(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6673574号
(P6673574)
(45)発行日　令和2年3月25日(2020.3.25)　(24)登録日　令和2年3月9日(2020.3.9)

(51)Int.Cl.　　F I
G06F 9/315 (2006.01)　　G06F 9/315 S
G06F 9/38 (2006.01)　　G06F 9/38 370A

請求項の数 25　(全 38 頁)

(21)出願番号　特願2017-528541(P2017-528541)
(86)(22)出願日　平成27年11月25日(2015.11.25)
(65)公表番号　特表2018-506096(P2018-506096A)
(43)公表日　平成30年3月1日(2018.3.1)
(86)国際出願番号　PCT/US2015/062564
(87)国際公開番号　WO2016/105819
(87)国際公開日　平成28年6月30日(2016.6.30)
審査請求日　平成30年11月20日(2018.11.20)
(31)優先権主張番号　14/583,636
(32)優先日　平成26年12月27日(2014.12.27)
(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
龍華国際特許業務法人
(72)発明者　ウルド－アハメド－ヴァル、エルムスタファ
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

最終頁に続く

(54)【発明の名称】ベクトルビットシャッフルを実行するための方法および装置

(57)【特許請求の範囲】
【請求項１】
複数のソースデータ要素を格納するための第１のベクトルレジスタと、
複数の制御要素を格納するための第２のベクトルレジスタであって、前記複数の制御要素の各々は複数のビットフィールドを含んで前記第１のベクトルレジスタにおける前記複数のソースデータ要素のうちの別々の１つに対応し、各ビットフィールドはデスティネーションマスクレジスタ内の単一のビット位置に対応し、各ビットフィールドは、前記デスティネーションマスクレジスタ内の対応する前記単一のビット位置へコピーされるべき、対応する前記ソースデータ要素からの一のビットを識別する、第２のベクトルレジスタと、
前記対応するソースデータ要素からの一のビットを識別すべく前記第２のベクトルレジスタから各ビットフィールドを読み出し且つこれに応じて前記対応するソースデータ要素からの識別された前記ビットを、前記デスティネーションマスクレジスタ内の前記ビットフィールドに対応する単一のビット位置にコピーするためのベクトルビットシャッフルロジックと、を備え、
前記デスティネーションマスクレジスタのビット数は、前記複数のソースデータ要素の個数と、当該複数の制御要素の各々における前記複数のビットフィールドの個数との乗算値に等しい、プロセッサ。
【請求項２】
前記ベクトルビットシャッフルロジックは、前記複数の制御要素の各々における前記複

数のビットフィールドに従い、前記複数のソースデータ要素の各々からビットのセットを選択するための１または複数のマルチプレクサを含む、請求項１に記載のプロセッサ。
【請求項３】
前記複数のソースデータ要素の各々は６４ビットデータ要素を含み、各ビットフィールドは、複数の前記６４ビットデータ要素の各々からのビットを識別するための少なくとも６ビットを含む、請求項１または２に記載のプロセッサ。
【請求項４】
前記複数のビットフィールドの各々は制御バイトを含み、前記６ビットは複数の前記制御バイトの各々から選択され、複数の前記６４ビットデータ要素の各々からの各ビットを識別する、請求項３に記載のプロセッサ。
【請求項５】
８個の前記制御バイトを使用して、各データ要素から８ビットが選択される、請求項４に記載のプロセッサ。
【請求項６】
各データ要素からの前記８ビットは、前記デスティネーションマスクレジスタ内で連結される、請求項５に記載のプロセッサ。
【請求項７】
前記第１のベクトルレジスタは、複数の前記６４ビットデータ要素のうち８個の６４ビットデータ要素を格納し、前記デスティネーションマスクレジスタは前記８個の６４ビットデータ要素から選択された８個の対応する８ビット値を格納する、請求項６に記載のプロセッサ。
【請求項８】
前記デスティネーションマスクレジスタ内の前記ビットは、前記プロセッサによって実行される１または複数の後続の命令のためのマスク演算の実行に使用される、請求項７に記載のプロセッサ。
【請求項９】
前記ベクトルビットシャッフルロジックは、前記プロセッサ内のデコードロジックによってデコードされ、前記プロセッサ内の実行ロジックによって実行されるベクトルビットシャッフル命令に応答して動作する、請求項１から８のいずれか一項に記載のプロセッサ。
【請求項１０】
複数のソースデータ要素を第１のベクトルレジスタ内に格納する段階と、
複数の制御要素を第２のベクトルレジスタ内に格納する段階であって、前記複数の制御要素の各々は複数のビットフィールドを含んで前記第１のベクトルレジスタにおける前記複数のソースデータ要素のうちの別々の１つに対応し、各ビットフィールドはデスティネーションマスクレジスタ内の単一のビット位置に対応し、各ビットフィールドは、前記デスティネーションマスクレジスタ内の対応する前記単一のビット位置へコピーされるべき、対応する前記ソースデータ要素からの一のビットを識別する、複数の制御要素を第２のベクトルレジスタ内に格納する段階と、
前記対応するソースデータ要素からの一のビットを識別すべく前記第２のベクトルレジスタから各ビットフィールドを読み出し且つこれに応じて前記対応するソースデータ要素からの識別された前記ビットを、前記デスティネーションマスクレジスタ内の前記ビットフィールドに対応する単一のビット位置にコピーする段階と、を備え、
前記デスティネーションマスクレジスタのビット数は、前記複数のソースデータ要素の個数と、当該複数の制御要素の各々における前記複数のビットフィールドの個数との乗算値に等しい、方法。
【請求項１１】
前記複数の制御要素の各々における前記複数のビットフィールドに従い、１または複数のマルチプレクサを用いて、前記複数のソースデータ要素の各々からビットのセットを選択する段階をさらに備える、請求項１０に記載の方法。

【請求項１２】
前記複数のソースデータ要素の各々は６４ビットデータ要素を含み、各ビットフィールドは複数の前記６４ビットデータ要素の各々からのビットを識別するための少なくとも６ビットを含む、請求項１０または１１に記載の方法。
【請求項１３】
前記複数のビットフィールドの各々は制御バイトを含み、前記６ビットは、複数の前記制御バイトの各々から選択され、複数の前記６４ビットデータ要素の各々からの各ビットを識別する、請求項１２に記載の方法。
【請求項１４】
８個の前記制御バイトを使用して、各データ要素から８ビットが選択される、請求項１３に記載の方法。
【請求項１５】
各データ要素からの前記８ビットは、前記デスティネーションマスクレジスタ内で連結される、請求項１４に記載の方法。
【請求項１６】
前記第１のベクトルレジスタは複数の前記６４ビットデータ要素のうち８個の６４ビットデータ要素を格納し、前記デスティネーションマスクレジスタは前記８個の６４ビットデータ要素から選択された８個の対応する８ビット値を格納する、請求項１５に記載の方法。
【請求項１７】
前記デスティネーションマスクレジスタ内の前記ビットは、プロセッサによって実行される１または複数の後続の命令のためのマスク演算の実行に使用される、請求項１６に記載の方法。
【請求項１８】
プログラムコードおよびデータを格納するためのメモリと、
指定されたキャッシュ管理ポリシーに従い前記プログラムコードおよびデータをキャッシュするための複数のキャッシュレベルを有するキャッシュ階層と、
ユーザからの入力を受信するための入力デバイスと、
前記プログラムコードを実行し且つ前記ユーザからの前記入力に応じて前記データを処理するためのプロセッサと、を備え、
前記プロセッサは、
複数のソースデータ要素を格納するための第１のベクトルレジスタと、
複数の制御要素を格納するための第２のベクトルレジスタであって、前記複数の制御要素の各々は複数のビットフィールドを含んで前記第１のベクトルレジスタにおける前記複数のソースデータ要素のうちの別々の１つに対応し、各ビットフィールドはデスティネーションマスクレジスタ内の単一のビット位置に対応し、各ビットフィールドは、前記デスティネーションマスクレジスタ内の対応する前記単一のビット位置へコピーされるべき、対応する前記ソースデータ要素からの一のビットを識別する、第２のベクトルレジスタと、
前記対応するソースデータ要素からの一のビットを識別すべく前記第２のベクトルレジスタから各ビットフィールドを読み出し且つこれに応じて前記対応するソースデータ要素からの識別された前記ビットを、前記デスティネーションマスクレジスタ内の前記ビットフィールドに対応する単一のビット位置にコピーするためのベクトルビットシャッフルロジックと、を含み、
前記デスティネーションマスクレジスタのビット数は、前記複数のソースデータ要素の個数と、当該複数の制御要素の各々における前記複数のビットフィールドの個数との乗算値に等しい、システム。
【請求項１９】
前記ベクトルビットシャッフルロジックは、前記複数の制御要素の各々における前記複数のビットフィールドに従い、前記複数のソースデータ要素の各々からビットのセットを

選択するための１または複数のマルチプレクサを含む、請求項１８に記載のシステム。
【請求項２０】
前記複数のソースデータ要素の各々は６４ビットデータ要素を含み、各ビットフィールドは複数の前記６４ビットデータ要素の各々からのビットを識別するための少なくとも６ビットを含む、請求項１８に記載のシステム。
【請求項２１】
前記複数のビットフィールドの各々は制御バイトを含み、前記６ビットは複数の前記制御バイトの各々から選択され、複数の前記６４ビットデータ要素の各々からの各ビットを識別する、請求項２０に記載のシステム。
【請求項２２】
８個の前記制御バイトを使用して、各データ要素から８ビットが選択される、請求項２１に記載のシステム。
【請求項２３】
各データ要素からの前記８ビットは、前記デスティネーションマスクレジスタ内で連結される、請求項２２に記載のシステム。
【請求項２４】
前記第１のベクトルレジスタは複数の前記６４ビットデータ要素のうち８個の６４ビットデータ要素を格納し、前記デスティネーションマスクレジスタは前記８個の６４ビットデータ要素から選択された８個の対応する８ビット値を格納する、請求項２３に記載のシステム。
【請求項２５】
前記デスティネーションマスクレジスタ内の前記ビットは、前記プロセッサによって実行される１または複数の後続の命令のためのマスク演算の実行に使用される、請求項２４に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、概してコンピュータのプロセッサ分野に関する。具体的には、本発明は、ベクトルビットシャッフルを実行するための方法および装置に関する。
【背景技術】
【０００２】
命令セットまたは命令セットアーキテクチャ（ＩＳＡ）は、ネイティブデータタイプ、命令、レジスタアーキテクチャ、アドレス指定モード、メモリアーキテクチャ、割り込みおよび例外処理並びに外部入力および出力（Ｉ／Ｏ）を含む、プログラミングに関するコンピュータアーキテクチャの一部である。本明細書において、「命令」という用語は概してマクロ命令を指すことに留意されたい。マクロ命令とは、実行のためにプロセッサに供給される命令であり、これに対し、マイクロ命令またはマイクロｏｐとは、マクロ命令をデコーディングするプロセッサのデコーダの結果である。マイクロ命令またはマイクロｏｐは、プロセッサの実行ユニットに対し、マクロ命令に関連するロジックを実装するための演算を実行するよう命令するように構成可能である。
【０００３】
ＩＳＡは、命令セットの実装に使用される一連のプロセッサ設計技術であるマイクロアーキテクチャとは区別される。異なるマイクロアーキテクチャを持つプロセッサは、共通の命令セットを共有可能である。例えば、インテル（登録商標）ＰＥＮＴＩＵＭ（登録商標）４プロセッサ、インテル（登録商標）コア（商標）プロセッサおよびカリフォルニア州サニーベールのＡｄｖａｎｃｅｄ　Ｍｉｃｒｏ　Ｄｅｖｉｃｅｓ社のプロセッサは、ｘ８６命令セット（より新しいバージョンに追加されたいくつかの拡張機能を持つ）とほぼ同一バージョンを実装するが、内部設計が異なる。例えば、ＩＳＡの同一のレジスタアーキテクチャは、専用の物理レジスタ、レジスタリネーミングメカニズムを使用（例えば、レジスタエイリアステーブル（ＲＡＴ）、リオーダバッファ（ＲＯＢ）およびリタイアメ

ントレジスタファイルの使用）して動的に割り当てられた１または複数の物理レジスタを含む周知の技術を使用して異なるマイクロアーキテクチャに異なる方法で実装されてよい。別途の記載がない限り、本明細書において、レジスタアーキテクチャ、レジスタファイルおよびレジスタという文言は、ソフトウェア／プログラマに可視であるもの、および命令がレジスタを指定する方法を指すために使用される。区別が必要な場合、「論理」、「アーキテクチャ」または「ソフトウェアビジブル」なる形容詞が、レジスタアーキテクチャにおけるレジスタ／ファイルを示すために使用される一方で、異なる形容詞が、特定のマイクロアーキテクチャにおけるレジスタ（例えば、物理レジスタ、リオーダバッファ、リタイアメントレジスタ、レジスタプール）を指すために使用される。

【０００４】

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。特定の命令は、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。命令ストリームとは、特定の命令シーケンスであり、シーケンス内の各命令は、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つにおける）内の命令の出現である。

【図面の簡単な説明】

【０００５】

以下の詳細な説明に以下の添付図面を組み合わせると、本発明のより良い理解が得られる。

【０００６】

【図１Ａ】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１Ｂ】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。

【０００７】

【図２Ａ】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すブロック図である。

【図２Ｂ】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すブロック図である。

【図２Ｃ】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すブロック図である。

【図２Ｄ】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すブロック図である。

【０００８】

【図３】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。

【０００９】

【図４Ａ】本発明の実施形態による、例示的なインオーダフェッチ、デコード、リタイア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。

【００１０】

【図４Ｂ】本発明の実施形態によるプロセッサに含まれる、インオーダフェッチ、デコー

ド、リタイア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。
【００１１】
【図５Ａ】オンダイ相互接続ネットワークへの接続を伴う単一のプロセッサコアのブロック図である。
【００１２】
【図５Ｂ】本発明の実施形態による図５Ａ中のプロセッサコアの一部の拡大図を示す。
【００１３】
【図６】本発明の実施形態による統合メモリコントローラおよびグラフィックを持つ単一のコアプロセッサおよびマルチコアプロセッサのブロック図である。
【００１４】
【図７】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。
【００１５】
【図８】本発明の実施形態による第２のシステムのブロック図を示す。
【００１６】
【図９】本発明の実施形態による第３のシステムのブロック図を示す。
【００１７】
【図１０】本発明の実施形態によるシステムオンチップ（ＳｏＣ）のブロック図を示す。
【００１８】
【図１１】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を示す。
【００１９】
【図１２】本発明の実施形態が実装されてよい例示的なプロセッサを示す。
【００２０】
【図１３】本発明の一実施形態によるベクトルビットシャッフルロジックを示す。
【００２１】
【図１４】本発明の一実施形態による方法を示す。
【発明を実施するための形態】
【００２２】
以下の詳細な説明には、後述の本発明の実施形態に係る完全な理解を共すべく、説明目的で多数の具体的な詳細が記載されている。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の一部を省いても実施可能であることは当業者に自明なところである。他の例においては、本説明の実施形態に係る根本的な原理を曖昧にしないように、周知の構造およびデバイスはブロック図内に詳細に示されていない。
［例示的なプロセッサアーキテクチャおよびデータタイプ］
【００２３】
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、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］

【００２４】

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。

Ａ［汎用ベクトル向け命令フォーマット］

【００２５】

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特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を使用する。

【００２６】

図１Ａ～１Ｂ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１Ａ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１Ｂ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１００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１０５命令テンプレートおよびメモリアクセス１２０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）いう用語は、いずれの特定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。

【００２７】

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。

【００２８】

図１Ａ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１０５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１１０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１１５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１２０命令テンプレート内に、メモリアクセス、一時的１２５命令テンプレートおよびメモリアクセス、非一時的１３０命令テンプレートが存在するように図示されている。図１Ｂ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１０５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、部分的なラウンド制御タイプ演算１１２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、ｖｓｉｚｅタイプ演算１１７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１２０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１２７命令テンプレートが存在するように図示されている。

【００２９】

汎用ベクトル向け命令フォーマット１００は、以下に挙げられるフィールドを図１Ａおよび図１Ｂ中に図示される順序で含む。

【００３０】

フォーマットフィールド１４０。このフィールド内の特定の値（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内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。

【００３１】

ベース演算フィールド１４２。その内容が、異なるベース演算を区別する。

【００３２】

レジスタインデックスフィールド１４４。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、ＰｘＱ（例えば、３２×５１２、１６×１２８、３２×１０２４、６４×１０２４）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）。

【００３３】

修飾子フィールド１４６。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１０５命令テンプレートおよびメモリアクセス１２０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。

【００３４】

拡張演算フィールド１５０。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１６８、アルフ

ァフィールド１５２およびベータフィールド１５４に分割される。拡張演算フィールド１５０は、２、３または４個の命令ではなく、単一の命令の中で共通の演算グループが実行されることを可能にする。

【００３５】

スケールフィールド１６０。その内容が、メモリアドレス生成のための（例えば、２^{scale}＊インデックス＋ベースを使用するアドレス生成のための）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。

【００３６】

変位フィールド１６２Ａ。その内容が、メモリアドレス生成（例えば、２^{scale}＊インデックス＋ベース＋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。

【００３７】

変位係数フィールド１６２Ｂ（変位係数フィールド１６２Ｂの直接の上位に、変位フィールド１６２Ａ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）のサイズ分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（例えば、２^{scale}＊インデックス＋ベース＋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１７４（本明細書で後述の）およびデータ操作フィールド１５４Ｃ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１６２Ａおよび変位係数フィールド１６２Ｂは、それらがメモリアクセスなし１０５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。

【００３８】

データ要素幅フィールド１６４。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。

【００３９】

書き込みマスクフィールド１７０。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、要素のスパンが第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１７０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１７０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１７０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１７０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。

【００４０】

即値フィールド１７２。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。

【００４１】

クラスフィールド１６８。その内容が、異なるクラスの命令間を区別する。図１Ａおよび図１Ｂ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１Ａおよび図１Ｂ中、特定の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１Ａおよび図１Ｂ中、クラスフィールド１６８に対し、それぞれクラスＡ　１６８Ａおよびクラス　Ｂ　１６８Ｂ）。

［クラスＡの命令テンプレート］

【００４２】

クラスＡのメモリアクセスなし１０５命令テンプレートの場合、アルファフィールド１５２はＲＳフィールド１５２Ａとして解釈され、ＲＳフィールド１５２Ａ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１５２Ａ．１およびデータ変換１５２Ａ．２がそれぞれ、メモリアクセスなし、ラウンドタイプ演算１１０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１１５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１５４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１０５命令テンプレートには、スケールフィールド１６０、変位フィールド１６２Ａおよび変位スケールフィールド１６２Ｂは存在しない。

［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］

【００４３】

メモリアクセスなしの完全ラウンド制御タイプ演算１１０命令テンプレートでは、ベータフィールド１５４はラウンド制御フィールド１５４Ａとして解釈され、ラウンド制御フィールド１５４Ａ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１５４Ａは、すべての浮動小数点の例外を抑制（ＳＡＥ）フィールド１５６およびラウンド演算制御フィールド１５８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１５８のみを有してよい）。

【００４４】

ＳＡＥフィールド１５６。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１５６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。

【００４５】

ラウンド演算制御フィールド１５８。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１５８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１５０の内容で、そのレジスタ値を上書きする。

【００４６】
［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］
【００４７】
メモリアクセスなしのデータ変換タイプ演算１１５命令テンプレートでは、ベータフィールド１５４はデータ変換フィールド１５４Ｂとして解釈され、データ変換フィールド１５４Ｂ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。
【００４８】
クラスＡのメモリアクセス１２０命令テンプレートの場合、アルファフィールド１５２はエビクションヒントフィールド１５２Ｂとして解釈され、エビクションヒントフィールド１５２Ｂ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１Ａ中、一時的１５２Ｂ．１および非一時的１５２Ｂ．２がそれぞれ、メモリアクセスの一時的１２５命令テンプレートおよびメモリアクセスの非一時的１３０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１５４はデータ操作フィールド１５４Ｃとして解釈され、データ操作フィールド１５４Ｃ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１２０命令テンプレートは、スケールフィールド１６０を含み、随意で変位フィールド１６２Ａまたは変位スケールフィールド１６２Ｂを含む。
【００４９】
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。
［メモリアクセス命令テンプレート－一時的］
【００５０】
一時的データとは、キャッシュの利益を十分得るべく、間もなく再使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。
［メモリアクセス命令テンプレート－非一時的］
【００５１】
非一時的データとは、第１のレベル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。
［クラスＢの命令テンプレート］
【００５２】
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１５２は書き込みマスク制御（Ｚ）フィールド１５２Ｃとして解釈され、書き込みマスク制御（Ｚ）フィールド１５２Ｃの内容が、書き込みマスクフィールド１７０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。
【００５３】
クラスＢのメモリアクセスなし１０５命令テンプレートの場合、ベータフィールド１５４の一部はＲＬフィールド１５７Ａとして解釈され、ＲＬフィールド１５７Ａ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１５７Ａ．１およびベクトル長（ＶＳＩＺＥ）１５７Ａ．２がそれぞれ、メモリアクセスなし、書き込みマスク制御、部分的なラウンド制御タイプ演算１１２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、ＶＳＩＺＥタイプ演算１１７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１５４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１０５命令テンプレートには、スケールフィールド１６０、変位フィールド１６２Ａおよび変位スケールフィールド１６２Ｂが存在しない。

【００５４】

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１１０命令テンプレートでは、ベータフィールド１５４の残部はラウンド演算フィールド１５９Ａ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）。

【００５５】

ラウンド演算制御フィールド１５９Ａ。まさにラウンド演算制御フィールド１５８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１５９Ａ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本発明の一実施形態において、ラウンド演算制御フィールド１５０の内容で、そのレジスタ値を上書きする。

【００５６】

メモリアクセスなし、書き込みマスク制御、ＶＳＩＺＥタイプ演算１１７命令テンプレートでは、ベータフィールド１５４の残部はベクトル長フィールド１５９Ｂとして解釈され、ベクトル長フィールド１５９Ｂ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。

【００５７】

クラスＢのメモリアクセス１２０命令テンプレートの場合、ベータフィールド１５４の一部はブロードキャストフィールド１５７Ｂとして解釈され、ブロードキャストフィールド１５７Ｂ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否かを区別し、一方で、ベータフィールド１５４の残部はベクトル長フィールド１５９Ｂとして解釈される。メモリアクセス１２０命令テンプレートは、スケールフィールド１６０を含み、随意で変位フィールド１６２Ａまたは変位スケールフィールド１６２Ｂを含む。

【００５８】

汎用ベクトル向け命令フォーマット１００に関しては、フルオペコードフィールド１７４は、フォーマットフィールド１４０、ベース演算フィールド１４２およびデータ要素幅フィールド１６４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１７４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１７４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１７４は、オペレーションコード（オペコード）を提供する。

【００５９】

拡張演算フィールド１５０、データ要素幅フィールド１６４および書き込みマスクフィールド１７０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。

【００６０】

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。

【００６１】

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の高性能な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、本発明の範囲内に属する）。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。

Ｂ．［例示的な特定ベクトル向け命令フォーマット］

【００６２】

図２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すブロック図である。図２は特定ベクトル向け命令フォーマット２００を示す。特定ベクトル向け命令フォーマット２００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的である。特定ベクトル向け命令フォーマット２００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機能（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、いくつかの拡張機能を備えた既存の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１のフィールドが図２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。

【００６３】

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１００に照らし特定ベクトル向け命令フォーマット２００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２００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２００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１００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１６４は、特定ベクトル向け命令フォーマット２００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１００は、データ要素幅フィールド１６４の他のサイズを想定している）。

【００６４】

特定ベクトル向け命令フォーマット２００は、以下に挙げられるフィールドを図２Ａに図示される順序で含む。

【００６５】

ＥＶＥＸプレフィクス（バイト０‐３）２０２。これは４バイト形式でエンコードされる。

【００６６】

フォーマットフィールド１４０（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１４０であり、フォーマットフィールド１４０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）。

【００６７】

第２から第４のバイト（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。

【００６８】

ＲＥＸフィールド２０５（ＥＶＥＸバイト１、ビット［７－５］）。これは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１５７ＢＥＸバイト１、ビット［５］－Ｂから成る。ＥＶＥＸ．Ｒビットフィールド、ＥＶＥＸ．Ｘビットフィールドおよび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１１１１Ｂとしてエンコードされ、ＺＭＭ１５は０００００Ｂ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、ＥＶＥＸ．Ｒ、ＥＶＥＸ．ＸおよびＥＶＥＸ．Ｂを追加することによって形成されてよい。

【００６９】

ＲＥＸ'フィールド１１０。これはＲＥＸ'フィールド１１０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'Ｒｒｒｒが形成される。

【００７０】

オペコードマップフィールド２１５（ＥＶＥＸバイト１、ビット［３：０］－ｍｍｍｍ）。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。

【００７１】

データ要素幅フィールド１６４（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）。

【００７２】

ＥＶＥＸ．ｖｖｖｖ２２０（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２２０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。

【００７３】

ＥＶＥＸ．Ｕ　１６８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。

【００７４】

プレフィクスエンコーディングフィールド２２５（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。

【００７５】

アルファフィールド１５２（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。

【００７６】

ベータフィールド１５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ。これは$ＥＶＥＸ．ｓ_{2-0}$、$ＥＶＥＸ．ｒ_{2-0}$、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。

【００７７】

ＲＥＸ'フィールド１１０。これはＲＥＸ'フィールドの残部であり、ＲＥＸ'フィールド１１０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．ｖｖｖｖを組み合わせることにより、Ｖ'ＶＶＶＶが形成される。

【００７８】

書き込みマスクフィールド１７０（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）。

【００７９】

リアルオペコードフィールド２３０（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。

【００８０】

ＭＯＤ　Ｒ／Ｍフィールド２４０（バイト５）は、ＭＯＤフィールド２４２、Ｒｅｇフ

ィールド２４４およびＲ／Ｍフィールド２４６を含む。上記の通り、ＭＯＤフィールド２４２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２４４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２４６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。

【００８１】

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１５０の内容は、メモリアドレス生成に使用される。ＳＩＢ．ｘｘｘ２５４およびＳＩＢ．ｂｂｂ　２５６。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。

【００８２】

変位フィールド１６２Ａ（バイト７‐１０）。ＭＯＤフィールド２４２に１０が含まれる場合、バイト７‐１０は変位フィールド１６２Ａであり、変位フィールド１６２Ａ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。

【００８３】

変位係数フィールド１６２Ｂ（バイト７）。ＭＯＤフィールド２４２に０１が含まれる場合、バイト７は変位係数フィールド１６２Ｂである。このフィールドの場所は、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１６２Ｂはｄｉｓｐ８と再解釈される。変位係数フィールド１６２Ｂ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１６２Ｂ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１６２Ｂ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、エンコーディング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だけ変位をスケーリングする必要がある）。

【００８４】

即値フィールド１７２は、上記の通り動作する。

［フルオペコードフィールド］

【００８５】

図２Ｂ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２００のフルオペコードフィールド１７４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１７４は、フォーマットフィールド１４０、ベース演算フィールド１４２およびデータ要素幅（Ｗ）フィールド１６４を含む。ベース演算フィールド１４２は、プレフィクスエンコーディングフィールド２２５、オペコードマップフィールド２１５およびリアルオペコードフィールド２３０を含む。

［レジスタインデックスフィールド］

【００８６】

図２Ｃ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２００のレジスタインデックスフィールド１４４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１４４は、ＲＥＸフィールド２０５、ＲＥＸ'フィールド２１０、ＭＯＤＲ／Ｍ．ｒｅｇフィールド２４４、ＭＯＤＲ／Ｍ．ｒ／ｍフィールド２４６、ＶＶＶＶフィールド２２０、ｘｘｘフィールド２５４およびｂｂｂフィールド２５６を含む。

［拡張演算フィールド］

【００８７】

図２Ｄ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２００の拡張演算フィールド１５０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１６８が０を含む場合、それはＥＶＥＸ．Ｕ０（クラスＡ　１６８Ａ）を表わす。クラス（Ｕ）フィールド１６８が１を含む場合、それはＥＶＥＸ．Ｕ１（クラスＢ　１６８Ｂ）を表わす。Ｕ＝０で且つＭＯＤフィールド２４２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１５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ）は、ｒｓフィールド１５２Ａとして解釈される。ｒｓフィールド１５２Ａが１を含む場合（ラウンド１５２Ａ．１）、ベータフィールド１５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ）はラウンド制御フィールド１５４Ａとして解釈される。ラウンド制御フィールド１５４Ａは、１ビットのＳＡＥフィールド１５６および２ビットのラウンド演算フィールド１５８を含む。ｒｓフィールド１５２Ａが０を含む場合（データ変換１５２Ａ．２）、ベータフィールド１５４（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１５４Ｂとして解釈される。Ｕ＝０で且つＭＯＤフィールド２４２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１５２（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１５２Ｂとして解釈され、ベータフィールド１５４（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１５４Ｃとして解釈される。

【００８８】

Ｕ＝１の場合、アルファフィールド１５２（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１５２Ｃとして解釈される。Ｕ＝１で且つＭＯＤフィールド２４２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１５４の一部（ＥＶＥＸバイト３、ビット［４］－$Ｓ_{0}$）は、ＲＬフィールド１５７Ａとして解釈される。ＲＬフィールド１５７Ａが１を含む場合（ラウンド１５７Ａ．１）、ベータフィールド１５４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１５９Ａとして解釈され、一方で、ＲＬフィールド１５７Ａが０を含む場合（ＶＳＩＺＥ　１５７．Ａ２）、ベータフィールド１５４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１５９Ｂ（ＥＶＥＸバイト３、ビット［６－５］－$Ｌ_{1-0}$）として解釈される。Ｕ＝１で且つＭＯＤフィールド２４２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１５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ）は、ベクトル長フィールド１５９Ｂ（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１５７Ｂ（ＥＶＥＸバイト３、ビット［４］－Ｂ）として解釈される。

Ｃ．［例示的なレジスタアーキテクチャ］

【００８９】

図３は、本発明の一実施形態による、レジスタアーキテクチャ３００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３１０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２００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159Bを含まない命令テンプレート	A(図1A; U=0)	110, 115, 125, 130	zmmレジスタ（ベクトル長は64バイト）
	B(図1B; U=1)	112	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159Bを含む命令テンプレート	B(図1B; U=1)	117, 127	ベクトル長フィールド159B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)

【００９０】

換言すると、ベクトル長フィールド１５９Ｂ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１５９Ｂを持た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２００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。

【００９１】

図示された実施形態中の書き込みマスクレジスタ３１５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３１５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キングを無効にする。

【００９２】

図示された実施形態中の汎用レジスタ３２５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。

【００９３】

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）３４５について、ＭＭＸパックド整数フラットレジスタファイル３５０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。

【００９４】

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。

Ｄ．［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］

【００９５】

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コアインオーダコア、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、その後に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。

【００９６】

図４Ａは、本発明の実施形態による、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４Ｂ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４Ａ～図４Ｂ中の実線ボックスは、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、アウトオブオーダ発行／実行パイプラインおよびコアを示す。インオーダの態様はアウトオブオーダ態様のサブセットであると想定して、アウトオブオーダ態様について以下記載する。

【００９７】

図４Ａ中、プロセッサパイプライン４００は、フェッチステージ４０２、長さデコードステージ４０４、デコードステージ４０６、割り当てステージ４０８、リネーミングステージ４１０、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４１２、レジスタ読み取り／メモリ読み取りステージ４１４、実行ステージ４１６、ライトバック／メモリ書き込みステージ４１８、例外処理ステージ４２２およびコミットステージ４２４が含まれる。

【００９８】

図４Ｂは、実行エンジンユニット４５０に連結されたフロントエンドユニット４３０を含むプロセッサコア４９０を示し、フロントエンドユニット４３０および実行エンジンユニット４５０の両方はメモリユニット４７０に連結されている。コア４９０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４９０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、汎用コンピューティンググラフ

ィック処理ユニット（ＧＰＧＰＵ）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。

【００９９】

フロントエンドユニット４３０は、命令キャッシュユニット４３４に連結された分岐予測ユニット４３２を含み、命令キャッシュユニット４３４は、命令トランスレーションルックアサイドバッファ（ＴＬＢ）４３６に連結され、ＴＬＢ４３６は命令フェッチユニット４３８に連結され、命令フェッチユニット４３８はデコードユニット４４０に連結される。デコードユニット４４０（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４４０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４９０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４４０内またはフロントエンドユニット４３０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４４０は、実行エンジンユニット４５０内のリネーム／アロケータユニット４５２に連結される。

【０１００】

実行エンジンユニット４５０は、リタイアメントユニット４５４に連結されたリネーム／アロケータユニット４５２および１または複数のスケジューラユニット４５６のセットを含む。スケジューラユニット４５６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４５６は物理レジスタファイルユニット４５８に連結される。物理レジスタファイルユニット４５８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４５８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４５８がリタイアメントユニット４５４に重ねられている（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４５４および物理レジスタファイルユニット４５８は、実行クラスタ４６０に連結される。実行クラスタ４６０は、１または複数の実行ユニット４６２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４６４のセットを含む。実行ユニット４６２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、特定の関数または関数のセットに専用に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４５６、物理レジスタファイルユニット４５８および実行クラスタ４６０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４６４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または複数はアウトオブオーダ発行／実行であってよく、残りはインオーダであってよいことも理解されたい。

【０１０１】

メモリアクセスユニット４６４のセットがメモリユニット４７０に連結され、メモリユニット４７０はレベル２（Ｌ２）キャッシュユニット４７６に連結されたデータキャッシュユニット４７４に連結されたデータＴＬＢユニット４７２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４６４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４７０内のデータＴＬＢユニット４７２に連結される。命令キャッシュユニット４３４は、メモリユニット４７０内のレベル２（Ｌ２）キャッシュユニット４７６にさらに連結される。Ｌ２キャッシュユニット４７６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。

【０１０２】

例を挙げると、例示的なレジスタリネーミング、アウトオブオーダ発行／実行コアアーキテクチャは、パイプライン４００を以下のように実装してよい。すなわち、１）命令フェッチ４３８がフェッチステージ４０２および長さデコーディングステージ４０４を実行する。２）デコードユニット４４０がデコードステージ４０６を実行する。３）リネーム／アロケータユニット４５２が割り当てステージ４０８およびリネーミングステージ４１０を実行する。４）スケジューラユニット４５６がスケジューリングステージ４１２を実行する。５）物理レジスタファイルユニット４５８およびメモリユニット４７０がレジスタ読み取り／メモリ読み取りステージ４１４を実行する。実行クラスタ４６０が実行ステージ４１６を実行する。６）メモリユニット４７０および物理レジスタファイルユニット４５８がライトバック／メモリ書き込みステージ４１８を実行する。７）様々なユニットが例外処理ステージ４２２に関与してよい。８）リタイアメントユニット４５４および物理レジスタファイルユニット４５８がコミットステージ４２４を実行する。

【０１０３】

コア４９０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くつかの拡張を持つ）、カリフォルニア州サニーベールのＭＩＰＳ　Ｔｅｃｈｎｏｌｏｇｉｅｓ　ＭＩＰＳ命令セット、カリフォルニア州サニーベールのＡＲＭ　Ｈｏｌｄｉｎｇｓ　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４９０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。

【０１０４】

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ディング並びにインテル（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。

【０１０５】

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。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４３４およびデータキャッシュユニット４７４並びに共有Ｌ２キャッシュユニット４７６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。

【０１０６】

図５Ａ～５Ｂ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（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。

【０１０７】

図５Ａ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５０２への接続を備え、且つ、レベル２（Ｌ２）キャッシュ５０４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５００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５０６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５０８およびベクトルユニット５１０は、別個のレジスタセット（それぞれスカラレジスタ５１２およびベクトルレジスタ５１４）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５０６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。

【０１０８】

Ｌ２キャッシュのローカルサブセット５０４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５０４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５０４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５０４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、一方向当たり１０１２ビット幅である。

【０１０９】

図５Ｂは、本発明の実施形態による、図５Ａのプロセッサコアの一部の拡大図である。図５Ｂには、Ｌ１キャッシュ５０４の一部であるＬ１データキャッシュ５０６Ａに加え、ベクトルユニット５１０およびベクトルレジスタ５１４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５１０は、１６幅ベクトル処理ユニット（ＶＰＵ）（１６幅ＡＬＵ　５２８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５２０を用いるレジスタ入力のスウィズル、数値変換ユニット５２２Ａ～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５２４を用いる複製をサポートする。書き込みマスクレジスタ５２６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。

【０１１０】

図６は、本発明の実施形態による、プロセッサ６００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６中の実線ボックスは、単一のコア６０２Ａ、システムエージェント６１０、１または複数のバスコントローラユニット６１６のセットを備えたプロセッサ６００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６０２Ａ～Ｎ、システムエージェントユニット６１０内の１または複数の統合メモリコントローラユニット６１４のセット、および専用ロジック６０８を備えた代替的なプロセッサ６００を示す。

【０１１１】

故に、プロセッサ６００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６０８と、１または複数の汎用コアであるコア６０２Ａ～Ｎ（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）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６０２Ａ～Ｎを有するコプロセッサ、並びに３）多数の汎用インオーダコアであるコア６０２Ａ～Ｎを有するコプロセッサ、を含んでよい。故に、プロセッサ６００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６００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。

【０１１２】

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６０６のセットまたは１若しくは複数の共有キャッシュユニット６０６、および統合メモリコントローラユニット６１４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６０６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６１２は、統合グラフィックロジック６０８、共有キャッシュユニット６０６のセットおよびシステムエージェントユニット６１０／統合メモリコントローラユニット６１４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６０６およびコア６０２Ａ～Ｎ間で維持される。

【０１１３】

いくつかの実施形態において、コア６０２Ａ～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６１０は、コア６０２Ａ～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６１０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６０２Ａ～Ｎおよび統合グラフィックロジック６０８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。

【０１１４】

コア６０２Ａ～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６０２Ａ～Ｎ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。

【０１１５】

図７～図１０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。

【０１１６】

ここで図７を参照すると、本発明の一実施形態によるシステム７００のブロック図が示されている。システム７００は、１または複数のプロセッサ７１０、７１５を含んでよく、当該１または複数のプロセッサ７１０、７１５は、コントローラハブ７２０に連結される。一実施形態において、コントローラハブ７２０は、グラフィックメモリコントローラハブ（ＧＭＣＨ）７９０および入／出力ハブ（ＩＯＨ）７５０（別個のチップ上に存在してよい）を含む。ＧＭＣＨ７９０は、メモリ７４０およびコプロセッサ７４５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。ＩＯＨ７５０は、入出力（Ｉ／Ｏ）デバイス７６０をＧＭＣＨ７９０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の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７４０およびコプロセッサ７４５は、プロセッサ７１０と、単一のチップ内のＩＯＨ７５０を持つコントローラハブ７２０とに直接連結される。

【０１１７】

図７中、破線を用いて、追加のプロセッサ７１５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７１０、７１５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６００の何らかのバージョンであってよい。

【０１１８】

メモリ７４０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７２０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７９５を介して、プロセッサ７１０、７１５と通信する。

【０１１９】

一実施形態において、コプロセッサ７４５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７２０は統合グラフィックアクセラレータを含んでよい。

【０１２０】

物理リソース７１０と７１５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。

【０１２１】

一実施形態において、プロセッサ７１０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７１０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７４５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７１０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７４５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７４５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。

【０１２２】

ここで図８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８００のブロック図を示す。図８に図示の通り、マルチプロセッサシステム８００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８５０を介して連結された第１のプロセッサ８７０および第２のプロセッサ８８０を含む。プロセッサ８７０および８８０の各々は、プロセッサ６００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８７０および８８０は、それぞれプロセッサ７１０および７１５である一方で、コプロセッサ８３８はコプロセッサ７４５である。別の実施形態においては、プロセッサ８７０および８８０は、それぞれプロセッサ７１０およびコプロセッサ７４５である。

【０１２３】

プロセッサ８７０および８８０は、それぞれ統合メモリコントローラ（ＩＭＣ）ユニット８７２および８８２を含むように図示されている。プロセッサ８７０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８７６および８７８を含み、同様に第２のプロセッサ８８０はＰ－Ｐインタフェース８８６および８８８を含む。プロセッサ８７０、８８０は、Ｐ－Ｐインタフェース回路８７８、８８８を使用して、ポイントツーポイント（Ｐ－Ｐ）インタフェース８５０を介して情報を交換してよい。図８に図示の通り、ＩＭＣ８７２および８８２はプロセッサをそれぞれのメモリ、すなわちメモリ８３２およびメモリ８３４に連結する。メモリ８３２およびメモリ８３４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。

【０１２４】

プロセッサ８７０、８８０はそれぞれ、ポイントツーポイントインタフェース回路８７６、８９４、８８６、８９８を使用して、個々のＰ－Ｐインタフェース８５２、８５４を介して、チップセット８９０と情報を交換してよい。随意で、チップセット８９０は、高性能インタフェース８３９を介してコプロセッサ８３８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８３８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。

【０１２５】

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。

【０１２６】

チップセット８９０が、インタフェース８９６を介して第１のバス８１６に連結されてよい。一実施形態において、第１のバス８１６はペリフェラルコンポーネントインターコネクト（ＰＣＩ）バス、すなわちＰＣＩ　Ｅｘｐｒｅｓｓ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。

【０１２７】

図８に図示の通り、様々なＩ／Ｏデバイス８１４がバスブリッジ８１８と共に第１のバス８１６に連結されてよく、バスブリッジ８１８は第１のバス８１６を第２のバス８２０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ＭＩＣプロセッサ、ＧＰＧＰＵ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（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８１５が第１のバス８１６に連結される。一実施形態において、第２のバス８２０はローピンカウント（ＬＰＣ）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８２０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８２２、通信デバイス８２７および命令／コードおよびデータ８３０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８２８が含まれる。さらに、オーディオＩ／

Ｏ８２４が第２のバス８２０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８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。

【０１２８】

ここで図９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９００のブロック図が示されている。図８および図９中で同様の要素は同様の参照符号が付されており、図９の他の態様を不明瞭にするのを回避すべく、図８の特定の態様は図９で省略されている。

【０１２９】

図９は、プロセッサ８７０、８８０が統合メモリおよびＩ／Ｏ制御ロジック（「ＣＬ」）８７２および８８２をそれぞれ含んでよいことを示す。故に、ＣＬ８７２、８８２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９は、メモリ８３２、８３４がＣＬ８７２、８８２に連結されるだけでなく、Ｉ／Ｏデバイス９１４も制御ロジック８７２、８８２に連結されることも示している。レガシＩ／Ｏデバイス９１５がチップセット８９０に連結される。

【０１３０】

ここで図１０を参照すると、本発明の一実施形態によるＳｏＣ　１０００のブロック図が示されている。図６中と同様の要素は同一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ＳｏＣ上でのオプションの機能である。図１０中、相互接続ユニット１００２は、アプリケーションプロセッサ１０１０と、システムエージェントユニット６１０と、バスコントローラユニット６１６と、統合メモリコントローラユニット６１４と、コプロセッサ１０２０のセットまたは１若しくは複数のコプロセッサ１０２０と、スタティックランダムアクセスメモリ（ＳＲＡＭ）ユニット１０３０と、ダイレクトメモリアクセス（ＤＭＡ）ユニット１０３２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１０４０とに連結される。アプリケーションプロセッサ１０１０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６０６を含む。コプロセッサ１０２０のセットまたは１若しくは複数のコプロセッサ１０２０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１０２０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、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ような専用プロセッサを含む。

【０１３１】

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。

【０１３２】

図８に図示されたコード８３０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。

【０１３３】

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。

【０１３４】

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。

【０１３５】

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。

【０１３６】

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。

【０１３７】

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。

【０１３８】

図１１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１１は、高水準言語１１０２のプログラムが、ｘ８６バイナリコード１１０６を生成するｘ８６コンパイラ１１０４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１１０６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１１１６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１１１６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１１０４は、ｘ８６バイナリコード１１０６（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１１０６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１１１６上で実行可能である。同様に、図１１は、高水準言語１１０２のプログラムが、代替的な命令セットバイナリコード１１１０を生成する代替的な命令セットコンパイラ１１０８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１１１０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１１１４（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１１１２は、ｘ８６バイナリコード１１０６を、ｘ８６命令セットコアを持たないプロセッサ１１１４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１１１０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１１１２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１１０６を実行できるようにする。

［ベクトルビットシャッフルを実行するための方法および装置］

【０１３９】

ベクトルビットシャッフル命令について以下説明する。当該命令は、第１のソースオペランドを制御として、第２のソースオペランドをデータとして使用し、ビットシャッフルを実行する。この命令は、複数のビット操作ルーチンを実装するために効率的に使用されてよい。例示であり限定ではないが、当該命令を使用して、可変ビット置換を実装し、現行のＶＥＸ実装またはＥＶＥＸ実装に対し、最大８倍の速度アップをもたらしてよい。

【０１４０】

図１２に図示の通り、本発明の実施形態が実装されてよい例示的なプロセッサ１２５５は、汎用レジスタ（ＧＰＲ）セット１２０５、ベクトルレジスタセット１２０６、およびマスクレジスタセット１２０７を含む。一実施形態において、複数のベクトルデータ要素が、２個の２５６ビット値、４個の１２８ビット値、８個の６４ビット値、１６個の３２ビット値等を格納するための５１２ビット幅を有してよい各ベクトルレジスタ１２０６内にパックされる。しかしながら、本発明の根本的な原理は、いずれの特定のサイズ／タイプのベクトルデータにも限定されない。一実施形態において、マスクレジスタ１２０７は、ベクトルレジスタ１２０６内に格納された値にビットマスク演算を実行するために使用される８個の６４ビットオペランドマスクレジスタ（例えば、上記のマスクレジスタｋ０～ｋ７として実装される）を含む。しかしながら、本発明の根本的な原理は、いずれの特定のマスクレジスタのサイズ／タイプにも限定されない。

【０１４１】

簡単にするために、単一のプロセッサコア（「コア０」）の詳細が図１２中に示されている。しかしながら、図１２に図示の各コアは、コア０と同一のロジックセットを有してよいことを理解されたい。例えば、各コアは、指定されたキャッシュ管理ポリシーに従い、命令およびデータをキャッシュするための専用のレベル１（Ｌ１）キャッシュ１２１２およびレベル２（Ｌ２）キャッシュ１２１１を含んでよい。Ｌ１キャッシュ１２１２は、命令を格納するための別個の命令キャッシュ１２２０およびデータを格納するための別個のデータキャッシュ１２２１を含む。様々なプロセッサキャッシュ内に格納された命令およびデータは、固定サイズ（例えば、６４、１２８、５１２バイト長）であってよいキャッシュラインの粒度で管理される。この例示的な実施形態の各コアは、メインメモリ１２

００および／または共有レベル３（Ｌ３）キャッシュ１２１６から命令をフェッチするための命令フェッチユニット１２１０、命令をデコーディング（例えば、プログラム命令をマイクロオペレーションまたは「μｏｐ」へとデコーディング）するためのデコードユニット１２２０、命令を実行するための実行ユニット１２４０、および命令をリタイアし、結果をライトバックするためのライトバックユニット１２５０を有する。

【０１４２】

命令フェッチユニット１２１０は、メモリ１２００（または複数のキャッシュのうちの１つ）からフェッチされる次の命令のアドレスを格納するための次の命令ポインタ１２０３、アドレス変換速度を改善すべく最近使用された仮想命令アドレスと物理命令アドレスのマップを格納するための命令トランスレーションルックアサイドバッファ（ＩＴＬＢ）１２０４、命令分岐アドレスを投機的に予測するための分岐予測ユニット１２０２、および分岐アドレスおよびターゲットアドレスを格納するための分岐ターゲットバッファ（ＢＴＢ）１２０１を含む、様々な周知のコンポーネントを含む。いったんフェッチされた命令は、その後デコードユニット１２３０、実行ユニット１２４０およびライトバックユニット１２５０を含む命令パイプラインの残りのステージにストリームされる。これらのユニットの各々の構造および機能は当業者に十分理解されており、本発明の異なる実施形態の関連態様を不明瞭にするのを避けるべく、ここでは詳細に記載しない。

【０１４３】

一実施形態において、デコードユニット１２３０は、本明細書に記載のベクトルビットシャッフル命令を（例えば、一実施形態において、一連のマイクロオペレーションへと）デコーディングするためのベクトルビットシャッフルデコードロジック１２３１を含み、実行ユニット１２４０は、当該命令を実行するためのベクトルビットシャッフル実行ロジック１２４１を含む。

【０１４４】

上記の通り、一実施形態において、ベクトルビットシャッフル命令は、第１のソースを制御として、第２のソースをデータとして使用し、結果をデスティネーションレジスタ内に出力する、ビットギャザー選択を実行する。一実施形態において、デスティネーションの各ビットは、第１のソースからの６つの制御ビットを使用して、第２のソースから識別される。

【０１４５】

図１３は例示的な実施形態を示し、そこには、制御ビットを格納するための第１のソースレジスタであるＳＲＣ２、ソースデータを格納するための第２のソースレジスタであるＳＲＣ３、およびベクトルビットシャッフル命令の結果を格納するためのデスティネーションレジスタであるＤＳＴが含まれる。一実施形態において、ＳＲＣ３は、５１２ビットベクトルレジスタ内にパックされた６４ビットデータの８個のレーン０‐７を含み、ＳＲＣ２は、これもまた５１２ビットベクトルレジスタ内にパックされた８個の制御バイトから成る８個のセット１３００～１３０２...１３０７を含み、ＤＳＴは、６４ビットのマスクレジスタ１３２０を含む。しかしながら、上記の通り、本発明の根本的な原理は、いずれの特定のサイズ／タイプのオペランドまたはレジスタにも限定されない。簡略にするため、図１３には、ＳＲＣ３内に格納されたデータおよびＳＲＣ２内に格納された制御ビットの一部のみが図示されていることに留意されたい。

【０１４６】

演算では、８個の制御バイトから成る各セットの各バイトが、その対応する６４ビットレーン内の特定のビットを識別する。従って、８個の制御バイト１３００の各々は、レーン０内のビットを識別し、８個の制御バイト１３０１の各々は、レーン１内のビットを識別するといった具合である。一実施形態において、各制御バイト１３００～１３０２...１３０７において、８ビットのうちの６ビットのみが使用され、ＳＲＣ３内のビットを識別する（2^6＝６４なので、６ビットで十分である）。残りの２ビットは無視されてよい。

【０１４７】

一実施形態において、８個の制御バイト１３００～１３０７の各々によるこれらの６ビットが、選択ロジック１３１０～１３１２...１３１７（例えば、マルチプレクサのセット）に適用され、各レーンから８ビットが選択される。従って、選択ロジック１３１０はレーン０から８ビットを選択し、選択ロジック１３１１はレーン１から８ビットを選択し、選択ロジック１３１２はレーン２から８ビットを選択し、選択ロジック１３１７はレーン７から８ビットを選択する（上記の通り、簡略化のため、レーン３～６および関連の選択ロジックは図示されていない）。

【０１４８】

最終結果は、８ビットの８セットがＳＲＣ３から読み出されるということである。一実施形態において、当該８ビットの８セットがＤＳＴ内で共に連結され、６４ビットマスク値１３２０を形成する。いったん形成された６４ビットマスク値１３２０は、後続のマスク演算に使用されてよい。

【０１４９】

図１４に、本発明の一実施形態による方法が図示されている。方法は、上記のアーキテクチャの文脈において実行されてよいが、当該方法はいずれの特定のシステムアーキテクチャにも限定されない。

【０１５０】

１４０１において、ベクトルビットシャッフル命令がシステムメモリからフェッチされ、またはキャッシュ（例えば、Ｌ１、Ｌ２またはＬ３キャッシュ）から読み出される。１４０２において、ベクトルビットシャッフル命令のデコーディング／実行に応答して、シャッフルされる入力ベクトルデータが第１のソースレジスタ内に格納される。上記の通り、一実施形態において、第１のソースレジスタは５１２ビットベクトルレジスタであり、ベクトルデータは８個の６４ビットデータレーンを含む。１４０３において、ベクトルビットシャッフルを実行するために必要な制御データが第２のソースレジスタ内に格納され、第２のソースレジスタは上記の通り、別の５１２ビットベクトルレジスタであってよい。

【０１５１】

１４０４において、第１のソースレジスタ内の各レーンからビットのセットが、第２のソースレジスタ内の関連付けられた制御ビットのセットを使用して、識別される。上記の通り、一実施形態において、各レーンに８個の制御バイトが提供され、各制御バイトのうちの６ビットが使用され、対応するレーンのビットを識別する。最終結果は、第１のソースレジスタから８ビットの８セットが読み出されるということである。最後に、１４０５において、デスティネーションマスクレジスタ内で当該ビットのセットが連結される。上記の実施形態においては、例えば、８ビットの８セットが連結され、６４ビットのマスク値を形成する。

【０１５２】

一実施形態において、ＥＶＥＸエンコード実装においては、第１のソースオペランド、第２のソースのオペランドおよびデスティネーションオペランドはすべてＺＭＭレジスタである。一実施形態において、ベクトルビットシャッフル命令は以下の形態を取る。ここでＤＥＳＴはデスティネーション、ＳＲＣ２は制御データを含むソースを含み、ＳＲＣ３はシャッフルされるデータを含むソースを含む。

【数１】

VPSHUFBITQMB DEST, SRC2, SRC3

【０１５３】

以下の擬似コードは、本発明の一実施形態により実行される演算の典型例を提供する。

【数２】

```
VSHUFBITQMB DEST, SRC2, SRC3
(KL, VL) = (16, 128), (32, 256), (64, 512)
FOR i:= 0 to KL/8 – 1; Qword
        FOR j:= 0 to 7; Byte
                IF EVEX.b AND SRC3 *is memory*
                THEN
                        Data:= SRC3.qword[0];
                ELSE
                        Data:= SRC3.qword[i];
                m:= SRC2.qword[i].byte[j] & 0x3F
                k1[i*8+j]:=Data.bit[m]
        END FOR
END FOR
k1[MAX_KL-1:KL]:=0
```

【０１５４】

従って、ＫＬ＝６４且つＶＬ＝５１２と想定すると、外側ＦＯＲループ（ｉに基づく）は異なる６４ビットレーン（Ｑワード）の各々を選択するために使用され、内側ＦＯＲループ（ｊに基づく）は制御バイトで指定された制御値を使用して各レーン内の８ビットを選択するために使用される。「EVEX.b AND SRC3 *is memory*」を持つＩＦステートメントは、「ｂ」ビットがＥＶＥＸビットフィールドに設定される場合（通常、ソースブロードキャスト、ラウンド制御（Ｌ'Ｌとの組み合わせ）または抑制例外に使用）且つソースデータがシステムメモリから読み出されている場合、単一の６４ビットソース値がすべてのレーン（（ＫＬ，ＶＬ）＝（６４，５１２））にコピーされることを示す。さもなければ、使用されるクワッドワード／レーンは、ｉの現在の値（Data:= SRC3.qword[i]）に基づいて選択される。また、各６４ビットレーン内のビット値を識別するために、各制御バイトのうちの６ビットのみが使用されるので、０ｘ３Ｆの値はインデックス値SRC2.qword[i].byte[j]とＡＮＤ演算される（すなわち、０ｘ３ＦとのＡＮＤ演算で上位２ビットを除去する）。

【０１５５】

上記の明細書において、本発明の実施形態は、本発明の具体的な例示的実施形態を参照して記載されている。しかしながら、添付の特許請求の範囲に記載の本発明のより広範な精神および範囲から逸脱することなく、様々な修正および変更がそこに加えられ得ることは自明であろう。従って、明細書および図面は限定的な意味ではなく、例示的な意味において解釈されるべきである。

【０１５６】

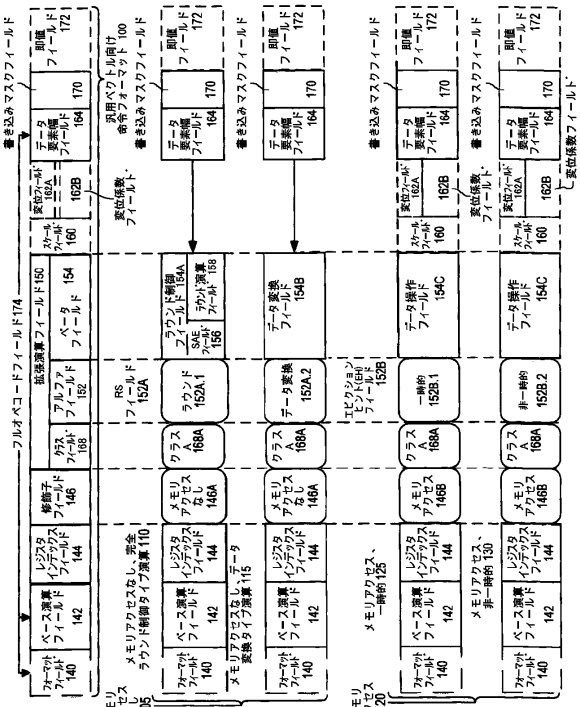
本発明の実施形態は、上記の様々な段階を含んでよい。当該段階は機械で実行可能な命令に具現化されてよく、当該命令を使用して、汎用プロセッサまたは専用プロセッサに当該段階を実行させてよい。代替的に、これらの段階は具体的なハードウェアコンポーネン

トによって実行されてよく、当該ハードウェアコンポーネントは、当該段階を実行するためのハードワイヤードされたロジックを含む。またはこれらの段階はプログラムされたコンピュータコンポーネントおよびカスタムのハードウェアコンポーネントの任意の組み合わせによって実行されてよい。

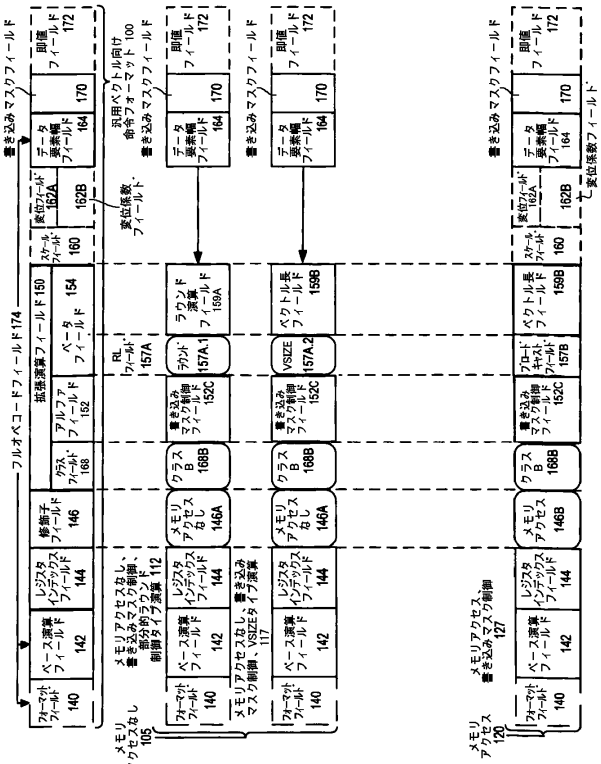
【０１５７】

本明細書で上記の通り、命令とは、特定の処理を実行するように構成された若しくは予め定められた機能を有する特定用途向け集積回路（ＡＳＩＣ）等のハードウェアの特定の構成、または、非一時的コンピュータ可読媒体に具現化されたメモリ内に格納されたソフトウェア命令を指してよい。故に、図面中に図示された技術は、１または複数の電子デバイス（例えば、エンドステーション、ネットワーク要素等）に格納され、当該電子デバイス上で実行されるコードおよびデータを使用して実装可能である。このような電子デバイスは、コンピュータ機械可読媒体を使用してコードおよびデータを格納および通信し（内部的におよび／またはネットワーク経由で他の電子デバイスと共に）、このようなコンピュータ機械可読媒体としては、非一時的コンピュータ機械可読記録媒体（例えば、磁気ディスク、光ディスク、ランダムアクセスメモリ、リードオンリメモリ、フラッシュメモリデバイス、相変化メモリ）および一時的コンピュータ機械可読通信媒体（例えば、搬送波、赤外線信号、デジタル信号等、電気、光、音響または他の形態の伝搬信号）が挙げられる。また、このような電子デバイスは通常、１または複数のストレージデバイス（非一時的機械可読記録媒体）、ユーザ入力／出力デバイス（例えば、キーボード、タッチスクリーンおよび／またはディスプレイ）およびネットワーク接続等の１または複数の他のコンポーネントに連結された１または複数のプロセッサのセットを含む。プロセッサのセットと他のコンポーネントとの連結は通常、１または複数のバスおよびブリッジ（またバスコントローラとも呼ばれる）を介してなされる。ストレージデバイスおよびネットワークトラフィックを搬送する信号はそれぞれ、１または複数の機械可読記録媒体および機械可読通信媒体を表わす。故に、特定の電子デバイスのストレージデバイスは通常、その電子デバイスの１または複数のプロセッサのセット上で実行されるためのコードおよび／またはデータを格納する。もちろん、本発明の実施形態に係る１または複数の部分が、ソフトウェア、ファームウェア、および／またはハードウェアの異なる組み合わせを使用して実装されてもよい。詳細な説明にわたり、本発明の完全な理解を共すべく、多数の具体的な詳細が説明目的で記載された。しかしながら、本発明はこれらの具体的な詳細の一部を省いても実施可能であることは当業者に自明なところである。特定の例においては、本発明の主題を不明瞭にするのを避けるべく、周知の構造および機能は詳細には記載されていない。従って、本発明の範囲および精神は以降の特許請求の範囲に照らし判断するものとする。

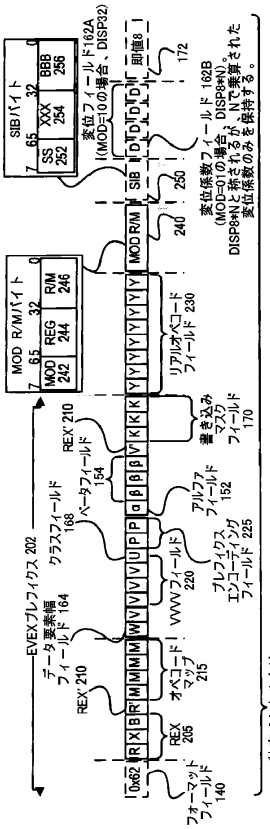
【図１Ａ】



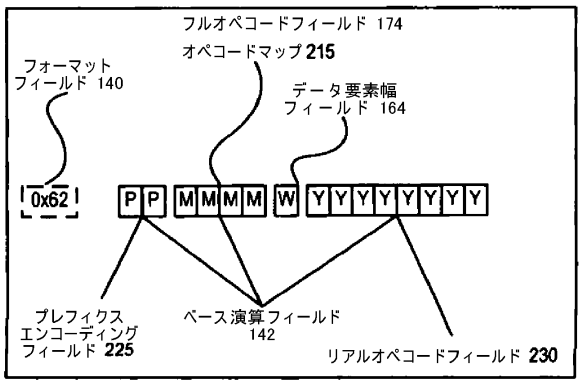
【図１Ｂ】



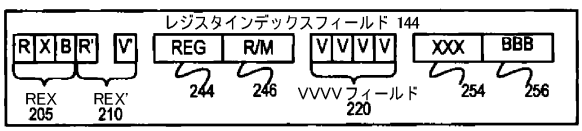
【図２Ａ】



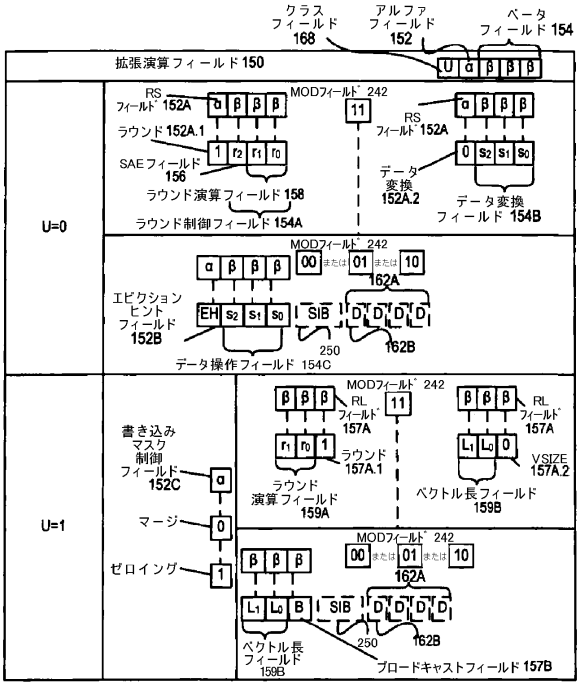
【図２Ｂ】



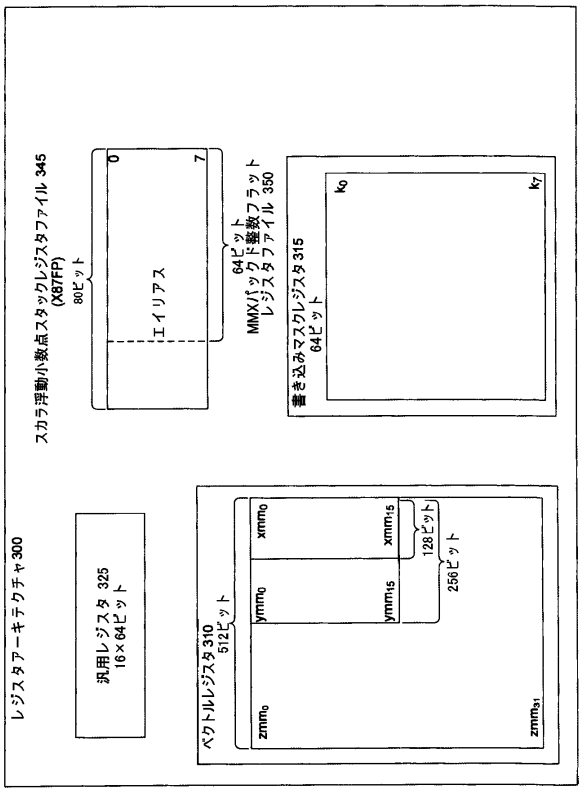
【図２Ｃ】



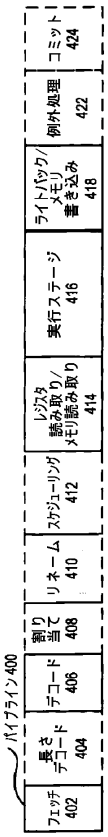
【図２Ｄ】



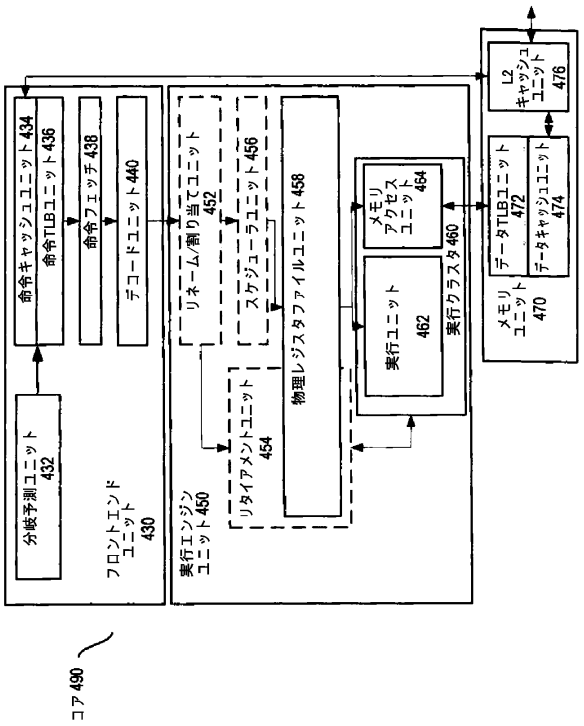
【図３】



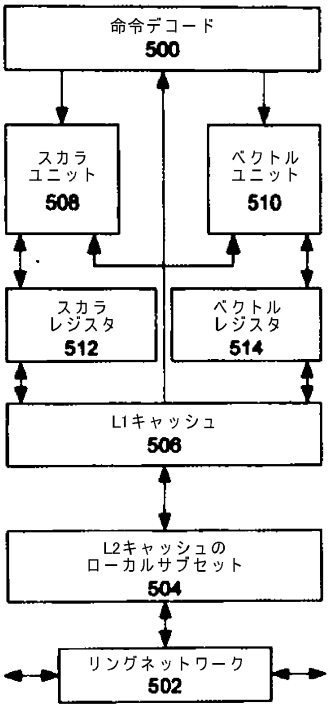
【図４Ａ】



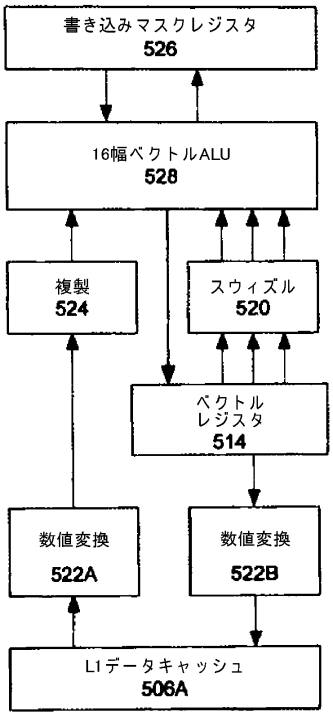
【図４Ｂ】



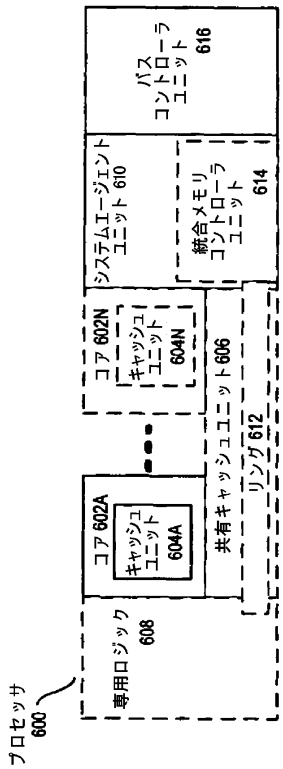
【図５Ａ】


命令デコード
500
スカラ
ユニット
508
ベクトル
ユニット
510
スカラ
レジスタ
512
ベクトル
レジスタ
514
L1キャッシュ
506
L2キャッシュの
ローカルサブセット
504
リングネットワーク
502

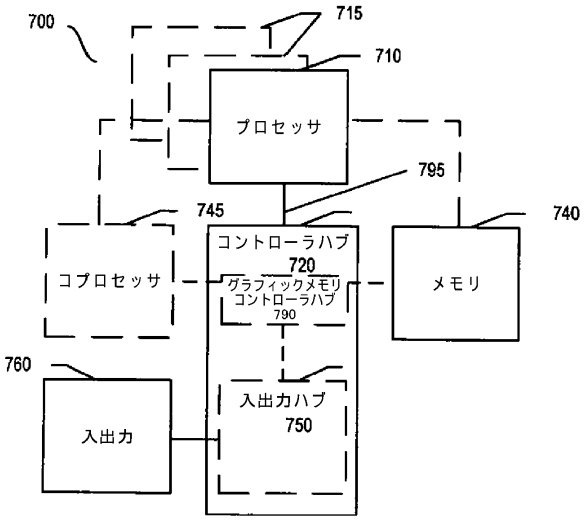
【図５Ｂ】


書き込みマスクレジスタ
526
16幅ベクトルALU
528
複製
524
スウィズル
520
ベクトル
レジスタ
514
数値変換
522A
数値変換
522B
L1データキャッシュ
506A

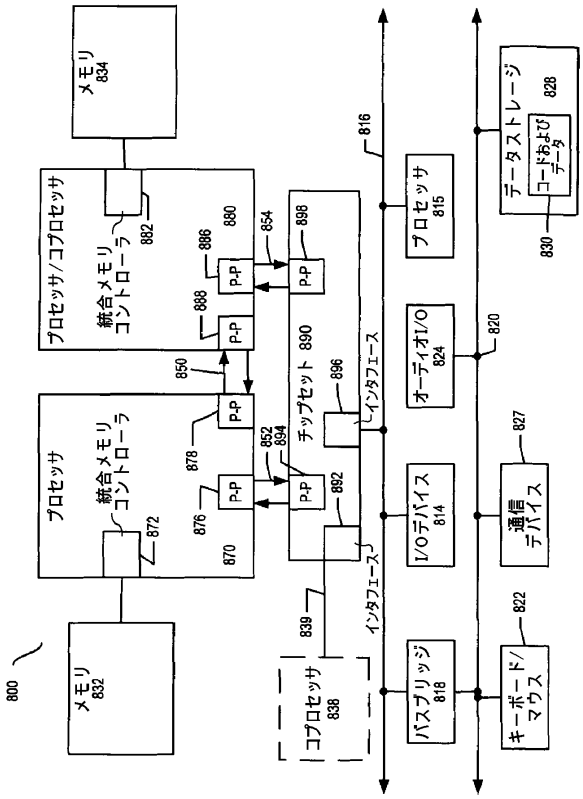
【図６】


バス
コントローラ
ユニット
616
システムエージェント
ユニット 610
統合メモリ
コントローラ
ユニット
614
コア 602N
キャッシュ
ユニット
604N
共有キャッシュユニット606
リング 612
コア 602A
キャッシュ
ユニット
604A
専用ロジック
608
プロセッサ
600

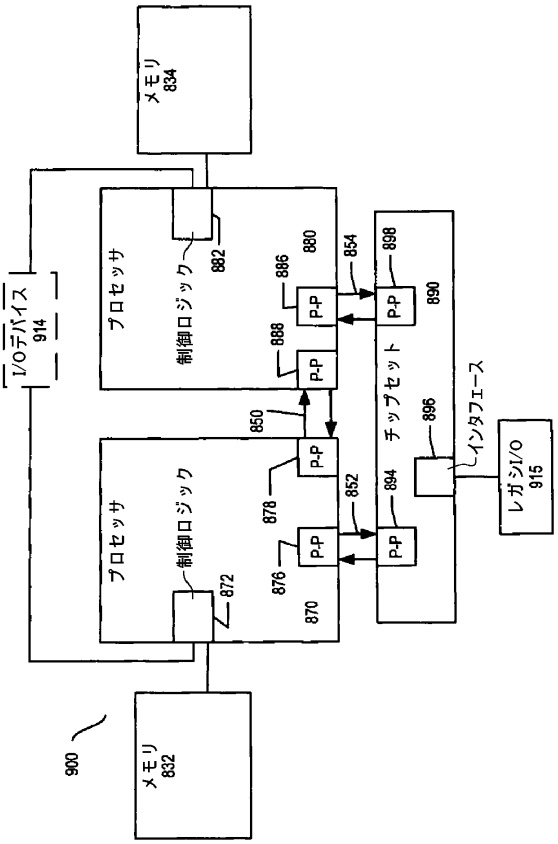
【図７】


700
715
710
プロセッサ
795
745
740
コントローラハブ
720
グラフィックメモリ
コントローラハブ
790
コプロセッサ
メモリ
760
入出力ハブ
750
入出力

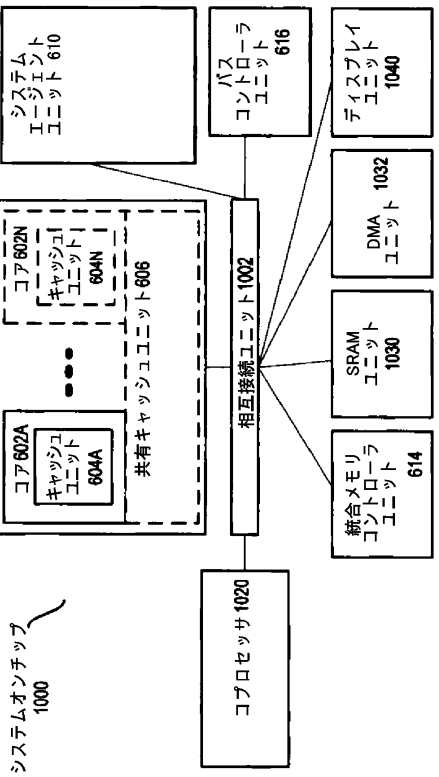
【図８】


800
メモリ 832
プロセッサ
統合メモリコントローラ 872
プロセッサ/コプロセッサ
統合メモリコントローラ 882
メモリ 834
P-P
870
876
878
880
886
888
850
852
854
894
896
898
チップセット 890
インタフェース
インタフェース 839
コプロセッサ 838
バスブリッジ 818
I/Oデバイス 814
オーディオI/O 824
プロセッサ 815
816
820
通信デバイス 827
データストレージ 828
コードおよびデータ 830
キーボード/マウス 822

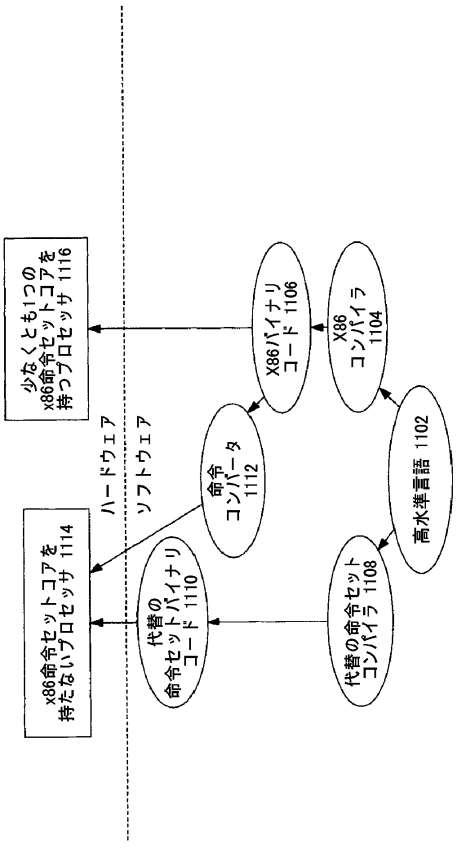
【図９】


900
メモリ 832
プロセッサ
制御ロジック 872
プロセッサ
制御ロジック 882
メモリ 834
I/Oデバイス 914
P-P
870
876
878
880
886
888
850
852
854
894
896
898
890
チップセット
インタフェース
レガシI/O 915

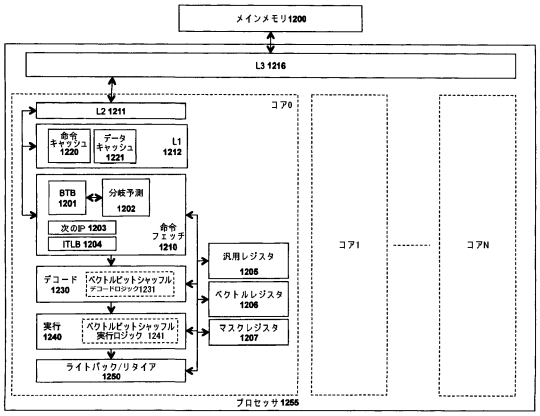
【図１０】


システムオンチップ 1000
アプリケーションプロセッサ 1010
コア 602A
キャッシュユニット 604A
コア 602N
キャッシュユニット 604N
共有キャッシュユニット 606
システムエージェントユニット 610
バスコントローラユニット 616
相互接続ユニット 1002
ディスプレイユニット 1040
DMAユニット 1032
SRAMユニット 1030
統合メモリコントローラユニット 614
コプロセッサ 1020

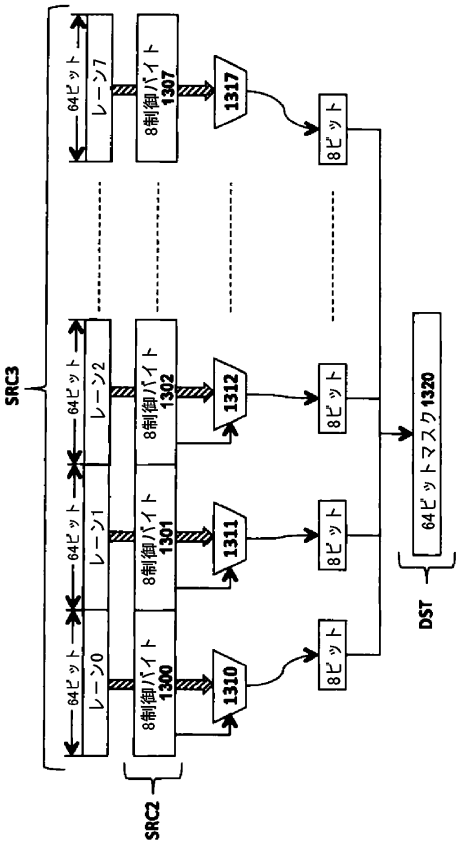
【図１１】


ハードウェア
ソフトウェア
少なくとも1つのx86命令セットコアを持つプロセッサ 1116
x86命令セットコアを持たないプロセッサ 1114
命令コンバータ 1112
代替の命令セットバイナリコード 1110
X86バイナリコード 1106
X86コンパイラ 1104
高水準言語 1102
代替の命令セットコンパイラ 1108

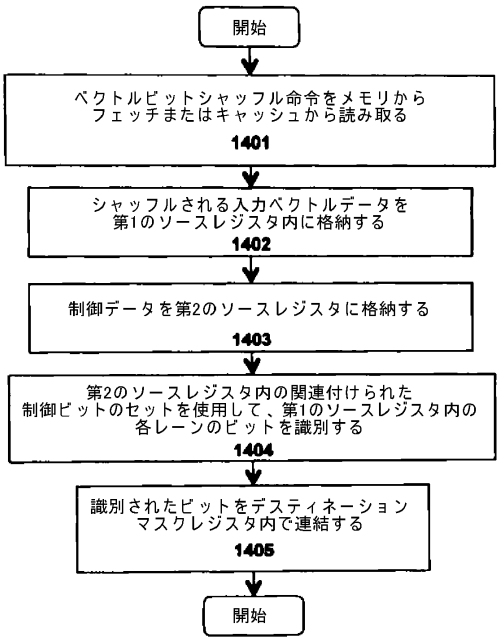
【図１２】


メインメモリ1200
L3 1216
コア0
L2 1211
命令キャッシュ 1220
データキャッシュ 1221
L1 1212
BTB 1201
分岐予測 1202
次のIP 1203
ITLB 1204
命令フェッチ 1210
デコード 1230
ベクトルビットシャッフルデコードロジック1231
実行 1240
ベクトルビットシャッフル実行ロジック 1241
ライトバック/リタイア 1250
汎用レジスタ 1205
ベクトルレジスタ 1206
マスクレジスタ 1207
コア1
コアN
プロセッサ 1255

【図１３】


SRC3
SRC2
64ビット
レーン7
レーン2
レーン1
レーン0
8制御バイト 1307
8制御バイト 1302
8制御バイト 1301
8制御バイト 1300
1317
1312
1311
1310
8ビット
64ビットマスク1320
DST

【図１４】


開始
ベクトルビットシャッフル命令をメモリから
フェッチまたはキャッシュから読み取る
1401
シャッフルされる入力ベクトルデータを
第1のソースレジスタ内に格納する
1402
制御データを第2のソースレジスタに格納する
1403
第2のソースレジスタ内の関連付けられた
制御ビットのセットを使用して、第1のソースレジスタ内の
各レーンのビットを識別する
1404
識別されたビットをデスティネーション
マスクレジスタ内で連結する
1405
開始

フロントページの続き

(72)発明者　コーバル、ジーザス
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　バレンタイン、ロバート
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　チャーニー、マーク　ジェイ.
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　ソレ、グイレム
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　エスパサ、ロジャー
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

審査官　清木　泰

(56)参考文献　特許第６５２６１７５（ＪＰ，Ｂ２）
特表２０１８－５００６６６（ＪＰ，Ａ）
特開２０１２－３３０３２（ＪＰ，Ａ）
特表２００７－５２６５３６（ＪＰ，Ａ）
米国特許出願公開第２０１４／００５９３２３（ＵＳ，Ａ１）
米国特許出願公開第２０１４／００１３０８８（ＵＳ，Ａ１）
米国特許出願公開第２０１３／０３１８３２８（ＵＳ，Ａ１）
米国特許出願公開第２０１３／０２１２３６０（ＵＳ，Ａ１）
米国特許出願公開第２００５／０１３９６４７（ＵＳ，Ａ１）
米国特許出願公開第２００２／００３５６７８（ＵＳ，Ａ１）
米国特許第６９６３３４１（ＵＳ，Ｂ１）
国際公開第２０１４／１５０９１３（ＷＯ，Ａ２）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／３０－　９／３５５
Ｇ０６Ｆ　９／３８
Ｇ０６Ｆ１７／１０－１７／１８