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

特許第4246922号
(P4246922)

(45)発行日　平成21年4月2日(2009.4.2)　(24)登録日　平成21年1月16日(2009.1.16)

(51)Int.Cl.　F I
G06F 12/12 (2006.01)　G06F 12/12 551
G06F 12/08 (2006.01)　G06F 12/12 557Z
G06F 12/08 507G
G06F 12/08 543B
G06F 12/08 523B

請求項の数 21　(全 15 頁)

(21)出願番号　特願2000-618844 (P2000-618844)
(86)(22)出願日　平成12年5月17日 (2000.5.17)
(65)公表番号　特表2002-544624 (P2002-544624A)
(43)公表日　平成14年12月24日 (2002.12.24)
(86)国際出願番号　PCT/US2000/013569
(87)国際公開番号　WO2000/070468
(87)国際公開日　平成12年11月23日 (2000.11.23)
審査請求日　平成17年1月12日 (2005.1.12)
(31)優先権主張番号　09/314,233
(32)優先日　平成11年5月18日 (1999.5.18)
(33)優先権主張国　米国(US)

(73)特許権者　500271546
インテル・コーポレーション
ＩＮＴＥＬ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　９５０５１　カリフォルニア、サンタ・クララ、ウォルシュ・アベニュ　２６２５
(74)代理人　100088454
弁理士　加藤　紘一郎
(72)発明者　ウイルカーソン，クリストファー，ビイ
アメリカ合衆国　オレゴン州　９７２０９　ポートランド　ホーリー・ストリート　ノースウエスト　１８０７

最終頁に続く

(54)【発明の名称】保護機能付き最近最低使用頻度置換方法

(57)【特許請求の範囲】
【請求項１】
キャッシュメモリシステムにデータを記憶させる方法であって、
キャッシュメモリに記憶させたデータのリストを含むキャッシュデータエントリーリストを作成し、
キャッシュデータエントリーリストを再使用リストとフィルタリストに分割し、
新しいキャッシュデータエントリーをフィルタリストに割り当てることによりフィルタリストにエントリーを記入し、
昇進プロセスにより再使用リストにエントリーを記入し、
選択されたキャッシュデータエントリーを再使用リストとフィルタリストのいずれかから捨てるように動作する保護プロセスの作動によりフィルタリスト及び再使用リストからキャッシュデータエントリーを排除するステップより成る、キャッシュメモリシステムへのデータ記憶方法。
【請求項２】
キャッシュデータエントリーの再使用リストとフィルタリストへの分割は、各キャッシュデータエントリーに関連する再使用ビットを所定の値にセットすることにより行われる請求項１の方法。
【請求項３】
再使用リストへのエントリーの記入は、フィルタリストの一部である間にアクセスされたキャッシュデータエントリーを再使用リストに昇進させることにより行われる請求項１

の方法。
【請求項４】
再使用リストへのエントリーの記入は、指名プロセスにより選択されたキャッシュデータエントリーを再使用リストに昇進させることにより行われる請求項１の方法。
【請求項５】
再使用リストへのエントリーの記入は、キャッシュデータエントリーをフィルタリストから再使用リストへランダムに昇進させることにより行われる請求項１の方法。
【請求項６】
キャッシュデータエントリーは、キャッシュメモリシステムに現在記憶されていないデータに対するリクエストであるキャッシュミスに限り応答して、フィルタリストから再使用リストへランダムに昇進される請求項５の方法。
【請求項７】
再使用リストへのエントリーの記入は、さらに、フィルタリストから、新しく割り当てられたキャッシュラインをランダムに選択して再使用リストへ含まれるようにすることである請求項１の方法。
【請求項８】
再使用リストまたはフィルタリストからのキャッシュデータエントリーの排除は、
再使用リスト及びフィルタリストの各々の長さを計算し、
再使用リスト及びフィルタリストの長さを所望の長さと比較し、
キャッシュメモリシステムに記憶させる新しいキャッシュデータエントリーを受け、
キャッシュデータエントリーを何れかのリストから排除して、新しいキャッシュデータエントリーの記憶場所を提供することより成り、
キャッシュデータエントリーは、計算した長さと所望の長さとの比較結果に応じて選択されたリストから排除される請求項１の方法。
【請求項９】
データを電子的に記憶させ、取り出すように作動可能で、キャッシュメモリを実現するコンピュータシステムであって、
キャッシュメモリへのデータの記憶は、
キャッシュメモリに記憶させたデータのリストを含むキャッシュデータエントリーリストを作成し、
キャッシュデータエントリーリストを再使用リストとフィルタリストに分割し、
新しいキャッシュデータエントリーをフィルタリストに割り当てることによりフィルタリストにエントリーを記入し、
フィルタリストからキャッシュデータエントリーを昇進させることにより再使用リストにエントリーを記入し、
選択されたキャッシュデータエントリーを再使用リストとフィルタリストのいずれかから捨てるように動作する保護プロセスの作動によりフィルタリスト及び再使用リストからキャッシュデータエントリーを排除するように行われるコンピュータシステム。
【請求項１０】
キャッシュメモリにデータを記憶させる方法であって、
キャッシュメモリに記憶させたデータのリストを含むキャッシュデータエントリーリストを作成し、
キャッシュデータエントリーリストを再使用リストとフィルタリストに分割し、
キャッシュデータエントリーリストの各キャッシュデータエントリーの使用順序を追跡し、
新しいキャッシュデータエントリーをフィルタリストに割り当てることによりフィルタリストにエントリーを記入し、
フィルタリストからキャッシュデータエントリーを昇進させることにより再使用リストにエントリーを記入し、
選択されたキャッシュデータエントリーを再使用リストとフィルタリストのいずれかか

ら捨てるように動作する保護プロセスの作動によりフィルタリスト及び再使用リストからキャッシュデータエントリーを排除するステップより成る、キャッシュメモリへのデータ記憶方法。

【請求項１１】

再使用リストへのエントリーの記入は、さらに、フィルタリストの一部である間にアクセスされたキャッシュデータエントリーを再使用リストに昇進させることにより行われる請求項１０の方法。

【請求項１２】

再使用リストへのエントリーの記入は、さらに、キャッシュデータエントリーがフィルタリストにあるが、フィルタリストにおける最近使用頻度が最も高いキャッシュデータエントリーではない間に使用されると、フィルタリストから再使用リストに昇進させることにより行われる請求項１０の方法。

【請求項１３】

再使用リストへのエントリーの記入は、さらに、キャッシュデータエントリーがフィルタリストにあるが、キャッシュデータエントリーリストにおける最近使用頻度が最も高いキャッシュデータエントリーではない間に使用されると、フィルタリストから再使用リストに昇進させることにより行われる請求項１０の方法。

【請求項１４】

再使用リストへのエントリーの記入は、さらに、フィルタリストから、新しく割り当てられたキャッシュラインをランダムに選択して再使用リストへ含まれるようにすることである請求項１０の方法。

【請求項１５】

フィルタリストから再使用リストへの昇進のための新しく割り当てられたキャッシュラインのランダムな選択は、再使用リストのサイズが大きくなるにつれてますます可能性が低くなる請求項１４の方法。

【請求項１６】

保護プロセスは、さらに、

キャッシュメモリに現在記憶されていないデータに対するリクエストであるキャッシュミスを、再使用リストの最近使用頻度が最も低いキャッシュデータエントリーが不変である間に、ストライクカウンタにおいてカウントし、

ストライクカウンタの値を最大許容ストライクしきい値と比較し、

ストライクカウンタの値が最大許容ストライクしきい値よりも大きい場合、キャッシュデータエントリーリストの最近使用頻度が最も低いキャッシュデータエントリーを排除するステップより成る請求項１０の方法。

【請求項１７】

保護プロセスは、さらに、

再使用リストの最近使用頻度が最も低いキャッシュデータエントリーを参照する毎にカウンタを増分し、

フィルタリストの最近使用頻度が最も低いキャッシュデータエントリーを参照する毎にカウンタを減分し、

カウンタの値に基づき再使用リストの最大許容サイズを計算し、

再使用リストの実際のサイズを計算し、

再使用リストの実際のサイズが再使用リストの最大許容サイズより大きい場合、キャッシュデータエントリーリストの最近使用頻度が最も低いキャッシュデータエントリーを排除し、

フィルタリストの最近使用頻度が最も低いキャッシュデータエントリーが存在し、且つ再使用リストの実際のサイズが再使用リストの最大許容サイズよりも大きくない場合、フィルタリストのかかるキャッシュデータエントリーを排除し、

フィルタリストが空であり、再使用リストの実際のサイズが再使用リストの最大許容サイズより大きくない場合、キャッシュデータエントリーリストの最近使用頻度が最も低い

キャッシュデータエントリーを排除するステップより成る請求項１０の方法。
【請求項１８】
保護プロセスはさらに、
カウンタの値をしきい値と比較し、
しきい値に対するカウンタの値に基づき再使用リストの最大許容サイズを発生させるステップより成る請求項１７の方法。
【請求項１９】
保護プロセスは、さらに、
フィルタリストの最近使用頻度が最も低いキャッシュデータエントリーへの中間的な参照なしに再使用リストの最近使用頻度が最も低いキャッシュデータエントリーを連続して参照すると、１つおきの参照毎にカウンタを増分し、
再使用リストの最近使用頻度が最も低いキャッシュデータエントリーへの中間的な参照なしにフィルタリストの最近使用頻度が最も低いキャッシュデータエントリーを連続して参照すると、１つおきの参照毎にカウンタを減分し、
カウンタの値に基づき再使用リストの最大許容サイズを計算し、
再使用リストの実際のサイズを計算し、
再使用リストの実際のサイズが再使用リストの最大許容サイズより大きい場合、キャッシュデータエントリーリストの最近使用頻度が最も低いキャッシュデータエントリーを排除し、
フィルタリストの最近使用頻度が最も低いキャッシュデータエントリーが存在し、再使用リストの実際のサイズが再使用リストの最大許容サイズよりも大きくない場合、フィルタリストのかかるキャッシュデータエントリーを排除し、
フィルタリストが空であり、再使用リストの実際のサイズが再使用リストの最大許容サイズより大きくない場合、キャッシュデータエントリーリストの最近使用頻度が最も低いキャッシュデータエントリーを排除するステップより成る請求項１０の方法。
【請求項２０】
キャッシュメモリにデータを記憶させるように作動可能な回路を有する集積回路であって、
キャッシュデータエントリーリストを再使用リストとフィルタリストに分割し、
キャッシュデータエントリーリストの各キャッシュデータエントリーの使用順序を追跡し、
新しいキャッシュデータエントリーをフィルタリストに割り当てることによりフィルタリストにエントリーを記入し、
フィルタリストからキャッシュデータエントリーを昇進させることにより再使用リストにエントリーを記入し、
選択されたキャッシュデータエントリーを再使用リストとフィルタリストのいずれかから捨てるように動作する保護プロセスの作動によりフィルタリスト及び再使用リストからキャッシュデータエントリーを排除するように動作する集積回路。
【請求項２１】
データを電子的に記憶させ、取り出すように作動可能で、キャッシュメモリより成るコンピュータシステムであって、
キャッシュメモリへのデータの記憶は、
キャッシュデータエントリーリストを再使用リストとフィルタリストに分割し、
キャッシュデータエントリーリストの各キャッシュデータエントリーの使用順序を追跡し、
新しいキャッシュデータエントリーをフィルタリストに割り当てることによりフィルタリストにエントリーを記入し、
フィルタリストからキャッシュデータエントリーを昇進させることにより再使用リストにエントリーを記入し、
選択されたキャッシュデータエントリーを再使用リストとフィルタリストのいずれかか

ら捨てるように動作する保護プロセスの作動によりフィルタリスト及び再使用リストからキャッシュデータエントリーを排除するように行われるコンピュータシステム。

【発明の詳細な説明】

【０００１】

【発明の分野】

本発明は、一般的に、電子コンピュータシステムのキャッシュメモリに記憶されたデータの置換に関し、さらに詳細には、望ましくない置換が起こらないように保護する最近最小使用頻度置換方法に関する。

【０００２】

【発明の背景】

キャッシュメモリは、データへの高速アクセスのため、高いシステム速度を必要とする高速で高価なメモリを備えた高性能コンピュータシステムに使用されることが多い。プロセッサの動作速度が速ければ速いほど、プロセッサはメモリからデータ及び命令を高速で取り出す必要がある。このため、高速でアクセス可能なメモリが必要となる。このメモリは、非常に高いクロック速度が使用されることが多いため、プロセッサに近いところに配置するのが一般的である。しかしながら、キャッシュとして動作可能な高速メモリは高価であり、大きなまたは特注のメモリをプロセッサに近いところに配置するのは実際的でない場合が多い。従って、限られた数のキャッシュメモリのバンクを、大きい主システムメモリとは別個に、プロセッサのコア近くに配置することが多い。

【０００３】

キャッシュメモリは、通常、スタティックランダムアクセスメモリ（ＳＲＡＭ）のような高速メモリと、キャッシュコントローラとより成る。コントローラは、プロセッサがどのデータを早期に必要とする可能性があるかの予測に応じて比較的低速の主メモリからキャッシュメモリへコピーされるデータを管理する。キャッシュメモリの容量は、システムの全メモリ容量の１０％と１％の間であることが多いが、部分的にコンピュータシステムのメモリアクセス特性の予測可能性に応じて大きな範囲で変化する。

【０００４】

通常、メモリへのアクセスはメモリアドレスの比較的小さな領域で次々に起こるため、アクセス頻度の最も高いデータをキャッシュメモリに記憶させると、システム性能を有意に改善することができる。使用頻度の最も高いデータが比較的高速のキャッシュメモリからアクセスされるようにすると、低速の主メモリからデータがアクセスされる間プロセッサが待機を余儀なくされることがなく、これは「キャッシュヒット」と呼ばれる。プロセッサが必要とするデータがキャッシュに存在せず、主メモリから取り出さなければならない場合、そのリクエストは同様に、「キャッシュミス」と呼ばれる。

【０００５】

キャッシュがメモリへのアクセス速度を事実上増加させる度合は、キャッシュヒットとキャッシュミスとのメモリリクエスト数により測定可能である。キャッシュコントローラ設計者の目標は、プロセッサが必要とする可能性が最も高いデータをキャッシュに配置して、キャッシュミスに対するキャッシュヒットの比率を最大限にすることである。かかる方式を用いると、システムは、高速メモリを備える利点を最大限に活用すると共に大部分のデータを比較的低価格の低速メモリに記憶させてシステムの全体コストを減少させることができる。

【０００６】

使用頻度の低いデータを使用頻度の高いデータに置換するかかる方式の１つは、キャッシュに記憶されたデータのうち最近の使用頻度が最低（ＬＲＵ）であるデータに着目してそのデータをキャッシュにロードされる新しいデータと置換することである。かかる方式は通常、データが最後にアクセスされた時からの時間を示すデータを含む、キャッシュに記憶された各データブロックに関連する多数のＬＲＵビットを使用する。かかる方式は、キャッシュのサイズに限りがあるため、再び必要になる前にデータを大きなデータセットから変位させる。さらに、ＬＲＵアルゴリズムは、最近の使用頻度の高いデータを、そのデ

ータの使用頻度が格段に低く、たとえただ１回使用されたものであったとしても、どちらかと言えば使用頻度の高いデータと置換する。従って、再びリクエストされる可能性の高いデータが、最近使用されたが使用頻度が低いデータに置換されないようにする方式が必要とされる。

【０００７】

【発明の概要】

本発明は、キャッシュメモリシステムにデータを記憶させる方法を提供する。キャッシュデータエントリーリストは、キャッシュメモリに記憶されるキャッシュデータエントリーを記録するが、このリストは、フィルタリストと、再使用リストに分割されている。フィルタリストは、フィルタリストに新しいキャッシュデータエントリーを全て記憶させることによりエントリーが行われ、また再使用リストは、キャッシュデータエントリーをフィルタリストから選択的に昇進させることによりエントリーが行われる。選択されたキャッシュデータエントリーを再使用リストとフィルタリストのいずれかから捨てるように動作する保護プロセスが作動すると、フィルタリストと再使用リストとから要素が排除される。

【０００８】

【実施例の詳細な説明】

本発明の以下の詳細な説明において、本願の一部であり、本発明の特定の実施例を例示する添付図面を参照する。これらの実施例は、当業者が本発明を実施できるように十分に詳しく記載されている。他の実施例も可能であり、本発明の範囲から逸脱することなく構造的、論理的及び電気的な設計変更を行うことができる。従って、以下の詳細な説明は限定的な意味に解すべきでなく、本発明の範囲は、頭書の特許請求の範囲と、特許請求の範囲が当然受けるべき均等物の全範囲とにより規定される。

【０００９】

ＬＲＵキャッシュメモリシステムを改良して、新たなキャッシュスペースが必要になると最近最低使用頻度のキャッシュデータエントリーをただ排除する場合に生じる問題の影響を減少する方法が必要とされている。本発明は、単純なＬＲＵ置換方式を改良するものであり、キャッシュサイズに限りがあるため、再使用の前に大きなデータセットからデータを排除する問題に対処するものである。また、使用頻度の高いデータを使用頻度は低いが最近使用されたデータと置換する問題に対処するものである。本発明は、ＬＲＵキャッシュのこれら共通の問題に対する解決法を提供する。

【００１０】

典型的なキャッシュメモリは、種々のバンクで同じ短縮アドレスを有する多数のメモリセルを備えており、これらのバンクは「ウェイ」と呼ばれる。例えば、アドレス長が１６ビットのシステムのメモリは、４ウェイのキャッシュを備えており、各々が８ビットのアドレスを有する４つのバンクのメモリより成る。主メモリからのデータをキャッシュに記憶させる場合、データの主メモリアドレスの８つの最上位ビットが捨てられて、残りの８ビットがデータが記憶されるキャッシュアドレスとなる。所望のアドレスを有する４つのウェイ、またはメモリバンクが存在するため、最近最低使用頻度のデータを有するウェイは、通常、データが記憶されるウェイとして選択される。これは、他のメモリのウェイに記憶された使用頻度の高いデータの前に、最近最低使用頻度のデータが必要とされる可能性が低いことによる。この仮定は常に正しい訳ではなく、ＬＲＵデータが実際、使用頻度が高い場合がある。多数の異なるキャッシュアーキテクチャ及びキャッシュ記入／置換アルゴリズムが存在するが、上記説明は、最近最低使用頻度（ＬＲＵ）置換アルゴリズムを用いる１つのかかるマルチウェイキャッシュの一例にすぎない。

【００１１】

本発明は、キャッシュメモリから排除されるデータが早期使用の可能性が最低ではないデータを含むウェイから選択される普通の方式である、容量による変位法及び関連性による変位法の問題を克服するように設計されたＬＲＵアルゴリズムを含む。

【００１２】

容量による変位は、データの使用頻度が低いため、最近最低使用頻度のデータの早期アクセス可能性が、最近使用頻度の高いウェイのデータよりも実際に大きい場合に起こる。例えば、容量による変位は、１６ｋＢのキャッシュに一部が記憶される３２ｋＢの大きさのデータより成る画像に対する画像編集プロセスに、その画像を反復してアクセスする必要がある場合に、影響を与える。画像を構成するデータバイトが次々にアクセスされ、アクセスと共にキャッシュに記憶される場合、キャッシュには常に１６ｋＢの大きさの画像情報が存在するが、そのプロセスは、キャッシュに実際に記憶された情報を再び必要とする前に、もう１つの１６ｋＢの画像情報をアクセスする必要がある。画像ファイルの第１の部分を再びアクセスする時、キャッシュは画像の第２の部分からの最近使用頻度の最も高いデータだけを含んでいる。同様に、プロセスが画像の第２の部分からのデータを必要とする時、そのデータは、画像の第１の部分からより最近アクセスされたデータと置換されている。かかるプロセスにはキャッシュミスが必ず大きな割合で起こるが、その理由は、キャッシュが反復使用される全てのデータを記憶する容量をただ備えていないためである。

【００１３】

データがキャッシュから排除されるもう１つの普通の方式は、関連性による変位である。これは、ある特定のデータが、繰り返し必要とされるが、使用頻度の低い多量のデータがアクセスされるため、排除される時に起こる。例えば、データ構造Ａと、データ要素Ｂとを取り扱うただ４つのエントリーを有するキャッシュを考える。データがＡ１、Ａ２、Ａ３、Ａ４、Ａ５，Ｂ、Ａ６、Ａ７、Ａ８、Ａ９、Ｂ、Ａ１０、Ａ１１、Ａ１２、Ａ１３、Ａ１４、Ｂなどの順序でアクセスされる場合、キャッシュはＢを決して含まず、データ構造Ａの最近使用頻度の高い要素を含む。Ｂはデータ構造Ａの何れの単一要素より使用頻度は格段に高いが、従来のＬＲＵアルゴリズムはＢのようなアクセス頻度の高いデータは優先しない。

【００１４】

ＬＲＵアルゴリズムを実現するためには、キャッシュコントローラは、ウェイ内の特定のアドレスについて何れのウェイが最近使用頻度が最も高いかを示すデータを処理しなければならない。キャッシュコントローラは、何れのウェイが最近使用頻度が最も高いかに関するデータを記憶させると共に、そのデータを復号して、何れのウェイが、最近使用頻度が最低であり、新しく使用されるデータと置換しなければならないかを決定する役割を有する。図１は、４ウェイキャッシュの特定のアドレスについて何れのウェイの最近使用頻度が最低かを追跡する１つの方式を部分的に示す表である。

【００１５】

図１は、各ウェイ内の特定のアドレスの使用順序を記録するために使用可能な要素（ｉ，ｊ）の二次元マトリックスより成る表であり、この要素（ｉ，ｊ）は、ウェイ（ｉ）のデータがウェイ（ｊ）のデータより使用頻度が高い場合は１であり、ウェイ（ｉ）のデータがウェイ（ｊ）のデータより使用頻度が低い場合は０である。マトリックスの対角線上要素であるアドレス（０，０）乃至（３，３）は、（ｉ）及び（ｊ）が共に同じウェイを参照するため、無視できるから、意味のある情報を含まないことに注意されたい。また、この対角線の一方の側の要素は他方の側の要素の逆数であるため、（３，２）が１で、ウェイ３がウェイ２よりも使用頻度が高いことを意味する場合、ウェイ２の最近使用頻度はウェイ３より高いことはありえず、（２、３）は論理０でなければならない。従って、対角線の一方の側の要素は対角線と共に捨てることが可能であり、使用順序の記録に重要なマトリックスの数で表される要素だけを残す。従って、これら５個の数字を付したマトリックス要素が、５個のマトリックス要素（ｉ，ｊ）のアレイＬで表される５ビットの列により表され、Ｌ＝｛（０，１），（０，２），（１，２），（０，３），（１，３），（２，３）｝であり、各ビットは、要素（ｉ）が要素（ｊ）よりもより最近使用頻度が高ければ１を含む。アレイＬのこれら５つのビットは、単一の数で容易に参照することができるため、Ｌ＝（０，１，２，３，４，５）であり、各ビットはアレイＬの前のマトリックス表示で与えられる順序のマトリックス要素に対応する。

【００１６】
図２は、新しいデータにアクセスし、新しくアクセスしたデータをキャッシュに書き込む時、アレイＬの５つのビットを更新するためにキャッシュのこの例の動作に用いる更新表を示す。この更新マトリックスは、新しくアクセスするデータを記憶させるために何れのウェイを使用するかに応じて、Ｌの何れのビットを特定のアドレスに対して変化させるかを示す。キャッシュは４ウェイキャッシュであり、４つのウェイのうちの任意の１つがデータを記憶させるために選択可能であるため、マトリックスの１つの軸はキャッシュシステムの４つのウェイを表す。もう１つの軸は、４つのウェイのうち最近使用頻度が最も高いウェイにデータを記憶させ、もう１つの軸に表されるようにアクセスされるウェイに対応する、アレイＬの５つのビットを表す。例えば、ウェイ２がワードＬのチェックにより特定のアドレスについて最近使用頻度が最も低いと判定される動作では、ウェイ２は入来データと置換される。ウェイ軸のウェイ２を参照して、マトリックスは、ビット０、３及び４は不変であるが、ビット１及び２は０にセットされ、ビット５は１に変化していることを示す。アレイＬのマトリックス要素表示法を再び参照して、その式は、ビット１及び２を０にセットすると、ウェイ０がウェイ２よりも最近使用頻度が低く、ウェイ１がウェイ２よりも最近使用頻度が低いことを示す。同様に、ビット５を１にセットすると、ウェイ２がウェイ３よりも最近使用頻度が高いことを示す。アレイＬのマトリックス表示法をさらに検討すると、ビット０、３及び４はウェイ２についての情報を含まず、そのため不変であることがわかる。キャッシュへの書き込みの毎にこの方法に従ってビットを更新することにより、アレイＬは、キャッシュのウェイの使用順序についての正確な情報を常に含むことになるため、キャッシュコントローラが新しいデータをキャッシュの最近使用頻度が最も低いウェイに割り当てるために使用することができる。

【００１７】
キャッシュコントローラは、図２の更新表の補足物に過ぎない犠牲者選択表を用いてこれを行う。図３は、マトリックスの全ての１が０で置換され、全ての０が１で置換され、全ての不変ビットが不変のままであるような、この例のキャッシュの犠牲者選択表を示す。マトリックスから読み取られる要素の補足物の発見は、エレクトロニクスまたは他の手段により全く容易に行うことが可能であるため、犠牲者選択表を維持する必要はないが、図２の更新表からデータを取り出しながら変換することによって行うことができる。図３の各要素は動作時にマスクとして使用されるため、マスクの内容がアレイＬに記憶されたデータと比較される。４つのマスクのうちの任意のもののデータがアレイＬのデータとマッチする場合、そのマスクに関連するウェイが、最近使用頻度が最も低く新しいデータの目標となるウェイとして選択される。図３のマスクが、ウェイ間で互いに排他的なビットデータを含むため、４つのマスクのうち１つだけがアレイＬの任意可能な組み合わせのビットにマッチし、特定のマスクを最近使用頻度が最も低いものとして指示することに注意されたい。これは、マスクとアレイＬとは同時に比較可能であることを意味し、ただ１つのウェイだけがキャッシュの書き込み動作のための目標として選択される。

【００１８】
例えば、特定のアドレスのウェイが２、３、０、１の順序で使用されており、ウェイ１が最近使用頻度が最も高く、ウェイ２が最近使用頻度が最も低い場合を考える。図２の更新表を用いると、全ての更新を行った後、アレイＬはデータＬ＝（０，１，１，１，１，０）を含む。このワードを図３のマスクと比較すると、ウェイ０では、ビット０、１及び３が不正確であり、ウェイ１では、ビット１及び３が不正確であり、ウェイ２では、ビット１、２、５が不正確であることがわかる。ウェイ２では、犠牲者選択マトリックスのマスクのビット１、２及び５はアレイＬのデータとマッチするため、ウェイ２は最近使用頻度が最も低いウェイとして適正に選択される。

【００１９】
前の段落で説明した例は、キャッシュシステムにおけるＬＲＵ置換の一例であり、前述したような普通のＬＲＵ置換方式の短所を有する。本発明は、上述した方式を改良するもので、関連性及び容量による変位のありふれた問題に対処するものである。

【００２０】
保護機能付きＬＲＵ、即ち、ＬＲＵＰは、従来のＬＲＵウェイリストを少なくとも２つのサブリストに分割したものである。１つの実施例では、従来のＬＲＵウェイリストを改良するために、各ウェイが、そのウェイに記憶されるデータが再使用サブリストまたはフィルタサブリストの一部であるかを示す再使用／フィルタビットを含むようにする。ＬＲＵの順序は大域的に維持されるため、ＬＲＵウェイから最近使用頻度が最も高いウェイまでの任意のウェイは、大域的リストに加えて、再利用リストまたはフィルタリストのいずれか１つの一部である。再使用リスト及びフィルタリストのデータを別個に管理する種々のプロセスは、その後、変位の問題に対処し、ＬＲＵＰキャッシュの性能を最適化するように調整される。

【００２１】
本発明の１つの実施例において、キャッシュ要素を、フィルタリストの一部として最初に割り当てる。フィルタリストの選択要素を、指名プロセスにより再使用リストに昇進させ、再使用リストへエントリーとして記入する。指名によりキャッシュラインがフィルタリストから再使用リストへ移動するため、指名を引き続き行うと、再使用リストが無限に大きくなって、最終的にキャッシュ要素がフィルタリストから押し出されることになる。従って、保護プロセスは、入来データを記憶させるために新しいキャッシュラインを割り当てると、再使用リストまたはフィルタリストの間で排除するメンバーの選択を行うことにより、フィルタリストと再使用リストの間で要素が適正に分配されるようにする。

【００２２】
フィルタリストから再使用リストへの要素の指名は、望ましくない変位を減少させるように調整された多数のプロセスにより行われる。関連性による変位を減少させるトリガーによる指名は、記憶されるデータがフィルタリストにある間に再使用されるなら、キャッシュラインをフィルタリストから再使用リストへ移動させることを含む。容量による変位を減少させるランダムな指名では、再使用リストに含まれるようにするために、フィルタリストから、新しく割り当てられたキャッシュラインがランダムに選択される。トリガーによる指名とランダムな指名との組み合わせ及び変更は、本発明の範囲内に含まれると考えられ、かかるものの幾つかの実施例を以下において説明する。

【００２３】
トリガーによる指名は、ある特定の要素を再使用リストに昇進させるのがそれほど望ましくない特定の状況を評価することにより、フィルタリストにある間使用されるキャッシュラインを再使用リストに昇進させるという簡単な規則を変更するものである。例えば、大域リストの最近使用頻度の最も高い位置にある間に参照される要素は、単一の演算または計算の一部として再アクセスされるため、２つの異なる時間にアクセスされる要素より再アクセスの可能性が低い。これらの要素は、その後、昇進から除外されるが、後で再使用または排除されるまでは、フィルタリストの一部として留まる。同様に、フィルタリストの最近使用頻度が最も高い位置にある間に参照される要素は、トリガーによる指名の間、再使用リストへの昇進から除外されるか、または、大域リストまたはフィルタリストの何れかの最近使用頻度の高い多数の位置にある要素が指名から除外される。

【００２４】
ランダムな指名は、ランダムに選択したキャッシュラインを再使用リストへ指名することを含む。通常のランダムな指名の変形例は、大域リストに対する再使用リストのサイズをモニターし、再使用リストが大きくなるにつれてランダムな指名をますます可能性の低いものとするランダムな選択アルゴリズムを用いることを含む。上述の例に加えて、データをランダムに選択するための多数の他の方法が存在するが、これらは本発明の範囲内に含まれるものと考えられる。

【００２５】
トリガーによる指名とランダムな指名との組み合わせを用いて、関連性による変位及び容量による変位の両方に対処することが可能であり、これら両方のアルゴリズム及びそれらの変形例を、特定のキャッシュ構成についてキャッシュヒットとキャッシュミスの比率

を最適化するように選択する。例えば、本発明との間に一貫性がある１つの指名プロセスの実施例は、再使用リストのサイズが増大するにつれてランダムな指名の可能性がますます低くなるランダムな指名アルゴリズムと共に、フィルタリストの最近使用頻度の最も高い要素を排除するトリガーによる指名方式を使用する。かかる組み合わせによると、トリガーによる指名アルゴリズムによる指名と、ランダムな指名アルゴリズムによる指名とが自動的に平衡するという利点があるが、その理由は、ランダムな指名アルゴリズムが再使用リストのサイズに依存するからである。これにより、再使用リストのサイズの安定化が得られると共に、関連性による変位と容量による変位の両方への対処を依然として効果的に行うことができる。

【００２６】

再使用リストへのエントリーの記入は、データの特定の特性に応じてキャッシュデータのエントリーを再使用リストに昇進させる分類方式などのような、他の指名プロセスにより行うことができる。種々の指名プロセスにより昇進されるキャッシュラインは、再使用リストからデータを昇進させるか、または新しく割り当てられたキャッシュデータエントリーを昇進させるか、もしくは別の方法により選択されたキャッシュデータエントリーを昇進させる。指名プロセスを組み合わせて用いることにより、容量及び関連性による変位、あるいはキャッシュを非効率にする他の問題に対処することができる。再使用リストへの昇進のためにキャッシュラインを選択する他の指名プロセスが存在し、また展開可能であると予想されるが、これらは本発明の範囲内であると考えるべきである。

【００２７】

保護アルゴリズムは、大域リストまたはフィルタリストから排除される要素を選択することにより、再使用リストのサイズをさらに制御する。大域リストはフィルタリストと再使用リストの両方から構成されるため、大域リストからの排除は、再使用リストまたはフィルタリストの何れかからの排除であることに注意されたい。２つの保護アルゴリズム、即ち、ストライクアルゴリズムと、ＬＲＵカウンタアルゴリズムについて説明する。

【００２８】

ストライクアルゴリズムは、ストライクカウンタを用いて、再使用リストの最近使用頻度が最も低い要素が不変である間にメモリへのアクセスがキャッシュミスとなった回数を追跡する。最大許容ストライクしきい値を予めセットし、ストライクカウンタの値と比較して、ストライクカウンタの値が最大許容ストライクしきい値を超えると、大域リストのＬＲＵ要素が排除されるようにする。再使用リストにおける要素の寿命は、再使用リストを大きくするか、または最大許容ストライクしきい値を増加させることにより延長可能であるが、最大許容ストライクしきい値を大きくすると、大域リストの要素の寿命が増加する。従って、比較的大きな再使用リストでは、最大許容ストライクしきい値を低くするのが望ましく、比較的小さい再使用リストでは、このしきい値を高くするのが望ましい。従って、ストライクアルゴリズムの１つの実施例は、再使用リストのサイズに応じてストライクしきい値を変化させる方法を使用する。

【００２９】

ＬＲＵカウンタ保護アルゴリズムは、フィルタリストからのキャッシュラインの排除に対する再使用リストからのキャッシュラインの排除コストを予想しようとする。カウンタは、フィルタリスト及び再使用リストの両方のＬＲＵ要素への参照を追跡するため、各リストのＬＲＵ要素がアクセスされる可能性の見積りを比較することができる。かかる方式の１つの実施例は、０でスタートし、再使用リストのＬＲＵ要素が参照される毎にカウントアップし、フィルタリストのＬＲＵ要素が参照される毎にカウントダウンするカウンタを使用する。別法として、ヒステリシスビットをカウンタに追加して、フィルタリストまたは再使用リストのいずれか一方のリストのＬＲＵビットが連続して（もう一方のリストのＬＲＵへの中間的な参照なしに）参照されると、１つおきの参照のたびにカウンタが増分または減分されるようにする。

【００３０】

排除が必要な場合、ＬＲＵのカウンタの値を用いて、再使用リストの最大許容サイズを

求め、再使用リストまたはフィルタリストのいずれからキャッシュラインを排除すべきであるかを決定する。一例として、１つの実施例で排除が必要であり、カウンタが再使用リストへの頻繁なアクセスを示す正の数をホールドするケースを考える。その場合、ＬＲＵカウンタの値を、再使用リストのしきい値と比較する。カウンタ値がしきい値を越える場合、所定の大きなサイズのリストが示唆されており、カウンタ値がしきい値よりも小さい場合、所定の小さいサイズのリストが必要とされる。その後、選択された所定のリストサイズを再使用リストの実際のサイズと比較し、所定のサイズが再使用リストの実際のサイズよりも小さい場合に限り、再使用リストからラインを排除する。そうでなければ、フィルタリストからラインを排除する。表は、多数の所定しきい値と、それらに対応する再使用リストのサイズとを含み、その目的は再使用リストのサイズ及びそのリストからの排除の最適化にあることに注意されたい。かかる方式は、再使用リストのサイズを制御するだけでなく、フィルタリストまたは再使用リストの何れにあるかに関係なく、使用される可能性が最も低いデータを排除する作用を有する。

【００３１】

動作時、再使用リストとフィルタリストとを管理するには、各キャッシュ要素が何れのリストの一部であるかを示すデータビットを使用する必要がある。高速キャッシュを最適動作させて、大きなキャッシュヒット対ミスの比率から得られる利点を得るためには、各リストの順序付けを、それぞれ独立に、且つそれらを組み合わせた大域リストとして、迅速且つ効率良く行わなければならない。

【００３２】

再使用リスト指示ビットを使用するＬＲＵＰキャッシュの実施例を、かかるキャッシュの再使用リスト及びフィルタリストを管理する１つの方法の一例として、ここで説明する。

【００３３】

再使用リストまたはフィルタリストのメンバーであるキャッシュ要素間の識別は、かかるキャッシュにおいて、各キャッシュ要素に関連する再使用ビットをチェックすることにより容易に行うことができる。例えば、図４において、キャッシュ要素が再使用リストのメンバーである場合、再使用ビットは、ウェイ１及び２のように、１にセットされている。要素がフィルタリストのメンバーである場合、再使用ビットは、ウェイ０及び３のように、０にセットされている。キャッシュコントローラが、ＬＲＵＰキャッシュの一部として説明するプロセスの１つを実行するために、フィルタリストまたは再使用リストの何れのＬＲＵウェイかを判定する必要がある場合、キャッシュコントローラが求める順序の少なくともそのリストについて使用順序を維持して、大域リストを２つのリストに分離しなければならない。このプロセスを、サブリスト順序抽出と呼び、かかるプロセスの一例を以下において説明する。

【００３４】

この例の方法におけるサブリスト順序抽出の目的は、大域リストのＬＲＵ要素が抽出を受けるサブリストのＬＲＵ要素であるように、順序を変更した大域リストを作成することである。大域リストのＭＲＵ位置へ、抽出されないサブリストの要素を再び挿入する方式についてここで説明するが、この方法は、抽出されるリストの順序を保持して、リストのＬＲＵビットが正しく識別されるようにするが、抽出されないリストの要素順序情報は失われる。

【００３５】

図４を再び参照して、再使用リストのＬＲＵ要素の抽出について説明する。抽出されるサブリスト（再使用リスト）は、関連の再使用ビットにより示されるように、要素１及び２より成り、要素０及び３が抽出中でないサブリストのメンバーであることも示す。図４の要素０－４を表すために数字を付した各行は、リストのメンバーが新しいアレイを引き出すために抽出中でない場合、アレイＬに適用されるマスクを表す。この新しいアレイは、ＬＲＵが抽出中のサブリストのＬＲＵである順序を変更した大域リストを表す。

【００３６】

例えば、「古い」と記した行は、元のＬＲＵリストを表し、３、２、１、０の順序を示すが、３は最近使用頻度が最も低く、０は最近使用頻度が最も高いキャッシュ要素である。再使用リストを抽出するため、再使用リストの一部でない要素０及び３のマスクがアレイＬに適用される。これにより、「新しい」と記した行に示す新しいアレイが生まれるが、これは抽出中のリストの要素の順序を反映する一方、抽出中でないリストの要素がＭＲＵ位置にあることをアレイが示すようにする。しかしながら、抽出中のリストの一部でない要素のマスクが、古いアレイＬに、アレイＬのそれらの古い位置を表さない順序で適用されるため、抽出中でないリストの順序情報が失われる。

【００３７】

この特定の例では、新しいアレイ（０、０、０、０、１、１）は２、１、０、３の順序を表すが、２はＬＲＵ要素、３はＭＲＵ要素である。ウェイ０及び３のマスクがランダムな順序で適用されたため、新しいアレイは、０が実際に３より最近使用頻度が高い時、３が０より最近使用頻度が高いことを示す。しかしながら、この情報はアレイＬに保管されず、また抽出中のリストの順序の維持だけに興味があるため、このタイプのエラーは本願にとっては重要ではない。

【００３８】

あたらしいアレイの作成によるサブリストの抽出は、１つの例において組み合わせ論理を用いることにより行うことが可能であり、従って、非常に迅速に完了することができる。サブリストの順序を異なる方式で取り出す他の実施例が存在するが、それは、抽出中でないサブリストの順序を保持することを含む。サブリストを抽出するについて新しいアレイを作成し使用した後、新しいアレイは古いアレイＬに書き戻さない。従って、アレイＬはそのままであり、実際の要素の順序が保持され、必要に応じて組み合わせ論理のような方法により、サブリストの非破壊的抽出を繰り返し行うことができる。

【００３９】

ここで説明するようなキャッシュシステムは、図５に示すシステムにおけるようなコンピュータに使用されるプロセッサの一部である。キャッシュシステムはプロセッサコアの一部である必要はないが、大きなプロセッサモジュールの一部として、別個のデバイスとして、またはプロセッサへ接続されるものとして実現可能である。システムは、システムバス５０２に取り付けられたプロセッサコア５０１を有する。システムバスとプロセッサは、キャッシュコントローラ５０３によりリンクされ、このコントローラはキャッシュ５０４にも接続されている。バスは、ハードディスクドライバ５０５のようなデータ記憶装置に接続され、またポート５０６のような１またはそれ以上の入出力ポートに接続されている。キャッシュコントローラは、プロセッサからメモリフェッチリクエストを受信し、リクエストされたデータがそこに記憶されておれば、キャッシュからリクエストに対応し、そうでなければ、主メモリからそのリクエストに対応する。キャッシュコントローラは、前述したように、本発明に従って、キャッシュ要素置換のＬＲＵＰ方式により、キャッシュ５０４におけるデータを維持する。

【００４０】

キャッシュコントローラ５０３のようなキャッシュコントローラは、集積回路上の論理回路として物理的に実現して、集積回路の少なくとも一部が、前述したように、ＬＲＵＰ方式によりキャッシュにおいてデータを維持するように作動可能である。例えば、上述したように、キャッシュを再使用リストと、フィルタリストに分割し、再使用リストと、フィルタリストにエントリーを記入し、再使用リスト及びフィルタリストからデータエントリーを排除する機能を実行可能なモジュールを備えた集積回路は、本発明の実施例と一貫性を有するものである。特に、本発明は、プロセッサまたはプロセッサモジュール、マザーボードチップセット、キャッシュコントローラチップ、または他の集積回路の少なくとも一部より成る回路で実施されることを企図されており、これらは全て本発明の範囲内にあると考えるべきである。

【００４１】

図６は、データ記憶サブシステムを示す。このシステムは、コンピュータシステムのシス

テムバス６０２に記憶コントローラ６０１が接続されたものである。記憶コントローラは、多量のデータの非揮発性記憶装置として一般的に使用される、１またはそれ以上のハードディスクドライブ６０３にも接続されている。記憶コントローラはさらに、キャッシュ６０４に接続され、このキャッシュは、ハードディスクドライブ６０３よりも迅速にデータの記憶及び取り出しを行なうことができるが、一般的に高価で小容量である。記憶コントローラ６０１は、ハードディスクドライブからのデータを記憶し、取り出すリクエストに応答し、キャッシュ６０４に使用頻度の高いデータを維持する。キャッシュコントローラは、本願のＬＲＵＰキャッシュの説明に従って、キャッシュのデータを管理する。

【００４２】

上述したような保護機能付きＬＲＵは、従来のＬＲＵ置換方式の共通の問題に対する解決法を与えることにより、従来のＬＲＵ置換方式に対して実質的な性能上の利点を提供する。キャッシュエントリーリストをフィルタサブリストと、再使用リストに分割することにより、エントリー記入及び保護プロセスを実行して、関連性及び容量による変位を減少させることができる。新しいキャッシュエントリーは最初にフィルタリストに記憶され、フィルタリストから昇進したエントリーが再使用リストに記入される。フィルタリスト及び再使用リストからの排除は、フィルタリスト、再使用リストまたは大域リストからデータエントリーを排除する保護プロセスにより行われる。保護及び排除プロセスの多くの変形例を、それぞれが従来のＬＲＵ置換方式に存在する望ましくない変位の問題の影響を軽減することにより得られる利点と共に説明した。

【００４３】

特定の実施例を図示説明したが、当業者にとっては、同じ目的を達成する任意の構成を図示の特定の実施例に置き換えることが可能であることがわかるであろう。本願は、本発明の任意の変形例または設計変更を包含するように意図されている。本発明は、特許請求の範囲及びその均等物の全範囲によってのみ限定されることを意図されている。

【図面の簡単な説明】

【図１】　図１は、本発明の実施例と一貫する、最近最小使用頻度（ＬＲＵ）順序マトリックスを表す表である。

【図２】　図２は、本発明の実施例と一貫する、特定のウェイのデータへのアクセスに応答して変化するマルチウェイのＬＲＵ置換キャッシュワードのビットを表す表である。

【図３】　図３は、本発明の実施例と一貫する、マルチウェイＬＲＵキャッシュ置換における犠牲者選択に使用する復号マスクを表す表である。

【図４】　図４は、本発明の実施例と一貫する、サブリスト順序付けマスクと、関連のＬＲＵウェイ保護ビットを備えたＬＲＵワードへのかかるマスクの適用例とを表す表である。

【図５】　図５は、本発明の実施例と一貫する、プロセッサ、メモリ及びＬＲＵＰキャッシュを備えた汎用コンピュータを示す。

【図６】　図６は、本発明の実施例と一貫する、ＬＲＵＰキャッシュを組み込んだデータ記憶システムを示す。

【図１】

LRU ウェイ表

i \ j	0	1	2	3
0	–	0	1	3
1		–	2	4
2			–	5
3				–

L={(0,1),(0,2),(1,2),(0,3),(1,3),(2,3)}
OR L−{0,1,2,3,4,5}

【図２】

更新表

使用されるウェイ \ 変化したビット	0	1	2	3	4	5
0	1	1	–	1	–	–
1	0	–	1	–	1	–
2	–	0	0	–	–	1
3	–	–	–	0	0	0

1= 最近使用頻度が高い

【図３】

犠牲者選択復号表

検知されたウェイ \ Lビットマスク	0	1	2	3	4	5
0	0	0	–	0	–	–
1	1	–	0	–	0	–
2	–	1	1	–	–	0
3	–	–	–	1	1	1

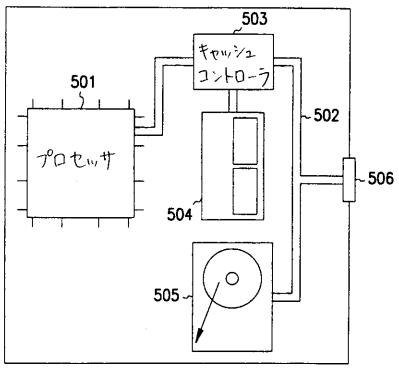
【図４】

LRUP 大域リストウェイ表

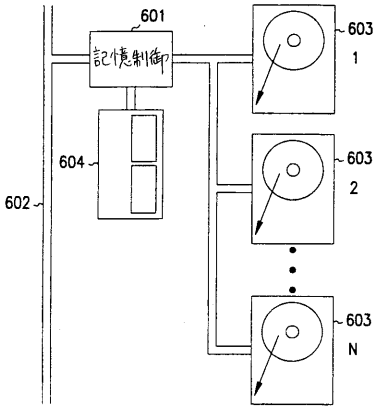
ウェイ \ Lビット	0	1	2	3	4	5	再使用
0	0	0	0	–	–	–	0
1	1	–	–	0	0	–	1
2	–	1	–	1	–	0	1
3	–	–	1	–	1	1	0
例：（マスク 0.3 を適用）							
古い	0	0	0	0	0	0	0
新しい	0	0	0	0	0	1	1

XX=0
X1=1
01=0
0X=0
00=0
11=1

【図５】



【図６】



フロントページの続き

(72)発明者　ウェード，ニコラス，デイ
アメリカ合衆国　オレゴン州　９７２２９　ポートランド　エイティファースト・ストリート　ノースウエスト　２７３０

審査官　清木　泰

(56)参考文献　特開平０４－０９２９４１（ＪＰ，Ａ）
特開平１１－３３８７７５（ＪＰ，Ａ）
特開平１０－３０７７５６（ＪＰ，Ａ）
特開平０６－１６１８９８（ＪＰ，Ａ）
特開平０５－１８９３１６（ＪＰ，Ａ）
特開平０３－０７１３４７（ＪＰ，Ａ）
特開平０２－２８１３５０（ＪＰ，Ａ）
米国特許第０５７７８４４２（ＵＳ，Ａ）
Ramakrishina Karedla, J. Spencer Love, Bradley G. Wherry, ＨＤＤの性能を改善するためのキャッシュ方式を提案, 日経エレクトロニクス, 日本, 日経ＢＰ社, １９９４年　9月１２日, No:617, Pages:107-121
Method for Improving Least Recently Used Buffer Performance Using Multiple Stack Insertion Points Based on Data Types, IBM Technical Disclosure Bulletin, 米国, International Business Machines Corporation, １９９３年　8月, Vol:36, No:08, Pages:479-480

(58)調査した分野(Int.Cl., ＤＢ名)
G06F12/08-12/12