(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4268332号
(P4268332)
(45)発行日　平成21年5月27日(2009.5.27)　(24)登録日　平成21年2月27日(2009.2.27)

(51)Int.Cl.　F I
G06F 12/10 (2006.01)　G06F 12/10 505B

請求項の数 10　(全 31 頁)

(21)出願番号　特願2000-329869(P2000-329869)
(22)出願日　平成12年10月30日(2000.10.30)
(65)公開番号　特開2001-175536(P2001-175536A)
(43)公開日　平成13年6月29日(2001.6.29)
審査請求日　平成19年6月29日(2007.6.29)
(31)優先権主張番号　09/430793
(32)優先日　平成11年10月31日(1999.10.31)
(33)優先権主張国　米国(US)

(73)特許権者　599142110
エマージング・アーキテクチャーズ・エルエルシー
Ｅｍｅｒｇｉｎｇ　Ａｒｃｈｉｔｅｃｔｕｒｅｓ，Ｌ．Ｌ．Ｃ．
アメリカ合衆国９５０１４カリフォルニア州クパーチノ、プルーンリッジ・アベニュー　１９４４７　ヒューレット・パッカード・カンパニー内
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ウイリアム・アール・バイグ
アメリカ合衆国９５０７０カリフォルニア州サラトガ、ペレゴ・ウェイ　１８６３０

最終頁に続く

(54)【発明の名称】仮想アドレスからページ・テーブル・インデックスを計算する方法および装置

(57)【特許請求の範囲】
【請求項１】
ページ・テーブルのエントリを参照するエントリ・アドレスを仮想アドレスから形成する方法であって、前記仮想アドレスが、領域を識別する活動領域識別子を参照する領域部分を含み、前記ページ・テーブルが、ロング・フォーマットおよびショート・フォーマットを仮定することができ、
前記仮想アドレスをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するステップであって、前記仮想アドレスの前記領域部分に関連付けられた領域の好ましいページ・サイズが2^Jバイトであるステップと、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記仮想アドレスの前記領域部分によって参照される前記領域識別子および前記ハッシュ・ページ番号を組み合わせることによりハッシュ・インデックスを形成するステップと、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ロング・フォーマット・ページ・テーブル・エントリが2^Kバイト長であるステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによりハッシュ・インデックスを形成するステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハ

ッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ショート・フォーマット・ページ・テーブル・エントリが２^{L}バイト長であるステップと、
前記ページ・テーブルのサイズに基づきマスクを形成するステップと、
前記ページ・テーブルのベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成するステップと、
前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成するステップと、
少なくとも前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップと、を含む方法。

【請求項２】
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブルの前記ベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップとをさらに含む方法であって、
前記少なくとも前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップが、前記ページ・テーブル領域、前記第１アドレス部分および第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップであって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるステップを含む、請求項１に記載の方法。

【請求項３】
前記仮想アドレスからハッシュ・ページ番号を形成するステップが、前記仮想アドレスの実装された部分だけをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するステップであって前記仮想アドレスの前記領域部分に関連付けられた領域の好ましいページ・サイズが２^{J}バイトであるステップを含む、請求項１に記載の方法。

【請求項４】
２^{N}バイトの最小ページ・テーブル・サイズが定義されており、前記ページ・テーブルのベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成するステップが、
前記ページ・テーブルの前記ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブルの前記ベース・アドレス、および前記マスクの下位Ｎビットを含まない前記マスクを使用して第１アドレス部分を形成するステップを含み、
前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成するステップが、
前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクを使用して第２アドレス部分を形成するステップを含み、
前記少なくとも前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップが、前記第１アドレス部分および前記第２アドレス部分を組み合わせて結果を形成し、その結果をＮビット左にシフトし、その結果と前記テーブル・オフセットの下位Ｎビットを組み合わせることにより前記エントリ・アドレスを形成するステップを含む、請求項１に記載の方法。

【請求項５】
ページ・テーブルのエントリを参照するエントリ・アドレスを仮想アドレスから形成する方法であって、前記仮想アドレスが、領域を識別する活動領域識別子を参照する領域部分を含み、前記ページ・テーブルの最小サイズが２^{N}バイトであり、前記ページ・テーブルが、ロング・フォーマットおよびショート・フォーマットを仮定することができ、

前記仮想アドレスの実装された部分だけをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するステップであって、前記仮想アドレスの前記領域部分に関連付けられた前記領域の好ましいページ・サイズが2^{J}バイトであるステップと、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・ページ番号、前記仮想アドレスの前記領域部分、および前記仮想アドレスの前記領域部分から参照される前記領域識別子を組み合わせることによってハッシュ・インデックスを形成するステップであって、前記仮想アドレスをＪビット右にシフトすることに基づき空いていることが分かっている前記ハッシュ・ページ番号のビット位置に前記仮想アドレスの前記領域部分が組み合わされて挿入されるステップと、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ロング・フォーマット・ページ・テーブル・エントリが2^{K}バイト長であるステップと、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブルのベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップと、

前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成するステップと、

前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ショート・フォーマット・ページ・テーブル・エントリが2^{L}バイト長であるステップと、

前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップと、

２をＭ乗し１を引いてマスクを形成するステップであって、2^{M}が前記ページ・テーブルのサイズであるステップと、

前記ページ・テーブルの前記ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブルの前記ベース・アドレス、および前記マスクの反転の下位Ｎビットを含まない前記マスクの反転に対して論理積演算を実行することによって第１アドレス部分を形成するステップと、

前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクに対して論理積演算を実行することによって第２アドレス部分を形成するステップと、

前記第１アドレス部分および前記第２アドレス部分に対して論理和演算を実行して第１の結果を形成し、この第１の結果をＮビット左にシフトして第２の結果を形成し、前記ページ・テーブル領域、前記第２の結果、および前記テーブル・オフセットの下位Ｎビットに対して論理和演算を実行することによって前記エントリ・アドレスを形成するステップであって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるステップと、を含む方法。

【請求項６】

領域を識別する活動領域識別子を参照する領域部分を含む仮想アドレスによってアドレス指定される仮想アドレス空間を定義するアーキテクチャを有するコンピュータ・システムであって、

ページ・テーブル・ベース・アドレスによって固定されたページ・テーブルを含むメモリ・ユニットであって、ページ・テーブルがロング・フォーマットおよびショート・フォーマットを仮定できるメモリ・ユニットと、

命令を実行するプロセッサであって、仮想アドレスから前記ページ・テーブルにエント

リ・アドレスを生成することができるページ・テーブル・エントリ・アドレス生成ユニットであるプロセッサと、を有し、
前記エントリ・アドレス生成ユニットは、
前記仮想アドレスをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するハッシュ・ページ番号生成回路であって、前記仮想アドレスの前記領域部分に関連付けられた前記領域の好ましいページ・サイズが2^{J}バイトであるハッシュ・ページ番号生成回路と、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・ページ番号と前記仮想アドレスの前記領域部分から参照される前記領域識別子を組み合わせることによってハッシュ・インデックスを形成し、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成するハッシュ・インデックス生成回路と、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ロング・フォーマット・ページ・テーブル・エントリが2^{K}バイト長であり、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ショート・フォーマット・ページ・テーブル・エントリが2^{L}バイト長であるテーブル・オフセット生成回路と、
前記ページ・テーブルのサイズに基づきマスクを形成するマスク生成回路と、
前記ページ・テーブル・ベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成する第１アドレス部分生成回路と、
前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成する第２アドレス部分生成回路と、
少なくとも前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するエントリ・アドレス生成回路と、を含むコンピュータ・システム。
【請求項７】
前記プロセッサの前記ページ・テーブル・エントリ・アドレス生成ユニットは、前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブル・ベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成し、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するページ・テーブル領域生成回路をさらに備え、
前記エントリ・アドレス生成回路は、前記ページ・テーブル領域、前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するものであり、ここで前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるものである、請求項６に記載のコンピュータ・システム。
【請求項８】
前記ハッシュ・ページ番号生成回路が、前記仮想アドレスの実装されている部分だけをＪビット右にシフトすることによって前記仮想アドレスから前記ハッシュ・ページ番号を形成するコンピュータ・システムであって、前記仮想アドレスの前記領域部分に関連付けられた前記領域の前記好ましいページ・サイズが2^{J}バイトである、請求項６に記載のコンピュータ・システム。
【請求項９】
前記第１アドレス部分生成回路は、前記ページ・テーブル・ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブル・ベース・アドレスおよび前記マスクの下位Ｎビットを含まない前記マスクを使用して前記第１アドレス部分を形成し、
前記第２アドレス部分生成回路は、前記テーブル・オフセットの下位Ｎビットを含まな

い前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクを使用して前記第２アドレス部分を形成し、

前記エントリ・アドレス生成回路は、前記第１アドレス部分および前記第２アドレス部分を組み合わせて結果を形成し、前記結果をＮビット左にシフトし、前記結果と前記テーブル・オフセットの下位Ｎビットを組み合わせることによって前記エントリ・アドレスを形成する、請求項６に記載のコンピュータ・システム。

【請求項１０】

領域を識別する活動領域識別子を参照する領域部分を含む仮想アドレスによってアドレス指定される仮想アドレス空間を定義するアーキテクチャを有するコンピュータ・システムであって、

ページ・テーブル・ベース・アドレスによって固定されたページ・テーブルを含むメモリ・ユニットであって、ページ・テーブルがロング・フォーマットおよびショート・フォーマットを仮定でき、最小サイズが2^{N}ビットであるメモリ・ユニットと、

命令を実行するプロセッサであって、仮想アドレスからページ・テーブルにエントリ・アドレスを生成することができるページ・テーブル・エントリ・アドレス生成ユニットであるプロセッサとを含み、

前記エントリ・アドレス生成ユニットは、

前記仮想アドレスの実装されている部分だけをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するハッシュ・ページ番号生成回路であって、前記仮想アドレスの前記領域部分に関連付けられた領域の好ましいページ・サイズが2^{J}バイトであるハッシュ・ページ番号生成回路と、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・ページ番号、前記仮想アドレスの前記領域部分、および前記仮想アドレスの前記領域部分から参照される前記領域識別子を組み合わせることによってハッシュ・インデックスを形成し、ここで前記仮想アドレスの前記領域部分が、前記仮想アドレスをＪビット右にシフトすること基づき空いていることが分かっている前記ハッシュ・ページ番号のビット位置に組み合わされて挿入され、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成するハッシュ・インデックス生成回路と、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ロング・フォーマット・ページ・テーブル・エントリが2^{K}バイト長であり、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ショート・フォーマット・ページ・テーブル・エントリが2^{L}バイト長であるテーブル・オフセット生成回路と、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブル・ベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成し、ページ・テーブルのフォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するページ・テーブル領域生成回路と、

２をＭ乗し１を引くことによってマスクを形成するマスク生成回路であって、2^{M}が前記ページ・テーブルのサイズであるマスク生成回路と、

前記ページ・テーブル・ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブル・ベース・アドレス、および前記マスクの反転の下位Ｎビットを含まない前記マスクの反転に対して論理積演算を実行することによって第１アドレス部分を形成する第１アドレス部分生成回路と、

前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクに対して論理積演算を実行することに

よって第２アドレス部分を形成する第２アドレス部分生成回路と、
前記第１アドレス部分および前記第２アドレス部分に対して論理和演算を実行して第１の結果を形成し、前記第１の結果をＮビット左にシフトして第２の結果を形成し、前記ページ・テーブル領域、前記第２の結果、および前記テーブル・オフセットの下位Ｎビットに対して論理和演算を実行することによって前記エントリ・アドレスを形成するエントリ生成回路であって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるエントリ生成回路と、を含むコンピュータ・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はコンピュータ・システム内のメモリ機構に関する。より詳細には、本発明は、ハッシュ関数を介してアクセスされるページ・テーブルを有する仮想メモリ・システムに関する。

【０００２】

【従来の技術】

従来型のコンピュータ・システムは、実際に存在するより多くの論理メモリをシミュレートし、コンピュータがいくつかのプログラムをそのサイズに関係なく同時に実行することを可能にする、仮想メモリと呼ばれる技術を使用する。同時実行するユーザ・プログラムは、オペレーティング・システムによって割り当てられた仮想アドレスを介してメインメモリ・アドレスにアクセスする。仮想アドレスからメインメモリの物理アドレスへのマッピングは、仮想アドレス変換として知られている処理である。仮想アドレス変換を達成する技術は多数あり、それらによってプロセッサがメインメモリ内の所要の情報にアクセスすることが可能になる。

【０００３】

仮想アドレスおよび物理アドレス空間は通常、ページと呼ばれる等しいサイズのメモリ・ブロックに分割され、ページ・テーブルが仮想アドレスと物理アドレスの間の変換を提供する。各ページ・テーブル・エントリは通常、仮想アドレスおよび／または物理アドレス、ならびにそのページに関する保護およびステータス情報を含む。ステータスは通常、そのページが受けたアクセスのタイプについての情報を含む。例えばダーティ・ビットは、そのページ内のデータに変更があったことを示す。通常、ページ・テーブルは大きいのでメモリ内に格納される。従って通常のメモリ・アクセス毎に、変換を獲得する第１のアクセスおよび物理メモリ位置にアクセスする第２のアクセスという、少なくとも２つのアクセスが実際には必要とされ得る。

【０００４】

仮想アドレス変換をサポートする多数のコンピュータ・システムは、変換索引バッファ（ＴＬＢ）を使用する。ＴＬＢは通常、小型で高速な連想記憶であり、普通はプロセッサ・ユニット上または近傍に置かれ、最近使用された仮想アドレスおよび物理アドレスの対を格納する。ＴＬＢはページ・テーブル内の変換のサブセットを含み、はるかに速くアクセスすることができる。処理ユニットは、メインメモリからの情報を必要とするとき、ＴＬＢに対して仮想アドレスを送る。ＴＬＢは仮想アドレス・ページ番号を受け取り、物理ページ番号を返す。物理ページ番号は、メインメモリ内の所要のバイトまたはワードにアクセスするために下位アドレス情報と組み合わされる。

【０００５】

ほとんどの場合、ＴＬＢはページ・テーブル全体を含むことができないので、ＴＬＢを更新する手順を実装する必要がある。ＴＬＢ内に変換がない仮想ページがアクセスされるとき、この仮想ページ番号から物理ページ番号への変換を決定するためにページ・テーブルがアクセスされ、この情報がＴＬＢ内に入れられる。ページ・テーブルに対するアクセスは、ＴＬＢに対するアクセスより２０倍長くかかる可能性があり、したがってＴＬＢ内で変換を利用し続けることでプログラムの実行速度が最適化される。

【０００６】

今日のコンピュータ・システムの大半は、コンピュータ内の物理ランダム・アクセス・メモリ（ＲＡＭ）を増強するためにある種の大量記憶、通常はディスクを使用する。このメインメモリの増強により、メインメモリだけが利用可能な場合より大きなプログラムを実施することができる。さらにディスク・メモリはＲＡＭよりかなり廉価であるが、数桁低速でもある。プログラムの長さおよびメインメモリに対する他のプログラムとの競合次第で、任意の特定時点においてプログラムの一部がメインメモリ内にあり、一部がディスク上にある可能性がある。直ちにアクセスする必要のあるプログラムの一部はメインメモリ内に移動されるが、現在使用されていない部分はディスク上に残される。

【０００７】

例えば１メガバイトのメインメモリを有するコンピュータ上で実行される２メガバイト長の単一プログラムを考慮されたい。このプログラムは２メガバイトの仮想アドレス空間を必要とする。メインメモリが１メガバイトしか保有できないので、所与のいかなる時にもメインメモリ内にはせいぜいプログラムの半分しか存在することができず、仮想アドレス空間の残りはディスク上に格納される。メインメモリ内の情報に対するアクセスは通常どおり行われる。すなわち変換を有するかどうかを見るためにＴＬＢが最初に調べられ、ＴＬＢにない場合、ページ・テーブルからの情報を用いてＴＬＢが更新され、次いで希望する変換情報を得るためにＴＬＢが再度参照される。

【０００８】

メインメモリにない情報に対するアクセスが発生する場合には、その変換を求めてＴＬＢが最初にアクセスされるが、そこにはないことになる。次いでＴＬＢを更新する変換情報を得るためにページ・テーブルが参照される。しかしページ・テーブルにはメインメモリ内の情報に対する変換しかないので、必要な変換情報はないことになる。この状態はページ・フォルトと呼ばれる。ページ・フォルトに応答して、ページ・フォルト・ハンドラが空き物理ページを見つけ、その物理ページにディスク上に格納されている必要な仮想ページをロードし、その変換をページ・テーブルに挿入する。全ての物理ページが他の仮想ページに既に関連付けられている場合、ページ・フォルト・ハンドラは、現在物理メモリ内に格納されているどの仮想ページをディスクにスワップ・アウトするかを選択する必要がある。このタスクを実行する、先入れ先出し、ＬＲＵ（least-recentry-used）アルゴリズムなど多数のアルゴリズムがある。当技術分野でよく知られているように、ＴＬＢ更新処理がハードウェアまたはソフトウェアのいずれかによって処理されるのに対し、ページ・フォルト・ハンドラは通常ソフトウェアに実装される。

【０００９】

図１は上述の処理を示す。ステップ１１２において、ＴＬＢに仮想アドレスが提示される。この仮想アドレスに対する変換がＴＬＢ内にある場合には（ＴＬＢヒット）、関連付けられた物理アドレスがＴＬＢから得られ、物理メモリにアクセスするために利用される（ステップ１１４）。この仮想アドレスに対する変換がＴＬＢにない場合には（ＴＬＢミス）、その変換を求めてページ・テーブルがアクセスされる（ステップ１１６）。その変換がページ・テーブルにある場合には、この情報がＴＬＢに挿入され（ステップ１１８）、仮想アドレスが再び提示される（ステップ１１２）。今度はＴＬＢヒットになり、得られた物理アドレスが物理メモリにアクセスするために使用される。

【００１０】

仮想アドレスが、物理アドレス・ページが関連付けられていない仮想アドレス・ページ内にある場合には、ページ・テーブル内にこのページに対するエントリがないことになり、ページ・フォルトが発生する。この状況では、ソフトウェアのページ・フォルト・ハンドラ（ステップ１２０）が仮想ページに物理ページを割り当て、ディスクから物理ページにページをコピーし、ページ・テーブルを更新する。次いでＴＬＢに仮想アドレスが再び提示される。ＴＬＢにまだ変換がないので、もう１度ＴＬＢミスが起こりページ・テーブルからＴＬＢが更新される。その後、仮想アドレスがＴＬＢに再び提示され、今度はＴＬＢヒットが確実で、得られた物理アドレスが物理メモリにアクセスするために使用される。

【００１１】

図２は、仮想アドレスの提示に応答してバッファ（ＴＬＢ）内のエントリにアクセスする方法を単純化して示したものである。普通ならＴＬＢにはより多数のエントリがあるが、例を単純化するために、例示したＴＬＢには１つしかエントリがない。仮想アドレスがレジスタ２０１にロードされる。この仮想アドレスは２つの部分、仮想ページ番号２０３および物理オフセット２０５からなる。物理オフセットはページ・サイズに対応する。４キロバイトのページ・サイズを有するコンピュータ・システムの場合、物理オフセット２０５はアドレスの下位１２ビット（ビット１１～０）であり、ページ内の特定バイトを示す。レジスタ内の残りのビットは仮想ページ番号を指す。用語「ページ・オフセット」は当業界でしばしば使用される用語であり、用語「物理オフセット」と同義である。仮想アドレスは、「アドレス空間識別子」ビット、「領域（region）識別子」ビットなど、物理ページ番号に対する変換を一意に指定する際に使用される他のビットを含むことがある。

【００１２】

図に示した例の場合、仮想ページ番号はＴＬＢ比較器２０７に対して一入力を与える仮想タグになる。ＴＬＢ２０９は、２つの連結した部分、ＴＬＢタグ２１１および関連付けられた物理ページ番号２１３を有する。ＴＬＢタグ２１１はＴＬＢ比較器２０７に第２の入力を与え、比較器はＴＬＢタグと仮想タグを比較する。タグが一致する場合には、比較器はＴＬＢヒットを示し、物理（実）メモリ・アドレスを与えるために物理ページ番号２１３と物理オフセット２０５が組み合わされる。タグが一致しない場合には、ＴＬＢミスが起こっており、ＴＬＢを更新するために、図１に関連して述べたＴＬＢミス処理が用いられる。

【００１３】

図３は、ＴＬＢミスの後にＴＬＢを更新するために必要となる、仮想ページ番号が与えられた物理ページ情報を取り出す処理を示す。上述のようにページ・テーブル内で仮想から物理へのマッピングが維持される。所与の仮想アドレスを物理アドレスに変換する場合、一つの方法は、ページ・テーブルへのインデックスを形成するために仮想アドレスに対して多対一（ハッシュ）関数を実行することである。これはリンクされたエントリ・リストに対するポインタを与える。次いでこれらのエントリから一致するものが検索される。一致を判定するためには、仮想ページ番号とページ・テーブル内のエントリ（仮想タグ）が比較される。両者が等しい場合、そのページ・テーブル・エントリが物理アドレス変換を与える。

【００１４】

図に示した例では、インデックスを形成するために仮想ページ番号２０３に対してハッシュ関数３０１が実行される。このインデックスはページ・テーブル３０３へのオフセットである。図に示すように、インデックスは０、すなわちインデックスはページ・テーブル内の第１のエントリ３０５を指す。ページ・テーブル内の各エントリは、複数の部分からなるが、通常少なくとも仮想タグ３０７、物理ページ３０９およびポインタ３１１を含む。仮想ページ番号２０３が仮想タグ３０７に等しい場合、物理ページ３０９は所要の物理（実）メモリ・ページ・アドレスを与える。仮想タグが一致しない場合、ポインタ３１１は、仮想から物理への変換情報を含むメモリ内のエントリのチェイン（連鎖）を指す。同じページ・テーブル・エントリに２つ以上の仮想ページ番号が付与される（ハッシュされる）可能性があるときは、そのチェインに含まれる別の情報が必要である。

【００１５】

図に示すように、ポインタ３１１は連鎖セグメント３１３を指す。このチェイン・セグメントは、最初のページ・テーブル・エントリと同じタイプの情報を含む。前と同様に、一致するかどうかを見るために仮想ページ番号２０３と次の仮想タグ３１５が比較される。一致する場合には、関連付けられた物理ページ３１７が所要の物理メモリ・ページのアドレスを与える。一致しない場合には、もしあれば次のチェイン・セグメントを見つけるためにポインタ３１９が検査される。図に示すようにポインタ３１９が他のチェイン・セグメントを指さない場合には、ページ・フォルトが起こっている。その場合、図１に関連して述べたように、ページ・テーブルを更新するためにページ・フォルト・ソフトウェア・

プログラムが使用される。

【００１６】

上述の方法は、仮想タグがコンピュータの基本データ・パス・サイズ以下のシステムに対してはよく機能する。しかし仮想タグがそのデータ・パス・サイズより大きい場合には、仮想タグと仮想ページ番号が同じであるかどうかをテストするために２度比較する必要がある。

【００１７】

「Computer Memory Address Control Apparatus Utilizing Hashed Address Tags in Page Tables Which Are Compared to a Combined Address Tag and Index Which Are Longer than the Basic Data Width of the Associated Computer」という名称で、本明細書に参照により組み込まれたDale Morris等の米国特許第5,724,538号は、仮想タグのサイズを減らし、それによって仮想タグと仮想ページ番号がおなじであるかどうかをテストするのに必要な比較の回数を減らす方式を開示している。基本的に、Morris等は、仮想アドレスの一部は既にハッシュ・インデックスによって表されており、したがってアドレスのその部分は仮想タグによって表される必要がないことを認識した。

【００１８】

図４は、Dale Morris等によって開示された実施形態の単純化されたブロック図を示す。図４においてページ・テーブル４１３は、「ハッシュ・タグ」４２１、４２３を含む。ハッシュ・インデックス４０９は、仮想ページ番号ビット４０１を取り、そのビットに対してインデックス・ハッシュ関数４０５を実行することによって形成され、その結果は、コンピュータの基本データ幅より大きくない。同様にハッシュ・タグは、仮想ページ番号ビット４０１を取り、タグ・ハッシュ関数４２７を実行することによって形成され、その結果得られたハッシュ・タグは、コンピュータの基本データ幅より大きくない。図４はタグ・ハッシュ関数４２７とハッシュ・タグ４２１、４２３の間の明示的な関係を示してはいないが、タグ・ハッシュ関数４２７によって表されるアルゴリズムが、ハッシュ・タグ４２１、４２３が生成され、ページ・テーブル４１３に挿入されるときに使用されることに留意されたい。

【００１９】

インデックス・ハッシュ関数４０５およびタグ・ハッシュ関数４２７は、所与の仮想ページ番号に対して、得られたハッシュ・インデックスおよび得られたハッシュ・タグの組合せが一意であるという範囲において望ましい（complimentary）。したがって仮想ページがアクセスされるときには、ページ・テーブル４１３内の（ハッシュ・タグ４２１、４２３などの）ハッシュ・タグを指すハッシュ・インデックスを生成するために、仮想ページ番号４０１がインデックス・ハッシュ関数４０５に与えられる。テーブル４１３から与えられたハッシュ・タグは比較関数４２９に回される。同時に仮想ページ番号４０１は、ハッシュ・タグ４２５を生成するためにハッシュ関数４２７にも与えられる。ハッシュ・タグ４２５とページ・テーブル４１３のハッシュ・タグが一致する場合には、（エントリ３１７、４１７に格納された物理ページなどの）物理ページがメモリ・アクセス動作を完了するために使用される。タグが一致しない場合、チェイン・セグメントが存在するかどうかを見るために、ページ・テーブル・エントリの（ポインタ３１９、４１９などの）ポインタがアクセスされる。チェイン・セグメントがないか、または全てのチェイン・セグメントを検索して一致するものが見つからない場合には、上述のようにオペレーティング・システムのページ・フォルト・ハンドラが呼び出される。

【００２０】

インデックス・ハッシュ関数４０５およびタグ・ハッシュ関数４２７はハードウェアおよびソフトウェアの両方からアクセスされることに留意されたい。ハードウェアは、仮想ページ番号を物理ページ番号に変換するときにこのハッシュ関数にアクセスしなければならず、ソフトウェアは、ページ・テーブルを初期化するとき、およびページ・フォルトに対応するときに必要となるような、ページ・テーブルにアクセスし変更するときに、このハッシュ関数にアクセスしなければならない。従来の技術においてハッシュ・アルゴリズム

は、本質的に２つの形態で提供された。コンピュータ・ハードウェアは、仮想から物理への変換を迅速に進められるようにハードウェアに基づくバージョンのハッシュ・アルゴリズムを有し、オペレーティング・システムは、ページ・テーブルの初期化、アクセスまたは変更時に仮想から物理への変換を生成するためにソフトウェアに基づくバージョンのハッシュ・アルゴリズムを有した。

【００２１】

いくつかのコンピュータは、領域（region）をサポートすることによって仮想アドレッシングの概念を拡大した。領域は、仮想アドレス空間を等しいサイズの領域に分割することにより、仮想アドレス空間内で独立したローカル・アドレス空間、共用アドレス空間およびグローバル・アドレス空間を効果的に作り出す機能を可能にする。通常任意の時点では、領域の１サブセットだけが活動化される。各領域には、所与の領域のアドレス変換に一意にタグを付ける領域識別子が関連付けられる。ある領域の領域識別子が特定の処理に割り当てられる場合、この領域空間は、その処理に対してローカルになる。ある領域の領域識別子が処理間で共用される場合、この領域空間は共用になる。ある領域の領域識別子が全ての処理によって共用される場合、この領域はグローバルになる。ローカル領域に対する領域識別子の変更により、仮想アドレスは１つの処理のローカル空間から別の処理のローカル空間に効果的にスワップされる。したがって領域は、処理を交換するときにＴＬＢをフラッシュする必要を実質的になくし、それによってシステム性能全体が改善される。

【００２２】

【発明が解決しようとする課題】

本発明は、仮想アドレスからページ・テーブル・インデックスを計算する方法および装置である。本発明は、単一のコンピュータ・アーキテクチャにおいて構成レジスタおよび事前定義定数を介して２つの異なるハッシュ・ページ・テーブル構成をサポートする複合型ハッシュ・アルゴリズムによって実施される。

【００２３】

【課題を解決するための手段】

本発明は例えば、上位３ビットが仮想領域部分を形成する６４ビット仮想アドレスを有する仮想アドレス指定方式と共に使用することができる。したがって所与の任意の時点で仮想アドレスによって８領域を指定することができる。仮想アドレスの残りの６１ビットは、各領域内でメモリをアドレス指定するために使用され、それによって各領域に2^{61}バイトの仮想メモリを提供する。各メモリ・ページには２４ビットの領域識別子が関連付けられる。したがってオペレーティング・システムは、個々の仮想アドレス空間を2^{24}まで割り当てることができる。メモリ・ページは、サイズが４キロバイトから２５６メガバイトの範囲を取ることができる。

【００２４】

第１のハッシュ・ページ・テーブル構成は、領域に基づく直線的ページ・テーブルをサポートし、本明細書において「ショート・フォーマット」ページ・テーブルと呼ばれる。ショート・フォーマット・ページ・テーブルは、各仮想領域に提供され、直線的(linear)であり、領域内の各変換に対して直線的エントリを有する。ショート・フォーマット・ページ・テーブルはチェイン・セグメントを必要とせず、ショート・フォーマット・ページ・テーブルはハッシュ・タグ・エントリを含まない。第２のハッシュ・ページ・テーブル構成は、コンピュータ・システム全体のための単一ページ・テーブルをサポートし、本明細書において「ロング・フォーマット」ページ・テーブルと呼ばれる。ロング・フォーマット・ページ・テーブルはチェイン・セグメントをサポートし、ロング・フォーマット・ページ・テーブル・エントリはハッシュ・タグ・フィールドを含む。

【００２５】

一実施形態において本発明の方法は、仮想アドレスからエントリ・アドレスを形成し、このエントリ・アドレスはページ・テーブルのエントリを参照する。エントリ・アドレスを形成するために、最初に仮想アドレスをＪビット右にシフトすることによって仮想アドレスからハッシュ・ページ番号が形成され、ここで仮想アドレスの領域部分に関連付けられ

た領域の好ましいページ・サイズは、2^{J}バイトである。

【００２６】

コンピュータ・システムがロング・フォーマット・ページ・テーブルで動作している場合、次のステップは、ハッシュ・ページ番号と仮想アドレスの領域部分から参照される領域識別子とを組み合わせることによってハッシュ・インデックスを形成すること、およびハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成することであり、ここでロング・フォーマット・ページ・テーブル・エントリはそれぞれ、2^{K}バイト長である。

【００２７】

しかしコンピュータ・システムがショート・フォーマット・ページ・テーブルで動作している場合、次のステップは、ハッシュ・インデックスをハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成すること、およびハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成することであり、ここでショート・フォーマット・ページ・テーブル・エントリはそれぞれ、2^{L}バイト長である。

【００２８】

次にページ・テーブルのサイズに基づきマスクが形成される。次いでページ・テーブルのベース・アドレスおよびマスクを使用して第１アドレス部分が形成され、テーブル・オフセットおよびマスクを使用して第２アドレス部分が形成される。最後に第１、第２アドレス部分を組み合わせることによってエントリ・アドレスが形成される。

【００２９】

他の実施形態では、領域部分がエントリ・アドレス内に挿入される。フォーマットがロングに設定される場合、領域部分はページ・テーブルのベース・アドレスの領域部分から得られる。しかし領域がショートに設定される場合、領域部分は仮想アドレスの領域部分から得られる。

【００３０】

他の実施形態では、フォーマットがロングに設定されるときに仮想アドレスの領域部分をハッシュ・ページ番号に挿入することによって、ロング・フォーマット・ページ・テーブルの最大サイズが増やされる。

【００３１】

本発明は、ある実施依存パラメータに基づき本発明を実施するために使用されるロジック量を減らすいくつかの実施形態もまた含む。ロングおよびショート・フォーマット・ページ・テーブル双方のためのページ・テーブル・エントリを生成することのできる単一のアルゴリズムを提供することにより、本発明は、実行スピードに重大な影響を与えることなく、両方のページ・テーブル・フォーマットにアクセスするのに必要なロジック量を減らす。

【００３２】

【発明の実施の形態】

本発明は、仮想アドレスからページ・テーブル・インデックスおよびハッシュ・タグを計算する方法および装置である。本発明は、単一のコンピュータ・アーキテクチャにおいて構成レジスタを介して２つの異なるハッシュ・テーブル構成をサポートする複合型（combined）ハッシュアルゴリズム、および仮想アドレスからハッシュ・タグを生成するアルゴリズムによって実施される。

【００３３】

本発明をより詳細に議論する前に、本発明を実施することのできるアーキテクチャの枠組みを最初に考慮されたい。Stephen Burger等による「A Method and Appratus for Exposing a Hardware-based Virtual Memory Hash Scheme to Software」という名称の係属中の米国特許出願を参照により本明細書に組み込む。この出願は、本発明と同じ譲受人に譲渡され、１９９８年１０月１２日に出願され、米国出願番号09/170,143が割り当てられている。Burger等は、ページ・テーブルにアクセスするためにハードウェアによって使用され

るハッシュ・アルゴリズムをソフトウェアに公開する２つの命令を開示している。第１の命令は、ＴＨＡＳＨ（Translation Hashed Entry Address）命令であり、ページ・テーブル内のエントリを指すハッシュ・インデックスを仮想アドレスから生成する。第２の命令は、ＴＴＡＧ（Translation Hashed Entry Tag）命令であり、ハッシュ・インデックスから参照されるページ・テーブルのエントリ内に格納されるハッシュ・タグを仮想アドレスから生成する。この２つの命令を提供することによりＢｕｒｇｅｒ等は、コンピュータ・オペレーティング・システム（または他のシステム・ソフトウェア）がコンピュータ・ハードウェアによって使用されるハッシュ・アルゴリズムをコードする必要がないことを教示した。より正確には、ＴＨＡＳＨおよびＴＴＡＧ命令は、ハードウェアによって使用されるハッシュ・アルゴリズムにソフトウェアがアクセスすることを可能にするインタフェースを提供する。本発明は、ＴＨＡＳＨ命令によって使用することのできる可能な１アルゴリズムを本発明が提供するという点で上記出願に関係する。さらにＴＴＡＧ命令によって使用することのできる可能な１アルゴリズムが以下に開示される。

【００３４】

本発明は、図５に示す仮想アドレス指定方式５０１をサポートする。仮想アドレス５０２は６４ビット・アドレスである。上位３ビットは仮想領域番号（ＶＲＮ）５０３を形成する。したがって仮想アドレスにより、随時８領域を指定することができる。仮想アドレス５０２の残り６１ビットは、各領域内でメモリをアドレス指定するために使用され、それによって各領域に2^{61}バイトの仮想メモリを提供する。（ページ５０４などの）各メモリ・ページには２４ビットの領域識別子（ＲＩＤ）が関連付けられている。したがってオペレーティング・システムは、個々の仮想アドレス空間を2^{24}まで割り当てることができる。以下により詳述するように、メモリ・ページはサイズが４キロバイトから２５６メガバイトの範囲を取ることができる。Stephen Burger等による同時係属中の「A Method and Apparatus for Pre-validating Regions in a Virtual Addressing Scheme」という名称の米国特許出願の中に仮想領域を説明する追加情報が見られる。この出願は参照により本明細書に組み込まれ、本発明と同じ譲受人に譲渡され、１９９８年１０月１２日に出願され、米国出願番号09/170,140が割り当てられている。

【００３５】

本発明は、２つのページ・テーブル・フォーマットを提供するアーキテクチャをサポートする。第１のフォーマットは、領域に基づく直線的（linear）ページ・テーブルであり、本明細書において「ショート・フォーマット」ページ・テーブルと呼ばれる。ショート・フォーマット・ページ・テーブルは、図５に示すように各仮想領域に対して提供される。ショート・フォーマット・ページ・テーブルは直線的であり、領域内の各変換のための直線的エントリを有する。したがってショート・フォーマット・ページ・テーブルはチェイン・セグメントを必要とせず、ショート・フォーマット・ページ・テーブルはハッシュ・タグ・エントリを含まない。

【００３６】

第２のフォーマットは、コンピュータ・システム全体のための単一の大きなページ・テーブルをサポートし、本明細書において「ロング・フォーマット」ページ・テーブルと呼ばれる。ロング・フォーマット・ページ・テーブルは、チェイン・セグメントをサポートし、ロング・フォーマット・ページ・テーブル・エントリは、ハッシュ・タグ・フィールドを含む。

【００３７】

図６はショート・フォーマット・ページ・テーブル・エントリ６０１を示す。ショート・フォーマット・エントリ６０１は単一の６４ビットのワードであり、したがって合計サイズ８バイトを有することに留意されたい。ショート・フォーマット・エントリ６０１内のフィールドを以下のテーブル１に述べる。

【００３８】

【表１】

エントリ・フィールド	説明
ｐ	存在ビット。マッピングされた物理ページがメモリ内に実際にあるかどうかを示す。
ｒｖ	予備
ｍａ	メモリ属性―マッピングされた物理ページのキャッシュ可能性、コヒーレンシ、書込みポリシおよびスペキュレーティブ属性を示す。
ａ	アクセスト・ビット―ページ・フォルトにどのように対応するかを示す。
ｄ	ダーティ・ビット―ページに対するデータ書き込みによって起こるフォルトにどのように対応するかを示す。
ｐｌ	特権レベル―ページの特権レベルを示す。
ａｒ	アクセス権限―ページ・レベルの読取り、書込み、実行許可および特権制御。
ｐｐｎ	物理ページ番号（ＰＰＮ）―マッピングされた物理アドレスのうち最も重要なビット。マッピングにおいて使用されるページ・サイズ次第では、最も重要でないＰＰＮビットのいくつかは無視される。
ｉｇ	オペレーティング・システムに利用可能なソフトウェア・フィールド。ＣＰＵからは無視される。
ｅｄ	例外延期―例外またはフォルトを延期すべきかどうかを示す。

【００３９】
領域内の各ページに対してショート・フォーマット・エントリが存在し、仮想ハッシュ・ページ・テーブル（ＶＨＰＴ）内のショート・フォーマット・エントリの位置によって変換の仮想ページ番号（ｖｐｎ）が暗示されることに留意されたい。ページ・サイズが領域内で一定であることにも留意されたい。
【００４０】
したがって以下に述べるように、ページ・サイズは、領域に関連付けられた構成レジスタのpreferred_page_sizeフィールドにアクセスすることによって入手可能である。
【００４１】
図７はロング・フォーマット・ページ・テーブル・エントリ７０１を示す。ロング・フォーマット・エントリ７０１は４つの６４ビット・ワードを含み、したがって合計サイズが３２バイトであることに留意されたい。ロング・フォーマット・エントリ７０１の第１ワードはショート・フォーマット・エントリ６０１と同一であり、したがって第１ワード内のフィールドは上記テーブル１に述べられている。以下のテーブル２はロング・フォーマット・エントリ７０１の残りのフィールドを記載する。
【００４２】
【表２】

エントリ・フィールド	説明
ｒｖ	予備
ｐｓ	ページ・サイズーマッピングのページ・サイズ。４Ｋバイトより大きいページ・サイズの場合、ＰＰＮおよびＶＰＮの下位ビットは無視される。ページ・サイズは2^{ps}バイトと定義される。
ｋｅｙ	保護キーー保護ドメインに対する変換に一意にタグを付ける。
ｔａｇ	変換タグ。このタグは、ロング・フォーマット・ハッシュ・インデックスと共に用いて、変換を一意に識別するために使用される。
ｔｉ	タグ無効ビット。タグが無効であることを示す。ソフトウェアはロング・フォーマット・エントリを無効にするためにこのビットを使用することができる。
ｉｇ	オペレーティング・システムに利用可能なソフトウェア・フィールド。ＣＰＵからは無視される。ロング・フォーマットＶＨＰＴエントリの最後の６４ビット（オフセット＋２４で開始）は通常オペレーティング・システムにより、ロング・フォーマットＶＨＰＴの同じ最初の(initial)エントリに２つ以上の仮想から物理への変換が混在する場合に別のロング・フォーマットＶＨＰＴエントリに対する関連を格納するために使用されることに留意されたい。

【００４３】
全ての仮想アドレスに対して単一のロング・フォーマット・ページ・テーブルが使用されること、エントリは共にチェインすることができること、通常必ずしも全てのページに対して最初のエントリがあるわけではないことに留意されたい。したがってロング・フォーマット・ページ・テーブル・エントリは、ページ・サイズ（ｐｓ）、タグなどの追加情報を含む。ＶＰＮはハッシュ・インデックスおよびタグによって一意に表される。
【００４４】
本発明のアルゴリズムを以下に議論する前に、次のフィールドが本発明のアルゴリズムに利用可能であることを最後に述べる。これらのフィールドのいくつかは構成レジスタに格納されるプログラム可能変数を表し、したがって特定のコンピュータ・システム上で動作するソフトウェアによって変更することができることに留意されたい。他のフィールドは、コンピュータ・システムの特定の実施態様中で変化しない定数を表し、したがって本発明のアルゴリズムの特定の実施態様中にハード・コード化されてよい。これらのフィールドを以下のテーブル３に示す。
【００４５】
【表３】

構成レジスタフィールドまたは定数	説明
page_table_format	ロング・フォーマットまたはショート・フォーマット・ページ・テーブルが使用されているかどうかを示す。このプログラム可能フィールドはグローバルであり、メモリ内の全てのページに適用される。
preferred_page_size	ページ内のバイト数を示す。ページ・サイズが2^Nバイトの場合にページ・サイズはNとコード化される。このプログラム可能フィールドは、各領域に対して指定することができる。preferred_page_sizeは各ロング・フォーマット・ページ・テーブル・エントリの（ｐｓ）フィールドにコピーされることに留意されたい。
page_table_size	このプログラムの可能フィールドは、ページ・テーブルの直線部分のバイト数を示す。ショート・フォーマット・ページ・テーブルでは、page_table_sizeは各仮想領域に対して与えられ、またショート・フォーマット・ページ・テーブルは、仮想領域内のページ毎に１エントリずつ保有しなければならないので、仮想領域のサイズも決定される。ショート・テーブル・フォーマットが直線的であり成長しないので、page_table_sizeはショート・フォーマット・ページ・テーブルの正確なサイズを示す。ロング・フォーマット・ページ・テーブルにおいて、page_table_sizeはページ・テーブルの直線部分の長さを示し、チェイン・セグメントが追加されるにつれてページ・テーブルはさらに深く(deeper)成長することができる。ページ・テーブルのサイズが2^Nバイトの場合にページ・テーブル・サイズはNとコード化される。
page_table_base	このプログラム可能フィールドは、メモリ内の第１ページ・テーブル・エントリのアドレスを示す。ショート・フォーマット・ページ・テーブルが使用されるとき、各仮想領域は自分自身のページ・テーブルを含み、page_table_baseは各仮想領域に対して与えられる。ロング・フォーマット・ページ・テーブルが使用されるとき、単一のページ・テーブルが提供され、単一のpage__table__baseが第１ロングフォーマット・ページ・テーブル・エントリのアドレスを示す。ビット｛６３：min_pt_size｝（下記参照）だけしか格納する必要がないことに留意されたい。page_table_baseが$2^{page_table_size}$の境界上になければならないことにも留意されたい。
impl_va_msb	この定数は、特定のコンピュータ・システムによってサポートされる仮想アドレスのうち最も重要なビットを示す。
min_pt_size	この定数は、ロングおよびショート・フォーマット・ページ・テーブル双方の（バイト単位の）最小サイズを示す。ページ・テーブルの最小サイズ数が2^Nの場合、最小ページ・テーブル・サイズはNと表される。

【００４６】

上述のように領域に基づくショート・フォーマットにおいて、各領域に対する直線的ページ・テーブルは、参照される領域自体の中にある。結果としてショート・フォーマットＶＨＰＴは、各領域内でpage_table_baseのビット｛６０：min_pt_size｝によって固定される別々の領域毎のページ・テーブルからなる。ＶＨＰＴが利用できる領域に対してオペレ

ーティング・システムは、領域毎の直線的ページ・テーブルを維持する必要がある。以下のショート・フォーマット・アルゴリズムにおいて定義するように、ショート・フォーマットＶＨＰＴへの直線的インデックスを計算するために、変換すべき仮想アドレス（ＶＡ）、領域のpreferred_page_size、page_table_baseおよびpage_table_sizeが使用される。

【００４７】

ショート・フォーマットＶＨＰＴのサイズ（page_table_size）は、マッピングされた仮想アドレス空間のサイズを定義する。ショート・フォーマットにおいてアーキテクチャに適合する最大テーブル・サイズは、領域毎に2^{52}バイトである。４キロバイトのページを使用して領域全体（2^{61}バイト）をマッピングするには、$2^{(61-12)}$（あるいは2^{49}）のページがマップ可能でなければならない。ショート・フォーマットＶＨＰＴエントリは８バイト（あるいは2^{3}バイト）大である。結果として最大テーブル・サイズは、領域毎に$2^{(61-12+3)}$（あるいは2^{52}）バイトである。2^{61}より小さいアドレス空間をマッピングするためにショート・フォーマットが使用される場合、より小さいショート・フォーマット・テーブル（page_table_size＜５２）を使用することができる。４キロバイトのページで2^{N}のアドレス空間をマッピングするには、（Ｎ－９）の最小page_table_sizeが必要である。

【００４８】

ショート・フォーマットＶＨＰＴを使用するとき、（上述の）ＴＨＡＳＨ命令は領域に基づくショート・フォーマット・インデックスを返す。ＴＴＡＧ命令は、これもまた上述したが、ショート・フォーマットでは使用されない。以下のショート・フォーマット・ハッシング・アルゴリズムでは、ＶＨＰＴエントリ・アドレスの要求されている仮想アドレス（ＶＡ）が関数tlb_vhpt_hash_shortに渡される。この関数は、仮想アドレスに対応するエントリのアドレス（vhpt_addr）を返す。

【００４９】

【表４】

ショート・フォーマット・ハッシング・アルゴリズム

```
1:    tlb_vhpt_hash_short (VA)
2:    {
3:    hash_page_number = VA{impl_va_msb:0} u>> preferred_page_size;
4:    hash_index = hash_page_number;
5:    vhpt_offset = hash_index << 3;
6:    vhpt_region= VA{63:61};
7:    pmask = 2 page_table_size - 1;
8:    vhpt_addr = (vhpt_region<<61) |
9:     (((page_table_base{60:min_pt_size} & ~pmask{60:min_pt_size}) |
10:    (vhpt_offset{60:min_pt_size} & pmask{60:min_pt_size})) << min_pt_size) |
11:    vhpt_offset{min_pt_size-1:0};
12:   return vhpt_addr;
13:   }
```

【００５０】

ショート・フォーマット・ハッシング・アルゴリズムのライン１において、関数tlb_vhpt_hash_shortが呼び出され、仮想アドレス（ＶＡ）がこの関数に渡される。ライン３では、preferred_page_sizeによってＶＡを除することによりhash_page_numberが計算される。（定数impl_va_msbによって定義されるように）ＶＡのうち特定のコンピュータ・システムの実施によって使用されるビットだけしか使用されないことに留意されたい。除算演算は、ＶＡをＮビット右にシフトすることによって達成され、ここで、ページ・サイズは2^Nである。右へのシフトは符号なしである。上述のように一実施形態では、ページ・サイズが４キロバイトと２５６メガバイトの間で変化する可能性があるので、ＶＡは１２ビットないし２８ビット右にシフトされる。

【００５１】

ライン４において、hash_indexがhash_page_numberに等しく設定される。ショート・フォーマット・アルゴリズムにおいてこのステップは多少冗長であるが、以下に見られるようにショート・フォーマットとロング・フォーマットのアルゴリズムを調和させるために含められている。上記の議論のように、ショート・フォーマットＶＨＰＴ内の各エントリは８バイト幅である。したがってライン５において、ページ・テーブルへのオフセット（vhpt_offset）がhash_page_numberに８をかけることによって計算される。これはhash_indexを３ビット位置だけ左にシフトすることによって実行される。

【００５２】

ライン６ではＶＨＰＴの領域（vhpt_region）が計算される。上記で議論したように、ショート・フォーマットＶＨＰＴを使用するとき各領域がそれ自体のＶＨＰＴを含むので、ＶＨＰＴの領域はＶＡの領域と同じである。したがってＶＨＰＴの領域は、単純にＶＡのビット｛６３：６１｝である。

【００５３】

ライン７では、page_table_sizeに対応するビット数だけ２を累乗し、１を引くことによってマスク（pmask）が形成される。例えば最小４キロバイトのpreferred_page_sizeを用いて領域全体（2^{61}バイト）をマッピングするには、$2^{(61-12)}$（あるいは2^{49}）ページがマッピング可能でなければならない。各ショート・フォーマットＶＨＰＴエントリが８（あるいは2^3）バイトなので、最大page_table_sizeは2^{52}である。この第１の例では、pmaskの上位１２ビットが「０」であり、下位５２ビットが「１」である。同様に最大２５６メガバイトのpreferred_page_sizeを用いて領域全体（2^{61}バイト）をマッピングするには、$2^{(61-28)}$（あるいは2^{33}）ページがマッピング可能でなければならない。各ショート・フォーマットＶＨＰＴエントリが８（あるいは2^3）バイトなので、（全領域をマッピングするときの）最小page_table_sizeは2^{36}である。この第２の例では、pmaskの上位２８ビットが「０」であり、下位３６ビットが「１」である。もちろん2^{61}バイトの領域全体より少ないバイトをマッピングすることもまた可能であり、preferred_page_size次第で2^{36}より小さいpage_table_sizeになる可能性がある。以下に述べるように、マスクは、ＶＡに対応する得られたＶＨＰＴエントリのアドレスを形成する構成要素を選択するために使用される。

【００５４】

ライン８～１１では、いくつかの構成要素を合わせて論理和を取ることにより、ＶＡに対応するＶＨＰＴのエントリのアドレス（vhpt_addr）が計算される。最初にライン８において、vhpt_regionを６１ビット左にシフトすることによって領域構成要素が計算され、それによってvhpt_regionをvhpt_addrの適切な位置に位置付ける。

【００５５】

ライン９～１１を議論する前に、min_pt_sizeがコンピュータ・システムの各実施態様に対して定義される定数であることを考慮されたい。この定数min_pt_sizeはＮと表され、ここでページ・テーブルの最小サイズは2^Nバイトである。したがってvhpt_addrのビット｛min_pt_size－１：０｝がvhpt_offsetによって提供されることが常に分かる。しかしビット｛６０：min_pt_size｝は、page_table_sizeに基づきpage_table_baseまたはvhpt_offsetのいずれかによって提供される。したがってライン７で計算されたpmaskは、page_

table_sizeに基づきpage_table_baseおよびvhpt_offsetの適切なビットを選択するために使用される。

【００５６】

最小ページ・テーブル・サイズを定義することは、本発明によるコンピュータ・システムを実施するのに必要なロジック量を（ある程度）減らす。例えば各page_table_baseを保有するレジスタは、ビット｛６３：min_pt_size｝しか格納する必要がない。ライン９、１０を参照して以下に議論する論理積および論理和演算の幅もまたmin_pt_sizeビットだけ減らすことができる。一実施形態においてmin_pt_sizeは１５であり、その結果、最小ページ・テーブル・サイズは３２キロバイトとなる。

【００５７】

したがってライン９において、page_table_baseのビット｛６０：min_pt_size｝は、pmaskのビット｛６０：min_pt_size｝の反転との論理積を取られ、ライン１０において、vhpt_offsetのビット｛６０：min_pt_size｝は、pmaskのビット｛６０： min_pt_size｝との論理積を取られる。この２つの論理積演算の結果を合わせて論理和が取られ、その結果がmin_pt_sizeビット位置だけ左にシフトされる。したがってライン９、１０は、このvhpt_addrの構成要素を形成するためにpmaskおよびmin_pt_sizeを使用し、この構成要素はＶＨＰＴのサイズに基づき変化し、min_pt_sizeに基づきvhpt_offsetだけからは与えられないことが分かる。この構成要素がライン８で計算された領域構成要素との論理和を取られることに留意されたい。

【００５８】

最後にライン１１において、vhpt_offset（ビット｛min_pt_size－１：０｝）だけに基づくvhpt_addrの構成要素は、vhpt_addrを形成するために上記で計算された他の２つの構成要素との論理和を取られる。ライン１２において、関数tlb_vhpt_hash_shortが終了し、vhpt_addrを呼び出しルーチンに返す。

【００５９】

ショート・フォーマットＶＨＰＴにおいて各ＶＨＰＴエントリは、仮想アドレスに一意に対応する。しかしロング・フォーマットＶＨＰＴでは、複数の仮想アドレスがＶＨＰＴへの第１エントリを共用する可能性があり、オペレーティング・システムによって第１エントリにチェインされたＶＨＰＴエントリにその後の変換が格納される。第１エントリがアクセスされた後に、仮想から物理への変換に対応する（図７に示す）タグを見つけるために、第１エントリおよびリンクされたエントリを検索することにより、仮想から物理への適切な変換が見つけられる。ロング・フォーマット・アルゴリズムを以下に説明する。混乱を避けるために、全てのアルゴリズムに対して一意のライン番号が使用されていることに留意されたい。

【００６０】

【表５】

ロング・フォーマット・ハッシング・アルゴリズム

```
14:   tlb_vhpt_hash_long (VA, region_id)
15:   {
16:   hash_page_number = VA{impl_va_msb:0} u>> preferred_page_size;
17:   hash_index = ((VA{63:61} << 52) | hash_page_number) ^ region_id;
18:   vhpt_offset = hash_index << 5;
19:   vhpt_region= page_table_base{63:61};
20:   pmask = 2 page_table_size - 1;
21:   vhpt_addr = (vhpt_region<<61) |
22:    (((page_table_base{60:min_pt_size} & ~pmask{60:min_pt_size}) |
23:    (vhpt_offset{60:min_pt_size} & pmask{60:min_pt_size})) << min_pt_size) |
24:    vhpt_offset{min_pt_size-1:0};
25:   return vhpt_addr;
26:   }
```

【００６１】
ロング・フォーマット・ハッシングアルゴリズムのライン１４において、関数tlb_vhpt_hash_longが呼び出され、その関数に仮想アドレス（ＶＡ）および２４ビットのregion_idが渡される。ライン１６において、ＶＡをpreferred_page_sizeで除することによりhash_page_numberが計算される。（impl_va_msbによって定義されるように）ＶＡのうち特定のコンピュータ・システムの実施によって使用されるビットだけしか使用されないことに留意されたい。除算演算は、ＶＡをＮビット右にシフトすることによって達成され、ここでページ・サイズは2^Nである。右のシフトは符号なしである。上述のように、一実施形態においてページ・サイズは、４キロバイトと２５６メガバイトの間で変化する可能性があるので、ＶＡは１２ビットないし２８ビット右にシフトされる。
【００６２】
ライン１７においてhash_indexが形成される。上述のように、hash_page_numberはＶＡを少なくとも１２ビット右にシフトすることによって形成される。したがってhash_page_numberの最大値は2^{52}であり、hash_page_numberのビット｛６４：５２｝は「０」である。ライン１７の最初の部分は、ＶＡのビット｛６３：６１｝（ＶＡの領域部分）を５２ビット左にシフトし、その結果とhash_page_numberとの論理和を取る。これにより、ロング・フォーマットＶＨＰＴの最大可能サイズが2^{52}エントリ（hash page numbersの最大値）から2^{55}エントリに増える。最後にhash_indexを形成するために、ライン１７の最初の部分の結果と２４ビットのregion_idとの排他的論理和が取られる。
【００６３】
上述のように、ロング・フォーマットＶＨＰＴエントリは３２（あるいは2^5）バイトである。したがってライン１８では、hash_indexを５ビット位置だけ左にシフトすることによってvhpt_offsetが形成される。ライン１９では、page_table_baseのビット｛６３：６１｝を取り出すことによってvhpt_regionが形成される。各領域内に存在するショート・フォーマットＶＨＰＴとは対照的に、ロング・フォーマットＶＨＰＴはシステム全体に対して１つしか定義されない。
【００６４】

ライン１７～１９においてhash_index、vhpt_offset、vhpt_regionを計算したのち、ライン２０～２４ではpmaskおよびvhpt_addrが計算される。ロング・フォーマット・アルゴリズムのライン２０～２４は、ショート・フォーマット・アルゴリズムのライン７～１１と同一であることに留意されたい。したがってvhpt_addrは、ショート・フォーマット・アルゴリズムを参照しながら上に述べたのと同じ方法で形成される。最後にライン２５において、関数tlb_vhpt_hash_longが終了し、vhpt_addrを呼び出しルーチンに返す。

【００６５】

ロング・フォーマットＶＨＰＴを使用するときにvhpt_addrは、ロング・フォーマットＶＨＰＴエントリ内に格納されたタグと組み合わせて、仮想から物理への変換を一意に識別することに留意されたい。ロング・フォーマット・ハッシングアルゴリズムで一意性を保証するタグ・アルゴリズムを以下に説明する。

【００６６】

【表６】

タグ・アルゴリズム

```
27:     tag(VA, region_id)
28:     {
29:     pmask = 2 page_table_size - 1;
30:     tpn = VA & -pmask;
31:     tag_for_entry = (region_id << 40) | (tpn >> 12);
32:     return tag_for_entry;
33:     }
```

【００６７】

本発明にしたがって設計されたコンピュータ・システムは、ロングおよびショート・フォーマットＶＨＰＴの両方をサポートする。上に議論したように、ロングおよびショート・フォーマット・ハッシングアルゴリズムはハードウェアにて実施されることが好ましい。当技術分野で知られているように、特定の関数を実施するのに必要なトランジスタの数を最小化しながら関数の実行スピードを最小化することが常に望ましい。

【００６８】

ロングおよびショート・フォーマット・アルゴリズムを検査すると、アルゴリズム間の多数の類似点が注目されよう。本発明によれば、組み合わされたショートおよびロング・フォーマット・アルゴリズムは以下に与えられる。ショートおよびロング・フォーマット・アルゴリズムを組み合わせることにより両方のアルゴリズムを実施するのに必要なトランジスタ数は、どちらのアルゴリズムの実行スピードにも大きな影響を与えることなく最小化される。複合型ハッシングアルゴリズムを以下に説明する。

【００６９】

【表７】

複合型ハッシング・アルゴリズム

```
34:  tlb_vhpt_hash_combined (VA, region_id)
35:  {
36:  hash_page_number = VA{impl_va_msb:0} u>> preferred_page_size;
37:  if (page_table_format == long) {
38:   hash_index = ((VA{63:61} << 52) | hash_page_number) ^ region_id;
39:   vhpt_offset = hash_index << 5;
40:   vhpt_region= page_table_base{63:61};
41:  }
42:  else {
43:   hash_index = hash_page_number;
44:   vhpt_offset = hash_index << 3;
45:   vhpt_region= VA{63:61};
46:  }
47:  pmask = 2 page_table_size - 1;
48:  vhpt_addr = (vhpt_region<<61) |
49:   (((page_table_base{60:min_pt_size} & ~pmask{60:min_pt_size}) |
50:   (vhpt_offset{60:min_pt_size} & pmask{60:min_pt_size})) << min_pt_size) |
51:   vhpt_offset{min_pt_size-1:0};
52:  return vhpt_addr;
53:  }
```

【００７０】

基本的に、複合型ハッシングアルゴリズムは、ロングおよびショート・フォーマット・ハッシングアルゴリズムの共通要素を組合わせ、アルゴリズムの異なる部分は、page_table_formatが「ロング」に設定されているかどうかをテストするIF-THEN-ELSEブロック内で与えられる。したがって複合型ハッシングアルゴリズムのライン３４において、関数tlb_vhpt_hash_combinedが呼び出され、その関数に仮想アドレス（ＶＡ）および２４ビットのregion_idが渡される。page_table_formatが「ロング」に設定されていない場合、region_idが使用されないことに留意されたい。ライン３６では、ショート・フォーマット・ハッシングアルゴリズムのライン３およびロング・フォーマット・ハッシングアルゴリズムのライン１６と同様に、ＶＡをprefferd_page_sizeで割ることによってhash_page_numberが計算される。

【００７１】

ライン３７ではpage_table_formatが、「ロング」に設定されているかどうかを見るためにテストされる。設定されている場合には、ロング・フォーマット・ハッシングアルゴリズムのライン１７、１８、１９とそれぞれ同様に、hash_index、vhpt_offset、vhpt_regionがライン３８、３９、４０においてそれぞれ計算される。page_table_formatが「ロング」に設定されていない場合、ショート・フォーマット・ハッシングアルゴリズムのライン４、５、６とそれぞれ同様に、hash_index、vhpt_offset、vhpt_regionがライン４３、

４４、４５においてそれぞれ計算される。その後、pmaskおよびvhpt_addrが計算され、ライン４７～５２において（呼び出しルーチンにvhpt_addrを返して）関数が終了する。ライン４７～５２が、ショート・フォーマット・ハッシング・アルゴリズムのライン７～１２と同一であり、またロングフォーマット・ハッシングアルゴリズムのライン２０～２５と同一であることに留意されたい。

【００７２】

したがって本発明は、（各ＶＨＰＴエントリが仮想から物理への変換を一意に識別する）ショート・フォーマットＶＨＰＴ、または（格納されたタグと組み合わされた各最初のＶＨＰＴが仮想から物理への変換を一意に識別する）ロングフォーマットＶＨＰＴのいずれかに対するインデックスを生成することのできる、複合型ハッシングアルゴリズムを提供する。本発明の複合型アルゴリズムを実施する者はまた、ロングおよびショート・フォーマットに対して別々に実行される複合型ハッシングアルゴリズムの一部に別の共通点を見つけるであろう。例えばライン３９、４４で実行される左のシフトは、page_table_formatが「ロング」に設定されている場合にさらに２ビット位置左にシフトする単一のシフト回路によって実施することができる。同様にライン４０、４５のvhpt_regionの計算は、page_table_base、またはpage_table_formatに基づくＶＡから、ビット６３～６１を選択するためにマルチプレクサを使用することができる。本発明の複合型ハッシングアルゴリズムを実施するために論理回路を設計する者は、本発明の実施に必要なロジックを最小化する他の方法にも気付くであろう。

【００７３】

好ましい実施形態を参照しながら本発明を述べて来たが、本発明の精神および範囲を逸脱することなく、形態および詳細において変更を加えることができることを当業者なら認識するであろう。

【００７４】

本発明は例として以下の実施態様を含む。

【００７５】

（１）ページ・テーブルのエントリを参照するエントリ・アドレスを仮想アドレスから形成する方法であって、前記仮想アドレスが、領域を識別する活動領域識別子を参照する領域部分を含み、前記ページ・テーブルが、ロング・フォーマットおよびショート・フォーマットを仮定することができ、

前記仮想アドレスをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するステップであって、前記仮想アドレスの前記領域部分に関連付けられた領域の好ましいページ・サイズが2^{J}バイトであるステップと、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記仮想アドレスの前記領域部分によって参照される前記領域識別子および前記ハッシュ・ページ番号を組み合わせることによりハッシュ・インデックスを形成するステップと、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ロング・フォーマット・ページ・テーブル・エントリが2^{K}バイト長であるステップと、

前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによりハッシュ・インデックスを形成するステップと、

前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ショート・フォーマット・ページ・テーブル・エントリが2^{L}バイト長であるステップと、

前記ページ・テーブルのサイズに基づきマスクを形成するステップと、

前記ページ・テーブルのベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成するステップと、

前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成するステップと、
前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップと、を含む方法。
【００７６】
（２）前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブルの前記ベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップとをさらに含む方法であって、
前記ページ・テーブル領域、前記第１アドレス部分および第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップであって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるステップを含む前記（１）に記載の方法。
【００７７】
（３）前記仮想アドレスからハッシュ・ページ番号を形成するステップが、前記仮想アドレスの実装された部分だけをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するステップであって前記仮想アドレスの前記領域部分に関連付けられた領域の好ましいページ・サイズが2^{J}バイトであるステップを含む前記（１）に記載の方法。
【００７８】
（４）前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記仮想アドレスの前記領域部分から参照される前記領域識別子および前記ハッシュ・ページ番号を組み合わせることによってハッシュ・インデックスを形成するステップが、前記仮想アドレスの前記領域部分と前記ハッシュベージ番号を組み合わせるステップを含む前記（１）に記載の方法。
【００７９】
（５）前記仮想アドレスの前記領域部分と前記ハッシュ・ページ番号を組み合わせるステップが、前記仮想アドレスの前記領域部分のビットを前記ハッシュ・ページ番号内に、前記仮想アドレスをＪビット右にシフトすることに基づき空いていると分かっている前記ハッシュ・ページ番号のビット位置に挿入するステップを含む前記（４）に記載の方法。
【００８０】
（６）前記ページ・テーブルの前記サイズに基づきマスクを形成するステップが、前記マスクを$2^{M}-1$に等しく設定するステップであって、2^{M}が前記ページ・テーブルの前記サイズであるステップを含む前記（１）に記載の方法。
【００８１】
（７）前記ページ・テーブルのベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成するステップが、
前記ページ・テーブルの前記ベース・アドレスおよび前記マスクの反転に対して論理積演算を実行することによって第１アドレス部分を形成するステップを含み、
前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成するステップが、
前記テーブル・オフセットおよび前記マスクに対して論理積演算を実行することによって第２アドレス部分を形成するステップを含む前記（６）に記載の方法。
【００８２】
（８）前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップが、前記第１アドレス部分および前記第２アドレス部分に対して論理和演算を実行することによって前記エントリ・アドレスを形成するステップを含む前記（１）に記載の方法。

【００８３】
（９）2^Nバイトの最小ページ・テーブル・サイズが定義されており、前記ページ・テーブルのベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成するステップが、
前記ページ・テーブルの前記ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブルの前記ベース・アドレス、および前記マスクの下位Ｎビットを含まない前記マスクを使用して第１アドレス部分を形成するステップを含み、
前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成するステップが、
前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクを使用して第２アドレス部分を形成するステップを含み、
前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するステップが、
前記第１アドレス部分および前記第２アドレス部分を組み合わせて結果を形成し、その結果をＮビット左にシフトし、その結果と前記テーブル・オフセットの下位Ｎビットを組み合わせることにより前記エントリ・アドレスを形成するステップを含む前記（１）に記載の方法。
【００８４】
（１０）ページ・テーブルのエントリを参照するエントリ・アドレスを仮想アドレスから形成する方法であって、前記仮想アドレスが、領域を識別する活動領域識別子を参照する領域部分を含み、前記ページ・テーブルの最小サイズが2^Nバイトであり、前記ページ・テーブルが、ロング・フォーマットおよびショート・フォーマットを仮定することができ、
前記仮想アドレスの実装された部分だけをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するステップであって、前記仮想アドレスの前記領域部分に関連付けられた前記領域の好ましいページ・サイズが2^Jバイトであるステップと、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・ページ番号、前記仮想アドレスの前記領域部分、および前記仮想アドレスの前記領域部分から参照される前記領域識別子を組み合わせることによってハッシュ・インデックスを形成するステップであって、前記仮想アドレスをＪビット右にシフトすることに基づき空いていることが分かっている前記ハッシュ・ページ番号のビット位置に前記仮想アドレスの前記領域部分が組み合わされて挿入されるステップと、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ロング・フォーマット・ページ・テーブル・エントリが2^Kバイト長であるステップと、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブルのベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成するステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成するステップであって、各ショート・フォーマット・ページ・テーブル・エントリが2^Lバイト長であるステップと、
前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するステ

ップと、
２をＭ乗し１を引いてマスクを形成するステップであって、2^Mが前記テーブルのサイズであるステップと、
前記ページ・テーブルの前記ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブルの前記ベース・アドレス、および前記マスクの反転の下位Ｎビットを含まない前記マスクの反転に対して論理積演算を実行することによって第１アドレス部分を形成するステップと、
前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクに対して論理積演算を実行することによって第２アドレス部分を形成するステップと、
前記第１アドレス部分および前記第２アドレス部分に対して論理積演算を実行して第１の結果を形成し、この第１の結果をＮビット左にシフトして第２の結果を形成し、前記ページ・テーブル領域、前記第２の結果、および前記テーブル・オフセットの下位Ｎビットに対して論理和演算を実行することによって前記エントリ・アドレスを形成するステップであって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるステップと、を含む方法。

【００８５】

（１１）領域を識別する活動領域識別子を参照する領域部分を含む仮想アドレスによってアドレス指定される仮想アドレス空間を定義するアーキテクチャを有するコンピュータ・システムであって、
ページ・テーブル・ベース・アドレスによって固定されたページ・テーブルを含むメモリ・ユニットであって、ページ・テーブルがロング・フォーマットおよびショート・フォーマットを仮定できるメモリ・ユニットと、
命令を実行するプロセッサであって、仮想アドレスから前記ページ・テーブルにエントリ・アドレスを生成することができるページ・テーブル・エントリ・アドレス生成ユニットであるプロセッサと、を有し、
前記エントリ・アドレス生成ユニットは、
前記仮想アドレスをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するハッシュ・ページ番号生成回路であって、前記仮想アドレスの前記領域部分に関連付けられた前記領域の好ましいページ・サイズが2^Jバイトであるハッシュ・ページ番号生成回路と、
前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・ページ番号と前記仮想アドレスの前記領域部分から参照される前記領域識別子を組み合わせることによってハッシュ・インデックスを形成し、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成するハッシュ・インデックス生成回路と、前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ロング・フォーマット・ページ・テーブル・エントリが2^Kバイト長であり、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ショート・フォーマット・ページ・テーブル・エントリが2^Lバイト長であるテーブル・オフセット生成回路と、
前記ページ・テーブルの前記サイズに基づきマスクを形成するマスク生成回路と、
前記ページ・テーブル・ベース・アドレスおよび前記マスクを使用して第１アドレス部分を形成する第１アドレス部分生成回路と、
前記テーブル・オフセットおよび前記マスクを使用して第２アドレス部分を形成する第２アドレス部分生成回路と、
前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成するエントリ・アドレス生成回路と、を含むコンピュータ・システ

ム。
【００８６】
（１２）前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブル・ベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成し、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するページ・テーブル領域生成回路と、
前記ページ・テーブル領域、前記第１アドレス部分および前記第２アドレス部分を組み合わせることによって前記エントリ・アドレスを形成する前記エントリ・アドレス生成回路であって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるエントリ・アドレス生成回路とを、前記プロセッサの前記ページ・テーブル・エントリ・アドレス生成ユニットがさらに備える前記（１１）に記載のコンピュータ・システム。
【００８７】
（１３）前記ハッシュ・ページ番号生成回路が、前記仮想アドレスの実装されている部分だけをＪビット右にシフトすることによって前記仮想アドレスから前記ハッシュ・ページ番号を形成するコンピュータ・システムであって、前記仮想アドレスの前記領域部分に関連付けられた前記領域の前記好ましいサイズが2^{J}バイトである前記（１１）に記載のコンピュータ・システム。
【００８８】
（１４）前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックス生成ユニットが前記ハッシュ・ページ番号、前記仮想アドレスの前記領域部分から参照される前記領域識別子、および前記仮想アドレスの前記領域部分を組み合わせることによって前記ハッシュ・インデックスを形成する前記（１１）に記載のコンピュータ・システム。
【００８９】
（１５）前記ハッシュ・インデックス生成ユニットが、前記ハッシュ・ページ番号に前記仮想アドレスの前記領域部分のビットを、前記仮想アドレスをＪビット右にシフトすることに基づき空いていることが分かっている前記ハッシュ・ページ番号のビット位置に挿入もする前記（１４）に記載のコンピュータ・システム。
【００９０】
（１６）前記マスク生成ユニットが、前記マスクを$2^{M}-1$に等しく設定することによって前記マスクを形成するコンピュータ・システムであって、2^{M}が前記ページ・テーブルのサイズである前記（１５）に記載のコンピュータ・システム。
【００９１】
（１７）前記第１アドレス部分生成回路が前記ページ・テーブル・ベース・アドレスと前記マスクの反転に対して論理積演算を実行することによって前記第１アドレス部分を形成し、前記第２アドレス部分生成回路が前記テーブル・オフセットと前記マスクに対して論理積演算を実行することによって前記第２アドレス部分を形成する前記（１６）に記載のコンピュータ・システム。
【００９２】
（１８）前記エントリ・アドレス生成回路が前記第１アドレス部分および前記第２アドレス部分に対して論理和演算を実行することによって前記エントリ・アドレスを形成する前記（１１）に記載のコンピュータ・システム。
【００９３】
（１９）2^{N}バイトの最小ページ・テーブル・サイズが定義され、前記第１アドレス部分生成回路が、前記ページ・テーブル・ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブル・ベース・アドレスおよび前記マスクの下位Ｎビットを含まない前記マスクを使用して前記第１アドレス部分を形成し、前記第２アドレス生成回路が、前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクを使用して前記第２アドレス部分を形成し、前記エ

ントリ・アドレス生成回路が、前記第１アドレス部分および前記第２アドレス部分を組み合わせて結果を形成し、前記結果をＮビット左にシフトし、前記結果と前記テーブル・オフセットの下位Ｎビットを組み合わせることによって前記エントリ・アドレスを形成する前記（１１）に記載のコンピュータ・システム。

【００９４】

（２０）領域を識別する活動領域識別子を参照する領域部分を含む仮想アドレスによってアドレス指定される仮想アドレス空間を定義するアーキテクチャを有するコンピュータ・システムであって、

ページ・テーブル・ベース・アドレスによって固定されたページ・テーブルを含むメモリ・ユニットであって、ページ・テーブルがロング・フォーマットおよびショート・フォーマットを仮定でき、最小サイズが2^Nビットであるメモリ・ユニットと、

命令を実行するプロセッサであって、仮想アドレスからページ・テーブルにエントリ・アドレスを生成することができるページ・テーブル・エントリ・アドレス生成ユニットであるプロセッサとを含み、

前記エントリ・アドレス生成ユニットは、

前記仮想アドレスの実装されている部分だけをＪビット右にシフトすることによって前記仮想アドレスからハッシュ・ページ番号を形成するハッシュ・ページ番号生成回路であって、前記仮想アドレスの前記領域部分に関連付けられた領域の好ましいページ・サイズが2^Jバイトであるハッシュ・ページ番号生成回路と、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・ページ番号、前記仮想アドレスの前記領域部分、および前記仮想アドレスの前記領域部分から参照される前記領域識別子を組み合わせることによってハッシュ・インデックスを形成し、ここで前記仮想アドレスの前記領域部分が、前記仮想アドレスをＪビット右にシフトすること基づき空いていることが分かっている前記ハッシュ・ページ番号のビット位置に組み合わされて挿入され、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスを前記ハッシュ・ページ番号に等しく設定することによってハッシュ・インデックスを形成するハッシュ・インデックス生成回路と、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ハッシュ・インデックスをＫビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ロング・フォーマット・ページ・テーブル・エントリが2^Kバイト長であり、前記ページ・テーブルの前記フォーマットがショートに設定されている場合に、前記ハッシュ・インデックスをＬビット左にシフトすることによってテーブル・オフセットを形成し、ここで各ショート・フォーマット・ページ・テーブル・エントリが2^Lバイト長であるテーブル・オフセット生成回路と、

前記ページ・テーブルの前記フォーマットがロングに設定されている場合に、前記ページ・テーブル・ベース・アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成し、ページ・テーブルのフォーマットがショートに設定されている場合に、前記仮想アドレスから前記領域部分を抽出することによってページ・テーブル領域を形成するページ・テーブル領域生成回路と、

２をＭ乗し１を引くことによってマスクを形成するマスク生成回路であって、2^Mが前記テーブルのサイズであるマスク生成回路と、

前記ページ・テーブル・ベース・アドレスの下位Ｎビットを含まない前記ページ・テーブル・ベース・アドレス、および前記マスクの反転の下位Ｎビットを含まない前記マスクの反転に対して論理積演算を実行することによって第１アドレス部分を形成する第１アドレス部分生成回路と、

前記テーブル・オフセットの下位Ｎビットを含まない前記テーブル・オフセット、および前記マスクの下位Ｎビットを含まない前記マスクに対して論理積演算を実行することによって第２アドレス部分を形成する第２アドレス部分生成回路と、

前記第１アドレス部分および前記第２アドレス部分に対して論理積演算を実行して第１の

結果を形成し、前記第１の結果をＮビット左にシフトして第２の結果を形成し、前記ページ・テーブル領域、前記第２の結果、および前記テーブル・オフセットの下位Ｎビットに対して論理和演算を実行することによって前記エントリ・アドレスを形成するエントリ生成回路であって、前記ページ・テーブル領域が前記エントリ・アドレスの領域部分に挿入されるエントリ生成回路と、を含むコンピュータ・システム。

【図面の簡単な説明】

【図１】プログラムの実行中に提示される仮想アドレスに応答する従来技術の手順を示す図。

【図２】変換索引バッファ（ＴＬＢ）内のエントリにアクセスする従来技術の方法を示す図。

【図３】ＴＬＢミスの後に、ＴＬＢを更新するための物理ページ情報を取り出す従来技術の方法を示す図。

【図４】ページ・テーブル内にハッシュ・タグが格納される従来技術のページ・テーブル指定方式を示す図。

【図５】本発明によってサポートされる仮想アドレス指定方式を示す図。

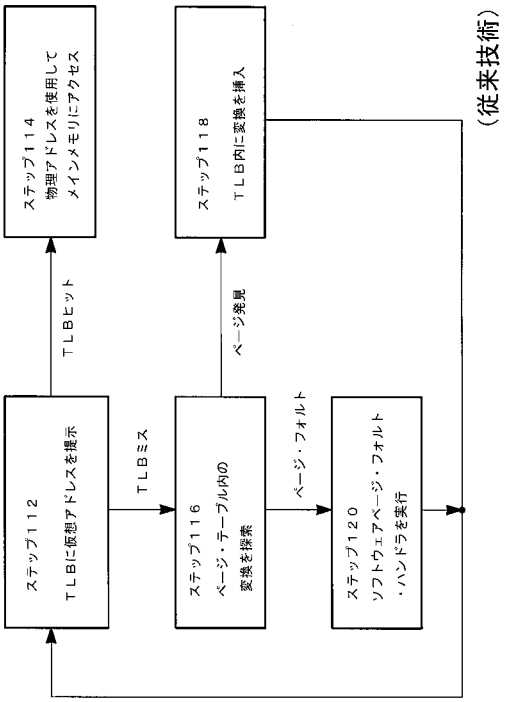
【図６】本発明のハッシュ関数の適用によってアクセスすることのできる「ショート・フォーマット」仮想ハッシングページ・テーブルのエントリを示す図。

【図７】本発明のハッシュ関数の適用によってアクセスすることのできる「ロング・フォーマット」仮想ハッシングページ・テーブルのエントリを示す図。

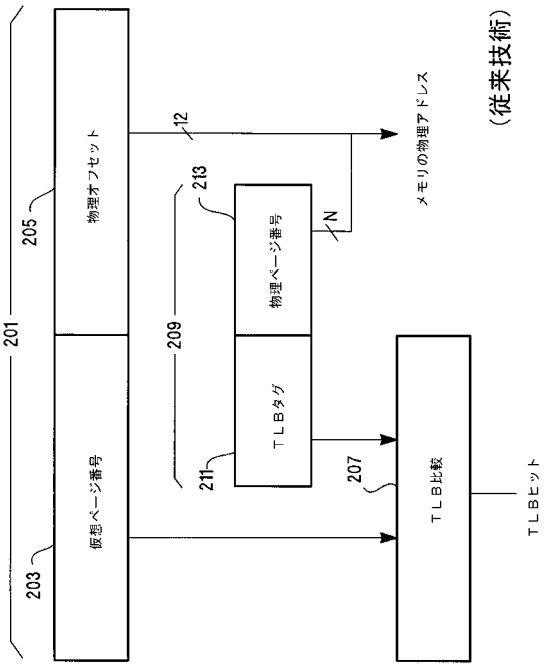
【符号の説明】

５０１　仮想アドレス指定方式
５０２　仮想アドレス
５０３　仮想領域番号
５０４　ページ

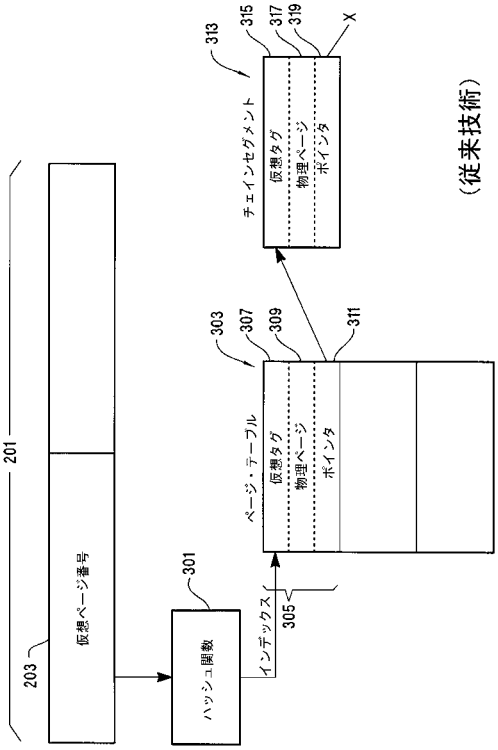
【図１】



【図２】

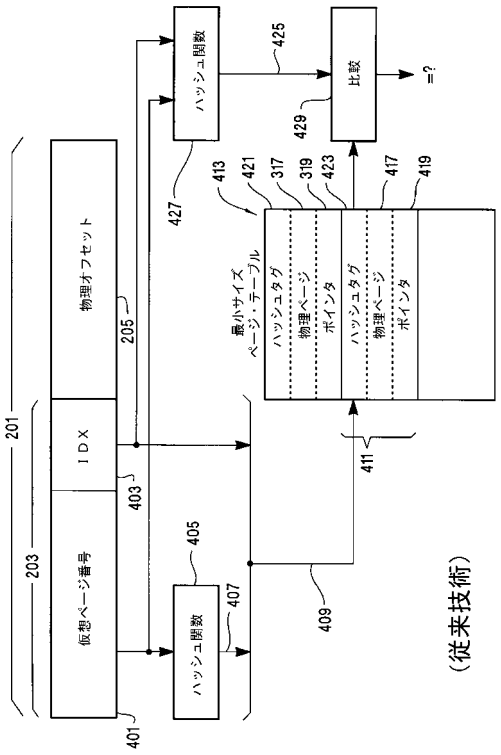


【図３】



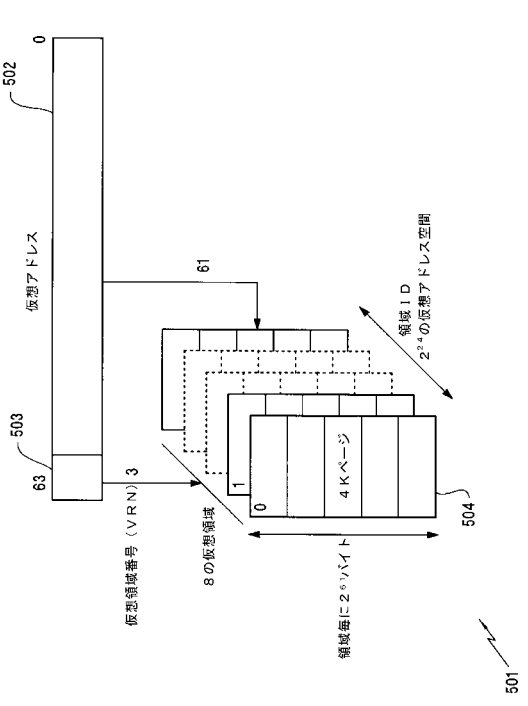
（従来技術）

【図４】

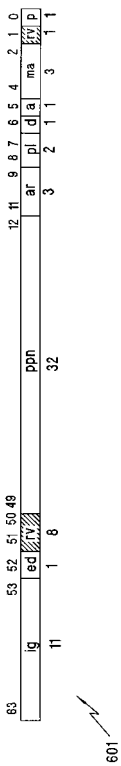


（従来技術）

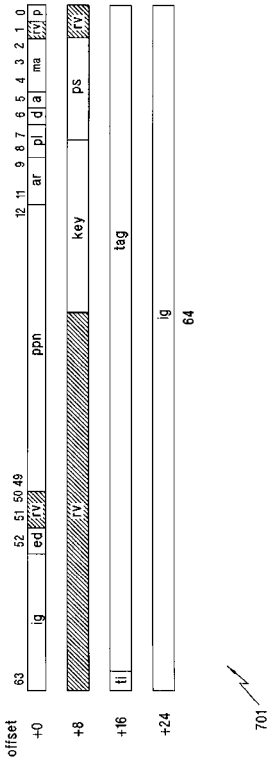
【図５】



【図６】



【図７】


offset
63
52 51 50 49
12 11
9 8 7
6 5 4
3 2 1 0
+0
ig
ed
rv
ppn
ar
pl
d
a
ma
rv
p
+8
rv
key
ps
rv
+16
ti
tag
+24
ig
64
701

フロントページの続き

(72)発明者 ステファン・ジー・バーガー
アメリカ合衆国９５０５０カリフォルニア州サンタ・クララ、フォーブス・アベニュー ２２５７
(72)発明者 ガリー・エヌ・ハモンド
アメリカ合衆国８０５２５コロラド州フォート・コリンズ、ソウグラス・コート ５１０１
(72)発明者 ジェームス・オー・ハイズ
アメリカ合衆国９５１２５カリフォルニア州サン・ノゼ、キャンベル・アベニュー １７２２
(72)発明者 ジェローム・シー・ハック
アメリカ合衆国９４３０３カリフォルニア州パロ・アルト、タリスマン・ドライブ ８５１
(72)発明者 ジョナサン・ケー・ロス
アメリカ合衆国９４０８７カリフォルニア州サニーヴェイル、ローン・ウェイ ９７４
(72)発明者 スニル・サクシーナ
アメリカ合衆国９４０８７カリフォルニア州サニーヴェイル、カーロウ・コート １５６
(72)発明者 コウイチ・ヤマダ
アメリカ合衆国９５１３４カリフォルニア州サン・ノゼ、エラン・ヴィレッジ・レーン ３７１、ナンバー１１６

審査官 清木 泰

(56)参考文献 特開２０００－１２２９２８（ＪＰ，Ａ）
特開２０００－１２２９２７（ＪＰ，Ａ）
特開平１０－２２８４１９（ＪＰ，Ａ）
特開平１０－１３３９５０（ＪＰ，Ａ）
特開平１０－０９１５２５（ＪＰ，Ａ）
特開平０７－０４９８１２（ＪＰ，Ａ）
特開平０４－３２３７４８（ＪＰ，Ａ）
特開昭６０－１２３９４７（ＪＰ，Ａ）
特開昭５６－１０５３８２（ＪＰ，Ａ）
国際公開第９８／０４４４１９（WO，Ａ１）
米国特許第０５８０９５６３（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F12/10