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

特許第5575477号
(P5575477)

(45)発行日　平成26年8月20日(2014.8.20)　(24)登録日　平成26年7月11日(2014.7.11)

(51)Int.Cl.　　F I
G06F 9/305 (2006.01)　　G06F 9/30 340C
G06F 15/80 (2006.01)　　G06F 15/80

請求項の数 43　(全 28 頁)

(21)出願番号　特願2009-529420 (P2009-529420)
(86)(22)出願日　平成19年9月21日 (2007.9.21)
(65)公表番号　特表2010-504594 (P2010-504594A)
(43)公表日　平成22年2月12日 (2010.2.12)
(86)国際出願番号　PCT/US2007/079234
(87)国際公開番号　WO2008/036945
(87)国際公開日　平成20年3月27日 (2008.3.27)
審査請求日　平成21年3月23日 (2009.3.23)
(31)優先権主張番号　11/525,981
(32)優先日　平成18年9月22日 (2006.9.22)
(33)優先権主張国　米国(US)

前置審査

(73)特許権者　593096712
インテル　コーポレイション
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　ブールバード　２２００
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(72)発明者　ジュリア，マイケル，エー.
アメリカ合衆国　９７１２３　オレゴン州　ヒルズボロ　サウスウェスト　３２１スト　プレイス　１６７３０

最終頁に続く

(54)【発明の名称】文字列を処理するための命令及び論理回路

(57)【特許請求の範囲】
【請求項１】
命令を記憶した機械読み取り可能媒体であって、前記命令は、機械により実行されると、前記機械に
第１のパック化オペランドの有効なデータ要素のすべてと、第２のパック化オペランドの有効なデータ要素のすべてとの全組み合わせについて比較する段階であって、前記比較の第１の結果は、前記第１のオペランドの有効データ要素と前記第２のオペランドの有効データ要素との間の比較のみの比較結果を含む段階と、
前記比較の第１の結果を記憶する段階と
を含む方法を実行させる、媒体。
【請求項２】
前記第１の結果は前記データ要素のいずれかが等しいかどうか示す、請求項１に記載の機械読み取り可能媒体。
【請求項３】
前記第１の結果は、前記第１のオペランドに示された一範囲のデータ要素について、その範囲に前記第２のオペランドのデータ要素があるか示す、請求項１に記載の機械読み取り可能媒体。
【請求項４】
前記第１の結果は前記第１のオペランドの各データ要素が、前記第２のオペランドの各データ要素と等しいかどうか示す、請求項１に記載の機械読み取り可能媒体。

【請求項５】
前記第１の結果は前記第１のオペランドのデータ要素の一部の順序が、前記第２のオペランドのデータ要素の一部の順序と等しいかどうか示す、請求項１に記載の機械読み取り可能媒体。
【請求項６】
前記第１の結果の一部をネゲートする、請求項１に記載の機械読み取り可能媒体。
【請求項７】
前記第１の結果は、マスク値またはインデックス値のいずれかにより表される、請求項１に記載の機械読み取り可能媒体。
【請求項８】
第１の単一命令複数データ（ＳＩＭＤ）オペランドの有効なデータ要素のすべてと、第２のＳＩＭＤオペランドの有効なデータ要素のすべてとの全組み合わせについて比較し、ＳＩＭＤオペランド比較結果を生成する比較ロジックと、
前記比較ロジックを制御して前記ＳＩＭＤオペランド比較結果に関するインジケータを記憶する第１の制御信号とを有する装置。
【請求項９】
前記第１と第２のオペランドのデータ要素の有効性を明示的に示す、請求項８に記載の装置。
【請求項１０】
前記第１と第２のオペランドのデータ要素の有効性を黙示的に示す、請求項８に記載の装置。
【請求項１１】
前記第１の制御信号は、前記比較ロジックが符号付きまたは符号無しの値を比較するかどうか示す符号制御信号を含む、請求項８に記載の装置。
【請求項１２】
前記第１の制御信号は、どれかが等しい、範囲内にある、及びそれぞれ等しいよりなるリストから選択した集約機能を前記比較ロジックが実行するかどうか示す集約機能信号を含む、請求項１１に記載の装置。
【請求項１３】
前記第１の制御信号は、ネゲート信号を含み、前記比較ロジックに前記比較の結果の少なくとも一部をネゲートさせる、請求項１２に記載の装置。
【請求項１４】
前記第１の制御信号は、前記比較ロジックが前記比較の結果のＭＳＢまたはＬＳＢのインデックスを生成するかどうか示すインデックス信号を含む、請求項１３に記載の装置。
【請求項１５】
前記第１の制御信号は、前記比較ロジックが前記比較の結果としてゼロ延長マスクまたはバイト又はワードへの拡張マスクを生成するかどうかを示すマスク信号を含む、請求項１４に記載の装置。
【請求項１６】
前記第１の制御信号は、複数のビットを記憶する制御フィールドである、請求項１５に記載の装置。
【請求項１７】
単一命令複数データ（ＳＩＭＤ）比較命令を記憶する第１のメモリと、
前記ＳＩＭＤ比較命令を実行して、第１のオペランドの有効なデータ要素のすべてと、第２のオペランドの有効なデータ要素のすべてとの全組み合わせについて比較して、ＳＩＭＤオペランド比較結果を生成し、前記第１のオペランドと前記第２のオペランドとは前記ＳＩＭＤ比較命令で示され、前記ＳＩＭＤオペランド比較結果に関するインジケータを記憶するプロセッサと
を有する、システム。
【請求項１８】

前記第１のオペランドを、第１のレジスタのアドレスにより前記命令内に示す、請求項１７に記載のシステム。
【請求項１９】
前記第２のオペランドを、メモリアドレスまたは第２のレジスタにより前記命令内に示す、請求項１８に記載のシステム。
【請求項２０】
前記命令は前記プロセッサに対する制御信号を示すイミーディエイトフィールドを含む、請求項１９に記載のシステム。
【請求項２１】
イミーディエイトフィールドは、前記オペランドが符号付きバイト、符号無しバイト、符号付きワード、または符号無しワードを含むかどうかを示す、請求項２０に記載のシステム。
【請求項２２】
前記イミーディエイトフィールドは集約機能を前記プロセッサが実行することを示す、請求項２１に記載のシステム。
【請求項２３】
前記イミーディエイトフィールドは、マスクまたはインデックスを前記命令の実行に応じて生成するかどうかを示す、請求項２２に記載のシステム。
【請求項２４】
前記命令は、前記第１及び第２のオペランドの明示的に有効データ要素のみを比較させる、請求項１７に記載のシステム。
【請求項２５】
前記命令は、前記第１及び第２のオペランドの黙示的に有効データ要素のみを比較させる、請求項１７に記載のシステム。
【請求項２６】
第１のテキストストリングに対応する第１のパック化オペランドを記憶する第１の記憶領域と、
第２のテキストストリングに対応する第２のパック化オペランドを記憶する第２の記憶領域と、
前記第１のパック化オペランドのすべての有効データ要素を、前記第２のパック化オペランドのすべての有効データ要素と比較する比較ロジックと、
第１のパック化オペランドの有効データ要素と第２のパック化オペランドの有効データ要素との間の比較のみの比較結果を含む、前記比較ロジックが実行した前記比較の結果アレイを記憶する第３の記憶領域とを有するプロセッサ。
【請求項２７】
前記比較ロジックは値の２次元のアレイを生成し、前記アレイのエントリーは前記第１のパック化オペランドの有効データ要素と前記第２のパック化オペランドの有効データ要素との間の比較に対応する、請求項２６に記載のプロセッサ。
【請求項２８】
前記比較ロジックは、前記値の２次元のアレイに、いずれかが等しい、範囲内にある、各々が等しい、及び非連続的サブストリングがマッチするよりなる集約機能の１つを実行する、請求項２７に記載のプロセッサ。
【請求項２９】
前記結果アレイは、マスク値またはインデックス値のいずれかにより表される、請求項２８に記載のプロセッサ。
【請求項３０】
ＳＩＭＤ比較命令をデコードするデコーダと、
整数レジスタと浮動小数点レジスタとを含むレジスタファイルと、
前記レジスタを前記レジスタファイルにリネームするレジスタリネーミングであって、前記レジスタファイルは第１のパック化オペランドを格納する第１のソースレジスタと第

２のパック化オペランドを格納する第２のソースレジスタとを有するレジスタリネーミングと、
前記デコーダからのマイクロ演算を保持するキューと、
前記ソースレジスタ中の依存したオペランドの準備状況に基づき、前記オペランドの実行をスケジューリングするスケジューラと、
前記第１のパック化オペランドの有効なデータ要素のすべてと、前記第２のパック化オペランドの有効なデータ要素のすべてとの全組み合わせについて比較する実行ユニットであって、前記実行ユニットの比較結果は有効な第１のオペランドのデータ要素と有効な第２のオペランドのデータ要素との間の比較のみの比較結果を含む、前記レジスタファイルに結合した実行ユニットと、
前記実行ユニットの比較結果を記憶するデスティネーションレジスタとを有するプロセッサ。
【請求項３１】
前記実行ユニットは第１の制御信号により制御される、請求項３０に記載のプロセッサ。
【請求項３２】
前記第１と第２のオペランドのデータ要素の有効性を明示的に示す、請求項３１に記載のプロセッサ。
【請求項３３】
前記第１と第２のオペランドのデータ要素の有効性を黙示的に示す、請求項３１に記載の装置。
【請求項３４】
前記第１の制御信号は、前記実行ユニットが符号付きの値または符号無しの値を比較するかどうか示す符号制御信号を含む、請求項３１に記載のプロセッサ。
【請求項３５】
前記第１の制御信号は、どれかが等しい、範囲内にある、及びそれぞれ等しいよりなるリストから選択した集約機能を前記実行ユニットが実行するかどうか示す集約機能信号を含む、請求項３１に記載のプロセッサ。
【請求項３６】
前記第１の制御信号は、前記実行ユニットに前記比較結果の少なくとも一部をネゲートさせるネゲート信号を含む、請求項３１に記載のプロセッサ。
【請求項３７】
前記第１の制御信号は、前記実行ユニットが前記比較結果のＭＳＢまたはＬＳＢのインデックスを生成するかどうか示すインデックス信号を含む、請求項３１に記載のプロセッサ。
【請求項３８】
前記第１の制御信号は、前記比較ロジックが前記比較結果としてゼロ延長マスクまたはバイト又はワードへの拡張マスクを生成するかどうかを示すマスク信号を含む、請求項３１に記載のプロセッサ。
【請求項３９】
前記実行ユニットは、整数演算と浮動小数点演算とを実行するロジックを有する、請求項３０に記載のプロセッサ。
【請求項４０】
前記スケジューラは、高速スケジューラ、低速／汎用浮動点スケジューラ、単純浮動小数点スケジューラのうち１つ以上を有する、請求項３０に記載のプロセッサ。
【請求項４１】
ＳＩＭＤ比較命令をデコードするデコーダと、
整数レジスタと浮動小数点レジスタとを含む複数のレジスタを格納したレジスタファイルと、
前記レジスタを前記レジスタファイルにリネームするレジスタリネーミングであって、

前記レジスタファイルは第１のパック化オペランドを格納する第１のソースレジスタと第２のパック化オペランドを格納する第２のソースレジスタとを有するレジスタリネーミングと、
前記デコーダからのマイクロ命令を格納するキューと、
前記ソースレジスタ中の依存したオペランドの準備状況に基づき、前記オペランドの実行をスケジューリングするスケジューラと、
前記第１のパック化オペランドの有効なデータ要素のすべてと、前記第２のパック化オペランドの有効なデータ要素のすべてとの全組み合わせについて比較する、前記レジスタファイルに結合した実行ユニットであって、前記第１と第２のパック化オペランドのデータ要素は８ビット、１６ビット、または３２ビットである、実行ユニットと、
前記第１と第２のパック化オペランドの対応するデータ要素が等しいと前記実行ユニットが判断すると、すべてが１である結果配列を格納するデスティネーションレジスタとを有する、プロセッサ。
【請求項４２】
前記実行ユニットは値の２次元のアレイを生成し、前記アレイのエントリーは前記第１のパック化オペランドの有効データ要素と前記第２のパック化オペランドの有効データ要素との間の比較に対応する、請求項４１に記載のプロセッサ。
【請求項４３】
前記実行ユニットは、前記値の２次元のアレイに、いずれかが等しい、範囲内にある、各々が等しい、及び非連続的サブストリングがマッチするよりなる集約機能の１つを実行する、請求項４２に記載のプロセッサ。
【発明の詳細な説明】
【技術分野】
【０００１】
本願開示は、論理的及び数学的演算を行う、処理装置並びに関連するソフトウェア及びソフトウェア列の分野に関する。
【背景技術】
【０００２】
計算機システムは、我々の社会でますます普及している。計算機の処理能力により、広範な職業で働く人々の効率と生産性が向上している。計算機を買って所有する費用は落ち続けている。従って、より多くの消費者が、より新しく、より速い計算機を活用できるようになっている。更に、多くの人は、ノート型の計算機を、その自由度ゆえに、楽しんで利用している。可搬型計算機により、利用者は、職場を離れても旅行中でも、簡単にデータを持ち運ぶことができ、仕事もできる。このような場面は、営業職や管理職にとって、また学生にとってすらも、ありふれた光景である。
【０００３】
処理装置の技術が進むにつれ、より新しいソフトウェアも開発が進んでいる。このソフトウェアは、進んだ処理装置を持つ計算機で走る。利用者は概して、自分の計算機に、より高い性能を期待し要求する。このことは、使うソフトウェアの種類には無関係である。このような性能に関する問題が起こりうるのは、処理装置の内部で実際に実行される命令及び演算の種類からである。ある種類の演算は、完了するのに、他の演算よりも時間がかかる。その理由は、演算が複雑であるせいか、演算に必要な回路の型のせいか、その両方のせいである。このことが、ある種の複雑な演算を処理装置の内部で実行する方法を、最適化する動機である。
【０００４】
通信の応用が、１０年以上に渡って、超小型処理装置の進歩を駆り立ててきた。実際、計算と通信の間の境界線は、ますますぼやけてきている。この理由は、部分的には、通信の応用で文字列を使うからである。文字列の応用は、一般消費者向け市場で広まっている。また、文字列の応用は、多数の装置で広まっている。装置とは例えば携帯電話から個人用計算機までである。このような装置は、文字列情報を、一層より高速に処理することを

求めている。文字列を通信する装置は、計算し通信する装置に進化し続けている。計算し通信する装置は、次のような形の応用を行う。即ち、マイクロソフト（登録商標）インスタントメッセンジャー（商標）、電子メールの応用（例えばマイクロソフト（登録商標）アウトルック（商標））、及び携帯電話メールの応用である。その結果、将来における、個人の計算及び通信の体験は、文字列を扱う能力について、更により豊かになると期待される。

【０００５】

従って、計算又は通信する装置同士の間で交換される文字列情報を、処理すること又は構文解析することは、現在の計算装置及び通信装置にとって、一段と重要性を増している。とりわけ、通信又は計算する装置が文字情報の列を解釈することは、文字列データに対して行う最も重要な演算のうちの、いくつかを含む。このような演算では、計算量が嵩むにしても、データの並列度は高い水準であってもよい。この並列度を利用して、様々なデータ格納装置を使う効率的な実装を行える。格納装置とは、例えば、単一命令複数データ（ＳＩＭＤ）型のレジスタである。数多くの現在の計算機アーキテクチャはまた、次のことを要求する。即ち、複数の演算、複数の命令、又は複数の下位命令（よく「マイクロ命令」又は「μｏｐ」という。）を使って、様々な論理的及び数学的演算を、多数の演算対象に対して行う。このことにより、処理速度を上げ、その論理的及び数学的演算を行うのに必要なクロック周期の数を減らす。

【発明の概要】

【発明が解決しようとする課題】

【０００６】

例えば、多数の命令から成る命令列が、次のことを行うために必要であってもよい。即ち、文字列の中の特定の語を解釈するのに必要な１つ以上の演算である。この演算は、処理装置、システム、又は計算機プログラムの内部の様々なデータ型が表現する、２つ以上の文字列語を比べることを含む。しかし、このような従来の技術では、多数の処理周期が必要になることがあり、処理装置又はシステムは、結果を得るために、不要な電力を消費してしまうことがある。更に、いくつかの従来技術では、演算の対象としてもよいデータ型として、限られたものしか使えないことがある。

【課題を解決するための手段】

【０００７】

本発明の一態様によると、命令を記憶した機械読み取り可能媒体が提供される。前記命令は、機械により実行されると、前記機械に第１のパック化オペランドの各データ要素を、第２のパック化オペランドの各データ要素と比較する段階と、前記比較の第１の結果を記憶する段階を含む方法を実行させる。

【図面の簡単な説明】

【０００８】

【図１Ａ】計算機システムの区画図である。計算機システムは、処理装置を含む。処理装置は、実行部を含む。実行部は、命令を実行する。命令は、文字列比較演算を行う。この命令は、本願発明の１つの実施例による。

【図１Ｂ】本願発明の別の実施例による、別の例の計算機システムの区画図である。

【図１Ｃ】本願発明の更に別の実施例による、更に別の例の計算機システムの区画図である。

【図２】１つの実施例による処理装置のマイクロアーキテクチャの区画図である。この実施例は、論理回路を含む。この論理回路は、本願発明による文字列比較演算を１つ以上行う。

【図３Ａ】本願発明の１つの実施例による、マルチメディアレジスタにおける種々のパック化データ型の表現を示す。

【図３Ｂ】別の実施例による、パック化データ型を示す。

【図３Ｃ】本願発明の１つの実施例による、マルチメディアレジスタにおける種々の符号付き及び符号無しのパック化データ型の表現を示す。

【図３Ｄ】演算の符号化（即ち命令符号）の形式の１つの実施例を示す。
【図３Ｅ】演算の符号化（即ち命令符号）の別な形式を示す。
【図３Ｆ】演算の符号化の更に別な形式を示す。
【図４】論理回路の区画図である。この論理回路は、本願発明の１つの実施例により、少なくとも１つの文字列比較演算を、１つ以上の単精度パック化データ演算対象に対して行う。
【図５】配列の区画図である。この配列を使って、１つの実施例による少なくとも１つの文字列比較演算を行ってもよい。
【図６】本発明の１つの実施例で行ってもよい演算を示す。
【発明を実施するための形態】
【０００９】
本願発明を実施例を使って説明する。本願発明は、実施例及び添付の図面によっては、限定されない。
【実施例】
【００１０】
以下の記載が記述するのは、技法の実施例である。この技法は、処理装置、計算機システム、又はソフトウェアプログラムの内部で、文字列の要素同士の間を比べる演算を行う。以下の記載では、多数の個別の詳細を記述する。詳細とは例えば処理装置の型、マイクロアーキテクチャの事情、事象、実施可能な機構、等である。詳細を記載する目的は、本願発明のより深い理解を与えるためである。しかし、当業者は次の点に注意。即ち、本発明を、そのような個別の詳細を抜きに実施してもよい。加えて、いくつかの周知の構造、回路などは、詳細を示していない。これは、本願発明を不要に複雑に示すのを避けるためである。
【００１１】
以下の実施例を、処理装置を参照して記述する。しかし、他の実施例を、他の型の集積回路や論理部品に応用できる。本願発明と同じ技術及び教示を、他の型の回路又は半導体部品に容易に応用できる。他の型の回路又は半導体部品も、より高いパイプライン効率及び改善した性能から、利益を受けることができる。本願発明の教示は、データの演算を行う、いかなる処理装置又は機械にも、応用できる。なお、本願発明は、２５６ビット、１２８ビット、６４ビット、３２ビット、又は１６ビットのデータの演算を行う処理装置又は機械に限定されない。本願発明を、パック化データを演算する必要がある、いかなる処理装置及び機械にも、応用できる。
【００１２】
以下の記載では、説明のために、多数の個別の詳細を記述する。詳細を記載する目的は、本願発明の徹底的な理解を与えるためである。しかし、当業者は次の点を理解することになる。即ち、これらの個別の詳細は、本願発明を実施するために必要ではない。場合により、周知の電気的な構造及び回路については、特に詳しくは記載していない。これは、本願発明を不要に複雑に示すのを避けるためである。加えて、以下の記載は、例を示す。添付の図面は、様々な例を示す。これらの例を示すのは、説明のためである。しかし、これらの例を、本願発明を限定する意味で解釈してはならない。これらの例は、本願発明の例を示すことを、意図しているだけである。これらの例は、本願発明の全ての可能な実装を網羅する一覧を示すことを、意図していない。
【００１３】
以下の例では、命令の取り扱い及び分散を、実行部及び論理回路の文脈で記述する。しかし、本願発明の他の実施例を、ソフトウェアによっても実現できる。１つの実施例では、本願発明の方法を、機械が実行可能な命令に実施する。この命令を使って、次のことを行える。即ち、汎用処理装置又は専用処理装置をこの命令によってプログラムし、本願発明の工程を実行させる。本願発明を、計算機プログラム又はソフトウェアとして提供してもよい。この計算機プログラム又はソフトウェアは、機械可読媒体又は計算機可読媒体を含んでもよい。機械可読媒体又は計算機可読媒体は、命令を内部に格納して持つ。この命

令を使って、計算機（又は他の電子装置）をプログラムしてもよい。このプログラムにより、本願発明による処理を行う。代わりに、本願発明の工程を、特定のハードウェア部品によって実行してもよい。特定のハードウェア部品は、本願発明の工程を実行するための、配線を固定した論理回路を含む。又は、本願発明の工程を、プログラムされた計算機部品と専用ハードウェア部品との、いかなる組み合わせによっても実行してもよい。このようなソフトウェアを、システムの記憶装置の内部に格納できる。同様に、命令を分散できる。この分散を、網により行う。又は、この分散を、他の計算機可読媒体を使って行う。

【００１４】

従って、機械可読媒体は、機械（例えば計算機）が読める形式で情報を格納又は伝達するための、いかなる機構を含んでもよい。機械可読媒体は、次のものを含むが、これらに限定されない：フロッピー（登録商標）ディスケット；光学ディスク；コンパクトディスク；ＣＤ－ＲＯＭ；光磁気ディスク；ＲＯＭ；ＲＡＭ；ＥＰＲＯＭ；ＥＥＰＲＯＭ；磁気カード若しくは光学カード；フラッシュ記憶装置；インターネット上の伝送；電気的、光学的、音響的、若しくは他の形態の伝搬する信号（例えば搬送波、赤外線信号、デジタル信号、等）；又は、同様のもの。従って、計算機可読媒体は、機械（例えば計算機）が読める形式で、電子的な命令又は情報を、格納又は伝達するのに適した、いかなる型の媒体及び機械可読媒体をも含む。更に、本願発明を、計算機プログラムとしてダウンロードしてもよい。即ち、プログラムを、遠隔の計算機（例えばサーバー）から転送して、要求する計算機（例えばクライアント）に取り込んでもよい。プログラムの転送を、次の信号によって行ってもよい。即ち、電気的、光学的、音響的、又は他の形態のデータ信号。これらの信号を、搬送波又は他の伝搬媒体に実施する。これらの信号は、通信接続（例えばモデム接続、網接続等）を経由する。

【００１５】

設計は、様々な段階を踏んでもよい。即ち、設計は、創案からシミュレーションを経て製造に至る。設計を表現するデータは、その設計を多数の方法で表現してもよい。まず、シミュレーションで便利なのは、次の方法である。即ち、ハードウェアを、ハードウェア記述言語又は別の機能記述言語を使って表現してもよい。加えて、論理の及び／又はトランジスターのゲート水準の回路モデルを、設計の過程の何らかの段階で作ってもよい。更に、ほとんどの設計者は、何らかの段階で、ハードウェアモデルにおける、種々の素子の物理的な配置を表現する水準のデータに辿り着く。従来の半導体の製造技術を使う場合には、このハードウェアモデルを表現するデータは、半導体マスクの様々な層に種々の特徴が有るか無いかを指定するデータであってもよい。このマスクを使って集積回路を作る。設計におけるいかなる表現でも、そのデータをいかなる形態の機械可読媒体に格納してもよい。機械可読媒体とは、次のものでもよい。即ち、そのような情報を伝送するために、変調した若しくは他の方法で生成した、光学的若しくは電気的な波、記憶装置、又は磁気若しくは光学的な格納器（例えば円盤）。これらの媒体のいかなるものも、設計又はソフトウェア情報を「担う」又は「示す」のでもよい。符号又は設計を示す又は担う電気的な搬送波を伝送する場合に、その電気信号の複写、蓄積、又は再送を行うと、新しい複写ができる。従って、通信設備者又は網提供者は、本願発明の技術を実施する物（即ち搬送波）の複写を作ってもよい。

【００１６】

近年の処理装置では、多数の異なる実行部を使って、様々な命令を処理し実行する。全ての命令が平等に作られている訳では無い。即ち、ある命令は他の命令よりも早く完了する。別の命令は、完了するのに莫大なクロックサイクルを費やすことがある。命令の実行速度が速ければ速いほど、処理装置の全体的な性能はより良いことになる。従って、有利なのは、なるべく多くの命令を、なるべく速く実行することである。しかし、いくつかの命令は、他の命令よりも遥かに複雑である。従って、実行時間と処理装置の資源を、他の命令よりも多く必要とする。そのような命令の例としては、浮動小数点命令、記憶装置からの読み込み／記憶装置への書き出し操作、データの移動命令等がある。

【００１７】

ますます多くの計算機システムを、インターネット、文章作成、及びマルチメディアのアプリケーションで使うようになったので、時が経つにつれ、処理装置に、それらを支援する機能が追加されてきた。例えば、単一命令複数データ（ＳＩＭＤ）型の整数及び浮動小数点命令、並びに、ストリーミングＳＩＭＤ拡張（ＳＳＥ）のような命令は、特定のプログラムの仕事を実行するのに必要な命令の総数を減らす。このことにより、消費電力を減らすこともできる。このような命令がソフトウェアの性能を高速化できるのは、複数のデータ要素に並列に演算を行うことによる。その結果、広範な応用で性能を上げられる。応用は、映像の処理、発話の処理、及び画像や写真の処理を含む。ＳＩＭＤ命令の実装は、超小型処理装置や類似の論理回路で行われている。このような実装は、通常、多数の問題を孕んでいる。更に、ＳＩＭＤ演算は複雑なので、大抵は追加の回路が必要になる。追加の回路により、データを正しく処理して演算する。

【００１８】

現在、少なくとも２つのパック化演算対象のデータ要素の各々を比べるＳＩＭＤ命令は存在しない。本発明の１つの実施例で行うようなＳＩＭＤパック化比較命令が無いと、応用プログラムで同じ結果を得るために、多数の命令及びデータレジスタが必要になることがある。応用プログラムは、例えば文字列についての、解釈、圧縮及び復元、処理、並びに演算を行う。本願で開示する実施例では、「文字列」の比較と「列」の比較を、相互に交換可能なように参照する。しかし、本発明の実施例を、情報のいかなる列（例えば、文字の列、数値の列、又は他のデータの列）にも適用してよい。

【００１９】

従って、本願発明の実施例による、少なくとも１つの文字列比較命令は、プログラムのオーバーヘッド及び必要な資源を減らせる。本願発明の実施例は、文字列を構文解析する演算を、ＳＩＭＤ関連のハードウェアを利用する算法として実装する方法を提供する。現在、ＳＩＭＤレジスタにあるデータについて、文字列を構文解析する演算を行うことは、やや困難で手間がかかる。算法によっては、算術演算を実行する肝心の命令の数よりも、算術演算のためにデータを配置する命令に、より多くの数を必要とするほどである。本願発明の実施例による文字列比較演算の実施例を実装することにより、文字列を処理するために必要な命令の数を大幅に減らせる。

【００２０】

本願発明の実施例は、文字列を比べる１つ以上の演算を実装するための命令を含む。文字列を比べる演算は、一般に、データの２つの列からのデータ要素を比較することに関する。この比較により、どのデータ要素が合致するかを判断する。別の変形例を、汎用の文字列比較算法について作ってもよい。この算法も後で開示する。一般化した意味では、文字列比較演算の１つの実施例を、２つのパック化演算対象中にある個々のデータ要素に適用する。２つのパック化演算対象は、データの２つの列を示す。この文字列比較演算の実施例を、次のように汎用的に示せる：

ＤＥＳＴ１　＜－　ＳＲＣ１　ｃｍｐ　ＳＲＣ２；

１つのパック化したＳＩＭＤデータ演算対象について、この汎用演算を、各演算対象の各データ要素の位置に適用できる。

【００２１】

上記の動作において、「ＤＥＳＴ」と「ＳＲＣ」は、対応するデータや動作の送信先と送信元を表す一般的な用語である。実施形態では、レジスタ、またはメモリ、または図示したものとは異なる名称や機能を有するその他の記憶領域により実施できる。例えば、一実施形態では、ＤＥＳＴ１は一時的記憶レジスタやその他の記憶領域であり、ＳＲＣ１とＳＲＣ２は送信先の第１と第２の記憶レジスタまたはその他の記憶領域である。他の実施形態では、ＳＲＣ及びＤＥＳＴ記憶領域は同一記憶領域内（例えば、ＳＩＭＤレジスタ）の異なるデータ記憶要素に対応する。

【００２２】

さらに、一実施形態では、ストリング比較動作により、あるソースレジスタの各要素が他のソースレジスタの各要素と等しいかどうかのインジケータを生成し、そのインジケー

タをＤＥＳＴ１等のレジスタに記憶する。一実施形態では、インジケータはインデックス値である。他の実施形態では、インジケータはマスク値である。他の実施形態では、インジケータはその他のデータ構造やポインタを表す。

【００２３】

図１Ａはコンピュータシステムの一例を示すブロック図である。このコンピュータシステムはプロセッサを有する。このプロセッサは、本発明の一実施形態によるストリング比較動作の命令を実行する実行ユニットを含む。システム１００は、ここに説明する実施形態のような、本発明により、データを処理するアルゴリズムを実行する論理回路を含む実行ユニットを利用する、プロセッサ１０２等のコンポーネントを含む。システム１００は、カリフォルニア州サンタクララ市のインテルコーポレイションから入手可能なＰＥＮＴＩＵＭ（登録商標）ＩＩＩ、ＰＥＮＴＩＵＭ（登録商標）４、Ｘｅｏｎ（商標）、Ｉｔａｎｉｕｍ（登録商標）、ＸＳｃａｌｅ（登録商標）、ＳｔｒｏｎｇＡＲＭ（登録商標）に基づくプロセッシングシステムを表す。しかし、（他のマイクロプロセッサを有するＰＣ、エンジニアリングワークステーション、セットトップボックス等を含む）他のシステムを使うことも可能である。一実施形態では、サンプルシステム１００は、ワシントン州レドモンド市のマイクロソフトコーポレーションのウィンドウズ（登録商標）オペレーティングシステムの一バージョンを実行するが、他のオペレーティングシステム（ユニックス、リナックス等）、組み込みソフトウェア、及び／またはグラフィカルユーザインターフェイス等を用いても良い。このように、本発明の実施形態は、ハードウェア回路とソフトウェアの特定の組み合わせには限定されない。

【００２４】

実施形態はコンピュータシステムには限定されない。本発明の別の実施形態は、その他のデバイス、例えばハンドヘルドデバイスや組み込みアプリケーション等で利用することもできる。ハンドヘルドデバイスの例としては、セルラ電話、インターネットプロトコルデバイス、デジタルカメラ、パーソナルデジタルアシスタント（ＰＤＡ）、ハンドヘルドＰＣなどがある。組み込みアプリケーションには、マイクロコントローラ、デジタルシグナルプロセッサ（ＤＳＰ）、システムオンチップ、ネットワークコンピュータ（ＮｅｔＰＣ）、セットトップボックス、ネットワークハブ、ワイドエリアネットワーク（ＷＡＮ）スイッチ、その他のオペランドにストリング比較演算を実行するシステムがある。さらに、複数のデータ（several data）に対して同時に命令を実行してマルチメディアアプリケーションの効率を向上させるアーキテクチャを組み込んだ。データのタイプとボリュームが大きくなるにつれ、コンピュータやそのプロセッサはより効率的な方法でデータを操作するように高機能化（enhanced）されねばならない。

【００２５】

図１Ａは、コンピュータシステム１００のブロック図であり、プロセッサ１０２を有する。プロセッサ１０２は、１つまたは複数のオペランド（operands）のデータ要素を比較するアルゴリズムを実行する１つまたは複数の実行ユニット１０８を含む。一実施形態をシングルプロセッサデスクトップまたはサーバシステムについて説明するが、別の実施形態をマルチプロセッサシステムで利用することができる。システム１００はハブアーキテクチャの一例である。コンピュータシステム１００は、データ信号を処理するプロセッサ１０２を含む。プロセッサ１０２は、ＣＩＳＣ（complex instruction set computer）マイクロプロセッサ、ＲＩＳＣ（reduced instruction set computing）マイクロプロセッサ、ＶＬＩＷ（very long instruction word）マイクロプロセッサ、複数の命令セットの組み合わせを実装したプロセッサ、その他のデジタルシグナルプロセッサ等の任意のプロセッサである。プロセッサ１０２は、プロセッサバス１１０と結合し、プロセッサバス１１０により、プロセッサ１０２とシステム１００の他のコンポーネントとの間でデータ信号を送信できる。システム１００の要素は、本技術分野の当業者に周知である従来の機能を実行する。

【００２６】

一実施形態では、プロセッサ１０２はレベル１（Ｌ１）内部キャッシュメモリ１０４を

含む。アーキテクチャによって、プロセッサ１０２は単一内部キャッシュを有しても、複数内部キャッシュレベルを有していてもよい。あるいは、他の実施形態では、キャッシュメモリはプロセッサ１０２の外部にあってもよい。他の実施形態では、具体的な実施形態及び必要性に応じて内部キャッシュと外部キャッシュを組み合わせてもよい。レジスタファイル１０６は、整数レジスタ、浮動小数点レジスタ、ステータスレジスタ、命令ポインタレジスタを含む様々なレジスタに相異なるタイプのデータを格納できる。

【００２７】

プロセッサ１０２には、実行ユニット１０８もあり、整数及び浮動小数点の演算を実行する論理回路を含む。プロセッサ１０２は、マクロ命令のマイクロコードを格納するマイクロコード（μコード）ROMも含む。この実施形態では、実行ユニット１０８はパック化命令セット１０９を処理する論理回路を含む。一実施形態では、パック化命令セット１０９は、複数のオペランドの要素を比較するパック化ストリング比較命令（packed string comparison instruction）を含む。汎用プロセッサ１０２の命令セットにパック化命令セット１０９を含めることにより、その命令を実行する関連回路とともに、多くのマルチメディアアプリケーションで利用する演算を汎用プロセッサ１０２においてパック化データを用いて実行することができる。このように、プロセッサのデータバスの幅を最大限に用いてパック化データ（packed data）に演算を行ことにより、多くのマルチメディアアプリケーションを高速化し、より効率的に実行することができる。これにより、プロセッサのデータバスを介してデータを小さい単位で転送して、一度に一データ要素に演算を実行する必要が無くなる。

マイクロコントローラ、組み込みプロセッサ、グラフィックスデバイス、DSP、その他のタイプの論理回路において、実行ユニット１０８の別の実施形態を利用することもできる。システム１００は、メモリ１２０を含む。メモリ１２０は、ＤＲＡＭ（dynamic random access memory）デバイス、ＳＲＡＭ（static random access memory）デバイス、フラッシュメモリデバイス、その他のメモリデバイスである。メモリ１２０は、プロセッサ１０２により実行できる、データ信号で表された命令及び／またはデータを格納できる。

システム論理チップ１１６はプロセッサバス１１０とメモリ１２０に結合している。例示した実施形態では、システム論理チップ１１６はメモリコントローラハブ（ＭＣＨ）である。プロセッサ１０２は、プロセッサバス１１０を介してＭＣＨ１１６と通信できる。ＭＣＨ１１６は、命令とデータの格納、グラフィックスコマンド、データ、及びテクスチャの格納のために、メモリ１２０への広帯域幅メモリパス１１８を提供する。ＭＣＨ１１６は、プロセッサ１０２、メモリ１２０、及びシステム１００のその他のコンポーネントの間でデータ信号を方向付け（direct）、プロセッサバス１１０、メモリ１２０、及びシステムＩ／Ｏ１２２間のデータ信号をブリッジする。実施形態によっては、システム論理チップ１１６は、グラフィックスコントローラ１１２に結合するためのグラフィックスポートを提供する。ＭＣＨ１１６は、メモリインターフェイス１１８を通してメモリ１２０に結合している。グラフィックスカード１１２は、ＡＧＰ（Accelerated Graphics Port）インターコネクト１１４によりＭＣＨ１１６に結合されている。

【００２８】

システム１００は、独自のハブインターフェイスバス１２２を用いて、ＭＣＨ１１６をＩ／Ｏコントローラハブ（ＩＣＨ）１３０に結合する。ＩＣＨ１３０は、ローカルＩ／Ｏバスを介してＩ／Ｏデバイスに直接接続する。ローカルＩ／Ｏバスは、メモリ１２０、チップセット、及びプロセッサ１０２に周辺機器を接続する高速Ｉ／Ｏバスである。例としては、オーディオコントローラ、ファームウェアハブ（フラッシュＢＩＯＳ）１２８、ワイヤレストランシーバ１２６、データストレージ１２４、ユーザ入力及びキーボードインターフェイスを含むレガシーＩ／Ｏコントローラ、ＵＳＢ（Universal Serial Bus）等のシリアル拡張ポート、及びネットワークコントローラ１３４がある。データストレージデバイス１２４は、ハードディスクドライブ、フロッピー（登録商標）ディスクドライブ、ＣＤ－ＲＯＭデバイス、フラッシュメモリデバイス、その他の大容量ストレージデバイ

スである。

【００２９】

システムの他の実施形態の場合、ストリング比較命令を含むアルゴリズムを実行する実行ユニットをシステムオンチップ（system on a chip）で利用できる。システムオンチップの一実施形態は、プロセッサ及びメモリである。かかるシステムのメモリはフラッシュメモリである。フラッシュメモリはプロセッサ及びその他のシステムコンポーネントと同じダイ（die）にあってもよい。また、他の論理ブロック、例えばメモリコントローラまたはグラフィックスコントローラ等がシステムオンチップ上にあってもよい。

【００３０】

図１Ｂは、本発明の一実施形態の原理を化体するデータ処理システム１４０を示す。当業者には言うまでもなく、本発明の範囲から逸脱することなく、ここに説明する実施形態を別の処理システムで利用することもできる。

【００３１】

コンピュータシステム１４０は、ストリング比較演算を含むＳＩＭＤ演算を実行できるプロセッシングコア１５９を有する。一実施形態では、プロセッシングコア１５９は、任意タイプのアーキテクチャの処理ユニットを表し、ＣＩＳＣ、ＲＩＳＣ、ＶＬＩＷなど各タイプのアーキテクチャを含むが、これらには限定されない。プロセッシングコア１５９は、１つまたは複数のプロセステクノロジーでの生産に適しており、機械読み取り可能媒体で十分に詳しく表せるので、生産が容易になる。

【００３２】

プロセッシングコア１５９は、実行ユニット１４２、一組のレジスタファイル１４５、及びデコーダ１４４を有する。プロセッシングコア１５９は、この他の回路（図示せず）も含むが、この回路は本発明を理解するためには必要ない。実行ユニット１４２は、プロセッシングコア１５９が受け取った命令を実行するために使用する。実行ユニット１４２は、一般的なプロセッサ命令を認識するのに加え、パック化命令セット１４３の命令を認識して、パック化データフォーマットに演算を実行する。パック化命令セット１４３は、ストリング比較演算をサポートする命令を含み、他のパック化命令を含んでも良い。実行ユニット１４２は内部バスによりレジスタファイル１４５に結合している。レジスタファイル１４５は、データを含む情報を格納する、プロセッシングコア１５９上の記憶領域を表す。上記の通り、パック化データを記憶するのに用いる記憶領域は必須ではない。実行ユニット１４２はデコーダ１４４に結合している。デコーダ１４４は、プロセッシングコア１５９が受け取った命令を制御信号及び／またはマイクロコードエントリーポイント（microcode entry points）にデコードするために用いられる。実行ユニット１４２は、これらの制御信号及び／またはマイクロコードエントリーポイントに応じて適切な演算を実行する。

【００３３】

プロセッシングコア１５９は、他の様々なシステムデバイスと通信するためにバス１４１と結合されている。

システムデバイスには、シンクロナスＤＲＡＭ（ＳＤＲＡＭ）コントロール１４６、スタティックＲＡＭ（ＳＲＡＭ）コントロール１４７、バーストフラッシュメモリインターフェイス１４８、ＰＣＭＣＩＡ／コンパクトフラッシュ（登録商標）（ＣＦ）カードコントロール１４９、液晶ディスプレイ（ＬＣＤ）コントロール１５０、ＤＭＡコントローラ１５１、代替バスマスターインターフェイス１５２が含まれるが、これらには限定されない。一実施形態では、データプロセッシングシステム１４０は、Ｉ／Ｏバス１５３を介して様々なＩ／Ｏデバイスと通信するためのＩ／Ｏブリッジ１５４も有する。Ｉ／Ｏデバイスには、例えばＵＡＲＴ１５５、ＵＳＢ１５６、ブルートゥースワイヤレスＵＡＲＴ１５７、及びＩ／Ｏ拡張インターフェイス１５８が含まれるが、これらには限定されない。

【００３４】

データプロセッシングシステム１４０の一実施形態は、ストリング比較演算を含むＳＩＭＤ演算を実行できる、モバイル、ネットワーク及び／またはワイヤレス通信およびプロ

セッシングコア１５９である。プロセッシングコア１５９は、様々なオーディオ、ビデオ、画像化、及び通信アルゴリズムでプログラムすることができる。これらのアルゴリズムには、例えば、ウォルシュ・アダマール変換、高速フーリエ変換、離散余弦変換（ＤＣＴ）、これらのそれぞれの逆変換；色空間変換等の圧縮・解凍方法、ビデオエンコード動き予測、またはビデオデコード動き補償；パルスコード変調（ＰＣＭ）等の変復調（ＭＯＤＥＭ）機能等が含まれる。

【００３５】

図１Ｃは、ＳＩＭＤストリング比較演算を実行できるデータ処理システムのさらに別の実施形態を示す。別の一実施形態によるデータプロセッシングシステム１６０は、メインプロセッサ１６６、ＳＩＭＤコ・プロセッサ１６１、キャッシュメモリ１６７、及び入出力システム１６８を含む。入出力システム１６８は、任意的に、ワイヤレスインターフェイス１６９に結合している。ＳＩＭＤコ・プロセッサ１６１は、ストリング比較演算を含むＳＩＭＤ演算を実行できる。プロセッシングコア１７０は、１つまたは複数のプロセステクノロジーでの生産に適しており、機械読み取り可能媒体で十分に詳しく表せるので、プロセッシングコア１７０を含むデータプロセッシングシステム１６０の全部または一部の生産が容易になる。

【００３６】

一実施形態では、ＳＩＭＤコ・プロセッサ１６１は、実行ユニット１６２と一組のレジスタファイル１６４を有する。メインプロセッサ１６５の一実施形態は、実行ユニット１６２が実行するＳＩＭＤストリング比較命令を含む命令セット１６３の命令を認識するデコーダ１６５を有する。別の実施形態では、ＳＩＭＤコ・プロセッサ１６１は、デコーダ１６５Ｂの少なくとも一部を有し、命令セット１６３の命令をデコードする。プロセッシングコア１７０は、この他の回路（図示せず）も含むが、この回路は本発明の実施形態を理解するためには必要ない。

【００３７】

動作中、メインプロセッサ１６６は、キャッシュメモリ１６７や入出力システム１６８とのインターラクションを含む、一般的なタイプのデータ処理演算を制御するデータ処理命令ストリーム（stream of data processing instructions）を実行する。ＳＩＭＤコ・プロセッサ命令はデータ処理命令ストリームの中に組み込まれている。メインプロセッサ１６６のデコーダ１６５は、ＳＩＭＤコ・プロセッサ命令を、付随するＳＩＭＤコ・プロセッサ１６１が実行すべきタイプであるとして認識する。従って、メインプロセッサ１６６は、これらのＳＩＭＤコ・プロセッサ命令（または、ＳＩＭＤコ・プロセッサ命令を表す制御信号）をコ・プロセッサバス１６６上に発行し、付随するＳＩＭＤコ・プロセッサはコ・プロセッサバス１６６からコ・プロセッサ命令を受け取る。この場合、ＳＩＭＤコ・プロセッサ１６１は、それに宛てられたＳＩＭＤコ・プロセッサ命令を受け取り、実行する。

【００３８】

ＳＩＭＤコ・プロセッサ命令が処理するデータは、ワイヤレスインターフェイス１６９を介して受け取ってもよい。一例として、音声通信をデジタル信号の形式で受信して、ＳＩＭＤコ・プロセッサ命令で処理して、その音声通信を表すデジタルオーディオサンプルを再生する。他の一例として、圧縮オーディオ及び／またはビデオをデジタルビットストリームの形式で受信して、ＳＩＭＤコ・プロセッサ命令で処理して、そのデジタルオーディオサンプル及び／またはモーションビデオフレームを再生してもよい。プロセッシングコア１７０の一実施形態では、メインプロセッサ１６６とＳＩＭＤコ・プロセッサ１６１は単一のプロセッシングコア１７０に集積されている。プロセッシングコア１７０は、実行ユニット１６２、一組のレジスタファイル１６４、及びデコーダ１６５を有し、ＳＩＭＤストリング比較命令を含む命令セット１６３の命令を認識する。

【００３９】

図２は、プロセッサ２００のマイクロアーキテクチャを示すブロック図である。プロセッサ２００は、本発明の一実施形態によるストリング比較命令を実行する論理回路を含む

。ストリング比較命令の一実施形態では、第１のオペランドの各データ要素を第２のオペランドの各データ要素と比較して、各比較結果が一致したかを示すインジケータを格納する。実施形態では、サイズがバイト、ワード、ダブルワード、クアッドワード（quadword）等であり、データタイプが整数や浮動小数点であるデータ要素に、ストリング比較命令を演算することができる。一実施形態では、インオーダー（in-order）フロントエンド２０１がプロセッサ２００の一部となっており、実行するマクロ命令をフェッチして、後でプロセッサパイプラインで使用するように準備する。フロントエンド２０１は複数のユニットを含む。一実施形態では、命令プリフェッチャ２２６は、メモリからマクロ命令をフェッチして、命令デコーダ２２８に供給（feed）する。命令デコーダ２２８は、マクロ命令を、機械が実行可能なマイクロ命令またはマイクロ演算（micro opやμopsとも呼ぶ）と呼ばれるプリミティブ（primitives）にデコードする。一実施形態では、トレースキャッシュ２３０は、デコードされたマイクロ演算を取って、プログラムオーダーシーケンス（program ordered sequences）またはトレース（traces）を組立、実行のためにマイクロ演算キュー２３４に入れる。トレースキャッシュ２３０が複雑なマクロ命令を見つける（encounter）と、マイクロコードＲＯＭ２３２がその演算を完了するのに必要なマイクロ演算を供給する。

【００４０】

多数のマクロ命令は単一のマイクロ演算に変換されるが、他のマクロ命令はその演算を完全に完了するのに複数の（several）マイクロ演算を必要とする。一実施形態では、１つのマクロ命令を完了するのに５つ以上のマイクロ演算が必要であれば、デコーダ２２８はマイクロコードＲＯＭ２３２にアクセスしてマクロ命令を実行する。一実施形態では、パック化ストリング比較命令を少数のマイクロ演算にデコードして、命令デコーダ２２８で処理する。他の実施形態では、演算を行うのに多数のマイクロ演算が必要な場合、パック化ストリング比較アルゴリズムをマイクロコードＲＯＭ２３２内に格納することもできる。トレースキャッシュ２３０は、マイクロコードＲＯＭ２３２のストリング比較アルゴリズムのマイクロコードシーケンスを読むための、正しいマイクロ命令ポインタを決定するエントリーポイントのプログラマブルロジックアレイ（ＰＬＡ）である。マイクロコードＲＯＭ２３２がカレントの（current）マクロ命令のマイクロ演算のシーケンス決定（sequencing）を終了すると、マシンのフロントエンド２０１は、トレースキャッシュ２３０からマイクロ演算のフェッチを再開する。

【００４１】

一部のＳＩＭＤその他のマルチメディアタイプの命令は複雑な命令であると考えられる。浮動小数点関係の命令もほとんどが複雑な命令である。そこで、命令デコーダ２２８は複雑なマクロ命令が来ると（encounter）、マイクロコードＲＯＭ２３２の適切な場所にアクセスして、そのマクロ命令のマイクロコードシーケンスを読み出す。そのマクロ命令を実行するのに必要な様々なマイクロ演算を、アウトオブオーダー（out-of-order）実行エンジン２０３に送り、適切な整数実行ユニット及び浮動小数点実行ユニットで実行する。

【００４２】

アウトオブオーダー実行エンジン２０３は、マイクロ命令の実行準備をするところである。アウトオブオーダー実行論理回路は、多数のバッファを有し、マイクロ命令がパイプラインを下り、実行スケジューリングがなされるにつれ、実行を最適化するように、マイクロ命令のフローをスムースにして並べ替える。アロケータロジックは、各マイクロ演算を実行するために必要なマシンバッファとリソースをアロケートする。レジスタリネーミングロジックは、ロジックレジスタをレジスタファイルのエントリーにリネーム（rename）する。アロケータは、命令スケジューラであるメモリスケジューラ、高速スケジューラ２０２、低速・一般浮動小数点スケジューラ２０４、及び単純浮動小数点スケジューラ２０６の前にある、メモリ演算用と非メモリ演算用の２つのマイクロ演算キューの一方の各マイクロ演算にエントリーをアロケートする。マイクロ演算スケジューラ２０２、２０４、２０６は、マイクロ演算が依存する入力レジスタオペランドソースの準備状況（readin

ess）と、マイクロ演算がその演算を完了するのに必要とする実行リソースの利用可能性とに基づき、マイクロ演算がいつ実行できるか決定する。本実施形態の高速スケジューラ２０２は、メインクロックサイクルの半分ごとにスケジューリングをできるが、他のスケジューラはメインプロセッサクロックサイクルごとにしかスケジューリングができない。複数のスケジューラはディスパッチポートをアービトレーションしてマイクロ演算の実行をスケジューリングする。

【００４３】

レジスタファイル２０８、２１０はスケジューラ２０２、２０４、２０６と、実行ブロック２１１の実行ユニット２１２、２１４、２１６、２１８、２２０、２２２、２２４との間にある。整数演算と浮動小数点演算にはそれぞれ別のレジスタファイル２０８、２１０がある。他の実施形態では、整数レジスタ及び浮動小数点レジスタは同一レジスタファイルにあってもよい。本実施形態の各レジスタファイル２０８、２１０は、ちょうど完了した結果であってまだレジスタファイルに書き込まれていないものを、新しいディペンデント（dependent）なマイクロ演算にバイパスまたは転送するバイパスネットワークを含む。整数レジスタファイル２０８と浮動小数点レジスタファイル２１０は、互いにデータをやりとりすることができる。一実施形態では、整数レジスタファイル２０８は、下位３２ビット用と上位３２ビット用である２つの別々のレジスタファイルに分離されている。一実施形態の浮動小数点レジスタファイル２１０は、１２８ビット幅のエントリーを有する。浮動小数点命令は、一般的には６４ビットから１２８ビットの幅のオペランドを有するからである。

【００４４】

実行ブロック２１１は、実行ユニット２１２，２１４，２１６，２１８，２２０，２２２，２２４を含み、これらにより命令が実際に実行される。このセクションにはレジスタファイル２０８，２１０が含まれる。レジスタファイル２０８，２１０は、マイクロ命令の実行に必要な整数及び浮動小数点データオペランドの値が記憶される。本実施形態のプロセッサ２００は、複数の実行ユニット、すなわちアドレス生成ユニット（AGU）２１２、AGU２１４、高速ALU２１６、高速ALU２１８、低速ALU２２０、浮動小数点ＡＬＵ２２２、浮動小数点moveユニット２２４により構成されている。本実施形態では、浮動小数点実行ブロック２２２、２２４は、浮動小数点演算、ＭＭＸ演算、ＳＩＭＤ演算、及びＳＳＥ演算を実行する。本実施形態の浮動小数点ＡＬＵ２２２は、６４ビット対６４ビットの浮動小数点割り算器を含み、割り算、平方根、剰余のマイクロ演算を実行する。本発明の実施形態では、浮動小数点値が関わる動作は浮動小数点ハードウェアで行われる。例えば、整数形式と浮動小数点形式の間の変換には浮動小数点レジスタファイルが関与する。同様に、浮動小数点割り算演算は浮動小数点割り算器で行われる。一方、非浮動小数点型や整数型は整数ハードウェアリソースで処理される。単純かつ頻度が高いＡＬＵ演算は高速ＡＬＵ実行ユニット２１６、２１８に行く。本実施形態の高速ＡＬＵ２１６，２１８は、有効レイテンシーがクロックサイクルの半分である高速演算を実行できる。一実施形態では、ほとんどの複雑な整数演算は低速ALU２２０に行く。低速ALU２２０が、乗算、シフト、フラグロジック、ブランチ処理等のレイテンシーが長いタイプの演算用の整数実行ハードウェアを含むからである。メモリロード・ストア命令は、ＡＧＵ２１２，２１４で実行される。この実施形態は、整数ＡＬＵ２１６，２１８，２２０は、６４ビットデータオペランドに整数演算を実行するものとして説明した。別の実施形態では、ＡＬＵ２１６，２１８，２２０は、１６，３２，１２８，２５６等の様々なデータビットをサポートするように実施することもできる。同様に、浮動小数点ユニット２２２，２２４は、様々な幅のビットを有するある範囲のオペランドをサポートするように実施することもできる。一実施形態では、浮動小数点ユニット２２２、２２４は、ＳＩＭＤ命令やマルチメディア命令とともに、１２８ビット幅のパック化データオペランドに演算をすることができる。

【００４５】

本実施形態では、マイクロ演算スケジューラ２０２，２０４，２０６は、親のロード（load）の実行が終わる前に、ディペンデント演算（dependent operations）をディスパ

ッチする。マイクロ演算はプロセッサ２００においてスペキュレーティブ（speculatively）にスケジューリングされるので、プロセッサ２００はメモリミスを処理するロジックも含む。データキャッシュにおいてデータロードがミスすると、パイプライン中には、データが一時的に正しくないディペンデント演算がある。正しくないデータを使う命令をリプレイメカニズムが追跡し、再実行する。ディペンデント演算のみをリプレイする必要があり、インディペンデント演算は完了することができる。プロセッサの一実施形態のスケジューラとリプレイメカニズムは、ストリング比較演算の命令シーケンスを捉えるように設計されている。

【００４６】

「レジスタ」という用語は、オペランドを特定するマクロ命令の一部として使われる、オンボードプロセッサの記憶場所を言う。換言すると、ここでレジスタとは、プロセッサの外側から（プログラマーの視点から）見えるレジスタである。しかし、一実施形態のレジスタは、特定タイプの回路を意味していると限定すべきではない。むしろ、実施形態のレジスタは、データを記憶して供給し、本明細書に記載する機能を実行できるだけでよい。ここで説明したレジスタは、専用の物理的レジスタ、レジスタリネーミングを利用した動的割当ての物理的レジスタ、専用の物理的レジスタ及び動的割当の物理的レジスタの組み合わせなど、任意数の異なる技術を用いて、プロセッサ内の回路により実施することができる。一実施形態では、整数レジスタは３２ビットの整数データを記憶する。一実施形態のレジスタファイルは、パック化データ用に８個のマルチメディアＳＩＭＤレジスタも含む。以下の説明では、レジスタは、カリフォルニア州サンタクララ市のインテルコーポレイションのＭＭＸテクノロジーで実現された、マイクロプロセッサの６４ビット幅ＭＭＸ（登録商標）レジスタ（場合によっては「ｍｍ」レジスタとも呼ぶ）などの、パック化データを保持するように設計されたデータレジスタであるものとする。これらのＭＭＸレジスタは、整数形式と浮動小数点形式とがあるが、ＳＩＭＤ命令やＳＳＥ命令をともなうパック化データ要素に利用できる。同様に、ＳＳＥ２，ＳＳＥ３，ＳＳＥ４またはそれ以降（総称的に「ＳＳＥｘ」と呼ぶ）のテクノロジーに関する１２８ビット幅のＸＭＭレジスタも、このようなパック化データオペランドを保持するために用いることができる。本実施形態では、パック化データや整数データを記憶する際、レジスタは２つのデータタイプを区別する必要はない。

【００４７】

以下の図の実施例では、複数のデータオペランドを説明する。図３Ａは、本発明の一実施形態によるマルチメディアレジスタにおける様々なパック化データタイプを表した図である。図３Ａは、１２８ビット幅オペランドの、パック化バイト３１０、パック化ワード３２０、及びパック化ダブルワード３３０を示している。本実施例のパック化バイトフォーマット３１０は、１２８ビットの長さで、１６個のパック化バイトデータ要素を含む。ここでは、１バイトは８ビットのデータであると定義する。各バイトデータ要素の情報は、バイト０がビット７からビット０まで、バイト１がビット１５からビット８まで、バイト２がビット２３からビット１６まで、そして最終的にバイト１５がビット１２７からビット１２０までに記憶される。このように、レジスタのすべてのビットが利用される。このような記憶構成をとることにより、プロセッサの記憶効率が高まる。また、１６個のデータ要素にアクセスするので、１つの演算を１６個のデータ要素に並行に演算することができる。

【００４８】

一般的に、データ要素は、単一のレジスタや記憶場所（memory location）に格納される個別のデータ（individual piece of data）であり、他のデータ要素と同じ長さのものである。ＳＳＥｘテクノロジーに関連するパック化データシーケンスでは、ＸＭＭレジスタに格納されるデータ要素数は、１２８ビットを個々のデータ要素のビット長で割った数である。ＭＭＸ及びＳＳＥテクノロジーに関連するパック化データシーケンスでは、ＭＭＸレジスタに格納されるデータ要素数は、６４ビットを個々のデータ要素のビット長で割った数である。図３Ａに示したデータタイプは１２８ビット長であるが、本発明の実

施形態は、６４ビット幅でもその他のサイズのオペランドでも動作可能である。本実施例のパック化ワードフォーマット３２０は、１２８ビットの長さで、８個のパック化ワードデータ要素を含む。各パック化ワードは１６ビットの情報を含む。図３Ａのパック化ダブルワードフォーマット３３０は、１２８ビットの長さで、４個のパック化ダブルワードデータ要素を含む。各パック化ダブルワードデータ要素は３２ビットの情報を含む。パック化クアドワード（quadword）は、１２８ビットの長さであり、２つのパック化クアドワードデータ要素を含む。

【００４９】

図３Ｂは、別のレジスタ内データ記憶フォーマットを示す図である。各パック化データは独立した２つ以上のデータ要素を含んでいても良い。パック化ハーフ３４１、パック化シングル３４２、及びパック化ダブル３４３である３つのパック化データフォーマットを示した。パック化ハーフ３４１、パック化シングル３４２、及びパック化ダブル３４３の一実施形態は、固定小数点データ要素である。別の実施形態では、パック化ハーフ３４１、パック化シングル３４２、及びパック化ダブル３４３は、浮動小数点データ要素を含んでいてもよい。パック化ハーフ３４１の別の一実施形態は、８個の１６ビットデータ要素を含む１２８ビット長データである。パック化シングル３４２の一実施形態は、１２８ビットの長さであり、４個の３２ビットデータ要素を含む。パック化ダブル３４３の一実施形態は、１２８ビットの長さであり、２つの６４ビットデータ要素を含む。言うまでもなく、かかるパック化データフォーマットは、例えば、９６ビット、１６０ビット、１９２ビット、２２４ビット、２５６ビット、またはそれ以上のレジスタ長に拡張することができる。

【００５０】

図３Ｃは、本発明の一実施形態によるマルチメディアレジスタにおける様々な符号付き及び符号無しのパック化データタイプを表した図である。符号無しパック化バイト表現３４４は、ＳＩＭＤレジスタにおける符号無しパック化バイトの記憶を示す。各バイトデータ要素の情報は、バイト０がビット７からビット０まで、バイト１がビット１５からビット８まで、バイト２がビット２３からビット１６まで、そして最終的にバイト１５がビット１２７からビット１２０までに格納される。このように、レジスタのすべてのビットが利用される。このような記憶構成をとることにより、プロセッサの記憶効率が高まる。また、１６個のデータ要素にアクセスするので、１つの演算を１６個のデータ要素に並行に演算することができる。符号付きパック化バイト表現３４５は、符号付きパック化バイトの記憶を示す。各バイトデータ要素の８番目のビットは符号インジケータである。符号無しパック化ワード表現３４６は、ワード７からワード０までがどのようにＳＩＭＤレジスタに記憶されるかを示している。符号付きパック化ワード表現３４７は、符号無しパック化ワードレジスタ内表現３４６と同様である。各ワードデータ要素の１６番目のビットは符号インジケータである。符号無しパック化ダブルワードデータ表現３４８は、ダブルワードデータ要素がどのように格納されるか示している。符号付きパック化ダブルワード表現３４９は、符号無しパック化ダブルワードレジスタ内表現３４８と同様である。必要な符号ビットは、各ダブルワードデータ要素の３２番目のビットである。一実施形態では、オペランドは定数でもよく、それが付随する命令によって変化しない。

【００５１】

図３Ｄは、オペレーションエンコーディング（opcode）フォーマット３６０の一実施形態を示す。これは、３２ビット以上であり、レジスタ・メモリオペランドのアドレッシングモードは、「IA-32 Intel Architecture Software Developer's Manual Volumne 2: Instruction Set Reference」に記載されたopcodeフォーマットのタイプに対応している。このマニュアルは、ワールドワイドウェブintel.com/design/litcentrで、カリフォルニア州サンタクララ市のインテルコーポレイションから入手できる。一実施形態では、ストリング比較演算は１つまたは複数のフィールド３６１及び３６２でエンコードされる。２つまでのソースオペランド識別子３６４と３６５を含め、一命令につき２つまでのオペランドの場所が特定される。ストリング比較命令の一実施形態では、デスティネ

ーションオペランド識別子３６６はソースオペランド識別子３６４と同じであり、他の実施形態では異なる。別の実施形態では、デスティネーションオペランド識別子３６６はソースオペランド識別子３６５と同じであり、他の実施形態では異なる。ストリング比較命令の一実施形態では、ソースオペランド識別子３６４と３６５により特定されるソースオペランドの一方は、ストリング比較命令の結果により上書きされる。一方、他の実施形態では、識別子３６４はソースレジスタ要素に対応し、識別子３６５はデスティネーションレジスタ要素に対応する。ストリング比較命令の一実施形態では、オペランド識別子３６４と３６５は、３２ビットまたは６４ビットのソース及びデスティネーションオペランドを特定するために用いられる。

【００５２】

図３Ｅは、４０ビットまたはそれ以上の、別のオペレーションエンコーディング（opcode）フォーマット３７０を示す。opcodeフォーマット３７０は、opcodeフォーマット３６０に対応し、任意的なプレフィックスバイト３７８を含む。ストリング比較演算のタイプは、１つまたは複数のフィールド３７８、３７１及び３７２でエンコードされる。１つの命令につき２つまでのオペランドの場所がソースオペランド識別子３７４と３７５、及びプレフィックスバイト３７８により特定される。ストリング比較命令の一実施形態では、プレフィックスバイト３７８は、３２ビット、６４ビット、または１２８ビットのソース及びデスティネーションオペランドを特定するために用いられる。ストリング比較命令の一実施形態では、デスティネーションオペランド識別子３７６はソースオペランド識別子３７４と同じであり、他の実施形態では異なる。別の実施形態では、デスティネーションオペランド識別子３７６はソースオペランド識別子３７５と同じであり、他の実施形態では異なる。一実施形態では、ストリング比較演算は、オペランド識別子３７４と３７５により特定されるオペランドの各要素を、オペランド識別子３７４と３７５により特定される他のオペランドの各要素と比較、その各要素をストリング比較演算の結果により上書きする。一方、他の実施形態では、識別子３７４と３７５により特定されるオペランドのストリング比較は、他のレジスタの他のデータ要素に書き込まれる。opcodeフォーマット３６０と３７０では、ＭＯＤフィールド３６３と３７３、及び任意的なスケール・インデックス・ベース及びディスプレースメントバイトにより部分的に規定される、レジスタからレジスタ、メモリからレジスタ、メモリによるレジスタ、レジスタによるレジスタ、イミーディエイト（immediate）によるレジスタ、レジスタからメモリへのアドレッシングが可能である。

【００５３】

次に図３Ｆを参照して、別の実施形態では、６４ビット単一命令複数データ（ＳＩＭＤ）算術演算は、コ・プロセッサデータ処理（ＣＤＰ）命令により実行される。オペレーションエンコーディング（opcode）フォーマット３８０は、ＣＤＰopcodeフィールド３８２と３８９を有するかかるＣＤＰ命令を示す。ストリング比較演算の別の実施形態では、ＣＤＰ命令のタイプは、１つまたは複数のフィールド３８３、３８４、３８７及び３８８でエンコードされる。２つまでのソースオペランド識別子３８５と３９０と、１つのデスティネーションオペランド識別子３８６とを含め、一命令につき３つまでのオペランドの場所を特定できる。コ・プロセッサの一実施形態は、８、１６、３２及び６４ビット値で動作できる。一実施形態では、ストリング比較演算は整数データ要素に実行される。実施形態では、ストリング比較命令は、条件フィールド３８１を用いて、条件付きで実行してもよい。ストリング比較命令によっては、ソースデータサイズはフィールド３８３によりエンコードできる。ストリング比較命令の実施形態では、ＳＩＭＤフィールドでゼロ（Ｚ）、ネガティブ（Ｎ）、キャリー（Ｃ）、オーバーフロー（Ｖ）の検出をできる。命令によっては飽和のタイプをフィールド３８４でエンコードしてもよい。

【００５４】

一実施形態では、ストリング比較演算の結果が非ゼロであることを示すために、フィールドまたは「フラグ」を用いてもよい。実施形態によっては、ソース要素が無効であることを示すフラグや、ストリング比較演算の結果のＬＳＢまたはＭＳＢを示すフラグなどの

他のフィールドを使ってもよい。

【００５５】

図４は、本発明による、パック化データオペランドにストリング比較演算を実行するロジックの一実施形態を示すブロック図である。本発明の実施形態は、上記のような様々なタイプのオペランドで機能するように実施できる。一実施形態では、本発明によるストリング比較演算は、特定のデータタイプに作用する命令セットとして実施する。例えば、整数と浮動小数点を含む３２ビットデータタイプの比較を実行するパック化ストリング比較命令を提供する。同様に、整数と浮動小数点を含む６４ビットデータタイプの比較を実行するパック化ストリング比較命令を提供する。以下の説明と実施例により、データ要素が何を表しているかに関わらずデータ要素を比較する比較命令の動作を説明する。説明を簡単にするため、一部の実施例は、データ要素がテキストの言葉である１つまたは複数のストリング比較命令の実行を示す。

【００５６】

一実施形態では、ストリング比較命令は、第１のデータオペランドDATA　A　４１０の各要素を、第２のデータオペランドDATA　B　４２０の各要素と比較し、各比較の結果をRESULTANT　４４０レジスタに格納する。以下の説明では、DATA　A、DATA　B、及びRESULTANTはレジスタであるものとする。しかし、そのようには限定されず、レジスタ、レジスタファイル、及びメモリの記憶場所を含む。一実施形態では、テキストストリング比較命令（例えば、「PCMPxSTRy」）は１つのマイクロ演算にデコードされる。別の実施形態では、各命令は、データオペランドにテキストストリング比較演算を行う様々な数のマイクロ演算にデコードできる。この実施例では、オペランド４１０、４２０は、ワード幅のデータ要素を有するソースレジスタ・メモリに格納された１２８ビット幅の情報である。一実施形態では、オペランド４１０、４２０は、１２８ビットＳＳＥｘ　ＸＭＭレジスタ等の１２８ビット長ＳＩＭＤレジスタに保持される。一実施形態では、ＲＥＳＵＬＴＡＮＴ４４０はＸＭＭデータレジスタでもある。他の実施形態では、ＲＥＳＵＬＴＡＮＴ４４０は、拡張レジスタ（例えば、「ＥＡＸ」）などの他のタイプのレジスタであってもよく、メモリの記憶場所であってもよい。実施形態によっては、オペランドとレジスタは３２、６４、２５６ビットなどの長さであっても良く、バイト、ダブルワード、またはクアドワードサイズのデータ要素を有していてもよい。この実施例のデータ要素はワードサイズであるが、同じコンセプトをバイトやダブルワードサイズの要素に拡張することができる。一実施形態では、データオペランドが６４ビット幅であれば、ＸＭＭレジスタの替わりにＭＭＸレジスタを用いる。

【００５７】

一実施形態では、第１のオペランド４１０は、Ａ７，Ａ６，Ａ５，Ａ４，Ａ３，Ａ２，Ａ１及びＡ０の８つのデータ要素により構成されている。第１と第２のオペランドの要素間の各比較は、結果４４０中のデータ要素の位置に対応してもよい。一実施形態では、第２のオペランド４２０は、Ｂ７，Ｂ６，Ｂ５，Ｂ４，Ｂ３，Ｂ２，Ｂ１及びＢ０の８つのデータセグメントにより構成されている。ここでデータセグメントとは、長さが等しく、１データワード（１６ビット）より構成される。しかし、データ要素とデータ要素位置はワード以外の粒度（granularities）を有していてもよい。各データ要素がバイト（８ビット）、ダブルワード（３２ビット）、またはクアドワード（６４ビット）であるとき、１２８ビットオペランドは１６バイト幅、４ダブルワード幅、または２クアドワード幅のデータ要素をそれぞれ有する。本発明の実施形態は特定の長さのデータオペランドやデータセグメントに限定されず、各実施形態に適切なサイズを利用できる。

【００５８】

オペランド４１０，４２０は、レジスタ、メモリの記憶場所、レジスタファイル、またはこれらの組み合わせ（mix）のどれにあってもよい。データオペランド４１０、４２０は、テキストストリング比較命令とともに、プロセッサの実行ユニットのストリング比較ロジック４３０に送られる。一実施形態では、命令が実行ユニットに到着する時までに、その命令はプロセッサパイプラインで早めにデコードされる。このように、ストリング比

較命令はマイクロ命令（μop）またはその他のデコードされたフォーマットの形式であり得る。一実施形態では、２つのデータオペランド４１０，４２０をストリング比較ロジック４３０が受け取る。一実施形態では、テキストストリング比較ロジックは、２つのデータオペランドの要素が等しいかどうかの表示を生成する。一実施形態では、各オペランドの有効要素のみを比較する。有効要素は、各オペランドの各要素について他のレジスタまたはメモリの記憶場所により示される。一実施形態では、オペランド４１０の各要素をオペランド４２０の各要素と比較する。この比較により、オペランド４１０の要素数にオペランド４２０の要素数をかけた数に等しい比較結果ができる。例えば、各オペランド４１０と４２０が３２ビット値である場合、結果レジスタ４４０は、ストリング比較ロジック４３０で実行されたテキスト比較演算の３２×３２までの結果インジケータを記憶する。一実施形態では、第１と第２のオペランドからのデータ要素は単精度（例えば、３２ビット）であり、他の実施形態では、第１と第２のオペランドのデータ要素は倍精度（例えば、６４ビット）である。他の実施形態では、第１と第２のオペランドは、８、１６、３２ビットを含む任意サイズの整数要素を含み得る。

【００５９】

一実施形態では、すべてのデータ位置のデータ要素は並行に処理される。他の実施形態では、データ要素位置の一部は同時に処理できる。一実施形態では、RESULTANT４４０は、オペランド４１０と４２０に格納された各データ要素間の比較の複数の結果により構成される。具体的には、一実施形態では、結果（RESULTANT）はオペランド４１０または４２０の一方のデータ要素数の２乗だけの比較結果を記憶してもよい。

【００６０】

一実施形態では、RESULTANTは、オペランド４１０と４２０の有効なデータ要素の間の比較のみの比較結果を記憶する。一実施形態では、各オペランドのデータ要素は、明示的または黙示的に有効であると示され得る。例えば、一実施形態では、各オペランドデータ要素は、有効レジスタなどの他の記憶領域内に記憶される、有効ビットなどの有効性インジケータに対応する。一実施形態では、両方のオペランドの各要素の有効性ビットは、同じ有効レジスタに記憶される。しかし、他の実施形態では、１つのオペランドの有効性ビットは、第１の有効レジスタに記憶され、他のオペランドの有効性ビットは第２の有効レジスタに記憶される。有効な要素間でのみ比較を行うように、オペランドデータ要素を比較する前に、またはそれと共に、（例えば、対応する有効ビットをチェックすることにより）両方のデータ要素が有効であるか判断してもよい。

【００６１】

一実施形態では、各オペランドの有効データ要素は、オペランドの一方または両方に記憶されたヌルまたは「ゼロ」フィールドの使用により黙示的に示され得る。例えば、一実施形態では、ヌルバイト（または他のサイズ）を要素に記憶して、ヌルバイトより重要な（significant）データ要素はすべて無効であり、一方、ヌルバイトより重要でないデータ要素はすべて有効であるので、他のオペランドの対応する有効なデータ要素と比較すべきことを示してもよい。さらに、一実施形態では、（上記の通り）１つのオペランドの有効データ要素を明示的に示し、一方、他のオペランドの有効データ要素をヌルフィールドを用いて黙示的に示しても良い。一実施形態では、有効データ要素は、１つ以上のソースオペランド内の有効なデータ要素またはサブエレメント（sub-elements）の数に対応するカウントにより示される。

【００６２】

各オペランドの有効データ要素を示す方法にかかわらず、少なくとも１つの実施形態では、有効であると示された各オペランドのデータ要素を比較する。有効データ要素のみの比較は、様々な実施形態で複数の方法で実行できる。詳細かつ理解可能な説明をする目的では、２つのテキストストリングオペランド間で有効なデータ要素のみを比較する方法は、以下によりもっともよく概念的に説明できる。しかし、以下の説明は、テキストストリングオペランドの有効データ要素のみの比較を以下に概念的に説明または実施するかの一例に過ぎない。他の実施形態では、他の概念的説明や方法を用いて、有効なデータ要素を

いかに比較するかを示す。

【００６３】

一実施形態では、オペランドの有効なデータ要素数が（例えば、有効性レジスタの有効ビットや、最下位から始めて有効なバイト・ワードの数をカウントすることにより）明示的に示されているか、（例えば、オペランド内のヌルキャラクタにより）黙示的に示されているかにかかわらず、各オペランドの有効データ要素のみを互いに比較する。一実施形態では、有効性インジケータの集計と比較するデータ要素を、図５を参照して概念的に説明する。

【００６４】

図５を参照して、一実施形態では、アレイ５０１と５０５は、第１のオペランドと第２のオペランドの各要素がそれぞれ有効であるかどうかを示すエントリーを含む。例えば、上記の例では、アレイ５０１は、第１のオペランドが対応する有効データ要素を含む各アレイ要素には「１」を含む。同様に、アレイ５０５は、第２のオペランドが対応する有効データ要素を含む各アレイ要素に「１」を含む。一実施形態では、アレイ５０１と５０５は、２つのオペランドにある各有効要素に対して、アレイ要素０から始まり１を含む。例えば、一実施形態では、第１のオペランドが４つの有効要素を含む場合、アレイ５０１は最初の４つのアレイ要素にのみ１を含み、アレイ５０１の他のアレイ要素はすべてゼロである。

【００６５】

一実施形態では、アレイ５０１と５０５はサイズが１６要素であり、２つの１２８ビットオペランドの１６個のデータ要素を表し、各々はサイズが８ビット（１バイト）である。他の実施形態では、オペランドのデータ要素のサイズが１６ビットであり、アレイ５０１と５０５は８要素のみを含む。他の実施形態では、アレイ５０１と５０５は、対応するオペランドのサイズに応じて大きくても小さくてもよい。

【００６６】

一実施形態では、第１のオペランドの各データ要素を第２のオペランドの各データ要素と比較し、その結果をｉ×ｊアレイ５１０で表す。例えば、テキストストリングを表す第１のオペランドの第１のデータ要素を、例えば、他のテキストストリングを表す他のオペランドの各データ要素と比較し、アレイ５１０の第１の行内の各アレイ要素に記憶された「１」は、第１のオペランドの第１のデータ要素と第２のオペランドの各データ要素の間の一致に対応する。これは、アレイ５１０が完了するまで、第１のオペランドの各データ要素に対して繰り返される。

【００６７】

一実施形態では、ｉ×ｊエントリーの第２のアレイ５１５が生成され、有効なオペランドのデータ要素のみが等しいかどうかの表示を記憶する。例えば、一実施形態では、アレイ５１０の最初の行５１１の各エントリーを対応する有効なアレイ要素５０６及び有効なアレイ要素５０２と論理的にＡＮＤを取って、その結果をアレイ５１５の対応する要素５１６に配置する。ＡＮＤ演算は、アレイ５１０の各要素と、有効なアレイ５０１及び５０５の対応する要素との間で実行し、その結果をアレイ５２０の対応する要素に配置してもよい。

【００６８】

一実施形態では、結果アレイ５２０は、一オペランドのデータ要素のうち他のオペランドのデータ要素と関係するものがあるか示す。例えば、結果アレイ５２０は、アレイ５１５の要素のペアをＡＮＤ演算し、ＡＮＤのすべての結果をＯＲ演算することにより、他のオペランドのデータ要素により決まる範囲内にデータ要素があるか示すビットを記憶することができる。

【００６９】

図５は、少なくとも２つのパック化オペランドのデータ要素間の比較に関する様々なインジケータを記憶する結果アレイ５２０も示す。例えば、結果アレイ５２０は、アレイ５１５の対応する要素をＯＲ演算することにより、２つのオペランド間に等しいデータ要素

はあるかどうかを示すビットを記憶する。アレイ５１５のアレイ要素のどれかが、例えば、オペランドの有効なデータ要素間に一致するものがあることを示す「１」を含む場合、これは結果アレイ５２０に反映される。結果アレイ５２０の要素をＯＲ演算して、オペランドの有効なデータ要素が等しいか判断することもできる。

【００７０】

一実施形態では、アレイ内の隣接する「１」を検出することにより、結果アレイ５２０内の、２つのオペランドのデータ要素間の有効な一致の連続を検出する。一実施形態では、これは、連続する結果アレイ要素を一度にＡＮＤ演算し、「０」を検出するまで一ＡＮＤ演算の結果と次の結果とをＡＮＤ演算することにより、実現できる。他の実施形態では、他の論理を用いて２つのパック化演算のデータ要素の有効な一致の範囲を検出してもよい。

【００７１】

一実施形態では、結果アレイ５２０は、対応する結果アレイエントリーに「１」を返すことにより、両方のオペランドの各データ要素が一致するか示すこともできる。すべてのエントリーが等しいか判断するため、結果アレイエントリーにＸＯＲ演算を実行してもよい。他の実施形態では、他の論理を用いて２つのオペランドの有効データ要素が等しいか判断してもよい。

【００７２】

一実施形態では、データ要素のストリングがデータ要素の他のストリング内のどこかにあることを、テストストリングを他のストリングの同じサイズの部分と比較して、テストストリングと他のストリングのその部分との一致を結果アレイに示すことにより、検出できる。例えば、一実施形態では、第１のオペランドの３つのデータ要素に対応する３つのキャラクタのテストストリングを、第２のストリングの３つのデータ要素の第１のセットと比較する。一致を検出したら、その一致を結果アレイに反映させる。これは、一致に対応する３つの結果エントリーのグループに「１」を格納することにより行う。テストストリングを他のオペランドの次の３つのデータ要素と比較する。または、比較されるにつれてテストストリングが他のオペランドに沿って「スライド」するように、前のオペランドのデータ要素の２つと新しい第３のデータ要素を、テストストリングと比較してもよい。

【００７３】

一実施形態では、アプリケーションに応じて、結果アレイのエントリーを反転、または否定してもよい。他の実施形態では、結果エントリーの一部のみを、例えば２つのオペランドのデータ要素間の有効な一致に対応するものだけを否定（negate）する。他の実施形態では、他の演算を結果アレイ５２０の結果エントリーに実行してもよい。例えば、実施形態によっては、結果アレイ５２０はマスク値として表される。他の実施形態では、結果アレイはインデックス値で表され、レジスタなどの記憶場所に記憶される。インデックスは、一実施形態では結果アレイのＭＳＢのグループにより表され、他の実施形態ではアレイのＬＳＢで表される。一実施形態では、インデックスは、設定されているＬＳＢまたはＭＳＢへのオフセット値により表される。マスクは、一実施形態ではゼロ拡張であり、他の実施形態ではバイト／ワードマスク、またはその他の粒度（granularity）である。

【００７４】

様々な実施形態では、ＳＩＭＤオペランドの各要素の比較する際の上記の各相違は、個々の命令として実行される。他の実施形態では、上記の相違は、命令に付随するフィールド（immediate fields）などの単一の命令の属性を変えることにより実行され得る。図６は、１つまたは複数の命令により実行される、２つまたはそれ以上のＳＩＭＤオペランドの各データ要素を比較する様々な動作を示す図である。一実施形態では、図６の動作により比較されるオペランドはテキストストリングである。他の実施形態では、オペランドはその他のデータ情報やデータである。

【００７５】

図６を参照して、動作６１０において、第１のＳＩＭＤオペランド６０１と第２のＳＩＭＤオペランド６０５の各要素を互いに比較する。一実施形態では、一方のオペランドは

ＸＭＭレジスタなどのレジスタに記憶され、他方のオペランドは他のＸＭＭレジスタまたはメモリに記憶されている。一実施形態では、比較のタイプは、図６に示した動作を実行する命令に対応するイミーディエイトフィールド（immediate field）により制御される。例えば、一実施形態では、２ビットのイミーディエイトフィールド（例えば、ＩＭＭ８［１：０］）を用いて、比較するデータ要素が符号付きバイトか、符号付きワードか、符号無しバイトか、符号無しワードか示す。一実施形態では、比較結果によりｉ×ｊアレイ（例えば、ＢｏｏｌＲｅｓ［ｉ，ｊ］）、またはｉ×ｊアレイの一部ができる。

【００７６】

動作６１３において、並行して、オペランド６０１と６０５がそれぞれ表すストリングの終わりを見つけて、オペランド６０１と６０５の各要素の有効性を判断する。一実施形態では、レジスタまたはメモリの記憶場所内の対応する１つまたは複数のビットを設定することにより、オペランド６０１と６０５の各要素の有効性を明示的に示す。一実施形態では、その１つまたは複数のビットは、オペランド６０１と６０５のＬＳＢの位置から始まる連続した有効データ要素（例えば、バイト）の数に対応する。例えば、オペランドのサイズにもよるが、ＥＡＸレジスタやＲＡＸレジスタなどのレジスタを用いて、第１のオペランドの各データ要素の有効性を示すビットを記憶する。同様に、オペランドのサイズによっては、ＥＤＸレジスタやＲＤＸレジスタなどのレジスタを用いて、第２のオペランドの各データ要素の有効性を示すビットを記憶する。他の実施形態では、オペランド６０１と６０５の各要素の有効性を、本開示ですでに説明した手段により、黙示的に示しても良い。

【００７７】

一実施形態では、動作６１５において、比較と有効性に関する情報を集約機能（aggregation function）により結合して、２つのオペランドの要素の比較結果を生成する。一実施形態では、集約機能を、２つのオペランドの要素の比較を実行する命令に付随するイミーディエイトフィールドにより決定する。例えば、一実施形態では、２つのオペランドのデータ要素が等しいか、２つのオペランドのデータ要素の範囲が等しいか、２つのオペランドの各データ要素が等しいか、オペランドの少なくともデータ要素の一部の並びが同じか、比較により示すかどうか、イミーディエイトフィールド（immediate field）が示す。

【００７８】

動作６２０において、一実施形態では、（例えば、ＩｎｔＲｅｓ１に記憶された）集約機能の結果をネゲートする。一実施形態では、イミーディエイトフィールドのビット（例えば、ＩＭＭ８［６：５］）により、集約機能の結果に実行するネゲート機能のタイプを制御する。例えば、イミーディエイトフィールドは、集約結果をまったくネゲート（negate）しない、集約機能の結果をすべてネゲートする、オペランドの有効要素に対応する集約結果のみをネゲートすることを示してもよい。一実施形態では、ネゲート演算の結果をアレイ（例えば、ＩｎｔＲｅｓ２アレイ）に記憶する。

【００７９】

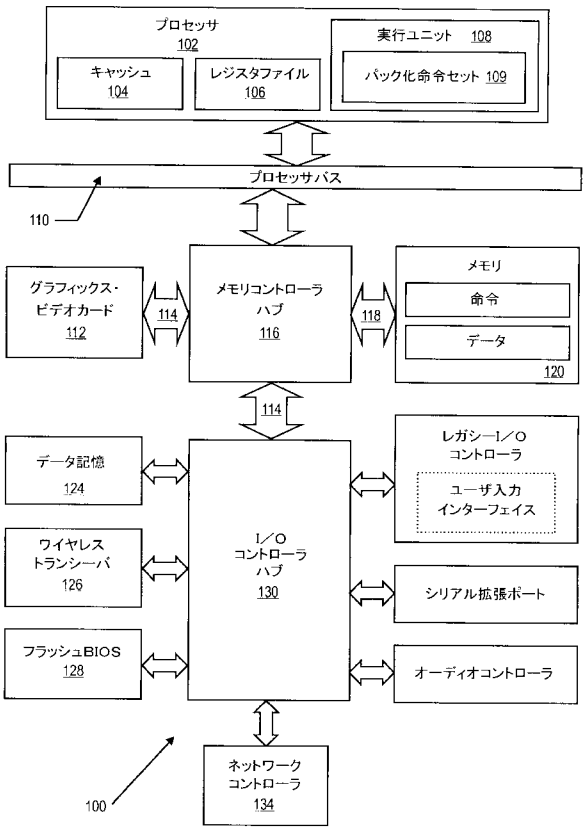
一実施形態では、それぞれ動作６２５と６３０において、ネゲート演算により生成される結果のアレイをインデックス値またはマスク値に変換する。ネゲート演算結果をインデックスに変換する場合、イミーディエイトフィールドのビット（例えば、ＩＭＭ８［６］）により、比較結果のＭＳＢまたはＬＳＢをインデックスにエンコードするかどうか、その結果をレジスタ（例えば、ＥＣＸまたはＲＣＸ）に記憶するかどうか制御する。一実施形態では、ネゲート演算の結果をマスク値で表す場合、イミーディエイトフィールドのビット（例えば、ＩＭＭ８［６］）を用いて、マスクをゼロ延長（zero-extended）拡張するか、バイト（またはワード）に拡張するか制御する。

【００８０】

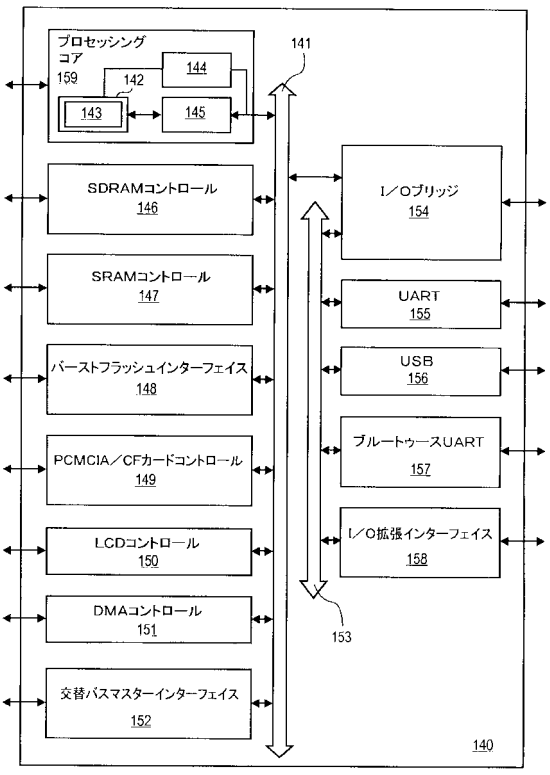
このように、ストリング比較演算の実行方法を開示する。実施形態の例を説明し、添付した図面に示したが、言うまでもなく、かかる実施形態は本発明の単なる例示であって制約するものではなく、本開示を研究すれば当業者には様々な修正に想到するので、本発明

は図示し説明した具体的な構成に限定はされない。本技術分野等では、成長が速く進歩が容易には予見できないので、本発明の原理や添付したクレームの範囲から逸脱することなく技術的な進歩を可能とすることにより容易になるので、開示の実施形態を構成と詳細において容易に修正できる。

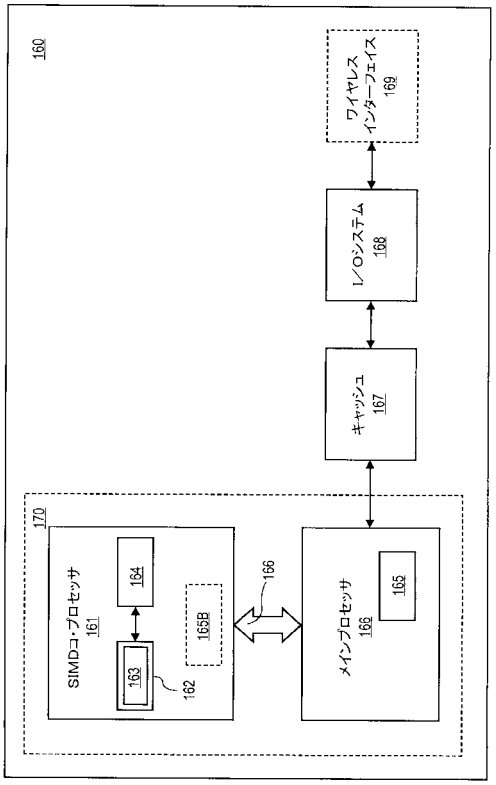
【図１Ａ】



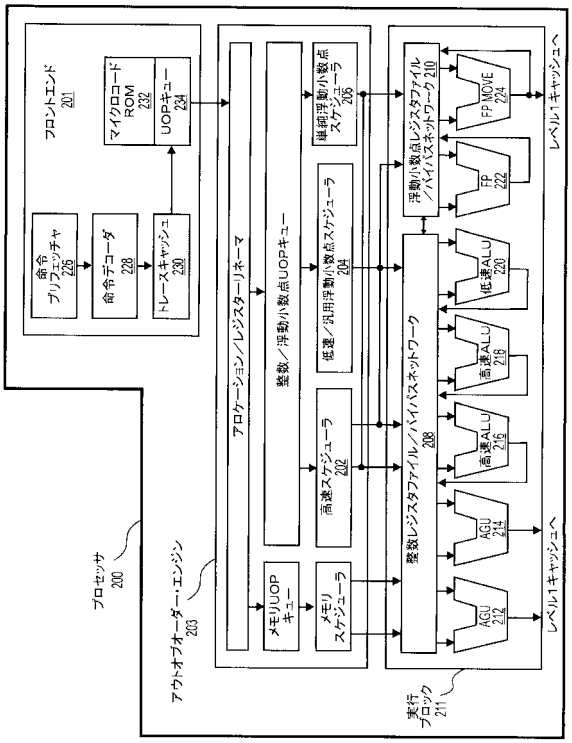
【図１Ｂ】



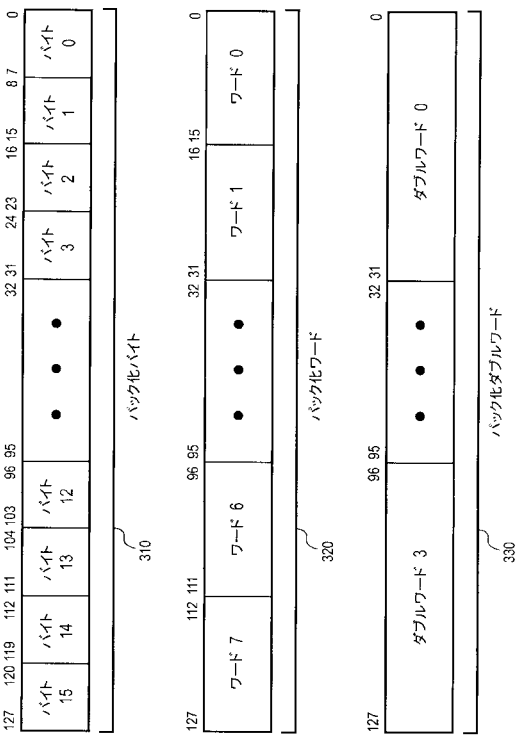
【図１Ｃ】



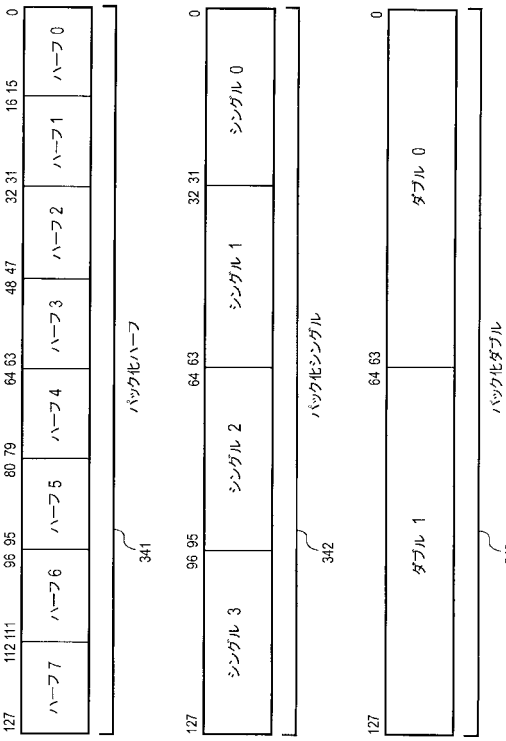
【図２】



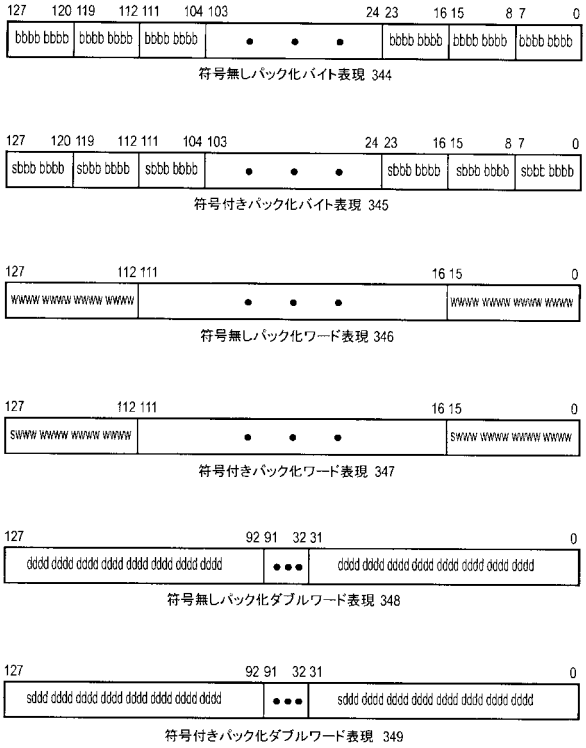
【図３Ａ】



【図３Ｂ】



【図３Ｃ】



【図３Ｄ】

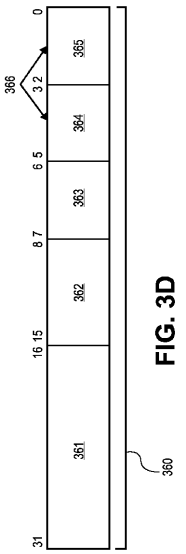


FIG. 3D

【図３Ｅ】

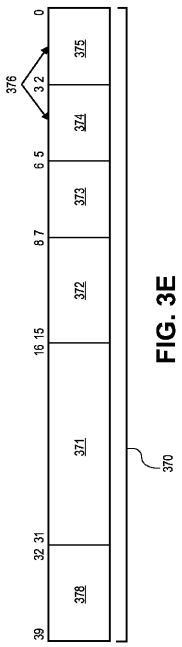


FIG. 3E

【図３Ｆ】

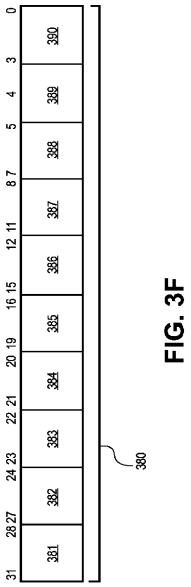
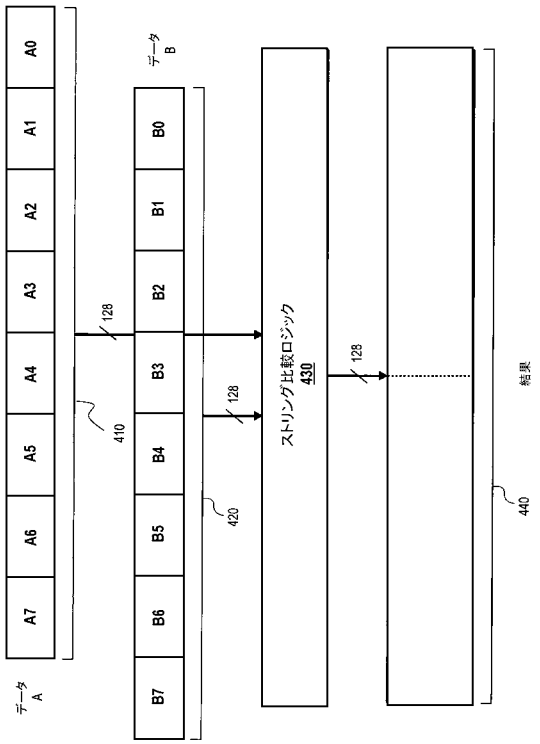
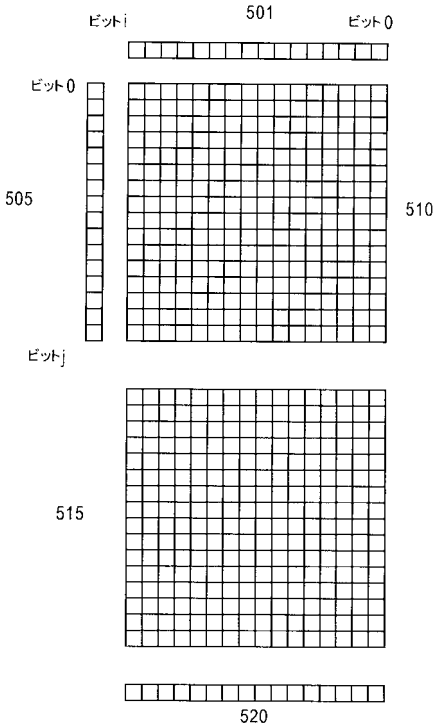


FIG. 3F

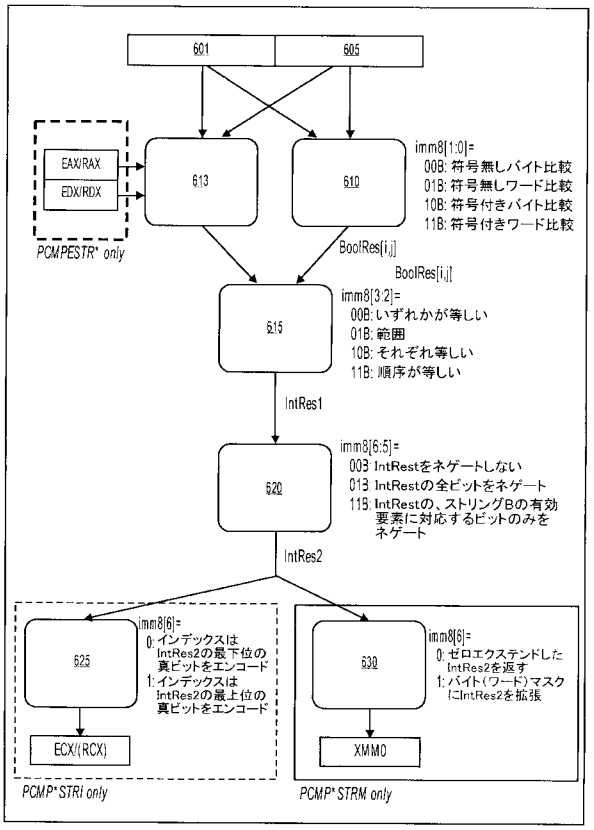
【図４】


A0
A1
A2
A3
A4
A5
A6
A7
データA
B0
B1
B2
B3
B4
B5
B6
B7
データB
128
410
128
420
ストリング比較ロジック
430
128
結果
440

【図５】


ビットi
501
ビット0
ビット0
505
510
ビットj
515
520

【図６】


601
605
EAX/RAX
EDX/RDX
PCMPESTR* only
613
610
imm8[1:0]=
00B: 符号無しバイト比較
01B: 符号無しワード比較
10B: 符号付きバイト比較
11B: 符号付きワード比較
BoolRes[i,j]
BoolRes[i,j]
615
imm8[3:2]=
00B: いずれかが等しい
01B: 範囲
10B: それぞれ等しい
11B: 順序が等しい
IntRes1
620
imm8[6:5]=
00B IntResをネゲートしない
01B IntResの全ビットをネゲート
11B: IntResの、ストリングBの有効要素に対応するビットのみをネゲート
IntRes2
625
imm8[6]=
0: インデックスはIntRes2の最下位の真ビットをエンコード
1: インデックスはIntRes2の最上位の真ビットをエンコード
ECX/(RCX)
PCMP*STRI only
630
imm8[6]=
0: ゼロエクステンドしたIntRes2を返す
1: バイト(ワード)マスクにIntRes2を拡張
XMM0
PCMP*STRM only

フロントページの続き

(72)発明者 グレイ，ジェフリー，ディー．
アメリカ合衆国 ９７２１２ オレゴン州 ポートランド ノースイースト ノット ストリート ３７１１
(72)発明者 チェヌパティー，スリニヴァス
アメリカ合衆国 ９７２２９ オレゴン州 ポートランド ノースウェスト ウィズマー ドライヴ １５７２７
(72)発明者 マーケス，ショーン，ピー．
アメリカ合衆国 ９７００７ オレゴン州 ビーヴァートン サウスウェスト コーモラント ドライヴ １６０５５
(72)発明者 セコニ，マーク，ピー．
アメリカ合衆国 ９７００６ オレゴン州 ビーヴァートン サウスウェスト エスチュアリ ドライヴ １６３８８ ナンバー１０６

審査官 三坂 敏夫

(56)参考文献 特表平１０－５１２０７０（ＪＰ，Ａ）
特開昭５８－１０６６３６（ＪＰ，Ａ）
国際公開第２００５／００６１８３（WO，Ａ１）
特表平１１－５１１５７５（ＪＰ，Ａ）
特開昭６２－２３３８８４（ＪＰ，Ａ）
特開平０６－１６２０６７（ＪＰ，Ａ）
特開平０１－２７１８７５（ＪＰ，Ａ）
特開昭６１－００７９７６（ＪＰ，Ａ）
国際公開第００／０３６５２７（WO，Ａ１）
特開２００４－１４５４９３（ＪＰ，Ａ）
インテル株式会社，「インテル・アーキテクチャ・ソフトウェア・ディベロッパーズ・マニュアル 中巻：命令セットリファレンス」，日本，ＣＱ出版株式会社，１９９７年，3-338～3-343
Alex Peleg et al., "Intel MMX for Multimedia PCs", Communications of the ACM , 米国, ACM, １９９７年 1月 1日, Volume 40 Issue 1, pages:25-38
インテル株式会社，「インテル・アーキテクチャ・ソフトウェア・ディベロッパーズ・マニュアル 上巻：基本アーキテクチャ」，日本，ＣＱ出版株式会社，１９９７年，2-5～2-12，8-1～8-15
Alex Peleg et al., "Intel MMX for Multimedia PCs", Communications of the ACM, 米国, ACM, １９９７年 1月 1日, Volume 40 Issue 1, pages:25-38
freescale semiconductor, "AltiVec TechnologyProgramming Interface Manual", 米国, freescale semiconductor, １９９６年 6月, Rev.0, pages:50,51,179,180,205,225, ［平成25年1月17日検索］インターネットＵＲＬ：http://www.freescale.com/files/32bit/doc/ref_manual/ALTIVECPIM.pdf

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ９／３０５
Ｇ０６Ｆ １５／８０