(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2016-510461
(P2016-510461A)
(43)公表日　平成28年4月7日(2016.4.7)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/315	(2006.01)	G06F	9/30	340D	5B033
G06F 17/16	(2006.01)	G06F	17/16	D	5B056
		G06F	17/16	E	

審査請求 未請求　予備審査請求 未請求　（全 49 頁）

(21)出願番号　特願2015-553179 (P2015-553179)
(86)(22)出願日　平成25年11月21日 (2013.11.21)
(85)翻訳文提出日　平成27年6月24日 (2015.6.24)
(86)国際出願番号　PCT/IB2013/060309
(87)国際公開番号　WO2014/114997
(87)国際公開日　平成26年7月31日 (2014.7.31)
(31)優先権主張番号　13/748,543
(32)優先日　平成25年1月23日 (2013.1.23)
(33)優先権主張国　米国 (US)

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MACHINES CORPORATION
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
New Orchard Road, Armonk, New York 10504, United States of America
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】ＶＥＣＴＯＲＥＬＥＭＥＮＴＲＯＴＡＴＥＡＮＤＩＮＳＥＲＴＵＮＤＥＲＭＡＳＫ命令を処理するためのコンピュータ・プログラム、コンピュータ・システム及び方法

(57)【要約】
【課題】　中央演算処理ユニットにおいてマシン命令を実行するためのコンピュータ・プログラム、コンピュータ・システム及び方法を提供する。
【解決手段】　Ｖｅｃｔｏｒ　Ｅｌｅｍｅｎｔ　Ｒｏｔａｔｅ　ａｎｄ　Ｉｎｓｅｒｔ　Ｕｎｄｅｒ　Ｍａｓｋ命令である。命令の第２のオペランドの各要素が、指定されたビット数だけ指定された方向にローテートされる。１に設定された命令の第３のオペランド内の各ビットについて、第２のオペランド内のローテートされた要素の対応するビットが、命令の第１のオペラ内の対応するビットに置き換わる。
【選択図】　図２０

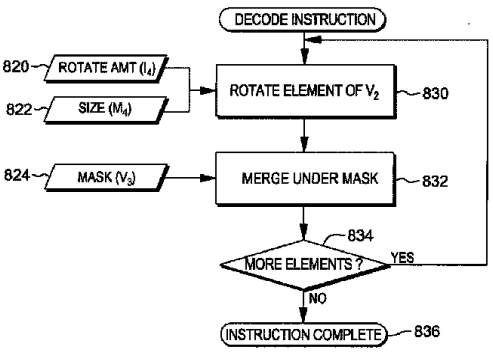


FIG. 8B

【特許請求の範囲】
【請求項１】
中央演算処理ユニットにおいてマシン命令を実行するためのコンピュータ・プログラムであって、前記コンピュータ・プログラムは、方法を実施するための命令を含み、
前記方法は、
プロセッサにより、実行のためのマシン命令を取得することであって、前記マシン命令は、コンピュータ・アーキテクチャに従ったコンピュータ実行のために定められ、
Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ Ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ操作を識別するオペコードを与えるための少なくとも１つのオペコード・フィールドと、
第１のオペランドを含む第１のレジスタを指定するのに用いられる第１のレジスタ・フィールドと、
第２のオペランドを含む第２のレジスタを指定するのに用いられる第２のレジスタ・フィールドと、
第３のオペランドを含む第３のレジスタを指定するのに用いられる第３のレジスタ・フィールドと、
を含む、取得することと、
前記マシン命令を実行することと、
を含み、
前記実行することは、
前記第２のオペランドの１つ又は複数の要素を、指定されたローテート量だけ選択された方向にローテートさせることと、
特定の値を有する１つ又は複数のデータ単位に関して前記第３のオペランドをチェックすることと、
前記特定の値をもつ前記１つ又は複数のデータ単位を有する前記第３のオペランドに基づいて、前記第１のオペランド内の対応するデータ単位の１つ又は複数の値を、ローテートされた前記第２のオペランド内の対応するデータ単位の１つ又は複数の値に置き換えることと、
を含む、コンピュータ・プログラム。
【請求項２】
前記選択された方向は左方向を含み、前記指定されたローテート量は指定されたビット数を含む、請求項１に記載のコンピュータ・プログラム。
【請求項３】
前記ローテートさせることは、前記第２のオペランドの各要素を前記指定されたビット数だけ左へローテートさせることを含み、要素の左端ビット位置から外にシフトされた各ビットは、前記要素の右端ビット位置に再び入る、請求項２に記載のコンピュータ・プログラム。
【請求項４】
前記第３のオペランドは複数の要素を含み、前記複数の要素は複数のマスクを含み、前記複数のマスクのうちの１つのマスクは複数のビットを含む、請求項１に記載のコンピュータ・プログラム。
【請求項５】
前記１つ又は複数のデータ単位は１つ又は複数のビットを含み、前記特定の値は１を含み、前記置き換えることは、１に設定された前記第３のオペランド内の各ビットについて、前記第１のオペランド内の対応するビットの値を、ローテートされた前記第２のオペランド内の対応するビットの値に置き換えることを含む、請求項４に記載のコンピュータ・プログラム。
【請求項６】
前記第２のオペランドは複数の要素を含み、前記指定されたローテート量は、前記マシン命令の第４のオペランド内に含まれ、前記第４のオペランドは、前記第２のオペランド

の各要素をローテートするためのビット数を指定する符号なし２進整数を含み、
前記方法は、
前記符号なし２進整数が選択されたオペランドの要素のビット数より大きいかどうかを判断することと、
前記符号なし２進整数が前記選択されたオペランドの前記要素の前記ビット数より大きいことに基づいて、前記符号なし２進整数を法として前記選択されたオペランドの前記要素の前記ビット数だけ減らすことをさらに含む、請求項１に記載のコンピュータ・プログラム。
【請求項７】
前記方法は、前記マシン命令のマスク・フィールドを用いて、前記選択されたオペランドの前記要素の前記ビット数を求めることをさらに含み、前記マスク・フィールドは、前記マシン命令の１つ又は複数のオペランドの要素のサイズを示すための要素サイズ制御を含む、請求項６に記載のコンピュータ・プログラム。
【請求項８】
前記マシン命令は、１つ又は複数のレジスタを指定するのに用いられる拡張フィールドをさらに含み、前記第１のレジスタ・フィールドと前記拡張フィールドの第１の部分とが組み合わされて前記第１のレジスタが指定され、前記第２のレジスタ・フィールドと前記拡張フィールドの第２の部分とが組み合わされて前記第２のレジスタが指定され、前記第３のレジスタ・フィールドと前記拡張フィールドの第３の部分とが組み合わされて前記第３のレジスタが指定される、請求項１に記載のコンピュータ・プログラム。
【請求項９】
前記第２のオペランドは１つ又は複数の第２のオペランド要素を含み、前記選択された方向は左方向であり、前記データ単位はビットを含み、前記指定されたローテート量は指定されたビット数を含み、前記第３のオペランドは１つ又は複数の第３のオペランド要素を含み、各々の第３のオペランド要素は複数のビットを有するマスクを含み、
前記ローテートさせることは、前記第２のオペランドの各要素を前記指定されたビット数だけ左へローテートさせることを含み、要素の左端ビット位置から外にシフトされた各ビットは、前記要素の右端ビット位置に再び入り、
前記置き換えることは、１に設定された前記第３のオペランド内の各ビットについて、前記第１のオペランド内の対応するビットの値を、ローテートされた前記第２のオペランドの対応するビットの値に置き換えることを含む、請求項１に記載のコンピュータ・プログラム。
【請求項１０】
前記マシン命令はマスク・フィールドをさらに含み、前記マスク・フィールドは、前記第１のオペランド、前記第２のオペランド及び前記第３のオペランドのうちの１つ又は複数の要素のサイズを示すための要素サイズ制御を含む、請求項１に記載のコンピュータ・プログラム。
【請求項１１】
中央演算処理ユニットにおいてマシン命令を実行するためのコンピュータ・システムであって、前記コンピュータ・システムは、
メモリと、
前記メモリと通信を行うプロセッサと、
を含み、方法を実施するように構成され、
前記方法は、
プロセッサにより、実行のためのマシン命令を取得することであって、前記マシン命令は、コンピュータ・アーキテクチャに従ったコンピュータ実行のために定められ、
Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ Ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ操作を識別するオペコードを与えるための少なくとも１つのオペコード・フィールドと、
第１のオペランドを含む第１のレジスタを指定するのに用いられる第１のレジスタ・

フィールドと、
　第２のオペランドを含む第２のレジスタを指定するのに用いられる第２のレジスタ・フィールドと、
　第３のオペランドを含む第３のレジスタを指定するのに用いられる第３のレジスタ・フィールドと、
を含む、取得することと、
　前記マシン命令を実行することと、
を含み、
　前記実行することは、
　前記第２のオペランドの１つ又は複数の要素を、指定されたローテート量だけ選択された方向にローテートさせることと、
　特定の値を有する１つ又は複数のデータ単位に関して前記第３のオペランドをチェックすることと、
　前記特定の値をもつ前記１つ又は複数のデータ単位を有する前記第３のオペランドに基づいて、前記第１のオペランド内の対応するデータ単位の１つ又は複数の値を、ローテートされた前記第２のオペランド内の対応するデータ単位の１つ又は複数の値に置き換えることと、
を含む、コンピュータ・システム。
【請求項１２】
　前記選択された方向は左方向を含み、前記指定されたローテート量は指定されたビット数を含み、前記ローテートさせることは、前記第２のオペランドの各要素を前記指定されたビット数だけ左へローテートさせることを含み、要素の左端ビット位置から外にシフトされた各ビットは、前記要素の右端ビット位置に再び入る、請求項１１に記載のコンピュータ・システム。
【請求項１３】
　前記第３のオペランドは複数の要素を含み、前記複数の要素は複数のマスクを含み、前記複数のマスクのうちの１つのマスクは複数のビットを含み、前記１つ又は複数のデータ単位は１つ又は複数のビットを含み、前記特定の値は１を含み、前記置き換えることは、１に設定された前記第３のオペランド内の各ビットについて、前記第１のオペランド内の対応するビットの値を、ローテートされた前記第２のオペランド内の対応するビットの値に置き換えることを含む、請求項１１に記載のコンピュータ・システム。
【請求項１４】
　前記第２のオペランドは複数の要素を含み、前記指定されたローテート量は、前記マシン命令の第４のオペランド内に含まれ、前記第４のオペランドは、前記第２のオペランドの各要素をローテートするためのビット数を指定する符号なし２進整数を含み、
　前記方法は、
　前記符号なし２進整数が選択されたオペランドの要素のビット数より大きいかどうかを判断することと、
　前記符号なし２進整数が前記選択されたオペランドの前記要素の前記ビット数より大きいことに基づいて、前記符号なし２進整数を法として前記選択されたオペランドの前記要素の前記ビット数だけ減らすことをさらに含む、請求項１１に記載のコンピュータ・システム。
【請求項１５】
　前記マシン命令は、１つ又は複数のレジスタを指定するのに用いられる拡張フィールドをさらに含み、前記第１のレジスタ・フィールドと前記拡張フィールドの第１の部分とが組み合わされて前記第１のレジスタが指定され、前記第２のレジスタ・フィールドと前記拡張フィールドの第２の部分とが組み合わされて前記第２のレジスタが指定され、前記第３のレジスタ・フィールドと前記拡張フィールドの第３の部分とが組み合わされて前記第３のレジスタが指定される、請求項１１に記載のコンピュータ・システム。
【請求項１６】

前記第２のオペランドは１つ又は複数の第２のオペランド要素を含み、前記選択された方向は左方向であり、前記データ単位はビットを含み、前記指定されたローテート量は指定されたビット数を含み、前記第３のオペランドは１つ又は複数の第３のオペランド要素を含み、各々の第３のオペランド要素は複数のビットを有するマスクを含み、
前記ローテートさせることは、前記第２のオペランドの各要素を前記指定されたビット数だけ左へローテートさせることを含み、要素の左端ビット位置から外にシフトされた各ビットは、前記要素の右端ビット位置に再び入り、
前記置き換えることは、１に設定された前記第３のオペランド内の各ビットについて、前記第１のオペランド内の対応するビットの値を、ローテートされた前記第２のオペランドの前記対応するビットの値に置き換えることを含む、請求項１１に記載のコンピュータ・システム。

【請求項１７】
前記マシン命令はマスク・フィールドをさらに含み、前記マスク・フィールドは、前記第１のオペランド、前記第２のオペランド及び前記第３のオペランドの要素のサイズを示すための要素サイズ制御を含む、請求項１１に記載のコンピュータ・システム。

【請求項１８】
中央演算処理ユニットにおいてマシン命令を実行する方法であって、
プロセッサにより、実行のためのマシン命令を取得することであって、前記マシン命令は、コンピュータ・アーキテクチャに従ったコンピュータ実行のために定められ、
Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ Ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ操作を識別するオペコードを与えるための少なくとも１つのオペコード・フィールドと、
第１のオペランドを含む第１のレジスタを指定するのに用いられる第１のレジスタ・フィールドと、
第２のオペランドを含む第２のレジスタを指定するのに用いられる第２のレジスタ・フィールドと、
第３のオペランドを含む第３のレジスタを指定するのに用いられる第３のレジスタ・フィールドと、
を含む、取得することと、
前記マシン命令を実行することと、
を含み、
前記実行することは、
前記第２のオペランドの１つ又は複数の要素を、指定されたローテート量だけ選択された方向にローテートさせることと、
特定の値を有する１つ又は複数のデータ単位に関して前記第３のオペランドをチェックすることと、
前記特定の値をもつ前記１つ又は複数のデータ単位を有する前記第３のオペランドに基づいて、前記第１のオペランド内の対応するデータ単位の１つ又は複数の値を、ローテートされた前記第２のオペランド内の対応するデータ単位の１つ又は複数の値に置き換えることと、
を含む、方法。

【請求項１９】
前記第２のオペランドは１つ又は複数の第２のオペランド要素を含み、前記選択された方向は左方向であり、前記データ単位はビットを含み、前記指定されたローテート量は指定されたビット数を含み、前記第３のオペランドは１つ又は複数の第３のオペランド要素を含み、各々の第３のオペランド要素は複数のビットを有するマスクを含み、
前記ローテートさせることは、前記第２のオペランドの各要素を前記指定されたビット数だけ左へローテートさせることを含み、要素の左端ビット位置から外にシフトされた各ビットは、前記要素の右端ビット位置に再び入り、
前記置き換えることは、１に設定された前記第３のオペランド内の各ビットについて、

前記第１のオペランド内の対応するビットの値を、ローテートされた前記第２のオペランド内の対応するビットの値に置き換えることを含む、請求項１８に記載の方法。
【請求項２０】
前記マシン命令はマスク・フィールドをさらに含み、前記マスク・フィールドは、前記第１のオペランド、前記第２のオペランド及び前記第３のオペランドの要素のサイズを示すための要素サイズ制御を含む、請求項１８に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
１つ又は複数の態様は、一般に、コンピューティング環境内での処理に関し、特定的には、こうした環境内でのベクトル処理に関する。
【背景技術】
【０００２】
コンピューティング環境内での処理は、１つ又は複数の中央演算処理ユニット（ＣＰＵ）の動作を制御することを含む。通常、中央演算処理ユニットの動作は、ストレージ内の命令によって制御される。命令は、異なる形式を有し、多くの場合、種々の動作の実施に用いられるレジスタを指定することができる。
【０００３】
中央演算処理ユニットのアーキテクチャによって、例えば、例を挙げると汎用レジスタ、専用レジスタ、浮動小数点レジスタ、及び／又はベクトル・レジスタを含む種々のタイプのレジスタを使用することができる。異なるタイプのレジスタを、異なるタイプの命令と共に使用することができる。例として、浮動小数点レジスタは、浮動小数点命令により用いられる浮動小数点数を格納し、ベクトル・レジスタは、ベクトル命令を含む、Ｓｉｎｇｌｅ Ｉｎｓｔｒｕｃｔｉｏｎ、Ｍｕｌｔｉｐｌｅ Ｄａｔａ（ＳＩＭＤ）命令によって実施されるベクトル処理のためのデータを保持する。
【先行技術文献】
【特許文献】
【０００４】
【特許文献１】米国特許第５，５５１，０１３号明細書
【特許文献２】米国特許第６，００９，２６１号明細書
【特許文献３】米国特許第５，５７４，８７３号明細書
【特許文献４】米国特許第６，３０８，２５５号明細書
【特許文献５】米国特許第６，４６３，５８２号明細書
【特許文献６】米国特許第５，７９０，８２５号明細書
【非特許文献】
【０００５】
【非特許文献１】ＩＢＭ（登録商標）、「ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ Ｐｒｉｎｃｉｐｌｅｓ ｏｆ Ｏｐｅｒａｔｉｏｎ」、ＩＢＭ（登録商標）出版番号ＳＡ２２－７８３２－０９、第１０版、２０１２年９月
【非特許文献２】インターナショナル・ビジネス・マシーンズ・コーポレーション、「Ｐｏｗｅｒ ＩＳＡ（商標） Ｖｅｒｓｉｏｎ ２．０６ 改定Ｂ」、２０１０年７月２３日
【非特許文献３】「Ｉｎｔｅｌ（登録商標）６４ ａｎｄ ＩＡ－３２ Ａｒｃｈｉｔｅｃｔｕｒｅｓ Ｄｅｖｅｌｏｐｅｒ’ｓ Ｍａｎｕａｌ：Ｖｏｌ．２Ｂ，Ｉｎｓｔｒｕｃｔｉｏｎｓ Ｓｅｔ Ｒｅｆｅｒｅｎｃｅ，Ａ－Ｌ」、注文番号２５３６６６－０４５ＵＳ、２０１３年１月
【非特許文献４】「Ｉｎｔｅｌ（登録商標）６４ ａｎｄ ＩＡ－３２ Ａｒｃｈｉｔｅｃｔｕｒｅｓ Ｄｅｖｅｌｏｐｅｒ’ｓ Ｍａｎｕａｌ：Ｖｏｌ．２Ｂ，Ｉｎｓｔｒｕｃｔｉｏｎｓ Ｓｅｔ Ｒｅｆｅｒｅｎｃｅ，Ｍ－Ｚ」、注文番号２５３６６７－０４５ＵＳ、２０１３年１月

【発明の概要】
【発明が解決しようとする課題】
【０００６】
中央演算処理ユニットにおいてマシン命令を実行するためのコンピュータ・プログラム、コンピュータ・システム及び方法を提供する。
【課題を解決するための手段】
【０００７】
マシン命令を実行するためのコンピュータ・プログラム製品を提供することにより、従来技術の欠点が克服され、利点がもたらされる。このコンピュータ・プログラム製品は、処理回路により読み出し可能であり、且つ、方法を実施するための、処理回路による実行される命令を格納するコンピュータ可読ストレージ媒体を含む。この方法は、例えば、プロセッサにより、実行のためのマシン命令を取得することであって、マシン命令は、コンピュータ・アーキテクチャに従ったコンピュータ実行のために定められ、且つ、Ｖｅｃｔｏｒ　Ｅｌｅｍｅｎｔ　Ｒｏｔａｔｅ　Ａｎｄ　Ｉｎｓｅｒｔ　Ｕｎｄｅｒ　Ｍａｓｋ操作を識別するオペコードを与えるための少なくとも１つのオペコード・フィールドと、第１のオペランドを含む第１のレジスタを指定するのに用いられる第１のレジスタ・フィールドと、第２のオペランドを含む第２のレジスタを指定するのに用いられる第２のレジスタ・フィールドと、第３のオペランドを含む第３のレジスタを指定するのに用いられる第３のレジスタ・フィールドとを含む、取得することと、マシン命令を実行することとを含み、実行することは、第２のオペランドの１つ又は複数の要素を、指定されたローテート量だけ選択された方向にローテートさせることと、特定の値を有する１つ又は複数のデータ単位に関して第３のオペランドをチェックすることと、特定の値をもつ１つ又は複数のデータ単位を有する第３のオペランドに基づいて、第１のオペランド内の対応するデータ単位の１つ又は複数の値を、ローテートされた第２のオペランド内の対応するデータ単位の１つ又は複数の値に置き換えることと、を含む。
【０００８】
１つ又は複数の態様に関連する方法及びシステムも、本明細書で説明され、特許請求される。さらに、１つ又は複数の態様に関連するサービスも、本明細書で説明され、特許請求され得る。
１つ又は複数の態様の技術を通じて、付加的な特徴及び利点が実現される。他の実施形態及び態様は、本明細書で詳細に説明され、特許請求の範囲の一部と見なされる。
【０００９】
１つ又は複数の態様が、本明細書の最後にある特許請求の範囲において、例として具体的に示され、明確に特許請求されている。上記及び他の目的、特徴、並びに利点は、添付図面と関連して用いられる以下の詳細な説明から明らかである。
【図面の簡単な説明】
【００１０】
【図１】１つ又は複数の態様を組み込み、用いるためのコンピューティング環境の一例を示す。
【図２】１つ又は複数の態様を組み込み、用いるためのコンピューティング環境の別の例を示す。
【図３】図２のメモリの更なる詳細を示す。
【図４】レジスタ・ファイルの一例を示す。
【図５】Ｖｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令の形式の一例を示す。
【図６】図５のＶｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令の第３のオペランドのビット値の一例を示す。
【図７】図５のＶｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令と関連付けられた論理の一実施形態を示す。
【図８】図５のＶｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃ

ｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令の実行のブロック図の一例を示す。
【図９】２進浮動小数点データの種々のクラスの定義の一例を示す。
【図１０】Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令の形式の一例を示す。
【図１１】図１０のＶｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令と関連付けられた論理の一実施形態を示す。
【図１２】図１０のＶｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令の実行のブロック図の一例を示す。
【図１３】Ｖｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ　Ａｃｃｕｍｕｌａｔｅ命令の形式の一例を示す。
【図１４】図１３のＶｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ　Ａｃｃｕｍｕｌａｔｅ命令と関連付けられた論理の一実施形態を示す
【図１５】図１３のＶｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ　Ａｃｃｕｍｕｌａｔｅ命令の実行のブロック図の一例を示す。
【図１６】Ｖｅｃｔｏｒ　Ｇｅｎｅｒａｔｅ　Ｍａｓｋ命令の形式の一例を示す。
【図１７】図１６のＶｅｃｔｏｒ　Ｇｅｎｅｒａｔｅ　Ｍａｓｋ命令と関連付けられた論理の一実施形態を示す。
【図１８】図１６のＶｅｃｔｏｒ　Ｇｅｎｅｒａｔｅ　Ｍａｓｋ命令の実行のブロック図の一例を示す。
【図１９】Ｖｅｃｔｏｒ　Ｅｌｅｍｅｎｔ　Ｒｏｔａｔｅ　ａｎｄ　Ｉｎｓｅｒｔ　Ｕｎｄｅｒ　Ｍａｓｋ命令の形式の一例を示す。
【図２０】図１９のＶｅｃｔｏｒ　Ｅｌｅｍｅｎｔ　Ｒｏｔａｔｅ　ａｎｄ　Ｉｎｓｅｒｔ　Ｕｎｄｅｒ　Ｍａｓｋ命令と関連付けられた論理の一実施形態を示す
【図２１】図１９のＶｅｃｔｏｒ　Ｅｌｅｍｅｎｔ　Ｒｏｔａｔｅ　ａｎｄ　Ｉｎｓｅｒｔ　Ｕｎｄｅｒ　Ｍａｓｋ命令の実行のブロック図の一例を示す。
【図２２】ベクトル例外コード（Ｖｅｃｔｏｒ　Ｅｘｃｅｐｔｉｏｎ　Ｃｏｄｅ）の一例を示す。
【図２３】図２２のベクトル例外コードを設定するための論理の一実施形態を示す。
【図２４】１つ又は複数の態様を組み込むコンピュータ・プログラム製品の一実施形態を示す。
【図２５】ホスト・コンピュータ・システムの一実施形態を示す。
【図２６】コンピュータ・システムの更に別の例を示す。
【図２７】コンピュータ・ネットワークを含むコンピュータ・システムの別の例を示す。
【図２８】コンピュータ・システムの種々の要素の一実施形態を示す。
【図２９】図２８のコンピュータ・システムの実行ユニットの一実施形態を示す。
【図３０】図２８のコンピュータ・システムの分岐ユニットの一実施形態を示す。
【図３１】図２８のコンピュータ・システムのロード／ストア・ユニットの一実施形態を示す。
【図３２】エミュレートされたホスト・コンピュータ・システムの一実施形態を示す。
【発明を実施するための形態】
【００１１】
１つ又は複数の態様によると、種々のベクトル命令、並びにベクトル例外処理を含むベクトル・ファシリティが提供される。本明細書で説明される命令の各々は、１つ又は複数のベクトル・レジスタ（本明細書では、ベクトルとも呼ばれる）を用いるＳｉｎｇｌｅ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｍｕｌｔｉｐｌｅ　Ｄａｔａ（ＳＩＭＤ）命令である。ベクトル・レジスタは、例えば、中央演算処理ユニット（ＣＰＵ）又は他のプロセッサの一部として利用可能な小容量のストレージである（例えば、主メモリではない）プロセッサ・レジスタ（ハードウェア・レジスタとも呼ばれる）である。各ベクトル・レジスタは、１つ又は複数の要素を有するベクトル・オペランドを含み、要素の長さは、例えば、１バイト、２バイト、４バイト、又は８バイトである。他の実施形態において、要素は他のサイズのものとすることができ、また、ベクトル命令はＳＩＭＤ命令である必要はない。

【００１２】
１つ又は複数の態様を組み込み、用いるためのコンピューティング環境の一実施形態が、図１を参照して説明される。コンピューティング環境１００は、例えば１つ又は複数のバス１０８及び／又は他の接続を介して互いに結合された、例えばプロセッサ１０２（例えば中央演算処理ユニット）、メモリ１０４（例えば主メモリ）、並びに１つ又は複数の入力／出力（Ｉ／Ｏ）デバイス及び／又はインターフェース１０６を含む。
【００１３】
一例において、プロセッサ１０２は、インターナショナル・ビジネス・マシーンズ・コーポレーションにより提供されるｚ／Ａｒｃｈｉｔｅｃｔｕｒｅに基づいており、同じくインターナショナル・ビジネス・マシーンズ・コーポレーションにより提供され、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅを実装する、Ｓｙｓｔｅｍ ｚサーバなどのサーバの一部である。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅの一実施形態は、ＩＢＭ（登録商標）の刊行物である非特許文献１に記載されている。一例において、プロセッサは、同じくインターナショナル・ビジネス・マシーンズ・コーポレーションにより提供される、ｚ／ＯＳなどのオペレーティング・システムを実行する。ＩＢＭ（登録商標）、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ（登録商標）、及びＺ／ＯＳ（登録商標）は、米国ニューヨーク州アーモンク所在のインターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。本明細書で使用される他の名称は、インターナショナル・ビジネス・マシーンズ・コーポレーション又は他の会社の登録商標、商標、又は製品名である場合がある。
【００１４】
さらに別の実施形態において、プロセッサ１０２は、インターナショナル・ビジネス・マシーンズ・コーポレーションにより提供されるＰｏｗｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅに基づいている。Ｐｏｗｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅの一実施形態は、非特許文献２に記載されている。Ｐｏｗｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅ（登録商標）は、インターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。
【００１５】
さらに別の実施形態において、プロセッサ１０２は、Ｉｎｔｅｌ Ｃｏｒｐｏｒａｔｉｏｎにより提供されるＩｎｔｅｌアーキテクチャに基づいている。Ｉｎｔｅｌアーキテクチャの一実施形態は、非特許文献３及び非特許文献４に記載されている。Ｉｎｔｅｌ（登録商標）は、カリフォルニア州サンタクララ所在のＩｎｔｅｌ Ｃｏｒｐｏｒａｔｉｏｎの登録商標である。
【００１６】
１つ又は複数の態様を組み込み、用いるためのコンピューティング環境の別の実施形態が、図２を参照して説明される。この例において、コンピューティング環境２００は、例えば１つ又は複数のバス２０８及び／又は他の接続を介して互いに結合された、例えばネイティブ中央演算処理プロセッサ２０２、メモリ２０４、並びに１つ又は複数の入力／出力（Ｉ／Ｏ）デバイス及び／又はインターフェース２０６を含む。例として、コンピューティング環境２００は、ニューヨーク州アーモンク所在のインターナショナル・ビジネス・マシーンズ・コーポレーションにより提供されるＰｏｗｅｒＰＣプロセッサ、ｐＳｅｒｉｅｓサーバ、又はｘＳｅｒｉｅｓサーバ、及びカリフォルニア州Ｐａｌｏ Ａｌｔｏ所在のＨｅｗｌｅｔｔ Ｐａｃｋａｒｄ Ｃｏ.，により提供される、Ｉｎｔｅｌ Ｉｔａｎｉｕｍ ＩＩプロセッサを搭載するＨＰ Ｓｕｐｅｒｄｏｍｅ、及び／又はインターナショナル・ビジネス・マシーンズ・コーポレーション、Ｈｅｗｌｅｔｔ Ｐａｃｋａｒｄ、Ｉｎｔｅｌ、Ｏｒａｃｌｅ、又はその他により提供されるアーキテクチャに基づく他のマシンを含むことができる。
【００１７】
ネイティブ中央演算処理ユニット２０２は、環境内での処理の際に用いられる、１つ又は複数の汎用レジスタ及び／又は１つ又は複数の専用レジスタなどの１つ又は複数のネイティブ・レジスタ２１０を含む。これらのレジスタは、任意の特定の時点での環境の状態を表す情報を含む。

【００１８】
さらに、ネイティブ中央演算処理ユニット２０２は、メモリ２０４に格納された命令及びコードを実行する。１つの具体的な例においては、中央演算処理ユニットは、メモリ２０４に格納されたエミュレータ・コード２１２を実行する。このコードにより、１つのアーキテクチャで構成された処理環境が、別のアーキテクチャをエミュレートすることが可能になる。例えば、エミュレータ・コード２１２は、ＰｏｗｅｒＰＣプロセッサ、ｐＳｅｒｉｅｓサーバ、ｘＳｅｒｉｅｓサーバ、ＨＰ　Ｓｕｐｅｒｄｏｍｅサーバなどのような、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ以外のアーキテクチャに基づくマシンが、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅをエミュレートし、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅに基づき開発されたソフトウェア及び命令を実行することを可能にする。

【００１９】
エミュレータ・コード２１２に関する更なる詳細が、図３を参照して説明される。メモリ２０４に格納されたゲスト命令２５０が、ネイティブＣＰＵ２０２のもの以外のアーキテクチャで実行されるように開発されたソフトウェア命令（例えば、マシン命令に関する）を含む。例えば、ゲスト命令２５０は、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅプロセッサ１０２上で実行されるように設計されているが、代わりに、例えばＩｎｔｅｌ　Ｉｔａｎｉｕｍ　ＩＩプロセッサとすることができるネイティブＣＰＵ２０２上でエミュレートされてもよい。一例において、エミュレータ・コード２１２は、メモリ２０４から１つ又は複数のゲスト命令２５０を取得し、取得した命令のためのローカル・バッファリングを随意的に提供するための命令フェッチ・ルーチン２５２を含む。エミュレータ・コード２１２はまた、取得したゲスト命令のタイプを判断し、ゲスト命令を１つ又は複数の対応するネイティブ命令２５６に変換するための命令変換ルーチン２５４も含む。この変換は、例えば、ゲスト命令により実施される機能を識別すること、及びその機能を実施するためのネイティブ命令を選択することを含む。

【００２０】
さらに、エミュレータ２１２は、ネイティブ命令を実行させるためのエミュレーション制御ルーチン２６０を含む。エミュレーション制御ルーチン２６０は、ネイティブＣＰＵ２０２に、既に取得した１つ又は複数のゲスト命令をエミュレートするネイティブ命令のルーチンを実行させ、こうした実行の最後に、制御を命令フェッチ・ルーチンに戻して、次のゲスト命令又はゲスト命令のグループの取得をエミュレートすることができる。ネイティブ命令２５６の実行は、データをメモリ２０４からレジスタにロードすること、データをレジスタから再びメモリに格納すること、又は変換ルーチンによって求められるような何らかのタイプの算術演算又は論理演算を実施することを含むことができる。

【００２１】
各ルーチンは、例えば、メモリに格納され、ネイティブ中央演算処理ユニット２０２によって実行される、ソフトウェアの形で実装される。他の例においては、ルーチン又は演算は、ファームウェア、ハードウェア、ソフトウェア、又はそれらの何らかの組み合わせの形で実装される。エミュレートされるプロセッサのレジスタは、ネイティブＣＰＵのレジスタ２１０又はメモリ２０４内の位置を使用して、エミュレートすることができる。実施形態において、ゲスト命令２５０、ネイティブ命令２５６、及びエミュレータ・コード２１２は、同一のメモリ内にあっても、又は異なるメモリ・デバイス間に分散されてもよい。

【００２２】
本明細書で用いられるファームウェアは、例えば、プロセッサのマイクロコード、ミリコード、及び／又はマクロコードを含む。ファームウェアは、例えば、上位レベルのマシン・コードの実装に用いられるハードウェア・レベルの命令及び／又はデータ構造体を含む。一実施形態において、ファームウェアは、例えば、典型的には、信頼できるソフトウェアを含むマイクロコード、又は基礎をなすハードウェアに特有のマイクロコードとして配信される独自のコードを含み、システム・ハードウェアへのオペレーティング・システムのアクセスを制御する。

【００２３】
一例において、取得され、変換され、実行されるゲスト命令２５０は、本明細書で説明される１つの命令である。１つのアーキテクチャ（例えば、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ）のものであるこの命令が、メモリからフェッチされ、変換され、別のアーキテクチャ（例えば、ＰｏｗｅｒＰＣ、ｐＳｅｒｉｅｓ、ｘＳｅｒｉｅｓ、Ｉｎｔｅｌ等）のネイティブ命令２５６のシーケンスとして表される。次に、これらのネイティブ命令が実行される。
【００２４】
一実施形態において、本明細書で説明される命令は、ベクトル・ファシリティの一部であるベクトル命令である。ベクトル・ファシリティは、例えば、１つの要素から１６の要素までの範囲の固定サイズのベクトルを提供する。各ベクトルは、ファシリティ内で定められたベクトル命令により操作されるデータを含む。一実施形態において、ベクトルが複数の要素で構成される場合、各要素は、他の要素と共に並行処理される。全ての要素の処理が完了するまで、命令は完了しない。他の実施形態において、要素は部分的に並行処理され、及び／又は逐次的に処理される。
【００２５】
ベクトル命令は、ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ、Ｐｏｗｅｒ、ｘ８６、ＩＡ－３２、ＩＡ－６４等を含むがこれらに限定されるものではない、種々のアーキテクチャの一部として実装することができる。本明細書で説明される実施形態はｚ／Ａｒｃｈｉｔｅｃｔｕｒｅに関するものであるが、本明細書で説明されるベクトル命令及び１つ又は複数の他の態様は、他の多数のアーキテクチャに基づくものであってもよい。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅは一例にすぎない。
【００２６】
ベクトル・ファシリティがｚ／Ａｒｃｈｉｔｅｃｔｕｒｅの一部として実装される一実施形態において、ベクトル・レジスタ及び命令を使用するために、指定された制御レジスタ（例えば、制御レジスタ０）におけるベクトル・イネーブルメント制御（ｖｅｃｔｏｒ　ｅｎａｂｌｅｍｅｎｔ　ｃｏｎｔｒｏｌ）及びレジスタ制御が、例えば１に設定される。ベクトル・ファシリティがインストールされており、ベクトル命令がイネーブルメント制御の設定なしに実行される場合、データ例外が認識される。ベクトル・ファシリティがインストールされておらず、ベクトル命令が実行される場合、演算例外が認識される。
【００２７】
一実施形態においては、３２のベクトル・レジスタが存在し、他のタイプのレジスタは、ベクトル・レジスタの象限にマッピングすることができる。例えば、図４に示すように、レジスタ・ファイル３００は３２のベクトル・レジスタ３０２を含み、各レジスタの長さは１２８ビットである。長さが６４ビットである１６の浮動小数点レジスタ３０４は、ベクトル・レジスタに重ね合わせることができる。従って、一例として、浮動小数点レジスタ３０４が修正されると、ベクトル・レジスタ３０２も修正される。他のタイプのレジスタに対する他のマッピングも可能である。
【００２８】
ベクトル・データは、ストレージにおいて、例えば、他のデータ形式と同じ左から右への順序で現れる。０－７の番号が付けられたデータ形式のビットは、ストレージ内の左端の（最小番号を付された）バイト位置のバイトを構成し、ビット８－１５は、次の順次位置のバイトを形成し、以下同様である。さらに別の例において、ベクトル・データは、ストレージにおいて、右から左などの別の順序で現れることがある。
【００２９】
本明細書で説明されるベクトル命令の各々は複数のフィールドを有し、フィールドの１つ又は複数は、それに付随する下付き数字を有する。命令フィールドに付随する下付き数字は、そのフィールドが適用されるオペランドを示す。例えば、ベクトル・レジスタV_1に付随する下付き数字１は、V_1のレジスタが第１のオペランドを含むことを示し、以下同様である。レジスタ・オペランドは、長さが１レジスタであり、これは例えば１２８ビ

ットである。

【００３０】

さらに、ベクトル・ファシリティが与えられるベクトル命令の多くは、指定ビットのフィールドを有する。レジスタ拡張ビット（ｒｅｇｉｓｔｅｒ ｅｘｔｅｎｓｉｏｎ ｂｉｔ）又はＲＸＢと呼ばれるこのフィールドは、ベクトル・レジスタ指定のオペランドの各々についての最上位ビットを含む。命令によって指定されないレジスタ指示のためのビットは、予約され、ゼロに設定される。最上位ビットは、例えば、４ビットのレジスタ指示の左に連結されて、５ビットのベクトル・レジスタ指示を作成する。

【００３１】

一例において、ＲＸＢフィールドは４ビット（例えば、ビット０－３）を含み、これらのビットは、以下のように定義される。：

０－命令の第１のベクトル・レジスタ指定（例えば、ビット８－１１内）のための最上位ビット。

１－もしあれば、命令の第２のベクトル・レジスタ指定（例えば、ビット１２－１５内）のための最上位ビット。

２－もしあれば、命令の第３のベクトル・レジスタ指定（例えば、ビット１６－１９内）のための最上位ビット。

３－もしあれば、命令の第４のベクトル・レジスタ指定（例えば、ビット３２－３５内）のための最上位ビット。

【００３２】

各ビットは、例えば、レジスタ番号に応じて、アセンブラによりゼロ又は１に設定される。例えば、レジスタ０－１５に対してビットは０に設定され、レジスタ１６－３１に対してビットは１に設定される、などである。

【００３３】

一実施形態において、各ＲＸＢビットは、１つ又は複数のベクトル・レジスタを含む命令における特定の位置のための拡張ビットである。例えば、１つ又は複数のベクトル命令において、ＲＸＢのビット０は、位置８－１１のための拡張ビットであり、これが例えばV_1に割り当てられ、ＲＸＢのビット１は、位置１２－１５のための拡張ビットであり、これが例えばV_2に割り当てられ、以下同様である。さらに別の実施形態において、ＲＸＢフィールドは付加的なビットを含み、１つより多くのビットが、各ベクトル又は位置のための拡張として用いられる。

【００３４】

ＲＸＢフィールドを含む、一態様に従って提供される１つの命令が、Ｖｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ Ｔｅｓｔ Ｄａｔａ Ｃｌａｓｓ Ｉｍｍｅｄｉａｔｅ（ＶＦＴＣＩ）命令であり、その一例を図５に示す。一例において、Ｖｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ Ｔｅｓｔ Ｄａｔａ Ｃｌａｓｓ Ｉｍｍｅｄｉａｔｅ命令４００は、Ｖｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ Ｔｅｓｔ Ｄａｔａ Ｃｌａｓｓ Ｉｍｍｅｄｉａｔｅ操作を指示するオペコード・フィールド４０２ａ（例えば、ビット０－７）、４０２ｂ（例えば、ビット４０－４７）と、第１のベクトル・レジスタ（V_1）を指示するのに用いられる第１のベクトル・レジスタ・フィールド４０４（例えば、ビット８－１１）と、第２のベクトル・レジスタ（V_2）を指示するのに用いられる第２のベクトル・レジスタ・フィールド４０６（例えば、ビット１２－１５）と、ビットマスクを含むための即値フィールド（I_3）４０８（例えば、ビット１６－２７）と、第１のマスク・フィールド（M_5）４１０（例えば、ビット２８－３１）と、第２のマスク・フィールド（M_4）４１２（例えば、ビット３２－３５）と、ＲＸＢフィールド４１４（例えば、ビット３６－３９）とを含む。フィールド４０４－４１４の各々は、一例においては、オペコード・フィールドから分離し、独立している。さらに、一実施形態において、これらのフィールド４０４－４１４は互いに分離し、独立しているが、他の実施形態においては、１つより多くのフィールドを組み合わせてもよい。これらのフィールドの使用に関する更なる情報を以下に説明する。

【００３５】
一例において、オペコード・フィールド４０２ａにより指示されるオペコードの選択されたビット（例えば、最初の２ビット）は、命令の長さを指定する。この特定の例において、選択されたビットは、長さが３ハーフワード（ｈａｌｆｗｏｒｄ）であることを示す。さらに、命令の形式は、拡張されたオペコード・フィールドを伴うベクトル・レジスタ・アンド即値操作（ｖｅｃｔｏｒ　ｒｅｇｉｓｔｅｒ－ａｎｄ－ｉｍｍｅｄｉａｔｅ）操作である。ベクトル（Ｖ）フィールドの各々は、ＲＸＢによって指定されたその対応する拡張ビットと共に、ベクトル・レジスタを指示する。特に、ベクトル・レジスタについては、オペランドを含むレジスタが、例えば、その対応するレジスタ拡張ビット（ＲＸＢ）を最上位ビットとして付加したレジスタ・フィールドの４ビット・フィールドを用いて指定される。例えば、４ビット・フィールドが０１１０であり、拡張ビットが０である場合、５ビット・フィールド００１１０はレジスタ番号６を示す。
【００３６】
さらに、ＶＦＴＣＩ命令の一実施形態において、$V_1$４０４及び$V_2$４０６は、それぞれ、命令のための第１のオペランド及び第２のオペランドを含むベクトル・レジスタを指定する。付加的に、以下でさらに詳細に説明されるように、$I_3$４０８は複数のビットを有するビットマスクを含み、各ビットは２進浮動小数点要素のクラス及び符号（正又は負）を表すのに用いられる。
【００３７】
さらに別の実施形態において、ビットマスクは、例として、汎用レジスタ、メモリ、（要素ごとに異なる）ベクトル・レジスタの要素において、又はアドレス計算から、提供することができる。ビットマスクは、命令の明示的オペランドとして含ませてもよく、又は暗黙オペランド若しくは入力として含ませてもよい。
【００３８】
M_5フィールド４１０は、例えば、４ビット（０－３）を有し、例えばビット０における単一要素制御（Ｓ）を指定する。ビット０が１に設定される場合、操作は、ベクトル内のゼロ・インデックス付の要素に対してのみ行われる。第１のオペランド・ベクトル内の他の全ての要素のビット位置は、予測不能である。ビット０がゼロに設定される場合、操作は、ベクトル内の全ての要素に対して行われる。
【００３９】
M_4フィールド４１２は、例えば、命令の第２のオペランド内の浮動小数点数のサイズを指定するのに用いられる。一例において、このフィールドは、倍精度２進浮動小数点数を示す３に設定される。他の例も可能である。
【００４０】
Ｖｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令の一実施形態の実行において、第３のオペランドから１つ又は複数のビットを選択するために、第２のオペランドの浮動小数点要素のクラス及び符号を検査する。選択されたビットが設定された場合、第１のオペランド内の対応する要素の全てのビット位置が１に設定され、そうでなければ、これらはゼロに設定される。つまり、第２のオペランドの要素内に含まれる浮動小数点数のクラス／符号が、第３のオペランド内の設定されたビット（即ち、例えば１に設定されたビット）と一致した場合、第２のオペランドの要素に対応する第１のオペランドの要素は、１に設定される。一例において、全てのオペランド要素は、長形式（ｌｏｎｇ　ｆｏｒｍａｔ）のＢＦＰ（２進浮動小数点）数を含む。
【００４１】
本明細書で示されるように、第３のオペランドの１２ビット、すなわち命令テキストのビット１６－２７を用いて、ＢＦＰデータのクラス及び符号の１２の組み合わせを指定する。一例において、図６に示されるように、ＢＦＰオペランド要素は、６つのクラス４３０：即ち、ゼロ、正規数、非正規数、無限大、クワイエット（ｑｕｉｅｔ）ＮａＮ（ＮａＮ：Ｎｏｔ－ａ－Ｎｕｍｂｅｒ（非数））、及びシグナリング（ｓｉｇｎａｌｉｎｇ）Ｎ

ａＮに分けられ、各クラスは、これと関連付けられた符号４３２（正又は負のいずれか）を有する。従って、例えば、I_3のビット０は正符号を有するゼロ・クラスを指定し、ビット１は負符号を有するゼロ・クラスを指定する等である。

【００４２】

第３のオペランドの１つ又は複数のビットを１に設定することができる。さらに、一実施形態において、命令は、同時に１つ又は複数の要素を操作することができる。

ＩＥＥＥ例外を引き起こすことなく、ＳＮａＮ（シグナリングＮａＮ）及びＱＮａＮ（クワイエットＮａＮ）を含むオペランド要素が検査される。

【００４３】

全ての要素に対する結果のサマリ条件コード：

０　全ての要素について、選択されたビットは１である（一致）

１　全ての要素ではないが、少なくとも１つの要素について、選択されたビットは１である（Ｓビットがゼロである場合）

２　－－

３　全ての要素について、選択されたビットは０である（非一致）

【００４４】

ＩＥＥＥ例外：なし

プログラム例外：

＊ベクトル・ファシリティがイネーブルにされていないことを示す、データ例外コード（ＤＸＣ）ＦＥを伴うデータ、ベクトル命令

＊演算（ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅのためのベクトル・ファシリティがインストールされていない場合）

＊指定

＊トランザクション制限

【００４５】

プログラミング上の注意：

１．この命令は、例外のリスクなしに又はＩＥＥＥフラグを設定せずにオペランド要素をテストする方法を提供する。

２．Ｓビットが設定された場合、１の条件コードは使用されない。

【００４６】

Ｖｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ Ｔｅｓｔ Ｄａｔａ Ｃｌａｓｓ Ｉｍｍｅｄｉａｔｅ命令の一実施形態に関する更なる詳細が、図７及び図８を参照して説明される。特に、図７は、プロセッサ（例えば、ＣＰＵ）によって実施されるＶｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ Ｔｅｓｔ Ｄａｔａ Ｃｌａｓｓ Ｉｍｍｅｄｉａｔｅ命令と関連付けられた論理の一実施形態を示し、図８は、Ｖｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ Ｔｅｓｔ Ｄａｔａ Ｃｌａｓｓ Ｉｍｍｅｄｉａｔｅ命令の実行を図示するブロック図の一例を示す。

【００４７】

図７を参照すると、最初に、要素インデックス（Ｅｉ）と呼ばれる変数をゼロに初期化する（ステップ４５０）。次に、この場合は要素０である要素Ｅｉの値を、命令の第２のオペランドから（例えば、V_2によって指示されるレジスタに格納されたオペランドから）抽出する（ステップ４５２）。以下で説明されるように、長形式の２進浮動小数点の値であるこの値をタイプ番号に変換し、第２のオペランドの浮動小数点要素についてのクラス及び符号を取得する（ステップ４５４）。一例において、浮動小数点数のサイズ４５３を変換論理に入力する。図６を参照して説明されるように、取得したクラス及び符号を特定のクラス／符号ビットと関連付ける。例えば、浮動小数点数が正の正規数であることが変換により示される場合、ビット２をその浮動小数点数と関連付ける。

【００４８】

変換に続いて、変換に基づいて決定された特定のビットに対応する第３のオペランド内のビット（選択されたビットと呼ばれる）をチェックする（ステップ４５６）。選択され

たビットが設定された場合（問い合わせ４５８）、要素（Ｅｉ）に対応する第１のオペランド内の要素を全て１に等しくなるように設定し（ステップ４６０）、そうでなければ、第１のオペランド内の要素をゼロと等しくなるように設定する（ステップ４６２）。例えば、要素０内の浮動小数点数の変換が正の正規数を示す場合、ビット２をその数と関連付ける。従って、第３のオペランドのビット２をチェックし、これが１に設定された場合、第１のオペランドの要素０を全て１に設定する。

【００４９】

その後、Ｅｉが第２のオペランドの要素の最大数に等しいかどうかについての判断を行う（問い合わせ４６４）。等しくない場合、Ｅｉを例えば１だけインクリメントし（ステップ４６６）、処理はステップ４５２を続行する。そうではなく、Ｅｉが要素の最大数に等しい場合、サマリ条件コードを生成する（ステップ４６８）。サマリ条件コードは、第２のオペランドの全ての要素についての処理を要約する。例えば、全ての要素について、選択されたビットが１である場合（一致）、結果の条件コードはゼロである。一方、要素の全てではないが少なくとも１つについて、選択されたビットが１である場合（Ｓビットがゼロでない場合）、条件コードは１であり、要素の全てについて選択されたビットが０である場合（非一致）、条件コードは３である。

【００５０】

上記の処理は、図８のブロック図に示される。示されるように、ベクトル・レジスタ４８０は、各々が浮動小数点数を含む、複数の要素４８２ａ－４８２ｎを含む。浮動小数点数４８３ａ－４８３ｎの各々の浮動小数点数及びサイズは、タイプ番号への変換（ｃｏｎｖｅｒｔ－ｔｏ－ｔｙｐｅ　ｎｕｍｂｅｒ）論理４８４ａ－４８４ｎに入力され、その出力は、その浮動小数点数についてのクラス／符号を表す特定のビットである。次に、特定のビットの各々に対応する各マスク４８６ａ－４８６ｎ内の選択されたビットをチェックする。選択されたビットが設定されているかどうかに応じて、ベクトル・レジスタ４８０内の第１のオペランドが設定される。例えば、第２のオペランドの要素０に対して、選択されたビットが設定されている場合、第１のオペランドの要素４９０ａは全て１に設定される。同様に、第２のオペランドの要素１に対する選択されたビットが設定されていない（例えば、ゼロに設定されている）場合、第１のオペランドの要素４９０ｂは全てゼロに設定される。

【００５１】

ここで、タイプ番号への変換論理の一実施形態の更なる詳細を説明する。最初に、標準的なＩＥＥＥ２進浮動小数点数である浮動小数点数を、既知のように、３つの部分：即ち符号部、指数部（８ビット）＋１２７、及び仮数部（２３ビット）に変換する。次に、図９に示されるように、３つの部分全ての値をチェックして、クラス及び符号を判断する。例えば、符号は符号部の値であり、クラス（図９では、エンティティとしても知られる）は、指数部及び仮数部の値に基づいている（図９の単位ビットは、仮数部の暗黙ビットである）。一例として、指数部及び仮数部（単位ビットを含む）の値がゼロの場合、クラスはゼロであり、符号部が正の場合、符号は正である。従って、ビット０（図６）は、この浮動小数点数のクラス／符号を表す。

【００５２】

上述したのは、ベクトル内の要素の浮動小数点クラスをテストし、結果のビットマスクを設定するための命令の一実施形態である。Ｖｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令は、検出するための浮動小数点数のクラスを各ビットが表す即値フィールドを有する。入力ベクトルの各浮動小数点の要素をテストして、値が、命令によって指定されるクラスのいずれかの中にあるかどうかを確認する。浮動小数点要素がクラスの１つの中にある場合、出力ベクトルの対応する要素のビット位置は１に設定される。これは、いかなる例外又は割り込みも引き起こすことなく、２進浮動小数点数に関する何らかの特性（例えば、クラス及び符号）を判断するための技術を提供する。

【００５３】

さらに別の実施形態において、第３のオペランドのどのビットが（例えば１に）設定されているかをチェックし、次いで、第２のオペランドの１つ又は複数の要素のクラス／符号が設定されたビットの１つと同じであるかどうかを判断することにより、テストを行うことができる。次いで、比較に基づいて第１のオペランドを設定する。

【００５４】

さらに別の態様において、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令が提供され、その一例が図１０に示される。一例において、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令５００は、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ操作を示すオペコード・フィールド５０２ａ（例えば、ビット０－７）、５０２ｂ（例えば、ビット４０－４７）と、第１のベクトル・レジスタ（$Ｖ_1$）を指示するのに用いられる第１のベクトル・レジスタ・フィールド５０４（例えば、ビット８－１１）と、第２のベクトル・レジスタ（$Ｖ_2$）を指示するのに用いられる第２のベクトル・レジスタ・フィールド５０６（例えば、ビット１２－１５）と、第３のベクトル・レジスタ（$Ｖ_3$）を指示するのに用いられる第３のベクトル・レジスタ・フィールド５０８（例えば、ビット１６－１９）と、ＲＸＢフィールド５１０（例えば、ビット３６－３９）とを含む。フィールド５０４乃至５１０の各々は、一例において、オペコード・フィールドから分離され、独立している。さらに、一実施形態において、これらのフィールド５０４乃至５１０は互いに分離し、独立しているが、他の実施形態においては、１つより多くのフィールドを組み合わせてもよい。

【００５５】

さらに別の実施形態において、第３のベクトル・レジスタ・フィールドは、命令の明示的オペランドとしては含まれず、代わりに、暗黙オペランド又は入力である。さらに、オペランドにおいて提供される値を、例えば汎用レジスタで、メモリで、アドレス計算としてなど、他の手法で提供することもできる。

【００５６】

さらに別の実施形態において、第３のオペランドは、明示的であれ又は暗黙的であれ、全く提供されない。

【００５７】

一例において、オペコード・フィールド５０２ａにより指示されるオペコードの選択されたビット（例えば、最初の２ビット）は、命令の長さを指定する。この特定の例において、選択されたビットは、長さが３ハーフワードであることを示す。さらに、命令の形式は、拡張されたオペコード・フィールドを伴うベクトル・レジスタ・アンド・レジスタ（ｖｅｃｔｏｒ　ｒｅｇｉｓｔｅｒ－ａｎｄ－ｒｅｇｉｓｔｅｒ）操作である。ベクトル（Ｖ）フィールドの各々は、ＲＸＢによって指定されたその対応する拡張ビットと共に、ベクトル・レジスタを指示する。特に、ベクトル・レジスタについては、オペランドを含むレジスタは、例えば、その対応するレジスタ拡張ビット（ＲＸＢ）を最上位ビットとして付加したレジスタ・フィールドの４ビット・フィールドを用いて指定される。

【００５８】

Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令の一実施形態の実行において、例えばワード（ｗｏｒｄ）サイズである第２のオペランドからの要素が、第３のオペランドの選択された要素、例えば第３のオペランドのワード１内の要素と共に、１つずつ加算される。（別の実施形態においては、第３のオペランドの選択された要素の加算は任意である。）和は、第１のオペランドの選択された位置、例えばワード１に入れられる。第１のオペランドの他のワード要素、例えばワード要素０及び２－３に、ゼロが入れられる。ワード・サイズ要素は、全て３２ビットの符号なし２進整数として扱われる。要素のそれぞれの加算後、例えば和のビット位置０のキャリー出力を、例えば、第１のオペランドのワード要素１における結果のビット位置３１に加算する。

【００５９】

条件コード：コードは変更されないままである。

【００６０】

プログラム例外：

＊ベクトル・ファシリティがイネーブルにされていないことを示す、データ例外コード（ＤＸＣ）ＦＥを伴うデータ、ベクトル命令
＊演算（ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅのためのベクトル・ファシリティがインストールされていない場合）
＊トランザクション制限

【００６１】

プログラミング上の注意：
１．第３のオペランドのコンテンツは、チェックサム計算アルゴリズムの開始時にゼロを含むことになる。
２．１６ビット・チェックサムは、例えば、ＴＣＰ／ＩＰアプリケーションにおいて用いられる。３２ビット・チェックサムを計算した後、以下のプログラムを実行することができる。

【表１】

VERLLF V2,V1,16(0)　(VERLLF - Vector Element Rotate Left Logical – 4-byte value)

VAF V2,V1,V2　(VAF – Vector Add – 4 byte value)

要素２内のハーフワードは、１６ビット・チェックサムを含む。

【００６２】

Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令に関する更なる詳細が、図１１及び図１２を参照して説明される。一例において、図１１は、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令の実行においてプロセッサにより実施される論理の一実施形態を示し、図１２は、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令の実行の一例のブロック図を示す。

【００６３】

図１１を参照すると、最初に、第１のオペランド（ＯＰ１）の要素インデックス（Ｅｙ）を、例えば第１のオペランドの要素１を示す１に設定する（ステップ５３０）。同様に、第３のオペランド（ＯＰ３）の要素インデックス（Ｅｘ）を、例えば第３のオペランドの要素１を示す１に設定する（ステップ５３２）。次に、要素インデックス（Ｅｉ）を０と等しくなるように設定し、要素インデックス（Ｅｙ）における要素、即ちこの例では要素１をゼロに初期化する（ステップ５３４）。さらに別の実施形態において、Ｅｘ及びＥｙは、任意の有効な要素インデックスに設定することができる。

【００６４】

ＯＰ１（Ｅｙ）＝ＯＰ１（Ｅｙ）＋ＯＰ２（Ｅｉ）＋ＯＰ２（Ｅｉ＋１）である、エンド・アラウンド・キャリー（ＥＡＣ）加算を行う（ステップ５３６）。従って、出力ベクトル（ＯＰ１）の要素１を、その要素のコンテンツに第２のオペランド（ＯＰ２）の要素０内の値及び第２のオペランドの要素１内の値を加算したものと等しくなるように、設定する。エンド・アラウンド・キャリー加算を用いて、加算演算を行い、加算からのあらゆるキャリー出力を和に戻し、新しい和を生成する。

【００６５】

さらに別の実施形態においては、上述のように加算する代わりに、以下を実施する。すなわち、一時的アキュムレータ値を定め、ゼロに初期化し、次に、一度に１つの要素を加算する。さらに別の実施形態として、全てのワードを並列に加算し、一時的アキュムレータは存在しない。他の変形も可能である。

【００６６】

その後、第２のオペランド内に加算するべき付加的な要素があるかどうかについての判断を行う（問い合わせ５３８）。例えば、Ｅｉ－２＜第２のオペランドの要素の番号であるか。加算すべき第２のオペランドの要素がさらにある場合、Ｅｉを、例えば２だけイン

クリメントし（ステップ５４０）、処理はステップ５３６を続行する。

【００６７】

第２のオペランドにわたって要素を加算した後、結果を第３のオペランド内の値に加算する。例えば、第１のオペランドの要素（Ｅｙ）（これは、全ての第２のオペランドの要素にわたるＥＡＣ加算の和である）と第３のオペランド（ＯＰ３）の要素（Ｅｘ）内の値とのエンド・アラウンド・キャリー加算を行う（即ち、ＥＡＣ　ＡＤＤ　ＯＰ１（Ｅｙ）＋ＯＰ３（Ｅｘ））（ステップ５４２）。これは、図１２に示される。

【００６８】

図１２に示されるように、第２のオペランド５５０は複数の要素５５２ａ－５５２ｎを含み、これらの要素が、第３のオペランド５６０のワード１内の要素（５６２）と共に１つずつ加算される。結果は、第１のオペランド５７０の要素１（５７２）に入れられる。これは、式$Ey = Ex + E_i$の総和、により数学的に表され、ここで、ｉは０乃至ｎであり、加算はエンド・アラウンド・キャリー加算である。

【００６９】

上述したのは、レーン（ｌａｎｅ）算術計算を行う代わりに、ベクトル・レジスタの要素にわたってチェックサムを行うＶｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令の一実施形態である。一実施形態において、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令は、エンド・アラウンド・キャリー加算によりｓｕｍ－ａｃｒｏｓｓを実施することにより、チェックサムを行う。一例において、Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令は、ベクトル・レジスタから４つの４バイト整数要素を取り出し、これらを加算する。加算からのあらゆるキャリー（桁上げ、ｃａｒｒｙ）が戻されて加えられる。４バイトの和を別のオペランド内の４バイト要素に加算し、次に、さらに別のベクトル・レジスタに保存する（例えば、ベクトル・レジスタの下位４バイト要素が、ベクトル・レジスタの上位要素内に格納される）。

【００７０】

さらに別の実施形態において、値を保存するためにさらに別のベクトル・レジスタ又は別のレジスタが使用されず、代わりに、他のレジスタ（即ち、オペランド）の１つがアキュムレータとして使用される。

【００７１】

与えられるチェックサムを用いて、データの完全性を保つことができる。受信したデータが正しいことを検証するために、チェックサムが、データに適用され、ノイズのあるチャネル上で送られることが多い。この例では、本明細書で説明されるように、連続する４バイト整数を加算することによって、チェックサムが計算される。整数算術演算のキャリー出力がある場合、キャリー及び付加的な１が、累積和に加算される。

【００７２】

本明細書ではチェックサムが説明されるが、類似の技術を他のエンド・アラウンド・キャリー加算に対して用いることができる。

【００７３】

一態様に従って提供されるさらに別の命令は、Ｖｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ　Ａｃｃｕｍｕｌａｔｅ（ＶＧＦＭＡ）命令であり、その一例を図１３に示す。一例において、Ｖｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ　Ａｃｃｕｍｕｌａｔｅ命令６００は、Ｖｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ　Ａｃｃｕｍｕｌａｔｅ操作を示す、オペコード・フィールド６０２ａ（例えば、ビット０－７）、６０２ｂ（例えば、ビット４０－４７）と、第１のベクトル・レジスタ（V_1）を指示するのに用いられる第１のベクトル・レジスタ・フィールド６０４（例えば、ビット８－１１）と、第２のベクトル・レジスタ（V_2）を指示するのに用いられる第２のベクトル・レジスタ・フィールド６０６（例えば、ビット１２－１５）と、第３のベクトル・レジスタ（V_3）を指示するのに用いられる第３のベクトル・レジスタ・フィールド６０８（例えば、ビット１６－１９）と、マスク・フィールド（M_5）６１０（例えば、ビット２０－２３）と、第４のベクトル・レジスタ（V_4）を指示するのに用いられる第４のベクトル・レジ

スタ・フィールド６１２（例えば、ビット３２－３５）と、ＲＸＢフィールド６１４（例えば、ビット３６－３９）とを含む。フィールド６０４－６１４の各々は、一例において、オペコード・フィールドから分離し、独立している。さらに、一実施形態においては、これらのフィールド６０４－６１４は互いに分離し、独立しているが、他の実施形態においては、１つより多くのフィールドを組み合わせてもよい。

【００７４】

一例において、オペコード・フィールド６０２ａにより指示されるオペコードの選択されたビット（例えば、最初の２ビット）は、命令の長さを指定する。この特定の例において、選択されたビットは、長さが３ハーフワードであることを示す。さらに、命令の形式は、拡張されたオペコード・フィールドを伴うベクトル・レジスタ・アンド・レジスタ操作である。ベクトル（Ｖ）フィールドの各々は、ＲＸＢによって指定されたその対応する拡張ビットと共に、ベクトル・レジスタを指示する。特に、ベクトル・レジスタについては、例えば、その対応するレジスタ拡張ビット（ＲＸＢ）を最上位ビットとして付加したレジスタ・フィールドの４ビット・フィールドを用いて、オペランドを含むレジスタが指定される。

【００７５】

M_5フィールド６１０は、例えば４ビット（０－３）を含み、要素サイズ（ＥＳ）制御を指定する。要素サイズ制御は、ベクトル・レジスタ・オペランド２及び３内の要素のサイズを指定し、第１のオペランド及び第４のオペランド内の要素は、ＥＳ制御によって指定されるものの２倍のサイズである。例えば、M_5における０の値はバイト・サイズの要素を示し、例として、１はハーフワードを示し、２はワードを示し、３はダブルワード（ｄｏｕｂｌｅｗｏｒｄ）を示す。

【００７６】

Ｖｅｃｔｏｒ Ｇａｌｏｉｓ Ｆｉｅｌｄ Ｍｕｌｔｉｐｌｙ Ｓｕｍ ａｎｄ Ａｃｃｕｍｕｌａｔｅ命令の一実施形態の実行においては、ガロア体（Ｇａｌｏｉｓ ｆｉｅｌｄ）（即ち、有限数の要素を有する有限フィールド）において、第２のオペランドの各要素に、第３のオペランドの対応する要素を乗算する。つまり、キャリーレス（桁上げなし、ｃａｒｒｙｌｅｓｓ）乗算を用いて、第２のオペランドの各要素に、対応する第３のオペランドの要素を乗算する。一例において、ガロア体は、２の位数を有する。この乗算は、標準的な２進乗算に類似しているが、シフトされる被乗数を加算する代わりに、排他的論理和演算（ＸＯＲ）される。例えば、ダブル要素サイズの積の結果として得られる偶数－奇数の対を互いに排他的論理和演算し、第４のオペランドの、例えばダブルワイド要素などの対応する要素と排他的論理和演算する。結果は、例えば、第１のオペランドのダブルワイド要素に入れられる。

【００７７】

条件コード：コードは変更されないままである。

【００７８】

プログラム例外：

＊ベクトル・ファシリティがイネーブルにされていないことを示す、データ例外コード（ＤＸＣ）ＦＥを伴うデータ、ベクトル命令

＊演算（ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅのためのベクトル・ファシリティがインストールされていない場合）

＊指定

＊トランザクション制限

【００７９】

さらに別の実施形態において、命令は、１つ又は複数のより少ない数のオペランドを含むことができる。例えば、第４のオペランドの代わりに、排他的論理和演算されるべき値は第１のオペランド内にあり、この値は結果も含む。他の変形も可能である。

【００８０】

Ｖｅｃｔｏｒ Ｇａｌｏｉｓ Ｆｉｅｌｄ Ｍｕｌｔｉｐｌｙ Ｓｕｍ ａｎｄ Ａｃ

ｃｕｍｕｌａｔｅ命令の実行の一実施形態に関する更なる詳細が、図１４及び図１５を参照して説明される。一例において、図１４は、Ｖｅｃｔｏｒ Ｇａｌｏｉｓ Ｆｉｅｌｄ Ｍｕｌｔｉｐｌｙ Ｓｕｍ ａｎｄ Ａｃｃｕｍｕｌａｔｅ命令を実行するためにプロセッサによって実施される論理の一実施形態を示し、図１５は、その論理の実行を示すブロック図の一例を示す。

【００８１】

図１４を参照すると、最初に、第２のオペランド（ＯＰ２）、第３のオペランド（ＯＰ３）、及び第４のオペランド（ＯＰ４）から、偶数／奇数の対を抽出し（ステップ６３０）、ｃａｒｒｙｌｅｓｓ ｍｕｌｔｉｐｌｙ ｓｕｍ ａｃｃｕｍｕｌａｔｅ関数を実施する（ステップ６３２）。例えば、２の累乗のガロア体において動作する場合、キャリーレス乗算はシフト・アンドＸＯＲ（排他的ＯＲ）であり、これはあらゆるキャリーを事実上無視する。結果は、第１のオペランド（ＯＰ１）に入れられ（ステップ６３４）、抽出されるべき対がさらにあるかどうかについての判断が行われる（問い合わせ６３６）。対がさらにある場合、処理はステップ６３０を続行し、他の場合には、処理は完了する（ステップ６３８）。一例において、要素サイズ６３１は、ステップ６３０－６３４への入力である。

【００８２】

ステップ６３２のｃａｒｒｙｌｅｓｓ ｍｕｌｔｉｐｌｙ ｓｕｍ ａｃｃｕｍｕｌａｔｅ関数の更なる詳細が、図１５を参照して説明される。示されるように、オペランドの対ＯＰ２Ｈ ６５２ａ、ＯＰ２Ｌ ６５２ｂが、第２のオペランド６５０から抽出される。さらに、オペランドの対ＯＰ３Ｈ ６６２ａ、ＯＰ３Ｌ ６６２ｂが、第３のオペランド６６０から抽出され、オペランドの対ＯＰ４Ｈ ６７２ａ及びＯＰ４Ｌ ６７２ｂが、第４のオペランド６７０から抽出される。キャリーレス乗算によって、オペランドＯＰ２Ｈ ６５２ａに、オペランドＯＰ３Ｈ ６６２ａを乗算して、結果Ｈ ６８０ａが与えられる。同様に、キャリーレス乗算によって、オペランドＯＰ２Ｌ ６５２ｂに、オペランドＯＰ３Ｌ ６６２ｂを乗算して、結果Ｌ ６８０ｂが与えられる。次に、結果Ｈ ６８０ａを結果Ｌ ６８０ｂと排他的論理和演算し、その結果をオペランドＯＰＨ ６７２ａ及びオペランドＯＰ４Ｌ ６７２ｂと排他的論理和演算し、結果を、ＯＰ１Ｈ ６９０ａ、ＯＰ１Ｌ ６９０ｂに入れる。

【００８３】

本明細書で説明されたのは、キャリーレス乗算演算を行い、次に最終的な排他的論理和演算を行って累積和を生成するベクトル命令である。この技術は、２の位数を有する有限体で演算を行うエラー検出コード及び暗号化の種々の態様と共に使用することができる。

【００８４】

一例において、命令は、ベクトル・レジスタの複数の要素に対して、キャリーレス乗算演算を行って和を得る。さらに、命令は、和に対して最終的な排他的論理和演算を行って累積和を生成する。命令は、実行されると、ガロア体において第２のベクトル及び第３のベクトルの対応する要素を乗算し、シフトされた被乗数を排他的論理和演算する。各々のダブルワイドの積を互いに排他的論理和演算し、結果を、第１のベクトルの対応するダブルワイド要素と排他的論理和演算する。結果は、第１のベクトル・レジスタに格納される。ダブルワード要素が上述されているが、他の要素サイズのワード・サイズ要素を用いてもよい。命令は、多数の異なる要素サイズで動作することができる。

【００８５】

一態様に従って提供されるさらに別の命令は、Ｖｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ（ＶＧＭ）命令であり、その一例が図１６を参照して説明される。一例において、Ｖｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ命令７００は、Ｖｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ操作を示すオペコード・フィールド７０２ａ（例えば、ビット０－７）、７０２ｂ（例えば、ビット４０－４７）と、第１のベクトル・レジスタ（$Ｖ_1$）を指示するのに用いられる第１のベクトル・レジスタ・フィールド７０４（例えば、ビット８－１１）と、第１の値を指定するのに用いられる第１の即値フィールド$Ｉ_2$ ７０６（例えば、ビ

ット１６－２４）と、第２の値を指定するのに用いられる第２の即値フィールド（I_3）７０８（例えば、ビット２４－３２）と、マスク・フィールド（M_4）７１０（例えば、ビット３２－３５）と、ＲＸＢフィールド７１２（例えば、ビット３６－３９）とを含む。フィールド７０４－７１２の各々は、一例においては、オペコード・フィールドから分離し、独立している。さらに、一実施形態において、これらのフィールド７０４－７１２は互いに分離し、独立しているが、他の実施形態においては、１つより多くのフィールドを組み合わせてもよい。

【００８６】

さらに別の実施形態において、第１の値及び／又は第２の値は、例として、汎用レジスタ、メモリ、（要素ごとに異なる）ベクトル・レジスタの要素において、又は、アドレス計算から、提供することができる。値は、命令の明示的オペランドとして含ませてもよく、又は暗黙オペランド若しくは入力として含ませてもよい。

【００８７】

一例において、オペコード・フィールド７０２ａにより指示されるオペコードの選択されたビット（例えば、最初の２ビット）は、命令の長さを指定する。この特定の例において、選択されたビットは、長さが３ハーフワードであることを示す。さらに、命令の形式は、拡張されたオペコード・フィールドを伴うベクトル・レジスタ・アンド即値操作である。ベクトル（Ｖ）フィールドの各々は、ＲＸＢによって指定されたその対応する拡張ビットと共に、ベクトル・レジスタを指示する。特に、ベクトル・レジスタについては、オペランドを含むレジスタが、例えば、その対応するレジスタ拡張ビット（ＲＸＢ）を最上位ビットとして付加したレジスタ・フィールドの４ビット・フィールドを用いて指定される。

【００８８】

M_4フィールドは、例えば、要素サイズ制御（ＥＳ）を指定する。要素サイズ制御は、ベクトル・レジスタ・オペランド内の要素のサイズを指定する。一例において、M_4フィールドのビット０は１バイトを指定し、ビット１はハーフワード（例えば２バイト）を指定し、ビット２はワード（例えば４バイト、別名フルワード）を指定し、ビット３はダブルワードを指定する。

【００８９】

Ｖｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ命令の一実施形態の実行において、第１のオペランド内の各要素について、ビットマスクが生成される。マスクは、例えばI_2内の符号なし整数値が指定するビット位置から開始して、例えばI_3内の符号なし整数値が指定するビット位置で終了する、１に設定されたビットを含む。他の全てのビット位置はゼロに設定される。一例において、指定された要素サイズのビット位置の全てを表すのに必要なビット数のみが、I_2フィールド及びI_3フィールドから使用され、他のビットは無視される。I_2フィールド内のビット位置がI_3フィールド内のビット位置より大きい場合、ビット範囲は、指定された要素サイズの最大ビット位置でラップする。例えば、バイト・サイズの要素を仮定すると、I_2＝１及びI_3＝６であれば、結果のマスクは、$X^{\xi}7E^{\xi}$、又は$B^{\xi}01111110^{\xi}$である。しかしながら、I_2＝６及びI_3＝１であれば、結果のマスクは、$X^{\xi}81^{\xi}$、又は$b^{\xi}10000001^{\xi}$である。

【００９０】

条件コード：コードは変更されないままである。

【００９１】

プログラム例外：

＊ベクトル・ファシリティがイネーブルにされていないことを示す、データ例外コード（ＤＸＣ）ＦＥを伴うデータ、ベクトル命令

＊演算（ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅのためのベクトル・ファシリティがインストールされていない場合）

＊指定

＊トランザクション制限

【００９２】
Ｖｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ命令の一実施形態に関する更なる詳細が、図１７及び図１８を参照して説明される。特に、図１７は、プロセッサによって実施されるＶｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ命令と関連付けられた論理の一実施形態を示し、図１８は、Ｖｅｃｔｏｒ Ｇｅｎｅｒａｔｅ Ｍａｓｋ命令の実行の一実施形態を図示するブロック図の一例を示す。
【００９３】
図１７を参照すると、最初に、第１のオペランド内の各要素について、マスクを生成する（ステップ７２０）。このステップは、第２のオペランド・フィールド内で開始位置（７２２）として指定された値と、第３のオペランド・フィールド内で終了位置（７２４）として指定された値と、M_4フィールド内で指定された要素サイズ（７２６）とを含む種々の入力を用いる。これらの入力は、マスクを生成し、第１のオペランド（Ｏｐ１）の、例えば要素０などの選択された要素の位置を埋める（ステップ７３０）のに用いられる。例えば、第１のオペランド（Ｏｐ１）の要素０は、複数の位置（例えば、ビット位置）を含み、I_2内の符号なし整数値が指定する位置で開始し、I_3内の符号なし整数値が指定する位置で終了し、第１のオペランドの要素０の位置（例えば、ビット）は、１に設定される。他のビット位置は０に設定される。その後、第１のオペランド内に要素がさらにあるかどうかについての判断を行う（問い合わせ７３４）。要素がさらにある場合、処理はステップ７２０を続行する。他の場合には、処理は完了する（ステップ７３６）。
【００９４】
マスクを生成すること及び第１のオペランドを埋めることが、図１８に示される。示されるように、入力（例えば、７２２－７２６）を用いて、第１のオペランドの各要素についてのマスクが生成され（７２０）、マスクを生成した結果は、第１のオペランドの要素７４０内に格納される。
【００９５】
上記で詳細に説明されるのは、ベクトルの各要素についてビットマスクを生成するための命令である。一実施形態において、命令は、開始ビット位置及び終了ビット位置を利用してビットマスクを生成し、このビットマスクは各要素について複製される。命令は、ビット範囲を指定し、範囲内の各ビットは、ベクトル・レジスタの各要素について１に設定されるが、他のビットはゼロに設定される。
【００９６】
一実施形態において、ビットマスクを生成するための命令を用いることにより、例えば、命令ストリームのキャッシュ・フットプリントを増大させ、必要なマスク数に応じて重要ループにおける待ち時間を増大させる場合があるメモリからビットマスクをロードすることに勝る利点が提供される。
【００９７】
１つの態様に従って提供されるさらに別の命令は、Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ（ＶＥＲＩＭ）命令であり、その一例が図１９に示される。一例において、Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ命令８００は、Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ操作を示すオペコード・フィールド８０２ａ（例えば、ビット０－７）、８０２ｂ（例えば、ビット４０－４７）と、第１のベクトル・レジスタ（V_1）を指示するのに用いられる第１のベクトル・レジスタ・フィールド８０４（例えば、ビット８－１１）と、第２のベクトル・レジスタ（V_2）を指示するのに用いられる第２のベクトル・レジスタ・フィールド８０６（例えば、ビット１２－１５）と、第３のベクトル・レジスタ（V_3）を指示するのに用いられる第３のベクトル・レジスタ・フィールド８０８（例えば、ビット１６－１９）と、例えば各要素をローテートするためのビット数を指定する符号なし２進整数を含む即値フィールド（I_4）８１２（例えば、ビット２４－３１）と、マスク・フィールド（M_5）８１４（例えば、ビット３２－３５）と、ＲＸＢフィールド８１６（例えば、ビット３

６－３９）とを含む。フィールド８０４－８１６の各々は、一例において、オペコード・フィールドから分離し、独立している。さらに、一実施形態において、これらのフィールド８０４－８１６は互いに分離し、独立しているが、他の実施形態においては、１つより多くのフィールドを組み合わせてもよい。

【００９８】

一例において、オペコード・フィールド８０２ａにより指示されるオペコードの選択されたビット（例えば、最初の２ビット）は、命令の長さを指定する。この特定の例において、選択されたビットは、長さが３ハーフワードであることを示す。さらに、命令の形式は、拡張されたオペコード・フィールドを伴うベクトル・レジスタ・アンド即値操作である。ベクトル（Ｖ）フィールドの各々は、ＲＸＢによって指定されたその対応する拡張ビットと共に、ベクトル・レジスタを指示する。特に、ベクトル・レジスタについては、オペランドを含むレジスタが、例えば、対応するレジスタ拡張ビット（ＲＸＢ）を最上位ビットとして付加したレジスタ・フィールドの４ビット・フィールドを用いて指定される。

【００９９】

M_5フィールドは、要素サイズ制御（ＥＳ）を指定する。要素サイズ制御は、ベクトル・レジスタ・オペランド内の要素のサイズを指定する。一例において、M_5フィールドのビット０はバイトを指定し、ビット１はハーフワード（例えば、２バイト）を指定し、ビット２はワード（例えば、４バイト、別名フルワード）を指定し、ビット３はダブルワードを指定する。

【０１００】

Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ命令の一実施形態の実行において、第２のオペランドの各要素は、第４のオペランドによって指定されたビット数だけ左にローテートされる。要素の左端のビット位置から外にシフトされた各ビットは、要素の右端のビット位置に再び入る。第３のオペランドは、各要素内にマスクを含む。第３のオペランド内の１である各ビットについて、第２のオペランド内のローテートされた要素の対応するビットが、第１のオペランド内の対応するビットに置き換わる。つまり、ローテートされた要素の対応するビットの値が、第１のオペランド内の対応するビットの値に置き換わる。第３のオペランド内の０である各ビットについて、第１のオペランドの対応するビットは変更されないままである。第１のオペランドが第２のオペランド又は第３のオペランドのいずれかと同一である場合を除いて、第２のオペランド及び第３のオペランドは変更されないままである。

【０１０１】

第４のオペランドは、例えば、第２のオペランド内の各要素をローテートするビット数を指定する、符号なし２進整数である。この値が指定された要素サイズのビット数より大きい場合、要素内のビット数を法として、値が減らされる。

【０１０２】

一例において、第３のオペランドに含まれるマスクは、本明細書で説明されるＶＧＭ命令を用いて生成される。

【０１０３】

条件コード：コードは変更されないままである。

【０１０４】

プログラム例外：

＊ベクトル・ファシリティがイネーブルにされていないことを示す、データ例外コード（ＤＸＣ）ＦＥを伴うデータ、ベクトル命令

＊演算（ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅのためのベクトル・ファシリティがインストールされていない場合）

＊指定

＊トランザクション制限

【０１０５】

プログラミング上の注意：

１．ＶＥＲＩＭとＶＧＭとの組み合わせを用いて、Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｓｅｌｅｃｔｅｄ Ｂｉｔｓ命令の完全な機能を達成することができる。

２．I_4フィールドのビットは、各要素を左にローテートするためのビット数を指定する符号なし２進整数を含むように定義されるが、右へのローテート量を効果的に指定する負の数をコード化してもよい。

【０１０６】

Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ命令の実行に関する更なる詳細が、図２０及び図２１を参照して説明される。特に、図２０は、プロセッサによって実施されるＶｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ命令と関連付けられた論理の一実施形態を示し、図２１は、Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ命令の実行の一例を図形的に示す。

【０１０７】

図２０を参照すると、第２のオペランドの選択された要素が、第４のオペランドで指定された量（８２０）だけローテートされる（ステップ８３０）。第４のオペランドで指定された値が要素サイズ（８２２）で指定されたビット数より大きい場合、その値は要素内のビット数を法として減らされる。

【０１０８】

要素のビットをローテートさせた後、ｍｅｒｇｅ ｕｎｄｅｒ ｍａｓｋを実施する（ステップ８３２）。例えば、第３のオペランド内の１である各ビット（８２４）について、第２のオペランド内のローテートされた要素の対応するビットが、第１のオペランド内の対応するビットに置き換わる。

【０１０９】

その後、ローテートされるべき要素がさらにあるかどうかについての判断が行われる（問い合わせ８３４）。ローテートされるべき要素がさらにある場合、処理はステップ８３０を続行する。そうでない場合には、処理は完了する（ステップ８３６）。

【０１１０】

図２１を参照すると、示されるように、第２のオペランドの要素は、入力８２０及び８２２に基づいてローテートされる（８３０）。さらに、入力８２４を用いて、ｍｅｒｇｅ ｕｎｄｅｒ ｍａｓｋ（８３２）を実施する。出力は、第１のオペランド８５０に与えられる。

【０１１１】

上述したのは、Ｖｅｃｔｏｒ Ｅｌｅｍｅｎｔ Ｒｏｔａｔｅ ａｎｄ Ｉｎｓｅｒｔ Ｕｎｄｅｒ Ｍａｓｋ命令の一例である。この命令は、定められたビット数だけ、選択されたオペランド内の要素をローテートするのに用いられる。ビットが指定されるが、さらに別の実施形態においては、位置の数だけ要素をローテートさせることができ、位置は、ビット以外であってもよい。さらに、命令は、異なる要素サイズと共に用いることができる。

【０１１２】

一例として、こうした命令は、テーブル・ルックアップの数から特定のビット範囲を選択するために用いられる。

【０１１３】

特定のベクトル命令又は他のＳＩＭＤ操作の実行の際、例外が発生することがある。ＳＩＭＤ操作において例外が発生した場合、通常、ベクトル・レジスタのどの要素が例外を引き起こしたのかは未知である。どの要素が例外を引き起こしたのかを判断するために、ソフトウェア割り込みハンドラは、各要素を抽出し、スカラー・モードで計算をやり直さなければならない。しかしながら、一態様によれば、マシン（例えば、プロセッサ）がベクトル演算に起因するプログラム割り込みを処理するとき、例えば、例外を引き起こしたベクトル内の最小インデックス付き要素を示す要素インデックスが報告される。そして、ソフトウェア割り込みハンドラは、即座に当該要素にスキップし、いずれかの必要な又は

所望のアクションを実施することができる。

【０１１４】

例えば、一実施形態において、ベクトル・データ例外がプログラム割り込みを発生させた場合、ベクトル例外コード（ＶＸＣ）が、例えば実メモリ位置（例えば、位置１４７（Ｘ^{ξ}９３^{ξ}））に格納され、ゼロが、実メモリ位置１４４－１４６（Ｘ^{ξ}９０^{ξ}－Ｘ^{ξ}９２^{ξ}））に格納される。さらに別の実施形態において、指定された制御レジスタ（例えば、ＣＲ０）の指定されたビット（例えば、ビット４５）が１である場合、ＶＸＣは、浮動小数点制御レジスタのデータ例外コード（ＤＸＣ）フィールドにも入れられる。制御レジスタ０のビット４５が０であり、制御レジスタ０のビット４６が１である場合、ＦＰＣレジスタのＤＸＣ、及び、位置１４７（Ｘ^{ξ}９３^{ξ}）におけるストレージのコンテンツは予測不能である。

【０１１５】

一実施形態において、ＶＸＣは、種々のタイプのベクトル浮動小数点例外を区別して、どの要素が例外を引き起こしたのかを示す。一例において、図２２に示されるように、ベクトル例外コード９００が、ベクトル・インデックス（ＶＩＸ）９０２と、ベクトル割り込みコード（ＶＩＣ）９０４とを含む。一例において、ベクトル・インデックスは、ベクトル例外コードのビット０－３を含み、その値は、例外を認識した選択されたベクトル・レジスタの左端の要素のインデックスである。さらに、ベクトル割り込みコードは、ベクトル例外コードのビット４－７に含まれ、例として以下の値を有する。

０００１　ＩＥＥＥ無効操作
００１０　ＩＥＥＥゼロ除算
００１１　ＩＥＥＥオーバーフロー
０１００　ＩＥＥＥアンダーフロー
０１０１　ＩＥＥＥ不正確

【０１１６】

さらに別の実施形態において、ＶＸＣは、例外を引き起こした要素のベクトル・インデックス又は他の位置インジケータのみを含む。

【０１１７】

一実施形態において、ＶＸＣは、例えば以下の命令を含む多数の命令：即ち、例として、Ｖｅｃｔｏｒ Ｆｌｏａｔｉｎｇ Ｐｏｉｎｔ（ＦＰ） Ａｄｄ、Ｖｅｃｔｏｒ ＦＰ Ｃａｍｐａｒｅ Ｓｃａｌａｒ、Ｖｅｃｔｏｒ ＦＰ Ｃｏｍｐａｒｅ Ｅｑｕａｌ、Ｖｅｃｔｏｒ ＦＰ Ｃｏｍｐａｒｅ Ｈｉｇｈ ｏｒ Ｅｑｕａｌ、Ｖｅｃｔｏｒ ＦＰ Ｃｏｎｖｅｒｔ Ｆｒｏｍ Ｆｉｘｅｄ ６４－Ｂｉｔ、Ｖｅｃｔｏｒ ＦＰ Ｃｏｎｖｅｒｔ Ｆｒｏｍ Ｌｏｇｉｃａｌ ６４－Ｂｉｔ、Ｖｅｃｔｏｒ ＦＰ Ｃｏｎｖｅｒｔ ｔｏ Ｆｉｘｅｄ ６４－Ｂｉｔ、Ｖｅｃｔｏｒ ＦＰ Ｃｏｎｖｅｒｔ ｔｏ Ｌｏｇｉｃａｌ ６４－Ｂｉｔ、Ｖｅｃｔｏｒ ＦＰ Ｄｉｖｉｄｅ、Ｖｅｃｔｏｒ Ｌｏａｄ ＦＰ Ｉｎｔｅｇｅｒ、Ｖｅｃｔｏｒ ＦＰ Ｌｏａｄ Ｌｅｎｇｔｈｅｎｅｄ、Ｖｅｃｔｏｒ ＦＰ Ｌｏａｄ Ｒｏｕｎｄｅｄ、Ｖｅｃｔｏｒ ＦＰ Ｍｕｌｔｉｐｌｙ、Ｖｅｃｔｏｒ ＦＰ Ｍｕｌｔｉｐｌｙ ａｎｄ Ａｄｄ、Ｖｅｃｔｏｒ ＦＰ Ｍｕｌｔｉｐｌｅ ａｎｄ Ｓｕｂｔｒａｃｔ、Ｖｅｃｔｏｒ ＦＰ Ｓｑｕａｒｅ Ｒｏｏｔ、及びＶｅｃｔｏｒ ＦＰ Ｓｕｂｔｒａｃｔ、並びに他のタイプのベクトル浮動小数点命令及び／又は他の命令によって設定することができる。

【０１１８】

ベクトル例外コードの設定に関する更なる詳細が、図２３を参照して説明される。一実施形態において、コンピューティング環境のプロセッサがこの論理を実施する。

【０１１９】

図２３を参照すると、最初に、上に列挙した命令の１つ又は別の命令などの、ベクトル・レジスタ上で動作する命令が実行される（ステップ９２０）。命令の実行の際、例外条件が引き起こされる（ステップ９２２）。一例において、例外条件は割り込みを発生させる。ベクトル・レジスタのどの要素が例外を引き起こしたかについての判断が行われる（

ステップ９２４）。例えば、ベクトル・レジスタの１つ又は複数の要素の計算を実施しているプロセッサの１つ又は複数のハードウェア・ユニットが例外を判断し、信号を出す。例えば、複数のハードウェア・ユニットがベクトル・レジスタの複数の要素の計算を並行して実施し、要素の１つ又は複数の処理中に例外が引き起こされた場合、例外を引き起こした処理を実施しているハードウェア・ユニットは、例外条件、並びに処理していた要素の表示を信号で送る。さらに別の実施形態においては、ベクトルの要素が逐次的に処理され、要素の処理中に例外に遭遇した場合、ハードウェアは、例外が発生したときにシーケンス内のどの要素にとりかかっていたかを示す。

【０１２０】

例外の信号送信に基づいて、ベクトル例外コードが設定される（ステップ９２６）。これは、例えば、例外を引き起こしたベクトル・レジスタ内の要素の位置、並びに割り込みコードを示すことを含む。

【０１２１】

上記で詳細に説明したのは、効率的なベクトル例外処理を提供するベクトル例外コードである。一例において、マシンがベクトル演算に起因するプログラム割り込みを処理する際、例外を引き起こしたベクトル・レジスタ内の最小インデックス付き要素を示す要素インデックスが報告される。特定の例として、ベクトル加算（ｖｅｃｔｏｒ　ａｄｄ）が実施されており、ベクトル・レジスタ毎に、Ａ０＋Ｂ０及びＡ１＋Ｂ１を与える２つの要素があり、且つ、Ａ１＋Ｂ１ではなくＡ０＋Ｂ０に関する不正確の結果を受信した場合、ＶＩＸは０に設定され、ＶＩＣは０１０１に等しくなるように設定される。さらに別の例においては、Ａ０＋Ｂ０は例外を受信しないが、Ａ１＋Ｂ１は例外を受信した場合、ＶＩＸは１に等しくなるように設定される（ＶＩＣ＝０１０１）。両方とも例外を引き起した場合、それが左端インデックス付き位置であるので、ＶＩＸは０に設定され、ＶＩＣ＝０１０１である。

【０１２２】

上記で詳細に説明したのは、種々のベクトル命令、並びに、ベクトル・レジスタ内の例外の位置を示すベクトル例外コードである。与えられたフロー図では、幾つかの処理は逐次的に見えることがあるが、１つ又は複数の実施形態においては、要素は並行処理され、従って、例えば、処理されるべき要素がさらにあるかどうかをチェックする必要がないことがある。他の多くの変形も可能である。

【０１２３】

付加的に、さらに別の実施形態において、命令の１つ又は複数のフィールドのコンテンツを、例として、汎用レジスタ、メモリ、（要素ごとに異なる）ベクトル・レジスタの要素内に、又は、アドレス計算から提供することができる。コンテンツは、命令の明示的オペランドとして、又は、暗黙オペランド若しくは入力として含ませることができる。さらに、１つ又は複数の命令がより少ない数のオペランド又は入力を用いることがあり、代わりに、１つ又は複数のオペランドを複数の操作又はステップに用いることもある。

【０１２４】

さらに、本明細書で説明されたように、命令のフィールドに要素サイズ制御を含む代わりに、要素サイズ制御を他の手法で提供することができる。付加的に、要素サイズはオペコードによって指示することができる。例えば、命令の特定のオペコードは、操作、並びに要素サイズ等を指示する。

【０１２５】

本明細書において、メモリ、主メモリ、ストレージ、及び主ストレージは、明示的に又は文脈によって特に断りのない限り、交換可能に使用される。

【０１２６】

当業者であれば理解するように、態様は、システム、方法又はコンピュータ・プログラムとして具体化することができる。従って、態様は、完全にハードウェアの実施形態、完全にソフトウェアの実施形態（ファームウェア、常駐ソフトウェア、マイクロコード等を含む）、又はソフトウェアの態様とハードウェアの態様とを組み合わせた実施形態の形態

をとることができ、本明細書においては、これらは全て、一般的に「回路」、「モジュール」又は「システム」と呼ぶことがある。さらに、態様は、媒体内に具体化されたコンピュータ可読プログラム・コードを有する、１つ又は複数のコンピュータ可読媒体内に具体化されたコンピュータ・プログラムの形態をとることができる。

【０１２７】

１つ又は複数のコンピュータ可読媒体のいずれかの組み合わせを用いることもできる。コンピュータ可読媒体は、コンピュータ可読ストレージ媒体とすることができる。コンピュータ可読ストレージ媒体は、例えば、これらに限定されるものではないが、電子、磁気、光学、電磁気、赤外線若しくは半導体のシステム、装置若しくはデバイス、又はこれらのいずれかの適切な組み合わせとすることができる。コンピュータ可読ストレージ媒体のより具体的な例（非網羅的なリスト）として、以下のもの、即ち、１つ又は複数の配線を有する電気的接続、ポータブル・コンピュータ・ディスケット、ハード・ディスク、ランダム・アクセス・メモリ（ＲＡＭ）、読み出し専用メモリ（ＲＯＭ）、消去可能なプログラム可能読み出し専用メモリ（ＥＰＲＯＭ又はフラッシュ・メモリ）、光ファイバ、ポータブル・コンパクト・ディスク型読み出し専用メモリ（ＣＤ－ＲＯＭ）、光記憶装置、磁気記憶装置、又は上記のいずれかの適切な組み合わせが挙げられる。本明細書の文脈においては、コンピュータ可読ストレージ媒体は、命令実行システム、装置若しくはデバイスによって又はそれらと関連して用いるためのプログラムを収容又は格納することが可能な、任意の有形媒体とすることができる。

【０１２８】

図２４を参照すると、一例において、コンピュータ・プログラム製品１０００は、例えば、本発明の１つ又は複数の態様を提供し、容易にするように、その上にコンピュータ可読プログラム・コード手段又は論理１００４を格納するための、１つ又は複数の一時的でないコンピュータ可読ストレージ媒体１００２を含む。

【０１２９】

コンピュータ可読媒体上に具体化されたプログラム・コードは、これらに限定されるものではないが、無線、有線、光ファイバ・ケーブル、ＲＦ等、又はこれらの任意の適切な組み合わせを含む、適切な媒体を用いて伝送することができる。

【０１３０】

態様のための動作を実行するためのコンピュータ・プログラム・コードは、Ｊａｖａ、Ｓｍａｌｌｔａｌｋ（登録商標）、Ｃ＋＋などのようなオブジェクト指向型プログラミング言語、及び、「Ｃ」プログラミング言語などのような従来の手続き型プログラミング言語、アセンブラ、又は類似のプログラミング言語を含む、１つ又は複数のプログラミング言語の任意の組み合わせで記述することができる。プログラム・コードは、全体をユーザのコンピュータ上で実行することができ、独立型ソフトウェア・パッケージとして部分的にユーザのコンピュータ上で実行することができ、一部をユーザのコンピュータ上で実行し、一部を遠隔コンピュータ上で実行することができ、又は全体を遠隔コンピュータ若しくはサーバ上で実行することができる。後者のシナリオにおいては、遠隔コンピュータは、ローカル・エリア・ネットワーク（ＬＡＮ）若しくは広域ネットワーク（ＷＡＮ）を含むいずれかのタイプのネットワークを通じてユーザのコンピュータに接続されるか、又は、若しくはコンバージド・ネットワークを含むがこれらに限定されるものではない、いずれかのタイプのネットワーク若しくは通信システムを通じてユーザのコンピュータに接続することができ、又は（例えば、インターネット・サービス・プロバイダを用いたインターネットを通じて）外部コンピュータへの接続をなすことができる。

【０１３１】

態様は、本明細書において、１つ又は複数の実施形態による方法、装置（システム）及びコンピュータ・プログラムのフローチャート図及び／又はブロック図を参照して説明される。フローチャート図及び／又はブロック図の各ブロック、並びにフローチャート図及び／又はブロック図におけるブロックの組み合わせは、コンピュータ・プログラム命令によって実装できることが理解されるであろう。これらのコンピュータ・プログラム命令を

、汎用コンピュータ、専用コンピュータ、又は他のプログラム可能データ処理装置のプロセッサに与えて、マシンを製造し、その結果、コンピュータ又は他のプログラム可能データ処理装置のプロセッサによって実行される命令が、フローチャート及び／又はブロック図の１つ又は複数のブロックにおいて指定された機能／動作を実装するための手段を生成するようにすることができる。

【０１３２】

これらのコンピュータ・プログラム命令を、コンピュータ、他のプログラム可能データ処理装置、又は他のデバイスを特定の方式で機能させるように指示することができるコンピュータ可読媒体内に格納し、その結果、そのコンピュータ可読媒体内に格納された命令が、フローチャート及び／又はブロック図の１つ又は複数のブロックにおいて指定された機能／動作を実装する命令を含む製品を製造するようにすることもできる。

【０１３３】

コンピュータ・プログラム命令を、コンピュータ、他のプログラム可能データ処理装置、又は他のデバイス上にロードして、そのコンピュータ、他のプログラム可能装置、又は他のデバイス上で一連の動作ステップを行わせてコンピュータ実装プロセスを生成し、それにより、そのコンピュータ又は他のプログラム可能装置上で実行される命令が、フローチャート及び／又はブロック図の１つ又は複数のブロックにおいて指定された機能／動作を実施するためのプロセスを提供するようにすることもできる。

【０１３４】

図面内のフローチャート及びブロック図は、種々の実施形態による、システム、方法及びコンピュータ・プログラムの可能な実装のアーキテクチャ、機能及び動作を示す。この点に関して、フローチャート又はブロック図内の各ブロックは、指定された論理機能を実行するための１つ又は複数の実行可能な命令を含む、モジュール、セグメント又はコードの一部を表すことができる。幾つかの代替的な実施において、ブロック内に記された機能は、図面内に記された順序とは異なる順序で行われることがあることにも留意すべきである。例えば、連続して示された２つのブロックは、関与する機能に応じて、実際には実質的に同時に実行されることもあり、又はこれらのブロックは、ときには逆の順序で実行されることもある。ブロック図及び／又はフローチャート図の各ブロック、並びにブロック図及び／又はフローチャート図内のブロックの組み合わせは、指定された機能又は動作を行う専用ハードウェアベースのシステム、又は専用ハードウェアとコンピュータ命令との組み合わせによって実装することができることにも留意されたい。

【０１３５】

上記に加えて、１つ又は複数の態様は、顧客環境の管理を提供するサービス・プロバイダによって供与、提供、配置、管理、サービス等を行うことができる。例えば、サービス・プロバイダは、１又は複数の顧客のために１つ又は複数の態様を実施するコンピュータ・コード及び／又はコンピュータ・インフラストラクチャの作成、保守、サポート等を行うことができる。見返りに、サービス・プロバイダは、例として、予約申し込み及び／又は報酬契約の下で顧客から支払いを受けることができる。付加的に又は代替的に、サービス・プロバイダは、１又は複数の第三者に対する広告コンテンツの販売から支払いを受けることができる。

【０１３６】

１つの態様において、１つ又は複数の態様を実施するために、アプリケーションを配置することができる。一例として、アプリケーションの配置は、１つ又は複数の態様を実施するように動作可能なコンピュータ・インフラストラクチャを提供することを含む。

【０１３７】

更に別の態様として、コンピュータ可読コードをコンピュータ・システムに統合することを含む、コンピュータ・インフラストラクチャを配置することができ、そこでは、コードは、コンピューティング・システムと協働して、１つ又は複数の態様を実施することができる。

【０１３８】

更に別の態様として、コンピュータ可読コードをコンピュータ・システムに統合することを含む、プロセスを提供することができる。コンピュータ・システムは、コンピュータ可読媒体を含み、ここで、コンピュータ媒体は、１つ又は複数の態様を含む。コードは、コンピュータ・システムと協働して、１つ又は複数の態様を実施することができる。

【０１３９】

種々の実施形態が上述されたが、これらは例にすぎない。例えば、他のアーキテクチャのコンピューティング環境は、１つ又は複数の態様を組み込み、使用することができる。さらに、他のサイズのベクトルを使用することができ、１つ又は複数の態様から逸脱することなく、命令への変更をなすことができる。加えて、他の実施形態においては、ベクトル・オペランドは、ベクトル・レジスタの代わりに、メモリ位置とすることができる。他の変形も可能である。

【０１４０】

さらに、他のタイプのコンピューティング環境は、１つ又は複数の態様から利益を得ることができる。一例として、プログラム・コードを格納及び／又は実行するのに適しており、システム・バスを介してメモリ要素に直接又は間接的に結合された少なくとも２つのプロセッサを含む、データ処理システムを使用することができる。メモリ要素は、例えば、プログラム・コードの実際の実行中に用いられるローカル・メモリ、大容量記憶装置、及び実行中に大容量記憶装置からコードを取り出さなければならない回数を減らすために少なくとも幾つかのプログラム・コードの一時的なストレージを提供するキャッシュ・メモリを含む。

【０１４１】

入力／出力即ちＩ／Ｏデバイス（これらに限定されるものではないが、キーボード、ディスプレイ、ポインティング・デバイス、ＤＡＳＤ、テープ、ＣＤ、ＤＶＤ、サムドライブ及び他のメモリ媒体等）は、直接システムに結合することもでき、又は介在するＩ／Ｏコントローラを介してシステムに結合することができる。ネットワーク・アダプタをシステムに結合させて、データ処理システムが、介在する私的ネットワーク又は公衆ネットワークを通じて他のデータ処理システム又は遠隔プリンタ若しくはストレージ・デバイスに結合できるようにすることもできる。モデム、ケーブル・モデム及びイーサネット・カードは、ネットワーク・アダプタの利用可能なタイプのうちのほんの数例である。

【０１４２】

図２５を参照すると、１つ又は複数の態様を実装するためのホスト・コンピュータ・システム５０００の代表的なコンポーネントが描かれる。代表的なホスト・コンピュータ５０００は、コンピュータ・メモリ（即ち、中央ストレージ）５００２と通信する１つ又は複数のＣＰＵ５００１と、他のコンピュータ若しくはＳＡＮなどとの通信のためのストレージ媒体デバイス５０１１及びネットワーク５０１０に対するＩ／Ｏインターフェースとを含む。ＣＰＵ５００１は、アーキテクチャ化命令セット及びアーキテクチャ化機能を有するアーキテクチャに準拠している。ＣＰＵ５００１は、プログラム・アドレス（仮想アドレス）をメモリの実アドレスに変換するための動的アドレス変換（ＤＡＴ）５００３を有することができる。ＤＡＴは、典型的には、変換をキャッシュに入れるための変換ルックアサイド・バッファ（ＴＬＢ）５００７を含み、後でコンピュータ・メモリ５００２のブロックにアクセスしたときにアドレス変換による遅延を必要とせずに済むように変換をキャッシュに入れるための、変換ルックアサイド・バッファ（ＴＬＢ）５００７を含む。典型的には、キャッシュ５００９は、コンピュータ・メモリ５００２とプロセッサ５００１との間で用いられる。キャッシュ５００９は、１つより多くのＣＰＵが利用できる大型キャッシュと、大型キャッシュと各ＣＰＵとの間のより小型で高速な（下位レベルの）キャッシュとを有する階層構造とすることができる。幾つかの実施において、下位レベルのキャッシュは、命令フェッチ及びデータ・アクセスのための個別の下位レベル・キャッシュを提供するように分割される。一実施形態において、命令は、命令フェッチ・ユニット５００４によりメモリ５００２からキャッシュ５００９を介してフェッチされる。命令は、命令デコード・ユニット５００６内でデコードされ（幾つかの実施形態においては他の

命令と共に）、命令実行ユニット５００８にディスパッチされる。典型的には、幾つかの実行ユニット５００８、例えば、算術演算実行ユニット、浮動小数点実行ユニット及び分岐命令実行ユニットが用いられる。命令は、実行ユニットにより、必要に応じて命令が指定するレジスタ又はメモリからのオペランドにアクセスすることにより実行される。オペランドがメモリ５００２からアクセスされる（ロードされる又はストアされる）場合には、典型的には、ロード／ストア・ユニット５００５が、実行されている命令の制御下でアクセスを取り扱う。命令は、ハードウェア回路若しくは内部マイクロコード（ファームウェア）、又はこの両方の組み合わせにより実行することができる。

【０１４３】

既述のように、コンピュータ・システムは、ローカル（又は、主）ストレージ内の情報、並びにアドレス指定、保護、並びに参照及び変更記録を含む。アドレス指定の幾つかの態様は、アドレスの形式、アドレス空間の概念、アドレスの種々のタイプ及び１つのタイプのアドレスが別のタイプのアドレスに変換される方法を含む。主ストレージの一部は、恒久的に割り当てられたストレージ位置を含む。主ストレージは、システムに、直接アドレス可能なデータの高速アクセス・ストレージを提供する。データ及びプログラムの両方とも、これらが処理される前に（入力デバイスから）主ストレージにロードされる。

【０１４４】

主ストレージは、キャッシュと呼ばれることがある、１つ又は複数のより小型の高速アクセス・バッファ・ストレージを含むことができる。キャッシュは、典型的には、ＣＰＵ又はＩ／Ｏプロセッサと物理的に関連付けられる。物理的構造の、性能を除いた効果及び別個のストレージ媒体の使用は、一般に、プログラムにより観察することができない。

【０１４５】

命令及びデータ・オペランドに対して、別個のキャッシュを維持することができる。キャッシュ内の情報は、キャッシュ・ブロック又はキャッシュ・ライン（又は、簡単に言えばライン）と呼ばれる整数境界上の連続バイトで維持される。モデルは、キャッシュ・ラインのサイズをバイト単位で戻すＥＸＴＲＡＣＴ　ＣＡＣＨＥ　ＡＴＴＲＩＢＵＴＥ命令を提供することができる。モデルはまた、データ又は命令キャッシュへのストレージのプリフェッチ又はキャッシュからのデータの解放を行うＰＲＥＦＥＴＣＨ　ＤＡＴＡ及びＰＲＥＦＥＴＣＨ　ＤＡＴＡ　ＲＥＬＡＴＩＶＥ　ＬＯＮＧ命令も提供することができる。

【０１４６】

ストレージは、ビットの水平の長い文字列として見ることができる。殆どの操作では、ストレージへのアクセスは、左から右への順で進行する。ビットの文字列は、８ビット単位で細分される。この８ビットの単位はバイトと呼ばれ、これは全ての情報形式の基本構成単位である。ストレージ内の各々のバイト位置は、負でない固有の整数により識別され、この整数がバイト位置のアドレス、即ち、簡単にバイト・アドレスである。隣接するバイト位置は連続するアドレスを有し、左端の０から始まって左から右へ順に進行する。アドレスは、符号なしの２進整数であり、２４ビット、３１ビット又は６４ビットである。

【０１４７】

情報は、ストレージとＣＰＵ又はチャネル・サブシステムとの間で、一度に１バイトずつ、又は１グループ分のバイトで伝送される。特に断りのない限り、例えばｚ／Ａｒｃｈｉｔｅｃｔｕｒｅにおいて、ストレージ内のバイト・グループは、グループの左端のバイトによりアドレス指定される。グループ内のバイト数は、実行される操作により暗黙的に決定される場合、又は明示的に決定される場合がある。ＣＰＵ操作に使用される場合、バイト・グループはフィールドと呼ばれる。各々のバイト・グループ内において、例えばｚ／Ａｒｃｈｉｔｅｃｔｕｒｅにおいて、ビットは、左から右の順に番号付けされる。ｚ／アーキテクチャにおいて、左端のビットを「最上位」ビットと呼び、右端のビットを「最下位」ビットと呼ぶことがある。しかしながら、ビット番号はストレージ・アドレスではない。アドレス指定できるのはバイトだけである。ストレージ内の１つのバイトの個々のビットに対して操作を行うためには、そのバイト全体にアクセスされる。１バイトの中のビットには、（ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅにおいて）左から右に０から７までの番号

が付けられる。１つのアドレスの中のビットには、２４ビット・アドレスの場合は、８－３１若しくは４０－６３の番号が付けられ、又は３１ビット・アドレスの場合は、１－３１若しくは３３－６３の番号が付けられ、又は６４ビット・アドレスの場合は、０－６３の番号が付けられる。複数バイトの任意の他の固定長形式において、形式を構成するビットは、０から始まって連続的に番号が付けられる。エラー検出のため、また好ましくは訂正のため、各々のバイト又はバイト・グループと共に１又は複数の検査ビットを伝送することができる。こうした検査ビットは、マシンにより自動的に生成され、プログラムにより直接制御することはできない。ストレージ容量は、バイト数で表現される。ストレージ・オペランド・フィールドの長さが命令のオペコードで暗黙指定される場合、そのフィールドは固定長を有するといわれ、この長さは１バイト、２バイト、４バイト、８バイト又は１６バイトとすることができる。幾つかの命令に対しては、より大きいフィールドが暗黙指定される。ストレージ・オペランドの長さが暗黙指定されず、明示的に指定される場合は、そのフィールドは可変長を有するといわれる。可変長オペランドは、１バイトのインクリメント（又は幾つかの命令では、２バイトの倍数又は他の倍数で）で長さが変化し得る。情報がストレージ内に配置されると、ストレージへの物理的パスの幅が格納されるフィールドの長さより大きい場合であっても、指定したフィールドに含まれているバイト位置のコンテンツのみが置き換えられる。

【０１４８】

情報の特定の単位は、ストレージ内の整数境界上にあるべきである。境界は、そのストレージ・アドレスがバイトでの単位の長さの倍数である場合に、情報の単位に対して整数であると呼ばれる。整数境界上にある２バイト、４バイト、８バイト、及び１６バイトのフィールドには、特別な名称が与えられる。ハーフワードは、２バイト境界上にある２個の連続したバイトのグループであり、これは命令の基本構成単位である。ワードは、４バイト境界上にある４個の連続したバイトのグループである。ダブルワードは、８バイト境界上にある８個の連続したバイトのグループである。クワッドワードは、１６バイト境界上にある１６個の連続したバイトのグループである。ストレージ・アドレスが、ハーフワード、ワード、ダブルワード、及びクワッドワードを指定するとき、そのアドレスの２進表現では、それぞれ１個、２個、３個、又は４個の右端の０ビットを含む。命令は、２バイト整数境界上にあるべきである。殆どの命令のストレージ・オペランドは、境界位置合わせ要件を有さない。

【０１４９】

命令及びデータ・オペランドに対して別個のキャッシュを実装するデバイスにおいては、ストアが後にフェッチされる命令を変更するかどうかに関係なく、プログラムが、後にフェッチされるキャッシュ・ラインに格納される場合、著しい遅延が生じ得る。

【０１５０】

一実施形態において、本発明は、ソフトウェア（ライセンス内部コード、ファームウェア、マイクロコード、ミリコード、ピココードなどとも呼ばれる場合があるが、そのいずれも１つ又は複数の態様と整合性がある）により実施することができる。図２５を参照すると、１つ又は複数の態様を具体化するソフトウェア・プログラム・コードは、ＣＤ－ＲＯＭドライブ、テープドライブ、又はハードドライブといった長期ストレージ媒体デバイス５０１１から、ホスト・システム５０００のプロセッサ５００１によりアクセスすることができる。ソフトウェア・プログラム・コードは、ディスケット、ハードドライブ、又はＣＤ－ＲＯＭのようなデータ処理システムと共に使用するための種々の周知の媒体のいずれかの上で具体化することができる。コードは、こうした媒体上に分散させても、又はコンピュータ・メモリ５００２からユーザに分散させても、又はこうした他のシステムのユーザが使用するために、ネットワーク５０１０上の１つのコンピュータ・システムのストレージから他のコンピュータ・システムに分散させてもよい。

【０１５１】

ソフトウェア・プログラム・コードは、種々のコンピュータ・コンポーネント及び１つ又は複数のアプリケーション・プログラムの機能及び相互作用を制御するオペレーティン

グ・システムを含む。プログラム・コードは、通常、ストレージ媒体デバイス５０１１から相対的により高速のコンピュータ・ストレージ５００２にページングされ、そこでプロセッサ５００１による処理のために利用可能になる。ソフトウェア・プログラム・コードをメモリ内、物理的媒体上で具体化し、及び／又は、ネットワークを介してソフトウェア・コードを配布する技術及び方法は周知であり、ここではこれ以上論じない。プログラム・コードは、有形の媒体（これらに限定されるものではないが、電子メモリ・モジュール（ＲＡＭ）、フラッシュ・メモリ、コンパクト・ディスク（ＣＤ）、ＤＶＤ、磁気テープ等）上に作成され格納されたとき、「コンピュータ・プログラム」と呼ばれることが多い。コンピュータ・プログラム媒体は、典型的には、処理回路による実行のために、好ましくはコンピュータ・システム内の処理回路によって読み取り可能である。

【０１５２】

図２６は、１つ又は複数の態様を実施できる代表的なワークステーション又はサーバ・ハードウェア・システムを示す。図２６のシステム５０２０は、随意的な周辺機器を含む、パーソナル・コンピュータ、ワークステーション、又はサーバなどの代表的なベース・コンピュータ・システム５０２１を含む。ベース・コンピュータ・システム５０２１は、１つ又は複数のプロセッサ５０２６と、周知の技術に従ってプロセッサ５０２６とシステム５０２１の他のコンポーネントを接続し、これらの間の通信を可能にするために用いられるバスとを含む。バスは、プロセッサ５０２６を、ハードドライブ（例えば、磁気媒体、ＣＤ、ＤＶＤ及びフラッシュ・メモリのいずれかを含む）又はテープドライブを含むことができる、メモリ５０２５及び長期ストレージ５０２７に接続する。システム５０２１はまた、バスを介して、マイクロプロセッサ５０２６を、キーボード５０２４、マウス５０２３、プリンタ／スキャナ５０３０、及び／又はタッチ・センシティブ・スクリーン、デジタル化された入力パッド等のいずれかのユーザ・インターフェース機器とすることができる他のインターフェース機器といった、１つ又は複数のインターフェース機器に接続する、ユーザ・インターフェース・アダプタを含むこともできる。バスはまた、ディスプレイ・アダプタを介して、ＬＣＤスクリーン又はモニタなどのディスプレイ装置５０２２をマイクロプロセッサ５０２６にも接続する。

【０１５３】

システム５０２１は、ネットワーク５０２９と通信する５０２８ことができるネットワーク・アダプタを介して、他のコンピュータ又はコンピュータ・ネットワークと通信することができる。例示的なネットワーク・アダプタは、通信チャネル、トークン・リング、イーサネット又はモデムである。代替的に、システム５０２１は、ＣＤＰＤ（セルラー・デジタル・パケット・データ）カードのような無線インターフェースを用いて通信することもできる。システム５０２１は、ローカル・エリア・ネットワーク（ＬＡＮ）又は広域ネットワーク（ＷＡＮ）又はシステム５０２１内のこうした他のコンピュータと関連付けることができ、又は、別のコンピュータ等とのクライアント／サーバ構成におけるクライアントとすることができる。これら構成の全て、並びに、適切な通信ハードウェア及びソフトウェアは、当技術分野において周知である。

【０１５４】

図２７は、１つ又は複数の態様を実施することができるデータ処理ネットワーク５０４０を示す。データ処理ネットワーク５０４０は、各々が複数の個々のワークステーション５０４１、５０４２、５０４３、５０４４を含むことができる、無線ネットワーク及び有線ネットワークのような複数の個々のネットワークを含むことができる。さらに、当業者であれば理解するように、１つ又は複数のＬＡＮを含ませることができ、そこで、ＬＡＮは、ホスト・プロセッサに結合された複数のインテリジェント・ワークステーションを含むことができる。

【０１５５】

さらに図２７を参照すると、ネットワークはまた、ゲートウェイ・コンピュータ（クライアント・サーバ５０４６）、又はアプリケーション・サーバ（データ・リポジトリにアクセスすることができ、且つ、ワークステーション５０４５から直接アクセスすることも

できる遠隔サーバ５０４８）のような、メインフレーム・コンピュータ又はサーバを含むこともできる。ゲートウェイ・コンピュータ５０４６は、各々の個々のネットワークへの入力点のとして働く。ゲートウェイは、１つのネットワーク・プロトコルを別のものに接続するときに必要とされる。ゲートウェイ５０４６は、通信リンクによって別のネットワーク（例えば、インターネット５０４７）に結合できることが好ましい。ゲートウェイ５０４６はまた、通信リンクを用いて、１つ又は複数のワークステーション５０４１、５０４２、５０４３、５０４４に直接結合することもできる。ゲートウェイ・コンピュータは、インターナショナル・ビジネス・マシーンズ・コーポレーションから入手可能なＩＢＭ　ｅＳｅｒｖｅｒ　Ｓｙｓｔｅｍ　ｚサーバを用いて実装することができる。

【０１５６】

図２６及び図２７を同時に参照すると、本発明の１つ又は複数の態様を具体化することができるソフトウェア・プログラム・コードには、一般的に、ＣＤ－ＲＯＭドライブ又はハードドライブといった長期ストレージ媒体５０２７から、システム５０２０のプロセッサ５０２６によってアクセスすることができる。ソフトウェア・プログラム・コードは、ディスケット、ハードドライブ、又はＣＤ－ＲＯＭといった、データ処理システムと共に用いるための種々の周知の媒体のいずれかの上で具体化することができる。コードは、そのような媒体上で分散させても、又はメモリからユーザ５０５０、５０５１に分散させても、又は、こうした他のシステムのユーザが用いるために、ネットワーク上の１つのコンピュータ・システムのメモリ若しくはストレージから他のコンピュータ・システムに分散させてもよい。

【０１５７】

代替的に、プログラム・コードをメモリ５０２５内で具体化し、プロセッサ・バスを用いてプロセッサ５０２６によってプログラム・コードにアクセスすることができる。このようなプログラム・コードは、種々のコンピュータ・コンポーネント及び１つ又は複数のアプリケーション・プログラム５０３２の機能及び相互作用を制御するオペレーティング・システムを含む。プログラム・コードは、通常、ストレージ媒体５０２７から高速メモリ５０２５にページングされ、そこでプロセッサ５０２６による処理のために利用可能になる。ソフトウェア・プログラム・コードをメモリ内、物理的媒体上で具体化し、及び／又は、ネットワークを介してソフトウェア・コードを配布する技術及び方法は周知であり、ここではこれ以上論じない。プログラム・コードは、有形の媒体（これらに限定されるものではないが、電子メモリ・モジュール（ＲＡＭ）、フラッシュ・メモリ、コンパクト・ディスク（ＣＤ）、ＤＶＤ、磁気テープなどを含む）に格納されたとき、「コンピュータ・プログラム」と呼ばれることが多い。コンピュータ・プログラム媒体は、典型的には、処理回路による実行のために、好ましくはコンピュータ・システム内の処理回路によって読み取り可能である。

【０１５８】

プロセッサが最も容易に利用できるキャッシュ（通常、プロセッサの他のキャッシュよりも高速で小さい）は、最下位（Ｌ１又はレベル１）のキャッシュであり、主ストア（主メモリ）は、最上位レベルのキャッシュ（３つのレベルがある場合にはＬ３）である。最下位レベルのキャッシュは、実行されるマシン命令を保持する命令キャッシュ（Ｉ－キャッシュ）と、データ・オペランドを保持するデータ・キャッシュ（Ｄ－キャッシュ）とに分割されることが多い。

【０１５９】

図２８を参照すると、プロセッサ５０２６についての例示的なプロセッサの実施形態が示される。典型的には、メモリ・ブロックをバッファに入れてプロセッサ性能を向上させるために、１つ又は複数のレベルのキャッシュ５０５３が用いられる。キャッシュ５０５３は、用いられる可能性が高いメモリ・データのキャッシュ・ラインを保持する高速バッファである。典型的なキャッシュ・ラインは、６４バイト、１２８バイト、又は２５６バイトのメモリ・データである。データをキャッシュに入れるのではなく、命令をキャッシュに入れるために、別個のキャッシュが用いられることが多い。キャッシュ・コヒーレン

ス（メモリ及びキャッシュ内のラインのコピーの同期）は、多くの場合、当技術分野において周知の種々の「スヌープ」アルゴリズムによって与えられる。プロセッサ・システムの主メモリ・ストレージ５０２５は、キャッシュと呼ばれることが多い。４つのレベルのキャッシュ５０５３を有するプロセッサ・システムにおいて、主ストレージ５０２５は、典型的にはより高速であり、且つ、コンピュータ・システムが利用できる不揮発性ストレージ（ＤＡＳＤ、テープ等）の一部だけを保持するので、レベル５（Ｌ５）のキャッシュと呼ばれることがある。主ストレージ５０２５は、オペレーティング・システムによって主ストレージ５０２５との間でページングされるデータのページを「キャッシュに入れる」。

【０１６０】

プログラム・カウンタ（命令カウンタ）５０６１は、実行される現行の命令のアドレスを常時監視している。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅプロセッサのプログラム・カウンタは６４ビットであり、従来のアドレッシング制限をサポートするために、３１ビット又は２４ビットに切り捨てることができる。プログラム・カウンタは、典型的には、コンテキスト・スイッチの際に持続するように、コンピュータのＰＳＷ（プログラム状況ワード）内で具体化される。従って、例えば、オペレーティング・システムにより、プログラム・カウンタ値を有する進行中のプログラムに割り込みをかけることが可能である（プログラム環境からオペレーティング・システム環境へのコンテキスト・スイッチ）。プログラムのＰＳＷは、プログラムがアクティブでない間、プログラム・カウンタ値を保持し、オペレーティング・システムが実行されている間、オペレーティング・システムの（ＰＳＷ内の）プログラム・カウンタが用いられる。典型的には、プログラム・カウンタは、現行の命令のバイト数に等しい量だけインクリメントされる。ＲＩＳＣ命令は、典型的には固定長であり、ＣＩＳＣ命令は、典型的には可変長である。ＩＢＭ ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅの命令は、２バイト、４バイト、又は６バイトの長さを有するＣＩＳＣ命令である。例えば、コンテキスト・スイッチ操作又は分岐命令の分岐成立操作により、プログラム・カウンタ５０６１が変更される。コンテキスト・スイッチ操作において、現行のプログラム・カウンタ値は、実行されるプログラムについての他の状態情報（条件コードのような）と共にプログラム状況ワード内に保存され、実行される新しいプログラム・モジュールの命令を指し示す新しいプログラム・カウンタ値がロードされる。分岐成立操作を行い、分岐命令の結果をプログラム・カウンタ５０６１にロードすることにより、プログラムが判断を下すこと又はプログラム内でループすることを可能にする。

【０１６１】

典型的には、プロセッサ５０２６の代わりに命令をフェッチするために、命令フェッチ・ユニット５０５５が用いられる。フェッチ・ユニットは、「次の順次命令」、分岐成立命令のターゲット命令、又はコンテキスト・スイッチの後のプログラムの最初の命令のいずれかをフェッチする。今日の命令フェッチ・ユニットは、プリフェッチされた命令を用いることができる可能性に基づいて、命令を投機的にプリフェッチするプリフェッチ技術を用いることが多い。例えば、フェッチ・ユニットは、次の順次命令を含む１６バイトの命令と、付加的なバイトの更なる順次命令とをフェッチすることができる。

【０１６２】

次いで、フェッチされた命令が、プロセッサ５０２６によって実行される。一実施形態において、フェッチされた命令は、フェッチ・ユニットのディスパッチ・ユニット５０５６に渡される。ディスパッチ・ユニットは命令をデコードし、デコードされた命令についての情報を適切なユニット５０５７、５０５８、５０６０に転送する。実行ユニット５０５７は、典型的には、命令フェッチ・ユニット５０５５からデコードされた算術命令についての情報を受け取り、命令のオペコードに従ってオペランドに関する算術演算を行う。オペランドは、好ましくは、メモリ５０２５、アーキテクチャ化レジスタ５０５９、又は実行される命令の即値フィールドのいずれかから、実行ユニット５０５７に与えられる。実行の結果は、格納された場合には、メモリ５０２５、レジスタ５０５９、又は他のマシン・ハードウェア（制御レジスタ、ＰＳＷレジスタなどのような）内に格納される。

【０１６３】
プロセッサ５０２６は、典型的には、命令の機能を実行するための１つ又は複数の実行ユニット５０５７、５０５８、５０６０を有する。図２９を参照すると、実行ユニット５０５７は、インターフェース論理５０７１を介して、アーキテクチャ化された汎用レジスタ５０５９、デコード／ディスパッチ・ユニット５０５６、ロード・ストア・ユニット５０６０、及び他のプロセッサ・ユニット５０６５と通信することができる。実行ユニット５０５７は、幾つかのレジスタ回路５０６７、５０６８、５０６９を用いて、算術論理演算ユニット（ＡＬＵ）５０６６が動作する情報を保持することができる。ＡＬＵは、加算、減算、乗算、及び除算などの算術演算、並びに、論理積、論理和、及び排他的論理和、ローテート及びシフトのような論理関数を実行する。ＡＬＵは、設計に依存する専用の演算をサポートすることが好ましい。他の回路は、例えば条件コード及び回復サポート論理を含む、他のアーキテクチャ化ファシリティ５０７２を提供することができる。典型的には、ＡＬＵ演算の結果は、出力レジスタ回路５０７０に保持され、この出力レジスタ回路５０７０が、結果を種々の他の処理機能に転送することができる。多数のプロセッサ・ユニットの構成が存在し、本説明は、一実施形態の代表的な理解を与えることのみを意図している。

【０１６４】
例えばＡＤＤ命令は、算術及び論理機能を有する実行ユニット５０５７で実行され、一方、例えば浮動小数点命令は、特化された浮動小数点能力を有する浮動小数点実行部で実行される。実行ユニットは、オペランドに対してオペコードが定めた関数を行うことにより、命令が特定したオペランドに対して動作することが好ましい。例えば、ＡＤＤ命令は、命令のレジスタ・フィールドによって特定された２つのレジスタ５０５９内に見出されるオペランドに対して、実行ユニット５０５７により実行することができる。

【０１６５】
実行ユニット５０５７は、２つのオペランドに対して算術加算を実行し、結果を第３オペランドに格納し、ここで第３オペランドは、第３のレジスタであっても又は２つのソース・レジスタのいずれかであってもよい。実行ユニットは、シフト、ローテート、論理積、論理和、及び排他的論理和のような種々の論理関数、並びに、加算、減算、乗算、除法のいずれかを含む、種々の代数関数を実行することができる算術論理演算ユニット（ＡＬＵ）５０６６を用いることが好ましい。スカラー演算のために設計されたＡＬＵ５０６６もあり、浮動小数点のために設計されたものＡＬＵ５０６６もある。データは、アーキテクチャに応じて、ビッグ・エンディアン（最下位のバイトが最も高いバイト・アドレスである）、又はリトル・エンディアン（最下位のバイトが最も低いバイト・アドレスである）とすることができる。ＩＢＭ　ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅは、ビッグ・エンディアンである。符号付きフィールドは、アーキテクチャに応じて、符号及び大きさ、１の補数、又は２の補数とすることができる。２の補数における負の値又は正の値は、ＡＬＵ内で加法しか必要としないため、ＡＬＵが減算能力を設計する必要がないという点で、２の補数は有利である。数値は、通常、省略表現で記述され、１２ビット・フィールドは、４，０９６バイトブロックのアドレスを定め、通常、例えば４Ｋバイト（キロバイト）ブロックのように記述される。

【０１６６】
図３０を参照すると、分岐命令を実行するための分岐命令情報が、典型的には、分岐ユニット５０５８に送られ、この分岐ユニット５０５８は、多くの場合、分岐履歴テーブル５０８２のような分岐予測アルゴリズムを用いて、他の条件付き演算が完了する前に分岐の結果を予測する。条件付き演算が完了する前に、現行の分岐命令のターゲットがフェッチされ、投機的に実行される。条件付き演算が完了すると、投機的に実行された分岐命令は、条件付き演算の条件及び投機された結果に基づいて、完了されるか又は破棄される。典型的な分岐命令は、条件コードをテストし、条件コードが分岐命令の分岐要件を満たす場合、ターゲット・アドレスに分岐することができ、ターゲット・アドレスは、例えば、命令のレジスタ・フィールド又は即値フィールド内に見出されるものを含む幾つかの数に

基づいて計算することができる。分岐ユニット５０５８は、複数の入力レジスタ回路５０７５、５０７６、５０７７と、出力レジスタ回路５０８０とを有するＡＬＵ５０７４を用いることができる。分岐ユニット５０５８は、例えば、汎用レジスタ５０５９、デコード・ディスパッチ・ユニット５０５６、又は他の回路５０７３と通信することができる。

【０１６７】

例えば、オペレーティング・システムによって開始されるコンテキスト・スイッチ、コンテキスト・スイッチを発生させるプログラム例外又はエラー、コンテキスト・スイッチを発生させるＩ／Ｏ割り込み信号、又は（マルチスレッド環境における）複数のプログラムのマルチスレッド活動を含む様々な理由により、命令のグループの実行に割り込みがかけられることがある。コンテキスト・スイッチ動作は、現在実行中のプログラムについての状態情報を保存し、次いで、起動される別のプログラムについての状態情報をロードすることが好ましい。状態情報は、例えば、ハードウェア・レジスタ又はメモリ内に保存することができる。状態情報は、実行される次の命令を指し示すプログラム・カウンタ値と、条件コードと、メモリ変換情報と、アーキテクチャ化されたレジスタのコンテンツとを含むことが好ましい。コンテキスト・スイッチの活動は、ハードウェア回路、アプリケーション・プログラム、オペレーティング・システム・プログラム、又はファームウェア・コード（マイクロコード、ピココード、又はライセンス内部コード（ＬＩＣ））単独で又はその組み合わせで実施することができる。

【０１６８】

プロセッサは、命令により定義された方法に従ってオペランドにアクセスする。命令は、命令の一部の値を用いて即値オペランドを与えることができ、汎用レジスタ又は専用レジスタ（例えば、浮動小数点レジスタ）のいずれかを明示的に指し示す１つ又は複数のレジスタ・フィールドを与えることができる。命令は、オペコード・フィールドによって、オペランドとして識別されるインプライド・レジスタを用いることができる。命令は、オペランドのためのメモリ位置を用いることができる。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅの長変位ファシリティにより例示されるように、オペランドのメモリ位置を、レジスタ、即値フィールド、又はレジスタと即値フィールドの組み合わせによって与えることができ、命令は、ベース・レジスタ、索引レジスタ、及び即値フィールド（変位フィールド）を定め、これらが、例えば互いに加算されてメモリ内のオペランドのアドレスをもたらす。ここでの位置は、典型的には、特に断りのない限り、主メモリ（主ストレージ）内の記憶位置を意味する。

【０１６９】

図３１を参照すると、プロセッサは、ロード／ストア・ユニット５０６０を用いて、ストレージにアクセスする。ロード／ストア・ユニット５０６０は、メモリ５０２５内のターゲット・オペランドのアドレスを取得し、オペランドをレジスタ５０５９又は別のメモリ５０２５の記憶位置にロードすることによってロード操作を行うことができ、或いは、メモリ５０２５内のターゲット・オペランドのアドレスを取得し、レジスタ５０５９又は別のメモリ５０２５の記憶位置から取得したデータをメモリ５０２５内のターゲット・オペランドの記憶位置に格納することによって、ストア操作を行うことができる。ロード／ストア・ユニット５０６０は、投機的なものであってもよく、命令シーケンスに対してアウト・オブ・オーダー式の順序でメモリにアクセスすることができるが、プログラムに対して、命令がインオーダー式に実行されたという外観を維持することになる。ロード／ストア・ユニット５０６０は、汎用レジスタ５０５９、デコード／ディスパッチ・ユニット５０５６、キャッシュ／メモリ・インターフェース５０５３、又は他の要素５０８３と通信することができ、ストレージ・アドレスを計算し、且つ、パイプライン処理を順に行って操作をインオーダー式に保持するための、種々のレジスタ回路、ＡＬＵ５０８５、及び制御論理５０９０を含む。一部の動作は、アウト・オブ・オーダー式とすることができるが、ロード／ストア・ユニットは、アウト・オブ・オーダー式動作が、プログラムに対して、当技術分野において周知のようなインオーダー式に実行されたように見えるようにする機能を提供する。

【０１７０】
好ましくは、アプリケーション・プログラムが「見ている」アドレスは、仮想アドレスと呼ばれることが多い。仮想アドレスは、「論理アドレス」及び「実効アドレス」と呼ばれることもある。これらの仮想アドレスは、これらに限定されるものではないが、単に仮想アドレスをオフセット値にプリフィックス付加すること、１つ又は複数の変換テーブルを介して仮想アドレスを変換することを含む、種々の動的アドレス変換（ＤＡＴ）技術の１つによって、物理的メモリ位置にリダイレクトされるという点で仮想のものであり、変換テーブルは、少なくともセグメント・テーブル及びページ・テーブルを単独で又は組み合わせて含むことが好ましく、セグメント・テーブルは、ページ・テーブルを指し示すエントリを有することが好ましい。ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅでは、領域第１テーブル、領域第２テーブル、領域第３テーブル、セグメント・テーブル、及び随意的なページ・テーブルを含む、変換の階層構成が提供される。アドレス変換の性能は、仮想アドレスを関連した物理的メモリ位置にマッピングするエントリを含む変換ルックアサイド・バッファ（ＴＬＢ）を用いることにより改善されることが多い。ＤＡＴが変換テーブルを用いて仮想アドレスを変換したときに、エントリが作成される。次いで、後に仮想アドレスを用いることで、低速の順次変換テーブル・アクセスではなく、高速のＴＬＢのエントリを用いることが可能になる。ＴＬＢのコンテンツは、ＬＲＵを含む種々の置換アルゴリズムによって管理することができる。

【０１７１】
プロセッサがマルチプロセッサ・システムのプロセッサである場合には、各プロセッサは、コヒーレンシのために、Ｉ／Ｏ、キャッシュ、ＴＬＢ、及びメモリといった共有リソースをインターロック状態に保持する責任を負う。キャッシュ・コヒーレンシを保持する際に、一般的には「スヌープ」技術が用いられる。スヌープ環境においては、共有を容易にするために、各キャッシュ・ラインを、共有状態、排他的状態、変更状態、無効状態等のいずれか１つの状態にあるものとしてマーク付けすることができる。

【０１７２】
Ｉ／Ｏユニット５０５４（図２８）は、プロセッサに、例えば、テープ、ディスク、プリンタ、ディスプレイ、及びネットワークを含む周辺機器に取り付けるための手段を与える。Ｉ／Ｏユニットは、ソフトウェア・ドライバによってコンピュータ・プログラムに提示されることが多い。ＩＢＭによるＳｙｓｔｅｍ ｚのようなメインフレームにおいては、チャネル・アダプタ及びオープン・システム・アダプタが、オペレーティング・システムと周辺機器との間に通信をもたらすメインフレームのＩ／Ｏユニットである。

【０１７３】
さらに、他のタイプのコンピューティング環境が、１つ又は複数の態様から利益を得ることができる。一例として、環境は、特定のアーキテクチャ（例えば、命令実行、アドレス変換などのアーキテクチャ化された機能、及びアーキテクチャ化されたレジスタを含む）又はそのサブセットをエミュレートする（例えば、プロセッサ及びメモリを有するネイティブ・コンピュータ・システム上で）エミュレータ（例えば、ソフトウェア又は他のエミュレーション機構）を含むことができる。このような環境においては、エミュレータを実行しているコンピュータが、エミュレートされる機能とは異なるアーキテクチャを有することができたとしても、エミュレータの１つ又は複数のエミュレーション機能は、本発明の１つ又は複数の態様を実施することができる。一例として、エミュレーション・モードにおいては、エミュレートされる特定の命令又は操作がデコードされ、適切なエミュレーション機能が構築され、個々の命令又は操作を実施する。

【０１７４】
エミュレーション環境においては、ホスト・コンピュータは、例えば、命令及びデータを格納するためのメモリと、メモリから命令をフェッチし、随意的に、フェッチされた命令のためのローカル・バッファリングを提供するための命令フェッチ・ユニットと、フェッチされた命令を受信し、フェッチされた命令のタイプを判断するための命令デコード・ユニットと、命令を実行するための命令実行ユニットとを含む。実行は、データをメモリ

からレジスタ内にロードすること、データをレジスタから再びメモリに格納すること、又はデコード・ユニットにより判断されるように、何らかのタイプの算術演算又は論理演算を実行することを含むことができる。一例においては、各ユニットは、ソフトウェアで実装される。例えば、ユニットが実行する演算は、エミュレータ・ソフトウェア内の１つ又は複数のサブルーチンとして実装される。

【０１７５】

より具体的には、メインフレームにおいて、アーキテクチャ化されたマシン命令は、通常、プログラマによって、多くの場合コンパイラ・アプリケーションを介して、今日では「Ｃ」プログラマによって用いられる。ストレージ媒体内に格納されたこれらの命令は、ｚ／ＡｒｃｈｉｔｅｃｔｕｒｅのＩＢＭサーバにおいて、又は代替的に他のアーキテクチャを実行するマシンにおいて、ネイティブに実行することができる。これらの命令は、既存の及び将来のＩＢＭメインフレーム・サーバにおいて、及び、ＩＢＭの他のマシン（例えば、Ｐｏｗｅｒ　Ｓｙｓｔｅｍｓサーバ及びＳｙｓｔｅｍ　ｘ（登録商標）サーバ）上で、エミュレートすることができる。これらの命令は、ＩＢＭ（登録商標）社、Ｉｎｔｅｌ（登録商標）社、ＡＭＤ（商標）社などによって製造されたハードウェアを用いて種々のマシン上でＬｉｎｕｘ（登録商標）を実行しているマシンにおいて実行することができる。Ｚ／Ａｒｃｈｉｔｅｃｔｕｒｅ下でそのハードウェア上で実行することに加えて、Ｌｉｎｕｘを用いること、並びに、一般に実行がエミュレーション・モードにあるＨｅｒｃｕｌｅｓ、ＵＭＸ、又はＦＳＩ（Ｆｕｎｄａｍｅｎｔａｌ　Ｓｏｆｔｗａｒｅ，Ｉｎｃ）によるエミュレーションを用いるマシンを用いることもできる。エミュレーション・モードにおいては、ネイティブ・プロセッサによって、エミュレーション・ソフトウェアが実行され、エミュレートされたプロセッサのアーキテクチャをエミュレートする。

【０１７６】

ネイティブ・プロセッサは、一般的に、エミュレートされたプロセッサのエミュレーションを実行するためにファームウェア又はネイティブ・オペレーティング・システムのいずれかを含むエミュレーション・ソフトウェアを実行する。エミュレーション・ソフトウェアは、エミュレートされたプロセッサ・アーキテクチャの命令のフェッチと実行を担当する。エミュレーション・ソフトウェアは、エミュレートされたプログラム・カウンタを維持し、命令境界を常時監視している。エミュレーション・ソフトウェアは、一度に１つ又は複数のエミュレートされたマシン命令をフェッチし、ネイティブ・プロセッサにより実行するために、その１つ又は複数のエミュレートされたマシン命令を、対応するネイティブマシン命令のグループに変換することができる。これらの変換された命令は、より速い変換を達成できるようにキャッシュに入れることができる。それにも関わらず、エミュレーション・ソフトウェアは、エミュレートされたプロセッサ・アーキテクチャのアーキテクチャ規則を維持して、オペレーティング・システム及びエミュレートされたプロセッサのために書かれたアプリケーションが正確に動作することを保証しなければならない。さらに、エミュレーション・ソフトウェアは、これらに限定されるものではないが、制御レジスタ、汎用レジスタ、浮動小数点レジスタ、例えばセグメント・テーブル及びページ・テーブルを含む動的アドレス変換機能、割り込み機構、コンテキスト・スイッチ機構、時刻（ＴＯＤ）クロック、及びＩ／Ｏサブシステムへのアーキテクチャ化インターフェースを含む、エミュレートされたプロセッサのアーキテクチャによって識別されるリソースを提供し、オペレーティング・システム又はエミュレートされたプロセッサ上で実行するように設計されたアプリケーション・プログラムが、エミュレーション・ソフトウェアを有するネイティブ・プロセッサ上で実行することができる。

【０１７７】

エミュレートされた特定の命令がデコードされ、個々の命令の機能を実行するためのサブルーチンが呼び出される。エミュレートされたプロセッサの機能をエミュレートするエミュレーション・ソフトウェア機能は、例えば、「Ｃ」サブルーチン若しくはドライバにおいて、又は１つ又は複数の実施形態の説明を理解した後で当業者の技術の範囲内にあるような特定のハードウェアのためにドライバを提供する他の何らかの方法で実装される。

Ｂｅａｕｓｏｌｅｉｌ他による「Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒ ｆｏｒ Ｈａｒｄｗａｒｅ Ｅｍｕｌａｔｉｏｎ」という名称の特許文献１、Ｓｃａｌｚｉ他による「Ｐｒｅｐｒｏｃｅｓｓｉｎｇ ｏｆ Ｓｔｏｒｅｄ Ｔａｒｇｅｔ Ｒｏｕｔｉｎｅｓ ｆｏｒ Ｅｍｕｌａｔｉｎｇ Ｉｎｃｏｍｐａｔｉｂｌｅ Ｉｎｓｔｒｕｃｔｉｏｎｓ ｏｎ ａ Ｔａｒｇｅｔ Ｐｒｏｃｅｓｓｏｒ」という名称の特許文献２、Ｄａｖｉｄｉａｎ他による「Ｄｅｃｏｄｉｎｇ Ｇｕｅｓｔ Ｉｎｓｔｒｕｃｔｉｏｎ ｔｏ Ｄｉｒｅｃｔｌｙ Ａｃｃｅｓｓ Ｅｍｕｌａｔｉｏｎ Ｒｏｕｔｉｎｅｓ ｔｈａｔ Ｅｍｕｌａｔｅ ｔｈｅ Ｇｕｅｓｔ Ｉｎｓｔｒｕｃｔｉｏｎｓ」という名称の特許文献３、Ｇｏｒｉｓｈｅｋ他による「Ｓｙｍｍｅｔｒｉｃａｌ Ｍｕｌｔｉｐｒｏｃｅｓｓｉｎｇ Ｂｕｓ ａｎｄ Ｃｈｉｐｓｅｔ Ｕｓｅｄ ｆｏｒ Ｃｏｐｒｏｃｅｓｓｏｒ Ｓｕｐｐｏｒｔ Ａｌｌｏｗｉｎｇ Ｎｏｎ－Ｎａｔｉｖｅ Ｃｏｄｅ ｔｏ Ｒｕｎ ｉｎ ａ Ｓｙｓｔｅｍ」という名称の特許文献４、Ｌｅｔｈｉｎ他による「Ｄｙｎａｍｉｃ Ｏｐｔｉｍｉｚｉｎｇ Ｏｂｊｅｃｔ Ｃｏｄｅ Ｔｒａｎｓｌａｔｏｒ ｆｏｒ Ａｒｃｈｉｔｅｃｔｕｒｅ Ｅｍｕｌａｔｉｏｎ ａｎｄ Ｄｙｎａｍｉｃ Ｏｐｔｉｍｉｚｉｎｇ Ｏｂｊｅｃｔ Ｃｏｄｅ Ｔｒａｎｓｌａｔｉｏｎ Ｍｅｔｈｏｄ」という名称の特許文献５、Ｅｒｉｃ Ｔｒａｕｔによる「Ｍｅｔｈｏｄ ｆｏｒ Ｅｍｕｌａｔｉｎｇ Ｇｕｅｓｔ Ｉｎｓｔｒｕｃｔｉｏｎｓ ｏｎ ａ Ｈｏｓｔ Ｃｏｍｐｕｔｅｒ Ｔｈｒｏｕｇｈ Ｄｙｎａｍｉｃ Ｒｅｃｏｍｐｉｌａｔｉｏｎ ｏｆ Ｈｏｓｔ Ｉｎｓｔｒｕｃｔｉｏｎｓ」という名称の特許文献６及び他の多くを含むがこれらに限定されない、種々のソフトウェア及びハードウェア・エミュレーションの特許は、当業者が利用可能なターゲット・マシンのための異なるマシン用に設計された命令形式のエミュレーションを達成する様々な既知の方法を示す。

【０１７８】

図３２において、ホスト・アーキテクチャのホスト・コンピュータ・システム５０００’をエミュレートする、エミュレートされたホスト・コンピュータ・システム５０９２の一例が提供される。エミュレートされたホスト・コンピュータ・システム５０９２では、ホスト・プロセッサ（ＣＰＵ）５０９１は、エミュレートされたホスト・プロセッサ（又は、仮想ホスト・プロセッサ）であり、且つ、ホスト・コンピュータ５０００’のプロセッサ５０９１のものとは異なるネイティブな命令セット・アーキテクチャを有するエミュレーション・プロセッサ５０９３を含む。エミュレートされたホスト・コンピュータ・システム５０９２は、エミュレーション・プロセッサ５０９３がアクセス可能なメモリ５０９４を有する。例示的な実施形態において、メモリ５０９４は、ホスト・コンピュータ・メモリ５０９６の部分と、エミュレーション・ルーチン５０９７の部分とにパーティション化される。ホスト・コンピュータ・メモリ５０９６は、ホスト・コンピュータ・アーキテクチャに従い、エミュレートされたホスト・コンピュータ・システム５０９２のプログラムに利用可能である。エミュレーション・プロセッサ５０９３は、エミュレートされたプロセッサ５０９１のもの以外のアーキテクチャのアーキテクチャ化された命令セットのネイティブ命令を実行し、このネイティブ命令はエミュレーション・ルーチン・メモリ５０９７から取得されたものであり、且つ、エミュレーション・プロセッサ５０９３は、シーケンス及びアクセス／デコード・ルーチンにおいて取得される１つ又は複数の命令を用いることにより、ホスト・コンピュータ・メモリ５０９６の中のプログラム由来の実行のためのホスト命令にアクセスすることができ、このシーケンス及びアクセス／デコード・ルーチンは、アクセスされたホスト命令をデコードして、アクセスされたホスト命令の機能をエミュレートするためのネイティブ命令実行ルーチンを判断することができる。ホスト・コンピュータ・システム５０００’のアーキテクチャのために定められた、例えば、汎用レジスタ、制御レジスタ、動的アドレス変換、及びＩ／Ｏサブシステムのサポート、並びにプロセッサ・キャッシュといったファシリティを含む他のファシリティを、アーキテクチャ化ファシリティ・ルーチンによってエミュレートすることができる。エミュレーション・ルーチンは、エミュレーション・ルーチンの性能を高めるために、エミュレーション・プロセッサ５０９３において利用可能な（汎用レジスタ、及び仮想アドレスの動的

変換といった）機能を利用することもできる。ホスト・コンピュータ５０００’の機能をエミュレートする際にプロセッサ５０９３を補助するために、専用のハードウェア及びオフ・ロード・エンジンを設けることもできる。

【０１７９】

本明細書で用いられる用語は、特定の実施形態を説明する目的のためのものにすぎず、本発明の限定であることを意図したものではない。本明細書で用いられる場合、単数形「１つの（a）」、「１つの（an）」及び「その（the）」は、文脈が特に明示しない限り、複数形も同様に含むことを意図したものである。「含む（comprise）」及び／又は「含んでいる（comprising）」という用語は、本明細書で用いられる場合、記述された特徴、整数、ステップ、動作、要素、及び／又はコンポーネントの存在を指示するが、１つ又は複数の他の特徴、整数、ステップ、動作、要素、コンポーネント、及び／又はそれらのグループの存在又は追加を排除するものではないこともさらに理解されるであろう。

【０１８０】

以下の特許請求の範囲に存在する場合、「手段又はステップと機能との組み合わせ（ミーンズ又はステップ・プラス・ファンクション）」要素の対応する構造、材料、動作及び均等物は、明確に特許請求された他の請求要素と共に機能を実行するための任意の構造体、材料、又は行為を含むことを意図したものである。１つ又は複数の態様の説明は、例証及び説明のためだけに提示されたものであり、網羅的であること又は本発明を開示した形態に限定することを意図したものではない。１つ又は複数の態様の範囲から逸脱することなく、当業者には、多くの修正及び変形が明らかとなるであろう。実施形態は、１つ又は複数の態様の原理及び実際の用途を最もよく説明するため、及び、当業者が、企図した特定の用途に適するように種々の修正を有する種々の実施形態に関して１つ又は複数の態様を理解することができるように、選択され記述された。

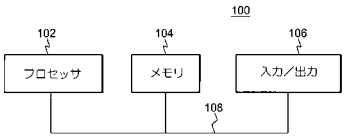
【符号の説明】

【０１８１】

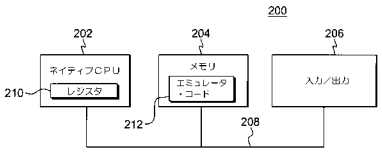
１００、２００：コンピューティング環境
１０２、５０２６：プロセッサ
１０４、２０４、５０２５：メモリ
１０６、２０６：入力／出力デバイス及び／又はインターフェース
１０８、２０８：バス
２０２：ネイティブ中央演算処理ユニット（ＣＰＵ）
２１０：ネイティブ・レジスタ
２１２：エミュレータ・コード
２５０：ゲスト命令
２５２：命令フェッチ・ユニット
２５４：命令変換ルーチン
２５６：ネイティブ命令
２６０：エミュレーション制御ルーチン
３００：レジスタ・ファイル
３０２、４８０：ベクトル・レジスタ
３０４：浮動小数点レジスタ
４００：Ｖｅｃｔｏｒ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｔｅｓｔ　Ｄａｔａ　Ｃｌａｓｓ　Ｉｍｍｅｄｉａｔｅ命令
５００：Ｖｅｃｔｏｒ　Ｃｈｅｃｋｓｕｍ命令
６００：Ｖｅｃｔｏｒ　Ｇａｌｏｉｓ　Ｆｉｅｌｄ　Ｍｕｌｔｉｐｌｙ　Ｓｕｍ　ａｎｄ
７００：Ｖｅｃｔｏｒ　Ｇｅｎｅｒａｔｅ　Ｍａｓｋ命令
８００：Ｖｅｃｔｏｒ　Ｅｌｅｍｅｎｔ　Ｒｏｔａｔｅ　ａｎｄ　Ｉｎｓｅｒｔ　Ｕｎｄｅｒ　Ｍａｓｋ命令
９００：ベクトル例外コード
９０２：ベクトル・インデックス

９０４：ベクトル割り込みコード
１０００：コンピュータ・プログラム製品

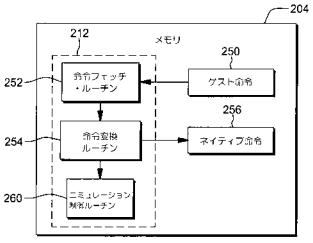
【図１】



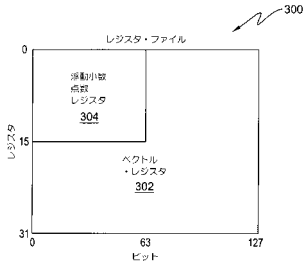
【図２】



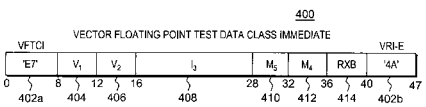
【図３】



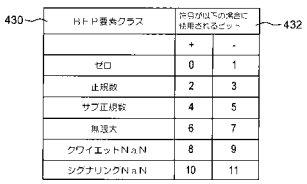
【図４】



【図５】

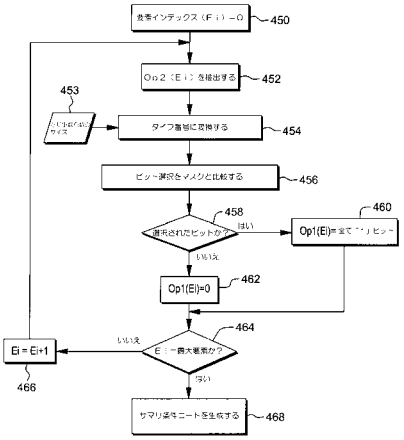


【図６】

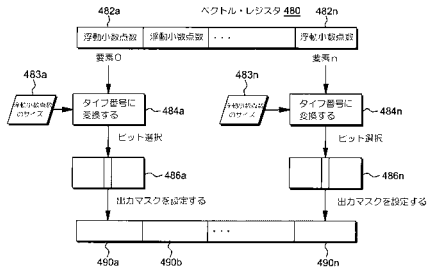


BFP要素クラス	符号が以下の場合に使用されるビット	
	+	-
ゼロ	0	1
正規数	2	3
サブ正規数	4	5
無限大	6	7
クワイエットNaN	8	9
シグナリングNaN	10	11

【図７】



【図８】

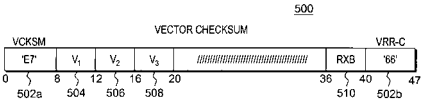


【図９】

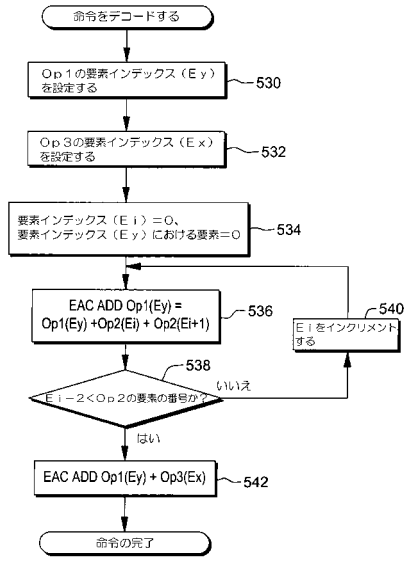
エンティティ	符号	バイアス指数	単位ビット*	仮数
負のゼロ	±	0	0	0
非正規化数	±	0**	0	非０
正規化数	±	非０、全て１ではない	1	任意
無限大	±	全て１	-	0
クワイエットＮａＮ	±	全て１	-	Ｆ０=1, Ｆ≠ 任意
シグナリングＮａＮ	±	全て１	-	Ｆ０=0, Ｆ≠0

説明：
* 単位ビットは論理指定される
** バイアス指数は、これが値１を有するかのように数学的に処理される
NaN 非数
F0 仮数の左側ビット
F- 仮数の残りのビット

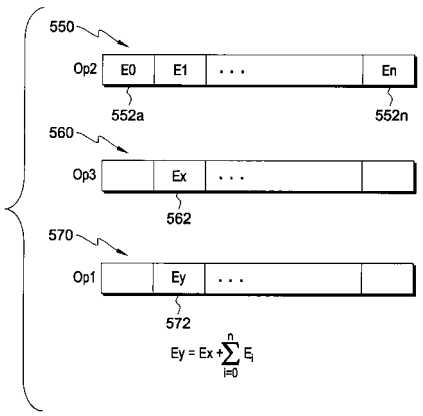
【図１０】



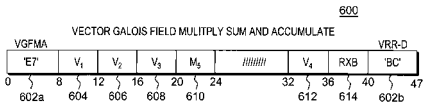
【図１１】



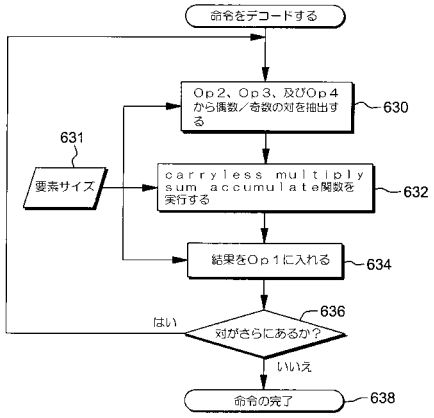
【図１２】



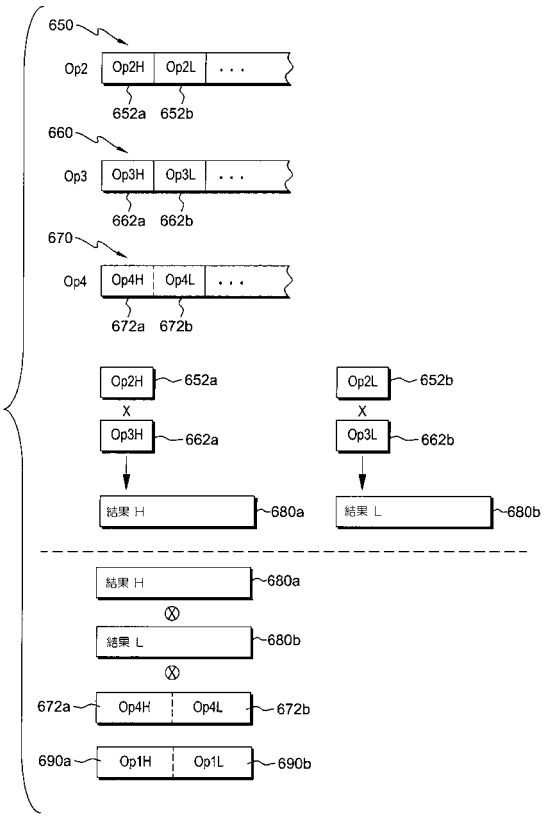
【図１３】



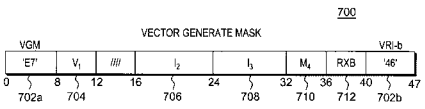
【図１４】



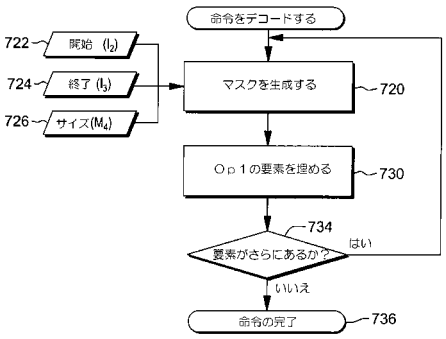
【図１５】



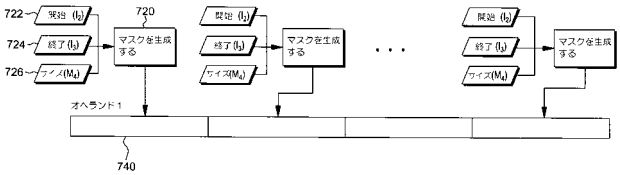
【図１６】



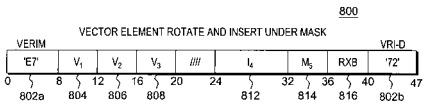
【図１７】



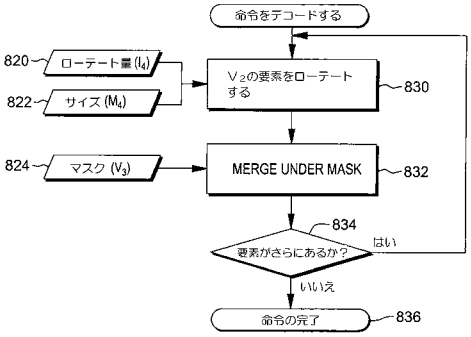
【図１８】



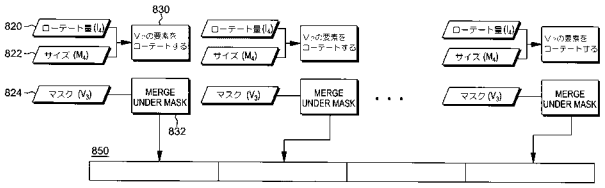
【図１９】



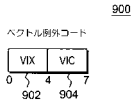
【図２０】



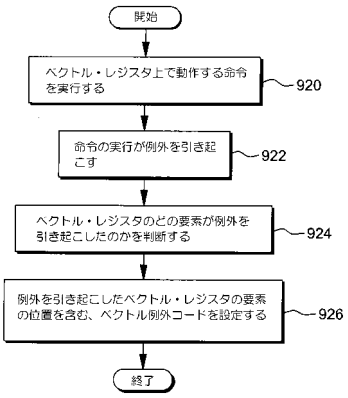
【図２１】



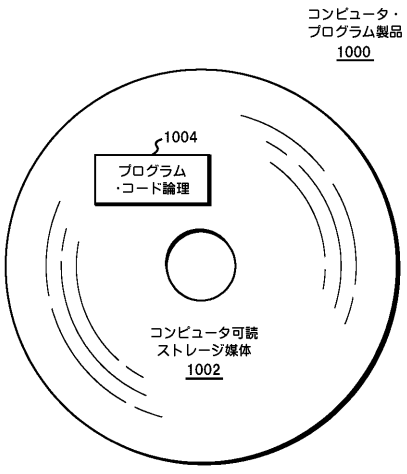
【図２２】



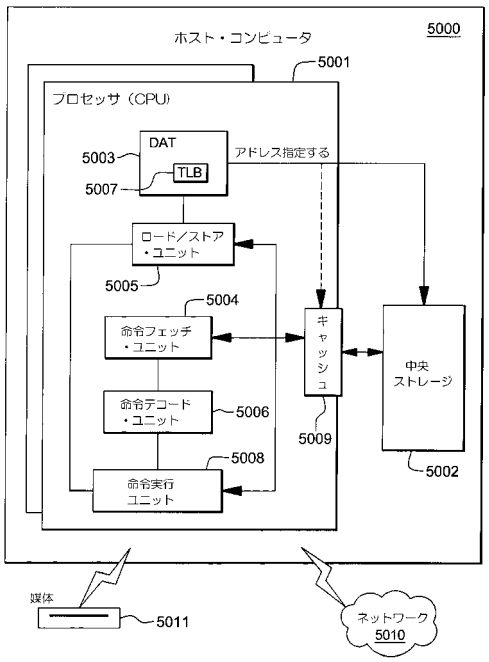
【図２３】



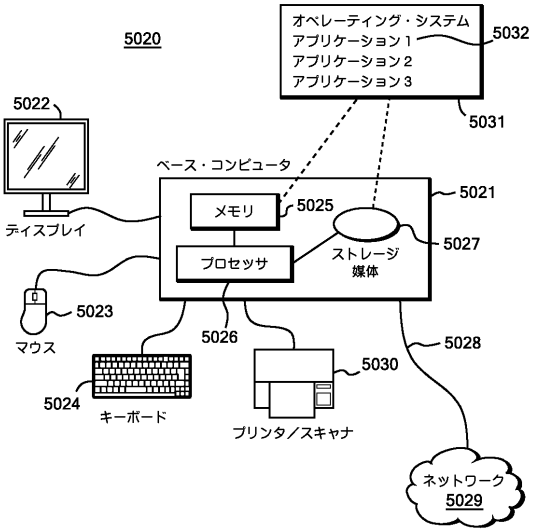
【図２４】



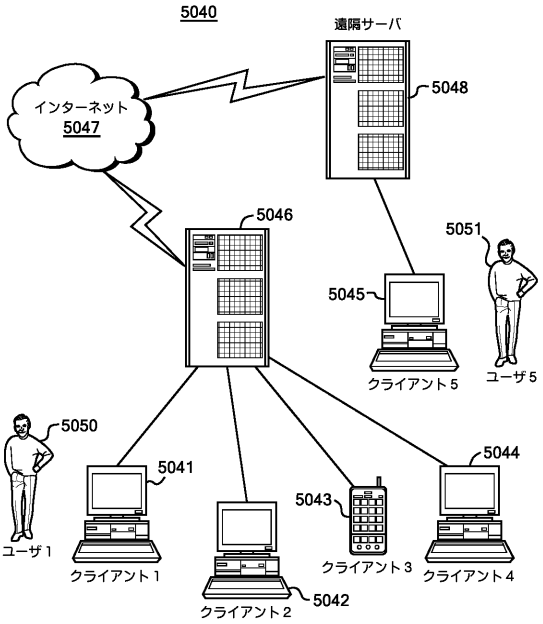
【図２５】



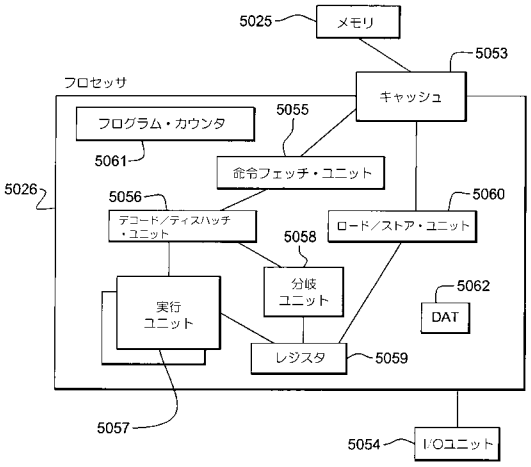
【図２６】



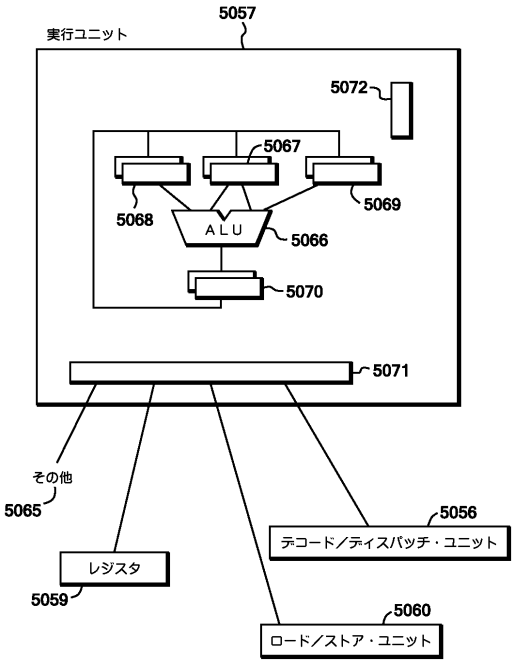
【図２７】



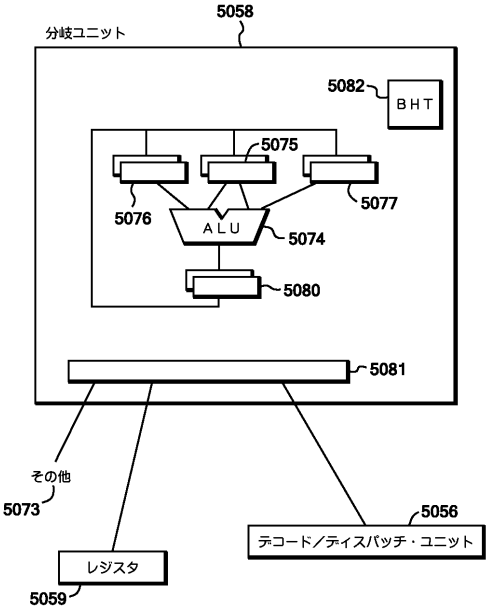
【図２８】



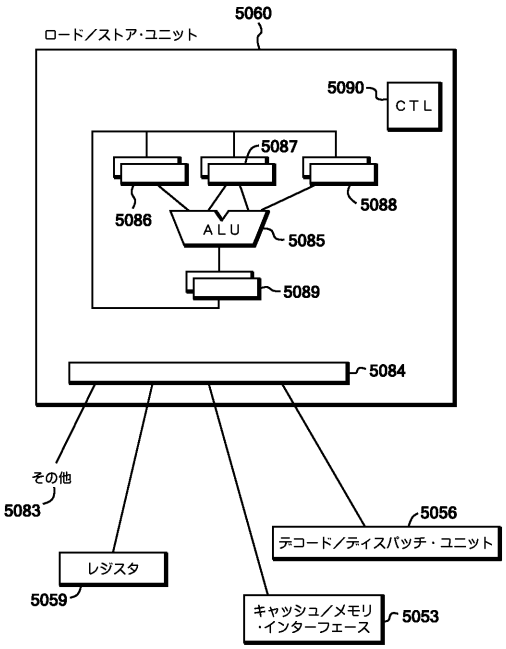
【図２９】



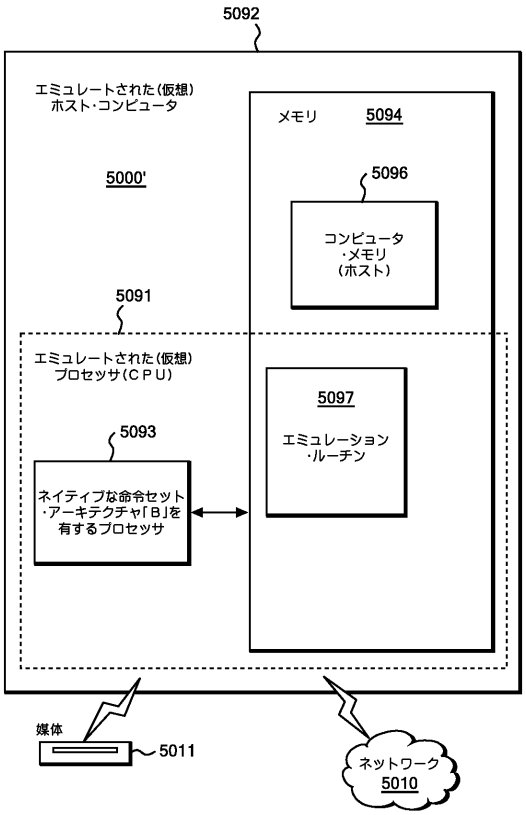
【図３０】


5058
分岐ユニット
5082
BHT
5075
5076
ALU
5077
5074
5080
5081
その他
5073
5056
デコード/ディスパッチ・ユニット
レジスタ
5059

【図３１】


5060
ロード/ストア・ユニット
5090
CTL
5087
5086
ALU
5088
5085
5089
5084
その他
5083
5056
デコード/ディスパッチ・ユニット
レジスタ
5059
キャッシュ/メモリ
・インターフェース
5053

【図３２】


5092
エミュレートされた(仮想)
ホスト・コンピュータ
5000'
メモリ
5094
5096
コンピュータ
・メモリ
(ホスト)
5091
エミュレートされた(仮想)
プロセッサ(CPU)
5097
エミュレーション
・ルーチン
5093
ネイティブな命令セット
・アーキテクチャ「B」を
有するプロセッサ
媒体
5011
ネットワーク
5010

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2013/060309

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. G06F9/305(2006.01)i, G06F9/315(2006.01)i, G06F15/80(2006.01)i, G06F17/16(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.Cl. G06F9/305, G06F9/315, G06F15/80, G06F17/16

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2014
Registered utility model specifications of Japan 1996-2014
Published registered utility model applications of Japan 1994-2014

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4569016 A (Hsieh T. HAO et al.) 1986.02.14, column11,line1 - column13,line42, Table 2(a), FIG.3 & JP 60-14336 A	1-6, 9, 11-14, 16, 18-19
Y		8, 15
A		7, 10, 17, 20
Y	US 2012/0265967 A1 (Michael Karl GSCHWIND et al.) 2012.10.18, paragraph77 - 134, FIG.1A - 1C (family none)	8 , 15
A	WO 2006/121444 A1 (TELAIRITY SEMICONDUCTOR INC.) 2006.11.16, paragraph68 (family none)	1-20

☑ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27.02.2014	11.03.2014

Name and mailing address of the ISA/JP	Authorized officer
Japan Patent Office 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	Toshio MISAKA 5B 4178 Telephone No. +81-3-3581-1101 Ext. 3545

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2013/060309

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6760837 B1 (Gillbert LAURENTI et al.) 2004.07.06, column12,line5 - line31, FIG.10B & JP 2000-284960 A	1-20
A	US 2008/0077643 A1 (Kenichi HANDA) 2008.03.27, paragraph44 - 54, FIG.1, FIG.6 - 9 & JP 2008-83795 A	1-20

Form PCT/ISA/210 (continuation of second sheet) (July 2009)

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CZ,DE,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IR,IS,JP,KE,KG,KN,KP,KR,KZ,LA,LC,LK,LR,LS,LT,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,SM,ST,SV,SY,TH,TJ,TM,TN,TR,TT,TZ,UA,UG,US

（特許庁注：以下のものは登録商標）
１．ＪＡＶＡ
２．イーサネット

(74)代理人 100112690
弁理士 太佐 種一
(74)復代理人 110000316
特許業務法人ピー・エス・ディ
(72)発明者 ブラッドベリ、ジョナサン、デービッド
アメリカ合衆国 １２６０１－５４００ ニューヨーク州 ポキプシー サウス・ロード ２４５５
(72)発明者 スレゲル、ティモシー
アメリカ合衆国 １２６０１－５４００ ニューヨーク州 ポキプシー サウス・ロード ２４５５
(72)発明者 シュワルツ、エリック、マーク
アメリカ合衆国 １２６０１－５４００ ニューヨーク州 ポキプシー サウス・ロード ２４５５
(72)発明者 エネンケル、ロバート、フレデリック
カナダ国 Ｌ６Ｇ １Ｃ７ オンタリオ州 マーカム ワーデン・アベニュー ８２００

Ｆターム(参考) 5B033 BD03 CA12 CA18
5B056 BB35 FF08 FF10 FF11