(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6340097号
(P6340097)

(45)発行日　平成30年6月6日(2018.6.6)　(24)登録日　平成30年5月18日(2018.5.18)

(51)Int.Cl.		F I		
G06F 9/315	(2006.01)	G06F	9/30	340D
G06F 9/32	(2006.01)	G06F	9/32	330A
G06F 9/38	(2006.01)	G06F	9/38	370A
G06F 17/16	(2006.01)	G06F	17/16	E
G06F 8/41	(2018.01)	G06F	9/44	322G

請求項の数 12　(全 31 頁)

(21)出願番号　特願2017-7593(P2017-7593)
(22)出願日　平成29年1月19日(2017.1.19)
(62)分割の表示　特願2015-531915(P2015-531915)の分割
原出願日　平成25年6月12日(2013.6.12)
(65)公開番号　特開2017-107579(P2017-107579A)
(43)公開日　平成29年6月15日(2017.6.15)
審査請求日　平成29年1月21日(2017.1.21)
(31)優先権主張番号　13/630,118
(32)優先日　平成24年9月28日(2012.9.28)
(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　プロトニコフ、ミカイル
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

最終頁に続く

(54)【発明の名称】リードマスク及びライトマスクにより制御されるベクトル移動命令

(57)【特許請求の範囲】
【請求項１】
第１のマスクレジスタ、第２のマスクレジスタ、第１のソースレジスタおよび第２のソースレジスタを特定するための複数のフィールドを有する命令をデコードするデコード回路と、
デコードされた前記命令を実行し、前記第１のソースレジスタにおける１または複数のターゲットデータエレメントを前記第２のソースレジスタにおける１または複数のソースデータエレメントで置き換える実行回路と、を備え、
各ソースデータエレメントの位置は、第２の値を有する前記第２のマスクレジスタ内のマスク値に対応し、各ターゲットデータエレメントは、第１の値を有する前記第１のマスクレジスタ内のマスク値に対応する
装置。
【請求項２】
前記第１の値を有する前記第１のマスクレジスタの前記マスク値の各々に対し、前記実行回路は、前記１または複数のソースデータエレメントのうちの１つとして、前記第２のマスクレジスタ内の前記第２の値の、対応するマスク値を有するデータエレメントを探索するために前記第２のソースレジスタを検索する
請求項１に記載の装置。
【請求項３】
前記実行回路は、前記第１のソースレジスタ内の１または複数のデータエレメントに対

して再帰計算を実行し、前記第１のソースレジスタ内に前記再帰計算の複数の結果を蓄積する
請求項１または２に記載の装置。
【請求項４】
前記実行回路は、前記第１のソースレジスタ内の１または複数のデータエレメントがさらに計算を必要としなくなるまで、前記第１のソースレジスタ内の１または複数のデータエレメントに対して再帰計算を実行し、前記第１のマスクレジスタを用いて、前記１または複数のデータエレメントが前記１または複数のターゲットデータエレメントであると示す
請求項１から３の何れか一項に記載の装置。
【請求項５】
前記第１の値は前記第２の値の逆である
請求項１から４の何れか一項に記載の装置。
【請求項６】
前記第１の値は前記第２の値と同一である
請求項１から４のいずれか一項に記載の装置。
【請求項７】
第１のマスクレジスタ、第２のマスクレジスタ、第１のソースレジスタおよび第２のソースレジスタを特定するための複数のフィールドを有する命令を、デコーダ回路を用いてデコードする段階と、
デコードされた前記命令を、実行回路を用いて実行し、前記第１のソースレジスタにおける１または複数のターゲットデータエレメントを前記第２のソースレジスタにおける１または複数のソースデータエレメントで置き換える段階と、を備え、
各ソースデータエレメントの位置は、第２の値を有する前記第２のマスクレジスタ内のマスク値に対応し、各ターゲットデータエレメントは、第１の値を有する前記第１のマスクレジスタ内のマスク値に対応する
方法。
【請求項８】
前記第１の値を有する前記第１のマスクレジスタの前記マスク値の各々に対し、前記実行回路が前記１または複数のソースデータエレメントのうちの１つとして、前記第２のマスクレジスタ内の前記第２の値の、対応するマスク値を有するデータエレメントを探索するために前記第２のソースレジスタを検索する
請求項７に記載の方法。
【請求項９】
前記実行回路は、前記第１のソースレジスタ内の１または複数のデータエレメントに対して再帰計算を実行し、前記第１のソースレジスタ内に前記再帰計算の複数の結果を蓄積する
請求項７または８に記載の方法。
【請求項１０】
前記実行回路は、前記第１のソースレジスタ内の１または複数のデータエレメントがさらに計算を必要としなくなるまで、前記第１のソースレジスタ内の１または複数のデータエレメントに対し再帰計算を実行し、前記第１のマスクレジスタを用いて、前記１または複数のデータエレメントが前記１または複数のターゲットデータエレメントであると示す
請求項７から９の何れか一項に記載の方法。
【請求項１１】
前記第１の値は前記第２の値の逆である
請求項７から１０の何れか一項に記載の方法。
【請求項１２】
前記第１の値は前記第２の値と同一である
請求項７から１０の何れか一項に記載の方法。

【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、プロセッサまたは他のプロセッシングロジックにより実行されると、論理演算、数学的演算、または他の関数演算を実行する、プロセッシングロジック、マイクロプロセッサ及び関連する命令セットアーキテクチャの分野に関する。
【背景技術】
【０００２】
命令セット、すなわち命令セットアーキテクチャ（ＩＳＡ）は、プログラミングに関連するコンピューターアーキテクチャの一部分であり、複数のネイティブデータタイプ、複数の命令、レジスタアーキテクチャ、複数のアドレッシングモード、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）を含みうる。命令という用語は、本明細書においては、マクロ命令をデコードするプロセッサのデコーダの結果であるマイクロ命令またはマイクロオペレーション（マイクロ－ｏｐｓ）とは対照的に、実行すべく、プロセッサ（または、命令を、プロセッサにより処理される１または複数の他の命令へと、変換（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）、モーフィング、エミュレート、または別の方法でコンバートする命令変換部）に提供される命令である、マクロ命令を概して指す。
【０００３】
ＩＳＡは、命令セットを実装するプロセッサの内部設計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複数のプロセッサは共通の命令セットを共有できる。例えば、Ｉｎｔｅｌ（登録商標）Ｃｏｒｅ（商標）プロセッサ及びカリフォルニア州サニーベールのアドバンスト・マイクロ・デバイセズの複数のプロセッサは、ほとんど同じバージョンのｘ８６命令セット（複数のより新しいバージョンと共に追加された幾つかの拡張を有する）を実装しているが、内部設計は異なる。例えば、ＩＳＡの同一のレジスタアーキテクチャは、複数の専用の物理的レジスタ、レジスタリネーミングメカニズムを用いる、１または複数の動的に割り当てられた物理的レジスタ等を含むよく知られた技法を用いて、異なるマイクロアーキテクチャに異なる方法で実装されうる。
【０００４】
多くの最新のＩＳＡは単一命令、複数データ（ＳＩＭＤ）オペレーションをサポートする。たった１つまたは２つのデータエレメントに対して実行するスカラ命令ではなく、ベクトル命令(パックドデータ命令またはＳＩＭＤ命令とも称される)は、同時に、またはパラレルに、複数のデータエレメントに、または、データエレメントの複数のペアに対して実行しうる。プロセッサは、ベクトル命令に応答するパラレルな実行ハードウェアを有し得て、同時に、またはパラレルに複数のオペレーションを実行する。ＳＩＭＤオペレーションは、１つのオペレーションにおいて、１つのベクトルレジスタまたはメモリ位置内にパックされた複数のデータエレメントに対して実行する。これらのデータエレメントは、パックドデータまたはベクトルデータと称される。ベクトルエレメントのそれぞれは、他とは別個に独立して扱われうる、別個で独立した１つのデータ（例えば、ピクセルの色、など）を表しうる。
【０００５】
幾つかのシナリオにおいて、ＳＩＭＤオペレーションは、独立したベクトルデータエレメントに対して再帰的に実行し得て、その際データエレメントが異なればイタレーション回数は異なる。したがって、幾つかのデータエレメントに対する計算は、その他のデータエレメントがさらに多くのイタレーションを必要とする一方で、終了しうる。再帰計算の１つの例が、ＷＨＩＬＥループオペレーションである。この例において、Ｎエレメントのデータアレイ Ｘ［ｉ］（ｉ＝０，...，Ｎ－１）は、条件（Ｘ［ｉ］）が真（満たされる）である間、再帰計算が施される。Ｘ［ｉ］のこの計算は、条件（Ｘ［ｉ］）が偽となる場合、終了する。その状況の一例は、Ｘ［ｉ］＞０でありうる。

【数１】

```
for (i=0; i<N; i++){
    while (condition(X[i])){
        X[i]=computation(X[i]);}}
```

【０００６】

上記計算は、Ｘ［ｉ］の異なる複数のデータエレメントでＷＨＩＬＥループのイタレーション回数が異なれば、容易にベクトル化され得ない。１つのアプローチとしては、プロセッサが、その条件を満たさないそれらのエレメントに対して計算を実行し、それから、それらのエレメントから得られた結果を捨てるというアプローチが考えられる。しかしながら、プロセッサがこれらのエレメントに対して不必要な計算を実行するだけでなく、これらのエレメントが占めるベクトルレジスタスロットを使用することもできないので、このアプローチは低効率である。

【図面の簡単な説明】

【０００７】

複数の実施形態は例として示されるものであって、添付の複数の図面における複数の図に限定されるものではない。

【０００８】

【図１】一実施形態による、複数のベクトルレジスタ及び複数のマスクレジスタを含む命令処理装置のブロック図である。

【図２】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。

【図３】一実施形態による、ベクトル演算シーケンスの一例を示す。

【図４Ａ】一実施形態による、プロセッサにベクトルレジスタ及びマスクレジスタ上で複数のオペレーションを実行させる複数の命令の疑似コードの一例を示す。

【図４Ｂ】一実施形態による、図４Ａの複数の命令を使用するためのコードセグメントの一例を示す。

【図５Ａ】一実施形態による、マスク更新命令及びベクトル移動命令を用いるコードセグメントに応答して実行される複数のオペレーションを例示するフロー図である。

【図５Ｂ】一実施形態による、マスク更新命令に応答して実行されるオペレーションを例示するフロー図である。

【図５Ｃ】一実施形態による、ベクトル移動命令に応答して実行されるオペレーションを例示するフロー図である。

【図６】一実施形態による、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変換器の使用を例示するブロック図である。

【図７Ａ】一実施形態による、インオーダ及びアウトオブオーダパイプラインのブロック図である。

【図７Ｂ】一実施形態による、インオーダ及びアウトオブオーダコアのブロック図である。

【図８Ａ】一実施形態による、より具体的な例示的インオーダコアアーキテクチャのブロック図である。

【図８Ｂ】一実施形態による、より具体的で例示的なインオーダコアアーキテクチャのブロック図である。

【図９】一実施形態による、プロセッサのブロック図である。

【図１０】一実施形態による、システムのブロック図である。

【図１１】一実施形態による、第２のシステムのブロック図である。

【図１２】本発明のある実施形態による、第３のシステムのブロック図である。

【図１３】一実施形態による、システムオンチップ（ＳｏＣ）のブロック図である。

【発明を実施するための形態】

【０００９】
以下の説明において、数々の具体的な詳細を記載する。しかしながら、本発明の複数の実施形態は、これらの具体的な詳細なしに実現可能であることが理解される。他の例において、公知の回路、構造及び技術については、説明を理解する妨げにならないよう、詳細には示していない。
【００１０】
本明細書にて説明される複数の実施形態は、複数の独立したデータエレメントに対する再帰的なベクトル演算の効率性を向上させる複数の命令を提供する。その複数の命令は、ベクトルレジスタのペア及びマスクレジスタのペアを用いて再帰的なベクトル演算を実行する。第１のベクトルレジスタは、ベクトル演算結果を蓄積するアキュムレータとして機能し、第２のベクトルレジスタは、第１のベクトルレジスタの複数の未使用スロット（未使用または終了した複数のデータエレメント位置）を埋める複数の新しいデータエレメントを提供する。マスクレジスタは、その対応する複数のベクトルレジスタにおけるどのデータエレメントがさらに計算を必要としているかを示すべく用いられる。
【００１１】
一実施形態において、第１のベクトルレジスタ（すなわちアキュムレータ）は、レジスタが完全ベクトルで満たされるまで、複数の入力データエレメントを蓄積する。プロセッサはそれから非マスク（すなわちデンス（ｄｅｎｓｅ）な）ベクトル演算を用いて、これらのデータエレメントに対して計算を実行する。計算後、アキュムレータ中の幾つかの（計算が終了した）エレメントは、メモリまたは他のストレージ場所に送り戻され得て、他の（計算が終了していない）エレメントは、イタレーション回数を追加すべく、アキュムレータ中に保たれ得る。アキュムレータ中の、計算が終了したデータエレメント位置を、同一の再帰計算もまた必要とする複数の新しいデータエレメントが使用できる。
【００１２】
ＲＷＭＡＳＫＵＰＤＡＴＥ及びＳＰＡＲＳＥＭＯＶという２つの命令を本明細書にて説明する。これらの命令は、多くのシナリオにおいて、ベクトル化の効率性を向上させる。例えば、一シナリオにおいて、入力データエレメントは、１または複数のスパース（ｓｐａｒｓｅ）なベクトルデータセットに由来する。そのベクトルデータセットのそれぞれは、アキュムレータ全体（すなわち、第１のベクトルレジスタ）を埋めるのに十分なエレメントを有していない。さらに、異なるデータセットからの入力データエレメントは、計算中、異なるイタレーション回数を必要とし得る。したがって、これ以上計算を必要としない、それらのデータエレメントからの未使用スロットがアキュムレータに残される。本明細書にて説明される命令は、これらの未使用スロットが複数の有用なエレメントで埋められるようにするので、完全ベクトルに対して再帰計算が可能となる。以下でさらに詳細に説明するように、ＳＰＡＲＳＥＭＯＶ命令は、第２のベクトルレジスタからアキュムレータへ有用なデータエレメント（すなわち、計算を必要とするデータエレメント）を移動させるベクトル移動命令である。ＲＷＭＡＳＫＵＰＤＡＴＥ命令は、（第２のベクトルレジスタに関連付けられる）リードマスクレジスタ及び（アキュムレータに関連付けられる）ライトマスクレジスタの両方を更新し、これら２つのベクトルレジスタにおける有用なデータエレメントの位置を特定する。
【００１３】
ＳＰＡＲＳＥＭＯＶと組み合わせてＲＷＭＡＳＫＵＰＤＡＴＥを使用すれば、再帰計算法において必要になる命令の総数を低減し、第２のベクトルレジスタにおける複数の有用なデータエレメント（すなわち、複数のソースデータエレメント）の数が、第１のベクトルレジスタにおける未使用スロット（すなわち、ターゲット位置）の数と一致しない、オーバーフロー及びアンダーフローの場合を単純化する。更新されたリード及びライトマスクは、２つのベクトルレジスタ間のデータの移動を制御すべく用いられる。特に、０のライトマスクビットは、アキュムレータ中のターゲット位置を特定すべく用いられ、１のリードマスクビットは、第２のベクトルレジスタ中の複数のソースデータエレメントを特定すべく用いられる。ターゲット位置を特定すべく、反転させたライトマスクビットを用い

れば、スパースかつ再帰的な計算のベクトル化におけるデータ蓄積を単純化する。

【００１４】

図１は、ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令を含む命令を実行すべく動作可能な回路を含む実行ユニット１４０を有する命令処理装置１１５の実施形態のブロック図である。いくつかの実施形態では、命令処理装置１１５は、電子システムの、プロセッサ、マルチコアプロセッサのプロセッサコア、またはプロセッシングエレメントでありうる。

【００１５】

デコーダ１３０は、高レベルの機械語命令またはマクロ命令の形態の、入力命令を受信し、それらをデコードし、元の高レベルの命令を反映する、及び／または、元の高レベルの命令から得られる、低レベルのマイクロオペレーション、マイクロコードエントリーポイント、マイクロ命令、または、他の低レベルの命令または制御信号を生成する。低レベルの命令または制御信号は、低レベル（例えば、回路レベルまたはハードウェアレベル）のオペレーションを通して、高レベルの命令のオペレーションを実行しうる。デコーダ１３０は、様々な異なるメカニズムを用いて実装されうる。適切なメカニズムの複数の例は、限定されるものではないが、マイクロコード、ルックアップテーブル、ハードウェア実装、プログラム可能ロジックアレイ（ＰＬＡ）、当技術分野において既知のデコーダを実装すべく用いられる他のメカニズム等を含む。

【００１６】

デコーダ１３０は、キャッシュ１１０、メモリ１２０、または他のソース用の、入力命令を受信しうる。デコードされた命令は、実行ユニット１４０へ送られる。実行ユニット１４０はデコーダ１３０から、受信した命令を反映する、または受信した命令から得られる、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ーポイント、マイクロ命令、他の命令、または他の制御信号を受信しうる。実行ユニット１４０は、レジスタファイル１７０、キャッシュ１１０、及び/またはメモリ１２０からのデータ入力を受信し、かつ、レジスタファイル１７０、キャッシュ１１０、及び/またはメモリ１２０へのデータ出力を生成する。

【００１７】

一実施形態において、レジスタファイル１７０は、レジスタとも称される、アーキテクチャレジスタを含む。別途特定しないか、または間違いなく明らかでない限り、複数のアーキテクチャレジスタ、レジスタファイル、及び複数のレジスタという文言は、ソフトウェア、及び／またはプログラマがアクセスできる（例えば、ソフトウェアビジブルな）複数のレジスタを、及び／またはオペランドを特定すべくマクロ命令が特定する複数のレジスタ、を指すべく本明細書中で用いられる。これらのレジスタは、与えられたマイクロアーキテクチャ上の他の複数の非アーキテクチャレジスタ（例えば、複数のテンポラリレジスタ、複数のリオーダバッファ、複数のリタイヤメントレジスタ等）とは対照的である。

【００１８】

説明を分かりにくくしないように、比較的簡易な命令処理装置１１５を示し、説明する。他の複数の実施形態においては、１より多くの実行ユニットを有しうることが理解されよう。例えば、装置１１５は、例えば、複数の演算ユニット、複数の演算ロジックユニット（ＡＬＵ）、複数の整数ユニット、複数の浮動小数点ユニットなどの、複数の異なるタイプの実行ユニットを含み得る。命令処理装置またはプロセッサの、さらなる他の複数の実施形態は、複数のマルチコア、複数の論理プロセッサ、または複数の実行エンジンを有しうる。命令処理装置１１５の、幾つかの実施形態を、図７から図１３に関して提供する。

【００１９】

一実施形態によると、レジスタファイル１７０は、１セットのベクトルレジスタ１７５及び１セットのマスクレジスタ１８５を備え、その両方は、ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令の複数のオペランドをストアする。各ベクトルレジスタ１７５は、５１２ビット幅、２５６ビット幅、１２８ビット幅であり得て、また、異なるベ

クトル幅が用いられうる。各マスクレジスタ１８５は、幾つかのマスクビットを含み、その各マスクビットは、複数のベクトルレジスタ１７５のうちの１つのベクトルレジスタの１つのデータエレメントに対応している。各マスクビットは、ベクトルレジスタのデータエレメントをマスクすべく用いられるので、６４ビットのマスクレジスタは、５１２ビットのレジスタの６４個の８ビットデータエレメントをマスクすべく用いられ得る。異なる幅（例えば、２５６ビットまたは１２８ビット）、及び異なるサイズ（例えば、１６ビット、３２ビットまたは６４ビット）の複数のデータエレメントを有するベクトルレジスタに対して、異なる数のマスクビットをベクトル演算に関連して用いてもよい。

【００２０】

図２は、本明細書において説明される複数の命令をサポートする、根本的なレジスタアーキテクチャ２００のある実施形態を示す。 レジスタアーキテクチャ２００は、ｘ８６、ＭＭＸ（商標）、ストリーミングＳＩＭＤ拡張（ＳＳＥ）命令、ＳＳＥ２命令、ＳＳＥ３命令、ＳＳＥ４．１命令、及びＳＳＥ４．２命令を含む命令セットも、アドバンストベクトル拡張命令（ＡＶＸ）（ＡＶＸ１及びＡＶＸ２）を指すＳＩＭＤ拡張命令の追加的なセットも実装する、Ｉｎｔｅｌ（登録商標）Ｃｏｒｅ（商標）プロセッサに基づいている、しかしながら、異なるレジスタ長、異なるレジスタタイプ、及び／またはレジスタの異なる個数をサポートする、異なるレジスタアーキテクチャもまた用いることができるということが理解されよう。

【００２１】

図示す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２１０がある。これらのレジスタはｚｍｍ０からｚｍｍ３１と参照される。下側の１６個のｚｍｍレジスタの下位２５６ビットは、レジスタｙｍｍ０－１６に重ねられている。下側の１６個のｚｍｍレジスタの下位１２８ビット（ｙｍｍレジスタの下位１２８ビット）は、レジスタｘｍｍ０－１５に重ねられている。図示する実施形態において、８個のマスクレジスタ２２０（ｋ０からｋ７）があり、それぞれ６４ビットの長さがある。代替的な実施形態においては、マスクレジスタ２２０は１６ビット幅である。

【００２２】

図示する実施形態において、レジスタアーキテクチャ２００は、１６個の６４ビット汎用（ＧＰ）レジスタ２３０をさらに備える。ある実施形態において、それらは、現存するｘ８６アドレッシングモードと共に用いられ、メモリオペランドをアドレッシングする。その実施形態は、ＲＦＬＡＧＳレジスタ２６０、ＲＩＰレジスタ２７０及びＭＸＣＳＲレジスタ２８０も図示している。

【００２３】

その実施形態は、スカラ浮動小数点（ＦＰ）スタックレジスタファイル（ｘ８７スタック）２４０も図示し、そのファイル上で、ＭＭＸパックド整数フラットレジスタファイル２５０がエイリアスされる。図示する実施形態において、ｘ８７スタックは、ｘ８７命令セットの拡張を使用して、３２／６４／８０ビットの浮動小数点データ上で、複数のスカラ浮動小数点オペレーションを実行すべく用いられる８個のエレメントのスタックである。一方、ＭＭＸレジスタは、ＭＭＸとｘｍｍレジスタとの間で実行される幾つかのオペレーションに対する複数のオペランドを保持すべく、また同様に、６４ビットのパックド整数データ上で複数のオペレーションを実行すべく用いられる。

【００２４】

本発明の複数の代替的実施形態は、より幅の広い、または狭いレジスタを使用しうる。

【００２５】

追加的に、本発明の複数の代替的実施形態は、よりたくさんの、より少ない、または異なる、レジスタファイル及びレジスタを使用しうる。

【００２６】

図３は、独立したデータエレメントに対する計算を効率的にベクトル化すべく、プロセッサ（例えば、命令処理装置１１５）が実行するオペレーションの一例を図示している。説明を単純化するべく、この例における各ベクトルレジスタは、８つのデータエレメント

のみを有するように示される。複数の代替的実施形態においては、複数のベクトルレジスタ内に、異なる数のデータエレメントを有してよい。ベクトルレジスタは、１２８ビット幅、２５６ビット幅、または５１２ビット幅（例えば、図２のｘｍｍ、ｙｍｍ、またはｚｍｍレジスタ）であることが可能であり、または異なる幅が用いられうる。各ベクトルレジスタには８つのデータエレメントがあるので、各ベクトルレジスタに関連して８つのマスクビットのみが用いられる。

【００２７】

この例において、ベクトルレジスタＶ１はアキュムレータとして用いられ、ベクトルレジスタＶ２は、新しいデータエレメントをＶ１に提供すべく用いられる。マスクレジスタＫ１（ライトマスク）及びＫ２（リードマスク）はそれぞれ、Ｖ１及びＶ２のデータエレメントをマスクすべく用いられる。この例において、０のマスクビットは、対応するデータエレメントが計算からマスクされる（すなわち、さらなる計算は必要ない）ことを示し、１のマスクビットは、対応するデータエレメントがさらなる計算を必要とすることを示す。代替的実施形態においては、マスクビット値の意味は逆であり得て、例えば、１のマスクビットは、対応するデータエレメントがさらなる計算を必要としないことを示すべく用いられ、０のマスクビットは、対応するデータエレメントがさらなる計算を必要とすることを示すべく用いられうる。

【００２８】

初めに、アキュムレータＶ１は、入力ベクトル：Ａ及びＢとして２セットのデータをストアし、そのそれぞれは、スパースなデータアレイの一部でありうると仮定する。Ａｊ及びＢｊの下付きの文字ｊは、データエレメントが受けたイタレーション回数を示す。例えば、Ａ０は、あらゆるイタレーションの前のエレメントＡであり、Ａ１は第１のイタレーション３１０後のエレメントである。説明を単純化すべく、同一イタレーション内の同一データセットからの異なるデータエレメントは同一の識別子を有するように示される。例えば、入力ベクトルの、位置０のＡ０及び位置２のＡ０は、２つの異なるエレメントであり、同一または異なる値を有し得て、入力ベクトルの位置１のＢ０及び位置３のＢ０は、２つの異なるエレメントであり、同一または異なる値を取りうる。マスクレジスタＫ１の複数のマスクビットの初期値は全て１であり、Ｖ１の初期の入力ベクトルが完全ベクトルであり、かつＶ１の全エレメントがベクトル演算の第１のイタレーション３１０に関与できることを示している。

【００２９】

この例において、各イタレーションは、再帰的なベクトル演算が実行されるＷＨＩＬＥループのイタレーションを表している。第１のイタレーション３１０の後、アキュムレータＶ１は複数のＡ及び複数のＢからなる１セットを含み、ここで、下付き文字は、これらのエレメントが第１のイタレーションを終了したことを示す。ＡのエレメントはＷＨＩＬＥループの１回のイタレーションを、Ｂのエレメントは２回のイタレーションを必要とすると仮定しよう。そうすると、ＷＨＩＬＥループの１回のイタレーション後、Ｂエレメントに対してはもう１回イタレーションが必要な一方、Ａエレメントに対する計算は終了した。この時点で、複数のＡエレメントのそれぞれに対する条件は偽であり（なぜなら、さらに計算するための条件を満たさないから）、複数のＢエレメントのそれぞれに対する条件は真である（なぜなら、さらに計算するための条件を満たすから）。したがって、複数のＡに対応するそれらのマスクビットに対し、Ｋ１の複数のマスクビット値は０に設定され、複数のＢに対応するそれらのマスクビットに対しては１が設定される。

【００３０】

一実施形態において、０のマスクビットは、対応するエレメント位置における結果が、ベクトルレジスタ全体（この場合、Ｖ１）に対するベクトル演算後に捨てられるであろうことを示す。複数の代替の実施形態において、０のマスクビットは、対応するエレメント位置に対する計算が以後実行されず、故にそのエレメント位置は使用されないことを示す。いずれのシナリオにおいても、アキュムレータＶ１に複数のＡ１を保持しておくと、ベクトルリソースの無駄使いであるし、ベクトル演算の効率性を低減する。よって、本発明

の一実施形態によると、第２のベクトルレジスタＶ２が用いられ、複数のＡ１が残した未使用スロット（すなわち、データエレメント位置）を埋めるべく、Ｖ１に新しいデータエレメントを提供する。複数のＡ１のデータエレメントは、メモリ、キャッシュ、または他のデータストレージに保存可能である。

【００３１】

図３の例において、ベクトルレジスタＶ２は、別のスパースなベクトルアレイの一部でありうるデータセットＣのエレメントをストアする。「＊」でマーク付けされたＶ２中の位置は、「ドントケア」を表し、再帰的なベクトル演算の目的のための有用なデータエレメントを含まないことを意味する。Ｃの各データエレメントがＷＨＩＬＥループの３回のイタレーションを受ける必要があると仮定しよう。Ｃのエレメントに代えて、または追加して、Ｖ２は、ＷＨＩＬＥループ（及び、故にさらなる計算）の１または複数のイタレーション受ける必要がある、Ａ及び／またはＢ（例えば、複数の$Ａ_0$，複数の$Ｂ_0$、及び／または複数のＢ）の複数の新しいデータエレメントを提供しうる。さらなる計算を必要とするＶ２中のこれらのデータエレメントは、「ソースデータエレメント」と称される。Ｖ２中のこれらのソースデータエレメントは、複数のＡ（「ターゲットデータエレメント」と称される）が残したＶ１中の未使用スロットを埋めることができる。説明を簡単にすべく、さらなる計算を必要とする、Ｖ１及び／またはＶ２中のデータエレメントを「有用なデータエレメント」と呼ぶ。したがって、併合操作３２０が、Ｖ１及びＶ２中の複数の有用なデータエレメントをマージすべく実行され、Ｖ２中の複数のソースデータエレメントが、複数のターゲットデータエレメントによって占有されるＶ１の位置に移され、再帰計算は、Ｖ１中の複数の追加された有用なデータエレメントで、第２のイタレーション３３０に進むことができるようになる。

【００３２】

そのような併合操作においては、オーバーフロー、アンダーフロー、及び完全一致の、３つのシナリオが起こりうる。完全一致とは、Ｖ２中の有用なデータエレメントが、Ｖ１に残された未使用スロットの数と同一数あることを示す。したがって、完全一致においては、Ｖ２中の複数のソースデータエレメントの全ては、Ｖ１に残された未使用スロットへ移動（すなわち、入れ替え）する。結果として、Ｖ１は次のイタレーションを開始する完全ベクトルを有し、Ｋ１は更新され、全て１を含む。Ｖ２にはこれ以上ソースデータエレメントは残っていないので、Ｋ２は更新されて、全て０を含む。

【００３３】

併合操作３２０は、新しいデータエレメント（Ｃ０）の数が、Ｋ１中の０値のマスクビットの数（すなわちＡ１の数）よりも大きい、オーバーフローのシナリオを示している。したがって、Ｖ２中の新しいデータエレメントの全てがＶ１へ移動するわけではない。この例において、Ｖ２の位置７の、丸で囲ったＣ０はＶ２中に残される。一方、位置２、４および６の、その他のＣ０はＶ１へ移動した。この実施形態において、Ｖ２の下位のエレメントがＶ１へ移され、複数の代替の実施形態においては、Ｖ２の上位のエレメントがＶ１へ移されうる。併合操作３２０は、Ｋ１及びＫ２における、対応するマスクビットの更新もする。

【００３４】

併合操作３２０後、Ｖ１は８つのエレメントからなる完全ベクトルを含み、第２のイタレーション３３０を開始し、Ｖ２は位置７に残された１つのＣ０を有するのみである。この時点（併合操作３２０後）における、対応するマスクレジスタＫ１は全て１を含み、Ｋ２は位置７に、１の値を有する、たった１つのマスクビットを含む。

【００３５】

第２のイタレーション３３０後、アキュムレータＶ１は複数のＢ２及び複数のＣ１の組み合わせを含む。複数のＢエレメントに対する計算はこのイタレーション後に終了したので、それらのＢ２はメモリ、キャッシュ、または他のデータストレージへ保存できる。したがって、複数のＢエレメントのそれぞれに対する条件は偽（さらに計算するための条件を満たさないから）であり、複数のＣエレメントのそれぞれに対する条件は真（さらに計

算するための条件を満たすから）である。したがって、Ｋ１の複数のマスクビット値は、複数のＢ２に対応するそれらのマスクビットに対して０が設定され、複数のＣ１に対応するそれらのマスクビットに対して１が設定される。

【００３６】

複数のＢ２によって残された未使用スロットは、Ｖ２の残りの複数のソースデータエレメントが埋めることができ、この場合、それはＶ２の位置７のＣ０である。しかしながら、Ｃ０の数は、Ｂ２の数より少ないので、後に続く併合操作３４０においてアンダーフローが起きる。図３に示すアンダーフローのシナリオにおいて、Ｖ１における最下位のＢ２がＣ０で置き換えられ、複数の代替の実施形態においては、Ｖ１における最上位のＢ２がＣ０で置き換えられうる。その併合操作３４０は、Ｋ１及びＫ２における、対応するマスクビットの更新もする。

【００３７】

併合操作３４０後、アキュムレータＶ１は完全には埋まっておらず、Ｖ２は、Ｖ１へ移動できる有用なデータエレメントをこれ以上有していない。この時点（併合操作３４０後）におけるマスクレジスタＫ１は、複数のＣエレメントに対応する複数の位置に１を含み、Ｋ２は全て０を含む。複数の有用なデータエレメントの全てが処理され、Ｖ２にもはやソースデータエレメントが残らなくなるまで、Ｖ１へ移動させる、追加の有用なデータエレメントを、Ｖ２はロードし得るし、３２０及び／または３４０の併合操作を繰り返すことが可能である。この時点において、Ｖ１の複数のエレメントの全てが、必要とされるイタレーション回数に達するまで、Ｖ１は、追加の回数のイタレーションを受けうる。

【００３８】

０または１のマスクビット値の意味は、図３の例において示されるものと逆であり得る、例えば、０のマスクビット値は条件が満たされることを意味すべく用いられ得て、１のマスクビット値は条件が満たされないことを意味すべく用いられ得る、ということが理解されよう。いくつかの実施形態では、Ｋ１マスクビット値の意味は、Ｋ２マスクビット値の意味とは逆であり得る、例えば、１のＫ１マスクビット値は条件が満たされないことを意味すべく用いられ得て、１のＫ２マスクビット値は条件が満たされることを意味すべく用いられ得る。したがって、各マスクレジスタにおける各マスクビットの意味が、一貫性のある解釈を可能にすべく、矛盾なく定義される限り、同一のシナリオに対して、図３の例において異なるマスクビット値を使用可能である。

【００３９】

本発明の一実施形態によると、図３に関連して説明されるオペレーションは、ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令を含む複数のベクトル命令に応答して、プロセッサ（例えば、命令処理装置１１５）が実行する。ＳＰＡＲＳＥＭＯＶ命令は、条件を満たさない、Ｖ１中の複数のターゲットエレメント（例えば、もはや計算が必要ない複数のエレメント）を置き換えながら、複数のソースデータエレメントをベクトルレジスタＶ２からベクトルレジスタＶ１へと移動させるべく用いられ得る。ＲＷＭＡＳＫＵＰＤＡＴＥ命令は、マスクレジスタＫ１及びマスクレジスタＫ２の更新に用いることができ、それにより、それぞれ、条件を満たすＶ１及びＶ２における複数のデータエレメント（例えば、もっと計算を必要とする複数のエレメント）の複数の位置を特定する。一実施形態において、ＲＷＭＡＳＫＵＰＤＡＴＥは、２つのオペランドＫ１及びＫ２を有し、ＳＰＡＲＳＥＭＯＶは４つのオペランドＫ１、Ｖ１、Ｋ２及びＶ２を有する。複数の代替の実施形態においては、ＲＷＭＡＳＫＵＰＤＡＴＥ及び／またはＳＰＡＲＳＥＭＯＶの複数のオペランドのうちの幾つかは黙示的である。

【００４０】

図４Ａは、一実施形態による、ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令用の疑似コード４０１および４０２の一例を示す。疑似コード４０１および４０２において、ＫＬは、各ベクトルレジスタ（例えば、Ｖ１及びＶ２のそれぞれ）における、複数のデータエレメントの総数であるベクトルの長さを表す。ｚｍｍレジスタを、８ビットのデータエレメントを有するアキュムレータとして用いるならば、ＫＬ＝５１２／８＝６

４である。疑似コード４０１はＲＷＭＡＳＫＵＰＤＡＴＥ命令を、疑似コード４０２はＳＰＡＲＳＥＭＯＶ命令を記述する。プロセッサは、疑似コード４０１および４０２に示されるものとは異なる複数のオペレーションまたはロジックを有するＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令を実装しうることに留意されたい。

【００４１】

ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令はそれぞれ、複数のマスクレジスタを更新し、及び複数のデータエレメントを複数のベクトルレジスタ間で移動させる。これらの命令の結果を用いるべく、複数の追加的な命令を実行できて、それにより、再帰的なベクトル演算をより効率的に実行できる。図４Ｂは、一実施形態による、ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令を用いるコードセグメント４００の一例を示す。プロセッサが実行すると、コードセグメント４００は、プロセッサに、アレイＸの複数の独立したデータエレメントに対して再帰的なベクトル演算を実行させる。アレイＸは、メモリ、キャッシュ、または他のデータのストア位置にストアされうる。コードセグメント４００は、初期化部４１０、初期マージ部４２０、後続マージ部４３０、演算部４４０、及び残余部４５０を備える。４１０－４５０の各部におけるオペレーションを、プロセッサ（例えば、図１の命令処理装置１１５）が実行する方法５００の実施形態を示す図５Ａのフロー図を参照し後述する。

【００４２】

初期化部４１０において、マスクレジスタＫ１及びマスクレジスタＫ２の両方は、それらが対応するベクトルレジスタＶ１及びＶ２において有用なデータエレメントはないということを示す、０に初期化される。「複数の有用なデータエレメント」という用語は、計算を必要とする複数のデータエレメントを意味する。イタレーションは初期マージ部４２０から始まる。４２０において、まず、有用なデータエレメントがＶ２に残っていないかどうか決定すべく、Ｋ２をチェックする（ブロック５３１）。Ｖ２に有用なデータがなければ、複数の入力データエレメントがアレイＸからＶ２へとロードされ（ブロック５３２）、Ｋ２におけるそれらの対応するマスクビットがしかるべく設定される。

【００４３】

後続マージ部４３０は、Ｖ２が複数の有用なデータエレメントを含むシナリオを処理する。複数の有用なデータエレメントが前のオーバーフローからＶ２に残され得て、ブロック５３２においてＶ２へとロードされうる。ＳＰＡＲＳＥＭＯＶ命令４３１に応答して、Ｖ２中のこれらの有用なデータエレメントは、Ｋ１及びＫ２におけるマスクビットに従い、Ｖ１へと移される（ブロック５３３）。

【００４４】

ブロック５３３における移動の後、ＲＷＭＡＳＫＵＰＤＡＴＥ命令４３３に応答して、マスクレジスタＫ１及びマスクレジスタＫ２は更新され、それぞれ、Ｖ１及びＶ２中の複数の有用なデータエレメントの現在の位置を特定する（ブロック５３４）。

【００４５】

後続マージ部４３０において、第２のＳＰＡＲＳＥＭＯＶ命令４３２が実行され、Ｖ２からＶ１へと移された、アレイＸ中の複数のデータエレメントのインデックス（位置）をストアし、計算結果がアレイＸ中のそれらの元の位置にストアし戻され得るようになる。

【００４６】

演算部４４０は、（対応するマスクが全て１であることによって示されるような、すなわち、ＩｓＦｕｌｌＭａｓｋ（Ｋ１）が真である場合の）完全ベクトルのベクトル演算を処理する。Ｖ１が、有用なデータエレメントの完全ベクトルを有さず（ブロック５３５）、Ｖ１に、ロードされなかった複数の入力データエレメントがあれば（ブロック５３８）、複数の追加的な入力データエレメントが、Ｖ２を介してＶ１へロードされうる（ブロック５３２－５３４）ことを示す。Ｖ１が完全ベクトルを有さず、Ｖ１にロードされる入力データエレメントがもはや無いならば（ブロック５３８）、計算が終了し、複数の結果がアレイＸに保存し戻されるまで、Ｖ１中の複数の残存するデータエレメントが計算される、残余部４５０（ブロック５３９）に、複数のオペレーションが進むことを示す。

【００４７】
Ｖ１が複数の有用なデータエレメントからなる完全ベクトルを有するならば（ブロック５３５）、Ｖ１についてベクトル演算が実行され得る（ブロック５３６）。Ｖ１中のデータエレメントがどれもこれ以上計算を必要としないならば、マスクレジスタＫ１は更新される。１または複数のデータエレメントが、アレイＸに保存し戻される時点（ブロック５３７）において、Ｖ１中のそれらのデータエレメントがこれ以上計算を必要としなくなる（Ｋ１中の対応する０値のマスクビットで示されるように）まで、ベクトル演算は続く。示すようなその実施形態において、複数のデータエレメントはＳＣＡＴＴＥＲ命令と共に保存され得て、Ｋ１中の複数の０値のマスクビットが、関数ｋｎｏｔ（Ｋ１）を用いて特定され得る。ＲＷＭＡＳＫＵＰＤＡＴＥ命令及びＳＰＡＲＳＥＭＯＶ命令を除いては、ＳＣＡＴＴＥＲ、ｋｎｏｔ、ＩｓＦｕｌｌＭａｓｋ、などのコードセグメント４００において用いる複数の特定の命令及び関数を、複数の代替的な命令シーケンスがエミュレートできる。

【００４８】
Ｖ２を通してＶ１にロードされる入力データエレメントがもはやなくなる（ブロック５３８）、すなわち、アレイＸ中の入力データエレメントの全てがＶ２にロードされ、Ｖ２中の有用なデータエレメントの全てがＶ１に移されたとき、まで、ブロック５３１－５３７のオペレーションは繰り返される。これが、残余部４５０が始まるときである。この時点で、Ｖ１は複数の有用なデータエレメントから成る完全ベクトルを有し得ないが、Ｖ１中のそれらのデータエレメントはさらに計算を必要とする。ベクトル演算は、Ｖ１中の残存するデータエレメントの全てが、必要とされるイタレーション回数に達するまで続く（ブロック５３９）。この時点において、Ｖ１における演算結果がアレイＸに保存し戻され得る（例えば、ＳＣＡＴＴＥＲ命令を用いて）（ブロック５３９）。

【００４９】
図５Ｂは、一実施形態による、ＲＷＭＡＳＫＵＰＤＡＴＥ命令を実行する方法５１０のブロックフロー図である。方法５１０は、プロセッサ（例えば、図１の命令処理装置１１５）の、第１のマスクレジスタ及び第２のマスクレジスタを特定するマスク更新命令の受信で始まる（ブロック５１１）。プロセッサはマスク更新命令をデコードする（ブロック５１２）。そのデコードされたマスク更新命令に応答して、プロセッサは、第１のマスクレジスタ内の与えられた数のマスクビットを、例えばこれらのマスクビットを第１のビット値（例えば、０）から第２のビット値（例えば、１）に設定することによって反転させる段階（ブロック５１３）と、第２のマスクレジスタ内の与えられた数のマスクビットを、例えばこれらのマスクビットを第２のビット値（例えば、１）から第１のビット値（例えば、０）に設定することによって反転させる段階（ブロック５１４）と、を含む複数のオペレーションを実行する。与えられた数とは、第１のビット値を有する第１のマスクレジスタ内のマスクビットの数、及び、第２のビット値を有する第２のマスクレジスタ内のマスクビットの数のうちの小さい方である。代替的実施形態において、第１のビット値は１であり得て、第２のビット値は０であり得る。

【００５０】
図５Ｃは、一実施形態による、ＳＰＡＲＳＥＭＯＶ命令を実行する方法５２０のブロックフロー図である。方法５２０は、プロセッサ（例えば、図１の命令処理装置１１５）の、第１のマスクレジスタ、第２のマスクレジスタ、第１のベクトルレジスタ、及び第２のベクトルレジスタを特定するベクトル移動命令の受信から始まる（ブロック５２１）。プロセッサは、ベクトル移動オペレーションをデコードする（ブロック５２２）。デコードされたベクトル移動命令に応答して、かつ第１のマスクレジスタ及び第２のマスクレジスタ内の複数のマスクビット値に基づき、プロセッサは、第１のベクトルレジスタ内の与えられた数の複数のターゲットデータエレメントを、第２のベクトルレジスタ内の与えられた数の複数のソースデータエレメントで置き換える（ブロック５２３）。一実施形態において、各ソースデータエレメントは、第２のビット値（例えば、１）を有する第２のマスクレジスタ内のマスクビットに対応し、各ターゲットデータエレメントは、第１のビット

値（例えば、０）を有する第１のマスクレジスタ内のマスクビットに対応する。代替的実施形態においては、第１のビット値は１であり得て、第２のビット値は０であり得る。与えられた数とは、第１のビット値を有する第１のマスクレジスタ内のマスクビットの数、及び、第２のビット値を有する第２のマスクレジスタ内のマスクビットの数のうちの小さい方である。

【００５１】

様々な実施形態において、図５Ａ－５Ｃの方法は、汎用プロセッサ、専用プロセッサ（例えば、グラフィクスプロセッサまたはデジタルシグナルプロセッサ）、または別のタイプのデジタル論理デバイスまたは命令処理装置が実行しうる。いくつかの実施形態では、図５Ａ－５Ｃの方法は、図１の命令処理装置１１５、または図７－１３に示す実施形態などの、同様のプロセッサ、装置またはシステムが実行しうる。さらに、図７－１３に示すプロセッサ、装置及びシステムと同様、図１の命令処理装置１１５は、図５Ａ－５Ｃの複数の方法の実施形態と同一の、同様の、または異なる、複数のオペレーション及び複数の方法の実施形態を実行しうる。

【００５２】

いくつかの実施形態では、図１の命令処理装置１１５は、命令をソース命令セットからターゲット命令セットへと変換する命令変換部と共に動作しうる。例えば、命令変換部は、命令を、変換（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）、モーフィング、エミュレート、または別の方法でコンバートし、コアが処理する１または複数の他の命令にする。命令変換部は、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うる。命令変換部は、プロセッサ上、プロセッサ外、または、一部はプロセッサ上かつ一部はプロセッサ外でありうる。

【００５３】

図６は、本発明の複数の実施形態による、ソフトウェア命令変換器の使用を対比させたブロック図である。図示した実施形態においては、命令変換部はソフトウェア命令変換器であるが、代替的に命令変換部はソフトウェア、ファームウェア、ハードウェア、またはそれらの組み合わせにおいて実装されてよい。図６は、高級言語６０２で書かれたプログラムを、ｘ８６コンパイラ６０４を用いてコンパイルできて、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６１６がネイティブで実行しうる、ｘ８６バイナリコード６０６を生成することを示している。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６１６は、少なくとも１つのｘ８６命令セットコアを有するインテル社製プロセッサと実質的に同一な結果を得るべく、（１）インテル社製ｘ８６命令セットコアの命令セットのかなりの部分、または（２）複数のアプリケーションのオブジェクトコードバージョン、または少なくとも１つのｘ８６命令セットコアを有するインテル社製プロセッサ上で実行することを目的とされる他のソフトウェア、を互換性のある状態で実行、または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社製プロセッサと実質的に同一な複数の機能を実行可能な任意のプロセッサを表わす。ｘ８６コンパイラ６０４は、追加的なリンケージ処理有りで、または無しで、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６１６上で実行され得る、ｘ８６バイナリコード６０６（例えば、オブジェクトコード）を生成すべく動作可能なコンパイラを表わす。

【００５４】

同様に図６は、高級言語６０２で書かれたプログラムは、代替的な命令セットのコンパイラ６０８を用いてコンパイルされ得て、少なくとも１つのｘ８６命令セットコアを有さないプロセッサ６１４（例えば、カリフォルニア州サニーベールにあるミップス・テクノロジーズのＭＩＰＳ命令セットを実行する、及び／または、カリフォルニア州サニーベールにある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）がネイティブで実行しうる代替的な命令セットのバイナリコード６１０を生成することを示す。命令変換部６１２は、ｘ８６バイナリコード６０６を、ｘ８６命令セットコアを有さないプロセッサ６１４がネイティブで実行し得るコードへと変換すべく用いられる。変換後のコードは、代替的な命令セットのバイナリコード６１０と同一である可

能性が低い。なぜなら、これができる命令変換部は作るのが困難だからである。しかしながら、変換後のコードは一般的なオペレーションを達成し、代替的な命令セットからの複数の命令から生成されるであろう。したがって、命令変換部６１２は、エミュレーション、シミュレーション、または任意の他の処理を通して、ｘ８６命令セットプロセッサまたはコアを有さないプロセッサまたは他の電子デバイスが、ｘ８６バイナリコード６０６を実行できるようにする、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わす。　＜例示的なコアアーキテクチャ＞　＜インオーダ及びアウトオブオーダコアブロック図＞

【００５５】

図７Ａは、本発明の複数の実施形態による、例示的なインオーダパイプライン、及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図示するブロック図である。図７Ｂは、本発明の複数の実施形態による、例示的な、インオーダアーキテクチャコアの実施形態、及びプロセッサに含まれる、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図示するブロック図である。図７Ａ及び図７Ｂ内の複数の実線で囲まれたボックスは、インオーダパイプライン及びインオーダコアを示し、一方で複数の点線で囲まれた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／実行パイプライン及びコアを示す。インオーダ態様がアウトオブオーダ態様のサブセットであるとして、アウトオブオーダ態様を説明する。

【００５６】

図７Ａにおいて、プロセッサパイプライン７００はフェッチ段階７０２、レンス（ｌｅｎｇｔｈ）デコード段階７０４、デコード段階７０６、割り当て段階７０８、リネーム段階７１０、スケジュール段階（ディスパッチまたは発行としても知られる）７１２、レジスタリード／メモリリード段階７１４、実行段階７１６、ライトバック／メモリライト段階７１８、例外処理段階７２２、及びコミット段階７２４を備える。

【００５７】

図７Ｂは、実行エンジンユニット７５０に連結されるフロントエンドユニット７３０を含むプロセッサコア７９０を示し、実行エンジンユニット７５０及びフロントエンドユニット７３０の両方はメモリユニット７７０に連結されている。コア７９０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コアタイプもしくは代替的な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７９０は、例えば、ネットワークコアまたは通信コア、圧縮エンジン、コプロセッサコア、汎用コンピューティンググラフィックスプロセッシングユニット（ＧＰＧＰＵ）コア、グラフィックスコア等の専用コアであってよい。

【００５８】

フロントエンドユニット７３０は、命令キャッシュユニット７３４に連結される分岐予測ユニット７３２を備え、この命令キャッシュユニットは命令トランスレーションルックアサイドバッファ（ＴＬＢ）７３６に連結され、このＴＬＢは命令フェッチユニット７３８に連結され、この命令フェッチユニットはデコードユニット７４０に連結されている。デコードユニット７４０（またはデコーダ）は、命令をデコードし、その元の命令からデコードされるか、または元の命令を反映するか、または元の命令から得られる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ーポイント、マイクロ命令、他の命令、または他の制御信号を出力として生成しうる。デコードユニット７４０は様々な異なるメカニズムを用いて実装されうる。適切なメカニズムの複数例には、限定されるものではないが、ルックアップテーブル、ハードウェア実装、プログラム可能ロジックアレイ（ＰＬＡ）、マイクロコード読み出し専用メモリ（ＲＯＭ）等が挙げられる。一実施形態において、コア７９０は、（例えば、デコードユニット７４０内で、またはフロントエンドユニット７３０内において）複数の特定のマクロ命令用のマイクロコードをストア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７４０は、実行エンジンユニット７５０内のリネーム／アロケータユニット７５２に連結されている。

【００５９】
実行エンジンユニット７５０は、リタイアメントユニット７５４、及び１セットの、１または複数のスケジューラユニット７５６に連結されるリネーム／アロケータユニット７５２を含む。スケジューラユニット（複数）７５６は、複数のリザベーションステーション、中央命令ウィンドウ等を含む任意の数の異なるスケジューラを表す。スケジューラユニット（複数）７５６は物理レジスタファイル（複数）ユニット（複数）７５８に連結されている。複数の、物理レジスタファイル（複数）ユニット７５８のそれぞれは、１または複数の物理レジスタファイルを表し、その複数の物理レジスタファイルのうちの異なるものは、スカラ整数、スカラ浮動小数点、パックド整数、パックド浮動小数点、ベクトル整数、ベクトル浮動小数点、ステータス（例えば、次に実行される命令のアドレスである命令ポインタ）等の、１または複数の異なるデータタイプをストアする。一実施形態において、物理レジスタファイル（複数）ユニット７５８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、及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うる。物理レジスタファイル（複数）ユニット（複数）７５８は、（例えば、リオーダバッファ（複数）及びリタイアメントレジスタファイル（複数）を用いて；フューチャーファイル（複数）、履歴バッファ（複数）、及びリタイアメントレジスタファイル（複数）を用いて；レジスタマップ及びレジスタのプールを用いて；等により）レジスタリネーミング及びアウトオブオーダ実行が実装されうるさまざまな方法を示すべく、リタイアメントユニット７５４によって重ね合わされる。リタイアメントユニット７５４及び物理レジスタファイル（複数）ユニット（複数）７５８は、実行クラスタ（複数）７６０に連結されている。実行クラスタ（複数）７６０は、１セットの、１または複数の実行ユニット７６２、及び、１セットの、１または複数のメモリアクセスユニット７６４を備える。複数の実行ユニット７６２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、様々な演算（例えば、シフト、加算、減算、乗算）を実行しうる。幾つかの実施形態が、特定の複数の機能または特定の複数セットの機能専用の幾つかの実行ユニットを含みうる一方で、他の複数の実施形態は、全機能を全て実行する１つの実行ユニットのみを含むか、または複数の実行ユニットを含む。スケジューラユニット（複数）７５６、物理レジスタファイル（複数）ユニット（複数）７５８、及び実行クラスタ（複数）７６０については、複数個ある可能性があるように示した。なぜなら、複数の特定の実施形態は、複数の特定のタイプのデータ／オペレーションに対して別個の複数のパイプライン（例えば、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、及び／または、それぞれが自身のスケジューラユニット、物理レジスタファイル（複数）ユニット及び／または実行クラスタを有するメモリアクセスパイプライン、－及び別個のメモリアクセスパイプラインの場合においては、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（複数）７６４を有する複数の特定の実施形態が実装される）を作成するからである。複数の別個のパイプラインが使用されると、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、残りはインオーダ発行／実行でありうることも理解されるべきである。

【００６０】
１セットのメモリアクセスユニット７６４はメモリユニット７７０に連結され、このメモリユニットは、レベル２（Ｌ２）キャッシュユニット７７６に連結されるデータキャッシュユニット７７４に連結されるデータＴＬＢユニット７７２を備える。１つの例示的な実施形態において、複数のメモリアクセスユニット７６４は、ロードユニット、ストアアドレスユニット、及びストアデータユニットを含んでよく、それぞれは、メモリユニット７７０内のデータＴＬＢユニット７７２に連結されている。命令キャッシュユニット７３４は、メモリユニット７７０内のレベル２（Ｌ２）キャッシュユニット７７６にさらに連結されている。Ｌ２キャッシュユニット７７６は、１または複数の他のレベルのキャッシ

ュに連結されており、最終的にはメインメモリに連結されている。
【００６１】
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、以下のようなパイプライン７００を実装しうる：１）命令フェッチ７３８が、フェッチ段階７０２及びレンスデコード段階７０４を実行する；２）デコードユニット７４０が、デコード段階７０６を実行する；３）リネーム／アロケータユニット７５２が、割り当て段階７０８及びリネーム段階７１０を実行する；４）スケジューラユニット（複数）７５６が、スケジュール段階７１２を実行する；５）物理レジスタファイル（複数）ユニット（複数）７５８及びメモリユニット７７０が、レジスタリード／メモリリード段階７１４を実行する；実行クラスタ７６０が実行段階７１６を実行する；６）メモリユニット７７０及び物理レジスタファイル（複数）ユニット（複数）７５８が、ライトバック／メモリライト段階７１８を実行する；７）様々なユニットが例外処理段階７２２に関与しうる；及び８）リタイアメントユニット７５４及び物理レジスタファイル（複数）ユニット（複数）７５８がコミット段階７２４を実行する。
【００６２】
コア７９０は、本明細書で説明する命令（複数）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と共に追加された幾つかの拡張を有する）；カルフォルニア州サニーベールにあるミップス・テクノロジーズのＭＩＰＳ命令セット；カルフォルニア州サニーベールにあるＡＲＭホールディングスのＡＲＭ命令セット（ＮＥＯＮなどの任意追加の拡張を有する））をサポートしうる。一実施形態において、コア７９０はパックドデータ命令セット拡張（例えば、ＳＳＥ、ＡＶＸ１、ＡＶＸ２等）をサポートすべくロジックを備え、それにより、多数のマルチメディアアプリケーションが用いる複数のオペレーションを、パックドデータを使用して実行できるようになる。
【００６３】
コアはマルチスレッド化（２またはそれ以上のパラレルなセットのオペレーションまたはスレッドを実行する）をサポートし得て、タイムスライスマルチスレッド化、同時マルチスレッド化（物理コアが同時にマルチスレッド化している複数のスレッドのそれぞれに対して、単一の物理コアが論理コアを提供する）、またはそれらの組み合わせ（例えば、Ｉｎｔｅｌ（登録商標）のハイパースレッディングテクノロジ中などのタイムスライスフェッチ及びデコード、並びにその後の同時マルチスレッド化）を含む様々な方法でそれを行いうることが理解されるべきである。
【００６４】
レジスタリネーミングがアウトオブオーダ実行との関連で説明される一方で、レジスタリネーミングはインオーダアーキテクチャにおいて使用されうることが理解されるべきである。図示したプロセッサの実施形態が、別個の命令、及びデータキャッシュユニット７３４／７７４及び共有のＬ２キャッシュユニット７７６を備える一方で、代替的実施形態は、例えば、レベル１（Ｌ１）の内部キャッシュ、または複数レベルの内部キャッシュ等の、複数の命令及びデータの両方に対する単一の内部キャッシュを有しうる。いくつかの実施形態では、システムは、内部キャッシュ、及び、コア及び／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む。代替的に、全てのキャッシュがコア及び／またはプロセッサの外部にあってよい。＜具体的で例示的なインオーダコアアーキテクチャ＞
【００６５】
図８Ａ－８Ｂは、より具体的な例示的インオーダコアアーキテクチャのブロック図を示す。そのコアは、チップ内の幾つかの論理ブロック（同一タイプ及び／または異なるタイプの複数の他のコアを含む）のうちの１つでありうる。複数の論理ブロックは、アプリケーションに応じて、ある固定のファンクションロジック、メモリＩ／Ｏインターフェース、及び他の必要なＩＯロジックと、高帯域幅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。
【００６６】

図８Ａは、本発明の複数の実施形態による、自身のオンダイ相互接続ネットワーク８０２への接続、及び自身のレベル２（Ｌ２）のキャッシュのローカルサブセット８０４を伴う、単一のプロセッサコアのブロック図である。一実施形態において、命令デコーダ８００は、パックドデータ命令セット拡張を有するｘ８６命令セットをサポートする。Ｌ１キャッシュ８０６は、スカラユニット及びベクトルユニットへの、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（設計を単純化するための）において、スカラユニット８０８及びベクトルユニット８１０は、複数の別個のレジスタセット（それぞれ、複数のスカラレジスタ８１２及び複数のベクトルレジスタ８１４）を使用し、それらの間を伝送されるデータがメモリへ書き込まれ、その後レベル１（Ｌ１）キャッシュ８０６からリードバックされる一方で、本発明の複数の代替的実施形態は異なるアプローチ（例えば、単一のレジスタセットを用いるか、または、書き込まれもリードバックもされずに、２つのレジスタファイル間をデータが伝送され得るようにする通信パスを含む）を使用しうる。

【００６７】

Ｌ２キャッシュのローカルサブセット８０４は、１つのプロセッサコアにつき１つの、複数の別個のローカルサブセットに分割されるグローバルＬ２キャッシュの一部である。各プロセッサコアは、Ｌ２キャッシュの、自身のローカルサブセット８０４へのダイレクトアクセスパスを有する。プロセッサコアがリードするデータは、自身のＬ２キャッシュのサブセット８０４へストアされ、他のプロセッサコアが自身のローカルＬ２キャッシュのサブセットにアクセスするのとパラレルに、高速にアクセス可能である。プロセッサコアが書き込むデータは、自身のＬ２キャッシュのサブセット８０４にストアされ、必要ならば複数の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に対するコヒーレンシを保証する。リングネットワークは、複数のプロセッサコア、複数のＬ２キャッシュ、及び複数の他の論理ブロックが、チップ内で互いに通信できるようにする双方向性のネットワークである。各リングデータパスは１つの方向につき１０１２ビット幅である。

【００６８】

図８Ｂは、本発明の複数の実施形態による、図８Ａのプロセッサコアの一部の拡大図である。図８Ｂは、ベクトルユニット８１０及びベクトルレジスタ８１４に関するより詳細な内容と共に、Ｌ１キャッシュ８０６のＬ１データキャッシュ８０６Ａ部分を備える。

【００６９】

具体的に、ベクトルユニット８１０は、１６ビットのベクトル処理ユニット（ＶＰＵ）であり（１６幅ＡＬＵ８２８を参照）、整数命令のうちの１または複数、単精度浮動小数点命令、及び倍精度浮動小数点命令を実行する。ＶＰＵは、スウィズルユニット８２０による複数のレジスタ入力のスウィズル、複数の数値変換ユニット８２２Ａ－Ｂによる数値変換、及び、メモリ入力に対する複製ユニット８２４による複製をサポートする。ライトマスクレジスタ８２６は複数の結果のベクトル書き込みのプレディケートを可能にする。

＜統合型メモリコントローラ及び統合型グラフィックを有するプロセッサ＞

【００７０】

図９は、本発明の複数の実施形態による、２つ以上のコア、統合型メモリコントローラ、及び統合型グラフィックスを有しうるプロセッサ９００のブロック図である。図９中の複数の実線で囲まれたボックスは、単一のコア９０２Ａ、システムエージェント９１０、１セットの１または複数のバスコントローラユニット９１６を有するプロセッサ９００を図示する。複数の点線で囲まれたボックスの任意の追加は、複数のコア９０２Ａ－Ｎ、システムエージェントユニット９１０内の１セットの１または複数の統合型メモリコントローラユニット９１４、及び専用ロジック９０８を有する代替的なプロセッサ９００を図示する。

【００７１】

したがって、プロセッサ９００の複数の異なる実装は、１）統合型グラフィックス及び／または科学的（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９０８、及び１または複数の汎用コア（例えば、複数の汎用インオーダコア、複数の汎用アウトオブオーダコア、その２つの組み合わせ）である複数のコア９０２Ａ－Ｎを有するＣＰＵ；２）主にグラフィクス及び／または科学（スループット）向けの多数の専用コアである複数のコア９０２Ａ－Ｎを有するコプロセッサ；及び、３）多数の汎用インオーダコアである複数のコア９０２Ａ－Ｎを有するコプロセッサを含みうる。したがって、プロセッサ９００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（汎用グラフィックスプロセッシングユニット）、高スループットメニーインテグレーテッドコア（ＭＩＣ）コプロセッサ（３０以上のコアを含む）、組込みプロセッサ等の、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りうる。プロセッサは、１または複数のチップ上に実装されうる。プロセッサ９００は、例えば、ＢｉＣＭＯＳ，ＣＭＯＳ、またはＮＭＯＳなどの幾つかのプロセス技術のうちの任意のものを用いる１または複数の基板の一部であり得て、及び／または、１または複数の基板上に実装されうる。

【００７２】

メモリ階層は、複数のコア内の、１または複数レベルのキャッシュ、１セットの１または複数の共有キャッシュユニット９０６、及び、１セットの統合型メモリコントローラユニット９１４に連結された外部メモリ（図示せず）を備える。１セットの共有キャッシュユニット９０６は、レベル２（Ｌ２）、レベル３（Ｌ３）、レベル４（Ｌ４）、または他の複数レベルのキャッシュ等の１または複数の中間レベルのキャッシュ、ラストレベルキャッシュ（ＬＬＣ）、及び／またはそれらの組み合わせを含みうる。一実施形態において、リングベースの相互接続ユニット９１２は、統合型グラフィックスロジック９０８、１セットの共有キャッシュユニット９０６、及びシステムエージェントユニット９１０／統合型メモリコントローラユニット（複数）９１４を相互接続する一方で、複数の代替的実施形態は、そのような複数のユニットを相互接続する、任意の数の公知の技術を使用しうる。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９０６と複数のコア９０２Ａ－Ｎとの間で維持される。

【００７３】

いくつかの実施形態では、コア９０２Ａ－Ｎのうちの１または複数は、マルチスレッド化することが可能である。システムエージェント９１０は、複数のコア９０２Ａ－Ｎを調整し、操作する複数の構成要素を備える。システムエージェントユニット９１０は、例えば、電力制御ユニット（ＰＣＵ）及びディスプレイユニットを含みうる。ＰＣＵは、複数のコア９０２Ａ－Ｎ及び統合型グラフィックスロジック９０８の電力状態の調整に必要なロジック及び複数の構成要素でありうるか、またはそれらを含みうる。ディスプレイユニットは、１または複数の外部接続のディスプレイの駆動用である。

【００７４】

コア９０２Ａ－Ｎは、アーキテクチャ命令セットに関して同種または異種でありうる。すなわち、２またはそれ以上のコア９０２Ａ－Ｎは同一の命令セットを実行可能であり得て、一方で、他のコアはその命令セットのサブセットまたは異なる命令セットのみを実行可能でありうる。＜例示的なコンピューターアーキテクチャ＞

【００７５】

図１０－図１３は、例示的なコンピューター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、ハンドヘルド型ＰＣ、携帯情報端末（ＰＤＡ）、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イヤ、ハンドヘルドデバイス、及び様々な他の電子デバイスの当技術分野において既知の複数の他のシステム設計及び構成も適している。一般的には、本明細書で開示するような、プロセッサ及び／または他の実行ロジックを組み込み可能な多様なシステムまたは電子デバイスが概して適している。

【００７６】

次に図１０を参照すると、本発明の一実施形態による、システム１０００のブロック図が示される。システム１０００は、コントローラハブ１０２０に連結される、１または複数のプロセッサ１０１０、１０１５を含みうる。一実施形態において、コントローラハブ１０２０は、グラフィックスメモリコントローラハブ（ＧＭＣＨ）１０９０及び入出力ハブ（ＩＯＨ）１０５０（複数の別個のチップ上にあってもよい）を含み、ＧＭＣＨ１０９０は、メモリ１０４０及びコプロセッサ１０４５に連結されるメモリ及びグラフィクスコントローラを含み、ＩＯＨ１０５０は、入出力（Ｉ／Ｏ）デバイス１０６０をＧＭＣＨ１０９０に連結する。代替的に、メモリ及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、（本明細書で説明するように）プロセッサ内に統合されており、メモリ１０４０及びコプロセッサ１０４５は、プロセッサ１０１０と、ＩＯＨ１０５０を有する単一チップ内のコントローラハブ１０２０とに直接連結されている。

【００７７】

追加のプロセッサ１０１５の、任意選択される特性が図１０に破線で示される。各プロセッサ１０１０、１０１５は、本明細書で説明される、複数のプロセッサコアのうちの１または複数を含み得て、プロセッサ９００のあるバージョンでありうる。

【００７８】

メモリ１０４０は、例えば、ダイナミックランダムアクセスメモリ（ＤＲＡＭ）、相変化メモリ（ＰＣＭ）、またはその２つの組み合わせでありうる。少なくとも１つの実施形態に対して、コントローラハブ１０２０は、フロントサイドバス（ＦＳＢ）などのマルチドロップバス、クイックパスインターコネクト（ＱＰＩ）などの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１０９５を介してプロセッサ（複数）１０１０，１０１５と通信する。

【００７９】

一実施形態において、コプロセッサ１０４５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、組込みプロセッサ等の専用プロセッサである。一実施形態において、コントローラハブ１０２０は、統合型グラフィックスアクセラレータを含みうる。

【００８０】

アーキテクチャ特性、マイクロアーキテクチャ特性、熱特性、電力消費特性等を含む、メリットのメトリクスの範囲に関して、複数の物理リソース１０１０，１０１５間の違いは様々あり得る。

【００８１】

一実施形態において、プロセッサ１０１０は一般的なタイプの、データプロセッシングオペレーション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がその複数の命令内に組み込まれうる。プロセッサ１０１０はこれらのコプロセッサ命令を、接続されたコプロセッサ１０４５によって実行されるべきタイプのものであると認識する。従って、プロセッサ１０１０はコプロセッサ１０４５に対して、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わす複数の制御信号）をコプロセッサバスまたは他の相互接続上に発行する。コプロセッサ（複数）１０４５は複数の受信したコプロセッサ命令を受け付け、実行する。

【００８２】

次に図１１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１１００のブロック図が示される。図１１に示されるように、マルチプロセッサシステム１１００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１１５０を介して連結される、第１のプロセッサ１１７０及び第２のプロセッサ１１８０を備える。プロセッサ１１７０および１１８０のそれぞれは、プロセッサ９００のあるバージョンでありうる。本発明の一実施形態において、プロセッサ１１７０および１１８０はそれぞれ、プロセッサ１０１０および１０１５であり、一方コプロセッサ１１３８はコプロセッサ１０４５である。別の実施形態では、プロセッサ１１７０および１１８０はそれぞれ、プロセッサ１０１０及びコプロセッサ１０４５である。

【００８３】
プロセッサ１１７０および１１８０はそれぞれ、統合型メモリコントローラ（ＩＭＣ）ユニット１１７２および１１８２を備えるように示されている。プロセッサ１１７０は、自身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１１７６および１１７８も備え、同様に、第２のプロセッサ１１８０はＰ－Ｐインターフェース１１８６および１１８８を備える。プロセッサ１１７０、１１８０は、ポイントツーポイント（Ｐ－Ｐ）インターフェース回路１１７８、１１８８を用いて、Ｐ－Ｐインターフェース１１５０を介して情報を交換しうる。図１１に示されるように、ＩＭＣ１１７２および１１８２は、複数のプロセッサを、各メモリ、すなわち、各プロセッサにローカルに接続されたメインメモリの複数部分でありうる、メモリ１１３２及びメモリ１１３４に連結する。
【００８４】
プロセッサ１１７０、１１８０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１１７６、１１９４、１１８６、１１９８を用いて、個々のＰ－Ｐインターフェース１１５２、１１５４を介してチップセット１１９０と各情報を交換しうる。チップセット１１９０は、高性能インターフェース１１３９を介してコプロセッサ１１３８と任意選択的に情報を交換しうる。一実施形態において、コプロセッサ１１３８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、組込みプロセッサ等の専用プロセッサである。
【００８５】
共有キャッシュ（図示せず）は、いずれかのプロセッサ内、または両方のプロセッサの外部に含まれ得て、さらにＰ－Ｐ相互接続を介してその複数のプロセッサに接続され得て、プロセッサが低電力モードになった場合に、いずれかの、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ストアされるようにする。
【００８６】
チップセット１１９０は、インターフェース１１９６を介して第１のバス１１１６に連結しうる。一実施形態において、第１のバス１１１６は、ペリフェラルコンポーネントインターコネクト（ＰＣＩ）バスであってもよいし、またはＰＣＩエクスプレスバスまたは別の第３世代Ｉ／Ｏ相互接続バス等のバスであってもよいが、本発明の範囲はそのように限定されない。
【００８７】
図１１に示されるように、第１のバス１１１６を第２のバス１１２０に連結するバスブリッジ１１１８と共に、様々なＩ／Ｏデバイス１１１４を、第１のバス１１１６に連結しうる。一実施形態においては、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）ユニットなどの）、フィールドプログラマブルゲートアレイ、または任意の他のプロセッサ等の１または複数の追加のプロセッサ１１１５を第１のバス１１１６に連結する。一実施形態において、第２のバス１１２０はローピンカウント（ＬＰＣ）バスでありうる。
【００８８】
一実施形態において、様々なデバイスを、例えば、キーボード及び／またはマウス１１２２、通信デバイス１１２７、及び、複数の命令／コード及びデータ１１３０を含みうるディスクドライブまたは他の大容量ストレージデバイス等のストレージユニット１１２８を含む第２のバス１１２０に連結しうる。さらに、オーディオＩ／Ｏ１１２４を、第２のバス１１２０に連結してよい。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、システムは、図１１のポイントツーポイントアーキテクチャではなく、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うる。
【００８９】
次に図１２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１２００のブロック図が示される。図１１及び図１２中の同様のエレメントには同様の

参照番号を付し、図１１中の複数の特定の態様を図１２から省き、図１２中の複数の他の態様を分かりにくくしないようにした。

【００９０】

図１２は、プロセッサ１１７０、１１８０がそれぞれ、統合されたメモリ及びＩ／Ｏ制御ロジック（「ＣＬ」）１１７２及び１１８２を備えうることを示している。したがって、ＣＬ１１７２、１１８２は、複数の統合型メモリコントローラユニットを備え、Ｉ／Ｏ制御ロジックを備える。図１２は、ＣＬ１１７２、１１８２に連結されるのはメモリ１１３２、１１３４だけでなく、ＩＯデバイス１２１４も制御ロジック１１７２、１１８２に連結されることを示している。レガシーＩ／Ｏデバイス１２１５はチップセット１１９０に連結される。

【００９１】

次に図１３を参照すると、本発明の実施形態による、ＳｏＣ１３００のブロック図が示される。図９中の同様のエレメントには同様の参照番号が付してある。また、点線で囲まれたボックスは、より高度なＳｏＣ上の任意選択の機能である。図１３において、相互接続ユニット（複数）１３０２は、１セットの１または複数のコア２０２Ａ－Ｎ及び共有キャッシュユニット（複数）９０６を備えるアプリケーションプロセッサ１３１０；システムエージェントユニット９１０；バスコントローラユニット（複数）９１６；統合型メモリコントローラユニット（複数）９１４；統合型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及びビデオプロセッサを含みうる１セットの１または複数のコプロセッサ１３２０；スタティックランダムアクセスメモリ（ＳＲＡＭ）ユニット１３３０；ダイレクトメモリアクセス（ＤＭＡ）ユニット１３３２；及び１または複数の外部のディスプレイに連結するためのディスプレイユニット１３４０に連結される。一実施形態において、コプロセッサ（複数）１３２０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、組込みプロセッサ等の専用プロセッサを備える。

【００９２】

本明細書にて開示される複数のメカニズムの複数の実施形態は、ハードウェア、ソフトウェア、ファームウェア、またはそのような実装手法の組み合わせにおいて実装されうる。本発明の複数の実施形態は、少なくとも１つのプロセッサ、ストレージシステム（揮発及び不揮発性メモリ、及び／またはストレージエレメントを含む）、少なくとも１つの入力デバイス、及び少なくとも１つの出力デバイスを備え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うる。

【００９３】

図１１に図示するコード１１３０などのプログラムコードは、入力命令に適用され、本明細書で説明される機能を実行し、出力情報を生成する。出力情報は、既知の方法で１または複数の出力デバイスに適用されうる。こうした適用のために、プロセッシングシステムは、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などのプロセッサを有する任意のシステムを備える。

【００９４】

プログラムコードは、高級手続き型プログラミング言語またはオブジェクト指向型プログラミング言語で実装され、プロセッシングシステムと通信する。プログラムコードは、望むならば、アセンブリ言語又は機械言語で実装されてもよい。実際には、本明細書で説明される複数のメカニズムは、何れの特定のプログラミング言語へも範囲を限定されるものではない。いずれの場合であっても、言語はコンパイラ型言語またはインタプリタ型言語でありうる。

【００９５】

少なくとも１つの実施形態の１または複数の態様は、機械がリードすると、機械に、本明細書で説明した複数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わす、機械可読媒体上にストアされた複数の代表的な命令により実装されう

る。「ＩＰコア」として知られるそのような表現は、有形のマシン可読媒体上にストアされ、様々な顧客または製造施設に供給されて、実際にロジックまたはプロセッサを作成する複数の製造機械にロードされうる。

【００９６】

そのような機械可読記憶媒体は、限定はしないが、ハードディスク、フロッピー（登録商標）ディスク、光ディスク、コンパクトディスクリードオンリメモリ（ＣＤ－ＲＯＭ）、コンパクトディスクリライタブル（ＣＤ－ＲＷ）、及び光磁気ディスクを含む任意の他のタイプのディスク、リードオンリメモリ（ＲＯＭ）、ダイナミックランダムアクセスメモリ（ＤＲＡＭ）、スタティックランダムアクセスメモリ（ＳＲＡＭ）等のランダムアクセスメモリ（ＲＡＭ）、消去可能プログラマブルリードオンリメモリ（ＥＰＲＯＭ）、フラッシュメモリ、電気的消去可能プログラマブルリードオンリメモリ（ＥＥＰＲＯＭ）、相変化メモリ（ＰＣＭ）などの半導体デバイス、磁気カードまたは光カード、または電子命令をストアするのに適した任意の他のタイプの媒体等のストレージ媒体を含む、機械またはデバイスによって製造または形成される物品の、非一時的で有形な複数の構成を含みうる。

【００９７】

従って、本発明の複数の実施形態は、複数の命令を含むか、または、本明細書にて説明する複数の構造、複数の回路、複数の装置、複数のプロセッサ、及び／または複数のシステムの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、非一時的で有形な機械可読媒体も含む。そのような複数の実施形態はプログラム製品と称されることもある。

【００９８】

複数の特定の例示的な実施形態が添付の複数の図面において説明され、示されてきたが、複数のそのような実施形態は、広義の発明の例示に過ぎず、それを限定するものではなく、かつ、本発明は、示され、説明される具体的な複数の構造及び複数の構成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。なぜなら、本開示を研究すれば、当業者は様々な他の変更形態を考え付き得るからである。成長が早く、さらなる進歩を容易には予測し得ないこのような技術の領域においては、複数の開示した実施形態は、本開示の原理、または添付の特許請求の範囲から逸脱することなく、技術的進歩を可能にすることによって容易になるような、構成及び詳細な内容を、たやすく変更可能でありうる。本明細書によれば、以下の各項目に記載の事項もまた開示される。

［項目１］

第１のマスクレジスタ、第２のマスクレジスタ、第１のベクトルレジスタ、及び第２のベクトルレジスタを含むレジスタファイルと、

前記レジスタファイルに連結される実行回路と、を備え、

前記実行回路は、命令を実行し、前記第１のベクトルレジスタ内の与えられた数の複数のターゲットデータエレメントを、前記第２のベクトルレジスタ内の前記与えられた数の複数のソースデータエレメントで置き換え、

前記複数のソースデータエレメントの各々は、第２のビット値を有する前記第２のマスクレジスタ内のマスクビットに対応し、前記複数のターゲットデータエレメントの各々は、第１のビット値を有する前記第１のマスクレジスタ内のマスクビットに対応する

装置。

［項目２］

前記第１のビット値を有する前記第１のマスクレジスタの前記マスクビットの各々に対し、前記実行回路は、前記複数のソースデータエレメントのうちの１つとして、前記第２のマスクレジスタ内の前記第２のビット値の、対応するマスクビットを有するデータエレメントを探索するために前記第２のベクトルレジスタを検索する

項目１に記載の装置。

［項目３］

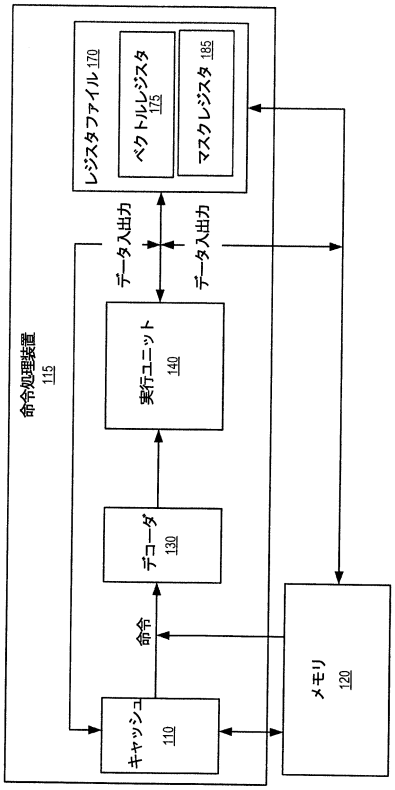
前記実行回路は、前記第１のベクトルレジスタ内の複数のデータエレメントに対して再

帰計算を実行し、前記第１のベクトルレジスタ内に前記再帰計算の複数の結果を蓄積する
項目１または２に記載の装置。
［項目４］
前記実行回路は、前記第１のベクトルレジスタ内の１または複数のデータエレメントがさらに計算を必要としなくなるまで、前記第１のベクトルレジスタ内の複数のデータエレメントに対して再帰計算を実行し、前記第１のマスクレジスタを用いて、前記１または複数のデータエレメントが前記複数のターゲットデータエレメントであると示す
項目１から３の何れか一項に記載の装置。
［項目５］
前記第１のベクトルレジスタ内の前記与えられた数の複数のターゲットデータエレメントは、前記第１のベクトルレジスタ内の複数の下位のエレメントまたは複数の上位のエレメントのいずれかであり、前記第２のベクトルレジスタ内の前記与えられた数の複数のソースデータエレメントは、前記第２のベクトルレジスタ内の複数の下位のエレメントまたは複数の上位のエレメントのいずれかである
項目１から４の何れか一項に記載の装置。
［項目６］
前記第１のビット値は前記第２のビット値の逆である
項目１から５の何れか一項に記載の装置。
［項目７］
前記第１のビット値は前記第２のビット値と同一である
項目１から６のいずれか一項に記載の装置。
［項目８］
前記与えられた数は、前記第１のビット値を有する前記第１のマスクレジスタ内のマスクビットの数、及び、前記第２のビット値を有する前記第２のマスクレジスタ内のマスクビットの数のうちの小さい方である
項目１から７の何れか一項に記載の装置。
［項目９］
プロセッサが、第１のマスクレジスタ、第２のマスクレジスタ、第１のベクトルレジスタ、及び第２のベクトルレジスタを特定するベクトル移動命令を受信する段階と、
前記ベクトル移動命令に応答して、前記第１のベクトルレジスタ内の与えられた数の複数のターゲットデータエレメントを、前記第２のベクトルレジスタ内の前記与えられた数の複数のソースデータエレメントで置き換える段階と、を含み、
前記複数のソースデータエレメントの各々は、第２のビット値を有する前記第２のマスクレジスタ内のマスクビットに対応し、前記複数のターゲットデータエレメントの各々は、第１のビット値を有する前記第１のマスクレジスタ内のマスクビットに対応する
方法。
［項目１０］
前記第１のビット値を有する前記第１のマスクレジスタの前記マスクビットの各々に対し、前記複数のソースデータエレメントのうちの１つとして、前記第２のマスクレジスタ内の前記第２のビット値の、対応するマスクビットを有するデータエレメントを探索するために前記第２のベクトルレジスタを検索する段階をさらに備える
項目９に記載の方法。
［項目１１］
前記第１のベクトルレジスタ内の複数のデータエレメントに対して再帰計算を実行する段階と、前記第１のベクトルレジスタ内に前記再帰計算の複数の結果を蓄積する段階と、をさらに備える
項目９または１０に記載の方法。
［項目１２］
前記第１のベクトルレジスタ内の１または複数のデータエレメントがさらに計算を必要としなくなるまで、前記第１のベクトルレジスタ内の複数のデータエレメントに対し再帰

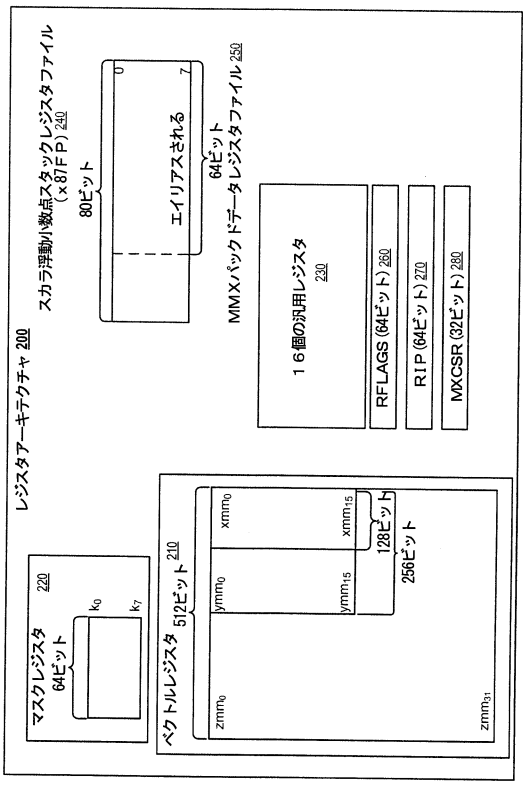
計算を実行する段階と、
前記第１のマスクレジスタを用いて、前記１または複数のデータエレメントが前記複数のターゲットデータエレメントであると示す段階と、をさらに備える
項目９から１１の何れか一項に記載の方法。
［項目１３］
前記第２のベクトルレジスタ内の前記与えられた数の複数のソースデータエレメントは、前記第２のベクトルレジスタ内の複数の下位のデータエレメントまたは複数の上位のデータエレメントのいずれかであり、前記第１のベクトルレジスタ内の前記与えられた数の複数のターゲットデータエレメントは、前記第１のベクトルレジスタ内の複数の下位のデータエレメントまたは複数の上位のデータエレメントのいずれかである
項目９から１２の何れか一項に記載の方法。
［項目１４］
前記第１のビット値は前記第２のビット値の逆である
項目９から１３の何れか一項に記載の方法。
［項目１５］
前記第１のビット値は前記第２のビット値と同一である
項目９から１４の何れか一項に記載の方法。
［項目１６］
前記与えられた数は、前記第１のビット値を有する前記第１のマスクレジスタ内のマスクビットの数、及び、前記第２のビット値を有する前記第２のマスクレジスタ内のマスクビットの数のうちの小さい方である
項目９から１５の何れか一項に記載の方法。
［項目１７］
複数の入力データエレメントを含む入力データアレイをストアするメモリと、
第１のマスクレジスタ、第２のマスクレジスタ、第１のベクトルレジスタ、及び第２のベクトルレジスタを含むレジスタファイルと、
前記メモリ及び前記レジスタファイルに連結される実行回路と、を備え、前記実行回路は、複数のイタレーションの間、前記第１のベクトルレジスタに対し再帰計算を実行し、前記複数のイタレーションのうちの２回以上は、
前記入力データアレイから前記第２のベクトルレジスタへ前記複数の入力データエレメントのうちの少なくとも一部をロードするベクトルロードオペレーションと、
前記第２のベクトルレジスタ内の前記複数の入力データエレメントを前記第１のベクトルレジスタへ移動させるベクトル移動オペレーションと、
前記第１のマスクレジスタ及び前記第２のマスクレジスタを更新して、それぞれ、さらに計算を必要とする前記第１のベクトルレジスタ及び前記第２のベクトルレジスタ内の、複数のデータエレメントを特定するマスク更新オペレーションと、
前記第１のベクトルレジスタ内の前記複数のデータエレメントに対して実行するベクトル演算オペレーションと、
前記ベクトル演算オペレーションの複数の結果を前記メモリへストアするベクトルストアオペレーションからなる複数のオペレーションを含む
システム。
［項目１８］
前記実行回路は、前記第１のベクトルレジスタ内の与えられた数の複数のターゲットデータエレメントを、前記第２のベクトルレジスタ内の前記与えられた数の複数のデータエレメントで置き換える前記ベクトル移動オペレーションを実行し、各ソースデータエレメントは、第２のビット値を有する前記第２のマスクレジスタ内のマスクビットに対応し、各ターゲットデータエレメントは、第１のビット値を有する前記第１のマスクレジスタ内のマスクビットに対応する
項目１７に記載のシステム。
［項目１９］

前記第１のビット値は前記第２のビット値の逆である
項目１７に記載のシステム。
［項目２０］
前記第１のビット値は前記第２のビット値と同一である
項目１７に記載のシステム。
［項目２１］
前記与えられた数は、前記第１のビット値を有する前記第１のマスクレジスタ内のマスクビットの数、及び、前記第２のビット値を有する前記第２のマスクレジスタ内のマスクビットの数のうちの小さい方である
項目１７に記載のシステム。
［項目２２］
前記実行回路は前記マスク更新オペレーションを実行して、前記第１のマスクレジスタ及び前記第２のマスクレジスタを更新し、前記第１のマスクレジスタ及び前記第２のマスクレジスタの複数のマスクビットがそれぞれ、さらに計算を必要とする、前記第１のベクトルレジスタ及び前記第２のベクトルレジスタの、複数の対応するデータエレメントを特定するようにする
項目１７から２１の何れか一項に記載のシステム。
［項目２３］
前記第１のベクトルレジスタ、前記第１のマスクレジスタ、前記第２のベクトルレジスタ、及び前記第２のマスクレジスタを複数のオペランドとして特定するベクトル移動命令に応答して、前記実行回路は前記ベクトル移動オペレーションを実行する
項目１７から２２の何れか一項に記載のシステム。
［項目２４］
前記第１のマスクレジスタ及び前記第２のマスクレジスタを複数のオペランドとして特定するマスク更新命令に応答して、前記実行回路は前記マスク更新オペレーションを実行する
項目１７から２３の何れか一項に記載のシステム。

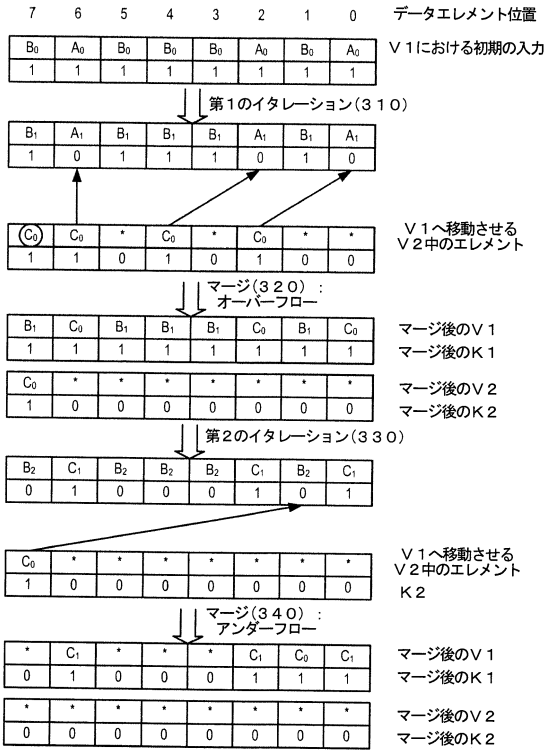
【図１】



【図２】



【図３】



【図４Ａ】

401

```
rwmaskupdate(K1,K2)
for (i = 0, k = 0; i < KL; i++){
        if (!K1[i]){
                for (; k < KL; k++){
                        if (K2[k]){
                                K2[k] = 0;
                                K1[i] = 1;
                                k++;
                                break;
                        }
                }
        }
}
```

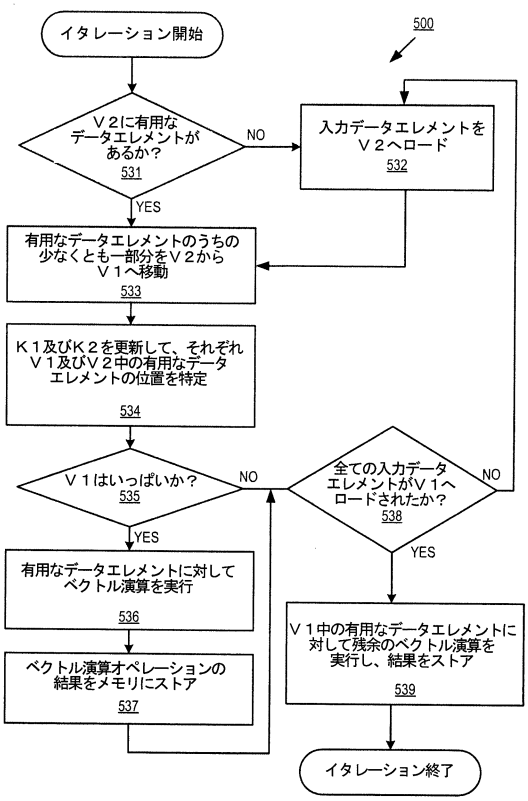
402

```
sparsemov(K1,V1,K2,V2)
for (i = 0, k = 0; i < KL; i++){
        if (!K1[i]){
                for(; k < KL; k++){
                        if (K2[k]){
                                V1[i] = V2[k];
                                k++;
                                break;
                        }
                }
        }
}
```

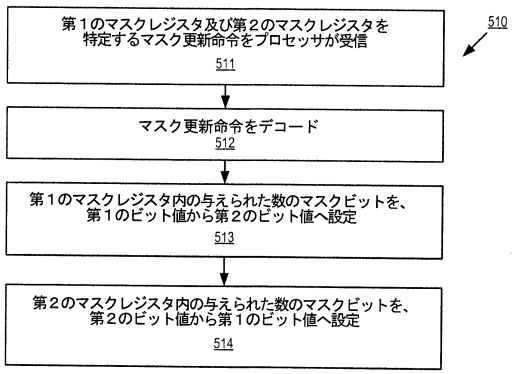
【図４Ｂ】

```
400
i = 0;                                  // ループカウンタを初期化
v_index = -1:-2:...:-KL+1:-KL           // インデックスの初期ベクトル          410
v_KL = KL:KL:....:KL:KL                 // インデックスのベクトルに対するインクリメント
K1 = 0;                                 // アキュムレータは初期は空である
K2 = 0;                                 // オーバーフローは未だない
do{                                                                         420
  if (K2 == 0){                         // 前のオーバーフローからエレメントが１つも残されていないならば
    V2 = vector_load(X[i+KL-1:i]);      // Ｘアレイの新しいＫＬエレメントをロード
    K2 = condition(V2);                 // 新しいエレメントに対してリードマスクを生成
    i += KL;                            // ループカウンタをインクリメント
    v_index += v_KL;                    // インデックスベクトルをインクリメント
  }                                     // その他はリードマスクＫ２で続行
431 sparsemov(K1, V1, K2, V2);          // Ｖ１．．．に新しいエレメントを追加   430
432 sparsemov(K1, V3, K2, v_index);     // Ｖ３にそれらのインデックスを追加
433 rwmaskupdate(K1,K2);                // リードマスク及びライトマスクを更新
IsFullMask(K1){                         // いっぱいになったアキュムレータのみに対して
                                                                            440
  do{
    V1 = computation(V1);               // 蓄積されたデータに対するデンスコンピューテーション
    K1 = condition(V1);                 // 計算後、条件をチェック
  }while(K1 == 0xFFFF)                  //アキュムレータがまだいっぱい(Ｋ１内が全て１)であるかどうかチェック
  scatter(knot(K1),V1,V3,X);            //インデックスＶ３を有するＶ１からＸアレイへエレメントを分散
}
}while((i < N) || (K2 != 0));           // Ｖ２内にもっと入力ストリームまたは新しいデータが
                                        // ある場合、続行
                                        // 残余の計算を開始
K3 = K1;                                // 最終の分散用に残余のマスクをストア   450
do{
  V1{K1} = computation(V1);             // マスクＫ１下で計算
  K1 = condition(V1);                   // 計算後、条件をチェック
}while(K1 != 0)                         // 成すべきことが何も残されていないかどうかチェック
scatter(K3,V1,V3,X);                    // インデックスＶ３を有するＶ１からＸアレイへ
                                        // 残余のエレメントを分散
```

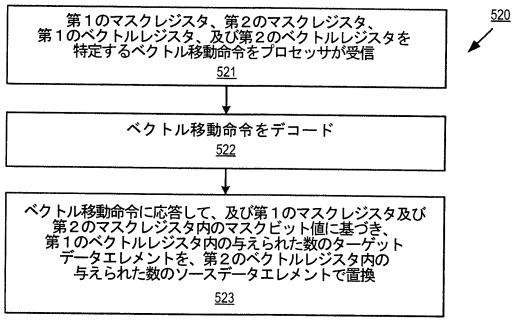
【図５Ａ】



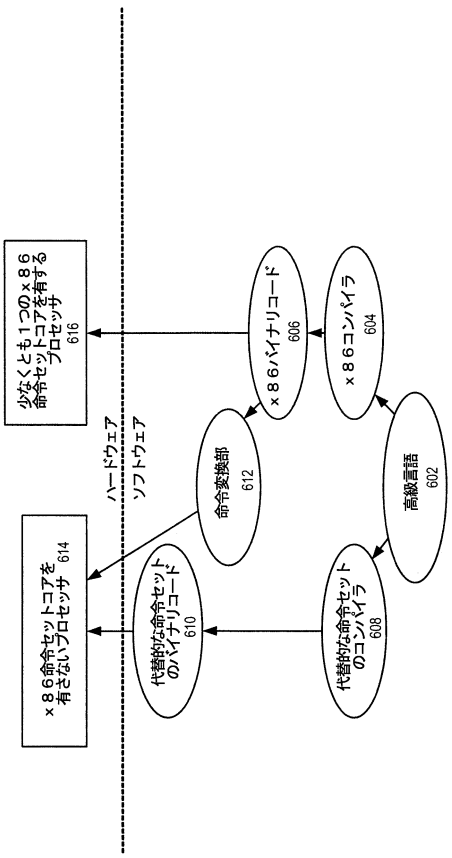
【図５Ｂ】



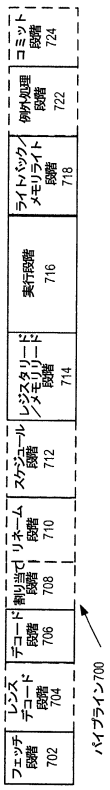
【図５Ｃ】



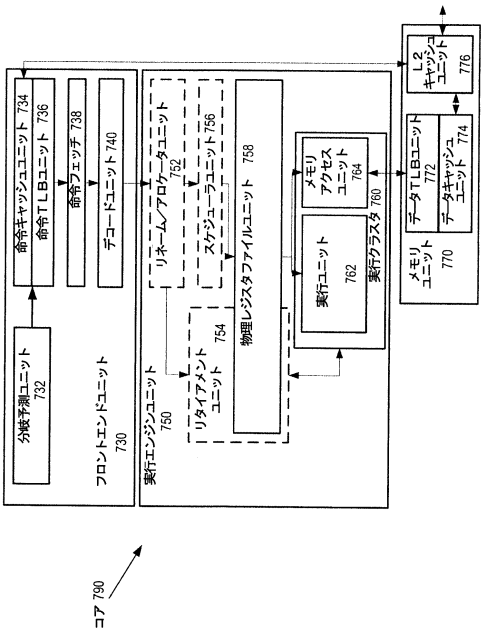
【図６】



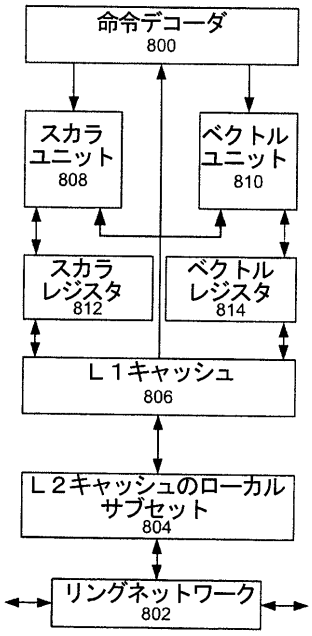
【図７Ａ】



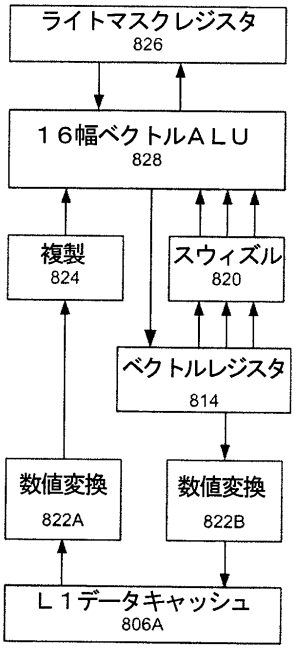
【図７Ｂ】



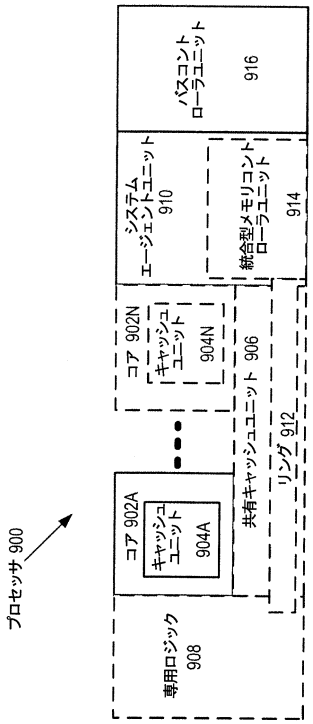
【図８Ａ】



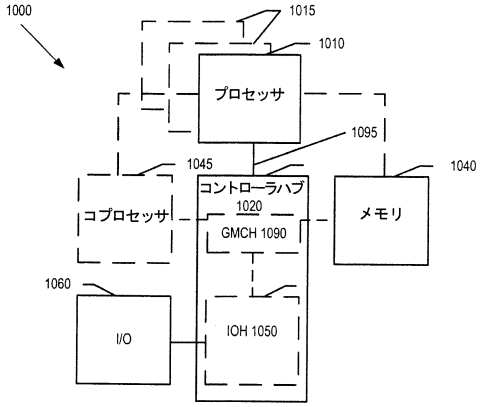
【図８Ｂ】



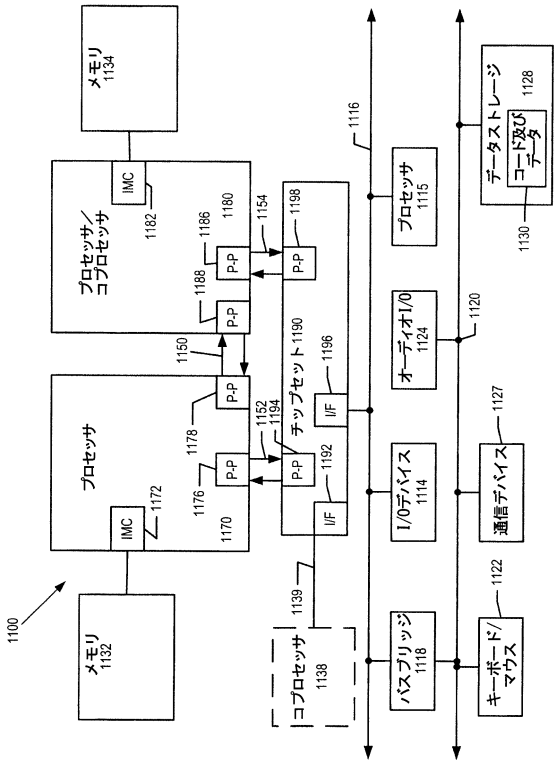
【図９】



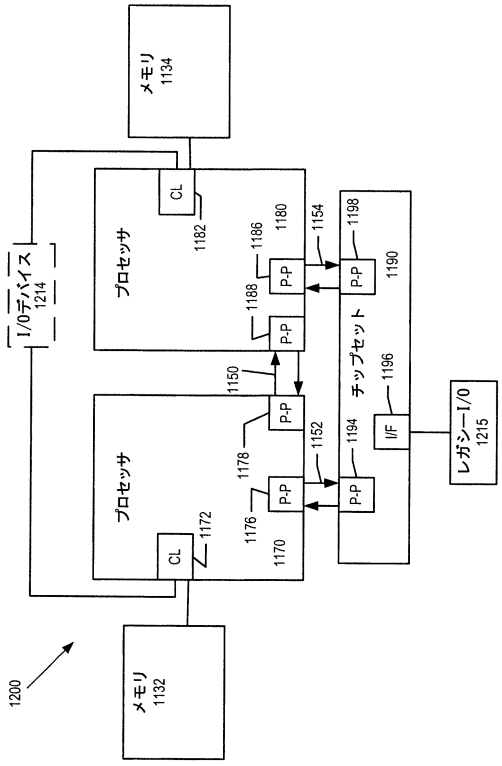
【図１０】



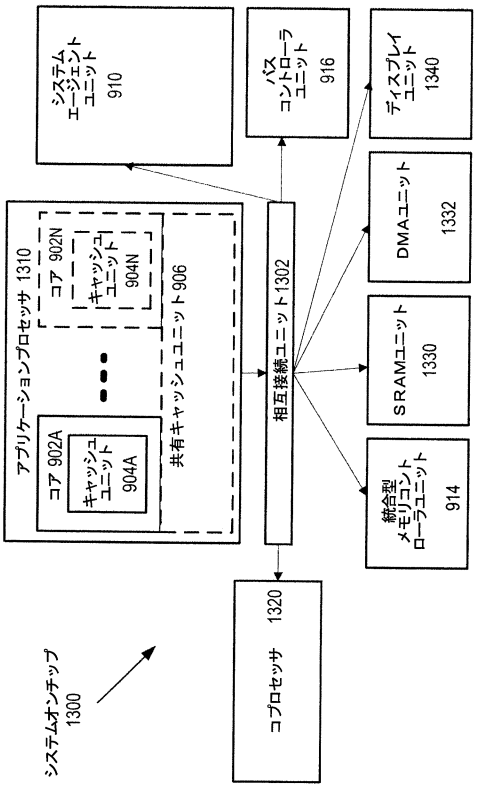
【図１１】



【図１２】



【図１３】


システムエージェントユニット 910
バスコントローラユニット 916
ディスプレイユニット 1340
アプリケーションプロセッサ 1310
コア 902N
キャッシュユニット 904N
コア 902A
キャッシュユニット 904A
共有キャッシュユニット 906
相互接続ユニット 1302
DMAユニット 1332
SRAMユニット 1330
統合型メモリコントローラユニット 914
コプロセッサ 1320
システムオンチップ 1300

フロントページの続き

(72)発明者　ナライキン、アンドレー
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　ヒューズ、クリストファー
アメリカ合衆国　９５０５４　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・２２００　インテル・コーポレーション内

審査官　清木　泰

(56)参考文献　特表２０１５－５２４９７８（ＪＰ，Ａ）
特開平１１－１２６２００（ＪＰ，Ａ）
特開平１１－３２２６（ＪＰ，Ａ）
特開平７－１４１３２６（ＪＰ，Ａ）
特開平６－４４２９２（ＪＰ，Ａ）
特開昭６０－５９４６９（ＪＰ，Ａ）
米国特許第７４８０７８７（ＵＳ，Ｂ１）
NECスーパーコンピュータSX-9シリーズ　中央処理装置機能説明書　UOM-G1AZ124-1，日本電気株式会社，２００８年　2月，初版，Pages:219-220
廣松悠介，黒田久泰，金田康正，複雑な制御構造を持つプログラムのSIMD命令セットによる最適化，情報処理学会論文誌，日本，社団法人情報処理学会，２００７年　3月１５日，第48巻，第SIG4(PRO32)号，Pages:62-72

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／３０－　９／４２
Ｇ０６Ｆ１７／１６
Ｇ０６Ｆ　８／４１
Ｇ０６Ｆ１５／８０
Ｇ０６Ｆ　７／５７－　７／５７５