社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－ARC－164 (2)
2005／8／3

スーパスカラ・プロセッサのための物理レジスタ2段階解放

山本哲弘† 安藤秀樹† 島田俊夫†

近年のプロセッサではプロセッサのクロック速度とメモリの速度の差は大きくなり，大きな性能低下を生んでいる．この問題を解決する手法として，データのプリフェッチが挙げられる．本論文では，物理レジスタの2段階解放と呼ぶ手法を導入し，プログラムに潜在するILPを利用したデータのプリフェッチを実現する．本手法は，第1段階として，リネームステージでレジスタを解放することにより，レジスタ不足によるストールをなくす．命令の実行終了には最終的な2段階目の解放を待つ必要があるが，実行待ちの命令に先行実行を行わせることにより，ロードデータのキャッシュへのプリフェッチを実現した．SPECfp2000ベンチマークを用いて測定したところ，物理レジスタ数が64個の場合，平均で32%もの大きな性能向上を達成できることを確認した．

Two-Step Physical Register Deallocation for Superscalar Processors

Akihiro Yamamoto,† Hideki Ando† and Toshio Shimada†

Recently the difference between the clock speed of a processor and the speed of a memory access grows and it results in large performance degradation. One of its solutions is data prefetching techniques. This paper proposes a two-step physical register deallocation scheme that allows data prefetching by exploiting potential ILP in programs. Our scheme deallocats physical registers in the renaming stage as a first step and removes stalls caused by physical register shortage. Instructions wait the final deallocation as a second step in the instruction window. While waiting, the scheme enables prefetching of load data into the cache by allowing pre-execution of instructions. Our evaluation results using SPECfp2000 benchmarks show that our scheme achieves significant speedups of 32% on average in the typical case of 64 physical registers.

1. はじめに

近年のプロセッサはますますクロック周波数が高くなり，メモリアクセス速度との解離が非常に大きくなっている．このため，長いロードレイテンシが引き起こす性能低下は非常に深刻なものとなっている．現在の主なプロセッサではロードレイテンシを削減し，クリティカルパスを短縮するため，キャッシュの階層化が行われている．しかし，キャッシュの大きさはクロック周波数などの点から制限され，依然としてキャッシュミスによる性能低下は大きい．

この問題を解決する方法として，データのプリフェッチが挙げられる．実際にメモリアクセスを行う前にデータをキャッシュやバッファに取り込むことによりキャッシュミスを回避する．従来の手法は大きく分けてハードウェアによる方法とソフトウェアによる方法に分類できる．

ハードウェアによる方法では実行時にロード命令のアクセスパターンから法則を検出し，次にアクセスされるデータのアドレスを予測することによりプリフェッチを行う．この方法ではいつ，どのデータをプリフェッチするかが重要となる．早すぎるプリフェッチは他の有用なデータの上書きを招き，遅すぎてはレイテンシを削減できない．無駄なデータのプリフェッチは逆にキャッシュミスを増加させる可能性がある．また，単純なアクセスパターンを検出するのは比較的簡単であるが，複雑なアクセスパターンを検出するには大きなハードウェアを要する．

ソフトウェアによる方法では，コンパイル時にコードを解析することによって，複雑なアクセスパターンを検出することができる．しかし，キャッシュミスという動的なイベントを予測することは難しく，適切にプリフェッチを行うのは困難である．また，プリフェッチ命令によりコード量が増加する問題や，バイナリ互換の問題がある．

これに対して本論文では，正確なタイミングで予測に基づかず実際に参照されるデータをプリフェッチす

† 名古屋大学大学院 工学研究科
Graduate School of Engineering, Nagoya University

る手法を提案する．本手法は，プロセッサが利用している命令レベル並列性 (ILP:Instruction-Level Prallelism) より，プログラムに潜在する ILP が高いことを利用するものである．プロセッサが利用している ILP は，プロセッサ内の様々な資源によって制限されるが，本手法はレジスタファイルの大きさによって制限されることに着眼している．一般に，プログラムに潜在する ILP を十分に引き出すだけの大きなレジスタファイルを実装することは必ずしも得策ではない．その理由は，大きなレジスタファイルのアクセス時間は長く，クロック速度に悪影響を与えるからである．また，消費電力・面積も大きくなるという問題がある[2]．このような理由から実際のプロセッサでは，限られた大きさのレジスタファイルしか実装されない．

本手法は，物理レジスタの 2 段階解放と呼ぶ手法を導入し，データのプリフェッチを実現する．本方式では，従来コミット時に行われていたレジスタ解放を，パイプラインの異なるステージにおいて 2 段階で行う．第 1 段階として，リネームステージで行う．これにより，レジスタ不足により命令の処理がリネームステージでストールすることはなくなる．命令はパイプライン中を前進し，命令ウィンドウで 2 段階目のレジスタ解放を待ち合わせる．2 段階目のレジスタ解放は，通常のレジスタ解放と同じくコミットステージで行われる．命令は，ソースオペランドが利用可能になり，自身のデスティネーションレジスタの 2 段階目の解放が行われた時，発行可能となる．

データのプリフェッチを行うため，命令ウィンドウで待機している間，ソースオペランドが利用可能となったとき，デスティネーションレジスタの 2 段階目の解放が行われる前に，1 度だけその命令を実行するよう制御する．レジスタへの書き込みは行えないので，実行を終了することはできないが，先行実行が実現される．結果をバイパス経由で依存命令に渡すことにより，先行実行は連鎖的に行われる．先行実行は物理レジスタが十分にあるときの ILP を利用して進むため，本実行よりも速く進む．この結果，ロード命令の先行実行が行われると，データキャッシュへデータがプリフェッチされる．これにより，本実行時のデータキャッシュミスを回避することができる．

本手法ではハードウェアによるプリフェッチ手法の問題点である，いつ，どのデータをプリフェッチするかという問題を解決している．まず，先行実行されるロード命令は，後に本実行される命令そのものである．したがって，プリフェッチされるデータは必ず使用するものである．また，プログラムに潜在する ILP と実際の ILP の違いを利用してプリフェッチを起動するため，適切なタイミングでデータを得ることができる．

本論文では，第 2 章で提案手法について説明し，第 3 章で評価を行う．最後に第 4 章で本論文をまとめる．

2. 2 段階での物理レジスタ解放

この章では，物理レジスタを 2 段階で解放する手法を提案する．最初に基本的な方式を説明し，次に先行実行を実現するよう方式を拡張する．

2.1 基本方式

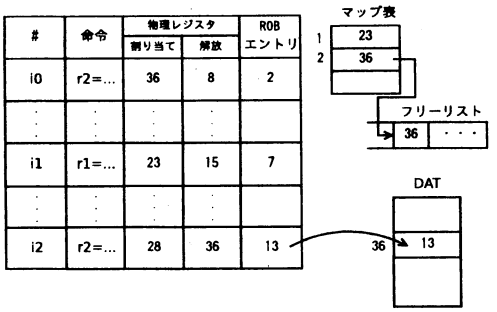
2.1.1 1 段階目の解放

1 段階目の物理レジスタの解放はリネームステージで行われる．リネームステージには，従来のマップ表とフリーリストの他，**解放表** (DAT: Deallocation Table) と呼ぶ表を用意する．この表の各エントリは物理レジスタに対応し，当該物理レジスタを解放する命令が登録されるリオーダバッファ(ROB: Reorder Buffer) のエントリ番号を保持する．

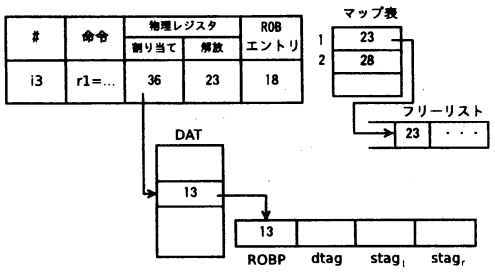
動作は次の通りである．まず，命令の論理デスティネーションレジスタに現在割り当てられている物理レジスタを解放し，フリーリストに追加する．この時，解放する物理レジスタに対応する DAT のエントリに，自身が割り当てられた ROB のエントリ番号を書き込む．また，従来と同じく，フリーリストより空き物理レジスタを得，論理デスティネーションレジスタに割り当てる．この時，DAT を参照し，当該物理レジスタを解放する ROB エントリ番号 ROBP を得る．ROBP は，2 段階目のレジスタの解放タイミングを知らせるための命令のタグとなる．

図 1 にこの動作の例を示す．表は命令に割り当てられた物理レジスタ，解放される物理レジスタ，および，割り当てられた ROB のエントリ番号を示す．同図 (a) は，命令 `i0,i1` がすでにリネームされており，今，`i2` がリネームされる時の動作を示している．まず，論理デスティネーションレジスタ `r2` に現在割り当てられている物理レジスタ 36(命令 `i0` の「割り当て」の欄参照) を解放し，フリーリストに追加する．同時に，DAT の第 36 エントリに割り当てられた ROB エントリ番号 13 を書き込む．

次に，命令 `i2` が解放した物理レジスタ 36 が命令 `i3` に割り当てられる時の動作を同図 (b) に示す．前述のように，物理レジスタの解放と割り当てを行った後，DAT を参照し，割り当てられた物理レジスタ 36 を最終的に解放する (2 段階目の解放) 命令が格納されている ROB のエントリ番号 13 を得る．この番号は，命令に付加されるタグ ROBP となり，デスティネー



(a) 命令 i2 のリネーム



(b) 命令 i3 のリネーム

図 1 リネーム時の動作

ションタグ (dtag)、2 つのソースレジスタタグ (stag$_l$ および stag$_r$) とともに，命令ウィンドウに書き込まれる．なお，従来は，オペランドタグとして物理レジスタ番号が用いられたが，本方式では，任意のユニークな番号をタグとすることとする．本方式では，リネームステージで物理レジスタを解放するので，物理レジスタ番号では異なる依存関係をもはや識別できないからである．

2.1.2 2 段階目の解放

命令は，命令ウィンドウにおいて，自身のデスティネーション物理レジスタの 2 段階目の解放を待ち合わせる．2 段階目の解放は，コミット時に行われる．この時解放される物理レジスタは，従来と同じく，コミットされる命令の論理レジスタに以前に割り当てられていた物理レジスタである．従来と異なるところは，解放の際に，ROB のエントリ番号 ROBP を命令ウィンドウに放送する点である．放送された ROBP が，命令ウィンドウで待ち合わせている命令の ROBP タグと一致したなら，その命令は実行結果の書き込みを許可される．

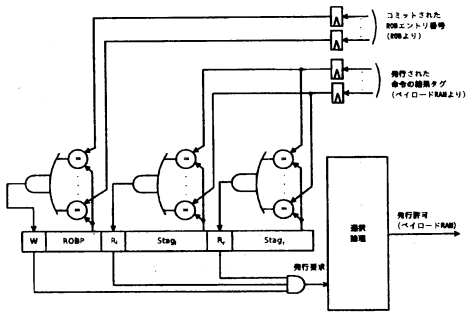


図 2 2 段階レジスタ解放の基本方式におけるウェイクアップ論理

図 1 の例では，ROB の第 13 エントリに格納されている命令 i2 がコミットされたなら，物理レジスタ 36 の 2 段階目の解放が行われる．そして，ROB のエントリ番号 13 を命令ウィンドウに放送する．その結果，命令 i3 の ROBP タグと一致し，書き込み許可を得る．

図 2 に命令ウィンドウのウェイクアップ論理を示す．命令に対応して，ROBP，stag$_l$，および，stag$_r$ の 3 つのタグを持つ．各タグに対応して，3 つのフラグ W，R$_l$，R$_r$ を持つ．R$_l$ と R$_r$ は，ソースオペランドが用意されたことを示す従来のレディフラグである．W は，当該命令が書き込み許可を得たことを示すフラグである．命令が発行されるとデスティネーションタグ (あるいは結果タグとも呼ばれる) が放送され，stag$_l$，stag$_r$ に付加された比較器で一致がチェックされ，一致したならばレディフラグがセットされる．これは従来通りである．一方，ROB より送られてきた ROB エントリ番号は，ROBP タグと比較され，一致した場合，W フラグがセットされる．R$_l$，R$_r$，W が全てセットされたとき，選択論理に発行要求が出される．

2.2 命令の先行実行

前節では，命令ウィンドウで待機している命令は，書き込み許可が得られるまで発行しないとした．しかし，書き込み許可を得る前に，ソースオペランドが利用可能となったとき一度だけ先行実行することで，データのキャッシュへのプリフェッチを行うことができる．これにより本実行時のデータキャッシュミスペナルティを低減できる．

2.2.1 ウェイクアップ論理

図 3 に，先行実行を実現するウェイクアップ論理を示す．図 2 に示した基本方式のウェイクアップ論理との違いは，一度の実行を制御するために T-FF を設けること，および発行された命令の結果タグにその命令が先行実行であることを示す **pexec フラグ**を付加す

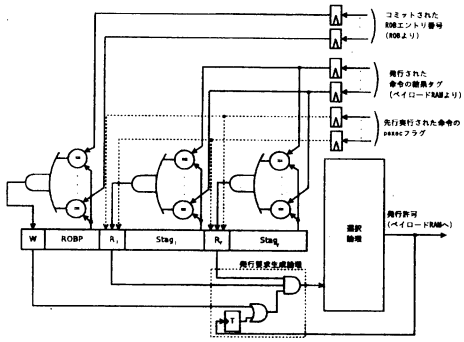


図 3　先行実行を実現するウェイクアップ論理

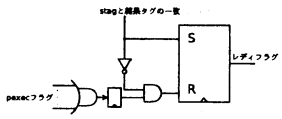


図 4　レディフラグをクリアする回路

ることである．

命令はウェイクアップ論理のエントリに書き込まれる際に，タグを書き込むとともに，T-FF をセットする．命令の実行終了時には，結果タグが命令ウィンドウに放送され，タグが一致するエントリは実行結果を取り込み，レディフラグをセットする．両ソースオペランドがそろったら，発行要求生成論理により，W フラグが 0 であっても，T-FF が 1 を保持しているため，発行要求が選択論理に送出される．発行が許可されると先行実行が行われ，書き込みは許可されないが，実行結果は依存命令にバイパス論理経由で渡され，次の先行実行を生じさせる．発行許可信号により，T-FF はトグルし 0 となる．これにより以後，W フラグが 1 にならない限り発行要求は生じない．なお，先行実行の結果はレジスタに書かれないので，pexec フラグによりセットされたレディフラグを次のサイクルにクリアする．これは**図 4** のような単純な回路で実現できる．

先行実行による結果はレジスタには書かれないが，多くの場合，バイパス論理経由で連鎖的に先行実行を生じさせることができる．その理由の一つは，1 オペランド命令の割合が多いことが挙げられる．2 オペランドの命令であっても，一方のオペランドがすでにレディであれば，もう一方のオペランドさえ先行実行によって得られれば良い．もう一つの理由は，依存命令間の距離は短く，先行実行された命令に依存する命令が命令ウィンドウ内に存在する可能性は高いことである．

このウェイクアップ論理のクリティカルパスは，コミットされた ROB エントリ番号と ROBP の一致チェックから，発行要求が出されるパスである．従来のウェイクアップ論理のクリティカルパスとの違いは，OR ゲートが 1 段挿入されたことと，発行要求を出力する AND ゲートのファンインが 2 から 3 に増えたことだけである．また，ウェイクアップ論理の 1 エントリは主として ROBP フィールドと W フラグ分長くなるが，W フラグから発行要求生成論理までの配線長に影響するだけである．以上よりウェイクアップ論理の遅延の増加は軽微といえる．

2.3　その他の効果

本方式は，先行実行によるデータプリフェッチ効果以外に次のような効果により性能を向上させることができる．1 段階目の解放はリネーム時に行われるため，物理レジスタ不足で命令の処理がストールすることはない．したがって，命令がリネームから命令ウィンドウに入るまでの間，割り当てられた物理レジスタが以前に割り当てられていた命令の実行と処理をオーバラップさせることができる．この結果，並列度が高まる

3. 評価結果

3.1　評価環境

SimpleScalar Tool Set Version 3.0a[1] をベースにシミュレータを作成し，評価した．命令セットは，MIPS R10000 を拡張した SimpleScalar/PISA である．ベンチマークプログラムとして，データセットの大きなプログラムを含む**表 1** に示す SPECfp2000 の 8 本を使用した．バイナリは，gcc ver.2.7.2.3 を用い，-O6 -funroll-loops のオプションでコンパイルし作成した．入力には ref 入力を用いた．シミュレーション時間が過大にならないように，命令のスキップと実行命令数の制限を行った．

以下のモデルについて評価した：

- **BASE**: 通常のレジスタ解放を行うモデルである．
- **PE**: 2 段階解放によって，命令の先行実行を行うモデルである．

各評価モデルにおいて共通する測定条件を**表 2** に示す．

3.2　IPC の評価

本節では，使用可能なレジスタ数を実行が継続可能となる最小値☆から最大 128 個まで変化させ，IPC を

☆ 最小値とは，論理レジスタ数に 1 つの命令が書き込みに使用するレジスタ数を加えた値である．

表 1　各ベンチマークに与えた入力 (SPECfp2000)

ベンチマーク	入力
ammp	ammp.in
applu	applu.in
apsi	
art	-scanfile c756hel.in -trainfile1 a10.img -trainfile2 hc.img -stride 2 -startx 110 -starty 200 -endx 160 -endy 240 -objects 10
equake	inp.in
mesa	-frames 1000 -meshfile mesa.in -ppmfile mesa.ppm
mgrid	mgrid.in
swim	swim.in

表 2　測定条件

命令フェッチ	最大 8 命令
命令デコード	最大 8 命令 リネームから発行まで 3 ステージ
命令発行	最大 8 命令
命令コミット	最大 8 命令
ROB	128 エントリ
LSQ	64 エントリ
命令ウィンドウ	64 エントリ
機能ユニット	8 iALU, 4 iMULT/DIV, 4 Ld/St, 6 fpALU, 4 fpMULT/DIV/SQRT
命令キャッシュ	64KB, 2 ウェイセットアソシアティブ, ライン長 32 バイト,
データキャッシュ	64KB, ノンブロッキング 2 ウェイセットアソシアティブ, ライン長 32 バイト, 4 ポート, ヒットレイテンシ 2 サイクル
2 次キャッシュ	統合, 2MB, 2 ウェイセットアソシアティブ, ライン長 64 バイト, ヒットレイテンシ 12 サイクル ミスペナルティ70 サイクル
分岐予測機構	gshare 17 ビット履歴, 256K エントリ PHT, 分岐予測ミスペナルティ10 サイクル

測定した．**図 5** に，評価結果を示す．縦軸は IPC のベンチマーク平均であり，横軸は使用可能な整数物理レジスタ数である．整数物理レジスタとして実行が継続可能となる最小値から k 個の物理レジスタを加えた場合，浮動小数点物理レジスタの数は，その最小値に k を加えた数とした．

図より，PE モデルは BASE モデルに比べて，レジスタ数が少ないときに高い性能向上率を達成できることがわかる．物理レジスタ数が 64 個の場合，32%と大きな性能向上を得ることができた．物理レジスタが

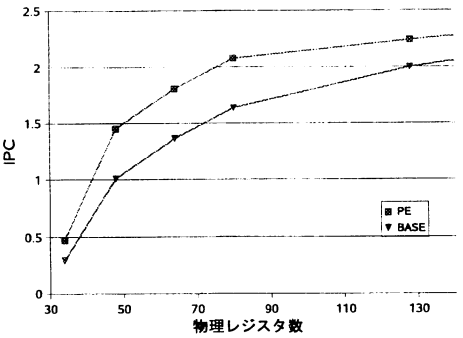


図 5　IPC vs. レジスタ数

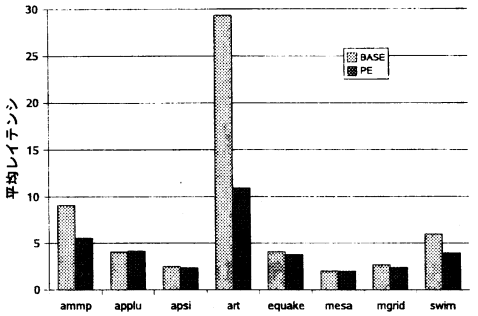


図 6　ロード命令の平均レイテンシ

128 個の時でも 12%の性能向上である．レジスタ数が少ないことによって本実行の速度は制限されるが，先行実行はプログラムの潜在的な ILP を利用して進む．これによってプリフェッチが進むためである．

3.3　先行実行の効果

本節では，ロード命令の先行実行によるデータのプリフェッチの効果を定量的に調べる．

次にデータのプリフェッチ効果を見るために，ロード命令の平均レイテンシを測定した．物理レジスタ数は 64 個とした．測定結果を**図 6** に示す．縦軸はロード命令の 1 次データキャッシュミス率である．PE モデルの場合，先行実行の際のロード命令はカウントしていない．

図からわかるように，ammp, art, swim で非常に大きな削減が見られる．**表 3** に，BASE モデルでのデータキャッシュミス率と，PE モデルでコミットされた命令数に対する先行実行された命令数の割合を示す．大きくレイテンシが削減された理由は，これらのベンチマークで，データキャッシュミス率が高く，先行実行命令数の割合も多いためである．

表 3 データキャッシュミス率と先行実行命令数の割合

ベンチマーク	DC ミス率	先行実行命令
ammp	9%	33%
applu	5%	43%
apsi	1%	16%
art	48%	45%
equake	5%	28%
mesa	1%	18%
mgrid	3%	52%
swim	13%	70%

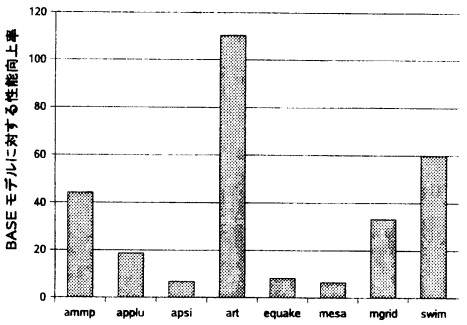


図 7 ベンチマーク毎の正規化した IPC

3.4 ベンチマーク毎の性能

物理レジスタ 64 個の場合を例にとって，BASE モデルに対する性能向上率を示す．**図 7** に測定結果を示す．

図 7 からわかるように，PE モデルでは多くのベンチマークで大きな性能向上が得られている．ammp, art, swim では特に大きな性能向上を達成している．これは，図 6 からわかるように，プリフェッチによりロード命令の平均レイテンシが大きく削減されたことが原因である．逆に apsi, mesa は表 3 からわかるように，キャッシュミス率が非常に小さく，プリフェッチによる性能向上の余地がほとんどない．equake のキャッシュミス率は非常に小さいというわけではないが，先行実行命令数の割合が小さく，プリフェッチが良く行われなかったことから大きな性能向上がないものと思われる．applu はレイテンシの削減はできないものの比較的大きな性能向上がある．これは，クリティカルパス上にあるロード命令のレイテンシが削減されたか，2.3 節で示したオーバラップ効果によるものと思われる．

4. ま と め

本論文では，プログラムに潜在する ILP を利用し，データのプリフェッチを行う 2 段階のレジスタ解放手法を提案した．

本手法では，従来行われているコミットステージでの解放に先立ち，第 1 段階として，リネームステージでレジスタを解放する．命令は，命令ウィンドウにおいて，第 1 段階の解放を受けたレジスタが最終的な解放を受けるのを待ち合わせるが，この時，オペランドがそろった命令を先行実行する．先行実行が連鎖的に進み，ロード命令に至ることによりロードデータのキャッシュへのプリフェッチを可能とする．

本手法は従来のプリフェッチ手法とは異なり，予測ではなく，必要なデータのみをプリフェッチする．また，プログラムに潜在する ILP と実際の ILP の差を利用したプリフェッチであるため，適切なタイミングでデータを得ることができる．

SPECfp2000 ベンチマークを使い，評価を行った結果，本手法の適用により，プリフェッチが有効に行われ，高い性能向上を達成することが可能であることがわかった．たとえば，64 個の物理レジスタの場合，本手法を適用することにより，その性能を SPECfp2000 でそれぞれ平均 32%も大きく改善できることを確認した．

謝辞 本研究の一部は，文部科学省科学研究費補助金基盤研究（C）課題番号 15500036，文部科学省 21 世紀 COE プログラム，財団法人大川情報通信基金の支援により行った．

参 考 文 献

[1] D. Burger, et al. The simplescalar tool set version 2.0. Technical Report CS-TR-97-1342, University of Wisconsin, 1997.

[2] R. P. Preston, et al. Design of an 8-wide superscalar risc microprocessor with simultaneous multithreading. *2002 IEEE Int. Solid-State Circuits Conference Digest of Technical Papers*, pp. 334–335, February 2002.