

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2012-203729

(P2012-203729A)

(43) 公開日 平成24年10月22日(2012.10.22)

(51) Int.Cl.	F I	テーマコード (参考)
G 0 6 F 12/08 (2006.01)	G 0 6 F 12/08 5 0 7 E	5 B 0 0 5
	G 0 6 F 12/08 5 1 1 E	
	G 0 6 F 12/08 5 6 5	

審査請求 未請求 請求項の数 10 O L (全 35 頁)

(21) 出願番号	特願2011-68861 (P2011-68861)	(71) 出願人	000005223
(22) 出願日	平成23年3月25日 (2011. 3. 25)		富士通株式会社
			神奈川県川崎市中原区上小田中4丁目1番1号
		(74) 代理人	100074099
			弁理士 大菅 義之
		(74) 代理人	100133570
			弁理士 ▲徳▼永 民雄
		(72) 発明者	山村 周史
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	森田 國樹
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		Fターム(参考)	5B005 JJ11 KK12 MM01 NN31

(54) 【発明の名称】 演算処理装置および演算処理装置の制御方法

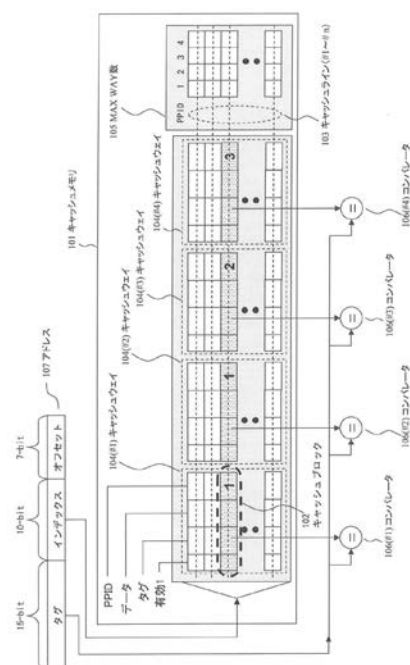
(57) 【要約】

【課題】 キャッシュメモリを実装した演算処理装置およびキャッシュメモリ制御装置において、プロセスIDに対応してキャッシュメモリ領域をブロック単位で任意に分割可能として、プロセッサの実効性能を向上することを可能とする。

【解決手段】 各セット103のキャッシュブロック102毎に物理プロセスID (PPID) が記憶されるとともに、#1から#nの各インデックス値毎に、各PPID値に対するMAX WAY数105が記憶される。或るインデックス値における或るPPID値に対応するMAX WAY数105は、そのインデックス値において記憶可能なそのPPID値を有するキャッシュブロック102の最大数を示す。各インデックス値毎に、各PPID値のMAX WAY数105が守られるように、キャッシュミス時のウェイ数の制御が実施される。

【選択図】 図1

キャッシュメモリの実施形態のブロック図



【特許請求の範囲】

【請求項 1】

複数の命令を含むプロセスを実行するとともに、インデックス情報とタグ情報を含むメモリアクセス要求を発行する命令制御部と、

タグと、前記メモリアクセス要求に対応するデータと、前記命令制御部が実行するプロセスを識別するプロセス識別子を保持するブロックを、複数のインデックス各々に対応して有するキャッシュウェイを複数備えたキャッシュメモリ部と、

受信したメモリアクセス要求に含まれるインデックス情報をデコードし、前記デコードしたインデックス情報に対応するブロックを選択するインデックスデコード部と、

受信したメモリアクセス要求に含まれるタグ情報と前記インデックスデコード部が選択したブロックに含まれるタグを比較し、前記タグ情報と前記タグが一致する場合には前記インデックスデコード部が選択したブロックに含まれるデータを出力する比較部と、

前記プロセス識別子毎に設定された最大キャッシュウェイ数情報に基づき、前記キャッシュメモリ部のインデックス毎に、前記プロセス識別子で識別されるプロセスが使用するキャッシュウェイ数を決定する制御部と、

を有することを特徴とする演算処理装置。

【請求項 2】

前記命令制御部は、制御プログラムを実行して、プロセス識別子毎に設定された最大キャッシュウェイ数情報に基づき、前記キャッシュメモリ部のインデックス毎に、前記プロセス識別子で識別されるプロセスが使用するキャッシュウェイ数を決定することを特徴とする請求項 1 記載の演算処理装置。

【請求項 3】

前記演算処理装置において、

前記比較部による比較の結果、前記タグ情報に一致するタグが選択したブロックに存在せずキャッシュミスが発生した場合、前記キャッシュメモリ部は、前記演算処理装置に接続されたメインメモリから読み出した前記メモリアクセス要求に対応するデータを、設定された最大キャッシュウェイ数情報を超えて使用しているプロセスが使用中のブロックのいずれかが保持するデータと置換することを特徴とする請求項 1 又は 2 記載の演算処理装置。

【請求項 4】

前記制御部は、

プロセス識別子毎に、各プロセス識別子に割り当てる最大ブロック数を、1つのキャッシュウェイあたりのブロック数で除算して、各プロセス識別子に割り当てるキャッシュウェイ数を算出し、

プロセス識別子毎に、各プロセス識別子に割り当てる最大ブロック数を、1つのキャッシュウェイあたりのブロック数で除算した剰余を算出して、各プロセス識別子における1キャッシュウェイ分に満たないキャッシュウェイ数を算出し、

プロセス識別子毎に、前記キャッシュメモリ部内のすべてのインデックスについて、前記各プロセス識別子に割り当てるキャッシュウェイ数を前記各プロセス識別子に対応する最大キャッシュウェイ数として設定し、

プロセス識別子毎に、前記算出した各プロセス識別子における1キャッシュウェイ分に満たないブロック数のインデックス分だけ、前記各プロセス識別子に対応する最大キャッシュウェイ数を加算し、

前記加算後の最大キャッシュウェイ数を、前記各プロセス識別子で識別されるプロセスが使用するキャッシュウェイ数として決定することを特徴とする請求項 1 記載の演算処理装置。

【請求項 5】

前記比較部による比較の結果、前記タグ情報に一致するタグが選択したブロックに存在せずキャッシュミスが発生した場合、前記メモリアクセス要求を発生させたプロセスを識別する要求元プロセス識別子と、前記メモリアクセス要求により特定されるインデックス

10

20

30

40

50

の各キャッシュウェイに対応して前記キャッシュメモリ部に保持されているプロセス識別子と、前記メモリアクセス要求により特定されるインデックスに対応して決定される前記プロセス識別子毎の最大キャッシュウェイ数とに基づいて、前記メモリアクセス要求に対応するインデックスにおいて前記要求元プロセス識別子に対応するプロセスへの前記キャッシュメモリ部の領域の割当てを行なうキャッシュメモリ制御部を有することを特徴とする請求項 4 記載の演算処理装置。

【請求項 6】

前記キャッシュメモリ制御部は、

前記比較部による比較の結果、前記タグ情報に一致するタグが選択したブロックに存在せずキャッシュミスが発生した場合、前記メモリアクセス要求に含まれるインデックスの各キャッシュウェイに対応して前記キャッシュメモリ部に保持されている各プロセス識別子が前記要求元プロセス識別子と一致するか否かを値 1 または 0 で示すビットマスクを生成するマスク生成部と、

前記生成されたビットマスクの「1」または「0」の数を計数する計数部と、

前記計数部が計数した値の数が前記要求元プロセス識別子に対応する最大キャッシュウェイ数に満たない場合には、前記マスク生成部が出力するビットマスクの各ビットを反転したビットマスクを出力し、前記計数部が計数した所定値の数が前記要求元プロセス識別子に対応する最大キャッシュウェイ数に達している場合には、前記マスク生成部が出力したビットマスクを出力するビットマスク選択部と、

前記ビットマスク選択部が出力したビットマスクに基づき、前記複数のキャッシュウェイから置換するキャッシュウェイを決定する置換ウェイ決定部を備えることを特徴とする請求項 5 記載の演算処理装置。

【請求項 7】

プロセス識別子に割り当てるキャッシュウェイ数が 0 である場合、前記メモリアクセス要求に含まれる要求アドレスに含まれる部分アドレス情報を、前記プロセス識別子における 1 キャッシュウェイ分に満たないブロック数で除算した剰余に前記所定のインデックス開始位置を加えた値を、前記インデックスデコード部の出力とし、プロセス識別子に割り当てるキャッシュウェイ数が 0 でない場合、前記要求アドレスに含まれるインデックス情報を前記インデックスデコード部の出力とするアドレスハッシュ生成部を備えることを特徴とする請求項 4 記載の演算処理装置。

【請求項 8】

前記キャッシュメモリ部は、前記最大キャッシュウェイ数を前記複数のインデックス毎および前記プロセス識別子毎に記憶するメモリを備え、

前記制御部は、前記メモリアクセス要求では使用されないアドレスを指定して、前記最大キャッシュウェイ数の更新を指示し、

前記キャッシュメモリ部は、前記制御部が指定する前記アドレスを前記メモリのアドレス空間のアドレスに変換して、前記プロセス識別子に対応する最大キャッシュウェイ数を更新することを特徴とする請求項 4 に記載の演算処理装置。

【請求項 9】

前記プロセス識別子は、前記命令制御部が実行するプロセスを複数の種類にグループ化したときの該各グループを識別し、前記命令制御部が実行するプロセスの実プロセス ID と前記プロセス識別子との対応関係を保持する連想メモリを備え、

前記命令制御部が実行するプロセスの実プロセス ID をキーとして前記連想メモリ部を検索して、前記実プロセス ID に対応するプロセス識別子を取得して、前記キャッシュメモリ制御部に出力するプロセス ID マップ部を備えることを特徴とする請求項 1 記載の演算処理装置。

【請求項 10】

タグと、データと、実行対象のプロセスに対応するプロセス識別子を保持するブロックとを複数のインデックスに対応して有するキャッシュウェイを複数備えたキャッシュメモリ部を有する演算処理装置の制御方法において、

10

20

30

40

50

前記演算処理装置が有する命令制御部が、複数の命令を含むプロセスを実行するとともに、インデックス情報とタグ情報を含む、前記データに対するメモリアクセス要求を発行し、

前記演算処理装置が有するインデックスデコード部が、受信したメモリアクセス要求に含まれるインデックス情報をデコードし、前記デコードしたインデックス情報に対応するブロックを選択し、

前記演算処理装置が有する比較部が、受信したメモリアクセス要求に含まれるタグ情報と前記インデックスデコード部が選択したブロックに含まれるタグを比較するとともに、前記タグ情報と前記タグが一致する場合には前記インデックスデコード部が選択したブロックに含まれるデータを出力し、

前記演算処理装置が有する制御部が、前記プロセス識別子毎に設定された最大キャッシュウェイ数情報に基づき、前記キャッシュメモリ部のインデックス毎に、前記プロセス識別子で識別されるプロセスが使用するキャッシュウェイ数を決定することを特徴とする演算処理装置の制御方法。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、演算処理装置および演算処理装置の制御方法に関する。

【背景技術】

【0002】

近年のプロセッサの動作周波数の向上により、相対的にプロセッサ内部からメインメモリに対するメモリアクセスの遅延時間が長くなり、メモリアクセスの遅延時間がシステム全体の性能を左右するに至っている。多くのプロセッサは、メモリアクセス遅延時間を隠蔽する目的で、キャッシュメモリと呼ぶ小容量の高速メモリを搭載している。

【0003】

キャッシュメモリは、データを複数のキャッシュライン（もしくは単に「ライン」）またはキャッシュブロック（もしくは単に「ブロック」）と呼ばれる単位で管理する。プロセッサからデータのアクセス要求があった時に、そのデータがキャッシュ内のいずれかのラインに存在しているか否かを高速に検索する必要がある。

【0004】

このためキャッシュメモリを分割して検索等の処理を行なうことが行われる。

プロセッサが実行するオペレーティングシステム(Operating System; OS)によって共有キャッシュ領域を分割管理する手法として従来、Modified LRU Replacement方式と呼ばれる第1の従来技術が知られている。この第1の従来技術では、システム上で動作する全プロセスについて、プロセスが使用しているキャッシュブロック数がカウントされる。

【0005】

また、キャッシュブロック内のタグ（キャッシュタグ）にプロセッサが実行するプロセスを識別するプロセスIDを記憶し、プロセスIDによってキャッシュフラッシュを制御する第2の従来技術が知られている。

【0006】

さらに、キャッシュタグ内にプロセスIDを記録して、キャッシュアクセス時に要求元プロセスIDとキャッシュタグ内のプロセスIDとを比較することで、キャッシュフラッシュを制御する第3の従来技術が知られている。

【先行技術文献】

【特許文献】

【0007】

【特許文献1】特開平3-235143号公報

【特許文献2】特許2700148号公報

【非特許文献】

【0008】

10

20

30

40

50

【非特許文献 1】 Suh, G.E. and Devadas, S. and Rudolph, L., "A new memory monitoring scheme for memory-aware scheduling and partitioning", High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on, pp.117--128.

【発明の概要】

【発明が解決しようとする課題】

【0009】

しかし、第 1 の従来技術では、使用中のキャッシュブロック数を、全プロセスについて正しく把握するような機構が必要となつて、ハードウェア規模が増大してしまう。また、マルチプロセス環境においてはその効率的な動作の観点から問題があった。

10

【0010】

また、第 2 の従来技術では、各キャッシュタグにプロセス ID を固定的に割り当てるのみである。このため、キャッシュメモリ中でのプロセス ID 間の全体的なサイズ割当てを変更するためには、全キャッシュタグの書換えが必要となってしまうという問題点を有していた。

【0011】

さらに、第 3 の従来技術も、キャッシュメモリ中でのプロセス ID 間の全体的なサイズ割当てを変更するような機構は備えていないという問題点を有していた。

このため、キャッシュメモリのより効率的な動作が望まれていた。

【0012】

本発明の 1 つの側面では、プロセス ID に対応してキャッシュメモリ領域をブロック単位で任意に分割可能として、プロセッサの実効性能を向上することを可能とすることにある。

20

【課題を解決するための手段】

【0013】

態様の一例では、複数の命令を含むプロセスを実行するとともに、インデックス情報とタグ情報を含むメモリアクセス要求を発行する命令制御部と、タグと、メモリアクセス要求に対応するデータと、命令制御部が実行するプロセスを識別するプロセス識別子を保持するブロックを、複数のインデックス各々に対応して有するキャッシュウェイを複数備えたキャッシュメモリ部と、受信したメモリアクセス要求に含まれるインデックス情報をデコードし、デコードしたインデックス情報に対応するブロックを選択するインデックスデコード部と、受信したメモリアクセス要求に含まれるタグ情報とインデックスデコード部が選択したブロックに含まれるタグを比較し、タグ情報とタグが一致する場合にはインデックスデコード部が選択したブロックに含まれるデータを出力する比較部と、プロセス識別子毎に設定された最大キャッシュウェイ数情報に基づき、キャッシュメモリ部のインデックス毎に、プロセス識別子で識別されるプロセスが使用するキャッシュウェイ数を決定する制御部とを有するように構成する。

30

【発明の効果】

【0014】

キャッシュメモリ領域をキャッシュブロック単位で任意に分割し、各プロセスに適切なキャッシュブロック数を割り当てることが可能となる。これにより、キャッシュメモリをリソースとして管理し、プロセススケジューリングを最適化することが可能となり、プロセッサの実効性能を向上させることが可能となる。

40

【図面の簡単な説明】

【0015】

【図 1】 キャッシュメモリの実施形態のブロック図である。

【図 2】 OS が各 P P I D 値に与えるキャッシュブロック数のテーブルのデータ構成例を示す図である。

【図 3】 キャッシュメモリの分割例を示す図である。

【図 4】 キャッシュミス発生時のリプレース動作を示す説明図である。

50

【図 5】ハッシュユニットを示す図である。

【図 6】プロセス ID マップユニットを示す図である。

【図 7】キャッシュタグ部のハードウェア構成例を示す図（その 1）である。

【図 8】キャッシュタグ部のハードウェア構成例を示す図（その 2）である。

【図 9】OS が各 P P I D 値に与えるキャッシュブロック数に基づいて M A X W A Y 数を決定する処理を示すフローチャートである。

【図 10】OS が各 P P I D 値に与えるキャッシュブロック数に基づいて M A X W A Y 数を決定する処理を示すプログラム擬似コードである。

【図 11】置換ウェイ制御回路のハードウェア構成例を示す図である。

【図 12】M A X W A Y 数更新機構を示す図である。

10

【図 13】ハッシュユニットのハードウェア構成例を示す図である。

【図 14】ハッシュユニットの動作説明図（その 1）である。

【図 15】ハッシュユニットの動作説明図（その 2）である。

【図 16】プロセス ID マップユニットのハードウェア構成例を示す図である。

【図 17】P P I D 書込み機構を示す図である。

【図 18】本実施形態のキャッシュメモリシステムを備えるプロセッサシステムの構成例を示す図である。

【図 19】同時にスケジューリングされる各プロセスが要求するウェイ数の合計が、実装されているキャッシュメモリのウェイ数を超えている場合の動作例を示す説明図である。

【図 20】時間と優先度でキャッシュブロックをスケジューリングする動作を示すフローチャートである。

20

【発明を実施するための形態】

【0016】

プロセッサの実効性能の向上のためには、キャッシュメモリの高速動作が必要である。

データがキャッシュメモリ内のいずれかのラインに存在しているか否かを高速検索するため、各キャッシュセット（以下、単にセットと略記）を構成するキャッシュブロックは、有効か否かを示す有効フラグ、タグ、及びデータから構成されている。キャッシュブロックのサイズは例えば、有効フラグが 1 ビット、タグが 15 ビット、データが 128 バイトである。ここで、キャッシュセットとは、分割されたキャッシュメモリの領域をいい、各キャッシュセットは複数のキャッシュブロックを含む。

30

【0017】

一方、プログラムによって指定されるメモリアクセスのための例えば 32 ビットのアドレスは例えば、下位から 7 ビットがキャッシュライン内オフセット、10 ビットがインデックス、上位 15 ビットがタグとして使用される。

【0018】

アドレスに対するデータ読出しが要求されると、アドレス内のインデックスアドレスが示すセットが選択される。さらに、選択されたセット内の各キャッシュブロックに対応する形で記憶されているタグがアドレス内のタグと一致するか否かが判定され、タグが一致する場合にはキャッシュヒットが検出され、タグが一致しない場合にはキャッシュミスが検出される。

40

【0019】

このとき、セット内に複数ウェイのキャッシュブロック（データとタグの組）を持てば、同じインデックス値を有するエントリでも上位アドレス値（タグ値）が異なる複数のデータを格納することが可能となる。このようなキャッシュメモリのデータ格納方式はセット・アソシアティブ (Set Associative) 方式と呼ばれる。メモリのアドレス空間より小さい空間となっているキャッシュのアドレス空間をセット（集合）に分割し、例えば要求アドレスをそのセットの数で割った余りの数をインデックスとすればセットの数はインデックス数に対応する。各セット（インデックス）は複数のブロックを含むが、インデックスの指定によって同時に出力されるブロック数がウェイ数である。1 ラインが n 個のタグにより構成される n 個のブロックを同時に出力する場合、 n ウェイセット・アソシアティブ

50

(n-way Set Associative)方式と呼ぶ。

【0020】

書き込まれるデータのサイズがインデックスで指定可能なアドレスの範囲よりも大きい場合に、複数のデータにおいてアドレスの一部分であるインデックスの値が一致し、それらのデータがキャッシュラインを奪い合う競合が発生する可能性がある。このような場合であっても、セット・アソシアティブ方式のキャッシュメモリにおいては、インデックスが同じラインが指定されたとしても、キャッシュラインの競合を発生することなく複数のウェイからキャッシュブロックを選択できる。例えば4ウェイ構成のキャッシュメモリでは、同じインデックスを持つ最大で4つまでのデータに対応することができる。

【0021】

指定されたラインのどのウェイのキャッシュブロックにおいてもタグの一致が検出されなかった、またはタグの一致が検出されたキャッシュブロックの有効フラグが無効を示していたら、キャッシュミスとなり、メインメモリ（主記憶装置）からアクセス対象のデータが読み出される。キャッシュミスの発生時には、指定されたセット上から未使用のウェイが選択されて、そのウェイのキャッシュブロックにメインメモリから読み出されたデータが新たに保持される。これにより、保持されたデータが次回アクセス時にキャッシュヒットし、メインメモリへのアクセスが不要となるため、高速なアクセスが実行される。キャッシュミス時にどのウェイも使用中の場合には、例えばLRU（Least Recently Used）と呼ばれるアルゴリズムによって、使用中のウェイから1つが選択されて、そのウェイのキャッシュブロックのデータが置換される。LRUアルゴリズムでは、使われてから最も長い時間が経っているキャッシュブロックのデータがメインメモリに追い出されるとともに、メインメモリから読み出されたデータに置換される。

セット・アソシアティブ方式のキャッシュメモリは以上のような構成を有す。

【0022】

以下、本発明を実施するための形態について図面を参照しながら詳細に説明する。

図1は、キャッシュメモリの実施形態のブロック図である。

本実施形態によるキャッシュメモリ101は、例えば4ウェイまたは8ウェイのセット・アソシアティブ方式のキャッシュメモリである。

キャッシュメモリ101は、データを#1から#nの複数行からなるセット103、および各セット103に属するキャッシュブロック102の単位で管理される。例えば、 $n = 1024$ である。

【0023】

図1の実施形態においては、各セット103を構成するキャッシュブロック102は、有効フラグ（例えば1ビット）、タグ（例えば15ビット）、データ（例えば128バイト）に加えて、物理プロセスID（以下、「PPID」と称する）を有する。PPIDは、オペレーティングシステムが管理するプロセスID（以下、「PID」と称する）を、後述するプロセスIDマップユニットによって変換して得られるプロセス識別情報である。PPIDは、例えば2ビットのデータであり、例えば0～3の4つのPPID値を識別することができる。PPIDを記憶することで、各キャッシュブロック102がどのプロセスに割り当てられているかを区別することができる。

【0024】

キャッシュメモリ101のデータサイズ定義：は、「キャッシュブロック102のデータサイズ×キャッシュインデックス数×キャッシュウェイ数で計算され、例えば、1024バイトを1キロバイトとして、4ウェイのキャッシュメモリ101の場合、

$(128 \text{ バイト} \times 1024 \text{ インデックス} \times 4 \text{ ウェイ}) \div 1024 = 512 \text{ キロバイト}$ である。

【0025】

一方、プログラムによって指定されるメモリアクセスのためのアドレス107は、例えば32ビットで指定され、下位から7ビットがキャッシュライン内オフセット、10ビットがインデックス、上位15ビットがタグとして使用される。

【0026】

また、本実施形態では、プログラムを実行する場合にオペレーティングシステムから指定されるPIDを、プロセスIDマップユニットによって変換して得られるPPIDが、キャッシュメモリ101に与えられる。

【0027】

以上の構成により、アドレス107に対するデータの読み出し又は書き込みのアクセスが指定されると、アドレス107内の10ビットのインデックスにより、セット103内の#1～#nのキャッシュブロックのうちの1つが指定される。

【0028】

その結果、#1～#4の各キャッシュウェイ104から、セット103上の各キャッシュブロック102（#i）のタグ値が読み出され、それぞれ#1～#4のコンパレータ106に入力する。

10

【0029】

#1～#4のコンパレータ106は、読み出された各キャッシュブロック102（#i）内のタグ値と、指定されたアドレス107内のタグ値との一致／不一致を検出する。この結果、#1～#4のうちタグ値の一致が検出されたコンパレータ106において読み出されているキャッシュブロック102（#i）がキャッシュヒットとなり、そのキャッシュブロック102（#i）に対してデータが読み書きされる。

【0030】

どのコンパレータ106でもタグ値の一致が検出されなかった、又はタグ値の一致が検出されたキャッシュブロック102（#i）の有効フラグが無効を示していたら、キャッシュミスとなり、メインメモリ上のアドレスがアクセスされる。キャッシュミスの発生時には、指定されたライン上から選択された未使用のウェイのキャッシュブロックにデータが新たに保持される。これにより、次回アクセス時にキャッシュヒットとなり、メインメモリへのアクセスが不要となるため、高速なアクセスが実行される。

20

【0031】

キャッシュミス時にどのウェイも使用中の場合には、本実施形態では、以下に示されるような追い出し制御が実施される。

まず、本実施形態では、各セット103のキャッシュブロック102毎にPPIDが記憶されるとともに、#1から#nのインデックス値毎に、各PPID値（例えば1～4）に対するMAX WAY数（最大ウェイ数）105が記憶される。或るインデックス値における或るPPID値に対応するMAX WAY数105は、そのインデックス値において記憶可能なそのPPID値を有するキャッシュブロック102の最大数を示す。本実施形態では、各インデックス値毎に、各PPID値のMAX WAY数105が守られるように、追い出し制御が実施される。

30

【0032】

各PPID値毎のMAX WAY数105の割合は、オペレーティングシステム（OS）が定めたPPID値毎のキャッシュブロック数に基づいて決定される。この場合、キャッシュメモリ101中でのPPID値間のサイズ割当て、すなわち各PPIDが使用できるキャッシュメモリの領域のサイズを変更する場合には、各インデックス値へのアクセス発生時に、そのインデックス値の各PPID値毎のMAX WAY数105を順次変更する。キャッシュメモリ101を単純にPPID値によって分割すると、分割量を変更する場合にキャッシュメモリ101内の全キャッシュブロック102のPPID情報を書き換えなくてはならないので更新オーバーヘッドが大きくなる。これに対して、本実施形態では、一時期に全キャッシュブロック102の書換えを行わなくても、インデックス値単位で動的にPPID間のサイズ割当ての変更が可能となるため、情報の更新を最小限に抑えることにより、低オーバーヘッドで分割量の変更を行うことが可能となる。

40

【0033】

図2は、OSが保有する、OSが各PPID値に与える最大キャッシュブロック数のテーブルのデータ構成例を示す図である。PPID値がP1、P2、P3のとき、最大キャ

50

ッシュブロック数は例えば、それぞれ64, 21, 11である。さらに図3は、図2に示されるテーブル内容に従って本実施形態において実施されるキャッシュメモリ101の分割例を示す図である。この分割処理においては、キャッシュウェイ104の数が8ウェイである場合の例について示されている。キャッシュメモリのインデックス数は10ビットもしくは11ビットで表わされる数存在するが、ここでは説明を簡単にするため、インデックス方向には例として16個のインデックスがあるものとして説明を行なう。インデックス値毎に、PPID値（図3ではP1、P2、P3）毎のMAX WAY数105が保持される。そして、キャッシュメモリ101全体で、各PPID値に与えられるMAX WAY数105が、図2のテーブルで設定されたOSが各PPID値に与えるキャッシュブロック数に等しくなるように、各インデックス値毎のMAX WAY数105が設定される。

【0034】

指定されたインデックス値において、或るPPID値を有するキャッシュブロック102についてキャッシュミスが発生したときには、次のような動作が実行される。すなわち、そのセット103上で、そのPPID値に関して既に配分済みのキャッシュウェイ数の合計と、そのPPID値に対応して記憶されているMAX WAY数105とが比較される。配分済みのキャッシュウェイ数の合計がMAX WAY数105に満たない場合は、以下の動作が実行される。すなわち、そのインデックス値上で、他のPPID値に対して配分済みのキャッシュブロック102の中で、配分済みのキャッシュウェイ数の合計が当該PPID値に対応するMAX WAY数105を越えているキャッシュブロックの中から置換ブロックが選択される。

【0035】

図4は、キャッシュミス発生時のキャッシュブロックのリプレース動作を示す説明図である。キャッシュミスが発生したとき、図4に示されるように、例として、PPID値P1に4ブロック、P2に3ブロック、P3に1ブロックが割り当てられていたとする。ここで、P1についてキャッシュミスが発生したときには、P1はそのインデックス値上でのMAX WAY数105を超えておらず、一方、P2がそのインデックス値上でのMAX WAY数105を既に越えている。このため、PPID値としてP2を有するキャッシュブロック102の中からリプレース候補が選択され、図4中の矢印のブロックのデータがメインメモリから読み出されたデータに置換され、PPID値P1の要求するデータがロードされる。

【0036】

このように、本実施形態では、キャッシュミスのアクセスが発生したタイミングで、各PPIDに対するキャッシュサイズの割当てが動的に変更される。

キャッシュメモリ101における各PPIDに対するキャッシュサイズの割当てを変更する場合は、MAX WAY数105のマップを変更するだけでよい。MAX WAY数105の指示は、キャッシュアクセス命令に付随させて行うことができる。従来技術では、キャッシュメモリ101内の全キャッシュブロック102のプロセスIDを書き換えることが必要であった。これに対して、本実施形態では、キャッシュアクセス命令に付随して随時各PPIDに対するキャッシュサイズの割当てを変化させることができる。なお、全てのインデックス値について、一括して書き換えてもよい。

【0037】

また、同時にスケジューリングされる各プロセスが要求するウェイ数の合計が、実装されているキャッシュメモリ101のウェイ数を超えていても、ウェイの取り合いになるだけでシステム停止などの問題は発生しない。

【0038】

図2のテーブル例の場合、PPID値3に与えるキャッシュブロック数は11である。このため、PPID値3は、インデックス値の数（図3では16インデックス）の全てに対して、キャッシュブロックを割り当てることができない。このため、図3に示されるキャッシュメモリ101の分割例において、次のようなインデックス方向の割り当て変更が

必要となる。すなわち例えば、インデックス方向の先頭5インデックスの領域ではP P I D値P 3に対するM A X W A Y数1 0 5は0とされ、それ以後1 1インデックスの領域でのみP P I D値P 3に対するM A X W A Y数1 0 5が1とされる。このため、P P I D値3に対応するキャッシュアクセスが発生した場合、命令アドレス中のインデックスによって、先頭5インデックスの領域が指定されないようにし、常に後側1 1インデックスの領域が指定されるようにする必要がある。

【0 0 3 9】

この機能として、本実施形態では、図5に示されるハッシュ機構としてのアドレスハッシュユニット5 0 1が実装される。このハッシュ機構により、指定された命令アドレスをハッシュして得られるインデックスが、禁止された領域のインデックスを生成しないようにされる。

10

【0 0 4 0】

また、O Sが管理するプロセスI Dは、例えば1 6ビット以上の値を持つ。従って、1 6ビット以上の値で示されるプロセスI Dをキャッシュメモリ1 0 1中の各キャッシュブロック1 0 2に保持すると、ハードウェア追加量が大きくなる。そこで、本実施形態では、図6に示されるように、プロセスI Dマップユニット6 0 1が実装される。このプロセスI Dマップユニット6 0 1は、キャッシュアクセス命令を実行しているプロセスのプロセスI Dを、キャッシュメモリ1 0 1のハードウェアが取扱い可能な物理プロセスI D P P I Dにマップする。P P I Dは、例えば分割されたセット数を指定する2ビットの値を持てばよい。例えば1 6ビット以上の値で示されるプロセスI Dを保持する場合よりも、キャッシュメモリ1 0 1のハードウェア量の増大を防ぐことができる。

20

【0 0 4 1】

以上のハードウェア機構により、O Sは、プロセッサをプロセス間の共有資源として時分割でスケジューリングして使用するのと同様に、キャッシュメモリ1 0 1をプロセス間の共有資源としてサイズと時間で自由にスケジューリング可能となる。

【0 0 4 2】

例えば、図2のテーブル例のように各P P I D値にキャッシュブロック数を割り当てた場合には、下記のようにキャッシュブロック数と当該キャッシュブロック数の使用期間を乗算した値が大きくなれば、優先度を下げる、もしくはキャッシュ割当てブロック数を減らすといったスケジューリングを行うことができる。

30

P1: 64 × 1000 マイクロ秒 = 64,000 → 例: 優先度を下げる

P2: 21 × 500 マイクロ秒 = 10,500

P4: 11 × 2000 マイクロ秒 = 22,000

【0 0 4 3】

以上のように、本実施形態では、キャッシュメモリ領域をキャッシュブロック単位で任意に分割することができる。従って、共有キャッシュメモリをプロセッサが有する演算器等の演算リソースと同様にリソースとして管理し、プロセススケジューリングを最適化することが可能となり、プロセッサの実効性能を向上させることが可能となる。

【0 0 4 4】

図7および図8は、図1に示されるキャッシュメモリ1 0 1のブロック構成に対応するハードウェア構成例を示す図である。図7および図8において、図1の場合と同じ機能部分には同じ番号を付してある。

40

【0 0 4 5】

図1に示されるキャッシュブロック1 0 2は、例えば、データ部（キャッシュデータ部）とタグ部（キャッシュタグ部）が別々のR A M (R a n d o m A c c e s s M e m o r y)によって実装される。図7および図8の実装例では、キャッシュタグ部7 0 1において、各セット1 0 3を構成するキャッシュブロック1 0 2のタグ情報7 0 2としては、有効フラグ（1ビット）、タグ（15ビット）、P P I D（2ビット）が記憶される。また、各インデックス値毎の各P P I D値に対するM A X W A Y数1 0 5も、キャッシュタグ部7 0 1内に保持される。

50

【0046】

なお、タグ情報702とMAX WAY数105は、さらに別々のRAMに記憶されてもよい。

【0047】

図7において、メモリアクセス要求によりキャッシュアクセスが発生すると、#1～#4の各キャッシュウェイ104から、指定されたインデックス値上の各キャッシュブロック102（#i）のタグ値が読み出され、#1～#4のコンパレータ106に入力する。この結果、図1で説明したように、#1～#4のコンパレータ106のうち、要求元タグ値との一致を検出したコンパレータ106がタグ値を比較したキャッシュブロック102（#i）が、キャッシュヒットしたということになる。そして、キャッシュヒットが検出されたキャッシュブロック102（#i）に対してキャッシュデータ部（後述する図18の1804を参照）上のデータが読み書きされる。

10

【0048】

一方、図8において、メモリアクセス要求によりキャッシュアクセスが発生すると、#1～#4の各キャッシュウェイ104から、指定されたインデックス値上の各キャッシュブロック102（#i）のPPID値が読み出され、#1～#4のコンパレータ801に入力する。

【0049】

#1～#4のコンパレータ801は、読み出された各キャッシュブロック102（#i）内のPPID値と、要求元PPIDの値との一致／不一致を検出する。要求元PPIDは、キャッシュアクセス命令を実行しているプロセスのプロセスIDをプロセスIDマップユニット601（図6）で変換して得られる値である。この結果、キャッシュブロック102（#i）のPPID値が要求元PPIDの値と一致するウェイのコンパレータ801の出力は例えば1、一致しないウェイのコンパレータ801の出力は例えば0となる。

20

【0050】

従って、#1～#4のコンパレータ801は、キャッシュブロック102（#i）のPPID値が要求元PPIDの値と一致するウェイを示すビットマップを出力することになる。

【0051】

本実施形態では、このビットマップに含まれる1の数を数え上げることにより、キャッシュミスが発生したインデックス値上で、キャッシュミスを発生させたPPID値に関して既に配分済みのキャッシュウェイ数の合計を算出することができる。そして、前述したように、そのインデックス値上で、キャッシュミスを発生させたPPID値に関して既に配分済みのキャッシュウェイ数の合計と、そのPPID値に対応して記憶されているMAX WAY数105とが比較される。MAX WAY数105としては、図7または図8に示されるように、各インデックス毎に、図2または図3に示される各PPID値P1、P2、P3に対応する値が、キャッシュタグ部701に記憶されている。図2、図3には示していないが、P4についても同様である。そして、上記P1、P2、P3、P4などに対応するMAX WAY数105のうち、要求元PPIDに対応するMAX WAY数が、配分済みのキャッシュウェイ数の合計との比較処理の対象とされる。そして、配分済みのキャッシュウェイ数の合計がMAX WAY数105に満たない場合は、そのインデックス値上で、他のPPID値に対して配分済みのキャッシュブロック102の中で当該PPID値に対応するMAX WAY数105を越えているものの中から置換ブロックが選択されることになる。

30

40

【0052】

#1～#4のコンパレータ801が出力するビットマップに対して置換ブロックを決定するための置換ウェイ制御回路のハードウェア構成については、図11で後述する。

【0053】

図9は、OSが各PPID値に与えるキャッシュブロック数のテーブル（図2）に基づいて、各インデックス値毎に各PPID値に対するMAX WAY数105（図3）を決

50

定する処理を示す動作フローチャートである。この処理は例えば、図 7、図 8 を含むキャッシュシステムを制御するプロセッサ（例えば後述する CPU コア 1802）が実行する OS の処理の一部である。

【0054】

まず、図 2 のテーブル構成が参照され、最初のプロセスに割り当てるブロック数を、1 ウェイあたりのインデックス方向のブロック数で除算した値を C とする（ステップ S901）。すなわち、C は、キャッシュメモリ全体で当該プロセスに割り当てられるウェイ数である。

【0055】

次に、当該プロセスに割り当てるブロック数を 1 ウェイあたりのブロック数で割った余りの値を R とする（ステップ S902）。

例えば図 2 の最初の P P I D 値 P 1 のキャッシュブロック数は 64 である。また、図 3 において、1 ウェイあたりのインデックス方向のブロック数は 16 ブロックである。従って、 $C = 64 / 16 = 4$ 、その除算の余りは 0 であるから $R = 0$ となる。

【0056】

次に、すべてのインデックスについて MAX WAY 数 = C を設定する（ステップ S903）。上述の P P I D 値 P 1 の例では、MAX WAY 数 105 = 4 が設定される。

次に、初期値 0 から始まって前回の R の値を順次累算することで、R 個分の MAX WAY 数の増加処理を行う開始位置（MAX WAY 数増加開始位置）を更新する（ステップ S904）。続いて、MAX WAY 数増加開始位置から R 個のインデックス分だけ、MAX WAY 数 105 を 1 ずつ増加する（ステップ S905）。上述の P P I D 値 P 1 の例では、 $R = 0$ であるため、ステップ S905 の増加処理は実行されず、また、MAX WAY 数増加開始位置は、初期値 0 のままである。

【0057】

次に、 $C = 0$ であるか否かを判定する（ステップ S904）。

$C = 0$ でなくステップ S904 の判定が NO ならば、ステップ S908 に移行する。この結果、P P I D 値 P 1 に関する MAX WAY 数 105 は、図 3 に示されるように、すべてのインデックス値に対して 4 となる。

【0058】

ステップ S904 の判定の後、図 2 のテーブル構成例に対応するデータ構成を参照して、次のプロセスがあるか否かが判定される（ステップ S908）。

次のプロセスがありステップ S908 の判定が YES ならば、ステップ S901 からの処理を繰り返す。

【0059】

図 2 のテーブル構成例において、P P I D 値 P 1 の次にまだ P P I D 値 P 2 がある。このため、ステップ S901、S902 が再び実行される。図 2 の P P I D 値 P 2 のキャッシュブロック数は 21 であるため、 $C = 21 / 16 = 1$ 、その除算の余りは 5 であるから $R = 5$ となる。

さらに、ステップ S903 が実行される。P P I D 値 P 2 の例では、MAX WAY 数 105 = 1 が設定される。

【0060】

次に、ステップ S904 および S905 が実行される。P P I D 値 P 2 の例では、まず、MAX WAY 数増加開始位置は、前記アクセスの P 1 における $R = 0$ を使って、初期値 $0 + R = 0$ となる。そして、今回の $R = 5$ であるため、MAX WAY 数増加開始位置 = 0 から $R = 5$ 個分だけ MAX WAY 数 105 が +1 される。この結果、P P I D 値 P 2 に関する MAX WAY 数 105 は、図 3 に示されるように、最初の 5 インデックス値に対して 2、残りの 11 インデックス値に対して 1 となる。

【0061】

ステップ S905 の処理の後、ステップ S906 の判定が NO となって、ステップ S908 が判定される。図 2 のテーブル構成例において、P P I D 値 P 2 の次にまだ P P I D

10

20

30

40

50

値 P 3 がある。このため、ステップ S 9 0 8 の判定が Y E S となり、ステップ S 9 0 1、S 9 0 2 が再び実行される。図 2 の P P I D 値 P 3 のキャッシュブロック数は 1 1 であるため、 $C = 11 / 16 = 0$ 、その除算の余りは 1 1 であるから $R = 11$ となる。

【0062】

さらに、ステップ S 9 0 3 が実行される。P P I D 値 P 3 の例では、M A X W A Y 数 1 0 5 = 0 が設定される。

【0063】

次に、ステップ S 9 0 4 および S 9 0 5 が実行される。P P I D 値 P 3 の例では、まず、M A X W A Y 数増加開始位置は、前記アクセスの P 2 における $R = 5$ が累算されて 5 となる。そして、今回の $R = 11$ であるため、M A X W A Y 数増加開始位置 = 5 から $R = 11$ 個分だけ M A X W A Y 数 1 0 5 が + 1 される。この結果、P P I D 値 P 3 に関する M A X W A Y 数 1 0 5 は、図 3 に示されるように、最初の 5 インデックス値に対して 0、残りの 1 1 インデックス値に対して 1 となる。

【0064】

次に、 $C = 0$ であるからステップ S 9 0 6 の判定が Y E S となって、ステップ S 9 0 7 が実行される。

ここでは、P P I D 値 P 3 について、図 5 のアドレスハッシュユニット 5 0 1 を動作させるためのハッシュ有効化レジスタ（後述する図 1 3 の 1 3 0 2 の P 3 の行を参照）をセットする。

【0065】

ステップ S 9 0 7 の処理の後、図 2 のテーブル構成例において、P P I D 値 P 3 の次にはもう P P I D 値がない。このため、ステップ S 9 0 8 の判定が N O となって、図 9 のフローチャートによる M A X W A Y 数 1 0 5 の決定処理を終了する。なお、P P I D 値 P 4 がある場合には、さらに P 4 についても同様の処理が繰り返される。

【0066】

以上説明したフローチャートにより、O S が各 P P I D 値に与えるキャッシュブロック数のテーブル（図 2）に基づき、各インデックス値毎に各 P P I D 値に対する M A X W A Y 数 1 0 5（図 3）を適切に決定可能となる。

【0067】

図 1 0 は、図 9 のフローチャートの処理をプログラム処理として実行した場合のプログラム擬似コードである。各プログラムステップの左には、図 9 の対応する処理のステップ番号を付してある。

【0068】

まず、各変数 NP, NB, C, B, R, 0 が以下のように定義される。

NP : Number of Processes プロセス数

NB : Number of Blocks per way 1 ウェイあたりのブロック数

C[p] : プロセス p に割り当てるウェイト数

B[p] : プロセス p に割り当てるブロック数

R[p] : プロセス p において 1 ウェイ分に満たないブロック数

0[p] : M A X W A Y 数増加開始位置

【0069】

まず、図 2 のテーブル構成から参照される各プロセス p について、プロセス p に割り当てるブロック数 B[p] を、1 ウェイあたりのインデックス方向のブロック数で除算することにより、プロセス p に割り当てるウェイト数 C[p] を算出する（ステップ S 9 0 1）。

【0070】

次に、プロセス p に割り当てるブロック数 B[p] を、1 ウェイあたりのインデックス方向のブロック数で除算した余りとして、プロセス p において 1 ウェイ分に満たないブロック数 R[p] を算出する（ステップ S 9 0 2）。

【0071】

次に、M A X W A Y 数増加開始位置 0[p] = s とする（ステップ S 9 0 4）。また、s = s

10

20

30

40

50

+R[p]として更新する（ステップS905）。

次に、プロセスpについて、C[p]=0であるならば（ステップS906）、set_reg_hash_val(p)関数を呼び出し、図5のアドレスハッシュユニット501を動作させるためのハッシュ有効化レジスタ（後述する図13の1302を参照）をセットする（ステップS907）。

【0072】

以上の動作が図2のテーブル構成から参照される全プロセスについて実行される。この結果、各プロセスp毎に、プロセスpに割り当てるウェイ数C[p]、プロセスpにおいて1ウェイ分に満たないブロック数R[p]、およびMAX WAY数増加開始位置0[p]が算出される。

10

【0073】

これらの値を使って、まず、各プロセスp毎に、キャッシュタグ部701内のすべてのインデックスについて、MAX WAY数=C[p]を設定するSTORE命令（後述する図12参照）が実行される。

【0074】

次に、各プロセスp毎に、キャッシュタグ部701内のMAX WAY数増加開始位置からR[p]個のインデックス分だけ、MAX WAY数=C[p]+1を設定するSTORE命令（後述する図12参照）が実行される。

【0075】

以上のプログラム処理により、図9のフローチャートに対応するMAX WAY数105の決定処理が実行される。

20

図11は、図8の#1～#4のコンパレータ801が出力するビットマップに対して置換ブロックを決定するための置換ウェイ制御回路のハードウェア構成例を示す図である。置換ウェイ制御回路は、ビット数え上げ器1101と置換ウェイ候補決定回路1102と置換ウェイマスク生成回路1103とから構成される。

【0076】

PPIDマッチしたビットマスク1108は、図8の#1～#4のコンパレータ801の出力である。また、MAX WAY数105は、キャッシュタグ部701（図8参照）において、現在のキャッシュアクセスのインデクス値に対応して読み出される各PPID値に対応するMAX WAY数105である。

30

【0077】

まず、ビット数え上げ器1101は、ビットマスク1108のビットのうち1となっているビットを数え上げる。この結果、現在のキャッシュアクセスを発生させたPIDに対応するPPID（要求元PPID）に現在割り当てられているキャッシュウェイ数の合計が算出される。

【0078】

次に、選択回路1104が、各PPID値に対応するMAX WAY数105のうち、要求元PPIDに対応するMAX WAY数105を選択して出力する。

比較器1105は、ビット数え上げ器1101が出力する要求元PPIDに現在割り当てられているキャッシュウェイ数と、選択回路1104が出力する要求元PPIDに対応するMAX WAY数105とを比較する。

40

【0079】

比較器1105が比較した結果、要求元PPIDに現在割り当てられているキャッシュウェイ数の合計が要求元PPIDに対応するMAX WAY数105に満たない場合には、選択回路1107は次のように動作する。すなわち、選択回路1107は、ビットマスク1108の各ビットをインバータ1106で反転して得られるビットマスクを選択し、置換ウェイ候補を示すビットマスク1109として出力する。これにより、現在のキャッシュアクセスに対応するセット103上で、要求元PPID値以外の他のPPID値に対して配分済みのキャッシュブロック10が存在するウェイが、置換ウェイ候補とされる。

【0080】

50

一方、比較器 1105 が比較した結果、要求元 P P I D に現在割り当てられているキャッシュウェイ数の合計が要求元 P P I D に対応する MAX WAY 数 105 に達している場合には、選択回路 1107 は次のように動作する。すなわち、選択回路 1107 は、ビットマスク 1108 をそのまま選択し、置換ウェイ候補を示すビットマスク 1109 として出力する。これにより、現在のキャッシュアクセスに対応するセット 103 上で、要求元 P P I D 値に対して配分済みのキャッシュブロック 10 が存在するウェイが、置換ウェイ候補とされる。

【0081】

置換ウェイマスク生成回路 1103 は、置換ウェイ候補を示すビットマスク 1109 が示す置換ウェイ候補から、置換ウェイを選択し、置換ウェイを示す置換ウェイマスクを生成して出力する。より具体的には、ビットマスク 1109 が要求元 P P I D 以外の P P I D を置換ウェイ候補として示しているときには、置換ウェイマスク生成回路 1103 は、次のように動作する。すなわち、置換ウェイマスク生成回路 1103 は、キャッシュアクセスに対応するセット 103 上で、他の P P I D 値に対し配分済みのキャッシュブロック 102 の中で、配分済みのキャッシュウェイ数の合計が当該 P P I D 値に対応する MAX WAY 数 105 を越えているキャッシュブロックが選択される。そして、選択されたキャッシュブロックのウェイに対応するビット位置のみが 1 となる、4 ビットからなる置換ウェイマスクを生成する。ビットマスク 1109 が要求元 P P I D を置換ウェイ候補として示しているときには、置換ウェイマスク生成回路 1103 は、例えば L R U アルゴリズムによって、最も長い期間アクセスなされなかったウェイから選択された置換ウェイのみが 1 となる 4 ビットからなる置換ウェイマスクを生成する。

【0082】

キャッシュミスしたメモリアクセス要求に対応するデータはキャッシュデータ部に、また、タグおよび P P I D はキャッシュタグ部 701（図 7 参照）内の、置換ウェイマスクの 4 ビットのデータのうち値が 1 となるビット位置に対応するウェイに出力される。また、メモリアクセス要求内のインデックスが、キャッシュデータ部、キャッシュタグ部 701 のセット 103 を指定する。

【0083】

これにより、キャッシュデータ部およびキャッシュタグ部 701 において、指定されたセット 103 の選択されたウェイのキャッシュブロック 102 に、データ、タグ、および P P I D が書き込まれる。

【0084】

なお、キャッシュデータ部に書き込まれるデータは、メモリアクセス要求が読み出し要求の場合には、図示しないメインメモリ上の対応するアドレスから読み出されたデータである。また、メモリアクセス要求が書き込み要求の場合には、当該書き込み要求に指定されている書き込みデータである。

【0085】

図 12 は、各インデックス値の MAX WAY 数 105 を更新するための MAX WAY 数更新機構を示す実施例を説明する図である。

MAX WAY 保持部 1201 には、プロセッサの命令制御部（例えば後述する図 18 の 1806）からアドレスを指定して MAX WAY 数 105 の更新値を書き込むことができる。

【0086】

このとき、命令制御部は、MAX WAY 数 105 を更新するための STORE 命令が指定する物理アドレスは、例えば、52 ビットの物理アドレス空間を有するとする。

上記の STORE 命令が指定する物理アドレスは、MAX WAY 数保持部 1201 内のアドレスマップユニット 1202 によって、キャッシュのインデックス数に等しいアドレス空間を有する RAM 1203 上の該当する記憶領域をアクセス可能なアドレスとして、例えば「0x00C」に変換される。すなわちアドレスマップユニット 1202 は例えば、指定されたアドレス「0x1000000000000000C」から上位のアドレス情報「

0x1000000000」を削除して、アドレスを「0x00C」に変換する処理を実行する。そして、この変換されたアドレスによって指定されるRAM1203内の記憶領域、例えば「0x00C」に、STORE命令によって4バイトのデータ、例えば「0x04020101」が書き込まれる。そして例えば、この4バイトのデータのうち、最上位の1バイト「04」が、図2または図3に示されるPPID=P1に対応するMAXWAY数105=4を指定する。また、次の上位1バイト「02」が、同じくPPID=P2に対応するMAXWAY数105=2を指定する。同様に、次の上位1バイト「01」が、同じくPPID=P3に対応するMAXWAY数105=1を指定する。そして、最下位の1バイト「01」が、図2や図3では図示しないが、PPID=P4に対応するMAXWAY数105=1を指定する。この1つのSTORE命令によって書き込まれる4バイト1組のデータが、図7または図8に示される1つのインデックス値上のP1~P4に対応する1組のMAXWAY数105となる。

10

【0087】

このように、RAM1203上のデータは、4バイトを1組として管理されるため、RAM1203を更新するために命令制御部によって指定される物理アドレスは、4バイトおきに指定されることになる。例えば、「0x1000000000000000」の次は「0x1000000000000004」のごとくである。

【0088】

なお、図8等で前述したように、キャッシュアクセス時には、例えば、メモリアクセスのためのアドレス107内のインデックスの値によって、キャッシュタグ部701が、キャッシュメモリ101に含まれるRAM1203上の該当する記憶領域にアクセスする。

20

【0089】

前述したように、キャッシュメモリ101のPPID値毎の容量割当てを変更する場合は、MAXWAY数105を保持するキャッシュタグ部701内のRAM1203内での各インデックス値毎のMAXWAY数105の割当てを変更すればよい。この場合に、上述のSTORE命令によるMAXWAY数105の更新指示は、キャッシュアクセス命令に付随させて行ってもよいし、全インデックス値について一括して実行されてもよい。

【0090】

以上説明した図12のMAXWAY数更新処理は、例えば後述する図18に示されるキャッシュシステム1801内のキャッシュメモリ制御部1805が、CPUコア1802内の命令制御部1806からの指示に従って実行する。

30

【0091】

図13は、図5のアドレスハッシュユニット501のハードウェア構成例を示す図である。

ハッシュ有効化レジスタ1302は、PPID値毎に、有効ビット、インデックス数、およびオフセットインデックス数を記憶する。有効ビットとしては、例えば、ハッシュ処理を実行する場合に有効を示す値1を、実行しない場合に無効を示す値0がセットされる。インデックス数としては、1ウェイ分に満たない分のインデックス増加処理を行うブロック数R[p]がセットされる。オフセットインデックス数としては、上記増加処理の実行を開始するインデックス位置=MAXWAY数増加開始位置O[p]がセットされる。

40

【0092】

図9及び図10で前述したように、プロセスpについて、C[p]=0であるならば、set_reg_hashval(p)関数が呼び出されて、ハッシュ有効化レジスタ1302へのセットが実行される。

【0093】

次に、図13において、選択回路1303は、ハッシュ有効化レジスタ1302上の要求元PPID値と一致するPPID値に対応するエントリから、有効ビット、インデックス数、およびオフセットインデックス数を読み出して、モジュロ演算器1301に与える。要求元PPID値は、キャッシュアクセス命令を実行しているプロセスのプロセスID

50

を、プロセスIDマップユニット601（図6）で変換して得られる値である。

【0094】

モジュロ演算器1301には、選択回路1303から要求元PPIDに対応する有効ビット、インデックス数、オフセットインデックス数が入力するほか、キャッシュアクセス命令によって指定されるアドレス107の上位ビット部分が入力する。

【0095】

モジュロ演算器1301は、有効ビットがセットされているアドレス107の上位ビット部分をインデックス数で割った余りにオフセットインデックス数を加えた値を計算する。計算結果は、キャッシュタグ部701（図7）およびキャッシュデータ部（後述する図18の1804を参照）に、新たなインデックスとして出力される。

10

【0096】

モジュロ演算器1301は、もし有効ビットがセットされていなければ、アドレス107のインデックスをそのまま、キャッシュタグ部701（図7）およびキャッシュデータ部（後述する図18の1804を参照）に、新たなインデックスとして出力する。

【0097】

以上の構成を有するアドレスハッシュユニット501の具体的な動作について、図14および図15の動作説明図と、前述した図2、図3を用いて説明する。

ここで、図7および図8に示されるキャッシュタグ部701のハードウェア構成においては、キャッシュタグ部701の具体的なサイズは例えば次のようになる。すなわち、プログラムで指定される32ビットのアドレス107において、下位7ビットでキャッシュライン内オフセットが指定され、その上位10ビットでインデックス、さらにその上位15ビットでタグが指定される例が示されている。従って、この例の場合は、10ビットのインデックスによって指定されるセット103のライン数 n は2の10乗＝1024であるが、キャッシュタグ部701のサイズはこれに限定されるものではなく、システムごとに適切なその他のサイズ値を採用することができる。システムごとに適切なその他のサイズ値を採用する場合には、アドレス107も適切なビット幅を採用できる。

20

【0098】

図14および図15では、理解を容易にするために、アドレス107が16ビット、キャッシュライン内オフセットが7ビット、インデックスが4ビット、タグが5ビットである場合の例について説明する。この例の場合、セット103のライン数 n は、図3のインデックス方向の行数として示されるように、2の4乗＝16となる。

30

【0099】

図13のハッシュ有効化レジスタ1302において、図3に示されているPPID値がP1、P2、P3と、これらP1、P2、P3以外のP-OTHERSの場合に、 $C=0$ となるケースは、PPID値＝P3のときで、全ブロック数はインデックス方向のインデックス数16に満たない。このためP3のインデックス数としては、1ウェイ分に満たない分のブロック数 $R[P3]=5$ （図10参照）がセットされる。オフセットインデックス数としては、上記増加処理の実行を開始するインデックス位置＝MAXWAY数増加開始位置 $O[p]$ がセットされる。例えば図3において、P3のケースの場合、 $R[P2]=5$ 、すなわち $C=0$ となる直前のプロセスP2において図9のS902で計算された当該プロセスP2に割り当てるブロック数15を1ウェイあたりのブロック数10で割った余り $R[P2]=5$ に等しい値5が、 $O[P3]$ としてセットされる。

40

【0100】

図9及び図10で前述したように、プロセス p について、 $C[p]=0$ であるならば、`set_reg_hashval(p)`関数が呼び出されて、ハッシュ有効化レジスタ1302へのセットが実行される。

【0101】

すなわち、PPID値＝P3について、 $C[P3]=0$ となるため、ハッシュ有効化レジスタ1302のP3に対応するエントリに、以下の値がセットされる。すなわち、図14に示されるように、有効ビット＝1、インデックス数＝ $R[P3]=11$ 、オフセット

50

インデックス数 = $R[P2] = 5$ がセットされる。その他の P P I D 値 $P1$, $P2$ については、 $C[p] = 0$ とならないため、ハッシュ有効化レジスタ 1302 の各 P P I D 値 $P1$, $P2$ に対応するエントリの内容は、図 14 に示されるように、共に各値が 0 のクリア状態となる。

【0102】

ここで、図 14 に示されるように、要求元 P P I D 値として「3」が入力したとする。この結果、選択回路 1303 が、ハッシュ有効化レジスタ 1302 上の要求元 P P I D 値と一致する P P I D 値 = $P3$ に対応するエントリから、有効ビット = 1、インデックス数 = 11、およびオフセットインデックス数 = 5 を読み出す。そして、選択回路 1303 は、それらの数値データをモジュロ演算器 1301 に与える。モジュロ演算器 1301 は、上述のようにもし有効ビットが 1 にセットされていれば、アドレス 107 のタグ + インデックスの上位 9 ビットのビット値をインデックス数 = 11 で割った余りにオフセットインデックス数 = 5 を加算し、その加算結果を新たなインデックスとして出力する。

10

【0103】

ここで例えば、要求元 P P I D 値 = 3 で、アドレス 107 として、下記のアドレスがそれぞれ入力した場合を考える。

0 x D 1 5 2
0 x D 1 D 2
0 x D 2 5 2
0 x D 2 D 2
0 x D 3 5 2
0 x D 3 D 2
0 x D 4 5 2
0 x D 4 D 2
0 x D 5 5 2
0 x D 5 D 2
0 x D 6 5 2
0 x D 6 D 2
0 x D 7 5 2

20

【0104】

図 14 では、アドレス 107 として「0 x D 5 5 2」が入力された場合が示されている。

30

これらの場合、上位 9 ビットのビット値および各々に対応する 10 進値は、それぞれ以下になる。

1 1 0 1 0 0 0 1 0 = 4 1 8
1 1 0 1 0 0 0 1 1 = 4 1 9
1 1 0 1 0 0 1 0 0 = 4 2 0
1 1 0 1 0 0 1 0 1 = 4 2 1
1 1 0 1 0 0 1 1 0 = 4 2 2
1 1 0 1 0 0 1 1 1 = 4 2 3
1 1 0 1 0 1 0 0 0 = 4 2 4
1 1 0 1 0 1 0 0 1 = 4 2 5
1 1 0 1 0 1 0 1 0 = 4 2 6
1 1 0 1 0 1 0 1 1 = 4 2 7
1 1 0 1 0 1 1 0 0 = 4 2 8
1 1 0 1 0 1 1 0 1 = 4 2 9
1 1 0 1 0 1 1 1 0 = 4 3 0

40

【0105】

図 14 では、アドレス 107 「0 x D 5 5 2」における上位ビット 9 ビットが、「1 1 0 1 0 1 0 1 0」で、10 進表現は「4 2 6」であることが示されている。

50

【0106】

モジュロ演算器1301は、上記各上位9ビット値に対して、それぞれ以下のようにして、インデックス数=11で割った余りにオフセットインデックス数=5を加算し、その加算結果を新たなインデックスとして出力する。

$418 \div 11 = 38$ 余り0、余り0+オフセットインデックス数5=5
 $419 \div 11 = 38$ 余り1、余り1+オフセットインデックス数5=6
 $420 \div 11 = 38$ 余り2、余り2+オフセットインデックス数5=7
 $421 \div 11 = 38$ 余り3、余り3+オフセットインデックス数5=8
 $422 \div 11 = 38$ 余り4、余り4+オフセットインデックス数5=9
 $423 \div 11 = 38$ 余り5、余り5+オフセットインデックス数5=10
 $424 \div 11 = 38$ 余り6、余り6+オフセットインデックス数5=11
 $425 \div 11 = 38$ 余り7、余り7+オフセットインデックス数5=12
 $426 \div 11 = 38$ 余り8、余り8+オフセットインデックス数5=13
 $427 \div 11 = 38$ 余り9、余り9+オフセットインデックス数5=14
 $428 \div 11 = 38$ 余り10、余り10+オフセットインデックス数5=15
 $429 \div 11 = 39$ 余り0、余り0+オフセットインデックス数5=5
 $430 \div 11 = 39$ 余り1、余り1+オフセットインデックス数5=6

10

【0107】

図14では、上位ビット9ビット値=110101010（10進値=426）をインデックス数11で割った余りが8であり、その余りにオフセットインデックス数5を加算することにより、新たなインデックスの値13が得られることが示されている。

20

【0108】

以上の具体例により、図3のP3の11個のブロックを順にアクセスすることができることがわかる。すなわち、新たなインデックスの値は、0～15までの全インデックス範囲のうち、5から15までの範囲（P3）に収まる。すなわち、PPID値P3に対応する命令が実行されるときに、アドレス107のインデックスは、図3のインデックス方向の全域で指定される可能性がある。これに対して、モジュロ演算器1301は、インデックスの値が5から15までの11個のインデックス範囲のみが指定されるように、マッピングを行うことができる。

【0109】

30

一方、図15に示されるように、要求元PPID値として「1」（または「2」）が入力したとする。この結果、選択回路1303が、ハッシュ有効化レジスタ1302上の要求元PPID値と一致するPPID値=P1（またはP2）に対応するエントリから、有効ビット=0、インデックス数=0、およびオフセットインデックス数=0を読み出す。そして、選択回路1303は、それらの数値データをモジュロ演算器1301に与える。モジュロ演算器1301は、上述のようにもし有効ビットが1にセットされていなければ、以下のように動作する。すなわち、アドレス107中の4ビットのインデックスをそのまま、キャッシュタグ部701（図7）およびキャッシュデータ部（後述する図16の1604を参照）に、新たなインデックスとして出力する。

【0110】

40

ここで例えば、要求元PPID値=1で、アドレス107として、前述と同様の「0xD152」から「0xD752」までのアドレスが入力したとする。

図15では、アドレス107として「0xD552」が入力された場合が示されている。

【0111】

これらの場合、アドレス107中のインデックスおよび各々に対応する10進値は、それぞれ以下ようになる。

$0010 = 2$
 $0011 = 3$
 $0100 = 4$

50

0 1 0 1 = 5
 0 1 1 0 = 6
 0 1 1 1 = 7
 1 0 0 0 = 8
 1 0 0 1 = 9
 1 0 1 0 = 1 0
 1 0 1 1 = 1 1
 1 1 0 0 = 1 2
 1 1 0 1 = 1 3
 1 1 1 0 = 1 4

10

モジュロ演算器 1 3 0 1 は、上記各 4 ビットのインデックスをそのまま、新たなインデックスとして出力する。

【0 1 1 2】

図 1 5 では、アドレス 1 0 7 中のインデックス「1 0 1 0」（1 0 進数で 1 0）がそのまま、新たなインデックスとして出力されることが示されている。

以上の具体例により、図 3 の P P I D 値 = P 1 または P 2 については、0 ~ 1 5 までの全インデックス範囲をインデックスとして指定できることがわかる。

【0 1 1 3】

このようにして、或るプロセス p について、図 2 のテーブルによって指定されるブロック数が 1 ウェイ分に満たない場合には、次のような制御が実行される。すなわち、MAX WAY 数増加開始位置 O [P] から、プロセス p について割り当て可能な 1 ウェイ分に満たないブロック数 R [p] に対応するインデックス範囲でのみインデックス指定がされるように、新たなインデックスがマッピングされる。

20

【0 1 1 4】

ここで、図 9 または図 1 0 のステップ S 9 0 7 によってハッシュ有効化レジスタ 1 3 0 2 の内容を更新するときには、次のようなアドレス指定を行うことができる。すなわち、図 1 2 の MAX WAY 数 1 0 5 の更新処理の場合と同様に、メインメモリ等へのメモリアクセス時には使用されないような特定のアドレス空間にマップされた領域を介して、ハッシュ有効化レジスタ 1 3 0 2 に対する読み書きを実行できる。

【0 1 1 5】

以上説明した図 1 3 のアドレスハッシュユニット 5 0 1 の構成により、指定された命令アドレス 1 0 7 のインデックスをハッシュして得られるインデックスが、禁止された領域のインデックスを生成しないように制御することが可能となる。

30

【0 1 1 6】

図 1 6 は、図 6 のプロセス I D マップユニット 6 0 1 のハードウェア構成例を示す図である。

プロセス I D マップユニット 6 0 1 は、OS が管理する P I D と、キャッシュメモリ 1 0 1 のハードウェアが取扱い可能な物理プロセス I D である P P I D の変換を行う。

【0 1 1 7】

プロセス I D マップユニット 6 0 1 は、変換のマップを格納するとともに検索が可能な連想メモリ 1 6 0 1 で構成されている。なお、プロセス I D マップユニット 6 0 1 は、レジスタで構成されてもよい。要求元 P I D の値をキーとして連想メモリ 1 6 0 1 の検索を行い、マッチした P P I D の値を出力する。

40

【0 1 1 8】

連想メモリ 1 6 0 1 に格納する値は、図 1 2 の MAX WAY 数 1 0 5 の更新処理の場合と同様に、メインメモリ等へのメモリアクセス時には使用されないような特定のアドレス空間にマップされた領域を介して読み書きできる。

【0 1 1 9】

図 1 7 は、P P I D 書込み機構を示す図である。

図 1 6 に示されるプロセス I D マップユニット 6 0 1 から出力される要求元 P P I D の

50

値で、キャッシュタグ部 701 (図 7) 内のキャッシュブロック 102 の更新が行われる。このときキャッシュブロック 102 をアクセスするインデックスは、図 13 に示されるアドレスハッシュユニット 501 から出力された値を用いる。

【0120】

図 18 は、本実施形態のキャッシュメモリシステムを備える演算処理装置としてのプロセッサの構成例を示す図である。

キャッシュシステム 1801 は、図 7 に示したキャッシュタグ部 701 (MAX WAY 数保持部 1201 を含む)、図 5 および図 13 に示したアドレスハッシュユニット 501、図 6 および図 16 に示したプロセス ID マップユニット 601 を備える。また、キャッシュシステム 1801 は、キャッシュデータを保持するキャッシュデータ部 1804 およびキャッシュタグ部 701 およびキャッシュデータ部 1804 へのキャッシュアクセスを制御するキャッシュメモリ制御部 1805 を備える。

10

【0121】

キャッシュメモリ制御部 1805 は、#1 ~ #4 の CPU コア 1802 内の命令制御部 1806 から発行されるメモリアクセス命令をデコードして、メインメモリ 1803 に対するアクセスがキャッシュデータ部 1804 へのアクセスであるかを判断する。

【0122】

キャッシュメモリ制御部 1805 は、デコードの結果、メモリアクセス命令がキャッシュデータ部 1804 へのアクセスである場合には、キャッシュタグ部 701 およびキャッシュデータ部 1804 に対して、メモリアクセス命令に含まれるアドレス 107 (図 1、図 7 等参照) を発行する。このアドレス 107 は、アドレスハッシュユニット 501 で処理された後に、キャッシュタグ部 701 およびキャッシュデータ部 1804 に出力する。

20

【0123】

また、キャッシュメモリ制御部 1805 は、メモリアクセス命令がキャッシュデータ部 1804 へのアクセスである場合には、メモリアクセス命令が実行される PID をプロセス ID マップユニット 601 に出力する。プロセス ID マップユニット 601 は、PID を P PID に変換し、キャッシュタグ部 701 に要求元 P PID として出力する。

【0124】

キャッシュメモリ制御部 1805 は、図 11 および図 12 に示したハードウェア機構を含み、前述した置換ウェイの制御や MAX WAY 数 105 の更新制御等を実行する。

30

キャッシュシステム 1801 においてキャッシュミスが発生した場合には、メインメモリ 1803 からデータが読み出されるとともに、キャッシュメモリ制御部 1805 内の図 11 のハードウェア構成によって生成される置換ウェイマスクに対応する置換ウェイのキャッシュブロック 102 に、そのデータが記憶される。これにより、次回アクセス時にキャッシュヒットとなり、高速なアクセスが実行される。

【0125】

また、キャッシュメモリ制御部 1805 は、命令制御部 1806 から、MAX WAY 数 105 の更新を指示する STORE 命令が発行されている場合には (図 12 参照)、次のような動作を実行する。すなわち、キャッシュメモリ制御部 1805 は、MAX WAY 数 105 を保持するキャッシュタグ部 701 内の RAM 1203 (図 12) 内の、上記 STORE 命令で指定される物理アドレスに対して、STORE 命令で指定される 4 バイトのデータを書き込む。これにより、該当するインデックス値上の各 P PID 値 (P1, P2, P3, P4) 毎の MAX WAY 数 105 が更新される。STORE 命令による MAX WAY 数 105 の更新指示は、命令制御部 1806 からの指示により、キャッシュアクセスを発生させるメモリアクセス命令によるメモリアクセスがなされた場合に付随させて行ってもよいし、全インデックス値について一括して実行されてもよい。

40

【0126】

図 19 は、本実施形態において、同時にスケジューリングされる各プロセスが要求するウェイ数の合計が、実装されているキャッシュメモリのウェイ数を超過している場合の動作例を示す説明図である。

50

【0127】

この動作例において、まず、P P I D 値 P 1、P P I D 値 P 2、P P I D 値 P 3 の各 P P I D 値に対する M A X ウェイ数の設定値を例えば、5、5、3 とする。

まず、P P I D 値 P 3 のプロセスに含まれる L O A D 命令実行によりキャッシュミスが発生する（ステップ S 1 7 0 1）。P 3 のブロック数 = 1 は、P 3 の M A X W A Y 数 = 3 よりも小さいため、他の P P I D 値のウェイ、図 1 9 の例では P P I D 値 P 2 のウェイを置換する。

【0128】

さらに、P P I D 値 P 3 のプロセスに含まれる L O A D 命令実行によりキャッシュミスが発生する（ステップ S 1 7 0 2）。P 3 のブロック数 = 2 は、P 3 の M A X W A Y 数 = 3 よりも小さいため、さらに他の P P I D 値のウェイ、図 1 9 の例では P P I D 値 P 1 のウェイを置換する。

【0129】

このようにして、P P I D 値 P 3 に割り当てられているブロック数は、最初 1 ブロックしかないが、P P I D 値 P 3 のプロセスに含まれるメモリアクセス要求があると M A X W A Y 数 = 3 まで他の P P I D のブロックを置換することにより増加する。

【0130】

さらに、P P I D 値 P 3 のプロセスに含まれる L O A D 命令実行によりキャッシュミスが発生したものとする（ステップ S 1 7 0 3）。P 3 のブロック数 = 3 は、P 3 の M A X W A Y 数 = 3 以下であるため、自 P P I D である P P I D 値 P 3 に対応するウェイを置換する。

【0131】

このように、M A X W A Y 数以上の P P I D 値 P 3 の要求があっても、P P I D 値 P 3 に対するキャッシュブロック数は M A X W A Y 数以上には増えない。

次に、P P I D 値 P 2 のプロセスに含まれる L O A D 命令実行によりキャッシュミスが発生したものとする（ステップ S 1 7 0 4）。P 2 のブロック数 = 1 は、P 2 の M A X W A Y 数 = 5 よりも小さいため、P P I D 値 P 1 のウェイを置換する。

【0132】

その後、P P I D 値 P 1 のプロセスに含まれるメモリアクセス要求があり、同様にして M A X W A Y 数 = 5 まで増加する（ステップ S 1 7 0 5、S 1 7 0 6・・・）。このように、M A X W A Y 数に近づくように各 P P I D 値に対応するブロック数が変化することで、実装されているウェイ数を超える M A X W A Y 数を設定した場合でも問題なくキャッシュ分割を行うことが可能となる。

【0133】

図 2 0 は、時間と優先度でキャッシュブロックをスケジュールする動作を示すフローチャートである。

このフローチャートの処理は、一定時間（例えば 1 0 マイクロ秒）毎に実行される。

【0134】

まず、キャッシュブロックを割り当てている各プロセス毎に、キャッシュブロック割当て数 [blocks] とプロセス割当て時間 [us] の積 A を計算する（ステップ S 2 0 0 1）。

次に、 $A > T$ となるプロセスが存在するか否かが判定される（ステップ S 2 0 0 2）。ここで、T は、システム依存の定数（しきい値）とする。

【0135】

$A > T$ となるプロセスが存在しステップ S 2 0 0 2 の判定が Y E S ならば、プロセス実行優先度が下げられて（ステップ S 2 0 0 3）、今回の処理を終了する。

$A > T$ となるプロセスが存在せずステップ S 2 0 0 2 の判定が N O ならば、何もせずに今回の処理を終了する。

【0136】

上述した実施形態において、M A X W A Y 数は、キャッシュタグ部内に設けるようにしたが、O S の管理下で制御されるような構成が採用されてもよい。

10

20

30

40

50

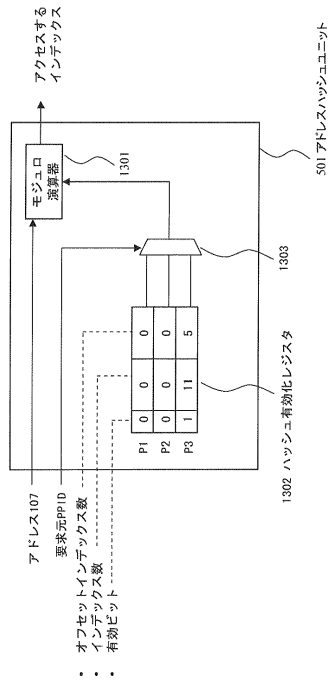
【符号の説明】

【0137】

101	キャッシュメモリ	
102	キャッシュブロック	
103	キャッシュライン	
104	キャッシュウェイ	
105	MAX WAY数	
106, 801	コンパレータ	
107	アドレス	
501	アドレスハッシュユニット	10
601	プロセスIDマップユニット	
701	キャッシュタグ部	
702	タグ情報	
1101	ビット数え上げ器	
1102	置換ウェイ候補決定回路	
1103	置換ウェイマスク生成回路	
1104, 1107, 1303	選択回路	
1105	比較器	
1106	インバータ	
1108	PPIDマッチしたビットマスク	20
1109	置換ウェイ候補を示すビットマスク	
1201	MAX WAY数105保持部	
1202	アドレスマップユニット	
1203	RAM	
1301	モジュロ演算器	
1302	ハッシュ有効化レジスタ	
1601	連想メモリ	
1801	キャッシュシステム	
1802	CPUコア	
1803	メインメモリ	30
1804	キャッシュデータ部	
1805	キャッシュメモリ制御部	
1806	命令制御部	

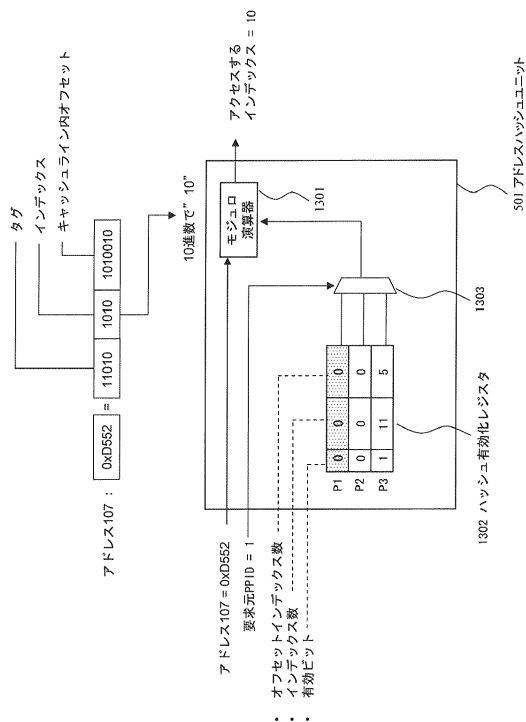
【図 13】

ハッシュユニットのハードウェア構成例を示す図



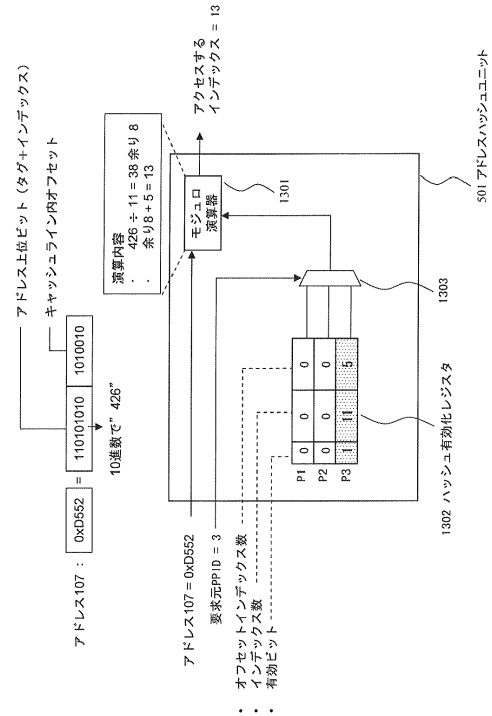
【図 15】

ハッシュユニットの動作説明図(その2)

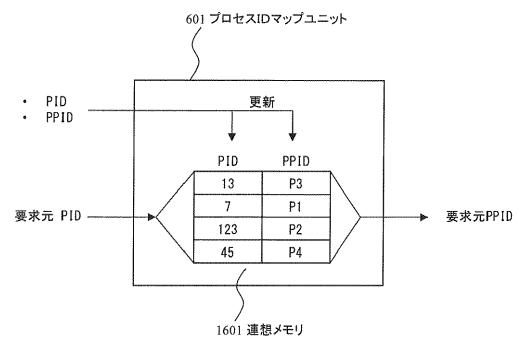


【図 14】

ハッシュユニットの動作説明図(その1)

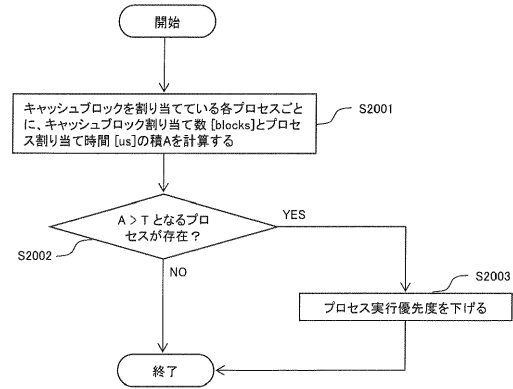


【図 16】

プロセスIDマップユニットの
ハードウェア構成例を示す図

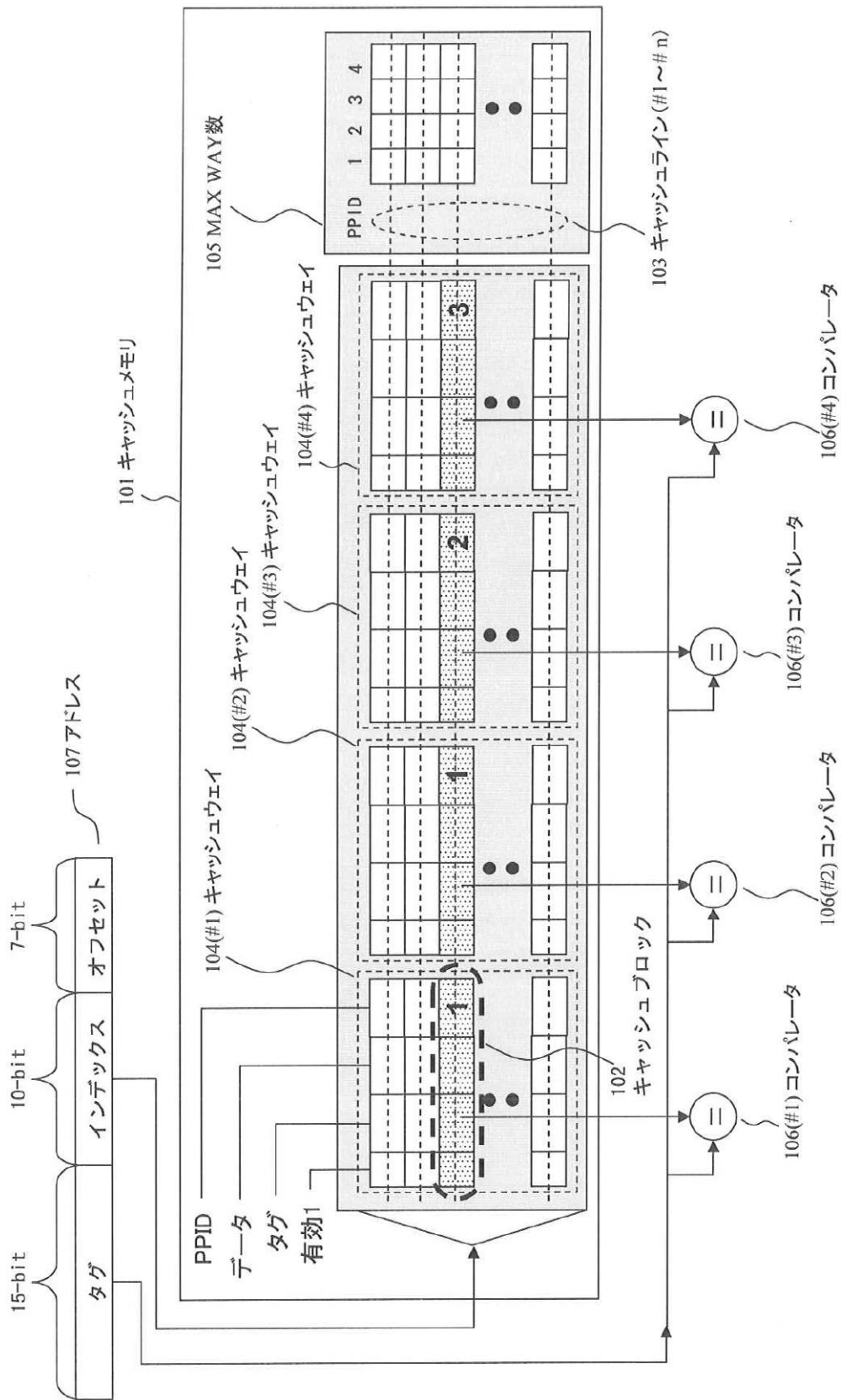
【図 20】

時間と優先度でキャッシュブロックを
スケジュールする動作を示すフローチャート



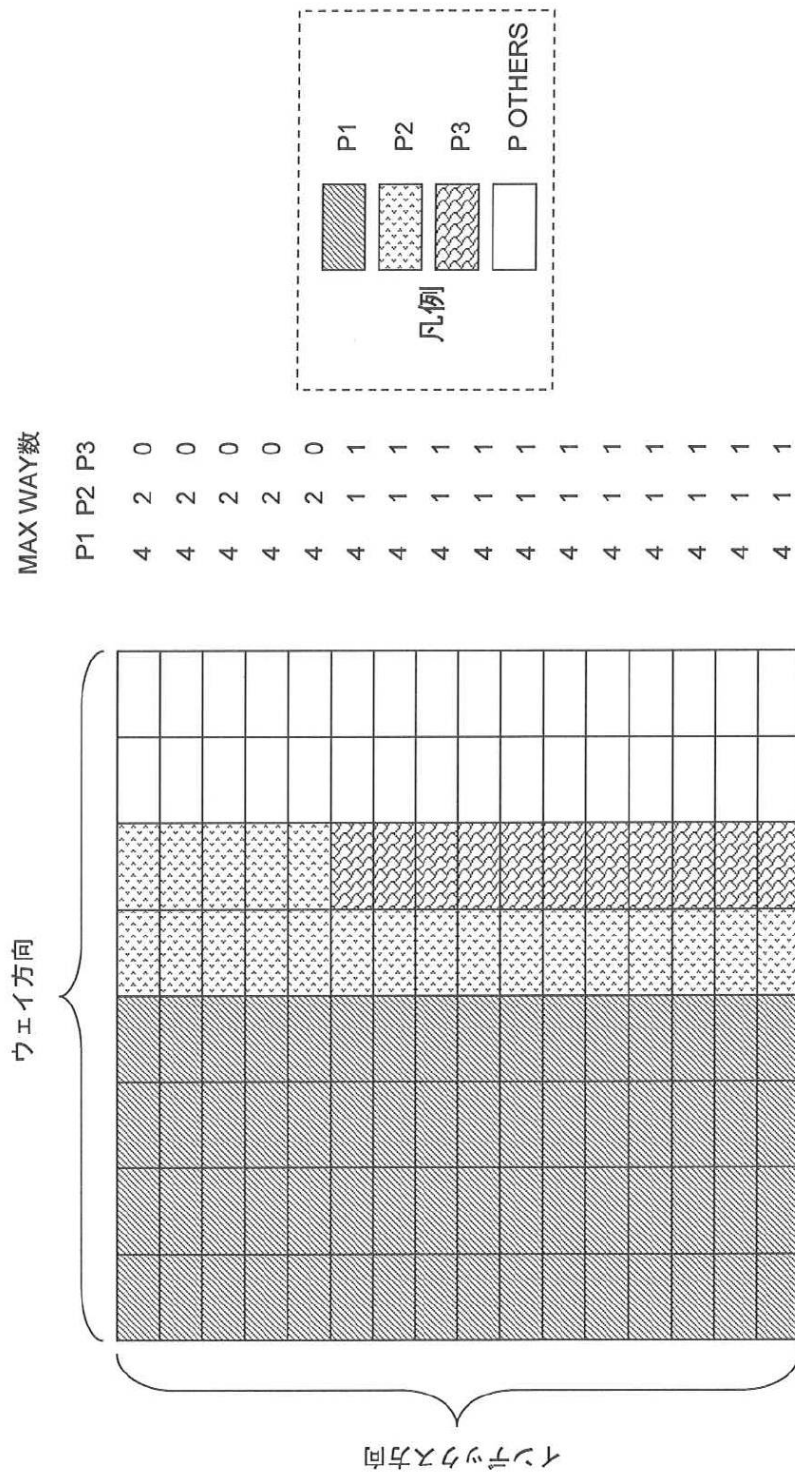
【図 1】

キャッシュメモリの実施形態のブロック図



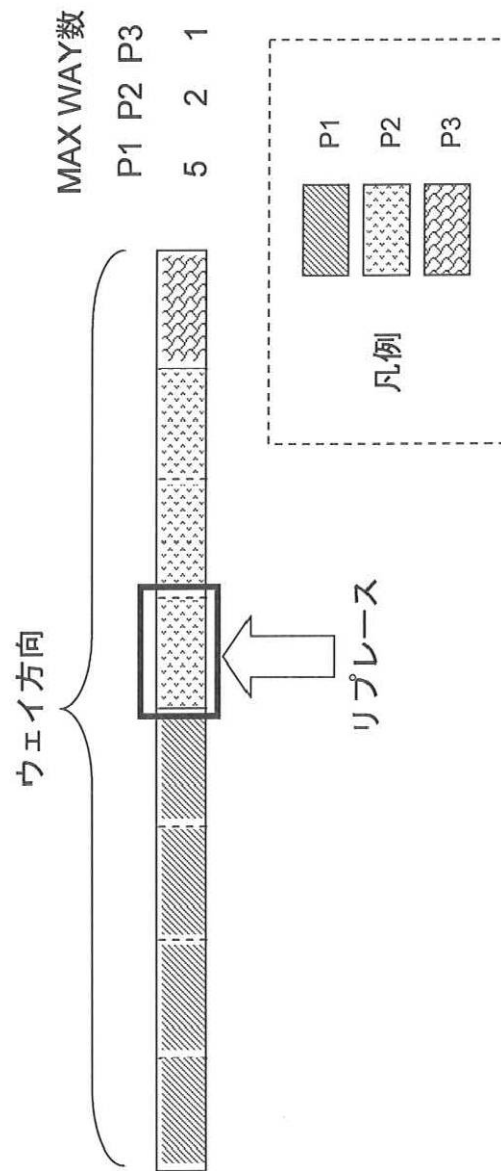
【図 3】

キャッシュメモリの分割例を示す図



【図 4】

キャッシュミス発生時の リプレイス動作を示す説明図



【図 5】

ハッシュユニットを示す図



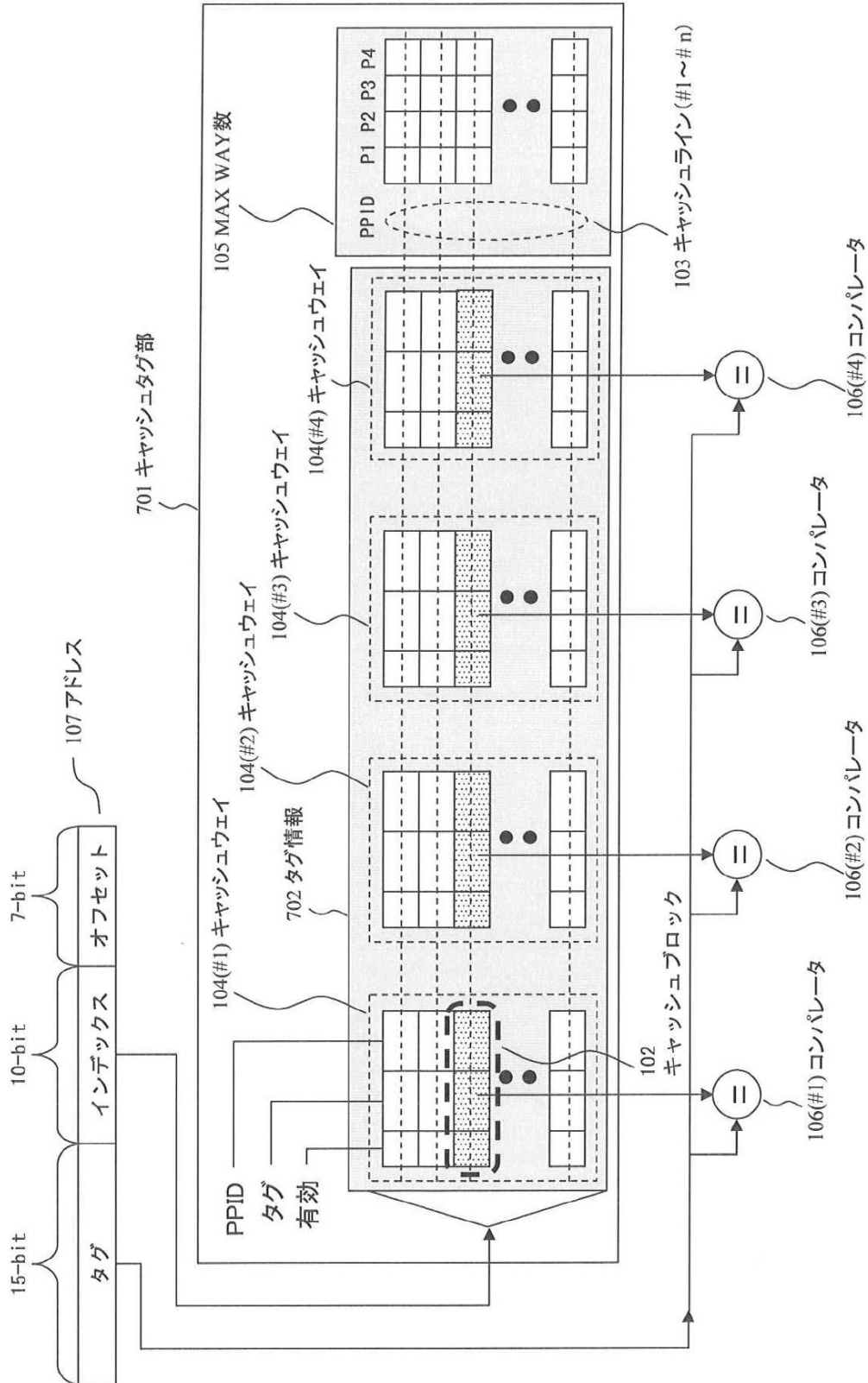
【図 6】

プロセスIDマップユニットを示す図

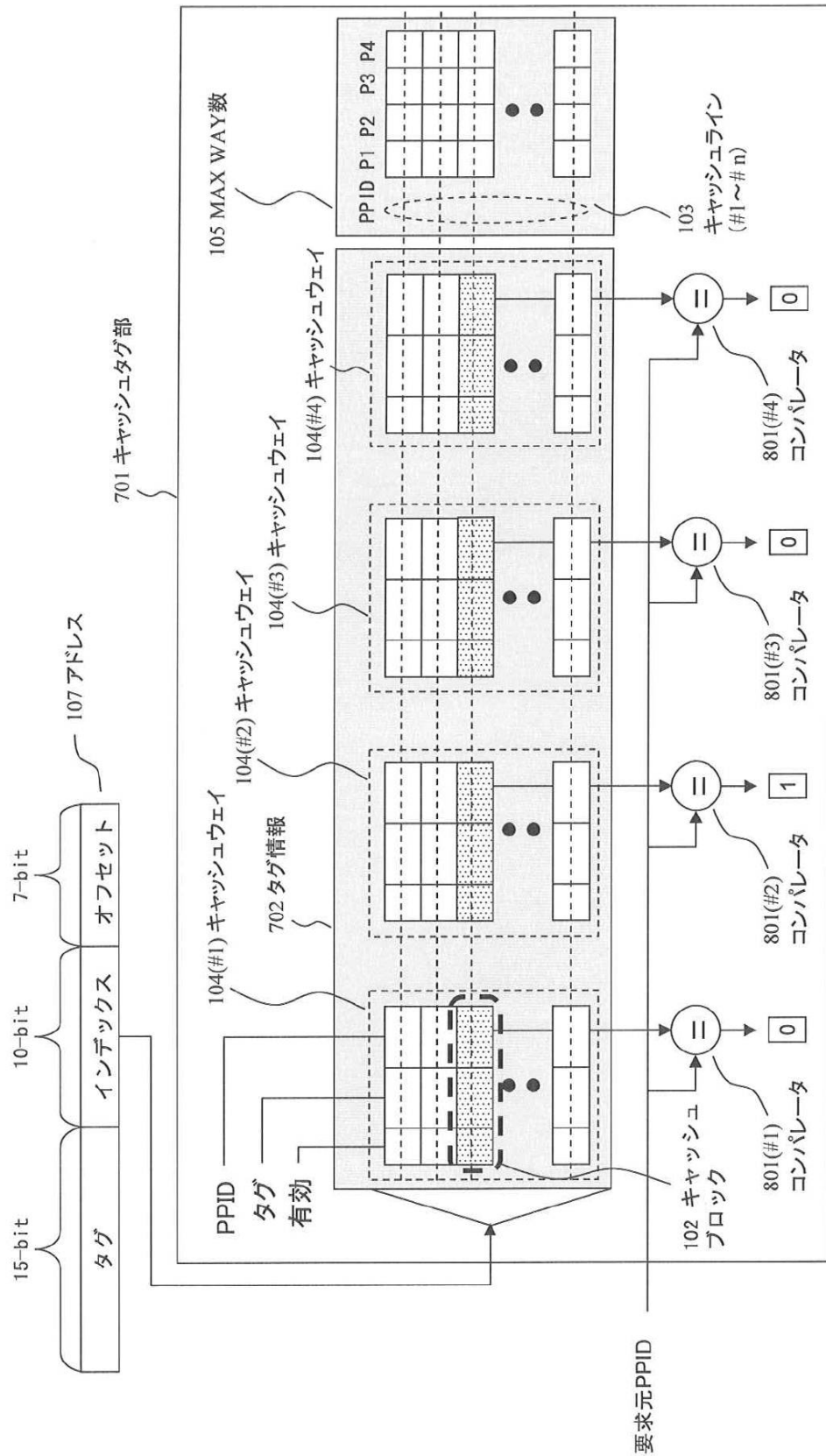


【図 7】

キャッシュタグ部のハードウェア構成例を示す図(その1)

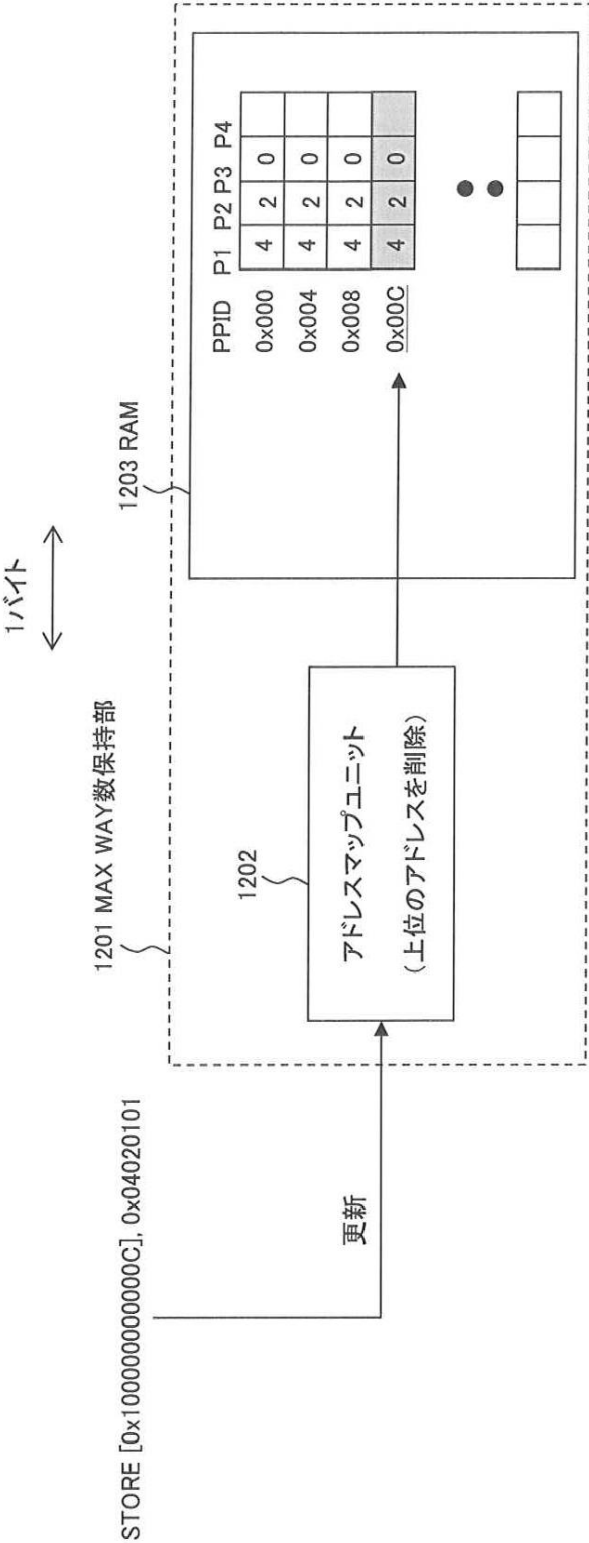


キャッシュタグ部のハードウェア構成例を示す図(その2)



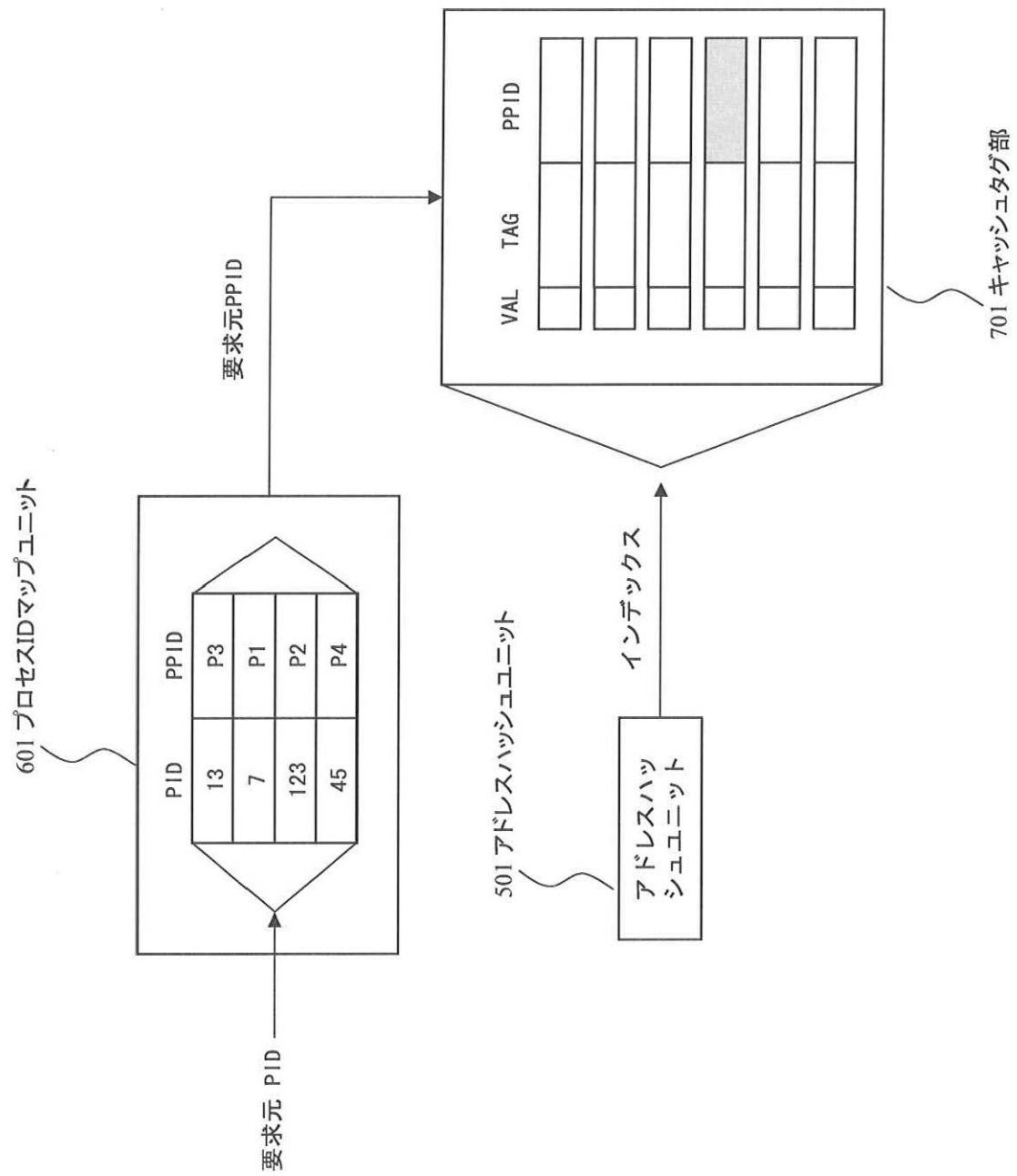
【図 1 2】

MAX WAY数更新機構を示す図



【図 17】

PPID書込み機構を示す図



同時にスケジューリングされる各プロセスが要求する
ウェイ数の合計が、実装されているキャッシュメモリの
ウェイ数を超えている場合の動作例を示す説明図

