(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6738579号
(P6738579)
(45)発行日　令和2年8月12日(2020.8.12)　(24)登録日　令和2年7月22日(2020.7.22)

(51)Int.Cl.			F I		
G06F	7/57	(2006.01)	G06F	7/57	202
G06F	9/30	(2018.01)	G06F	9/30	372
G06F	9/38	(2006.01)	G06F	9/30	310A
G06F	9/302	(2006.01)	G06F	9/38	370C
G06F	9/318	(2006.01)	G06F	9/302	A

請求項の数 18　(全 41 頁)　最終頁に続く

(21)出願番号　特願2017-527720 (P2017-527720)
(86)(22)出願日　平成27年11月23日 (2015.11.23)
(65)公表番号　特表2018-507453 (P2018-507453A)
(43)公表日　平成30年3月15日 (2018.3.15)
(86)国際出願番号　PCT/US2015/062056
(87)国際公開番号　WO2016/105754
(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19日 (2018.11.19)
(31)優先権主張番号　14/581,815
(32)優先日　平成26年12月23日 (2014.12.23)
(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　コーベル、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

最終頁に続く

(54)【発明の名称】命令フローを最適化するチェックを実行するための装置および方法

(57)【特許請求の範囲】
【請求項１】
１または複数のソースオペランドを用いて複数の数学的命令を実行するための演算論理装置（ＡＬＵ）と、
現在の数学的命令のための前記１または複数のソースオペランドを評価し、前記評価に基づいて、前記ＡＬＵによって前記現在の数学的命令を実行することを含むデフォルト演算シーケンスを実行するか否か、または特定のタイプのソースオペランドを有する前記数学的命令の結果を、前記デフォルト演算シーケンスより効率的に提供する代替的な演算シーケンスにジャンプするか否かを判断するための命令チェックロジックとを備え、
前記１または複数のソースオペランドの前記評価を実行すると、前記命令チェックロジックは、前記デフォルト演算シーケンスまたは前記代替的なシーケンスを実行するか否かを示すベクトル出力と、前記代替的なシーケンスで処理されるべき要素をシグナリングするマスク出力と、前記１または複数のソースオペランドに対する数学的オペレーションの実行から生じた１または複数の例外を示すべく出力された計算毎の例外フラグとを生成する
プロセッサ。
【請求項２】
前記現在の数学的命令は、前記１または複数のソースオペランドにより指定された分子および分母を有する除算命令を含み、
前記命令チェックロジックは、非正規化オペランドであるか、無限大に等しいか、非数

（ＮａＮ）オペランドであるか、およびゼロによる除算をもたらすかのうちの少なくとも１つである分子または分母のいずれかに応答して、前記代替的な演算シーケンスへのジャンプを生じさせる、請求項１に記載のプロセッサ。

【請求項３】

前記命令チェックロジックは、ゼロ（ＺＥ）による除算、無効な演算（ＩＥ）、および非正規化オペランド（ＤＥ）のうちの少なくとも１つを含む、１または複数の例外フラグを前記プロセッサ内で設定する、請求項２に記載のプロセッサ。

【請求項４】

前記現在の命令は、平方根演算を実行するソースオペランド値を有する平方根命令を含み、

前記命令チェックロジックは、負の数であるか、非正規化オペランドであるか、無限大に等しいか、および非数（ＮａＮ）オペランドであるかのうちの少なくとも１つである前記ソースオペランドに応答して、前記代替的な演算シーケンスへのジャンプを生じさせる、請求項１～３のいずれか１項に記載のプロセッサ。

【請求項５】

前記現在の命令は、平方根演算を実行するソースオペランド値（ｘ）を有する平方根命令を含み、

前記命令チェックロジックは、ｘを用いて実行される比較に応答して前記代替的な演算シーケンスへのジャンプを生じさせる、請求項１～４のいずれか１項に記載のプロセッサ。

【請求項６】

前記デフォルト演算シーケンスは、デフォルトシーケンスの命令またはマイクロオペレーションを含み、

前記代替的な演算シーケンスは、代替的なシーケンスの命令またはマイクロオペレーションを含む、請求項１～５のいずれか１項に記載のプロセッサ。

【請求項７】

現在の数学的命令のための１または複数のソースオペランドを取得する段階と、

前記現在の数学的命令のための前記１または複数のソースオペランドを評価する段階と、

前記評価に基づいて、前記現在の数学的命令を実行することを含むデフォルト演算シーケンスを実行するか否か、または特定のタイプのソースオペランドを有する前記数学的命令の結果を、前記デフォルト演算シーケンスより効率的に提供する代替的な演算シーケンスにジャンプするか否かを判断する段階と、

前記１または複数のソースオペランドの前記評価を実行すると、前記デフォルト演算シーケンスまたは前記代替的なシーケンスを実行するか否かを示すベクトル出力を生成する段階と、

前記代替的なシーケンスで処理されるべき要素をシグナリングするマスク出力を生成する段階と、

前記１または複数のソースオペランドに対する数学的オペレーションの実行から生じた１または複数の例外を示すべく出力された計算毎の例外フラグを生成する段階と、を備える、方法。

【請求項８】

前記現在の数学的命令は、前記１または複数のソースオペランドにより指定された分子および分母を有する除算命令を含み、

前記方法は、非正規化オペランドであるか、無限大に等しいか、非数（ＮａＮ）オペランドであるか、およびゼロによる除算をもたらすかのうちの少なくとも１つである分子または分母のいずれかに応答して、前記代替的な演算シーケンスにジャンプする、請求項７に記載の方法。

【請求項９】

ゼロ（ＺＥ）による除算、無効な演算（ＩＥ）、および非正規化オペランド（ＤＥ）の

うちの少なくとも１つを含む、１または複数の例外フラグを前記方法において設定する段階を更に備える、請求項８に記載の方法。

【請求項１０】

前記現在の命令は、平方根演算を実行するソースオペランド値を有する平方根命令を含み、

前記方法は、負の数であるか、非正規化オペランドであるか、無限大に等しいか、および非数（ＮａＮ）オペランドであるかのうちの少なくとも１つである前記ソースオペランドに応答して、前記代替的な演算シーケンスにジャンプする段階を更に備える、請求項７～９のいずれか１項に記載の方法。

【請求項１１】

前記現在の命令は、平方根演算を実行するソースオペランド値（ｘ）を有する平方根命令を含み、

前記方法は、ｘを用いて実行される比較に応答して前記代替的な演算シーケンスにジャンプする段階を更に備える、請求項７～１０のいずれか１項に記載の方法。

【請求項１２】

前記デフォルト演算シーケンスは、デフォルトシーケンスの命令またはマイクロオペレーションを含み、

前記代替的な演算シーケンスは、代替的なシーケンスの命令またはマイクロオペレーションを含む、請求項７～１１のいずれか１項に記載の方法。

【請求項１３】

数学的命令およびグラフィックス命令を含む命令およびデータを格納するためのメモリと、

前記数学的命令を実行して前記データを処理するための複数のコアと、

前記グラフィックス命令に応答してグラフィックスオペレーションを実行するためのグラフィックスプロセッサユニットと、

ネットワークを介してデータを受信および送信するためのネットワークインタフェースと、

マウスまたはカーソル制御デバイスからユーザ入力を受信するためのインタフェースと、

１または複数のソースオペランドを用いて複数の数学的命令を実行するための演算論理装置（ＡＬＵ）と、

現在の数学的命令のための前記１または複数のソースオペランドを評価し、前記評価に基づいて、前記ＡＬＵにより前記現在の数学的命令を実行することを含むデフォルト演算シーケンスを実行するか否か、または特定のタイプのソースオペランドを有する前記数学的命令についての結果を、前記デフォルト演算シーケンスより効率的に提供する代替的な演算シーケンスにジャンプするか否かを判断する命令チェックロジックとを備え、

前記１または複数のソースオペランドの前記評価を実行すると、前記命令チェックロジックは、前記デフォルト演算シーケンスまたは前記代替的なシーケンスを実行するか否かを示すベクトル出力と、前記代替的なシーケンスで処理されるべき要素をシグナリングするマスク出力と、前記１または複数のソースオペランドに対する数学的オペレーションの実行から生じた１または複数の例外を示すべく出力された計算毎の例外フラグとを生成する

システム。

【請求項１４】

前記現在の数学的命令は、前記１または複数のソースオペランドにより指定された分子および分母を有する除算命令を含み、

前記命令チェックロジックは、非正規化オペランドであるか、無限大に等しいか、非数（ＮａＮ）オペランドであるか、およびゼロによる除算をもたらすかのうちの少なくとも１つである分子または分母のいずれかに応答して、前記代替的な演算シーケンスへのジャンプを生じさせる、請求項１３に記載のシステム。

【請求項１５】
前記命令チェックロジックは、ゼロ（ＺＥ）による除算、無効な演算（ＩＥ）、および非正規化オペランド（ＤＥ）のうちの少なくとも１つを含む、１または複数の例外フラグをプロセッサ内で設定する、請求項１４に記載のシステム。
【請求項１６】
前記現在の命令は、平方根演算を実行するソースオペランド値を有する平方根命令を含み、
前記命令チェックロジックは、負の数であるか、非正規化オペランドであるか、無限大に等しいか、および非数（ＮａＮ）オペランドであるかのうちの少なくとも１つである前記ソースオペランドに応答して、前記代替的な演算シーケンスへのジャンプを生じさせる、請求項１３～１５のいずれか１項に記載のシステム。
【請求項１７】
前記現在の命令は、平方根演算を実行するソースオペランド値（ｘ）を有する平方根命令を含み、
前記命令チェックロジックは、ｘを用いて実行される比較に応答して前記代替的な演算シーケンスへのジャンプを生じさせる、請求項１３～１６のいずれか１項に記載のシステム。
【請求項１８】
前記命令チェックロジックは、マイクロオペレーションを実行するためのものである
請求項１～６のいずれか１項に記載のプロセッサまたは請求項１３～１７のいずれか１項に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、概ね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、本発明は、命令フローを最適化するチェック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、ネイティブなデータタイプ、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、ならびに外部入出力（Ｉ／Ｏ）を含むプログラミングに関連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、一般に、マクロ命令をデコードするプロセッサデコーダの結果であるマイクロ命令またはマイクロオプとは異なり、実行のためにプロセッサに提供される命令であるマクロ命令を指すことに留意されたい。マイクロ命令またはマイクロオプは、プロセッサ上の実行ユニットにマクロ命令に関連するロジックを実装するためのオペレーションの実行を命令するように構成され得る。
【０００３】
ＩＳＡは、命令セットを実装するべく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とは区別される。異なるマイクロアーキテクチャを有するプロセッサは、共通の命令セットを共有し得る。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）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にあるＡｄｖａｎｃｅｄ Ｍｉｃｒｏ Ｄｅｖｉｃｅｓ，Ｉｎｃのプロセッサは、ほぼ同一のバージョンの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伴う）を実装するが、異なる内部設計を有する。例えば、ＩＳＡの同一のレジスタアーキテクチャは、専用物理レジスタ、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）、およびリタイアメントレジスタファイルの使用）等を用いる１または複数の動的に割り当てられる物理レジスタを含む周知技術を用いて、異なる態様で異なるマイクロアーキテクチャに実装され得る。別途指定されない限り、レジスタアーキテクチャ、レジスタファイ

ル、およびレジスタという文言は、本明細書においてソフトウェア／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態様に言及するために用いられる。区別が必要とされる場合、「論理的」、「アーキテクチャの」、または「可視のソフトウェア」という形容詞は、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が、異なる形容詞が所与のマイクロアーキテクチャ（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）におけるレジスタを指すために用いられる。

【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。所与の命令フォーマットは、とりわけ、実行されるオペレーションおよび当該オペレーションが実行されるオペランドを指定する様々なフィールド（ビットの数、ビットの位置）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）の定義により更に分類される。例えば、所与の命令フォーマットの命令テンプレートは、異なるサブセットの命令フォーマットのフィールドを有するものと定義され（含まれるフィールドは、通常は同じ順序であるが、少なくともいくつかは、より少ないフィールドが含まれているので、異なるビット位置を有する）、および／または異なる解釈をされる所与のフィールドを有するものと定義され得る。所与の命令は、所与の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、当該命令フォーマットの命令テンプレートのうちの所与の１つで）表され、オペレーションおよびオペランドを指定する。命令ストリームは、命令の特定のシーケンスであり、シーケンスにおける各命令は、命令フォーマットにおける命令の発生である（定義される場合、当該命令フォーマットの命令テンプレートのうちの所与の１つ）。

【図面の簡単な説明】

【０００５】

本発明のより良い理解は、以下の図面と併せて以下の詳細な説明から得られ得る。

【０００６】

【図１Ａ】本発明の実施形態による一般的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一般的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。

【００１０】

【図４Ｂ】本発明の実施形態による、プロセッサに含まれるべきインオーダフェッチ、デコード、リタイアコアの例示的な実施形態、および例示的なレジスタリネーム、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。

【００１１】

【図５Ａ】オンダイ相互接続ネットワークとの接続を伴うシングル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スを用いるシングル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、ソース命令セットのバイナリ命令を、ターゲット命令セットの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と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得る例示的なプロセッサを示す。
【００２０】
【図１３】命令の入力をチェックし、これに応じて次の命令シーケンスを決定するのに使用できる出力のセットを生成するための命令チェックロジックを示す。
【００２１】
【図１４】異なるタイプの入力値に対する除算チェック命令に対して生成される出力を示す。
【００２２】
【図１５】異なるタイプの入力値のための平方根チェック命令に対して生成される出力を示す。
【００２３】
【図１６】チェックオペレーションを実行し、第１の命令シーケンスまたは第２の命令シーケンスを選択するための方法の一実施形態を示す。
【発明を実施するための形態】
【００２４】
以下の説明において、説明の目的のために、下記の本発明の実施形態の完全な理解を提供するべく、多数の具体的な詳細が記載される。しかし、当業者には、本発明の実施形態がこれらの具体的な詳細のいくつかを用いることなく実施され得ることが明らかであろう。本発明の実施形態の基礎となる原理を不明瞭にするのを避けるべく、他の例において、周知の構造およびデバイスがブロック図の形態で示される。
【００２５】
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
命令セットは、１または複数の命令フォーマットを含む。所与の命令フォーマットは、とりわけ、実行されるオペレーション（オペコード）および当該オペレーションが実行されるオペランドを指定する様々なフィールド（ビットの数、ビットの位置）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）の定義により更に分類される。例えば、所与の命令フォーマットの命令テンプレートは、異なるサブセットの命令フォーマットのフィールド（含まれるフィールドは、通常は同じ順序であるが、少なくともいくつかは、より少ないフィールドが含まれているので、異なるビット位置を有する）を有するものと定義され、および／または異なる解釈をされる所与のフィールドを有するものと定義され得る。従って、ＩＳＡの各命令は、所与の命令フォーマッ

トを用いて（および定義される場合には、当該命令フォーマットの命令テンプレートのうちの所与の１つで）表され、オペレーションおよびオペランドを指定するためのフィールドを含む。例えば、例示的なＡＤＤ命令は、特定のオペコード、ならびに当該オペコードを指定する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。命令ストリームにおけるこのＡＤＤ命令が生じることにより、特定のオペランドを選択するオペランドフィールドに特定の内容を有する。Ａｄｖａｎｃｅｄ Ｖｅｃｔｏｒ Ｅｘｔｅｎｓｉｏｎｓ（ＡＶＸ）（ＡＶＸ１およびＡＶＸ２）と呼ばれ、ベクトル拡張（ＶＥＸ）符号化スキームを用いるＳＩＭＤ拡張のセットが、リリースおよび／または公開されている（例えば、Ｉｎｔｅｌ（登録商標）６４ ａｎｄ ＩＡ－３２ Ａｒｃｈｉｔｅｃｔｕｒｅｓ Ｓｏｆｔｗａｒｅ Ｄｅｖｅｌｏｐｅｒｓ Ｍａｎｕａｌ，Ｏｃｔｏｂｅｒ ２０１１およびＩｎｔｅｌ（登録商標）Ａｄｖａｎｃｅｄ Ｖｅｃｔｏｒ Ｅｘｔｅｎｓｉｏｎｓ Ｐｒｏｇｒａｍｍｉｎｇ Ｒｅｆｅｒｅｎｃｅ，Ｊｕｎｅ ２０１１を参照されたい）。

【００２６】

例示的な命令フォーマット

本明細書に説明される命令の実施形態は、異なるフォーマットで実施され得る。更に、例示的なシステム、アーキテクチャ、およびパイプラインが以下に詳述される。命令の実施形態は、そのようなシステム、アーキテクチャ、およびパイプライン上で実行され得るが、詳述されるものに限定されない。

【００２７】

Ａ．一般的ベクトル向け命令フォーマット

ベクトル向け命令フォーマットは、ベクトル命令に好適な命令フォーマットである。（例えば、ベクトルオペレーションに固有の一定のフィールドが存在する）。ベクトルおよびスカラオペレーションの両方がベクトル向け命令フォーマットによりサポートされる実施形態が説明されるが、代替的な実施形態は、ベクトル向け命令フォーマットによるベクトルオペレーションのみを用いる。

【００２８】

図１Ａ～図１Ｂは、本発明の実施形態による一般的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、一般的ベクトル向け命令フォーマット、およびそのクラスＡ命令テンプレートを示すブロック図である。図１Ｂは、本発明の実施形態による、一般的ベクトル向け命令フォーマット、およびそのクラスＢ命令テンプレートを示すブロック図である。具体的には、クラスＡおよびクラスＢ命令テンプレートは、一般的ベクトル向け命令フォーマット１００に対して定義され、これらの両方は、非メモリアクセス１０５の命令テンプレートおよびメモリアクセス１２０の命令テンプレートを含む。ベクトル向け命令フォーマットの文脈における一般的という用語は、いずれの特定の命令セットにも関係しない命令フォーマットを指す。

【００２９】

ベクトル向け命令フォーマットが、３２ビット（４バイト）または６４ビット（８バイト）のデータ要素幅（またはサイズ）を有する６４バイトのベクトルオペランド長（またはサイズ）（従って、６４バイトのベクトルは、１６個のダブルワードサイズの要素、または代替的に８クワッドワードサイズの要素のいずれかからなる）、１６ビット（２バイト）または８ビット（１バイト）のデータ要素幅（またはサイズ）を有する６４バイトの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）のデータ要素幅（またはサイズ）を有する３２バイトの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）のデータ要素幅（またはサイズ）を有する１６バイトのベクトルオペランド長（またはサイズ）をサポートする本発明の実施形態が説明されるが、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）のデータ要素幅）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バイトのベクトルオペランド）をサポートし得る。

【００３０】

図１Ａにおけるクラス命令テンプレートは、１）非メモリアクセス１０５の命令テンプレート中に示される非メモリアクセス、フルラウンド制御タイプオペレーション１１０の命令テンプレート、および非メモリアクセス、データ変換タイプオペレーション１１５の命令テンプレート、ならびに２）メモリアクセス１２０の命令テンプレート中に示されるメモリアクセス、一時的１２５の命令テンプレート、およびメモリアクセス、非一時的１３０の命令テンプレートを含む。図１Ｂのクラス命令テンプレートは、１）非メモリアクセス１０５の命令テンプレート中に示される非メモリアクセス、ライトマスク制御、部分的ラウンド制御タイプオペレーション１１２の命令テンプレート、および非メモリアクセス、ライトマスク制御、ＶＳＩＺＥタイプオペレーション１１７の命令テンプレート、ならびに２）メモリアクセス１２０の命令テンプレート中に示されるメモリアクセス、ライトマスク制御１２７の命令テンプレートを含む。

【００３１】

一般的ベクトル向け命令フォーマット１００は、図１Ａ～図１Ｂにおいて示される順序で以下に列挙される、次のフィールドを含む。

【００３２】

フォーマットフィールド１４０。このフィールドにおける特定の値（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、従って、命令ストリーム中のベクトル向け命令フォーマットにおける命令の発生を一意に識別する。従って、このフィールドは、一般的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で任意選択である。

【００３３】

ベースオペレーションフィールド１４２。その内容は、異なるベースオペレーションを区別する。

【００３４】

レジスタインデックスフィールド１４４。その内容は、レジスタ内であれ、メモリ内であれ、直接に、またはアドレス生成により、ソースオペランドおよびデスティネーションオペランドの位置を指定する。これらは、ＰｘＱ（例えば、３２ｘ５１２、１６ｘ１２８、３２ｘ１０２４、６４ｘ１０２４）のレジスタファイルからＮ個のレジスタを選択するのに十分な数のビットを含む。一実施形態においては、Ｎは、最大で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であり得るが、代替的な実施形態は、より多いか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得る（例えば、最大で２つのソースをサポートし得、この場合、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機能し、最大で３つのソースをサポートし得、この場合、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機能し、最大で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得る）。

【００３５】

修飾子フィールド１４６。その内容は、一般的ベクトル命令フォーマット中におけるメモリアクセスを指定する命令の発生とメモリアクセスを指定しない命令の発生とを、すなわち、非メモリアクセス１０５の命令テンプレートとメモリアクセス１２０の命令テンプレートを区別する。メモリアクセスオペレーションは、メモリ階層を読み出し、および／またはこれに書き込むが（いくつかの場合には、レジスタにおける値を用いて、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）、非メモリアクセスオペレーションは、これを行わない（例えば、ソースおよびデスティネーションは、レジスタである）。また、一実施形態において、このフィールドは、３つの異なる態様を選択し、メモリアドレス計算を実行するが、代替的な実施形態は、より多い、より少ない、または異なる態様をサポートし、メモリアドレス計算を実行し得る。

【００３６】
追加オペレーションフィールド１５０。その内容は、ベースオペレーションに加えて、種々様々な異なるオペレーションのうちのどれが実行されるかを区別する。このフィールドは、コンテキストに固有である。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。追加オペレーションフィールド１５０は、オペレーションの共通グループが２、３、または４つの命令ではなく、単一の命令で実行されることを可能にする。
【００３７】
スケールフィールド１６０。その内容は、メモリアドレス生成のための（例えば、２^{scale}＊ｉｎｄｅｘ＋ｂａｓｅを用いるアドレス生成のための）インデックスフィールド内容のスケーリングを可能にする。
【００３８】
変位フィールド１６２Ａ。その内容は、メモリアドレス生成の一部として（例えば、２^{scale}＊ｉｎｄｅｘ＋ｂａｓｅ＋ｄｉｓｐｌａｃｅｍｅｎｔを用いるアドレス生成に）用いられる。
【００３９】
変位係数フィールド１６２Ｂ（変位係数フィールド１６２Ｂの直ぐ上に変位フィールド１６２Ａを並置することにより、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）。その内容は、アドレス生成の一部として用いられる。変位係数フィールド１６２Ｂは、メモリアクセス（Ｎ）のサイズに対して調整される変位係数を指定する。Ｎは、（例えば、２^{scale}＊ｉｎｄｅｘ＋ｂａｓｅ＋ｓｃａｌｅｄ ｄｉｓｐｌａｃｅｍｅｎｔを用いるアドレス生成のための）メモリアクセスにおけるバイトの数である。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、有効なアドレスを計算するときに使用される最終的変位を生成するべく、メモリオペランドの合計サイズ（Ｎ）で乗算される。Ｎの値は、フルオペコードフィールド１７４（本明細書において後述される）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り決定される。変位フィールド１６２Ａおよび変位係数フィールド１６２Ｂは、非メモリアクセス１０５の命令テンプレートに用いられず、および／または異なる実施形態が１つのみまたは２つのうちいずれも実装しない場合があるという意味で任意選択である。
【００４０】
データ要素幅フィールド１６４。その内容は、（いくつかの実施形態において全ての命令に対して、他の実施形態において命令のうちいくつかのみに対して）いくつかのデータ要素幅のうちのどれが用いられるかを区別する。このフィールドは、１つのデータ要素幅のみがサポートされ、および／またはオペコードのいくつかの態様を用いて、データ要素幅がサポートされる場合には必要とされないという意味で任意選択である。
【００４１】
ライトマスクフィールド１７０。その内容は、データ要素位置ベースで、デスティネーションベクトルオペランドにおける当該データ要素位置がベースオペレーションおよび追加オペレーションの結果を反映するか否かを制御する。クラスＡ命令テンプレートは、マージング・ライトマスキングをサポートするが、クラスＢ命令テンプレートは、マージングおよびゼロ書き込みマスキングの両方をサポートする。マージングする場合、ベクトルマスクは、デスティネーションにおける要素の任意のセットが（ベースオペレーションおよび追加オペレーションにより指定された）任意のオペレーションの実行中に更新から保護されることを可能にする。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持する。対照的に、ゼロ書き込みする場合、ベクトルマスクは、デスティネーションにおける要素の任意のセットが（ベースオペレーションおよび追加オペレーションにより指定された）任意のオペレーションの実行中にゼロにされることを可能にする。一実施形態において、対応するマスクビットが０の値を有する場合、デスティネーションの要素は、０に設定される。この機能のサブセットは、実行されるオペレーションのベクトル長を制御する能力である（すなわち、要素のスパンは、最

初のものから最後のものに変更される）。しかし、変更される要素が連続している必要はない。従って、ライトマスクフィールド１７０は、ロード、ストア、演算、論理等を含む部分的ベクトルオペレーションを可能にする。本発明の実施形態は、ライトマスクフィールド１７０の内容が、用いられるべきライトマスクを含むいくつかのライトマスクレジスタのうちの１つを選択する（従って、ライトマスクフィールド１７０の内容が実行されるべき当該マスキングを間接的に識別する）ものとして説明されているが、代替的な実施形態はこれに代えて、または更に、マスクライトフィールド１７０の内容が実行されるべきマスキングを直接に指定することを可能にする。

【００４２】

即値フィールド１７２。その内容は、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一般的ベクトル向けフォーマットの実装において存在せず、即値を用いない命令中に存在しないという意味で任意選択である。

【００４３】

クラスフィールド１６８。その内容は、命令の異なるクラスを区別する。図１Ａ～図１Ｂを参照すると、このフィールドの内容は、クラスＡ命令またはクラスＢ命令を選択する。図１Ａ～図１Ｂにおいて、角が丸い四角形は、特定の値がフィールド内に存在することを示すべく用いられる（例えば、図１Ａおよび図１Ｂにおけるクラスフィールド１６８のクラスＡ１６８Ａおよびクラス Ｂ１６８Ｂの各々）。

【００４４】

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１０５の命令テンプレートの場合に、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、その内容は、異なる追加オペレーションタイプのうちのどれが実行されるかを区別するが（例えば、ラウンド１５２Ａ．１およびデータ変換１５２Ａ．２は、各々、非メモリアクセス、ラウンドタイプオペレーション１１０、および非メモリアクセス、データ変換タイプオペレーション１１５の命令テンプレートに対して指定される）、ベータフィールド１５４は、指定されたタイプのオペレーションのうちいずれが実行されるかを区別する。非メモリアクセス１０５の命令テンプレートにおいて、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。

【００４５】

非メモリアクセス命令テンプレート‐フルラウンド制御タイプオペレーション　非メモリアクセスのフルラウンド制御タイプオペレーション１１０の命令テンプレートにおいて、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、その内容は、静的ラウンドを提供する。本発明の説明される実施形態において、ラウンド制御フィールド１５４Ａは、抑圧全浮動小数点例外（ＳＡＥ）フィールド１５６およびラウンドオペレーション制御フィールド１５８を含み、代替的な実施形態は、これら両方のコンセプトをサポートおよびエンコードして同一のフィールドとすることができ、またはこれらのコンセプト／フィールドのうち一方または他方のみを有し得る（例えば、ラウンドオペレーション制御フィールド１５８のみを有し得る）。

【００４６】

ＳＡＥフィールド１５６。その内容は、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、抑圧が有効にされたことを示す場合、所与の命令は、いずれの種類の浮動小数点例外フラグも報告せず、いずれの浮動小数点例外ハンドラも立ち上げない。

【００４７】

ラウンドオペレーション制御フィールド１５８。その内容は、ラウンドオペレーション（例えば、ラウンドアップ、ラウンドダウン、ゼロへのラウンド、および近似値へのラウンド）のグループのうちのどれが実行されるべきかを区別する。このように、ラウンドオペレーション制御フィールド１５８は、命令ベース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制御レジスタを含む本発明の一実施形態におい

て、ラウンドオペレーション制御フィールド１５０の内容は、当該レジスタの値を上書きする。

【００４８】

非メモリアクセス命令テンプレート‐データ変換タイプオペレーション　非メモリアクセスのデータ変換タイプオペレーション１１５の命令テンプレートにおいて、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、その内容は、いくつかのデータ変換のうちのどれが実行されるかを区別する（例えば、非データ変換、スウィズル、ブロードキャスト）。

【００４９】

クラスＡのメモリアクセス１２０の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、その内容は、エビクションヒントのうちのどれが用いられるかを区別するが（図１Ａにおいて、一時的１５２Ｂ．１および非一時的１５２Ｂ．２は、各々、メモリアクセス、一時的１２５の命令テンプレート、およびメモリアクセス、非一時的１３０の命令テンプレートに対して指定される）、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、その内容は、いくつかのデータ操作オペレーション（プリミティブとしても知られる）のうちのどれが実行されるかを区別する（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の命令テンプレートは、スケールフィールド１６０を含み、変位フィールド１６２Ａまたは変位スケールフィールド１６２Ｂを任意選択で含む。

【００５０】

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のように、ベクトルメモリ命令は、データ要素の様式で、メモリから／にデータを転送し、実際に転送される要素は、ライトマスクとして選択されるベクトルマスクの内容により規定される。

【００５１】

メモリアクセス命令テンプレート‐一時的　一時的データは、キャッシュから利益を得るのに十分なほど速やかに再利用される可能性が高いデータである。しかし、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、異なる態様で一時的データを実装してもよい。

【００５２】

メモリアクセス命令テンプレート‐非一時的　非一時的データは、レベル１のキャッシュにおけるキャッシュから利益を得るのに十分なほど速やかに再利用される可能性が低いデータであり、追い出しの優先権を与えられるべきである。しかし、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、異なる態様で非一時的データを実装してもよい。

【００５３】

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１５２は、ライトマスク制御（Ｚ）フィールド１５２Ｃとして解釈され、その内容は、ライトマスクフィールド１７０により制御されるライトマスキングがマージングであるべきか、またはゼロ書き込みであるべきかを区別する。

【００５４】

クラスＢの非メモリアクセス１０５の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、その内容は、異なる追加オペレーションタイプのどれが実行されるかを区別するが（例えば、ラウンド１５７Ａ．１およびベクトル長（ＶＳＩＺＥ）１５７Ａ．２は各々、非メモリアクセス、ライトマスク制御、部分的ラウンド制御タイプオペレーション１１２の命令テンプレート、および非メモリアクセス、ライトマスク制御、ＶＳＩＺＥタイプオペレーション１１７の命令テンプレートに対して指定される）、ベータフィールド１５４の残りは、指定されたタイプのどのオペレーションが実行されるかを区別する。非メモリアクセス１０５の命令テンプレートにお

いて、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。

【００５５】

非メモリアクセス、ライトマスク制御、部分的ラウンド制御タイプオペレーション１１０の命令テンプレートにおいて、ベータフィールド１５４の残りは、ラウンドオペレーションフィールド１５９Ａとして解釈され、例外イベント報告は、無効にされる（所与の命令は、いずれの種類の浮動小数点例外フラグも報告せず、いずれの浮動小数点例外ハンドラも立ち上げない）。

【００５６】

ラウンドオペレーション制御フィールド１５９Ａ。ちょうどラウンドオペレーション制御フィールド１５８のように、その内容は、ラウンドオペレーション（例えば、ラウンドアップ、ラウンドダウン、ゼロへのラウンド、および近似値へのラウンド）のグループのうちのどれが実行されるべきかを区別する。このように、ラウンドオペレーション制御フィールド１５９Ａは、命令ベース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制御レジスタを含む本発明の一実施形態において、ラウンドオペレーション制御フィールド１５０の内容は、そのレジスタの値を上書きする。

【００５７】

非メモリアクセス、ライトマスク制御、ＶＳＩＺＥタイプオペレーション１１７の命令テンプレートにおいて、ベータフィールド１５４の残りは、ベクトル長フィールド１５９Ｂとして解釈され、その内容は、いくつかのデータベクトル長のうちのどれが実行されるかを区別する（例えば、１２８、２５６、または５１２バイト）。

【００５８】

クラスＢのメモリアクセス１２０の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、その内容は、ブロードキャストタイプのデータ操作オペレーションが実行されるべきか否かを区別するが、ベータフィールド１５４の残りは、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の命令テンプレートは、スケールフィールド１６０を含み、変位フィールド１６２Ａまたは変位スケールフィールド１６２Ｂを任意選択で含む。

【００５９】

一般的ベクトル向け命令フォーマット１００に関連して、フォーマットフィールド１４０、ベースオペレーションフィールド１４２、およびデータ要素幅フィールド１６４を含む、フルオペコードフィールド１７４が示される。一実施形態として、フルオペコードフィールド１７４がこれらのフィールドの全てを含むものが示されているが、これら全てをサポートしない実施形態において、フルオペコードフィールド１７４は、これらのフィールド全てより少ないものを含む。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。

【００６０】

追加オペレーションフィールド１５０、データ要素幅フィールド１６４、およびライトマスクフィールド１７０は、一般的ベクトル向け命令フォーマットにおける命令ベースで、これらの機能が指定されることを可能にする。

【００６１】

ライトマスクフィールドおよびデータ要素幅フィールドを組み合わせることで、マスクが異なるデータ要素幅に基づいて適用されることを可能にするように、型付き命令を生成する。

【００６２】

クラスＡおよびクラスＢ内に見出され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、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、または両方のクラスをサポートし得る。例えば、汎用演算用の高性能汎用アウトオブオーダコアは、クラスＢのみをサポートし得、主にグラフィックスおよび／またはサイエンティフィック（スループット）演算用

のコアは、クラスＡのみをサポートし得、両方用のコアは、両方をサポートし得る（勿論、両方のクラスの全てのテンプレートおよび命令ではないが、両方のクラスのテンプレートおよび命令のいくつかのミックスを有するコアは、本発明の範囲内である）。また、シングルプロセッサは、複数のコアを含み得、これらの全てが同じクラスをサポートし、または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スコアおよび汎用コアを有するプロセッサにおいて、主にグラフィックスおよび／またはサイエンティフィック演算用のグラフィックスコアのうちの１つは、クラスＡのみをサポートし得るが、汎用コアのうちの１または複数は、クラスＢのみをサポートする汎用演算用のアウトオブオーダ実行およびレジスタリネームを用いる高性能汎用コアであり得る。別個のグラフィックスコアを有し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、もう１つの汎用インオーダまたはオウトオブオーダのコアを含み得る。勿論、本発明の異なる実施形態において、あるクラスの機能は、他のクラスにおいても実装され得る。高水準言語で書かれたプログラムは、１）実行のためにターゲットプロセッサによりサポートされるクラスの命令のみを有する形式、または２）全てのクラスの命令の異なる組み合わせを用いて書かれた代替的なルーチンを有し、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命令に基づいて、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を含む、種々様々な実行可能な形式にされる（例えば、実行時コンパイルまたは静的コンパイル）。

【００６３】

Ｂ．例示的な特定ベクトル向け命令フォーマット

図２Ａ～図２Ｄ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２Ａ～図２Ｄは、フィールドの位置、サイズ、解釈、および順序、ならびにそれらのフィールドのいくつかに対する値を指定するという意味で具体的な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、ｘ８６命令セットを拡張するために用いられ得、従ってフィールドのうちのいくつかは、既存のｘ８６命令セットおよびその拡張（例えば、ＡＶＸ）において用いられるものと類似するか、または同じである。このフォーマットは、拡張された既存のｘ８６命令セットのプレフィックスエンコードフィールド、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の整合性を保つ。図２Ａ～図２Ｄがフィールドにマッピングされる図１Ａ～図１Ｂのフィールドが示される。

【００６４】

本発明の実施形態は、例示的目的で一般的ベクトル向け命令フォーマット１００の文脈において、特定ベクトル向け命令フォーマット２００を参照して説明されるが、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に限定されないことを理解されたい。例えば、一般的ベクトル向け命令フォーマット１００は、様々なフィールドのために考えられる様々なサイズを企図するが、特定ベクトル向け命令フォーマット２００は、特定サイズのフィールドを有するものとして示される。具体的な例として、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における１つのビットフィールドとして示されるが、本発明は、そのようには限定されない（すなわち、一般的ベクトル向け命令フォーマット１００は、他のサイズのデータ要素幅フィールド１６４を企図する）。

【００６５】

一般的ベクトル向け命令フォーマット１００は、図２Ａにおいて示される順序で以下に列挙される、次のフィールドを含む。

【００６６】

ＥＶＥＸプレフィックス（バイト０～３）２０２。４バイト形式でエンコードされる。

【００６７】

フォーマットフィールド１４０（ＥＶＥＸバイト０、ビット［７：０］）。最初のバイト（ＥＶＥＸバイト０）は、フォーマットフィールド１４０であり、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べく用いられる一意な値）。

【００６８】

第２～第４のバイト（ＥＶＥＸバイト１～３）は、特定の能力を提供するいくつかのビットフィールドを含む。

【００６９】

ＲＥＸフィールド２０５（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ＥＶＥＸバイト１、ビット［５］－Ｂ）からなる。ＥＶＥＸ．Ｒ、ＥＶＥＸ．Ｘ、およびＥＶＥＸ．Ｂの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、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、当技術分野で既知のレジスタインデックスの下位の３つのビット（ｒｒｒ、ｘｘｘ、およびｂｂｂ）をエンコードし、従って、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得る。

【００７０】

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、拡張された３２個のレジスタセットの上位の１６個または下位の１６個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。本発明の一実施形態において、このビットは、以下に示される他のものと共に、（周知のｘ８６の３２ビットモードで）ＢＯＵＮＤ命令から区別するべく、ビット反転フォーマットで格納される。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（下記）にはＭＯＤフィールドにおける１１の値を受け付けない。本発明の代替的な実施形態は、これおよび以下に示される他のビットを反転フォーマットで格納しない。１の値は、下位の１６個のレジスタをエンコードするべく用いられる。換言すると、Ｒ'Ｒｒｒｒは、ＥＶＥＸ．Ｒ'、ＥＶＥＸ．Ｒ、および他のフィールドの他のＲＲＲを組み合わせることにより形成される。

【００７１】

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）。その内容は、暗黙に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。

【００７２】

データ要素幅フィールド１６４（ＥＶＥＸバイト２、ビット［７］－Ｗ）は、ＥＶＥＸ．Ｗという表記により表される。ＥＶＥＸ．Ｗは、データタイプの粒度（サイズ）（３２ビットのデータ要素または６４ビットのデータ要素のいずれか）を定義するべく用いられる。

【００７３】

ＥＶＥＸ．ｖｖｖｖ２２０（ＥＶＥＸバイト２、ビット［６：３］－ｖｖｖｖ）。ＥＶＥＸ．ｖｖｖｖの役割は、以下を含み得る。１）ＥＶＥＸ．ｖｖｖｖは、反転（１の補数）形式で指定された第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用いる命令に対して有効である。２）ＥＶＥＸ．ｖｖｖｖは、一定の複数のベクトルシフトに対して１の補数形式で指定された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、フィールドは予約され、１１１１ｂを含むはずである。従って、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、指定子サイズを３２個のレジスタに拡張するべく、追加の異なるＥＶＥＸビットフィールドが用いられる。

【００７４】

ＥＶＥＸ．Ｕ１６８クラスフィールド（ＥＶＥＸバイト２、ビット［２］－Ｕ）。ＥＶ

ＥＸ．Ｕ＝０である場合、クラスＡまたはＥＶＥＸ．Ｕ０を示す。ＥＶＥＸ．Ｕ＝１である場合、クラスＢまたはＥＶＥＸ．Ｕ１を示す。

【００７５】

プレフィックスエンコードフィールド２２５（ＥＶＥＸバイト２、ビット［１：０］－ｐｐ）は、ベースオペレーションフィールドのために追加ビットを提供する。ＥＶＥＸプレフィックスフォーマットでレガシＳＳＥ命令のためのサポートを提供することに加えて、これは、ＳＩＭＤプレフィックスを圧縮するという利益も有する（ＥＶＥＸプレフィックスは、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、２ビットのみを必要とする）。一実施形態において、レガシフォーマットおよびＥＶＥＸプレフィックスフォーマットの両方でＳＩＭＤプレフィックス（６６Ｈ、Ｆ２Ｈ、Ｆ３Ｈ）を用いるレガシＳＳＥ命令をサポートするべく、これらのレガシＳＩＭＤプレフィックスは、ＳＩＭＤプレフィックスエンコードフィールドへとエンコードされ、（ＰＬＡが変更なしにレガシフォーマットおよびこれらのレガシ命令のＥＶＥＸフォーマットの両方を実行し得るように）ランタイム時にデコーダのＰＬＡに提供される前にレガシＳＩＭＤプレフィックスへと拡張される。より新しい命令は、ＥＶＥＸプレフィックスエンコードフィールドの内容をオペコード拡張として直接に用い得るが、ある実施形態は、整合のために同様に拡張されるが、これらのレガシＳＩＭＤプレフィックスにより指定される異なる意味を可能にする。代替的な実施形態は、２ビットのＳＩＭＤプレフィックスエンコードをサポートするようにＰＬＡを再設計し得、従って拡張を必要としない。

【００７６】

アルファフィールド１５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ。ＥＶＥＸ．ＥＨ、ＥＶＥＸ．ｒｓ、ＥＶＥＸ．ＲＬ、ＥＶＥＸ．ライトマスク制御、およびＥＶＥＸ．Ｎとしても知られる。αでも示される）。上記ように、このフィールドは、コンテキストに固有である。

【００７７】

ベータフィールド１５４（ＥＶＥＸバイト３、ビット［６：４］－ＳＳＳ、ＥＶＥＸ．s_{2-0}、ＥＶＥＸ．r_{2-0}、ＥＶＥＸ．ｒｒ１、ＥＶＥＸ．ＬＬ０、ＥＶＥＸ．ＬＬＢとしても知られる。βββでも示される）。上記のように、このフィールドは、コンテキストに固有である。

【００７８】

ＲＥＸ'フィールド１１０。これは、ＲＥＸ'フィールドの残りであり、拡張された３２個のレジスタセットの上位の１６個または下位の１６個をエンコードするべく用いられ得るＥＶＥＸ．Ｖ'ビットフィールドのいずれかである（ＥＶＥＸバイト３、ビット［３］－Ｖ'）。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、下位の１６個のレジスタをエンコードするべく用いられる。換言すると、Ｖ'ＶＶＶＶは、ＥＶＥＸ．Ｖ'、ＥＶＥＸ．ｖｖｖｖを組み合わせることにより形成される。

【００７９】

ライトマスクフィールド１７０（ＥＶＥＸバイト３、ビット［２：０］－ｋｋｋ）。その内容は、上記のライトマスクレジスタにおけるレジスタのインデックスを指定する。本発明の一実施形態において、特定の値ＥＶＥＸ．ｋｋｋ＝０００は、特定の命令のために非ライトマスクが用いられることを暗に示す、特別な動作を有する（これは、全ての１に対するハードワイヤされたライトマスクの使用、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、様々な態様で実装され得る）。

【００８０】

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。オペコードの一部は、このフィールド内に指定される。

【００８１】

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。上記のように、ＭＯＤフィールド２４２の内容は、メモリアクセスオペレーションおよび非メモリアクセスオペレーショ

ンを区別する。Ｒｅｇフィールド２４４の役割は、２つの状況に要約され得る。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し、またはオペコード拡張として扱われ、任意の命令オペランドをエンコードするためには用いられない。Ｒ／Ｍフィールド２４６の役割は、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し、またはデスティネーションレジスタオペランドもしくはソースレジスタオペランドのいずれかをエンコードすることを含み得る。

【００８２】

スケール、インデックス、ベース（ＳＩＢ）バイト（バイト６）。上記のように、スケールフィールド１５０の内容は、メモリアドレス生成に用いられる。ＳＩＢ．ｘｘｘ２５４およびＳＩＢ．ｂｂｂ２５６。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に関連して既に言及された。

【００８３】

変位フィールド１６２Ａ（バイト７～１０）。ＭＯＤフィールド２４２が１０を含む場合、バイト７～１０は、変位フィールド１６２Ａであり、これはレガシ３２ビット変位（ｄｉｓｐ３２）と同様に機能し、バイト粒度で機能する。

【００８４】

変位係数フィールド１６２Ｂ（バイト７）。ＭＯＤフィールド２４２が０１を含む場合、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）の位置と同じである。ｄｉｓｐ８は、符号拡張されるので、－１２８～１２７バイトのオフセットのみをアドレス指定し得る。６４バイトのキャッシュラインに関しては、ｄｉｓｐ８は、４つの本当に有用な値である－１２８、－６４、０、および６４のみに設定され得る８ビットを用いる。多くの場合に、より広いレンジが必要とされるので、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とは対照的に、変位係数フィールド１６２Ｂは、ｄｉｓｐ８の再解釈である。変位係数フィールド１６２Ｂを用いる場合、実際の変位は、メモリオペランドアクセスのサイズ（Ｎ）で乗算される変位係数フィールドの内容により決定される。このタイプの変位は、ｄｉｓｐ８＊Ｎと称される。これにより、平均命令長（単一バイトだが、はるかに広いレンジの変位に用いられる）を小さくする。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、エンコードされる必要がないという前提に基づく。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。従って、ｄｉｓｐ８がｄｉｓｐ８＊Ｎにオーバーロードされることのみを例外として、変位係数フィールド１６２Ｂは、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（従って、ＭｏｄＲＭ／ＳＩＢエンコードルールに変更はない）。換言すると、エンコードルールまたはエンコードの長さに変更はないが、（バイト的アドレスオフセットを得るべく、メモリオペランドのサイズにより変位を調節する必要がある）ハードウェアによる変位値の解釈のみには変更がある。

【００８５】

即値フィールド１７２は、上記のように動作する。

【００８６】

フルオペコードフィールド

図２Ｂは、本発明の一実施形態による、フルオペコードフィールド１７４を構成する特定ベクトル向け命令フォーマット２００の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、ベースオペレーションフィールド１４２、およびデータ要素幅（Ｗ）フィールド１６４を含む。ベースオペレーションフィールド１４２は、プレフィックスエンコードフィールド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。

【００８７】

レジスタインデックスフィールド

図２Ｃは、本発明の一実施形態による、レジスタインデックスフィールド１４４を構成する特定ベクトル向け命令フォーマット２００の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。

【００８８】

追加オペレーションフィールド

図２Ｄは、本発明の一実施形態による、追加オペレーションフィールド１５０を構成する特定ベクトル向け命令フォーマット２００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１６８が０を含む場合、ＥＶＥＸ．Ｕ０（クラスＡ１６８Ａ）を意味する。１を含む場合、ＥＶＥＸ．Ｕ１（クラスＢ１６８Ｂ）を意味する。Ｕ＝０、かつＭＯＤフィールド２４２が１１を含む場合（非メモリアクセスオペレーションを意味する）、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（ラウンド１５２Ａ．１）を含む場合、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ビットのラウンドオペレーションフィールド１５８を含む。ＲＳフィールド１５２Ａが０（データ変換１５２Ａ．２）を含む場合、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であり、かつＭＯＤフィールド２４２が００、０１、または１０を含む場合（メモリアクセスオペレーションを意味する）、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。

【００８９】

Ｕ＝１である場合、アルファフィールド１５２（ＥＶＥＸバイト３、ビット［７］－ＥＨ）は、ライトマスク制御（Ｚ）フィールド１５２Ｃとして解釈される。Ｕ＝１であり、かつＭＯＤフィールド２４２が１１を含む場合（非メモリアクセスオペレーションを意味する）、ベータフィールド１５４（ＥＶＥＸバイト３、ビット［４］－$Ｓ_{0}$）の一部は、ＲＬフィールド１５７Ａとして解釈される。１（ラウンド１５７Ａ．１）を含む場合、ベータフィールド１５４（ＥＶＥＸバイト３、ビット［６－５］‐$Ｓ_{2-1}$）の残りは、ラウンドオペレーションフィールド１５９Ａとして解釈されるが、ＲＬフィールド１５７Ａが０（ＶＳＩＺＥ１５７．Ａ２）を含む場合、ベータフィールド１５４（ＥＶＥＸバイト３、ビット［６‐５］‐$Ｓ_{2-1}$）の残りは、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であり、ＭＯＤフィールド２４２が００、０１、または１０を含む場合（メモリアクセスオペレーションを意味する）、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。

【００９０】

Ｃ．例示的なレジスタアーキテクチャ

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。示される実施形態において、５１２ビット幅の３２個のベクトルレジスタ３１０が存在する。これらのレジスタは、ｚｍｍ０～ｚｍｍ３１として参照される。下位の１６個のｚｍｍレジスタの下位の２５６ビットは、レジスタｙｍｍ０～１５上にオーバーレイされる。下位の１６個のｚｍｍレジスタの下位の１２８ビット（ｙｍｍレジスタの下位の１２８ビット）は、レジスタｘｍｍ０～１５上にオーバーレイされる。以下の表に示されるよ

うに、特定ベクトル向け命令フォーマット２００は、これらのオーバーレイされたレジスタファイルで動作する。

【表１】

調整可能なベクトル長	クラス	オペレーション	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）

【００９１】

換言すると、ベクトル長フィールド１５９Ｂは、最大長、および１もしくは複数の他のより短い長さのうちから選択され、そのような各々のより短い長さは、先述の長さの半分の長さである。ベクトル長フィールド１５９Ｂを用いない命令テンプレートは、最大ベクトル長で動作する。更に、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、パックドもしくはスカラ単精度／倍精度浮動小数点データおよびパックドもしくはスカラ整数データで動作する。スカラオペレーションは、ｚｍｍ／ｙｍｍ／ｘｍｍレジスタにおける最下位のデータ要素位置で実行されるオペレーションである。より高位のデータ要素位置は、命令前と同じままであるか、または実施形態に応じてゼロにされる。

【００９２】

ライトマスクレジスタ３１５。示される実施形態において、８つのライトマスクレジスタ（ｋ０～ｋ７）が存在し、各々は６４ビットのサイズである。代替的な実施形態において、ライトマスクレジスタ３１５は１６ビットのサイズである。上記のように、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、ライトマスクとして使用され得ない。通常、ｋ０を示すエンコードがライトマスクに用いられる場合、これは、０ｘＦＦＦＦのハードワイヤードライトマスクを選択し、その命令に対するライトマスキングを実質的に無効にする。

【００９３】

汎用レジスタ３２５。示される実施形態において、メモリオペランドをアドレス指定する既存のｘ８６のアドレス指定モードと共に用いられる１６個の６４ビット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の名称により参照される。

【００９４】

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上に、ＭＭＸパックド整数フラットレジスタファイル３５０がエイリアスされ、示される実施形態において、ｘ８７スタックは、ｘ８７命令セット拡張を用いて、３２／６４／８０ビット浮動小

数点データにスカラ浮動小数点オペレーションを実行するべく用いられる８つの要素のスタックである。ＭＭＸレジスタは、６４ビットパックド整数データにオペレーションを実行すると共に、ＭＭＸレジスタとＸＭＭレジスタとの間で実行されるいくつかのオペレーションのためのオペランドを保持するべく用いられる。

【００９５】

本発明の代替的な実施形態は、より広いか、またはより狭いレジスタを用い得る。更に、本発明の代替的な実施形態は、より多いか、より少ないか、または異なるレジスタファイルおよびレジスタを用い得る。

【００９６】

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

プロセッサコアは、異なる態様で異なる目的のために異なるプロセッサに実装され得る。例えば、そのようなコアの実装は、１）汎用演算用の汎用インオーダコア、２）汎用演算用の高性能汎用アウトオブオーダコア、３）主にグラフィックスおよび／またはサイエンティフィック（スループット）演算用の専用コアを含み得る。異なるプロセッサの実装は、１）汎用演算用の１もしくは複数の汎用インオーダコア、および／または汎用演算用の１もしくは複数の汎用アウトオブオーダコアを含むＣＰＵ、ならびに２）主にグラフィックスおよび／またはサイエンティフィック（スループット）用の１もしくは複数の専用コアを含むコプロセッサを含み得る。そのような異なるプロセッサは、異なるコンピュータシステムアーキテクチャをもたらし、異なるコンピュータシステムアーキテクチャは、１）ＣＰＵの別個のチップ上のコプロセッサ、２）ＣＰＵと同一のパッケージにおける別個のダイ上のコプロセッサ、３）ＣＰＵと同一のダイ上のコプロセッサ（この場合、そのようなコプロセッサは、場合によっては統合グラフィックスおよび／またはサイエンティフィック（スループット）ロジック等の専用ロジック、または専用コアとして言及される）、および４）同一のダイ上に、説明されたＣＰＵ（場合によっては、アプリケーションコアもしくはアプリケーションプロセッサとして言及される）、上記のコプロセッサ、および追加の機能性を含み得るシステムオンチップを含み得る。例示的なコアアーキテクチャが次に説明され、その次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の説明が続く。

【００９７】

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の例示的な実施形態および例示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４Ａ～図４Ｂにおける実線ボックスは、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、破線ボックスの任意選択の追加は、レジスタリネーム、アウトオブオーダ発行／実行のパイプラインおよびコアを示す。インオーダの態様がアウトオブオーダの態様のサブセットであることを考慮して、アウトオブオーダの態様が説明される。

【００９８】

図４Ａにおいて、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、アロケーションステージ４０８、リネームステージ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、レジスタ読み出し／メモリ読み出しステージ４１４、実行ステージ４１６、ライトバック／メモリライトステージ４１８、例外処理ステージ４２２、およびコミットステージ４２４を含む。

【００９９】

図４Ｂは、実行エンジンユニット４５０に結合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、これら両方はメモリユニット４７０に結合されている。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

ドもしくは代替的なコアタイプであり得る。なおも別の選択肢として、コア４９０は、例えば、ネットワークコアもしくは通信コア、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ス処理ユニット（ＧＰＧＰＵ）コア、グラフィックスコア等のような専用コアであってもよい。

【０１００】

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に結合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に結合され、ＴＬＢ４３６は、命令フェッチユニット４３８に結合され、命令フェッチユニット４３８は、デコードユニット４４０に結合される。デコードユニット４４０（もしくはデコーダ）は、命令をデコードして、出力として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、または元の命令からデコードされ、もしくは別の方法で元の命令を反映し、もしくは元の命令から派生した他の制御信号を生成し得る。デコードユニット４４０は、様々な異なるメカニズムを用いて実装され得る。好適なメカニズムの例としては、ルックアップテーブル、ハードウェア実装、プログラマブルロジックアレイ（ＰＬＡ）、マイクロコードリードオンリメモリ（ＲＯＭ）等が挙げられるが、これらに限定されない。一実施形態において、コア４９０は、特定のマクロ命令用のマイクロコードを（例えば、デコードユニット４４０内、またはそうでなければフロントエンドユニット４３０内に）格納するマイクロコードＲＯＭまたは他のメディアを含む。デコードユニット４４０は、実行エンジンユニット４５０におけるリネーム／アロケータユニット４５２に結合される。

【０１０１】

実行エンジンユニット４５０は、リタイアメントユニット４５４に結合されたリネーム／アロケータユニット４５２と、１または複数のスケジューラユニット４５６のセットとを含む。スケジューラユニット４５６は、リザベーションステーション、中央命令ウィンドウ等を含む任意の数の異なるスケジューラを表す。スケジューラユニット４５６は、物理レジスタファイルユニット４５８に結合される。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し、これらの異なるものが、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、実行されるべき次の命令のアドレスである命令ポインタ）等のような１または複数の異なるデータタイプを格納する。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、ライト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得る。（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用い、フューチャファイル、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用い、レジスタマップおよびレジスタのプールを用いる等して）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得る様々な態様を示すべく、物理レジスタファイルユニット４５８は、リタイアメントユニット４５４と重ね合わされている。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に結合される。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）に対して様々なオペレーション（例えば、シフト、加算、減算、乗算）を実行し得る。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のいくつかの実行ユニットを含み得るが、他の実施形態は、１つの実行ユニットのみ、または全てがあらゆる関数を実行する複数の実行ユニットを含み得る。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は、場合によっては複数のものとして示される。なぜなら、ある実施形態は、一定のタイプのデータ／オペレーションのための別個のパイプライン（例えば、各々が自身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し、別個のメモリアクセスパイプラインの場合に、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を有する一定の実施形態が実装される）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を生成するからである。また、別個のパイプラインが用いられる場合に、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／実行であり、残りはインオーダであり得ることを理解されたい。

【０１０２】

メモリアクセスユニット４６４のセットは、メモリユニット４７０に結合される。メモリユニット４７０は、データキャッシュユニット４７４に結合されたデータＴＬＢユニット４７２を含み、データキャッシュユニット４７４は、レベル２（Ｌ２）キャッシュユニット４７６に結合される。例示的な一実施形態において、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み得、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に結合される。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に更に結合される。Ｌ２キャッシュユニット４７６は、１つまたは複数の他のレベルのキャッシュに結合され、最終的にはメインメモリに結合される。

【０１０３】

例として、例示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、パイプライン４００を以下のように実装し得る。１）命令フェッチ４３８は、フェッチステージ４０２および長さデコードステージ４０４を実行する。２）デコードユニット４４０はデコードステージ４０６を実行する。３）リネーム／アロケータユニット４５２は、アロケーションステージ４０８およびリネームステージ４１０を実行する。４）スケジューラユニット４５６は、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は、レジスタ読み出し／メモリ読み出しステージ４１４を実行し、実行クラスタ４６０は、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は、ライトバック／メモリライトステージ４１８を実行する。７）様々なユニットは、例外処理ステージ４２２に関与してもよい。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は、コミットステージ４２４を実行する。

【０１０４】

コア４９０は、本明細書において説明される命令を含む、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を追加された、いくつかの拡張を伴う）、カリフォルニア州サニーベールのＭＩＰＳ　ＴｅｃｈｎｏｌｏｇｉｅｓのＭＩＰＳ命令セット、カリフォルニア州サニーベールのＡＲＭ　ＨｏｌｄｉｎｇｓのＡＲＭ命令セット（ＮＥＯＮ等の任意選択の追加拡張を伴う）をサポートし得る。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）をサポートするためのロジックを含み、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オペレーションが、パックドデータを用いて実行されることを可能にする。

【０１０５】

コアは、（オペレーションまたはスレッドの２またはそれより多い並列セットを実行する）マルチスレッディングをサポートし得、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（物理コアが同時にマルチスレッディングするスレッドの各々のための論理コアを、単一の物理コアが提供する）、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デコードを行い、その後にインテル（登録商標）ハイパースレッディング技術等の同時マルチスレッディングを行う）を含む様々な態様でこれを実行し得ることを理解されたい。

【０１０６】

レジスタリネームは、アウトオブオーダ実行の文脈で説明されているが、レジスタリネームは、インオーダアーキテクチャにおいて用いられ得ることを理解されたい。プロセッ

サの示される実施形態は、別個の命令およびデータキャッシュユニット４３４／４７４、ならびに共有Ｌ２キャッシュユニット４７６も含むが、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のレベルの内部キャッシュ等の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得る。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、ならびにコアおよび／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み得る。あるいは、キャッシュの全てがコアおよび／またはプロセッサの外部にあってもよい。

【０１０７】

図５Ａ～図５Ｂは、より具体的な例示的なインオーダコアアーキテクチャのブロック図を示し、このコアは、チップにおける（同じタイプの他のコアおよび／または異なるタイプを含む）いくつかの論理ブロックのうちの１つである。アプリケーションに応じて、論理ブロックは、ある固定機能ロジック、メモリＩ／Ｏインタフェース、および他の必要なＩ／Ｏロジックを用いる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。

【０１０８】

図５Ａは、本発明の実施形態によるシングルプロセッサコアのブロック図であり、オンダイ相互接続ネットワーク５０２への接続に加え、レベル２（Ｌ２キャッシュ５０４のローカルサブセットを有する。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用いるｘ８６命令セットをサポートする。Ｌ１キャッシュ５０６は、スカラユニットおよびベクトルユニット内のキャッシュメモリへの低レイテンシアクセスを可能にする。一実施形態において（設計を簡略化するべく）、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（各々、スカラレジスタ５１２およびベクトルレジスタ５１４）を用い、それらの間で転送されるデータは、メモリに書き込まれ、次にレベル１（Ｌ１）キャッシュ５０６からリードバックされる。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用い、またはライトバック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の間で、データが転送されることを可能にする通信パスを含む）を用い得る。

【０１０９】

Ｌ２キャッシュ５０４のローカルサブセットは、プロセッサコア毎に１つずつ、別個のローカルサブセットに分割される全体的なＬ２キャッシュの一部である。各プロセッサコアは、Ｌ２キャッシュ５０４の自身のローカルサブセットへの直接のアクセス経路を有する。プロセッサコアにより読み出されたデータは、そのＬ２キャッシュサブセット５０４に格納され、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して迅速にアクセスされ得る。プロセッサコアにより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、必要であれば他のサブセットからフラッシュされる。リングネットワークは、共有データ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。各リングのデータパスは、１方向毎に１０１２ビット幅である。

【０１１０】

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は、Ｌ１キャッシュ５０４のＬ１データキャッシュ５０６Ａ部分、ならびにベクトルユニット５１０およびベクトルレジスタ５１４に関する更なる詳細を含む。具体的には、ベクトルユニット５１０は、１６ワイドのベクトル処理ユニット（ＶＰＵ）（１６ワイドのＡＬＵ５２８を参照）であり、これは整数命令、単精度浮動小数点命令、および倍精度浮動小数点命令のうちの１または複数を実行する。ＶＰＵは、スウィズルユニット５２０によるレジスタ入力のスウィズル、数値変換ユニット５２２Ａ～Ｂによる数値変換、およびメモリ入力時のレプリケーションユニット５２４によるレプリケーションをサポートする。ライトマスクレジスタ５２６は、結果として生じるベクトル書き込みを予測（ｐｒｅｄｉｃａｔｉｎｇ）することを可能にする。

【０１１１】
図６は、本発明の実施形態による、２つ以上のコアを有し得、統合メモリコントローラを有し得、統合グラフィックスを有し得るプロセッサ６００のブロック図である。図６の実線ボックスは、シングルコア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のセットを有するプロセッサ６００を示すが、破線ボックスの任意選択の追加は、複数のコア６０２Ａ～Ｎ、システムエージェントユニット６１０における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を有する代替的なプロセッサ６００を示す。
【０１１２】
従って、プロセッサ６００の異なる実装は、１）統合グラフィックスおよび／またはサイエンティフィック（スループット）ロジック（１または複数のコアを含み得る）である専用ロジック６０８、および１または複数の汎用コア（例えば、汎用インオーダコア、汎用アウトオブオーダコア、２つの組み合わせ）であるコア６０２Ａ～Ｎを用いるＣＰＵ、２）主にグラフィックスおよび／またはサイエンティフィック（スループット）用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、ならびに３）多数の汎用インオーダコアであるコア６０２Ａ～Ｎを用いるコプロセッサを含み得る。従って、プロセッサ６００は、例えば、ネットワークプロセッサもしくは通信プロセッサ、圧縮エンジン、グラフィックスプロセッサ、ＧＰＧＰＵ（汎用グラフィックス処理ユニット）、（３０またはそれより多いコアを含む）高スループット多集積コア（ＭＩＣ）コプロセッサ、エンベデッドプロセッサ等のような汎用プロセッサ、コプロセッサもしくは専用プロセッサであってもよい。プロセッサは、１または複数のチップ上に実装され得る。プロセッサ６００は、１または複数の基板の一部であり得、および／または例えば、ＢｉＣＭＯＳ、ＣＭＯＳ、またはＮＭＯＳ等のいくつかの処理技術のいずれかを用いてこれらの基板上に実装され得る。
【０１１３】
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、１セットまたは１も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のセットに結合された外部メモリ（図示せず）を含む。共有キャッシュユニット６０６のセットは、レベル２（Ｌ２）、レベル３（Ｌ３）、レベル４（Ｌ４）等の１または複数の中間レベルのキャッシュ、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／またはこれらの組み合わせを含み得る。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、統合グラフィックスロジック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を相互接続し、代替的な実施形態は、そ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の技術を用い得る。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とコア６０２Ａ～Ｎとの間で保持される。
【０１１４】
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することができる。システムエージェント６１０は、コア６０２Ａ～Ｎを調整および動作させ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み得る。ＰＣＵは、コア６０２Ａ～Ｎおよび統合グラフィックスロジック６０８の電力状態を調整するのに必要とされるロジックおよびコンポーネントであるか、またはこれらを含み得る。ディスプレイユニットは、１または複数の外部接続ディスプレイを駆動するためのものである。
【０１１５】
コア６０２Ａ～Ｎは、アーキテクチャ命令セットの観点からは同種または異種であり得る。すなわち、コア６０２Ａ～Ｎのうちの２またはそれより多くのものは、同じ命令セットを実行することができる場合があるが、他のものは、当該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することができる場合がある。

【０１１６】
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、情報携帯端末、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、エンベデッドプロセッサ、デジタル信号プロセッサ（ＤＳＰ）、グラフィックスデバイス、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。一般に、本明細書において開示されるプロセッサおよび／または他の実行ロジックを組み込むことができる多種多様なシステムまたは電子デバイスが、概ね好適である。

【０１１７】
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る。システム７００は、１または複数のプロセッサ７１０、７１５を含み得、これらは、コントローラハブ７２０に結合される。一実施形態において、コントローラハブ７２０は、グラフィックスメモリコントローラハブ（ＧＭＣＨ）７９０および入出力ハブ（ＩＯＨ）７５０（別個のチップ上にあり得る）を含む。ＧＭＣＨ７９０は、メモリコントローラおよびグラフィックスコントローラを含み、これらにメモリ７４０およびコプロセッサ７４５が結合される。ＩＯＨ７５０は、入出力（Ｉ／Ｏ）デバイス７６０をＧＭＣＨ７９０に結合する。あるいは、メモリおよびグラフィックスコントローラの一方または両方が（本明細書に説明される）プロセッサ内に集積され、メモリ７４０およびコプロセッサ７４５は、プロセッサ７１０と、ＩＯＨ７５０を有する単一のチップのコントローラハブ７２０とに直接に結合される。

【０１１８】
追加のプロセッサ７１５の任意選択の性質が図７に破線を用いて示されている。各プロセッサ７１０、７１５は、本明細書に説明される処理コアのうちの１または複数を含み得、プロセッサ６００のいくつかのバージョンであり得る。

【０１１９】
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もよい。少なくとも一実施形態においては、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相互接続（ＱＰＩ）等のポイントツーポイントインタフェース、または類似の接続７９５を介してプロセッサ７１０、７１５と通信する。

【０１２０】
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もしくは通信プロセッサ、圧縮エンジン、グラフィックスプロセッサ、ＧＰＧＰＵ、エンベデッド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、統合グラフィックスアクセラレータを含み得る。

【０１２１】
物理リソース７１０と物理リソース７１５との間には、アーキテクチャ、マイクロアーキテクチャ、熱、電力消費の特性等を含む幅広い価値基準に関して様々な違いが存在し得る。

【０１２２】
一実施形態において、プロセッサ７１０は、一般なタイプのデータ処理オペレーションを制御する命令を実行する。コプロセッサ命令は、命令中に埋め込まれ得る。プロセッサ７１０は、取り付けられたコプロセッサ７４５により実行されるべきタイプとしてこれらのコプロセッサ命令を認識する。従って、プロセッサ７１０は、コプロセッサバスまたは他の相互接続において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す制御信号）をコプロセッサ７４５に発する。コプロセッサ７４５は、受信したコプロセッサ命令を受け取って実行する。

【０１２３】
ここで図８を参照すると、本発明の実施形態による第１のより具体的な例示的システム８００のブロック図が示される。図８に示されるように、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を介して結合される第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のいくつかのバージョンであり得る。本発明の一実施形態において、プロセッサ８７０および８８０は各々、プロセッサ７１０および７１５であるが、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、プロセッサ８７０および８８０は各々、プロセッサ７１０およびコプロセッサ７４５である。

【０１２４】
統合メモリコントローラ（ＩＭＣ）ユニット８７２および８８２を各々含むプロセッサ８７０および８８０が示される。プロセッサ８７０は、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も含む。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を用いて、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得る。図８に示されるように、ＩＭＣ８７２および８８２は、プロセッサを各メモリ、すなわち、各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り得るメモリ８３２およびメモリ８３４に結合する。

【０１２５】
プロセッサ８７０、８８０は各々、ポイントツーポイントインタフェース回路８７６、８９４、８８６、８９８を用い、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と情報を交換し得る。任意選択で、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と情報を交換し得る。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もしくは通信プロセッサ、圧縮エンジン、グラフィックスプロセッサ、ＧＰＧＰＵ、エンベデッドプロセッサ等のような専用プロセッサである。

【０１２６】
共有キャッシュ（図示せず）は、どちらかのプロセッサに含まれ、または両方のプロセッサの外部にあり得るが、プロセッサが低電力モードにされると、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に格納され得るように、Ｐ‐Ｐ相互接続を介してプロセッサとなおも接続され得る。

【０１２７】
チップセット８９０は、インタフェース８９６を介して第１のバス８１６に結合され得る。一実施形態において、第１のバス８１６は、周辺構成要素相互接続（ＰＣＩ）バス、またはＰＣＩ　Ｅｘｐｒｅｓｓバス等のバス、または別の第３世代Ｉ／Ｏ相互接続バスであり得るが、本発明の範囲はそのように限定されない。

【０１２８】
図８に示されるように、様々なＩ／Ｏデバイス８１４が、第１のバス８１６を第２のバス８２０に結合するバスブリッジ８１８と共に、第１のバス８１６に結合され得る。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、アクセラレータ（例えば、グラフィックス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、またはその他のプロセッサ等、１または複数の追加のプロセッサ８１５は、第１のバス８１６に結合される。一実施形態において、第２のバス８２０は、低ピンカウント（ＬＰＣ）バスであり得る。様々なデバイスは、一実施形態において、例えば、キーボードおよび／またはマウス８２２、通信デバイス８２７、ならびに命令／コードおよびデータ８３０を含み得るディスクドライブもしくは他の大容量ストレージデバイス等のストレージユニット８２８を含む第２のバス８２０に結合され得る。更に、オーディオＩ／Ｏ８２４は、第２のバス８２０に結合され得る

。他のアーキテクチャが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得る。

【０１２９】

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のブロック図を示す。図８および図９の同一の要素は、同一の参照番号を有し、図９の他の態様を不明瞭にするのを避けるべく、図８の特定の態様は、図９から省略されている。

【０１３０】

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を各々含み得ることを示す。従って、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に結合されていることのみならず、Ｉ／Ｏデバイス９１４も制御ロジック８７２、８８２に結合されていることを示す。レガシＩ／Ｏデバイス９１５は、チップセット８９０に結合される。

【０１３１】

ここで図１０を参照すると、本発明の一実施形態によるＳｏＣ１０００のブロック図が示される。図６における類似の要素は、同一の参照番号を有する。また、破線ボックスは、より高度なＳｏＣの任意選択の特徴である。図１０において、相互接続ユニット１００２は、１もしくは複数のコア６０６Ａ～Ｎおよび共有キャッシュユニット６０６のセットを含むアプリケーションプロセッサ１０１０、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、統合メモリコントローラユニット６１４、統合グラフィックスロジック、画像プロセッサ、オーディオプロセッサ、およびビデオプロセッサを含み得る１もしくは複数のコプロセッサ１０２０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１０３０、ダイレクトメモリアクセス（ＤＭＡ）ユニット１０３２、および１または複数の外部ディスプレイを結合するディスプレイユニット１０４０に結合される。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もしくは通信プロセッサ、圧縮エンジン、ＧＰＧＰＵ、高スループットＭＩＣプロセッサ、エンベデッドプロセッサ等のような専用プロセッサを含む。

【０１３２】

本明細書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、またはそのような実装アプローチの組み合わせで実装され得る。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メモリ、ならびに／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有するプログラマブルシステム上で実行するコンピュータプログラムまたはプログラムコードとして実装され得る。

【０１３３】

図８に示されるコード８３０等のプログラムコードは、本明細書に説明される機能を実行して出力情報を生成するべく、入力命令に適用され得る。出力情報は、既知の様式で１または複数の出力デバイスに適用され得る。本願の目的のために、処理システムは、例えば、デジタル信号プロセッサ（ＤＳＰ）、マイクロコントローラ、特定用途向け集積回路（ＡＳＩＣ）、またはマイクロプロセッサ等のプロセッサを有する任意のシステムを含む。

【０１３４】

プログラムコードは、高水準手続き型プログラミング言語またはオブジェクト指向プログラミング言語で実装され、処理システムと通信し得る。所望であれば、プログラムコードは、アセンブリ言語または機械語でも実装され得る。実際には、本明細書に説明されるメカニズムは、範囲においていずれの特定のプログラミング言語にも限定されない。いずれの場合においても、言語は、コンパイル型言語またはインタープリタ型言語であってもよい。

【０１３５】
少なくとも一実施形態の１または複数の態様は、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代表的命令により実装され得、命令は、機械により読み出されると、当該機械に本明細書に説明される技術を実行するためのロジックを生成させる。「ＩＰコア」として知られるそのような表現は、有形機械可読媒体上に格納され、様々な顧客または製造設備に供給され、ロジックまたはプロセッサを実際に作成する製造機械にロードされ得る。
【０１３６】
そのような機械可読ストレージ媒体としては、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）、書き換え可能コンパクトディスク（ＣＤ‐ＲＷ）、および光磁気ディスク等、その他のタイプのディスクを含むストレージ媒体、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）、および相変化メモリ（ＰＣＭ）等の半導体デバイス、磁気もしくは光カード、または電子命令を格納するのに好適なその他のタイプの媒体を含む、機械またはデバイスにより製造または形成される、非一時的で有形な構成の物品が挙げられ得るが、これらに限定されない。
【０１３７】
従って、本発明の実施形態は、命令を含み、または本明細書に説明される構造体、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有形機械可読媒体も含む。そのような実施形態は、プログラム製品と称され得る。
【０１３８】
いくつかの場合に、命令コンバータは、ソース命令セットからターゲット命令セットへと命令を変換するべく用いられ得る。例えば、命令コンバータは、（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用いて）命令を、コアにより処理されるべき１または複数の他の命令に翻訳し、モーフィングし、エミュレートし、または別の方法で変換し得る。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得る。命令コンバータは、プロセッサにあり、プロセッサから離れ、またはプロセッサ上の一部であり、プロセッサから離れた一部であり得る。
【０１３９】
図１１は、本発明の実施形態による、ソース命令セットのバイナリ命令を、ターゲット命令セットの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と対比するブロック図である。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が、あるいは、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得る。図１１は、高水準言語１１０２によるプログラムが、少なくとも１つのｘ８６命令セットコア１１１６を有するプロセッサによりネイティブに実行され得るｘ８６バイナリコード１１０６を生成するべく、ｘ８６コンパイラ１１０４を用いてコンパイルされ得ることを示す。少なくとも１つのｘ８６命令セットコア１１１６を有するプロセッサは、少なくとも１つのｘ８６命令セットコアを用いるインテル（登録商標）プロセッサと実質的に同一の結果を実現するべく、（１）インテル（登録商標）ｘ８６命令セットコアの命令セットの実質的部分、または（２）少なくとも１つのｘ８６命令セットコアを有するインテル（登録商標）プロセッサ上で起動することを目的とする、オブジェクトコードバージョンのアプリケーションまたは他のソフトウェアを互換的に実行し、または別の方法で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インテル（登録商標）プロセッサと実質的に同一の機能を実行し得る任意のプロセッサを表す。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す。ｘ８６バイナリコード１１０６は、追加のリンク処理を用いるか、または用いることなく少なくとも１つのｘ８６命令セットコア１１１６を有するプロセッサにおいて実行され得る。同様に、図１１は、高水準言語１１０２によるプログラムが、少なくとも１つのｘ８６命令セットコア１１１４を有しないプロセッサ（例えば、カリフォルニア州サニーベールのＭＩＰＳ　ＴｅｃｈｎｏｌｏｇｉｅｓのＭＩＰＳ命令セットを実行し、および／またはカリフォルニア州サニーベールのＡＲＭ　ＨｏｌｄｉｎｇｓのＡＲＭ命令セットを実行するコアを有するプロセッサ）によりネイティブに実行され得る代替的な命令セットバイナリコード１１１０を生成するべく、代替的な命令セットコンパイラ１１０８を用いてコンパイルされ得ることを示す。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１１１４を有しないプロセッサによりネイティブに実行され得るコードに変換するべく用いられる。この変換済みコードは、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が低い。なぜなら、これが可能な命令コンバータは、作成することが困難だからである。しかし、変換済みコードは、一般的なオペレーションを実現し、代替的な命令セットの命令から構成される。従って、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し、これらは、エミュレーション、シミュレーション、またはその他の処理により、ｘ８６命令セットプロセッサまたはコアを有し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を実行することを可能にする。

【０１４０】

命令フローを最適化するチェックを実装するための方法および装置

除算および平方根のようなＩＥＥＥの正確な丸め関数は、２つの態様で実装され得る。すなわち、基数ベースのハードウェアディバイダを用いて機能をネイティブに提供するか、またはソフトウェア命令／マイクロコード（ｕｃｏｄｅ）シーケンス、通常は、ニュートン・ファフソンアルゴリズムの形式／変形を実装し、最終的結果を得るべくシードに対してイテレートするかである。

【０１４１】

ソフトウェアシーケンスに関する１つの問題は、大抵の入力についてアルゴリズムは、非常に効率的に処理され得るのに対して、（例えば、入力非正規化を処理する場合に）一桁遅いシーケンスをもたらすいくつかの特定のコーナーケースには、追加に処理が必要とされるということである。更に、入力がトリビアルな結果を有する場合に結果をパッチするべく、最終的な後処理（ゼロによる除算等）が必要とされ、これによりシーケンスの限界レイテンシが増加する。

【０１４２】

一実施形態において、命令シーケンスは、最良の場合を想定しており、入力においてコーナーの場合が存在するときには、代替的または「低速」のシーケンスにジャンプし、デフォルトシーケンスのクリティカルパスからチェックアウトする。本発明の実施形態は、主要のデフォルト計算とは別個のパスでこの条件付きジャンプを予め計算する「チェック」命令またはマイクロオペレーション（ｕｏｐ）のタイプを含む。いくつかの実施形態において、命令チェックオペレーションは、トリビアルな入力に対するデフォルト応答も提供する。下記の実施形態は、除算および平方根（ｓｑｒｔ）の実装に焦点を当てるが、本発明の基礎となる原理は、いずれの特定のタイプの数学関数にも限定されない。より具体的には、一実施形態は、（例えば、プレディケーション／マスクを用いる）パックド平方根／除算のＡＶＸ‐５１２実装に基づくが、このコンセプトは、いずれの命令セットアーキテクチャにも拡張され得る。

【０１４３】

「チェック」命令／ｕｏｐの一実施形態は、入力に基づいて、デフォルト／ファーストフローを用いるアルゴリズムを処理するか否か、または代替的にスローフローにジャンプし、ブーリアンベクトルをマスクに残しておくべきかを決定する。更に、一実施形態は、計算毎の例外フラグを生成し（無効、非正規、ゼロによる除算）、マスクされない場合に

は、実行をアボートする。最終的には、本発明の一実施形態は、トリビアルな入力についてはデフォルト応答を予め計算する（例えば、無限大で割る等）。

【０１４４】

図１２に示されるように、本発明の実施形態が実装され得る例示的なプロセッサ１２５５は、命令チェック命令をデコードするためのプロセッサパイプラインのデコードステージ１２３０内における命令チェックデコードロジック１２３１と、命令チェック命令を実行するためのパイプラインの実行ステージ１２４０内における命令チェック実行ロジック１２４１とを含む。一実施形態において、命令チェックデコードロジック１２３１は、命令チェック命令を複数のマイクロオペレーションにデコードするためのハードウェアを備え、複数のマイクロオペレーションは次に、命令チェック実行ロジック１２４１により実行される。しかし、本発明の基礎となる原理は、マイクロコード化プロセッサに限定されない。言及されたように、一実施形態において、命令は、除算チェック命令（ＤＩＶＣＨＫ（））および平方根チェック（ＳＱＲＴＣＨＫ（））を含む。しかし、本発明の基礎となる原理は、これら特定の実装に限定されない。

【０１４５】

図１２に示される他のプロセッサコンポーネントは、汎用レジスタ（ＧＰＲ）１２０５のセット、ベクトルレジスタ１２０６のセット、マスクレジスタ１２０７のセット、および制御レジスタ１２０８のセットを含む。図１２に明示的に示されていないが、制御レジスタ１２０８は、別個の制御レジスタバス（ＣＲＢｕｓ）インタフェースを介してアクセスされ得る。一実施形態において、マスクレジスタ１２０７は、（例えば、上記のマスクレジスタｋ０～ｋ７として実装される）ベクトルレジスタ１２０６に格納された値にビットマスキングオペレーションを実行するために用いられる８個の６４ビットオペランドのマスクレジスタを含む。しかし、本発明の基礎となる原理は、いずれの特定のマスクレジスタサイズ／タイプにも限定されない。一実施形態において、複数のベクトルデータ要素は、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ビット幅を有し得る各ベクトルレジスタ１２０６にパックされ得る。しかし、本発明の基礎となる原理は、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。

【０１４６】

シングルプロセッサコア（「コア０」）の詳細は、簡略化するために図１２に示される。しかし、図１２に示される各コアは、コア０と同じセットのロジックを有し得ることが理解されよう。例えば、各コアは、指定されたキャッシュ管理ポリシに従って命令およびデータをキャッシュするための専用レベル１（Ｌ１）キャッシュ１２１２およびレベル２（Ｌ２）キャッシュ１２１１を含み得る。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と、データを格納するための別個のデータキャッシュ１２２１とを含む。様々なプロセッサキャッシュに格納された命令およびデータは、固定サイズ（例えば、６４バイト、１２８バイト、５１２バイトの長さ）であり得る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１２００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と、命令をデコードする（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「マイクロオプ」にデコードする）ためのデコードユニット１２２０と、命令を実行するための実行ユニット１２４０と、命令をリタイアして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とを有する。

【０１４７】

命令フェッチユニット１２１０は、メモリ１２００（またはキャッシュのうちの１つ）からフェッチされるべき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と、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させ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と、命令分岐アドレスを推論的に予測するための分岐予測ユニット１２０２と、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）１２０

１とを含む、様々な周知のコンポーネントを含む。フェッチされると、命令は次に、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ミングされる。これらのユニットの各々の構造および機能は、当業者により良く理解されており、本発明の異なる実施形態の関連する態様を不明瞭にするのを避けるべく、ここでは詳細に説明されない。

【０１４８】

図１３は、現在の命令に関連する複数の入力値１３０１～１３０３（例えば、除算命令または平方根命令のための入力）に基づいて、特定の命令シーケンス（例えば、デフォルト／高速シーケンスまたは代替的／低速シーケンス）を選択するためのベクトル出力およびマスク出力１３１１を生成する命令チェックロジック１３００を示す。除算命令および平方根命令のためのベクトル出力およびマスク出力の様々な具体的な例が以下に提供される（例えば、図１４（除算）および図１５（平方根）を参照）。更に、いくつかの実施形態において、命令チェックロジック１３００は、特定のタイプの入力（例えば、無限大で割るなどしたトリビアルな入力）についての結果１３１２を生成し得、後続の命令シーケンス１３１１が使用できる例外フラグ１３１３を生成し得る。例として、例外フラグは、ゼロによる除算（ＺＥ）、無効な演算（ＩＥ）、および非正規化オペランド（ＤＥ）等の浮動小数点例外フラグを含み得る。

【０１４９】

上述のように、本明細書に説明される技術を実装する命令チェックロジック１３００は、命令を複数のｕｏｐにデコードするデコードステージ１２３０（例えば、命令チェックデコードロジック１２３１）と、ｕｏｐを実行する実行ステージ１２４０（例えば、命令チェック実行ロジック１２４１）内のロジックとを含み得る。しかし、本発明の基礎となる原理は、マイクロコード化プロセッサアーキテクチャに限定されないことに留意されたい。

【０１５０】

除算命令に関連する入力および他の変数にチェックを実行し、これに応じて適切な／効率的な命令シーケンスを選択して除算オペレーションを実装する除算チェック（ＤＩＶＣＨＫ）命令の一実施形態が、ここで説明される。一実施形態において、除算のためのマイクロコードフロー手順は、以下の通りである。

【数１】

```
division(x,y)
{
        start_fast_sequence(x,y);
        if(DIVCHK())
                jump slow_flow;
        else
                finish_fast_flow(x,y)
                return res;
slow_flow:
        do_slow_flow(x,y)
        return res;
}
```

【０１５１】

上記のコードにおいて、ＤＩＶＣＨＫは、入力（ｘ，ｙ）を評価し、代替的な「ｓｌｏｗ＿ｆｌｏｗ」命令シーケンスにジャンプするか否か、または通常の「ｆｉｎｉｓｈ＿ｆａｓｔ＿ｆｌｏｗ」シーケンスを完了するか否かを決定する。一実施形態において、以下

のＤＩＶＣＨＫ仕様が使用される。

【０１５２】

一実施形態において、マイクロオプＵＯＰ＿｛ＪＤＩＶＣＨＫ｝｛ＰＳ，ＰＤ｝が使用され、第１のパラメータ（Ｐａｒａｍ１）は、除算入力をチェックし、第２のパラメータ（Ｐａｒａｍ２）は、オペレーションが単精度入力を伴うか、または倍精度入力を伴うかを示す。ＵＯＰは、第１のソース（ｓｒｃＡ）を分子として、第２のソース（ｓｃｒＢ）を分母として用い得、除算シーケンスに対する入力の特性をチェックして、３つの異なる情報を返す。　１．特別の場合の入力のデフォルト応答によるベクトル出力（例えば、ゼロによる除算は、ＩＮＦを返す）。一実施形態において、これは、スローパスアルゴリズムを用いて実際の出力が生成される必要がある場合に、特別なインジケーション（例えば、＋１．０）を出力する。　２．ファーストパスアルゴリズムを用いて処理され得る要素をシグナリングするマスク出力。一実施形態において、ゼロのマスクは、結果がトリビアルなものであるか、またはスローフローシーケンス（＋１．０等の特別なインジケーションにより指定された）により計算が行われる必要があるかのいずれかを意味する。　３．計算毎の例外フラグの生成。これは、例えば、ゼロによる除算（ＺＥ）、無効な演算（ＩＥ）、および非正規化オペランド（ＤＥ）等の浮動小数点例外フラグを含み得る。

【０１５３】

図１４に示される表は、２つの異なる入力タイプ（ＸはＩＮＰＵＴ１であるが、ＹはＩＮＰＵＴ２である）の関数における３つの出力を要約する。この表において、Ｘ入力は行に示され、Ｙ入力は、列に示されている。結果として生じる出力は、（ｋｄｓｔ，ｖｄｓｔ，ｆｌａｇｓ）として構成され、「ｋｄｓｔ」はマスク出力を含み、「ｖｄｓｔ」はベクトル出力を含み、「ｆｌａｇｓ」は例外フラグを含む。入力「ｄｅｎｏｒｍ」は非正規化オペランドを表し、「ｎｏｒｍａｌ」は通常のオペランドを表し、「Ｉｎｆ」は無限を表す。ＮａＮは「非数（ｎｏｔ　ａ　ｎｕｍｂｅｒ）」を表し、ＱＮａＮはｑｕｉｅｔ　ＮａＮを表し、ＳＮａＮはｓｉｇｎａｌｉｎｇ　ＮａＮを表す。ＱＮａＮは、最上位の端数ビットを１にセットしたＮａＮであり、ＳＮａＮは、最上位の端数ビットをクリアしたＮａＮである。

【０１５４】

一実施形態において、両方の入力が正規数である場合（表において「以下を参照」で示される）は、以下のように処理される。

【数２】

```
xExp = biased exponent of x;   // 指数のメモリ形式
yExp = biased exponent of y;   // 指数のメモリ形式
eDiff = xExp – yExp;
      if (     (eDiff ≥ bias + 1 ) || // 商がオーバーフローする場合がある
               (eDiff ≤ (1 – bias) ) || // 商がアンダーフローする場合がある
               (yExp ≥ (2*bias – 1) ) || // 逆数がアンダーフローする場合がある
               (xExp ≤ 2*precision ) // 剰余がアンダーフローする場合がある
      ) {
         return (0,+1.0,None);
      } else
         return (1,+0.0,None);
      }
```

従って、上記のコードは、商のオーバーフロー／アンダーフロー、逆数アンダーフロー、および剰余のアンダーフローについてテストする。これらの条件のいずれかが検出されると、ベクトル出力は＋１．０であり、そうでなければベクトル出力は＋０．０である。

剰余テストにおいては、ＩＳＡから生じる入出力の非正規数が内部計算においてフラッシュされ得る。

【０１５５】

一実施形態において、ＤＩＶＣＨＫオペレーションは、以下の形式を取る。

【数３】

```
// UOP_DIVCHK_PS (.INPUT1(src_a), .INPUT2(src_b))
for(n=0; n<2; n+=1)
{
        i = n*32
        {MASK_OUTPUT[n], VECT_OUTPUT[i+31:i]}
                = divchk(INPUT1[i+31:i], INPUT2[i+31:i]) // (x, y)
}
```

【０１５６】

例外は、単精度および倍精度の浮動小数点データ要素の両方に対する非正規（ＤＡＺ＝０の場合）、無効、およびゼロによる除算に対して生成され得る。

【０１５７】

平方根命令に関連する入力および他の変数にチェックを実行し、これに応じて適切な／効率的な命令シーケンスを選択して平方根演算を実装する平方根チェック（ＳＱＲＴＣＨＫ）命令の一実施形態が、ここで説明される。一実施形態において、平方根のマイクロコードフロー手順は、以下のとおりである。

【数４】

```
sqrt(x)
{
        start_fast_sequence(x);
if(SQRTCHK())
        jump slow_flow;
else
        finish_fast_flow(x)
        return res;
Slow_flow:
        do_slow_flow(x)
        return res;
}
```

【０１５８】

上記のコードにおいて、ＳＱＲＴＣＨＫは、入力ｘを評価し、代替的な「ｓｌｏｗ＿ｆｌｏｗ」命令シーケンスにジャンプするか否か、または通常の「ｆｉｎｉｓｈ＿ｆａｓｔ＿ｆｌｏｗ」シーケンスを完了するか否かを決定する。一実施形態において、以下のＳＱＲＴＣＨＫ仕様が使用される。

【０１５９】

ＵＯＰ＿｛ＪＳＱＲＴＣＨＫ｝｛ＰＳ，ＰＤ｝は、一実施形態において使用されるマイクロオペレーションであり、第１のパラメータ｛ＳＱＲＴＣＨＫ｝は、平方根命令入力をチェックし、第２のパラメータ｛ＰＳ，ＰＤ｝は、オペレーションが単精度入力または倍精度入力を伴うかを示す。ＵＯＰは、平方根シーケンスに対する入力の特性をチェックし

、３つの異なる情報を返す。　１．特別の場合の入力のデフォルト応答によるベクトル出力（例えば、負の数の平方根は、ＮａＮを返す）。一実施形態において、これは、ファーストパスアルゴリズムを用いて実際の出力が生成される必要がある場合に、特別なインジケーション（例えば、＋１．０）を出力する。　２．スローパスアルゴリズムを用いて処理されなければならない要素をシグナリングするマスク出力。　３．計算毎の例外フラグの生成。これは、例えば、無効な演算（ＩＥ）、および非正規化オペランド（ＤＥ）等の浮動小数点例外フラグを含み得る。

【０１６０】

図１５に示される表は、ｘ（第１の列）に対する様々な異なる入力タイプおよび（ｋｄｓｔ，ｖｄｓｔ，ｆｌａｇｓ）として構成された、結果として生じる出力を列挙し、「ｋｄｓｔ」はマスク出力を含み、「ｖｄｓｔ」はベクトル出力を含み、「ｆｌａｇｓ」は例外フラグを含む。入力は、－ｎｏｒｍを含む（－無限大＋／－非正規化オペランド、＋／－０、＋無限大、ｑＮａＮ、およびｓＮａＮを含む）。制限ＢｉａｓＥｘｐｏｎｅｎｔ（ｘ）＜２＊の精度（表において条件（ａ）として識別される）は、ｄ＝ｘ－ｓ＊ｓの計算におけるアンダーフローを防止することが意図される。

【数５】

```
// UOP_SQRTCHK_PS (.INPUT1(src_b))
for(n=0; n<2; n+=1)
{
        i = n*32
        {MASK_OUTPUT[n], VECT_OUTPUT[i+31:i]}
        = sqrtchk(INPUT1[i+31:i]);
}
```

【０１６１】

例外は、単精度および倍精度の浮動小数点データ要素の両方に対する非正規（ＤＡＺ＝０の場合）および無効に対して生成され得る。

【０１６２】

本発明の一実施形態による方法は、図１６に示される。本方法は、上記のアーキテクチャの文脈内で実装され得るが、いずれの特定のアーキテクチャにも限定されない。

【０１６３】

１６０１において、数学関数から入力が取得される。例えば、数学関数が除算である場合、入力は、分子（ｘ）および分母（ｙ）を含む。数学関数が平方根である場合、入力は、平方根が必要とされる数（ｘ）を含む。１６０２において、数学関数に応じて入力の様々な特性が評価され、代替的／低速の命令シーケンスにジャンプを行うべきか否かを示す出力を生成する。例えば、上述のように、出力は、特別の場合の入力のデフォルト応答によるベクトル出力（例えば、ゼロによる除算は、ＩＮＦを返す）、（例えば、マスクＫ２における）ファーストパスアルゴリズムを用いて処理され得る要素をシグナリングするマスク出力、および計算毎の例外フラグを含み得る。更に、１６０２において、数学関数のうちの少なくともいくつかの結果がトリビアルなものか否かについて判断される。

【０１６４】

１６０３において少なくともいくつかの結果がトリビアルなものであることが判断された場合、次に１６０４において、トリビアルな結果が計算され、返される。いくつかの例において、トリビアルな結果以外の結果は、必要とされない。更なる結果が必要とされる場合、１６０５において１６０２の結果に基づいて、代替的／低速の命令シーケンスにジャンプするか否かについて判断される。唯一の結果が１６０４におけるトリビアルな結果である場合、代替的／低速シーケンスは保証されない。代替的／低速シーケンスが保証される場合、１６０６において、代替的／低速の命令シーケンスが実行される。１６０５に

おける判断が代替的／低速の命令シーケンスは、保証されないというものである場合、１６０７において、通常／高速命令シーケンスが実行される（例えば、通常のオペランド値を用いて除算、平方根、または他の数学関数の結果を計算する）。

【０１６５】

上記の明細書において、本発明の実施形態は、その特定の例示的な実施形態を参照して説明されている。しかし、添付の特許請求の範囲に記載される本発明のより広い趣旨および範囲を逸脱することなく、様々な修正および変更がなされ得ることは明らかであろう。従って、明細書および図面は、限定的意味ではなく、例示的意味で顧慮されるものである。

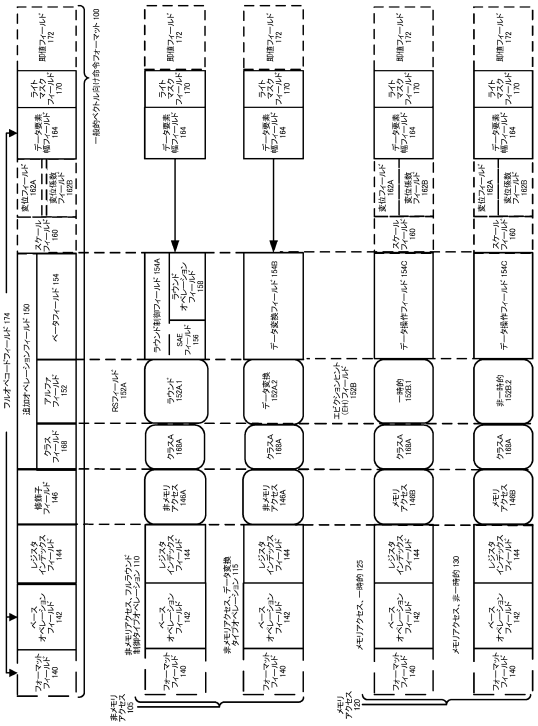
【０１６６】

本発明の実施形態は、上記の様々な段階を含み得る。段階は、汎用または専用プロセッサに段階を実行させるべく用いられ得る機械実行可能命令で実施され得る。あるいは、これらの段階は、段階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ハードウェアコンポーネントにより、またはプログラミング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、実行され得る。

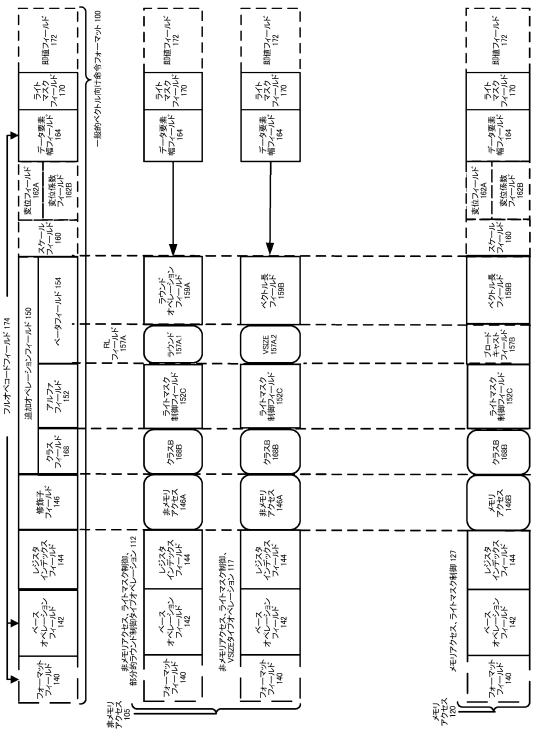
【０１６７】

本明細書において説明されるように、命令は、特定のオペレーションを実行するよう構成され、または予め定められた機能または非一時的コンピュータ可読媒体で実施されるメモリに格納されたソフトウェア命令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を指し得る。従って、図面に示される技術は、１または複数の電子デバイス（例えば、終端局およびネットワーク要素等）上に格納され、実行されるコードおよびデータを用いて実装され得る。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ストレージ媒体（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、電気的、光、音響、もしくは搬送波、赤外線信号、デジタル信号等の他の形態の伝搬信号）等の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格納し、（内部でおよび／またはネットワークを介する他の電子デバイスと）通信する。更に、そのような電子デバイスは通常、１もしくは複数のストレージデバイス（非一時的機械可読ストレージ媒体）、ユーザ入出力デバイス（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）、ならびにネットワーク接続等、１もしくは複数の他のコンポーネントに結合された１もしくは複数のプロセッサのセットを含む。プロセッサおよび他のコンポーネントのセットの結合は通常、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）を介して行われる。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、各々、１または複数の機械可読ストレージ媒体および機械可読通信媒体を表す。従って、所与の電子デバイスのストレージデバイスは通常、当該電子デバイスの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。勿論、本発明の実施形態の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組み合わせを用いて実装され得る。この詳細な説明を通じて、説明の目的のために、多数の具体的な詳細が、本発明の完全な理解を提供するべく記載された。しかし、当業者には本発明がこれら具体的な詳細のいくつかがなくても実施され得ることが明らかであろう。特定の例において、周知の構造および機能は、本発明の主題を不明瞭にするのを避けるべく、精巧詳細に説明されていない。従って、本発明の範囲および趣旨は、以下の特許請求の範囲の観点から判断されるべきである。

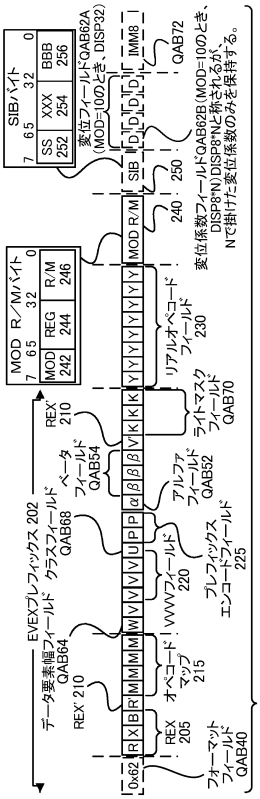
【図１Ａ】



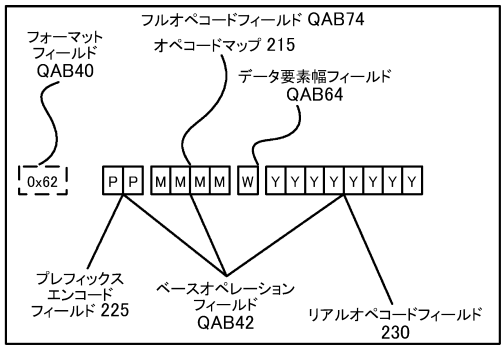
【図１Ｂ】



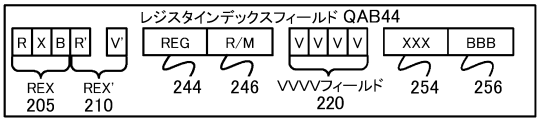
【図２Ａ】



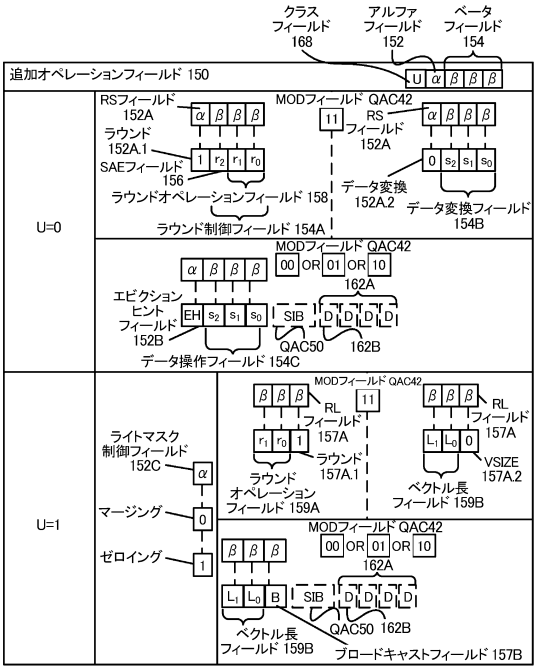
【図２Ｂ】



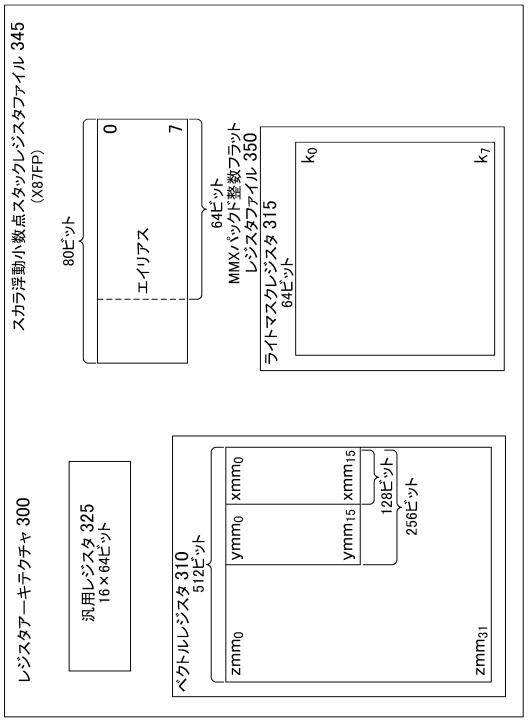
【図２Ｃ】



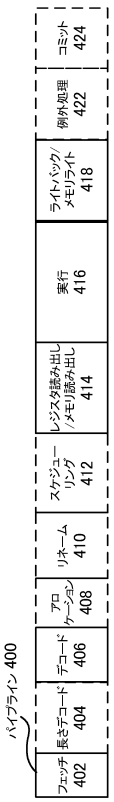
【図２Ｄ】



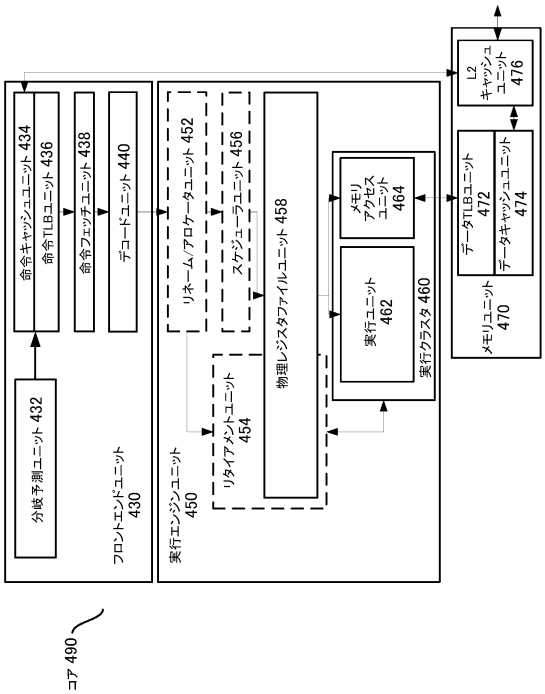
【図３】



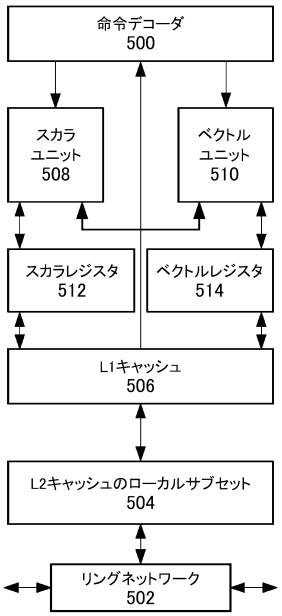
【図４Ａ】



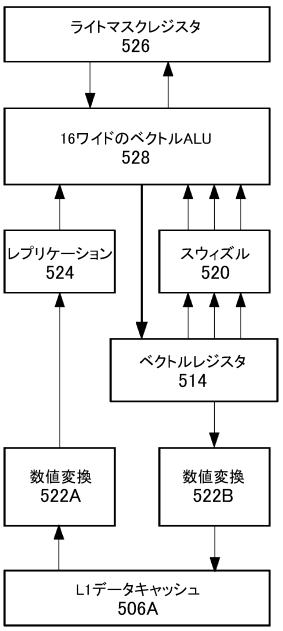
【図４Ｂ】



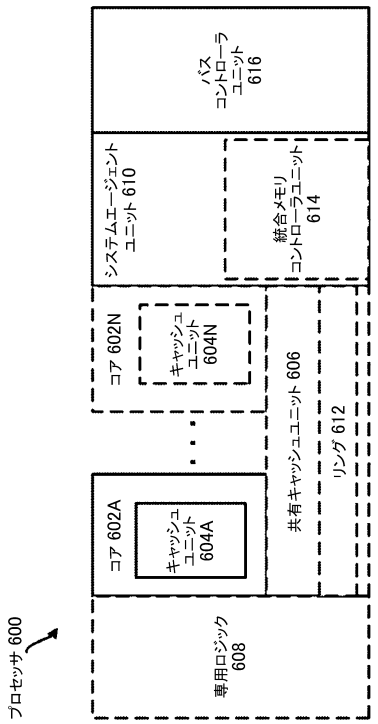
【図５Ａ】


命令デコーダ
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

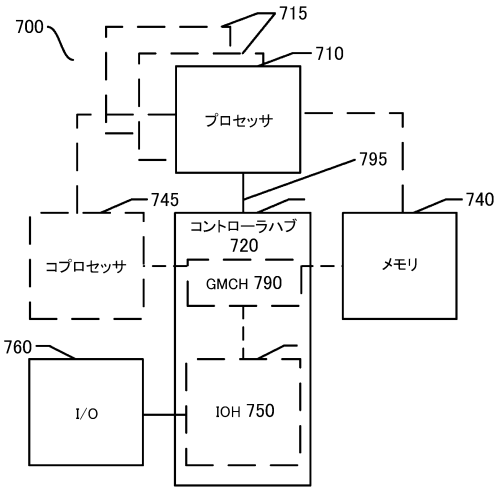
【図５Ｂ】


ライトマスクレジスタ
526
16ワイドのベクトルALU
528
レプリケーション
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

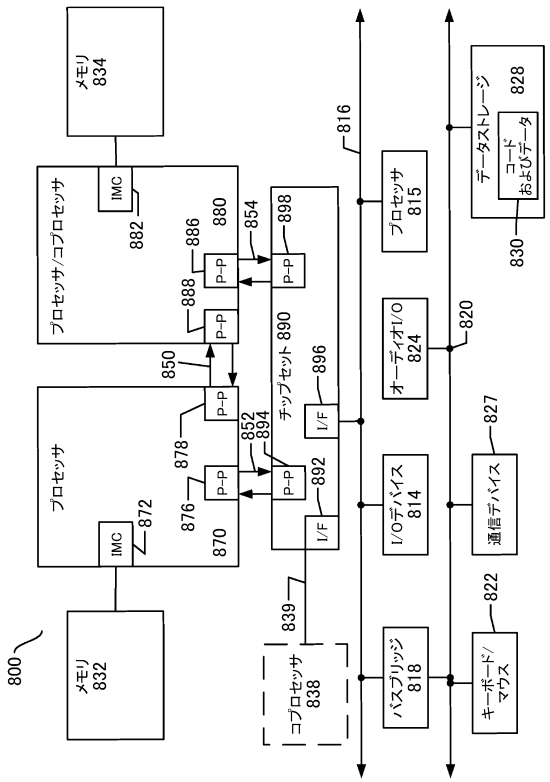
【図６】


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
コア 602A
キャッシュ
ユニット
604A
共有キャッシュユニット 606
リング 612
専用ロジック
608
プロセッサ 600

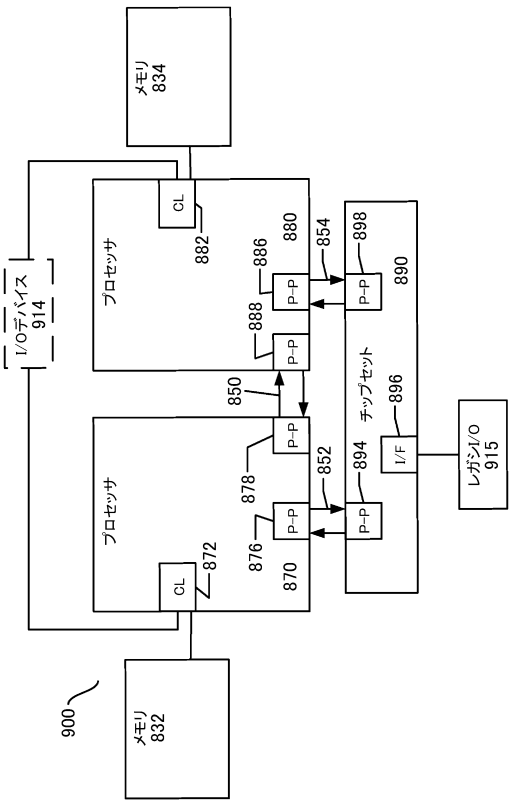
【図７】


700
715
710
プロセッサ
795
745
740
コントローラハブ
720
コプロセッサ
GMCH 790
メモリ
760
I/O
IOH 750

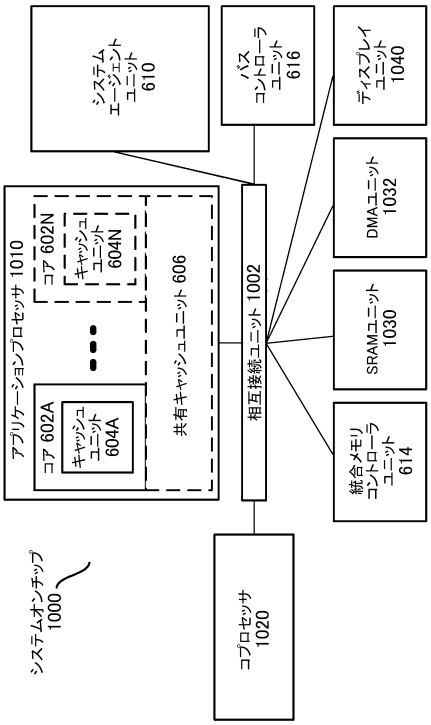
【図８】


800
メモリ 832
プロセッサ
IMC 872
870
876
878
P-P
852
894
892
I/F
チップセット 890
896
850
プロセッサ/コプロセッサ
IMC 882
886
888
880
854
898
メモリ 834
コプロセッサ 838
839
816
バスブリッジ 818
I/Oデバイス 814
オーディオI/O 824
820
通信デバイス 827
822
キーボード/マウス
プロセッサ 815
データストレージ 828
コードおよびデータ 830

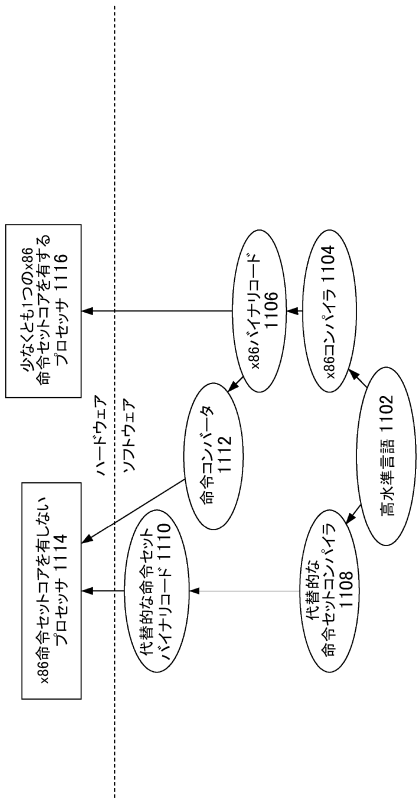
【図９】


900
メモリ 832
プロセッサ
CL 872
870
876
878
P-P
852
894
892
I/F
チップセット 890
896
850
プロセッサ
CL 882
886
888
880
854
898
890
メモリ 834
I/Oデバイス 914
レガシI/O 915

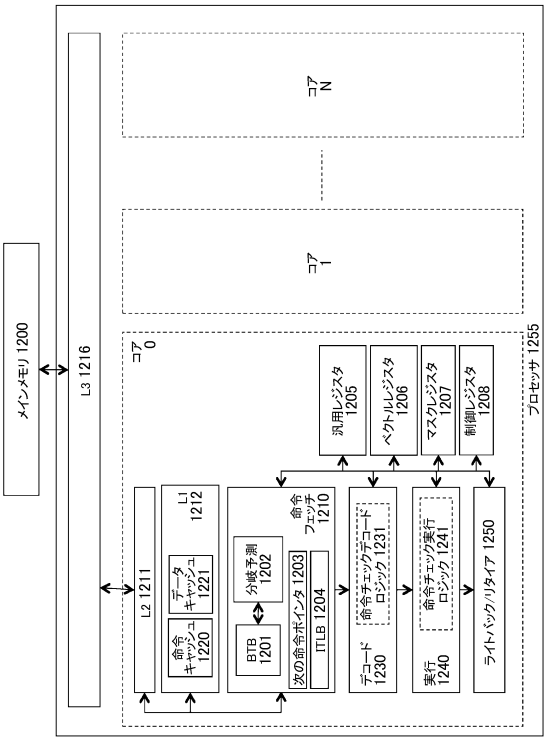
【図１０】


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
相互接続ユニット 1002
バスコントローラユニット 616
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

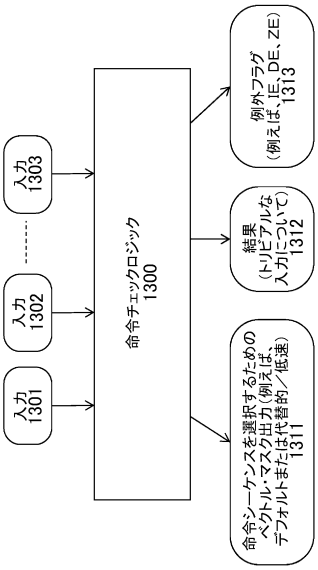
【図１１】


少なくともひとつのx86命令セットコアを有するプロセッサ 1116
x86バイナリコード 1106
x86コンパイラ 1104
ハードウェア
ソフトウェア
命令コンバータ 1112
高水準言語 1102
x86命令セットコアを有しないプロセッサ 1114
代替的な命令セットバイナリコード 1110
代替的な命令セットコンパイラ 1108

【図１２】



【図１３】



【図１４】

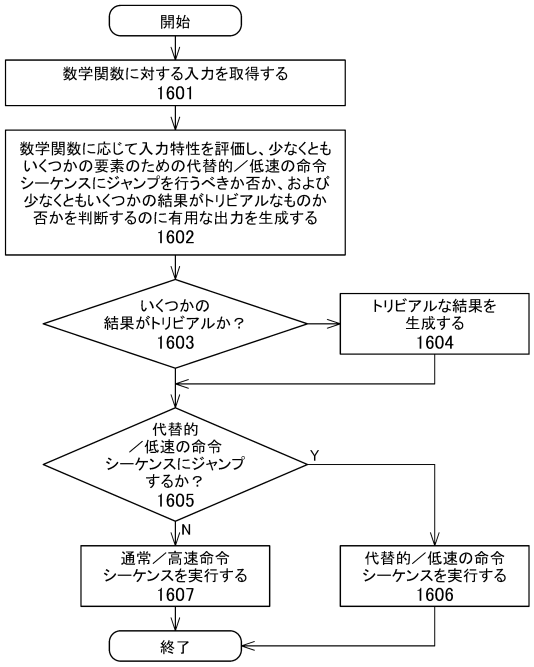
x\y	±0	±denorm	±norm	±inf	qNaN	sNaN
±0	(0, QtieN_Ind,#IE)	(0, ±0, #DE)	(0, ±0, None)	(0, ±0, None)	(0,y,None)	(0,QtieN(y),#IE)
±denorm	(0,±Inf,#ZE)	(0,+1,#DE)	(0,+1,#DE)	(0, ±0, #DE)	(0,y,None)	(0,QtieN(y),#IE)
±norm	(0,±Inf,#ZE)	(0,+1, #DE)	See Below	(0, ±0, None)	(0,y,None)	(0,QtieN(y),#IE)
±Inf	(0,±Inf,None)	(0,±Inf,#IE)	(0,±Inf,None)	(0,QtieN_Ind,#IE)	(0,y,None)	(0,QtieN(y),#IE)
qNaN	(0,x,None)	(0,x,None)	(0,x,None)	(0,x,None)	(0,x,None)	(0,x,#IE)
sNaN	(0,QtieN(x), #IE)	(0,QtieN(x),#IE)	(0,QtieN(x),#IE)	(0,QtieN(x),#IE)	(0,QtieN(x),#IE)	(0,QtieN(x),#IE)

Fig. 14

【図１５】

入力(x)	kdst.vdst.flags	タイプ
−norm(including −infinity)	(0,NaN,#IE)	特殊数
−denorm	(0,NaN,#IE)	特殊数
−0	(0,x,None)	特殊数
+0	(0,x,None)	特殊数
+denorm	(0,+1.0, #DE)	スローフロー
+infinity	(0,x,None)	特殊数
qNaN	(0,x,None)	特殊数
sNaN	(0,qNaN(x), #IE)	特殊数
BiasExponent(x) ≤ 2*precision[a]	(0,+1.0, None)	スローフロー
BiasExponent(x) = 2*bias[b]	(0,+1.0, None)	スローフロー
Other	(1,+0.0, None)	ファーストフロー

【図１６】


開始
数学関数に対する入力を取得する
1601
数学関数に応じて入力特性を評価し、少なくともいくつかの要素のための代替的／低速の命令シーケンスにジャンプを行うべきか否か、および少なくともいくつかの結果がトリビアルなものか否かを判断するのに有用な出力を生成する
1602
いくつかの結果がトリビアルか？
1603
トリビアルな結果を生成する
1604
代替的／低速の命令シーケンスにジャンプするか？
1605
Y
N
通常／高速命令シーケンスを実行する
1607
代替的／低速の命令シーケンスを実行する
1606
終了

フロントページの続き

(51)Int.Cl.		FI		
G06F 9/32 (2006.01)		G06F	9/302	C
		G06F	9/318	A
		G06F	9/32	320F

(72)発明者 ハネク、ロバート エヌ.
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 ファーガスン、ワーレン イー.
アメリカ合衆国 95054 カリフォルニア州・サンタクララ・ミッション カレッジ ブーレバード・2200 インテル・コーポレーション内
(72)発明者 バーラミ、タラネー
アメリカ合衆国 95054 カリフォルニア州・サンタクララ・ミッション カレッジ ブーレバード・2200 インテル・コーポレーション内
(72)発明者 テベット、アヴィ エイ.
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 ブラッドフォード、デニス アール.
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
(72)発明者 フェリー、マイケル
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
(72)発明者 ジャン、ジンヴェ
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 豊田 真弓

(56)参考文献 特表2002－508864（JP, A）
特開昭62－236073（JP, A）
特開2001－243063（JP, A）
特開平10－143355（JP, A）
特開平07－248897（JP, A）
特開2000－122850（JP, A）
特開2000－040081（JP, A）

(58)調査した分野(Int.Cl., DB名)
G06F 7／57
G06F 9／30
G06F 9／302
G06F 9／318
G06F 9／32
G06F 9／38