修士論文

省電力キャッシュメモリにおける スラックを用いたデータ割りあて手法

電気通信大学　大学院情報システム学研究科
情報システム基盤学専攻

0753001　　板倉 佑輔

指導教員　　本多弘樹
古賀久志
近藤正章

2009年 1月 29日 提出

目 次

図 目 次

表 目 次

1 はじめに

1.1 研究背景

近年，マイクロプロセッサの消費電力増大が問題となっている．消費電力の増大はバッテリー駆動型モバイル機器の駆動時間に大きな影響を及ぼし，バッテリー駆動型モバイル機器の特徴である携帯性が損なわれてしまう．また，電力を消費することによる発熱は機器へ負荷を与え機器の寿命を早めてしまう．特に，機器内部が高密度化されているものであれば熱を放出することは難しく機器への負担が大きくなる．発熱による機器への負荷を減らすために冷却装置を備えているものもあるが，電力を消費することによる発熱が与える機器への負荷を緩和できても冷却装置に消費電力がかかり，結果的に機器全体の消費電力量は増えてしまう．

現在では，家庭の PC でも一昔前のスーパーコンピュータ並の性能が達成されており，プロセッサの性能向上は著しい．プロセッサの性能が向上したのは，半導体技術と製造技術の向上によりプロセスルールの細微化とトランジスタの高集積化が可能となって動作周波数を上げることができた結果である．

マイクロプロセッサの消費電力はトランジスタのスイッチングに要する動的消費電力とリーク電流による静的消費電力に分けられる．近年では，プロセスルールの細微化が進むことで，リーク電流が増大し，リーク電流の存在が無視できない状況になっており，静的消費電力の割合が高くなっている．このため，動的消費電力だけでなく静的消費電力も削減することが必要である．

一方で，プロセッサ上で実行されるソフトウェアは大規模化及び複雑化してきておりプロセッサの高速化が求められてきた．プロセッサと主記憶は共に高速化されているが，主記憶はプロセッサの動作周波数向上と同様な高速化までには至っておらずプロセッサと主記憶の速度差は増大の傾向にある [17]．プロセッサと主記憶の速度差が増大してしまうと，プログラムを実行している際に，主記憶へのアクセスがボトルネックになって，プログラム実行速度に悪影響がでてしまう．

そのため，主記憶へのアクセス回数を減らすために小容量で高速なキャッシュメモリがプロセッサと主記憶との間に設けられるようになりキャッシュメモリによって，プログラムの実行速度への悪影響を軽減できるようになった．また，トランジスタの小型化と高集積化によってプロセッサチップ上にキャッシュメモリを多く載せることが可能となり，キャッシュメモリの容量は増加の一途をたどっている．これにより，キャッシュメモリがプロセッサの面積の大部分を占めるようになり，それに伴ってキャッシュメモリの消費電力が占める割合はプロセッサ中の 50%以上を占

めていると報告されている [14].

そのため，近年ではプロセッサの消費電力を削減するために，キャッシュメモリの消費電力を削減する手法が研究されている [1][6][7][14].

1.2 研究の目的とアプローチ

本研究は，プロセッサの消費電力のうちその割合の大きいキャッシュメモリに着目して，キャッシュメモリの消費電力を削減することを目的とする．

消費電力のうち，動的消費電力については従来通り電源電圧を下げることによって省電力化が可能である．また，静的消費電力については，リーク電流の小さなトランジスタでキャッシュを構成することで省電力化が可能である．しかし，動的消費電力と静的消費電力の削減に伴ってアクセスレイテンシは増大してキャッシュメモリの性能は低下してしまう問題が発生する．

そこで以下の手法が提案されている [1][18]．すなわち，キャッシュメモリに，低レイテンシな領域と高レイテンシな領域を用意する．低レイテンシな領域は，高い電源電圧が供給され，しきい値電圧の低いトランジスタで構成される．このためアクセス時の電力及びリーク電流は大きいがデータは高速に供給される．一方，高レイテンシな領域は，低い電源電圧が供給され，しきい値電圧の高いトランジスタで構成される．このためアクセス時の電力とリーク電流が抑えられるがデータは低速に供給される．これらの領域のうち，低レイテンシな領域には重要度の高いデータを配置する．一方，重要度の低いデータは高レイテンシな領域に配置する．このようにデータ配置の最適化を行うことで処理性能を低下させずに動的消費電力と静的消費電力を削減することを可能としている．

キャッシュの省電力問題に限らず，データの重要度の決定方法としては，時間的局所性に基づく方法，クリティカルパス情報に基づく方法がよく用いられており，文献 [1] のキャッシュ分割手法においてもクリティカルパス情報を用いている．しかし，クリティカルパス情報には重要度が 2 段階でしか表現できないことや，命令がクリティカルパス上にあるか否かの判定が困難であるといった欠点がある．一方，命令スケジューリングの分野では，命令のスラック [12] をクリティカルパス情報の代わりに利用する手法が提案されている [2][4].

そこで本研究では，前述の高レイテンシと低レイテンシの 2 つの領域を持つ L1 キャッシュメモリにおいて，データの重要度の決定を命令のスラックを用いて行い，スラックに基づいたデータの移動や命令の重要度に応じたキャッシュへのアクセスを行う方式を提案する．スラックの小さい命令によってアクセスされたデータは重

要度が高いとして，低レイテンシな領域に配置することでプログラムの実行が遅れないように，スラックの大きい命令によってアクセスされたデータは重要度が低いとして，高レイテンシな領域に配置することで電力の消費を抑えるようにする．これによりデータの重要度にクリティカルパス情報を用いるのに比べ，データの各領域への配置をより適切なものにできると考えた．この方式を採用した際にどの程度の電力の削減が認められるか，性能低下をどの程度に抑えられているかを検証して，その有効性を示す．

2　キャッシュメモリの省電力化

2.1　一般的なキャッシュメモリアーキテクチャ

キャッシュメモリは，プロセッサと主記憶との性能差を隠蔽するために用いる高速で小容量なメモリのことである．属性情報のセットを固定容量のメモリに格納するので，キャッシュメモリの階層化，データ格納構造，データ更新方式などに複数のアーキテクチャが存在している．以下に，キャッシュメモリの階層化，データ格納構造，データ更新方式について説明を行う．

- **キャッシュメモリの階層化**

  プロセッサの速度が速くなるにつれて，主記憶とプロセッサの速度のギャップが大きくなったためキャッシュを多段階層化することで速度のギャップを埋める試みが行われてきた．階層化されたキャッシュメモリは，プロセッサに近い順から L1 キャッシュメモリ，L2 キャッシュメモリが設けられ，近年では L3 キャッシュメモリまで設けられている例もある．

- **データ格納構造**

  キャッシュメモリは，データをある程度まとまった単位（ライン）で管理しており，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に存在しているか，存在しているならどのラインかを瞬時に検索する必要がある．そのためデータ格納アドレスの一部を用いある程度格納位置（セット）を限定することで検索速度を高めている．セットの決め方に以下の方式がある．

  – ダイレクトマップ方式

    セット中に 1 つしかラインを持たない方式，アドレスにより一意に配置が決まるため，検索のための構造が単純であるが，同一セットに異なるアドレスが転送されると必ずラインの入れ替えが発生するため，キャッシュヒット率は高くない．

  – セットアソシアティブ方式

    セット中に複数のラインを持つ方式，同一セットに異なるアドレスのデータを複数格納することができる．セット中のライン数を上げるほどキャッシュヒット率は上昇するが，検索のための構造は複雑になり実装は困難になっていく．1 セットが n 個のラインにより構成されたキャッシュメモリである場合，n ウェイセットアソシアティブ方式と呼ぶ．

– フルアソシアティブ方式

セットを 1 つしか持っておらず，全てのラインが検索対象となる方式．ラインの入れ替えが発生しにくくヒット率は最も優れているが，実装コストや複雑度の面から通常は用いられない．

- **データ更新方式**

メモリへの Write アクセスが発生すると，キャッシュメモリのデータは更新される．更新されたデータは，下位レベルのメモリのデータにも反映させ，記憶階層全体で一貫性（キャッシュコヒーレンシ）を保たなくてはならない．更新されたデータを下位レベルのメモリに反映させるタイミングの相違によって 2 つのアルゴリズムが存在する．

– ライトスルー方式

メモリへの Write アクセスが発生した場合，キャッシュメモリのデータを更新すると同時に，下位レベルのメモリにも書き戻す方式．単純な構造で実現でき，キャッシュのコヒーレンシを保つことも容易にできる．また，読み出しミスによって下位レベルのメモリへデータ追い出しが発生しないのでその場合のペナルティが小さくてすむ．ただし下位レベルのメモリへ同時に書き込みを行っているので，書き込み速度は下位レベルのメモリの速度に束縛されてしまう．

– ライトバック方式

メモリへの Write アクセスが発生した場合，キャッシュメモリのデータのみ更新を行い，下位レベルのメモリへの書き戻しは読み出しミスによって下位レベルのメモリへデータの追い出しが必要な場合に行う．書き込み速度がキャッシュメモリの速度で実行することができ，下位レベルのキャッシュへの書き戻しの際に，ライン中の複数のデータを一度の書き戻しで行うことができる．

2.2 省電力キャッシュメモリアーキテクチャ

本節では本研究が対象としている省電力キャッシュメモリアーキテクチャについて述べる．本研究で対象とするキャッシュメモリは一般的なプロセッサの L1・L2 キャッシュとライトバッファを持つライトバック方式のキャッシュメモリを想定し，さらに L1 のデータキャッシュメモリを性能指向な低レイテンシなウェイと，省電力指向な高レイテンシなウェイに分割する．低レイテンシなウェイは閾値電圧の低いトラ

ンジスタで構成され，高い電源電圧が供給される．このため，アクセスに必要な電力と回路のリーク電流は大きいが，データ供給は高速に行われる．一方，高レイテンシなウェイは閾値電圧の高いトランジスタで構成され，低い電源電圧が供給される．このため，アクセスに必要な電力と回路のリーク電流は小さいが、データの供給は低速に行われる．以降本研究では，性能指向の低レイテンシなウェイの集合を『高速な領域』，省電力指向の高レイテンシなウェイの集合を『低速な領域』と呼ぶ．

このキャッシュメモリでは，重要度の高いデータは高速な領域に配置されることで，プログラムの実行が遅れないようにすることができ，重要度の低いデータは低速な領域に配置されることで，プログラムの実行を遅れさせずに消費電力を削減することができる．また，キャッシュメモリを分割した場合は，キャッシュメモリへアクセスする領域がキャッシュメモリを分割しない場合と比べて小さくなるため，1 回のアクセスで消費する電力を削減することができる．

このキャッシュメモリでは，同一セット内のデータの重要度をラインごとに設定する必要がある．これはデータの読み出しがラインごとに行われるためである．データの重要度を決める方法として，時間的局所性に基づく方法 [1]，クリティカルパス情報に基づく方法 [1] が既存の手法として考案されているが，本研究では新たに，**スラックに基づく方法**を提案する．スラックについては 3 章で詳細を述べる．

時間的局所性に基づいてデータの重要度を決める場合は，LRU(Least Recently Used) 情報を用いて重要度を決定する．この場合，LRU の情報は高速な領域と低速な領域で別々に管理する．クリティカルパス情報に基づいてデータの重要度を決める場合は，クリティカルパス上の命令によってアクセスされたデータであるかを識別するためにラインごとにフラグを設けて，フラグを参照することで重要度を決定する．スラック情報を利用してデータの重要度を決める場合，ラインごとに命令のスラックの大きさをデータの重要度として保持するための領域をとり，保持されている値を用いて重要度を決定する．図 1 はスラックを重要度とするキャッシュメモリの構成を示している．重要度の決定方法は 4 章で詳細を述べる．

また，本研究では，一般的 RISC アーキテクチャを想定して，メモリアクセス命令は，キャッシュメモリからデータを読み出すロード命令，キャッシュメモリへデータを書き込むストア命令だけであると仮定する．このうちストア命令は，一般的なキャッシュメモリと同様に，書き込むデータをバッファに置くことで後続命令の実行を遅らせることはないと想定する．よって，プログラムの実行時間に影響を与えるのは，キャッシュメモリからデータを読み出すロード命令であり，ロード命令のスラックを用いてデータに重要度を付けることとする．

さらに本研究ではメモリアクセス命令ごとに低速又は高速な領域のいずれかにア

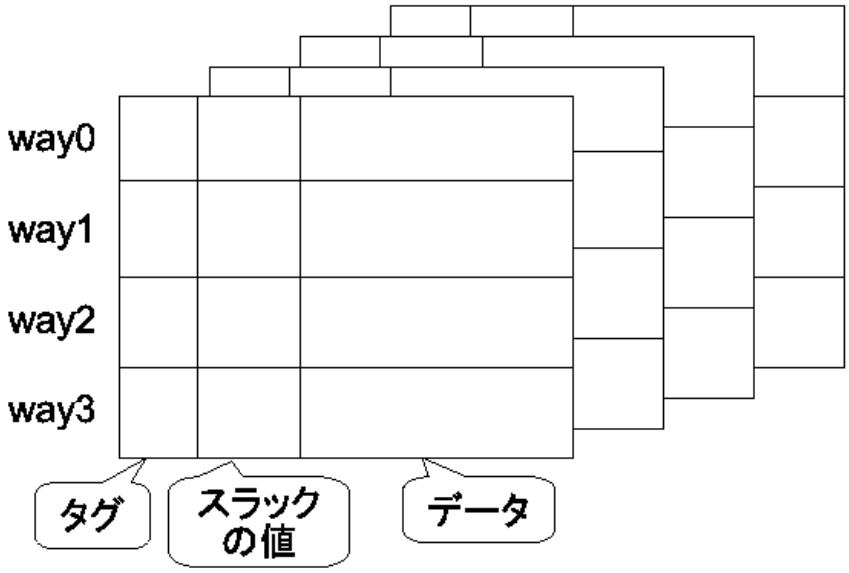

図 1: スラックを重要度とするキャッシュメモリ

クセスできることし，基準値によりある命令がどちらの領域にアクセスするか選択することとする．各領域への選択的アクセスについては 5 章で詳細を述べる．

図 2 は，プロセッサと主記憶の間に L1 と，L2 の二階層のキャッシュメモリを配置し，L1 を高速・低速な領域に分割している場合を表している．キャッシュメモリ内でのデータ移動は矢印のように行われる．

プロセッサは L1 キャッシュメモリの高速・低速な領域に読み出し・書き込み処理を行うことができる．一方 L1 キャッシュメモリと L2 キャッシュメモリとの間では，低速な領域からのみ L2 キャッシュメモリへの書き込みができるようになっている．L2 キャッシュメモリからの読み出しは，読み出した命令の重要度に応じて高速・低速どちらの領域に対しても行うことができる．L2 キャッシュメモリは主記憶に対して通常の書き込みと読み出しが行える．本研究では，図 2 に示した構成のキャッシュメモリ，データ配置方法を採用した際にどの程度の省電力化ができるかを検証していく．図 2 に示したデータ移動方式を採用して重要度に基づいたデータ移動を行うことで，L1 キャッシュメモリでは重要度の高いデータを高速な領域に，重要度の低いデータを低速な領域へ集める．

L1 キャッシュメモリを高速な領域と低速な領域に分割する場合，L1 キャッシュメモリの低速な領域を L2 キャッシュメモリ，L2 キャッシュメモリを L3 キャッシュメモリといったように，分割した領域を利用して新たな記憶階層を構築することも考えられる．本研究では上位階層に存在するデータは下位階層にも存在するキャッシュ

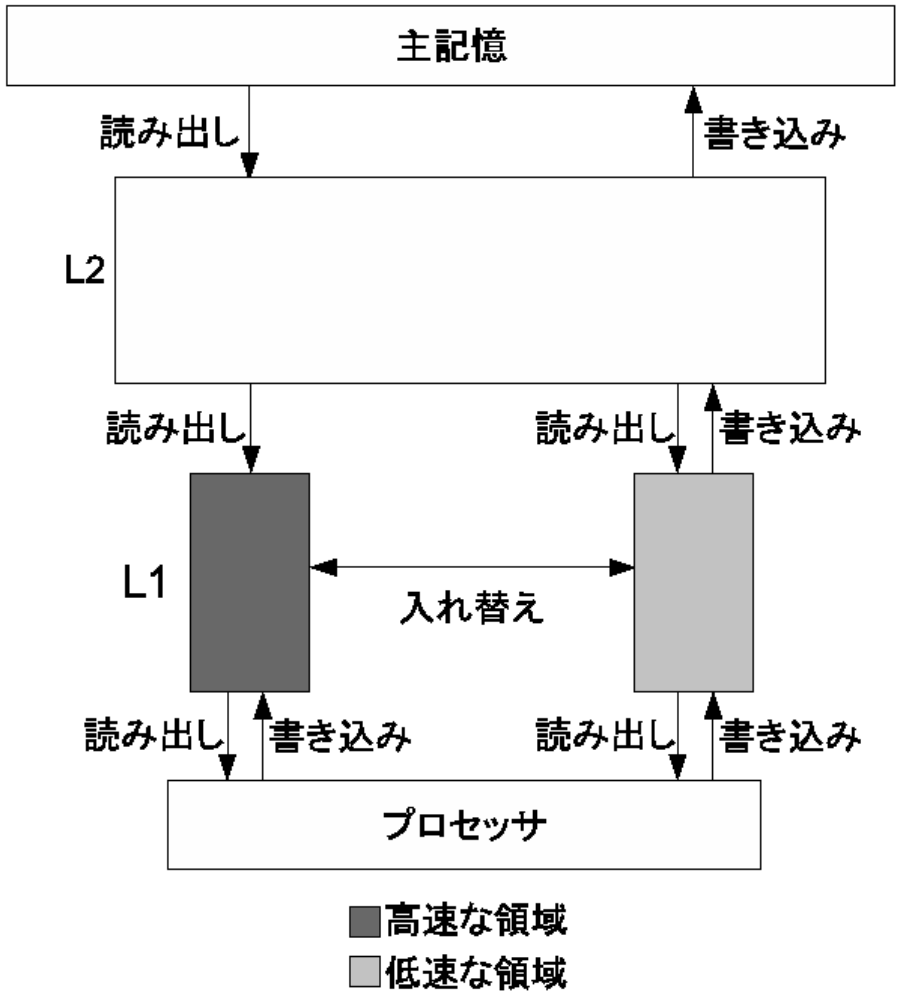

図 2: データの重要度を利用したキャッシュメモリ

メモリを想定している．よって，キャッシュメモリを分割する場合と分割しない場合でキャッシュメモリの総容量を変化させないとすると，分割した領域で階層化を行った場合，キャッシュメモリ内でアクセス可能なデータ量が減少しキャッシュの容量ミスを増加させてしまうことになる．また，本研究では，高速な領域と低速な領域のどちらに対してもプロセッサからアクセスできることとしているため，高速な領域と低速な領域との間で新たな記憶階層を構築してしまうと，プロセッサから低速な領域へアクセスができなくなってしまう．これらの理由のため、本研究では高速な領域と低速な領域との間で新たな記憶階層は構築しない．

3　スラック値によるデータの重要度に基づくデータ配置領域の選択

本研究で検証を行う省電力キャッシュアーキテクチャでは，高速な領域に重要度の高いデータを配置することで性能への影響を小さくする．また，低速な領域に重要度の低いデータを配置することでプログラムの実行を遅らせずに，消費電力を抑える．したがって，キャッシュメモリ内のデータの重要度を決める必要がある．データの重要度を決定方法として時間的局所性に基づく方法やクリティカルパス情報の基づく方法が既存の手法として考案されている．本研究では，新たにスラック [9] に基づく方法を提案し，既存の手法と比較する．

3.1　時間的局所性に情報に基づく方法

時間的局所性に基づく方法では，プログラムのデータアクセスの時間的局所性に着目する．プログラムには，最近アクセスされたデータには近い将来再びアクセスする可能性が高いという時間的局所性が存在する．最近アクセスされたデータほど再びアクセスされる可能性が高く，過去にアクセスされたデータになるほど再びアクセスされる可能性が低くなる．

頻繁にアクセスされるデータの供給が遅れるとプログラムの実行が遅れる可能性が高いと考えて，最近アクセスされたデータの重要度を高くする．

3.2　クリティカルパス情報に基づく方法

クリティカルパスとは，命令間の依存関係を結んだ鎖のうち，プログラム終了までに最も時間のかかる命令列で構成されるパスである．クリティカルパス上の命令の実行が遅れるとプログラム終了までにかかる時間も遅れてしまう．図 3 に命令列中に現れるクリティカルパスを表すデータフローグラフの例を示す．図 3 における矢印は命令間の依存関係を示す．つまり，依存している命令は矢印の始点にある命令すべてが実行終了していない限り実行することができない．全ての命令のレイテンシが 1 サイクルであるとすると，最も長いパスである命令 i0 → i3 → i4 → i6 → i7 を結んだパスがクリティカルパスとなる．

クリティカルパス情報を用いてデータの重要度を決定する場合には，クリティカルパス上のメモリアクセス命令によってアクセスされたデータの供給が遅れるとプ

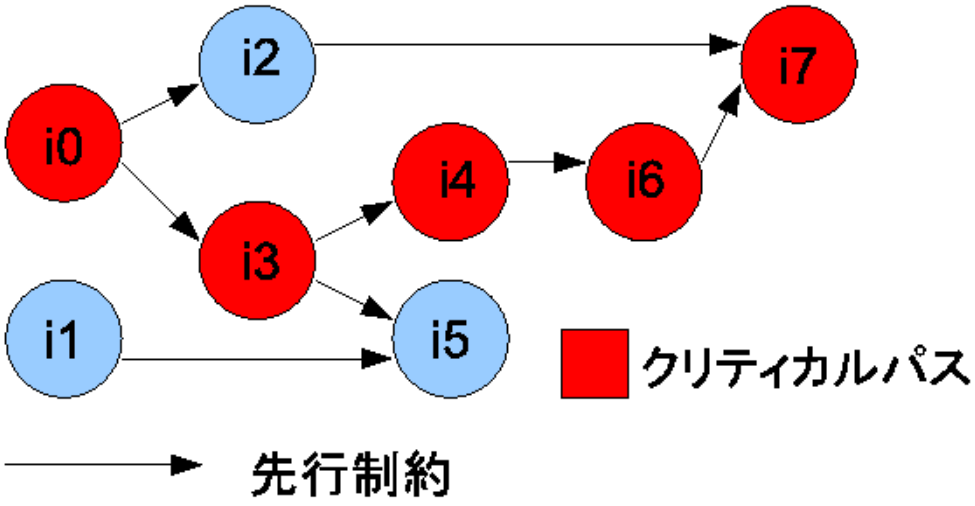

図 3: クリティカルパス

ログラムの実行が遅れる可能性が高いと考えて，当該アクセスによってアクセスされたデータの重要度を高くする．

3.3　命令のスラックに基づく方法

クリティカルパス情報は命令スケジューリングの分野で命令に重要度を与える指標として用いられているが，その重要度は命令がクリティカルパス上にあるかないかの 2 通りしかなく，命令は 2 種類にしか分類できない．

これに対し，クリティカルパスでなく命令のスラックの値によって命令の重要度を判定する手法が提案されている．命令 i のスラックとは，ある命令 i の実行を s サイクル遅らせてもプログラ全体のクリティカルパス長を長くしない s をいう．各命令のスラックが分かれば，各命令がクリティカルパス上にあるかどうかだけでなく，クリティカルパス上にない各命令に対して，そのスラック値に対応して命令の重要度に差を与えることができ，重要度の分解能を高くすることができる．スラックの最小値は 0 である．スラックの値が 0 の命令は，クリティカルパス上の命令ということになる．

スラックの例を図 4 のデータフローグラフを用いて説明する．演算器資源には制限がないとする．命令 i0，i2，i3，i5，i6 の実行レイテンシは 1 サイクルであるとし，命令 i1，i4 の実行レイテンシは 2 サイクルであるとする．図 4 に示した命令列をプ

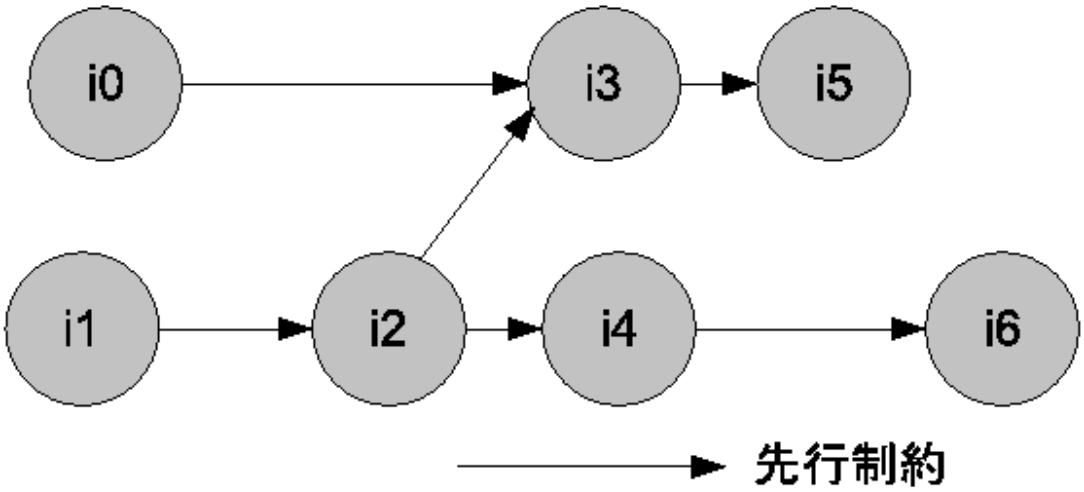

図 4: データフローグラフ (1)

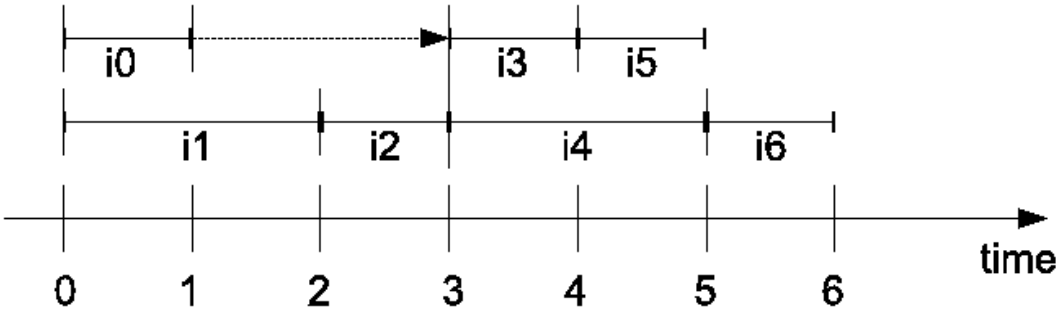

図 5: 命令実行のタイムチャート (1)

ロセッサ上で実行する過程を表したものが図 5 である．ここで命令 i0 に着目する．i0 の実行を 3 サイクル遅らせると，i0 に直接的，間接的に依存する i3，i5 の実行も遅れ，その結果 i5 はもっとも最後に実行される i6 と同時に終了することになるが，全体の実行時間を延ばすことにはならない (図 6)． 一方，i0 の実行レイテンシをこれ以上増加させると，図 4 の命令全体の実行完了が遅れてしまう．したがって，i0 のスラックは 3 とする．このように，ある命令のスラックを求めるためには，当該命令を遅らせたことによってプログラム全体が受けた影響を調べなくてはならない．これをプログラム実行中に求めるのは困難であり，スラックを求めることは難しい．

そこで，スラックではなく，**ローカルスラック**[9] を利用する手法が提案されてい

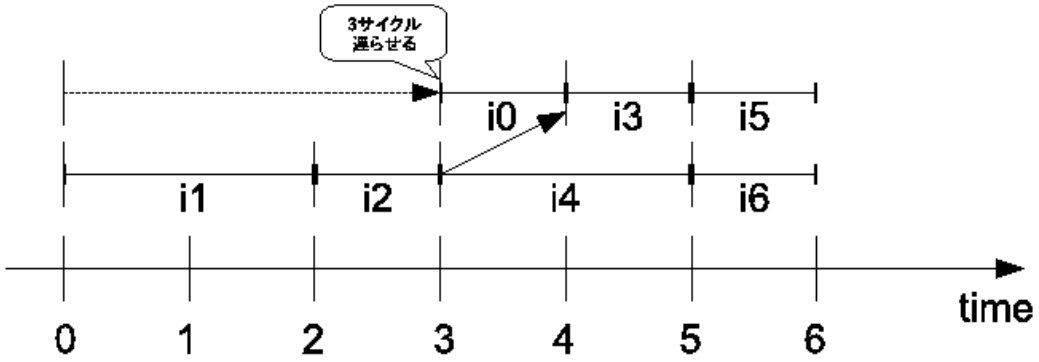

図 6: スラックの例

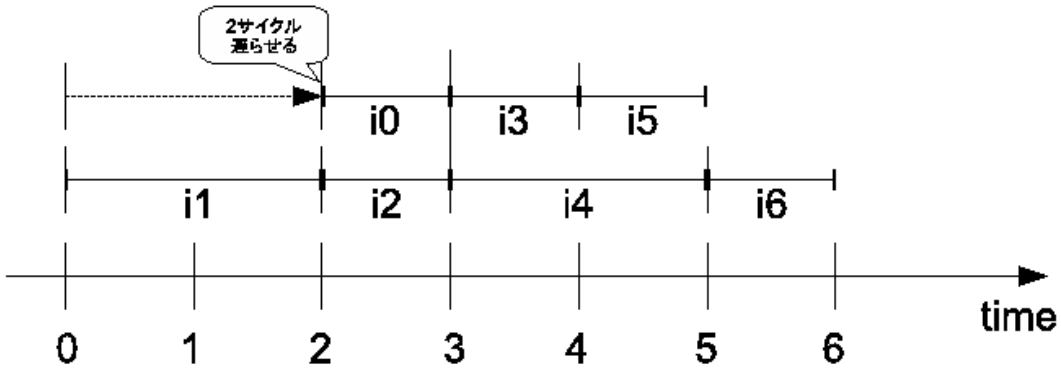

図 7: ローカルスラックの例

る [2][4]．命令のローカルスラックとは，資源制約の無い状態で命令が最適にスケジューリングされて実行される場合を想定し，ある命令 i がその命令に依存する後続命令の実行時刻も遅らせることなく増加させることのできるサイクル数である．

ローカルスラックの例も図 4 のデータフローグラフと図 5 のプロセッサ上での実行過程用いて説明する．ここで命令 i0 に着目する．i0 の実行を 2 サイクル遅らせた場合，後続命令である i3 の実行開始時刻を遅らせることはない (図 7)．しかし，これ以上実行レイテンシを増加させると，i0 に直接的，間接的に依存関係にある i3 と i5 の実行開始が最適にスケジューリングされている場合より遅れる．したがって，i0 のローカルスラックは 2 とする．このように，ある命令のローカルスラックは当該命令に後続する命令に着目するだけで容易に求めることができる．本研究でもスラッ

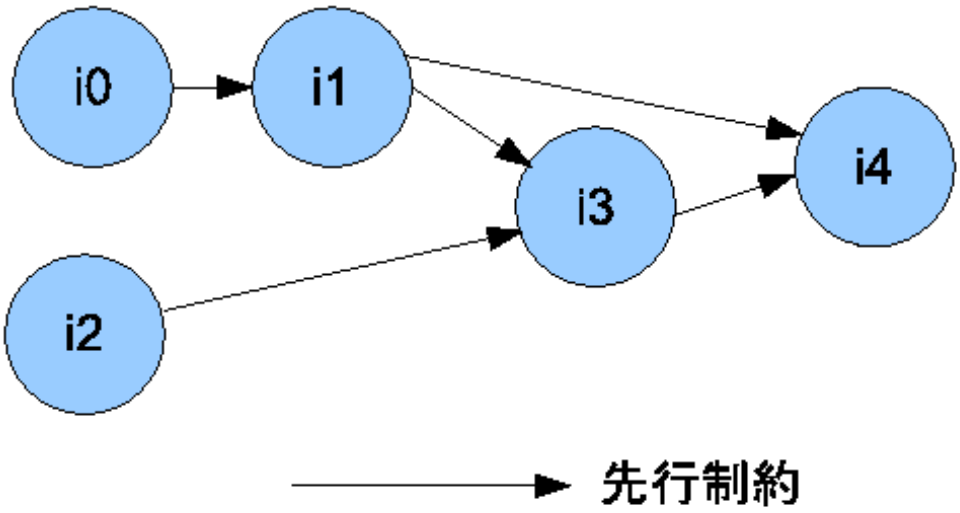

図 8: データフローグラフ

クではなくローカルスラックを各命令のスラックとして用いる．なお，前述のスラックは，ローカルスラックに対応してグローバルスラックとも呼ばれる [9]．本研究では以後，ローカルスラックをスラックと表記する．

命令のスラックに基づいてデータの重要度を決定する場合には、スラックの小さいメモリアクセス命令によってアクセスされたデータの供給が遅れるとプログラムの実行が遅れる可能性が高いと考えて，当該アクセスによってアクセスされたデータの重要度を高くする．

3.3.1 スラックの求め方

本研究では，ある命令のスラックは，その命令がレジスタ又は変数に値を定義した時刻と，定義されたレジスタ又は変数を他の命令が参照した時刻の差として求めることとする．この求め方は，本来のローカルスラックの定義の値とは異なる値となるが，命令が実際に実行された過程から求めるので，資源制約などの物理的制約を反映した値を得ることができる．

具体的な命令スラックの求め方を例を用いて説明する．説明に用いる命令列のデータフローグラフを図 8 に示す．命令 i0，i1，i3，i4 の実行レイテンシは 1 サイクルであるとし，命令 i2 の実行レイテンシは 2 サイクルとする．図 9 は，図 8 に示した命令列が実際に実行された様子を表すタイムチャートを示している．図 9 では，命令

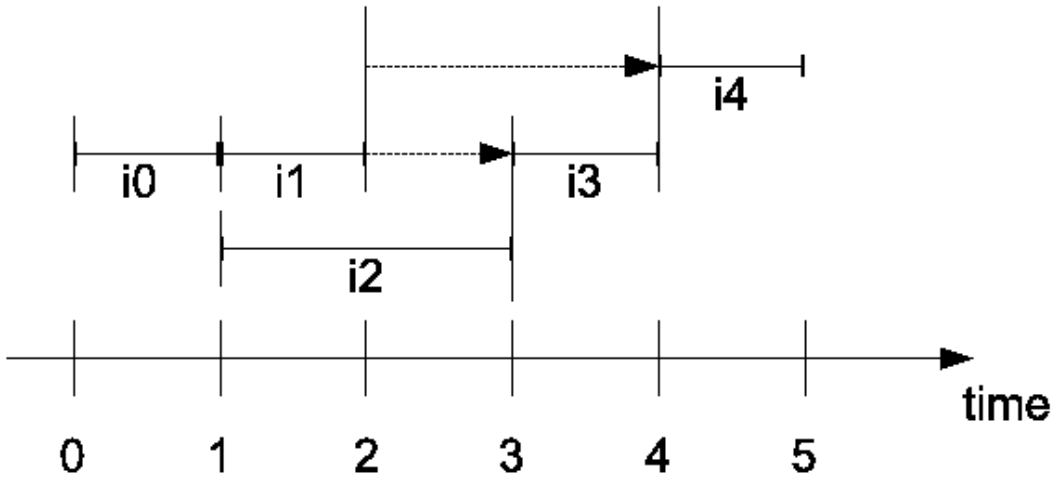

図 9: 命令実行のタイムチャート

i1 が定義した結果を最初に参照する命令は i3 である．命令 i1 のスラック値は，命令 i1 による定義時刻と命令 i3 による参照時刻の差をから求める．したがって図 9 では，命令 i1 のスラックは 1 サイクルとなる．なお，以下では命令のスラックの単位を省略し,『命令 i1 のスラックは 1』と表す.

また図 8 のデータフローグラフでは命令 i2 は特に先行制約のない命令であったが図 9 で，命令 i2 は，演算器資源不足などの理由により，理想的な場合より 1 サイクル遅れて実行されている．命令 i1 のスラックが 1 となるのは，命令 i2 の実行が遅れたことに起因している．資源制約を考慮しないローカルスラックの定義では，データ依存関係と命令の実行レイテンシだけでローカルスラックの値を求めるため，命令 i1 のスラックは 0 と定義される.

3.3.2 スラック予測器

本研究では，前述の通りメモリアクセス命令は，キャッシュメモリからデータを読み出すロード命令，キャッシュメモリへデータを書き込むストア命令としている．このうちストア命令は書き込むデータをバッファに置くことで後続命令の実行を遅らせることはないと想定している．プログラムの実行時間に影響を与えるのは，キャッシュメモリからデータを読み出すロード命令である．よってロード命令のスラックを用いてデータに重要度を付けることとする.

ロード命令のスラックをデータの重要度に利用するには，当該命令がフェッチさ

れた時点で，その命令のスラックの値が求められている必要がある．しかし，前項で述べた命令のスラックの求め方では命令を実行する前にその命令のスラックの値を求めることはできない．そのため命令のスラックを予測する必要がある．

あるロード命令のスラックを予測する方法として，そのロード命令が前回実行された時の命令のスラックを求めてこれを次回の実行の命令のスラックの値として用いることとする．

ロード命令のスラックを予測するための予測器は以下の 2 つの表からなる．

- **実行記録**　各データに対してそのデータがレジスタに設定された時刻，そのデータをレジスタに設定したロード命令を記録する表である．スラック表に記録するロード命令のスラックを計算するために用いられる．

  表は，物理レジスタファイルの各データに対して，データを設定した時刻，データを設定したロード命令を記録するフィールドを付加することで実現する．

  当該レジスタが参照された時点で，データと同時に表に記録されている時刻を読み出せば，表に記録されているロード命令のスラックを計算することができる．

- **スラック表**　命令の前回実行時のスラックの値を記録する表である．

  命令がフェッチされると同時にスラック表を参照することで前回実行時の命令のスラックの値を読み出し，それを今回の命令のスラックの予測値とする．

3.3.3 スラック予測器の動作

命令 I_d のスラックの予測値を求める動作を説明する．プログラム終了までに命令 I_d が N 回実行されるとする．命令 I_d の n 回目 $(n \in N)$ の実行を，肩付き数字を用いて，I_{d^n} のように表すことにする．また，I_{d^n} がレジスタに設定したデータを最初に使用する命令の実行を I_{u_n} と表す．なお，I_{d^i} と I_{d^j} $(i, j \in N, i \neq j)$ は同じ命令であるが，I_{u_i} と I_{u_j} は一般に異なる．

図 10 では，ロード命令 I_{d^1} とロード命令によってレジスタに設定されたデータを使う命令 I_{u_1} が，それぞれ，時刻 t_{d^1} および t_{u_1} に実行されている．スラック表への登録，および，同スラック表への参照，すなわち，予測は，次の様に行われる．

登録　登録は，以下のように，(a) I_{d^1} がレジスタにデータを設定するときと，(b) I_{u_1} がそのレジスタの値を使用するときの 2 つのフェーズからなる：

1. 定義： I_{d^1} がレジスタにデータを設定するとき，以下の操作が行われる：
   - 現時刻 t_{d^1} と I_{d^1} 自身（のアドレス）を表に書き込む（図 10 の W_D）.
2. 参照： I_{u_1} がそのレジスタの値を参照するとき，以下の 2 つの操作が連続して行われる：
   - 表を読み出して，レジスタにデータを設定した時刻 t_{d^1} とレジスタにデータを設定したロード命令 I_{d^1} の命令のアドレスを得る（図 10 の R_D）.
   - データ設定時刻 t_{d_1} と現時刻 tu_1 から，I_{d^1} のスラック s_{d^1} が，$s_{d_1} = t_{u_1} - t_{d^1}$ と求まる．スラック表のロード命令 I_{d^1} のエントリに，求めた s_{d^1} を書き込む（図 10 の W_S）.

予測　命令 I_d が再びフェッチされると，以下の操作が行われる：

- I_d のアドレスをインデクスとしてスラック表を直接読み出すことで，前回の命令のスラック s_{d^1} が得られる．I_{d^2} のスラック予測値として s_{d^1} の値を利用することができる（図 10 の R_S）.

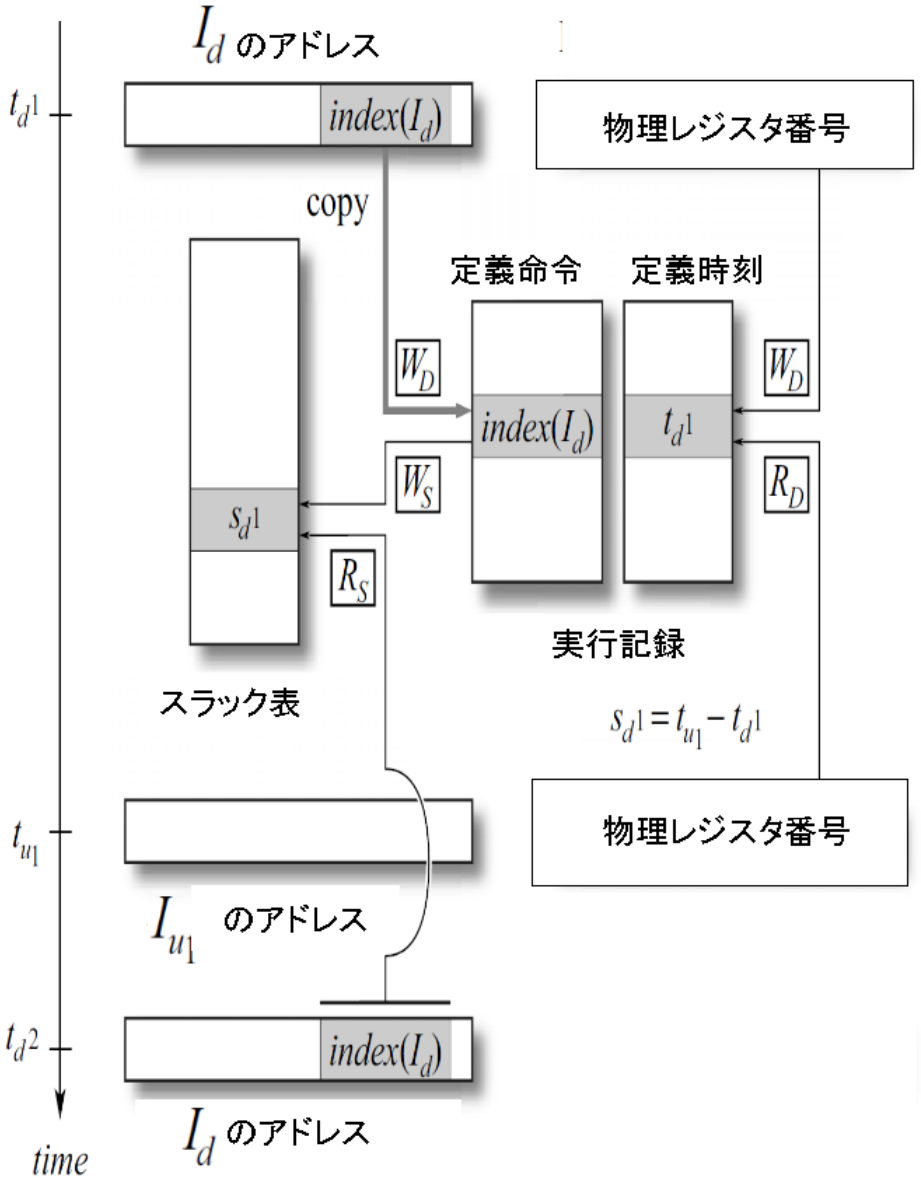

図 10: スラック予測器の動作

4 キャッシュメモリ内のデータの重要度決定方法とデータ移動

5 章で詳細を説明するが，本研究ではロード命令にスラックの値が予測されており，スラックの小さい命令によるアクセスとスラックの大きい命令によるアクセスに分けることができる．スラックの小さい命令によるアクセスの場合は，あまり実行時間を遅らせることができないため高速な領域からデータが供給されることが理想である．したがって供給されたデータがもともと高速な領域にあった場合はデータの移動は必要としないが，アクセスされたデータが低速な領域にあった場合は，次回以降のアクセスは高速な領域からアクセスされた方が実行時間を短くできるため，そのデータを低速な領域から高速な領域に移動させることとする．

また，L1 キャッシュメモリでミスをした場合に，L1 キャッシュメモリと L2 キャッシュメモリとの間でデータを移動させることが必要となる．

本章では，命令のスラックを基にデータの重要度を決定する方法と，その重要度を基にキャッシュメモリ内のデータを移動させる方法について説明する．

4.1 データの重要度の決定

本研究ではキャッシュメモリの構成としてセットアソシアティブ方式をとることを想定している．一般的に，キャッシュメモリでは，データの読み出しがラインごとに行われるため，データの重要度をラインごとに決定する必要がある．以下では 4way セットアソシアティブ方式のキャッシュメモリのセット１つを例にとって説明する．また，メモリアクセス命令はメモリからデータを読み出すロード命令，メモリへデータを書き込むストア命令の２つがあるが，3.3.2 項で述べたようにプログラム実行時間に影響を与えるのはロード命令であるため，データの重要度を決定することに用いるメモリアクセス命令は，ロード命令の場合のみとする．

- **時間的局所性に基づいた重要度の場合**

  時間的局所性を用いてデータの重要度を決定する場合，LRU 情報を用いて重要度を決定する．セット中の最もアクセスされていないラインの重要度が低いと決定される．

- **クリティカルパス情報に基づく重要度の場合**

  クリティカルパス情報に基づいて，データの重要度を決定する場合，セット中の各ラインがクリティカルパス上に存在する命令によってアクセスされたラ

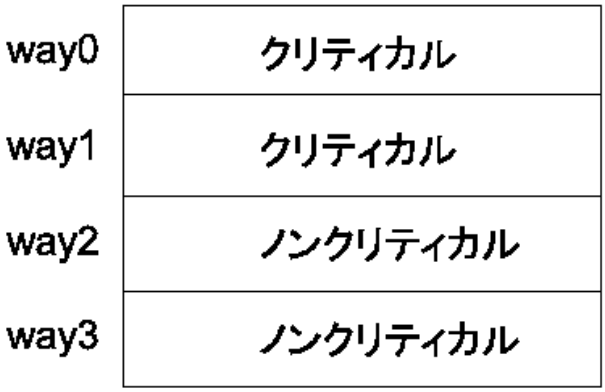

図 11: クリティカルパス情報に基づく重要度

イン（クリティカル）であるか，クリティカルパス上に存在しない命令によってアクセスされたライン（ノンクリティカル）であるかを保持することで重要度を決定する．

図 11 はセット中の各ラインがクリティカルなラインであるか，ノンクリティカルなラインであるかを示した一例である．図 11 の場合，way2，way3 のラインは way0，way1 のラインより重要度が低いと決定される．また．way0 のラインと way1 のラインとの間，及び way2 のラインと way3 のラインとの間では特に重要度に差は付けられない．

ラインがクリティカルであるか，ノンクリティカルであるかは，当該ラインにアクセスが行われるごとに更新される．つまり，最も最近アクセスした命令によって当該ラインの重要度が決定されることとなる．

また，ストア命令によるアクセスによってライン内のデータが書き換えられる場合，ストア命令にはクリティカリティが考慮されていないが，当該ラインに保持されているクリティカリティの情報には値を設定しなくてはならない．この場合，本研究では当該ラインはクリティカルであるとした．

- **命令のスラックに基づく重要度の場合**

  命令のスラックに基づいてデータの重要度を決定する場合，セット中の各ラインに重要度指標を保持する領域を付加する．ラインにアクセスがあった時点で，当該ラインにアクセスを行ったロード命令のスラックの値を付加した領域に重要度として記録し，各ラインに記録されている重要度指標を参照することで重要度を決定する．

  図 12 はセット中の各ラインに重要度指標が記録されて，命令のスラックに

図 12: 命令のスラックに基づく重要度

基づいてデータの重要度が決定される様子の一例を示したものである．命令のスラックの値が 0 であることは，クリティカルパス上の命令によってアクセスされた可能性が高いラインであることを表す．また，命令のスラックの値が 1 以上であることは，ノンクリティカルパス上の命令によってアクセスされた可能性が高いラインであることを表す．

図 12 の場合，way3 のラインが最も重要度が低いラインであると決定される．クリティカルパス情報に基づいて，データの重要度を決定する場合 (図 11) と比較すると，way2 のラインと way3 のラインとの間にも重要度に差をつけることができていることが特徴である．

ラインの重要度の値は，クリティカルパス情報に基づく方法と同様に，当該ラインにアクセスが行われるごとに更新される．つまり，最も最近アクセスした命令のスラックの値によって当該ラインの重要度が決定される．

また，ストア命令によるアクセスによってライン内のデータが書き換えられる場合，ストア命令のスラックの値は不明であるが，当該ラインに保持される重要度の値は何らかの値を設定しなくてはならない．この場合，本研究では重要度の値は 0 であるとした．

4.2 データ移動アルゴリズムの詳細

本節では，高速な領域と低速な領域との間でデータの移動が必要となる場合のデータ移動方法，L1 キャッシュメモリと L2 キャッシュメモリとの間でのデータ移動方法，

及び L2 キャッシュメモリと主記憶との間でのデータ移動方法についての詳細を説明する．

1 高速な領域と低速な領域との間のデータ移動について

1-1 スラックの小さい命令によるアクセスの場合

スラックの小さい命令によってアクセスされたデータがもともと高速な領域にあった場合はデータの移動は必要としないが，スラックの小さい命令によってアクセスされたデータが低速な領域にあった場合は，次回以降のアクセスは高速な領域からアクセスされた方が実行時間を短くできるため，そのデータを低速な領域から高速な領域に移動させることとする．

まず当該アクセスに対して，低速な領域から必要なデータを供給する．データ供給後，高速な領域へデータを移動させる．高速な領域の移動先となるラインの決定方法として，時間的局所性に基づいた方法，クリティカルパス情報に基づく方法，スラックに基づく方法の 3 種類の方法がある．それぞれの場合において移動方法を説明する．

* **時間的局所性に基づく方法の場合**

  高速な領域の同一セットに空きラインがある場合は，そこへラインを移動させる (図 13 の (a))．高速な領域の同一セットに空きラインがなければ最古にアクセスされた (LRU) 古いラインを置き換え対象として移動を行う (図 13 の (b))．

* **クリティカルパス情報に基づく方法の場合**

  高速な領域の同一セットに空きラインがある場合は，そこへラインを移動させる (図 14 の (a))．高速な領域の同一セットに空きラインがない場合は，ノンクリティカルなラインが存在するかを調べる．ノンクリティカルなラインが存在した場合は，そのラインを置き換え対象として移動を行う (図 14 の (b))．高速な領域の同一セットに空きラインがなく，ノンクリティカルなラインも存在しない場合は，置き換え対象のラインは LRU で決定する (図 14 の (c))．

* **スラックに基づく方法の場合**

  高速な領域の同一セットに空きラインがある場合は，そこへラインを移動させる (図 15 の (a))．高速な領域の同一セットに空きラインがない場合は，ラインの重要度の値を調べて，重要度の値の最も大

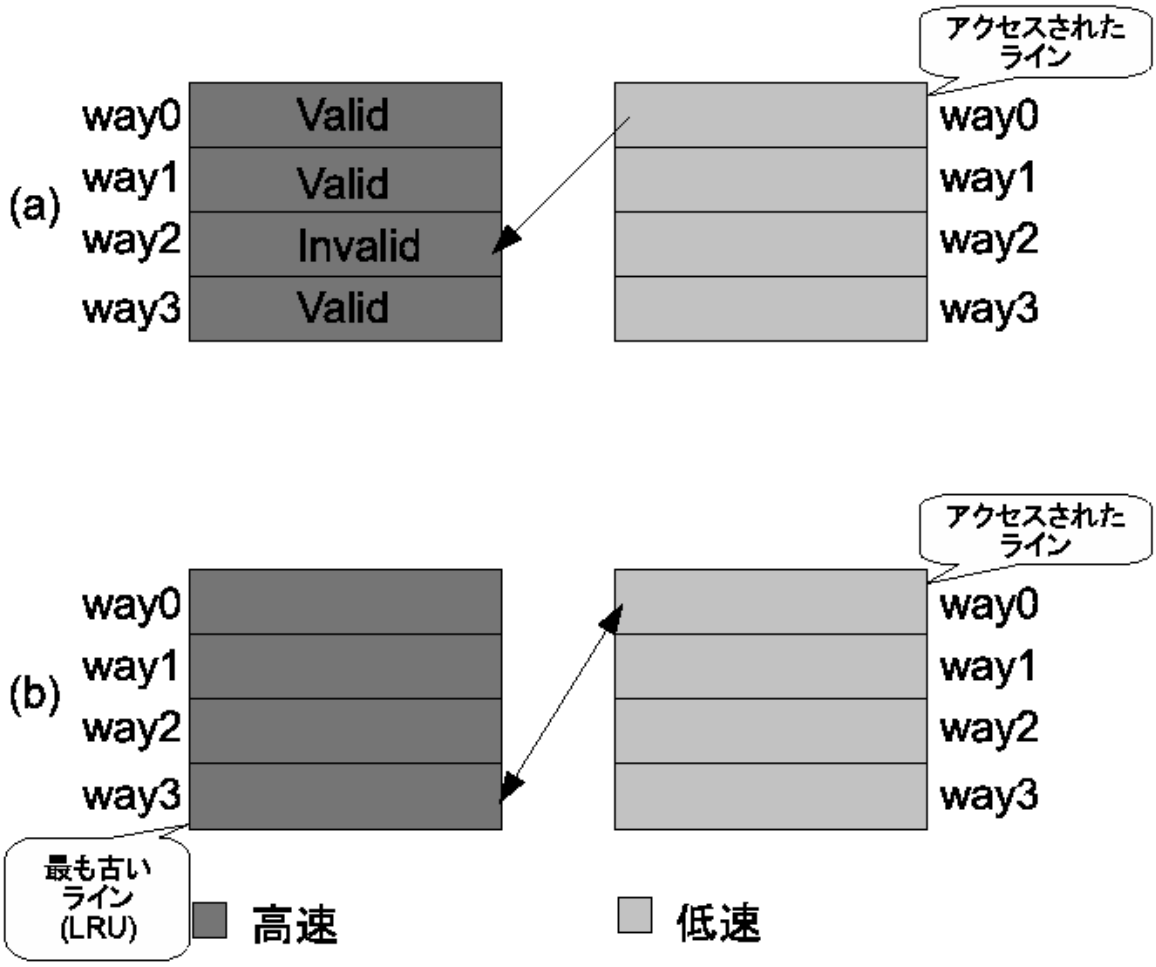

図 13: 時間的局所性に基づいたデータ移動

きいラインを置き換え対象として移動を行う (図 15 の (b)). 高速な領域の同一セットに空きラインがなく，全てのラインの重要度の値が同じである場合は，置き換え対象のラインは LRU で決定する (図 15 の (c)).

1-2 スラックの大きい命令によるアクセスの場合

スラックの大きい命令によるアクセスは時間に余裕があり，アクセスの際の消費電力を削減するためには，低速な領域からデータが供給されることが望ましい．したがって，アクセスされたデータがもともと低速な領域にあった場合はデータの移動を行う必要はないが，高速な領域にあった場合は低速な領域へ移動させた方が良いとも考えられる．しかし，データの移動によるレイテンシや電力のコストは大きく省電効果は期待できないという結果が出されている [1]．よってスラックの大きい命令に

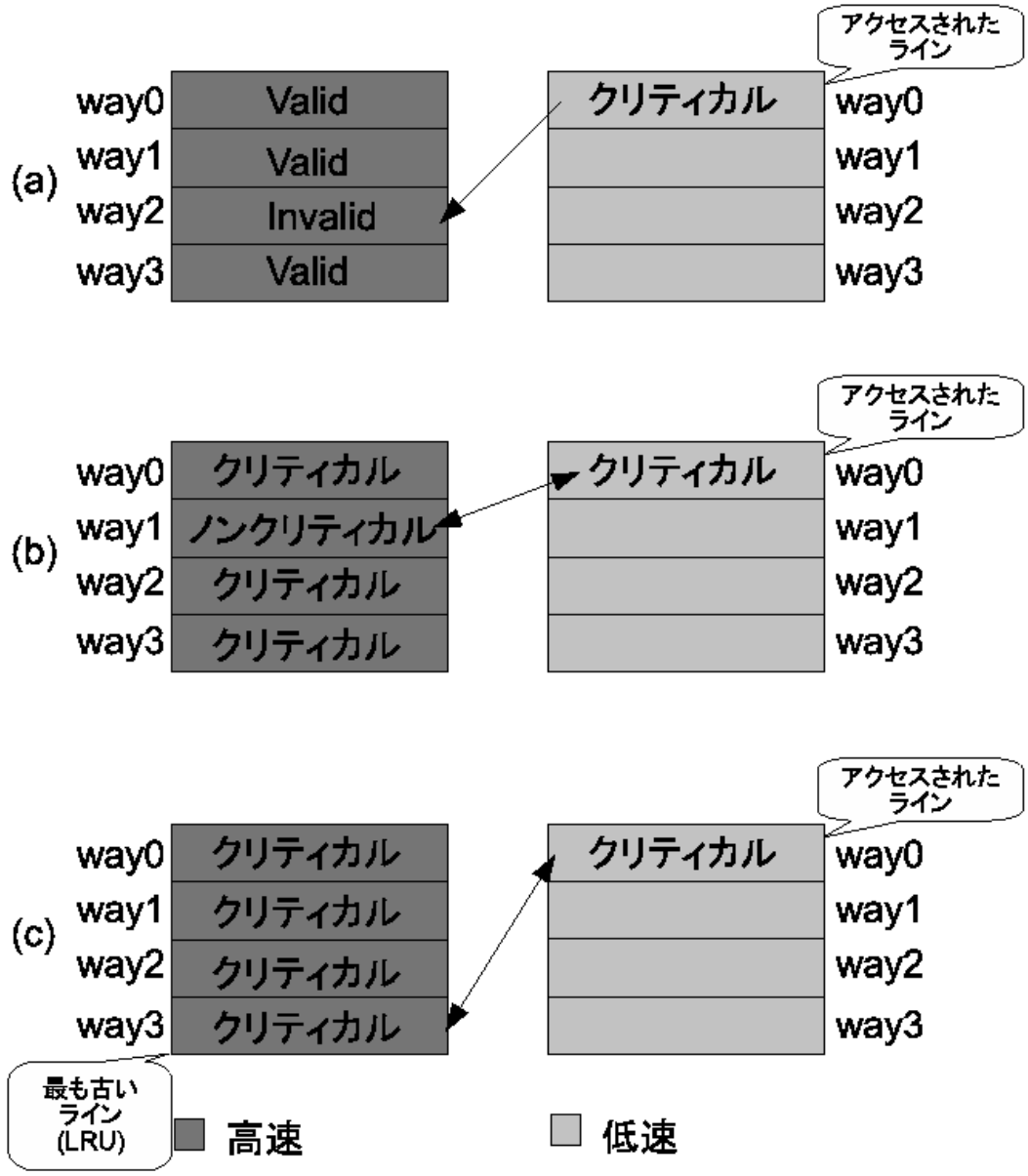

図 14: クリティカルパス情報に基づいたデータ移動

よるアクセスにおいては，データの移動は特に行わない．

2 L1 キャッシュメモリでミスが起こり，L2 キャッシュメモリと L1 キャッシュメモリとの間でデータの移動が必要となる場合について

2-1 スラックの小さい命令のアクセスの場合

スラックの小さい命令のアクセスの場合は，高速な領域からデータが供給されることが理想であるから L2 キャッシュメモリから供給されるデータは高速な領域に配置されるようにする．まず L2 キャッシュメモリから

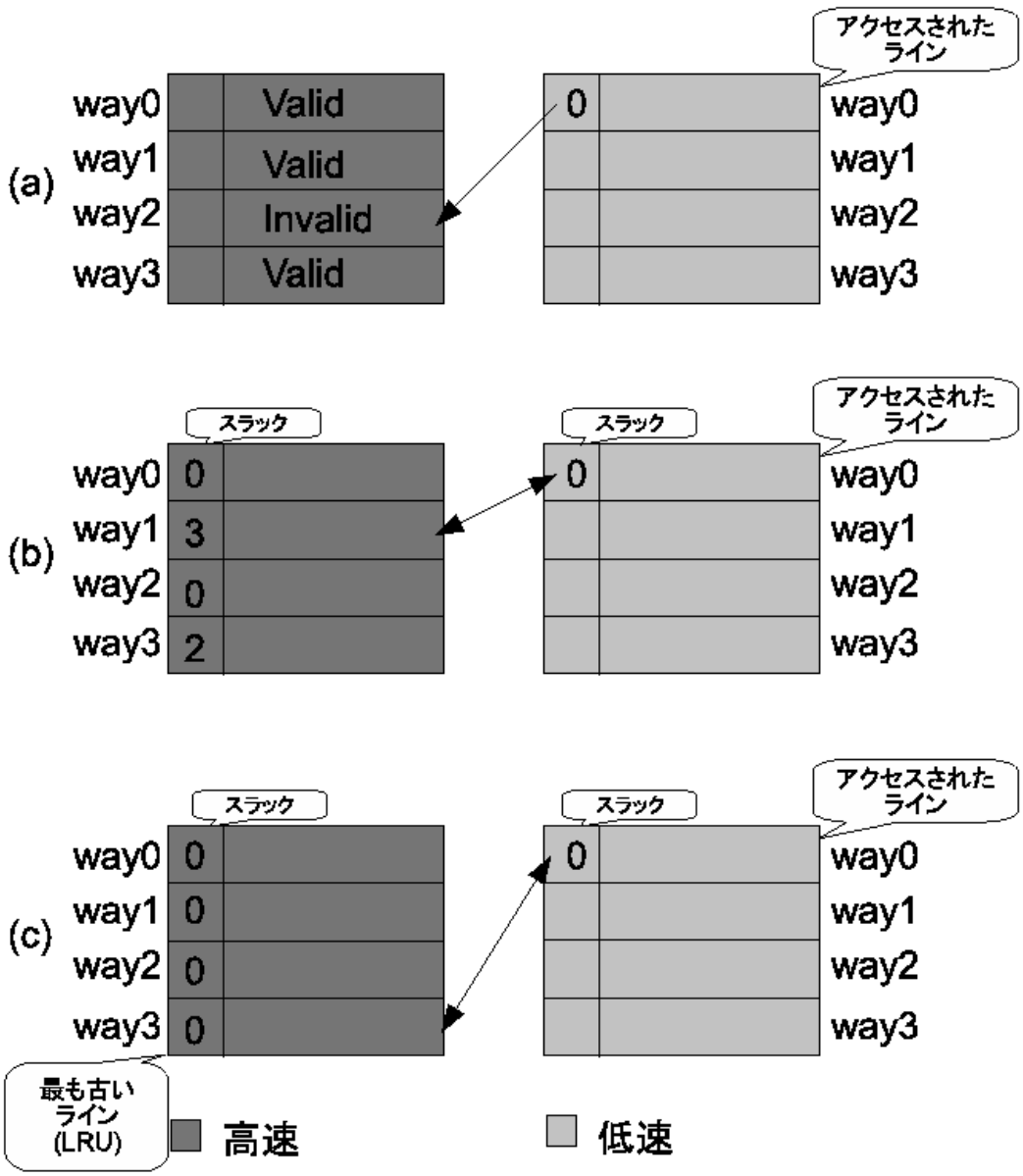

図 15: スラックに基づいたデータ移動

当該アクセスによるラインが読み出され，そのラインを高速な領域に書き込む図 16 の (1)）．高速な領域に空きがなく，当該アクセスによるラインを配置できない場合は，高速な領域から最も重要度の低いラインを低速な領域へ追い出す（図 16 の (2)）．低速な領域に空きがなく，高速な領域から追い出すラインを配置できない場合は，低速な領域から最も重要度の低いラインを L2 キャッシュメモリへ追い出す（図 16 の (3)）．このようにして L2 キャッシュメモリから供給されたデータが高速な領域からプロセッサに供給されるようになる．

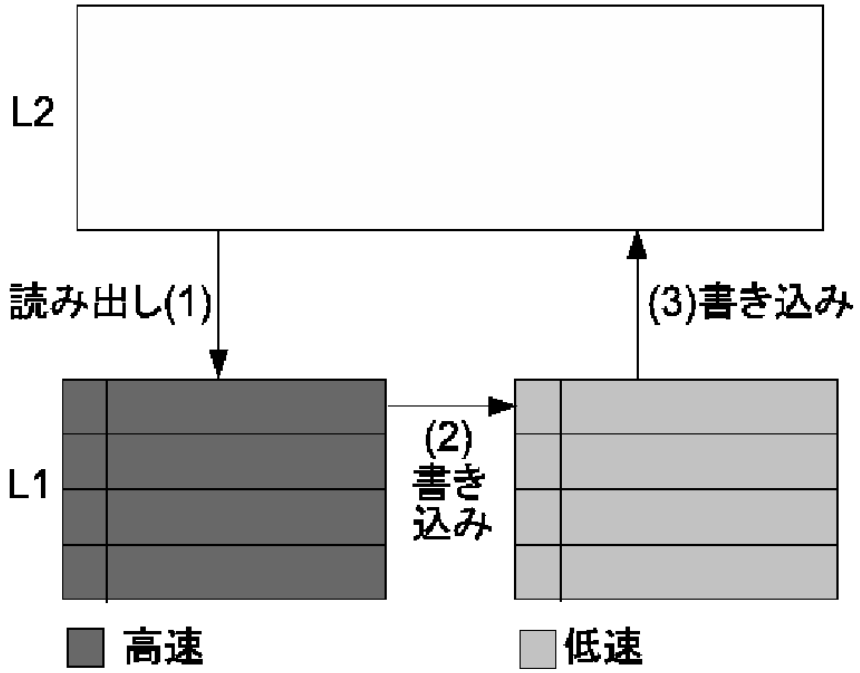

図 16: L1 キャッシュミス時のデータ移動 (1)

2-2 スラックの大きい命令のアクセスの場合

スラックの大きい命令のアクセスの場合は，次のアクセスからは低速な領域からデータが供給されればアクセスの際の電力を削減できるため，L2 キャッシュメモリから供給されるデータは低速な領域に配置されるようにする．まず L2 キャッシュメモリから当該アクセスによるラインが読み出され，そのラインを低速な領域に書き込む（図 17 の (1)）．低速な領域に空きがなく，当該アクセスによるラインを配置できない場合は，低速な領域から最も重要度の低いラインを L2 キャッシュメモリへ追い出す（図 17 の (2)）．このようにして L2 キャッシュメモリから供給されたデータが低速な領域からプロセッサに供給されるようになる．

3 L2 キャッシュメモリでもミスが起こった場合について

L2 キャッシュメモリでもミスが起こった場合を説明する．まず主記憶から当該アクセスによるラインが読み出され，そのラインを L2 キャッシュメモリに書き込む．L2 キャッシュメモリに空きがなく，このラインを配置できない場合，L2 キャッシュメモリから LRU によって決定されたラインを主記憶に追い出す．このようにして主記憶から L2 キャッシュメモリへデータが供給され，

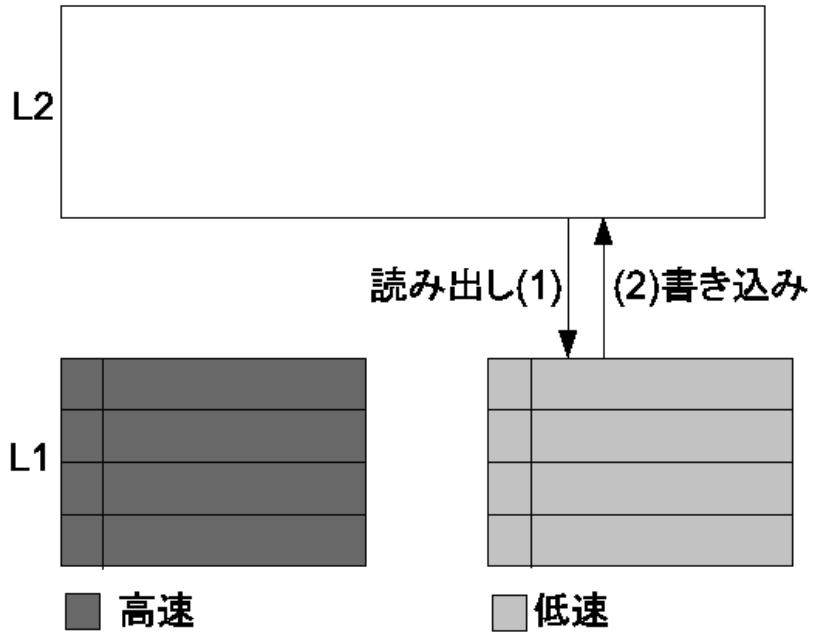

図 17: L1 キャッシュミス時のデータ移動 (2)

その供給されたデータは L1 キャッシュメモリへ供給される．

本研究において，L2 キャッシュメモリには，本研究が L1 キャッシュメモリに対して行っているクリティカルパス情報に基づく方法，及びスラックに基づく方法による重要度決定方法を適用することはできない．よって L2 キャッシュメモリの重要度決定は時間的局所性に基づく方法で行っている．

4.3 データ移動機構の実装

本キャッシュメモリで必要な高速な領域と低速な領域のデータ移動のため，L1 キャッシュメモリの高速な領域と低速な領域間に 2 ラインを保持できるバッファを付け加えラインの移動が必要な際に利用する．今回のアクセスで読み出されたラインはそのバッファに一時保管される．同様に置き換え先のラインをバッファに書き込む．高速な領域と低速な領域は階層化されていないので各領域に同一ラインが存在するとキャッシュコヒーレンシに問題が発生する．よって今回のアクセスによって読み出しがあったラインはバッファから移動先のウェイに書き込みを行うとともに，移動先に選ばれてバッファに書き込まれていたラインを今回のアクセスによって読み出しがあったラインがあったウェイに書き込む．以上でラインの移動が完了する（図

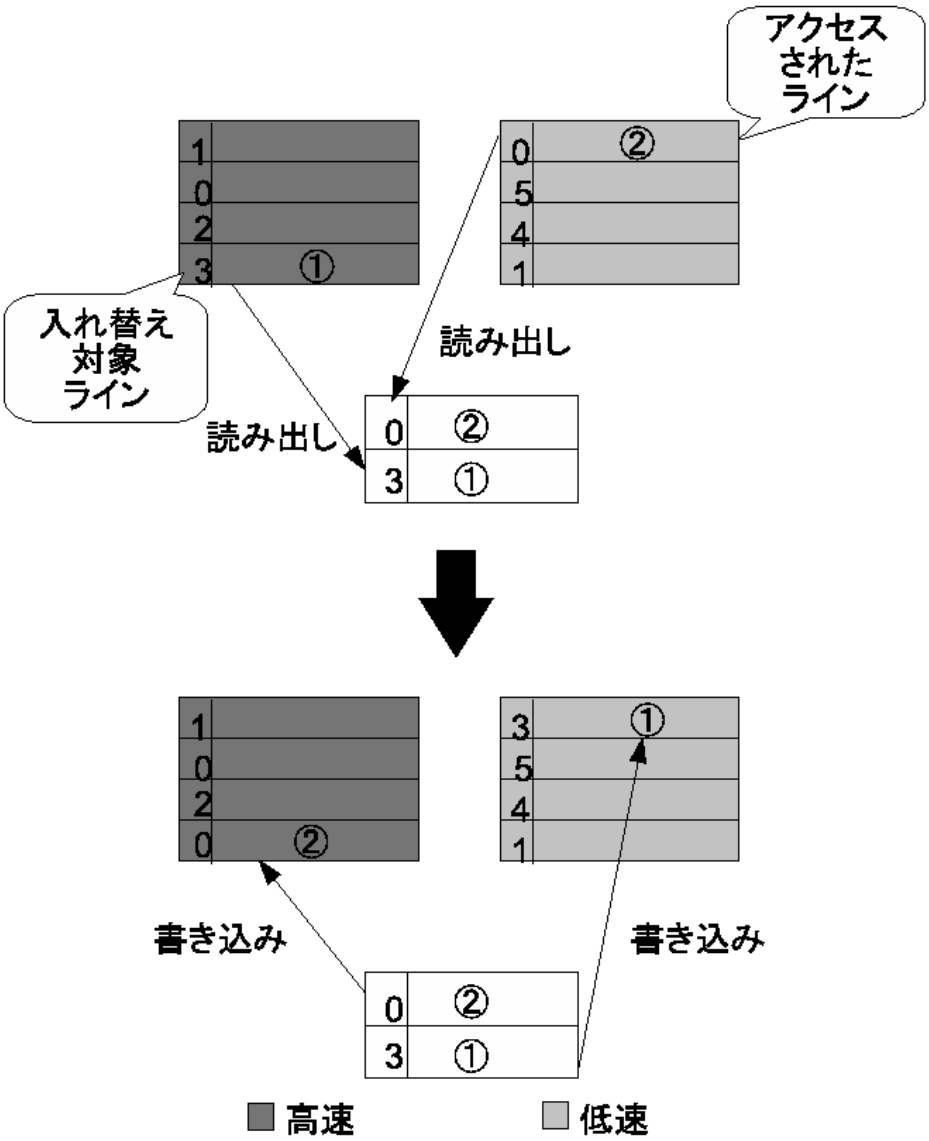

図 18: 高速な領域と低速な領域間でのデータ移動機構

18）.

このラインの移動において，移動が完了するまでは次のメモリアクセス命令はキャッシュメモリへアクセスできないとする.

なお，今回の実装では，低速な領域からラインを移動させる必要があり，高速な領域に空きラインがあって，そこへ移動させるだけでよい場合でも，実装を簡略化したため，移動先となった高速な領域のラインを低速な領域に移動させる仕様になっている.

5 命令のスラック値に基づく各領域への選択的アクセス方法

本研究では，キャッシュメモリへのアクセスはストア命令による Write アクセスとロード命令による Read アクセスと想定している．このうちロード命令による Read アクセスは予測器から得られる命令のスラックの値を利用して高速な領域と低速な領域のどちらにアクセスするかを選択することができる．スラックを用いてデータの重要度を決定している場合，高速な領域にはスラックの小さい命令によってアクセスされたデータが集まり，低速な領域にはスラックの大きい命令によってアクセスされたデータが集まると期待できる．この性質を利用してロード命令によるアクセスは命令のスラックの値に応じてアクセスする領域を決めることが有効と考えられる．

スラックが大きい命令であるか，スラックが小さい命令であるかの判定は，判定の基準となる値を設定して，基準値以下ならスラックの小さい命令，基準値より大きければスラックの大きい命令と判定する．例えば基準値を 2 に設定すると命令のスラックが 2 以下である命令をスラックの小さい命令であると判定し，命令のスラックが 2 より大きい命令をスラックの大きい命令であると判定する．

この章では各アクセスの手順を説明する．また，アクセス中に 4 章で述べたラインの移動が発生する場合がある．ロード命令によるアクセスの場合はラインの移動が発生する場合の動作も含めてデータがプロセッサへ供給されるまでの手順を説明し，ストア命令によるアクセスの場合はデータが L1 キャッシュに書き込まれるまでの手順を説明する．

5.1 スラックの小さいロード命令のアクセス

ロード命令によるアクセスの内，スラックの小さい命令のアクセスの手順を説明する．スラックの小さい命令のデータは高速な領域に集まりやすく，かつ高速な領域でヒットすれば実行時間を短くできる．そこで，まず高速な領域へアクセスすることとする (図 19 の (1))．

高速な領域でヒットした場合は，高速な領域から高速にデータを読み出すことができる．しかし高速な領域に該当するデータがなかった場合は，低速な領域へアクセスすることとする (図 19 の (2))．低速な領域にも該当するデータがなかった場合は L2 キャッシュメモリへアクセスする (図 19 の (3))．L2 キャッシュメモリにも該当するデータがなかった場合は，主記憶へアクセスする (図 19 の (4))．

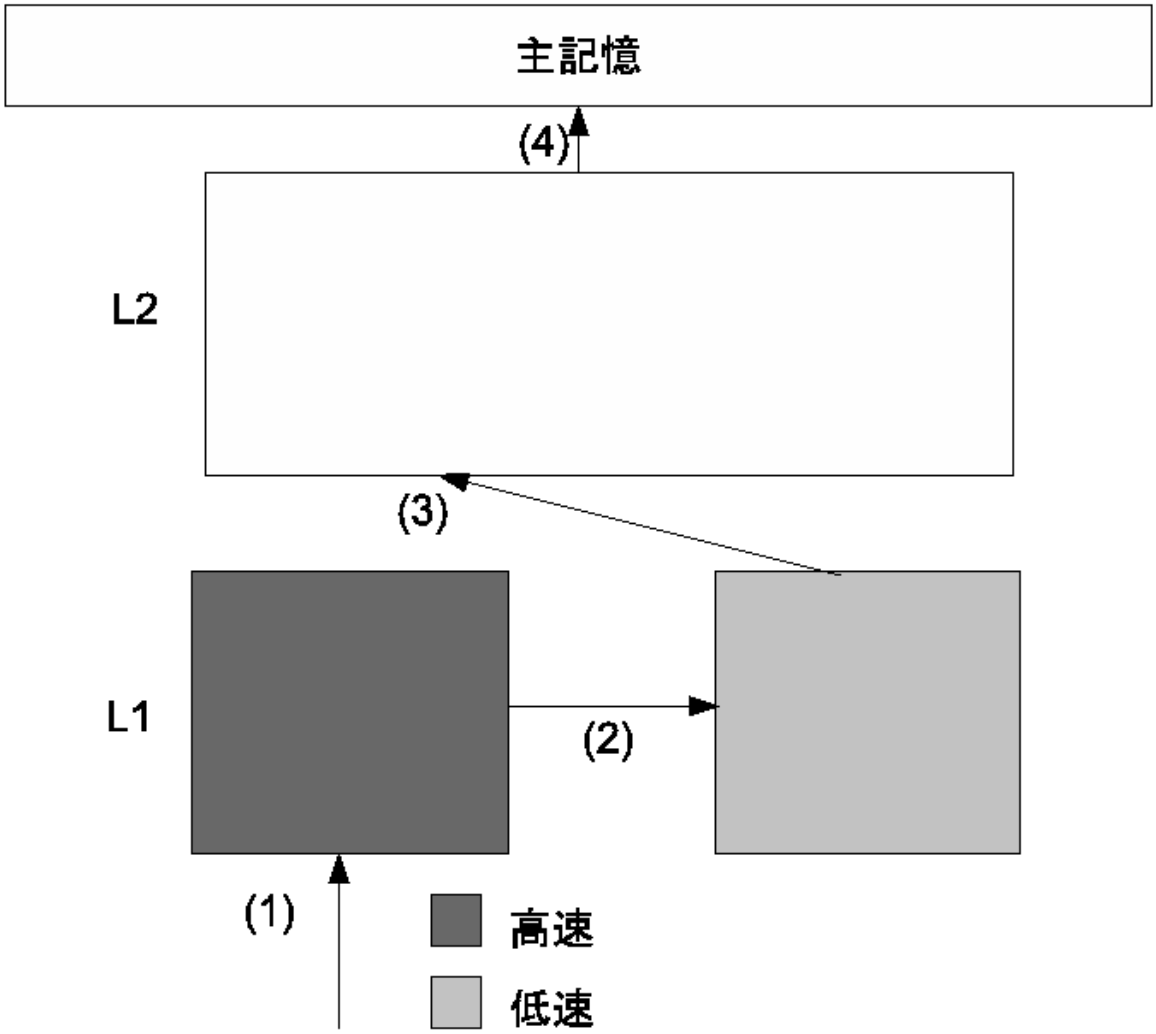

図 19: スラックの小さいロード命令のアクセス手順

低速な領域でヒットした場合は，4.2 節で述べた高速な領域と低速な領域との間でデータの移動を行うこととする (図 13，14，15)．また，L1 キャッシュメモリでミスして L2 キャッシュメモリへのアクセスが必要となった場合は，高速な領域からプロセッサへデータを供給できるようにするため，高速な領域と L2 キャッシュメモリとの間で 4.2 節で述べたデータの移動が行われる (図 16)．L2 キャッシュメモリでミスをして主記憶へのアクセスが必要となった場合は，L2 キャッシュメモリから高速な領域へデータを供給できるようにするため，L2 キャッシュメモリと主記憶との間で 4.2 節で述べたデータの移動が行われる．

スラックの小さい命令のアクセスのキャッシュメモリにおける動作手順を図 20 のフロチャートに示す．なお図 20 では，データの移動に関する詳細は省略している．

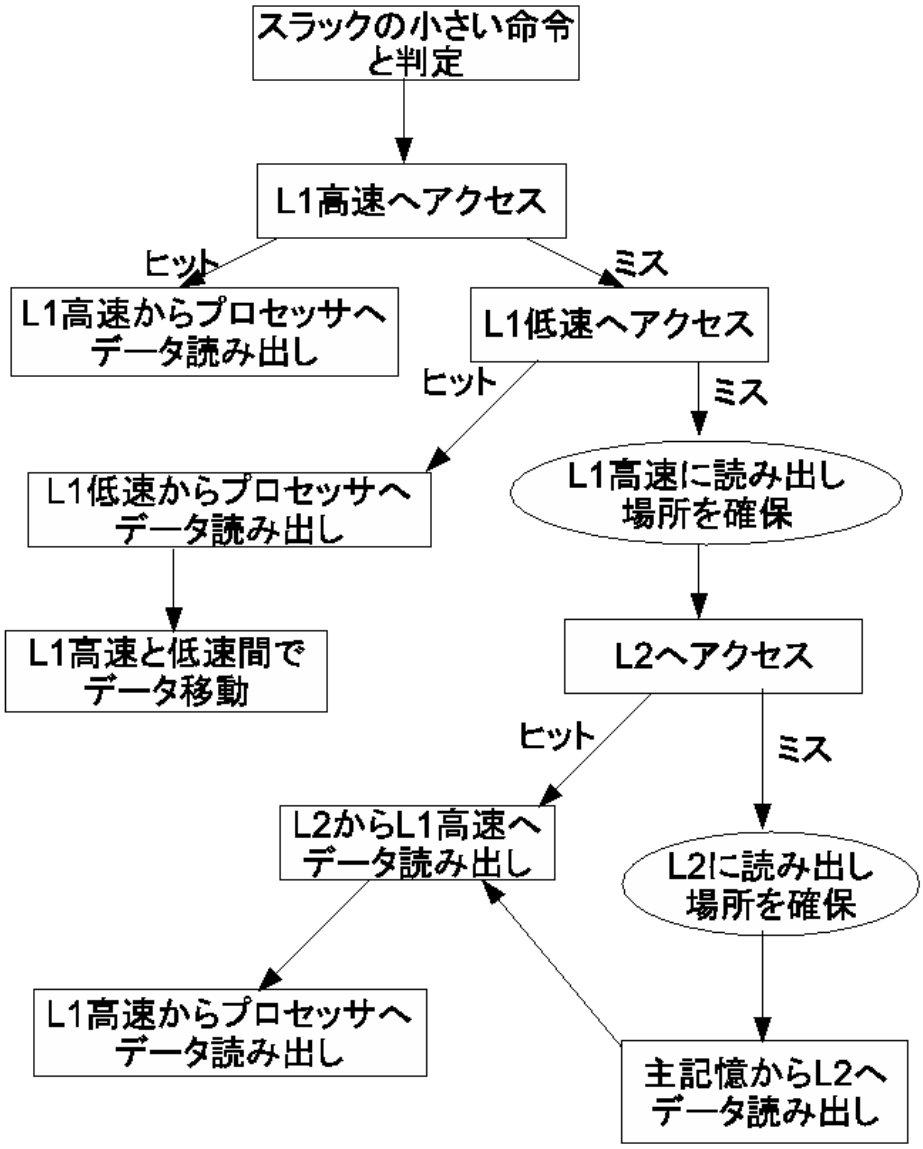

図 20: スラックの小さいロード命令のアクセスの動作手順

5.2 スラックの大きいロード命令のアクセス

ロード命令によるアクセスの内，スラックの大きい命令のアクセスの手順を説明する．スラックの大きい命令のデータは低速な領域に集まりやすく，またアクセスに時間的余裕があるので，低速な領域で読み出しができればアクセスの際の消費電力を削減することができると考えられる．よってまず低速な領域へアクセスすることとする (図 21 の (1))．該当するデータが低速な領域になかった場合は，高速な領域にアクセスする (図 21 の (2)) 高速な領域にも該当するデータがなかった場合は L2

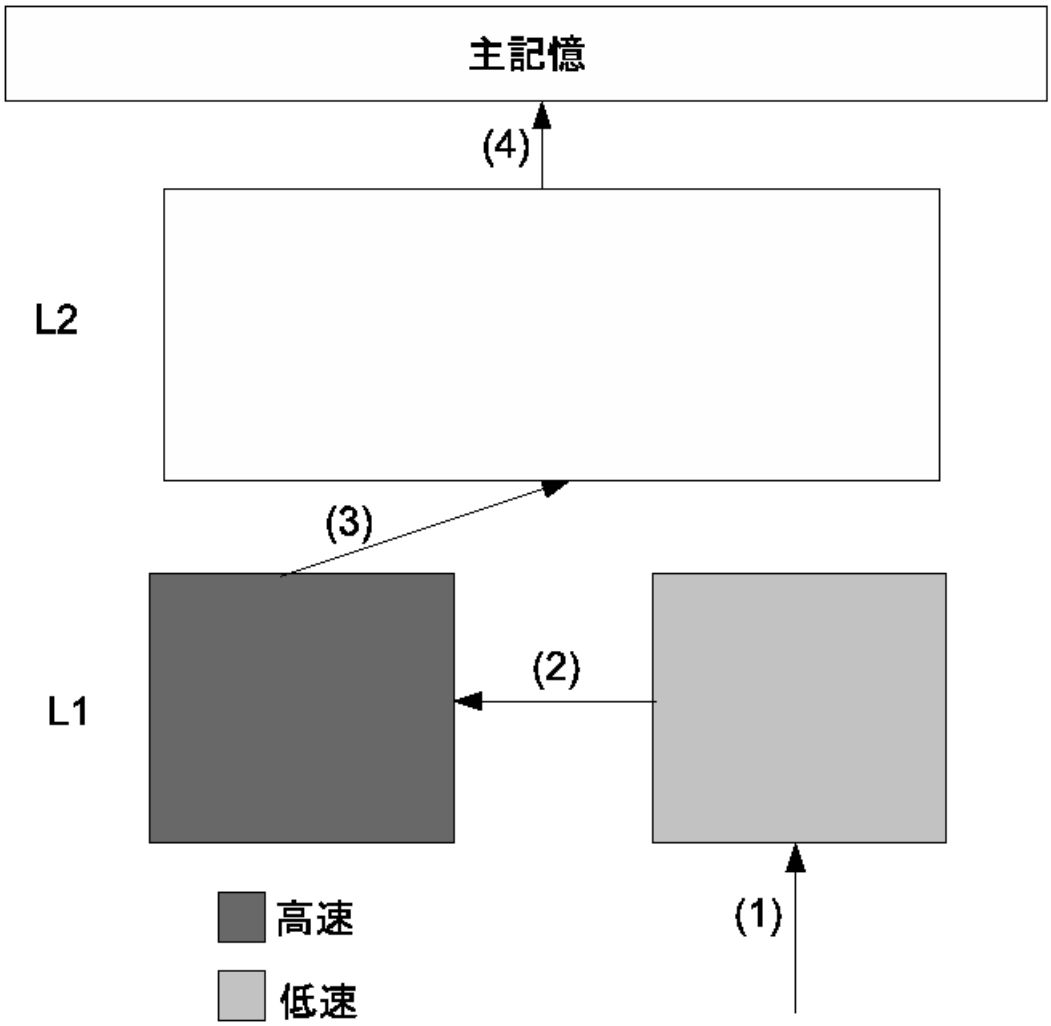

図 21: スラックの大きいロード命令のアクセス手順

キャッシュメモリにアクセスする (図 21 の (3))．L2 キャッシュメモリにも該当するデータがなかった場合は，主記憶へアクセスする (図 21 の (4))．

高速な領域と低速な領域との間では特にデータの移動は行わないこととするが，L1 キャッシュメモリでミスをして，L2 キャッシュメモリへのアクセスが必要となった場合は，低速な領域からデータを供給できるようにするため，低速な領域と L2 キャッシュメモリとの間で 4.2 節で述べたデータ移動が行われる (図 17)．L2 キャッシュメモリでミスをして主記憶へのアクセスが必要となった場合は，L2 キャッシュメモリから低速な領域へデータを供給できるようにするため，L2 キャッシュメモリと主記憶との間で 4.2 節で述べたデータ移動が行われる．

スラックの大きい命令のアクセスのキャッシュメモリにおける動作手順を図 22 の

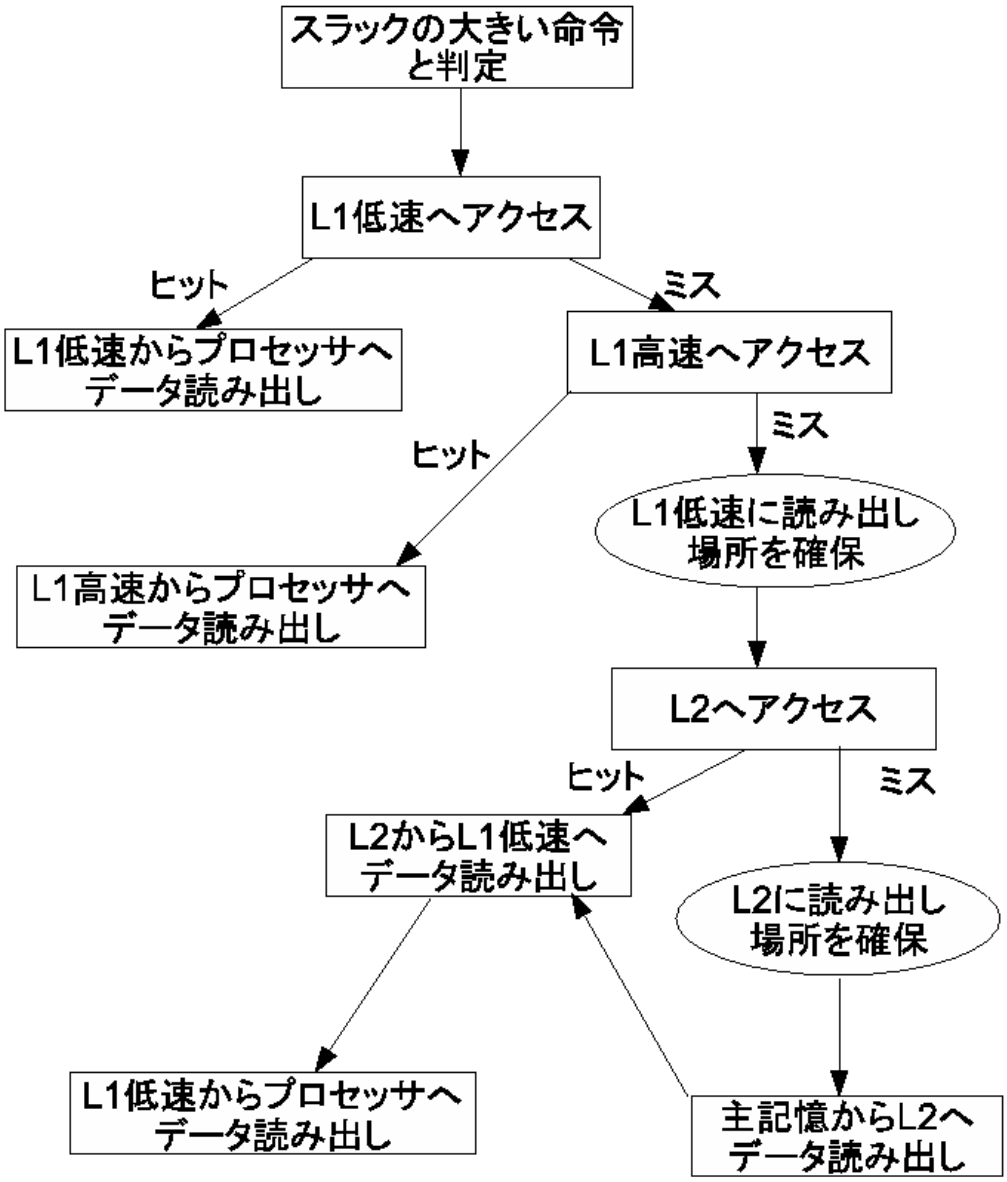

図 22: スラックの大きいロード命令のアクセスの動作手順

フロチャートに示す．なお図 22 では，データ移動に関する詳細は省略している．

5.3 ストア命令のアクセス

ストア命令のアクセスの手順を説明する．

初めに高速な領域にアクセスをする（図 23 の (1)）．ヒットすればプロセッサからのデータの書き込みを高速な領域に行う．高速な領域でミスをした場合は，低速な領域にアクセスする（図 23 の (2)）．ヒットすればプロセッサからのデータの書

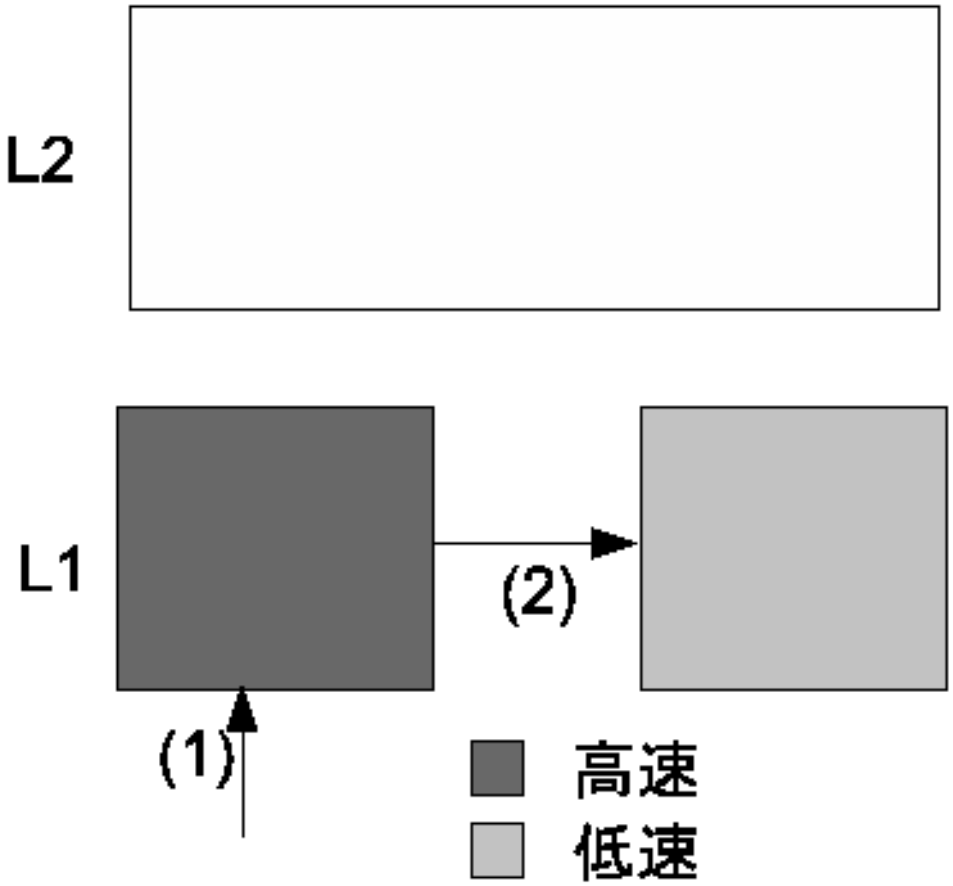

図 23: ストア命令のアクセス手順

き込みを低速な領域に行う．

L1 キャッシュでミスをした場合に L1 キャッシュにプロセッサからのデータを書き込む場所を確保する必要がある．本研究では書き込み場所は高速な領域に確保する．この場合，書き込むラインを L2 キャッシュメモリから高速な領域へ読み出すことが必要となる．当該ラインを L2 キャッシュメモリから高速な領域へ読み出す際に発生するデータの移動は，4.2 節で述べた図 16 の通りに行う．この L2 キャッシュメモリへのアクセスでミスをして，主記憶へのアクセスが必要となった場合は，L2 キャッシュメモリから当該アクセスで必要な書き込むラインを高速な領域へ供給できるようにするために，L2 キャッシュメモリと主記憶との間で 4.2 節で述べたデータの移動が行われる．

こうして高速な領域に書き込むラインが読み出された後で，高速な領域にプロセッサからのデータの書き込みを行う．

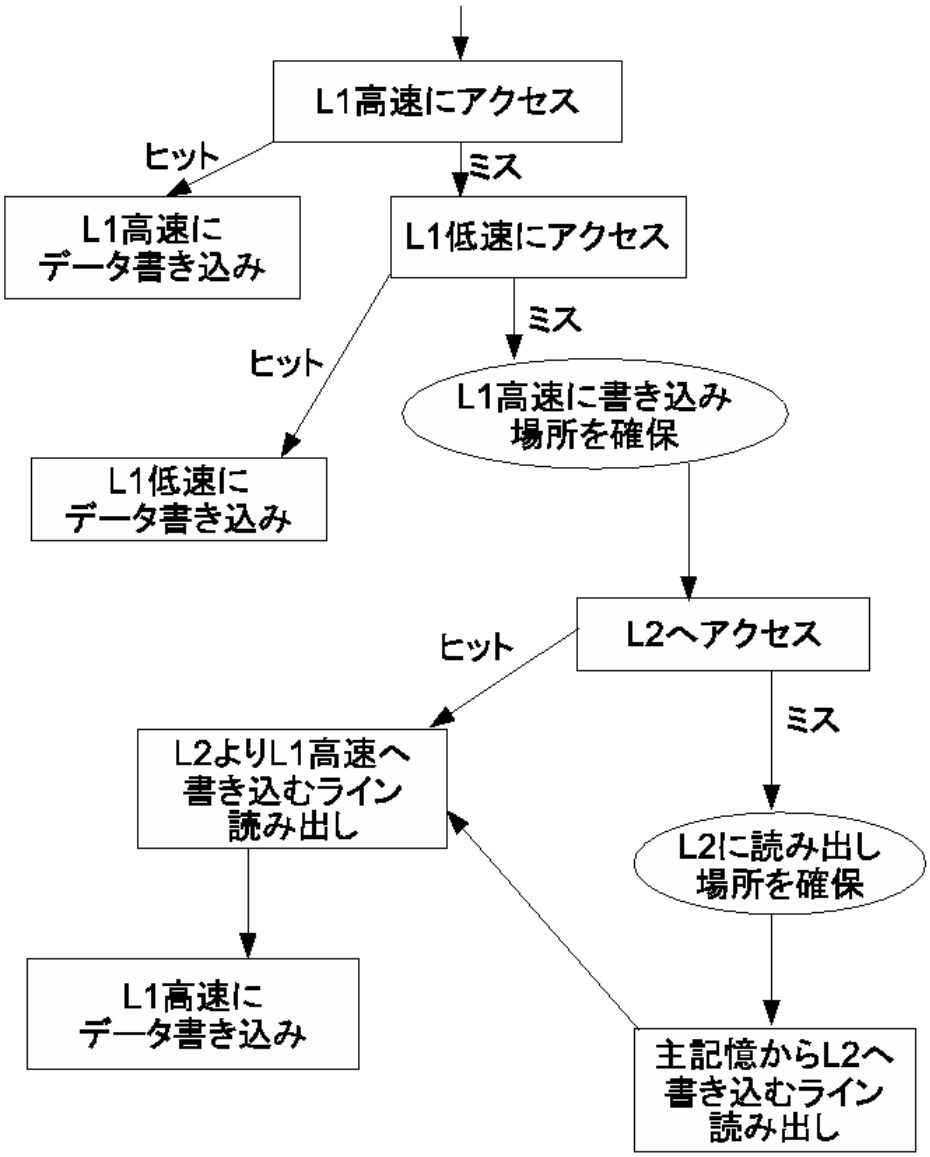

図 24: Write アクセスの動作手順

低速な領域でヒットした場合には，高速な領域へデータの移動をした方が良いとも考えられるが，もし低速な領域でヒットしたデータが重要なデータなのであれば，後にスラックの値の小さいロード命令においてデータの移動が発生することになるので，この時点では高速な領域への移動は行わないこととする．

Write アクセスのキャッシュにおける動作手順を図 24 のフロチャートに示す．なお図 24 では，データの移動に関する詳細は省略している．

表 1: プロセッサの構成

Clock Frequency	6 GHz
Fetch Bandwidth	4 instruction
Branch Predictor	512B-set 4-way set-associative BTB, 2K-entry 8-history-length gshare predictor, 8-entry return address stack, 3-cycle miss penalty, update at commit stage
Insn. Windows	16-entry instruction queue, 8-entry load/store queue
Issue Width	4 instruction
Commit Width	4 instruction
Function Units	iALU 4, iMULT/DIV 1, fpALU 4, fpMULT/DIV/SQRT 1
Insn. Cache	16KB, 4way, 32B blocks, 1-cycle hit latency
Data Cache	16KB, 8way, 32B blocks, 8-port, write-back, non-blocking load, 3-cycle fast hit latency, 5-cycle slow hit latency, 8-cycle swap penalty
L2 Cache	unified, 1MB, 8-way, 64B blocks, 18-cycle hit latency
Main memory	80-cycle latency

6 評価

6.1 評価する環境

SimpleScalar Tool Set[5] を用いて，本研究が提案するデータ移動機構，及びキャッシュへのアクセス機構を省電力キャッシュメモリに組み込んだシミュレーション環境を構築して評価を行った．評価に用いるプロセッサの構成を表 1 に示す．

高速な領域と低速な領域との間でデータの移動を行う場合には，高速・低速な領域へのアクセス時間を足した 8 サイクルをデータ移動のペナルティとして要することとした．

使用する命令セットは *Alpha* 命令セットであり，使用するベンチマークプログラムは SPEC 2000 CINT の中の 7 本で，それぞれの入力及び実行命令数は表 2 に示す通りである．

本研究では提案手法の適用範囲である L1 キャッシュを評価の対象とする．評価には，プロセッサの処理性能，キャッシュメモリの消費エネルギー量，エネルギー遅

表 2: ベンチマークプログラム

ベンチマーク	入力セット	実行命令数
gzip	input.compressed 2	3.3G
vpr	net.in arch.in	1.6G
gcc	cccp.i	2.0G
bzip	input.random 2	8.8G
parser	test.in	4.2G
vortex	lendian.raw	9.8G
mcf	inp.in	0.3G

表 3: キャッシュ容量 8KB におけるアクセスあたりのエネルギーとリーク電流

	access(pJ)	leak(mW)
Fast	11.83	6.60
Slow	9.78	4.44
Swap	43.21	-

延二乗積 (Energy-Delay Square Product: ED^2P)[19] を用いる．

キャッシュメモリの消費エネルギー量の求め方は，高速・低速な領域がそれぞれ何回ずつ使用されたかをカウントしてその回数とそれぞれのアクセスごとに消費されるエネルギーの積からアクセスによるエネルギーを求めることとし，プログラムの実行時間からリーク電流によるエネルギーを求めることとした．アクセスあたりのエネルギーとリーク電流は文献 [1] を参考に表 3 の通りに定める．

ED^2P はパワー削減による遅延の増加と省電力効果のトレードオフを定量的に評価するために用いられる指標である．電圧制御を用いて省電力化を試みる場合の評価に適している．ED^2P はキャッシュメモリのエネルギー消費量とプログラム実行時間から求めることとした．以上述べた評価指標を用いて評価を行った．

初めに，キャッシュメモリを高速な領域と低速な領域に分割する場合と分割しない場合とを比較して，キャッシュメモリを分割することによる省電力の効果を確認した．

次に，データの重要度の決定方法の違いによる比較を行い，データの重要度の決定に命令のスラックを用いた場合の省電力の効果を評価した．

最後に，キャッシュへのアクセスの際，スラックの小さい命令か大きい命令かを

判定する基準値を変化させた場合についての比較を行い，最適な基準値を調べるとともに省電力の効果を評価した．

6.2 領域分割の有無の違いによる評価

初めに，L1 キャッシュメモリを高速な領域と低速な領域に分割しない場合と分割を行った場合とを比較して，領域分割による消費電力削減の効果が認められるかを評価する．L1 キャッシュメモリを分割しない場合は，高速な領域のみで L1 キャッシュメモリを構成する場合と，低速な領域のみで L1 キャッシュメモリを構成する場合を考える．なお，領域分割を行う場合，領域分割を行わない場合どちらでも L1 キャッシュメモリ全体での総容量とウェイ数は同じとする．

領域分割を行わないキャッシュメモリは，一般的なキャッシュメモリと同様に L1 キャッシュメモリ対して全領域をアクセスする．

一方，領域分割を行う場合，命令の重要度の決定は，スラックの小さい命令は高速な領域から，スラックの大きい命令は低速な領域からアクセスを行うこととし，この評価実験では，命令のスラックが 0 の命令をスラックの小さい命令とし，命令のスラックが 1 以上の命令をスラックの大きい命令とした．また，データの重要度の決定は，スラックを用いて行うこととした．

6.2.1 プロセッサの処理性能の結果

L1 キャッシュメモリの分割の有無の違いによる IPC の結果を図 25 に示す．グラフの横軸はベンチマークごととその平均を表しており，fast は領域分割を行わずに、高速な領域のみで L1 キャッシュメモリを構成した場合の結果を表し，slow は領域分割を行わずに、低速な領域のみで L1 キャッシュメモリを構成した場合の結果を表している．また，slack は領域分割を行って，データの重要度の決定にスラックを用いた場合を表している．縦軸は処理性能の大きさを表しており，グラフの値が大きいほど処理性能が高いことを示している．なお，グラフは各ベンチマークごとに fast の場合で正規化されている．

全てのベンチマークにおいて，fast が最も良い結果となり，slow が最も悪い結果となった．slack はラインの移動によるペナルティが存在し，移動が頻発する場合では，移動の起こらない slow より処理性能が悪くなることも考えられたが，slow よりは良い結果を示すことができた．ラインの移動によるペナルティは処理性能に致命的になるほど悪影響は及ぼしていないといえる．

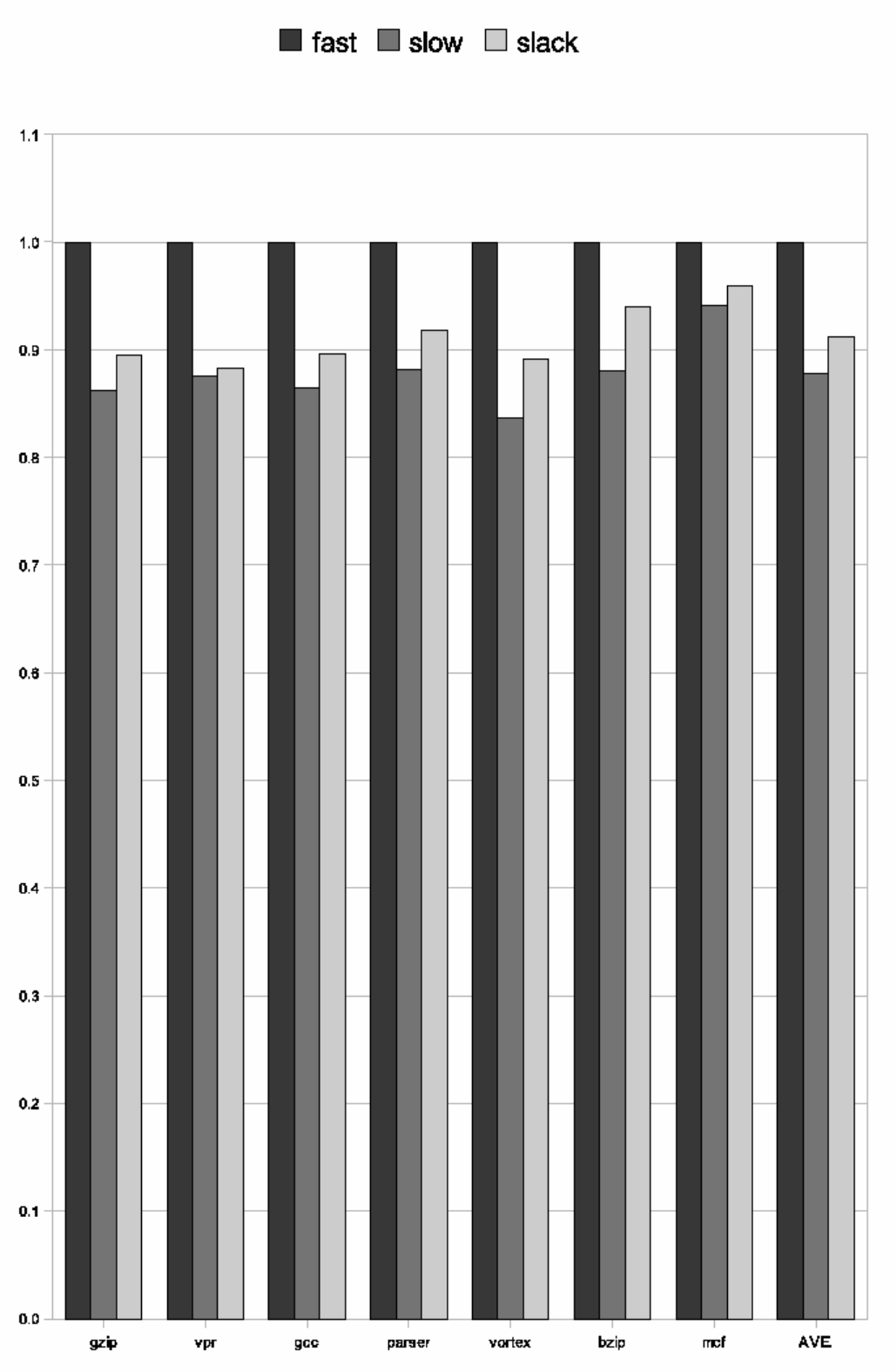

図 25: L1 キャッシュ分割の有無の違いによる IPC の結果

6.2.2 消費エネルギー量の結果

L1 キャッシュメモリの分割の有無の違いによる消費エネルギー量の結果を図 26 に示す．グラフの横軸はベンチマークごとに，データの重要度決定方法を変えた場合の結果を表し，縦軸は消費エネルギーの総量と，内訳を表している．access_fast は高速な領域でのアクセスによって消費したエネルギー量，access_slow は低速な領域でのアクセスによって消費したエネルギー量，swap はラインの移動に消費したエネルギー量を，leak はリーク電流によって消費したエネルギー量をそれぞれ表している．グラフは各ベンチマークごとに fast の場合で正規化されており，グラフの値が大きいほど消費電力が大きいことを示している．

fast，slow のどちらも領域全体へアクセスを行っているため 1 回のアクセスによる消費エネルギーが大きく，領域分割を行って分割した領域に対して順番にアクセスを行う slack に比べて，アクセスによって消費するエネルギーが大きくなっている．特に，slack は消費エネルギー量合計でも，slow より消費する電力を抑えることに成功しており，領域分割を行うことでアクセスによる消費エネルギーを大きく削減できているといえる．

6.2.3 エネルギー遅延二乗積の結果

L1 キャッシュメモリの分割の有無の違いによるエネルギー遅延二乗積 (ED^2P) の結果を図 27 に示す．グラフの横軸はベンチマークごとに，データの重要度決定方法を変えた場合の結果を表し，縦軸はエネルギー遅延二乗積を表している．グラフは各ベンチマークごとに fast の場合で正規化されており，グラフの値が小さいほど省電効果が高いことを示している．

slow は消費エネルギーは fast より小さかったが処理性能の低下が大きく，ED^2P では fast より悪い結果となっている．また，slack は，消費エネルギー量が小さかったので，fast より処理性能が悪くとも，ED^2P では良い結果を示し，領域分割によって省電効果が見込めることを示す結果となった．

6.3 データの重要度決定方法の違いによる評価

次に，データを配置する領域の選択において，データの重要度として時間的局所性を用いた場合，クリティカルパス情報を用いた場合，スラックを用いた場合の 3 つの場合についての重要度の決定方法の違いによる評価を行った．

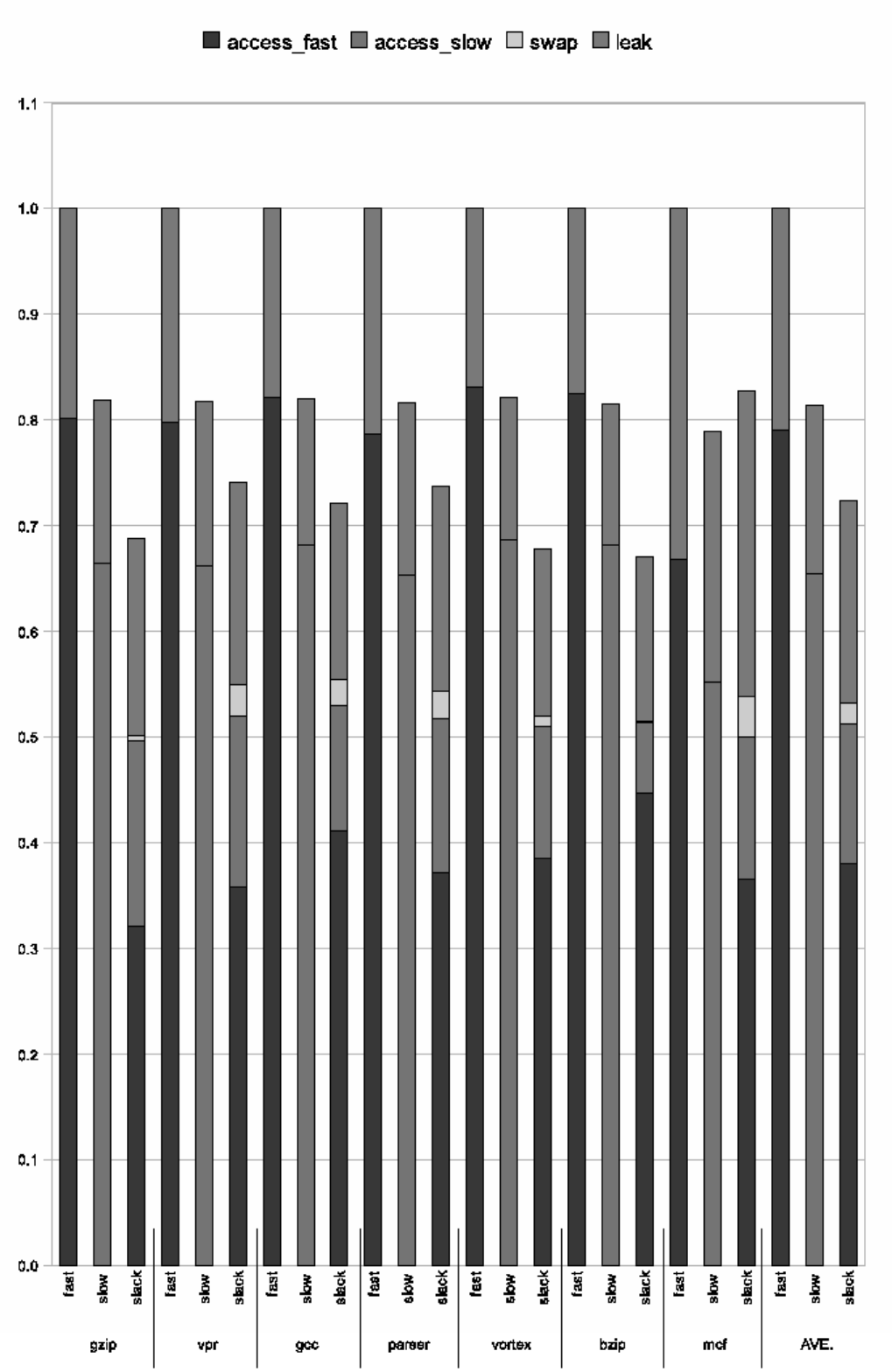

図 26: L1 キャッシュ分割の有無の違いによる消費エネルギー量の結果

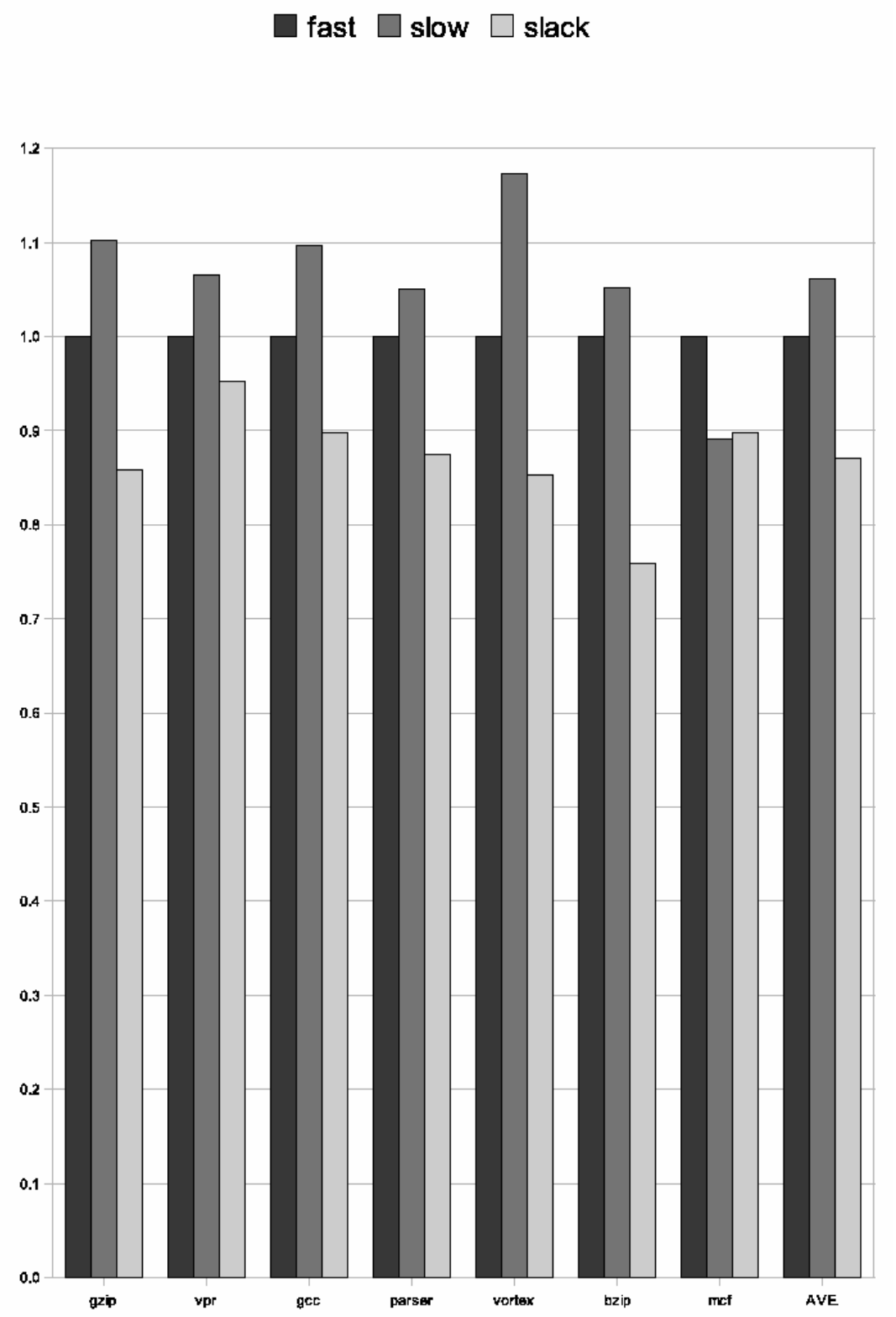

図 27: L1 キャッシュ分割の有無の違いによるエネルギー遅延二乗積の結果

アクセスする各領域の選択については，判定基準の値を 0 として，命令のスラックの値が 0 以下の命令は高速な領域からアクセスを行い，命令のスラックの値が 1 以上の命令は，低速な領域からアクセスを行うとした．

6.3.1 プロセッサの処理性能の結果

データの重要度の決定方法ごとの処理性能の結果を図 28 に示す．グラフの横軸はベンチマークとその平均を表しており，”LRU”はデータの重要度を LRU に基づき決定した場合，critical は重要度をクリティカルパス情報に基づき決定した場合，slack は重要度をスラックに基づき決定した場合の結果を表している．また，縦軸は処理性能の大きさを表している．グラフは各ベンチマークごとに LRU を用いた場合で正規化されており，グラフの値が大きいほど処理性能が高いことを示している．

図 29 に重要度の決定方法ごとにキャッシュアクセス命令が L1 キャッシュメモリのどの領域をアクセスしたかの内訳も示す．グラフの横軸は処理性能の結果と同様である．縦軸は，キャッシュへアクセスした割合を示している．fast_hit は高速な領域でキャッシュヒットする割合，fast_miss は高速な領域でキャッシュミスをする割合，slow_hit は低速な領域でキャッシュヒットする場合，slow_miss は低速な領域でキャッシュミスをする割合をそれぞれ表している．

まず，LRU と slack を比較する．gzip，gcc，vortex，bzip のベンチマークにおいて，slack の方が良い結果が得られた．これは，命令のスラックに基づくアクセスと，スラックに基づくデータ配置が上手くかみ合って，命令のスラックが 0 である命令が高速な領域でヒットする割合が増えてプログラムの実行が遅れにくかったためと考えられる．一方で vpr，parser，mcf のベンチマークでは，slack の方が悪い結果となった．これら 3 つのベンチマークでは，L1 キャッシュメモリ全体のキャッシュミス率が他のベンチマークに比べて大きく，スラックに基づくデータ配置を行っても，命令のスラックが 0 である命令が高速な領域でヒットする割合は増なかったため LRU より悪い結果になった．

次に，critical と slack を比較する．mcf 以外のベンチマークで slack の方が良い結果が得られた．ラインの移動が発生した場合に，移動先のラインとして重要度の値の大きいラインを選ぶことのできる slack の方が，より重要なラインを高速な領域に残すことができ，命令のスラックが 0 の命令が高速な領域でヒットする割合が大きくなっているためであると考えられる．

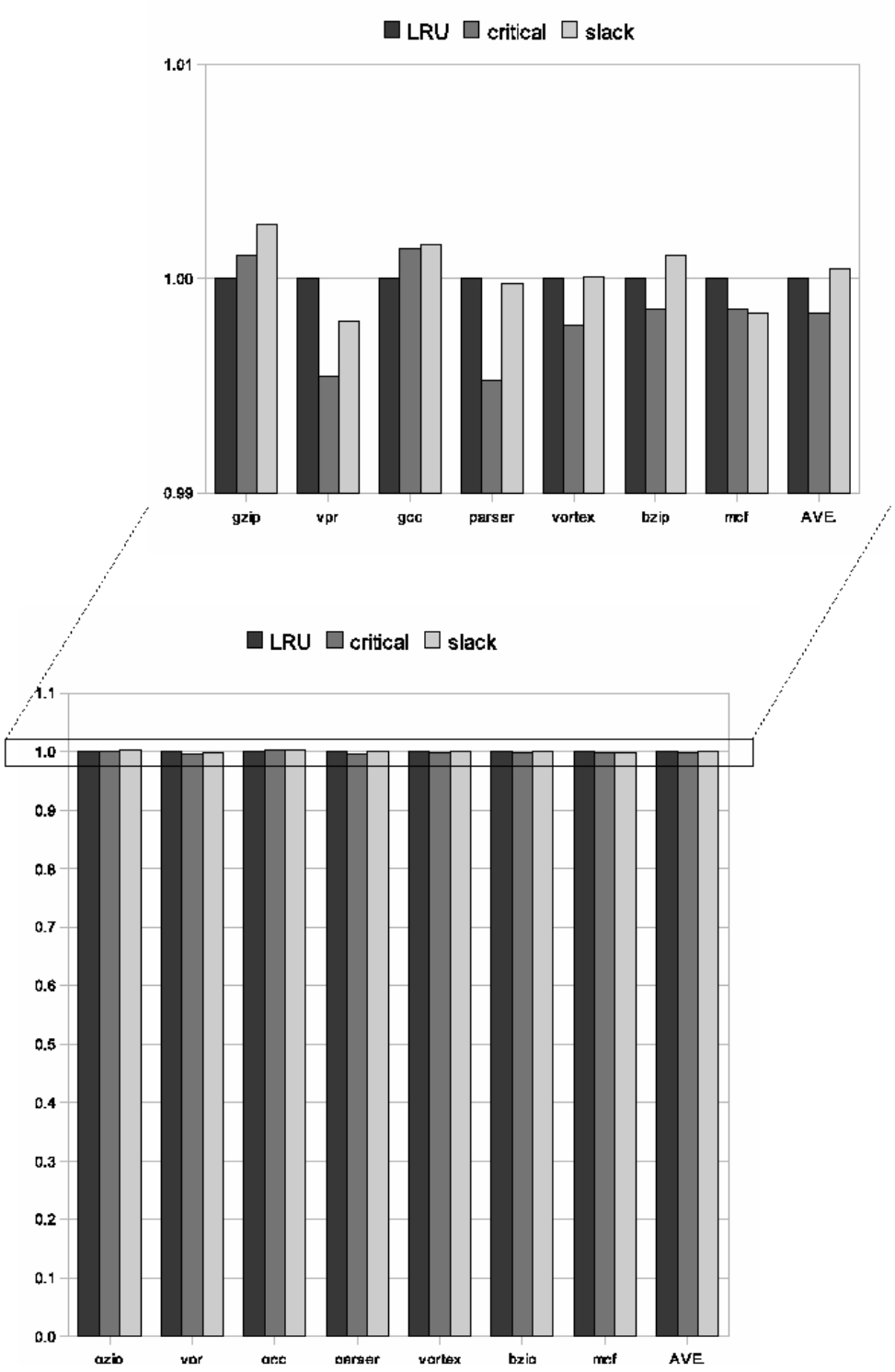

図 28: データの重要度決定方法ごとの IPC

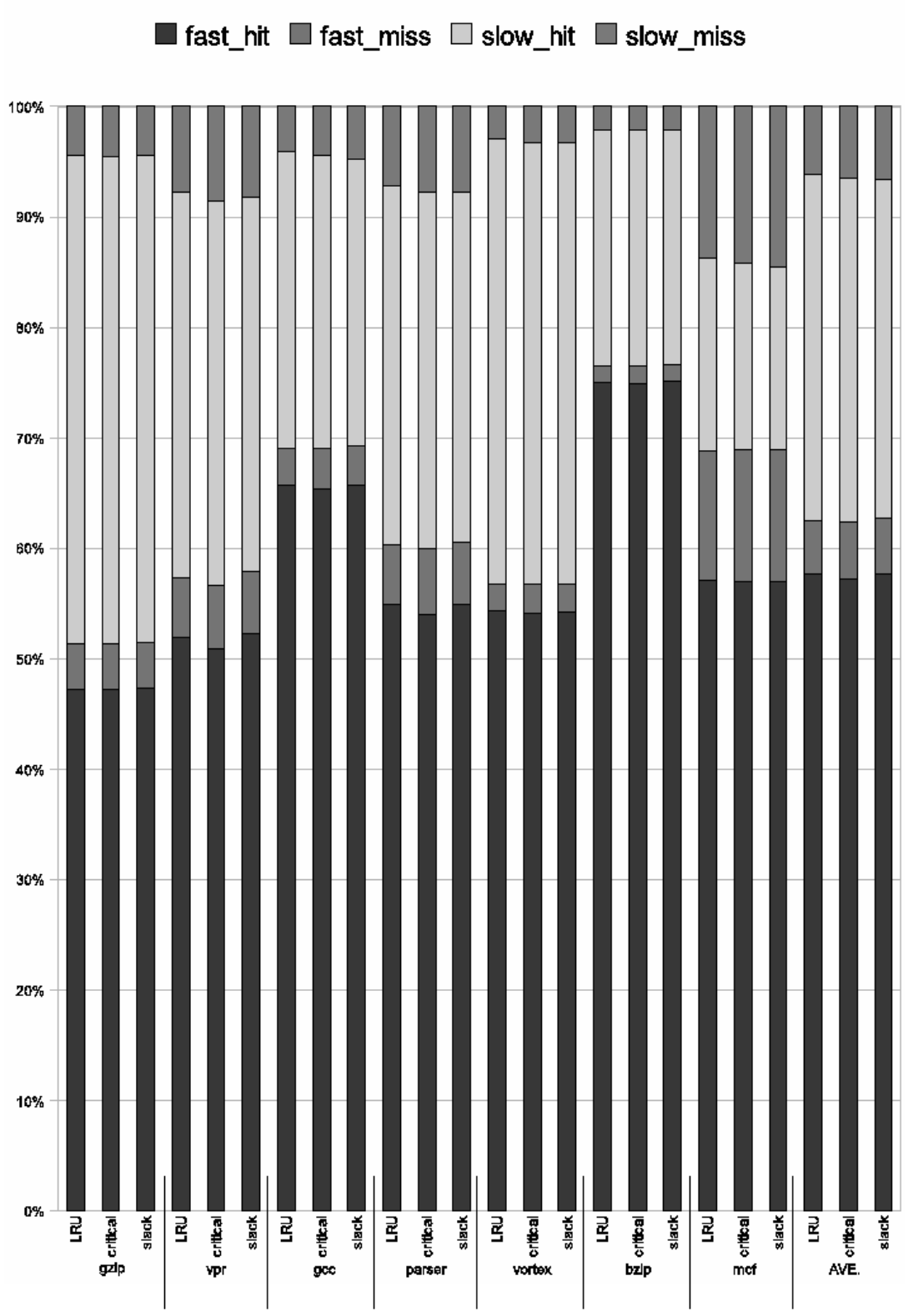

図 29: データの重要度決定方法ごとのキャッシュアクセス割合の内訳

6.3.2 消費エネルギー量の結果

データの重要度決定方法ごとの消費エネルギー量の結果を図 30 に示す．グラフの横軸はベンチマークごとに，データの重要度決定方法を変えた場合の結果を表し，縦軸は消費エネルギーの総量と，内訳を表している．access_fast は高速な領域でのアクセスによって消費したエネルギー量，access_slow は低速な領域でのアクセスによって消費したエネルギー量，swap はラインの移動に消費したエネルギー量を，leak はリーク電流によって消費したエネルギー量をそれぞれ表している．グラフは各ベンチマークごとに LRU を用いた場合で正規化されており，グラフの値が大きいほど消費電力が大きいことを示している．

まず，LRU と slack を比較する．vpr，gcc，parser，vortex，bzip のベンチマークにおいて，slack の方が消費するエネルギーが大きく gzip，mcf においては，slack の方が消費するエネルギーが小さくなっている．この要因として大きいのは，移動による消費エネルギー量の大きさのちがいであると考えられる．データ移動回数はベンチマークによって LRU と slack で大小が違っているがライン移動回数を抑えている方が全体の消費電力を小さくすることができている．

また，vpr，parser においては，slack の方が高速な領域のアクセス回数が大きくなってしまっていることも消費電力が大きくなっている原因と考えられる．これは命令のスラックが 1 以上の命令が低速な領域からアクセスした場合，低速な領域でミスをして高速な領域にアクセスしている割合が大きくなっていることを表す．vpr，parser は，性能評価でも slack の方が悪い結果となっており，スラックに基づくラインの移動が上手く機能しておらず最適なデータ配置となっていないと考えられる．

次に，critical と slack を比較する．gcc，vortex，bzip，vpr，parser において消費エネルギーに大きな差はでていないが，gzip と mcf では，ラインの移動回数が少ない slack の方が消費エネルギーは小さくなっている．

6.3.3 エネルギー遅延二乗積の結果

データの重要度決定方法ごとのエネルギー遅延二乗積 (ED^2P) の結果を図 31 に示す．グラフの横軸はベンチマークごとに，データの重要度決定方法を変えた場合の結果を表し，縦軸はエネルギー遅延二乗積を表している．グラフは各ベンチマークごとに LRU を用いた場合で正規化されており，グラフの値が小さいほど省電効果が高いことを示している．

初めに，LRU と slack を比較する．gzip，mcf においては，slack の方が良い結果になっている．これは，ラインの移動回数が少ないことで slack の方が消費エネルギー

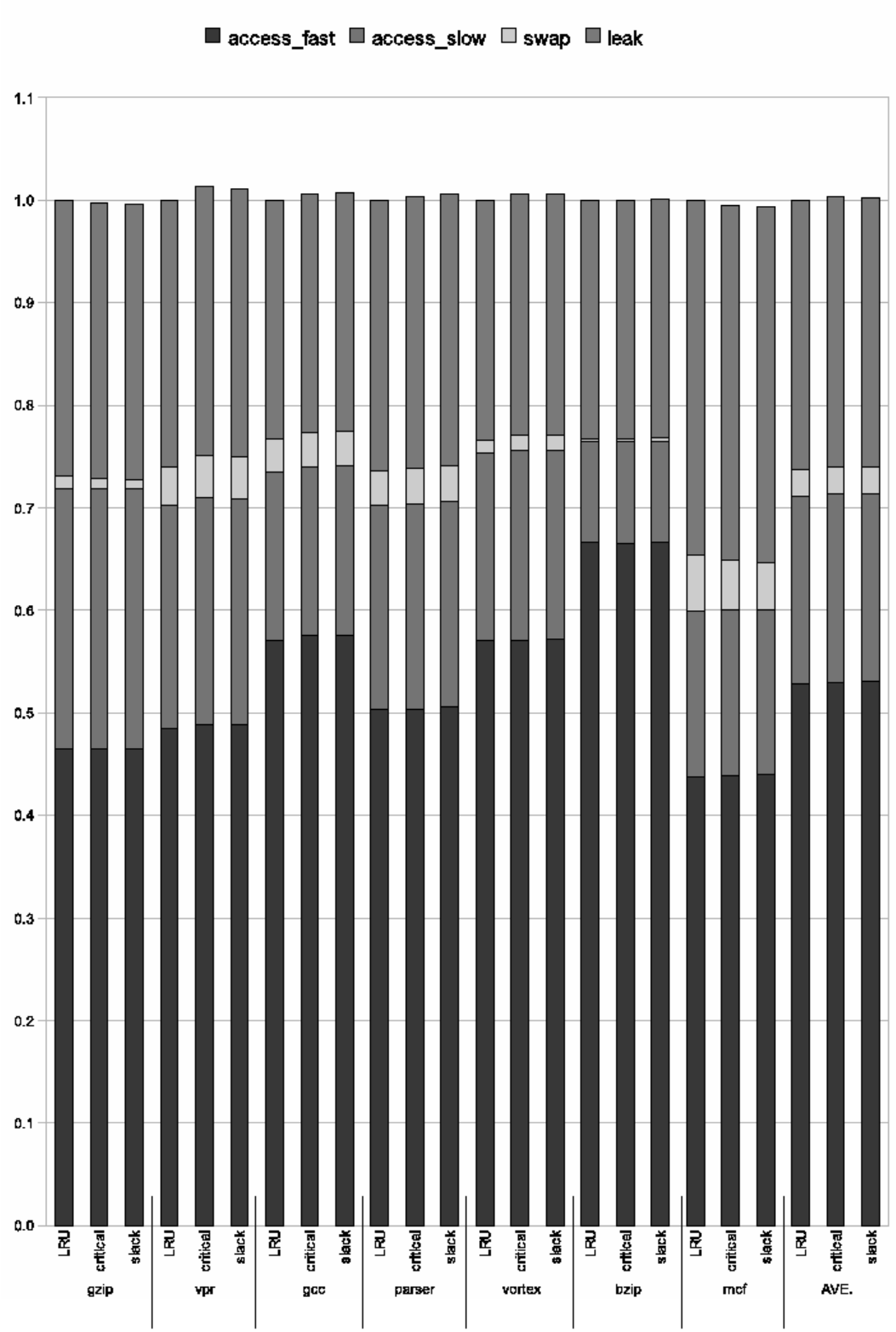

図 30: データの重要度決定方法ごとの消費エネルギー量

が小さくなった効果が表れた結果である．また bzip においては，消費エネルギーは slack の方が大きかったものの，性能も slack が大きかったったため，ED^2P は slack の方が良い結果となった．しかし vpr，gcc，parser，vortex においては，消費エネルギーが大きくなったことが影響して，slack の方が悪い結果となった．平均をみてみると，slack より LRU の方が省電効果が高い結果となった．

次に，critical と slack を比較する．gcc では critical の方が良い結果となったものの，そのほかのベンチマークでは，slack の方が良い結果となった．命令のスラックでデータの重要度を決定する場合，クリティカルパス情報では判別できなかった最も重要度の値の大きいラインを選べることの効果が表れた結果になった．

6.4 アクセス領域選択における選択基準値を変化させた場合の評価

最後に，命令のスラックの値に基づくアクセス領域選択における選択基準値を変化させた場合の評価を行った．命令のスラックの値が基準値以下なら高速な領域からアクセスを行い，命令のスラックの値が基準値より大きければ低速な領域からアクセスを行う．ここでは，データの重要度の決定は，スラックを用いて行うこととした．

6.4.1 プロセッサの処理性能の結果

領域選択基準値を変化させた場合の処理性能の結果を．図 32 に示す．グラフの横軸はベンチマークごとに，キャッシュメモリへのアクセス基準変えた場合の結果を表している．0，5，10，30，の数値はアクセスする領域を決定するための基準値を表しており，”無し”は基準値を設けないで，必ず高速な領域からアクセスを行った場合を表している．また，縦軸は処理性能の大きさを表している．グラフはベンチマークごとに基準値を 0 とした場合で正規化されており，グラフの値が大きいほど処理性能が高いことを示している．

図 33 に重要度の決定方法ごとにキャッシュアクセス命令が L1 キャッシュメモリのどの領域をアクセスしたかの内訳も示す．グラフの横軸は処理性能の結果と同様である．縦軸は，キャッシュへアクセスした割合を示している．fast_hit は高速な領域でキャッシュヒットする割合，fast_miss は高速な領域でキャッシュミスをする割合，slow_hit は低速な領域でキャッシュヒットする場合，slow_miss は低速な領域でキャッシュミスをする割合をそれぞれ表している．

どのベンチマークでもアクセス領域選択の基準値大きくするにつれて，高速な領

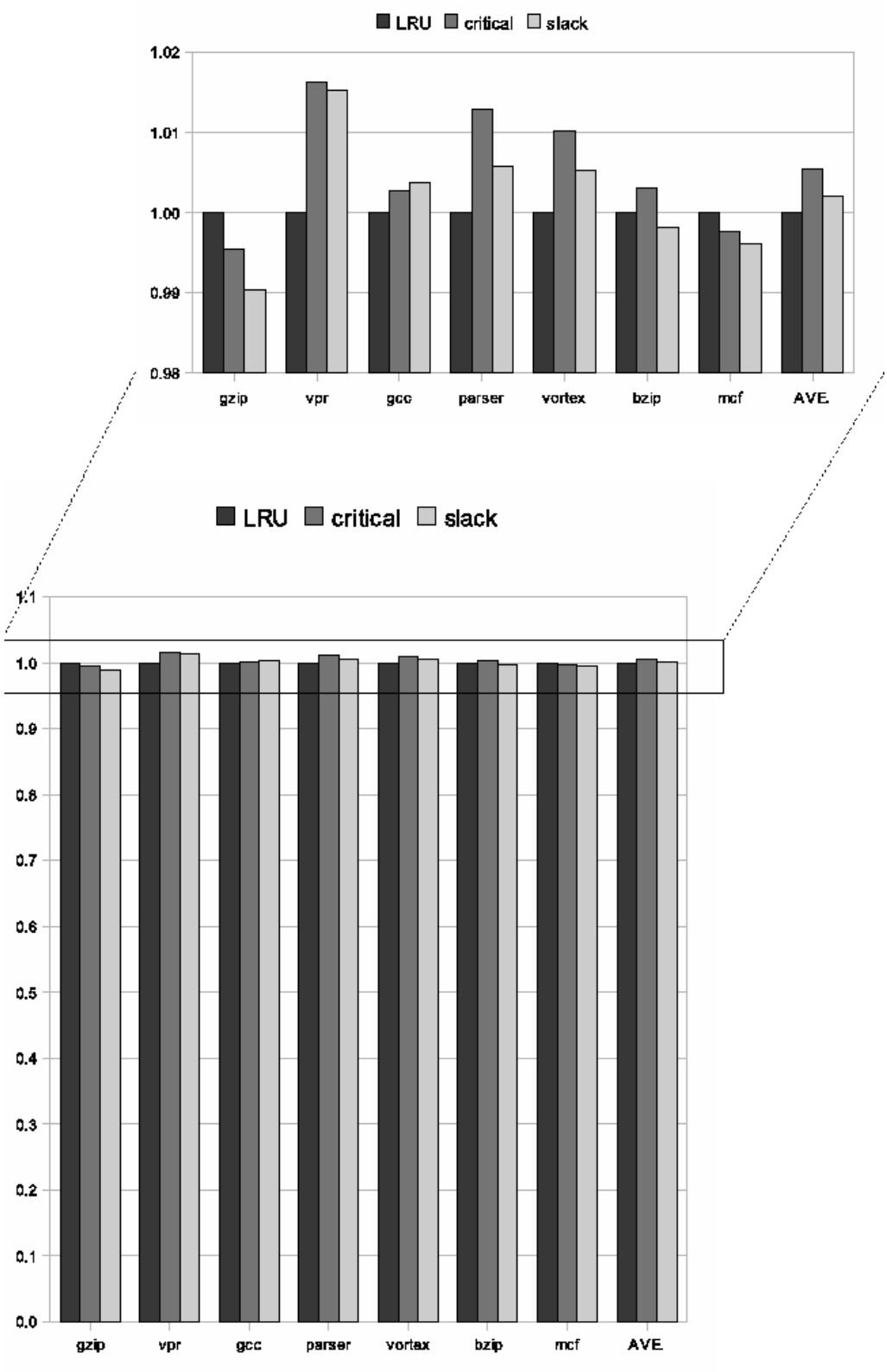

図 31: データの重要度決定方法ごとのエネルギー遅延二乗積

域からアクセスして，高速な領域でヒットする割合が大きくなるため，処理性能は大きくなっている様子が分かる．ただし高速な領域へのアクセスが多くなるとそれだけ消費エネルギーは大きくなっていく．

6.4.2 消費エネルギー量の結果

領域選択基準値を変化させた場合の消費エネルギー量の結果を図 34 に示す．グラフの横軸はベンチマークごとに，キャッシュメモリへのアクセス基準を変化させた場合の結果を表している．また，縦軸は消費エネルギーの総量と，内訳を表している．グラフは各ベンチマークごとに基準値を 0 とした場合で正規化されており，グラフの値が大きいほど消費エネルギーが大きいことを示している． まず，gzip，vortex，bzip に着目すると，アクセス領域選択の基準値大きくするにつれて高速な領域へのアクセスが増加することでアクセスによるエネルギーは大きくなっている．しかし処理性能向上によるプログラム実行時間の減少で，リーク電流による消費エネルギー総量は減少している．そのため消費エネルギー合計ではあまり差が出ていない．

次に vpr，gcc，parser，mcf に着目する，これらのベンチマークでは，アクセス領域選択の基準値大きくするにつれて高速な領域へのアクセスが増加することでのアクセスによるエネルギーは大きくなるものの，ラインの移動回数の減少で移動によるエネルギーの減少，処理性能向上によるプログラム実行時間の減少に伴うリーク電流による消費エネルギーの減少が大きく，消費エネルギー合計もアクセス領域選択の基準値を設けないで高速からアクセスする方が小さいという結果になった．

6.4.3 エネルギー遅延二乗積の結果

領域選択基準値を変化させた場合のエネルギー遅延二乗積 (ED^2P) の結果を図 35 に示す．グラフの横軸はベンチマークごとに，キャッシュメモリへのアクセス基準を変化させた場合の結果を表している．また，縦軸はエネルギー遅延二乗積を表している．グラフは各ベンチマークごとに基準値を 0 とした場合で正規化されており，グラフの値が小さいほど省電効果が高いことを示している．

どのベンチマークでも，基準値を大きくして高速からアクセスする割合を増やした方がよい結果になっている．高速な領域でキャッシュヒットすることによる処理性能向上の効果が大きく出ており，消費エネルギーが大きくなっても基準値を大きくして高速からアクセスする割合を増やした方がよいということになる．

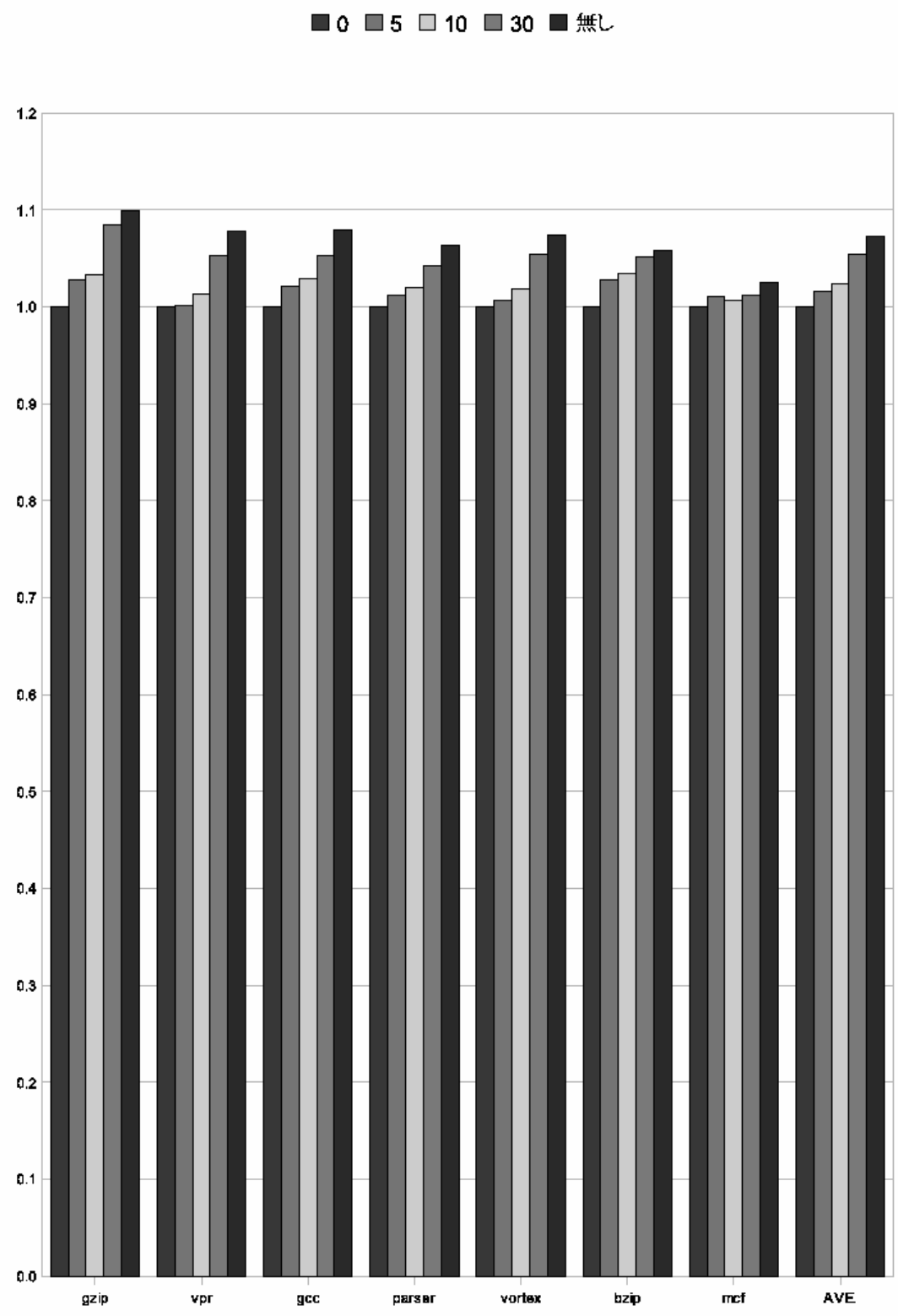

図 32: アクセス基準を変化させた場合の IPC

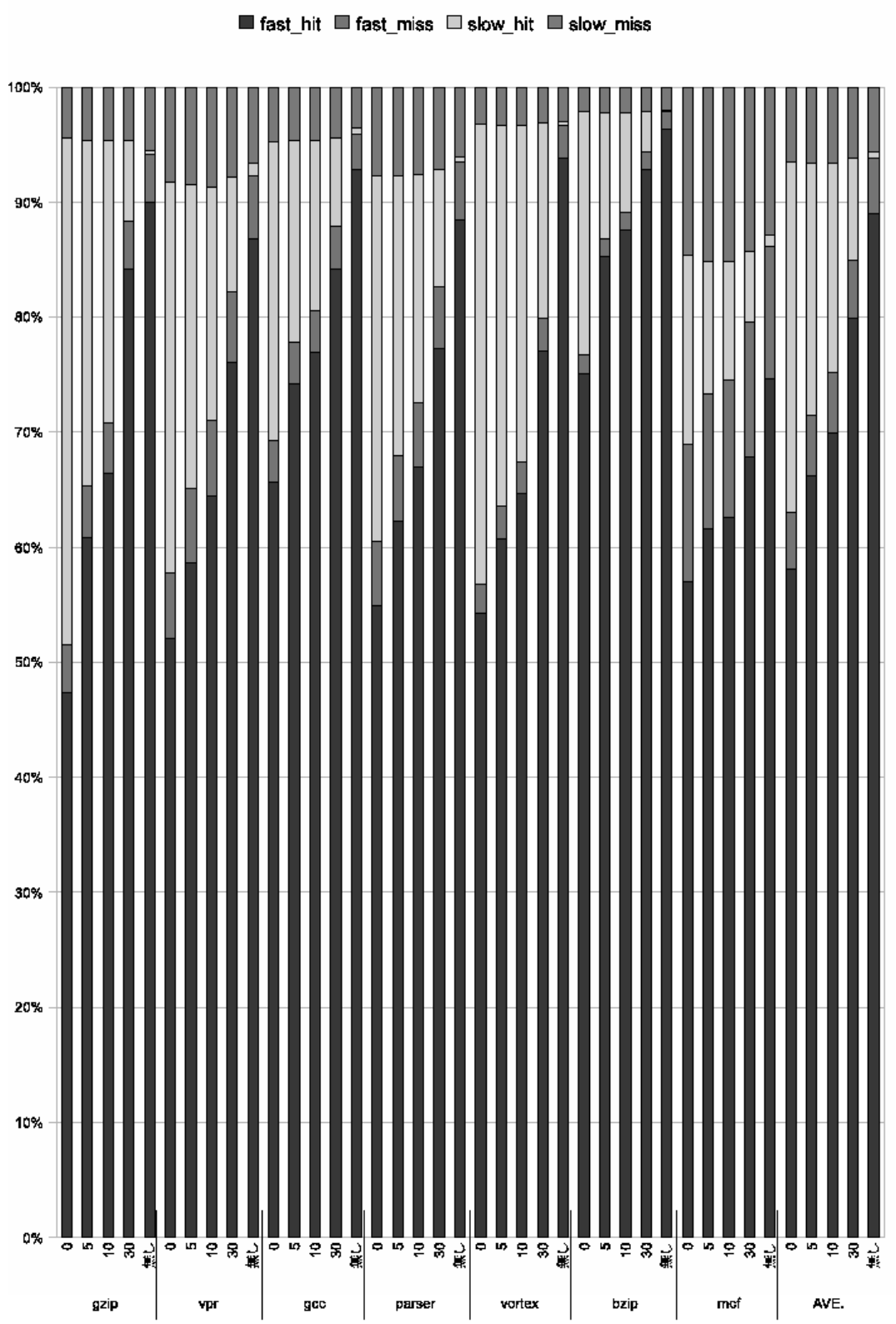

図 33: アクセス基準を変化させた場合のアクセス割合

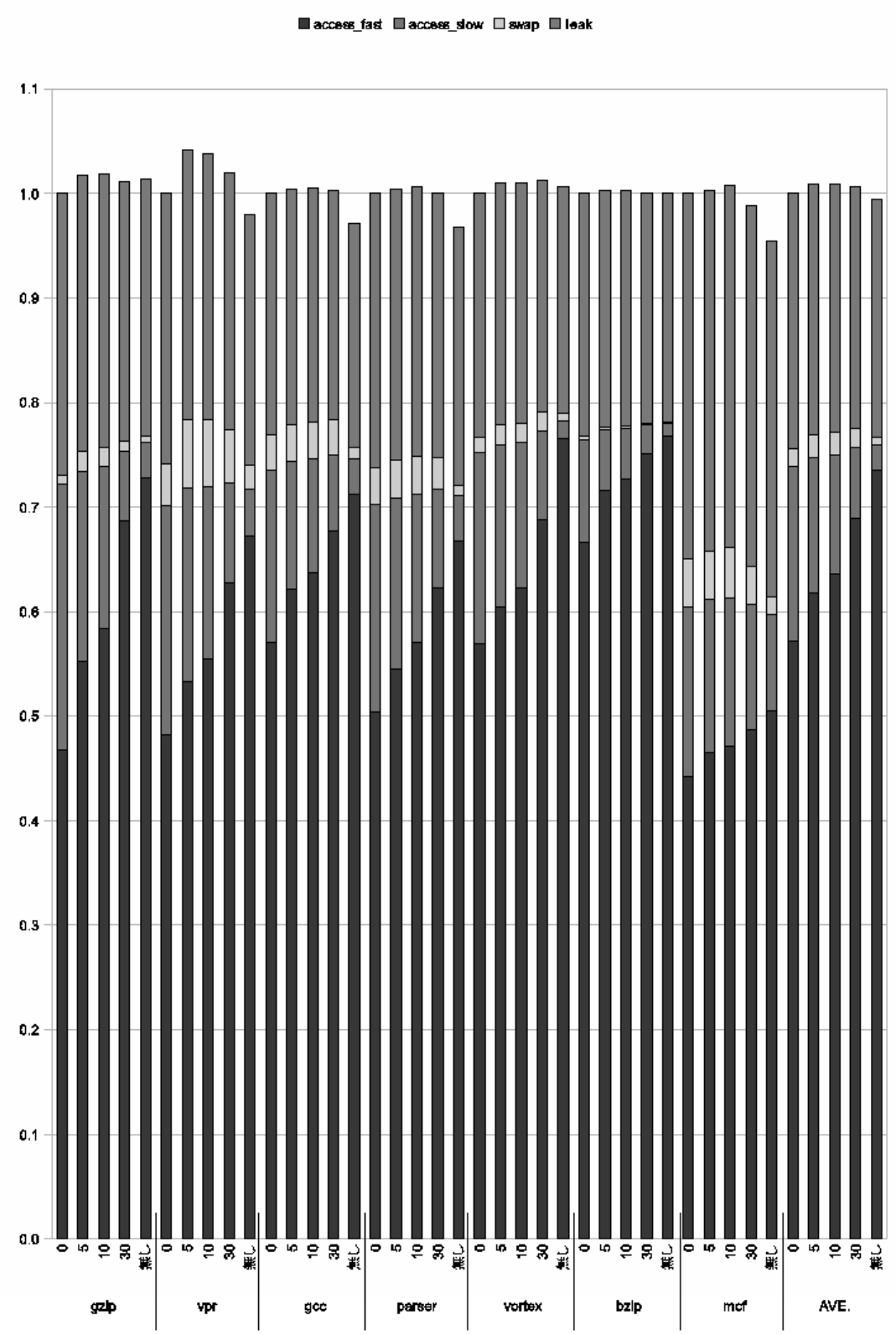

図 34: アクセス基準を変化させた場合の消費エネルギー量

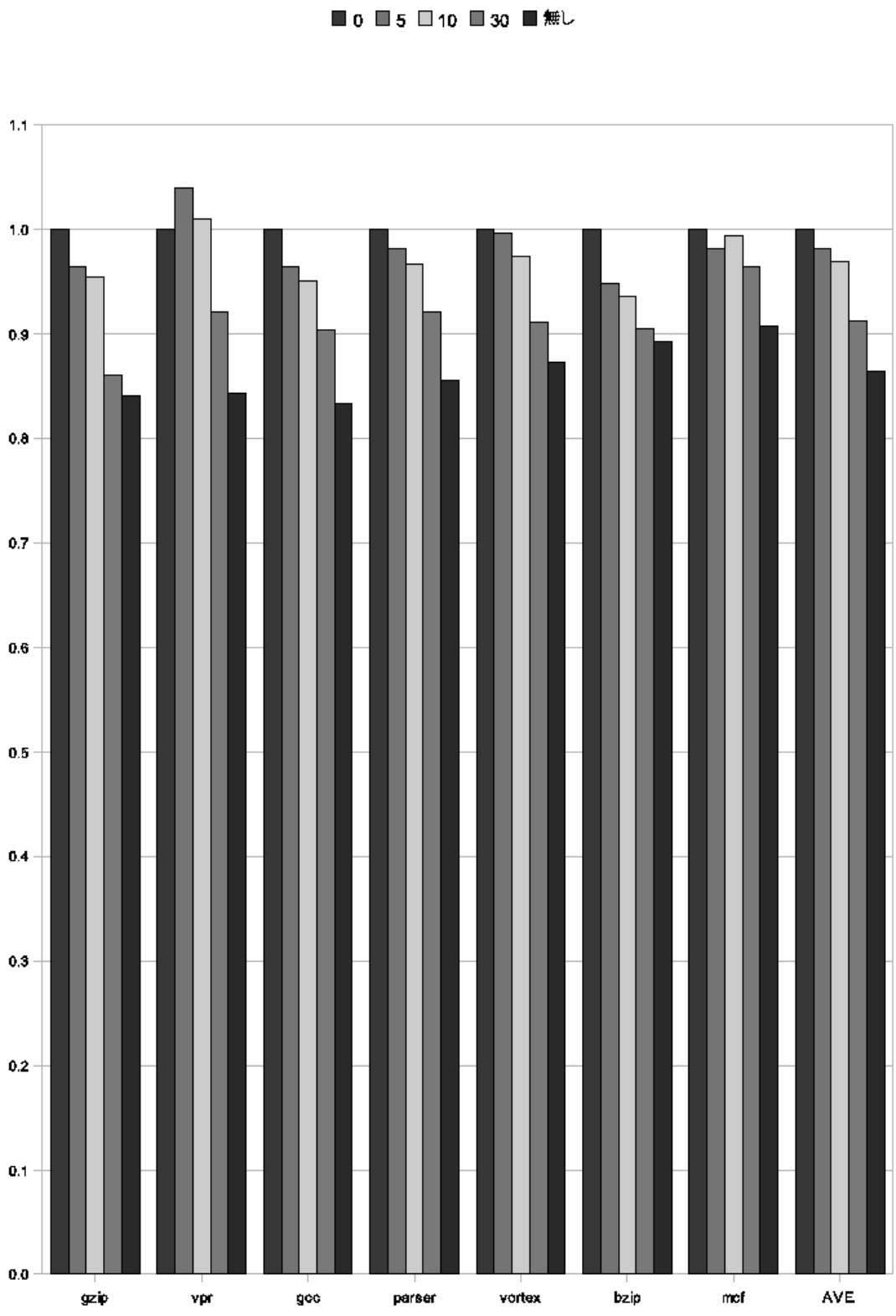

図 35: アクセス基準を変化させた場合のエネルギー遅延二乗積

6.5 考察

今回行ったキャッシュへのアクセス基準を変化させた場合の評価では，データ配置領域の選択のためのデータの重要度を命令スラックに基づいて行う方式を採用していた．そのため，アクセス基準を適切に設定することで，キャッシュへのアクセス領域の選択がキャッシュのデータの重要度とうまくかみ合う場合に処理性能の向上や消費電力の削減が見込めることを期待していた．しかし結果は，単純に高速からアクセスした方が良いという結果であった．以下に提案手法上手くいかなかった原因を考察する．

まず考えられるのが，本研究が利用している命令スラックがグローバルスラックではなくローカルスラックであることがあげられる．グローバルスラックを 1 以上持っている命令でも，ローカルスラックは 0 である場合は多い．このような命令は，低速な領域からデータが供給されても性能へ悪影響を与えにくいと考えられるが，ローカルスラックが 0 であるために高速な領域からデータが供給されており，本来消費電力を削減する事が可能な場合でも，消費電力を削減することができていなかった場合がある．予測で得られている命令スラックの値がグローバルスラックとは異なる値になっていたことが問題である．

次に考えられることが，データの重要度の決定方法に問題があることである．図 32 の結果では，命令スラックの値が 0 である命令のみ高速な領域からアクセスを行った場合と全ての命令で高速な領域からアクセスを行った場合との処理性能の差が，非常に大きくなってしまっている．低速な領域からアクセスする場合で低速な領域でキャッシュミスをしてしまうと高速な領域へもアクセスしなくてはならなくなる．この低速な領域でキャッシュミスすることのペナルティが大きく処理性能に大きな差が出てしまった．

今回の提案ではデータの重要度はラインごとに管理することとしているが，命令がアクセスするのはライン内のデータである．本研究では，最後にラインへアクセスした命令によって，その命令スラックの値を基にラインの重要度指標の更新が行われている．そのため，ラインに設定されている重要度指標が，ライン内のデータ全てに一致しているわけではなく，最後にラインへアクセスした命令のスラックの値が小さいとアクセスのあったラインの重要度の値は小さくなる．ところが，そのラインのデータの中には，データの供給が遅れてもでプログラムの実行時間に悪影響を与えないスラックの大きい命令によってアクセスされたデータが多く含まれているかもしれない．このような場合，そのラインは低速な領域に配置されるべきである．しかしラインに設定されているスラックの値が小さいため，移動の対象とな

らずそのラインは高速な領域にとどまったままになる．結果として，スラックの大きい命令が低速な領域でキャッシュミスをして，高速な領域へもアクセスをしなくてはならなくなってしまう．

このラインの重要度の決定方法の問題は，データの重要度の決定方法の違いによる評価において，データの重要度をスラックで決定した場合よりもLRUで決定した方が良い結果になったことにも関係する．データの重要度をスラックで決定してもアクセスする命令とラインの重要度が対応していない場合が多く，データアクセスの時間的局所性に基づいてLRUで重要度を決定した方が最適な配置になってしまったと考えられる．今後アクセスする命令に対応したラインの重要度の決定方法を考える必要がある．

7　終わりに

7.1　まとめ

本研究では，深刻な問題となっているプロセッサの動的・静的消費電力を削減するための省電力キャッシュアーキテクチャを検討した．本研究で検討したキャッシュアーキテクチャは，L1 キャッシュメモリを高速だが高消費電力な領域と低速だが省電力な領域に分割して，データの重要度に基づいてデータの配置を行った．データの重要度として命令のスラックを利用し，高速な領域には重要度の高いデータを配置し，低速な領域には重要度の低いデータを配置した．

また，アクセスする領域を選択するための基準値を設けて，命令のスラックの値が基準値以下の命令は高速な領域から，命令のスラックの値が基準値より大きい命令は低速な領域からアクセスする方法でアクセスを行うことで，処理性能を低下させることなく消費電力を削減することを目指した．

データの重要度の違いによる評価の結果，従来の手法であるクリティカルパス情報を用いた場合と比較して命令のスラックを用いた場合は処理性能の低下を小さくすることができ，それに伴って ED^2P も小さくすることができる評価を示し，データの重要度として命令のスラックを用いることの効果を示した．

アクセスする領域を選択するための基準値を変化させた場合の評価では，最適な基準値を設けた場合に最も ED^2P が小さくなることを期待したが，結果は命令のスラックによらず必ず高速からアクセスする場合，最も ED^2P が小さくなった．期待通りの評価が得られなかった原因としては，命令のスラックを考慮する際にグローバルスラックではなくローカルスラックを利用しているため，消費電力を削減する効果が十分に発揮されなかった事，データの重要度の決定をライン単位で行っているためアクセスする命令のスラックの値と，ラインに設定される重要度の値が食い違い，最適なデータ配置が行われていなかった事が考えられる．

7.2　今後の課題

データの重要度の違いによる評価で，命令のスラックを用いた場合の評価は，クリティカルパス情報を用いた場合の評価よりも良い結果を残した．しかし，今回行ったクリティカルパス情報の取得方法では，スラックが 0 の命令をクリティカルパス上の命令とし，スラックが 0 より大きい命令はスラックの値が決定できずクリティカルパス上にない命令とすることでクリティカルパス情報としていた．他のより正確なクリティカルパス情報を取得する方法と提案手法を比較して評価を行うことが

必要である．

また，今回の実装において，高速な領域と低速な領域との間でデータの移動を行う場合，高速な領域に空きラインがある場合は，低速な領域から高速な領域へのみラインの移動を行えば良いところを，移動の必要がない高速な領域の空きラインをわざわざ低速な領域へ移動させていた．必要のないデータ移動は行わないようにする必要がある．

また，アクセスする領域を選択するための基準値を変化させた場合の評価において，命令のスラックとしてローカルスラックを用いることでは，消費電力を削減する効果が十分に発揮されない場合が考えられた．グローバルスラックを用いての評価は困難であるが，今回用いたスラック予測手法よりもよりグローバルスラックに近い値を予測できる予測手法を用いることで良い評価を得られる可能性がある．

また，データの重要度の決定方法の問題を解決する必要がある．キャッシュメモリはデータの移動をライン単位行うため各ラインごとに重要度を決定する必要があるのだが，今回評価を行った命令のスラックを用いたデータの重要度の決定方法では，最後にラインへアクセスを行った命令のスラックがそのラインの重要度として決定されていた．この方法では，必ずしもライン内の全てのデータの重要度が反映されておらず，ラインの重要度はアクセスする命令のスラックに対応した値にならない場合がある．よって，ライン内の全てのデータを反映した重要度決定方法を導入することが課題である．

謝辞

本研究を進めるにあたり，数多くのご指導ならびにご助言をいただきました高性能コンピューティング学講座本多弘樹教授，近藤正章准教授，平澤将一助教に深く感謝いたします．また，研究を進める上でご支援やご助言をいただきました高性能コンピューティング学講座の皆様に深く感謝いたします．

参考文献

[1] 千代延昭宏，藤井誠一，佐藤寿倫:データの重要度を利用したキャッシュメモリの省電力化，情報処理学会論文誌，Vol．48，No．SIG8(ACS18)，pp．114-126 2007.

[2] 劉小路，小西将人，五島正裕，中島康彦，森眞一郎，富田眞治: クリティカリティ予測のためのスラック予測，先進的計算基盤システムシンポジウム SACSIS 2004，pp．187-196 2004.

[3] 福田匡則，小西将人，五島正裕，中島康彦，森眞一郎，富田眞治: グローバル分岐履歴を用いたスラック予測器，情報処理学会研究報告 2004-ARC-159 (SWoPP 2004)，pp．25-30 2004.

[4] 小林良太郎，林久紘，島田俊夫: 発見的手法に基づくローカル・スラック予測機構 先進的計算機基盤システムシンポジウム SACSIS 2006，pp．385-394 2006.

[5] Doug Burger and ToddM．Austin．The Simplescalar Tool Set Version 2．0．Technical Report CS-TR-97-1342，University of Wisconsin，June 1997.

[6] Jaume Abella，and Antonio Gonzalez:Power Efficient Data Cache Designs，In *Proceedings of the International Conference on Computer Design*，pp．8-13 October 2003.

[7] Rajeev Balasubramonian，Viji Srinivasan，Sandhya Dwarkadas，and Alper Buyuktosunoglu．Hot-and-cold: Using criticality in the design of energy-efficient caches．In *Proceedings of the Workshop on Power Aware Computer Systems*，pp．180-195，December 2003.

[8] Fields，B．and Blodik，S．R．R．: Focusing Processor Policies via Critical-Path Prediction，*28th Annual International Symposium on Computer Architecture (ISCA-28)*，pp． - 2001.

[9] Fields，B．，Bodik，R．，and Hill，M．D．:Slack Maximizing performance under technological constraints．In *29th Annual International Symposium on Computer Architecture (ISCA-29)*，pp． 47-58，May 25-29 2002.

[10] 小林良太郎，安藤秀樹，島田俊夫: データフロー・グラフの最長パスに着目したクラスタ化スーパースカラ・プロセッサにおける命令発行機構，並列処理シンポジウム JSPP 2001，pp．31-28 2001.

[11] 千代延昭宏，佐藤寿倫，有田五次郎:低消費電力プロセッサアーキテクチャ向けクリティカルパス予測器の提案，情報処理学会研究報告 2002-ARC-149 (SWoPP 2002)，pp. 1-6 2002.

[12] Casmira, J. and Grunwald, D. : Dynamic Instruction Scheduling Slack, *Kool Chips Workshop (in conjunction with MICRO-33)* 2000.

[13] Tune, E., Liang, D., Tullsen, D. M. and Calder, B. : Dynamic Prediction of Critical Path Instructions, *7th Int'l Symp. on High Performance Computer Architecture (HPCA7)* 2001.

[14] Afzal Malik, Bill Moyer, Dan Cermak: A Low Power Unified Cache Architecture Providing Power and Performance Flexibility, In *Internanional Symposium on Low Power Electronics and Design*, 2000.

[15] Krisztian Flautner, Num Sung Kim, Steve Martin, David Blaauw, Trevor Mudge, Drowsy Cache: Simple Techniques for Reducing Leakage Power, In *29th Annual International Symposium on Computer Architecture (ISCA-29)*, 2002.

[16] 千代延昭宏，佐藤寿倫:プログラム実行時における命令の重要度決定に関する検討，情報処理学会研究報告 2003-ARC-154 (SWoPP 2003)，pp. 1-6 2003.

[17] Internanional Technology Roadmap for Semiconductors. Semiconductor Industry Association, 2002.

[18] 千代延昭宏，佐藤寿倫:メモリアクセス命令の重要度を利用したキャッシュメモリの省電力化，情報処理学会九州支部 火の国情報シンポジウム，March 2004.

[19] Margaret Martonosi, David Brooks, and Pradip Bose: Modeling and analyzing cpu power and performance: Metrics, methods, and abstractions. Tutorials, SIGMETRICS 2001 / Performance 2001, June 2001.