# 「Control Independence」におけるリソース管理手法の提案

西川直樹† 孟林†† 西岡拓生†† 小柳滋†
†立命館大学情報理工学部 ††立命館大学大学院理工学研究科

## 1 はじめに

近年のスーパースカラプロセッサは，パイプラインが深化傾向にある。分岐予測ミスが発生した場合，パイプラインにおける分岐命令以降の命令をすべてフラッシュする。そのため，分岐予測ミスによるペナルティが，性能向上を阻害する大きな要因となっている。この問題を解決する手法として近年，Control Independence(CI) が注目されている。CI とは分岐に依存しない命令 (CI 命令) を再利用する手法である。これにより，ペナルティの軽減が期待できる。本研究では CI において，再利用できる可能性の高い命令を限定して保存する手法を提案する。これにより，リソースの最適化，並びに演算器等への負担軽減を目指す。

## 2 Control Independence の概要

ここでは，図1に示すような分岐命令を含む命令列を例として CI を説明する。命令 (E～F) は分岐の合流点なので，分岐の成否に依存しない。よって CI 命令である。プロセッサは分岐予測器の予測に応じて投機的に命令をパイプラインに取り込む。CI では，パイプライン中の合流点以降の命令を一時的に専用のバッファ等に保存する。そして予測ミスが発覚した後，フェッチした命令が合流点に達した際に再利用する。しかし，CI 命令の中にはデータ依存を持つ命令も存在する。図中 F のレジスタ r3 は D に対して依存を持つ。この際，F を再利用するためには再度，レジスタリネーミング，及び発行を行う必要がある。

先行研究では CI 実現のために，データ依存解消を中心とした様々な手法が提案されている。Walker[1] は CI 研究の先駆けであり，分岐予測ミスが発覚した時に，実行中の CI 命令すべてについてレジスタの再リネーム，及び変更のあった命令に関して再発行を行う。SBR[2] はデータ依存解消を行うが，分岐パターンが限定されている。そのため適応箇所が少ない。Skipper[3] は，低信頼な分岐命令をフェッチすると，間を飛ばし，合流点のフェッチ，実行を行う。そして分岐成否が決定した後に、分岐先以降の命令をフェッチ、実行する。Ginger[4] は先行研究それぞれの特徴を取り入れており，ハードウェア機構が非常に複雑になっている。また我々は，生成先が不明なソースオペランドレジスタをリネームする際に，2つのタグを渡すリネーム手法「Dual Renaming」を検討中である。これにより，リネーミング機構が簡略化され，比較的簡単なハードウェア機構での CI 実現が期待できる。

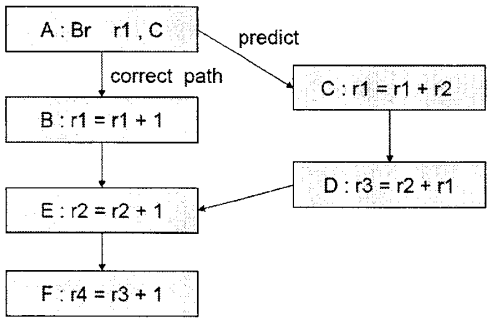


図 1: CI 例

## 3 提案手法

本研究は，保存する CI 命令の再利用率に着目する。命令の分岐は複雑な場合が多く，保存した CI 命令の全てが再利用できるわけではない。再利用できない CI 命令を含む分岐パターンは，大きく分けて2つ存在する。各パターンを図 2-a，図 2-b にそれぞれ示す。図 2-a は合流点が複数存在するパターンである。再び同じ命令へ合流しない場合，CI 命令を保存しても再利用はできない。図 2-b は，CI として保存する命令の中に，分岐命令を持つパターンである。そのため，すべての CI 命令を再利用できる機会は限られる。

再利用できない命令を保存してもリソースの無駄である。また図 2-b において，分岐命令 A における成否結果が，分岐命令 J の予測に変化を与える場合がある。この場合，新しい予測を無視することになり，誤った命令を実行させる可能性が高まる。並びに，リカバリ機構への負担も増加する。本研究は，再利用率の高い CI 命令を保存することを目指し，図 2-b の分岐パターンに注目した。手法としては，CI 命令の中に低信頼な分岐命令が存在する場合，それ以降の CI 命令を保存しないものとする。つまり，図中における四角で囲っ

---

The method of resource management for Control Independence
† Naoki NISHIKAWA
†† Lin MENG
†† Takuo NISHIOKA
† Shigeru OYANAGI
College of Information Science and Engineering, Ritsumeikan University (†)
Graduated School of Science and Engineering, Ritsumeikan University (††)

た命令 (F～J) のみを保存する。しかしこれは，再利用できる CI 命令も捨てる可能性がある事も意味しており，信頼度別による実験結果の考察が重要となる。またこの手法を実現するにあたって，信頼度を判別する専用ハードウェアが必要になるが，本研究ではまず，実行結果を用いてあらかじめ静的予測を行った上で信頼度を判別し，検証する。

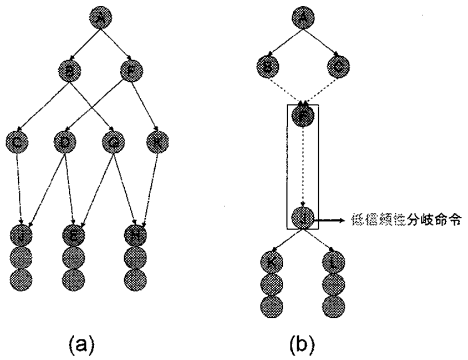


図 2: 不都合な分岐パターン

## 4 評価・考察

### 4.1 評価方法

プロセッサにおける汎用評価ツールである SimpleScalar を用いる。DualRenaming を適用した CI アーキテクチャをデフォルトとし，本手法を追加したものと比較，評価した。ベンチマークには SPECINT95 を使用した。また，前述したように，CI 命令中の各分岐命令における信頼度判別は静的に行う。実行結果をあらかじめ分析し，各分岐命令における予測ミス率を算出する。ミス率が 35，30，25，20%以上である分岐命令をそれぞれ低信頼と定義し，比較を行う。主な評価項目は，保存した命令の有効命令数，無効命令数の比とする。有効命令とは，保存して実際に再利用された CI 命令を指す。いかに有効命令数の比が大きくなっているかを見る。また本手法の性質上，有効命令も保存対象外としてしまう可能性があるため，IPC の低下が予想される。よって，IPC 低下率も加味して評価していく必要がある。

### 4.2 評価結果・考察

評価結果の一部を図 3，及び図 4 に示す。それぞれ，ベンチマーク li95，m88ksim の結果である。信頼度別による有効命令数，無効命令数の比を示している。また, 最大実行命令数は 50 万である。

li95,m88ksim のデフォルトにおいて，共に無効命令は約半数を占めている。本手法適用後には，有効命令数の比が向上していることが分かる。ミス率別に見ると，m88ksim の 25%，20%以上においては，非常に大きな効果が見られる。有効命令の割合は 9 割近くに達している。li95 においては, 有効命令数の比の向上に比べ, 数自体の減少も大きい。しかし,IPC の低下はほとんどみられなかった。

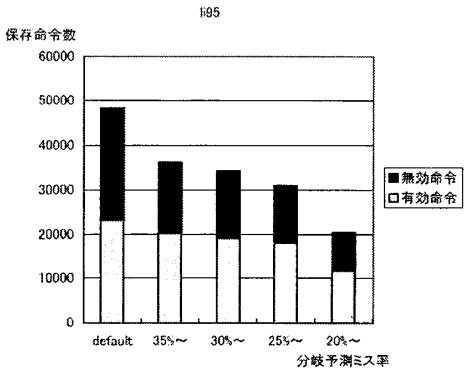


図 3: 評価結果 (li95)

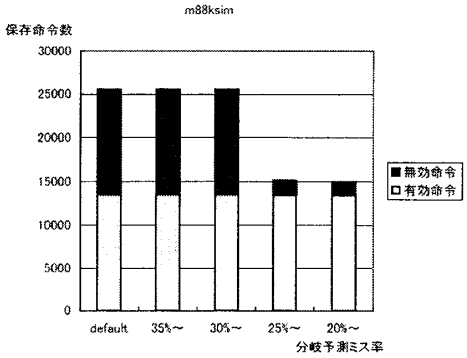


図 4: 評価結果 (m88ksim)

## 5 おわりに

追加する評価項目として，バッファに保存した命令が全て再利用される割合を追加する予定である。向上していれば，3 章に述べたようなメリットが期待できる。また，信頼度の判別を動的に行い，評価出来るように改善する予定である。


参考文献

[1] E. Rotenberg, Q. Jacobsen, and J. Smith. " A Study of Control Independence in Superscalar Processors. " In Proc. 5th Int'l Symposium on High Performance Computer Architecture, pages 115-124, Jan. 1999.

[2] A. Gandhi, H. Akkary, and S. Srinivasan. " Reducing Branch MispredictionPenalty via Selective Branch Recovery. " In Proc. 10th Int'l Symposium on High Performance Computer Architecture, pages 254-265, Feb. 2004.

[3] C. Cher and T. Vijaykumar. " Skipper: A Microarchitecture for Exploiting Control-Flow Independence. " In Proc. 34th Int'l Symposium on Microarchitecture, pages 4-15, Dec. 2001.

[4] A. Hilton, and A. Roth."Ginger: Control Independence Using Tag Rewriting. "In Proc. 35th Int'l Symposium on Computer Architecture, pages 436-447, May 2007.

[5] 安藤秀樹：命令レベル並列処理，コロナ社，(2005)