C-021

関数リターンとの相関を利用する分岐予測方式
A Branch Prediction using the Colleration with Returning from a Function

吉川 唯† 布目 淳‡ 平田 博章‡ 柴山 潔‡
Yui Yoshikawa Atsushi Nunome Hiroaki Hirata Kiyoshi Shibayama

1. はじめに

プロセッサの性能向上を目的として，これまでに様々な分岐予測方式が提案されている．予測対象とする条件分岐命令に至るまでの命令実行パスにおいて，他の条件分岐命令の実行結果が Taken であったか Not-Taken であったか（これを**グローバル履歴**と呼ぶ）によって分岐予測を行う方式[1]や，グローバル履歴を脳神経モデルを用いて処理するパーセプトロン予測方式[2]などが提案されている．

本稿では，このような条件分岐命令間の相関関係に加えて，関数（手続き）からのリターンと条件分岐命令との間の相関関係を利用する分岐予測方式を提案し，分岐予測精度の向上の可能性について評価する．

2. 関数リターンと分岐命令の相関

2.1 構造化プログラムに含まれる冗長性

例として，C 言語で記述した図 1 のプログラムを考える．関数 search() は，あるテーブルに対して，与えられた条件に適合するデータを検索する関数である．当該するデータが存在する場合はそのエントリへのポインタを戻り値として返し，また，存在しない場合は NULL を返す．while ループでテーブル中の各データを対象として処理を行い，ループ中の if 文でそのデータが検索条件を満たすかどうかを検査する．

一方，関数 search() を呼び出す側のプログラムでは，関数 search() が NULL を返すか否かでその後の制御の流れを変更する（例えば，NULL であればテーブルにデータを追加するなどの操作を行う）．しかし，ここでの関数 search() からの戻り値の検査は冗長である．テーブル中に当該データが存在するか否かは，関数 search() 内の if 文ですでに検査済みであり，同じ検査を 2 回行っている．

このような関数間での条件判定の冗長性は，構造化プログラミングによって生じるものであり，一般的に珍しくない．そこで，本稿では，このような冗長性を分岐予測に利用する．

2.2 グローバル履歴を用いた分岐予測

図 1 のソースプログラムをコンパイルした場合の命令実行の流れを図 2 に示す．図 2 中の条件分岐命令 1 を分岐予測の対象として議論を進める．

本稿では，分岐予測対象の条件分岐命令の直前に実行するリターン命令を**先行リターン命令**と呼ぶことにする．また，関数内において，先行リターン命令の直前に実行する条件分岐命令を**先行条件分岐命令**と呼ぶことにする．図 2 の例では，リターン命令 A またはリターン命令 B が先行リターン命令であり，条件分岐命令 2 または条件分岐命令 3 が先行分岐命令となる．

```
/* 関数呼び出し元のルーチン */
  TYPE  *p;
  if((p=search(…))==NULL){ /* 条件判定 1 */
      ……
  }
```

```
/* 関数 search() */
TYPE *search(……){
   ……
   while(……){       /* 条件判定 2 */
       ……
      if(……){       /* 条件判定 3 */
          return q;  /* リターン A */
      }
   }
   return NULL;      /* リターン B */
}
```

図 1　検索プログラムの例（C 言語）

一般に，グローバル履歴を用いて分岐予測を行う場合，予測対象の条件分岐命令に対して，その直近で実行された複数の条件分岐命令の分岐結果をグローバル履歴として利用する．図 2 の例では，特に先行条件分岐命令との相関が強いので，ここでは，先行条件分岐命令の分岐結果のみに着目する．

検索条件を満たすデータがテーブル中に存在しなかった場合，while ループの終了判定（条件判定 2）を行う条件分岐命令 2 の結果が Taken となり，リターン命令 A を実行して関数 search() から戻る．一方，目的のデータがテーブル中に存在する場合は，条件分岐命令 3 の結果が Not-Taken となり，リターン命令 B を実行する．このように，関数 search() が NULL を返すか否かで先行条件分岐命令の分岐結果が異なるため，グローバル履歴を用いることによって，条件分岐命令 1 の分岐予測を正確に行うことができる．

しかし，ソースプログラム中の条件判定に対してどのような条件分岐命令を用いるかは，コンパイラおよびその最適化の結果によって異なる．図 3 は，条件分岐命令の条件指定において，図 2 とは異なる戦略でコンパイラがコード生成した場合の命令実行の流れを示したものである．ループ本体を実行するときは条件分岐命令 2 が Taken となり，また，検索条件を満たさないときは条件分岐命令 3 が Taken となる．従って，検索条件を満たすデータがテーブル中に存在しなかった場合のグローバル履歴は，複数の Taken が続いた後，最後のみ Not-Taken となる．また，目的のデータがテーブル中に存在する場合のグローバル履歴も，複数の Taken が続いた後，最後のみ Not-Taken とな

†京都工芸繊維大学大学院工芸科学研究科情報工学専攻
‡京都工芸繊維大学大学院工芸科学研究科情報工学部門
Dept. of Information Science, Kyoto Institute of Technology

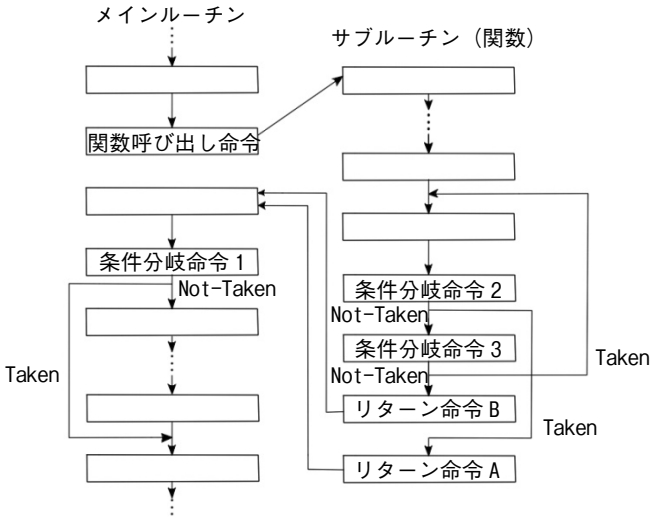

図 2　命令実行の流れ（例 1）

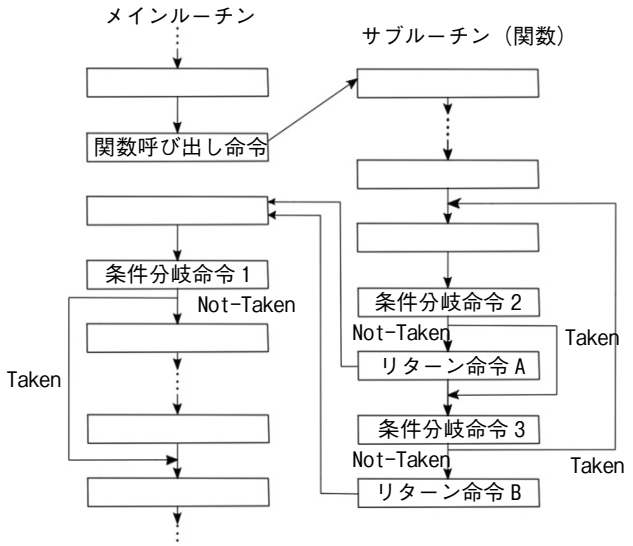

図 3　命令実行の流れ（例 2）

る．これより，図 3 の場合は，条件分岐命令 1 の分岐予測に対してグローバル履歴は役に立たない．

2.3 リターン命令アドレスを用いた分岐予測

2.1 で述べた関数リターン直後の条件判定において，グローバル履歴を用いる従来の方式では，必ずしも正確に分岐予測を行うことができるとは限らない．そこで，本稿では，先行リターン命令のアドレスを用いて分岐予測を行う．

関数リターン直後の条件分岐命令に対しては，過去の分岐結果と関連付けて，分岐予測用のテーブルに先行リターン命令のアドレスの一部を記憶する．

プログラム実行中にリターン命令に遭遇すると，そのアドレスの一部を保存する．リターン命令以外の分岐命令に遭遇すると，保存しているリターン命令アドレスを消去する．分岐予測を行う際に，リターン命令アドレスが保存されていれば，関数リターン直後の条件分岐命令であると判断できる．従って，保存されているアドレスが，分岐予測テーブル内の先行リターン命令のアドレスと一致すれば，それと関連付けた過去の分岐結果を用いて予測する．

3. 評価

本方式を実現するためには，分岐予測テーブルに記憶する先行リターン命令アドレスの数やビット数について検討しなければならないが，今回は，本方式で予測対象とする条件分岐命令が，実際のプログラムでどの程度出現するかを調査する．PowerPC の命令セットを対象とするシミュレータを作成し，SPEC CPU 2006 ベンチマークプログラム[3]中の 403.gcc，429.mcf，401.bzip2（いずれも入力データセットは test）に対して，以下の項目を計測する．

- プログラム中の条件分岐命令の実行回数 B
- 関数リターン後に最初に実行する条件分岐命令で，かつ，どの先行リターン命令が実行されたかによって，分岐方向が決まる条件分岐命令の実行回数 R
- 関数リターン後に最初に実行する条件分岐命令で，かつ，先行条件分岐命令の分岐結果によって，分岐方向が決まる条件分岐命令の実行回数 G

本稿では，R/B をリターン相関比率，G/B をグローバル相関比率，とそれぞれ呼ぶことにする．測定結果を表 1 に示す．今回のベンチマークプログラムの中では，特に，403.gcc において，条件分岐命令の約 4.1%について，本方式で正確な分岐予測が可能となることがわかった．グローバル履歴を用いても予測可能な条件分岐命令は 1.7%存在する（今回は R と G を独立に計測しており，G が完全に R と重複するとは限らない）ので，本方式を用いることにより，2.4～4.1%の条件分岐命令について，分岐予測精度の向上が見込める．

表 1　分岐予測対象命令の出現頻度

	リターン相関比率	グローバル相関比率
403.gcc	4.05%	1.70%
429.mcf	0.75%	0.02%
401.bzip2	0.04%	0.00%

4. むすび

本稿では，関数リターンとの相関を利用する分岐予測方式を提案し，関数リターンと条件分岐命令の相関を調査した．その結果，SPEC2006 ベンチマークプログラム中の 403.gcc において，実行されるすべての条件分岐命令のうち 4.1%について，正確な分岐予測が可能になることを確認した．今後は，実現性も含めて，具体的な分岐予測方式を検討する．

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）25330058）の補助による．

参考文献

[1] S. Pan, K. So, J. T. Rahmeh, “Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation,” Proc. of ASPLOS-V, pp. 76-84 (1992).
[2] D. A. Jimenez, “Fast Path-Based Neural Branch Prediction,” Proc. of MICRO-36, pp. 243–252 (2003).
[3] Standard Performance Evaluation Corporation, “SPEC CPU2006,” http://www.spec.org/cpu2006/.