(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6092904号
(P6092904)
(45)発行日　平成29年3月8日(2017.3.8)　(24)登録日　平成29年2月17日(2017.2.17)

(51)Int.Cl.			F I		
G06F	9/305	(2006.01)	G06F	9/30	340A
G06F	7/487	(2006.01)	G06F	7/487	
G06F	17/10	(2006.01)	G06F	17/10	S

請求項の数 20　外国語出願　(全 37 頁)

(21)出願番号　特願2015-11008(P2015-11008)
(22)出願日　平成27年1月23日(2015.1.23)
(65)公開番号　特開2015-191661(P2015-191661A)
(43)公開日　平成27年11月2日(2015.11.2)
審査請求日　平成27年1月23日(2015.1.23)
(31)優先権主張番号　14/229,183
(32)優先日　平成26年3月28日(2014.3.28)
(33)優先権主張国　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　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

最終頁に続く

(54)【発明の名称】プロセッサおよび方法

(57)【特許請求の範囲】
【請求項１】
メモリサブシステムから単一の二重乗算命令をフェッチする命令フェッチユニットであり、前記二重乗算命令は３つのソースオペランド値を有する、命令フェッチユニットと、
前記二重乗算命令をデコードして、単一のマイクロオペレーション（μｏｐ）を生成するデコードユニットと、
前記μｏｐを１回目に実行して、前記３つのソースオペランド値のうちの第１のソースオペランド値および第２のソースオペランド値を乗算して中間結果を生成し、前記μｏｐを２回目に実行して、前記中間結果を前記３つのソースオペランド値のうちの第３のソースオペランド値を用いて乗算して、最終結果を生成する実行ユニットと、
を備えるプロセッサ。
【請求項２】
前記実行ユニットは、前記μｏｐの前記２回目の実行の前に前記μｏｐを遅延する遅延バッファを含み、請求項１に記載のプロセッサ。
【請求項３】
前記実行ユニットは、さらに、少なくとも１つの機能ユニットによる実行のために、前記二重乗算命令をスケジュールするリザベーションステーションを含み、前記μｏｐは、前記リザベーションステーションから第１の機能ユニットに送信され、機能ユニットによる前記実行の前に前記遅延バッファにも提供される、請求項２に記載のプロセッサ。
【請求項４】

前記機能ユニットは、融合乗算および加算機能ユニットを有する、請求項３に記載のプロセッサ。

【請求項５】

前記μｏｐは、さらに、前記第１の機能ユニットが前記μｏｐの１回目の実行を完了し、前記中間結果を生成したときに前記遅延バッファから第２の機能ユニットに送信され、前記第２の機能ユニットは、前記中間結果を前記３つのソースオペランド値のうちの前記第３のソースオペランド値により乗算して、前記最終結果を生成する、請求項３または４に記載のプロセッサ。

【請求項６】

前記二重乗算命令の第１のソースオペランド値、第２のソースオペランド値、および第３のソースオペランド値は、浮動小数点値である、請求項１から５のいずれか一項に記載のプロセッサ。

【請求項７】

前記浮動小数点値は、単精度または倍精度浮動小数点値を有する、請求項６に記載のプロセッサ。

【請求項８】

前記二重乗算命令は、第１のソースオペランド値、第２のソースオペランド値、および第３のソースオペランド値のそれぞれの符号を示す即値を有する、請求項１から７のいずれか一項に記載のプロセッサ。

【請求項９】

前記即値は、前記第１のソースオペランド値、前記第２のソースオペランド値、および前記第３のソースオペランド値の符号を示す各ビットの値を有する３ビット値を有する、請求項８に記載のプロセッサ。

【請求項１０】

前記リザベーションステーションは、第１の実効ポートを介して前記μｏｐの前記１回目の実行をスケジュールするための第１のリザベーションステーション部分と、第２の実効ポートを介して前記μｏｐの前記２回目の実行をスケジュールするための第２のリザベーションステーション部分と、を含む、請求項３に記載のプロセッサ。

【請求項１１】

プロセッサにより実行される方法であって、

前記プロセッサにより、メモリサブシステムから単一の二重乗算命令をフェッチする段階であり、前記二重乗算命令は３つのソースオペランド値を有する、段階と、

前記プロセッサにより、単一のマイクロオペレーション（μｏｐ）を生成するべく前記二重乗算命令をデコードする段階と、

前記プロセッサにより、前記３つのソースオペランド値のうちの第１のソースオペランド値および第２のソースオペランド値を乗算して中間結果を生成するべく前記μｏｐを１回目に実行し、前記中間結果を前記３つのソースオペランド値のうちの第３のソースオペランド値を用いて乗算して、最終結果を生成するべく前記μｏｐを２回目に実行する段階と、

を備える方法。

【請求項１２】

前記プロセッサにより、前記μｏｐの前記２回目の実行の前に遅延バッファで前記μｏｐを遅延する段階をさらに備える、請求項１１に記載の方法。

【請求項１３】

前記プロセッサにより、少なくとも１つの機能ユニットによる実行のために、前記二重乗算命令をスケジュールする段階をさらに備え、前記μｏｐは、第１の機能ユニットに送信され、機能ユニットによる前記実行の前に前記遅延バッファにも提供される、請求項１２に記載の方法。

【請求項１４】

前記機能ユニットは、融合乗算および加算機能ユニットを有する、請求項１３に記載の

方法。
【請求項１５】
前記μｏｐは、さらに、前記第１の機能ユニットが前記μｏｐの１回目の実行を完了し、前記中間結果を生成したときに前記遅延バッファから第２の機能ユニットに送信され、前記第２の機能ユニットは、前記中間結果を前記３つのソースオペランド値のうちの前記第３のソースオペランド値により乗算して、前記最終結果を生成する、請求項１３または１４に記載の方法。
【請求項１６】
前記二重乗算命令の第１のソースオペランド値、第２のソースオペランド値、および第３のソースオペランド値は、浮動小数点値である、請求項１１から１５のいずれか一項に記載の方法。
【請求項１７】
前記浮動小数点値は、単精度または倍精度浮動小数点値を有する、請求項１６に記載の方法。
【請求項１８】
前記二重乗算命令は、第１のソースオペランド値、第２のソースオペランド値、および第３のソースオペランド値のそれぞれの符号を示す即値を有する、請求項１１から１７のいずれか一項に記載の方法。
【請求項１９】
前記即値は、前記第１のソースオペランド値、前記第２のソースオペランド値、および前記第３のソースオペランド値の符号を示す各ビットの値を有する３ビット値を有する、請求項１８に記載の方法。
【請求項２０】
前記スケジュールする段階は、第１の実効ポートを介して前記μｏｐの前記１回目の実行をスケジュールするための第１のリザベーションステーション部分と、第２の実効ポートを介して前記μｏｐの前記２回目の実行をスケジュールするための第２のリザベーションステーション部分と、を含むリザベーションステーションにより実行され、請求項１３に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、概して、コンピュータプロセッサの分野に関する。より具体的には、発明は、複数の乗算演算を実行するための方法及び装置に関する。
【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、本来のデータタイプ、命令、レジスタアーキテクチャ、アドレスモード、メモリアーキテクチャ、割り込み及び例外処理、及び外部入出力（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。ここでは、用語「命令」は、概して、マイクロ命令に対立するものとしてのマクロ命令（実行するためにプロセッサに提供される命令）またはマイクロオペレーション（プロセッサのデコーダがマクロ命令をデコードした結果）を参照することに留意すべきである。
【０００３】
ＩＳＡは、命令セットを実装するために用いられるプロセッサ設計技術のセットであるマイクロアーキテクチャから区別される。異なるマイクロアーキテクチャを有する複数のプロセッサは、共通の命令セットを共有する。例えば、Ｉｎｔｅｌ（登録商標）Ｐｅｎｔｉｕｍ（登録商標）４プロセッサ、Ｉｎｔｅｌ（登録商標）Ｃｏｒｅ（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アドバンスドマイクロデバイセズからのプロセッサは、ｘ８６命令セット（より新しいバージョンが追加された幾つかのエクステンションを有する）のほぼ同じバージョンを実装するが、異なる内部設計を有する。例えば、Ｉ

ＳＡの同じレジスタアーキテクチャは、専用の物理レジスタ、レジスタリネームメカニズム（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）、及びリタイアメントレジスタファイルの使用）を用い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用いて異なるマイクロアーキテクチャに異なる態様で実装されてよい。特に指定されない限り、レジスタアーキテクチャ、レジスタファイル、およびレジスタなるフレーズは、ここでは、ソフトウェア／プログラマにビジブルであるそれ、および複数の命令が複数のレジスタを特定する方法を参照するために用いられる。区別が必要の場合、「論理」、「アーキテクチャ上」、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用いられるとともに、異なる形容詞が、与えられたマイクロアーキテクチャにおいてレジスタを指定するために用いられる（例えば、物理レジスタ、リオーダバッファ、リタイヤメントレジスタ、レジスタプール）。

【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。与えられた命令フォーマットは、とりわけ、実行される演算およびその演算が実行されるオペランドを特定するために、様々なフィールド（ビットの数、ビットの位置）を定義する。幾つかの命令フォーマットは、さらに、複数の命令テンプレート（または複数のサブフォーマット）の定義を介して分解される。例えば、与えられた命令フォーマットの複数の命令テンプレートは、命令フォーマットの複数のフィールド（より少ない含まれたフィールドがあるので、含まれるフィールドは、一般的に、同じ順序であり、しかし少なくとも幾つかは異なるビット位置を有する。）の異なるサブセットを有するために定義されてよく、および／または異なって解釈される与えられたフィールドを有するために定義されてよい。与えられた命令は、与えられた命令フォーマットを用いて（および、定義されている場合には、その命令フォーマットの複数の命令テンプレートの与えられた１つにおいて）表され、演算および複数のオペランドを特定する。命令ストリームは、複数の命令の固有のシーケンスである。ただし、シーケンス内の各命令は、命令フォーマットにおける命令の発生である（および、定義されている場合には、その命令フォーマットの複数の命令テンプレートの与えられた１つである）。

【０００５】

科学、金融、自動ベクトル化の汎用、ＲＭＳ（認識、採鉱、および合成）、およびビジュアルおよびマルチメディアアプリケーション（例えば、２Ｄ／３Ｄグラフィック、画像処理、ビデオ圧縮／解凍、音声認識アルゴリズム、およびオーディオ操作）は、頻繁に、多数のデータアイテム（「データ並列処理」として参照される）上で実行される同じ演算を必要とする。単一命令複数データ（ＳＩＭＤ）は、プロセッサに複数のデータアイテム上の演算を実行させる命令のタイプを参照する。ＳＩＭＤ技術は、特に、レジスタ内の複数のビットを、それぞれが別個の値を表す固定サイズのデータ要素の数に論理的に分割できるプロセッサに好適である。例えば、６４ビットレジスタ内の複数のビットは、それぞれが別個の１６ビット値を表す４つの別個の１６ビットデータ要素として操作されるソースオペランドとして特定されてよい。このタイプのデータは、パックドデータタイプまたはベクトルデータタイプとして参照され、このデータタイプの複数のオペランドは、パックドデータオペランドまたはベクトルオペランドとして参照される。言い換えると、パックドデータアイテムまたはベクトルは、パックドデータ要素のシーケンスを参照し、パックドデータオペランドまたはベクトルオペランドは、ＳＩＭＤ命令（パックドデータ命令またはベクトル命令としても知られる）のソースまたはデスティネーションオペランドである。

【０００６】

例として、ＳＩＭＤ命令の一タイプは、２つのソースベクトルオペランド上で垂直式に実行されて、同じ数のデータ要素を有する同じサイズおよび同じデータエレメントの順序にあるデスティネーションベクトルオペランド（結果ベクトルオペランドとしても参照される）を生成するシングルベクトル演算を特定する。複数のソースベクトルオペランドに

おける複数のデータ要素は、複数のソースデータエレメントとして参照されるとともに、デスティネーションベクトルオペランド内の複数のデータ要素は、デスティネーションまたは結果データ要素と参照される。これらのソースベクトルオペランドは、同じサイズであり、同じ幅の複数のデータ要素を含み、従って、それらは同じ数のデータ要素を含む。２つのソースベクトルオペランド内の複数の同じビット位置内の複数のソースデータエレメントは、複数の組のデータ要素（対応するデータ要素としても参照される）を形成する。そのＳＩＭＤ命令により指定される演算は、これらの組のソースデータエレメントのそれぞれで別個に実行されて、マッチング数の結果データ要素を生成し、従って、各組のソースデータエレメントは対応する結果データ要素を有する。演算は垂直であるので、また結果ベクトルオペランドは同じ数のデータ要素を有する同じサイズであり、結果データ要素は複数のソースベクトルオペランドとして同じデータエレメントの順序で格納されるので、複数の結果データ要素は、複数のソースベクトルオペランド内の複数のソースデータエレメントのそれらの対応する組として、結果ベクトルオペランドの複数の同じビット位置内にある。この典型的なタイプのＳＩＭＤ命令に加えて、様々な他のタイプのＳＩＭＤ命令がある（例えば、１つのみまたは２以上のソースベクトルオペランドを有する、垂直式に演算する、異なるサイズの結果ベクトルオペランドを生成する、異なるサイズのデータ要素を有する、および／または異なるデータエレメントの順序を有する）。用語デスティネーションベクトルオペランド（またはデスティネーションオペランド）は、命令により指定される演算を実行することの直接の結果として、位置（そのレジスタ又はその命令により特定されるメモリアドレス）でそのデスティネーションオペランドのストレージを含めて定義され、それにより、それは別の命令により（別の命令によるその同じ位置の仕様により）ソースオペランドとしてアクセスされてよいことを理解されるべきである。

【０００７】

ｘ８６、ＭＭＸ、ストリーミングＳＩＭＤエクステンション（ＳＳＥ）、ＳＳＥ２、ＳＳＥ３、ＳＳＥ４．１、およびＳＳＥ４．２命令を含む命令セットを有するＩｎｔｅｌ（登録商標）Ｃｏｒｅ（商標）プロセッサにより使用されるようなＳＩＭＤ技術は、アプリケーションの性能の大幅な改善を可能にした（ＣｏｒｅおよびＭＭＸは、カリフォルニア州サンタクララのインテルの登録商標または商標である）。アドバンスドベクトルエクステンション（ＡＶＸ）と参照され、ＶＥＸコーディングスキームを用いる複数のＳＩＭＤエクステンションの追加的なセットも、設計され、公開されている。

【０００８】

本出願に特に関連する１つの命令は、乗算命令である。高性能コンピューティングプラットフォームにおける幾つかのアルゴリズムは、幾つかの演算値を乗算する。一般に、各乗算演算は、１つの命令の実行を必要とする。

【図面の簡単な説明】

【０００９】

本発明のより良い理解は、次の図面と併せて次の詳細な説明から得られることができる。

【図１Ａ】発明の実施形態に係る典型的なインオーダフェッチ、デコード、リタイヤパイプライン、および典型的なレジスタリネーム、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１Ｂ】発明の実施形態に係るインオーダフェッチ、デコード、リタイヤコアの典型的な実施形態、およびプロセッサ内に含まれる典型的なレジスタリネーム、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２】発明の実施形態に係るシングルコアプロセッサおよび統合メモリコントローラおよびグラフィックを有するマルチコアプロセッサのブロック図である。

【図３】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。

【図４】本発明の実施形態による第２システムのブロック図を示す。

【図５】本発明の実施形態による第３システムのブロック図を示す。

【図６】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。

【図７】発明の実施形態に係る、ソース命令セットにおけるバイナリ命令をターゲット命令セットにおけるバイナリ命令に変換する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。
【図８】発明の実施形態が使用されてよいプロセッサアーキテクチャの一実施形態を示す。
【図９Ａ】複数の乗算演算を実行するためのアーキテクチャの一実施形態を示す。
【図９Ｂ】複数の乗算演算を実行するためのアーキテクチャの別の実施形態を示す。
【図１０】複数の乗算演算を実行するための方法の一実施形態を示す。
【図１１Ａ】発明の実施形態に係る総称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１１Ｂ】発明の実施形態に係る総称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１２Ａ】発明の実施形態に係る典型的な特定ベクトル向け命令フォーマットのブロック図を示す。
【図１２Ｂ】発明の実施形態に係る典型的な特定ベクトル向け命令フォーマットのブロック図を示す。
【図１２Ｃ】発明の実施形態に係る典型的な特定ベクトル向け命令フォーマットのブロック図を示す。
【図１２Ｄ】発明の実施形態に係る典型的な特定ベクトル向け命令フォーマットのブロック図を示す。
【図１３】発明の一実施形態に係るレジスタアーキテクチャのブロック図である。
【発明を実施するための形態】
【００１０】
次の説明では、説明の目的のために、多くの特定の詳細が、以下に記載される発明の複数の実施形態の完全な理解を提供するために明らかにされる。しかし、発明の複数の実施形態はこれらの特定の詳細の一部がなくても実施されてよいことは、当業者には明らかであろう。他の複数の例において、既知の構造およびデバイスは、発明の実施形態の基礎となる原理を分かりにくくしないようにブロック図形式で示される。
【００１１】
典型的なプロセッサアーキテクチャおよびデータタイプ
図１Ａは、発明の実施形態に係る典型的なインオーダフェッチ、デコード、リタイヤパイプライン、および典型的なレジスタリネーム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１Ｂは、発明の実施形態に係るインオーダフェッチ、デコード、リタイヤコアの典型的な実施形態、およびプロセッサ内に含まれる典型的なレジスタリネーム、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１Ａおよび図１Ｂにおける実線のボックスは、パイプラインおよびコアのインオーダ部分を示し、一方、破線のボックスの任意の追加は、レジスタリネーム、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。
【００１２】
図１Ａにおいて、プロセッサパイプライン１００は、フェッチステージ１０２、レングスデコードステージ１０４、デコードステージ１０６、割り当てステージ１０８、リネームステージ１１０、スケジューリング（ディスパッチ又は発行としても知られる）ステージ１１２、レジスタ読み出し／メモリ読み出しステージ１１４、実行ステージ１１６、ライトバック／メモリ書き込みステージ１１８、例外ハンドリングステージ１２２、およびコミットステージ１２４を含む。
【００１３】
図１Ｂは、実行エンジンユニット１５０に連結されるフロントエンドユニット１３０を含むプロセッサコア１９０を示し、両方がメモリユニット１７０に連結される。コア１９０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、又はハイブリッドまたは代替

的コアタイプであってよい。さらに別のオプションとして、コア１９０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、汎用コンピュータグラフィックプロセッシングユニット（ＧＰＧＰＵ）コア、グラフィックコアなどのような特定の目的のコアであってよい。

【００１４】

フロントエンドユニット１３０は、命令キャッシュユニット１３４に連結される分岐予測ユニット１３２を含む。命令キャッシュユニット１３４は、命令変換索引バッファ（ＴＬＢ）１３６に連結される。ＴＬＢ１３６は、命令フェッチユニット１３８に連結される。命令フェッチユニット１３８は、デコードユニット１４０に連結される。デコードユニット１４０（またはデコーダ）は、複数の命令をデコードし、出力として、１または複数のマイクロ演算、複数のマイクロコードエントリポイント、複数のマイクロ命令、その他の複数の命令、または元の複数の命令からデコードされる、そうでなければそれらを反映する、またはそれらから導出されるその他の複数の制御信号を生成してよい。デコードユニット１４０は、様々な異なるメカニズムを用いて実装されてよい。適当なメカニズムの例は、これに限定されるものではないが、複数のルックアップテーブル、複数のハードウェア実装、複数のプログラマブルロジックアレイ（ＰＬＡ）、複数のマイクロコードリードオンリメモリ（ＲＯＭ）などを含む。一実施形態では、コア１９０は、特定の複数のマイクロ命令のマイクロコードを（例えば、デコードユニット１４０内に、そうでなければフロントエンドユニット１３０内に）格納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１４０は、実行エンジンユニット１５０内でリネーム／割り当てユニット１５２に連結される。

【００１５】

実行エンジンユニット１５０は、リタイアメントユニット１５４および１または複数のスケジューラユニット１５６のセットに連結されたリネーム／割り当てユニット１５２を含む。スケジューラユニット１５６は、複数の予約ステーション、中央の命令ウィンドウなど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１５６は、物理レジスタファイルユニット１５８に連結される。複数の物理レジスタファイルユニット１５８のそれぞれは、１または複数の物理レジスタファイル、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、ステータス（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）などのような１または複数の異なるデータタイプを格納する異なるものを表す。一実施形態では、物理レジスタファイルユニット１５８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、複数のアーキテクチャベクトルレジスタ、複数のベクトルマスクレジスタ、及び複数の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１５８は、リタイアメントユニット１５４により重ねられて、（例えば、リオーダバッファ及びリタイアメントレジスタファイルを用いて、将来のファイル、ヒストリバッファ、及びリタイアメントレジスタファイルを用いて、レジスタマップおよび複数のレジスタのプールを用いるなど）レジスタリネームおよびアウトオブオーダ実行が実装されてよい様々な態様を示す。リタイアメントユニット１５４および物理レジスタファイルユニット１５８は、実行クラスタ１６０に連結される。実行クラスタ１６０は、１または複数の実行ユニット１６２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１６４のセットを含む。実行ユニット１６２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）について実行してよい。幾つかの実施形態は、複数の特定の機能または複数の機能の複数のセットに専用の多くの実行ユニットを含んでよいとともに、他の実施形態は、すべての機能をすべて実行する実行ユニットの１つのみ又は複数の実行ユニットを含んでよい。特定の実施形態は、特定のタイプのデータ／複数の演算に対する別個のパイプラインを生成するので（例えば、それら自体のスケジューラユニットをそれぞれ有するスカラ整数パイプライン、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、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。別個のメモリアクセスパイプラインの場合、特定の実施形態は、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１６４を有するように実装される。）、スケジューラユニット１５６、物理レジスタファイルユニット１５８、及び実行クラスタ１６０は、場合により、複数あるように示される。別個のパイプラインが用いられる場合、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／実行され、残りがインオーダ発行／実行されてよいことは、理解されるべきでもある。

【００１６】

複数のメモリアクセスユニット１６４のセットは、メモリユニット１７０に連結される。メモリユニット１７０は、データＴＬＢユニット１７２を含む。データＴＬＢユニット１７２は、データキャッシュユニット１７４に連結される。データキャッシュユニット１７４は、レベル２（Ｌ２）キャッシュユニット１７６に連結される。一典型的な実施形態では、複数のメモリアクセスユニット１６４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、それぞれがメモリユニット１７０内のデータＴＬＢユニット１７２に連結される。命令キャッシュユニット１３４は、さらに、メモリユニット１７０内のレベル２（Ｌ２）キャッシュユニット１７６に連結される。Ｌ２キャッシュユニット１７６は、１または複数の他のレベルのキャッシュおよび最終的にはメインメモリに連結される。

【００１７】

例として、典型的なレジスタリネームアウトオブオーダ発行／実行コアアーキテクチャは、次のようにパイプライン１００を実装してよい。１）命令フェッチ１３８が、フェッチおよびレングスデコードステージ１０２および１０４を実行する。２）デコードユニット１４０が、デコードステージ１０６を実行する。３）リネーム／割り当てユニット１５２が、割り当てステージ１０８およびリネームステージ１１０を実行する。４）スケジューラユニット１５６が、スケジュールステージ１１２を実行する。５）物理レジスタファイルユニット１５８およびメモリユニット１７０が、レジスタ読み出し／メモリ読み出しステージ１１４を実行する。実行クラスタ１６０が、実行ステージ１１６を実行する。６）メモリユニット１７０および物理レジスタファイルユニット１５８が、ライトバック／メモリ書き込みステージ１１８を実行する。７）様々なユニットが、例外ハンドリングステージ１２２に関与されてよい。８）リタイアメントユニット１５４および物理レジスタファイルユニット１５８が、コミットステージ１２４を実行する。

【００１８】

コア１９０は、ここに記載される命令を含め、１または複数の命令セット（例えば、ｘ８６命令セット（複数のより新しいバージョンに追加された幾つかの拡張を有する））、カリフォルニア州サニーベールのＭＩＰＳテクノロジーズのＭＩＰＳ命令セット、カリフォルニア州サニーベールのＡＲＭホールディングスのＡＲＭ命令セット（ＮＥＯＮのような任意追加の複数の拡張を有する））をサポートしてよい。一実施形態では、コア１９０は、パックドデータ命令セットの拡張（例えば、ＡＶＸ１、ＡＶＸ２、および／または後述する総称ベクトル向け命令フォーマット（Ｕ＝０および／またはＵ＝１）の幾つかの形式）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り用いられる複数の演算をパックドデータを用いて実行されるようにする。

【００１９】

コアは、マルチスレッド（演算又はスレッドの２またはそれより多いパラレルセットを実行）をサポートしてよいし、時間スライスされたマルチスレッド、同時マルチスレッド（ただし、単一物理コアは、物理コアが同時にマルチスレッドする複数のスレッドのそれぞれに対して論理コアを提供する）、またはそれらの組み合わせ（例えば、インテルハイパースレッド技術におけるような時間スライスされたフェッチおよびデコードおよびそのあとの同時マルチスレッド）を含む様々な態様においてそうしてよいことが理解されるべきである。

【００２０】
レジスタリネームがアウトオブオーダ実行のコンテキストにおいて記載される限り、レジスタリネームがインオーダアーキテクチャにおいて用いられてよいことが理解されるべきである。プロセッサの示された実施形態が、別個の命令およびデータキャッシュユニット１３４／１７４および共有Ｌ２キャッシュユニット１７６も含むのに対して、代替的な実施形態は、例えばレベル１（Ｌ１）内部キャッシュまたは複数レベルの内部キャッシュのような命令およびデータの両方に対する単一の内部キャッシュを有してよい。幾つかの実施形態では、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との組み合わせを含んでよい。代替的に、キャッシュのすべては、コアおよび／またはプロセッサの外部にあってよい。

【００２１】
図２は、発明の実施形態に係る、１より多いコアを有してよく、統合メモリコントローラを有してよく、また統合グラフィクスを有してよいプロセッサ２００のブロック図である。図２における実線のボックスは、シングルコア２０２Ａ、システムエージェント２１０、および１または複数のバスコントローラユニット２１６のセットを有するプロセッサ２００を示すとともに、任意の追加の破線のボックスは、複数のマルチコア２０２Ａ－Ｎ、システムエージェントユニット２１０内の１または複数の統合メモリコントローラユニット２１４のセット、および専用ロジック２０８を有する代替例のプロセッサ２００を示す。

【００２２】
従って、プロセッサ２００の異なる実装は、１）統合グラフィクスおよび／または科学（スループット）ロジックである専用ロジック２０８を有するＣＰＵ（１または複数のコアを含んでよい）、および１または複数の汎用コアであるコア２０２Ａ－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、２つの組み合わせ）、２）グラフィックおよび／または科学（スループット）を主に意図する多数の専用コアであるコア２０２Ａ－Ｎを有するコプロセッサ、および３）多数の汎用インオーダコアであるコア２０２Ａ－Ｎを有するコプロセッサを含んでよい。従って、プロセッサ２００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、グラフィクスプロセッサ、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またはそれより多いコアを含む）、組み込みプロセッサなど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２００は、１または複数の基板の一部であってよいし、および／または、例えばＢｉＣＭＯＳ、ＣＭＯＳ、またはＮＭＯＳのような多くの処理技術のうちのいずれを用いてそれらの上に実装されてよい。

【００２３】
メモリ階層は、複数の統合メモリコントローラユニット２１４のセットに連結される複数のコア、セットまたは１または複数の共有キャッシュユニット２０６、および外部メモリ（不図示）内に１または複数のレベルのキャッシュを含む。共有キャッシュユニット２０６のセットは、レベル２（Ｌ２）、レベル３（Ｌ３）、レベル４（Ｌ４）、または他のレベルのキャッシュ、最後のレベルのキャッシュ（ＬＬＣ）、および／またはそれらの組み合わせのような１または複数の中間レベルキャッシュを含んでよい。一実施形態では、リングベースのインターコネクトユニット２１２は、統合グラフィクスロジック２０８、共有キャッシュユニット２０６のセット、およびシステムエージェントユニット２１０／統合メモリコントローラユニット２１４を相互接続するのに対して、代替的な実施形態は、そのような複数のユニットを相互接続する任意の数の周知の技術を用いてよい。一実施形態では、一貫性が、１または複数のキャッシュユニット２０６および複数のコア２０２Ａ－Ｎの間で維持される。

【００２４】
幾つかの実施形態では、１または複数のコア２０２Ａ－Ｎはマルチスレッドすることができる。システムエージェント２１０は、コア２０２Ａ－Ｎを調整および操作するそれら

の複数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２１０は、例えば、電力制御ユニット（ＰＣＵ）および表示ユニットを含んでよい。ＰＣＵは、コア２０２Ａ－Ｎおよび統合グラフィクスロジック２０８の電力状態をレギュレートするのに必要なロジックおよび複数のコンポーネントであってもまたは含んでもよい。表示ユニットは、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。

【００２５】

複数のコア２０２Ａ－Ｎは、アーキテクチャ命令セットの観点において同種または異種であってよい。すなわち、コア２０２Ａ－Ｎのうちの２またはそれより多いコアは同じ命令セットを実行できてよく、その他はその命令セットまたは異なる命令セットのサブセットのみを実行できてよい。一実施形態では、複数のコア２０２Ａ－Ｎは、異種であり、後述する複数の「小さい」コアおよび複数の「大きい」コアの両方を含む。

【００２６】

図３から図６は、典型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、携帯用情報端末、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、デジタルシグナルプロセッサ（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技術分野において既知の他のシステム設計及び構成も適当である。一般的に、ここに開示されるようなプロセッサおよび／または他の実行ロジックを組み込むことができる様々なシステムまたは電子デバイスが一般に適当である。

【００２７】

ここで図３を参照すると、本発明の一実施形態によるシステム３００のブロック図が示される。システム３００は、コントローラハブ３２０に連結される１または複数のプロセッサ３１０、３１５を含んでよい。一実施形態では、コントローラハブ３２０は、グラフィックスメモリコントローラハブ（ＧＭＣＨ）３９０および入出力ハブ（ＩＯＨ）３５０（別個の複数のチップ上にあってよい）を含む。ＧＭＣＨ３９０は、メモリ３４０およびコプロセッサ３４５に連結されるメモリおよびグラフィクスコントローラを含む。ＩＯＨ３５０は、入出力（Ｉ／Ｏ）デバイス３６０をＧＭＣＨ３９０に接続する。代替的に、メモリおよびグラフィクスコントローラのうちの１つまたは両方は、プロセッサに（ここに記載されるように）集積され、メモリ３４０およびコプロセッサ３４５は、ＩＯＨ３５０を有する単一チップ内でプロセッサ３１０およびコントローラハブ３２０に直接連結される。

【００２８】

複数の追加のプロセッサ３１５の任意の特性は、破線を用いて図３内に示される。各プロセッサ３１０、３１５は、ここに記載される処理コアの１または複数を含んでよく、またプロセッサ２００の幾つかのバージョンであってよい。

【００２９】

メモリ３４０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つの実施形態に対して、コントローラハブ３２０は、フロントサイドバス（ＦＳＢ）のようなマルチドロップバス、ＱｕｉｃｋＰａｔｈインターコネクト（ＱＰＩ）のようなポイントツーポイントインターフェース、または同様の接続３９５を介してプロセッサ３１０、３１５と通信する。

【００３０】

一実施形態では、コプロセッサ３４５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、グラフィクスプロセッサ、ＧＰＧＰＵ、組み込みプロセッサなどのような専用プロセッサである。一実施形態では、コントローラハブ３２０は、統合グラフィクスアクセラレータを含んでよい。

【００３１】

アーキテクチャ、マイクロアーキテクチャ、熱、電力消費特性などを含むメリットメトリクスの範囲の観点において、物理リソース３１０、３１５の間に様々な差があるはずである。

【００３２】

一実施形態では、プロセッサ３１０は、一般タイプのデータ処理演算を制御する複数の命令を実行する。複数のコプロセッサ命令は、複数の命令内に組み込まれてよい。プロセッサ３１０は、これらのコプロセッサ命令を、付属のコプロセッサ３４５により実行されるべきタイプとして認識する。従って、プロセッサ３１０は、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、コプロセッサバスまたは他のインターコネクト上でコプロセッサ３４５に発する。コプロセッサ３４５は、受信した複数のコプロセッサ命令を受け入れて実行する。

【００３３】

ここで図４を参照すると、本発明の実施形態による、第１のより具体的な典型的なシステム４００のブロック図を示す。図４に示されるように、マイクロプロセッサシステム４００は、ポイントツーポイントインターコネクトシステムであり、ポイントツーポイントインターコネクト４５０を介して連結された第１のプロセッサ４７０および第２のプロセッサ４８０を含む。プロセッサ４７０および４８０のそれぞれは、プロセッサ２００の幾つかのバージョンであってよい。発明の一実施形態では、プロセッサ４７０および４８０はそれぞれプロセッサ３１０および３１５であり、コプロセッサ４３８はコプロセッサ３４５である。別の実施形態では、プロセッサ４７０および４８０は、それぞれ、プロセッサ３１０およびコプロセッサ３４５である。

【００３４】

プロセッサ４７０および４８０は、それぞれ統合メモリコントローラ（ＩＭＣ）ユニット４７２および４８２を含めて示されている。プロセッサ４７０は、その複数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）インターフェース４７６および４７８も含む。同様に、第２のプロセッサ４８０は、Ｐ－Ｐインターフェース４８６および４８８を含む。プロセッサ４７０、４８０は、ポイントツーポイント（Ｐ－Ｐ）インターフェース４５０を介して、Ｐ－Ｐインターフェース回路４７８、４８８を用いて情報を交換してよい。図４に示されるように、ＩＭＣ４７２および４８２は、複数のプロセッサをそれぞれメモリ、すなわちそれぞれのプロセッサにローカルに付属するメインメモリの一部であってよいメモリ４３２およびメモリ４３４に接続する。

【００３５】

プロセッサ４７０、４８０は、それぞれ、ポイントツーポイントインターフェース回路４７６、４９４、４８６、４９８を用いて、個々のＰ－Ｐインターフェース４５２、４５４を介してチップセット４９０と情報を交換してよい。チップセット４９０は、必要に応じて、高性能インターフェース４３９を介してコプロセッサ４３８と情報を交換してよい。一実施形態では、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、グラフィクスプロセッサ、ＧＰＧＰＵ、組み込みプロセッサなどのようなコプロセッサ４３８は、専用プロセッサである。

【００３６】

共有キャッシュ（不図示）は、どちらかのプロセッサまたは両方のプロセッサの外部に含まれ、さらにＰ－Ｐインターコネクトを介して複数のプロセッサに接続され、それにより、プロセッサが低電力モードに配置されると、どちら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てよい。

【００３７】

チップセット４９０は、インターフェース４９６を介して、第１のバス４１６に連結されてよい。一実施形態では、第１のバス４１６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、またはＰＣＩエクスプレスバスまたは別の第３世代Ｉ／Ｏインターコネクトバスのようなバス、であってよいが、本発明の範囲はこれに限定されるものではない。

【００３８】
図４に示すように、様々なＩ／Ｏデバイス４１４は、第１のバス４１６を第２のバス４２０に接続するバスブリッジ４１８とともに、第１のバス４１６に連結されてよい。一実施形態では、複数のコプロセッサ、複数の高スループットＭＩＣプロセッサ、ＧＰＧＰＵの複数のアクセラレータ（例えば、複数のグラフィクスアクセラレータまたは複数のデジタル信号処理（ＤＳＰ）ユニット）、複数のフィールドプログラマブルゲートアレイ、またはいずれの他のプロセッサのような１または複数の追加のプロセッサ４１５は、第１のバス４１６に連結される。一実施形態では、第２のバス４２０は、ローピンカウント（ＬＰＣ）バスであってよい。一実施形態では、様々なデバイスは、例えば、キーボードおよび／またはマウス４２２、複数の通信デバイス４２７、および命令／コードおよびデータ４３０を含んでよい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のようなストレージユニット４２８を含めて、第２のバス４２０に連結されてよい。さらに、オーディオＩ／Ｏ　４２４は、第２のバス４２０に連結されてよい。なお、他のアーキテクチャも可能である。例えば、図４のポイントツーポイントアーキテクチャに代えて、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してよい。

【００３９】
ここで図５を参照すると、本発明の実施形態による第２のより具体的な典型的なシステム５００のブロック図が示される。図４および図５における同じ要素は同じ参照番号を与え、図４の特定の態様は、図５の他の態様を分かりにくくしないように図５から省略されている。

【００４０】
図５は、プロセッサ４７０、４８０が、統合メモリおよびそれぞれＩ／Ｏ制御ロジック（「ＣＬ」）４７２および４８２を含んでよいことを示す。従って、ＣＬ４７２、４８２は、複数の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５は、メモリ４３２、４３４がＣＬ４７２、４８２に連結されるだけでなく、Ｉ／Ｏデバイス５１４も制御ロジック４７２、４８２に連結されることも示す。複数のレガシＩ／Ｏデバイス５１５は、チップセット４９０に連結される。

【００４１】
ここで図６を参照すると、本発明の実施形態によるＳｏＣ６００のブロック図が示される。図２内の同様の要素は、同じ参照番号を与える。また、破線のボックスは、より高度なＳｏＣの任意の特徴である。図６において、インターコネクトユニット６０２は、１または複数のコア５０２Ａ－Ｎおよび共有キャッシュユニット５０６のセットを含むアプリケーションプロセッサ６１０、システムエージェントユニット５１０、バスコントローラユニット５１６、統合メモリコントローラユニット５１４、統合グラフィクスロジック、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１または複数のコプロセッサ６２０のセット、スタティックランダムアクセスメモリ（ＳＲＡＭ）ユニット６３０、ダイレクトメモリアクセス（ＤＭＡ）ユニット６３２、および１または複数の外部ディスプレイに連結するための表示ユニット６４０、に連結される。一実施形態では、コプロセッサ６２０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などのような専用プロセッサを含む。

【００４２】
ここに開示される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、またはそのような複数の実装アプローチの組み合わせにおいて実装されてよい。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メモリおよび／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複数のプログラマブルシステム上で実行する複数の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。

【００４３】
図４に示されるコード４３０のようなプログラムコードは、ここに記載の複数の機能を

実行し、出力情報を生成する複数の命令を入力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様式で適用されてよい。このアプリケーションの目的のために、処理システムは、例えば、デジタルシグナルプロセッサ（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のようなプロセッサを有するいずれのシステムを含む。

【００４４】

プログラムコードは、処理システムと通信するために、高級手続型またはオブジェクト指向型プログラミング言語において実装されてよい。プログラムコードは、必要に応じて、アセンブリまたは機械言語において実装されてもよい。実際、ここに記載の複数のメカニズムは、いずれの特定のプログラミング言語の範囲に限定されるものではない。いずれの場合において、言語は、コンパイル型またはインタプリタ型言語であってよい。

【００４５】

少なくとも１つの実施形態の１または複数の態様は、機械により読み込まれると、機械に、ここに記載の技術を実行するロジックを組み立てさせるプロセッサ内の様々なロジックを表す、機械可読媒体上に格納された典型的な複数の命令により実装されてよい。「ＩＰコア」として知られるそのような表現は、実際にロジックまたはプロセッサを製造する複数の製造機械にロードするために、有形の機械可読媒体上に格納されて、様々な顧客または製造施設に供給されてよい。

【００４６】

そのような機械可読記憶媒体は、これらに限定されないが、ハードディスク、フロッピー（登録商標）ディスクを含む他のタイプのディスク、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）、及び磁気光ディスクのようなストレージメディア、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）のようなランダムアクセスメモリ（ＲＡＭ）、スタティックランダムアクセスメモリ（ＳＲＡＭ）、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）、磁気または光カードのような半導体デバイス、または電子命令を格納するのに好適ないずれの他のタイプのメディアを含む、機械またはデバイスにより製造または形成される複数の物品の非一時的で有形の装置を含んでよい。

【００４７】

従って、発明の実施形態は、複数の命令を含む、またはここに記載の構造、回路、装置、プロセッサ、および／またはシステム特徴を規定するハードウェア記述言語（ＨＤＬ）のような設計データを含む非一時的な有形の機械可読媒体も含む。そのような実施形態は、プログラム製品と参照されてもよい。

【００４８】

幾つかの場合では、命令コンバータは、ソース命令セットからの命令をターゲット命令セットに変換するために用いられてよい。例えば、命令コンバータは、命令を、コアにより処理される１または複数の他の命令に翻訳（例えば、静的バイナリトランスレーション、動的コンパイルを含む動的バイナリトランスレーションを用いて）、モーフィング、エミュレート、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、またはそれらの組み合わせにおいて実装されてよい。命令コンバータは、プロセッサ上に、プロセッサ外に、または一部がプロセッサ上に、一部がプロセッサ外にあってよい。

【００４９】

図７は、発明の実施形態に係る、ソース命令セットにおけるバイナリ命令をターゲット命令セットにおけるバイナリ命令に変換する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。示された実施形態では、命令コンバータは、ソフトウェア命令コンバータであるが、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、またはそれらの様々な組み合わせにおいて実装されてよい。図７は、高級言語７

０２におけるプログラムが、ｘ８６コンパイラ７０４を用いてコンパイルされて、少なくとも１つのｘ８６命令セットコア７１６を用いて、プロセッサにより、本来的に実行されてよいｘ８６バイナリコード７０６を生成してよいことを示す。少なくとも１つのｘ８６命令セットコア７１６を有するプロセッサは、互換実行する、そうでなければ、少なくとも１つのｘ８６命令セットコアを用いるＩｎｔｅｌプロセッサと実質的に同じ結果を達成するよう、（１）インテルｘ８６命令セットコアの命令セットの相当の部分、または（２）少なくとも１つのｘ８６命令セットコアを用いてＩｎｔｅｌプロセッサ上で実行することを目標とされた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のバージョンを処理することにより、少なくとも１つのｘ８６命令セットコアを有するＩｎｔｅｌプロセッサと同じ機能を実質的に達成できるいずれのプロセッサを表す。ｘ８６コンパイラ７０４は、追加的なリンケージ処理を用いてまたは用いないで、少なくとも１つのｘ８６命令セットコア７１６を有するプロセッサ上で実行されることができるｘ８６バイナリコード７０６（例えば、オブジェクトコード）を生成するよう動作可能なコンパイラを表す。同様に、図７は、高級言語７０２におけるプログラムが、代替の命令セットコンパイラ７０８を用いてコンパイルされて、少なくとも１つのｘ８６命令セットコア７１４を用いないでプロセッサ（例えば、カリフォルニア州サニーベールのＭＩＰＳテクノロジーズの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ホールディングスのＡＲＭ命令セットを実行する複数のコアを有するプロセッサ）により本来的に実行されてよい代替の命令セットバイナリコード７１０を生成してよいことを示す。命令コンバータ７１２は、ｘ８６バイナリコード７０６を、ｘ８６命令セットコア７１４を用いないで、プロセッサにより本来的に実行されてよいコードに変換するために用いられる。この変換されたコードは、これが可能な命令コンバータは作るのが困難であるので、代替の命令セットバイナリコード７１０と同じである可能性は低い。しかし、変換されたコードは、一般的な演算を遂行し、代替の命令セットからの複数の命令から構成される。従って、命令コンバータ７１２は、エミュレーション、シミュレーション、またはいずれの他の処理を通じて、プロセッサまたはｘ８６命令セットプロセッサまたはコアを有さない他の電子デバイスに、ｘ８６バイナリコード７０６を実行させるソフトウェア、ファームウェア、ハードウェア、またはそれらの組み合わせを表す。

【００５０】

複数の乗算演算を実行するための方法および装置

以下に記載の発明の実施形態は、単一の命令において２つの乗算を実行する乗算命令のファミリーに対する複数のアーキテクチャ上の拡張を提供する。一実施形態では、複数のアーキテクチャ上の拡張は、インテル（登録商標）アーキテクチャ（ＩＡ）に提供されるが、発明の基礎となる原理はいずれの特定のＩＳＡに限定されるものではない。

【００５１】

既存のプロセッサアーキテクチャでは、各乗算命令は、単一の乗算演算を実行する。例えば、インテル（登録商標）アーキテクチャでは、ＶＭＵＬＳＳおよびＶＭＵＬＰＳは、２つの単精度浮動小数点値を乗算し、ＶＭＵＬＳＤおよびＶＭＵＬＰＤは、２つの倍精度浮動小数点値を乗算する。対照的に、ここに記載の二重乗算命令のファミリー（一実施形態においてＶＭＵＬ３命令とラベルされる）は、単一の命令において２つの乗算を実行し、それにより、電力を低減し、他の複数の命令の複数のデコードスロットを解放する。一実施形態では、２つの乗算は、３つのソースオペランド上で実行される。第２及び第３のソースオペランドは、まず乗算されて、そして第１のソースオペランドにより乗算される中間結果を生成する。

【００５２】

図８に示されるように、発明の実施形態が実装されてよい典型的なプロセッサ８５５は、ここに記載の複数のＶＭＵＬ３命令を実行するＶＭＵＬ３実行ロジック８４１とともに実行ユニット８４０を含む。実行ユニット８４０が命令ストリームを実行するので、レジスタセット８０５は、複数のオペランド、制御データ、および他のタイプのデータに対するレジスタストレージを提供する。

【００５３】
簡単のため、単一のプロセッサコア（「コア０」）の詳細が図８に示される。しかし、図８に示される各コアは、コア０のように、ロジックの同じセットを有してよいことが理解される。示されるように、各コアは、特定のキャッシュ管理ポリシーに従って複数の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８１２およびレベル２（Ｌ２）キャッシュ８１１を含んでよい。Ｌ１キャッシュ８１２は、複数の命令を格納するための別個の命令キャッシュ８２０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８２１を含む。様々なプロセッサキャッシュ内に格納される複数の命令およびデータは、固定サイズ（例えば、６４、１２８、５１２バイト長）であってよい複数のキャッシュラインの粒度で管理される。この典型的な実施形態の各コアは、メインメモリ８００および／または共有レベル３（Ｌ３）キャッシュ８１６から複数の命令をフェッチするための命令フェッチユニット８１０、複数の命令をデコードする（例えば、複数のプログラム命令を複数のマイクロ演算または複数の「μｏｐ」にデコードする）ためのデコードユニット８３０、複数の命令（例えば、ここに記載されるような複数のＶＭＵＬ３命令）を実行するための実行ユニット８４０、および複数の命令をリタイヤし、複数の結果をライトバックするためのライトバックユニット８５０を有する。

【００５４】
命令フェッチユニット８１０は、メモリ８００（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８０３、最近用いられた仮想物理命令アドレスのマップを格納して、アドレス変換の速度を向上するための命令変換索引バッファ（ＩＴＬＢ）８０４、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８０２、および分岐アドレスおよび目標アドレスを格納するための複数の分岐目標バッファ（ＢＴＢ）８０１を含む様々な既知のコンポーネントを含む。フェッチされると、複数の命令は、デコードユニット８３０、実行ユニット８４０、およびライトバックユニット８５０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それぞれの構造および機能は、当業者に良く理解されており、発明の異なる実施形態の適切な態様を分かりにくくしないようにここでは詳細に記載されない。

【００５５】
発明の一実施形態では、ＶＭＵＬ３実行ロジック８４１は、次のファミリーの命令を実行する。

```
VMUL3SS xmm1{k1}{z}, xmm2, xmm3/mV{er}
VMUL3PS zmm1{k1}{z}, zmm2, zmm3/B32(mV){er}
VMUL3SD xmm1{k1}{z}, xmm2, xmm3/mV{er}
VMUL3PD zmm1{k1}{z}, zmm2, zmm3/B64(mV){er}
```

ここで、ｘｍｍ１－３およびｚｍｍ１－３は、単精度（３２ビット）または倍精度（６４ビット）浮動小数点フォーマットのいずれかで、パックドまたはスカラ浮動小数点値を格納するレジスタセット８０５内のレジスタである。

【００５６】
特に、一実施形態では、ＶＭＵＬ３ＳＳは、ｘｍｍ１、ｘｍｍ２、およびｘｍｍ３に格納される３つのスカラ、単精度浮動小数点値を乗算する。演算において、（ｘｍｍ２からの）第２のオペランドは（ｘｍｍ３からの）第３のオペランドにより乗算されてよく、結果は（ｘｍｍ１からの）第１のオペランドにより（中間丸めを有して）乗算され、デスティネーションレジスタに格納されてよい。一実施形態では、デスティネーションレジスタは、第１のオペランド（例えば、ｘｍｍ１）を格納するために用いられる同じレジスタである。

【００５７】
一実施形態では、ＶＭＵＬ３ＰＳは、ｚｍｍ１、ｚｍｍ２、およびｚｍｍ３に格納された３つのパックド、単精度浮動小数点値を乗算する。演算において、（ｚｍｍ２からの）第２のオペランドは（ｚｍｍ３からの）第３のオペランドにより乗算されてよく、結果は

（ｚｍｍ１からの）第１のオペランドにより（中間丸めを有して）乗算され、デスティネーションレジスタに格納されてよい。一実施形態では、デスティネーションレジスタは、第１のオペランド（例えば、ｚｍｍ１）を格納するために用いられる同じレジスタである。

【００５８】

一実施形態では、ＶＭＵＬ３ＳＤは、ｘｍｍ１、ｘｍｍ２、およびｘｍｍ３に格納された３つのスカラ、倍精度浮動小数点値を乗算する。演算において、（ｘｍｍ２からの）第２のオペランドは（ｘｍｍ３からの）第３のオペランドにより乗算されてよく、結果は（ｘｍｍ１からの）第１のオペランドにより（中間丸めを有して）乗算され、デスティネーションレジスタに格納されてよい。一実施形態では、デスティネーションレジスタは、第１のオペランド（例えば、ｘｍｍ１）を格納するために用いられる同じレジスタである。

【００５９】

最後に、一実施形態では、ＶＭＵＬ３ＰＤは、ｚｍｍ１、ｚｍｍ２、およびｚｍｍ３に格納された３つのパックド、倍精度浮動小数点値を乗算する。演算において、（ｚｍｍ２からの）第２のオペランドは（ｚｍｍ３からの）第３のオペランドにより乗算されてよく、結果は（ｚｍｍ１からの）第１のオペランドにより（中間丸めを有して）乗算され、デスティネーションレジスタに格納されてよい。一実施形態では、デスティネーションレジスタは、第１のオペランド（例えば、ｚｍｍ１）を格納するために用いられる同じレジスタである。

【００６０】

一実施形態では、複数のＶＭＵＬ３命令のそれぞれの３つの即値ビット［２：０］は、複数の乗算の符号を制御するために用いられる。例えば、即値のビット０の値は、第１のオペランドの符号を制御してよい（例えば、１＝負および０＝正、またはその逆）。即値のビット１の値は、第２のオペランドの符号を制御してよい。また、即値のビット２の値は、第３のオペランドの符号を制御してよい。

【００６１】

一実施形態では、第１および第２のオペランドは、複数の単一命令複数データ（ＳＩＭＤ）レジスタから読まれ、第３のオペランドは、ＳＩＭＤレジスタまたはメモリ位置から読まれることができる。

【００６２】

図９Ａは、各ＶＭＵＬ３の複数のμｏｐに複数のリソースを割り当てるためのアロケータ９４０、および複数の機能ユニット９１２により実行されるＶＭＵＬ３の複数のμｏｐをスケジュールするためのリザベーションステーション９０２を含むＶＭＵＬ３実行ロジック８４１の一実施形態に関連する追加的な詳細を示す。演算では、各ＶＭＵＬ３命令が複数のμｏｐにデコードされるデコードステージ８３０に続いて、命令デコーダ８０６は、複数のμｏｐをレジスタエイリアステーブル（ＲＡＴ）９４１を含むアロケータユニット９４０に転送する。アウトオブオーダパイプラインにおいて、アロケータユニット９４０は、各入力μｏｐをリオーダバッファ（ＲＯＢ）９５０内の位置に割り当て、それにより、μｏｐの論理デスティネーションアドレスをＲＯＢ９５０内の対応する物理デスティネーションアドレスにマッピングする。ＲＡＴ９４１は、このマッピングを維持する。

【００６３】

ＲＯＢ９５０の複数のコンテンツは、最終的に、リアルレジスタファイル（ＲＲＦ）９５１内の複数の位置にリタイヤされてよい。ＲＡＴ９４１は、論理アドレスにより示された値が、リタイヤの後に、ＲＯＢ９５０内またはＲＲＦ９５１内の物理アドレスで見つかるかどうかを示すリアルレジスタファイルの有効ビットを格納してもよい。ＲＲＦ内に見つかると、値は、現在のプロセッサのアーキテクチャ状態の一部と考えられる。このマッピングに基づいて、ＲＡＴ９４１は、また、すべての論理ソースアドレスをＲＯＢ９５０またはＲＲＦ９５１内の対応する位置に結合する。

【００６４】

各入力μｏｐは、また、アロケータ９４０により割り当てられて、リザベーションステ

ーション（ＲＳ）９０２内のエントリに書き込まれる。リザベーションステーション９０２は、機能ユニット９１２による実行を待つＶＭＵＬ３の複数のμｏｐを組み立てる。簡単な場合において、２つの融合乗算および加算（ＦＭＡ）機能ユニットＦＭＡ０　９１０およびＦＭＡ１　９１１は、以下に記載されるように複数のＶＭＵＬ３命令を実行する複数の乗算演算を実行する。必要に応じて、複数の結果は、ライトバックバスを介してＲＳ９０２にライトバックされてよい。

【００６５】

一実施形態では、複数のリザベーションステーションエントリは、複数のグループに論理的に細分され、複数のエントリを読み出すおよび書き込むためにそれぞれ必要とされるリードおよびライトポートの数を減らす。図９Ａに示される実施形態では、２つのリザベーションステーションのグループＲＳ０　９００およびＲＳ１　９０１は、それぞれポート０および１を介してＦＭＡ０　９１０およびＦＭＡ１　９１１機能ユニットによるＶＭＵＬ３の複数のμｏｐの実行をスケジュールする。

【００６６】

一実施形態では、複数のＶＭＵＬ３命令のいずれかは、パイプラインを介して単一のμｏｐとして実行されてよい。特に、μｏｐは、まず、第２および第３のオペランドの第１の乗算を実行して（例えば、上述のようにｘｍｍ２／ｘｍｍ３またはｚｍｍ２／ｚｍｍ３から）、中間結果を生成するＦＭＡ０　９１０（ＲＳ０　９００を介して）により実行される。μｏｐは、バッファユニット９０５内で遅延され、そして、ＦＭＡ１　９１１（ＲＳ１　９０１を介して）により２回目に実行されて、中間結果と第１のオペランド（例えば、ｘｍｍ１／ｚｍｍ１から）とを乗算する。前述のように、最終結果は、ｘｍｍ１／ｚｍｍ１内に格納されてよい。更に、述べたように、ＶＭＵＬ３命令の即値は、３つのソースオペランドのそれぞれの符号を特定してよい。一実施形態では、μｏｐの第２の発行は、命令を再発行する前に、正確にＦＭＡレイテンシ（例えば、５クロックサイクル）待たされる（バッファ９０５を介して）。

【００６７】

様々な既存のデータバイパスは、ポート１のＦＭＡ１　９１１に中間結果を提供するために用いられてよい。一実施形態では、中間結果は、ＲＯＢ　９５０、またはＦＭＡ１　９１１によりそこから読み出され、用いられてよいいずれの他の記憶位置内に一時的に格納される。一実施形態では、ライトバックバスは、中間結果をポート１を介してＦＡＭ１　９１１に利用できるようにするＲＳ１　９０１に中間結果を提供するために用いられてよい。しかし、発明の基礎となる原理は、中間結果をＦＡＭ１　９１１に提供する任意の特定のやり方に限定されない。さらに、ＲＯＢ９５０が図９Ａに示されるように、幾つかのプロセッサの実装（例えば、複数のインオーダパイプライン）において、ＲＯＢ　９５０は用いられず、異なる形式のストレージが、中間結果および実行に続く最終結果を格納するために用いられてよいことが理解される。

【００６８】

図９Ｂに示されるように、２つの機能ユニットは、発明の基礎となる原理を実装するのに必要ではない。詳細には、この実施形態において、同じ機能ユニット（ＦＭＡ０　９１０）は、続けて２回、ＶＭＵＬ３のμｏｐを実行して、最終結果を生成する。すなわち、ＦＭＡ０　９１０は、第２および第３のオペランドの間の第１の乗算を実行し、中間結果およびμｏｐをそれ自体を介して戻して再循環して、第２の乗算（完了すると、パイプラインの残りを通過する）を実行する。一実施形態では、μｏｐの第２の反復は、リザベーションステーション９０２を介して送信するよう示され、再循環は、単に、機能ユニットステージ９１２内で実行される（すなわち、機能ユニットステージ９１２内で一時バッファストレージを用いてＦＭＡ０　９１０からそれ自体に直接）。さらに、別の実装では、複数の機能ユニット９１２のセット内の新しい専用の機能ユニットは、ＶＭＵＬ３命令を独立して（すなわち、融合乗算および加算機能ユニットを用いないで）実行する。

【００６９】

上記の実施形態は、１つの命令のみがデコードされたような、２つのＶＭＵＬ命令を用

いる場合より改善された電力消費を提供する。さらに、一時的なソースが複数のバイパスを介して読み出されることが保証されたことで、データはレジスタファイルから読み出される必要はない。

【００７０】

幾つかの要素がともに乗算される複数のアプリケーションでは、乗算命令の数は、ここに記載の複数のＶＭＵＬ３命令を利用することで２で除算されることができる。例として、ベクトル化されることができる、ただし複数の浮動小数点値が乗算される長いループに対して、ＶＭＵＬ３は、命令数を仮想的に２減らすのに用いられてよい。

【００７１】

複数の乗算演算を実行するための方法の一実施形態が、図１０に示される。１００１にて、単一のＶＭＵＬ３命令が、メモリサブシステムからフェッチされる。述べたように、ＶＭＵＬ３命令は、第１、第２、第３のソースオペランド、デスティネーションオペランド、および即値を含む。１００２にて、ＶＭＵＬ３命令は、複数のμｏｐにデコードされる。上述のように、一実施形態では、単一の乗算μｏｐが生成されてよい（および、ＶＭＵＬ３命令を完了するのに必要とされる２つの乗算演算のために２回実行されてよい）。

【００７２】

１００３にて、複数のソースオペランド値が、複数の機能ユニットによる実行のための準備として取り出される。この演算は、例えば、リザベーションステーション９０２および／またはアロケータユニット９４０により実行されてよい。

【００７３】

１００４にて、ＶＭＵＬ３命令が実行される。一実施形態では、乗算μｏｐが、一度、第２及び第３のオペランドを用いて実行されて、中間結果を生成する。μｏｐは、そして２回目に、中間結果および第１のオペランドを用いて実行されて、最終結果（すなわち、第１、第２、及び第３のソースオペランドの乗算）を生成する。述べたように、複数のソースオペランドのそれぞれの符号は、３ビット中間値として提供されてよい。

【００７４】

１００５にて、ＶＭＵＬ３命令の結果が、１または複数の続く演算のためにそこから読み出されてよいデスティネーションオペランドの位置（例えば、レジスタ）に格納される。

【００７５】

典型的な命令フォーマット

ここに記載の命令の複数の実施形態は、異なるフォーマットで実施されてよい。更に、典型的な複数のシステム、複数のアーキテクチャ、および複数のパイプラインが以下に詳述される。命令の複数の実施形態は、そ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、詳述されるそれらに限定されるものではない。

【００７６】

ベクトル向け命令フォーマットは、複数のベクトル命令（例えば、複数のベクトル演算に固有の特定の複数のフィールドがある）に好適な命令フォーマットである。複数の実施形態は、ベクトル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じてサポートされるよう記載され、代替的な複数の実施形態は、ベクトル向け命令フォーマットを通じてサポートされるベクトル演算のみを用いる。

【００７７】

図１１Ａおよび図１１Ｂは、発明の実施形態に係る総称ベクトル向け命令フォーマットおよびそれの複数の命令テンプレートを示すブロック図である。図１１Ａは、発明の実施形態に係る総称ベクトル向け命令フォーマットおよびそれのクラスＡの複数の命令テンプレートを示すブロック図であり、図１１Ｂは、発明の実施形態に係る総称ベクトル向け命令フォーマットおよびそれのクラスＢの複数の命令テンプレートを示すブロック図である。詳細には、総称ベクトル向け命令フォーマット１５００に対して、両方が非メモリアクセス１５０５の命令テンプレートおよびメモリアクセス１５２０の命令テンプレートを含

むクラスＡおよびクラスＢの命令テンプレートが定義される。ベクトル向け命令フォーマットのコンテキストにおける総称（generic）なる用語は、いずれの固有の命令セットに関連付けられていない命令フォーマットを意味する。

【００７８】

発明の複数の実施形態は、ベクトル向け命令フォーマットが以下をサポートするように記載される。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）（従って、１６ダブルワードサイズ要素または代替的に８クワッドワードサイズ要素のいずれからなる６４バイトベクトル）。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）。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）。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）を有する１６バイトベクトルオペランド長（またはサイズ）。また、代替的な複数の実施形態は、より多い、より少ない、または異なるデータ要素幅（例えば、１６８ビット（１６バイト）データ要素幅）を有するより多い、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。

【００７９】

図１１Ａ内のクラスＡの複数の命令テンプレートは、１）非メモリアクセス１５０５の複数の命令テンプレート内に示される非メモリアクセス、完全ラウンド制御型演算１５１０の命令テンプレートおよび非メモリアクセス、データ変換型演算１５１５の命令テンプレート、および２）メモリアクセス１５２０の複数の命令テンプレート内に示されるメモリアクセス、一時的１５２５の命令テンプレートおよびメモリアクセス、非一時的１５３０の命令テンプレートを含む。図１１Ｂ内のクラスＢの複数の命令テンプレートは、１）非メモリアクセス１５０５の複数の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、部分ラウンド制御型演算１５１６の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型演算１５１７の命令テンプレート、および２）メモリアクセス１５２０の複数の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１５２７の命令テンプレートを含む。

【００８０】

総称ベクトル向け命令フォーマット１５００は、図１１Ａおよび図１１Ｂに順に示され、以下に列挙される次の複数のフィールドを含む。

【００８１】

フォーマットフィールド１５４０－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を、従って、命令ストリームにおけるベクトル向け命令フォーマット内の複数の命令の複数の発生をユニークに特定し。そのように、このフィールドは、総称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必要とされないという意味において任意である。

【００８２】

ベース演算フィールド１５４２－そのコンテンツは、異なるベース演算を区別する。

【００８３】

レジスタインデックスフィールド１５４４－そのコンテンツは、直接またはアドレス生成を介して、複数のレジスタ内またはメモリ内にあるソースおよびデスティネーションオペランドの位置を特定する。これらは、ＰｘＱ（例えば、３２ｘ５１６、１６ｘ１６８、３２ｘ１０２４、６４ｘ１０２４）レジスタファイルからＮのレジスタを選択するのに十分な数のビットを含む。一実施形態では、Ｎは３つのソースおよび１つのデスティネーションレジスタに及んでよく、代替的な複数の実施形態はより多い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、２つのソースまでサポートしてよい。ただし、これらのソースのうちの１つはデスティネーションとしてもふる

まう。また、３つのソースまでサポートしてよい。ただし、これらのソースのうちの１つはデスティネーションとしてもふるまう。また、２つのソースおよび１つのデスティネーションまでサポートしてよい。）

【００８４】

修飾子フィールド１５４６－そのコンテンツは、そうでないものから、すなわち非メモリアクセス１５０５の複数の命令テンプレートおよびメモリアクセス１５２０の複数の命令テンプレートの間で、メモリアクセスを特定する総称ベクトル命令フォーマット内の複数の命令の複数の発生を区別する。複数のメモリアクセス演算は、（幾つかのケースでは、複数のレジスタ内の複数の値を用いてソースおよび／またはデスティネーションアドレスを特定する）メモリ階層を読み出すおよび／または書き込み、複数の非メモリアクセス演算はそれをしない（例えば、ソースおよび複数のデスティネーションはレジスタである）。一実施形態では、このフィールドは、また、３つの異なる態様の間で選択して、複数のメモリアドレス算出を実行し、代替的な複数の実施形態はより多い、より少ない、または異なる態様をサポートして、複数のメモリアドレス算出を実行してよい。

【００８５】

増加演算フィールド１５５０－そのコンテンツは、様々な異なる演算のうちのどの１つがベース演算に加えて実行されるかを区別する。このフィールドは、コンテキスト固有である。発明の一実施形態では、このフィールドは、クラスフィールド１５６８、アルファフィールド１５５２、およびベータフィールド１５５４に分割される。増加演算フィールド１５５０は、２、３、または４つの命令ではなく単一の命令において実行される複数の演算の共通グループを可能とする。

【００８６】

スケールフィールド１５６０－そのコンテンツは、メモリアドレス生成のためのインデックスフィールドのコンテンツのスケーリングを可能とする（例えば、アドレス生成に対して２のスケール乗のインデックス+ベースを用いる）。

【００８７】

変位フィールド１５６２Ａ－そのコンテンツは、メモリアドレス生成の一部として用いられる（例えば、アドレス生成に対して２のスケール乗のインデックス+ベース+変位を用いる）。

【００８８】

変位ファクタフィールド１５６２Ｂ（なお、変位ファクタフィールド１５６２Ｂの直上の変位フィールド１５６２Ａの並置は１または他が用いられることを示す）－そのコンテンツは、アドレス生成の一部として用いられる。それは、メモリアクセスのサイズ（Ｎ）によりスケールされる変位ファクタを特定する。ただし、Ｎは、メモリアクセスにおけるバイト数である（例えば、アドレス生成に対して２のスケール乗のインデックス+ベース+スケールされた変位を用いる）。冗長下位ビットは無視され、従って、変位ファクタフィールドのコンテンツは、複数のメモリオペランドの総サイズ（Ｎ）により乗算されて、実効アドレスの計算において用いられる最終変位を生成する。Ｎの値は、（ここに記載の）フルオペコードフィールド１５７４およびデータ操作フィールド１５５４Ｃに基づいて、実行時に、プロセッサハードウェアにより決定される。変位フィールド１５６２Ａおよび変位ファクタフィールド１５６２Ｂは、それらは非メモリアクセス１５０５の複数の命令テンプレートに対して用いられないという意味において任意であり、および／または異なる実施形態は２つのうちの１つのみを実装してよい、またはいずれも実装しなくてよい。

【００８９】

データ要素幅フィールド１５６４－そのコンテンツは、多くのデータ要素幅のうちのどの１つが用いられるかを区別する（幾つかの実施形態では、すべての命令に対して、他の複数の実施形態では、複数の命令のうちの幾つかのみに対して）。このフィールドは、複数のオペコードの幾つかの態様を用いて、１つのデータ要素幅のみがサポートされる、および／または複数のデータ要素幅がサポートされる場合、必要とされないという意味において任意である。

【００９０】
書き込みマスクフィールド１５７０－そのコンテンツは、データ要素の位置に基づいて、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の位置がベース演算および増加演算の結果を反映するかどうかを制御する。クラスＡの複数の命令テンプレートは、差込みライトマスクをサポートし、クラスＢの複数の命令テンプレートは、差込みおよびゼロ化ライトマスクの両方をサポートする。複数の差込み、ベクトルマスクは、デスティネーション内の複数の要素のいずれのセットに、いずれの演算（ベース演算および増加演算により特定される）の実行中のアップデートからプロテクトされることを可能とする。他の一実施形態では、対応するマスクビットが０を有するデスティネーションの各要素の古い値を保存する。対照的に、ゼロ化ベクトルマスクは、デスティネーション内の複数の要素のいずれのセットに、いずれの演算（ベース演算および増加演算により特定される）の実行中にゼロ化されることを可能とする。一実施形態では、対応するマスクビットが０値を有するとき、デスティネーションの要素が０にセットされる。この機能性のサブセットは、実行されている演算のベクトル長を制御する能力である（すなわち、複数の要素のスパンが１つめから最後の１つまで変更される）。しかし、変更される複数の要素が連続する必要はない。従って、書き込みマスクフィールド１５７０は、複数のロード、複数のストア、算術、論理等を含む複数の部分的なベクトル演算を可能とする。発明の複数の実施形態は、書き込みマスクフィールド１５７０のコンテンツが、用いられる書き込みマスクを含む多くの書き込みマスクレジスタのうちの１つを選択する（従って、書き込みマスクフィールド１５７０のコンテンツは、間接的に、実行されるマスキングを特定する）ように記載され、代替的な実施形態は、代わりにまたは追加的に、書き込みマスクフィールド１５７０のコンテンツに、直接、実行されるマスキングを特定させる。

【００９１】
即値フィールド１５７２－そのコンテンツは、即値の指定を可能とする。このフィールドは、即値をサポートしない総称ベクトル向けフォーマットの実装において存在せず、即値を用いない複数の命令において存在しないという意味において任意である。

【００９２】
クラスフィールド１５６８－そのコンテンツは、異なるクラスの複数の命令の間で区別する。図１１Ａおよび図１１Ｂを参照して、このフィールドのコンテンツは、クラスＡおよびクラスＢの複数の命令の間で選択する。図１１Ａおよび図１１Ｂにおいて、複数の丸角の正方形は、フィールド内に特定の値があることを示すために用いられる（例えば、図１１Ａおよび図１１Ｂのそれぞれにクラスフィールド１５６８に対してクラスＡ１５６８Ａ及びクラスＢ１５６８Ｂ）。

【００９３】
クラスＡの命令テンプレート
クラスＡの非メモリアクセス１５０５の複数の命令テンプレートの場合、アルファフィールド１５５２は、そのコンテンツが、複数の異なる増加演算型のどの１つが実行されるかを区別するＲＳフィールド１５５２Ａとして解釈され（例えば、ラウンド１５５２Ａ．１およびデータ変換１５５２Ａ．２はそれぞれ非メモリアクセス、ラウンドタイプ演算１５１０および非メモリアクセス、データ変換型演算１５１５の複数の命令テンプレートに対して特定される）、ベータフィールド１５５４は、指定される型の複数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１５０５の複数の命令テンプレート内には、スケールフィールド１５６０、変位フィールド１５６２Ａ、および変位スケールフィールド１５６２Ｂは存在しない。

【００９４】
非メモリアクセスの命令テンプレート－完全ラウンド制御型演算
非メモリアクセスの完全ラウンド制御型演算１５１０の命令テンプレートにおいて、ベータフィールド１５５４は、そのコンテンツが静的丸め込みを提供するラウンド制御フィールド１５５４Ａとして解釈される。発明の記載の複数の実施形態では、ラウンド制御フィールド１５５４Ａは、浮動小数点例外（ＳＡＥ）フィールド１５５６およびラウンド演

算制御フィールド１５５８のすべての抑制を含み、代替的な複数の実施形態は、これらのコンセプトの両方をサポートし、同じフィールドにエンコードしてよく、またはこれらのコンセプト／フィールドの１つまたは他を単に有する（例えば、ラウンド演算制御フィールド１５５８のみを有してよい）。

【００９５】

ＳＡＥフィールド１５５６－そのコンテンツは、例外イベント報告をディスエーブルするか否かを区別する。ＳＡＥフィールド１５５６のコンテンツが可能な抑制を示すと、与えられた命令はすべての種類の浮動小数点例外フラグを報告せず、すべての浮動小数点例外処理部を立ち上げない。

【００９６】

ラウンド演算制御フィールド１５５８－そのコンテンツは、複数のラウンド演算のグループのどの１つが実行するかを区別する（例えば、切り上げ、切り捨て、ゼロへの丸め、および最近接丸め）。従って、ラウンド演算制御フィールド１５５８は、命令に基づいてラウンド演算モードの変更を可能とする。プロセッサが複数のラウンド演算モードを指定するための制御レジスタを含む発明の一実施形態では、ラウンド演算制御フィールド１５５０のコンテンツは、そのレジスタ値を上書きする。

【００９７】

非メモリアクセスの命令テンプレート：データ変換型演算

非メモリアクセスのデータ変換型演算１５１５の命令テンプレートにおいて、ベータフィールド１５５４は、そのコンテンツが多くのデータ変換（例えば、データ変換なし、スウィズル、ブロードキャスト）のうちのどの１つが実行されるかを区別するデータ変換フィールド１５５４Ｂとして解釈される。

【００９８】

クラスＡのメモリアクセス１５２０の命令テンプレートの場合、アルファフィールド１５５２は、そのコンテンツが複数の追い出し示唆のうちのどの１つが用いられるかを区別する追い出し示唆フィールド１５５２Ｂとして解釈され（図１２Ａでは、一時的１５５２Ｂ．１および非一時的１５５２Ｂ．２は、それぞれ、メモリアクセス、一時的１５２５の命令テンプレートおよびメモリアクセス、非一時的１５３０の命令テンプレートに対して特定される）、ベータフィールド１５５４は、そのコンテンツが多くのデータ操作演算（プリミティブとも知られる）のうちのどの１つが実行されるかを区別するデータ操作フィールド１５５４Ｃとして解釈される（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、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１５２０の複数の命令テンプレートは、スケールフィールド１５６０、任意で変位フィールド１５６２Ａまたは変位スケールフィールド１５６２Ｂを含む。

【００９９】

複数の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。正規の複数のベクトル命令を用いるように、複数のベクトルメモリ命令は、データ要素ごとの様式で、実際に転送され、書き込みマスクとして選択されるベクトルマスクの複数のコンテンツにより命令される複数の要素を用いて、メモリから／へデータを転送する。

【０１００】

メモリアクセスの命令テンプレート－一時的

一時的なデータは、キャッシュにより利益を得るのに十分にすぐに再利用され得るデータである。しかし、これは示唆であり、異なるプロセッサは、示唆を完全に無視することを含め、それを異なる態様で実装してよい。

【０１０１】

メモリアクセスの命令テンプレート－非一時的

非一時的データは、第１レベルキャッシュにキャッシュすることより利益を得るのに十分にすぐに再利用され得るデータであり、削除の優先度を与えられるべきである。しかし、これは示唆であり、異なるプロセッサは、示唆を完全に無視することを含め、それを異

なる態様で実装してよい。

【０１０２】

クラスＢの命令テンプレート

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１５５２は、そのコンテンツが、書き込みマスクフィールド１５７０により制御される書き込みマスキングが差込みまたはゼロ化であるべきかどうかを区別する書き込みマスク制御（Ｚ）フィールド１５５２Ｃとして解釈される。

【０１０３】

クラスＢの非メモリアクセス１５０５の複数の命令テンプレートの場合、ベータフィールド１５５４の一部は、そのコンテンツが、異なる増加演算型のうちのどの１つが実行されるかを区別するＲＬフィールド１５５７Ａとして解釈され（例えば、ラウンド１５５７Ａ．１およびベクトル長（ＶＳＩＺＥ）１５５７Ａ．２は、それぞれ、非メモリアクセス、書き込みマスク制御の部分ラウンド制御型演算１５１６の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型演算１５１７の命令テンプレートに対して特定される）、ベータフィールド１５５４の残りは、指定される型の複数の演算のうちのどれが実行されるかを区別する。非メモリアクセス１５０５の複数の命令テンプレートには、スケールフィールド１５６０、変位フィールド１５６２Ａ、および変位スケールフィールド１５６２Ｂは存在しない。

【０１０４】

非メモリアクセス、書き込みマスク制御の部分ラウンド制御型演算１５１６の命令テンプレートでは、ベータフィールド１５５４の残りは、ラウンド演算フィールド１５５９Ａとして解釈され、例外イベント報告がディスエーブルされる（与えられた命令は、すべての種類の浮動小数点例外フラグを報告せず、すべての浮動小数点例外処理部を立ち上げない）。

【０１０５】

ラウンド演算制御フィールド１５５９Ａ－ラウンド演算制御フィールド１５５８と同じように、そのコンテンツは、複数のラウンド演算のグループのどの１つが実行するかを区別する（例えば、切り上げ、切り捨て、ゼロへの丸め、および最近接丸め）。従って、ラウンド演算制御フィールド１５５９Ａは、命令に基づいて、ラウンド演算モードの変更を可能とする。プロセッサがラウンド演算モードを指定するための制御レジスタを含む発明の一実施形態では、ラウンド演算制御フィールド１５５０のコンテンツはそのレジスタ値を上書きする。

【０１０６】

非メモリアクセス、書き込みマスク制御、ＶＳＩＺＥ型演算１５１７の命令テンプレートにおいて、ベータフィールド１５５４の残りは、そのコンテンツが多くのデータベクトル長のどの１つが実行されるか（例えば、１６８、２５６、または５１６バイト）を区別するベクトル長フィールド１５５９Ｂとして解釈される。

【０１０７】

クラスＢのメモリアクセス１５２０の命令テンプレートの場合、ベータフィールド１５５４の一部は、そのコンテンツがブロードキャストタイプのデータの操作演算が実行されるか否かを区別するブロードキャストフィールド１５５７Ｂとして解釈され、ベータフィールド１５５４の残りはベクトル長フィールド１５５９Ｂとし解釈される。メモリアクセス１５２０の複数の命令テンプレートは、スケールフィールド１５６０、および任意で変位フィールド１５６２Ａまたは変位スケールフィールド１５６２Ｂを含む。

【０１０８】

総称ベクトル向け命令フォーマット１５００に関連して、フルオペコードフィールド１５７４は、フォーマットフィールド１５４０、ベース演算フィールド１５４２、およびデータ要素幅フィールド１５６４を含んで示される。一実施形態は、フルオペコードフィールド１５７４がこれらのフィールドのすべてを含むように示され、フルオペコードフィールド１５７４は、それらのすべてをサポートしない複数の実施形態では、これらのフィー

ルドのすべてより少ないフィールドを含む。フルオペコードフィールド１５７４は、演算コード（オペコード）を提供する。

【０１０９】

増加演算フィールド１５５０、データ要素幅フィールド１５６４、および書き込みマスクフィールド１５７０は、これらの特徴を、命令に基づいて、総称ベクトル向け命令フォーマットにおいて特定されるようにする。

【０１１０】

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は、それらがマスクを異なるデータ要素幅に基づいて適用されることを可能とする型付けされた複数の命令を生成する。

【０１１１】

クラスＡおよびクラスＢ内の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。発明の幾つかの実施形態では、異なるプロセッサまたはプロセッサ内の異なるコアは、クラスＡのみ、クラスＢのみ、または両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のために意図された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のために意図されたコアは、クラスＡのみをサポートしてよく、両方のために意図されたコアは、両方をサポートしてよい（もちろん、両方のクラスからのすべてのテンプレートおよび命令ではなく、両方のクラスからの複数のテンプレートおよび複数の命令の幾つかのミックスを有するコアは発明の範囲内である）。また、単一のプロセッサは、すべてが同じクラスをサポートする、または異なるコアが異なるクラスをサポートするマルチコアを含んでよい。例えば、別個のグラフィックおよび複数の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のために意図された複数のグラフィックコアの１つは、クラスＡのみをサポートしてよく、複数の汎用コアのうちの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のために意図されたアウトオブオーダ実行およびレジスタリネームを有する高性能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、１つのクラスからの複数の機能は、発明の異なる実施形態において他のクラスに実装されてもよい。高級言語で書かれた複数のプログラムは、１）実行のために目標プロセッサによりサポートされるクラスの複数の命令のみを有する形式、または２）すべてのクラスの複数の命令の異なる組み合わせを用いて書かれた代替的な複数のルーチンを有し、現在コードを実行しているプロセッサにより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入れられる（例えば、ジャストインタイムにコンパイルされるまたは静的にコンパイルされる）。

【０１１２】

図１２Ａから図１２Ｄは、発明の複数の実施形態に係る典型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１２Ａから図１２Ｄは、複数のフィールドの位置、サイズ、解釈、および順序と、それらのフィールドの幾つかに対する複数の値を特定するという意味において固有である特定ベクトル向け命令フォーマット１６００を示す。特定ベクトル向け命令フォーマット１６００は、ｘ８６命令セットを拡張するために用いられてよく、従って、複数のフィールドのうちの幾つかは、既存のｘ８６命令セットおよびそのエクステンション（例えば、ＡＶＸ）において用いられるそれらと同様または同じである。このフォーマットは、複数のエクステンションを有する既存のｘ８６命令セットのプレフィックス符号化フィールド、実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、および複数の即値フィールドとの一致を維持する。図１２Ａから図１２Ｄからの複数のフィールドがマップされる図１１Ａおよび図１１Ｂからの複数のフィールドが示される。

【０１１３】

発明の複数の実施形態は、説明の目的のため、総称ベクトル向け命令フォーマット１５００のコンテキストにおいて特定ベクトル向け命令フォーマット１６００を参照して記載されるが、発明は、特許請求の範囲に記載されたものを除いて特定ベクトル向け命令フォーマット１６００に限定されるものではないことが理解されるべきである。例えば、総称ベクトル向け命令フォーマット１５００は、様々なフィールドの様々な可能なサイズを予想し、特定ベクトル向け命令フォーマット１６００は、固有の複数のサイズの複数のフィールドを有するように示される。具体的な例として、データ要素幅フィールド１５６４は、特定ベクトル向け命令フォーマット１６００内の１つのビットフィールドとして示されるが、発明はこれに限定されない（すなわち、総称ベクトル向け命令フォーマット１５００は、データ要素幅フィールド１５６４の他の複数のサイズを予想する）。

【０１１４】

総称ベクトル向け命令フォーマット１５００は、図１２Ａに順に示され、以下に列挙される次の複数のフィールドを含む。

【０１１５】

ＥＶＥＸ　Ｐｒｅｆｉｘ（バイト０－３）１６０２は、４バイト形式でエンコードされる。

【０１１６】

フォーマットフィールド１６４０（ＥＶＥＸバイト０、ビット［７：０］）－第１バイト（ＥＶＥＸバイト０）はフォーマットフィールド１６４０であり、０ｘ６２（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用いられるユニークな値）を含む。

【０１１７】

第２から第４バイト（ＥＶＥＸバイト１－３）は、固有の機能を提供する多くのビットフィールドを含む。

【０１１８】

ＲＥＸフィールド１６０５（ＥＶＥＸバイト１、ビット［７－５］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１５５７ＢＥＸバイト１、ビット［５］－Ｂ）からなる。ＥＶＥＸ．Ｒ、ＥＶＥＸ．Ｘ、およびＥＶＥＸ．Ｂビットフィールドは、対応する複数のＶＥＸビットフィールドと同じ機能性を提供し、１の補数形式を用いてエンコードされる、すなわち、ＺＭＭ０は１６１１Ｂとしてエンコードされ、ＺＭＭ１５は０００００Ｂとしてエンコードされる。当該分野において知られているように、複数の命令の他の複数のフィールドは、複数のレジスタインデックスのより低い３つのビット（ｒｒｒ、ｘｘｘ、およびｂｂｂ）をエンコードし、それにより、Ｒｒｒｒ、Ｘｘｘｘ、およびＢｂｂｂはＥＶＥＸ．Ｒ、ＥＶＥＸ．Ｘ、およびＥＶＥＸ．Ｂを加えることにより形成されてよい。

【０１１９】

ＲＥＸ'フィールド１６０５－これは、ＲＥＸ'フィールド１５１０の第１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用いられるＥＶＥＸ．Ｒ'ビットフィールド（ＥＶＥＸバイト１、ビット［４］－Ｒ'）である。発明の一実施形態では、このビットは、以下に示されるように他とともに、その実オペコードバイトが６２であるＢＯＵＮＤ命令から区別するためにビット反転フォーマットで（既知のｘ８６　３２ビットモードで）格納され、しかし、ＭＯＤ　Ｒ／Ｍフィールド内で、ＭＯＤフィールド内の１１の値を受け入れない。発明の代替的な複数の実施形態は、これと反転フォーマットで以下に示される他のビットを格納しない。１の値は、下位の１６のレジスタをエンコードするために用いられる。言い換えると、Ｒ'Ｒｒｒｒは、ＥＶＥＸ．Ｒ'、ＥＶＥＸ．Ｒ、および他の複数のフィールドからの他のＲＲＲを結合することにより形成される。

【０１２０】

オペコードマップフィールド１６１５（ＥＶＥＸバイト１、ビット［３：０］－ｍｍｍ

ｍ）－そのコンテンツは、暗黙の主要なオペコードバイト（０Ｆ、０Ｆ　３８、または０Ｆ　３）をエンコードする。

【０１２１】

データ要素幅フィールド１６６４（ＥＶＥＸバイト２、ビット［７］－Ｗ）は、標記ＥＶＥＸ．Ｗにより表される。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いずれ）の粒度（サイズ）を定義するために用いられる。

【０１２２】

ＥＶＥＸ．ｖｖｖｖ１６２０（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、反転（１の補数）形式で特定される第１のソースレジスタオペランドをエンコードし、２またはそれより多いソースオペランドを有する複数の命令に対して有効である。２）ＥＶＥＸ．ｖｖｖｖは、あるベクトルシフトに対して１の補数形式で特定されるデスティネーションレジスタオペランドをエンコードする。または、３）ＥＶＥＸ．ｖｖｖｖは、いずれのオペランドもエンコードせず、フィールドは残される。従って、ＥＶＥＸ．ｖｖｖｖフィールド１６２０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、余分の異なるＥＶＥＸビットフィールドは、指定子サイズを３２のレジスタに拡張するために用いられる。

【０１２３】

ＥＶＥＸ．Ｕ１６６８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。

【０１２４】

プレフィックス符号化フィールド１６２５（ＥＶＥＸバイト２、ビット［１：０］－ｐｐ）は、ベース演算フィールドに対して追加的な複数のビットを提供する。ＥＶＥＸプレフィックスフォーマットにおける複数のレガシＳＳＥ命令に対するサポートを提供することに加えて、これは、ＳＩＭＤプレフィックスをコンパクトにする利益も有する（ＳＩＭＤプレフィックスを表すバイトを必要とするのではなく、ＥＶＥＸプレフィックスは２ビットのみを必要とする）。一実施形態では、レガシフォーマットおよびＥＶＥＸプレフィックスフォーマットの両方においてＳＩＭＤプレフィックス（６６Ｈ、Ｆ２Ｈ、Ｆ３Ｈ）を用いる複数のレガシＳＳＥ命令をサポートするために、これらのレガシＳＩＭＤプレフィックスは、ＳＩＭＤプレフィックス符号化フィールドにエンコードされ、デコーダのＰＬＡに提供される前に、実行時に、レガシＳＩＭＤプレフィックスに拡張される（従って、ＰＬＡは、これらのレガシ命令のレガシおよびＥＶＥＸフォーマットの両方を変更することなく実行することができる）。より新しい複数の命令は、ＥＶＥＸプレフィックス符号化フィールドのコンテンツを直接、オペコード拡張として用いることができたが、ある実施形態は、一貫性のために、しかしこれらのレガシＳＩＭＤプレフィックスにより特定される異なる意味を認める同様の様式で拡張する。代替的な実施形態は、２ビットＳＩＭＤプレフィックスエンコードをサポートするＰＬＡを再設計してよく、従って、拡張を必要としない。

【０１２５】

アルファフィールド１６５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ、ＥＶＥＸ．ＥＨ、ＥＶＥＸ．ｒｓ、ＥＶＥＸ．ＲＬ、ＥＶＥＸ．ｗｒｉｔｅ　ｍａｓｋ　ｃｏｎｔｒｏｌ、およびＥＶＥＸ．Ｎとしても知られ、またαを用いて示される）－先述の通り、このフィールドはコンテキスト固有である。

【０１２６】

ベータフィールド１６５４（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、またβββを用いて示される）－先述の通り、このフィールドはコンテキスト固有である。

【０１２７】

ＲＥＸ'フィールド１６１０－これは、ＲＥＸ'フィールドの残りであり、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用いられてよい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。１の値は、下位１６のレジスタをエンコードするために用いられる。言い換えると、Ｖ'ＶＶＶＶは、ＥＶＥＸ．Ｖ'、ＥＶＥＸ．ｖｖｖｖ．を結合することにより形成される。

【０１２８】

書き込みマスクフィールド１６７０（ＥＶＥＸバイト３、ビット［２：０］－ｋｋｋ）－そのコンテンツは、前述のとおり、複数の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特定する。発明の一実施形態では、特定の値ＥＶＥＸ．ｋｋｋ＝０００は、特定の命令に対して書き込みマスクが用いられないことを暗示する特別な振る舞いを有する（これは、マスキングハードウェアをバイパスするすべてのものまたはハードウェアに配線される書き込みマスクの使用を含む様々な態様において実装されてよい）。

【０１２９】

リアルオペコードフィールド１６３０（バイト４）は、オペコードバイトとしても知られる。オペコードの一部は、このフィールド内で特定される。

【０１３０】

ＭＯＤ　Ｒ／Ｍフィールド１６４０（バイト５）は、ＭＯＤフィールド１６４２、Ｒｅｇフィールド１６４４、およびＲ／Ｍフィールド１６４６を含む。前述のとおり、ＭＯＤフィールド１６４２のコンテンツは、メモリアクセスおよび非メモリアクセス演算の間を区別する。Ｒｅｇフィールド１６４４の役割は、２つの状況にまとめられることができる。すなわち、デスティネーションレジスタオペランドまた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、いずれの命令オペランドをエンコードするために用いられない。Ｒ／Ｍフィールド１６４６の役割は、次を含んでよい。すなわち、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また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。

【０１３１】

スケール、インデックス、ベース（ＳＩＢ）バイト（バイト６）－前述のとおり、スケールフィールド１６５０のコンテンツは、メモリアドレス生成のために用いられる。ＳＩＢ．ｘｘｘ１６５４およびＳＩＢ．ｂｂｂ１６５６－これらのフィールドのコンテンツは、前に、レジスタインデックスＸｘｘｘおよびＢｂｂｂに関連して参照された。

【０１３２】

変位フィールド１６６２Ａ（バイト７－１０）－ＭＯＤフィールド１６４２が１０を含むと、バイト７－１０は変位フィールド１６６２Ａであり、それはレガシ３２ビット変位（ｄｉｓｐ３２）と同じように機能し、バイト粒度で機能する。

【０１３３】

変位ファクタフィールド１６６２Ｂ（バイト７）－ＭＯＤフィールド１６４２が０１を含むとき、バイト７は変位ファクタフィールド１６６２Ｂ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）のそれと同じである。ｄｉｓｐ８は符号拡張されるので、それは、１６８および１６７バイトオフセットの間でのみアドレスすることができる。６４バイトキャッシュラインの観点において、ｄｉｓｐ８は、－１６８、－６４、０、および６４のたった４つの実に有用な値にセットされることができる８ビットを用いる。より大きい範囲が頻繁に必要とされるので、ｄｉｓｐ３２が用いられる。しかし、ｄｉｓｐ３２は４バイトを必要とする。ｄｉｓｐ８およびｄｉｓｐ３２と対照的に、変位ファクタフィールド１６６２Ｂはｄｉｓｐ８の再解釈である。変位ファクタフィールド１６６２Ｂを用いると、実際の変位は、メモリオペランドアクセスのサイズ（Ｎ）により乗算された変位ファクタフィールドのコンテンツにより決定される。このタイプの変位は、ｄｉｓｐ８×Ｎとして参照される。これは、平均命令長を減らす（変位に対して用いられた、しかしはるかにより大きい範囲を有する単一

バイト）。そのような圧縮された変位は、有効な変位がメモリアクセスの粒度の倍数であるという仮定に基づくので、従って、アドレスオフセットの冗長下位ビットは、エンコードされる必要はない。言い換えると、変位ファクタフィールド１６６２Ｂは、レガシｘ８６命令セットの８ビット変位を代替する。従って、変位ファクタフィールド１６６２Ｂは、ｄｉｓｐ８はｄｉｓｐ８×Ｎに上書きされる例外のみを用いて、ｘ８６命令セットの８ビット変位と同じ態様でエンコードされる（ＭｏｄＲＭ／ＳＩＢエンコードルールに変更はない）。言い換えると、（変位をメモリオペランドのサイズによりスケールして、バイト単位のアドレスオフセットを得る必要がある）ハードウェアによる変位値の解釈のみを除いて、複数のエンコードルールまたは複数のエンコード長に変更はない。即値フィールド１６７２は、前述のように動作する。

【０１３４】

フルオペコードフィールド

図１２Ｂは、発明の一実施形態に係るフルオペコードフィールド１６７４を作成する特定ベクトル向け命令フォーマット１６００の複数のフィールドを示すブロック図である。詳細には、フルオペコードフィールド１６７４は、フォーマットフィールド１６４０、ベース演算フィールド１６４２、およびデータ要素幅（Ｗ）のフィールド１６６４を含む。ベース演算フィールド１６４２は、プレフィックス符号化フィールド１６２５、オペコードマップフィールド１６１５、およびリアルオペコードフィールド１６３０を含む。

【０１３５】

レジスタインデックスフィールド

図１２Ｃは、発明の一実施形態に係るレジスタインデックスフィールド１６４４を作成する特定ベクトル向け命令フォーマット１６００の複数のフィールドを示すブロック図である。詳細には、レジスタインデックスフィールド１６４４は、ＲＥＸフィールド１６０５、ＲＥＸ'フィールド１６１０、ＭＯＤＲ／Ｍ．ｒｅｇフィールド１６４４、ＭＯＤＲ／Ｍ．ｒ／ｍフィールド１６４６、ＶＶＶＶフィールド１６２０、ｘｘｘフィールド１６５４、およびｂｂｂフィールド１６５６を含む。

【０１３６】

増加演算フィールド

図１２Ｄは、発明の一実施形態に係る増加演算フィールド１６５０を生成する特定ベクトル向け命令フォーマット１６００の複数の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１６６８が０を含むと、それはＥＶＥＸ．Ｕ０（クラスＡ１６６８Ａ）を示す。それが１を含むと、それはＥＶＥＸ．Ｕ１（クラスＢ１６６８Ｂ）を示す。Ｕ＝０且つＭＯＤフィールド１６４２が１１（非メモリアクセス演算を示す）を含むと、アルファフィールド１６５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ）はｒｓフィールド１６５２Ａとして解釈される。ｒｓフィールド１６５２Ａが１（ラウンド１６５２Ａ．１）を含むと、ベータフィールド１６５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ）は、ラウンド制御フィールド１６５４Ａとして解釈される。ラウンド制御フィールド１６５４Ａは、１ビットのＳＡＥフィールド１６５６および２ビットのラウンド演算フィールド１６５８を含む。ｒｓフィールド１６５２Ａが０（データ変換１６５２Ａ．２）を含むと、ベータフィールド１６５４（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１６５４Ｂとして解釈される。Ｕ＝０且つＭＯＤフィールド１６４２が００、０１、または１０（メモリアクセス演算を示す）を含むと、アルファフィールド１６５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ）は、追い出し示唆（ＥＨ）フィールド１６５２Ｂとして解釈され、ベータフィールド１６５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ）は、３ビットデータ操作フィールド１６５４Ｃとして解釈される。

【０１３７】

Ｕ＝１のとき、アルファフィールド１６５２（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１６５２Ｃとして解釈される。Ｕ＝１且つＭＯＤフィールド１６４２が１１（非メモリアクセス演算を示す）を含むと、ベータフィー

ルド１６５４の一部（ＥＶＥＸバイト３、ビット［４］－$Ｓ_{0}$）は、ＲＬフィールド１６５７Ａとして解釈される。それが１（ラウンド１６５７Ａ．１）を含むと、ベータフィールド１６５４の残り（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１６５９Ａとして解釈され、ＲＬフィールド１６５７Ａが０（ＶＳＩＺＥ１６５７．Ａ２）を含むと、ベータフィールド１６５４の残り（ＥＶＥＸバイト３、ビット［６－５］－$Ｓ_{2-1}$）は、ベクトル長フィールド１６５９Ｂ（ＥＶＥＸバイト３、ビット［６－５］－$Ｌ_{1-0}$）として解釈される。Ｕ＝１且つＭＯＤフィールド１６４２が００、０１、または１０（メモリアクセス演算を示す）を含むと、ベータフィールド１６５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ）は、ベクトル長フィールド１６５９Ｂ（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１６５７Ｂ（ＥＶＥＸバイト３、ビット［４］－Ｂ）として解釈される。

【０１３８】

図１３は、発明の一実施形態に係るレジスタアーキテクチャ１７００のブロック図である。示される実施形態では、５１６ビット幅の３２のベクトルレジスタ１７１０がある。これらのレジスタは、ｚｍｍ０からｚｍｍ３１として参照される。より低い１６のｚｍｍレジスタの下位の２５６ビットは、レジスタｙｍｍ０－１６上に上書きされる。より低い１６のｚｍｍレジスタの下位の１６８ビット（ｙｍｍレジスタの下位の１６８ビット）は、レジスタｘｍｍ０－１５上に上書きされる。特定ベクトル向け命令フォーマット１６００は、下の表に示されるように、これらの上書きレジスタファイル上で動作する。

【０１３９】

【表１】

調整可能なベクトル長	クラス	演算	レジスタ
ベクトル長フィールド1559Bを含まない命令テンプレート	A(図11A; U=0)	1510，1515，1525，1530	zmmレジスタ（ベクトル長は64バイト
	B(図11B; U=1)	1516	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1559Bを含む命令テンプレート	B(図11B; U=1)	1517，1527	ベクトル長フィールド1559Bに依存する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

【０１４０】

言い換えると、ベクトル長フィールド１５５９Ｂは、最大長さと１または複数の他のより短い長さとの間で選択する。ただし、そのようなより短い長さのそれぞれは、前長の長さの半分であり、ベクトル長フィールド１５５９Ｂを有さない複数の命令テンプレートは、最大ベクトル長で動作する。さらに、一実施形態では、特定ベクトル向け命令フォーマット１６００のクラスＢの複数の命令テンプレートは、パックドまたはスカラ単／倍精度浮動小数点データおよびパックドまたはスカラ整数データで動作する。複数のスカラ演算は、ｚｍｍ／ｙｍｍ／ｘｍｍレジスタ内の最下位データ要素位置で実行される演算である。より高位の複数のデータ要素位置は、命令の前のそれらと同じ状態のままにされる、または実施形態に応じてゼロ化される。

【０１４１】

書き込みマスクレジスタ１７１５－示される実施形態では、それぞれが６４ビットサイズの８つの書き込みマスクレジスタ（ｋ０からｋ７）がある。代替的な実施形態では、書き込みマスクレジスタ１７１５は１６ビットサイズである。前述のとおり、発明の一実施形態では、ベクトルマスクレジスタｋ０は、書き込みマスクとして用いられない。通常ｋ０を示すエンコードが書き込みマスクに対して用いられると、それは、その命令に対する

書き込みマスキングを効率的にディスエーブルする０ｘＦＦＦＦのハードワイヤ書き込みマスクを選択する。

【０１４２】

複数の汎用レジスタ１７２５－示される実施形態では、複数のメモリオペランドをアドレスする既存の複数のｘ８６アドレスモードとともに用いられる１６の６４ビット汎用レジスタがある。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。

【０１４３】

ＭＭＸパックド整数フラットレジスタファイル１７５０がエイリアスされるスカラ浮動小数点のスタックレジスタファイル（ｘ８７スタック）１７４５－示される実施形態では、ｘ８７スタックは、ｘ８７命令セットエクステンションを用いて３２／６４／８０ビット浮動小数点データで複数のスカラ浮動小数点演算を実行するために用いられる８要素スタックである。複数のＭＭＸレジスタは、６４ビットパックド整数データで複数の演算を実行するため、同様にＭＭＸおよびＸＭＭレジスタの間で実行される同じ複数の演算に対して複数のオペランドを保持するために用いられる。

【０１４４】

発明の代替的な実施形態は、より広いまたはより狭い複数のレジスタを用いてよい。更に、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用いてよい。

【０１４５】

前述の明細書では、発明は、固有の典型的な複数の実施形態を参照して記載されている。しかし、様々な修正および変更が、添付の特許請求の範囲に記載されたように発明のより広い精神及び範囲から逸脱することなくなされてよいことは明らかであろう。従って、明細書及び図面は、限定の意味ではなく例示の意味として捉えられるべきである。

【０１４６】

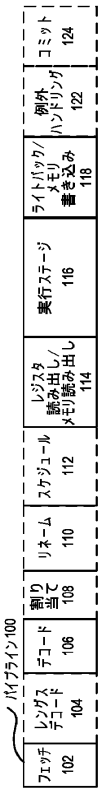
発明の複数の実施形態は、上述した様々なステップを含む。複数のステップは、汎用または専用プロセッサに複数のステップを実行させるために用いられてよい複数の機械実行可能命令において実装されてよい。代替的に、これらのステップは、複数のステップを実行するためのハードワイヤードロジックを含む特定の複数のハードウェアコンポーネントにより、またはプログラムされたコンピュータコンポーネントおよびカスタムハードウェアコンポーネントの任意の組み合わせにより、実行されてよい。

【０１４７】

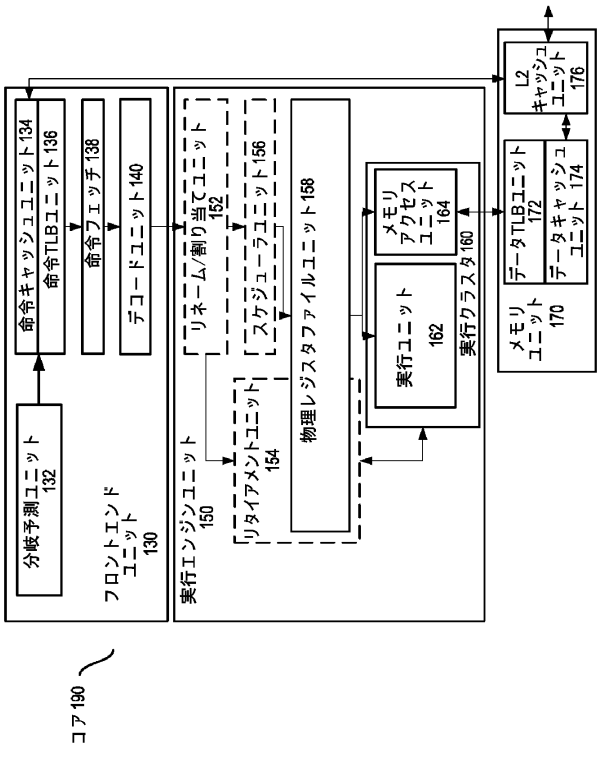
ここに記載されたように、複数の命令は、特定の複数の演算を実行するよう構成された、または所定の機能性または非一時的コンピュータ可読媒体に実装されるメモリに格納された複数のソフトウェア命令を有する特定用途向け集積回路（ＡＳＩＣ）のようなハードウェアの特定の複数の構成を参照してよい。従って、複数の図に示された複数の技術は、１または複数の電子デバイス（例えば、エンドステーション、ネットワーク要素等）上で格納および実行されるコードおよびデータを用いて実装されることができる。そのような電子デバイスは、非一時的コンピュータ機械可読記憶媒体（例えば、磁気ディスク、光ディスク、ランダムアクセスメモリ、リードオンリメモリ、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、デジタル信号などのような伝搬信号の電気、光、音、又は他の形式）のような、コンピュータ機械可読媒体を用いてコードおよびデータを（内部で、および／またはネットワークを介して他の電子デバイスを用いて）格納および通信する。さらに、そのような電子デバイスは、一般的に、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憶媒体）、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のような１または複数の他のコンポーネントと連結された１または複数のプロセッサのセットを含む。複数のプロセッサのセットおよび他の複数のコンポーネントの連結は、一般的に、１または複数のバスおよびブリッジ（バスコントローラとも呼ばれる）を介される。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複数の信号は、それぞれ、１または複数の機械可読記憶媒体および機械可読通信媒体を表す。従って、与えられた電子デバイスのストレージデバイスは、一般的に、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す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、発明の実施形態の１または複数の部分は、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複数の組み合わせを用いて実装されてよい。この発明の詳細な説明を通じて、説明の目的のために、多くの特定の詳細が、本発明の完全な理解を提供するために記載された。しかし、これらの特定の複数の詳細の幾つかが無くても本発明が実施されてよいことは、当業者に明らかである。特定の例において、周知の構造及び機能は、本発明の主題を分かりにくくしないよう精巧に詳細に記載されなかった。従って、発明の範囲および精神は、次の特許請求の範囲の観点において判断されるべきである。

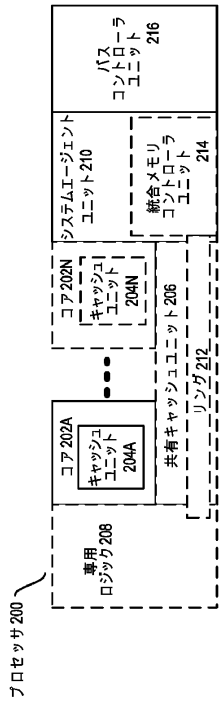
【図１Ａ】



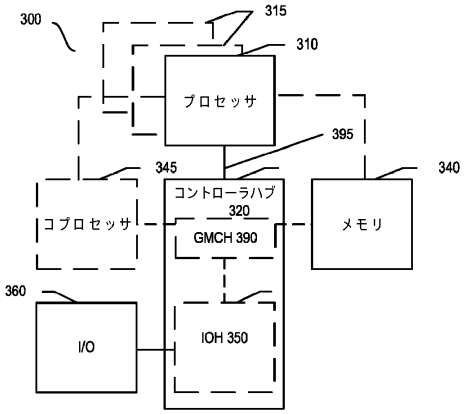
【図１Ｂ】



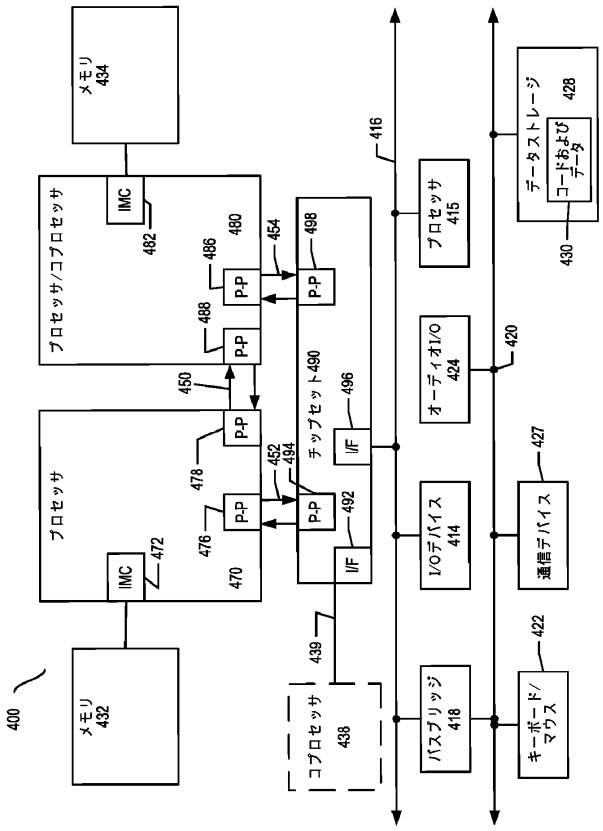
【図２】


プロセッサ200
専用ロジック208
コア202A
キャッシュユニット204A
共有キャッシュユニット206
リング212
コア202N
キャッシュユニット204N
システムエージェントユニット210
統合メモリコントローラユニット214
バスコントローラユニット216

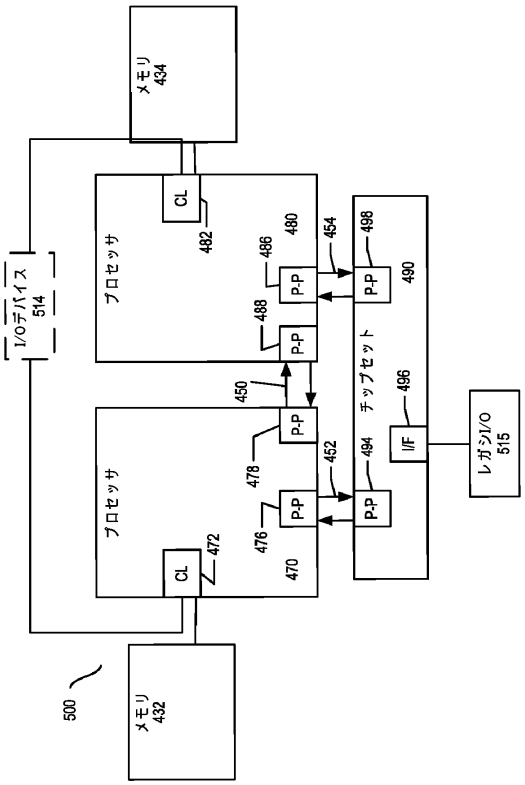
【図３】


300
315
310
プロセッサ
395
345
340
コプロセッサ
コントローラハブ
320
GMCH 390
メモリ
360
I/O
IOH 350

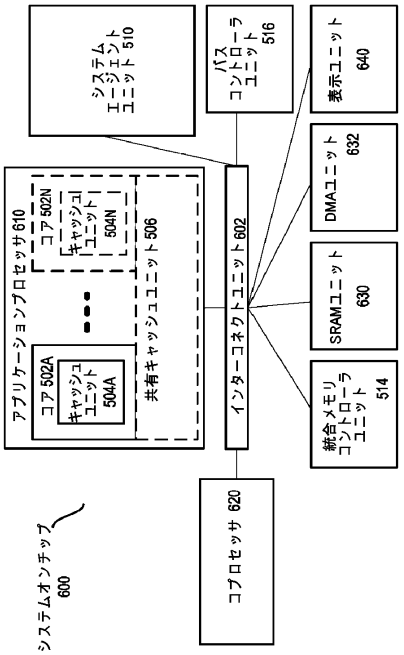
【図４】


400
メモリ
432
メモリ
434
プロセッサ
IMC
472
プロセッサ/コプロセッサ
IMC
482
コプロセッサ
438
P-P
470
476
478
480
486
488
450
452
454
490
494
496
498
I/F
492
439
チップセット
416
プロセッサ
415
オーディオI/O
424
420
I/Oデバイス
414
通信デバイス
427
バスブリッジ
418
422
キーボード/マウス
データストレージ
コードおよびデータ
428
430

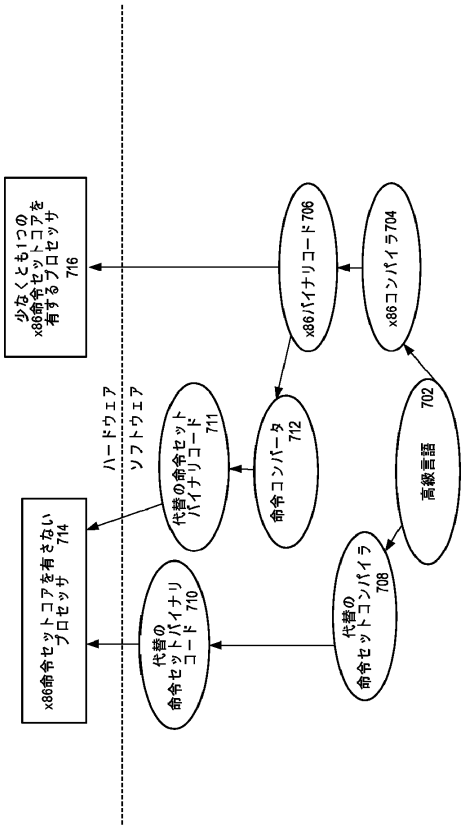
【図５】


500
メモリ
432
メモリ
434
I/Oデバイス
514
プロセッサ
CL
472
プロセッサ
CL
482
P-P
470
476
478
480
486
488
450
452
454
490
494
496
498
チップセット
I/F
レガシI/O
515

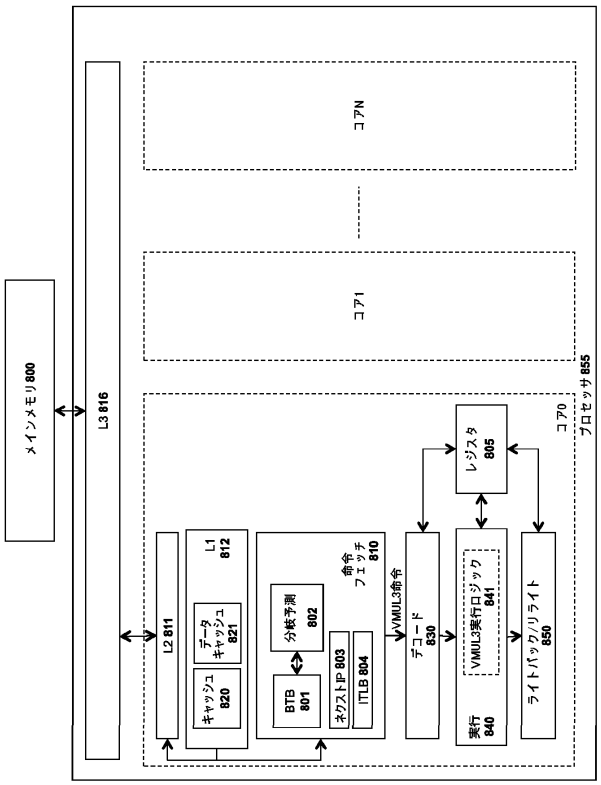
【図６】


システムオンチップ
600
コプロセッサ 620
インターコネクトユニット 602
共有キャッシュユニット 506
アプリケーションプロセッサ 610
コア 502A
キャッシュユニット 504A
コア 502N
キャッシュユニット 504N
システムエージェントユニット 510
バスコントローラユニット 516
表示ユニット 640
DMAユニット 632
SRAMユニット 630
統合メモリコントローラユニット 514

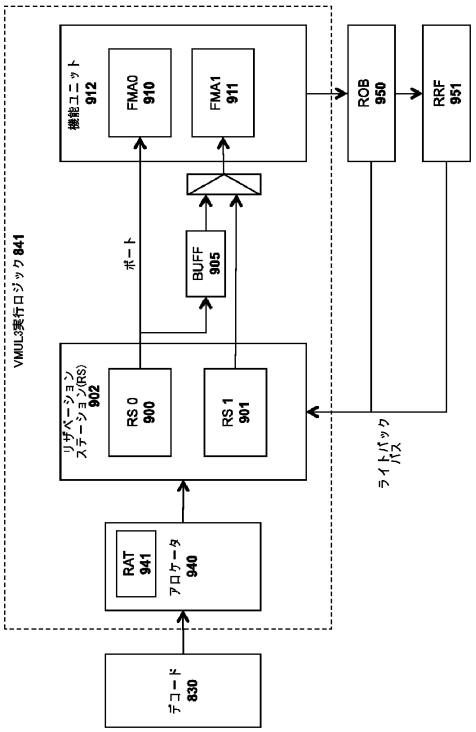
【図７】


ハードウェア
ソフトウェア
少なくとも1つのx86命令セットコアを有するプロセッサ 716
x86命令セットコアを有さないプロセッサ 714
代替の命令セットバイナリコード 710
代替の命令セットバイナリコード 711
命令コンバータ 712
代替の命令セットコンバイラ 708
高級言語 702
x86バイナリコード 706
x86コンパイラ 704

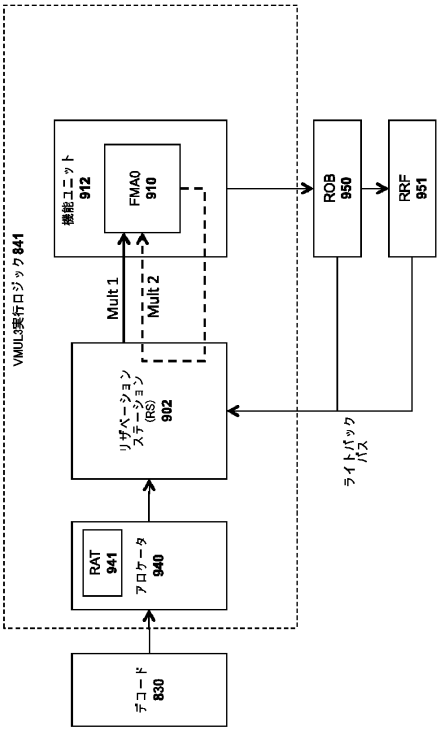
【図８】


メインメモリ 800
L3 816
コア1
コアN
コア0
プロセッサ 855
L2 811
L1 812
データキャッシュ 821
キャッシュ 820
BTB 801
分岐予測 802
ネクストIP 803
ITLB 804
命令フェッチ 810
VMUL3命令
デコード 830
実行 840
VMUL3実行ロジック 841
レジスタ 805
ライトバック/リタイア 850

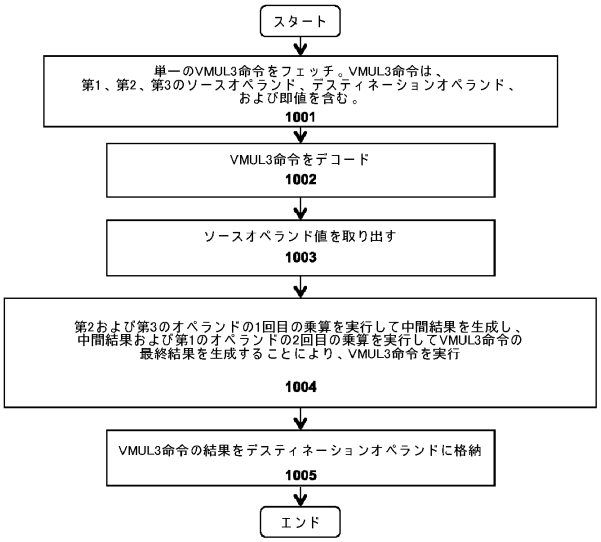
【図９Ａ】


VMUL3実行ロジック 841
デコード 830
RAT 941
アロケータ 940
リザベーションステーション(RS) 902
RS 0 900
RS 1 901
ポート
BUFF 905
機能ユニット 912
FMA0 910
FMA1 911
ROB 950
RRF 951
ライトバックバス

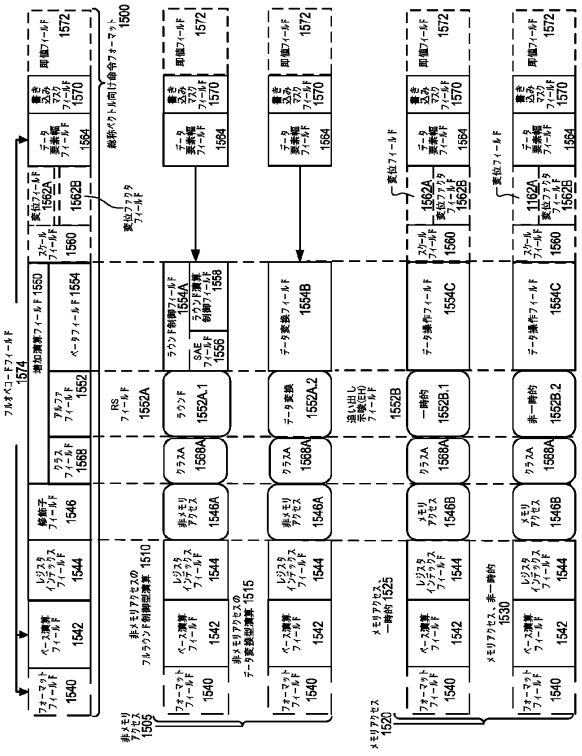
【図９Ｂ】



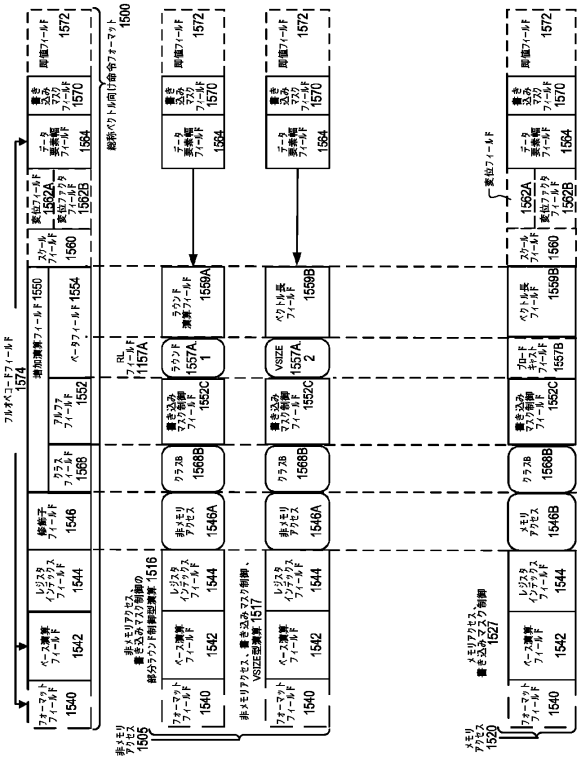
【図１０】



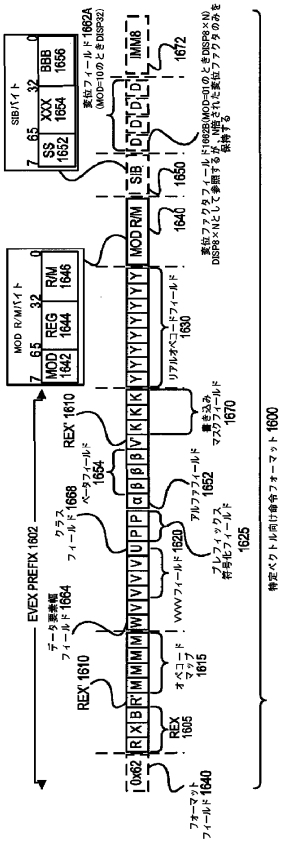
【図１１Ａ】



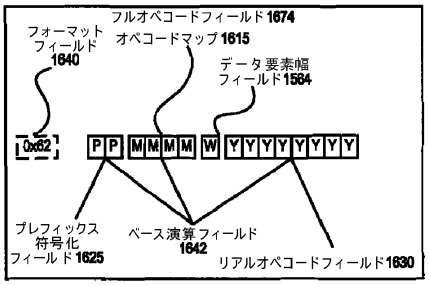
【図１１Ｂ】



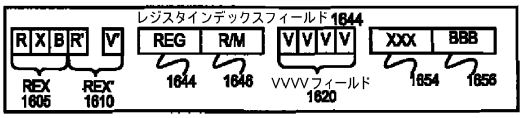
【図１２Ａ】



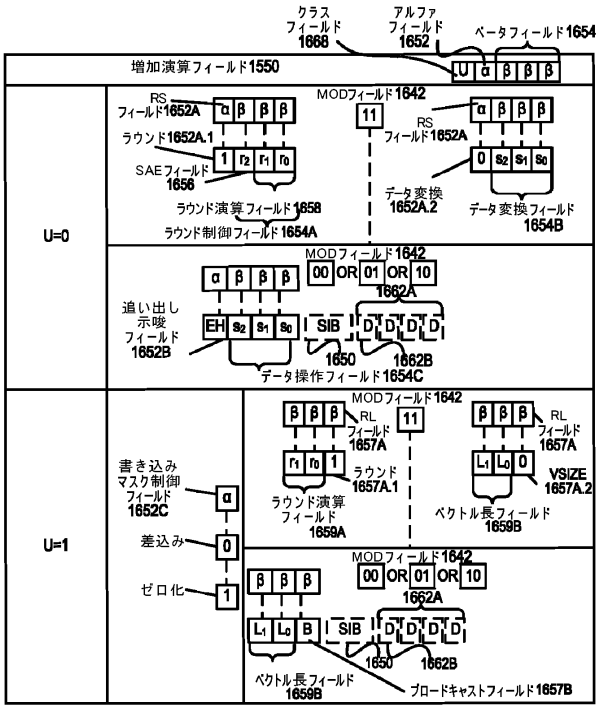
【図１２Ｂ】



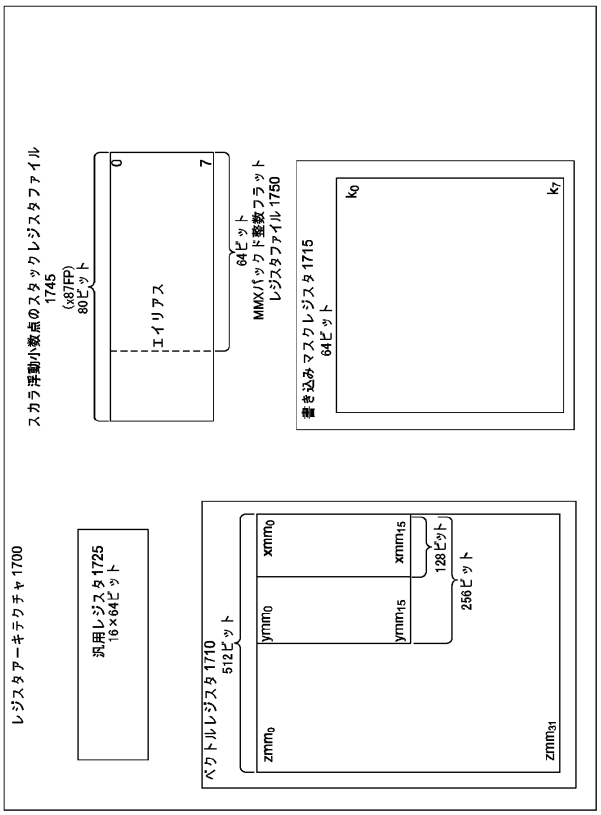
【図１２Ｃ】



【図１２Ｄ】



【図１３】



フロントページの続き

(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　フェルナンデス、マネル
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　田中　幸雄

(56)参考文献　特開２０１１－１３４３０５（ＪＰ，Ａ）
Hideaki KOBAYASHI, A FAST MULTI-OPERAND MULTIPLICATION SCHEME, Proceedings of the IEEE 5th Symposium on Computer Arithmetc, IEEE, １９８１年　5月１６日, Pages 246-250

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　　９／３０５
Ｇ０６Ｆ　　７／４８７
Ｇ０６Ｆ　１７／１０