C-002

先行実行に基づく実行フェーズの削減に関する研究
A Study on Reduction of Execution Phases Based on Pre-Execution

下村 佳生 †　　小林 良太郎 †
Yoshio Shimomura　　Ryotaro Kobayashi

1. はじめに

近年のマイクロプロセッサでは高い性能を達成するために並列性を使用する．並列性は値予測のような様々な依存によって制限されている．真の依存を解消するために，値予測 [1] と呼ばれる技術を利用する．結果待ちのオペランドが予測できれば，その命令は実行可能になる．

値予測は実行結果を**値履歴表** (VHT：Value History Table) に保存する．命令のフェッチ時，VHT を使って命令の実行結果を予測する．予測値が得られた場合，実行結果をオペランドとして使用する命令を投機的に実行することが可能になる．命令の予測値が，後の実行によって誤っていると明らかになった場合，予測値に基づいて実行した命令を再実行する．

従来機構ではフェッチステージにデスティネーションを予測するのが一般的である．我々はこれをソースについても予測を行うように拡張する．ソースの予測が行えた場合，パイプラインの外側で，それらの値を入力として命令を先行実行し，結果を保存する．もしソースオペランドの予測が成功していれば，その命令の実行フェーズを削除することができる．

2. ストライド値予測

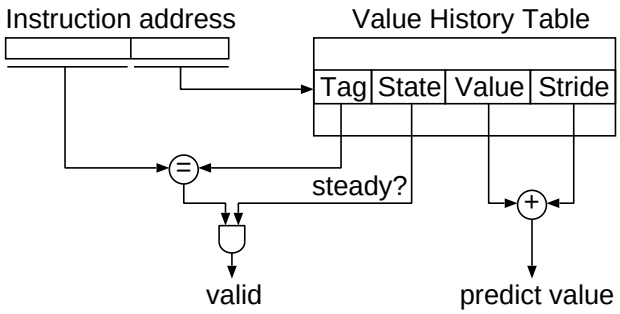

図 1　値予測機構

ストライド値予測は，命令の実行結果と，前回からの差分を基に，次回の実行結果を予測する．VHT の各エントリは Tag，State，Value，Stride を持つ．Tag は命令アドレスの上位ビットである．Value は命令の実行結果である．Stride は過去 2 回分の実行結果の差分である．State はエントリの状態を示し，以下のいずれかになる：“初期”，“遷移”，“定常”．それぞれ，ストライドが得られていない状態，ストライドが一定でない状態，ストライドが一定である状態を示す．

まず参照時の動作を示す．図 1 にストライド値予測機構の予測値の取得方法を示す．命令のフェッチ時，命令の下位ビットをインデクスとして VHT を参照する．命令の上位ビットと保存されているタグが一致し，状態が “定常” であれば予測が有効となり，保存されている値とストライドを足しあわせたものを予測値として得る．上位ビットとタグが一致しない場合や，状態が “定常” でない場合は予測を行わない．

次に更新時の動作を示す．命令のコミット時，命令アドレスの下位ビットをインデクスとして VHT にアクセスする．VHT に対応するエントリがない場合 (タグが一致しない場合)，命令の上位ビットをタグとして登録し，状態を “初期” にセットする，Value に実行結果を書き込む．対応するエントリがある場合，保存されている値と今回の実行結果からストライドを算出し，保存する．ストライドの値が前回のストライド値と一致した場合，状態を “定常” にセットする．前回ストライドが保存されていない場合や，一致しなかった場合，状態を “遷移” にセットする．

予測値が得られた場合，その命令に依存する後続の命令を投機的に実行することが可能となる．依存元の命令の実行結果が判明した時，予測値と一致しなければ，投機実行の結果も誤っているため，投機実行を行った命令と，それに後続する命令を再実行する．予測値が正しければ，投機実行の結果も正しいため，真の依存が解消でき，プログラムの並列性が向上する．

3. 提案手法

従来の値予測では，命令のデスティネーションを予測し，依存関係にある後続の命令を投機実行することによって，速度向上を図る．しかし，正しい予測値が得られた場合でも，依存関係にある命令がフェッチされる前に実行が終了している場合は，値予測によるメリットが得られない．例えば依存関係にある命令 `i1`, `i2` があった時，`i1` の正しい予測値を得られる状態にであり，`i1` の実行が終わる前に `i2` が実行待ちになった場合，投機実行により性能が向上する．しかし `i2` が実行待ちになる前に `i1` の実行が終了すると，値予測によるメリットはない．

† 豊橋技術科学大学大学院工学研究科
Graduate School of Engineering, Toyohashi University of Technology

本研究では，VHT を拡張し，ソースレジスタの値についても予測を行うように拡張する．VHT の更新時にソースレジスタが予測可能な状態になった場合，専用の ALU を用いて予め先行実行を行い，実行結果を VHT をに保存する．命令のフェッチ時，VHT を参照し，先行実行が行われている命令であれば，命令の実行をスキップできる．ソースの予測値が正しいことが確認出来れば，先行実行は成功である．ソースの予測値が誤っていた場合，スキップした命令と，その命令に後続する命令を再実行する．

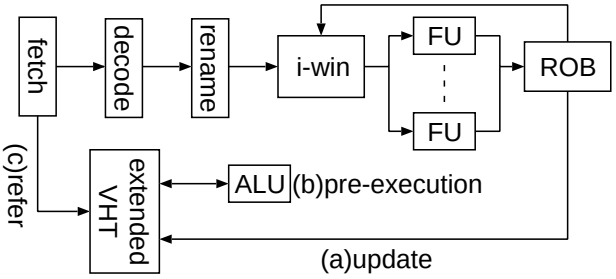

図 2　提案機構

図 2 に提案機構を適応した場合のプロセッサ構成を示す提案機構の VHT では，従来機構のフィールド (Tag, Value, Stride, State) に加え，ソース予測用のフィールドと先行実行用のフィールドを追加する．(Value_S1, Stride_S1, State_S1, Value_S2, Stride_S2, State_S2, EF).

VHT の更新時に，ソース値の両方が予測可能 (定常状態) になれば (図 2(a)) 専用の ALU を用いて先行実行を行い，Value フィールドに実行結果を保存する (図 2(b))．また，EF フラグをセットする．プログラムのフェッチ時，Tag がヒットし，EF フラグがセットされていれば，先行実行により実行結果が算出されているため，その命令の実行をスキップする (図 2(c))．オペランドの予測値が得られなかったが，デスティネーションの予測値が得られた場合，従来の値予測と同様の動作をする．

4.　評価

評価には SimpleScalarToolSet[2] のプロセッサ・シミュレータに本機構を組み込んだものを使用した．命令セットには MIPS R10000 を拡張した SimpleScalar/PISA を用いた．ベンチマーク・プログラムは, SPECint2000 から 8 本を使用した：bzip2, gcc, gzip, mcf, parser, perl, vortex, vpr．gcc では 10 奥命令，その他では 20 億命令をスキップした後，1 億命令を実行した．評価モデルとして以下のものを使用する．

- **基準モデル**
  値予測を行わない通常のスーパスカラ・プロセッサ．

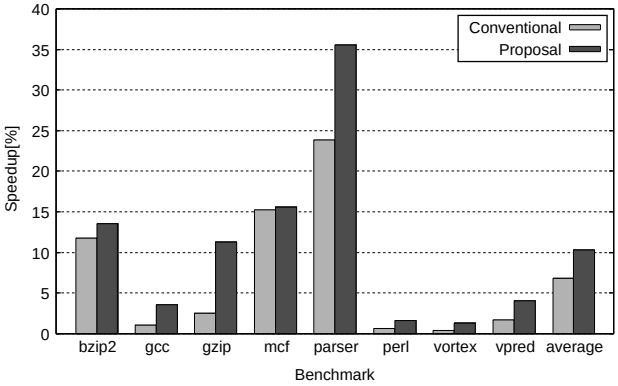

図 3　IPC 向上率

- **従来モデル**
  値予測を備えたスーパスカラ・プロセッサ．
- **提案モデル**
  提案機構を備えたスーパスカラ・プロセッサ．

図 3 に，値予測を使用した場合の置いて，提案モデルがプロセッサ性能に与える影響を評価した結果を示す．図の縦軸は，基準モデルに対するプロセッサ性能の向上率である．図の横軸はベンチマークを示す．2 本で組になっている棒グラフは左が従来モデルの場合，右が提案モデルの場合である．

提案モデルの速度向上率は，従来モデルに対し，平均 3.5 ポイント向上する．また，parser においては向上率が最も高く，11.7 ポイントにもなる．従来モデルではあまり性能向上が見られなかった vortex や perl においても，提案モデルではある程度の速度向上が見られるようになった．

5.　まとめ

ストライド値予測を拡張し，ソース値を予測することによって先行実行する機構を提案した．本提案手法により，値予測の効果をさらに向上させることが可能となった．評価の結果，従来モデルに比べ，平均 3.5 ポイント，最大 11.7 ポイントの性能向上が得られた．

謝辞

本研究の一部は，日本学術振興会グローバル COE プログラムの支援により行った。

参考文献

[1] F. Gabbay and A. Mendelson, “Speculative Execution based on Value Prediction,” EE Department TR #1080, Technion - Israel Institute of Technology, 1996.

[2] D. Burger and T.M. Austin, “The Simplescalar Tool Set Version 2.0,” ACM SIGARCH Computer Architecture News, vol.25, no.3, 1997.