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

特許第4065586号
(P4065586)

(45)発行日　平成20年3月26日(2008.3.26)　(24)登録日　平成20年1月11日(2008.1.11)

(51)Int.Cl.		F I		
G06F 12/08	(2006.01)	G06F	12/08	519D
G06F 12/00	(2006.01)	G06F	12/08	531E
G11C 15/04	(2006.01)	G06F	12/00	571A
		G11C	15/04	631W

請求項の数 7　(全 23 頁)

(21)出願番号　特願平9-264305
(22)出願日　平成9年9月29日(1997.9.29)
(65)公開番号　特開平10-133943
(43)公開日　平成10年5月22日(1998.5.22)
審査請求日　平成16年9月22日(2004.9.22)
(31)優先権主張番号　734-003
(32)優先日　平成8年10月18日(1996.10.18)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT－PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　３０００
(74)代理人　100075513
弁理士　後藤　政喜
(74)代理人　100084537
弁理士　松田　嘉夫
(72)発明者　ソリン・ラコボヴィッチ
アメリカ合衆国　カリフォルニア，サン・ジョセ，カウンテス・ドライヴ　５９９０

最終頁に続く

(54)【発明の名称】リンクリスト形成方法

(57)【特許請求の範囲】

【請求項１】

キャッシュコヒーレンシ機構を有するメインメモリと、

複数のリンクリストを格納する要求待ち行列を有するメモリコントローラと

を有する計算システムであって、

前記リンクリストのそれぞれは、第１の項目と、複数の追加項目とを有し、

前記追加項目のそれぞれは、前記リンクリスト中の、先にエントリされた項目に対するリンク参照を含み、前記追加項目のそれぞれ用のリンク参照は、すべて連想記憶（ＣＡＭ）に格納され、これにより、次の追加項目中にある、先にエントリされた項目に対する前記リンク参照を用いて連想探索を行うことにより、次の追加項目へのアクセスが可能となることを特徴とする計算システム。

【請求項２】

前記連想記憶は、前記追加項目ごとに、当該の追加項目が有効であるか否かを示す有効性ビットも格納することを特徴とする請求項１に記載の計算システム。

【請求項３】

前記追加項目のそれぞれに対する追加情報がランダムアクセスメモリに格納され、

前記追加情報は、メモリラインに対して実行される動作に関するものを含むことを特徴とする請求項１に記載の計算システム。

【請求項４】

前記連想記憶中には、前記第１の項目および前記複数の追加項目のそれぞれに対して頭

部フィールド、尾部フィールド、アドレスフィールド、有効性ビットが格納され、
前記頭部フィールドは、前記第１の項目に対しては「真」に対応する値を、前記複数の追加項目のそれぞれに対しては「偽」に対応する値を含み、
前記尾部フィールドは、前記リンクリスト中の最後の項目に対しては「真」に対応する値を含み、
前記アドレスフィールドは、メモリラインのアドレスを含み、
前記有効性ビットは、当該の第１の項目または追加項目が有効であるか否かを示す
ことを特徴とする請求項１に記載の計算システム。
【請求項５】
前記第１の項目に対して頭部待ち行列連想記憶にアドレスフィールドと有効性ビットとが格納され、
前記アドレスフィールドは、メモリラインのアドレスを含み、
前記有効性ビットは、当該の第１の項目が有効であるか否かを示し、
前記頭部待ち行列連想記憶が、前記連想記憶に追加されることを特徴とする請求項１に記載の計算システム。
【請求項６】
前記第１の項目の追加情報がランダムアクセスメモリに格納され、
前記追加情報は、
メモリラインに対して実行される動作と、
前記第１の項目が前記リンクリスト中において最後の項目であるときに、「真」に対応する値を含む最終フィールドと、
「有効」であるときに、前記リンクリスト内の最後の項目のインデックスを含む尾部フィールドと
を含むことを特徴とする請求項５に記載の計算システム。
【請求項７】
前記キャッシュコヒーレンシ機構はディレクトリ構造であることを特徴とする請求項１に記載の計算システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、計算装置に係り、とりわけ連想記憶を使用するリンクリストの形成に関する。
【０００２】
【従来の技術】
リンクリストは、多数の応用分野で有用な構造である。そのような応用分野の１つが、多重プロセッサ（ＭＰ）システムでのキャッシュコヒーレンシの確保である。同一のメモリラインへのアクセスをリンクリストに編成することによって、衝突が存在する場合でもこれらの要求を到着順にサービスできるようになる。これによって、公平さが保証され、再試行または他の方法を使用して衝突を解決する時に発生する可能性がある長時間の停止状態の問題が防止される。
【０００３】
キャッシュメモリとは、主記憶の内容のうち、近い将来にプロセッサによって使用されると思われる部分を一時的に保持するのに使用される、小容量で高速のバッファメモリである。キャッシュメモリの主目的は、データまたは命令の取出のメモリアクセスを実行するのに必要な時間を短縮することである。キャッシュメモリに置かれる情報は、主記憶に置かれた情報よりもはるかに短い時間でアクセスできる。したがって、キャッシュメモリを有するプロセッサは、命令およびオペランドの取出しまたは格納を待つのに費やす時間がはるかに短くなる。
【０００４】
キャッシュメモリは、１つまたは複数のワードのデータからなる複数のキャッシュラインから構成される。各キャッシュラインには、キャッシュラインにコピーされた主記憶のメ

モリラインを一意に識別するアドレスタグが関連付けられる。プロセッサがメモリ参照を行うたびに、アドレスタグとの比較を行って、要求されたデータのコピーがキャッシュメモリにあるかどうかを調べる。所望のメモリラインがキャッシュメモリ内にない場合には、そのメモリラインを主記憶から検索し、キャッシュラインとしてキャッシュメモリに記憶し、プロセッサに供給する。

【０００５】

主記憶からのデータ検索にキャッシュメモリを使用する他に、プロセッサは、データを主記憶に直接書き込む代わりにキャッシュメモリにデータを書き込むこともできる。プロセッサがメモリにデータを書き込もうとする際には、キャッシュメモリがアドレスタグ比較を行って、データを書き込まれるキャッシュラインがキャッシュメモリに存在するかどうかを判定する。そのキャッシュラインがキャッシュメモリに存在し、修正済み（ダーティ）または排他的である場合には、データはキャッシュメモリ内のキャッシュラインに書き込まれる。多くのシステムでは、キャッシュラインのデータ、「ダーティビット」がその後にセットされる。ダーティビットは、キャッシュライン内のデータがダーティ（すなわち、修正済み）であり、したがって、そのキャッシュラインをキャッシュメモリから削除する前に、修正済みのデータを主記憶に書き込まなければならないことを表す。データが書き込まれるキャッシュラインがキャッシュメモリに存在しない場合、キャッシュラインおよびメモリラインを排他的としてキャッシュメモリに取り出すか、データを主記憶に直接書き込まなければならない。

【０００６】

メモリ共用ＭＰシステムは、それぞれが独自のキャッシュを有する、潜在的に多数のプロセッサを有する。このようなシステムでメモリへのアクセスが行われる時には、アクセスされるデータの完全性を確実にするためのステップを実行する必要がある。例えば、ある実体がメモリからデータを読み取る時には、そのデータの更新された版（version）がそのシステム上のプロセッサのキャッシュに存在するかどうかを判定することが重要である。更新された版のデータが存在する場合には、その実体が更新された版のデータにアクセスすることを確実にするための処置を講ずる必要がある。更新された版のデータがメモリ参照で利用されることを確実にする機構を、本明細書ではコヒーレンシ機構と称する。

【０００７】

最も一般的なコヒーレンシ機構はスヌープ（snooping）であり、これは、通常はプロセッサがバスを共用することを必要とする。しかし、電気的な理由から、バスを共用できるプロセッサの個数には限界がある。したがって、ＭＰシステム内のプロセッサ数が多い時には、キャッシュコヒーレンシのためにスヌープを効率的に使用することが不可能になる。

【０００８】

多数のプロセッサを有するシステム用の最も一般的なキャッシュコヒーレンシ機構は、メモリ内のディレクトリ構造である。このディレクトリ構造内では、ライン状態情報が、メモリ内のメモリラインのそれぞれについて存在する。ライン状態情報は、各メモリライン用の複数のビットからなる。各メモリラインのビットは、そのメモリラインに関して、メモリラインの状態（私有、共用など）と、そのメモリライン状態に関連する余分な情報を表す。メモリラインが第１のプロセッサのキャッシュ内で「私有」として保持される時、これは、そのメモリラインが第１のプロセッサによって解放されるまでは他のプロセッサがそのメモリラインを使用できず、第１のプロセッサがそのメモリラインの内容を修正することを許可されていることを意味する。メモリラインが第１のプロセッサのキャッシュ内で「共用」として保持される時、これは、他のプロセッサがそのメモリラインを「私有」として保持しようとしない限り、他のプロセッサがそのメモリラインを使用でき、そのメモリラインが「共用」に保たれている間は、そのメモリラインの内容の修正が許可されないことを意味する。

【０００９】

あるプロセッサがメモリラインのアクセスを所望する時には、そのメモリラインに関する要求がメモリコントローラに送られる。メモリコントローラは、そのメモリラインのライ

ン状態情報を読み取って、要求されたメモリラインの現在の状態を判定する。要求されたメモリラインのライン状態情報ビットから、そのメモリラインが別のキャッシュ内で私有として保持されていることが示される場合、そのメモリラインは、メモリコントローラにリコールされる。メモリラインがメモリコントローラに戻った時に、メモリコントローラは、そのメモリラインを要求元に供給し、メモリラインのライン状態情報を更新し、必要があれば、メモリ内のそのメモリラインのデータを更新する。メモリラインが「私有」として要求され、メモリコントローラがライン状態情報を読み取り、そのメモリラインが「共用」であることがわかった場合には、メモリコントローラは、他のキャッシュ（ライン状態情報によって示される）内のメモリラインのコピーを無効化し、そのメモリラインを要求元に供給する。また、メモリコントローラは、メモリラインのライン情報に「私有」のタグを付け、そのメモリラインを現在所有しているプロセッサを示す。

【００１０】

メモリラインのリコールまたは無効化は、データを返すか無効化するのにかなりの時間を要する可能性がある。その間に、同一のメモリラインに関する次の要求を、メモリコントローラが受け取る可能性がある。これらの新しい要求の再試行は、公平さを実現し長時間の停止状態を防ぐ必要があるので、大規模システムでは厄介である。その代わりに、そのメモリラインに関する新しい要求を、メモリコントローラ内のそのメモリラインに関するリンクリストとして待ち行列化することができる。リコールされたデータまたは無効化の肯定応答を受け取ったならば、メモリコントローラは、要求を受け取った順序でリンクリストのそのメモリラインに関する要求をサービスする。リコールされるメモリラインごとに１つの複数のリンクリストが、どの時点でも、メモリコントローラ内に存在する可能性がある。

【００１１】

一般に、リンクリストは、ランダムアクセスメモリ（ＲＡＭ）構造を使用して実施される。上で説明したキャッシュコヒーレンシ機構の場合、あるメモリラインに関する要求を即座に満足できない時に、そのメモリラインに関する新しい要求に応答して、メモリコントローラは、ディレクトリからそのメモリラインを探索し、メモリラインが使用不能であることを発見した後に、そのメモリラインに関するリンクリストでその要求を待ち行列化する。これには、一般に、新しい要求のための新規項目の作成、リンクリストの末尾の突き止め、および、リンクリストの最終項目の次項目ポインタが新規項目を指すようにする更新が含まれる。その後、この新規項目が、リンクリストの尾部（末尾）として新たに指定される。

【００１２】

メモリラインが使用可能になった時に、メモリコントローラは、最初の（頭部）項目についてリンクリストにアクセスし、適当な処置を行う。リンクリスト内のポインタは、リンクリストの最初の項目の除去を反映するように更新される。

【００１３】

【発明が解決しようとする課題】

上の説明から明らかなとおり、すべての要求が、メモリコントローラによる１回または複数回のメモリラインに関するディレクトリへのアクセスをもたらす。リンクリスト項目が作成される可能性もある。その要求に代わってラインのリコールまたは無効化が発行される場合、メモリラインが返されるか無効化される時に、メモリコントローラは、要求を完了するためにディレクトリに関連するリンクリストをもう一度探索しなければならない。その結果、次項目の探索は、効率的でなければならない。そうでないと、多くの応用分野でかなりの性能損失がもたらされる可能性がある。

【００１４】

本発明は、上記の問題点に鑑みなされたもので、直接探索ではなく連想探索によって、リンクリストの現在の項目から次の項目を効率的に探索することを目的としている。

【００１５】

【課題を解決するための手段】

本発明の好ましい実施例によれば、計算システム内のリンクリスト構造に、最初の項目と追加の項目が含まれる。追加の項目のそれぞれには、リンクリスト内の前の項目へのリンク参照が含まれる。追加項目のそれぞれのリンク参照は、連想記憶内に格納される。例えば、前の項目へのリンク参照は、連想記憶内での前の項目のインデックスである。追加項目のそれぞれは、次の追加項目内の前項目へのリンク参照を使用する内容探索を実行することによってアクセスできる。すなわち、前項目へのリンク参照が、連想記憶内の前項目のインデックスを含むリンクフィールドである時に、次の追加項目は、次の追加項目の直前のリンクリスト内の項目のインデックスを求める連想記憶内での連想探索を実行することによって、発見される。

【００１６】

したがって、リンクリストは、例えばリンクリストの最初の項目をアクセスすることによって、横断される。リンクリストの第２の項目は、第１の項目への参照を連想記憶から探索する（例えば連想記憶での第１の項目のインデックスを使用して）ことによってアクセスされる。リンクリストの第３の項目は、第２の項目への参照を連想記憶から探索する（例えば連想記憶での第２の項目のインデックスを使用して）ことによってアクセスされる。以下同様である。

【００１７】

本発明のさまざまな実施例では、項目ごとに、その項目が有効であるかどうかを示す有効性ビットも連想記憶に格納される。

【００１８】

本発明のさまざまな実施例は、特定の応用分野に合わせて調整することができる。例えば、１実施例では、主記憶内のメモリラインのアクセスに使用されるメモリコントローラ内の要求待ち行列内でリンクリストが使用される。単一待ち行列の実施例では、第１の項目および追加項目のそれぞれについて、連想記憶内に、頭部フィールド、尾部フィールド、アドレスフィールドおよび有効性ビットが格納される。頭部フィールドには、第１項目では真、追加項目のそれぞれでは偽の値が格納される。尾部フィールドには、リンクリストの最終項目の場合に限って真になる値が格納される。アドレスフィールドには、主記憶内のメモリラインのアドレスが格納される。有効性ビットは、その項目が有効（使用中）であるかどうかを表す。各項目の追加情報は、対応するＣＡＭ項目または通常のインデックスデコーダからの「一致する」ビットを使ってアドレッシングされるランダムアクセスメモリに格納される。例えば、追加の情報には、メモリラインに対して実行される動作が含まれる。追加情報には、さらに、有効な時にそのメモリラインの現在のデータが格納されるデータフィールドを含めることができる。

【００１９】

本発明のもう１つの実施例では、主記憶のメモリラインのアクセスに使用されるメモリコントローラの要求待ち行列の２待ち行列実施態様内でリンクリストが使用される。２待ち行列実施態様では、例えば、各リンクリストの最初の項目のアドレスフィールドと有効性ビットが、別の（頭部待ち行列）連想記憶に格納される。頭部待ち行列連想記憶には、メモリラインに関する結果的なリンクリストの頭部項目だけが含まれる。アドレスフィールドには、メモリラインのアドレスが格納される。有効性ビットは、その項目が有効であるかどうかを示す。項目の追加情報は、ランダムアクセスメモリに格納される。

【００２０】

第１の項目の追加情報は、ランダムアクセスメモリに格納される。この追加情報には、例えば、メモリラインに対して実行される動作、最終フィールドおよび尾部フィールドが含まれる。最終フィールドには、第１の項目がそのリンクリストの最終項目である時に真になる値が格納される。尾部フィールドには、有効な時に、リンクリスト内の最終項目のインデックスが格納される。追加情報には、さらに、有効な時にメモリラインの現在のデータが格納されるデータフィールドを含めることができる。

【００２１】

第２の連想記憶には追加項目ごとに、ヘッド／リンクフィールドが格納される。ヘッド／

リンクフィールドには、リンク参照によって参照される前項目が、連想記憶と第２の連想記憶のどちらに存在するかを示す値が格納される。第２の連想記憶の項目に、有効フィールドとリンクフィールドを含めることができる。

【００２２】

本発明を用いると、特定の応用分野でリンクリストへのアクセスを大幅に単純化できる、リンクリストを実施するための効率的な方法が可能になる。

【００２３】

【発明の実施の形態】

図１は、計算システム内で相互接続されたさまざまな実体を示す図である。例えば、バス１１には、プロセッサ２４、２３およびメモリコントローラ２１が接続されている。プロセッサ２４には、キャッシュ２７が含まれる。プロセッサ２３には、キャッシュ２６が含まれる。メモリコントローラ２１には、据置き読取り待ち行列（ＤＲＱ）２５が含まれる。メモリコントローラ２１は、メモリ２２へのアクセスを制御する。他の実体も、バス１１に接続できる。

【００２４】

バス１２には、プロセッサ３１、３２が接続されている。プロセッサ３１には、キャッシュ３３が含まれる。プロセッサ３２には、キャッシュ３４が含まれる。他の実体も、バス１２に接続できる。バス１３には、プロセッサ４１およびプロセッサ４２が接続されている。プロセッサ４１には、キャッシュ４３が含まれる。プロセッサ４２には、キャッシュ４４が含まれる。他の実体も、バス１３に接続できる。バス１４には、プロセッサ４７およびメモリコントローラ４６が接続されている。プロセッサ４７には、キャッシュ４８が含まれる。メモリコントローラ４６は、メモリ４５へのアクセスを制御する。メモリコントローラ４６には、据置き読取り待ち行列（ＤＲＱ）４９が含まれる。他の実体も、バス１４に接続できる。

【００２５】

バス１１、１２、１３および１４は、相互接続１０によって示されるように、なんらかの形で相互接続される。さらに、相互接続１０によって他のバスを相互接続することができる。この意味では、図１は、計算システム内で相互接続される実体の組合せの例にすぎない。

【００２６】

そのようなシステムでメモリ２２へのアクセスが行われる時には、メモリコントローラ２１が、アクセスされるデータの保全性を確実にする。例えば、ある実体がメモリ２２からデータを読み取る時には、メモリコントローラ２１は、この計算システム内のプロセッサのいずれかのキャッシュにそのデータの更新された版が存在するかどうかを判定する。更新された版のデータが存在する場合、メモリコントローラ２１は、キャッシュコヒーレンシ機構を使用して、データの保全性を確実にする。

【００２７】

メモリ２２内のディレクトリ構造内には、メモリ２２内のメモリラインごとにライン状態情報が存在する。ライン状態情報は、メモリラインごとの複数のビットからなる。メモリラインごとのビット群は、例えば、そのメモリラインについて、メモリラインの状態（私有、共用など）と、そのメモリラインの状態に関連する余分な情報とを表す。メモリラインが第１のプロセッサのキャッシュ内で「私有」として保持される時、これは、第１のプロセッサがメモリラインを解放するまでは他のプロセッサがそのメモリラインを使用できず、第１のプロセッサがそのメモリラインの内容の変更を許可されていることを意味する。メモリラインが第１のプロセッサのキャッシュ内で「共用」として保持される時、これは、他のプロセッサがそのメモリラインを「私有」として保持しようとしない限り、他のプロセッサがそのメモリラインを使用でき、そのメモリラインが「共用」に保たれている間は、そのメモリラインの内容の修正が許可されないことを意味する。

【００２８】

あるプロセッサが、メモリ２２のメモリラインへのアクセスを所望する時には、そのメモ

リラインに関する要求がメモリコントローラ２１に送られる。メモリコントローラ２１は、そのメモリラインのライン状態情報を読み取って、要求されたメモリラインの現在の状態を判定する。要求されたメモリラインのライン状態情報ビットから、そのメモリラインが別のキャッシュで「私有」として保持されていることが示される場合、メモリコントローラ２１は、そのメモリラインのリコールを発行する。メモリラインがメモリコントローラ２１に返される時に、メモリコントローラ２１は、要求元にメモリラインを供給し、メモリラインのライン状態情報を更新し、必要であれば、メモリ２２内のメモリラインのデータを更新する。

【００２９】

メモリコントローラ２１がメモリラインを無効化するかメモリラインのリコールを要求するかし、メモリコントローラ２１が、リコールまたは無効化に対する応答を待っている間にそのメモリラインに関する追加の要求を受け取った時には、メモリコントローラは、図２に示された据置き読取り待ち行列（ＤＲＱ）に、そのメモリラインに関するリンクリストとして追加の要求を待ち行列化する。ＤＲＱ内のすべてのＤＲＱ項目に、リコール、無効化またはメモリからの読取りの処理中であるメモリラインに関する要求ならびに、同一のメモリラインに関する前の要求の完了を待っている、リンクリストに待ち行列化された要求が含まれる。

【００３０】

図２に示された据置き読取り待ち行列には、連想記憶（ＣＡＭ）５０と、ランダムアクセスメモリ（ＲＡＭ）６０とが含まれる。図２に示されたＤＲＱの実施態様では、頭部および尾部の両方の情報とリンクリスト実施態様のための単一の完全連想構造が使用される。

【００３１】

ＤＲＱ内の各ＤＲＱ項目は、さまざまなフィールドに分割される。ＣＡＭ５０内では、アドレスフィールド５３を使用して、ＤＲＱ項目に関連するメモリ２２内のメモリラインを示すアドレスを格納する。頭部（Ｈ）フィールド５１には、アドレスフィールド５３に格納されたアドレスのリンクリストの第１の項目がＤＲＱ項目に含まれるかどうかを示すビットが格納される。尾部（Ｔ）フィールド５２には、アドレスフィールド５３に格納されたアドレスのリンクリストの最後の項目がＤＲＱ項目に含まれるかどうかを示すビットが格納される。有効（Ｖ）フィールド５４には、ＤＲＱ項目に有効な要求が含まれるかどうかを示すビットが含まれる。リンクフィールド５５には、そのメモリラインのリンクリストに前の項目がある場合に、その項目のＤＲＱインデックスが格納される。リンクフィールドのビット数は、ＤＲＱのサイズに応じて変化する。図２には、９つのＤＲＱ項目（インデックス００１ないしインデックス００９）だけが図示されている。ＤＲＱ項目の数は、計算システムの構成、メモリ２２のサイズおよび他の多数の異なる要因に応じて大きく変化する可能性がある。

【００３２】

ＲＡＭ６０内では、動作（ＯＰ）フィールド６１が、ＤＲＱ項目に格納された要求の符号を示す。他フィールド６２には、ＤＲＱ項目に格納された要求を完了するのに必要な他の情報が格納される。任意選択のデータフィールド６３は、ＤＲＱ項目に格納された要求のためのメモリ読取りによって返されるデータを格納するのに使用できる。メモリラインのデータをキャッシュ記憶することによって、ラインのリコールでデータが返されない場合の待ち時間が大幅に改善される可能性がある。というのは、そのメモリラインが、前のオーナー（owner）によって修正されていないからである。

【００３３】

ＣＡＭ５０を探索する時に、探索されるフィールドは、ＣＡＭ５０の比較器機構を使用する時にＣＡＭ５０の他のフィールドを強制的に「無視（don't care）」にすることによって制限できる。例えば、第１のメモリラインに関する要求がＤＲＱ内にある場合、探索は、尾部フィールド５２、アドレスフィールド５３および有効フィールド５４を使用して実行できる。尾部フィールド５２を探索するには、値「論理１」と、ＣＡＭ５０内の各ＤＲＱ項目の尾部フィールド５２の内容との比較を行う。アドレスフィールド５３を探索する

には、第１のメモリラインのアドレスと、ＣＡＭ５０内の各ＤＲＱ項目のアドレスフィールド５３の内容との比較を行う。有効フィールド５４を探索するには、値「論理１」と、ＣＡＭ５０内の各ＤＲＱ項目の有効フィールド５４の内容との比較を行う。尾部フィールド５２、アドレスフィールド５３および有効フィールド５４を使用するこのような探索は、新しい要求を受け取り、その要求をＤＲＱに待ち行列化しなければならない時に行われる。一致が見つからない場合、第１のメモリライン用の新規のリンクリストを開始する。新規リンクリストの第１の項目には、その要求を説明する情報が格納される。新しい要求を受け取り、その要求をＤＲＱに待ち行列化しなければならない時に尾部フィールド５２、アドレスフィールド５３および有効フィールド５４を使用する探索を実行した後に、一致が見つかった場合には、その要求のための新規のＤＲＱ項目を、既存のリンクリストの末尾にリンクする。

【００３４】

図３ないし図９は、表１に示された要求を格納し、利用するための、図２に示されたＤＲＱの使用を示す図である。メモリコントローラ２１が要求１を受け取る時に、メモリコントローラ２１は、上で述べたように、尾部フィールド５２、アドレスフィールド５３および有効フィールド５４を使用して、ＣＡＭ５０からメモリラインＡの有効な項目を探索する。一致するＤＲＱがない場合、メモリコントローラ２１は、メモリ２２内のディレクトリを読み取って、例えば、計算システムのプロセッサのうちの１つのキャッシュによってメモリラインＡが「私有」として保持されていることを発見する。その結果、メモリコントローラは、オーナーキャッシュにメモリラインＡのリコールを発行する。メモリコントローラ２１は、ＤＲＱ内の空きＤＲＱ項目も見つけ、図３に示されるように、その空きＤＲＱ項目に要求１の情報を置く。本発明のいくつかの実施例では、メモリコントローラ２１が、メモリにアクセスする間にＤＲＱ内で要求を待ち行列化することもできる。

【００３５】

【表１】

要求１	私有	メモリラインＡ
要求２	共有	メモリラインＡ
要求３	共有	メモリラインＡ
要求４	共有	メモリラインＡ

【００３６】

図３では、要求１のＤＲＱが、インデックス００１のＤＲＱ項目にある。頭部フィールド５１と尾部フィールド５２にある値「１」は、この項目が、メモリラインＡのリンクリストの先頭であり、末尾でもあることを示す。頭部フィールド５１に「１」の値が含まれる時には、リンクフィールド５５の内容は通常は無視される。Ｖビットには、「１」（真）がセットされ、これが有効なＤＲＱ項目であることを示す。動作フィールド６１と他フィールド６２には、要求１に関連する情報が含まれる。要求１のＤＲＱ項目を書き込んだ後に、ＤＲＱの頭部ポインタを増分して、次の空きＤＲＱ項目を指すようにする。この実施例では、ＤＲＱの頭部ポインタを増分して次の空きＤＲＱ項目を指すようにするが、本発明の代替実施例では、異なる機構を使用して空きＤＲＱ項目を管理できる。

【００３７】

図示の例では、メモリラインＡのリコールを待っている間に、メモリコントローラ２１が、メモリラインＡに関する要求２を受け取る。メモリコントローラ２１は、アドレスフィールド５３、有効フィールド５４および尾部フィールド５２を使用して、メモリラインＡの有効な項目のうちでリンクリストの末尾にあるものをＣＡＭ５０から探索する。メモリコントローラ２１は、インデックス００１のＤＲＱ項目を見つける。メモリコントローラ

２１は、インデックス００１のＤＲＱ項目を変更し、その結果、インデックス００１のＤＲＱ項目の尾部フィールド５２が「０」になるようにする。１ビットだけが使用されるので、これは、特別なＣＡＭセルを使用するＣＡＭ照合の副作用として行うことができ、したがって、これのために別のＣＡＭアクセスを行う必要はない。メモリコントローラ２１は、ＤＲＱ内の空きＤＲＱ項目も見つけ、図４に示されるように、その空きＤＲＱ項目に要求２の情報を置く。図４では、インデックス００２のＤＲＱ項目が、要求１の後、要求２の前に到着した別のメモリラインに関する別の要求の格納に使用されたと仮定している。

【００３８】

図４では、要求２のＤＲＱが、インデックス００３のＤＲＱ項目にある。尾部フィールド５２の値「１」は、この項目がメモリラインＡのリンクリストの末尾であることを示す。インデックス００３のＤＲＱ項目のリンクフィールド５５には、メモリコントローラ２１によって、メモリラインＡの前の項目（リンクリストの以前の末尾）のＤＲＱインデックスであるインデックス００１が置かれる。Ｖビットには「１」（真）がセットされて、有効なＤＲＱ項目であることが示される。動作フィールド６１と他フィールド６２には、要求２に関連する情報が格納される。要求２のＤＲＱ項目を書き込んだ後に、ＤＲＱの頭部ポインタを増分して、次の空きＤＲＱ項目を指すようにする。さらに、他の構造を使用して、次の空きＤＲＱ項目を判定することができる。

【００３９】

まだメモリラインＡのリコールを待っている間に、メモリコントローラ２１は、要求３を受け取る。メモリコントローラ２１は、アドレスフィールド５３、有効フィールド５４および尾部フィールド５２を使用して、メモリラインＡの有効な項目のうちでリンクリストの末尾にあるものをＣＡＭ５０から探索する。メモリコントローラ２１は、インデックス００３のＤＲＱ項目を見つける。メモリコントローラ２１は、インデックス００３のＤＲＱ項目の項目を変更し、その結果、インデックス００３のＤＲＱ項目の尾部フィールド５２が「０」になるようにする。メモリコントローラ２１は、ＤＲＱ内の空きＤＲＱ項目も見つけ、図５に示されるように、その空きＤＲＱ項目に要求３の情報を置く。図５では、インデックス００４、００５および００６のＤＲＱ項目が、要求２の後、要求３の前に到着した他のメモリラインに関する他の要求の格納に使用されたと仮定している。

【００４０】

図５では、要求３のＤＲＱが、インデックス００７のＤＲＱ項目にある。尾部フィールド５２の値「１」は、この項目がメモリラインＡのリンクリストの末尾であることを示す。インデックス００７のＤＲＱ項目のリンクフィールド５５には、メモリコントローラ２１によって、メモリラインＡの前項目のＤＲＱインデックスであるインデックス００３が置かれる。Ｖビットには「１」（真）がセットされて、有効なＤＲＱ項目であることが示される。動作フィールド６１と他フィールド６２には、要求３に関連する情報が格納される。要求３のＤＲＱ項目を書き込んだ後に、ＤＲＱの頭部ポインタを増分して、次の空きＤＲＱ項目を指すようにする。

【００４１】

やはり、まだメモリラインＡのリコールを待っている間に、メモリコントローラ２１は、要求４を受け取る。メモリコントローラ２１は、アドレスフィールド５３、有効フィールド５４および尾部フィールド５２を使用して、メモリラインＡの有効な項目のうちでリンクリストの末尾にあるものをＣＡＭ５０から探索する。メモリコントローラ２１は、インデックス００７のＤＲＱ項目を見つける。メモリコントローラ２１は、インデックス００７のＤＲＱ項目の項目を変更し、その結果、インデックス００７のＤＲＱ項目の尾部フィールド５２が「０」になるようにする。メモリコントローラ２１は、ＤＲＱ内の空きＤＲＱ項目も見つけ、図６に示されるように、その空きＤＲＱ項目に要求４の情報を置く。図６では、インデックス００８のＤＲＱ項目が、要求３の後、要求４の前に到着した別のメモリラインに関する別の要求の格納に使用されたと仮定している。

【００４２】

図６では、要求４のＤＲＱが、インデックス００９のＤＲＱ項目にある。尾部フィールド５２の値「１」は、この項目がメモリラインＡのリンクリストの末尾であることを示す。インデックス００９のＤＲＱ項目のリンクフィールド５５には、メモリコントローラ２１によって、メモリラインＡの前項目のＤＲＱインデックスであるインデックス００７が置かれる。Ｖビットには「１」（真）がセットされて、有効なＤＲＱ項目であることが示される。動作フィールド６１と他フィールド６２には、要求４に関連する情報が格納される。要求４のＤＲＱ項目を書き込んだ後に、ＤＲＱの頭部ポインタを増分して、次の空きＤＲＱ項目を指すようにする。

【００４３】

メモリコントローラが、メモリラインＡの更新されたデータも含めて、メモリラインＡのオーナーからの応答を受け取る時には、メモリコントローラ２１は、アドレスＡを設定されたアドレスフィールド５３、１（真）の値の頭部フィールド５１および１（真）の値の有効フィールド５４を使用して、ＣＡＭ５０を探索する。この探索での他のフィールドは、「無視（don't care）」である。この場合では、インデックス００１のＤＲＱ項目が一致し、その内容が読み取られる。

【００４４】

インデックス００１のＤＲＱ項目を見つけたメモリコントローラ２１は、動作フィールド６１と他フィールド６２の情報を使用して、要求１を実行する。また、メモリコントローラ２１は、要求１が実行されたことを示すようにＤＲＱを更新する。

【００４５】

具体的にいうと、図７からわかるように、メモリコントローラ２１は、インデックス００１のＤＲＱ項目を無効化し、インデックス００１のＤＲＱ項目を再利用の対象にする。インデックス００１のＤＲＱ項目の場合、尾部フィールド５２が「０」に等しい（すなわち、真ではなく、メモリラインＡに関して待ち行列化された要求が他にも存在することを示す）ので、ＤＲＱでは、この項目のインデックス００１を使用して、リンクリストの次のリンクを探索する。メモリコントローラ２１は、リンクフィールド５５（００１がセットされている）と有効フィールド５４（１がセットされている）を使用して、リンクリストの次の項目をＣＡＭ５０から検索する。メモリコントローラ２１は、インデックス００３のＤＲＱ項目を見つける。メモリコントローラ２１は、動作フィールド６１と他フィールド６２を読み取って、要求の種類を判定する。この例では、メモリコントローラ２１は、要求２が「共用」読取りであることを発見する。メモリラインＡは、要求１の結果として私有として与えられるので、要求２は、メモリラインＡのリコールが終わるまでは実行できない。したがって、メモリコントローラは、メモリラインＡの新しいオーナーに、メモリラインＡの新規のリコールを発行する。また、メモリコントローラ２１は、インデックス００３のＤＲＱ項目の項目を変更し、その結果、インデックス００３のＤＲＱ項目の頭部フィールド５１が「１」になり、アドレスフィールド５３がアドレスＡになるようにする。インデックス００３のＤＲＱ項目のリンクフィールド５５は、現在はインデックス００３のＤＲＱ項目がメモリラインＡのリンクリストの先頭なので、「無視（don't care）」になる。この結果を図７に示す。

【００４６】

性能を最適化するために、メモリラインＡの更新されたデータを、インデックス００３のＤＲＱ項目のデータフィールド６３に格納することができる。その代わりに、メモリラインＡの更新されたデータを、メモリコントローラ２１内の局所バッファに取り込むか、メモリ２２に戻すか、その両方を行うことができる。ＤＲＱ内にデータをキャッシュ記憶することの唯一の目的は、メモリラインＡの新しいオーナーが、リコールの前にメモリラインＡをキャストアウトするか、そのキャッシュ内にある間にメモリラインＡを修正しない場合に、もう一度メモリ２２にアクセスしないようにすることである。

【００４７】

メモリコントローラが、メモリラインＡの更新されたデータも含めて、メモリラインＡの新しいオーナーからの応答を受け取る時には、メモリコントローラ２１は、アドレスＡが

セットされたアドレスフィールド５３、１（真）の値の頭部フィールド５１および１（真）の値の有効フィールド５４を使って、もう一度ＣＡＭ５０を探索する。この探索での他のフィールドは、「無視（don't care）」である。この場合では、インデックス００３のＤＲＱ項目が一致し、その内容が読み取られる。

【００４８】

インデックス００３のＤＲＱ項目を見つけたメモリコントローラ２１は、動作フィールド６１と他フィールド６２の情報を使用して、要求２を実行する。また、メモリコントローラ２１は、要求２が実行されたことを示すようにＤＲＱを更新する。

【００４９】

具体的にいうと、図８からわかるように、メモリコントローラ２１は、インデックス００３のＤＲＱ項目を無効化し、インデックス００３のＤＲＱ項目を再利用の対象にする。インデックス００３のＤＲＱ項目の場合、尾部フィールド５２が「０」に等しい（すなわち、真ではない）ので、ＤＲＱでは、リンクリストの次のリンクを探索する。メモリコントローラ２１は、リンクフィールド５５（００３がセットされている）と有効フィールド５４（１がセットされている）を使用して、リンクリストの次の項目をＣＡＭ５０から検索する。メモリコントローラ２１は、インデックス００７のＤＲＱ項目を見つける。要求３の実行には、メモリラインＡのリコールは不要なので、メモリコントローラ２１がインデックス００７のＤＲＱ項目の頭部フィールド５１に「１」（真）をセットする必要はない。その代わりに、インデックス００７のＤＲＱ項目の頭部フィールド５１は、無変更のままにされる。この結果を、図８に示す。

【００５０】

メモリコントローラ２１は、動作フィールド６１と他フィールド６２を読み取って、要求の種類を判定する。この例では、メモリコントローラ２１は、要求３が、メモリラインＡのリコールなしで実行できる「共用」読取りであることを発見する。したがって、メモリコントローラは、動作フィールド６１と他フィールド６２の情報を使用して、要求３を実行する。要求３は、直ちに実行できるので、メモリコントローラ２１は、ＤＲＱを更新して、要求３が実行されたことを示す。

【００５１】

具体的にいうと、図９からわかるように、メモリコントローラ２１は、インデックス００７のＤＲＱ項目を無効化し、インデックス００７のＤＲＱ項目を再利用の対象にする。インデックス００７のＤＲＱ項目の場合、尾部フィールド５２が「０」に等しい（すなわち、真ではない）ので、ＤＲＱでは、リンクリストの次のリンクを探索する。メモリコントローラ２１は、リンクフィールド５５（００７がセットされている）と有効フィールド５４（１がセットされている）を使用して、リンクリストの次の項目をＣＡＭ５０から検索する。メモリコントローラ２１は、インデックス００９のＤＲＱ項目を見つける。

【００５２】

メモリコントローラ２１は、動作フィールド６１と他フィールド６２を読み取って、要求の種類を判定する。この例では、メモリコントローラ２１は、要求４が、メモリラインＡのリコールなしで実行できる「共用」読取りであることを発見する。その結果、メモリコントローラ２１は、インデックス００９のＤＲＱ項目の頭部フィールド５１を無変更のままにすることができる。この結果を、図９に示す。

【００５３】

メモリコントローラ２１は、動作フィールド６１と他フィールド６２の情報を使用して、要求４を実行する。また、メモリコントローラ２１は、ＤＲＱを更新して、要求４が実行されたことを示す。

【００５４】

具体的にいうと、図１０からわかるように、メモリコントローラ２１は、インデックス００９のＤＲＱ項目を無効化し、インデックス００９のＤＲＱ項目を再利用の対象にする。インデックス００９のＤＲＱ項目の場合、尾部フィールド５２が「１」に等しい（すなわち、真である）ので、ＤＲＱでは、リンクリストの次のリンクを探索しない。メモリコン

トローラ２１は、メモリ２２内のメモリラインＡのライン状態情報に、「共用」のマークを付ける。

【００５５】

図１１に、２つの連想待ち行列を使用してＤＲＱが実施される、本発明の代替実施例を示す。この実施例は、性能上の理由から、リンクリスト実施態様の連想待ち行列の１つに複数のポートが必要な時に好ましい。この場合、ポートの数は、通常は、２つの連想待ち行列を使用することによって、リストのリンク部分からリストの頭部項目を分離することによって減らすことができる。

【００５６】

図１１に示された据置き読取り待ち行列（ＤＲＱ）は、２つの連想待ち行列を有する。第１の待ち行列には、連想記憶（ＣＡＭ）８０とランダムアクセスメモリ（ＲＡＭ）９０が含まれる。第２の待ち行列には、ＣＡＭ１００とＲＡＭ１１０が含まれる。これら２つの待ち行列のＤＲＱ項目のすべてに、リコールまたは無効化の処理中であるか、メモリからの読取りの処理中であるメモリラインに関する要求ならびに、同一のラインに関する要求の背後でリンクリストに待ち行列化された要求が格納される。ＣＡＭ８０とＲＡＭ９０は、各リンクリストの頭部が格納される頭部待ち行列（ＨＱ）を形成する。ＣＡＭ１００とＲＡＭ１１０は、各リンクリストの追加項目が格納されるリンク待ち行列（ＬＱ）を形成する。

【００５７】

ＣＡＭ８０内では、アドレスフィールド８２が、ＤＲＱ項目に関連するメモリ２２内のメモリラインを表すアドレスの格納に使用される。有効（Ｖ）フィールド８１は、ＤＲＱ項目に有効な要求が含まれるかどうかを示すビットからなる。

【００５８】

ＲＡＭ９０内では、動作（ＯＰ）フィールド９１が、ＤＲＱ項目に格納された要求の符号を示す。他フィールド９２には、ＤＲＱ項目に格納された要求を完了するのに必要な、他の情報が格納される。最終（Ｌ）フィールド９３（前のＤＲＱ実施例の尾部フィールド５２と同等）には、アドレスフィールド８２に格納されたアドレスのリンクリストの最終項目がそのＤＲＱ項目に含まれるかどうかを示すビットが格納される。尾部フィールド９４は、その項目のリンクリストの最終項目のインデックスを示す。

【００５９】

任意選択のデータフィールド９５は、ＤＲＱ項目に格納された要求のメモリラインのデータを格納するのに使用できる。データフィールド９５は、実施されるとしても、ＬＱのためには必要ない。というのは、ＨＱが、メモリからのデータを含む、あるアドレスのリンクリストの最初の項目が格納される待ち行列だからである。リンクの横断に割り込む必要がある場合（例えば、メモリラインが「私有」としてプロセッサに与えられ、リコールの必要があるので）、そのデータを待っている要求は、局所データレジスタ（すなわち、リンクリストの横断を高速化するために、完了直後または完了直前の前の頭部項目からデータを取得するレジスタ）からのデータを含めて、ＨＱにコピーされる。

【００６０】

ＣＡＭ１００内では、有効（Ｖ）フィールド１０１は、ＤＲＱ項目に有効な要求が含まれるかどうかを示すビットからなる。リンクフィールド１０３には、メモリラインの前項目のＤＲＱインデックスが格納される。リンクフィールドのビット数は、ＤＲＱのサイズ（ＨＱとＬＱの大きいほうのサイズ）に応じて変化する。頭部／リンク（Ｈ／Ｌ）フィールド１０２は、論理的にはリンクフィールド１０３の一部である。偽（論理０）のＨ／Ｌフィールド１０２は、リンクフィールド１０３がＨＱインデックスを表すことを示す。真のＨ／Ｌフィールド１０２は、リンクフィールド１０３がＬＱインデックスを表すことを示す。

【００６１】

ＲＡＭ１１０内では、動作（ＯＰ）フィールド１１１が、ＤＲＱ項目に格納された要求の符号を示す。他フィールド１１２には、ＤＲＱ項目に格納された要求を完了するのに必要

な、他の情報が格納される。

【００６２】

ＤＲＱ内の項目が、あるメモリラインのために作成され、ＨＱ内にそのメモリラインのアドレスに一致する項目がない時には、ＨＱ内で、ＨＱの次の空き項目の位置に、そのアドレスのための項目が作成される。この項目は、Ｖフィールド８１のＶビットを立てられ、アドレスフィールドは、待ち行列化されるメモリラインのアドレスになる。ＲＡＭ９０は、データが使用可能になった時に要求を完了するのに必要な情報を、動作フィールド９１と他フィールド９２に有する。さらに、Ｌフィールド９３の最終ビットを立てて、ＬＱにあふれるリンクリストがないことを示す。尾部フィールド９４の値は、他の項目へのリンクがないので「無視（Don't Care）」である。

【００６３】

同一のメモリラインに関するもう１つの要求を受け取る際に、メモリコントローラ２１は、メモリラインのアドレスを使ってＨＱを探索し、前にＨＱに置かれた一致するアドレスを見つける。その結果、メモリコントローラ２１は、新しい要求をＬＱに待ち行列化し、現在の「尾部」項目にリンクする。これは、次の空きＬＱ項目を使用することによって行われる。リンクフィールド１０３には、前の「尾部」項目のインデックスが置かれる。Ｖフィールド１０１の値のビットを立てる。一致するＨＱ項目のＬビットが立っているので、Ｈ／Ｌフィールド１０２の値は、立てないものとして書き込まれ（リンクフィールド１０３のリンクがＨＱインデックスであることを示す）、一致するＨＱ項目のインデックスが、リンクフィールド１０３に書き込まれる値になる。

【００６４】

メモリラインのＨＱ項目も、更新する必要がある。ＲＡＭ９０では、Ｌフィールド９３のビットを寝かせて、メモリラインの項目がＬＱにあふれていることを示す。さらに、尾部フィールド９４の値を更新して、メモリラインのリンクリストの最後のＬＱ項目のインデックスを含める。この場合、尾部フィールド９４の値は、次の空きＬＱ項目であった、書き込まれたばかりのＬＱ項目のインデックスである。

【００６５】

また、ＲＡＭ１１０は、データが使用可能になった時に要求を完了するのに必要なすべての情報を用いて更新される。同一のラインに関するもう１つの要求を受け取り、一致するＨＱ項目のＬフィールド９３のビットが立っていない場合には、この項目は、現在ＬＱの尾部インデックスにある項目の後ろの最終項目として、リンクリストで待ち行列化しなければならない。これは、この実施例では、次の空きＬＱ項目で、リンクフィールドに尾部インデックスを書き込み、Ｖを立て、Ｈ／Ｌを立てる（そのリンクがＬＱインデックスであることを意味する）ことによって行われる。一致するＨＱ項目のＲＡＭも、更新の必要があり、Ｌフィールド９３のビットは寝かせたままで、尾部フィールドにそのアドレスのリンクリストの最後のＬＱ項目のインデックス、この例では書き込んだばかりのＬＱ項目のインデックスを書き込む。そのＬＱ項目のＲＡＭ部分では、データが使用可能になった時に要求を完了するのに必要なすべての情報を用いて、動作フィールド９１と他フィールド９２が更新される。

【００６６】

ＨＱ連想待ち行列とＬＱ連想待ち行列を使用するＤＲＱの動作は、上で説明した１つの連想待ち行列だけを用いるＤＲＱの動作にかなり類似している。重要な相違は、Ｈ／Ｌビットの追加と、１つではなく２つの待ち行列を操作するという事実だけである。また、リストの最終項目を検出するのに使用される機構も、ＲＡＭ９０に配置された尾部フィールド９４とＬフィールド９３を使用するので、多少異なる。しかし、ＤＲＱの単一連想待ち行列と二重連想待ち行列の実施例の両方で、現在のインデックスをリンクとして使用する連想検索を介してリンクリストを横断するという提案の機構が使用される。

【００６７】

例えばリコール応答を受け取った時など、ある項目を待ち行列から解除するには、メモリラインの返されたアドレスを使用して、ＨＱのＣＡＭ８０を検索する。返されたアドレス

に一致するアドレスをアドレスフィールド８２に有するＤＲＱ項目がアクセスされる。動作フィールド９１と他フィールド９２のその項目の情報を使用して、要求を完了する。メモリラインのこの項目は、無効化される。

【００６８】

一致するＨＱ項目のＬフィールド９３のビットが立っている場合、これは、そのメモリラインのリンクリストに他の項目がないことを示す。しかし、Ｌフィールド９３のビットが寝ている場合、これは、ＬＱ内にそのメモリラインの追加項目があることを示す。そのメモリラインのリンクリストの次の項目は、そのメモリラインの一致したＨＱインデックスを用い、Ｖフィールド１０１に対応するビットを立て、Ｈ／Ｌフィールド１０２に対応するビットを寝かせてＬＱのＣＡＭ１００を探索することによってアクセスされる。Ｈ／Ｌフィールド１０２のビットは、リンクフィールド１０３の探索に使用されるインデックスがＨＱインデックスであることを示すために寝かされる。

【００６９】

この例の次のステップは、ＨＱ内のメモリラインに関する完了したばかりの要求の性質に応じて変化する。前の要求で、メモリラインが「共用」のままにされ、次の要求で、「私有」としてのメモリラインを必要としない場合、次の要求は、リンク横断の高速化専用の局所レジスタを使用して、即座に実行できる。この時点で、尾部フィールド９４の値と、一致するＬＱ項目のインデックスが等しい場合、これがそのメモリラインに関するリンクリスト横断の終りである。尾部フィールド９４の値が、一致するＬＱ項目のインデックスと等しくない場合、リンクリスト内のメモリラインの次の項目は、一致するＬＱインデックスを用い、Ｖフィールド１０１のビットを立て、Ｈ／Ｌフィールド１０２のビットを立て（ＬＱインデックスであることを示す）、ＬＱのＣＡＭ１００を探索することによって見つかる。

【００７０】

メモリラインが、ＬＱ内の最初の要求の結果としてまだ「共用」である場合、新しい要求は、前と同様に局所レジスタから実行できる。しかし、メモリラインが「私有」になる場合、そのメモリラインがリコールされ、ＬＱから取り出したメモリラインの最後の項目が、新しい応答を待っているリンクリストの先頭として、ＨＱ（次の空きＨＱ位置）に書き込まれる。これがそのアドレスのリンクリストの最後の項目であった（インデックスが、対応するＨＱ項目の尾部フィールド９４の値と等しかった）場合、そのアドレスの新しいＨＱ項目のＬフィールド９３のビットを立て、尾部フィールド９４の値を「無視（don't care）」にする。そのメモリラインのリンクリストの最終項目ではない場合には、Ｌフィールド９３のビットを寝かし、尾部フィールド９４の値は、そのメモリラインの前のＨＱ項目と同一のままにする。ここでは、そのメモリラインに関する新しい要求を受け取っていないと仮定している。ＬＱに格納されたそのメモリラインの次の要求は、そのメモリラインの最後の一致するＬＱインデックスを用い、Ｖフィールド１０１のビットを立て、Ｈ／Ｌフィールド１０２のビットを立て（これがＬＱインデックスであることを示す）、ＬＱのＣＡＭ１００を探索することによって見つかる。この探索によって見つかった項目を、ＨＱ内のメモリラインの新しい項目のインデックスをリンクフィールド１０３に置くことによって更新する。Ｈ／Ｌフィールド１０２のビットを寝かせる。

【００７１】

図２および図１１に示されたＤＲＱの実施例を使用して本発明を説明してきたが、本発明の原理は、リンクリストを使用するすべてのシステムに拡張できる。

【００７２】

図１２と図１３は、従来技術によるリンクリストと、本発明の好ましい実施例によるリンクリストとの間の本質的な相違を示す図である。

【００７３】

図１２は、従来技術によるリンクリストを示す図である。リンクリスト項目１２１には、リンク１３１が含まれる。リンク１３１は、次のリンクリスト項目１２２を識別するインデックスまたはアドレスである。リンクリスト項目１２２には、リンク１３２が含まれる

。リンク１３２は、次のリンクリスト項目１２３を識別するインデックスまたはアドレスである。リンクリスト項目１２３には、リンク１３３が含まれる。リンク１３３は、次のリンクリスト項目１２４を識別するインデックスまたはアドレスである。リンクリスト項目１２４には、リンク１３４が含まれる。リンク１３４は、次のリンクリスト項目を識別するインデックスまたはアドレスである。以下同様である。

【００７４】

横断の方向１３０でリンクリストを横断する時には、現在の項目のリンクを使用して、次の項目にアクセスする。したがって、リンク１３１は、リンクリスト項目１２２のアクセスに使用される。リンク１３２は、リンクリスト項目１２３のアクセスに使用される。リンク１３３は、リンクリスト項目１２４のアクセスに使用される。以下同様である。実装の観点から見ると、これは、リスト内で新規の項目を作成する時に、前の末尾項目を変更しなければならない（前の末尾項目のリンクフィールドが既知になるのはこの時だけである）ことを意味する。

【００７５】

図１３は、本発明によるリンクリストを示す図である。図１３に示されたリンクリストでは、項目を突き止めるのに連想探索が使用されるが、図１２のリンクリストでは、メモリ様のアドレッシングが使用される。

【００７６】

図１３では、リンクリスト項目１４１が、リンクリストの先頭である。リンクリスト項目１４１はリンクリストの先頭であるから、リンク１５１に保持される値は、「無視（don't care）」である。次のリンクリスト項目１４２には、リンク１５２が含まれる。リンク１５２は、前のリンクリスト項目１４１を指すインデックスまたはアドレスである。次のリンクリスト項目１４３には、リンク１５３が含まれる。リンク１５３は、前のリンクリスト項目１４２を指すインデックスまたはアドレスである。次のリンクリスト項目１４４には、リンク１５４が含まれる。リンク１５４は、前のリンクリスト項目１４４を指すインデックスまたはアドレスである。以下同様である。

【００７７】

横断の方向１５０でリンクリストを横断する時には、次の項目にアクセスするために、現在の項目のインデックスに対する探索を行う。現在の項目のインデックスは、次の項目のリンクに格納されるので、この探索は成功する。したがって、リンクリスト項目１４１からは、リンク１５２に格納されたリンクリスト項目１４１のインデックスを使用して、リンクリスト項目１４２にアクセスする。リンクリスト項目１４２からは、リンク１５３に格納されたリンクリスト項目１４２のインデックスを使って、リンクリスト項目１４３にアクセスする。リンクリスト項目１４３からは、リンク１５４に格納されたリンクリスト項目１４３のインデックスを使って、リンクリスト項目１４４にアクセスする。以下同様である。次の項目の中からの現在の項目のインデックスを使って次の項目を効率的に突き止めることができるのは、上で示したように、リンクリスト項目の適当なフィールドが連想記憶（ＣＡＭ）に格納されているからである。これによって、直接探索ではなく連想探索の実行が可能になる。この実施例では、リンクリストに追加項目を追加する時に、末尾（最終）ビットをリセットすることだけが必要なので、リスト内の前の末尾項目を修正する必要はない。末尾（最終）ビットのリセットは、このビット用の特殊なセルを使用するＣＡＭ探索の副作用として実行できる。その結果、ハードウェアでリンクリストを実装するブロックに必要な帯域幅が大幅に節約される。

【００７８】

ここまでの議論で、本発明の方法および実施態様の例を開示し、説明した。当業者に理解されるとおり、本発明は、その趣旨または本質的な特性から逸脱することなく、他の具体的な形態で実施できる。したがって、本発明の開示は、本発明の範囲の制限ではなく例示を目的とするものであり、本発明の範囲は、請求項によって示される。

【００７９】

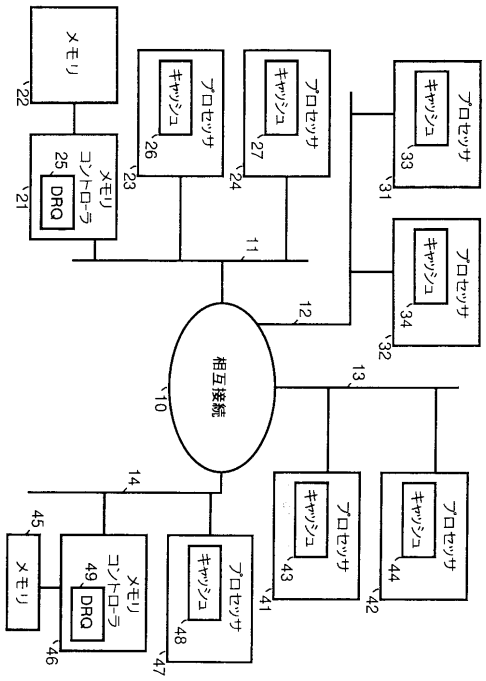
以下に本発明の実施の形態を要約する。

１．　リンクリスト（１４１～１４４）内の第１の項目（１４１）にアクセスし、
前記リンクリスト（１４１～１４４）内の第２の項目（１４２）にアクセスし、前記第２の項目（１４２）の少なくとも一部を含む連想記憶（５０）から前記リンクリスト（１４１～１４４）内の前記第１の項目（１４１）への参照（１５２）を探索する計算システムにおいて、
前記リンクリスト（１４１～１４４）を横断するためのリンクリスト形成方法。
【００８０】
２．　前記第２の項目（１４２）の少なくとも一部を含む前記連想記憶（５０）から前記リンクリスト（１４１～１４４）内の前記第１の項目（１４１）への前記参照（１５２）を探索する前記ステップが、探索される項目が有効であるかどうかの有効性ビット（５４）表示に対する探索を含む上記１に記載のリンクリスト形成方法。
【００８１】
３．前記第２の項目（１４２）の少なくとも一部を含む前記連想記憶（５０）から前記リンクリスト（１４１～１４４）内の前記第１の項目（１４１）への前記参照（１５２）を探索する前記ステップでの前記第１の項目（１４１）への前記参照（１５２）が、前記連想記憶（５０）内の前記第１の項目のインデックスである上記１または２に記載のリンクリスト形成方法。
【００８２】
４．　リンクリスト（１４１～１４４）内の第１の項目（１４１）と、
前記リンクリスト（１４１～１４４）内の追加項目とを含み、
前記追加項目のそれぞれが、前記リンクリスト（１４１～１４４）内の前の項目へのリンク参照（５５、１５１～１５４）を含み、各前記追加項目の前記リンク参照（５５、１５１～１５４）のすべてが、連想記憶（５０、１００）に格納され、これによって、次の追加項目内の前項目への前記リンク参照（５５、１５１～１５４）を使用する連想探索を実行することによって次の追加項目へのアクセスが可能になる計算システムにおけるリンクリスト（１４１～１４４）構造。
【００８３】
５．　前記連想記憶（５０、１００）に関連する追加項目が有効であるかどうかを示す有効性ビット（５４、１０１）が追加項目ごとに格納される上記４に記載のリンクリスト構造。
【００８４】
６．　前記第１の項目（１４１）の場合に真、前記追加項目のそれぞれの場合に偽になる値を含む頭部フィールド（５１）と、
前記リンクリスト（１４１～１４４）の最後の項目の場合に真になる値を含む尾部フィールド（５２）と、
メモリラインのアドレスを含むアドレスフィールド（５３）と、
関連する追加項目が有効であるかどうかを示す有効性ビット（５４）とが、第１の項目（１４１）および前記追加項目のそれぞれについて、連想記憶（５０）に格納される上記４または５に記載のリンクリスト構造。
【００８５】
７．　メモリラインのアドレスを含むアドレスフィールド（８２）と、
関連する追加項目が有効であるかどうかを示す有効性ビット（８１）とが、前記第１の項目（１４１）に関して頭部待ち行列連想記憶（８０）に記憶され、前記頭部待ち行列連想記憶（８０）が前記連想記憶（１００）に追加される上記４または５に記載のリンクリスト構造。
【００８６】
８．　前記第１の項目（１４１）の追加情報がランダムアクセスメモリ（９０）に格納され、追加情報が、
メモリラインに対して実行される動作と、
前記第１の項目（１４１）が前記リンクリスト（１４１～１４４）内の最後の項目である

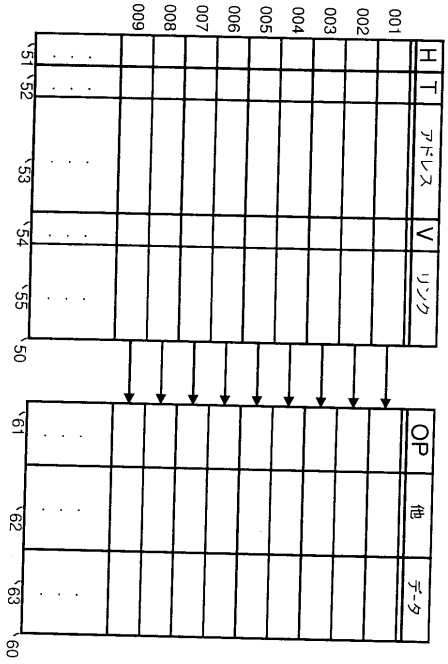
時に真になる値を含む最終フィールド（９３）と、
有効である時に前記リンクリスト（１４１～１４４）内の前記最後の項目のインデックスを含む尾部フィールド（９４）とを含む上記７に記載のリンクリスト構造。
【００８７】
９．　前記追加情報が有効である時に前記メモリラインの現在のデータを記憶するデータフィールド（９５）をさらに含む上記８に記載のリンクリスト構造。
【００８８】
１０．　前記リンク参照（５５、１５１～１５４）によって参照される前記前の項目が前記連想記憶（１００）内にあるか、前記頭部待ち行列連想記憶（８０）内にあるかを示す値を含む頭部／リンクフィールド（１０２）が、前記追加項目のそれぞれについて前記連想記憶（１００）に格納される上記８に記載のリンクリスト構造。
【００８９】
【発明の効果】
上述のように、本発明のリンクリスト形成方法によれば、リンクリスト項目の適当なフィールドが連想記憶（ＣＡＭ）に格納されることによって連想探索が実現可能となる。よって、リンクリストの現在の項目から次の項目を効率的に探索することができる。また、本実施例では、リンクリストに追加項目を追加する時に、末尾（最終）ビットをリセットすることだけが必要なので、リスト内の前の末尾項目を修正する必要はないため、ハードウェアでリンクリストを実装するブロックに必要な帯域幅が大幅に節約される。
【図面の簡単な説明】
【図１】本発明の好ましい実施例による計算システムのさまざまな実体の相互接続を示す図である。
【図２】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するためにメモリコントローラ内で使用される据置き読取り待ち行列を示す図である。
【図３】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図４】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図５】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図６】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図７】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図８】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図９】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図１０】本発明の好ましい実施例による、即座に満足できないメモリライン要求を保持するための据置き読取り待ち行列の使用を示すため、図２に示した据置き読取り待ち行列内の内容を示す図である。
【図１１】本発明の好ましい代替実施例による、即座に満足できないメモリライン要求を保持するためにメモリコントローラ内で使用される据置き読取り待ち行列を示す図である

。
【図１２】従来技術による単純化されたリンクリストを示す図である。
【図１３】本発明の好ましい実施例に従って構成された、単純化されたリンクリストを示す図である。
【符号の説明】
１０　相互接続
１１、１２、１３、１４　バス
２１、４６　メモリコントローラ
２２、４５　メモリ
２３、２４、３１、３２、４１、４２、４７　プロセッサ
２５、４９　据置き読取り待ち行列（ＤＲＱ）
２６、２７、３３、３４、４３、４４、４８　キャッシュ
５０、８０　連想記憶（ＣＡＭ）
５１　頭部（Ｈ）フィールド
５２、９４　尾部（Ｔ）フィールド
５３、８２　アドレスフィールド
５４、８１、１０１　有効（Ｖ）フィールド
５５、１０３　リンクフィールド
６０、９０、１１０　ランダムアクセスメモリ（ＲＡＭ）
６１、９１、１１１　動作（ＯＰ）フィールド
６２、９２、１１２　他フィールド
６３、９５　データフィールド
９３　最終（Ｌ）フィールド
１００　連想記憶（ＣＡＭ）
１０２　頭部／リンク（Ｈ／Ｌ）フィールド
１２１～１２４、１４１～１４４　リンクリスト項目
１３１～１３４、１５１～１５４　リンク

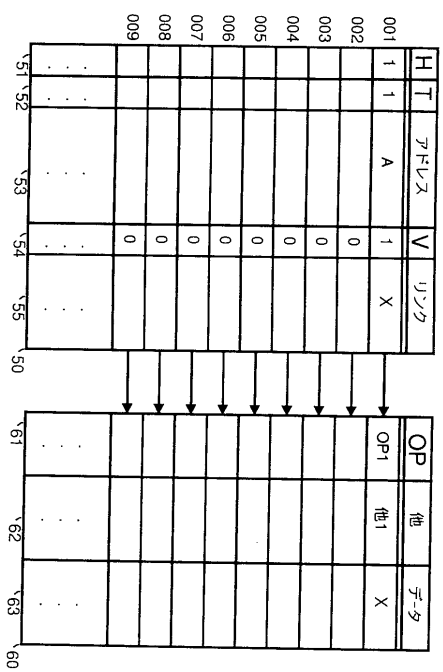
【図１】


メモリ
22
メモリ
コントローラ
25
DRQ
23
21
プロセッサ
キャッシュ
26
24
プロセッサ
キャッシュ
27
11
プロセッサ
キャッシュ
33
31
プロセッサ
キャッシュ
34
32
12
相互接続
10
13
14
プロセッサ
キャッシュ
43
41
プロセッサ
キャッシュ
44
42
45
メモリ
メモリ
コントローラ
49
DRQ
46
プロセッサ
キャッシュ
48
47

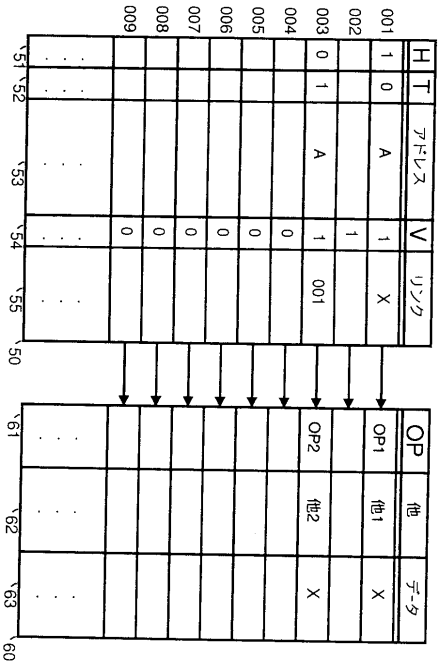
【図２】


001
002
003
004
005
006
007
008
009
H
T
アドレス
V
リンク
51
52
53
54
55
50
OP
他
データ
61
62
63
60

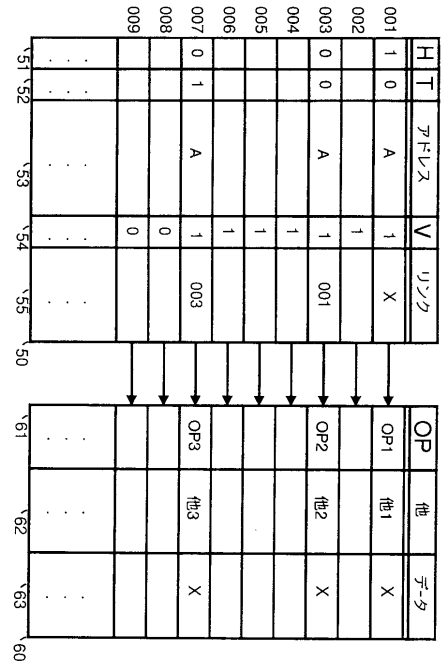
【図３】


001
002
003
004
005
006
007
008
009
H
T
アドレス
V
リンク
1
1
A
1
X
0
0
0
0
0
0
0
0
51
52
53
54
55
50
OP
他
データ
OP1
他1
X
61
62
63
60

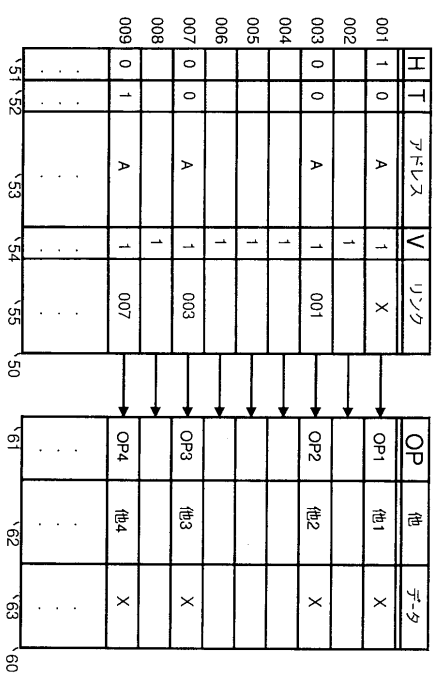
【図４】


001
002
003
004
005
006
007
008
009
H
T
アドレス
V
リンク
1
0
A
1
X
0
1
A
1
001
0
0
0
0
0
0
0
0
51
52
53
54
55
50
OP
他
データ
OP1
他1
X
OP2
他2
X
61
62
63
60

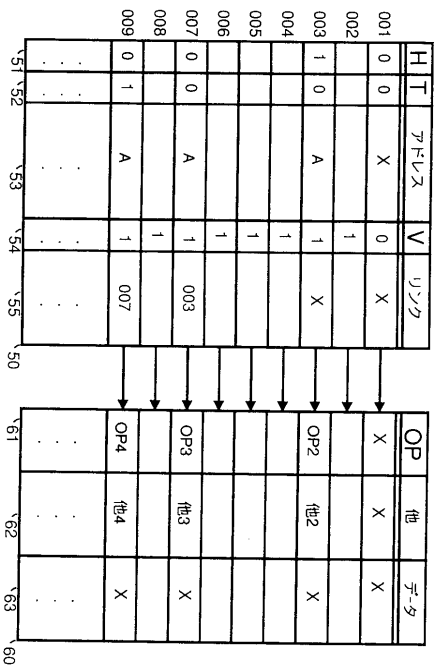
【図５】


H
T
アドレス
V
リンク
OP
他
データ
OP1
他1
X
OP2
他2
OP3
他3
001
002
003
004
005
006
007
008
009
50
51
52
53
54
55
60
61
62
63

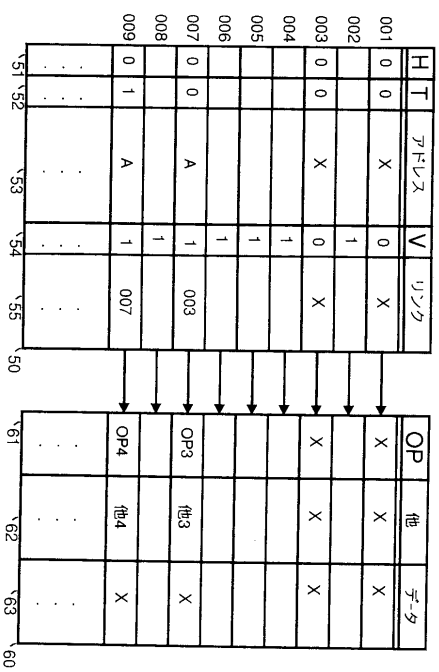
【図６】


H
T
アドレス
V
リンク
OP
他
データ
OP1
他1
X
OP2
他2
OP3
他3
OP4
他4
001
002
003
004
005
006
007
008
009
50
51
52
53
54
55
60
61
62
63

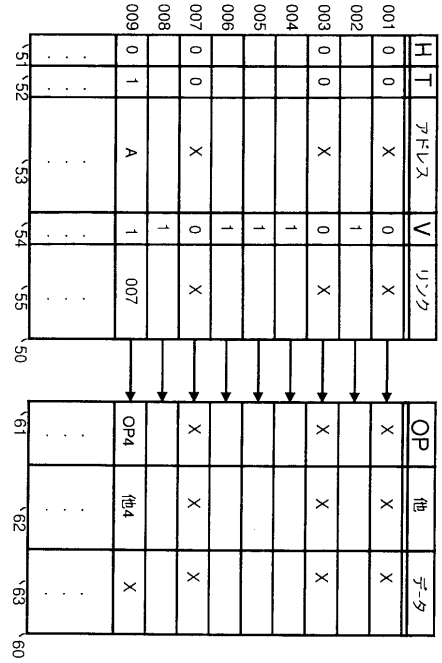
【図７】


H
T
アドレス
V
リンク
OP
他
データ
OP2
他2
OP3
他3
OP4
他4
001
002
003
004
005
006
007
008
009
50
51
52
53
54
55
60
61
62
63

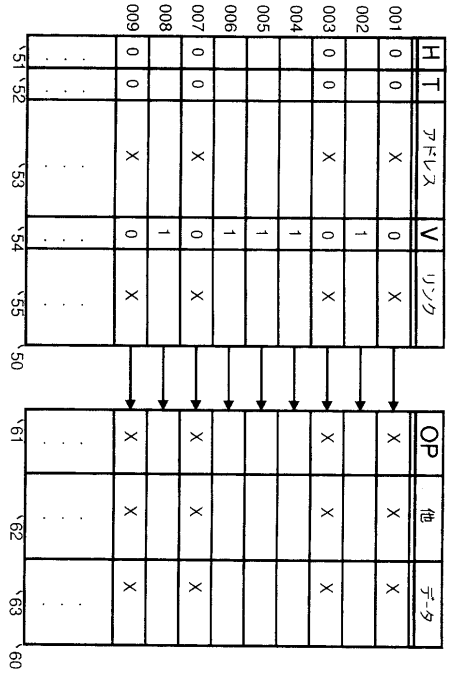
【図８】


H
T
アドレス
V
リンク
OP
他
データ
OP3
他3
OP4
他4
001
002
003
004
005
006
007
008
009
50
51
52
53
54
55
60
61
62
63

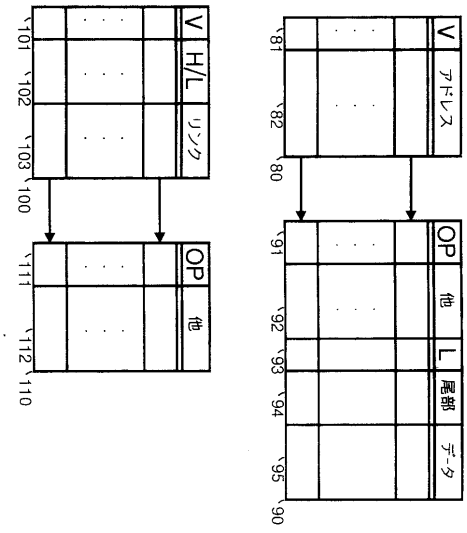
【図９】



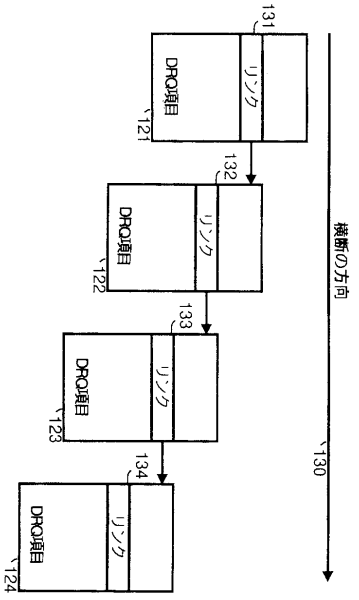
【図１０】



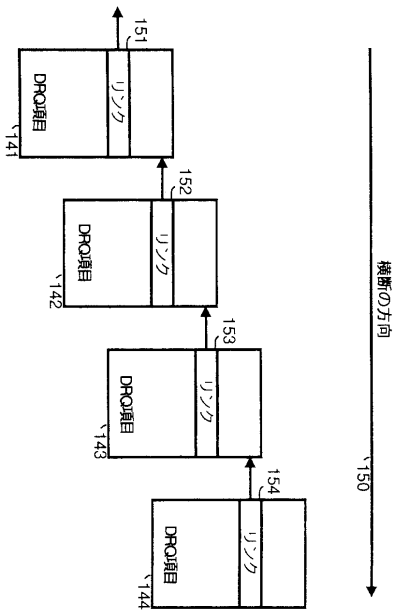
【図１１】



【図１２】



【図１３】


151
DRQ項目
141
リンク
152
DRQ項目
142
リンク
153
DRQ項目
143
リンク
154
DRQ項目
144
リンク
横断の方向
150

フロントページの続き

(72)発明者　ウィリアム・アール・ブリグ
アメリカ合衆国　カリフォルニア，サラトガ，ペレゴ・ウェイ　１８６３０
(72)発明者　ジョセフ・エッチ・ハスソウン
アメリカ合衆国　カリフォルニア，プレアサントン，ピー・オー・ボックス　１１９３７

審査官　清木　泰

(56)参考文献　特開昭６３－１２１９４１（ＪＰ，Ａ）
特開昭５９－００８１９５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F12/08-12/12
G06F12/00,550-12/06
G06F13/16-13/18
G11C15/00-15/06
G06F15/16-15/177