタ等) 内のDSX状態フラグ等の他の手段が利用されてもよい。状態がどこに格納されるかに関わらず、位置はプロセッサのハードウェアによってチェックされ、DSXが実際に発生していたかどうか判断される。

【0156】

DSXが発生していない場合、1503において、処理は実行されない。

【0157】

DSXが発生している場合、1505において、DSXネストカウントが1に等しいかどうかの判断がなされる。上記の通り、DSXネストカウントは通常、ネストカウントレジスタ内に格納される。DSXネストカウントが1でない場合、1507において、処理は実行されない。DSXネストカウントが1である場合、1509において、コミットおよびDSX再開がなされる。いくつかの実施形態において、コミットおよびDSX再開が発生する場合、以下のうちの1または複数が発生する。すなわち、1) DSX追跡ハードウェアがリセットされる(例えば、上記の通り)、2) フォールバックアドレスが計算される、および3) 前の投機的領域の投機的に実行された命令(書き込み)のコミットがなされる。

10

【0158】

図16は、YCONTINUE命令等の命令の実行を示す擬似コードの例を示す。

【0159】

[YABORT命令]

【0160】

時折、DSXをアボートするように要求する問題(投機ミス等)がDSX内に存在する。図17は、DSXをアボートする命令の実行に係る一実施形態を示す。本明細書に記載の通り、この命令は「YABORT」と称される。もちろん、当該命令は別の名前で称されてもよい。いくつかの実施形態において、この実行は、中央処理装置(CPU)、グラフィック処理ユニット(GPU)、アクセラレーテッド処理ユニット(APU)、デジタル信号プロセッサ(DSP)等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

20

【0161】

1701において、YABORT命令が受信/フェッチされる。例えば、命令はメモリから命令キャッシュへフェッチされ、または命令キャッシュからフェッチされる。フェッチされた命令は、後述のいくつかの形式のうちの1つを取ってよい。

30

【0162】

図18は、YABORT命令フォーマットのいくつかの例示的な実施形態を示す。一実施形態において、1801に図示の通り、YABORT命令はオペコード(YABORT)のみを含む。YABORT実装に応じ、DSX状態レジスタおよび/またはRTM状態レジスタに対する暗示的オペランドが使用される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい(フラグレジスタのような全体的な状態レジスタ等)といった具体である。

【0163】

1803に図示の通り、別の実施形態においては、YABORT命令はオペコードのみでなく、DSX状態レジスタ等のDSX状態レジスタに対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい(フラグレジスタのような全体的な状態レジスタ等)といった具体である。YABORT実装に応じ、RTM状態レジスタに対する暗示的オペランドが使用される。

40

【0164】

1805に図示の通り、別の実施形態においては、YABORT命令はオペコードのみでなく、DSX状態レジスタ等のDSX状態レジスタおよびRTM状態レジスタに対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく

50

、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）といった具体である。

【0165】

図17の参照に戻ると、1703において、フェッチ／受信されたYABORT命令がデコードされる。いくつかの実施形態において、命令は、後述のようなハードウェアデコーダによってデコードされる。いくつかの実施形態において、命令はマイクロオペレーション（マイクロop）へとデコードされる。例えば、一部のCISCベースの機械は通常、マクロ命令から派生したマイクロオペレーションを使用する。他の実施形態において、デコーディングは、ジャストインタイムコンパイル等のソフトウェアルーチンの一部である。

10

【0166】

1705において、デコードされたYABORT命令に関連付けられた任意のオペランドが取得される。例えば、DSXレジスタおよび／またはRTM状態レジスタのうちの1または複数からデータが取得される。

【0167】

1707において、デコードされたYABORT命令が実行される。命令がマイクロopへとデコードされる実施形態においては、これらのマイクロopが実行される。デコードされた命令の実行は、ハードウェアに対し、実行されるべき次の動作のうちの1または複数を実行させる。すなわち、1）RTMトランザクションがアクティブであることを判断し且つRTMトランザクションをアボートする、2）DSXがアクティブでないことを判断し且つ処理を実行しない、および／または3）任意のDSXネストカウントをリセットし、すべての投機的に実行された書き込みを破棄し、DSX状態を非アクティブに設定し且つ実行をフォールバックアドレスにロールバックすることによってDSXをアボートする。

20

【0168】

第1の動作に関し、RTM状態は通常、RTM状態および制御レジスタ内に格納される。このレジスタが、RTMトランザクションが発生していることを示す場合、YABORT命令が実行されるべきではなかった。よって、RTMトランザクションに関し問題が存在した場合、それはアボートするべきである。

【0169】

上記の通り、第2の動作および第3の動作に関し、DSXの状態は通常、図1に関し記載されたDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェックされ、DSXが実際に発生していたかどうか判断されてよい。このレジスタによって示されるDSXが存在しない場合、YABORT命令を実行する理由は存在せず、よって何も処理が実行されない（または同様の処理）。このレジスタによって示されるDSXが存在する場合、DSX追跡ハードウェアのリセット、すべての格納された投機的に実行された書き込みの破棄、DSX状態の非アクティブのリセット、および実行のロールバックを含む、DSXアボート処理が発生する。

30

40

【0170】

図19は、YABORT命令等の命令の実行に係る詳細な実施形態を示す。例えば、いくつかの実施形態において、このフローは図17のボックス1707である。いくつかの実施形態において、この実行は、中央処理装置（CPU）、グラフィック処理ユニット（GPU）、アクセラレーテッド処理ユニット（APU）、デジタル信号プロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

【0171】

例えば、RTMトランザクションをサポートするプロセッサのいくつかの実施形態において、1901において、RTMトランザクションが発生しているかどうかの判断がなさ

50

れる。例えば、R T Mをサポートするプロセッサのいくつかの実施形態において、R T Mトランザクションがアクティブであった場合、もともとD S Xアクティブが生じるべきではなかった。この例では、R T Mトランザクションにおいて何らかの問題が発生し、その終了手順がアクティブ化されるべきである。通常、R T Mトランザクションの状態はR T M制御および状態レジスタ等のレジスタに格納される。プロセッサのハードウェアは、このレジスタの内容を評価し、R T Mトランザクションが発生しているかどうかを判断する。R T Mトランザクションが発生している場合、1 9 0 3において、R T Mトランザクションは処理を続行する。

【0 1 7 2】

R T Mトランザクションが発生していない場合、またはR T Mがサポートされていない場合、1 9 0 5において、D S Xがアクティブかどうかの判断がなされる。D S Xの状態は通常、図1に関し記載したD S X状態および制御レジスタ（D S X S R）等のアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のD S X状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェックされ、D S Xが実際に発生していたかどうか判断されてよい。

【0 1 7 3】

このレジスタによって示されるD S Xが存在しない場合、1 9 0 7において、処理が実行されない。このレジスタによって示されるD S Xが存在する場合、1 9 0 9において、D S X追跡ハードウェアのリセット、すべての格納された投機的に実行された書き込みの破棄、D S X状態の非アクティブへのリセットおよび実行のロールバックを含む、D S Xアボート処理が発生する。

【0 1 7 4】

図20は、Y A B O R T命令等の命令の実行を示す擬似コードの例を示す。

【0 1 7 5】

[Y T E S T命令]

【0 1 7 6】

一般に、ソフトウェアが新しいD S X投機的領域を開始する前に、D S Xがアクティブか否かを認識することが望ましい。図21は、D S Xの状態をテストする命令の実行に係る一実施形態を示す。本明細書で記載の通り、この命令は「Y T E S T」として称され、フラグの使用を通して、D S Xがアクティブであるという指標を提供するために使用される。もちろん、当該命令は別の名前で称されてもよい。

【0 1 7 7】

いくつかの実施形態において、この実行は、中央処理装置（C P U）、グラフィック処理ユニット（G P U）、アクセラレーテッド処理ユニット（A P U）、デジタル信号プロセッサ（D S P）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

【0 1 7 8】

2 1 0 1において、Y T E S T命令が受信／フェッチされる。例えば、命令はメモリから命令キャッシュへフェッチされ、または命令キャッシュからフェッチされる。フェッチされた命令は、いくつかの形式のうちの1つを取ってよい。図22は、Y T E S T命令フォーマットのいくつかの例示的な実施形態を示す。2 2 0 1に図示の通り、一実施形態において、Y T E S T命令は、オペコード（Y T E S T）を含むが、明示的オペランドは含まない。D S X状態レジスタおよびフラグレジスタに対する暗示的オペランドが使用される。上記の通り、D S X状態レジスタは専用レジスタであってよく、レジスタ内のフラグはD S X状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）。例示的なフラグレジスタとしては、E F L A G Sレジスタが含まれる。特に、当該フラグレジスタは、ゼロフラグ（Z F）を格納する。

【0 1 7 9】

2 2 0 3に図示の通り、別の実施形態においては、Y T E S T命令はオペコードのみで

10

20

30

40

50

なく、DSX状態レジスタ等のDSX状態に対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタ等のような全体的な状態レジスタ等）。フラグレジスタに対する暗示的オペランドが使用される。例示的なフラグレジスタとしては、EFLAGレジスタが含まれる。特に、当該フラグレジスタはゼロフラグ（ZF）を格納する。

【0180】

2205に図示の通り、別の実施形態においては、YTEST命令はオペコードのみでなく、フラグレジスタに対する明示的オペランドも含む。例示的なフラグレジスタとしては、EFLAGレジスタが含まれる。特に、当該フラグレジスタは、ゼロフラグ（ZF）を格納する。DSX状態レジスタに対する暗示的オペランドが使用される。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）。

10

【0181】

2207に図示の通り、別の実施形態においては、YTEST命令はオペコードのみでなく、DSX状態レジスタ等のDSX状態およびフラグレジスタに対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい（フラグレジスタのような全体的な状態レジスタ等）。フラグレジスタに対する暗示的オペランドが使用される。例示的なフラグレジスタとしては、EFLAGレジスタが含まれる。特に、当該フラグレジスタは、ゼロフラグ（ZF）を格納する。

20

【0182】

図21の参照に戻ると、2103において、フェッチ／受信されたYTEST命令がデコードされる。いくつかの実施形態において、命令は、後述のようなハードウェアデコーダによってデコードされる。いくつかの実施形態において、命令はマイクロオペレーション（マイクロop）へとデコードされる。例えば、一部のCISCベースの機械は通常、マクロ命令から派生したマイクロオペレーションを使用する。他の実施形態において、デコーディングは、ジャストインタイムコンパイル等のソフトウェアルーチンの一部である。

【0183】

2105において、デコードされたYTEST命令に関連付けられた任意のオペランドが取得される。例えば、DSX状態レジスタからデータが取得される。

30

【0184】

2107において、デコードされたYTEST命令が実行される。命令がマイクロopへとデコードされる実施形態においては、これらのマイクロopが実行される。デコードされた命令の実行は、ハードウェアに対し、実行されるべき次の動作のうちの1または複数を実行させる。すなわち、1）DSX状態レジスタが、DSXがアクティブであると示すことを判断し、そうであれば、フラグレジスタ内のゼロフラグを0に設定する、または2）DSX状態レジスタが、DSXがアクティブでないと示すことを判断し、そうであれば、フラグレジスタ内のゼロフラグを1に設定する。もちろん、DSXがアクティブな状態を示すためにゼロフラグが使用される一方で、実施形態に応じ、他のフラグが使用される。

40

【0185】

図23は、YTEST命令等の命令の実行を示す擬似コードの例を示す。

【0186】

[YEND命令]

【0187】

DSXが問題なく終了すると（例えば、ループのイタレーションが最後まで実行された）、いくつかの実施形態において、投機的領域の終了を示すための命令が実行される。つまり、この命令の実行は、現在の投機的状態のコミットメント（書き込まれていないすべ

50

ての書き込み) および現在の投機的領域からの終了を発生させる。

【0188】

図24は、DSXを終了させる命令の実行に係る一実施形態を示す。本明細書に記載の通り、この命令は「YEND」と称され、DSXの終了をシグナリングするために使用される。もちろん、当該命令は別の名前で称されてもよい。

【0189】

いくつかの実施形態において、この実行は、中央処理装置(CPU)、グラフィック処理ユニット(GPU)、アクセラレーテッド処理ユニット(APU)、デジタル信号プロセッサ(DSP)等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

10

【0190】

2401において、YEND命令が受信／フェッチされる。例えば、命令はメモリから命令キャッシュへフェッチされ、または命令キャッシュからフェッチされる。フェッチされた命令は、いくつかの形式のうちの1つを取ってよい。図25は、YEND命令フォーマットのいくつかの例示的な実施形態を示す。2501に図示の通り、一実施形態において、YEND命令は、オペコード(YEND)を含むが、明示的オペランドは含まない。YEND実装に応じ、DSX状態、ネストカウントおよび／またはRTM状態に対する暗示的レジスタオペランドが使用される。

【0191】

2503に図示の通り、別の実施形態においては、YEND命令はオペコードのみでなく、DSX状態レジスタ等のDSX状態に対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい(フラグレジスタ等のような全体的な状態レジスタ等)。YEND実装に応じ、ネストカウントおよび／またはRTM状態に対する暗示的レジスタオペランドが使用される。

20

【0192】

2505に図示の通り、別の実施形態においては、YEND命令はオペコードのみでなく、DSXネストカウントレジスタ等のDSXネストカウントに対する明示的オペランドも含む。上記の通り、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい(全体的な状態レジスタ等)。YEND実装に応じ、DSX状態および／またはRTM状態に対する暗示的レジスタオペランドが使用される。

30

【0193】

2507に図示の通り、別の実施形態においては、YEND命令はオペコードのみでなく、DSX状態レジスタ等のDSX状態およびDSXネストカウントレジスタ等のDSXネストカウントに対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい(フラグレジスタ等のような全体的な状態レジスタ等)。そして、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい(全体的な状態レジスタ等)。YEND実装に応じ、RTM状態レジスタに対する暗示的オペランドが使用される。

40

【0194】

2509に図示の通り、別の実施形態においては、YEND命令はオペコードのみでなく、DSX状態レジスタ等のDSX状態およびDSXネストカウントレジスタ等のDSXネストカウントおよびRTM状態に対する明示的オペランドも含む。上記の通り、DSX状態レジスタは専用レジスタであってよく、レジスタ内のフラグはDSX状態に専用でなくてもよい(フラグレジスタ等のような全体的な状態レジスタ等)。そして、DSXネストカウントは専用レジスタであってよく、レジスタ内のフラグはDSXネストカウントに専用でなくてもよい(全体的な状態レジスタ等)。

【0195】

50

図24の参照に戻ると、2403において、フェッチ／受信されたYEND命令がデコードされる。いくつかの実施形態において、命令は、後述のようなハードウェアデコーダによってデコードされる。いくつかの実施形態において、命令はマイクロオペレーション（マイクロop）へとデコードされる。例えば、一部のCISCベースの機械は通常、マクロ命令から派生したマイクロオペレーションを使用する。他の実施形態において、デコーディングは、ジャストインタイムコンパイル等のソフトウェアルーチンの一部である。

【0196】

2405において、デコードされたYEND命令に関連付けられた任意のオペランドが取得される。例えば、DSXレジスタ、DSXネストカウントレジスタおよび／またはRTM状態レジスタのうちの1または複数からデータが取得される。

10

【0197】

2407において、デコードされたYEND命令が実行される。命令がマイクロopへとデコードされる実施形態においては、これらのマイクロopが実行される。デコードされた命令の実行は、ハードウェアに対し、実行されるべき以下の動作のうちの1または複数を実行させる。すなわち、1）DSXに関連付けられた投機的書き込みを確定する（それらをコミットする）、2）エラー（一般保護違反等）をシグナリングし、且つ処理を実行しない、3）DSXをアボートする、および／または4）RTMトランザクションを終了する。

【0198】

これらの動作のうちの第1のもの（投機的書き込みを確定する）は、DSXに関連付けられたすべての投機的書き込みがコミットされるようにし（それらがDSX外部からアクセス可能のように格納される）、DSXがDSX状態レジスタ内に存在しないことを示すためのDSX状態が設定される。例えば、DSXに関連付けられたすべての書き込み（キャッシュ、レジスタまたはメモリ内に格納されたもの）がコミットされ、その結果、それらが確定され且つDSXの外部から可視であるようにされる。通常、その投機実行のネストカウン트가ゼロである場合を除き、DSXは確定できない。ネストカウン트가ゼロより大きい場合、いくつかの実施形態においては、処理が実行されない。

20

【0199】

DSXを確定できない何らかの理由が存在した場合、他の3つの潜在的な動作のうちの1または複数が行われる。例えば、RTMをサポートするプロセッサのいくつかの実施形態において、RTMトランザクションがアクティブであった場合、もともとDSXアクティブが生じるべきではなかった。この例では、RTMトランザクションにおいて何らかの問題が発生し、その終了手順が、上記の第4の動作によって示される通りアクティブ化されるべきである。

30

【0200】

いくつかの実施形態において、DSXが存在しなかった場合、エラーが生成され、処理が実行されない（NOP）。例えば、上記の通り、DSXの状態は通常、図1に関し記載されたDSX状態および制御レジスタ（DSXSR）等のレジスタのようなアクセス可能な位置に格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。このレジスタは、コアのハードウェアによってチェックされ、DSXが実際に発生していたかどうか判断されてよい。

40

【0201】

いくつかの実施形態において、トランザクションのコミットにおいてエラーが存在する場合、アボート手順が実装される。例えば、RTMをサポートするプロセッサのいくつかの実施形態においては、RTMアボート手順がアクティブ化される。

【0202】

どの動作が実行されるかに関わらず、多くの実施形態において、その動作の後、保留中のDSXが存在しないことを示すために、DSX状態がリセット（設定されていた場合）される。

50

【0203】

図26は、YEND命令等の命令の実行に係る詳細な実施形態を示す。例えば、いくつかの実施形態において、このフローは、図24のボックス2407である。いくつかの実施形態において、この実行は、中央処理装置（CPU）、グラフィック処理ユニット（GPU）、アクセラレーテッド処理ユニット（APU）、デジタル信号プロセッサ（DSP）等のハードウェアデバイスの1または複数のハードウェアコアに対し実行される。他の実施形態においては、当該命令の実行はエミュレーションである。

【0204】

例えば、RTMトランザクションをサポートするプロセッサのいくつかの実施形態においては、2601において、RTMトランザクションが発生しているかどうかの判断がなされる。例えば、RTMをサポートするプロセッサのいくつかの実施形態において、RTMトランザクションがアクティブであった場合、もともとDSXアクティブが生じるべきではなかった。この例では、RTMトランザクションにおいて何らかの問題が発生し、その終了手順がアクティブ化されるべきである。通常、RTMトランザクションの状態はRTM制御および状態レジスタ等のレジスタに格納される。プロセッサのハードウェアは、このレジスタの内容を評価し、RTMトランザクションが発生しているかどうかを判断する。

10

【0205】

RTMトランザクションが発生している場合、2603において、そのRTMトランザクションを終了させる呼び出しがなされる。例えば、RTMトランザクションを終了させる命令が呼び出され、実行される。このような命令の例として、XENDが挙げられる。

20

【0206】

RTMトランザクションが発生していない場合、2605において、DSXがアクティブかどうかの判断がなされる。上記の通り、DSX状態は通常、図1中に示されるDSX状態および制御レジスタ（DSXSR）等の制御レジスタに格納される。しかしながら、専用ではない制御／状態レジスタ（フラグレジスタ等）内のDSX状態フラグ等の他の手段が利用されてもよい。状態がどこに格納されるかに関わらず、位置はプロセッサのハードウェアによってチェックされ、DSXが実際に発生していたかどうか判断される。

【0207】

DSXが発生していない場合、2607において、エラーが生成される。例えば、一般保護違反が生成される。また、いくつかの実施形態においては、処理が実行されない（nop）。

30

【0208】

DSXが発生している場合、2609において、DSXネストカウントがデクリメントされる。例えば、上記のようなDSXネストカウントレジスタに格納されたDSXネストカウントがデクリメントされる。

【0209】

2611において、DSXネストカウントがゼロに等しいかどうかの判断がなされる。上記の通り、DSXネストカウントは通常、レジスタ内に格納される。DSXネストカウントがゼロでない場合、いくつかの実施形態において、処理が実行されない。DSXネストカウントがゼロの場合、2615において、現在のDSXの投機的状態は確定され、コミットされる。

40

【0210】

2617において、コミットメントが成功したかどうかの判断がなされる。例えば、格納されたエラーが存在したかどうか？といったものである。存在しなかった場合、2621において、DSXがアボートされる。コミットメントが成功した場合、2619において、アクティブなDSXが存在しないことを示すためのDSX状態指標（DSX状態および制御レジスタ内に格納されるもの等）が設定される。いくつかの実施形態においては、この指標の設定は、2607のエラーの生成または2621のDSXのアボートの後に行われる。

50

【0211】

図27は、YEND命令等の命令の実行を示す擬似コードの例を示す。

【0212】

命令フォーマットの実施形態および上記命令を実行するための実行リソースについて以下記載する。

【0213】

命令セットは、1または複数の命令フォーマットを含む。特定の命令フォーマットは、とりわけ、実行されるべき演算（オペコード）およびその演算が実行されるべきオペランドを指定するための様々なフィールド（ビット数、ビット位置）を定義する。いくつかの命令フォーマットは、命令テンプレート（またはサブフォーマット）の定義を通して、さらに細分化されている。例えば、特定の命令フォーマットの命令テンプレートは、命令フォーマットのフィールドの異なるサブセットを有するように定義されてよく（含まれるフィールドは通常、同一順序であるが、少なくともいくつかは、含まれるフィールド数がより少ないので、異なるビット位置を有する）、および／または、異なって解釈される特定のフィールドを有するように定義されてよい。故に、ISAの各命令は、特定の命令フォーマット（また、定義されている場合には、その命令フォーマットの命令テンプレートのうちの特定の1つにおいて）を使用して表現され、演算およびオペランドを指定するためのフィールドを含む。例えば、例示的なADD命令は、特定のオペコード並びにそのオペコードを指定するためのオペコードフィールドおよびオペランド（ソース1／デスティネーションおよびソース2）を選択するためのオペランドフィールドを含む命令フォーマットを有する。命令ストリーム内にこのADD命令が出現すると、特定のオペランドを選択するオペランドフィールド内に特定の内容の有することになる。アドバンスベクトル拡張（AVX）（AVX1およびAVX2）と称され、ベクトル拡張（VEX）コーディングスキームを使用するSIMD拡張のセットがリリースおよび／または公開されている（例えば、2011年10月のインテル（登録商標）64およびIA-32アーキテクチャソフトウェアデベロッパーズマニュアル並びに2011年6月のインテル（登録商標）アドバンスベクトル拡張プログラミングリファレンスを参照）。

【0214】

[例示的命令フォーマット]

【0215】

本明細書に記載の命令の実施形態は異なる形式で具現化されてよい。また、例示的なシステム、アーキテクチャおよびパイプラインについて詳細に後述する。本命令の実施形態は、このようなシステム、アーキテクチャおよびパイプライン上で実行されてよいが、本発明の実施形態はそれらの具体的な内容に限定されるわけではない。

【0216】

[汎用ベクトル向け命令フォーマット]

【0217】

ベクトル向け命令フォーマットとは、ベクトル命令に好適な命令フォーマットである（例えば、ベクトル演算に特有の所定のフィールドが存在する）。実施形態は、ベクトル演算およびスカラ演算の両方がベクトル向け命令フォーマットを通してサポートされるように記載されているものの、代替的な実施形態は、ベクトル向け命令フォーマットのベクトル演算のみ使用する。

【0218】

図28A～28Bは、本発明の実施形態による、汎用ベクトル向け命令フォーマットおよびその命令テンプレートを示すブロック図である。図28Aは、本発明の実施形態による汎用ベクトル向け命令フォーマットおよびそのクラスA命令テンプレートを示すブロック図であり、これに対し、図28Bは、本発明の実施形態による汎用ベクトル向け命令フォーマットおよびそのクラスB命令テンプレートを示すブロック図である。具体的には、汎用ベクトル向け命令フォーマット2800に対し、クラスA命令テンプレートおよびクラスB命令テンプレートが定義され、クラスA命令テンプレートおよびクラスB命令テン

プレートは両方とも、メモリアクセスなし 2805 命令テンプレートおよびメモリアクセス 2820 命令テンプレートを含む。ベクトル向け命令フォーマットの文脈における汎用 (generic) という用語は、いずれの固有の命令セットにも関連付けられない命令フォーマットを指す。

【0219】

本発明の実施形態は、ベクトル向け命令フォーマットが次のものをサポートするように記載されている。すなわち、32ビット (4バイト) または64ビット (8バイト) データ要素幅 (またはサイズ) を備えた64バイトベクトルオペランド長 (またはサイズ) (つまり、64バイトベクトルは、16個のダブルワードサイズの要素または代替的に8個のクワッドワードサイズの要素のいずれかから成る) ; 16ビット (2バイト) または8ビット (1バイト) データ要素幅 (またはサイズ) を備えた64バイトベクトルオペランド長 (またはサイズ) ; 32ビット (4バイト) 、64ビット (8バイト) 、16ビット (2バイト) または8ビット (1バイト) データ要素幅 (またはサイズ) を備えた32バイトベクトルオペランド長 (またはサイズ) ; および32ビット (4バイト) 、64ビット (8バイト) 、16ビット (2バイト) または8ビット (1バイト) データ要素幅 (またはサイズ) を備えた16バイトベクトルオペランド長 (またはサイズ) 。一方で、代替的な実施形態は、より多い、より少ない、または異なるデータ要素幅 (例えば、128ビット (16バイト) データ要素幅) を備えたより多い、より少ない、および/または異なるベクトルオペランドサイズ (例えば、256バイトベクトルオペランド) をサポートしてよい。

【0220】

図28A中のクラスA命令テンプレートには次のものが含まれる。すなわち、1) メモリアクセスなし 2805 命令テンプレート内に、メモリアクセスなし、完全ラウンド制御タイプ演算 2810 命令テンプレートおよびメモリアクセスなし、データ変換タイプ演算 2815 命令テンプレートが存在するように図示されている。2) メモリアクセス 2820 命令テンプレート内に、メモリアクセス、一時的 2825 命令テンプレートおよびメモリアクセス、非一時的 2830 命令テンプレートが存在するように図示されている。図28B中のクラスB命令テンプレートには次のものが含まれる。すなわち、1) メモリアクセスなし 2805 命令テンプレート内に、メモリアクセスなし、書き込みマスク制御、部分的ラウンド制御タイプ演算 2812 命令テンプレートおよびメモリアクセスなし、書き込みマスク制御、vsizetypeタイプ演算 2817 命令テンプレートが存在するように図示されている。2) メモリアクセス 2820 命令テンプレート内に、メモリアクセス、書き込みマスク制御 2827 命令テンプレートが存在するように図示されている。

【0221】

汎用ベクトル向け命令フォーマット 2800 は、以下に挙げられるフィールドを図28Aおよび図28B中に図示される順序で含む。

【0222】

フォーマットフィールド 2840。このフィールド内の特定値 (命令フォーマット識別子の値) は、ベクトル向け命令フォーマットを一意に識別し、故に命令ストリームのベクトル向け命令フォーマットの命令の出現を一意に識別する。よって、このフィールドは、汎用ベクトル向け命令フォーマットのみを有する命令セットには不要であるという意味において任意的である。

【0223】

ベース演算フィールド 2842。その内容が、異なるベース演算を区別する。

【0224】

レジスタインデックスフィールド 2844。その内容が、直接的にまたはアドレス生成を介して、ソースオペランドおよびデスティネーションオペランドの位置を指定する。それらはレジスタ内またはメモリ内である。これらは、P x Q (例えば、32 x 512、16 x 128、32 x 1024、64 x 1024) レジスタファイルからN個のレジスタを選択するための十分なビット数を含む。一実施形態において、Nは最大3つのソースレジ

スタおよび1つのデスティネーションレジスタであってよく、一方で、代替的な実施形態は、それより多いまたは少ないソースレジスタおよびデスティネーションレジスタをサポートしてよい（例えば、最大2つのソースをサポートしてよく、この場合、これらのソースのうちの1つがデスティネーションとしても動作する。最大3つのソースをサポートしてよく、この場合、これらのソースのうちの1つがデスティネーションとしても動作する。最大2つのソースおよび1つのデスティネーションをサポートしてよい）。

【0225】

修飾子フィールド2846。その内容が、汎用ベクトル命令フォーマットの、メモリアクセスを指定する命令の出現を、メモリアクセスを指定しないものから区別する。すなわち、メモリアクセスなし2805命令テンプレートおよびメモリアクセス2820命令テンプレート間を区別する。メモリアクセス操作はメモリ階層に対し、読み取りおよび／または書き込みを行う（場合によっては、レジスタ内の値を使用してソースアドレスおよび／またはデスティネーションアドレスを指定する）が、メモリアクセスなし操作はそれを行わない（例えば、ソースおよびデスティネーションはレジスタである）。一実施形態において、このフィールドはまたメモリアドレス計算を実行するための3つの異なる方法の中で選択をする一方で、代替的な実施形態は、メモリアドレス計算を実行するためのより多い、より少ないまたは異なる方法をサポートしてよい。

【0226】

拡張演算フィールド2850。その内容が、ベース演算に加え、様々な異なる演算のうちどれが実行されるべきかを区別する。このフィールドは、コンテキストに特有のものである。本発明の一実施形態において、このフィールドは、クラスフィールド2868、アルファフィールド2852およびベータフィールド2854に分割される。拡張演算フィールド2850は、2、3または4の命令ではなく、単一の命令において共通の演算グループが実行されることを可能にする。

【0227】

スケールフィールド2860。その内容が、メモリアドレス生成（例えば、 $2scale * \text{インデックス} + \text{ベース}$ を使用するアドレス生成について）のためのインデックスフィールドの内容のスケールリングを可能にする。

【0228】

変位フィールド2862A。その内容が、メモリアドレス生成（例えば、 $2scale * \text{インデックス} + \text{ベース} + \text{変位}$ を使用するアドレス生成について）の一部として使用される。

【0229】

変位係数フィールド2862B（変位係数フィールド2862B上に直接、変位フィールド2862Aが並置されていることで、一方または他方が使用されることを示すことに留意されたい）。その内容が、アドレス生成の一部として使用される。その内容は、メモリアクセス（N）のサイズに応じてスケールリングされるべき変位係数を指定する。ここでNは、メモリアクセス（例えば、 $2scale * \text{インデックス} + \text{ベース} + \text{スケールされた変位}$ を使用するアドレス生成について）におけるバイト数である。冗長下位ビットは無視され、従って、変位係数フィールドの内容は、有効アドレスの計算に使用される最終的な変位を生成すべく、メモリオペランドの合計サイズ（N）によって乗算される。Nの値は、フルオペコードフィールド2874（本明細書で後述の）およびデータ操作フィールド2854Cに基づいて、ランタイムでプロセッサハードウェアによって判断される。変位フィールド2862Aおよび変位係数フィールド2862Bは、それらがメモリアクセスなし2805命令テンプレートには使用されない、および／または、異なる実施形態がそれら2つのうちの一方のみを実装してよい、またはいずれも実装しなくてよいという意味において任意的である。

【0230】

データ要素幅フィールド2864。その内容が、複数のデータ要素幅のうちどれが使用されるべきかを区別する（いくつかの実施形態においては、すべての命令に対し、他の実

10

20

30

40

50

施形態においては、命令の一部のみに対し）。1つのデータ要素幅のみがサポートされる、および／または、オペコードのいくつかの態様を使用して複数のデータ要素幅がサポートされる場合、このフィールドは不要であるという意味において、このフィールドは任意的なものである。

【0231】

書き込みマスクフィールド2870。その内容が、データ要素位置単位で、デスティネーションベクトルオペランド内のそのデータ要素位置が、ベース演算および拡張演算の結果を反映するかを制御する。クラスA命令テンプレートは、マージ-書き込みマスクをサポートする一方で、クラスB命令テンプレートは、マージ-書き込みマスクおよびゼロイング-書き込みマスクの両方をサポートする。マージの場合、ベクトルマスクは、任意の演算の実行中、デスティネーション内のあらゆる要素セットが更新されないように保護されることを可能にする（ベース演算および拡張演算によって指定される）。他の一実施形態においては、対応するマスクビットが0を有する場合、デスティネーションの各要素の古い値が保持される。これと対照的に、ゼロイングの場合、ベクトルマスクは、任意の演算の実行中、デスティネーション内のあらゆる要素セットがゼロにされることを可能にする（ベース演算および拡張演算によって指定される）。一実施形態においては、対応するマスクビットが0値を有する場合、デスティネーションの要素は0に設定される。この機能のうちのサブセットで、実行される演算のベクトル長（すなわち、要素のスペンが第1のものから最後のものへと変更される）を制御できる。しかしながら、変更される要素は連続的であることは必要ではない。故に、書き込みマスクフィールド2870は、ロード、ストア、算術、論理等を含む部分的なベクトル演算を可能にする。本発明の実施形態は、書き込みマスクフィールド2870の内容は、複数の書き込みマスクレジスタのうち使用されるべき書き込みマスクを含むものを選択（故に、書き込みマスクフィールド2870の内容は、実行されるべきマスキングを間接的に識別する）するように記載されているものの、代替的な実施形態は、代替的または追加的に、マスク書き込みフィールド2870の内容が、実行されるべきマスキングを直接指定することを可能にする。

【0232】

即値フィールド2872。その内容が、即値の指定を可能にする。このフィールドは即値をサポートしない汎用ベクトル向けフォーマットの実装には存在しない、および、このフィールドは即値を使用しない命令内には存在しないという意味において、このフィールドは、任意的なものである。

【0233】

クラスフィールド2868。その内容が、異なるクラスの命令間を区別する。図28Aおよび図28Bを参照すると、このフィールドの内容で、クラスA命令およびクラスB命令間を選択する。図28Aおよび図28B中、特定値がフィールド内に存在することを示すために、隅が丸められた四角が使用されている（例えば、図28Aおよび図28B中、クラスフィールド2868に対し、それぞれクラスA 2868AおよびクラスB 2868B）。

【0234】

[クラスAの命令テンプレート]

【0235】

クラスAのメモリアクセスなし2805命令テンプレートの場合、アルファフィールド2852はRSフィールド2852Aとして解釈され、RSフィールド2852Aの内容が、異なる拡張演算タイプのうちどれが実行されるべきか（例えば、ラウンド2852A.1およびデータ変換2852A.2がそれぞれ、メモリアクセスなし、ラウンドタイプ演算2810命令テンプレートおよびメモリアクセスなし、データ変換タイプ演算2815命令テンプレートに対し指定される）を区別し、一方で、ベータフィールド2854は指定されるタイプの演算のうちどれが実行されるべきかを区別する。メモリアクセスなし2805命令テンプレートには、スケールフィールド2860、変位フィールド2862Aおよび変位スケールフィールド2862Bは存在しない。

【0236】

[メモリアクセスなし命令テンプレート - 完全ラウンド制御タイプ演算]

【0237】

メモリアクセスなしの完全ラウンド制御タイプ演算2810命令テンプレートでは、ベータフィールド2854はラウンド制御フィールド2854Aとして解釈され、ラウンド制御フィールド2854Aの内容は静的ラウンドを提供する。本発明に記載の実施形態においては、ラウンド制御フィールド2854Aは、すべての浮動小数点の例外を抑制（SAE）フィールド2856およびラウンド演算制御フィールド2858を含み、一方で、代替的な実施形態は、これら両方の概念をサポートしてよく、且つこれら両方の概念を同一フィールドにエンコードしてよく、または代替的な実施形態はこれらの概念／フィールドのうち的一方または他方のみを有してよい（例えば、ラウンド演算制御フィールド2858のみを有してよい）。

10

【0238】

SAEフィールド2856。その内容が、例外イベント報告を無効にするか否かを区別する。SAEフィールド2856の内容が、抑制が有効になっていることを示す場合、特定の命令は、あらゆる種類の浮動小数点例外フラグを報告せず、浮動小数点例外ハンドラを発生させない。

【0239】

ラウンド演算制御フィールド2858。その内容が、ラウンド演算グループ（例えば、切り上げ、切り捨て、ゼロへの丸めおよび最近値への丸め）のうちどれが実行されるかを区別する。故に、ラウンド演算制御フィールド2858は、命令単位で、ラウンドモードの変更を可能にする。本発明の一実施形態において、プロセッサがラウンドモードを指定するための制御レジスタを含む場合、ラウンド演算制御フィールド2850の内容で、そのレジスタ値を上書きする。

20

【0240】

[メモリアクセスなし命令テンプレート - データ変換タイプ演算]

【0241】

メモリアクセスなしのデータ変換タイプ演算2815命令テンプレートでは、ベータフィールド2854はデータ変換フィールド2854Bとして解釈され、データ変換フィールド2854Bの内容が、複数のデータ変換（例えば、データ変換なし、スウィズル、ブロードキャスト）のうちどれが実行されるべきかを区別する。

30

【0242】

クラスAのメモリアクセス2820命令テンプレートの場合、アルファフィールド2852はエビクションヒントフィールド2852Bとして解釈され、エビクションヒントフィールド2852Bの内容が、エビクションヒントのうちどれが使用されるべきかを区別し（図28A中、一時的2852B.1および非一時的2852B.2がそれぞれ、メモリアクセスの一時的2825命令テンプレートおよびメモリアクセスの非一時的2830命令テンプレートに対し指定される）、一方で、ベータフィールド2854はデータ操作フィールド2854Cとして解釈され、データ操作フィールドの内容が、複数のデータ操作演算（プリミティブとしても知られる）のうちどれが実行されるべきかを区別する（例えば、操作なし、ブロードキャスト、ソースのアップコンバージョンおよびデスティネーションのダウンコンバージョン）。メモリアクセス2820命令テンプレートは、スケールフィールド2860を含み、随意で変位フィールド2862Aまたは変位スケールフィールド2862Bを含む。

40

【0243】

ベクトルメモリ命令は、変換サポートを用いて、メモリからのベクトルロードおよびメモリへのベクトルストアを実行する。通常のベクトル命令の場合と同様、ベクトルメモリ命令は、データ要素全体でデータをメモリから／メモリへ転送し、実際に転送される要素は、書き込みマスクとして選択されるベクトルマスクの内容によって記述されている。

【0244】

50

[メモリアクセス命令テンプレート - 一時的]

【0245】

一時的データとは、キャッシュから利益を十分得るために、間もなく再利用される可能性の高いデータのことである。しかしながら、これはヒントであり、異なるプロセッサは、ヒントを完全に無視することを含め、それを異なる方法で実装してよい。

【0246】

[メモリアクセス命令テンプレート - 非一時的]

【0247】

非一時的データとは、第1のレベルのキャッシュにおけるキャッシュから利益を十分得るために、間もなく再利用される可能性の低いデータのことであり、エビクションのための優先度が付与されるべきである。しかしながら、これはヒントであり、異なるプロセッサは、ヒントを完全に無視することを含め、それを異なる方法で実装してよい。

【0248】

[クラスBの命令テンプレート]

【0249】

クラスBの命令テンプレートの場合、アルファフィールド2852は書き込みマスク制御(Z)フィールド2852Cとして解釈され、書き込みマスク制御(Z)フィールド2852Cの内容が、書き込みマスクフィールド2870によって制御される書き込みマスキングが、マージであるべきか、またはゼロイングであるべきかを区別する。

【0250】

クラスBのメモリアクセスなし2805命令テンプレートの場合、ベータフィールド2854の一部はRLフィールド2857Aとして解釈され、RLフィールド2857Aの内容が、異なる拡張演算タイプのうちどれが実行されるべきかを区別し(例えば、ラウンド2857A.1およびベクトル長(VSIZ E)2857A.2がそれぞれ、メモリアクセスなし、書き込みマスク制御、部分的ラウンド制御タイプ演算2812命令テンプレートおよびメモリアクセスなし、書き込みマスク制御、VSIZ Eタイプ演算2817命令テンプレートに対し指定される)、一方で、ベータフィールド2854の残部が、指定されるタイプの演算のうちどれが実行されるべきかを区別する。メモリアクセスなし2805命令テンプレートには、スケールフィールド2860、変位フィールド2862Aおよび変位スケールフィールド2862Bが存在しない。

【0251】

メモリアクセスなし、書き込みマスク制御、部分的ラウンド制御タイプ演算2810命令テンプレートでは、ベータフィールド2854の残部はラウンド演算フィールド2859Aとして解釈され、例外イベント報告が無効にされる(特定の命令は、あらゆる種類の浮動小数点例外フラグを報告せず、浮動小数点例外ハンドラを発生させない)。

【0252】

ラウンド演算制御フィールド2859A。まさにラウンド演算制御フィールド2858と同様、その内容が、ラウンド演算グループ(例えば、切り上げ、切り捨て、ゼロへの丸めおよび最近値への丸め)のうちどれが実行されるかを区別する。故に、ラウンド演算制御フィールド2859Aは、命令単位で、ラウンドモードの変更を可能にする。本発明の一実施形態において、プロセッサがラウンドモードを指定するための制御レジスタを含む場合、ラウンド演算制御フィールド2850の内容で、そのレジスタ値を上書きする。

【0253】

メモリアクセスなし、書き込みマスク制御、VSIZ Eタイプ演算2817命令テンプレートでは、ベータフィールド2854の残部はベクトル長フィールド2859Bとして解釈され、ベクトル長フィールド2859Bの内容が、複数のデータベクトル長のうちのどれ(例えば、128、256または512バイト)に実行されるべきかを区別する。

【0254】

クラスBのメモリアクセス2820命令テンプレートの場合、ベータフィールド2854の一部はブロードキャストフィールド2857Bとして解釈され、ブロードキャストフ

10

20

30

40

50

フィールド 2857B の内容が、ブロードキャストタイプのデータ操作演算が実行されるかを否かを区別し、一方で、ベータフィールド 2854 の残部はベクトル長フィールド 2859B として解釈される。メモリアクセス 2820 命令テンプレートは、スケールフィールド 2860 を含み、随意で変位フィールド 2862A または変位スケールフィールド 2862B を含む。

【0255】

汎用ベクトル向け命令フォーマット 2800 に関しては、フルオペコードフィールド 2874 は、フォーマットフィールド 2840、ベース演算フィールド 2842 およびデータ要素幅フィールド 2864 を含むように表示されている。一実施形態は、フルオペコードフィールド 2874 がこれらのフィールドのうちすべてを含むように示されているものの、これらのフィールドのすべてをサポートしない実施形態においては、フルオペコードフィールド 2874 は、これらのフィールドのすべてより少ない数を含む。フルオペコードフィールド 2874 は、オペレーションコード（オペコード）を提供する。

10

【0256】

拡張演算フィールド 2850、データ要素幅フィールド 2864 および書き込みマスクフィールド 2870 は、汎用ベクトル向け命令フォーマット内でこれらの機能が、命令単位で指定されることを可能にする。

【0257】

書き込みマスクフィールドおよびデータ要素幅フィールドの組み合わせで、異なるデータ要素幅に基づいてマスクが適用されることを可能にするタイプの命令を作成する。

20

【0258】

クラス A およびクラス B 内に存在する様々な命令テンプレートは、異なる状況において有益である。本発明のいくつかの実施形態において、あるプロセッサ内の異なる複数のプロセッサまたは異なるコアが、クラス A のみ、クラス B のみ、またはこれら両方のクラスをサポートしてよい。例えば、汎用コンピューティング向け高性能な汎用アウトオブオーダーコアはクラス B のみをサポートしてよく、主にグラフィックおよび／または科学技術（スループット）コンピューティング向けのコアはクラス A のみをサポートしてよく、これら両方向けのコアは両方をサポートしてよい（もちろん、両方のクラスのテンプレートおよび命令がいくつか混在したものを有するが、両方のクラスのすべてのテンプレートおよび命令を有さないコアは、本発明の範囲に属する）。また、単一のプロセッサが複数のコアを含んでよく、それらのすべてが同一クラスをサポートし、またはそれらのうち異なるコアが異なるクラスをサポートする。例えば、別個のグラフィックコアおよび汎用コアを備えるプロセッサでは、主にグラフィックおよび／または科学技術コンピューティング向けのグラフィックコアのうちの 1 つはクラス A のみをサポートしてよく、一方で、汎用コアのうちの 1 または複数のは、クラス B のみをサポートする汎用コンピューティング向けのアウトオブオーダー実行およびレジスタリネーミングを備えた高性能な汎用コアであってよい。別個のグラフィックコアを有さない別のプロセッサは、クラス A およびクラス B の両方をサポートする 1 または複数の汎用インオーダーまたはアウトオブオーダーコアを含んでよい。もちろん、本発明の異なる実施形態において、一方のクラスに属する諸機能が、他方のクラスに実装されてもよい。高水準言語で記述されるプログラムは、様々な異なる実行可能な形式になされるであろう（例えば、ジャストインタイムコンパイルまたは静的コンパイル）。それらの形式としては、1）実行のためにターゲットプロセッサによってサポートされるクラスの命令のみを有する形式、または 2）すべてのクラスの命令の異なる組み合わせを使用して記述された代替的なルーチンを有し且つ現在コードを実行中のプロセッサによってサポートされる命令に基づき、実行するルーチンを選択する制御フローコードを有する形式が含まれる。

30

40

【0259】

[例示的な特定ベクトル向け命令フォーマット]

【0260】

図 29 は、本発明の実施形態による、例示的な特定ベクトル向け命令フォーマットを示

50

すブロック図である。図29は特定ベクトル向け命令フォーマット2900を示す。特定ベクトル向け命令フォーマット2900は、場所、サイズ、解釈およびフィールド順序に加え、これらのフィールドの一部の値を指定するという意味において特定の。特定ベクトル向け命令フォーマット2900は、x86命令セットを拡張するために使用されてよく、よって、当該フィールドのうちのいくつかは、既存のx86命令セットおよびその拡張（例えば、AVX）で使用されるフィールドと類似または同一である。このフォーマットは、いくつかの拡張を備えた既存のx86命令セットのプレフィクスエンコーディングフィールド、リアルオペコードバイトフィールド、MOD R/Mフィールド、SIBフィールド、変位フィールドおよび即値フィールドと、整合性が維持されている。図28のフィールドが図29のどのフィールドにマッピングされるかが図示されている。

10

【0261】

本発明の実施形態は、例示目的で、汎用ベクトル向け命令フォーマット2800に照らし特定ベクトル向け命令フォーマット2900に関し説明されているものの、本発明は特許請求される場合を除き、特定ベクトル向け命令フォーマット2900には限定されないことを理解されたい。例えば、特定ベクトル向け命令フォーマット2900は特定のサイズのフィールドを有するように図示されているものの、汎用ベクトル向け命令フォーマット2800は、様々なフィールドについて様々な考え得るサイズを想定している。特定の例示であるが、データ要素幅フィールド2864は、特定ベクトル向け命令フォーマット2900では1ビットフィールドとして図示されているものの、本発明はそのようには限定されない（すなわち、汎用ベクトル向け命令フォーマット2800は、データ要素幅フィールド2864の他のサイズを想定している）。

20

【0262】

特定ベクトル向け命令フォーマット2900は、以下に挙げられるフィールドを図29Aに図示される順序で含む。

【0263】

EVEXPレフィクス（バイト0～3）2902。これは4バイト形式でエンコードされる。

【0264】

フォーマットフィールド2840（EVEXPバイト0、ビット[7:0]）。第1のバイト（EVEXPバイト0）はフォーマットフィールド2840であり、フォーマットフィールド2840は0x62を含む（本発明の一実施形態において、ベクトル向け命令フォーマットを区別するために使用される一意の値）。

30

【0265】

第2から第4のバイト（EVEXPバイト1～3）は、特定の機能を提供する複数のビットフィールドを含む。

【0266】

REXフィールド2905（EVEXPバイト1、ビット[7:5]）。これはEVEXP.Rビットフィールド（EVEXPバイト1、ビット[7] - R）、EVEXP.Xビットフィールド（EVEXPバイト1、ビット[6] - X）および2857BEXバイト1、ビット[5] - B）から成る。EVEXP.Rビットフィールド、EVEXP.XビットフィールドおよびEVEXP.Bビットフィールドは、対応するVEXビットフィールドと同一の機能を提供し、それらは1の補数形式を使用してエンコードされ、すなわちZMM0は11111Bとしてエンコードされ、ZMM15は0000Bとしてエンコードされる。命令の他のフィールドは、レジスタインデックスの下位3ビットを当該技術分野で既知の方法（rrr、xxxおよびbbb）でエンコードし、その結果、Rrrr、XxxxおよびBbbbが、EVEXP.R、EVEXP.XおよびEVEXP.Bを追加することによって形成されてよい。

40

【0267】

REX'フィールド2910。これはREX'フィールド2910の第1の部分であり、拡張32レジスタセットの上位16または下位16のいずれかをエンコードするために使

50

用されるEVEX. R'ビットフィールド(EVEXバイト1、ビット[4] - R')である。本発明の一実施形態において、以下に示される他のものと共にこのビットは、ビット反転フォーマットで格納され、BOUND命令から区別(周知のx86の32ビットモードで)される。BOUND命令のリアルオペコードバイトは62であるが、MOD R/Mフィールド(後述)内では、MODフィールドの値11を受け付けない。本発明の代替的な実施形態は、このビットおよび後述される他のビットを反転フォーマットで格納しない。値1が使用され、下位16個のレジスタをエンコードする。換言すると、EVEX. R'、EVEX. Rおよび他のフィールドの他のRRRを組み合わせて、R'Rrrrrが形成される。

【0268】

オペコードマップフィールド2915(EVEXバイト1、ビット[3:0] - mmm)。その内容が暗示される先頭オペコードバイト(0F、0F38、または0F3)をエンコードする。

【0269】

データ要素幅フィールド2864(EVEXバイト2、ビット[7] - W)。これはEVEX. Wという表記で表される。EVEX. Wが使用され、データタイプの粒度(サイズ)を定義する(32ビットデータ要素または64ビットデータ要素のいずれか)。

【0270】

EVEX. vvvv2920(EVEXバイト2、ビット[6:3] - vvvv)。EVEX. vvvvの役割は以下を含んでよい。1) EVEX. vvvvは第1のソースレジスタオペランドを指定された反転(1の補数)形式にエンコードし、EVEX. vvvvは2以上のソースオペランドを持つ命令に対し有効である。2) EVEX. vvvvはデスティネーションレジスタオペランドを、指定された特定のベクトルシフト用の1の補数形式にエンコードする。または3) EVEX. vvvvはいずれのオペランドもエンコードせず、当該フィールドは予約され、1111bを含むべきである。故に、EVEX. vvvvフィールド2920は、反転(1の補数)形式で格納された第1のソースレジスタ指定子の4つの下位ビットをエンコードする。命令に応じて、追加の異なるEVEXビットフィールドが使用され、指定子サイズを32個のレジスタに拡張する。

【0271】

EVEX. U 2868クラスフィールド(EVEXバイト2、ビット[2] - U)。EVEX. U=0の場合、それはクラスAまたはEVEX. U0を示す。EVEX. U=1の場合、それはクラスBまたはEVEX. U1を示す。

【0272】

プレフィクスエンコーディングフィールド2925(EVEXバイト2、ビット[1:0] - pp)。これは、ベース演算フィールドの追加のビットを提供する。EVEXプレフィクスフォーマットにおけるレガシSSE命令のサポートの提供に加え、これはまた、SIMDプレフィクスのコンパクト化の利点を有する(SIMDプレフィクスを表わすために1バイトを要求する代わりに、EVEXプレフィクスは2ビットのみを要求する)。一実施形態において、レガシフォーマットおよびEVEXプレフィクスフォーマットの両方において、SIMDプレフィクス(66H、F2H、F3H)を使用するレガシSSE命令をサポートすべく、これらのレガシSIMDプレフィクスは、SIMDプレフィクスエンコーディングフィールドにエンコードされる。これらのレガシSIMDプレフィクスは、デコーダのPLAに提供される前に、ランタイムにレガシSIMDプレフィクスに拡張される(よって、PLAは、変更なしで、これらのレガシ命令のレガシフォーマットおよびEVEXフォーマットの両方を実行可能である)。より新しい命令はEVEXプレフィクスエンコーディングフィールドの内容を直接オペコード拡張として使用できるものの、特定の実施形態は、整合性のために同様の方法で拡張するが、これらのレガシSIMDプレフィクスによって指定される異なる手段を可能にする。代替的な実施形態は、2ビットSIMDプレフィクスエンコードをサポートするように、つまり拡張を要求しないように、PLAを再設計してよい。

10

20

30

40

50

【0273】

アルファフィールド2852 (EVEXバイト3、ビット[7] - EH。これはEVEX. EH、EVEX. rs、EVEX. RL、EVEX. 書き込みマスク制御およびEVEX. Nとしても知られる。またαを用いて図示)。上記の通り、このフィールドはコンテキストに特有のものである。

【0274】

ベータフィールド2854 (EVEXバイト3、ビット[6:4] - SSS。これはEVEX. s2-0、EVEX. r2-0、EVEX. r r 1、EVEX. LL0、EVEX. LLBとしても知られる。またβββを用いて図示)。上記の通り、このフィールドはコンテキストに特有のものである。

10

【0275】

REX'フィールド2910。これはREX'フィールドの残部であり、REX'フィールド2910は、拡張された32個のレジスタセットの上位16個または下位16個のいずれかをエンコードするために使用され得るEVEX. V'ビットフィールド (EVEXバイト3、ビット[3] - V') である。このビットは、ビット反転フォーマットで格納される。値1が使用され、下位16個のレジスタをエンコードする。換言すると、EVEX. V'、EVEX. v v v vを組み合わせてることにより、V'VVVVが形成される。

【0276】

書き込みマスクフィールド2870 (EVEXバイト3、ビット[2:0] - kkk)。上記の通り、その内容が書き込みマスクレジスタ内のレジスタのインデックスを指定する。本発明の一実施形態において、特定値EVEX. kkk=000は、特定の命令について書き込みマスクが使用されないことを暗示する特別な動作を有する (これは、すべて1にハードワイヤードされた書き込みマスクの使用またはマスキングハードウェアを迂回するハードウェアの使用を含む、様々な方法で実装されてよい)。

20

【0277】

リアルオペコードフィールド2930 (バイト4) は、オペコードバイトとしても知られる。このフィールドで、オペコードの一部が指定される。

【0278】

MOD R/Mフィールド2940 (バイト5) は、MODフィールド2942、Regフィールド2944およびR/Mフィールド2946を含む。上記の通り、MODフィールド2942の内容が、メモリアクセス操作およびメモリアクセスなし操作間を区別する。Regフィールド2944の役割は、デスティネーションレジスタオペランド若しくはソースレジスタオペランドのいずれかをエンコードすること、または、オペコード拡張として扱われ、命令オペランドをエンコードするために使用されないこと、という2つの状況に要約できる。R/Mフィールド2946の役割としては、メモリアドレスを参照する命令オペランドをエンコードすること、またはデスティネーションレジスタオペランド若しくはソースレジスタオペランドのいずれかをエンコードすることが含まれてよい。

30

【0279】

スケール、インデックス、ベース (SIB) バイト (バイト6)。上記の通り、スケールフィールド2850の内容は、メモリアドレス生成に使用される。SIB. xxx2954およびSIB. bbb 2956。これらのフィールドの内容は、レジスタインデックスXxx xおよびBbbbに関して記載済みである。

40

【0280】

変位フィールド2862A (バイト7 - 10)。MODフィールド2942に10が含まれる場合、バイト7 - 10は変位フィールド2862Aであり、変位フィールド2862Aはレガシ32 - ビット変位 (disp32) と同様に動作し、バイト粒度で動作する。

【0281】

変位係数フィールド2862B (バイト7)。MODフィールド2942に01が含まれる場合、バイト7は変位係数フィールド2862Bである。このフィールドの場所は、

50

レガシx86命令セットの8ビット変位(disp8)の場所と同一であり、レガシx86命令セットの8ビット変位(disp8)はバイト粒度で動作する。disp8は符号拡張されるので、disp8は-128~127バイトオフセット間のアドレス指定のみ可能である。64バイトのキャッシュラインに関しては、disp8は4つの実際に有用な値、-128、-64、0および64のみに設定可能な8ビットを使用する。通常、さらに広い範囲が必要であるので、disp32が使用されるが、disp32は4バイトを必要とする。disp8およびdisp32と対照的に、変位係数フィールド2862Bはdisp8と再解釈される。変位係数フィールド2862Bを使用する場合、実際の変位は、メモリオペランドアクセス(N)のサイズで乗算された変位係数フィールドの内容によって決定される。このタイプの変位は、disp8×Nと称される。これは、平均的な命令の長さ(変位に使用されるのは1バイトであるが、はるかにより広い範囲を備える)を低減する。このような圧縮された変位は、有効な変位は、メモリアクセスの粒度の倍数であり、従って、アドレスオフセットの冗長下位ビットはエンコードの必要がないという前提に基づいている。換言すると、変位係数フィールド2862Bは、レガシx86命令セットの8ビット変位に置き換わる。故に、変位係数フィールド2862Bは、disp8がdisp8×Nにオーバーロードされる点のみを除いては、x86命令セットの8ビット変位と同じ方法でエンコードされる(よって、ModRM/SIBエンコードルールの変更はない)。換言すると、ハードウェアによる変位値の解釈のみを除き、エンコードルールまたはエンコーディング長に変更はない(バイト単位のアドレスオフセットを取得するために、メモリオペランドのサイズに応じて変位をスケーリングする必要がある)。

10

20

【0282】

即値フィールド2872は、上記の通り動作する。

【0283】

[フルオペコードフィールド]

【0284】

図29Bは、本発明の一実施形態による、特定ベクトル向け命令フォーマット2900のフルオペコードフィールド2874を構成するフィールドを示すブロック図である。具体的には、フルオペコードフィールド2874は、フォーマットフィールド2840、ベース演算フィールド2842およびデータ要素幅(W)フィールド2864を含む。ベース演算フィールド2842は、プレフィクスエンコーディングフィールド2925、オペコードマップフィールド2915およびリアルオペコードフィールド2930を含む。

30

【0285】

[レジスタインデックスフィールド]

【0286】

図29Cは、本発明の一実施形態による、特定ベクトル向け命令フォーマット2900のレジスタインデックスフィールド2844を構成するフィールドを示すブロック図である。具体的には、レジスタインデックスフィールド2844は、REXフィールド2905、REX'フィールド2910、MODR/M.regフィールド2944、MODR/M.r/mフィールド2946、VVVVフィールド2920、xxxフィールド2954およびbbbフィールド2956を含む。

40

【0287】

[拡張演算フィールド]

【0288】

図29Dは、本発明の一実施形態による、特定ベクトル向け命令フォーマット2900の拡張演算フィールド2850を構成するフィールドを示すブロック図である。クラス(U)フィールド2868が0を含む場合、それはEVEX.U0(クラスA 2868A)を表わす。クラス(U)フィールド2868が1を含む場合、それはEVEX.U1(クラスB 2868B)を表わす。U=0で且つMODフィールド2942が11を含む場合(メモリアクセスなし操作を意味)、アルファフィールド2852(EVEXバイト

50

3、ビット [7] - EH) は、rs フィールド 2852A として解釈される。rs フィールド 2852A が 1 を含む場合 (ラウンド 2852A. 1)、ベータフィールド 2854 (EVEX バイト 3、ビット [6:4] - SSS) はラウンド制御フィールド 2854A として解釈される。ラウンド制御フィールド 2854A は、1 ビットの SAE フィールド 2856 および 2 ビットのラウンド演算フィールド 2858 を含む。rs フィールド 2852A が 0 を含む場合 (データ変換 2852A. 2)、ベータフィールド 2854 (EVEX バイト 3、ビット [6:4] - SSS) は 3 ビットのデータ変換フィールド 2854B として解釈される。U=0 で且つ MOD フィールド 2942 が 00、01 または 10 を含む場合 (メモリアクセス操作を意味)、アルファフィールド 2852 (EVEX バイト 3、ビット [7] - EH) は、エビクションヒント (EH) フィールド 2852B として解釈され、ベータフィールド 2854 (EVEX バイト 3、ビット [6:4] - SSS) は 3 ビットのデータ操作フィールド 2854C として解釈される。

10

【0289】

U=1 の場合、アルファフィールド 2852 (EVEX バイト 3、ビット [7] - EH) は、書き込みマスク制御 (Z) フィールド 2852C として解釈される。U=1 で且つ MOD フィールド 2942 が 11 を含む場合 (メモリアクセスなし操作を意味)、ベータフィールド 2854 の一部 (EVEX バイト 3、ビット [4] - S0) は、RL フィールド 2857A として解釈される。RL フィールド 2857A が 1 を含む場合 (ラウンド 2857A. 1)、ベータフィールド 2854 の残部 (EVEX バイト 3、ビット [6-5] - S2-1) はラウンド演算フィールド 2859A として解釈され、一方で、RL フィールド 2857A が 0 を含む場合 (VSIZ 2857. A2)、ベータフィールド 2854 の残部 (EVEX バイト 3、ビット [6-5] - S2-1) は、ベクトル長フィールド 2859B (EVEX バイト 3、ビット [6-5] - L1-0) として解釈される。U=1 で且つ MOD フィールド 2942 が 00、01 または 10 を含む場合 (メモリアクセス操作を意味)、ベータフィールド 2854 (EVEX バイト 3、ビット [6:4] - SSS) は、ベクトル長フィールド 2859B (EVEX バイト 3、ビット [6-5] - L1-0) およびブロードキャストフィールド 2857B (EVEX バイト 3、ビット [4] - B) として解釈される。

20

【0290】

[例示的なレジスタアーキテクチャ]

30

【0291】

図 30 は、本発明の一実施形態による、レジスタアーキテクチャ 3000 のブロック図である。図示される実施形態には、512 ビット幅の 32 個のベクトルレジスタ 3010 がある。これらのレジスタは、zmm0 から zmm31 と参照符号が付されている。下位 16 個の zmm レジスタの下位 256 ビットは、レジスタ ymm0 ~ ymm16 に重なっている。下位 16 個の zmm レジスタの下位 128 ビット (ymm レジスタの下位 128 ビット) は、レジスタ xmm0 ~ xmm15 に重なっている。特定ベクトル向け命令フォーマット 2900 は、これらの重なったレジスタファイルに対し、以下の表に示されるように動作する。

【表 1】

調整可能ベクトル長	クラス	演算	レジスタ
ベクトル長フィールド2859Bを含まない命令テンプレート	A(図28A; U=0)	2810, 2815, 2825, 2830	zmmレジスタ (ベクトル長は64バイト)
	B(図28B; U=1)	2812	zmmレジスタ (ベクトル長は64バイト)
ベクトル長フィールド2859Bを含む命令テンプレート	B(図28B; U=1)	2817, 2827	ベクトル長フィールド2859Bに応じて、zmm、ymmまたはxmmレジスタ(ベクトル長は64バイト、32バイトまたは16バイト)

10

【0292】

換言すると、ベクトル長フィールド2859Bは、最大長から1または複数の他のより短い長さまでの範囲内から選択する。ここで、当該より短い長さの各々は、1つ前の長さの半分であり、ベクトル長フィールド2859Bを持たない命令テンプレートは、最大ベクトル長に対し動作する。さらに、一実施形態において、特定ベクトル向け命令フォーマット2900のクラスB命令テンプレートは、パックド単精度／倍精度浮動小数点データまたはスカラ単精度／倍精度浮動小数点データおよびパックド整数データまたはスカラ整数データに対し、動作する。スカラ演算とは、zmm／ymm／xmmレジスタ内の最下位のデータ要素の位置で実行される演算である。実施形態に応じ、より上位のデータ要素の位置は、命令前と同じに保持されるか、ゼロにされるかのいずれかである。

20

【0293】

図示された実施形態中の書き込みマスクレジスタ3015には、8個の書き込みマスクレジスタ(k0からk7)が存在し、各々64ビットのサイズである。代替的な実施形態において、書き込みマスクレジスタ3015は、16ビットのサイズである。上記の通り、本発明の一実施形態において、ベクトルマスクレジスタk0は書き込みマスクとして使用不可である。通常k0を示すエンコーディングが書き込みマスクに使用される場合、それは0xFFFFのハードワイヤードされた書き込みマスクを選択し、有効にその命令に対し書き込みマスクングを無効にする。

30

【0294】

図示された実施形態中の汎用レジスタ3025には、メモリオペランドをアドレス指定するために既存のx86アドレス指定モードと共に使用される16個の64ビットの汎用レジスタが存在する。これらのレジスタは、RAX、RBX、RCX、RDX、RBP、RSI、RDI、RSPおよびR8～R15という名称で参照される。

40

【0295】

図示された実施形態中、スカラ浮動小数点スタックレジスタファイル(x87スタック)3045について、MMXパックド整数フラットレジスタファイル3050というエイリアスが示されているが、x87スタックは、x87命令セット拡張を使用して、32／64／80ビットの浮動小数点データにスカラ浮動小数点演算を実行するために使用される8個の要素のスタックである。MMXレジスタは、64ビットのパックド整数データに対し演算を実行するために使用されるが、MMXレジスタおよびXMMレジスタ間で実行されるいくつかの演算のためのオペランドを保持するためにも使用される。

【0296】

本発明の代替的な実施形態は、より範囲の広いまたは狭いレジスタを使用してよい。ま

50

た、本発明の代替的な実施形態は、より多い、より少ないまたは異なるレジスタファイルおよびレジスタを使用してもよい。

【0297】

[例示的なコアアーキテクチャ、プロセッサおよびコンピュータアーキテクチャ]

【0298】

プロセッサコアは、異なる方法で、異なる目的のために、および異なるプロセッサ内に実装されてよい。例えば、このようなコアの実装としては次のようなものが含まれてよい。すなわち、1) 汎用コンピューティング用の汎用インオーダーコアインオーダーコア、2) 汎用コンピューティング用の高性能汎用アウトオブオーダーコア、3) 主にグラフィックおよび／または科学技術（スループット）コンピューティング用の専用コア。異なるプロセッサの実装としては、次のようなものが含まれてよい。すなわち、1) 汎用コンピューティング用の1または複数の汎用インオーダーコアおよび／または汎用コンピューティング用の1または複数の汎用アウトオブオーダーコアを含むCPU、および2) 主にグラフィックおよび／または科学技術（スループット）用の1または複数の専用コアを含むコプロセッサ。このような異なるプロセッサは、異なるコンピュータシステムアーキテクチャをもたらし、それには次のようなものが含まれてよい。すなわち、1) CPUとは別個のチップ上のコプロセッサ、2) CPUと同一パッケージ内の別個のダイ上にあるコプロセッサ、3) CPUと同一ダイ上のコプロセッサ（この場合、このようなコプロセッサは、統合グラフィックおよび／または科学技術（スループット）ロジック等の専用ロジック、または専用コアと呼ばれることがある）および、4) 同一のダイ上に上記CPU（アプリケーションコアまたはアプリケーションプロセッサと呼ばれることがある）、上記コプロセッサおよび追加の機能を含み得るシステムオンチップ。例示的なコアアーキテクチャが次に記載され、その後例示的なプロセッサおよびコンピュータアーキテクチャが続く。

【0299】

[例示的なコアアーキテクチャ]

【0300】

[インオーダーおよびアウトオブオーダーコアのブロック図]

【0301】

図31Aは、本発明の実施形態による、例示的なインオーダーパイプラインおよび例示的なレジスタリネーミング、アウトオブオーダー発行／実行パイプラインの両方を示すブロック図である。図31Bは、本発明の実施形態による、プロセッサに含まれる、インオーダーアーキテクチャコアに係る例示的な実施形態および例示的なレジスタリネーミング、アウトオブオーダー発行／実行アーキテクチャコアの両方を示すブロック図である。図31A～図31B中の実線ボックスは、インオーダーパイプラインおよびインオーダーコアを示すが、オプションで追加される破線ボックスは、レジスタリネーミング、アウトオブオーダー発行／実行パイプラインおよびコアを示す。インオーダーの態様はアウトオブオーダー態様のサブセットであると想定して、アウトオブオーダー態様について以下記載する。

【0302】

図31A中、プロセッサパイプライン3100は、フェッチステージ3102、長さデコードステージ3104、デコードステージ3106、割り当てステージ3108、リネーミングステージ3110、スケジューリング（ディスパッチまたは発行としても知られる）ステージ3112、レジスタ読み取り／メモリ読み取りステージ3114、実行ステージ3116、ライトバック／メモリ書き込みステージ3118、例外処理ステージ3122およびコミットステージ3124が含まれる。

【0303】

図31Bは、実行エンジンユニット3150に連結されたフロントエンドユニット3130を含むプロセッサコア3190を示し、フロントエンドユニット3130および実行エンジンユニット3150の両方はメモリユニット3170に連結されている。コア3190は縮小命令セットコンピューティング（RISC）コア、複合命令セットコンピューティング（CISC）コア、超長命令語（VLIW）コア、またはハイブリッド若しくは

代替的なコアタイプであってよい。さらなる別のオプションとして、コア3190は、例えば、ネットワークコアまたは通信コア、圧縮エンジン、コプロセッサコア、汎用コンピュータグラフィック処理ユニット（GP GPU）コア、グラフィックコア等のような専用コアであってよい。

【0304】

フロントエンドユニット3130は、命令キャッシュユニット3134に連結された分岐予測ユニット3132を含み、命令キャッシュユニット3134は、命令トランスレーションルックアサイドバッファ（TLB）3136に連結され、TLB3136は命令フェッチユニット3138に連結され、命令フェッチユニット3138はデコードユニット3140に連結される。デコードユニット3140（すなわちデコーダ）は命令をデコードしてよく、および、1または複数のマイクロオペレーション、マイクロコードエントリポイント、マイクロ命令、他の命令または他の制御信号を出力として生成してよく、これらは元の命令からデコードされ、あるいは元の命令を反映し、あるいは元の命令から派生する。デコードユニット3140は、様々な異なるメカニズムを使用して実装されてよい。好適なメカニズムの例としては、限定はされないがルックアップテーブル、ハードウェア実装、プログラマブルロジックアレイ（PLA）、マイクロコードリードオンリメモリ（ROM）等が含まれる。一実施形態において、コア3190は、特定のマクロ命令のためのマイクロコードを格納（例えば、デコードユニット3140内またはフロントエンドユニット3130内部）するマイクロコードROMまたは他の媒体を含む。デコードユニット3140は、実行エンジンユニット3150内のリネーム／アロケータユニット3152に連結される。

【0305】

実行エンジンユニット3150は、リタイアメントユニット3154に連結されたリネーム／アロケータユニット3152および1または複数のスケジューラユニット3156のセットを含む。スケジューラユニット3156は、予約ステーション、中央命令ウィンドウ等を含む、任意の数の異なるスケジューラを表わす。スケジューラユニット3156は物理レジスタファイルユニット3158に連結される。物理レジスタファイルユニット3158の各々は、1または複数の物理レジスタファイルを表わし、それらの異なる1つ1つは、1または複数の異なるデータタイプを格納する。そのようなものとしては、スカラー整数、スカラー浮動小数点、パックド整数、パックド浮動小数点、ベクトル整数、ベクトル浮動小数点、状態（例えば、実行される次の命令のアドレスである命令ポインタ）等が挙げられる。一実施形態において、物理レジスタファイルユニット3158はベクトルレジスタユニット、書き込みマスクレジスタユニットおよびスカラーレジスタユニットを備える。これらのレジスタユニットは、アーキテクチャのベクトルレジスタ、ベクトルマスクレジスタおよび汎用レジスタを提供してよい。レジスタリネーミングおよびアウトオブオーダー実行が実装され得る様々な方法を示すため、物理レジスタファイルユニット3158がリタイアメントユニット3154に重ねられている（例えば、リオーダーバッファおよびリタイアメントレジスタファイルを使用する、将来のファイル、履歴バッファおよびリタイアメントレジスタファイルを使用する、レジスタマップおよびレジスタプールを使用する等）。リタイアメントユニット3154および物理レジスタファイルユニット3158は、実行クラスタ3160に連結される。実行クラスタ3160は、1または複数の実行ユニット3162のセットおよび1または複数のメモリアクセスユニット3164のセットを含む。実行ユニット3162は、様々な演算（例えば、シフト、加算、減算、乗算）を様々なタイプのデータ（例えば、スカラー浮動小数点、パックド整数、パックド浮動小数点、ベクトル整数、ベクトル浮動小数点）に行ってよい。いくつかの実施形態は、特定の関数または関数のセットに専用割り当てられた複数の実行ユニットを含んでよく、一方で、他の実施形態は、1つのみの実行ユニットまたは、それらすべてが全関数を実行する複数の実行ユニットを含んでよい。スケジューラユニット3156、物理レジスタファイルユニット3158および実行クラスタ3160が可能性として複数形で図示されているのは、特定の実施形態が特定のタイプのデータ／演算のために別個のパイプライン（例え

ば、スカラ整数のパイプライン、スカラ浮動小数点／パックド整数／パックド浮動小数点／ベクトル整数／ベクトル浮動小数点のパイプラインおよび／またはメモリアクセスパイプライン。これらの各々は独自のスケジューラユニット、物理レジスタファイルユニット、および／または実行クラスタを有する。別個のメモリアクセスパイプラインの場合、このパイプラインの実行クラスタのみがメモリアクセスユニット 3164 を有する特定の実施形態が実装される）を形成するからである。別個のパイプラインが使用される場合、これらのパイプラインのうちの 1 または複数のアウトオブオーダー発行／実行であってよく、残りはインオーダーであってよいことも理解されたい。

【0306】

メモリアクセスユニット 3164 のセットがメモリユニット 3170 に連結され、メモリユニット 3170 はレベル 2 (L2) キャッシュユニット 3176 に連結されたデータキャッシュユニット 3174 に連結されたデータ TLB ユニット 3172 を含む。一例示的な実施形態において、メモリアクセスユニット 3164 は、ロードユニット、ストアアドレスユニット、およびストアデータユニットを含んでよく、これらの各々はメモリユニット 3170 内のデータ TLB ユニット 3172 に連結される。命令キャッシュユニット 3134 は、メモリユニット 3170 内のレベル 2 (L2) キャッシュユニット 3176 にさらに連結される。L2 キャッシュユニット 3176 は、1 または複数の他のレベルのキャッシュに連結され、最終的にメインメモリに連結される。

【0307】

例を挙げると、例示的なレジスタリネーミング、アウトオブオーダー発行／実行コアアーキテクチャは、パイプライン 3100 を以下のように実装してよい。すなわち、1) 命令フェッチ 3138 がフェッチステージ 3102 および長さデコーディングステージ 3104 を実行する。2) デコードユニット 3140 がデコードステージ 3106 を実行する。3) リネーム／アロケータユニット 3152 が割り当てステージ 3108 およびリネーミングステージ 3110 を実行する。4) スケジューラユニット 3156 がスケジューリングステージ 3112 を実行する。5) 物理レジスタファイルユニット 3158 およびメモリユニット 3170 がレジスタ読み取り／メモリ読み取りステージ 3114 を実行する。実行クラスタ 3160 が実行ステージ 3116 を実行する。6) メモリユニット 3170 および物理レジスタファイルユニット 3158 がライトバック／メモリ書き込みステージ 3118 を実行する。7) 様々なユニットが例外処理ステージ 3122 に関与してよい。8) リタイアメントユニット 3154 および物理レジスタファイルユニット 3158 がコミットステージ 3124 を実行する。

【0308】

コア 3190 は、本明細書に記載の命令を含む、1 または複数の命令セット（例えば、x86 命令セット（より新しいバージョンに追加されたいいくつかの拡張を持つ）、カリフォルニア州サンニールの MIPS Technologies の MIPS 命令セット、カリフォルニア州サンニールの ARM Holdings の ARM 命令セット（NEON 等のオプションの追加拡張を持つ））をサポートしてよい。一実施形態において、コア 3190 は、パックドデータ命令セット拡張（例えば、AVX1、AVX2）をサポートするロジックを含み、それにより、多くのマルチメディアアプリケーションによって使用される演算がパックドデータを使用して実行されることを可能にする。

【0309】

コアは、マルチスレッディング（演算またはスレッドの 2 または 2 より多い並列セットの実行）をサポートしてよく、様々な方法でマルチスレッディングを実行してよいことを理解されたい。そのようなものとしては、時分割マルチスレッディング、同時マルチスレッディング（この場合、単一の物理コアは、物理コアが同時にマルチスレッディングを行うスレッドの各々に対し、論理コアを提供する）、またはこれらの組み合わせ（例えば、時分割フェッチおよび時分割デコード並びにインテル（登録商標）ハイパースレッディング技術等のそれら以降の同時マルチスレッディング）が含まれる。

【0310】

レジスタリネーミングはアウトオブオーダー実行の文脈で説明されているが、レジスタリネーミングはインオーダーアーキテクチャで使用されてよいことを理解されたい。図示されたプロセッサの実施形態はまた、別個の命令キャッシュユニット 3134 およびデータキャッシュユニット 3174 並びに共有 L2 キャッシュユニット 3176 を含むが、代替的な実施形態は、命令およびデータの両方のための例えば、レベル 1 (L1) 内部キャッシュまたは複数のレベルの内部キャッシュのような単一の内部キャッシュを有してよい。いくつかの実施形態において、システムは、内部キャッシュ並びにコアおよび／またはプロセッサの外部にある外部キャッシュの組み合わせを含んでよい。代替的に、すべてのキャッシュは、コアおよび／またはプロセッサの外部にあってよい。

【0311】

10

[特定の例示的なインオーダーコアアーキテクチャ]

【0312】

図 32A～32B は、より具体的な例示のインオーダーコアアーキテクチャのブロック図を示し、コア（同一タイプおよび／または異なるタイプの他のコアを含む）はチップ内のいくつかの論理ブロックの 1 つであろう。その適用に応じ、論理ブロックは、何らかの固有の機能ロジック、メモリ I/O インタフェースおよび他の必要な I/O ロジックを備えた高帯域幅の相互接続ネットワーク（例えば、リングネットワーク）を介して通信する。

【0313】

図 32A は、本発明の実施形態による、オンダイ相互接続ネットワーク 3202 への接続を備え、且つ、レベル 2 (L2) キャッシュ 3204 のローカルサブセットを備えた単一のプロセッサコアのブロック図である。一実施形態において、命令デコーダ 3200 は、パックドデータ命令セット拡張を備えた x86 命令セットをサポートする。L1 キャッシュ 3206 は、キャッシュメモリからスカラユニットおよびベクトルユニットへと読み出す低レイテンシアアクセスを可能にする。一実施形態（設計を簡易化した）において、スカラユニット 3208 およびベクトルユニット 3210 は、別個のレジスタセット（それぞれスカラレジスタ 3212 およびベクトルレジスタ 3214）を使用し、それらの間で転送されたデータはメモリに書き込まれた後、レベル 1 (L1) キャッシュ 3206 からリードバックされる一方で、本発明の代替的な実施形態は、異なるアプローチ（例えば、単一のレジスタセットを使用する、またはデータが書き込みおよびリードバックされることなく、2つのレジスタファイル間で転送されることを可能にする通信パスを含む）を使用してよい。

20

30

【0314】

L2 キャッシュのローカルサブセット 3204 は、1つのプロセッサコアにつき1つのローカルサブセットとして、別個の複数のローカルサブセットに分割されるグローバル L2 キャッシュの一部である。各プロセッサコアは、プロセッサコア自身の L2 キャッシュ 3204 のローカルサブセットへのダイレクトアクセスパスを有する。プロセッサコアによって読み取られたデータは、その L2 キャッシュサブセット 3204 に格納され、当該データは、他のプロセッサコアが、自身のローカル L2 キャッシュサブセットにアクセスするのと並列的に、迅速にアクセス可能である。プロセッサコアによって書き込まれたデータは、自身の L2 キャッシュサブセット 3204 に格納され、必要な場合、他のサブセットからはフラッシュされる。リングネットワークは、共有データのためのコヒーレンスを保証する。リングネットワークは双方向であり、プロセッサコア、L2 キャッシュおよび他の論理ブロック等のエージェントが、チップ内で互いに通信することを可能にする。各リングデータパスは、一方向当たり 1012 ビット幅である。

40

【0315】

図 32B は、本発明の実施形態による、図 32A のプロセッサコアの一部の拡大図である。図 32B には、L1 キャッシュ 3204 の一部である L1 データキャッシュ 3206A に加え、ベクトルユニット 3210 およびベクトルレジスタ 3214 に関しより詳細なものが含まれる。具体的には、ベクトルユニット 3210 は、16 幅ベクトル処理ユニット (VPU) (16 幅 ALU 3228 を参照) であり、整数命令、単精度浮動命令およ

50

び倍精度浮動命令のうちの1または複数を実行する。VPUは、スウィズルユニット3220を用いるレジスタ入力のスウィズル、数値変換ユニット3222A～Bを用いる数値変換およびメモリ入力での複製ユニット3224を用いる複製をサポートする。書き込みマスクレジスタ3226は、結果ベクトル書き込みのプレディケートを可能にする。

【0316】

〔統合メモリコントローラおよびグラフィックを持つプロセッサ〕

【0317】

図33は、本発明の実施形態による、プロセッサ3300のブロック図であり、当該プロセッサは、2以上のコアを有してよく、統合メモリコントローラを有してよく、統合グラフィックを有してよい。図33中の実線ボックスは、単一のコア3302A、システムエージェント3310、1または複数のバスコントローラユニット3316のセットを備えたプロセッサ3300を示す一方で、破線ボックスのオプションの追加は、複数のコア3302A～N、システムエージェントユニット3310内の1または複数の統合メモリコントローラユニット3314、および専用ロジック3308を備えた代替的なプロセッサ3300を示す。

【0318】

故に、プロセッサ3300の異なる実装は、次のもの、すなわち1) 統合グラフィックおよび/または科学技術(スループット)ロジック(1または複数のコアを含んでよい)である専用ロジック3308と、1または複数の汎用コアであるコア3302A～3302N(例えば、汎用インオーダーコア、汎用アウトオブオーダーコア、それら2つの組み合わせ)を有するCPU、2) 主にグラフィックおよび/または科学技術(スループット)向けの多数の専用コアであるコア3302A～3302Nを有するコプロセッサ、並びに3) 多数の汎用インオーダーコアであるコア3302A～3302Nを有するコプロセッサ、を含んでよい。故に、プロセッサ3300は、例えば、ネットワークプロセッサまたは通信プロセッサ、圧縮エンジン、グラフィックプロセッサ、GPGPU(汎用グラフィック処理ユニット)、高スループット多集積コア(MIC)コプロセッサ(30または30より多いコアを含む)、組み込みプロセッサ等のような汎用プロセッサ、コプロセッサ、または専用プロセッサであってよい。プロセッサは、1または複数のチップ上に実装されてよい。プロセッサ3300は、例えば、BiCMOS、CMOSまたはNMOS等の複数のプロセス技術のうちの任意のものを使用する1または複数の基板の一部であってよく、および/または当該基板上に実装されてよい。

【0319】

メモリ階層は、コア内の1または複数のレベルのキャッシュ、共有キャッシュユニット3306のセット若しくは1または複数の共有キャッシュユニット3306、および統合メモリコントローラユニット3314のセットに連結された外部メモリ(不図示)を含む。共有キャッシュユニットのセット3306は、レベル2(L2)、レベル3(L3)、レベル4(L4)等の1または複数の中レベルキャッシュまたは他のレベルのキャッシュ、ラストレベルキャッシュ(LLC)および/またはそれらの組み合わせを含んでよい。一実施形態において、リングベースの相互接続ユニット3312は、統合グラフィックロジック3308、共有キャッシュユニット3306のセットおよびシステムエージェントユニット3310/統合メモリコントローラユニット3314を相互接続する一方で、代替的な実施形態は、このようなユニットを相互接続するための任意の数の周知技術を使用してよい。一実施形態において、コヒーレンスは、1または複数のキャッシュユニット3306およびコア3302A～N間で維持される。

【0320】

いくつかの実施形態において、コア3302A～Nのうちの1または複数は、マルチスレッディングが可能である。システムエージェント3310は、コア3302A～Nを調整および操作するそれらのコンポーネントを含む。システムエージェントユニット3310は、例えば、電力制御ユニット(PCU)およびディスプレイユニットを含んでよい。PCUは、コア3302A～Nおよび統合グラフィックロジック3308の電力状態を統

制するために必要なロジックおよびコンポーネントであってよい、またはそれらを含んでよい。ディスプレイユニットは、1または複数の外部接続されたディスプレイを駆動するためのものである。

【0321】

コア3302A～Nは、アーキテクチャ命令セットの観点から同種または異種であってよい。すなわち、コア3302A～3302Nのうち2または2より多くは、同一命令セットを実行可能であってよいが、他のものはその命令セットのサブセットのみまたは異なる命令セットを実行可能であってよい。

【0322】

[例示的なコンピュータアーキテクチャ]

10

【0323】

図34～図37は、例示的なコンピュータアーキテクチャのブロック図である。ラップトップ、デスクトップ、ハンドヘルドPC、携帯情報端末、エンジニアリングワークステーション、サーバ、ネットワークデバイス、ネットワークハブ、スイッチ、組み込みプロセッサ、デジタル信号プロセッサ(DSP)、グラフィックデバイス、ビデオゲームデバイス、セットトップボックス、マイクロコントローラ、携帯電話、ポータブルメディアプレーヤ、ハンドヘルドデバイスおよび様々な他の電子デバイスのための当該技術分野で既知の他のシステム設計および構成も好適である。一般的に、本明細書に開示のプロセッサおよび／または他の実行ロジックを組み込み可能な非常に多種多様なシステムまたは電子デバイスが概して好適である。

20

【0324】

ここで図34を参照すると、本発明の一実施形態によるシステム3400のブロック図が示されている。システム3400は、1または複数のプロセッサ3410、3415を含んでよく、当該1または複数のプロセッサ3410、3415は、コントローラハブ3420に連結される。一実施形態において、コントローラハブ3420は、グラフィックメモリコントローラハブ(GMCH)3490および入／出力ハブ(IOH)3450(別個のチップ上に存在してよい)を含む。GMCH3490は、メモリ3440およびコプロセッサ3445が連結されたメモリコントローラおよびグラフィックコントローラを含む。IOH3450は、入出力(I/O)デバイス3460をGMCH3490に連結する。代替的に、メモリコントローラおよびグラフィックコントローラ的一方または両方がプロセッサ内に統合され(本明細書に記載の通り)、メモリ3440およびコプロセッサ3445は、プロセッサ3410と、単一のチップ内のIOH3450を持つコントローラハブ3420とに直接連結される。

30

【0325】

図34中、破線を用いて、追加のプロセッサ3415がオプションの性質であることが示されている。各プロセッサ3410、3415は、本明細書に記載の処理コアのうちの1または複数を含んでよく、プロセッサ3300の何らかのバージョンであってよい。

【0326】

メモリ3440は、例えば、ダイナミックランダムアクセス(DRAM)、相変化メモリ(PCM)、またはこれら2つの組み合わせであってよい。少なくとも1つの実施形態について、コントローラハブ3420は、フロントサイドバス(FSB)等のマルチドロップバス、QuickPathインターコネクト(QPI)等のポイントツーポイントインタフェースまたは類似の接続3495を介して、プロセッサ3410、3415と通信する。

40

【0327】

一実施形態において、コプロセッサ3445は、例えば、高スループットMICプロセッサ、ネットワークプロセッサまたは通信プロセッサプロセッサ、圧縮エンジン、グラフィックプロセッサ、GPGPU、組み込みプロセッサ等のような専用プロセッサである。一実施形態において、コントローラハブ3420は統合グラフィックアクセラレータを含んでよい。

50

【0328】

物理リソース3410と3415との間には、アーキテクチャ上、マイクロアーキテクチャ上、熱的、電力消費特性等を含む利点の様々な基準に関して、様々な差異が存在し得る。

【0329】

一実施形態において、プロセッサ3410は、汎用タイプのデータ処理演算を制御する命令を実行する。コプロセッサ命令が命令内に埋め込まれてよい。プロセッサ3410は、これらのコプロセッサ命令を取り付けられたコプロセッサ3445によって実行されるべきタイプのものであると認識する。従って、プロセッサ3410はこれらのコプロセッサ命令（またはコプロセッサ命令を表わす制御信号）を、コプロセッサ3445へのコプロセッサバスまたは他の相互接続上に発行する。コプロセッサ3445はコプロセッサ命令を受け取り、受信されたコプロセッサ命令を実行する。

10

【0330】

ここで図35を参照すると、本発明の一実施形態による第1のより具体的な例示的システム3500のブロック図を示す。図35に図示の通り、マルチプロセッサシステム3500は、ポイントツーポイント相互接続システムであり、ポイントツーポイント相互接続3550を介して連結された第1のプロセッサ3570および第2のプロセッサ3580を含む。プロセッサ3570および3580の各々は、プロセッサ3300の何らかのバージョンであってよい。本発明の一実施形態において、プロセッサ3570および3580は、それぞれプロセッサ3410および3415である一方で、コプロセッサ3538はコプロセッサ3445である。別の実施形態において、プロセッサ3570および3580は、それぞれプロセッサ3410およびコプロセッサ3445である。

20

【0331】

プロセッサ3570および3580は、それぞれ統合メモリコントローラ（IMC）ユニット3572および3582を含むように図示されている。プロセッサ3570はまた、そのバスコントローラユニットの一部として、ポイントツーポイント（P-P）インタフェース3576および3578を含み、同様に第2のプロセッサ3580はP-Pインタフェース3586および3588を含む。プロセッサ3570、3580は、P-Pインタフェース回路3578、3588を使用して、ポイントツーポイント（P-P）インタフェース3550を介して情報を交換してよい。図35に図示の通り、IMC3572および3582はプロセッサをそれぞれのメモリ、すなわちメモリ3532およびメモリ3534に連結する。メモリ3532およびメモリ3534は、それぞれのプロセッサにローカルに取り付けられたメインメモリの一部であってよい。

30

【0332】

プロセッサ3570、3580はそれぞれ、ポイントツーポイントインタフェース回路3576、3594、3586、3598を使用して、個々のP-Pインタフェース3552、3554を介して、チップセット3590と情報を交換してよい。随意で、チップセット3590は、高性能インタフェース3539を介してコプロセッサ3538と情報を交換してよい。一実施形態において、コプロセッサ3538は、例えば、高スループットMICプロセッサ、ネットワークプロセッサまたは通信プロセッサプロセッサ、圧縮エンジン、グラフィックプロセッサ、GPGPU、組み込みプロセッサ等のような専用プロセッサである。

40

【0333】

共有キャッシュ（不図示）が、いずれかのプロセッサの内部または両方のプロセッサの外部に含まれてよく、共有キャッシュはさらに当該プロセッサとP-P相互接続を介して接続されていてよく、その結果、プロセッサが低電力モードの場合、いずれかまたは両方のプロセッサのローカルキャッシュ情報が共有キャッシュ内に格納され得るようになる。

【0334】

チップセット3590が、インタフェース3596を介して第1のバス3516に連結されていてよい。一実施形態において、第1のバス3516はペリフェラルコンポーネントイ

50

ンターコネクト（P C I）バス、すなわち P C I E x p r e s s バス若しくは別の第 3 世代 I / O 相互接続バス等のバスであってよいが、本発明の範囲はそのようには限定されない。

【0335】

図 35 に図示の通り、様々な I / O デバイス 3514 がバスブリッジ 3518 と共に第 1 のバス 3516 に連結されてよく、バスブリッジ 3518 は第 1 のバス 3516 を第 2 のバス 3520 に連結する。一実施形態において、コプロセッサ、高スループット M I C プロセッサ、G P G P U のアクセラレータ（例えば、グラフィックアクセラレータまたはデジタル信号処理（D S P）ユニット等）、フィールドプログラマブルゲートアレイ、または任意の他のプロセッサ等の 1 または複数の追加のプロセッサ 3515 が第 1 のバス 3516 に連結される。一実施形態において、第 2 のバス 3520 はローピンカウント（L P C）バスであってよい。一実施形態において、様々なデバイスが第 2 のバス 3520 に連結されてよく、そのようなものとしては、例えば、キーボードおよび／またはマウス 3522、通信デバイス 3527 および命令／コードおよびデータ 3530 を含み得るディスクドライブまたは他の大容量ストレージデバイス等のストレージユニット 3528 が含まれる。さらに、オーディオ I / O 3524 が第 2 のバス 3520 に連結されてよい。他のアーキテクチャも可能であることに留意されたい。例えば、図 35 のポイントツーポイントアーキテクチャの代わりに、システムはマルチドロップバスまたは他のこのようなアーキテクチャを実装してよい。

【0336】

ここで図 36 を参照すると、本発明の実施形態による、第 2 のより具体的な例示的システム 3600 のブロック図が示されている。図 35 および図 36 中で同様の要素は同様の参照符号が付されており、図 36 の他の態様を不明瞭にするのを回避すべく、図 35 の特定の態様は図 36 で省略されている。

【0337】

図 36 は、プロセッサ 3570、3580 が統合メモリおよび I / O 制御ロジック（「C L」）3572 および 3582 をそれぞれ含んでよいことを示す。故に、C L 3572、3582 は、統合メモリコントローラユニットを含み、I / O 制御ロジックを含む。図 36 は、メモリ 3532、3534 が C L 3572、3582 に連結されるだけでなく、I / O デバイス 3614 はまた制御ロジック 3572、3582 に連結されることも示している。レガシ I / O デバイス 3615 がチップセット 3590 に連結される。

【0338】

ここで図 37 を参照すると、本発明の一実施形態による S o C 3700 のブロック図が示されている。図 33 中と同様の要素は同様の参照番号が付されている。また、破線ボックスは、より高度な S o C 上でのオプションの機能である。図 37 中、相互接続ユニット 3702 は、アプリケーションプロセッサ 3710 と、システムエージェントユニット 3310 と、バスコントローラユニット 3316 と、統合メモリコントローラユニット 3314 と、コプロセッサ 3720 のセットまたは 1 若しくは複数のコプロセッサ 3720 と、スタティックランダムアクセスメモリ（S R A M）ユニット 3730 と、ダイレクトメモリアクセス（D M A）ユニット 3732 と、1 または複数の外部ディスプレイに連結するためのディスプレイユニット 3740 とに連結される。アプリケーションプロセッサ 3710 は、1 または複数のコア 202A～N のセットおよび共有キャッシュユニット 3306 を含む。コプロセッサ 3720 のセットまたは 1 若しくは複数のコプロセッサ 3720 は、統合グラフィックロジック、イメージプロセッサ、オーディオプロセッサおよびビデオプロセッサを含んでよい。一実施形態において、コプロセッサ 3720 は、例えば、ネットワークプロセッサまたは通信プロセッサ、圧縮エンジン、G P G P U、高スループット M I C プロセッサ、組み込みプロセッサ等のような専用プロセッサを含む。

【0339】

本明細書に開示のメカニズムに係る実施形態は、ハードウェア、ソフトウェア、ファームウェアまたはこのような実装アプローチの組み合わせで実装されてよい。本発明の実施

形態は、少なくとも1つのプロセッサ、ストレージシステム（揮発性および不揮発性のメモリ並びに／またはストレージ要素を含む）、少なくとも1つの入力デバイスおよび少なくとも1つの出力デバイスを備えるプログラム可能なシステム上で実行されるコンピュータプログラムまたはプログラムコードとして実装されてよい。

【0340】

図35に図示されたコード3530等のプログラムコードは、本明細書に記載の機能を実行するための命令を入力するため、および出力情報を生成するために適用されてよい。出力情報は、1または複数の出力デバイスに既知の態様で適用されてよい。本願の目的において、処理システムには、例えば、デジタル信号プロセッサ（DSP）、マイクロコントローラ、特定用途向け集積回路（ASIC）、またはマイクロプロセッサ等のプロセッサを有する任意のシステムが含まれる。

10

【0341】

プログラムコードは、処理システムと通信するために、高水準の手順型プログラミング言語またはオブジェクト指向型プログラミング言語で実装されてよい。必要であれば、プログラムコードはまた、アセンブリ言語または機械言語で実装されてもよい。実際、本明細書に記載のメカニズムは、いずれの特定のプログラミング言語にも範囲限定されない。いずれの場合においても、言語はコンパイル型言語または解釈型言語であってよい。

【0342】

少なくとも1つの実施形態に係る1または複数の態様は、機械可読媒体上に格納された、プロセッサ内で様々なロジックを表わす典型的命令によって実装されてよく、当該命令は機械による読み取り時に、機械に対し、本明細書に記載の技術を実行するためのロジックを生成させる。このような「IPコア」として知られる典型的なものが、有形の機械可読媒体上に格納され、様々な顧客または製造施設に供給され、実際にロジックまたはプロセッサを作成する製造機械にロードされてよい。

20

【0343】

このような機械可読記録媒体としては、限定はされないが、機械またはデバイスによって製造または形成される複数の物品から成る非一時的な有形の構成が含まれてよく、それらとしては、ハードディスク、フロッピー（登録商標）ディスク、光ディスク、コンパクトディスクリードオンリメモリ（CD-ROM）、コンパクトディスクリライタブル（CD-RW）、および光磁気ディスクを含む任意の他のタイプのディスク、リードオンリメモリ（ROM）、ダイナミックランダムアクセスメモリ（DRAM）、スタティックランダムアクセスメモリ（SRAM）等のランダムアクセスメモリ（RAM）、消去可能プログラマブルリードオンリメモリ（EPROM）、フラッシュメモリ、電氣的消去可能プログラマブルリードオンリメモリ（EEPROM）、相変化メモリ（PCM）等の半導体デバイス、磁気カード若しくは光カードといった記録媒体または電子的命令を格納するのに好適な任意の他のタイプの媒体が含まれる。

30

【0344】

従って、また、本発明の実施形態は、命令を含む、または本明細書に記載の構造、回路、装置、プロセッサおよび／またはシステム機能を定義するハードウェア記述言語（HDL）等の設計データを含む非一時的な有形の機械可読媒体を含む。また、このような実施形態はプログラム製品としても称されてよい。

40

【0345】

[エミュレーション（バイナリ変換、コードモーフィング等）]

【0346】

いくつかの場合において、命令コンバータが使用され、命令をソース命令セットからターゲット命令セットへと変換してよい。例えば、命令コンバータは、ある命令を、コアによって処理されるべき1または複数の他の命令へと、トランスレート（例えば、静的バイナリ変換、動的コンパイルを含む動的バイナリ変換を使用して）、モーフィング、エミュレート、またはそれら以外の方法による変換を行ってよい。命令コンバータは、ソフトウェア、ハードウェア、ファームウェア、またはこれらの組み合わせで実装されてよい。命

50

令コンバータは、プロセッサ内、プロセッサ外、または部分的にプロセッサ内または部分的にプロセッサ外に存在してよい。

【0347】

図38は、本発明の実施形態による、ソース命令セット内のバイナリ命令をターゲット命令セット内のバイナリ命令に変換するためのソフトウェア命令コンバータの使用を対比するブロック図である。図示された実施形態において、命令コンバータはソフトウェア命令コンバータであるものの、代替的に、命令コンバータはソフトウェア、ファームウェア、ハードウェアまたはこれらの様々な組み合わせで実装されてもよい。図38は、高水準言語3802のプログラムが、x86バイナリコード3806を生成するx86コンパイラ3804を使用してコンパイルされ得ることを示しており、当該x86バイナリコード3806は、少なくとも1つのx86命令セットコアを持つプロセッサ3816によってネイティブに実行されてよい。少なくとも1つのx86命令セットコアを持つプロセッサ3816は、少なくとも1つのx86命令セットコアを持つインテルプロセッサと実質的に同一の諸機能を実行できる任意のプロセッサを表わしており、これは次のように行う。すなわち、少なくとも1つのx86命令セットコアを持つインテルのプロセッサと実質的に同一の結果を得るべく、(1)インテルx86命令セットコアの命令セットの大部分、または(2)少なくとも1つのx86命令セットコアを持つインテルプロセッサ上での実行を目的とするアプリケーションまたは他のソフトウェアのオブジェクトコードバージョン、を互換性のある状態で実行またはそれ以外の方法で処理することによってである。x86コンパイラ3804は、x86バイナリコード3806(例えばオブジェクトコード)を生成するように動作可能なコンパイラを表わし、当該x86バイナリコード3806は、追加のリンク処理と共に、または追加のリンク処理なしに、少なくとも1つのx86命令セットコアを持つプロセッサ3816上で実行可能である。同様に、図38は、高水準言語3802のプログラムが、代替的な命令セットバイナリコード3810を生成する代替的な命令セットコンパイラ3808を使用してコンパイルされ得ることを示しており、当該代替的な命令セットバイナリコード3810は、少なくとも1つのx86命令セットコアを持たないプロセッサ3814(例えば、カリフォルニア州サニーベールのMIPS TechnologiesのMIPS命令セットを実行する、および/または、カリフォルニア州サニーベールのARM HoldingsのARM命令セットを実行するコアを持つプロセッサ)によってネイティブに実行されてよい。命令コンバータ3812は、x86バイナリコード3806を、x86命令セットコアを持たないプロセッサ3814によってネイティブに実行可能なコードに変換されるのに使用される。これが可能な命令コンバータの作成は難しいので、この変換されたコードは、代替的な命令セットバイナリコード3810と同じである可能性は低いが、しかしながら、変換されたコードは、一般的な演算を達成し、代替的な命令セットに属する命令で構成されるであろう。故に、命令コンバータ3812は、ソフトウェア、ファームウェア、ハードウェアまたはこれらの組み合わせを表わし、それらは、エミュレーション、シミュレーションまたは任意の他の処理を介して、x86命令セットプロセッサまたはコアを有さないプロセッサまたは他の電子デバイスが、x86バイナリコード3806を実行できるようにする。

(項目1)

命令をデコードするためのハードウェアデコーダであって、上記命令は、オペコードおよびフォールバックアドレスの一部を格納するためのオペランドを含む、ハードウェアデコーダと、

上記デコードされた命令を実行するための実行ハードウェアであって、上記デコードされた命令は、投機的メモリアクセスを追跡し且つデータ投機的実行(DSX)領域における順序違反を検出するためのDSX追跡ハードウェアをアクティブ化し、且つ、上記フォールバックアドレスを格納することによって、上記DSX領域を開始する、実行ハードウェアと、を備える、装置。

(項目2)

上記フォールバックアドレスの上記一部は、上記実行ハードウェアによって、上記デコ

ードされた命令の直後にある命令の命令ポインタに追加されるべき変位値である、項目 1 に記載の装置。

(項目 3)

上記フォールバックアドレスの上記一部は、完全アドレスである、項目 1 に記載の装置。

(項目 4)

上記フォールバックアドレスの上記一部を格納するための上記オペランドは、即値である、項目 1 に記載の装置。

(項目 5)

上記フォールバックアドレスの上記一部を格納するための上記オペランドは、レジスタである、項目 1 に記載の装置。

(項目 6)

上記実行ハードウェアはさらに、RTM (Restricted Transactional Memory) トランザクションが発生していることを判断し且つ上記 RTM トランザクションを処理するためのものである、項目 1 に記載の装置。

(項目 7)

対応する DSX 領域の終了を備えていない DSX 領域の開始の数に対応する値を格納するための DSX ネストカウンタをさらに備える、項目 1 に記載の装置。

(項目 8)

ハードウェアデコーダを使用して、命令をデコードする段階であって、上記命令はオペコードおよびフォールバックアドレスの一部を格納するためのオペランドを含む、デコードする段階と、

上記デコードされた命令を実行する段階であって、上記デコードされた命令は、投機的メモリアクセスを追跡し且つデータ投機的実行 (DSX) 領域における順序違反を検出するための DSX 追跡ハードウェアをアクティブ化し且つ上記フォールバックアドレスを格納することによって、上記 DSX 領域を開始する、実行する段階と、を備える、方法。

(項目 9)

上記フォールバックアドレスの上記一部は、実行ハードウェアによって、上記デコードされた命令の直後にある命令の命令ポインタに追加されるべき変位値である、項目 8 に記載の方法。

(項目 10)

上記フォールバックアドレスの上記一部は、完全アドレスである、項目 8 に記載の方法。

(項目 11)

上記フォールバックアドレスの上記一部を格納するための上記オペランドは、即値である、項目 8 に記載の方法。

(項目 12)

上記フォールバックアドレスの上記一部を格納するための上記オペランドは、レジスタである、項目 8 に記載の方法。

(項目 13)

上記実行する段階は、RTM (Restricted Transactional Memory) トランザクションが発生していることを判断する段階および上記 RTM トランザクションを処理する段階をさらに含む、項目 8 に記載の方法。

(項目 14)

対応する DSX 領域の終了を備えていない DSX 領域の開始の数に対応する値を格納する段階をさらに備える、項目 8 に記載の方法。

(項目 15)

機械による実行時、回路が生成されるようにする命令を格納した非一時機械可読媒体であって、上記回路は、

10

20

30

40

50

命令をデコードするためのハードウェアデコーダであって、上記命令は、オペコードおよびフォールバックアドレスの一部を格納するためのオペランドを含む、ハードウェアデコーダと、

上記デコードされた命令を実行するための実行ハードウェアであって、上記デコードされた命令は、投機的メモリアクセスを追跡し且つデータ投機的実行（DSX）領域における順序違反を検出するためのDSX追跡ハードウェアをアクティブ化し且つ上記フォールバックアドレスを格納することによって、上記DSX領域を開始する、実行ハードウェアと、を含む、非一時機械可読媒体。

（項目16）

上記フォールバックアドレスの上記一部は、上記実行ハードウェアによって、上記デコードされた命令の直後にある命令の命令ポインタに追加されるべき変位値である、項目15に記載の非一時機械可読媒体。

（項目17）

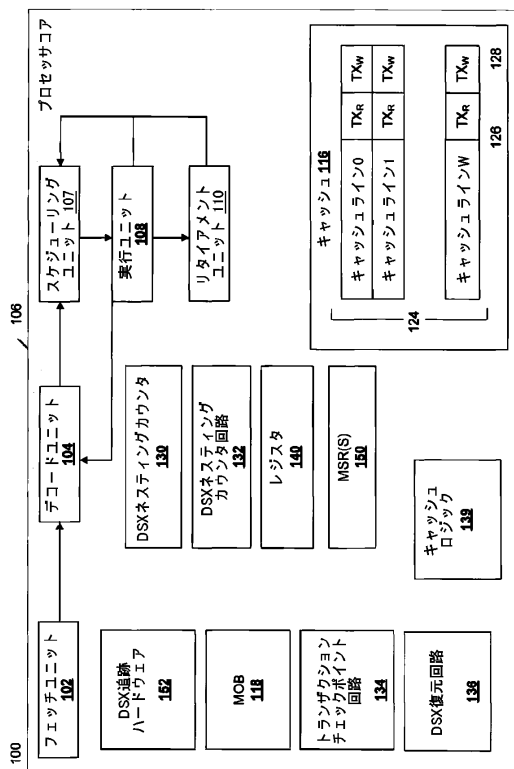
上記フォールバックアドレスの上記一部は、完全アドレスである、項目15に記載の非一時機械可読媒体。

（項目18）

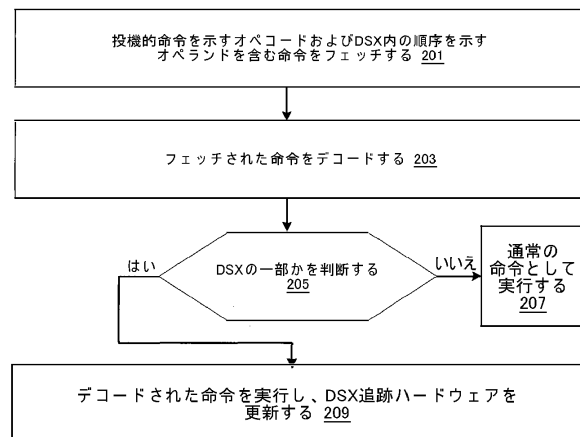
上記フォールバックアドレスの上記一部を格納するための上記オペランドは、即値である、項目15に記載の非一時機械可読媒体。

10

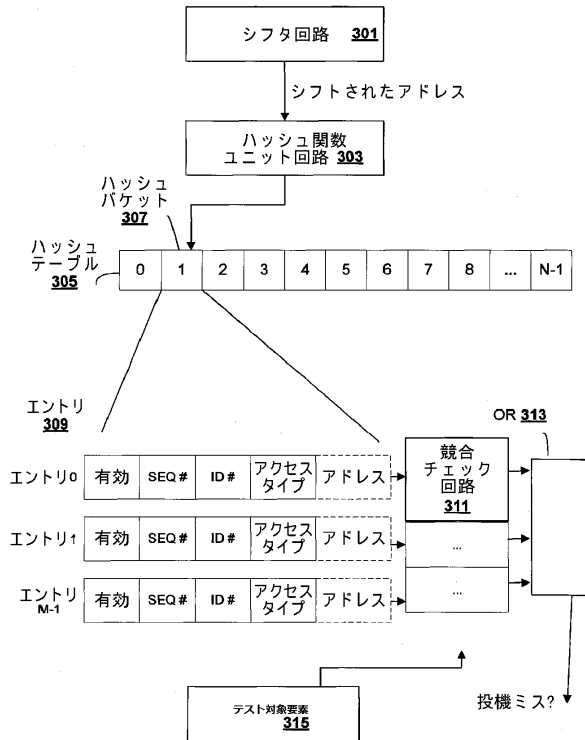
【図1】



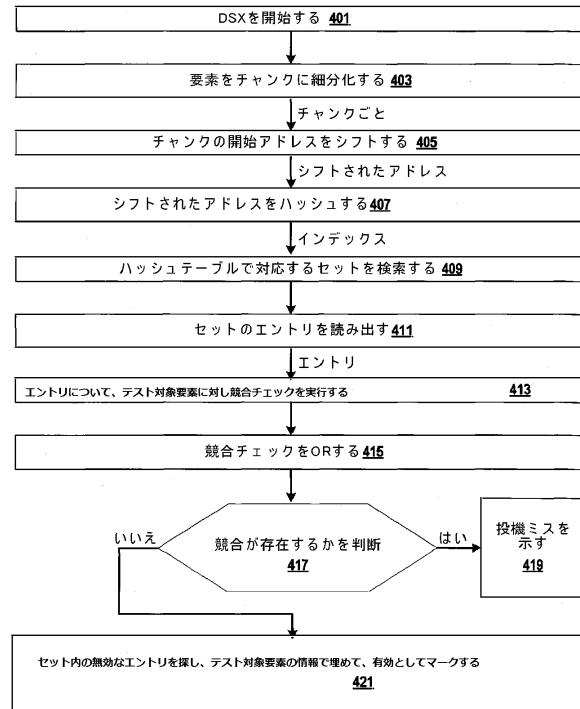
【図2】



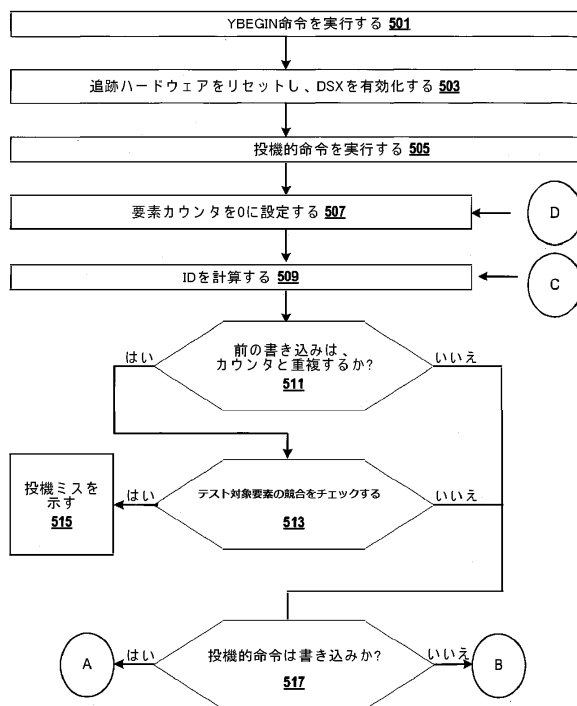
【図 3】



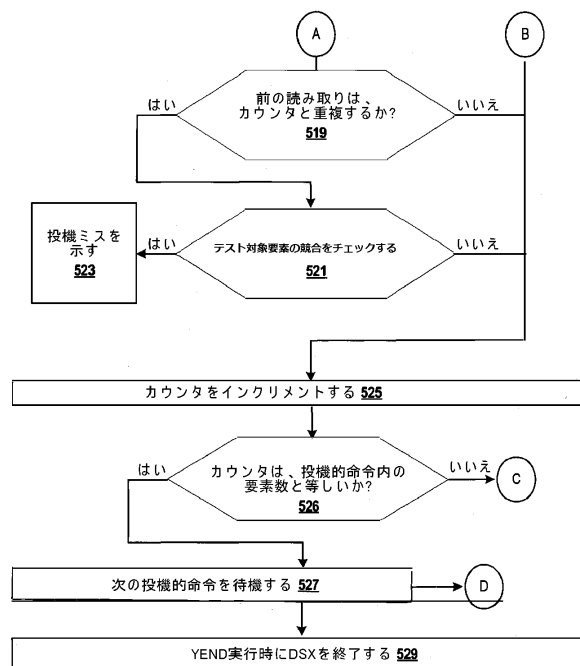
【図 4】



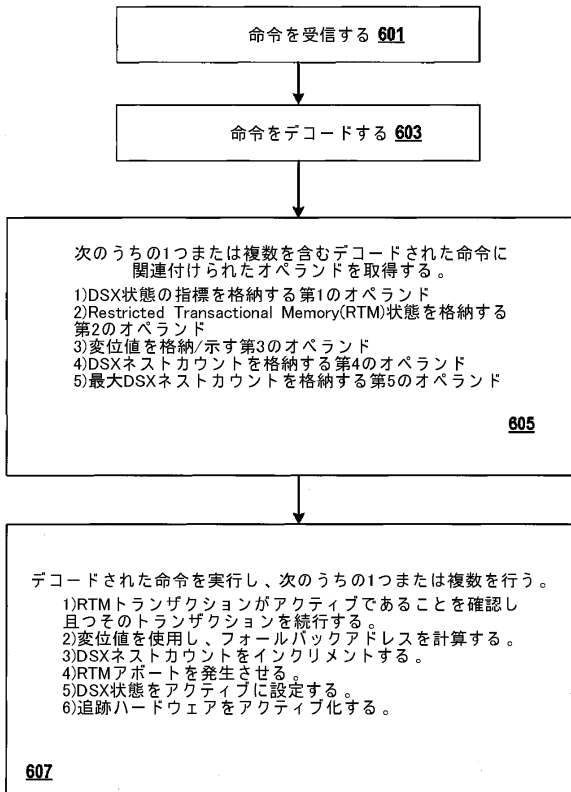
【図 5 A】



【図 5 B】



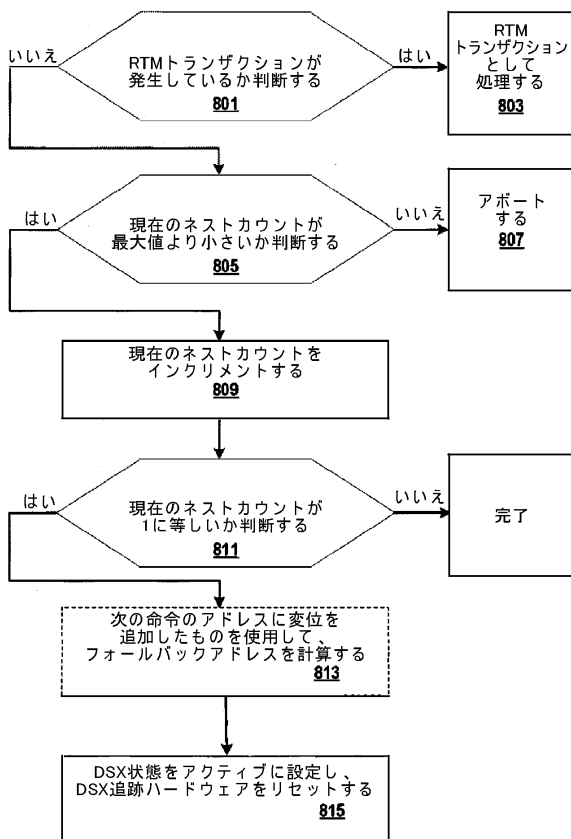
【図 6】



【図 7】

オペコード	明示的オペランド
701 YBEGIN	変位
703 YBEGIN	DSX状態レジスタ、変位
705 YBEGIN	DSXネストカウントレジスタ、変位
707 YBEGIN	DSX状態レジスタ、DSXネストカウントレジスタ、変位
709 YBEGIN	DSX状態レジスタ、DSXネストカウントレジスタ、RTM状態レジスタ、変位

【図 8】



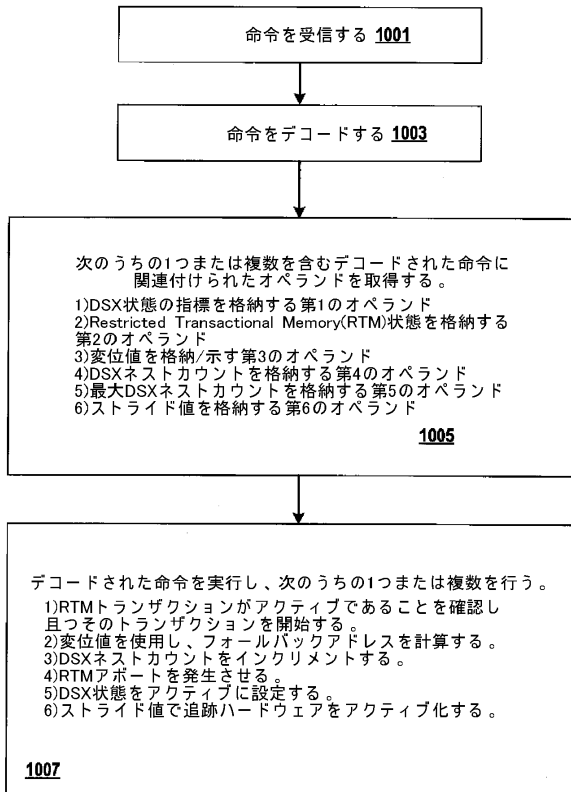
【図 9】

```

YBEGIN DISPLACEMENT
IF (RTM_ACTIVE)
  GOTO XBEGIN
IF(DSX_NESTING_COUNT < MAX_DSX_NESTING_COUNT)
  DSX_NEST_COUNT++
IF(DSX_NEST_COUNT == 1)
  FALLBACK_IP = CALC_AND_FAULT_CHECK_IP(RIP, DISPLACEMENT)
  DSX_ACTIVE = 1
ENTER_DSX_STATE(REGISTER, MEMORY_TRACKING)
FI
ELSE
  YABORT
FI
  
```

FIG. 9

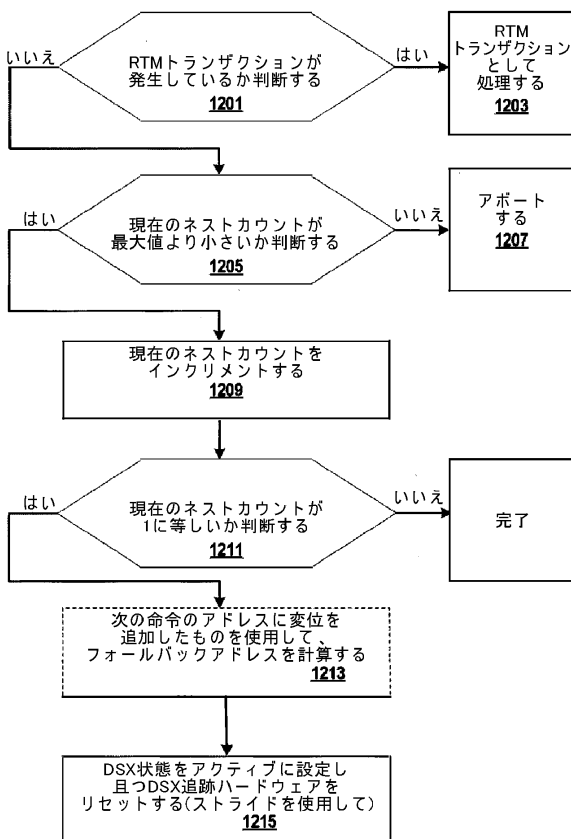
【図 10】



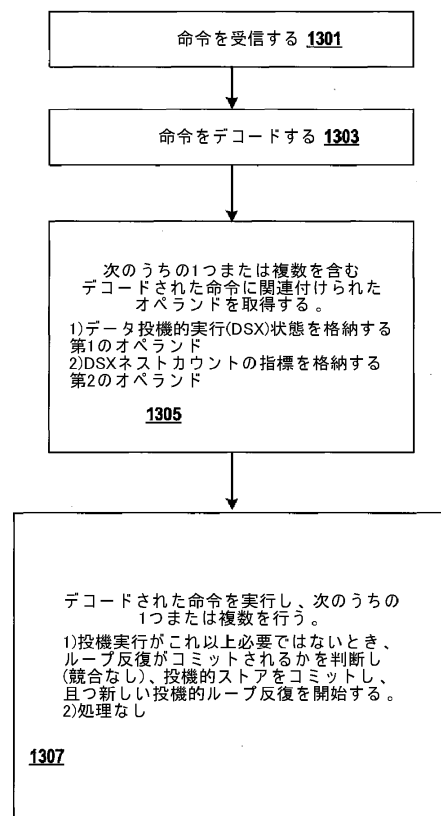
【図 11】

オペコード	明示的オペランド
1101 YBEGIN WITH STRIDE	変位、ストライド
1103 YBEGIN WITH STRIDE	DSX状態レジスタ、変位、ストライド
1105 YBEGIN WITH STRIDE	DSXネストカウントレジスタ、変位、ストライド
1107 YBEGIN WITH STRIDE	DSX状態レジスタ、DSXネストカウントレジスタ、変位、ストライド
1109 YBEGIN WITH STRIDE	DSXステータスレジスタ、DSXネストカウントレジスタ、RTM状態レジスタ、変位、ストライド

【図 12】



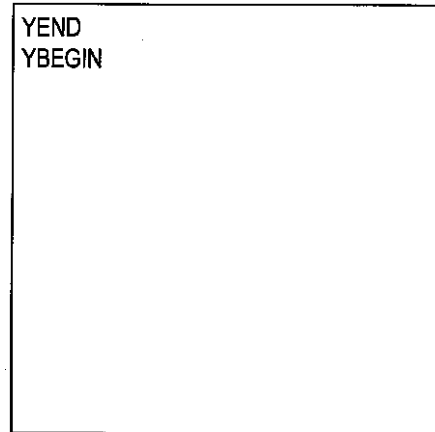
【図 13】



【図 14】

オペコード	明示的オペランド
1401 YCONTINUE	
1403 YCONTINUE	DSX状態レジスタ
1405 YCONTINUE	DSXネストカウントレジスタ
1407 YCONTINUE	DSX状態レジスタ、DSXネストカウントレジスタ

【図 16】



【図 15】

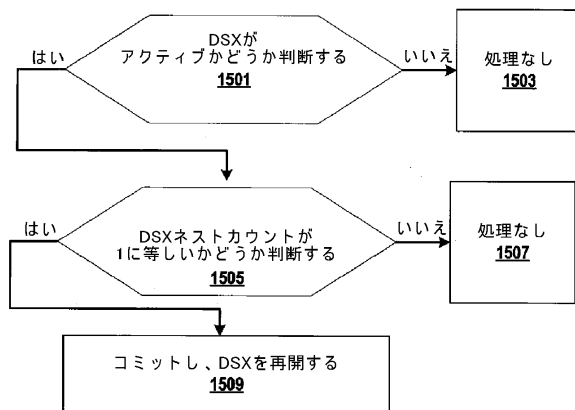
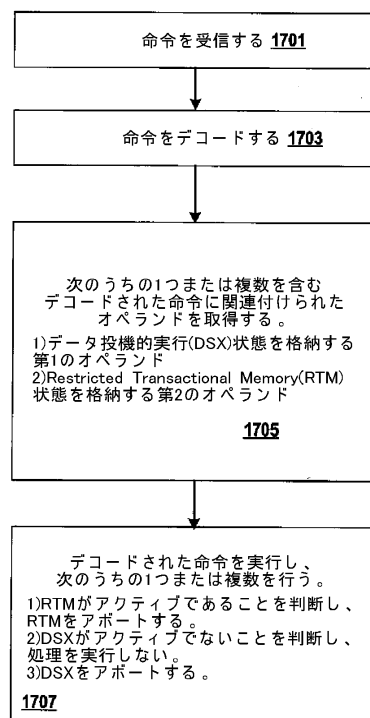


FIG. 16

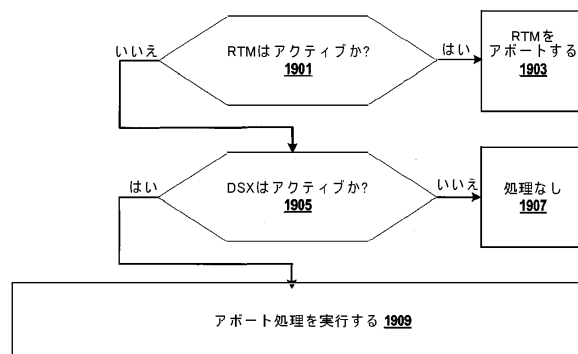
【図 17】



【図 18】

オペコード	明示的オペランド
1801 YABORT	
1803 YABORT	DSX状態レジスタ
1805 YABORT	DSX状態レジスタ、RTM状態レジスタ

【図 19】



【図 20】

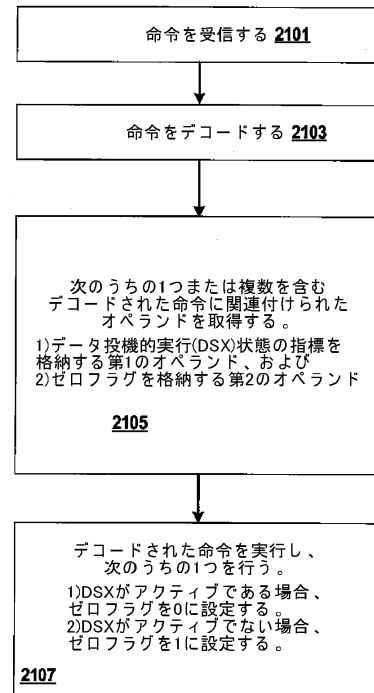
```

YABORT
IF(RTM_ACTIVE)
  GOTO XABORT
IF(!DSX_ACTIVE)
  NOP
ELSE
  DSX_ABORT_PROCESSING
  DSX_ACTIVE = 0

```

FIG. 20

【図 21】



【図 22】

オペコード	明示的オペランド
2201 YTEST	
2203 YTEST	DSX状態レジスタ
2205 YTEST	フラグレジスタ
2207 YTEST	DSX状態レジスタ、フラグレジスタ

【図 23】

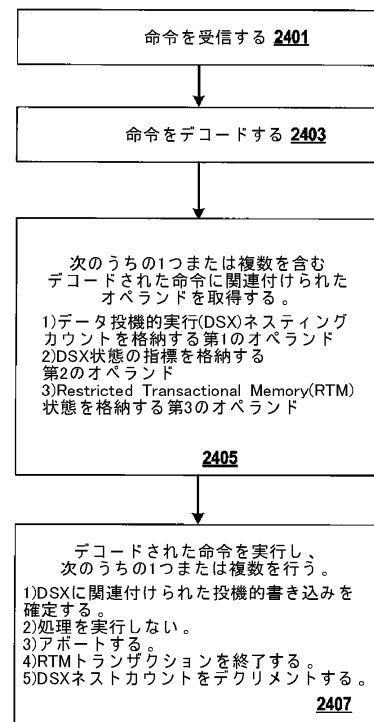
```

YTEST
If (DSX_ACTIVE)
  then ZF=0
ELSE
  ZF=1

```

FIG. 23

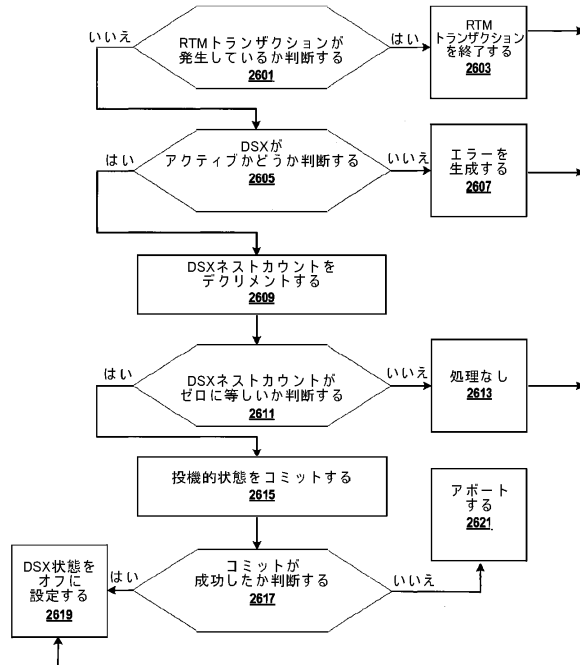
【図 24】



【図 25】

オペコード	明示的オペランド
2501 YEND	
2503 YEND	DSX状態レジスタ
2505 YEND	DSXネストカウントレジスタ
2507 YEND	DSX状態レジスタ、DSXネストカウントレジスタ
2509 YEND	DSX状態レジスタ、DSXネストカウントレジスタ、RTM状態レジスタ

【図 26】



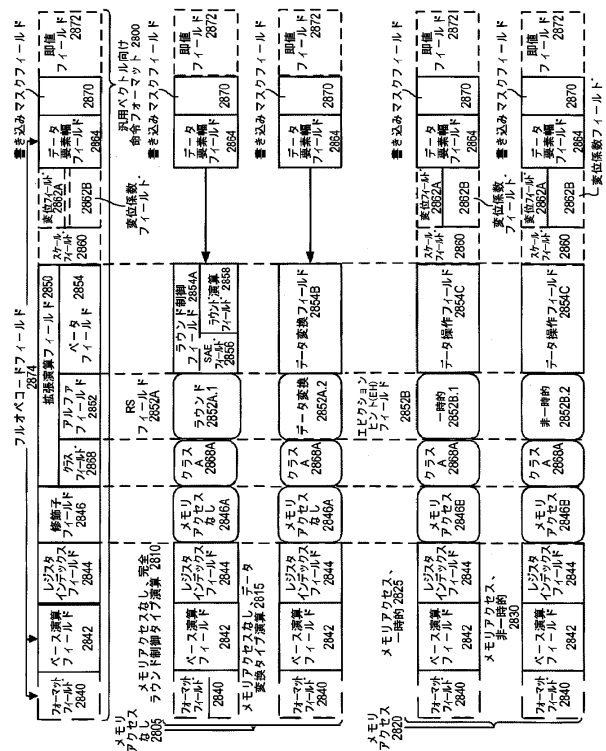
【図 27】

```

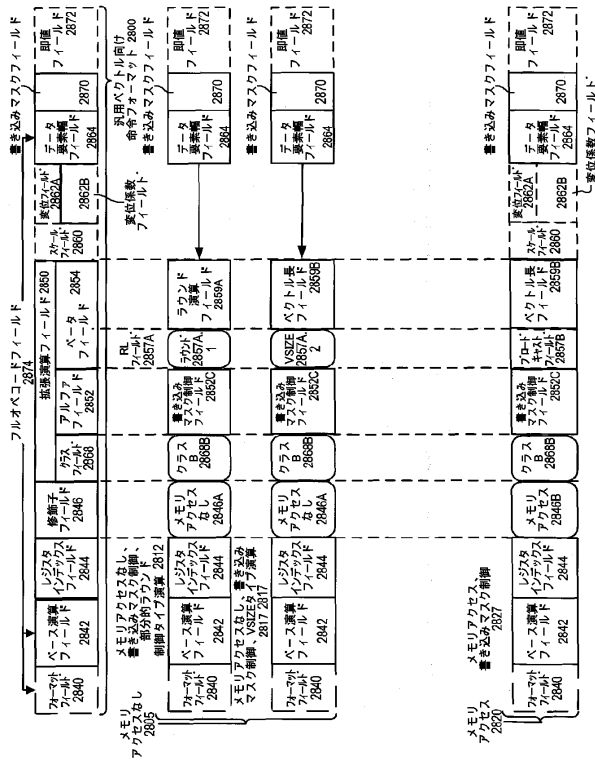
YEND
IF (RTM_ACTIVE)
GOTO XEND
IF (!DSX_ACTIVE)
#GP
ELSE
DSX_NEST_COUNT --
IF (DSX_NEST_COUNT == 0)
TRY TO COMMIT TRANSACTION
IF (FAIL)
GOTO RTM_ABORT
ELSE
DSX_ACTIVE = 0
  
```

FIG. 27

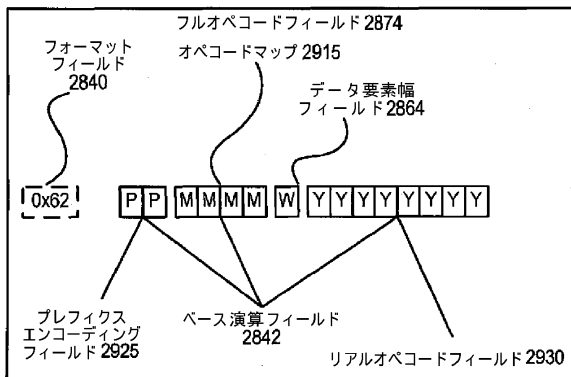
【図 28 A】



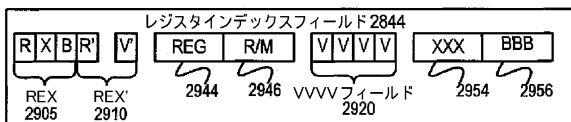
【図 28B】



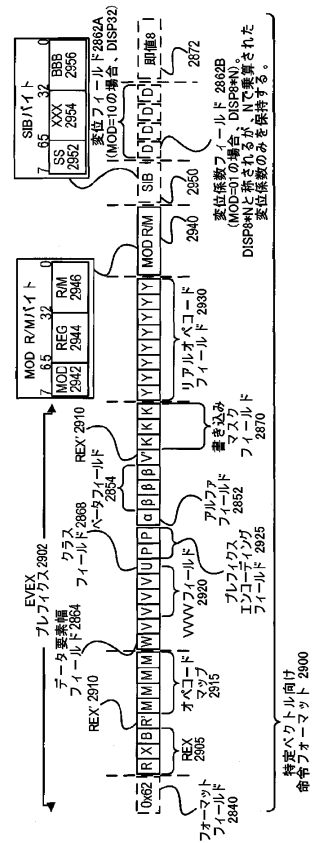
【図 29B】



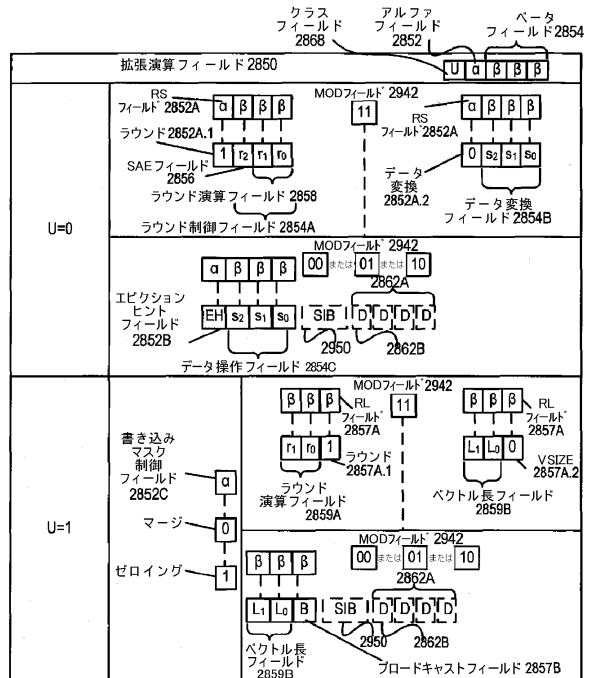
【図 29C】



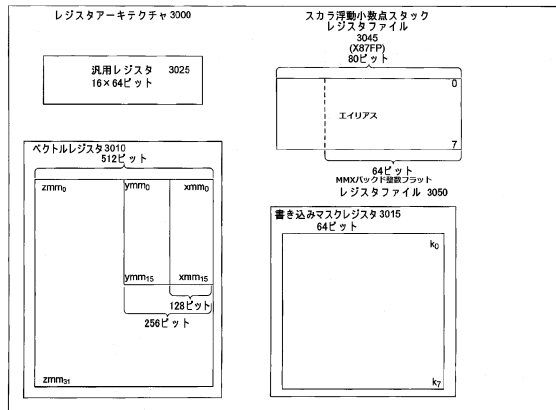
【図 29A】



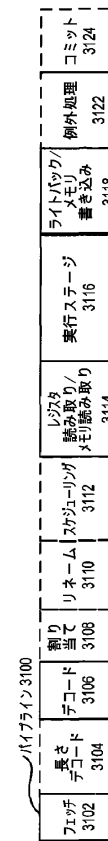
【図 29D】



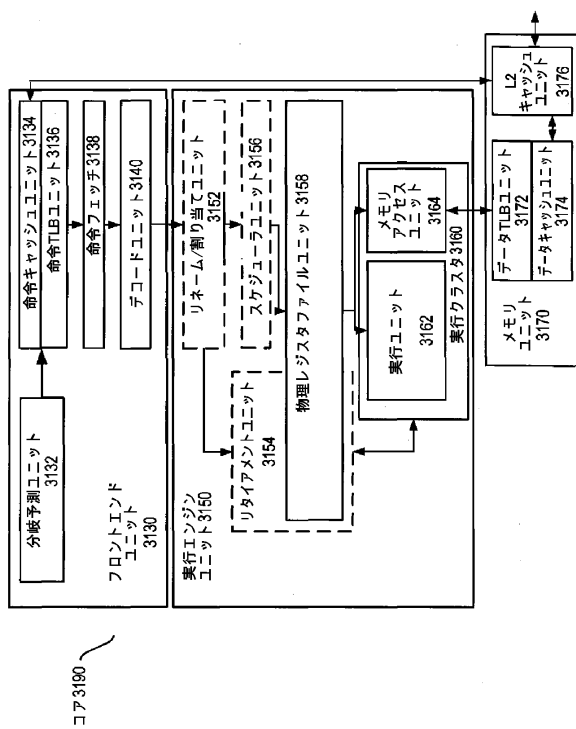
【図 30】



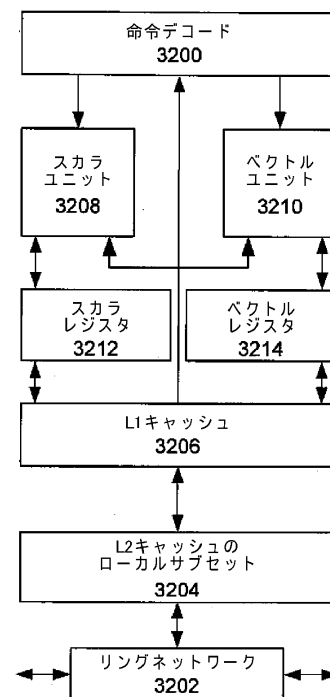
【図 31 A】



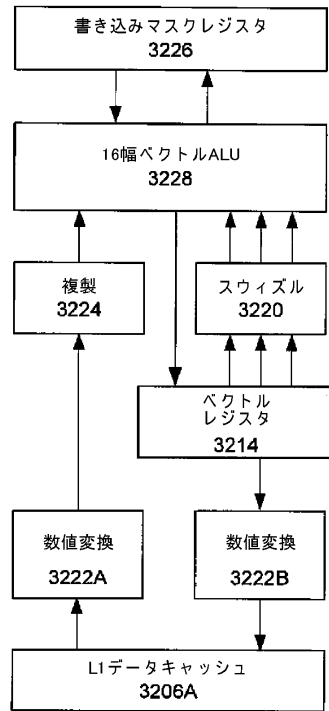
【図 31 B】



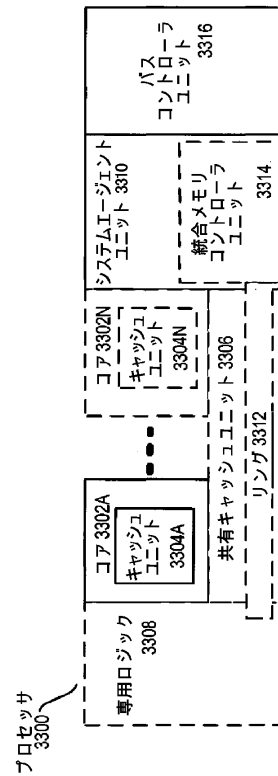
【図 32 A】



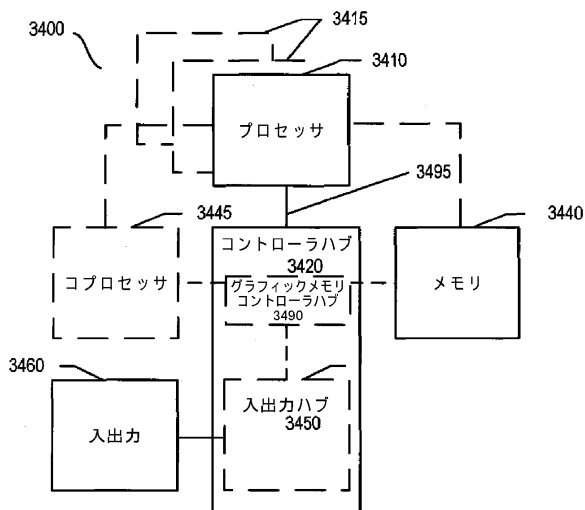
【図 3 2 B】



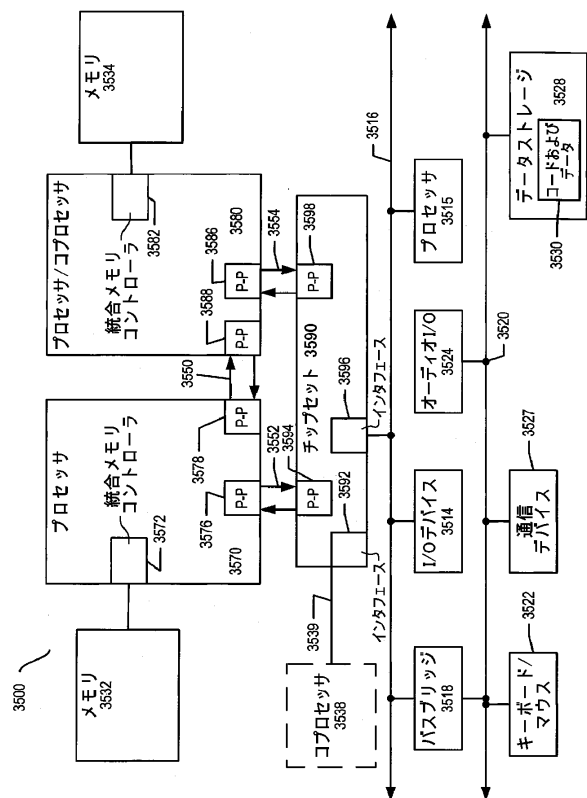
【図 3 3】



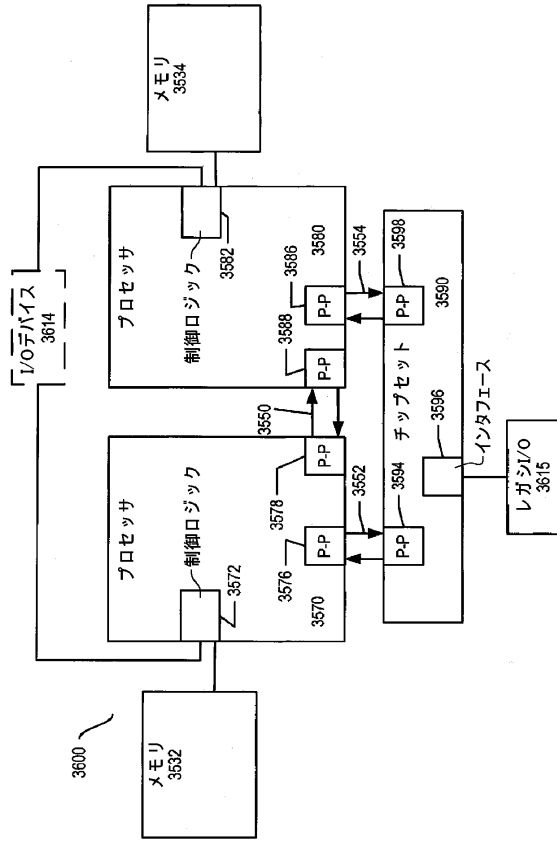
【図 3 4】



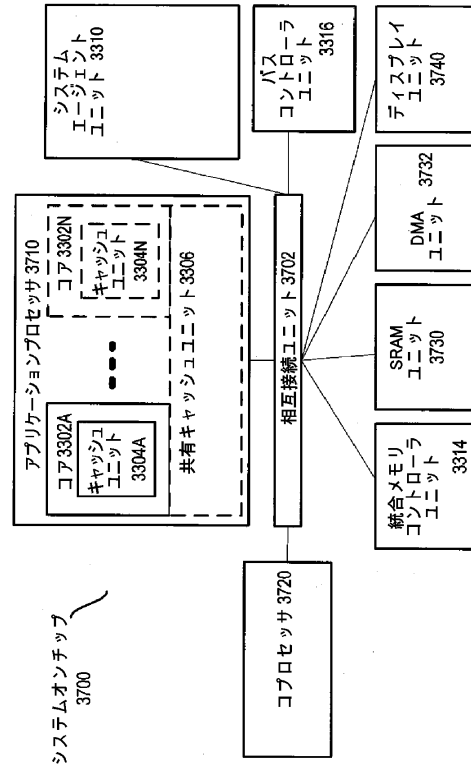
【図 3 5】



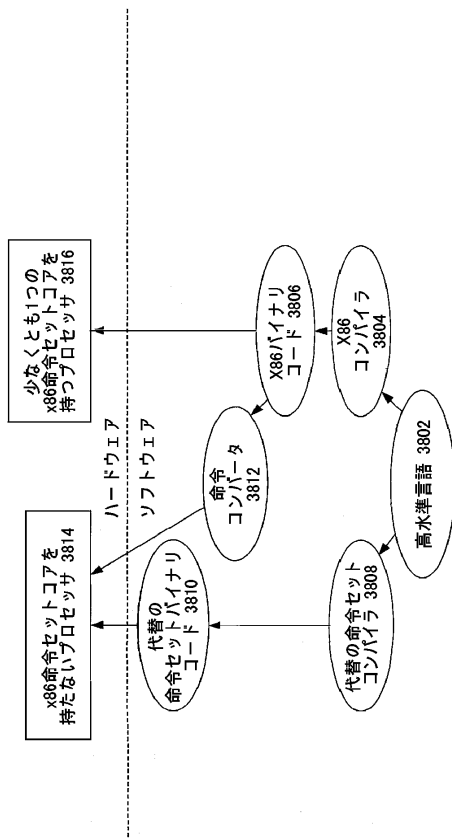
【図 36】



【図 37】



【図 38】



フロントページの続き

(51)Int.Cl. F I
G 0 6 F 8/41 1 3 0
G 0 6 F 17/16 G

- (72)発明者 ヒューズ、クリストファー ジェイ.
アメリカ合衆国 9 5 0 5 4 カリフォルニア州・サンタクララ・ミッション カレッジ ブレー
バード・2 2 0 0 インテル・コーポレーション内
- (72)発明者 バレンタイン、ロバート
アメリカ合衆国 9 5 0 5 4 カリフォルニア州・サンタクララ・ミッション カレッジ ブレー
バード・2 2 0 0 インテル・コーポレーション内
- (72)発明者 ジルカル、ミリンド ビー.
アメリカ合衆国 9 5 0 5 4 カリフォルニア州・サンタクララ・ミッション カレッジ ブレー
バード・2 2 0 0 インテル・コーポレーション内
- (72)発明者 イド、ヒデキ
アメリカ合衆国 9 5 0 5 4 カリフォルニア州・サンタクララ・ミッション カレッジ ブレー
バード・2 2 0 0 インテル・コーポレーション内
- (72)発明者 ウー、ヨウフェン
アメリカ合衆国 9 5 0 5 4 カリフォルニア州・サンタクララ・ミッション カレッジ ブレー
バード・2 2 0 0 インテル・コーポレーション内
- (72)発明者 ワン、チェン
アメリカ合衆国 9 5 0 5 4 カリフォルニア州・サンタクララ・ミッション カレッジ ブレー
バード・2 2 0 0 インテル・コーポレーション内

審査官 清木 泰

- (56)参考文献 米国特許出願公開第2005/0204119 (US, A1)
米国特許出願公開第2014/0059333 (US, A1)
米国特許出願公開第2013/0339961 (US, A1)
特表2018-500662 (JP, A)
特表2011-529603 (JP, A)
特表2010-510590 (JP, A)
米国特許出願公開第2015/0100754 (US, A1)
米国特許出願公開第2014/0040551 (US, A1)
米国特許出願公開第2013/0205119 (US, A1)
米国特許出願公開第2008/0263280 (US, A1)
米国特許出願公開第2008/0184011 (US, A1)
米国特許出願公開第2007/0266227 (US, A1)
石川隼輔, 山名早人, スレッドレベル投機的実行による依存距離不定運搬依存を持つループの並
列化, 情報処理学会研究報告, 日本, 社団法人情報処理学会, 2004年 3月19日, Vol.20
04, No.33, (2004-SLDM-114), pp.63~68
Jaewoong Chung, et al., ASF : AMD64 Extension for Lock-free Data Structures and Transa
ctional Memory, Proceedings of 43rd Annual IEEE/ACM International Symposium on Microar
chitecture, IEEE, 2010年12月 4日, pp.39-50
Advanced Micro Devices, Advanced Synchronization Facility Proposed Architectural Speci
fication, [online], developer.amd.com, 2009年 3月, pp.20-28, [令和元年9月25日検
索], インターネット, URL, http://developer.amd.com/wordpress/media/2013/09/45432-ASF_Spec_2.1.pdf

(58)調査した分野(Int.Cl., DB名)

G 0 6 F 9 / 3 8

G 0 6 F 9 / 3 0 - 9 / 3 5 5

G 0 6 F 9 / 4 5 5 - 9 / 5 4

G 0 6 F 1 7 / 1 0 - 1 7 / 1 8

G 0 6 F 8 / 4 0 - 8 / 5 4