(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7682613号
(P7682613)

(45)発行日　令和7年5月26日(2025.5.26)

(24)登録日　令和7年5月16日(2025.5.16)

(51)国際特許分類

		FI		
Ｇ０６Ｆ	*12/0842(2016.01)*	Ｇ０６Ｆ	12/0842	
Ｇ０６Ｆ	*12/123(2016.01)*	Ｇ０６Ｆ	12/123	
Ｇ０６Ｆ	*12/126(2016.01)*	Ｇ０６Ｆ	12/126	１００

請求項の数　21　外国語出願　（全46頁）

(21)出願番号　特願2020-150869(P2020-150869)
(22)出願日　令和2年9月8日(2020.9.8)
(65)公開番号　特開2021-86612(P2021-86612A)
(43)公開日　令和3年6月3日(2021.6.3)
審査請求日　令和5年9月4日(2023.9.4)
(31)優先権主張番号　16/696,548
(32)優先日　令和1年11月26日(2019.11.26)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　591003943
インテル・コーポレーション
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　ブーレバード・２２００
(74)代理人　110000877
弁理士法人ＲＹＵＫＡ国際特許事務所
(72)発明者　ネハ　ゴルカー
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　ブーレバード・２２００　インテル・コーポレーション内
(72)発明者　アキレシュ　クマー
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
最終頁に続く

(54)【発明の名称】　フレキシブルなキャッシュ割り当て技術の優先度ベースのキャッシュラインエビクションアルゴリズム

(57)【特許請求の範囲】

【請求項１】

複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービスレジスタ（ＣＬＯＳレジスタ）に関連付けられた、ＬＬＣと、

キャッシュ制御回路（ＣＣＣ）であって

前記ＬＬＣ内に無効キャッシュライン（無効ＣＬ）が存在する場合、前記無効ＣＬに、前記複数の優先度の１つである要求元優先度を有する後続キャッシュライン（後続ＣＬ）を格納し、

前記要求元優先度が前記複数の優先度のうちの最低のもので、前記要求元優先度が占有するウェイ数が１以上である、または前記要求元優先度について前記占有するウェイ数が最大である場合、前記後続ＣＬを、前記要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納し、

前記要求元優先度について前記占有するウェイ数が最小および前記最大の間である場合、前記要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納し、

前記要求元優先度について前記占有するウェイ数が前記最小よりも低く、前記より低い優先度を有するＣＬが存在する場合、前記より低い優先度を有するＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納し、

無効ＣＬまたは前記要求元優先度またはそれより低い優先度を有するＣＬが存在しな

い場合、より高い優先度のＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納する、ＣＣＣと、
　を備える、システム。
【請求項２】
　前記ＬＬＣは、複数セットのウェイを含み、前記複数のウェイは、前記複数セットの一部で、前記ＣＣＣは、前記後続ＣＬをどこに格納するかを決定する前に、前記後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、前記後続ＣＬが前記複数セットのいずれに含まれるかを判定するものとする、請求項１に記載のシステム。
【請求項３】
　ＬＬＣキャッシュエビクションに関するヒューリスティクスを維持するキャッシュ監視回路をさらに備え、
　より低い優先度を有する後続ＣＬを埋めるために空きを作るため、閾値よりも高い割合の高優先度を有するキャッシュラインがエビクションされる場合、前記高優先度に対する前記ＣＬＯＳレジスタが、前記占有する最小および最大ウェイを増やすように更新される、請求項１に記載のシステム。
【請求項４】
　前記複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である、請求項１に記載のシステム。
【請求項５】
　前記より低い優先度を有する前記ＬＲＵ　ＣＬが存在し、その代わりに前記後続ＣＬを格納する際、前記ＣＣＣは、前記ＬＲＵ　ＣＬを含むウェイ内にその他ＣＬが存在する場合、前記その他ＣＬをフラッシュさせるものである、請求項１に記載のシステム。
【請求項６】
　前記ＬＬＣおよび前記ＣＣＣを内蔵し、それぞれ仮想マシンを実装する１または複数のコアを有するプロセッサをさらに備え、前記ＣＣＣはハイパーバイザを含む、請求項１から５のいずれか一項に記載のシステム。
【請求項７】
　前記プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである、請求項６に記載のシステム。
【請求項８】
　複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービスレジスタ（ＣＬＯＳレジスタ）に関連付けられた、ＬＬＣを備えるシステムにおけるキャッシュ制御回路（ＣＣＣ）によって実行される方法であって、
　前記複数の優先度のうちのある要求元優先度を有する後続キャッシュライン（後続ＣＬ）を、前記ＬＬＣに格納する要求を受信する段階と、
　前記ＬＬＣ内に無効ＣＬが存在する場合、前記無効ＣＬに前記後続ＣＬを格納する段階と、
　前記要求元優先度が前記複数の優先度のうちの最低のもので前記要求元優先度が占有するウェイ数が１以上である、または前記要求元優先度について前記占有するウェイ数が最大である場合、前記後続ＣＬを、前記要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納する段階と、
前記要求元優先度について前記占有するウェイ数が最小および前記最大の間である場合、前記要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納する段階と、
前記要求元優先度について前記占有するウェイ数が前記最小よりも低く、前記より低い優先度を有するＣＬが存在する場合、前記より低い優先度を有するＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納する段階と、
　無効ＣＬまたは前記要求元優先度またはそれより低い優先度を有するＣＬが存在しない場合、より高い優先度のＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納する段階と、
　を含む、方法。

【請求項９】
前記ＬＬＣは、複数セットのウェイを含み、前記複数のウェイは、前記複数セットの一部で、前記ＣＣＣは、前記後続ＣＬをどこに格納するかを決定する前に、前記後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、前記後続ＣＬが前記複数セットのいずれに含まれるかを判定するものとする、請求項８に記載の方法。
【請求項１０】
ＬＬＣキャッシュ監視回路を使用して、ＬＬＣキャッシュエビクションに関するヒューリスティクスを維持し、
より低い優先度を有する後続ＣＬを埋めるために空きを作るため、閾値よりも高い割合の高優先度を有するキャッシュラインがエビクションされる場合、前記高優先度に対する前記ＣＬＯＳレジスタを、前記占有する最小および最大ウェイを増やすように更新する、請求項８に記載の方法。
【請求項１１】
前記複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である、請求項８に記載の方法。
【請求項１２】
前記より低い優先度を有する前記ＬＲＵ　ＣＬが存在し、その代わりに前記後続ＣＬを格納する際、前記ＣＣＣは、前記ＬＲＵ　ＣＬを含むウェイ内にその他ＣＬが存在する場合、前記その他ＣＬをフラッシュさせるものである、請求項８に記載の方法。
【請求項１３】
前記システムは、前記ＬＬＣおよび前記ＣＣＣを内蔵し、それぞれ仮想マシンを実装する１または複数のコアを有するプロセッサをさらに備え、前記ＣＣＣはハイパーバイザを含む、請求項８から１２のいずれか一項に記載の方法。
【請求項１４】
前記プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである、請求項１３に記載の方法。
【請求項１５】
複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられた、ＬＬＣを備えるシステムにおけるキャッシュ制御回路（ＣＣＣ）が実行する命令を含むコンピュータプログラムであって、前記実行は、
前記複数の優先度のうちのある要求元優先度を有する後続キャッシュライン（後続ＣＬ）を、前記ＬＬＣに格納する要求を受信することと、
前記ＬＬＣ内に無効ＣＬが存在する場合、前記無効ＣＬに前記後続ＣＬを格納することと、
前記要求元優先度が前記複数の優先度のうちの最低のもので、前記要求元優先度が占有するウェイ数が１以上である、または前記要求元優先度について前記占有するウェイ数が最大である場合、前記後続ＣＬを、前記要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納することと、
前記要求元優先度について前記占有するウェイ数が最小および前記最大の間である場合、前記要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納することと、
前記要求元優先度について前記占有するウェイ数が前記最小よりも低く、前記より低い優先度を有するＣＬが存在する場合、前記より低い優先度を有するＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納することと、
無効ＣＬまたは前記要求元優先度またはそれより低い優先度を有するＣＬが存在しない場合、より高い優先度のＬＲＵ　ＣＬの代わりに前記後続ＣＬを格納することと、
によって実施される、コンピュータプログラム。
【請求項１６】

前記ＬＬＣは、複数セットのウェイを含み、前記複数のウェイは、前記複数セットの一部で、前記ＣＣＣは、前記命令にさらに応答して、前記後続ＣＬをどこに格納するかを決定する前に、前記後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、前記後続ＣＬが前記複数セットのいずれに含まれるかを判定するものとする、請求項１５に記載のコンピュータプログラム。

【請求項１７】

前記システムは前記ＬＬＣおよび前記ＣＣＣを内蔵するプロセッサをさらに備え、前記プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである、請求項１５に記載のコンピュータプログラム。

【請求項１８】

前記複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である、請求項１５に記載のコンピュータプログラム。

【請求項１９】

前記より低い優先度を有する前記ＬＲＵ　ＣＬが存在し、その代わりに前記後続ＣＬを格納する際、前記ＣＣＣは、前記ＬＲＵ　ＣＬを含むウェイ内にその他ＣＬが存在する場合、前記その他ＣＬをフラッシュさせるものである、請求項１５に記載のコンピュータプログラム。

【請求項２０】

前記システムは、前記ＬＬＣおよび前記ＣＣＣを内蔵し、それぞれ仮想マシンを実装する１または複数のコアを有するプロセッサをさらに備え、前記ＣＣＣはハイパーバイザを含む、請求項１５から１９のいずれか一項に記載のコンピュータプログラム。

【請求項２１】

請求項１５から２０のいずれか一項に記載のコンピュータプログラムを格納する非一時的なコンピュータ読み取り可能な媒体。

【発明の詳細な説明】

【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、より詳細には、キャッシュ分割のための改良されたフレキシブルなキャッシュ割り当て技術（Ｆｌｅｘ－ＣＡＴ）の優先度ベースのエビクションアルゴリズムに関する。

【背景技術】

【０００２】

マルチテナンシーは、空間共有により高システム利用率および省コストを実現するソリューションであると確認されている。ユーザアプリケーションを実行する仮想マシン（ＶＭ）を各コアがホストする仮想化により、クラウド環境でマルチテナンシーが実現可能である。Ｆｕｎｃｔｉｏｎ　ａｓ　ａ　Ｓｅｒｖｉｃｅ（ＦａａＳ）等の新たなコンピューティングパラダイムは、多数の個別軽量機能をコンテナ内で実行するように、コンテナ型仮想化を利用する。典型的なマルチテナント環境においては、マルチコアプロセッサまたはコアのような同じコンピューティングリソース上に、高い優先度（ＨＰ）のジョブが低い優先度（ＬＰ）のジョブと共存する。ＨＰジョブがレイテンシに敏感なジョブであり、一方、ＬＰジョブは往々にして期限が緩い。ＨＰジョブの一部には、低レイテンシに加え、パフォーマンスの確定性が求められる。ジョブをサブミットするユーザは、サービス品質（ＱｏＳ）サービス水準合意（ＳＬＡ）をクラウドサービスプロバイダ（ＣＳＰ）と締結し、これらに応じてレイテンシまたはパフォーマンスの確定性の保証についてこれらに準拠する。ＣＳＰは、パフォーマンス変動、または同位置の他のＬＰジョブが引き起こすＨＰジョブのＱｏＳの低下を抑制することで、ＳＬＡを満たす必要がある。

【図面の簡単な説明】

【０００３】

本発明は、添付図面の図において限定ではなく例として示されており、それらの中で、同様の参照符号は同様の要素を示している。

【０００４】
【図１】いくつかの実施形態に係る、命令を実行する処理コンポーネントを示すブロック図である。
【０００５】
【図２】いくつかの実施形態に係る、仮想マシンを実行するマルチコアシステムを含むシステムを示すブロック図である。
【０００６】
【図３】いくつかの実施形態に係る、例示的なキャッシュ分割方式を示す。
【０００７】
【図４】いくつかの実施形態に係る、キャッシュラインエビクションを示すブロック図である。
【０００８】
【図５】いくつかの実施形態に係る、キャッシュフィル要求に応じてキャッシュ制御回路で実行される処理を示すブロックフロー図である。
【０００９】
【図６】いくつかの実施形態に係る、キャッシュフィル要求を処理するキャッシュ制御回路を示すフロー図である。
【００１０】
図７Ａおよび図７Ｂは、本発明のいくつかの実施形態に係る、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。
【００１１】
【図７Ａ】本発明のいくつかの実施形態に係る、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る。
【００１２】
【図７Ｂ】本発明のいくつかの実施形態に係る、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。
【００１３】
【図８Ａ】本発明のいくつかの実施形態に係る、例示的な特定ベクトル向け命令フォーマットを示すブロック図である。
【００１４】
【図８Ｂ】一実施形態による、特定ベクトル向け命令フォーマットの、フルオペコードフィールドを構成するフィールドを示すブロック図である。
【００１５】
【図８Ｃ】一実施形態による、特定ベクトル向け命令フォーマットの、レジスタインデックスフィールドを構成するフィールドを示すブロック図である。
【００１６】
【図８Ｄ】一実施形態による、特定ベクトル向け命令フォーマットの、拡張演算フィールドを構成するフィールドを示すブロック図である。
【００１７】
【図９】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。
【００１８】
【図１０Ａ】いくつかの実施形態に係る、例示的なインオーダパイプラインおよび例示的なレジスタリネーミング、アウトオブオーダ発行／実行パイプラインの両方を示すブロック図である。
【００１９】
【図１０Ｂ】いくつかの実施形態に係る、プロセッサに含まれる、インオーダアーキテクチャコアの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。
【００２０】
図１１Ａおよび図１１Ｂは、より具体的で例示的なインオーダコアアーキテクチャのブ

ロック図であり、そのコアはチップ内の（同じタイプおよび／または異なるタイプの他のコアを含む）いくつかの論理ブロックのうちの１つである。
【００２１】
【図１１Ａ】いくつかの実施形態に係る、オンダイ相互接続ネットワークへの接続を備え、かつ、レベル２（Ｌ２）キャッシュ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。
【００２２】
【図１１Ｂ】いくつかの実施形態に係る、図１１Ａのプロセッサコアの一部の拡大図である。
【００２３】
【図１２】いくつかの実施形態に係る、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、統合グラフィックを有してよい。
【００２４】
図１３から１６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。
【００２５】
【図１３】いくつかの実施形態に係る、システムのブロック図である。
【００２６】
【図１４】いくつかの実施形態に係る、第１のより具体的な例示的システムのブロック図である。
【００２７】
【図１５】いくつかの実施形態に係る、第２のより具体的な例示的システムのブロック図である。
【００２８】
【図１６】いくつかの実施形態に係る、システムオンチップ（ＳｏＣ）のブロック図である。
【００２９】
【図１７】いくつかの実施形態に係る、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】
【００３０】
以下の説明では、多数の具体的な詳細が示されている。ただし、いくつかの実施形態はこれら具体的な詳細無しで実施でき得ることが理解されよう。他の例では、本記載の理解を損なわないように、既知の回路、構造、および技術については詳細に示さない。
【００３１】
本明細書での「一実施形態」、「ある実施形態」、「例としての実施形態」等の表現は、記載された実施形態にある特徴、構造、または特性が含まれ得ることを示すが、必ずしもすべての実施形態がこれらの特徴、構造、または特性を含むわけではない。さらに、このような文言は、必ずしも同一の実施形態を指していない。さらに、ある特徴、構造、または特性がある実施形態について記載される場合、明示的に記載されていれば他の実施形態についてもこれらの特徴、構造、または特性に影響を与えることが当業者には自明であることを示す。
【００３２】
上述のように、クラウドサービスプロバイダ（ＣＳＰ）は、パフォーマンス変動、または同位置の他の低パフォーマンスＬＰジョブが引き起こす高パフォーマンス（ＨＰ）ジョブのサービス品質（ＱｏＳ）の低下を抑制することで、サービス水準合意（ＳＬＡ）を満たす必要がある。具体的には、開示の実施形態は、フレキシブルなキャッシュ割り当て技術（Ｆｌｅｘｉｂｌｅ　Ｃａｃｈｅ　Ａｌｌｏｃａｔｉｏｎ　Ｔｅｃｈｎｏｌｏｇｙ：Ｆｌｅｘ－ＣＡＴ）について説明する。これは、ＬＰジョブによる、ＨＰラストレベルキャッ

シュ（ＬＬＣ）エビクションを抑制する構造的ソリューションである。Ｆｌｅｘ－ＣＡＴ手法は、各種優先度により、各キャッシュセット内のキャッシュライン（ＣＬ）の利用に応じて、各セットに最適なウェイ数を動的に決定するものである。その境界（最小および最大ウェイ数）は、ＬＬＣキャッシュフィル時に、エビクション対象を選択する際のヒントを提供するモデル固有レジスタ（ＭＳＲ）内に指定される。

【００３３】

Ｆｌｅｘ－ＣＡＴは、キャッシュパーティションを指定するための、容易に設定可能で、それでいてフレキシブルなインタフェースを提供するという利点がある。Ｆｌｅｘ－ＣＡＴは、詳細な粒度でパーティションを管理するため、リアルタイムデータに基づく、優先度式ＬＬＣエビクション決定を下す動的キャッシュ分割方式をサポートする。Ｆｌｅｘ－ＣＡＴは、パフォーマンスの確定性、ならびに高パフォーマンスジョブおよび低パフォーマンスジョブの分離等の、ＱｏＳ保証を満たす構造的特徴に対する、クラウドサービスプロバイダの要求を満たすことに寄与する。本明細書において、低パフォーマンスジョブを「ノイジーネイバー（ｎｏｉｓｙ　ｎｅｉｇｈｂｏｒｓ；うるさい隣人）」とも称する場合がある。

【００３４】

より劣る代替的な手法は、コアにウェイの個別セットを割り当て、ＨＰおよびＬＰジョブを特定のコアに限定することで、リソースを共有するＨＰおよびＬＰジョブ間の不均衡を解消しようとしてきた。しかしそのような手法にはいくつか問題がある。例えば、当該メカニズムには、優先度という概念が存在しない。これら手法の一部は、すべてのキャッシュセットおよびＨＰコアに固有のウェイセットを割り当てることによりＬＰジョブからＨＰジョブを分離するが、このような固有のリソースは、ＨＰ作業負荷に利用されていないときには、ＬＰ作業負荷に利用することができない。さらに、これら手法の一部によると、特定の限定されたキャッシュセット（例えば、ｘセット）が他よりも飽和する（例えば、Ｎをキャッシュセットの総数とすると、Ｎ－ｘセット）。これらｘセットの飽和による、ＨＰジョブへの均等ウェイ割り当ては、Ｎ－ｘセット全体での過提供、ならびにこれらＮ－ｘの過少利用につながる可能性がある。さらに、コアに最大ウェイよりも少なく割り当てると、結合性の低下が生じ、競合性ミスが多くなり、パフォーマンス低下につながる。固定のキャッシュウェイをコアに割り当てる静的割り当てを利用した方法の場合、キャッシュエビクションおよびフィル時に柔軟性の入る余地がない。

【００３５】

一方で開示の実施形態は、動的に優先度とキャッシュパーティションを指定するため、フレキシブルなインタフェースを提供する。優先度は高くなる順で列挙される。フレキシブルなキャッシュパーティションは、優先度毎の最大および最小ウェイ数について指定可能である。いくつかのその他手法と異なり、Ｆｌｅｘ－ＣＡＴは各パーティションに割り当てられるキャッシュウェイを正確に指定するソフトウェアを必要としない。

【００３６】

そのような動的な優先度およびキャッシュ分割仕様は、クラスオブサービス（ＣＬＯＳ）レジスタによりサポートされる。同レジスタは、各ＣＬＯＳに対して以下の値を保持する。

・ＣＬＯＳ優先度Ｐ：Ｐｎビット
・優先度Ｐが占める最大ウェイ数：ｍｘｗｎビット
・優先度Ｐが占める最小ウェイ数：ｍｎｗｎビット

【００３７】

例えば、優先度最大数を４とすると、Ｐｎ＝ｌｏｇ（４）＝２。最大ウェイ数を１６とすると、ｍｘｗｎ＝ｍｎｗｎ＝ｌｏｇ（１６）＝４。

【表１】

CLOS[0]:(0,12,4)および CLOS[1]:(1,4,1)とする

CLOS	優先度	最大 W(mxwn ビット)	最小 W(mnwn ビット)
0	0	12	4
1	1	4	1

【００３８】
本明細書に開示の実施形態によると、要求元はＬＬＣに埋められる予定のＣＬのオーナである。要求元の優先度をＰＦとする。システムにおいて、ＰＬが最低優先度で、ＰＨが最高優先度である。ｌｏｃをＦｌｅｘ－ＣＡＴにより決定される要求元のＣＬに対する最終格納場所とする。要求元の占有数Ｏ［ＰＦ］は、インデックスされたキャッシュセット内で、要求元が占めるＣＬの数である。
【００３９】
Ｆｌｅｘ－ＣＡＴは、キャッシュセット粒度で、優先度式キャッシュ分割を実行する、新たなエビクションアルゴリズムである。要求元の占有数が最大ウェイ割り当て（ｍｘｗ）を超えない限り、優先的にＬＰ　ＣＬがエビクションされる。要求元の占有数が最大割り当てに達すると、Ｆｌｅｘ－ＣＡＴはその他優先度エビクションよりも自己エビクションを優先し、パーティション境界内に留まるようにする。これら前２段階で対象が発見できない限定的状況では、Ｆｌｅｘ－ＣＡＴは後続のキャッシュフィルの場所を作るため、ＨＰ　ＣＬをエビクションに選択する。このようなＦｌｅｘ－ＣＡＴの基本的考え方を図４に示す。
【００４０】
詳細なアルゴリズムを図４から図６のフローチャートに示し、以下に説明する。ＬＬＣフィル時に、開示の実施形態は、従来のハッシングアルゴリズムにより、要求元の後続キャッシュラインに対するキャッシュセットインデックスを判定する。
【００４１】
適切なキャッシュセットにインデックスした後、Ｆｌｅｘ－ＣＡＴは最初に、インデックスされたキャッシュセット内の無効なＬＬＣエントリを探す。キャッシュセットが一杯で、無効格納場所が見つからない場合、Ｆｌｅｘ－ＣＡＴはＬＬＣからエビクションされる必要のある対象ＣＬを判定する。これにより、Ｆｌｅｘ－ＣＡＴは飽和したキャッシュセットのみでイネーブルされ、コンテンションがなければ無用に作業負荷が課されないことが保証される。Ｆｌｅｘ－ＣＡＴはキャッシュセット全体を走査し、システムの各優先度について、ＬＲＵ　ＣＬのインデックス、その経過時間、および占有数を判定する。
【００４２】
要求元の占有数が最小割り当て未満（Ｏ［ＰＦ］＜ＰＦ［ｍｎｗ］）であれば、Ｆｌｅｘ－ＣＡＴはＬＰ　ＬＲＵ　ＣＬを優先してエビクションして、その占有数を上げる。要求元の占有数が最小割り当てに達し、かつ最大割り当て未満（ＰＦ［ｍｎｗ］≦Ｏ［ＰＦ］＜ＰＦ［ｍｘｗ］）であれば、Ｆｌｅｘ－ＣＡＴは、優先度の中から、ＬＲＵ対象を探しながら、さらにそのＬＲＵ　ＣＬを候補リストに加える。要求元の占有数が最大割り当てに達すると、Ｆｌｅｘ－ＣＡＴはＬＰ　ＬＲＵ候補を無視して、要求元のＬＲＵ（ＬＲＵＦ）を対象ＣＬとして選択し、要求元の占有数が決して上限（ＰＦ［ｍｘｗ］）を超えないことを保証する。前段階で対象が発見されなければ（すべてのラインがより高い優先度のオーナに属する場合）、Ｆｌｅｘ－ＣＡＴはＨＰエビクションを実施する。
【００４３】
図５および図６に示し、以下に説明するフローチャートに、適切なキャッシュセットにインデックスした後にＦｌｅｘ－ＣＡＴが経る段階を示す。

【００４４】
図１は、いくつかの実施形態に係る、命令を実行する処理コンポーネントを示すブロック図である。図示のとおり、ストレージ１０１は実行される命令（複数可）１０３を格納する。さらに後述するように、いくつかの実施形態において、システム１００（本明細書では「コンピューティングシステム」とも称する）は、行列を含む、パックドデータベクトルの複数要素を同時に処理するＳＩＭＤプロセッサである。
【００４５】
動作時、命令（複数可）１０３はフェッチ回路１０５によりストレージ１０１からフェッチされる。命令は、復号回路１０９により復号される。復号回路１０９は、フェッチされた命令１０７を、１または複数の動作に復号する。いくつかの実施形態において、この復号は実行回路（実行回路１１７等）により実行される複数のマイクロ動作を生成することを含む。復号回路１０９はさらに、（使用される場合）命令サフィクスとプレフィクスとを復号する。
【００４６】
いくつかの実施形態において、レジスタリネーミング、レジスタ割り当て、および／またはスケジューリング回路１１３は、以下のうちの１または複数のための機能を提供する。１）論理オペランド値を物理オペランド値にリネーミング（例えば、いくつかの実施形態においてレジスタエイリアステーブル）、２）ステータスビットおよびフラグを復号された命令に割り当て、３）命令プールのうち、復号された命令１１１を実行回路１１７上での実行用にスケジューリング（例えば、いくつかの実施形態においてリザベーションステーションを利用）。
【００４７】
レジスタ（レジスタファイル）および／またはメモリ１１５は、データを実行回路１１７により実行される命令１１１のオペランドとして格納する。例示的なレジスタの種類は、少なくとも図９を参照して以下にさらに説明し、示すように、書き込みマスクレジスタ、パックドデータレジスタ、汎用レジスタ、フローティングポイントレジスタを含む。
【００４８】
いくつかの実施形態において、ライトバック回路１１９は命令実行結果をコミットする。実行回路１１７およびシステム１００は、図２～図４、図１０Ａ、図１０Ｂ、図１１Ａ、図１１Ｂにさらに図示され、それらを参照に説明される。
【００４９】
図２は、いくつかの実施形態に係る、仮想マシンを実行するマルチコアプロセッサを含むシステムを示すブロック図である。図示のとおり、コンピューティングシステム２００は、ラストレベルキャッシュＬＬＣ２０４を共有するコア０　２０６Ａ、コア１　２０６Ｂ、…コアＮ　２０６Ｎまでを含むマルチコアプロセッサ２０２を含む。これとともに、プロセッサ２０２のリソースは、１または複数のクライアントにネットワークサービスを提供するように、クラウドサービスプロバイダ（ＣＳＰ）のコンピューティングプラットフォームの一部として機能できる。例えば、図示のとおり、コア０、１、からＮは、ＶＭ０　２１０Ａ、ＶＭ１　２１０Ｂ、…ＶＭＮ　２１０Ｎまでをサポートする。ＶＭ０　２１０Ａは、ＶＮＦアプリケーション０　２１２Ａ（仮想ネットワーク機能アプリケーション）と、ゲストＯＳ０　２１４Ａとをサポートする。同様に、ＶＭ１　２１０Ｂは、ＶＮＦアプリケーション１　２１２Ｂと、ゲストＯＳ１　２１４Ｂとをサポートする。同様に、ＶＭＮ　２１０Ｎ、はＶＮＦアプリケーション　２１２Ｎと、ゲストＯＳ　Ｎ　２１４Ｎとをサポートする。動作時、仮想マシンはハイパーバイザ／ＶＭＭ２０８を使用して起動および管理される。さらにシステムを管理するため、オペレーティングシステム２１４を呼び出すことが可能である。
【００５０】
いくつかの実施形態において、キャッシュ制御回路２０１は、ハイパーバイザ／ＶＭＭ２０８と協働して、本明細書に記載のキャッシュ分割方式を実施する。
【００５１】

いくつかの実施形態において、キャッシュ監視回路２０３は、高優先度キャッシュラインのエビクションを生じる低優先度キャッシュフィル要求の割合等の、キャッシュアクセス要求に係る統計およびヒューリスティクスを維持する。キャッシュ監視回路２０３は、代替的にプロセッサ２０２に組み込まれることが可能であれば、破線で囲っていることで示されるように、任意のものである。いくつかの例において、コンピューティングシステム２００はスタンドアロンコンピューティングプラットフォームであるが、別の例では、ネットワーク（図示せず）を介して別のコンピューティングプラットフォームに結合される。

【００５２】

いくつかの実施形態において、コンピューティングシステム２００はデータセンタ内のノードであって、１または複数のＶＮＦアプリケーションを個別に実行するＶＭをサポートする。同アプリケーションは、例えば、クラウドサービスプロバイダ、データベースネットワークサービス、ウェブサイトホスティングサービス、ルーティングネットワークサービス、ｅメールサービス、ファイアウォールサービス、ドメインネームサービス（ＤＮＳ）、キャッシングサービス、ネットワークアドレストランスレーション（ＮＡＴ）サービス、またはウイルススキャンネットワークサービスを含む。コンピューティングシステム２００におけるＶＭ２１０Ａ～２１０Ｎは、ハイパーバイザ／ＶＭＭ２０８等の、ハイパーバイザまたは仮想マシンマネージャー（ＶＭＭ）により管理または制御され得る。他の実施形態では、コンピューティングシステム２００は、同一の物理的エンクロージャ、シャーシ、またはコンテナに収められた、各種上述のコンピューティングリソースを有する、より従来どおりのサーバとして構成されてよい。

【００５３】

いくつかの実施形態によると、仮想マシンは、物理コンピュータ同様、オペレーティングシステムおよびアプリケーションを実行するソフトウェアコンピュータである。一部の仮想マシンは、構成ファイルセットにより構成され、ホストの物理リソースにより補助される。また、ハイパーバイザまたはＶＭＭは、仮想マシンを生成および管理する、コンピュータソフトウェア、ファームウェア、またはハードウェアである。ハイパーバイザが１または複数の仮想マシンを動作させるコンピュータを、ホストマシンと称し、各仮想マシンをゲストマシンと称する。ハイパーバイザまたはＶＭＭは、仮想オペレーティングプラットフォームによりゲストオペレーティングシステムを提示し、ゲストオペレーティングシステムの実行を管理する。様々なオペレーティングシステムの多数のインスタンスが、仮想化ハードウェアリソースを共有し得る。例えば、Ｌｉｎｕｘ（登録商標）、Ｗｉｎｄｏｗｓ（登録商標）、ｍａｃＯＳ（登録商標）インスタンスはすべて、マルチコアを有する単一の物理プロセッサ上で動作可能である。

【００５４】

いくつかの例において、図２に示すように、コンピューティングシステム２００に対するコンピューティングリソースの少なくとも一部は、共有ラストレベルキャッシュ（ＬＬＣ）２０４を有するＣＰＵ／コア２０６Ａ、２０６Ｂ、…２０６Ｎ等の、処理要素を含み得る。

【００５５】

ＬＬＣ２０４は、いくつかの例において、プロセッサ２０２の外部にある。いくつかの例によると、共有ＬＬＣ２０４は、アクセスレイテンシを最小限に抑えるため、ＣＰＵ／コア２０６Ａから２０６Ｎに対する共有ＬＬＣとして機能するため、比較的高速のアクセスメモリの種類であり得る。共有ＬＬＣ２０４に含まれる比較的高速のアクセスメモリの種類は、揮発性または不揮発性メモリを含み得るがこれらに限定されない。揮発性メモリの種類は、スタティックランダムアクセスメモリ（ＳＲＡＭ）もしくはダイナミックランダムアクセスメモリ（ＤＲＡＭ）、サイリスタＲＡＭ（ＴＲＡＭ）、またはゼロキャパシタＲＡＭ（ＺＲＡＭ）を含むことができるが、これらに限定されない。不揮発性メモリの種類は、カルコゲナイド相変化材料（例えばカルコゲナイドガラス）を含む３次元（３Ｄ）クロスポイントメモリ構造を有する、バイトまたはブロックアドレス指定可能な不揮発

性メモリの種類（以下、「３Ｄクロスポイントメモリ」と呼ぶ）を含むことができるが、これらに限定されない。不揮発性メモリの種類はさらに、バイトまたはブロックアドレス指定可能な不揮発性メモリの他の種類を含み得る。この例として、多閾値ＮＡＮＤフラッシュメモリ、ＮＯＲフラッシュメモリ、単一または複数相変化メモリ（ＰＣＭ）、抵抗性メモリ、ナノワイヤメモリ、強誘電体トランジスタランダムアクセスメモリ（ＦｅＴＲＡＭ）、メモリスタ技術を組み込んだ磁気抵抗ランダムアクセスメモリ（ＭＲＡＭ）、スピントランスファトルクＭＲＡＭ（ＳＴＴ－ＭＲＡＭ）、または上記の任意の組み合わせが挙げられるが、これらに限定されない。

【００５６】

図３は、いくつかの実施形態に係る、例示的なＬＬＣキャッシュ分割方式を示す。図示のように、方式３００は、図２に示すようなコンピューティングシステム２００の共有ＬＬＣ２０４として使用可能な方式の一例である。ここでＬＬＣ３０４は、各キャッシュウェイ３０２が８つのキャッシュラインを含む、８ウェイセットアソシアティブキャッシュとして示されている。ＬＬＣ３０４のキャッシュラインは一部が低優先度アプリケーション３０６に割り当てられ、一部が高優先度アプリケーション３１０に割り当てられ、一部が無効３０８である。説明を簡潔にするため、共有ＬＬＣ３５４は、各ウェイが１つのキャッシュラインを含む、８ウェイセットアソシアティブキャッシュとして示されている。ＬＬＣ３５４のキャッシュラインは一部が低優先度アプリケーション３５６に割り当てられ、一部が高優先度アプリケーション３６０に割り当てられ、一部が無効３５８である。図３に図示したものは、開示の実施形態を特定の構成に限定するものではない。別の方式は、より多くのまたはより少ないセット、セット内により多くのまたはより少ないウェイ、各ウェイ内により多くのまたはより少ないキャッシュラインなどを含んでもよい。例えば、ＬＬＣ２０４は、各セットがＭ個のキャッシュラインを含む、Ｎウェイセットアソシアティブキャッシュであり得る。ここで、ＮおよびＭは１以上の正の整数である。

【００５７】

動作時、さらに後述するように、ＬＬＣ２０４は、ＬＬＣを共有するアプリケーションの必要に応じて、動的に再分割される。開示の実施形態の利点として、低優先度アプリケーションにより、高優先度アプリケーションに割り当てられたキャッシュラインのエビクションを最小限に抑えることを図る。

【００５８】

図４は、Ｆｌｅｘ－ＣＡＴアルゴリズムのいくつかの実施形態に係る、キャッシュラインエビクションを示すブロック図である。開示の実施形態によると、Ｆｌｅｘ－ＣＡＴはキャッシュセット粒度で優先度式キャッシュ分割を行うエビクションアルゴリズムである。図示のように、方式４００は、優先度４０２の範囲内に入る優先度のアプリケーションを示す。円弧４０４、４０６、４０８は、要求元コアからの後続キャッシュラインで、キャッシュラインを満たす要求のために空きを作るためのキャッシュラインエビクションを示す。エビクション４０８等の一部のエビクションは、より低い優先度割り当てのエビクションにより、より高い優先度割り当てのための空きを作る。一部のエビクションは自己エビクションで、円弧４０６で示す（例えば、後続ＣＬのために空きを作るため、最大ウェイ数を既に割り当てた優先度が自己エビクションする）。要求元の占有数が最大割り当てに達すると、Ｆｌｅｘ－ＣＡＴはその他優先度エビクションよりも自己エビクションを優先し、パーティション境界内に留まるようにする。これら前２段階で対象が発見できない限定的状況では、Ｆｌｅｘ－ＣＡＴは後続のキャッシュフィルの場所を作るため、ＨＰＣＬをエビクションに選択する。Ｆｌｅｘ－ＣＡＴは、エビクション４０８のようなＬＰＣＬのエビクションの最大化を図り、エビクション４０４のようなＨＰ　ＣＬのエビクションの最小化を図る。

【００５９】

図５は、いくつかの実施形態に係る、キャッシュフィル要求に応じてキャッシュ制御回路で実行される処理を示すブロックフロー図である。例えば、図２のキャッシュ制御回路（ＣＣＣ）２０１によりフロー５００が実行可能である。図示のとおり、５０１において

フロー５００が開始し、ＣＣＣは、要求元優先度を有する後続のキャッシュフィル要求を要求元から受信するものとする。例えば、要求元は図２のコア２０６Ａ、２０６Ｂ、…２０６Ｎまでのうちの１つであり得、図１のメモリ１１５等のメモリから、後続キャッシュラインが取得可能である。要求元優先度は、要求元コア内で動作するアプリケーションに割り当てられた優先度を反映可能である。動作５０２において、ＣＣＣは、ＬＬＣ内に無効キャッシュライン（ＣＬ）が１つでも存在するかを判定するものとする。存在する場合、動作５０４において、ＣＣＣは、無効キャッシュラインの格納場所に後続ＣＬを書き込むものとし、５０５において格納場所が発見されるとフローが終了する。しかし、動作５０２において無効ＣＬが存在しないと示されると、５０６においてＣＣＣは、システム内の各優先度について、優先度（Ｐ）と、優先度Ｐ（ＬＲＵｐ）を有するＬＲＵ　ＣＬのインデックスと、ＬＲＵｐの経過時間と、優先度が占めるウェイ数とを決定するものとする。動作５０８において、ＣＣＣは、要求元優先度（$Ｐ_Ｒ$）が最低優先度であるかを判定するものとする。最低優先度である場合、ＣＣＣは動作５１０において、要求元優先度（Ｏ［$Ｐ_Ｒ$］）の占有数が０に等しいかを判定するものとする。等しい場合、フローは動作５２４に進む。ここでは、後続ＣＬ（$Ｐ_Ｒ$）のために空きを作るため、より高い優先度ＣＬ（$Ｐ_Ｈ$）がエビクションされる。

【００６０】

動作５２４は、より低い優先度キャッシュラインのために空きを作るため、高優先度キャッシュラインがエビクションされる状況を示す。これは、上述の場合を除き、開示の実施形態が可能な限り避けることを図る状況である。いくつかの実施形態において、図２のキャッシュ監視回路２０３等のキャッシュ監視回路は、動作５２４のインスタンスを含む、キャッシュフィル要求へのヒューリスティクス追跡応答を維持する。いくつかの実施形態において、ＣＣＣはヒューリスティクスを監視し、必要に応じて、各優先度に割り当てられた最小および最大ウェイ数を動的に調整する。これにより、最終的に動作５２４につながるような、より低い優先度のアプリケーションのエビクションに対するより高い優先度アプリケーションの積極性を調節する。いくつかの実施形態において、キャッシュ監視回路２０３はＬＬＣを再分割させる。またはいくつかの実施形態において、動作５２４の発生等の維持されたヒューリスティクスが所定の閾値を超える場合、キャッシュ監視回路は各種優先度に関連付けられたウェイの境界を調整させる。例えば、高優先度アプリケーションに割り当てられたウェイ数を、動作５２４の繰り返し実行につながるような、低優先度アプリケーションの繰り返し、積極的、および完全なエビクションを低減するように、低減可能である。ＣＣＣが動作５１０において要求元占有数が０ではないと判定すると、ＣＣＣは動作５１４において、要求元優先度の、最も長く使われていないＣＬをエビクションするものとする。

【００６１】

動作５０８に戻ると、ＣＣＣは要求元優先度が最低優先度ではないと判定すると、ＣＣＣは動作５１２において、要求元優先度の占有数が最大かを判定するものとする。最大であれば、ＣＣＣは動作５１４において、要求元優先度の、最も長く使われていないＣＬをエビクションするものとする。動作５１５において格納場所が発見されるとフローは終了する。

【００６２】

動作５１２に戻ると、ＣＣＣは要求元優先度の占有数が最大でないと判定すると、ＣＣＣは動作５１６において、要求元優先度の占有数が最大未満で、要求元優先度に対する最小以上であるかを判定する。そうであれば、フローは動作５１８に移行し、そうでなければ、フローは動作５２０に移行する。動作５１８において、ＣＣＣは要求元優先度（$Ｐ_Ｒ$）またはより低い優先度（$Ｐ_Ｌ$）に等しい優先度を有するＬＲＵ　ＣＬをエビクションするものとし、５１９において格納場所が発見されるとフローは終了する。動作５２０において、ＣＣＣは要求元優先度よりも低い優先度のＬＲＵ　ＣＬのエビクションを試みる。そのようなラインがあれば、５２２での格納場所はＮＵＬＬ以外と判定され、フローは５２３に移行し、ここで格納場所が発見されるとフローは終了する。そのようなＣＬがなけ

れば、５２２での格納場所はＮＵＬＬに等しくなり、フローは動作５２４に移行し、要求元優先度よりも高い優先度を有するＬＲＵ　ＣＬをエビクションする。そして、５２５において格納場所が発見されるとフローは終了する。

【００６３】

図６は、いくつかの実施形態に係る、キャッシュフィル要求を処理するキャッシュ制御回路（ＣＣＣ）により実行される方法を示すフロー図である。例えば、フロー６００は図２のキャッシュ制御回路（ＣＣＣ）２０１により実行可能である。図示のように、動作６０５においてＣＣＣは、複数の優先度のうちのある要求元優先度を有する後続キャッシュライン（ＣＬ）を、ラストレベルキャッシュ（ＬＬＣ）に格納する要求を受信するものとする。動作６１０において、ＬＬＣに無効キャッシュライン（ＣＬ）が存在すると、後続キャッシュライン（ＣＬ）は無効ＣＬに格納される。動作６１５において、要求元優先度が、複数の優先度のうち最低で、１または複数である占有数を有する、または、要求元優先度に対して占有数が最大である場合、要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに、後続ＣＬが格納される。動作６２０において、占有数が要求元優先度について最大と最小の間である場合、要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬが格納される。動作６２５において、占有数が最小未満で、より低い優先度を有するＣＬが存在する場合、より低い優先度を有するＬＲＵ　ＣＬの代わりに後続ＣＬが格納される。動作６３０において、無効ＣＬ、または要求元優先度またはそれより低い優先度を有するＣＬが存在しない場合、より高い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬが格納される。

［命令セット］

【００６４】

命令セットは、１または複数の命令のフォーマットを含んでよい。所与の命令形式は、とりわけ、行われるべき演算（例えば、オペコード）およびその演算が行われるべきオペランド（複数可）を指定するための様々なフィールド（例えば、ビット数、ビットの位置）および／または他のデータフィールド（複数可）（例えば、マスク）を決定し得る。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）の定義により更に分類される。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、特定のオペコードなら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／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を有することになる。アドバンストベクトル拡張（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一連のＳＩＭＤ拡張機能がリリースおよび／または公開されている（例えば、２０１４年９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ならびに２０１４年１０月のインテル（登録商標）アドバンストベクトル拡張プログラミングリファレンスを参照）。

［例示的な命令フォーマット］

【００６５】

本明細書で説明される命令の実施形態は、異なるフォーマットで具現化されてよい。さらに、例示的システム、アーキテクチャ、およびパイプラインが以下に詳述される。命令の複数の実施形態は、このような複数のシステム、複数のアーキテクチャおよび複数のパイプライン上で実行されてよいが、これらの詳細に限定されるものではない。

［汎用ベクトル向け命令フォーマット］

【００６６】

ベクトル向け命令フォーマットとは、ベクトル命令に適した命令フォーマットのことである（例えば、ベクトル演算に特有の特定のフィールドが存在する）。ベクトル演算およびスカラ演算の両方を、ベクトル向け命令フォーマットを介してサポートする実施形態を説明するが、代替として、ベクトル向け命令フォーマットによりベクトル演算のみを用いる実施形態もある。

【００６７】

図７Ａおよび図７Ｂは、本発明のいくつかの実施形態に係る、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７Ａは、本発明のいくつかの実施形態に係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７Ｂは、本発明のいくつかの実施形態に係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、クラスＡおよびクラスＢ命令テンプレートに対して定義される汎用ベクトル向け命令フォーマット７００は、どちらのクラスも、非メモリアクセス７０５の命令テンプレートおよびメモリアクセス７２０の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いて汎用という用語は、いかなる特定の命令セットにも結び付けられていない命令フォーマットに関連する。

【００６８】

本発明の実施形態が説明されるが、ここでベクトル向け命令フォーマットは以下のものをサポートする。つまり、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）を有する６４バイトベクトルオペランド長（またはサイズ）（したがって、６４バイトベクトルは、ダブルワードサイズの１６個の要素、または代わりにクワッドワードサイズの８個の要素で構成される）と、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）を有する６４バイトベクトルオペランド長（またはサイズ）と、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）を有する３２バイトベクトルオペランド長（またはサイズ）と、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）を有する１６バイトベクトルオペランド長（またはサイズ）である。代替的な実施形態は、より大きいデータ要素幅、より小さいデータ要素幅、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）を有する、より大きいベクトルオペランドサイズ、より小さいベクトルオペランドサイズ、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。

【００６９】

図７Ａ中のクラスＡ命令テンプレートは、１）非メモリアクセス７０５命令テンプレート内に示される、非メモリアクセスフルラウンド制御タイプ演算７１０命令テンプレート、および非メモリアクセスデータ変換タイプ演算７１５命令テンプレート、ならびに２）メモリアクセス７２０命令テンプレート内に示されるメモリアクセス一時的７２５命令テンプレート、およびメモリアクセス非一時的７３０命令テンプレートを含む。図７Ｂ中のクラスＢ命令テンプレートは、１）非メモリアクセス７０５命令テンプレート内に示される、非メモリアクセス書き込みマスク制御パーシャルラウンド制御タイプ演算７１２命令テンプレート、および非メモリアクセス書き込みマスク制御ＶＳＩＺＥタイプ演算７１７命令テンプレート、ならびに２）メモリアクセス７２０命令テンプレート内に示される、メモリアクセス書き込みマスク制御７２７命令テンプレートを含む。

【００７０】

汎用ベクトル向け命令フォーマット７００は、以下に示されるフィールドを図７Ａおよび図７Ｂに示される順序で含む。

【００７１】

フォーマットフィールド７４０－このフィールド内の特定の値（命令フォーマット識別子値）は、ベクトル向け命令フォーマット、したがって命令ストリーム内のベクトル向け命令フォーマット内の命令の出現を一意に識別する。したがって、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必要ないという意味で任意選択的である。

【００７２】

ベース演算フィールド７４２：その内容により、異なるベース演算を識別する。

【００７３】

レジスタインデックスフィールド７４４：その内容により、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がレジスタ内にあるか、メモリ内にあるかに関わらず、それらの位置を直接に、または、アドレス生成を通して指定する。これらは、Ｐ×Ｑ（例えば、３２×５１２、１６×１２８、３２×１０２４、６４×１０２４）個の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が、最大３個までのソースおよび１つのデスティネーションレジスタであり得るが、代替的な実施形態は、より多くのまたはより少ないソース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個までのソースをサポートしてよく、その場合、これらのソースの１つがデスティネーションとしても機能し、最大３個までのソースをサポートしてよく、その場合、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機能し、最大２個までのソースおよび１つのデスティネーションをサポートしてよい）。

【００７４】

修飾子フィールド７４６：その内容により、汎用ベクトル命令フォーマットにおける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命令から区別する。すなわち、非メモリアクセス７０５命令テンプレートと、メモリアクセス７２０命令テンプレートとの間を区別する。メモリアクセス動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、非メモリアクセス動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）。また、一実施形態において、このフィールドは、メモリアドレス計算を実行するために３つの別々の方法から選択をするが、代替的な実施形態によっては、メモリアドレス計算の実行のために、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。

【００７５】

拡張演算フィールド７５０：その内容により、ベース演算に加え、様々な異なる演算のうちのいずれが実行されるかを区別する。このフィールドは、コンテキストに特有のものである。いくつかの実施形態において、このフィールドは、クラスフィールド７６８、アルファフィールド７５２およびベータフィールド７５４に分割される。拡張演算フィールド７５０は、共通の演算グループを、２つ、３つ、または４つの命令ではなく、単一の命令で行うことを可能にする。

【００７６】

スケールフィールド７６０：その内容により、メモリアドレス生成のために（例えば、２ｓｃａｌｅ＊インデックス＋ベースを使用するアドレス生成のために）インデックスフィールドの内容のスケーリングを可能にする。

【００７７】

変位フィールド７６２Ａ：その内容は、メモリアドレス生成（例えば、２ｓｃａｌｅ＊インデックス＋ベース＋変位を用いるアドレス生成のための）の一部として用いられる。

【００７８】

変位係数フィールド７６２Ｂ（変位係数フィールド７６２Ｂの直接上に変位フィールド７６２Ａが併置されていることは、一方または他方が用いられることを示すことに留意されたい）：その内容は、アドレス生成の一部として用いられ、それは、メモリアクセス（Ｎ）のサイズによりスケーリングされるべき変位係数を指定する。ここで、Ｎは、メモリアクセスにおけるバイト数である（例えば、２ｓｃａｌｅ＊インデックス＋ベース＋スケー

リングされた変位を用いるアドレス生成のための）。冗長下位ビットは無視され、したがって、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７７４（後に本明細書で説明される）およびデータ操作フィールド７５４Ｃに基づいて、実行時にプロセッサハードウェアによって決定される。変位フィールド７６２Ａおよび変位係数フィールド７６２Ｂは、非メモリアクセス７０５の命令テンプレートに使用されない、および／または異なる実施形態は、これら２つのうちの１つだけを実装することができる、または全く実装しない場合がある、という意味で任意選択的である。

【００７９】

データ要素幅フィールド７６４：その内容により、複数のデータ要素幅のいずれが用いられるかを区別する（いくつかの実施形態においては、すべての命令について、他の実施形態においては、命令のうちの一部のみについて）。このフィールドは、ただ１つのデータ要素幅がサポートされるおよび／またはデータ要素幅がオペコードのいくらかの態様を使用してサポートされる場合に、それが必要とされないという意味において任意選択的である。

【００８０】

書き込みマスクフィールド７７０：その内容により、データ要素位置ベースごとに、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否かを制御する。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化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、（ベース演算および拡張演算によって指定される）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。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、ゼロ化の場合、ベクトルマスクは、（ベース演算および拡張演算によって指定される）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化されることを可能にする。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。この機能のサブセットは、実行される動作のベクトル長（すなわち、最初のものから最後のものまで、要素が変更されるスパン）を制御する能力であるが、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７７０は、ロード、格納、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７７０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７７０の内容により、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、書き込みマスクフィールド７７０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。

【００８１】

即値フィールド７７２：その内容により、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では存在せず、即値を用いない複数の命令に存在しないという意味で、任意選択的である。

【００８２】

クラスフィールド７６８：その内容により、異なるクラスの命令間を区別する。図７Ａおよび図７Ｂを参照すると、このフィールドの内容は、クラスＡ命令およびクラスＢ命令間を選択する。図７Ａおよび図７Ｂにおいて、角に丸みのある四角は、特定の値がフィールドに存在することを示すために使用される（例えば、図７Ａおよび図７Ｂにおいて、クラスフィールド７６８について、それぞれ、クラスＡ　７６８ＡおよびクラスＢ　７６８Ｂ）。

［クラスＡの命令テンプレート］

【００８３】
クラスＡの非メモリアクセス７０５命令テンプレートの場合、アルファフィールド７５２はＲＳフィールド７５２Ａとして解釈され、その内容により、異なる拡張演算タイプのうちのいずれが実行されるか（例えば、ラウンド７５２Ａ．１およびデータ変換７５２Ａ．２が、非メモリアクセスラウンドタイプ演算７１０および非メモリアクセスデータ変換タイプ演算７１５命令テンプレートのそれぞれに対し指定される）を区別する一方、ベータフィールド７５４により、指定されたタイプの演算のうちのいずれが実行されるかを区別する。非メモリアクセス７０５命令テンプレートには、スケールフィールド７６０、変位フィールド７６２Ａおよび変位係数フィールド７６２Ｂは存在しない。

［非メモリアクセス命令テンプレート－フルラウンド制御タイプ演算］

【００８４】
非メモリアクセスフルラウンド制御タイプ演算７１０命令テンプレートでは、ベータフィールド７５４がラウンド制御フィールド７５４Ａとして解釈され、その内容が静的ラウンディングを提供する。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７５４Ａは、すべての浮動小数点の例外を抑制（Ｓｕｐｐｒｅｓｓ　Ａｌｌ　ｆｌｏａｔｉｎｇ－ｐｏｉｎｔ　Ｅｘｃｅｐｔｉｏｎｓ：ＳＡＥ）フィールド７５６およびラウンド演算制御フィールド７５８を含み、一方で、代替的な実施形態は、これら両方の概念を同一フィールドに符号化してよく、または代替的な実施形態はこれらの概念／フィールドのうちの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７５８のみを有してよい）。

【００８５】
ＳＡＥフィールド７５６：その内容により、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７５６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。

【００８６】
ラウンド演算制御フィールド７５８：その内容により、一群のラウンド演算（例えば、切り上げ、切り捨て、ゼロへのラウンド、および近似値へのラウンド）のうちのいず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７５８は、命令単位で、ラウンドモードの変更を可能にする。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のいくつかの実施形態において、ラウンド演算制御フィールド７５０の内容で、そのレジスタ値を上書きする。

［非メモリアクセス命令テンプレート－データ変換タイプ演算］

【００８７】
非メモリアクセスのデータ変換タイプ演算７１５命令テンプレートでは、ベータフィールド７５４はデータ変換フィールド７５４Ｂとして解釈され、データ変換フィールド７５４Ｂの内容により、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）のうちいずれが実行されるかを区別する。

【００８８】
クラスＡのメモリアクセス７２０命令テンプレートの場合において、アルファフィールド７５２は、エビクションヒントフィールド７５２Ｂとして解釈され、その内容により、エビクションヒントのうちのいずれが用いられるべきかを区別し（図７Ａにおいて、一時的７５２Ｂ．１および非一時的７５２Ｂ．２が、それぞれ、メモリアクセス一時的７２５命令テンプレートおよびメモリアクセス非一時的７３０命令テンプレートについて指定される）、一方でベータフィールド７５４は、データ操作フィールド７５４Ｃとして解釈され、その内容により、複数のデータ操作の動作（プリミティブとしても知られる）のうちいずれが実行されるか（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）を区別する。メモリアクセス７２０命令テンプレートは、スケールフィールド７６０を含み、任意で変位フィールド７６２Ａまたは変位係数フィールド７６２Ｂを含む。

【００８９】
ベクトルメモリ命令は、変換サポートにより、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。
［メモリアクセス命令テンプレート：一時的］
【００９０】
一時的データとは、キャッシングによる恩恵を得るのに十分早く再使用される可能性が高いデータである。しかしながら、これはヒントであり、異なるプロセッサが、ヒントを完全に無視することを含む、異なる方法でそれを実装することができる。
［メモリアクセス命令テンプレート：非一時的］
【００９１】
非一時的データとは、第１レベルキャッシュ内にキャッシュすることによる恩恵を得るのに十分早く再使用される可能性が低いデータであり、これにはエビクションの優先度が与えられるべきである。しかしながら、これはヒントであり、異なるプロセッサが、ヒントを完全に無視することを含む、異なる方法でそれを実装することができる。
［クラスＢの命令テンプレート］
【００９２】
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７５２は、書き込みマスク制御（Ｚ）フィールド７５２Ｃとして解釈され、書き込みマスク制御（Ｚ）フィールド７５２Ｃの内容により、書き込みマスクフィールド７７０により制御される書き込みマスキングが、マージングであるべきかゼロ化であるべきかを区別する。
【００９３】
クラスＢの非メモリアクセス７０５命令テンプレートの場合、ベータフィールド７５４の一部はＲＬフィールド７５７Ａとして解釈され、その内容により、異なる拡張演算タイプのうちのいずれが実行されるかを区別する（例えば、ラウンド７５７Ａ．１およびベクトル長（ＶＳＩＺＥ）７５７Ａ．２が、非メモリアクセス書き込みマスク制御パーシャルラウンド制御タイプ演算７１２命令テンプレートおよび非メモリアクセス書き込みマスク制御ＶＳＩＺＥタイプ演算７１７命令テンプレートにそれぞれ指定される）一方、ベータフィールド７５４の残部により、指定されたタイプの演算のうちいずれが実行されるかを区別する。非メモリアクセス７０５命令テンプレートには、スケールフィールド７６０、変位フィールド７６２Ａおよび変位係数フィールド７６２Ｂは存在しない。
【００９４】
非メモリアクセス書き込みマスク制御パーシャルラウンド制御タイプ演算７１２命令テンプレートでは、ベータフィールド７５４の残部はラウンド演算フィールド７５９Ａ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）。
【００９５】
ラウンド演算制御フィールド７５９Ａ：ラウンド演算制御フィールド７５８と同様に、その内容により、一群のラウンド演算のうちのいずれが実行されるかを区別する（例えば、切り上げ、切り捨て、ゼロへのラウンド、および近似値へのラウンド）。故に、ラウンド演算制御フィールド７５９Ａは、命令単位で、ラウンドモードの変更を可能にする。いくつかの実施形態において、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７５０の内容で、そのレジスタ値を上書きする。
【００９６】
非メモリアクセス書き込みマスク制御ＶＳＩＺＥタイプ演算７１７命令テンプレートでは、ベータフィールド７５４の残部はベクトル長フィールド７５９Ｂとして解釈され、ベクトル長フィールド７５９Ｂの内容により、複数のデータベクトル長（例えば、１２８、２５６または５１２バイト）のうちのいずれで実行されるかを区別する。
【００９７】

クラスＢのメモリアクセス７２０命令テンプレートの場合において、ベータフィールド７５４の一部は、ブロードキャストフィールド７５７Ｂとして解釈され、その内容により、ブロードキャストタイプデータ操作の動作が実行されるか否かを区別し、一方でベータフィールド７５４の残部は、ベクトル長フィールド７５９Ｂとして解釈される。メモリアクセス７２０命令テンプレートは、スケールフィールド７６０を含み、任意で変位フィールド７６２Ａまたは変位係数フィールド７６２Ｂを含む。

【００９８】

汎用ベクトル向け命令フォーマット７００に関しては、フルオペコードフィールド７７４は、フォーマットフィールド７４０、ベース演算フィールド７４２およびデータ要素幅フィールド７６４を含むように示されている。フルオペコードフィールド７７４がこれらのフィールドのすべてを含む一実施形態が示されるが、該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、フルオペコードフィールド７７４は、該フィールドのすべてよりも少ないフィールドを含む。フルオペコードフィールド７７４は、演算コード（オペコード）を提供する。

【００９９】

拡張演算フィールド７５０、データ要素幅フィールド７６４、および書き込みマスクフィールド７７０は、これらの特徴が汎用ベクトル向け命令フォーマットにおいて命令ごとの単位で指定されることを可能とする。

【０１００】

書き込みマスクフィールドとデータ要素幅フィールドの組み合わせは、それらが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という点で、型付き命令を形成する。

【０１０１】

クラスＡおよびクラスＢ内で見つか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汎用アウトオブオーダコアはクラス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（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の何らかの組み合わせを有するものの、両方のクラスのすべてのテンプレートおよび命令を有してはいないコアは、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有するプロセッサにおいて、主にグラフィックおよび／または科学コンピューティング用のグラフィックコアのうちの１つは、クラスＡのみをサポートし得るが、汎用コアのうちの１または複数は、クラスＢのみをサポートする汎用コンピューティング用のアウトオブオーダ実行およびレジスタリネームを用いる高性能汎用コアであり得る。別個のグラフィックコアを持た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以上の汎用インオーダまたはアウトオブオーダコアを含んでよい。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複数のプログラムは、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）全クラスの複数の命令の異なる組み合わせを用いて記述される代替的な複数のルーチンを有し、コードを現在実行しているプロセッサによってサポートされる複数の命令に基づいて実行する、複数のルーチンを選択する制御フローコードを有する形式を含む、様々な異なる実行可能な形式にされる（例えば、ジャストインタイムでコンパイルされ、または静的にコンパイルされる）。

［例示的な特定ベクトル向け命令フォーマット］

【０１０２】

図８Ａは、本発明のいくつかの実施形態に係る、例示的な特定ベクトル向け命令フォーマットを示すブロック図である。図８Ａは、特定ベクトル向け命令フォーマット８００を示し、これは位置、サイズ、解釈、およびフィールドの順序、ならびにこれらのフィールドのいくつかに対する値を指定するという点で特定のものである。特定ベクトル向け命令フォーマット８００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符号化フィールド、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８Ａからのフィールドがマッピングされる図７Ａまたは図７Ｂからのフィールドが例示される。

【０１０３】

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７００に照らし特定ベクトル向け命令フォーマット８００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８００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８００は特定のサイズのフィールドを有するように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７００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。特定の実施例として、データ要素幅フィールド７６４は、特定ベクトル向け命令フォーマット８００で１ビットフィールドとして示されるが、本発明を限定するものではない（すなわち、汎用ベクトル向け命令フォーマット７００は、他のサイズのデータ要素幅フィールド７６４を想定している）。

【０１０４】

特定ベクトル向け命令フォーマット８００は、図８Ａに示された順序で下記に列挙された以下のフィールドを含む。

【０１０５】

ＥＶＥＸプレフィクス（バイト０－３）８０２。これは４バイト形式で符号化される。

【０１０６】

フォーマットフィールド７４０（ＥＶＥＸバイト０、ビット［７：０］）。第１のバイト（ＥＶＥＸバイト０）はフォーマットフィールド７４０であり、フォーマットフィールド７４０は０×６２を含む（いくつかの実施形態において、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）。

【０１０７】

第２～第４バイト（ＥＶＥＸ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。

【０１０８】

ＲＥＸフィールド８０５（ＥＶＥＸバイト１、ビット［７－５］）：ＥＶＥＸ．Ｒビットフィールド（ＥＶＥＸバイト１、ビット［７］－Ｒ）、ＥＶＥＸ．Ｘビットフィールド（ＥＶＥＸバイト１、ビット［６］－Ｘ）、およびＥＶＥＸ．Ｂビットフィールド（ＥＶＥＸバイト１、ビット［５］－Ｂ）で構成される。ＥＶＥＸ．Ｒビットフィールド、ＥＶＥＸ．ＸビットフィールドおよびＥＶＥＸ．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符号化され、すなわちＺＭＭ０は１１１１Ｂとして符号化され、ＺＭＭ１５は０００００Ｂとして符号化される。命令の他のフィールドは、当技術分野で知られているように、レジスタインデックスの下位３ビット（ｒｒｒ、ｘｘｘ、およびｂｂｂ）を符号化するので、ＥＶＥＸ．Ｒ、ＥＶＥＸ．Ｘ、およびＥＶＥＸ．Ｂを追加することによって、Ｒｒｒｒ、Ｘｘｘｘ、およびＢｂｂｂを形成することができる。

【０１０９】

ＲＥＸ'８１０Ａ：これはＲＥＸ'フィールド８１０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のいずれかを符号化するために使用されるＥＶＥＸ

．Ｒ'ビットフィールド（ＥＶＥＸバイト１、ビット［４］－Ｒ'）である。いくつかの実施形態において、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。下位１６個のレジスタを符号化するのに値１が使用される。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'Ｒｒｒｒが形成される。

【０１１０】

オペコードマップフィールド８１５（ＥＶＥＸバイト１、ビット［３：０］－ｍｍｍｍ）：その内容により、示唆される先頭オペコードバイト（０Ｆ、０Ｆ３８、または０Ｆ３）を符号化する。

【０１１１】

データ要素幅フィールド７６４（ＥＶＥＸバイト２、ビット［７］－Ｗ）は、ＥＶＥＸ．Ｗという表記で表される。ＥＶＥＸ．Ｗはデータタイプ（３２ビットデータ要素または６４ビットデータ要素）の粒度（サイズ）を規定するために使用される。

【０１１２】

ＥＶＥＸ．ｖｖｖｖフィールド８２０（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）形式に符号化し、ＥＶＥＸ．ｖｖｖｖは２またはそれより多い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、特定のベクトルシフト用の指定された１の補数形式に符号化する。または３）ＥＶＥＸ．ｖｖｖｖはいずれのオペランドも符号化せず、当該フィールドは予約され、１１１１ｂを含むべきである。したがって、ＥＶＥＸ．ｖｖｖｖフィールド８２０は、反転形式（１の補数）で格納された第１のソースレジスタ指定子の４つの下位ビットを符号化する。命令に応じて、追加の異なるＥＶＥＸビットフィールドを使用して、指定子サイズを３２個のレジスタに拡張する。

【０１１３】

ＥＶＥＸ．Ｕ　７６８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。

【０１１４】

プレフィクス符号化フィールド８２５（ＥＶＥＸバイト２、ビット［１：０］－ｐｐ）：ベース演算フィールドの追加的なビットを提供する。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プレフィクス符号化フィールドに符号化される。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プレフィクス符号化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させるが、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プレフィクス符号化をサポートするようにＰＬＡを再設計することができ、したがって、拡張を必要としない。

【０１１５】

アルファフィールド７５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ。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と示される）：先に記載したように、このフィールドはコンテキストに特有のものである。

【０１１６】

ベータフィールド７５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ。ＥＶＥＸ．S_{2-0}、ＥＶＥＸ．r_{2-0}、ＥＶＥＸ．ｒｒ１、ＥＶＥＸ．ＬＬ０、ＥＶＥＸ．ＬＬＢとしても知られる。またβββと示される）：先に記載したように、このフィールドは、コンテキストに特有のものである。

【０１１７】

ＲＥＸ'８１０Ｂ。これはＲＥＸ'フィールド８１０の残部であり、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個のいずれかを符号化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。下位１６個のレジスタを符号化するのに値１が使用される。換言すると、ＥＶＥＸ．Ｖ'、ＥＶＥＸ．ｖｖｖｖを組み合わせることにより、Ｖ'ＶＶＶＶが形成される。

【０１１８】

書き込みマスクフィールド７７０（ＥＶＥＸバイト３、ビット［２：０］－ｋｋｋ）：その内容により、前述のとおり、書き込みマスクレジスタにおけるレジスタのインデックスを指定する。いくつかの実施形態において、特定の値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）。

【０１１９】

リアルオペコードフィールド８３０（バイト４）は、オペコードバイトとしても知られる。オペコードの一部が、このフィールドにおいて指定される。

【０１２０】

ＭＯＤ　Ｒ／Ｍフィールド８４０（バイト５）は、ＭＯＤフィールド８４２、Ｒｅｇフィールド８４４およびＲ／Ｍフィールド８４６を含む。上記のとおり、ＭＯＤフィールド８４２の内容により、メモリアクセス動作および非メモリアクセス動作間を区別する。Ｒｅｇフィールド８４４の役割は、デスティネーションレジスタオペランドもしくはソースレジスタオペランドのいずれかを符号化すること、または、オペコード拡張として扱われ、命令オペランドを符号化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８４６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符号化すること、またはデスティネーションレジスタオペランドもしくはソースレジスタオペランドのいずれかを符号化することが含まれてよい。

【０１２１】

スケール、インデックス、ベース（ＳＩＢ）バイト８５０（バイト６）は、スケールに関するＳＳ８５２を含む。先に記載したように、スケールフィールド７６０は、メモリアドレス生成に使用される。ＳＩＢ．ｘｘｘ８５４およびＳＩＢ．ｂｂｂ８５６。これらのフィールドの内容は、レジスタインデックスＸｘｘｘおよびＢｂｂｂに関して記載済みである。

【０１２２】

変位フィールド７６２Ａ（バイト７－１０）：ＭＯＤフィールド８４２が１０を含む場合、バイト７－１０は、変位フィールド７６２Ａであり、これはレガシ３２ビット変位（ｄｉｓｐ３２）と同じく機能し、バイト粒度で機能する。

【０１２３】

変位係数フィールド７６２Ｂ（バイト７）：ＭＯＤフィールド８４２が０１を含む場合、バイト７が変位係数フィールド７６２Ｂである。このフィールドの位置は、バイト粒度で機能するレガシｘ８６命令セットの８ビット変位（ｄｉｓｐ８）のものと同じである。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７６２Ｂはｄｉｓｐ８の再解釈である。変位係数フィールド７６２Ｂ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、平均命令長（変位には単一のバイトが使用されるが、極めて大きい範囲を伴う）を低減させる。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、したがって、アドレスオフセットの冗長下位ビットは符号化の必要がないという前提に基づいている。換言すれば、変位係数フィールド７６２Ｂは、レガシｘ８６命令セットの８ビット変位を置換する。したがって、変位係数フィールド７６２Ｂは、ｄｉｓｐ８がｄｉｓｐ８＊Ｎにオーバーロードされることを唯一の例外として、ｘ８６命令セットの８ビット変位と同じ方法で符号化される（したがって、ＭｏｄＲＭ／ＳＩＢ符号化規則にはいかなる変更もない）。つまり、符号化規則または符号化長には変更がなく、ハードウェアによる変位値の解釈にのみ変更がある（ハードウェアはメモリオペランドのサイズにより変位をスケーリングしてバイト毎のアドレスオフセットを得る必要がある）。即値フィールド７７２は、上で説明したように演算を行う。

［フルオペコードフィールド］

【０１２４】

図８Ｂは、いくつかの実施形態に係る、特定ベクトル向け命令フォーマット８００のフルオペコードフィールド７７４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７７４は、フォーマットフィールド７４０と、ベース演算フィールド７４２と、データ要素幅（Ｗ）フィールド７６４とを含む。ベース演算フィールド７４２は、プレフィクス符号化フィールド８２５と、オペコードマップフィールド８１５と、リアルオペコードフィールド８３０とを含む。

［レジスタインデックスフィールド］

【０１２５】

図８Ｃは、いくつかの実施形態に係る、特定ベクトル向け命令フォーマット８００のレジスタインデックスフィールド７４４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７４４は、ＲＥＸフィールド８０５、ＲＥＸ'フィールド８１０、ＭＯＤＲ／Ｍ．ｒｅｇフィールド８４４、ＭＯＤＲ／Ｍ．ｒ／ｍフィールド８４６、ＶＶＶＶフィールド８２０、ｘｘｘフィールド８５４およびｂｂｂフィールド８５６を含む。

［拡張演算フィールド］

【０１２６】

図８Ｄは、本発明の一実施形態に係る、特定ベクトル向け命令フォーマット８００の拡張演算フィールド７５０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７６８が０を含む場合、それはＥＶＥＸ．Ｕ０（クラスＡ　７６８Ａ）を表す。クラス（Ｕ）フィールド７６８が１を含む場合、それはＥＶＥＸ．Ｕ１（クラスＢ　７６８Ｂ）を表す。Ｕ＝０でかつＭＯＤフィールド８４２が１１を含む場合（非メモリアクセス動作を意味する）、アルファフィールド７５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ）は、ｒｓフィールド７５２Ａとして解釈される。ｒｓフィールド７５２Ａが１を含む場合（ラウンド７５２Ａ．１）、ベータフィールド７５４（ＥＶＥＸバイト３、ビット［６：４］ＳＳＳ）はラウンド制御フィールド７５４Ａとして解釈される。ラウンド制御フィールド７５４Ａは、１ビットのＳＡＥフィールド７５６および２ビットのラウンド演算フィールド７５８を含む。ｒｓフィールド７５２Ａが０を含む場合（データ変換７５２Ａ．２）、ベータフィールド７５４（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７５４Ｂとして解釈される。Ｕ＝０でかつＭＯＤフィールド８４２が００、０１または１０を含む場合（メモリアクセス動作を意味する）、アルファ

フィールド７５２（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７５２Ｂとして解釈され、ベータフィールド７５４（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７５４Ｃとして解釈される。

【０１２７】

Ｕ＝１である場合、アルファフィールド７５２（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７５２Ｃとして解釈される。Ｕ＝１でかつＭＯＤフィールド８４２が１１を含む場合（非メモリアクセス動作を意味する）、ベータフィールド７５４の一部（ＥＶＥＸバイト３、ビット［４］$Ｓ_0$）は、ＲＬフィールド７５７Ａとして解釈される。ＲＬフィールド７５７Ａが１を含む場合（ラウンド７５７Ａ．１）、ベータフィールド７５４の残部（ＥＶＥＸバイト３、ビット［６－５］$Ｓ_{2-1}$）はラウンド演算フィールド７５９Ａとして解釈され、一方で、ＲＬフィールド７５７Ａが０を含む場合（ＶＳＩＺＥ　７５７．Ａ２）、ベータフィールド７５４の残部（ＥＶＥＸバイト３、ビット［６－５］$Ｓ_{2-1}$）は、ベクトル長フィールド７５９Ｂ（ＥＶＥＸバイト３、ビット［６－５］$Ｌ_{1-0}$）として解釈される。Ｕ＝１でかつＭＯＤフィールド８４２が００、０１または１０を含む場合（メモリアクセス動作を意味する）、ベータフィールド７５４（ＥＶＥＸバイト３、ビット［６：４］ＳＳＳ）は、ベクトル長フィールド７５９Ｂ（ＥＶＥＸバイト３、ビット［６－５］$Ｌ_{1-0}$）およびブロードキャストフィールド７５７Ｂ（ＥＶＥＸバイト３、ビット［４］Ｂ）として解釈される。

［例示的レジスタアーキテクチャ］

【０１２８】

図９は、いくつかの実施形態に係る、レジスタアーキテクチャ９００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９１０がある。これらのレジスタは、ｚｍｍ０からｚｍｍ３１として参照される。下位１６個のｚｍｍレジスタの下位２５６ビットは、レジスタｙｍｍ０～１５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８００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２】

調整可能ﾍﾞｸﾄﾙ長	クラス	演算	レジスタ
ベクトル長フィールド 759B を含まない命令テンプレート	A(図 7A;U=0)	710,715, 725,730	zmm レジスタ(ベクトル長=64 バイト)
	B(図 7B;U=1)	712	zmm レジスタ(ベクトル長=64 バイト)
ベクトル長フィールド 759B を含む命令テンプレート	B(図 7B;U=1)	717,727	ベクトル長フィールド 759B に応じて zmm、ymm、または xmm レジスタ (ベクトル長=64 バイト、32 バイト、または 16 バイト)

【０１２９】

換言すると、ベクトル長フィールド７５９Ｂは、最大長と、１または複数の他のより短い長さとの間で選択し、このようなより短い長さの各々は先行する長さの半分の長さであり、ベクトル長フィールド７５９Ｂのない命令テンプレートは、最大ベクトル長に対し、演算を行う。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８００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、演算を行う。スカラ演算は、ｚｍｍ／ｙｍｍ／ｘｍｍレジスタ内の最下位データ要素位

置において実行される演算であり、上位のデータ要素位置は、実施形態に応じて、命令の前と同じ状態のままにされるかまたはゼロ化される。

【０１３０】

図示された実施形態中の書き込みマスクレジスタ９１５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９１５は、１６ビットのサイズである。上記のとおり、いくつかの実施形態において、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を示す符号化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キングを無効にする。

【０１３１】

図示された実施形態中の汎用レジスタ９２５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からＲ１５の名前で参照される。

【０１３２】

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）９４５について、ＭＭＸパックド整数フラットレジスタファイル９５０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。

【０１３３】

代替的な実施形態では、より広いまたはより狭いレジスタを使用し得る。加えて、代替的な実施形態では、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も使用し得る。

［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］

【０１３４】

プロセッサコアは、異なる方法で、異なる目的のために、異なるプロセッサにおいて実装されてよい。例えば、そのようなコアの実装には、以下が含まれ得る。１）汎用コンピューティングを目的とした汎用インオーダコア。２）汎用コンピューティングを目的とした高性能汎用アウトオブオーダコア。３）主としてグラフィックおよび／または科学（スループット）コンピューティングを目的とした専用コア。異なるプロセッサの実装には、以下が含まれ得る。１）汎用コンピューティングを目的とした１または複数の汎用インオーダコアおよび／または汎用コンピューティングを目的とした１または複数の汎用アウトオブオーダコアを含むＣＰＵ、および２）主にグラフィックおよび／または科学（スループット）を目的とした１または複数の専用コアを含むコプロセッサ。そ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、それらには以下が含まれ得る。１）ＣＰＵとは別のチップ上のコプロセッサ。２）ＣＰＵと同じパッケージ内の別のダイ上のコプロセッサ。３）ＣＰＵと同じダイ上のコプロセッサ（この場合、そのようなコプロセッサは、統合グラフィックおよび／もしくは科学（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もある）。４）同じダイ上に記載のＣＰＵ（アプリケーションコア（複数可）またはアプリケーションプロセッサ（複数可）と呼ばれることもある）、上述したコプロセッサ、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説明され、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の説明が続く。

［例示的なコアアーキテクチャ］

［インオーダコアおよびアウトオブオーダコアのブロック図］

【０１３５】

図１０Ａは、本発明のいくつかの実施形態に係る、例示的なインオーダ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１０Ｂは、本発明のいくつかの実施形態に係る、プロセッサに含まれる、インオーダアーキテクチャコアの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１０Ａおよび図１０Ｂの実線で示されたボックスは、インオーダパイプラインおよびインオーダコアを図示する。一方、破線で示されたボックスの任意の追加は、レジスタリネーミング、アウトオブオーダ発行／実行パイプラインおよびコアを図示する。インオーダの態様がアウトオブオーダの態様のサブセットであることから、アウトオブオーダの態様を説明する。

【０１３６】

図１０Ａでは、プロセッサパイプライン１０００が、フェッチ段１００２、長さ復号段１００４、復号段１００６、割り当て段１００８、リネーミング段１０１０、スケジューリング（配付または発行としても知られる）段１０１２、レジスタ読み出し／メモリ読み出し段１０１４、実行段１０１６、ライトバック／メモリ書き込み段１０１８、例外処理段１０２２、およびコミット段１０２４を含む。

【０１３７】

図１０Ｂは、実行エンジンユニット１０５０に結合されたフロントエンドユニット１０３０を含むプロセッサコア１０９０を示し、両方ともメモリユニット１０７０に結合されている。コア１０９０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、あるいはハイブリッドまたは代替的なコアタイプであってよい。さらに別の選択肢として、コア１０９０は、例えば、ネットワークまたは通信コア、圧縮エンジン、コプロセッサコア、汎用コンピューティンググラフィック処理装置（ＧＰＧＰＵ）コア、グラフィックコア等の専用コアであり得る。

【０１３８】

フロントエンドユニット１０３０は、命令キャッシュユニット１０３４に結合された分岐予測ユニット１０３２を含み、命令キャッシュユニット１０３４は命令トランスレーションルックアサイドバッファ（ＴＬＢ）１０３６に結合され、命令トランスレーションルックアサイドバッファ（ＴＬＢ）１０３６は命令フェッチユニット１０３８に結合され、命令フェッチユニット１０３８は復号ユニット１０４０に結合されている。復号ユニット１０４０（すなわちデコーダ）は命令を復号してよく、また、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復号され、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。復号ユニット１０４０は、様々な異なるメカニズムを用いて実装され得る。好適なメカニズムの例としては、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）等が含まれるが、これらに限定されるものではない。一実施形態において、コア１０９０は、マイクロコードＲＯＭ、または特定のマクロ命令用のマイクロコードを格納する他の媒体を（例えば、復号ユニット１０４０に、またはそうでなければフロントエンドユニット１０３０の中に）含む。復号ユニット１０４０は、実行エンジンユニット１０５０の中のリネーム／アロケータユニット１０５２に連結される。

【０１３９】

実行エンジンユニット１０５０は、リタイアメントユニット１０５４と、１または複数のスケジューラユニット１０５６のセットとに結合されたリネーム／アロケータユニット１０５２を含む。スケジューラユニット１０５６は、複数のリザベーションステーション、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す。スケジューラユニット（複数可）１０５６は、物理レジスタファイルユニット（複数可）１０５８に結合される。物理レジスタファイルユニット１０５８の各々は、１または複数の物理レジスタファイルを表し、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、スカラ整数、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、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１０５８は、複数の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。物理レジスタファイルユニット１０５８は、（例えば、リオーダ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して、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（以上、複数可）を使用して、複数のレジスタマップおよびレジスタプールを使用してなど）レジスタリネーミングおよびアウトオブオーダ実行が実装され得る様々な方法を示すためにリタイアメントユニット１０５４とオーバーラップしている。リタイアメントユニット１０５４および物理レジスタファイルユニット（複数可）１０５８は、実行クラスタ（複数可）１０６０に結合される。実行クラスタ１０６０には、１または複数の実行ユニット１０６２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１０６４のセットが含まれる。実行ユニット１０６２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、特定の機能または機能のセットに専用の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、それらすべてが全機能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１０５６、物理レジスタファイルユニット１０５８および実行クラスタ１０６０が複数可として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（複数可）１０６４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。別々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または複数がアウトオブオーダ発行／実行であり、残りがインオーダであり得ることも理解されたい。

【０１４０】

複数のメモリアクセスユニット１０６４のセットがメモリユニット１０７０に結合され、メモリユニット１０７０は、レベル２（Ｌ２）キャッシュユニット１０７６に結合されたデータキャッシュユニット１０７４に結合されたデータＴＬＢユニット１０７２を含む。１つの例示的な実施形態において、メモリアクセスユニット１０６４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、これらはそれぞれ、メモリユニット１０７０のデータＴＬＢユニット１０７２に結合されている。命令キャッシュユニット１０３４は、メモリユニット１０７０のレベル２（Ｌ２）キャッシュユニット１０７６に更に結合されている。Ｌ２キャッシュユニット１０７６は、１または複数の他のレベルのキャッシュに結合され、最終的にはメインメモリに結合される。

【０１４１】

例として、例示的なレジスタリネーミング、アウトオブオーダ発行／実行コアアーキテクチャは、パイプライン１０００を次のように実装してよい。１）命令フェッチユニット１０３８が、フェッチ段１００２と、長さ復号段１００４とを実行し、２）復号ユニット１０４０が、復号段１００６を実行し、３）リネーム／アロケータユニット１０５２が、割り当て段１００８と、リネーミング段１０１０とを実行し、４）スケジューラユニット１０５６が、スケジュール段１０１２を実行し、５）物理レジスタファイルユニット１０５８およびメモリユニット１０７０が、レジスタ読み取り／メモリ読み取り段１０１４を実行し、実行クラスタ１０６０が実行段１０１６を実行し、６）メモリユニット１０７０および物理レジスタファイルユニット１０５８が、ライトバック／メモリ書き込み段１０

１８を実行し、７）様々なユニットが、例外処理段１０２２に関与してよく、かつ８）リタイアメントユニット１０５４および物理レジスタファイルユニット１０５８が、コミット段１０２４を実行する。

【０１４２】

コア１０９０は、本明細書で説明した命令を含む、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が追加されたいくつかの拡張がなされたもの）、カリフォルニア州サニーベールにあるＭＩＰＳ　ＴｅｃｈｎｏｌｏｇｉｅｓのＭＩＰＳ命令セット、カリフォルニア州サニーベールにあるＡＲＭ　ＨｏｌｄｉｎｇｓのＡＲＭ命令セット（ＮＥＯＮ等の任意の追加拡張機能が追加された））をサポートしてよい。一実施形態において、コア１０９０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、これによって、多くのマルチメディアアプリケーションによって用いられる複数の動作がパックドデータを用いて実行されることを可能にする。

【０１４３】

コアは、マルチスレッド化（演算またはスレッドの２つ以上の並列セットの実行）をサポートし、タイムスライスマルチスレッド化、（物理コアが同時にマルチスレッド化しているスレッドの各々に単一の物理コアが論理コアを提供する）同時マルチスレッド化、またはそれらの組み合わせ（例えば、インテル（登録商標）ハイパースレッディングテクノロジなどにおけるタイムスライスフェッチおよび復号ならびにその後の同時マルチスレッド化）を含む様々な方法でサポートすることができる。

【０１４４】

レジスタリネーミングはアウトオブオーダ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アーキテクチャで使用されてよいことを理解されたい。プロセッサの例示された実施形態はまた、別個の命令キャッシュユニット１０３４およびデータキャッシュユニット１０７４、ならびに共有Ｌ２キャッシュユニット１０７６を含み得るが、複数の代替的な実施形態が命令およびデータの両方用に、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等の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、内部キャッシュと、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もよい。あるいは、キャッシュのすべてがコアおよび／またはプロセッサの外部にあってよい。

［具体的な例示的インオーダコアアーキテクチャ］

【０１４５】

図１１Ａおよび図１１Ｂは、より具体的な例示のインオーダコアアーキテクチャのブロック図を示し、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１つであろう。論理ブロックは、アプリケーションに応じて、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）を通して、いくつかの固定機能論理、メモリＩ／Ｏインタフェース、および他の必要なＩ／Ｏ論理と通信する。

【０１４６】

図１１Ａは、本発明のいくつかの実施形態に係る、オンダイ相互接続ネットワーク１１０２への接続を備え、かつ、レベル２（Ｌ２）キャッシュのローカルサブセット１１０４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１１００はパックドデータ命令セット拡張を用いてｘ８６命令セットをサポートする。Ｌ１キャッシュ１１０６によって、キャッシュメモリに対して、スカラおよびベクトルユニット内部に低レイテンシのアクセスが可能になる。一実施形態において、（設計の単純化のために）スカラユニット１１０８およびベクトルユニット１１１０は、別個のレジスタセット（それぞれ、複数のスカラレジスタ１１１２および複数のベクトルレジスタ１１１４）を用い、これらの間で転送されるデータは、レベル１（Ｌ１）キャッシュ１１０６のメモリに書き込まれてから再読み出しされるが、本発明の複数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用いる、または書き込みおよび再読み

出しを行うことなく、２つのレジスタファイル間でのデータ転送を可能にする通信パスを含む）を用いてよい。

【０１４７】

Ｌ２キャッシュのローカルサブセット１１０４は、１つのプロセッサコアあたり１つずつ、別個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、独自のＬ２キャッシュのローカルサブセット１１０４に直接アクセスする経路を有する。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１１０４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１１０４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シを保証する。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、方向ごとに１０１２ビット幅である。

【０１４８】

図１１Ｂは、本発明のいくつかの実施形態に係る、図１１Ａのプロセッサコアの一部の拡大図である。図１１Ｂは、ベクトルユニット１１１０およびベクトルレジスタ１１１４に関するより詳細な点だけでなく、Ｌ１キャッシュ１１０６の一部であるＬ１データキャッシュ１１０６Ａを含む。具体的には、ベクトルユニット１１１０は、１６幅ベクトル処理ユニット（ＶＰＵ）（１６幅ＡＬＵ１１２８を参照）であり、整数命令、単精度浮動命令およ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１１２０を用いるレジスタ入力のスウィズル、数値変換ユニット１１２２Ａおよび１１２２Ｂを用いる数値変換およびメモリ入力での複製ユニット１１２４を用いる複製をサポートする。書き込みマスクレジスタ１１２６は、結果として生じるベクトル書き込みの予測を可能にする。

【０１４９】

図１２は、本発明のいくつかの実施形態に係る、プロセッサ１２００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１２の実線で示す箱はプロセッサ１２００を説明しており、単一コア１２０２Ａ、システムエージェントユニット１２１０、１または複数のバスコントローラユニット１２１６のセットを備える。破線で示す箱は任意の追加する別のプロセッサ１２００を説明しており、複数のコア１２０２Ａ～１２０２Ｎ、システムエージェントユニット１２１０内の１または複数の統合メモリコントローラユニット１２１４のセット、および専用ロジック１２０８を備える。

【０１５０】

よって、プロセッサ１２００の異なる実装としては、１）専用ロジック１２０８を統合グラフィックおよび／または科学（スループット）ロジック（これは、１または複数のコアを含んでよい）とし、コア１２０２Ａ～１２０２Ｎを１または複数の汎用コア（例えば、汎用インオーダコア、汎用アウトオブオーダコア、それら２つの組み合わせ）としたＣＰＵ、２）コア１２０２Ａ～１２０２Ｎをグラフィックおよび／または科学（スループット）を主な用途とする多数の専用コアとしたコプロセッサ、および３）コア１２０２Ａ～１２０２Ｎを多数の汎用インオーダコアとしたコプロセッサが挙げられてよい。故に、プロセッサ１２００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１２００は、例えば、ＢｉＣＭＯＳ、ＣＭＯＳ、またはＮＭＯＳ等の複数のプロセス技術のいずれかを用いて、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。
【０１５１】
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と、１または複数の共有キャッシュユニット１２０６のセットと、統合メモリコントローラユニット１２１４のセットに結合された外部メモリ（図示せず）とを含む。共有キャッシュユニット１２０６のセットは、レベル２（Ｌ２）、レベル３（Ｌ３）、レベル４（Ｌ４）、あるいは他のレベルのキャッシュ等の１または複数の中間レベルキャッシュ、ラストレベルキャッシュ（ＬＬＣ）、および／またはこ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１２１２は、統合グラフィックロジック１２０８（統合グラフィックロジック１２０８は専用ロジックの一例であり、本明細書で専用ロジックとも呼ばれる）、共有キャッシュユニット１２０６のセット、およびシステムエージェントユニット１２１０／統合メモリコントローラユニット１２１４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、コヒーレンシは、１または複数のキャッシュユニット１２０６とコア１２０２Ａ～１２０２Ｎとの間で維持される。
【０１５２】
いくつかの実施形態において、コア１２０２Ａ～１２０２Ｎのうち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ユニット１２１０は、コア１２０２Ａ～１２０２Ｎを調整し操作するこ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１２１０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もよい。ＰＣＵは、コア１２０２Ａ～１２０２Ｎおよび統合グラフィックロジック１２０８の電力状態を調整するのに必要とされるロジックおよび複数のコンポーネントであってよく、またはこれらを含んでもよい。ディスプレイユニットは、外部接続された１または複数のディスプレイを駆動するためのものである。
【０１５３】
コア１２０２Ａ～１２０２Ｎは、アーキテクチャ命令セットに関して同種または異種であってよい。すなわち、コア１２０２Ａ～１２０２Ｎのうちの２つ以上は、同じ命令セットの実行が可能であってよく、一方で他のものは、その命令セットのサブセットまたは異なる命令セットのみの実行が可能であってよい。
［例示的なコンピュータアーキテクチャ］
【０１５４】
図１３～図１６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、ネットワーク装置、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）、グラフィック装置、ビデオゲーム装置、セットトップボックス、マイクロコントローラ、携帯電話、携帯型メディアプレーヤ、ハンドヘルド装置、および他の様々な電子装置のための当技術分野において公知の他のシステム設計および構成も適している。一般的に、本明細書に開示されるプロセッサおよび／または他の実行ロジックを組み込むことができる多種多様なシステムまたは電子装置が一般に適している。
【０１５５】
ここで図１３を参照すると、本発明の１つの実施形態に係るシステム１３００のブロック図が示されている。システム１３００は、コントローラハブ１３２０に結合されている１または複数のプロセッサ１３１０、１３１５を含んでよい。一実施形態において、コントローラハブ１３２０は、グラフィックメモリコントローラハブ（ＧＭＣＨ）１３９０と、入出力ハブ（ＩＯＨ）１３５０（これは別のチップ上にあってよい）とを含み、ＧＭＣＨ１３９０は、メモリ１３４０およびコプロセッサ１３４５が結合されている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み、ＩＯＨ１３５０は、入出力（Ｉ／Ｏ）デバイス１３６０をＧＭＣＨ１３９０に結合する。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の一方または両方は、（本明細書で説明されるように）プロセ

ッサ内に統合され、メモリ１３４０およびコプロセッサ１３４５は、プロセッサ１３１０、およびＩＯＨ１３５０を有する単一のチップの中のコントローラハブ１３２０に直接結合される。

【０１５６】

追加的なプロセッサ１３１５の任意選択的な性質が、図１３において破線で示されている。各プロセッサ１３１０、１３１５は、本明細書で説明される複数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１２００の何らかのバージョンであってよい。

【０１５７】

メモリ１３４０は、例えば、ダイナミックランダムアクセスメモリ（ＤＲＡＭ）、相変化メモリ（ＰＣＭ）、またはその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は、コントローラハブ１３２０は、フロントサイドバス（ＦＳＢ）等のマルチドロップバス、ＱｕｉｃｋＰａｔｈ相互接続（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１３９５を介してプロセッサ１３１０、１３１５と通信する。

【０１５８】

一実施形態において、コプロセッサ１３４５は、例えば、高スループットＭＩＣプロセッサ、ネットワーク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、組み込みプロセッサ等の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１３２０は、統合グラフィックアクセラレータを含み得る。

【０１５９】

物理リソース１３１０、１３１５の間には、アーキテクチャ上のもの、マイクロアーキテクチャ上のもの、熱的なもの、および電力消費特性のもの等を含む様々な利点の基準に関して、多様な差異があり得る。

【０１６０】

一実施形態において、プロセッサ１３１０は、一般的タイプのデータ処理動作を制御する複数の命令を実行する。この命令内にコプロセッサ命令が組み込まれてよい。プロセッサ１３１０は、取り付けられたコプロセッサ１３４５により実行されるべきタイプのものとして、これらのコプロセッサ命令を認識する。したがって、プロセッサ１３１０は、コプロセッサバスまたは他の相互接続上で、これらのコプロセッサ命令（または複数のコプロセッサ命令を表す複数の制御信号）をコプロセッサ１３４５に発行する。コプロセッサ（複数可）１３４５は、受信したコプロセッサ命令を受け付け、実行する。

【０１６１】

ここで図１４を参照すると、本発明のある実施形態に従って、第１のより具体的な例示的システム１４００のブロック図が示されている。図１４に示されるように、マルチプロセッサシステム１４００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１４５０を介して結合される第１のプロセッサ１４７０および第２のプロセッサ１４８０を含む。プロセッサ１４７０および１４８０の各々は、いくつかのバージョンのプロセッサ１２００であり得る。本発明のいくつかの実施形態において、プロセッサ１４７０および１４８０は、それぞれプロセッサ１３１０および１３１５である一方で、コプロセッサ１４３８はコプロセッサ１３４５である。他の実施形態において、プロセッサ１４７０および１４８０は、それぞれ、プロセッサ１３１０およびコプロセッサ１３４５である。

【０１６２】

プロセッサ１４７０および１４８０は、それぞれ、統合メモリコントローラ（ＩＭＣ）ユニット１４７２および１４８２を含むものとして示されている。プロセッサ１４７０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１４７６および１４７８を含み、同様に、第２のプロセッサ１４８０はＰ－Ｐインタフェース１４８６および１４８８を含む。プロセッサ１４７０、１４８０は、ポイントツーポイント（Ｐ－Ｐ）インタフェース１４５０を介し、Ｐ－Ｐインタフェース回路１４７８、１４８８を用いて情報を交換してよい。図１４に図示のとおり、ＩＭＣ１４

７２および１４８２は、プロセッサをそれぞれのメモリ、すなわちメモリ１４３２およびメモリ１４３４に連結する。メモリ１４３２およびメモリ１４３４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。

【０１６３】

プロセッサ１４７０、１４８０はそれぞれ、個々のＰ－Ｐインタフェース１４５２、１４５４を介し、ポイントツーポイントインタフェース回路１４７６、１４９４、１４８６、１４９８を用いてチップセット１４９０と情報を交換してよい。チップセット１４９０は、任意で、高性能インタフェース１４９２を介してコプロセッサ１４３８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１４３８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、圧縮エンジン、グラフィックプロセッサ、ＧＰＧＰＵまたは組み込みプロセッサ等の専用プロセッサである。

【０１６４】

共有キャッシュ（図示せず）は、プロセッサ、またはＰ－Ｐ相互接続を介してプロセッサとすでに接続されている両方のプロセッサの外部のいずれかに含まれ、その結果、プロセッサが低電力モードに入れられた場合、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に格納されてよい。

【０１６５】

チップセット１４９０は、インタフェース１４９６を介して第１のバス１４１６に結合されてよい。一実施形態において、第１のバス１４１６は、周辺構成要素相互接続（ＰＣＩ）バス、またはＰＣＩ Ｅｘｐｒｅｓｓバスまたは別の第３世代Ｉ／Ｏ相互接続バス等のバスであり得るが、本発明の範囲は、そのようには限定されない。

【０１６６】

図１４に示されるように、第１のバス１４１６を第２のバス１４２０に結合するバスブリッジ１４１８と共に、様々なＩ／Ｏデバイス１４１４が第１のバス１４１６に結合されてよい。一実施形態において、コプロセッサ、高スループットＭＩＣプロセッサ、ＧＰＧＰＵ、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（ＤＳＰ）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１４１５が第１のバス１４１６に連結される。一実施形態において、第２のバス１４２０は、低ピン数（Ｌｏｗ Ｐｉｎ Ｃｏｕｎｔ：ＬＰＣ）バスであってよい。一実施形態において、例えばキーボードおよび／またはマウス１４２２、通信デバイス１４２７、ならびに複数の命令／コードおよびデータ１４３０を含み得るディスクドライブもしく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１４２８を含む様々なデバイスが第２のバス１４２０に結合され得る。更に、オーディオＩ／Ｏ１４２４は、第２のバス１４２０に結合されてよい。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１４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そのようなアーキテクチャを実装することができる。

【０１６７】

ここで図１５を参照すると、本発明のある実施形態に従って、第２のより具体的な例示的システム１５００のブロック図が示されている。図１４および図１５における同一の要素は、複数の同一の参照符号を有し、図１４の特定の態様は、図１５の他の態様を不明瞭にするのを避けるべく、図１５から省略されている。

【０１６８】

図１５は、プロセッサ１４７０、１４８０が、それぞれ統合メモリならびにＩ／Ｏ制御ロジック（「ＣＬ」）１５７２および１５８２を含んでもよいことを示す。したがって、ＣＬ１５７２、１５８２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１５は、メモリ１４３２、１４３４のみがＣＬ１５７２、１５８２に結合されるのではなく、複数のＩ／Ｏデバイス１５１４もＣＬ１５７２、１５８２に結合されることを示す。レガシＩ／Ｏデバイス１５１５は、チップセット１４９０に結合される。

【０１６９】

ここで図１６を参照すると、本発明のある実施形態に従って、ＳｏＣ１６００のブロック図が示されている。図１２の類似の複数の要素は同じ参照符号を有している。また、破線ボックスは、より高度なＳｏＣ上での任意の機能である。図１６中、相互接続ユニット１６０２は、アプリケーションプロセッサ１６１０と、システムエージェントユニット１２１０と、バスコントローラユニット１２１６と、統合メモリコントローラユニット１２１４と、１または複数のコプロセッサ１６２０のセットと、スタティックランダムアクセスメモリ（ＳＲＡＭ）ユニット１６３０と、ダイレクトメモリアクセス（ＤＭＡ）ユニット１６３２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１６４０とに連結される。アプリケーションプロセッサ１６１０は、キャッシュユニット１２０４Ａ～１２０４Ｎを含む、１または複数のコア１２０２Ａ～１２０２Ｎのセットおよび共有キャッシュユニット１２０６を含む。コプロセッサ１６２０のセットは、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１６２０は、例えば、ネットワークまたは通信プロセッサ、圧縮エンジン、ＧＰＧＰＵ、高スループットＭＩＣプロセッサ、組み込みプロセッサ等の専用プロセッサを含む。

【０１７０】

本明細書で開示されたメカニズムの実施形態は、ハードウェア、ソフトウェア、ファームウェア、またはそのような実装手段の組み合わせに実装されてよい。本発明の実施形態はプログラム可能なシステムで実行す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してよく、このプログラム可能なシステムは、少なくとも１つのプロセッサ、記憶装置システム（例えば揮発性および不揮発性メモリおよび／または記憶素子）、少なくとも１つの入力装置、および少なくとも１つの出力装置を備える。

【０１７１】

図１４に図示されるコード１４３０等のプログラムコードが、本明細書に説明される複数の機能を実行して出力情報を生成するための複数の命令を入力するのに適用されてよい。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。この適用を目的として、処理システムは、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備える任意のシステムを含む。

【０１７２】

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続型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。また、必要であれば、プログラムコードは、アセンブリ言語または機械言語で実装されてもよい。実際に、本明細書で記載されたメカニズムはその範囲において、いずれの特定のプログラミング言語にも限定されない。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インタープリタ型言語であってよい。

【０１７３】

少なくとも１つの実施形態の１または複数の態様が、機械可読媒体に格納された典型的な命令により実装されてよく、当該命令は、プロセッサ内の様々なロジックを表し、機械によって読み取られると、機械に、本明細書で説明された技術を実行するためのロジックを作成させる。複数の「ＩＰコア」として知られる、そのような複数の記述表現は、有形の機械可読媒体に格納されてよく、様々な顧客または製造施設に対し供給され、実際にロジックまたはプロセッサを作成する複数の製造機械にロードされてよい。

【０１７４】

このような機械可読ストレージ媒体は、機械または装置により製造または形成された物品の非一時的有形構成を含んでよく、このようなものとしては、ハードディスク等のストレージ媒体、フロッピーディスク、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）および磁気光ディスク等の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、電気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、磁気または光カード、または電子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が、これらに限定されない。

【０１７５】

したがって、本発明の実施形態は、本明細書に記述している構造、回路、装置、プロセッサおよび／またはシステム特徴を定義する命令を含む、または設計データを含む、ハードウェア記述言語（ＨＤＬ）等の非一時的な有形機械可読媒体も含む。このような実施形態はプログラム製品と呼んでもよい。　［エミュレーション（バイナリ変換、コードモーフィング等を含む）］

【０１７６】

いくつかの場合において、命令コンバータを用いて、ソース命令セットからの命令をターゲット命令セットへ変換してよい。例えば、命令コンバータは、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に、（例えば、スタティックバイナリ変換、ダイナミックコンパイルを含むダイナミックバイナリ変換を用いて）解釈し、モーフィングし、エミュレートし、またはそうでなければ変換してよい。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。

【０１７７】

図１７は、本発明のいくつかの実施形態に係る、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よい。図１７は、少なくとも１つのｘ８６命令セットコアを用いるプロセッサ１７１６によりネイティブに実行され得るｘ８６バイナリコード１７０６を生成するべくｘ８６コンパイラ１７０４を用いてコンパイルされ得る高水準言語１７０２のプログラムを示す。少なくとも１つのｘ８６命令セットコアを用いるプロセッサ１７１６は、少なくとも１つのｘ８６命令セットコアを用いるインテル（登録商標）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命令セットコアを用いるインテル（登録商標）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（登録商標）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命令セットコアを用いるインテル（登録商標）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１７０４は、さらなるリンク処理を用いて、または用いることなく、少なくとも１つのｘ８６命令セットコアを用いるプロセッサ１７１６上で実行可能なｘ８６バイナリコード１７０６（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す。同様に、図１７は、少なくとも１つのｘ８６命令セットコアを用いないプロセッサ１７１４（例えば、カルフォルニア州サニーベールのＭＩＰＳ　ＴｅｃｈｎｏｌｏｇｙのＭＩＰＳ命令セットを実行し、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複数のコアを用いるプロセッサ）によりネイティブに実行され得る代替的な命令セットバイナリコード１７１０を生成するべく代替的な命令セットのコンパイラ１７０８を用いてコンパイルされ得る高水準言語１７０２のプログラムを示す。命令コンバータ１７１２は、ｘ８６命令セットコアを用いないプロセッサ１７１４がネイティブに実行し得るコードに、ｘ８６バイナリコード１７０６を変換するのに用いられる。この変換済みコードは、代替の命令セットのバイナリコード１７１０と同じではない可能性がある。なぜなら、このことが可能な命令コ

ンバータは作るのが難しいからである。しかし、変換済みコードは一般的な演算を実現し、代替の命令セットの命令で構成される。故に、命令コンバータ１７１２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１７０６を実行できるようにする。

［さらなる例］

【０１７８】

例１は、１または複数のコアを備えるプロセッサと、ラストレベルキャッシュ（ＬＬＣ）と、キャッシュ制御回路（ＣＣＣ）と、を備える例示的システムを含む。ＬＬＣは、複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有し、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられている。ＣＣＣは、ＬＬＣ内に無効キャッシュライン（ＣＬ）が存在する場合、この無効ＣＬに複数の優先度のうちの要求元優先度を有する後続キャッシュラインを格納し、あるいは、要求元優先度が複数の優先度のうちの最低のもので、１または複数である占有数を有する、または占有数が要求元優先度について最大である場合、後続ＣＬを、要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納し、あるいは、占有数が要求元優先度について最小および最大の間である場合、要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬを格納し、あるいは、占有数が最小よりも低く、より低い優先度を有するＬＲＣ　ＣＬが存在する場合、このＬＲＵ　ＣＬの代わりに後続ＣＬを格納し、あるいは、要求元優先度またはそれより低い優先度を有するエビクション候補が存在しない場合、要求元優先度より高い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬを格納する。

【０１７９】

例２は、例１の例示的システムの内容を含み、ＬＬＣは、複数セットのウェイを含み、複数のウェイは、複数セットの一部で、ＣＣＣは、後続ＣＬをどこに格納するかを決定する前に、後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、後続ＣＬが複数セットのいずれに含まれるかを判定するものとする。

【０１８０】

例３は、例１の例示的システムの内容を含み、ＬＬＣキャッシュエビクションに関するヒューリスティクスを維持するキャッシュ監視回路をさらに備え、より低い優先度を有する後続ＣＬを埋めるために空きを作るため、閾値よりも高い割合の高優先度を有するキャッシュラインがエビクションされる場合、高優先度に対するＣＬＯＳレジスタが、占有する最小および最大ウェイを増やすように更新される。

【０１８１】

例４は、例１の例示的システムの内容を含み、複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である。

【０１８２】

例５は、例１の例示的システムの内容を含み、より低い優先度を有するＬＲＵ　ＣＬが存在し、その代わりに後続ＣＬを格納する際、ＣＣＣは、ＬＲＵ　ＣＬを含むウェイ内にその他ＣＬが存在する場合、その他ＣＬをフラッシュさせるものである。

【０１８３】

例６は、例１の例示的システムの内容を含み、１または複数のコアは、それぞれ仮想マシンを実装し、ＣＣＣはハイパーバイザを含む。

【０１８４】

例７は、例１の例示的システムの内容を含み、プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである。

【０１８５】

例８は、１または複数のコアを備えるプロセッサと、複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先

度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられたＬＬＣと、を備えるシステムにおけるキャッシュ制御回路（ＣＣＣ）によって実行される例示的方法を含む。複数の優先度のうちのある要求元優先度を有する後続キャッシュライン（ＣＬ）を、ＬＬＣに格納する要求を受信する段階と、ＬＬＣ内に無効ＣＬが存在する場合、この無効ＣＬに後続ＣＬを格納する段階と、あるいは、要求元優先度が複数の優先度のうちの最低のもので、１または複数である占有数を有する、または占有数が要求元優先度について最大である場合、後続ＣＬを、要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納する段階と、あるいは、占有数が要求元優先度について最小および最大の間である場合、要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬを格納する段階と、あるいは、占有数が最小よりも低く、より低い優先度を有するＬＲＣ　ＣＬが存在する場合、このＬＲＵ　ＣＬの代わりに後続ＣＬを格納し、あるいは、要求元優先度またはそれより低い優先度を有するエビクション候補が存在しない場合、要求元優先度より高い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬを格納する。

【０１８６】

例９は、例８の例示的方法の内容を含み、ＬＬＣは、複数セットのウェイを含み、複数のウェイは、複数セットの一部で、ＣＣＣは、後続ＣＬをどこに格納するかを決定する前に、後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、後続ＣＬが複数セットのいずれに含まれるかを判定するものとする。

【０１８７】

例１０は、例８の例示的方法の内容を含み、ＬＬＣキャッシュ監視回路を使用して、ＬＬＣキャッシュエビクションに関するヒューリスティクスを維持し、より低い優先度を有する後続ＣＬを埋めるために空きを作るため、閾値よりも高い割合の高優先度を有するキャッシュラインがエビクションされる場合、高優先度に対するＣＬＯＳレジスタを、占有する最小および最大ウェイを増やすように更新する。

【０１８８】

例１１は、例８の例示的方法の内容を含み、複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である。

【０１８９】

例１２は、例８の例示的方法の内容を含み、より低い優先度を有するＬＲＵ　ＣＬが存在し、その代わりに後続ＣＬを格納する際、ＣＣＣは、ＬＲＵ　ＣＬを含むウェイ内にその他ＣＬが存在する場合、その他ＣＬをフラッシュさせるものである。

【０１９０】

例１３は、例８の例示的方法の内容を含み、１または複数のコアは、それぞれ仮想マシンを実装し、ＣＣＣはハイパーバイザを含む。

【０１９１】

例１４は、例８の例示的方法の内容を含み、プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである。

【０１９２】

例１５は、１または複数のコアを備えるプロセッサと、複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられたＬＬＣと、を備えるシステムにおけるキャッシュ制御回路（ＣＣＣ）が応答する命令を含む例示的非一時的コンピュータ読み取り可能な媒体を含む。応答は、ＬＬＣ内に無効キャッシュライン（ＣＬ）が存在する場合、この無効ＣＬに複数の優先度のうちのある要求元優先度を有する後続ＣＬを格納することと、あるいは、要求元優先度が複数の優先度のうちの最低のもので、１または複数である占有数を有する、または占有数が要求元優先度について最大である場合、後続ＣＬを、要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納することと、あるいは、占有数が要求元優先度について最小および最大の間である場合、要

求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬを格納することと、あるいは、占有数が最小よりも低く、より低い優先度を有するＬＲＣ　ＣＬが存在する場合、このＬＲＵ　ＣＬの代わりに後続ＣＬを格納することと、あるいは、要求元優先度またはそれより低い優先度を有するエビクション候補が存在しない場合、要求元優先度より高い優先度のＬＲＵ　ＣＬの代わりに後続ＣＬを格納することと、によって実施される。

【０１９３】

例１６は、例１５の例示的方法の例示的非一時的コンピュータ読み取り可能な媒体の内容を含み、ＬＬＣは、複数セットのウェイを含み、複数のウェイは、複数セットの一部で、ＣＣＣは、命令にさらに応答して、後続ＣＬをどこに格納するかを決定する前に、後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、後続ＣＬが複数セットのいずれに含まれるかを判定するものとする。

【０１９４】

例１７は、例１５の例示的方法の例示的非一時的コンピュータ読み取り可能な媒体の内容を含み、プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである。

【０１９５】

例１８は、例１５の例示的方法の例示的非一時的コンピュータ読み取り可能な媒体の内容を含み、複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である。

【０１９６】

例１９は、例１５の例示的方法の例示的非一時的コンピュータ読み取り可能な媒体の内容を含み、より低い優先度を有するＬＲＵ　ＣＬが存在し、その代わりに後続ＣＬを格納する際、ＣＣＣは、ＬＲＵ　ＣＬを含むウェイ内にその他ＣＬが存在する場合、その他ＣＬをフラッシュさせるものである。

【０１９７】

例２０は、例１５の例示的方法の例示的非一時的コンピュータ読み取り可能な媒体の内容を含み、１または複数のコアは、それぞれ仮想マシンを実装し、ＣＣＣはハイパーバイザを含む。

【０１９８】

［ほかの考えられる項目］

（項目１）　複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられたＬＬＣと、キャッシュ制御回路（ＣＣＣ）であって上記ＬＬＣ内に無効キャッシュライン（ＣＬ）が存在する場合、上記無効ＣＬに、上記複数の優先度の１つである要求元優先度を有する後続キャッシュライン（ＣＬ）を格納し、上記要求元優先度が上記複数の優先度のうちの最低のもので、１または複数である占有数を有する、または上記占有数が上記要求元優先度について最大である場合、上記後続ＣＬを、上記要求元優先度の最も長く使われていない（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納し、上記占有数が上記要求元優先度について最小および上記最大の間である場合、上記要求元優先度またはそれより低い優先度のＬＲＵ　ＣＬの代わりに上記後続ＣＬを格納し、上記占有数が上記最小よりも低く、上記より低い優先度を有するＣＬが存在する場合、上記より低い優先度を有するＬＲＵ　ＣＬの代わりに上記後続ＣＬを格納し、無効ＣＬまたは上記要求元優先度またはそれより低い優先度を有するＣＬが存在しない場合、より高い優先度のＬＲＵ　ＣＬの代わりに上記後続ＣＬを格納する、ＣＣＣと、を備える、システム。

（項目２）　上記ＬＬＣは、複数セットのウェイを含み、上記複数のウェイは、上記複数セットの一部で、上記ＣＣＣは、上記後続ＣＬをどこに格納するかを決定する前に、上記後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、上記後続ＣＬが上記複数セットのいずれに含まれるかを判定するものとする、項目１に記載のシステム。

（項目３） ＬＬＣキャッシュエビクションに関するヒューリスティクスを維持するキャッシュ監視回路をさらに備え、より低い優先度を有する後続ＣＬを埋めるために空きを作るため、閾値よりも高い割合の高優先度を有するキャッシュラインがエビクションされる場合、上記高優先度に対する上記ＣＬＯＳレジスタが、上記占有する最小および最大ウェイを増やすように更新される、項目１に記載のシステム。

（項目４） 上記複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である、項目１に記載のシステム。

（項目５） 上記より低い優先度を有する上記ＬＲＵ ＣＬが存在し、その代わりに上記後続ＣＬを格納する際、上記ＣＣＣは、上記ＬＲＵ ＣＬを含むウェイ内にその他ＣＬが存在する場合、上記その他ＣＬをフラッシュさせるものである、項目１に記載のシステム。

（項目６） 上記ＬＬＣおよび上記ＣＣＣを内蔵し、それぞれ仮想マシンを実装する１または複数のコアを有するプロセッサをさらに備え、上記ＣＣＣはハイパーバイザを含む、項目１に記載のシステム。

（項目７） 上記プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである、項目６に記載のシステム。

（項目８） 複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられた、ＬＬＣを備えるシステムにおけるキャッシュ制御回路（ＣＣＣ）によって実行される方法であって、上記複数の優先度のうちのある要求元優先度を有する後続キャッシュライン（ＣＬ）を、上記ＬＬＣに格納する要求を受信する段階と、上記ＬＬＣ内に無効ＣＬが存在する場合、上記無効ＣＬに上記後続ＣＬを格納する段階と、上記要求元優先度が上記複数の優先度のうちの最低のもので、１または複数である占有数を有する、または上記占有数が上記要求元優先度について最大である場合、上記後続ＣＬを、上記要求元優先度の最も長く使われていない（Ｌｅａｓｔ Ｒｅｃｅｎｔｌｙ Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納する段階と、上記占有数が上記要求元優先度について最小および上記最大の間である場合、上記要求元優先度またはそれより低い優先度のＬＲＵ ＣＬの代わりに上記後続ＣＬを格納する段階と、上記占有数が上記最小よりも低く、上記より低い優先度を有するＣＬが存在する場合、上記より低い優先度を有するＬＲＵ ＣＬの代わりに上記後続ＣＬを格納する段階と、無効ＣＬまたは上記要求元優先度またはそれより低い優先度を有するＣＬが存在しない場合、より高い優先度のＬＲＵ ＣＬの代わりに上記後続ＣＬを格納する段階と、を含む、方法。

（項目９） 上記ＬＬＣは、複数セットのウェイを含み、上記複数のウェイは、上記複数セットの一部で、上記ＣＣＣは、上記後続ＣＬをどこに格納するかを決定する前に、上記後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、上記後続ＣＬが上記複数セットのいずれに含まれるかを判定するものとする、項目８に記載の方法。

（項目１０） ＬＬＣキャッシュ監視回路を使用して、ＬＬＣキャッシュエビクションに関するヒューリスティクスを維持し、より低い優先度を有する後続ＣＬを埋めるために空きを作るため、閾値よりも高い割合の高優先度を有するキャッシュラインがエビクションされる場合、上記高優先度に対するＣＬＯＳレジスタを、上記占有する最小および最大ウェイを増やすように更新する、項目８に記載の方法。

（項目１１） 上記複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である、項目８に記載の方法。

（項目１２） 上記より低い優先度を有する上記ＬＲＵ ＣＬが存在し、その代わりに上記後続ＣＬを格納する際、上記ＣＣＣは、上記ＬＲＵ ＣＬを含むウェイ内にその他ＣＬが存在する場合、上記その他ＣＬをフラッシュさせるものである、項目８に記載の方法。

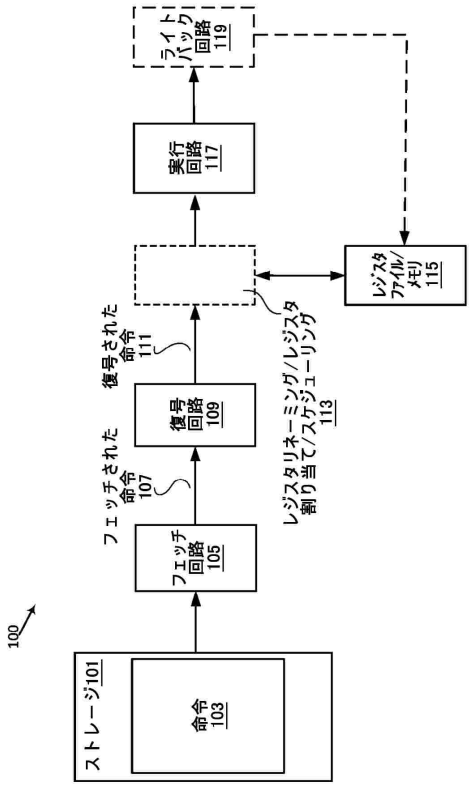
（項目１３） 上記システムは、上記ＬＬＣおよび上記ＣＣＣを内蔵し、それぞれ仮想マシンを実装する１または複数のコアを有するプロセッサをさらに備え、上記ＣＣＣはハイパーバイザを含む、項目８に記載の方法。

（項目１４） 上記プロセッサは、クラウドサービスプロバイダのデータセンタ内の複

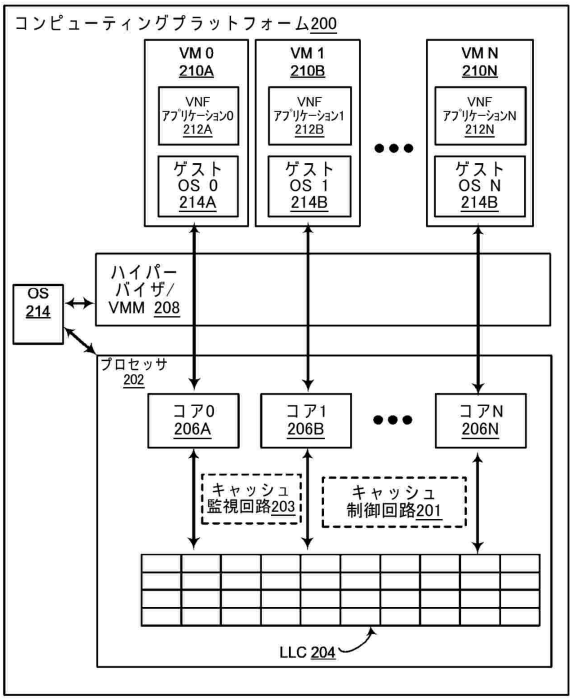
数のプロセッサの１つである、項目１３に記載の方法。
（項目１５） 複数の優先度の１つにそれぞれ割り当てられた複数のウェイを有するラストレベルキャッシュ（ＬＬＣ）であって、各優先度が、占有する最小および最大ウェイ数を指定するクラスオブサービス（ＣＬＯＳ）レジスタに関連付けられた、ＬＬＣを備えるシステムにおけるキャッシュ制御回路（ＣＣＣ）が応答する命令を含む非一時的なコンピュータ読み取り可能な媒体であって、上記応答は、上記複数の優先度のうちのある要求元優先度を有する後続キャッシュライン（ＣＬ）を、上記ＬＬＣに格納する要求を受信することと、上記ＬＬＣ内に無効ＣＬが存在する場合、上記無効ＣＬに上記後続ＣＬを格納することと、上記要求元優先度が上記複数の優先度のうちの最低のもので、１または複数である占有数を有する、または上記占有数が上記要求元優先度について最大である場合、上記後続ＣＬを、上記要求元優先度の最も長く使われていない（Ｌｅａｓｔ Ｒｅｃｅｎｔｌｙ Ｕｓｅｄ：ＬＲＵ）ＣＬの代わりに格納することと、上記占有数が上記要求元優先度について最小および上記最大の間である場合、上記要求元優先度またはそれより低い優先度のＬＲＵ ＣＬの代わりに上記後続ＣＬを格納することと、上記占有数が上記最小よりも低く、上記より低い優先度を有するＣＬが存在する場合、上記より低い優先度を有するＬＲＵ ＣＬの代わりに上記後続ＣＬを格納することと、無効ＣＬまたは上記要求元優先度またはそれより低い優先度を有するＣＬが存在しない場合、より高い優先度のＬＲＵ ＣＬの代わりに上記後続ＣＬを格納することと、によって実施される、非一時的なコンピュータ読み取り可能な媒体。
（項目１６） 上記ＬＬＣは、複数セットのウェイを含み、上記複数のウェイは、上記複数セットの一部で、上記ＣＣＣは、上記命令にさらに応答して、上記後続ＣＬをどこに格納するかを決定する前に、上記後続ＣＬの論理アドレスに実行されるハッシングアルゴリズムに基づいて、上記後続ＣＬが上記複数セットのいずれに含まれるかを判定するものとする、項目１５に記載の非一時的なコンピュータ読み取り可能な媒体。
（項目１７） 上記システムは上記ＬＬＣおよび上記ＣＣＣを内蔵するプロセッサをさらに備え、上記プロセッサは、クラウドサービスプロバイダのデータセンタ内の複数のプロセッサの１つである、項目１５に記載の非一時的なコンピュータ読み取り可能な媒体。
（項目１８） 上記複数のウェイは、それぞれＮ個のＣＬを含み、Ｎは１以上の正の整数である、項目１５に記載の非一時的なコンピュータ読み取り可能な媒体。
（項目１９） 上記より低い優先度を有する上記ＬＲＵ ＣＬが存在し、その代わりに上記後続ＣＬを格納する際、上記ＣＣＣは、上記ＬＲＵ ＣＬを含むウェイ内にその他ＣＬが存在する場合、上記その他ＣＬをフラッシュさせるものである、項目１５に記載の非一時的コンピュータ読み取り可能な媒体。
（項目２０） 上記システムは、上記ＬＬＣおよび上記ＣＣＣを内蔵し、それぞれ仮想マシンを実装する１または複数のコアを有するプロセッサをさらに備え、上記ＣＣＣはハイパーバイザを含む、項目１５に記載の非一時的コンピュータ読み取り可能な媒体。

【図面】

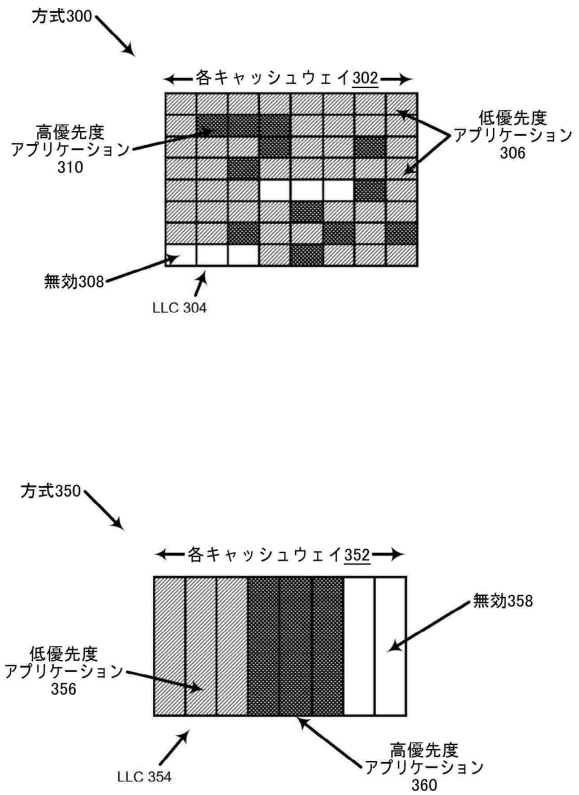
【図１】


ライトバック回路 119
実行回路 117
レジスタファイル/メモリ 115
復号された命令 111
レジスタリネーミング/レジスタ割り当て/スケジューリング 113
復号回路 109
フェッチされた命令 107
フェッチ回路 105
100
ストレージ101
命令 103

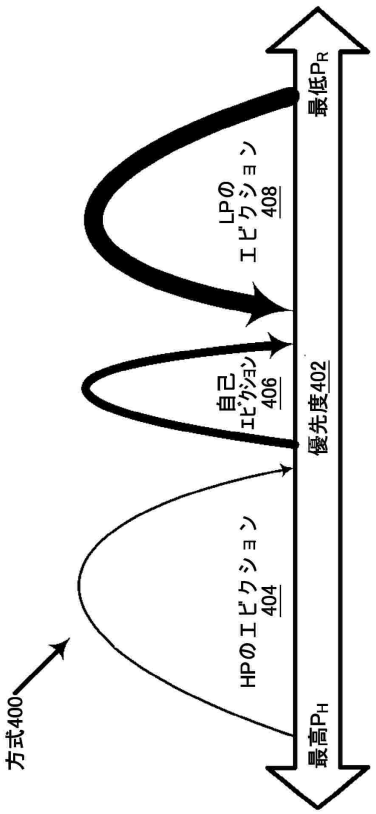
【図２】


コンピューティングプラットフォーム200
VM 0 210A
VM 1 210B
VM N 210N
VNF アプリケーション0 212A
VNF アプリケーション1 212B
VNF アプリケーションN 212N
ゲスト OS 0 214A
ゲスト OS 1 214B
ゲスト OS N 214B
OS 214
ハイパーバイザ/VMM 208
プロセッサ 202
コア0 206A
コア1 206B
コアN 206N
キャッシュ監視回路203
キャッシュ制御回路201
LLC 204

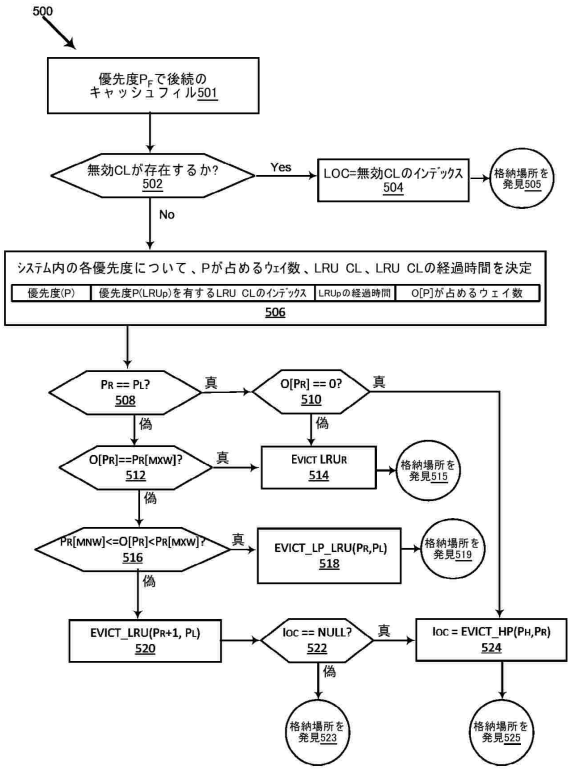
【図３】


方式300
各キャッシュウェイ302
高優先度アプリケーション 310
低優先度アプリケーション 306
無効308
LLC 304
方式350
各キャッシュウェイ352
無効358
低優先度アプリケーション 356
LLC 354
高優先度アプリケーション 360

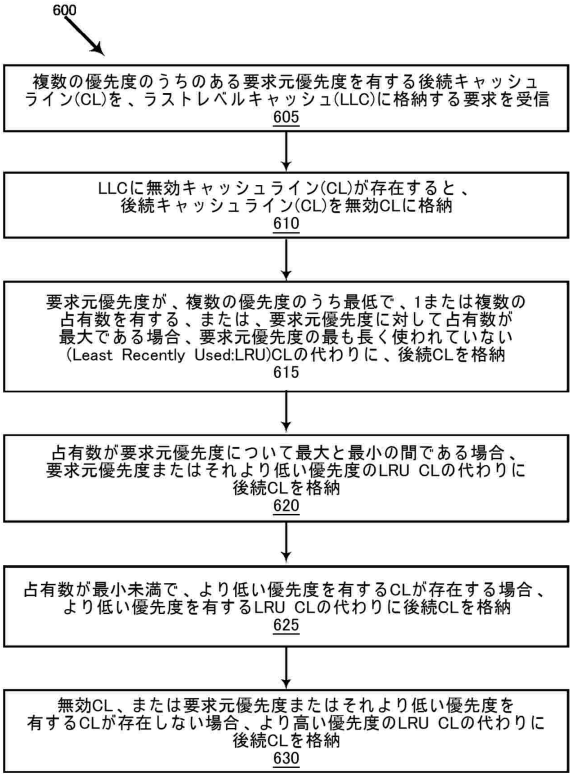
【図４】


最低P_R
LPのエビクション 408
自己エビクション 406
優先度402
HPのエビクション 404
方式400
最高P_H

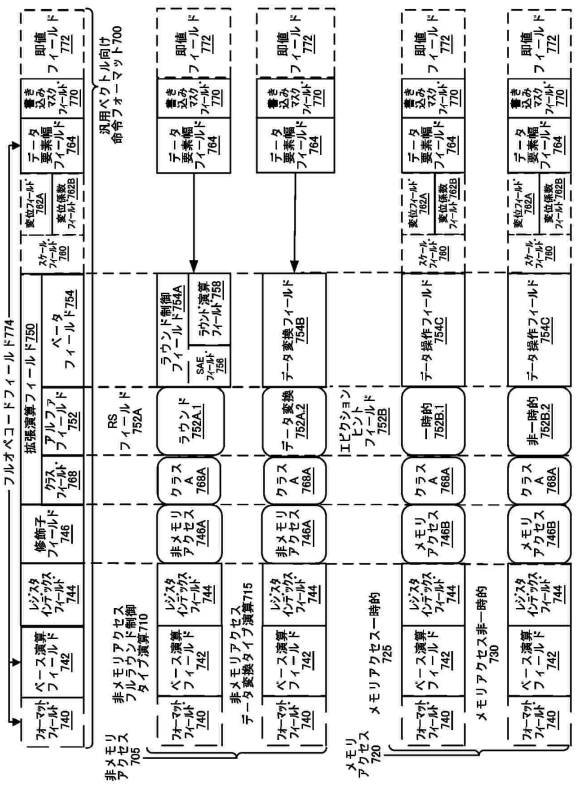
【図５】


500
優先度P_Fで後続のキャッシュフィル501
無効CLが存在するか?
502
Yes
LOC=無効CLのインデックス
504
格納場所を発見505
No
システム内の各優先度について、Pが占めるウェイ数、LRU CL、LRU CLの経過時間を決定
優先度(P)
優先度P(LRUp)を有するLRU CLのインデックス
LRUpの経過時間
O[P]が占めるウェイ数
506
PR == PL?
508
真
O[PR] == 0?
510
偽
O[PR]==PR[MXW]?
512
EVICT LRUR
514
格納場所を発見515
PR[MNW]<=O[PR]<PR[MXW]?
516
EVICT_LP_LRU(PR,PL)
518
格納場所を発見519
EVICT_LRU(PR+1, PL)
520
IOC == NULL?
522
IOC = EVICT_HP(PH,PR)
524
格納場所を発見523
格納場所を発見525

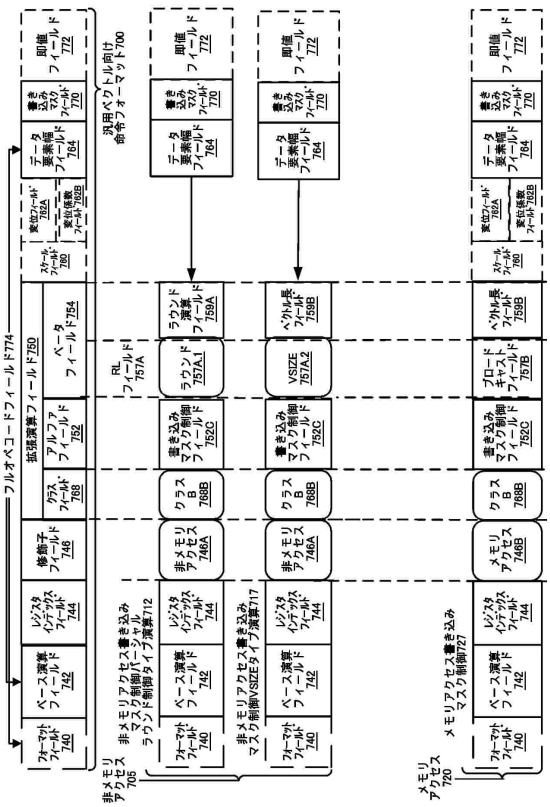
【図６】


600
複数の優先度のうちのある要求元優先度を有する後続キャッシュライン(CL)を、ラストレベルキャッシュ(LLC)に格納する要求を受信
605
LLCに無効キャッシュライン(CL)が存在すると、後続キャッシュライン(CL)を無効CLに格納
610
要求元優先度が、複数の優先度のうち最低で、1または複数の占有数を有する、または、要求元優先度に対して占有数が最大である場合、要求元優先度の最も長く使われていない(Least Recently Used:LRU)CLの代わりに、後続CLを格納
615
占有数が要求元優先度について最大と最小の間である場合、要求元優先度またはそれより低い優先度のLRU CLの代わりに後続CLを格納
620
占有数が最小未満で、より低い優先度を有するCLが存在する場合、より低い優先度を有するLRU CLの代わりに後続CLを格納
625
無効CL、または要求元優先度またはそれより低い優先度を有するCLが存在しない場合、より高い優先度のLRU CLの代わりに後続CLを格納
630

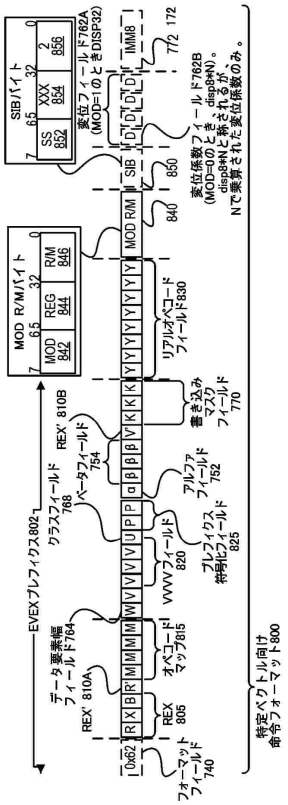
【図７Ａ】


フルオペコードフィールド774
拡張演算フィールド750
ベータフィールド754
アルファフィールド752
クラスフィールド768
修飾子フィールド746
レジスタインデックスフィールド744
ベース演算フィールド742
フォーマットフィールド740
即値フィールド772
書き込みマスクフィールド770
データ要素幅フィールド764
変位フィールド762A
変位係数フィールド762B
ストールフィールド780
汎用ベクトル向け命令フォーマット700
RSフィールド752A
ラウンド制御フィールド754A
SAEフィールド756
ラウンド演算フィールド758
ラウンド762A.1
データ変換フィールド754B
データ変換762A.2
エビクションヒントフィールド752B
データ操作フィールド754C
一時的762B.1
非一時的762B.2
クラスAフィールド768A
非メモリアクセス746A
メモリアクセス746B
非メモリアクセス705
非メモリアクセスフルラウンド制御タイプ演算710
非メモリアクセスデータ変換タイプ演算715
メモリアクセス720
メモリアクセス一時的725
メモリアクセス非一時的730

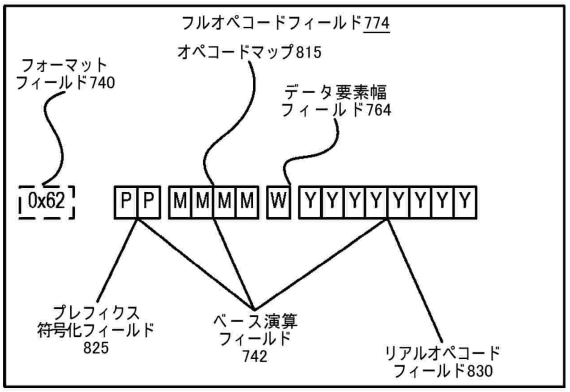
【図７Ｂ】


フルオペコードフィールド774
拡張演算フィールド750
ベータフィールド754
アルファフィールド752
RLフィールド757A
ラウンド757A.1
VSIZE757A.2
ブロードキャストフィールド757B
クラスフィールド768
クラスBフィールド768B
ラウンド演算フィールド768A
ベクトル長フィールド768B
修飾子フィールド746
レジスタインデックスフィールド744
ベース演算フィールド742
フォーマットフィールド740
即値フィールド772
書き込みマスクフィールド770
データ要素幅フィールド764
変位フィールド762A
変位係数フィールド762B
ストールフィールド780
汎用ベクトル向け命令フォーマット700
非メモリアクセス705
非メモリアクセス書き込みマスク制御ラウンドタイプ演算712
非メモリアクセス書き込みマスク制御VSIZEタイプ演算717
書き込みマスク制御752C
非メモリアクセス746A
メモリアクセス746B
メモリアクセス書き込みマスク制御727
メモリアクセス720

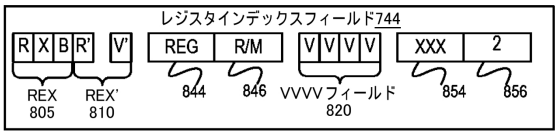
【図８Ａ】



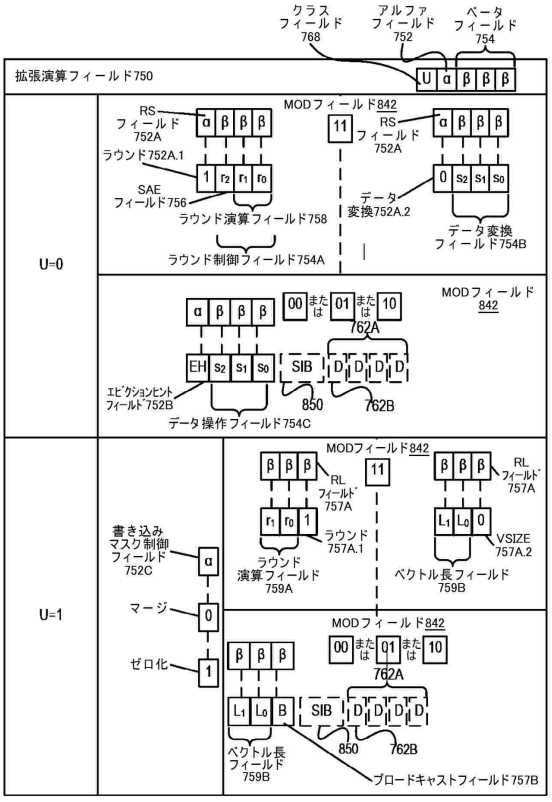
【図８Ｂ】



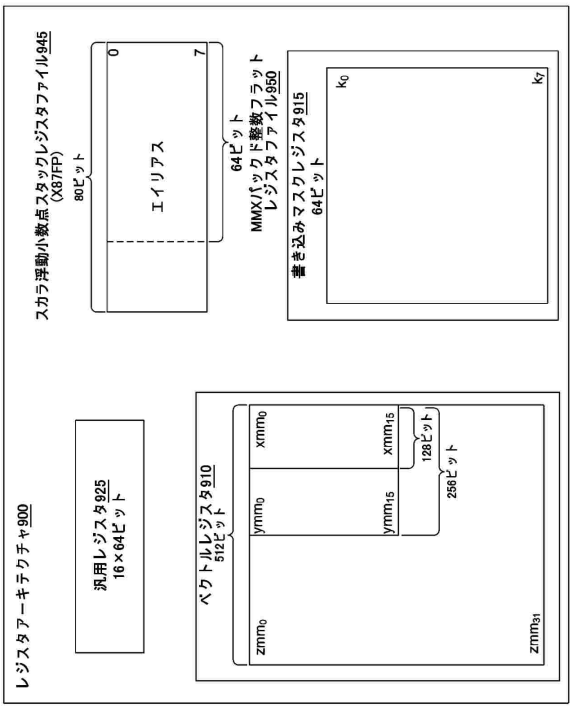
【図８Ｃ】



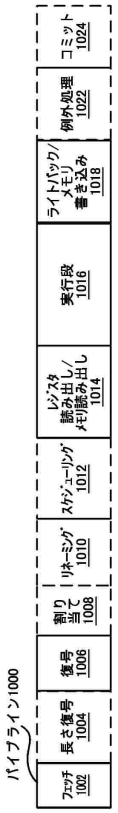
【図８Ｄ】



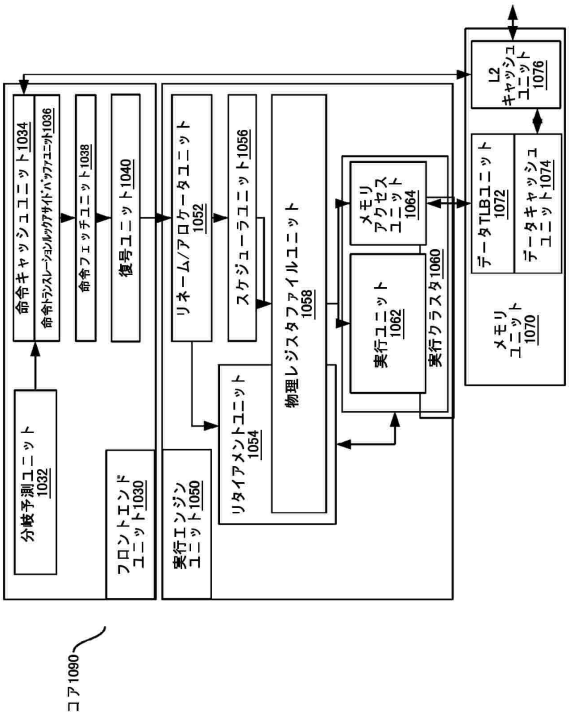
【図９】



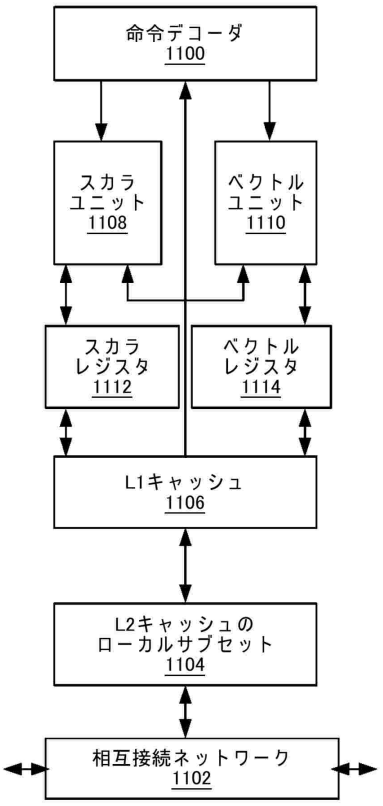
【図１０Ａ】



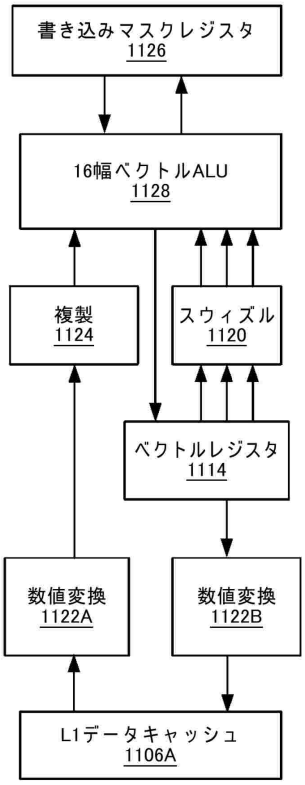
【図１０Ｂ】



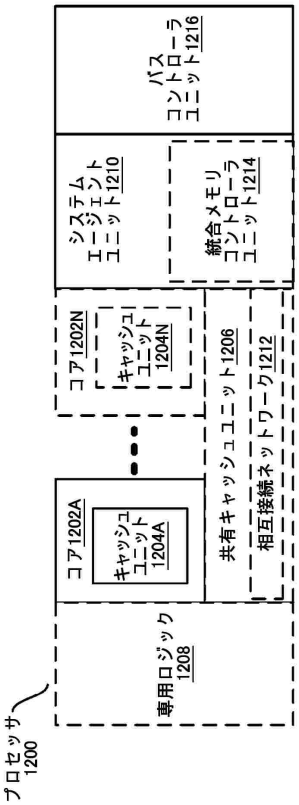
【図１１Ａ】



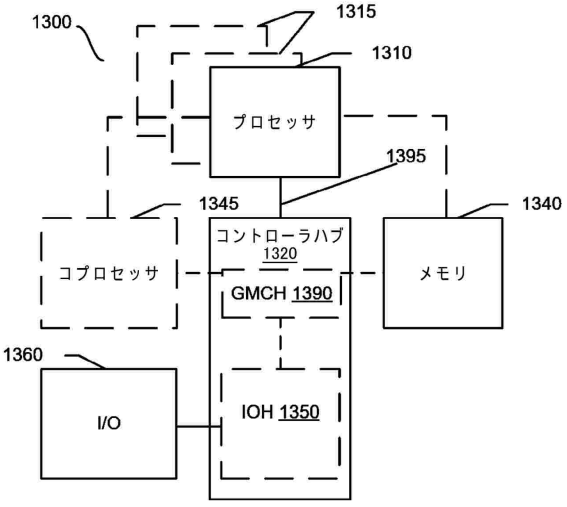
【図１１Ｂ】



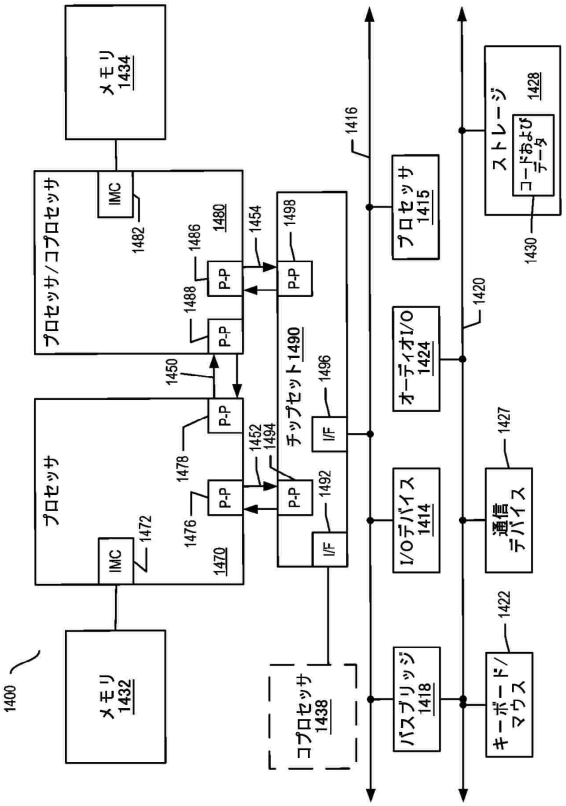
【図１２】



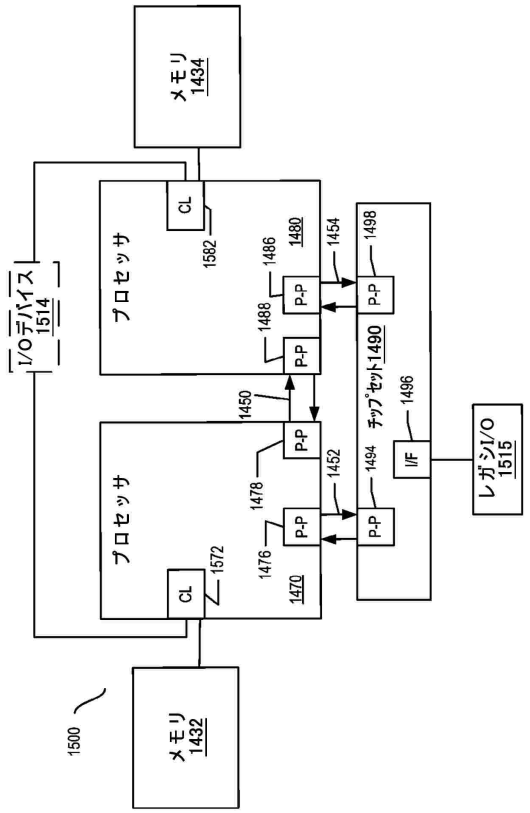
【図１３】



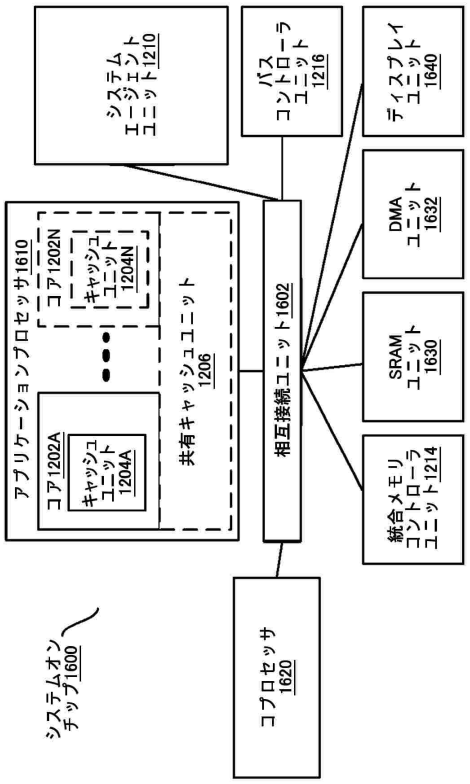
【図１４】



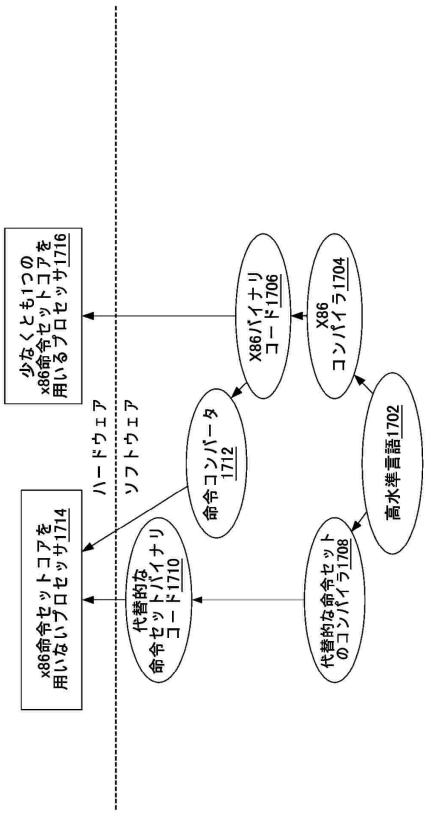
【図１５】


メモリ
1434
I/Oデバイス
1514
プロセッサ
CL
1582
1486
1480
1454
1498
P-P
1488
1450
1452
1494
チップセット1490
1496
I/F
レガシI/O
1515
1478
1476
1470
1572
1500
メモリ
1432

【図１６】


システム
エージェント
ユニット1210
バス
コントローラ
ユニット
1216
ディスプレイ
ユニット
1640
DMA
ユニット
1632
SRAM
ユニット
1630
統合メモリ
コントローラ
ユニット1214
アプリケーションプロセッサ1610
コア1202A
コア1202N
キャッシュ
ユニット
1204A
キャッシュ
ユニット
1204N
共有キャッシュユニット
1206
相互接続ユニット1602
コプロセッサ
1620
システムオン
チップ1600

【図１７】


ハードウェア
ソフトウェア
少なくとも1つの
x86命令セットコアを
用いるプロセッサ1716
x86命令セットコアを
用いないプロセッサ1714
x86バイナリ
コード1706
X86
コンパイラ1704
命令コンバータ
1712
高水準言語1702
代替的な
命令セットバイナリ
コード1710
代替的な命令セット
のコンパイラ1708

フロントページの続き

レッジ ブーレバード・２２００ インテル・コーポレーション内

審査官　北村 学

(56)参考文献　特開２００９－１６３４５０（ＪＰ，Ａ）
特開平０９－１０１９１６（ＪＰ，Ａ）
特開２０１１－０１８１９６（ＪＰ，Ａ）
米国特許出願公開第２００８／００４０５５４（ＵＳ，Ａ１）
米国特許出願公開第２００８／０２３５４５７（ＵＳ，Ａ１）
米国特許出願公開第２００９／０１７２３１５（ＵＳ，Ａ１）
米国特許出願公開第２０１５／００６７２６６（ＵＳ，Ａ１）
国際公開第２０１２／０９５９５７（ＷＯ，Ａ１）

(58)調査した分野　(Int.Cl.，ＤＢ名)
ＩＰＣ Ｇ０６Ｆ １２／０８ － １２／１２８