(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3771968号
(P3771968)
(45)発行日　平成18年5月10日(2006.5.10)　(24)登録日　平成18年2月17日(2006.2.17)

(51)Int.Cl.　F I
G06F 9/315 (2006.01)　G06F 9/30 340D

請求項の数 16 (全 31 頁)

(21)出願番号　特願平8-146619
(22)出願日　平成8年5月16日(1996.5.16)
(65)公開番号　特開平8-314717
(43)公開日　平成8年11月29日(1996.11.29)
審査請求日　平成15年5月16日(2003.5.16)
(31)優先権主張番号　9509987-5
(32)優先日　平成7年5月17日(1995.5.17)
(33)優先権主張国　英国(GB)

(73)特許権者　595008364
エスティーマイクロエレクトロニクス、リミテッド
ＳＴＭｉｃｒｏｅｌｅｃｔｒｏｎｉｃｓ Ｌｉｍｉｔｅｄ
イギリス国ブリストル、アーモンズベリー、アズテック、ウエスト、１０００
(74)代理人　100079049
弁理士　中島　淳
(74)代理人　100084995
弁理士　加藤　和詳
(74)代理人　100085279
弁理士　西元　勝一
(74)代理人　100099025
弁理士　福田　浩志

最終頁に続く

(54)【発明の名称】コンピュータ及びコンピュータ動作方法

(57)【特許請求の範囲】
【請求項１】
プロセッサと、
離散データの複数のサブストリングを有するデータストリングを処理するためのパック演算ユニットと、
を有するコンピュータであって、
前記パック演算ユニットは、入力バッファと、出力バッファと、を有し、
前記データストリングは、
同じビット長を有する第１サブストリング、
最終サブストリングおよび
少なくとも二つの中間サブストリング
を有し、
前記データストリングはオペランドを表し、
前記第１サブストリング、最終サブストリングおよび中間サブストリングは該オペランドのオブジェクトを表し、
前記コンピュータは少なくとも一つのデータストリング再構成インストラクションを含むインストラクションセットを有し、
少なくとも一つの前記再構成インストラクションのオブジェクトの各々のビット数を定義し、
前記プロセッサは、

前記入力バッファにデータストリングをロードし、
前記再構成インストラクションの指示にもとづき、前記入力バッファのサブストリングを対応する出力バッファの位置に出力し、または、該入力バッファの少なくとも二つの中間サブストリングの位置を互いに交換して、対応する出力バッファの位置に出力する、
コンピュータ。
【請求項２】
前記データストリングは偶数のサブストリングを含み、
前記データストリングの第１の半分を形成する前記サブストリングは第１シーケンスを形成し、
前記データストリングの第２の半分を形成する前記サブストリングは第２シーケンスを形成し、
前記プロセッサは、前記再構成インストラクションの指示にもとづいて、前記第１シーケンスから選択されたサブストリングと前記第２シーケンスから選択されたサブストリングとの位置を交換して、対応する出力バッファの位置に出力する、
請求項１に記載のコンピュータ。
【請求項３】
前記プロセッサは、前記再構成インストラクションの指示にもとづいて、前記第１および第２シーケンスのサブストリングを１つ置きに配置して、対応する出力バッファの位置に出力する、請求項２に記載のコンピュータ。
【請求項４】
前記プロセッサは、前記再構成インストラクションの指示にもとづいて、位置を変更しないで、前記第１および最終サブストリングを対応する出力バッファの位置に出力し、残りのサブストリングを１つ置きに配置して、対応する出力バッファの位置に出力する、請求項２または３に記載のコンピュータ。
【請求項５】
前記プロセッサは、前記再構成インストラクションの指示にもとづいて、前記データストリングの半分の奇数番目のサブストリング間に該データストリングの他方の半分の奇数番目のサブストリングを挿入して、対応する出力バッファの位置に出力する、請求項１～４のいずれか一項に記載のコンピュータ。
【請求項６】
前記入力バッファおよび出力バッファは各々が単一のアドレスによってアドレス可能な予め決められたビット容量を有する複数のレジスタ記憶装置である、請求項１～５のいずれか一項に記載のコンピュータ。
【請求項７】
データストリングは再構成インストラクションの実行の前に前記レジスタ記憶装置の一つに保持される、請求項６に記載のコンピュータ。
【請求項８】
データストリングは二つのレジスタ記憶装置の内容を有し、再構成インストラクションの実行の前に二つのレジスタ記憶装置に保持される、請求項６に記載のコンピュータ。
【請求項９】
プロセッサと、
プログラムメモリと、
実行ユニットと、
複数のレジスタ記憶装置と、
を有するコンピュータであって、
前記データ記憶回路の各々は、データストリング内の離散データのサブストリングを形成するビットシーケンスを保持する、単一のアドレスによって指定することができる、所定のビット容量を有し、
前記データストリングはオペランドを表し、前記サブストリングは該オペランドのオブジェクトを表し、

前記コンピュータはインストラクションシーケンスを前記プログラムメモリに記憶し、
前記インストラクションシーケンスは、
結果としてオペランドを生成するために、データストリング内の前記サブストリングの識別なしで、二つのレジスタ記憶装置のオペランドのオブジェクトへの共通動作を指定する第１インストラクションと、
データストリングを再構成するために、少なくとも一つのオブジェクトの各々のビット数を定義し、前記オペランドのデータストリング内の別個のサブストリングを識別し、少なくとも一つのサブストリングを変わらない位置で保持しつつ、データストリング内の少なくとも二つのサブストリングの位置を互いに交換する第２インストラクションと、
データストリング内の前記サブストリングの識別なしで、再構成されたデータストリングのサブストリングにさらに共通動作を指定する第３インストラクションと、
を含む、
コンピュータ。
【請求項１０】
前記第２インストラクションは前記データストリングにおいて前記データストリングの第１および最終サブストリングを変わらない位置で保持する、請求項９に記載のコンピュータ。
【請求項１１】
離散データを表し、データストリングの複数のサブストリングの一つを形成する複数のビットシーケンスを記憶し、
サブストリングの各々は、複数のビットを含み、
前記データストリングはオペランドを表し、
前記サブストリングは該オペランドのオブジェクトを表し、
結果としてオペランドを生成するために、データストリングの別個のサブストリングを識別せずに、２つのレジスタ記憶装置のオペランドのオブジェクトに共通の動作をもたらす第１のインストラクションを含むインストラクションシーケンスを実行し、
データストリングを再構成するために、少なくとも一つの前記再構成インストラクションのオブジェクトの各々のビット数を定義し、前記オペランドのデータストリングの別個のサブストリングを識別し、少なくとも１つのサブストリングの位置を変えずに、データストリングの少なくとも２つのサブストリングの位置を互いに交換する第２のインストラクションを実行し、
データストリングのサブストリングを識別せずに、再構成されたデータストリングのサブストリングに共通の動作をもたらす第３の命令を実行する、
コンピュータ動作方法。
【請求項１２】
各々が離散データを表し、データストリングの複数のサブストリングの一つを形成する複数のビットシーケンスを一以上のレジスタ記憶装置に記憶し、
サブストリングの各々は、複数のビットを含み、
前記データストリングはオペランドを表し、
前記サブストリングは該オペランドのオブジェクトを表し、
前記サブストリングは第１のサブストリングと、最終サブストリングと、少なくとも２つの中間サブストリングと、を含み、
サブストリングの各々のビット長は等しく、
前記レジスタ記憶装置の各々は、単一のアドレスで指定することができる所定のビット容量を有し、
少なくとも一つのデータストリング再構成インストラクションを含むインストラクションシーケンスを実行し、
前記少なくとも一つの再構成インストラクションは、オブジェクトの各々のビット数を定義し、
前記第１および最終サブストリングの位置を変えず、再構成データストリングの少なく

とも２つの中間サブストリングの位置を互いに交換するために、前記少なくとも１つの再構成インストラクションはデータストリングに動作する、
コンピュータ動作方法。
【請求項１３】
前記データストリングは前記データストリングの第１の半分を形成する第１ビットシーケンスおよび前記データストリングの第２の半分を形成する第２ビットシーケンスによって形成された偶数のサブストリングを含み、
前記再構成インストラクションが実行されて前記第１シーケンスから選択されたサブストリングと前記第２シーケンスから選択されたサブストリングとを交換する、
請求項１２に記載の方法。
【請求項１４】
前記再構成インストラクションが実行されて前記再構成データストリングにおいて前記第１および第２シーケンスからのサブストリングを１つ置きに配置する、請求項１３に記載の方法。
【請求項１５】
前記再構成インストラクションが実行されて各半分の１つ置きのサブストリングを変わらない位置で保持し、前記第１および最終サブストリングを変わらない位置で保持する請求項１３または１４に記載の方法。
【請求項１６】
前記再構成インストラクションが実行されてデータストリングの半分の少なくとも２つの奇数番目のサブストリング間に、前記データストリングの他方の半分からの奇数番目のサブストリングを挿入する、請求項１３～１５のいずれか一項に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明はデータの操作に関する。本発明は全面的という訳ではないが詳細にはコンピュータ及びコンピュータを動作させて所謂 "パックインストラクション（命令）”の実行に関連してデータ値の操作を実行する方法に関する。
【０００２】
【従来の技術及び発明が解決しようとする課題】
パックインストラクションはパックオペランドで実行される。パックオペランドは複数のサブストリングからなるビットストリングを有し、各サブストリングは特定のデータ値を画定し、本明細書中では "オブジェクト”と称される。従って、オペランドは複数のオブジェクトを有する。各オペランドはレジスタ記憶装置を有し、その記憶装置は単一のアドレスによってアドレス可能な予め決められたビット容量を有し、各ビット位置は個々にアドレスできない。従って、各オペランド内で個々のオブジェクトをアドレス且つ処理することはできない。
【０００３】
"パックインストラクション”によってオブジェクトを個々に識別せずにオペランド内の全てのオブジェクトに対して共通の動作が実行される。単一の共通動作はオペランドをメモリの中へ及び該メモリの外へロードすることである。
【０００４】
別の "パックインストラクション”は演算インストラクションであり、このインストラクションは個々のオペランドからのオブジェクトのペアに対して同じ演算動作を並列実行し、複数の結果オブジェクトを有するパック結果オペランドを提供する。パックインストラクションの実行のために、コンピュータはパック演算ユニットを提供し、このユニットは少なくとも一つがパックされている二つのソースオペランドに動作してパック結果を生成する。
【０００５】
メモリへのローディング及び記憶動作が減少し、各レジスタを埋めることによって使用可

能なレジスタ容量を最大にすることができるため、単一のオペランド内のオブジェクトのセットを一緒に処理することが有益であるのは明らかである。

【０００６】

しかしながら、オペランド内のオブジェクトのシーケンスはオブジェクトがメモリに記憶される順序によって予め決められているという不利益を有する。このシーケンスはメモリからオブジェクトを検索してそのオブジェクトを別個のレジスタに一時的に記憶し、異なる位置のメモリに再度書き込むことによってのみ変えることが可能である。これはマトリックス操作に特に要求される。この性質の動作はメモリへの繰り返しのアクセス及び長いインストラクションシーケンスを必要とする。インストラクションシーケンスはメモリの空間を占有する。インストラクションシーケンスの長さをできるだけ短くすることが所望される。メモリアクセスは動作を遅くするため、このアクセスを最小化することも所望される。

【０００７】

【課題を解決するための手段】

本発明の一態様に従うと、コンピュータが提供されており、そのコンピュータは、プロセッサ及び離散データの複数のサブストリングを有するデータストリングを処理するためのデータ記憶回路を有し、前記サブストリングはそれぞれ同じビット長を有する第１サブストリング、最終サブストリング及び少なくとも四つの中間サブストリングを有し、前記コンピュータは少なくとも一つのデータストリング再構成インストラクションを含むインストラクションセットを有し、前記再構成インストラクション又は各再構成インストラクションがデータストリングに実行されて前記第１サブストリング及び最終サブストリングを変わらない位置で保持し、再構成データストリングにおいて少なくとも二つの中間サブストリングの位置が互いに交換される。

【０００８】

従って単一の再構成インストラクションの実行によってオペランド内の二つのオブジェクトの配置をスワップすることが可能である。

【０００９】

この性質を有する本明細書中で述べられるインストラクションの主なカテゴリーは "フリップ”インストラクションである。

【００１０】

他に二つの画定されるインストラクションのカテゴリーがあり、それらは "ジップ”及び "アンジップ”インストラクションである。これらのインストラクションのいくつかもペアのオブジェクトのオペランド内で互いに位置をスワップする性質を有する。

【００１１】

以下の記述から明らかになるように、再構成インストラクションは単一のレジスタ記憶装置の内容又は二つのレジスタ記憶装置の内容に対して実行され、二つのレジスタ記憶装置の内容がデータストリングを画定する。

【００１２】

インストラクションが単一のレジスタ記憶装置の内容に関して実行されると、幾つかのインストラクションに対してデータストリングが考慮され、このインストラクションは前記データストリングの第１の半分を形成する第１ビットシーケンス及び前記データストリングの第２の半分を形成する第２ビットシーケンスによって形成された偶数のサブストリングを含む。再構成インストラクションが実行されて前記第１シーケンスから選択されたサブストリングと前記第２シーケンスから選択されたサブストリングとを交換する。或いは、各シーケンスの二つの中間サブストリングの位置を交換することもできる。第１シーケンス及び第２シーケンスは別個のレジスタ記憶装置に選択的に保持される。

【００１３】

再構成インストラクションが実行されて前記再構成データストリングにおける前記第１及び第２シーケンスからサブストリングを１つ置きに配置することができる。

【００１４】

再構成インストラクションが実行されてデータストリングの各半分の１つ置きのサブストリングを変わらない位置で保持し、前記第１及び最終サブストリングを変わらない位置で保持する。

【００１５】

再構成インストラクションが実行されて半分のデータストリングの奇数番目のサブストリング間に他方の半分のデータストリングの奇数番目のサブストリングを挿入することができる。

【００１６】

本発明はコンピュータを動作させる方法を提供し、その方法は、離散データを表しデータストリングの複数のサブストリングの一つを形成する複数のビットシーケンスをデータ記憶装置に保持し、各サブストリングはそれぞれ同じビット長を有する第１サブストリング、最終サブストリング及び少なくとも四つの中間サブストリングを含み、更に少なくとも一つのデータストリング再構成インストラクションを含むインストラクションシーケンスを実行し、再構成インストラクション又は各再構成インストラクションがデータストリングに実行されて前記第１及び最終サブストリングを変わらない位置で保持し、再構成されたデータストリングにおいて少なくとも二つの中間サブストリングの位置を交換する。

【００１７】

本発明はコンピュータを動作させる方法を提供し、その方法は、離散データを表しデータストリングの複数のサブストリングの一つを形成する複数のビットシーケンスをデータ記憶装置に保持し、インストラクションシーケンスを実行し、そのインストラクションシーケンスは、データストリング内の個々のサブストリングを識別せずに前記データストリングのサブストリングに共通動作を実行する第１インストラクション、データストリング内の個々のサブストリングを識別して少なくとも一つのサブストリングを変わらない位置で保持しつつデータストリング内の少なくとも二つの中間サブストリングの位置を交換してデータストリングを再構成する第２インストラクション、更にデータストリング内の前記サブストリングを識別せずに再構成されたデータストリングのサブストリングに共通動作を実行する第３インストラクションを含む。

【００１８】

この方法において、インストラクションのシーケンスが実行され、そのシーケンスにおいて、第１インストラクションはオペランド内の個々のオブジェクトを識別せず、第２インストラクションはオペランド内のオペランドを再配置し、第３インストラクションは更に共通動作を実行する。これはパック演算環境で動作を実行するのに特に有益である。

【００１９】

本発明はコンピュータを提供し、コンピュータはプロセッサ、メモリ及びデータストリング内の離散データのサブストリングを形成するビットシーケンスを保持するデータ記憶回路を有し、前記コンピュータはインストラクションシーケンスを前記メモリに記憶し、そのインストラクションシーケンスはデータストリング内の前記サブストリングの識別なしで前記データストリングのサブストリングに共通動作を指定する第１インストラクション、データストリング内の個々のサブストリングを識別して少なくとも一つのサブストリングを変わらない位置で残しつつデータストリング内の少なくとも二つのサブストリングの位置を互いに交換してデータストリングを再構成する第２インストラクション、及びデータストリング内の前記サブストリングの識別なしで再構成データストリングのサブストリングに更に共通動作を実行する第３インストラクションを含む。

【００２０】

本発明の再構成インストラクションはマトリックス操作を実行する場合に特に有益である。従って、本発明はコンピュータシステムを動作させてマトリックス転置動作を実行する方法を更に提供しており、その方法は、各々がマトリックスの行及び列の位置のデータ値を表す複数のサブストリングからデータストリングを形成し、本発明の再構成インストラクションを使用してデータストリングを再構成して選択された行及び列の値の交換を実行する。

【００２１】
本発明の請求項１の態様では、コンピュータであって、プロセッサ及び離散データの複数のサブストリングを有するデータストリングを処理するためのデータ記憶回路を有し、前記サブストリングは同じビット長を有する第１サブストリング、最終サブストリング及び少なくとも四つの中間サブストリングを有し、前記コンピュータは少なくとも一つのデータストリング再構成インストラクションを含むインストラクションセットを有し、前記再構成インストラクション又は各再構成インストラクションがデータストリングに実行されて前記第１及び最終サブストリングを変わらない位置で保持し、再構成されたデータにおいて少なくとも二つの中間サブストリングの位置を互いに交換する、ことを含む。
【００２２】
本発明の請求項２の態様では、請求項１の態様において、前記データストリングは前記データストリングの第１の半分を形成するビットの第１シーケンス及び前記データストリングの第２の半分を形成するビットの第２シーケンスによって形成された偶数のサブストリングを含み、前記再構成インストラクションが実行されて前記第１シーケンスから選択されたサブストリングと前記第２シーケンスから選択されたサブストリングを交換する、ことを含む。
【００２３】
本発明の請求項３の態様では、請求項２の態様において、前記再構成インストラクションが実行されて前記再構成データストリングにおいて前記第１及び第２シーケンスからのサブストリングを１つ置きに配置する、ことを含む。
【００２４】
本発明の請求項４の態様では、請求項２又は３の態様において、前記再構成インストラクションが実行されて前記データストリングの各半分の１つ置きのサブストリングを変わらない位置で保持し、変わらない位置の前記第１及び最終サブストリングを保持する。
【００２５】
本発明の請求項５の態様では、請求項１乃至４の態様において、前記再構成インストラクションが実行されてデータストリングの半分の奇数番目のサブストリング間に前記データストリングの他方の半分の奇数番目のサブストリングを挿入する、ことを含む。
【００２６】
本発明の請求項６の態様では、前述の請求項のいずれかの態様において、データ記憶回路は各々が単一のアドレスによってアドレス可能な予め決められたビット容量を有する複数のレジスタ記憶装置を有する。
【００２７】
本発明の請求項７の態様では、請求項６の態様において、データストリングは再構成インストラクションの実行の前に前記レジスタ記憶装置の一つに保持される。
【００２８】
本発明の請求項８の態様では、請求項６の態様において、データストリングは二つのレジスタ記憶装置の内容を有し再構成インストラクションの実行の前に二つのレジスタ記憶装置に保持される。
【００２９】
本発明の請求項９の態様では、コンピュータであって、前記コンピュータはプロセッサ、メモリ及びデータストリング内の離散データのサブストリングを形成するビットシーケンスを保持するデータ記憶回路を有し、前記コンピュータはインストラクションシーケンスを前記メモリに記憶し、そのインストラクションシーケンスはデータストリング内の前記サブストリングの識別なしで前記データストリングのサブストリングに共通動作を指定する第１インストラクション、データストリング内の別個のサブストリングを識別し、少なくとも一つのサブストリングを変わらない位置で保持しつつデータストリング内の少なくとも二つのサブストリングの位置を互いに交換してデータストリングを再構成する第２インストラクション及びデータストリング内の前記サブストリングの識別なしで再構成されたデータストリングのサブストリングに更に共通動作を指定する第３インストラクション

を含む。

【００３０】

本発明の請求項１０の態様では、請求項９の態様において、前記第２インストラクションは前記データストリングにおいて前記データストリングの第１及び最終サブストリングを変わらない位置で保持する。

【００３１】

本発明の請求項１１の態様では、コンピュータを動作させる方法であって、その方法は、離散データを表しデータストリングの複数のサブストリングの一つを形成する複数のビットシーケンスをデータ記憶装置に保持し、各サブストリングは同じビット長を有する第１サブストリング、最終サブストリング及び少なくとも４つの中間サブストリングを含み、更に少なくとも一つのデータストリング再構成インストラクションを含むインストラクションシーケンスを実行し、再構成インストラクション又は各再構成インストラクションが実行されて前記再構成データストリングの第１の半分を形成する第１ビットシーケンス及び前記再構成データストリングの第２の半分を形成する第２ビットシーケンスによって形成された偶数のサブストリングを含む再構成データストリングを生成し、前記再構成インストラクションが実行されて互いに隣接する原データストリングの第１サブストリング及び１つ置いた次のサブストリングを配置することによって第１シーケンスを生成し、互いに隣接する原データストリングの第２サブストリング及び１つ置いた次のサブストリングを配置することによって第２シーケンスを生成する。

【００３２】

本発明の請求項１２の態様では、コンピュータを動作させる方法であって、その方法は、各々が離散データを表しデータストリングの複数のサブストリングの一つを形成する複数のビットシーケンスをデータ記憶装置に保持し、前記サブストリングはそれぞれ同じビット長を有する第１サブストリング、最終サブストリング及び少なくとも二つの中間サブストリングを含み、更に少なくとも一つのデータストリング再構成インストラクションを含むインストラクションシーケンスを実行し、前記再構成インストラクション又は各再構成インストラクションがデータストリングに実行されて第１及び最終サブストリングを変わらない位置で保持し、再構成されたデータストリングにおいて少なくとも二つの中間サブストリングの位置を互いに交換する、ことを含む。

【００３３】

本発明の請求項１３の態様では、請求項１２の態様において、前記データストリングは前記データストリングの第１の半分を形成する第１ビットシーケンス及び前記データストリングの第２の半分を形成する第２ビットシーケンスによって形成された偶数のサブストリングを含み、前記再構成インストラクションが実行されて前記第１シーケンスから選択されたサブストリングと前記第２シーケンスから選択されたサブストリングを交換する、ことを含む。

【００３４】

本発明の請求項１４の態様では、請求項１３の態様において、前記再構成インストラクションが実行されて前記再構成データストリングにおいて前記第１及び第２シーケンスからのサブストリングを１つ置きに配置する、ことを含む。

【００３５】

本発明の請求項１５の態様では、請求項１３又は１４の態様において、前記再構成インストラクションが実行されて前記データストリングの各半分の１つ置きのサブストリングを変わらない位置で保持し変わらない位置の前記第１及び最終サブストリングを保持する。

【００３６】

本発明の請求項１６の態様では、請求項１３乃至１５の態様において、前記再構成インストラクションが実行されてデータストリングの半分の奇数番目のサブストリング間に前記データストリングの他方の半分からの奇数番目のサブストリングを挿入する、ことを含む。

【００３７】

【発明の実施の形態】
本発明をより理解し、どのように実行されるかを示すために、添付図面を例として参照される。
【００３８】
図１は本発明の第１実施の形態に従ったプロセッサを示す。プロセッサは三つの実行ユニットを有し、これらのユニットは従来通りの演算ユニット２及びメモリアクセスユニット４を有する。更に、パック演算ユニット６も有する。プロセッサはインストラクションフェッチャー８、インストラクションレジスタ１０、レジスタファイル１２及びインストラクションポインタ１４も有し、これらは全てプロセッサのコントロールユニット１６の制御によって動作する。レジスタファイルはレジスタのセットを有し、各レジスタは予め決められたビット容量を有し、単一のアドレスでアドレス可能である。レジスタ内の個々の位置にアドレスすることは出来ない。レジスタにアクセスすると、レジスタの全体的内容が考慮される。プロセッサは更に定数ユニット１８及び選択ユニット２０を有する。定数ユニット１８及び選択ユニット２０も制御ユニット１６の制御によって動作する。プロセッサはメモリ２２と協働して動作し、該メモリはプロセッサの動作を実行するためのインストラクション及びデータ値を保持する。データ値及びインストラクションはデータバス２４を介してメモリ２２へ及び該メモリ２２から与えられる。データバス２４はメモリデータ入力２６を介してメモリ２２へ及び該メモリ２２からデータ値を与える。データバス２４はフェッチャーデータ入力２８を介してインストラクションフェッチャー８にデータを与え、メモリアクセス読み取り入力３０を介してメモリアクセスユニット４にデータを与える。メモリは選択ユニット２０を介してアドレス入力３２でアドレスされる。選択ユニット２０は制御ユニット１６からのフェッチ信号３４を介して制御されてフェッチャー８からアドレス３６を選択するか又はメモリアクセスユニット４からアドレス３８を選択する。制御ユニット１６からの読み書き制御ライン４０、４２はメモリ２２から及び該メモリ２２への読み書き動作を制御する。インストラクションフェッチャー８は制御ユニット１６の制御によってメモリ２２からのインストラクションを以下のようにフェッチする。インストラクションが読み取られるアドレス３６が選択ユニット２０を介してメモリ２２に与えられる。これらのインストラクションはデータバス２４を介してフェッチャーデータ入力２８に与えられる。インストラクションフェッチャーが次にインストラクションをフェッチすると、又は任意のイベントにおいて次のインストラクションが実行可能になると、ライン４４を介してReady 信号が制御ユニット１６に出される。実行されるインストラクションはインストラクションラインInst４６に沿ってインストラクションレジスタ１０に与えられ、実行中該レジスタに保持される。インストラクションポインタ１４は、インストラクションポインタライン４８を介してフェッチャー８から与えられ実行されるインストラクションのアドレスを保持する。制御ユニット１６からのNewInst 信号５３に応答したGet 信号４７によって、インストラクションレジスタ１０はInstライン４６の次のインストラクションを記憶し、フェッチャー８は次のインストラクションの準備をする。NewInst 信号５３によってインストラクションポインタ１４は次のインストラクションのアドレスを記憶する。制御ユニット１１６からのブランチライン５０によってフェッチャー８はブランチを実行する。
【００３９】
インストラクションレジスタ１０は、ソース１及ソース２レジスタアドレスをReg1及びReg2としてレジスタファイルに提供する。結果レジスタアドレスはDestとして提供される。オペコードはライン５１に沿って制御ユニット１６に与えられる。更に、幾つかのインストラクションは一つの又は両方のソースレジスタをエンコードする代わりに定数オペランドを与える。定数は定数ユニット１８によって提供される。インストラクションのソース値は、入力E1,E2 でS1Reg 及びS2Reg 信号を適切に設定することによってソース１バス５２、ソース２バス５４に与えられる。ライン５１のオペコードに従って制御ユニット１６からのPack Ops、Mem Ops 及びALU Ops の適切な値を提供することによって正確な実行ユニットがイネーブルとなる。イネーブルになったユニットは普通結果バス５６に結果Res

を与える。これは普通レジスタファイル１２の選択された結果レジスタDestに記憶される。これには幾つかの例外がある。

【００４０】

幾つかのインストラクションはダブル長結果を提供する。これらのインストラクションは結果の第１部分を通常の方法で記憶する。連続する追加の段階では、結果の第２部分はDouble信号５８を表すことによってレジスタファイル１２の次のレジスタに記憶される。

【００４１】

Branch５０にはインストラクションポインタ１４を読み取り且つ調節することが要求される。これらのBranchによってS1Reg 信号は表されないため、インストラクションポインタ１４はライン６０にソース１の値を提供する。ソース２の値は通常の方法で（レジスタファイル１２のレジスタからか又は定数ユニット１８から）与えられる。演算ユニット２はブランチ演算を実行し、その結果はレジスタファイル１２ではなくNew IP入力６４でフェッチャー８に記憶されて制御ユニット１６からのBranchライン５０によって信号が送られる。これによって新しいアドレスからフェッチャーがスタートする。

【００４２】

条件ライン６２の状態に応じて二つの段階で条件ブランチが実行されなければならない。第１段階では、ReadDest信号４５を表すことによって別のソースとしてDestレジスタを使用する。条件が満たされると、次に通常のブランチソースオペランドが読み取られブランチが実行される。

【００４３】

コールはリターンアドレスをセーブしなければならない。これは、ブランチターゲットを計算する前にデスティネーションレジスタにインストラクションポインタ値を記憶することによって行われる。

【００４４】

本明細書で述べられるコンピュータは幾つかの重要な性質を有する。

【００４５】

ソースオペランドは常に自然ワード長である。一つ、二つ又は三つのソースオペランドがありうる。

【００４６】

結果は常に自然ワード長であるか又は自然ワード長の２倍の長さである。自然ワード長の２倍の長さである場合は、一つではなく二つのレジスタに記憶し、二つのレジスタを占有する特別の段階を踏むため動作ペナルティがある。このコンピュータでは、６４ビットの自然ワード長とする。つまり、レジスタファイルの各レジスタは６４ビットの予め決められた容量を有する。

【００４７】

実行ユニット２、４、６はインストラクション実行間に何らの状態も保持しない。従って、連続するインストラクションは独立的である。

【００４８】

アンパックインストラクション

演算ユニット２及びメモリアクセスユニット４は制御ユニット１６と共に従来通りのインストラクションセットの以下のインストラクションを実行することができる。以下の定義において、レジスタは、当該技術分野の技術者には公知であるように、レジスタの中身及び記憶位置としてのレジスタ自体を示すために使用される。

mov　定数又はレジスタをレジスタに移動する。

add　２個のレジスタを加算して結果を第３レジスタ（ソースのいずれかと同様である）に記憶する。

sub　２個のレジスタを減算し結果を第３レジスタに記憶する。

load　一つのレジスタをアドレスとして使用してメモリの位置から読み取り結果を別のレジスタに記憶する。

store　一つのレジスタをアドレスとして使用し、別のレジスタの中身をアドレスによっ

て特定された位置でメモリに記憶する。
compe　２個のレジスタ（又はレジスタと定数）が等しいかを比較する。等しい場合、１をデスティネーションレジスタに記憶し、そうでない場合はゼロを記憶する。
compge　２個のレジスタ（又はレジスタと定数）の順序性を比較する。２番目が１番目より小さくなければ、デスティネーションレジスタに１を記憶し、そうでない場合はゼロを記憶する。
jump　　新しい位置へ無条件にジャンプする。
jumpz　特定のレジスタの中身がゼロである場合、新しいプログラム位置にジャンプする。
jumpnz　　特定のレジスタの中身がゼロでなければ新しいプログラム位置にジャンプする。
shr　　定数又は別のレジスタによってレジスタのビット毎の右シフトを実行する。符号ビットはシフトの際複製されるため、シフトは符号が付けられる。
shl　　定数又は別のレジスタによってレジスタのビット毎の左シフトを実行して結果をデスティネーションレジスタに記憶する。
or/xor　２個のレジスタでビット毎のロジック動作（or/xor）を実行し、結果をデスティネーションレジスタに記憶する。

【００４９】

パックユニット

図２はパック演算ユニット６のブロック図を示す。これは別個のユニットの集合として示され、各々はパック演算インストラクションの幾つかのサブセットに対して応答可能である。別の実施では異なる方法で機能を組み合わせることがあり得るかもしれない。このユニットはバイト複製ユニット７０、ツイスト及びジップユニット７４、明確なパック演算ユニット８０及び本明細書では説明されないが他のパック演算ユニット７２、７６、７８を含む。これらは演算ユニット７０乃至８０を選択的に制御するルートオペコードユニット８２に応答して動作する。演算ユニット７０乃至８０のオペランドはソース１バス５２、ソース２バス５４に沿って与えられる。演算ユニットからの結果は結果バス５６に与えられる。ルートオペコードユニット８２へのOP入力は制御ユニット１６（図１）からのPack Opsインストラクションを受ける。ソース１バス及びソース２バスに与えられるオペランドは演算ユニットのそれぞれの入力バッファにロードされ、結果は一つ又は二つの出力バッファからレジスタファイル１２の一つ又は二つのディスティネーションレジスタに与えられる。

【００５０】

明確なパック演算

明確なパック演算ユニット８０は、二つのソースオペランドを各々が幾つかのパックオブジェクトを含むものとして扱い、二つのオペランドのオブジェクトの各ペアに動作して各ソースと同じ数のパックオブジェクトを含む結果を生成する。サポートされる動作は、加算、減算、比較、乗算、左シフト、右シフト等である。上記に説明されたように、単一のアドレスを使用してレジスタをアドレスすることによって、オペランドがアクセスされる。オペランドは個々にアドレスできない複数のオブジェクトを有する。

【００５１】

図３はパック演算ユニット６の演算ユニットを例示した図において使用される記号を示す。

【００５２】

図４は、パック１６ビット数の加算、減算、比較及び乗算を実行することができる明確なパック演算ユニットを示す。この場合、ソース及び結果バスの幅は６４ビットであるため、４つのパックオブジェクトがあり、それぞれは各バスで１６ビットの長さである。

【００５３】

明確なパック演算ユニット８０は４つの演算ロジカルユニットALU0-ALU3 を有し、各ユニットはライン１００のオペコードによって制御され、このラインは図２のルートオペコー

ドユニット８２から導出される。ソースレジスタ１SRC1から与えられた６４ビットワードは４つのパックオブジェクトS1[0]-S1[3] を含む。ソースレジスタ２SRC2から与えられた６４ビットワードは４つのパックオブジェクトS2[0]-S2[3] を含む。これらは第１入力バッファ９０及び第２入力バッファ９２に記憶される。第１演算ロジックユニットALU0は各オペランドの第１パックオブジェクトS1[0],S2[0] に動作して結果R[0]を生成する。第２乃至第４演算ロジックユニットALU1-ALU3 も同様にオブジェクトの第２ペア乃至第４ペアを使用してそれぞれ結果R[1]乃至R[3]を生成する。これらは結果バッファ１０２に記憶される。従って、結果ワードは４つのパックオブジェクトを含む。イネーブルユニット１０１はユニットのうちのいくつがアクティブであるべきかを決定し、出力バッファがその出力を表すかどうかを制御する。

【００５４】

インストラクションは以下の通りに表される。

add2p　各S1[i] にその２の補数S2[i] を加えてR[i]を生成する。オーバーフローは無視される。

sub2p　各S2[i] の２の補数S1[i] から各S2[i] を減算してR[i]を生成する。オーバーフローは無視される。

cmpe2p　S1[i] とS2[i] を比較する。等しい場合はR[i]を全て１に設定し、異なる場合はR[i]をゼロに設定する。

cmpge2ps　S1[i] と符号が付けられたその２の補数のS2[i] を比較する。S1[i] がS2[i] より大きいかS2[i] に等しい場合はR[i]を全てゼロに設定し、S1[i] がS2[i] より小さければR[i]をゼロに設定する。

mul2ps　S1[i] とその符号付き２の補数のS2[i] を乗算してR[i]をフル（３２ビット）プロダクトの最下位１６ビットに設定する。

【００５５】

幾つかの明確なパック演算インストラクションは、普通一つのパックソースオペランド及び一つのアンパックソースオペランドを使用する。図５はそのようなユニットを示す。

【００５６】

図５のパック演算ユニットの中身は、実質的には図４の中身と同様である。唯一異なる点は、第２ソースオペランドの入力バッファ９２’はアンパック形態のソースオペランドを受けることである。入力バッファ９２’は前のようにパック形態の第１ソースオペランドを受ける。アンパックソースオペランド及びパックソースオペランドを使用したインストラクションの一つの例は、シフトインストラクションであり、シフトの量はパックされないため、全てのパックオブジェクトに対して同じシフトが与えられる。シフト量はアンパックである必要はないが、この方がより有益である。

shl2p　S1[i] をS2（パックされていない）だけ左にシフトしてR[i]を結果に設定する。

shr2ps　各S1[i] をS2（パックされていない）だけ右にシフトしてR[i]を結果に設定する。シフトの際符号ビットが複製されるため、シフトは符号付きである。

【００５７】

同じセットの動作がパック８ビット及びパック３２ビットオブジェクトに与えられることが予想される。このインストラクションは類似した名前を有するが、"２”を "１”又は "４に置き換えたものである。

【００５８】

バイト複製

図６はバイト複製ユニット７０を示す。バイト複製ユニットは入力バッファ１０４を有し、このバッファは単一のオペランドを受け、このオペランドは図６では８つのパック８ビットオブジェクトS[0]乃至S[7]を含む６４ビットワードとして例示される。第１マルチプレクサ１０６は入力として第１オブジェクトS[0]及び第２オブジェクトS[1]を受ける。第２マルチプレクサ１０８は入力として第１オブジェクトS[0]及び第３オブジェクトS[2]を受ける。第３マルチプレクサ１１０は入力として第１マルチプレクサ１０８の出力及び第４オブジェクトS[3]を受ける。バイト複製ユニットは出力バッファ１１２も有する。出力

バッファは８つの８ビットオブジェクトR[0]乃至R[7]としてパックされた６４ビットワードを保持する。出力バッファ１１２の第１及び第５の８ビット位置は入力バッファ１０４の第１の８ビットに直接接続する。出力バッファ１１２の第２及び第６の８ビット位置は、第１マルチプレクサ１０６の出力を受けるように接続される。出力バッファ１１２の第３及び第７の８ビット位置は第２マルチプレクサ１０８の出力を受けるように接続される。出力バッファ１１２の第４及び第８の８ビット位置は第３マルチプレクサ１１０の出力を受けるように接続される。出力バッファの８ビット結果オブジェクトはR[0]乃至R[7]と称される。タイプユニット１１４は図２のルートオペコードユニット８２から導出されたライン１１８のオペコードを受ける。タイプユニットは複製されるオブジェクトのサイズを選択して３つの出力信号Do8,Do16,Do32 の内の一つを与える。これらの出力信号はORゲート１２０に与えられる。ORゲートの出力によって出力バッファ１１２がイネーブルとなる。Do16及びDo32信号は第２ORゲート１２２への入力であり、その出力は第１マルチプレクサ１０６を制御する。Do32信号自体が第２マルチプレクサ１０８及び第３マルチプレクサ１１０を制御する。従って、バイト複製ユニットはソースオペランドの最下位オブジェクト（８、１６、又は３２ビット）を使用してそれを８、４又は２倍に複製して出力バッファ１１２に保持されるパック６４ビット結果を生成する。動作は８ビットピースに分解され、S[i]及びR[i]の各々は８ビットである。幾つかのロジックが異なる複製において共有される。タイプユニット１１４は１６ビットシーケンス又は３２ビットシーケンスのどちらを複製するかを決定する。Do16又はDo32のどちらの信号も表されない場合、８ビットシーケンスが複製される。

【００５９】

バイト複製ユニットによってサポートされる三つのインストラクションは以下の通りである。 replp S[0]をR[0]乃至R[7]のそれぞれに複製する。

rep2p S[0]及びS[1]をi が０から３までのR[2i] 及びR[2i+1] に複製して１６ビットを複製する。

rep4p S[0]及びS[3]をi が０から１までのR[4i] からR[4i+3] に複製して３２ビットを複製する。

【００６０】

ツイスト及びジップ

ツイスト及びジップユニット７４によって実行される再構成動作には三つの種類がある。それらは以下の通りである。

shuffle (zip) オブジェクトストリングのペアからなるソースストリングを使用してオブジェクトストリングのペアからオブジェクトをインターリーブしてソースストリングと同じ長さの単一のストリングを生成する。これはパーフェクトシャッフルである。

Sort (unzip) オブジェクトペアを含むソースストリングを使用してこのペアをデインターリーブしてその結果デインターリーブされたペアの連結からなるストリングを生成する。これはパーフェクトソートである。

Transpose (flip) ４オブジェクトを含むソースストリングを使用し、適切なソースオブジェクトを交換することによって結果ストリングを生成し、マトリックス転置のセットを実行する。

【００６１】

これらの動作のうちの任意の一つは他の二つの動作の適切な組み合わせによって構成されることができる。

【００６２】

これら全ての変換に対してソースストリングは複数のベクトルからなり、各ベクトルは同じサイズのオブジェクトを同じ数だけ含む。これらの変換を挙げるためには３つの数字が必要である。

number of vectors ソース及び結果ストリングのベクトル数を特定する。

size of vector 各ベクトルのオブジェクトの数を特定する。

size of object 各オブジェクトのビット数を特定する。

【００６３】
インストラクションネームは変換タイプ（zip, unzip, flip）からなり、引き続きベクトルの数 "n"が付き、各ベクトルのサイズ "v"が付き、８ビットバイトの数として表されるオブジェクトサイズ "p"が付く。従って、インストラクションzip4n2vlp では、zip はインストラクションタイプを示し、4n2vlpはオペランドフォーマットを特定する。この場合、zip 動作は４ベクトルで実行され、それぞれは２つの１バイトオブジェクトである。この特定の動作を行うためには、各zip は２つのベクトルを要求するため、２つの別個のジップが実行される。
【００６４】
ソース及び結果ストリングが全体で６４又は１２８ビットである場合、図７に示されるような９個の固有ジップ及びアンジップ変換がある。
【００６５】
このジップ及びアンジップのセットはこの実施によってサポートされる６４及び１２８ビットストリングに対しては完全である。より長いストリングのジップ及びアンジップは従来のmoveインストラクションと協働したこれらのインストラクションのシーケンスによって実行することができる。
【００６６】
６４及び１２８ビットストリングに適切なflipは図８に示されている。これらのうちの幾つかは図７のジップ及びアンジップの幾つかと同様である。
【００６７】
ジップ及びアンジップと同様に、このフリップのセットも６４及び１２８ビットストリングに対して完全である。より長いストリングのフリップはフリップのシーケンス及び従来のmoveインストラクションによって実行可能である。
【００６８】
図９は６４ビットジップ及びアンジップを処理するツイスト及びジップユニット７４の部分を示す。図９に示されるツイスト及びジップユニットのジップ及びアンジップ部分は、S[0]からS[7]の８つのパック８ビットソースオブジェクトを含む入力バッファ１３０を含む。結果バッファ１３２はR[0]からR[7]の８個のパック８ビット結果オブジェクトを保持するために提供される。結果R[0]は第１ソースオブジェクトS[0]に直接接続する。第２ソースオブジェクトS[1]は第１入力として第１マルチプレクサ１３４、第２マルチプレクサ１３６及び第３マルチプレクサ１３８に与えられる。第１、第２及び第３マルチプレクサ１３４、１３６、１３８は、第２入力として第５ソースオブジェクトS[4]を受ける。第４マルチプレクサ１４０は一つの入力として第３ソースオブジェクトS[2]及び別の入力として第１マルチプレクサ１３４の出力を受ける。第４マルチプレクサの出力は第２結果オブジェクトR[1]に提供される。第２マルチプレクサ１３６の出力は第３結果オブジェクトR[2]を提供される。第５マルチプレクサ１４２は入力として第３マルチプレクサ１３８の出力及び第６ソースオブジェクトS[5]を受ける。第５マルチプレクサ１４２の出力は第４結果オブジェクトR[3]に与えられる。第６マルチプレクサ１４４は一つの入力として第４ソースオブジェクトS[3]及び別の入力として第７ソースオブジェクトS[6]を受ける。第６マルチプレクサの出力は、第７マルチプレクサ１４６の一つの入力として与えられ、該マルチプレクサ１４６の別の入力は第３ソースオブジェクトS[2]である。第７マルチプレクサ１４６の出力は第５結果オブジェクトR[4]に与えられる。第８マルチプレクサ１５０は一つの入力として第４ソースオブジェクトS[3]を受け、別の入力として第７ソースオブジェクトS[6]を受けて、出力を第６結果オブジェクトR[5]に与える。第９マルチプレクサ１５２は一つの入力として第４ソースオブジェクトS[3]を受け、別の入力として第７ソースオブジェクトS[6]を受ける。第９マルチプレクサ１５２の出力は第１０マルチプレクサ１５４に与えられ、該第１０マルチプレクサは第２入力として第６ソースオブジェクトS[5]を受ける。第１０マルチプレクサ１５４の出力は第７結果オブジェクトR[6]を提供する。第８ソースオブジェクトS7は第８結果オブジェクトR7を提供するように該結果オブジェクトに直接接続する。タイプユニット１６２は図２のルートオペコードユニット８２から導出

されたライン１６０のオペコードを受ける。タイプユニット１６２はツイスト及びジップユニット７４のジップ及びアンジップ部分で実行されるインストラクションを定める。この目的のために、該タイプユニットは４つの出力信号zip2n2v2p 、unzip2n4vlp 、zip2n4vlp 及びzip4n2vlp のうちの１つを与える。zip2n4vlp 及びzip4n2vlp 出力は第１ORゲート１６４に与えられ、その出力は第８マルチプレクサ１５０を制御する。出力信号zip4n2vlp は第２ORゲート１６６にも与えられ、このゲート１６６はunzip2n4vlp の出力を受ける。第２ORゲートの出力は第４、第５、第７及び第１０マルチプレクサを制御する。信号unzip2n4vlp は第３及び第６マルチプレクサを制御する。出力zip2n2v2p は第１及び第９マルチプレクサを制御する。タイプユニット１６２の４つ全ての出力は第３ORゲート１６８に与えられ、そのゲート１６８は出力バッファ１３２がイネーブルかどうかを決定する。図９においてロジックパスの幾つかは共有されるため、１０個の８ビットマルチプレクサしか必要としない。ソース及び結果はパック８ビットオブジェクトとして示される。しかしながら、ある一つのインストラクションではこの実施はパック１６ビットオブジェクトのみに画定され、これはソース及び結果８ビットオブジェクトのペアを使用することによって達成される。

【００６９】

６４ビットジップ及びアンジップは以下の通りである。

zip4n2v1p　２つの８ビットオブジェクトからなるベクトルをジップ（インターリーブ）する。これは同じベクトルのアンジップ（デインターリーブ）と同様である。

zip2n4v1p　４つの８ビットオブジェクトからなるベクトルをジップ（インターリーブ）する。

unzip1n4v1p ４つの８ビットオブジェクトからなるベクトルをアンジップ（デインターリーブ）する。

zip2n2v2p　２つの１６ビットオブジェクトからなるベクトルをジップ（インターリーブ）する。これは同じオブジェクトのアンジップ（デインターリーブ）と同様である。

【００７０】

図１０はダブル長８ビットジップ及びアンジップインストラクションを実行するツイスト及びジップユニットの部分を示す。このツイスト及びジップユニットの部分は第１及び第２入力バッファ１７０、１７２を有し、各バッファは６４ビットワードを保持する。入力バッファ１７０、１７２に保持される６４ビットワードは、S1[0] からS2[7] までラベル付けされた１６個のオブジェクトを有する連続データストリングとしてみなすことができる。第１及び第２出力バッファ１７４、１７６があり、それぞれ６４ビットワードを保持する。結果はライン１７８の出力である。６個の切り換えスイッチ１８０乃至１９０があり、それぞれ２つの入力及び２つの出力を有する。図１０に例示されるように、切り換えスイッチ１８０乃至１９０の入力は第１及び第２入力バッファ１７０、１７２の位置に接続する。図１０に例示されるように切り換えスイッチ１８０乃至１９０の出力は第１及び第２出力バッファ１７４、１７６の位置に接続する。図７に例示されたようなzip2n8v1p 動作又はunzip2n8v1p 動作のいずれかが実施されるように接続される。図１０から分かるように、第１入力バッファS1[0] の第１位置及び第２入力バッファS2[7] の最終位置はそれぞれ出力バッファの第１位置R[0]及び第２出力バッファの最終位置R[15] に接続する。このようにして、第１及び最終オブジェクトのデータストリングはジップ及びアンジップインストラクションに従ったデータストリングの再構成後も変化しないままである。タイプユニット１９２は図２のルートオペコードユニット８２から導出されたライン１６０のオペコードを受ける。タイプユニット１９２は、再構成インストラクションがジップインストラクションであるか又はアンジップインストラクションであるか、即ち、zip2n8v1p であるか又はunzip2n8v1p であるかに応じて２つの信号を出力する。これらの出力信号はORゲート１９６に与えられる。unzip2n8v1p 信号は切り換えスイッチ１８０乃至１９０を制御する。ORゲート１９６の出力は２つのAND ゲート１９８、２００に与えられる。AND ゲート１９８はDouble信号５８も受ける。AND ゲート２００は逆のDouble信号５８を受ける。AND ゲート２００は第１出力バッファ１７４を制御し、AND ゲート１９８は第２出力

バッファ１７６を制御する。２つの出力バッファはDouble信号によって制御され、該Double信号によって第１出力バッファ１７４はその中身をライン１７８に沿って第１デスティネーションレジスタに与え、次に状態を変化させ、第２出力バッファ１７６はその中身をライン１７８に沿ってレジスタファイル１２の連続レジスタに与える。

【００７１】

処理される２つのインストラクションは以下の通りである。

zip2n8v1p　８個の８ビットオブジェクトからなるベクトルをジップ（インターリーブ）する。

unzip2n8v1p ８個の８ビットオブジェクトからなるベクトルをアンジップ（デインターリーブ）する。

【００７２】

図１１はダブル長１６ビット及び３２ビットジップ及びアンジップインストラクションを実行するツイスト及びジップユニットの部分を示す。この部分は第１及び第２入力バッファ２０２、２０４を有し、それぞれは４つのパック形態の１６ビットオブジェクトを画定する６４ビットワードを保持する。２つのオブジェクトは３２ビットジップインストラクションを使用することによって一緒に処理することができる。第１及び第２出力バッファ２０６及び２０８はそれぞれ４つのパック１６ビットオブジェクトR[0]乃至R[3]、R[4]乃至R[7]を画定する６４ビットワードを保持する。結果はライン２１０に与えられる。Double信号５８は出力バッファがその出力を表すシーケンスを制御する。ツイスト及びジップユニットの他の部分と同様に、第１オブジェクトの第１入力バッファは第１出力バッファの第１オブジェクト位置に直接接続する。同様に、第２入力バッファ２０４の最終ソースオブジェクト位置は第２出力バッファ２０８の最終結果オブジェクト位置R[7]に直接接続する。

【００７３】

第１マルチプレクサ２１０は第１入力としてソースオブジェクトS1[1] 及び第２入力としてソースオブジェクトS1[2] を受ける。第２マルチプレクサ２１２は第１入力として第２ソースオブジェクトS1[1] 及び第２入力として第３ソースオブジェクトS1[2] を受ける。第３マルチプレクサ２１４は第１入力として第２ソースオブジェクトS1[1] 及び第２入力として第２入力バッファの第１ソースオブジェクトS2[0] を受ける。第４マルチプレクサ２１６は第１入力としてソースオブジェクトS1[3] 及び第２入力としてソースオブジェクトS2[2] を受ける。第５マルチプレクサ２１８は第１入力としてソースオブジェクトS2[1] 及び第２入力としてソースオブジェクトS2[2] を受ける。第６マルチプレクサ２２０は第１入力としてソースオブジェクトS2[1] 及び第２入力としてソースオブジェクトS2[2] を受ける。第１マルチプレクサ２１０の出力は第２出力バッファ２０８の第１結果オブジェクトR[4]に与えられる。第２マルチプレクサ２１２の出力は第７マルチプレクサ２２２に与えられ、この第７マルチプレクサは第２出力としてソースオブジェクトS2[0] を受ける。第２マルチプレクサ２２２の出力は第１出力バッファ２０６の第２結果オブジェクトR[1]に与えられる。第３マルチプレクサ２１４の出力は第１出力バッファ２０６の第３結果オブジェクトR[2]に与えられる。第４マルチプレクサ２１６の出力は第２出力バッファ２０８の第２結果オブジェクトR[5]に与えられる。第５マルチプレクサ２１８の出力は第８マルチプレクサ２２４の第１出力として与えられ、この第８マルチプレクサは第２入力としてソースオブジェクトS1[3] を受ける。第８マルチプレクサ２２４の出力は第２出力バッファ２０８の第３結果オブジェクトR[6]に与えられる。第６マルチプレクサ２２０の出力は第１出力バッファ２０６の第４結果オブジェクトR[3]に与えられる。タイプユニット２２６はライン１６０で図２のルートオペコードユニット８２から導出されたライン１６０のオペコードを受ける。タイプユニットはツイスト及びジップユニットによって実行される再構成動作のタイプに応じて３つの出力信号を生成する。これらの信号は、zip2n4vp2p、unzip2n4v2p 及びzip2n2v4p である。これらの信号はORゲート２２８に与えられ、その出力は二つのAND ゲート２３０及び２３２に与えられる。AND ゲート２３０はDouble信号も受ける。AND ゲート２３２はDouble信号の逆転バージョンを受ける。AND ゲート２

３０、２３２の出力は出力バッファ２０６、２０８の動作を制御する。
【００７４】
zip2n4v2p 信号は第７及び第８マルチプレクサ２２２、２２４を制御する。unzip2n4v2p 信号は第１、第２、第３、第４、第５及び第６マルチプレクサを制御する。
【００７５】
ツイスト及びジップユニットのこの部分によって処理される３つのインストラクションは以下の通りである。
zip2n4v2p　４個の１６ビットオブジェクトからなるベクトルをジップ（インターリーブ）する。
unzip2n4v2p　４個の１６ビットオブジェクトからなるベクトルをアンジップ（デインターリーブ）する。
zip2n2v4p　２個の３２ビットオブジェクトからなるベクトルをジップ（インターリーブ）する。同じベクトルのアンジップ（デインターリーブ）と同様である。
【００７６】
図１２は８ビットflipを実行するツイスト及びジップユニットの部分を示す。これはシングル長及びダブル長の両方の動作を行う。図１２において、２つの入力バッファ２３４及び２３６があり、それぞれは８ビットオブジェクトとしてパックされた６４ビットワードを含む。第１及び第２入力バッファ２３４、２３６の隣接ペアはそれぞれマルチプレクサ２３８－２５２に与えられる。マルチプレクサ２５４－２６４の第２セットは以下のように配置される。第２セットの第１マルチプレクサ２５４は第１入力として第１出力バッファ２３４の第２ソースオブジェクトを受け、第２入力として第１セットの第３マルチプレクサ２４２の出力を受ける。第２セットの第２マルチプレクサ２５６は第１入力として第１出力バッファ２３４の第５ソースオブジェクトを受け、第２入力として第１セットの第５マルチプレクサ２４６の出力を受ける。第２セットの第３マルチプレクサ２５８は第１入力として第１出力バッファ２３４の第４ソースオブジェクトを受け、第２入力として第１セットの第４マルチプレクサ２４４の出力を受ける。第２セットの第４マルチプレクサ２６０は第１入力として第１出力バッファ２３４の第７ソースオブジェクトを受け、第２入力として第１セットの第６マルチプレクサの出力を受ける。第２セットの第５マルチプレクサ２６２は第１入力として第１出力バッファの第６ソースオブジェクトを受け、第２入力として第１セットの第７マルチプレクサ２５０の出力を受ける。第２セットの第６マルチプレクサ２６４は第１入力として第１出力バッファ２３４の第８ソースオブジェクトを受け、第２入力として第１セットの第８マルチプレクサ２５２の出力を受ける。ツイスト及びジップユニットの８ビットフリップ部分も出力バッファ２６６を含み、８ビットパックオブジェクトとして６４ビットワードを収容する。第１結果オブジェクトは第１セットの第１マルチプレクサ２３８の出力として与えられる。第２ソースオブジェクトは第２セットの第２マルチプレクサ２５６の出力として与えられる。結果の第３オブジェクトは第１セットの第２マルチプレクサ２４０の出力として与えられる。結果の第４オブジェクトは第２セットの第４マルチプレクサ２６０の出力として与えられる。結果の第５オブジェクトは第２セットの第１マルチプレクサ２５４の出力として与えられる。結果の第６オブジェクトは第２セットの第５マルチプレクサ２６２の出力として与えられる。結果の第７オブジェクトは第２セットの第３マルチプレクサ２５８の出力として与えられる。結果の第８オブジェクトは第２セットの第６マルチプレクサ２６４の出力として与えられる。タイプユニット２６８はライン１６０でオペコードを受け、実行される再構成動作のタイプに応じて２つの信号を生成する。これらの信号はflip2n4v1p及びflip2n8v1pである。これらの信号はORゲート２７０に与えられ、その出力は出力バッファ２６６を制御する。Double信号５８は第１セットのマルチプレクサ２３８－２５２を制御する。Double信号はダブル長インストラクションの上部分に対してのみアクティブである。第２セットのマルチプレクサ２５４－２６４はflip2n8v1p信号によって制御される。
【００７７】
図１２では単一の６４ビット出力バッファのみが例示されている。flip2n4v1pインストラ

クションが実行されているとき、バッファは図９に示される単一の出力バッファに対応する。2n8v1pflipがインストラクションが実行されているとき、出力バッファは最初に結果のRESULT LOW部分を保持してその部分を与え、Double信号５８が表されると結果のRESULT HIGH 部分を保持し、その部分を与える。

【００７８】

ユニットによって処理される２つのインストラクションは以下の通りである。

flip2n4v1p　　４個の８ビットオブジェクトからなるベクトルをフリップする。

flip2n8v1p　　８個の８ビットオブジェクトからなるベクトルをフリップする。

【００７９】

図１３は１６ビット及び３２ビットフリップを実行するツイスト及びジップユニットの部分を示す。８ビットフリップユニットと同様に、この部分もシングル長及びダブル長の両方のフリップを実行する。３２ビットオブジェクトは１６ビットオブジェクトのペアとして処理される。

【００８０】

ユニットによって処理される３つのインストラクションは以下の通りである。

flip2n2v2p　　２個の１６ビットオブジェクトからなるベクトルをフリップする。

flip2n4v2p　　４個の１６ビットオブジェクトからなるベクトルをフリップする。

flip2n2v4p　　２個の３２ビットオブジェクトからなるベクトルをフリップする。

【００８１】

これら３つのフリップのうちの２つはジップのうちの２つと同様である。従って、両方のインストラクションセットがある場合、１つのハードウェアのセットのみを実施すればよい。

【００８２】

このツイスト及びジップユニットの部分は第１及び第２入力バッファ２７２、２７４を有し、それぞれのバッファは第１入力バッファではS1[0] からS1[3] まで、第２入力バッファではS2[0] からS2[3] までの４つの１６ビットオブジェクトとしてパックされた６４ビットワードを収容する。マルチプレクサ２７６乃至２９０の第１セットは第１及び第２入力バッファ２７２、２７４からの入力を以下のように受ける。第１セットの第１マルチプレクサ２７６は第１入力として第１ソースオブジェクトS1[0] を受け、第２入力として第３ソースオブジェクトS1[2] を受ける。第１セットの第２マルチプレクサ２７８は第１入力として第１ソースオブジェクトS1[0] を受け、第２入力として第２ソースオブジェクトS1[1] を受ける。第１セットの第３マルチプレクサ２８０は第１入力として第２ソースオブジェクトS1[1] を受け、第２入力として第４ソースオブジェクトS1[3] を受ける。第１セットの第４マルチプレクサ２８２は第１入力として第３ソースオブジェクトS1[2] を受け、第２入力として第４ソースオブジェクトS1[3] を受ける。第１セットの第５マルチプレクサ２８４は第１入力として第２バッファ２７４の第１ソースオブジェクトS2[0] を受け、第２入力として第３ソースオブジェクトS2[2] を受ける。第１セットの第６マルチプレクサ２８６は第１入力として第２バッファ２７４の第１ソースオブジェクトS2[0] を受け、第２入力として第２ソースオブジェクトS2[1] を受ける。第７マルチプレクサ２８８は第１入力として第２ソースオブジェクトS2[1] を受け、第２入力として第４ソースオブジェクトS2[3] を受ける。第８マルチプレクサ２９０は第１入力として第２入力バッファ２７４の第３ソースオブジェクトS2[2] を受け、第２入力として第４ソースオブジェクトS2[3] を受ける。マルチプレクサ２９２乃至２９８の第２セットは以下のように入力を受ける。第２セットの第１マルチプレクサ２９２は、第１セットの第１及び第２マルチプレクサ２７６、２７８の出力を入力として受ける。第２セットの第２マルチプレクサ２９４は第１セットの第３及び第６マルチプレクサ２８０、２８６からの出力を入力として受ける。第２セットの第３マルチプレクサ２９６は第１セットの第４及び第５マルチプレクサ２８２、２８４の出力を入力として受ける。第２セットの第４マルチプレクサ２９８は第１セットの第７及び第８マルチプレクサ２８８、２９０の出力を入力として受ける。第３セットのマルチプレクサ３００－３０４は以下のように入力を受ける。第３セットの第１

マルチプレクサ３００は第１入力バッファ２７２の第３ソースオブジェクトS1[2] 及び第２セットの第２マルチプレクサ２９４の出力を入力として受ける。第３セットの第２マルチプレクサ３０２は第１入力バッファ２７２の第２ソースオブジェクトS1[1] 及び第２セットの第３マルチプレクサ２９６の出力を入力として受ける。第３セットの第３マルチプレクサ３０４は第１入力バッファ２７２の第４ソースオブジェクトS1[3] 及び第２セットの第４マルチプレクサ２９８の出力を入力として受ける。

【００８３】

ツイスト及びジップユニットのこの部分も４つの１６ビットオブジェクトとしてパックされた６４ビットワードを収容することができる出力バッファ３０６を含む。第１結果オブジェクトR[0]は第２セットの第１マルチプレクサ２９２から導出される。第２結果オブジェクトR[1]乃至第３結果オブジェクトR[3]は第３セットのマルチプレクサ３００－３０４の出力から導出される。

【００８４】

タイプユニット３０６はライン１６０で図２のルートオペコードユニット８２からのオペコードを受ける。タイプユニットはユニットのこの部分によって実行される再構成インストラクションのタイプに応じて３つの信号を生成する。この信号は、flip2n2v2p、flip2n4v2p及びflip2n2v4pである。これらの信号はORゲート３０８に与えられ、このゲートの出力は出力バッファ３０６を制御する。Double信号５８は第１セットのマルチプレクサ２７６乃至２９０を制御する。flip2n2v4p信号は第２セットのマルチプレクサを制御する。flip2n2v2p信号は第３セットのマルチプレクサを制御する。

【００８５】

ユニットのこの部分がflip2n2v2pユニットを実行するために使用される場合、出力バッファはそのインストラクションのために図９に示された単一の出力バッファである。flip2n4v2p又はflip2n2v4pインストラクションを実行するためにユニットのこの部分が使用されると、出力バッファは図１２を参照して上記に述べられたように動作する。

【００８６】

バイト複製及びバイトツイスト及びジップインストラクションの使用例が与えられる。以下の例において、アセンブリ表記はレジスタオペランドＲｎ（ｎは任意の数）を示す。定数オペランドはｎである。ダブル長結果を生成するインストラクションはレジスタのペアの一番目のみを特定する。次に結果の上部分が次のレジスタに書き込まれる。ラベルは ":" が続くアルファベット及び数字からなるストリングによって示される。

【００８７】

一つの有益な動作はマトリックス転置である。

【００８８】

マトリックス転置

ジップ、アンジップ又はフリップはマトリックスを転置するために使用される。単一のインストラクションだけで転置できないマトリックスは、より長いサブユニットに動作する一連のステップで処理することができる。

【００８９】

マトリックスは、左上からスタートして順に各行に沿って右下まで進むことで表される。この行順序付け表示は機能ユニットの図で使用された表示と逆である。

【００９０】

フリップの使用

例えば、フリップを使用した１６ビットオブジェクトの４×４マトリックスの転置において、４つの４分割部分（それぞれは２×２の１６ビットオブジェクトである）は個々に転置され、４×４マトリックスの右上及び左下の４分割部分はスワッピングされる。これは、マトリックスを３２ビットオブジェクトの２つのインターリーブされた２×２マトリックスとして処理することによって実行することができる。図１４はこれを実行するための動作を示す。

【００９１】

転置を実行するためのアセンブリコードは外１に示される。

【００９２】

【外１】

```
;フリップを使用して４×４の１６ビットオブジェクトマトリックスを転置する
;マトリックスは最初レジスタＲ１－４にある
flip2n4v2p R6,R1,R2          ;トップの２つを転置する
flip2n4v2p R8,R3,R4          ;ボトムの２つを転置する
flip2n2v4p R1,R6,R8          ;第１インターリーブ部分を転置する
flip2n2v4p R3,R7,R9          ;第２インターリーブ部分を転置する
;転置されたマトリックスはレジスタＲ１、Ｒ３、Ｒ２、Ｒ４にある
```

【００９３】

ジップの使用

ジップ（パーフェクトシャッフル）を使用して同じマトリックスを転置するためには、１６ビットオブジェクト、次に１６ビットオブジェクトのペア、更に４つの１６ビットオブジェクトについての一連のシャッフルが要求される。

【００９４】

これを実行するためのアセンブリコードは外２に示される。

【００９５】

【外２】

```
;ジップを使用して４×４の１６ビットオブジェクトマトリックスを転置する
;マトリックスは最初レジスタＲ１－４にある
zip2n4v2p  R6,R1,R2          ;最初の２行をジップする
zip2n4v2p  R8,R3,R4          ;最後の２行をジップする
zip2n2v4p  R1,R6,R8          ;第１インターリーブ行をジップする
zip2n2v4p  R3,R7,R9          ;第２インターリーブ行をジップする
;ジップ結果は隣接レジスタにあるため、最後の２つのインストラクションは64ビット
オブジェクトのジップも行うことに注目
;転置したマトリックスはレジスタＲ１－Ｒ４にある
```

【００９６】

アンジップの使用

アンジップ（パーフェクトソート）を使用して同じマトリックスを転置するためには、１６ビットオブジェクトのソートが必要である。図１６はこれを実行するための動作を示す。

【００９７】

これを実行するためのアセンブリコードは外３に示される。

【００９８】

【外３】

```
；アンジップを使用して４×４のバイトマトリックスを転置する
；ソースマトリックスはレジスタ毎に２行でＲ０及びＲ２にある
unzip2n8v1pR2,R0,R1
unzip2n8v1pR4,R2,R3
；結果はＲ４及びＲ５にある
```

```
；アンジップを使用して４×４の２バイトマトリックスを転置する
；ソースマトリックスはレジスタ毎に１行でＲ０、Ｒ１、Ｒ２、Ｒ３にある
 unzip2n4v2pR4,R0,R1
 unzip2n4v2pR6,R2,R3
 unzip2n4v2pR8,R4,R6
 unzip2n4v2pR10,R5,R7
；結果はＲ８、Ｒ１０、Ｒ９及びＲ１１にある
```

【００９９】
外３はアンジップを使用したバイトの４×４マトリックスの転置を示す。
【０１００】
マトリックス乗算
マトリックス乗算は乗算累積のセットからなる。最も一般的なケースはベクトル（１次元）とマトリックス（２次元）を乗算して別のベクトルを生成することである。

$$V_0 \ldots V_{N-1} \qquad \begin{matrix} M_{0,0} \ldots M_{0,M-1} \\ M_{N-1,0} \ldots M_{N-1,M-1} \end{matrix}$$

【０１０１】
[V] 及び[M] が１６ビットデータを含むならば、演算を行うためにパック１６ビット乗算を使用することができる。
【０１０２】
乗算を行うための一つの方法は、バイト複製インストラクションを使用してベクトルの各要素を複製し、複製された各要素とマトリックスの正確な行のパック乗算を実行し、部分積のパック加算を行うことである。マトリックス転置の必要はないことに注意すべきである。これを実行するためのコードシーケンスは外４に示される。
【０１０３】
【外４】

```
;乗算和を使用してベクトルとマトリックスを乗算する
;[V] 及び [M]が与えられて[V][M]を計算する
;Vは４つの１６ビットエレメントを含み、Mは４×４の１６ビットエレメントである
;VはＲ１に含まれる
;MはＲ２－５に含まれる
rep2p      R6,R1              ;第１エレメントを複製する
mul2ps     R7,R6,R2           ;部分積の第１セット
shr        R6,R1,16           ;ベクトルの第２エレメントをシフトダウンする
rep2p      R5,R6              ;第２エレメントを複製する
mul2ps     R8,R6,R3           ;部分積の第２セット
add2p      R7,R7,R8           ;Ｒ７に加算する
shr        R6,R1,32           ;ベクトルの第３エレメントをシフトダウンする
rep2p      R6,R6              ;第３エレメントを複製する
mul2ps     R8,R6,R4           ;部分積の第３セット
add2p      R7,R7,R8           ;Ｒ７に加算する
shr        R6,R1,48           ;ベクトルの第４エレメントをシフトダウンする
rep2p      R5,R6              ;第４エレメントを複製する
mul2ps     R8,R6,R4           ;部分積の第４セット
add2p      R7,R7,R8           ;Ｒ７に加算する
;プロダクトはＲ７にある
```

【０１０４】

ベクトル要素の複製の別の方法はジップを使用することである。図１７はこれを実行するための動作を示す。

【０１０５】

マトリックス乗算のために行うコードシーケンスは外５に示される。

【０１０６】

【外５】

```
;乗算和を使用してベクトルとマトリックスを乗算する
;[V] 及び [M]が与えられて[V][M]を計算する
;Vは４つの１６ビットエレメントを含み、Mは４×４の１６ビットエレメントである
;VはＲ１に含まれる
;MはＲ２－Ｒ５に含まれる
zip2n4v2p  R6,R1,R2          ;ベクトルエレメントのペアを複製する
zip2n2v4p  R8,R6,R6           ;最初の２つのエレメントを４つ複製する
zip2n2v4p  R10,R7,R7          ;２番目の２つのエレメントを４つ複製する
mul2ps     R8,R8,R2           ;部分積の第１セット
mul2ps     R9,R9,R3           ;部分積の第２セット
mul2ps     R10,R10,R4         ;部分積の第３セット
mul2ps     R11,R11,R4         ;部分積の第４セット
add2p      R6,R8,R9           ;第１セットと第２セットを加える
add2p      R7,R10,R11         ;第３セットと第４セットを加える
add2p      R6,R6,R7           ;これらを加える
;プロダクトはＲ６にある
```

【０１０７】

データフォーマット変換

異なるフォーマット間の変換はジップ及びアンジップによって実行されうる。より大きなフォーマットへの符号付き変換には、符号ビットを複製することが必要であり、これは符号付き右シフトによって実行される。表１は種々の符号なしフォーマット間の変換に要求されるインストラクションを示し、表２は符号付き変換を示す。

【０１０８】

【表１】

符号なし変換

From→ To↓	8-bit	16-bit	32-bit	64-bit
8-bit		unzip2n8v1p R,S1,S2		
16-bit	zip2n8v1p R,S,0		unzip2n4v2p R,S1,S2	
32-bit		zip2n4v2p R,S,0		unzip2n2v4p R,S1,S2
64-bit			zip2n2v4p R,S,0	

【０１０９】
【表２】

符号付き変換

From→ To↓	8-bit	16-bit	32-bit	64-bit
8-bit		unzip2n8v1p R,S1,S2		
16-bit	shr1ps tmp,S,7 zip2n8v1p R,S,tmp		unzip2n4v2p R,S1,S2	
32-bit		shr2ps tmp,S,15 zip2n4v2p R,S,tmp		unzip2n2v4p R,S1,S2
64-bit			shr4ps tmp,S,31 zip2n2v4p R,S,tmp	

【０１１０】
ストリングサーチ
ストリングが特定のキャラクタを含んでいるかを知る必要があるときにストリングサーチが使用される。サーチキャラクタを複製し、パック比較を実行することによって、幾つかのキャラクタが同時にテストされることができる。このサーチのコードシーケンスは外６に示される。
【０１１１】
【外６】

```
;strchr
;Ｒ１はストリングをポイントする
;Ｒ２はサーチするキャラクタである
;ストリングはゼロのキャラクタで終了する
replp       R2,R2               ;サーチキャラクタを複製する
loop:
load        R3,R1               ;８バイトのストリングを得る
add         R1,R1,8             ;次の８をポイントする
cmpelp      R4,R3,0             ;ストリングの終わりをテストする
cmpelp      R5,R3,R2            ;所望のキャラクタをテストする
or          R6,R4,R5
jumpz       R6,loop             ;見つからなければ繰り返す
;それがストリングの終わりかキャラクタかを決定する
sub         R6,R4,1
xor         R6,R4,R6            ;ストリングの終わりの前をマスクする
and         R6,R5,R6            ;可能なターゲットキャラクタをマスクする
jumpz       R6,not_found
;特定のキャラクタが見つかったかを決定する
;このビットはカウントゼロのビットインストラクションリピートを画定していないことの
ループである
sub         R1,R1,1             ;ポインタを戻す
shl         R6,R6,8             ;８ビットをシフトアップする
jumpnz      R6,repeat           ;クリアーされなければ繰り返す
;Ｒ１は配置されたキャラクタをポイントする
```

【０１１２】

複製

１、２又は４バイトオブジェクトの複製を実行するためにジップ、アンジップ又はフリップを使用することが可能である。外７、外８及び外９はそれぞれどのようにして最右のバイトを複製するかを示す。

【０１１３】

【外７】

```
;ジップを使用して複製する
;ソースはＲ０にある
zip2n8v1p  R1,R0,R0
zip2n8v1p  R1,R1,R1
zip2n8v1p  R2,R1,R1
;複製値はＲ２にある
```

【０１１４】

【外８】

```
;アンジップを使用して複製する
;ソースはＲ０にある
unzip2n8v1pR1,R0,R0
unzip2n8v1pR1,R1,R1
unzip2n8v1pR2,R1,R1
;複製値はＲ２にある
```

【０１１５】

【外９】

```
;フリップを使用して複製する
;ソースはＲ０にある
flip2n8v1p R1,R0,R0
flip2n4v2p R1,R1,R1
flip2n2v4p R2,R1,R1
;複製値はＲ２にある
```

【０１１６】

ＲＧＢαと平面ビデオフォーマットとの変換

グラフィックス環境で使用するためには、ＲＧＢα（又はパック）フォーマットは、単一のピクセルのレッド、グリーン、ブルー及びアルファカラー情報を含む４つの連続バイトである。従って、各ピクセルは４つの連続バイトを含む。平面フォーマットはレッド、グリーン、ブルー及びアルファカラー情報の全てがメモリの別個の領域に記憶されている場合である。従って全ての同じカラー情報は連続的であり、各ピクセルはメモリの４つの非連続バイトに対応する。

【０１１７】

ＲＧＢαフォーマットと平面フォーマットの変換はジップ又はアンジップによって実行される。ジップを使用したＲＧＢαから平面への変換シーケンスは外１０に示され、アンジップを使用した変換シーケンスは外１１に示される。

【０１１８】

【外１０】

```
;ジップを使用してＲＧＢαから平面に変換する
;ソースはＲ０、Ｒ１、Ｒ２及びＲ３の８つのＲＧＢαピクセル（レジスタ毎に２つ）である
zip2n8v1p  R4,R0,R1
zip2n8v1p  R6,R2,R3
zip2n4v2p  R8,R4,R6
zip2n4v2p  R10,R5,R7
zip2n2v4p  R12,R8,R10
zip2n2v4p  R14,R9,R11
;結果は８つのα、８つのブルー、８つのグリーン及び８つのレッドとしてレジスタＲ１２、
Ｒ１３、Ｒ１４、Ｒ１５にある
```

【０１１９】

【外１１】

```
;アンジップを使用してＲＧＢαから平面に変換する
;ソースはＲ０、Ｒ１、Ｒ２及びＲ３の８つのＲＧＢαピクセル（レジスタ毎に２つ）である
unzip2n8v1pR4,R0,R1
unzip2n8v1pR6,R2,R3
unzip2n8v1pR8,R4,R6
unzip2n8v1pR10,R5,R7
;結果は８つのα、８つのブルー、８つのグリーン及び８つのレッドとしてレジスタＲ８、
Ｒ９、Ｒ１０、Ｒ１１にある
```

【０１２０】

ジップを使用した平面からＲＧＢαへの変換シーケンスは外１２に示され、アンジップを使用した変換シーケンスは外１３に示される。

【０１２１】

【外１２】

```
;ジップを使用して平面からＲＧＢαに変換する
;ソースはＲ０、Ｒ１、Ｒ２及びＲ３の８つのα、８つのブルー、８つのグリーン及び
                                                    ８つのレッドである
zip2n8v1p  R4,R0,R1
zip2n8v1p  R6,R2,R3
zip2n4v2p  R8,R4,R6
zip2n4v2p  R10,R5,R7
;結果はレジスタ毎に２ピクセルでレジスタＲ８、Ｒ９、Ｒ１０、Ｒ１１にある
```

【０１２２】

【外１３】

```
;アンジップを使用して平面からＲＧＢαに変換する
;ソースはＲ０、Ｒ１、Ｒ２及びＲ３の８つのα、８つのブルー、８つのグリーン及び
                                                    ８つのレッドである
unzip2n8v1pR4,R0,R1
unzip2n8v1pR6,R2,R3
unzip2n4v2pR8,R4,R6
unzip2n4v2pR10,R5,R7
unzip2n8v1pR12,R8,R10
unzip2n8v1pR14,R9,R11
;結果はレジスタ毎に２ピクセルでレジスタＲ１２、Ｒ１３、Ｒ１４、Ｒ１５にある
```

【０１２３】

フリップを使用して変換を実行することも可能であるが、ピクセルがインターリーブするため望ましくない。

【０１２４】

回転

ジップ又はアンジップによってマトリックス回転を実行することができる。このためのシーケンスは外１４及び外１５に示される。グラフィックオブジェクトの回転をサポートするために類似したシーケンスを使用することもできる。

【０１２５】

【外１４】

```
;ジップを使用して４×４バイトのマトリックスの反時計回りの回転を行う
;ソースマトリックスはレジスタ毎に２行でＲ０及びＲ１にある
zip2n8v1p  R2,R1,R0
zip2n8v1p  R4,R3,R2
;結果はＲ４及びＲ５にある
```

【０１２６】

【外１５】

```
;アンジップを使用して４×４バイトのマトリックスの時計回りの回転を行う
;ソースマトリックスはレジスタ毎に２行でＲ０及びＲ１にある
unzip2n8v1pR2,R0,R1
unzip2n8v1pR4,R3,R2
;結果はＲ５及びＲ４にある
```

【図面の簡単な説明】

【図１】コンピュータのプロセッサ及びメモリのブロック図である。

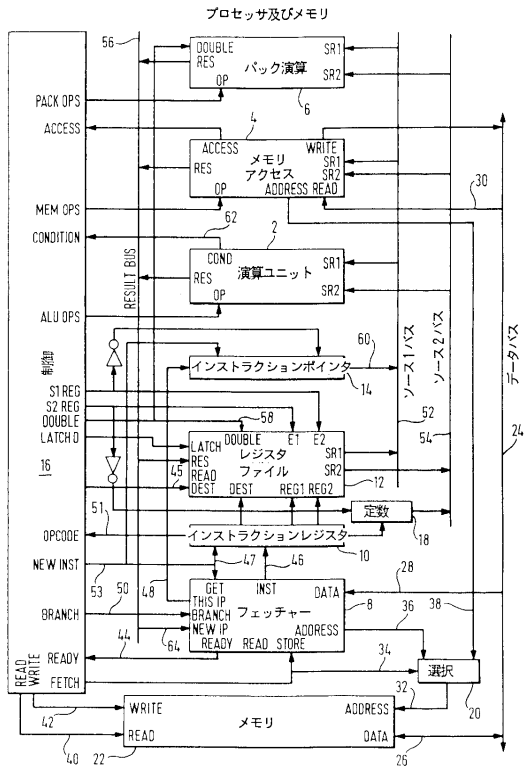
【図２】パック演算ユニットのブロック図である。

【図３】図面において使用される記号の意味を示す。

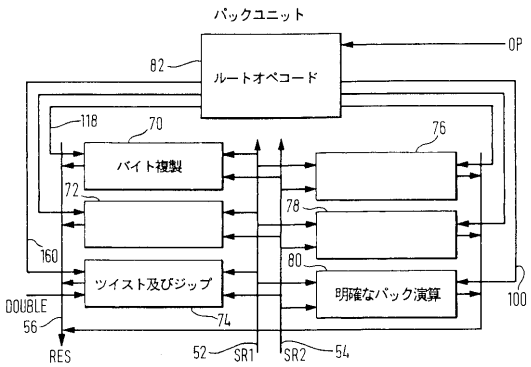
【図４】２つのパックソースオペランドに動作する明確なパック演算ユニットのブロック

図である。
【図５】パックソースオペランド及びアンパックソースオペランドに動作する明確な演算パックユニットのブロック図である。
【図６】バイト複製ユニットを示す。
【図７】ジップ及びアンジップ再構成動作を示す。
【図８】フリップ再構成動作を示す。
【図９】６４ビットジップ及びアンジップを実行するツイスト及びジップユニットの部分を示す。
【図１０】ダブル長８ビットジップ及びアンジップを実行するツイスト及びジップユニットの部分を示す。
【図１１】ダブル長１６ビット及び３２ビットジップ　及びアンジップを実行するツイスト及びジップユニットの部分を示す。
【図１２】８ビットフリップを実行するツイスト及びジップユニットの部分を示す。
【図１３】１６ビット及び３２ビットフリップを実行するツイスト及びジップユニットの部分を示す。
【図１４】フリップインストラクションを使用したマトリックス転置を示す。
【図１５】ジップインストラクションを使用したマトリックス転置を示す。
【図１６】アンジップインストラクションを使用したマトリックス転置を示す。
【図１７】ジップインストラクションを使用して複製がどのように実行されるかを示す。

【図１】

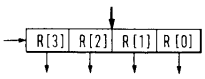


【図２】

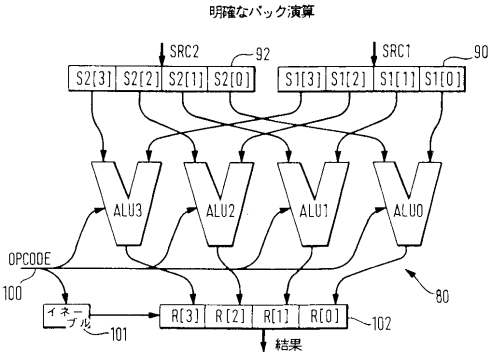


【図３】

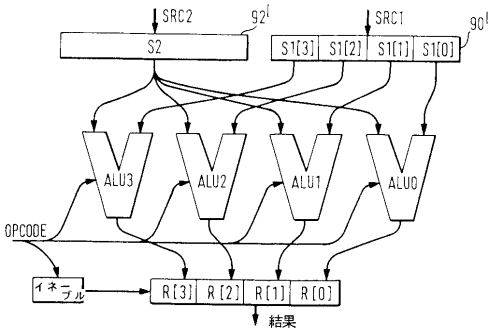
記号

記号	動作
	演算ロジックユニットである。 ２つのソース値を何らかの方法で加算して結果を生成する。
	バッファである。 このある特定の１個は１つの分解不可能入力及び４つの分解可能出力を有し、各出力は入力信号の１／４であり、出力イネーブル入力も有する。
	マルチプレクサである。 出力は２つの入力信号のうちの１つからなる。制御信号が表されなければ アンシェード入力信号が入力され、制御信号が表されるとシェード入力信号が出力される。
	切り換えスイッチである。 ２つの出力のそれぞれは入力信号の１つからなる。制御信号が表されなければアンシェード入力はアンシェード出力へ移行し、シェード入力はシェード出力へ移行する。制御信号が表されると出力はスワップする。

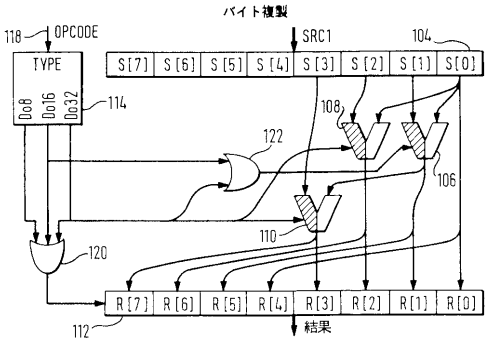
【図４】



【図５】



【図６】



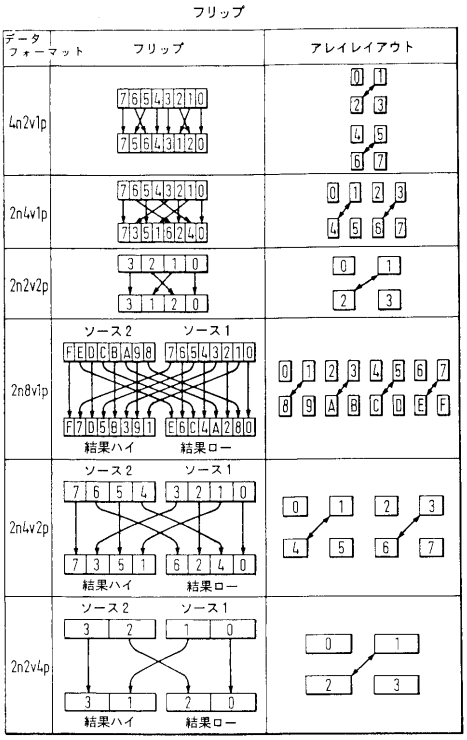
【図７】

ジップ及びアンジップ

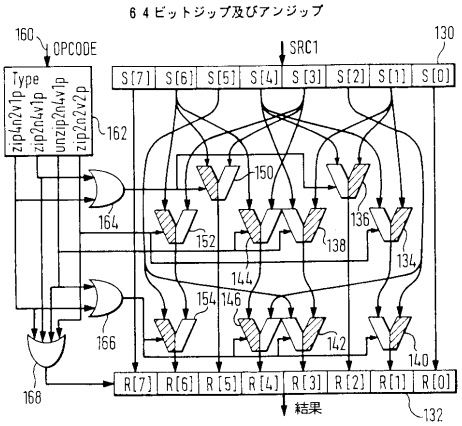
データフォーマット	ジップ	アンジップ
4n2v1p	7 6 5 4 3 2 1 0 → 7 5 6 4 3 1 2 0	7 6 5 4 3 2 1 0 → 7 5 6 4 3 1 2 0
2n4v1p	7 6 5 4 3 2 1 0 → 7 3 6 2 5 1 4 0	7 6 5 4 3 2 1 0 → 7 5 3 1 6 4 2 0
2n8v1p	ソース2 F E D C B A 9 8 / ソース1 7 6 5 4 3 2 1 0 → 結果ハイ F 7 E 6 D 5 C 4 / 結果ロー B 3 A 2 9 1 8 0	ソース2 F E D C B A 9 8 / ソース1 7 6 5 4 3 2 1 0 → 結果ハイ F D 8 9 7 5 3 1 / 結果ロー E C A 8 6 4 2 0
2n2v2p	3 2 1 0 → 3 1 2 0	3 2 1 0 → 3 1 2 0
2n4v2p	ソース2 7 6 5 4 / ソース1 3 2 1 0 → 結果ハイ 7 3 6 2 / 結果ロー 5 1 4 0	ソース2 7 6 5 4 / ソース1 3 2 1 0 → 結果ハイ 7 5 3 1 / 結果ロー 6 4 2 0
2n2v4p	ソース2 3 2 / ソース1 1 0 → 結果ハイ 3 1 / 結果ロー 2 0	ソース2 3 2 / ソース1 1 0 → 結果ハイ 3 1 / 結果ロー 2 0

影が付けられた行は対照的な動作を示す。

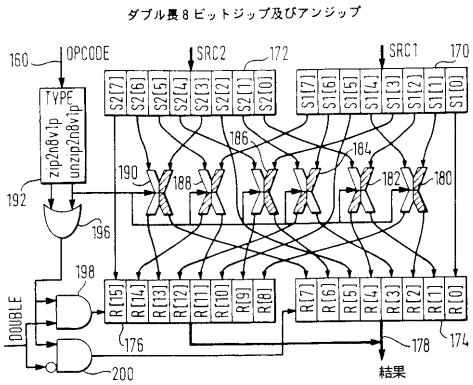
【図８】



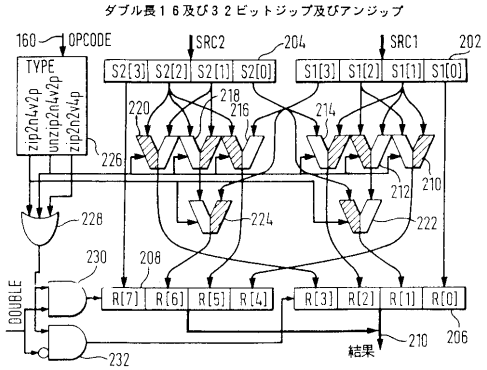
【図９】



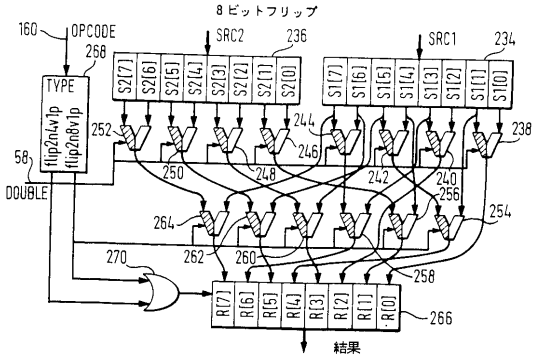
【図１０】



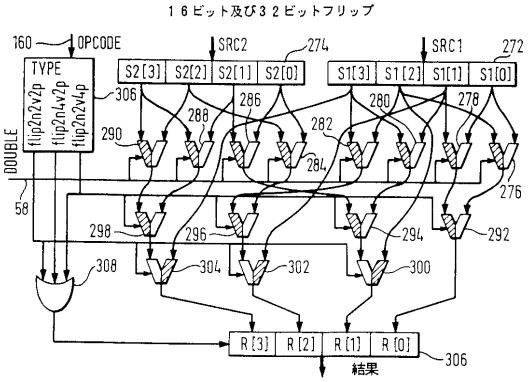
【図１１】



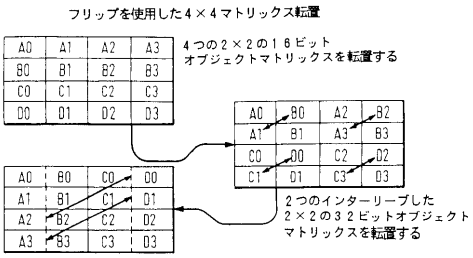
【図１２】



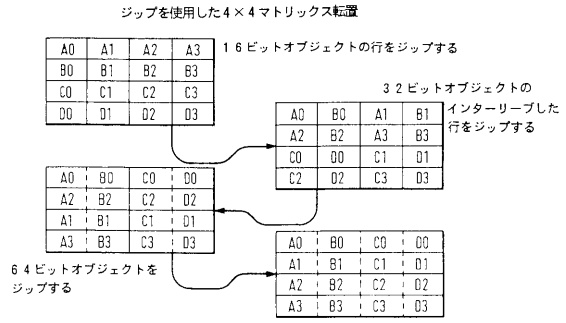
【図１３】



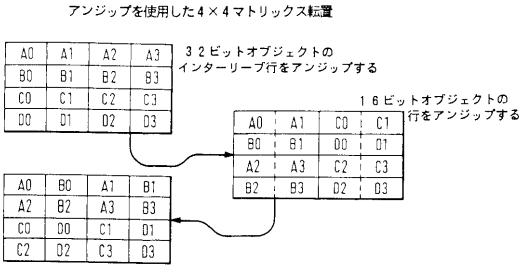
【図１４】



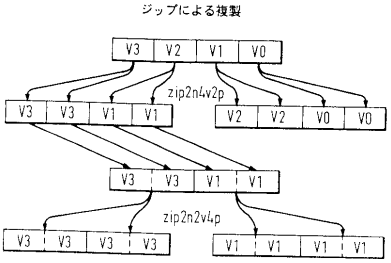
【図１５】



【図１６】



【図１７】



フロントページの続き

(72)発明者 ネイサン マッキンゼー サイドウェル
イギリス国 ＢＳ２ ９ＵＤ ブリストル セントワーバーフス クリーブ ストリート ２１
(72)発明者 キャサリン ルイス バーナビィー
イギリス国 ＢＳ１７ ２ＱＹ ブリストル コールピット ヒース ラウンドウェイズ １８３

審査官 後藤 彰

(56)参考文献 特開平３－２６８０２４（ＪＰ，Ａ）
特開昭５１－９６２５３（ＪＰ，Ａ）
特開平４－１４２６１８（ＪＰ，Ａ）
特表平１０－５１２０７０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/30 - 9/355