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

2008－ARC－180 (2)
2008／10／20

値予測を用いた物理レジスタ2段階解放による命令先行実行方式の性能向上

田中雄介† 安藤秀樹†

我々はこれまでに，物理レジスタの2段階解放 (TSD: Two-Step Physical Register Deallocation) と呼ぶ手法を提案している．TSD 手法では，本来であれば物理レジスタを割り当てることができず，実行ができない命令を先行実行する．これにより理想的には，物理レジスタが十分に存在する場合と同等のメモリ・レベル並列性 (MLP: Memory-Level Parallelism) を引き出すことが可能となる．しかし，実際には，種々の原因で先行実行できない場合や，先行実行のタイミングが遅れる場合があり，MLP の抽出は十分でなかった．原因の本質はデータ依存にある．そこで本論文では，値予測を用いることでデータ依存を削除し，より多くの命令が先行実行を可能とする手法を提案する．SPECfp2000 ベンチマークを用いて評価した結果，提案手法は従来の TSD 手法に比べて，75%のレジスタ・ファイルのサイズで同等の性能を達成できることを確認した．

Increasing the Effectiveness of Instruction Pre-Execution with Two-Step Physical Register Deallocation via Value Prediction

Yusuke Tanaka† and Hideki Ando†

We previously proposed *Two-step physical register deallocation* (TSD) scheme. The TSD pre-executes instructions that cannot be executed due to lack of a physical register in traditional architecture. This allows extraction of an equivalent amount of memory-level parallelism (MLP) to that in the case with an enough number of physical registers ideally. In practice, however, extraction of MLP is currently insufficient, because there are cases that pre-execution is not performed or timing of pre-execution is delayed due to several causes. The bottom line of the causes is data dependencies. This paper proposes use of value prediction to remove data dependencies, which allows more instructions to be pre-executed. Our evaluation results using SPECfp2000 benchmark show that our scheme can achieve equivalent performance to that of the TSD scheme without value prediction with 75% in the size of the register file.

1. はじめに

一般に，プロセッサがサポートする in-flight 命令の数を増加させれば，命令レベル並列性 (ILP: Instruction-Level Parallelism) およびメモリ・レベル並列性 (MLP: Memory-Level Parallelism) をより多く利用でき，スーパスカラ・プロセッサの性能を改善することができる．特に，メモリ・インテンシブなプログラムでは MLP 利用の効果は大きい．しかし，一般に，多くの in-flight 命令を保持するためにはそれだけ大きなリオーダ・バッファとレジスタ・ファイルを必要とする．これらの資源はクリティカル・パスであるため，クロック速度に大きな影響を与える．このため，リオーダ・バッファとレジスタ・ファイルの大きさには限界があり，従来のプロセッサではプログラムに内在する MLP を十分に引き出すことができるほどの in-flight 命令をサポートすることはできなかった．

これに対し我々は，**物理レジスタ2段階解放** (TSD: Two-Step Physical Register Deallocation) 手法と呼ぶ手法を提案した[4]．TSD 手法は，物理レジスタの解放を書き込み許可と割り当て許可の2段階に分け，当該レジスタへの書き込みが可能となるよりも早いタイミングで命令への割り当てを許可する．これにより，レジスタ・ファイルを従来よりも小さくしてもレジスタ不足によるストールを起こすことなく，命令をプロセッサ内に取り込むことが可能となる．このような命令は結果の書き込みはできないが，命令を実行することはできる．TSD 手法では，結果の書き込みが許可される前に仮の実行 (先行実行) を行い，書き込み許可が得られた後に改めて本来の実行 (本実行) を行う．この先行実行が理想的に行われた場合には，物理レジスタが十分に多く存在する場合と同じタイミングでメモリ・アクセスを行うことが可能となる．これにより，従来よりも小さなレジスタ・ファイルで多くの MLP を引き出すことができる．

TSD 手法では先行実行を行う命令が多いほど多くの MLP が利用できるが，以下の3つの原因により先行実行が妨げられるという問題がある．第1に，先行

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

実行の結果は保持することができないため，バイパス論理を介する以外に先行実行の結果を後続命令に伝達することができない．これにより，依存する命令の先行実行ができない場合がある．第2に，あるロードがキャッシュ・ミスを起こした場合に結果が得られるまで多くのサイクルが必要とするため，それに依存する命令の先行実行の先行性が損なわれる．第3に，データ依存の厳しいプログラムでは，本質的に先行実行の先行性を十分に確保できない．

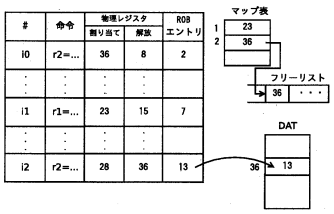
本論文では，これらの問題を解消する手法として値予測を用いる．命令の実行結果を予測することで，先行する命令との依存関係とは無関係に先行実行を行うことが可能となる．あるいは，実行結果ではなくメモリ命令の参照アドレスを予測すれば，ソースが利用可能でない状態でもロードの先行実行ができる．これによって，従来のTSD手法では先行実行できなかったロードを先行実行することが可能となり，より多くのMLPを引き出すことが可能となる．

本論文では，2節で関連研究を述べ，3節でTSD手法の紹介を行う．4節で従来の値予測の利用法の問題点を述べ，5節でTSD手法に値予測を適用した場合の効果について説明する．そして，6節で評価を行い，7節でまとめる．

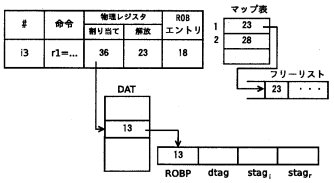
2. 関連研究

先行実行の研究は数多くあるが，そのほとんどはマルチスレッド環境を必要とする．TSD手法と同じく単一のスレッド環境で先行実行を行う手法としてRunahead実行[2]がある．この手法は，L2キャッシュ・ミスを起こした命令に依存する命令によって命令ウィンドウが塞がれてしまった時に，当該命令に不正を示す値を与えて一度命令ウィンドウから退出させ，後続する命令の実行を進めていくことで命令の先行実行 (Runahead実行) を行う手法である．この時，起点となるL2ミスしたロードに予測により結果を与えれば，それに依存する命令も先行実行でき，Runahead実行の効果を高めることができる．Mutluらは，ロードの参照アドレスから結果を予測する手法として，Address-Value Delta (AVD) 値予測[3]を提案した．AVD値予測は，ロードの参照アドレスと読み出されるデータの差が一定であることが多いことを利用しロードの結果を予測する手法である．

Zhouらは，値予測をILPの増加ではなくMLPの増加に利用する手法[5]を提案した．この手法では，値予測を行ったかどうかに関らず，全ての命令は通常通り実行され，値予測はデータ依存の削除ではなく，キャッシュのウォームアップのために用いられる．



(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

3. TSD手法

本節ではこれまで我々が提案したTSD手法[4]について説明する．

3.1 物理レジスタの2段階解放

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．

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 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．

3.2 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実行することで，本実行に先立つ先行実行ストリームを生成することができる．

先行実行を行うため，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．結果をバイパス論理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，物理レジスタが十分にある場合と同じタイミングで命令を実行することができる．

ここで，ロードが先行実行された場合について考える．もし，このロードがキャッシュ・ミスを起こした場合，物理レジスタが十分にある場合と同じタイミングでメモリからキャッシュへデータをフェッチすることが可能となる．この効果により，TSD 手法を用いることで，物理レジスタが少ない場合でも，物理レジスタが十分にある場合の MLP を利用することが可能となる．

4. 従来の値予測の利用法

値予測は，命令の過去の振舞から実行結果を予測する手法である．実行結果を予測することによってデータ依存を削除し，命令の投機的な並列実行により実行サイクル数を削減できる．

しかし，実際には，実行サイクル数の削減はそれほど容易ではない．その要因の 1 つは，値予測の誤りによる投機失敗からの回復方法にある．投機失敗からの回復方法として最も一般的なものは，分岐予測ミスからの回復と同じく，投機に失敗した命令に後続する命令をすべて削除し，投機に失敗した命令から再実行するという手法である．この手法は，値予測が分岐予測ほど精度が高くないため，回復のペナルティにより得られた利益の多くを失ってしまうという問題がある．投機に失敗した命令に依存する命令のみを選択的に再発行する手法[6] を用いれば，ペナルティは小さいが，複雑な機構を必要とする．

5. TSD 手法における値予測の利用

従来の TSD 手法では，以下に述べる場合には先行実行ができないか，先行実行ができたとしても十分な効果を得ることができないという問題があった．

まず，先行実行時に実行結果を後続命令に伝達できなかった場合である．先行実行命令は結果を物理レジスタに書き込むことができないため，バイパス論理によってしか依存する後続命令に結果を伝達することができない．このため，バイパスによって結果を伝達できなかった場合には，当該命令の本実行が行われるまでその命令に依存する命令の先行実行を行うことができなくなってしまう．

次に，先行実行した命令がキャッシュ・ミスを起こした場合である．先行実行が行われたロードがキャッシュ・ミスを起こした場合には，結果が判明するまでに長いレイテンシを要する．このため，キャッシュ・ミスを起こしたロードに依存する命令は，先行実行できたとしても本実行に対する先行性が損なわれる．TSD 手法では，先行実行命令の先行度が大きいほど，キャッ

シュ・ミス時の長いレイテンシを隠蔽することができる．よって，先行する命令のキャッシュ・ミスによって先行度を確保できなくなってしまう場合には十分な効果を得ることができない．

最後に，プログラム中のデータ依存が厳しい場合である．この場合も，先行実行は十分な先行度を確保することができなくなってしまうため，十分な効果を得ることはできない．

本論文では，これらの問題を解決するために値予測を利用する手法を提案する．値予測を用いることで，データ依存を無視して命令を実行することが可能となる．これにより，従来の TSD 手法では先行実行できなかったり，十分な先行性を確保できなかった場合にも，より効果的な先行実行を行うことが可能となる．

ここで，本手法では従来の値予測利用において必要とされていた状態の回復が不要であることに注意されたい．4 節で述べたように，値予測の有効性を低下させている要因は，投機失敗時の回復動作である．これに対して，先行実行時に値予測を用いる本手法の場合，投機に失敗したとしてもキャッシュの状態が変更される以外には副作用がない．このため，投機失敗からの回復を必要とせず値予測を利用することが可能である．

本論文では値予測の適用方法として 2 通りを試みる．1 つは，命令の実行結果を予測するものである．デスティネーション・レジスタを持つ全ての命令の結果の予測を試みる (後で述べるように実際に予測されるのは予測の信頼度が高い場合のみである)．ロードが直接あるいは間接に依存している結果が予測されれば，その結果を使う命令を連続して実行することによってロードを先行実行することができる．なお，ロードの結果を予測するとそのロードについてプリフェッチが行われないから無駄のように感じられるかもしれないが，そのロードに依存するロードがあれば，それが先行実行されるという利点があることに注意されたい．値予測を行うことにより，ある 1 つの命令は複数の値予測の結果により先行実行される可能性が生じる．しかし，そのように複数回先行実行を行っても無駄である．そこで，他の命令の結果が予測され，それが伝達され行われる先行実行と，予測を用いない従来の TSD 手法による先行実行をそれぞれ最大 1 回に制御する．

もう 1 つは，実行結果を予測するのではなく，ロード命令の参照アドレスを予測するものである．実行結果を予測する場合と異なり，予測を行ったロードはすぐにメモリ・アクセスを行うことが可能なため，結果を予測する前述の手法と比べ，依存命令が連続して実行される必要はなく，直接的に値予測の結果を利用できる．ただし，予測不能の場合があるので，一般的には最初の方法よりも値予測の恩恵を受ける機会は減少すると思われる．

表 1 測定条件

Pipeline width	8-instruction wide for each of fetch, decode, issue, and commit
ROB	128 entries
LSQ	64 entries
Instruction window	64 entries
Function unit	8 iALU, 4 iMULT/DIV, 4 Ld/St, 6 fpALU, 4 fpMULT/DIV/SQRT
L1 I-cache	64KB, 2-way, 32B line
L1 D-cache	64KB, 2-way, 32B line, 4 ports, 2-cycle hit latency, non-blocking
L2 cache	2MB, 4-way, 64B line, 12-cycle hit latency
Main memory	300-cycle min. latency, 8B/cycle bandwidth
Branch prediction	6-bit history gshare, 8K-entry PHT, 10-cycle misprediction penalty
Mem. disambiguation	perfect
Value predictor	stride predictor[1], 1024-entry direct-mapped VHT

6. 評価結果

6.1 評価環境

SimpleScalar Tool Set Version 3.0a 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，SPECfp2000 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．SPECfp2000 の入力には ref 入力を用い，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**: 通常の物理レジスタ解放を行い，先行実行を行わないモデル
- **TSD**: 物理レジスタ 2 段階解放によって，命令の先行実行を行うモデル
- **TSD_addr_pred**: TSD モデルにアドレス予測に基づく先行実行を加えたモデル
- **TSD_result_pred**: TSD モデルに結果予測に基づく先行実行を加えたモデル

ただし，現在のところ，TSD_result_pred モデルにおいては，値予測の結果が誤っていた場合も，正しい結果が後続命令に受け渡されるという楽観的なシミュレーションとなっている．

測定環境を**表 1** に示す．値予測のための履歴表

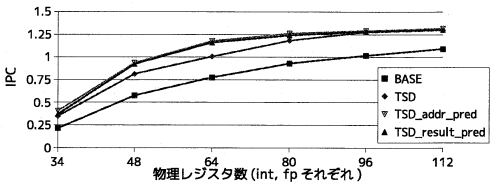


図 2 物理レジスタ数を変化させた場合の IPC

(VHT:Value History Table) の各エントリは，タグ，直前の値，ストライド，信頼度を持つ．信頼度は 1 ビットのフラグであり，直前の予測の成否によって予測を用いるかどうかを判断する．

メモリ曖昧性を完全に除去できるとした．TSD 手法では，先行実行時にロード・アドレスを計算することで，本実行時のメモリ曖昧性を早期に除去することが可能である．公平な評価のためには，Base モデルに既存のメモリ曖昧性除去方式 (たとえば，メモリ依存予測) を導入することが必要となるが，本評価ではそれを行わずに，全てのモデルでメモリ曖昧性を完全に除去するという方法を選択した．

6.2 物理レジスタの削減

図 2 に，物理レジスタ数 (int，fp それぞれ) を 34 から 112 まで変化させた場合の各モデルにおける IPC の幾何平均を示す．グラフより，TSD 手法による先行実行によって，物理レジスタ数を減少させていっても性能低下が抑制されていることがわかる．また，その効果は値予測を導入することによってさらに向上している．これにより，Base モデルと同一の性能をより少ない物理レジスタで達成できる．

ここで，物理レジスタの削減率の代表値を求めるために，基準となる物理レジスタ数を定める．総物理レジスタ数の基準値を，以下の式により求めた．

$$Npreg = ROBsize + Nlreg \qquad (1)$$

ここで，$Npreg$ は総物理レジスタ数，$ROBsize$ は ROB のサイズ，$Nlreg$ は総論理レジスタ数である．上式で得られる総物理レジスタ数の基準値は，以下の点で妥当である．1) ROB サイズはサポートする in-flight 命令の数を定め，ほとんどの in-flight 命令は書き込みレジスタとして物理レジスタを要する．2) コミットされた各論理レジスタは物理レジスタを要する．次に，得られた基準数を int，fp 物理レジスタに等しく分割する．これは，評価に用いたベンチマークにおいて，デスティネーション・レジスタが int または fp である割合がほぼ等しいからである．図 2 の評価では，総論理レジスタ数は 64，ROB サイズは 128 であるから，式 (1) を用いて，基準となる物理レジスタ数は，int, fp それぞれ 96 となる．

グラフより，基準の物理レジスタ数を持つ Base モデルとほぼ同等の性能を，従来の TSD 手法では物理レジスタ数 64 で，結果予測およびアドレス予測を用いた手法では物理レジスタ数 48 で，達成できることがわかる．つまり，従来の TSD 手法では物理レジスタ削減率は 33%であったが，値予測を用いることにより 50%にまで拡大できた．

アドレス予測を用いた場合と結果予測を用いた場合では，性能にほとんど変化がない．これは，ロードの参照アドレスがストライド・パターンを持ち予測可能な場合は，そのアドレスの生成に関与する命令の結果もストライド・パターンを持っているが，逆に，アドレスの生成に関与する命令の結果がストライド・パターンを持っており予測可能な場合で，当該アドレスがストライド・パターンを持っていない場合がほとんどないためと考えられる．たとえば，ループの誘導変数から配列のインデクスが生成される場合が前者に相当し，ループの誘導変数からリンク・リストのポインタが生成される場合が後者に相当する．明らかに，前者のようなコードは多いが，後者のようなコードは少ない．

6.3 値予測利用率

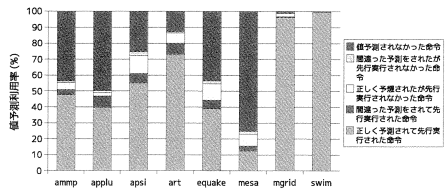
本節では，値予測を行った命令のうち，実際に先行実行に利用された命令の割合 (値予測利用率) を評価する．6.2 節で述べたように，BASE モデルで物理レジスタ数を 96 とした場合を基準とし，それとほぼ等しい性能となる場合，すなわち，TSD モデルでは物理レジスタ数 64，TSD_addr_pred モデルと TSD_result_pred モデルでは物理レジスタ数 48 の場合について測定を行った．**図 3** に測定結果を示す．同図 (a)(b) はそれぞれ，TSD_addr_pred モデル，TSD_result_pred モデルの場合である．各棒グラフは，5 つの部分にわかれており，それぞれ，値予測対象の命令 (結果予測では値を出力する全ての命令，アドレス予測ではロード命令) を以下の 5 つに分類した場合に対応する．

- 正しく予測されて先行実行された命令
- 誤った予測をされたが先行実行された命令
- 正しく予測されたが先行実行されなかった命令
- 誤った予測をされたが先行実行もされなかった命令
- 信頼度に従い予測が行われなかった命令

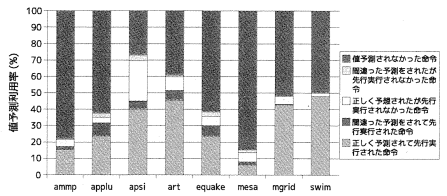
先行実行されない場合とは，値予測を行う時点で命令のソースが利用可能であった場合である．

グラフより，ほとんどのプログラムで値予測が行われた命令の大部分は先行実行されていることがわかる．

図 3(a) からわかるように，多くのプログラムでストライド予測のカバー率 (予測された命令の割合) は高く，ヒット率も高い．これは SPECfp2000 のような科学技術計算プログラムでは，メモリのアクセス・パ



(a) TSD_addr_pred モデル



(b) TSD_result_pred モデル

図 3 値予測利用率

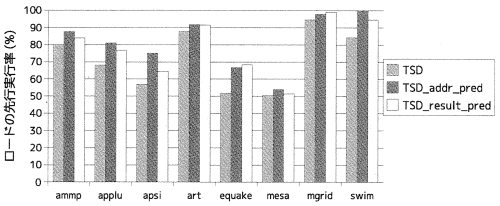


図 4 ロードの先行実行率

ターンには，ストライド・パターンが多いことを意味している．これに対して，ストライド・パターンに対応できる自動プリフェッチャは，ストライド予測器と同等のコストで容易に実現できる上，データ・フェッチ速度はより高い．なぜなら，値予測による手法では，命令フェッチ速度でデータ・フェッチ速度が制限されるが，自動プリフェッチャにはそのような制限はないからである．一方，自動プリフェッチャは，プリフェッチのタイミングが早すぎる，また，不正なデータのプリフェッチによるキャッシュの汚染という問題がある．以上のトレードオフを考慮に入れ，今後 Base モデルに自動プリフェッチャを実装し，評価していかなければならない．

アドレス予測に比べ結果予測は，カバー率が低い．よりカバー率が高く，精度の高い予測器，たとえば，コンテキスト・ベース予測器の導入が必要かもしれない．

6.4 ロードの先行実行率

本節では，全ての動的ロードのうち，1 度でも先行実行によるメモリ・アクセスを行ったロードの割合 (ロードの先行実行率) を評価する．本節でも 6.3 節と同様に物理レジスタ数を 48 として評価を行っている．**図 4** に，TSD モデル，TSD_addr_pred モデル，TSD_result_pred モデルの 3 モデルにおけるロードの先行実行率を示す．

グラフからわかるように，いずれのプログラムにおいても提案手法は従来の TSD 手法と比べてより多くのロードを先行実行できている．2 つの提案手法の間では，equake と mgrid では TSD_result_pred モデルの方が先行実行率が高いが，全体では TSD_addr_pred モデルの方がより高い先行実行率が得られていることがわかる．これは，6.3 節で述べたように結果予測のカバー率がアドレス予測よりも低いことに加え，バイパスによる結果の伝達だけでは先行実行を続けていくことが難しいためであると考えられる．

7. ま と め

本論文では，我々がこれまでに提案した TSD 手法において，値予測を用いることで従来は先行実行できなかった命令を先行実行可能にする手法を提案した．SPECfp2000 ベンチマークを用いて評価を行った結果，物理レジスタ数を 96 とした基準プロセッサと同等の性能を得るためには TSD 手法では 64 の物理レジスタが必要であったのに対し，提案手法によって必要な数を 48 まで削減することが可能となった．

謝辞 本研究の一部は，日本学術振興会 科学研究費補助金基盤研究 (C)（課題番号 19500041）による補助のもとで行われた．

参 考 文 献

[1] T. F. Chen, et al., Effective hardware-based data prefetching for high-performance processors, *IEEE Transactions on Computers*, Vol. 44, No. 5, pp. 609–623, May 1995.

[2] O. Mutlu, et al., Runahead execution: An effective alternative to large instruction windows, *IEEE Micro*, Vol. 23, No. 6, pp. 20–25, Dec. 2003.

[3] O. Mutlu, et al., Address-value delta (AVD) prediction: increasing the effectiveness of runahead execution by exploiting regular memory allocation patterns, In *Proc. MICRO-38*, pp. 233–244, Nov. 2005.

[4] A. Yamamoto, et al., Data prefetching and address pre-calculation through instruction pre-execution with two-step physical register deallocation, In *Proc. MEDEA 2007*, pp. 33–40, Sep. 2007.

[5] H. Zhou, et al., Enhancing memory level parallelism via recovery-free value prediction, In *Proc, ICS-17*, pp. 326–335, June 2003.

[6] 佐藤寿倫，命令再発行機構によるデータアドレス予測に基づく投機実行の効果改善，情報処理学会論文誌, Vol. 40, No. 5, pp. 2093–2108, 5 月 1999 年.