6X-5 『新風』プロセッサにおける分岐予測

原哲也　久我守弘　村上和彰　富田眞治
(九州大学大学院総合理工学研究科)

1. はじめに

我々は,汎用高速プロセッサ・アーキテクチャとしてSIMP(単一命令流/多重命令パイプライン)方式を提案し,これに基づく試作プロセッサとして『新風』を開発中である[1].『新風』プロセッサは,4本の命令パイプラインによって単一命令流の並列処理を行なう.このため,命令不在に起因するパイプラインの乱れによるペナルティは,単一命令パイプラインの時よりはるかに大きい.したがって命令の供給を行なう命令ブロック・フェッチステージでは,命令不在が生じないように有効な命令を命令パイプラインに供給する必要があり,この要求に対処すべく以下の機能を備える[2].

(1) 命令プリフェッチ機構:命令パイプラインの処理能力に応じた命令の供給
(2) 分岐予測機構:分岐命令による制御依存への対処
(3) 命令再フェッチ機構:分岐命令実行後の効果的な復元処理.

本稿では,分岐予測機構の構成,および,動作について述べる.

2. 分岐予測機構の構成

分岐予測機構は,分岐命令が分岐するか否か(Taken/Not-taken)を予測すると同時に,Takenと予測した際にはその分岐先アドレスを与える.この分岐予測には,分岐ターゲット・バッファ(BTB:Branch Target Buffer)方式を採用している.BTB方式は,対象となる分岐命令が過去において分岐したか否かを示す履歴情報をBTBに登録しておき,それに基づいて次節で述べる分岐予測アルゴリズムに従って分岐予測を行なうものである.

(1) BTBの構成

BTBの構成を図1に示す.BTBはマルチバンク命令キャッシュ内にあり,命令キャッシュと連動する.1キャッシュ・ライン(16命令)と4つのエントリを対応付け,局所性による分岐命令の偏りに対応している.各々のエントリは以下のフィールドから構成される.

① V (Valid):登録されている情報が有効かどうかを示す.
② U (Used):参照されたかどうかを示す.
③ BIA (Branch Instruction Address):登録されている分岐命令のアドレス(4ビット).
④ TA (Target Address):過去に分岐した際の分岐先アドレスで,分岐予測先アドレス(30ビット).
⑤ 履歴部 (History):分岐命令の過去2回の実行がTakenあるいは,Not-takenであったかを示す.

Branch Prediction in the 【ʃimpu:】 Processor
Tetsuya HARA, Morihiro KUGA, Kazuaki MURAKAMI, and Shinji TOMITA
Kyushu University

(2) 分岐予測方法

分岐予測は,命令キャッシュからの4命令(命令ブロック)の読出しと並行して,以下の手順で行なう.

① キャッシュ・ラインから読み出そうとしている4つの命令のアドレス下位4ビットと,そのラインに対応する4個のBTBエントリに登録されているBIAとの比較をそれぞれ行なう.
② もし一致するものがあればその命令は過去にTakenと実行された分岐命令であるので,その履歴部をもとに予測アルゴリズムにしたがって分岐予測を行なう.
③ ②でTakenと予測した場合,当該BTBエントリのTAを次にフェッチすべき命令ブロックアドレスをとして,プリフェッチ・カウンタに渡す.
④ また②でNot-takenと予測した場合,分岐予測機構では何も行わない.

3. 分岐予測アルゴリズム[3]

過去2回の履歴に基づく分岐予測アルゴリズムとして,図2に示すような状態機械で表現される3種類の予測アルゴリズムについて検討する.各状態機械は,ノード内の文字が予測を表わし,tならばTaken,nならばNot-takenと予測する.また,状態遷移を示すアーク上の文字が分岐結果を表わし,TakenならT,Not-takenならNで示されるアークに従って状態を遷移させる.

(1) 分岐命令の分岐パターン

履歴を用いた分岐予測を行なうことができるのは,分岐命令の分岐の仕方にある程度の規則性がある場合である.そこで,ある程度の規則性を持ついくつかの分岐パターンについてのみ,検討する.

分岐パターンとしては,連続する5回の実行結果について①全てTaken,②全てNot-taken,③5回のうち一度だけNot-taken,④5回のうち一度だけTaken,⑤TakenとNot-takenが交互に現れるもの,を挙げる.この5パターンについて,各パターンの存在頻度を調べた結果,これらでほぼ95%を占めていることが分かった.また,この5パターン以外のものは,不規則

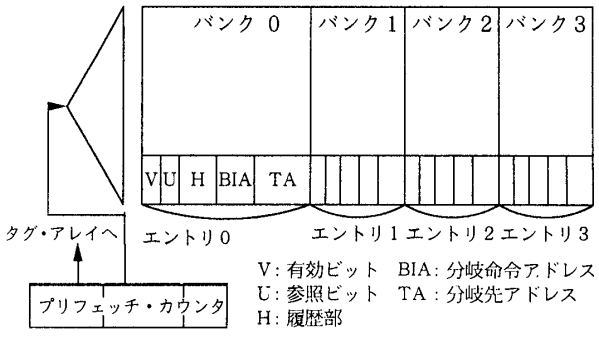

図1 分岐ターゲット・バッファ(BTB)の構成

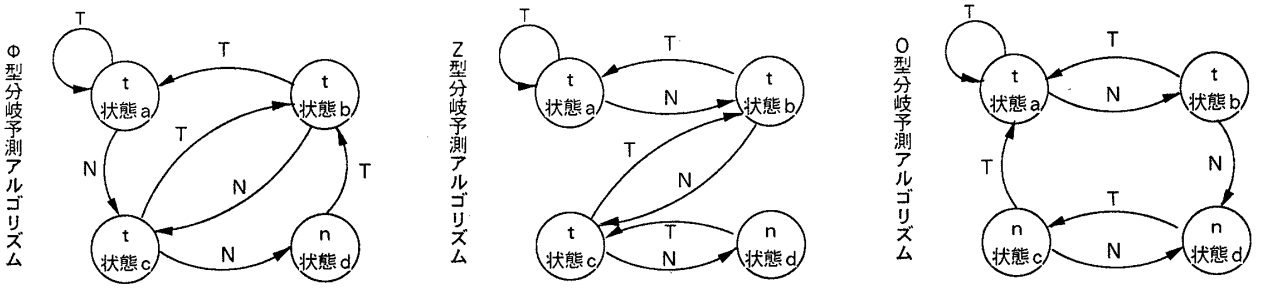

図2 分岐予測アルゴリズム

な分岐を行なうもので予測が難しく，これに対する予測アルゴリズムを検討してもあまり有効ではないと考えられる．したがって，上記の5パターンについて，予測アルゴリズムの有効性を評価する．

(2) アルゴリズムの評価

3つのアルゴリズムに，各分岐パターンを当てはめて，分岐予測がはずれる確率を調べることにより，各アルゴリズムの有効性を検討し，どのアルゴリズムを採用するか決定する．ここで，以下のことを考慮した．

① 初期状態

状態bおよびcを初期状態の候補とする．なぜなら，状態aはTaken，状態dはNot-takenと予測するのに一番安定した状態であり，分岐命令の挙動が不定である初期状態を割り当てるには望ましくないからである．

② 状態dの取扱い

状態dになった分岐命令は，BTBエントリからその登録を外す．なぜなら，状態dはNot-takenと予測する状態であり，この状態になった分岐命令をBTBから追い出しても，結果的にはNot-takenと予測することになるからである．このことによりBTBエントリの有効利用が図れる．

以上のことをふまえて，5つの分岐パターンに対し各予測アルゴリズムを適用した際の予測が外れる頻度を調べた．その結果を表1に示す．これより，Φ型あるいはZ型予測アルゴリズムで状態bを初期状態とするものが，予測が外れる頻度が最も小さいことがわかった．『新風』では，次節で述べる置換アルゴリズムとの関係でZ型予測アルゴリズムを採用し，初期状態は状態bとした．

表1 分岐予測が外れる割合（%）

Φ型アルゴリズム		Z型アルゴリズム		O型アルゴリズム	
(b)	(c)	(b)	(c)	(b)	(c)
6.0	8.0	6.0	8.0	8.3	19.5

初期状態　(b)：(b) 状態に登録　(c)：(c) 状態に登録

4. 置換アルゴリズム

一般には，約4命令に1命令が分岐命令と言われているが，偏りのため1つのキャッシュ・ライン中にTakenとなる分岐命令が5つ以上存在していることもある．このような状況で，置換を行なわないと，毎回参照されTakenとなる命令が，一度しか参照されなかった命令のために，BTBに登録されないという弊害が生じる．したがって，BTBを有効に利用して高い確率で分岐予測が当たるようにするためには，効率的にBTBエントリの置換を行なう必要がある．置換アルゴリズムとして，何を置換の基準にするかで，以下のアルゴリズムを考察する．

(1) 参照状況を置換の基準とするアルゴリズム

置換が必要な状況を実際のプログラムについて調査した結果，実行される分岐命令には偏りがあることが判明した．したがって，よく実行される分岐命令，すなわちよく参照されるエントリを優先的にBTBに残しておく方がよい．置換アルゴリズムとしては，LRU，FINUFOなどがあるが，ハードウェアが比較的簡単なFINUFOを採用する．

(2) 過去の履歴（状態）を置換の基準とするアルゴリズム

各エントリの状態に基づき，分岐しそうにない分岐命令を置換の対象とする．初期状態である状態bを置換の対象とするよりも，状態dに最も近い状態cを置換の対象とするのが効果があると考えた．この場合Φ型予測アルゴリズムでは，ループに代表されるような一度だけNot-takenとなる分岐命令がループから抜けたときに置換の対象となるので好ましくない．したがって『新風』ではZ型アルゴリズムを採用する．

最終的な置換アルゴリズムは以下の通りになる．

① まず，最も参照されていないエントリを置換対象とする．

② 置換対象が複数ある場合は状態cにあるエントリを選ぶ．

5. おわりに

以上，『新風』における分岐予測について述べた．現在，分岐予測アルゴリズムはシミュレーションによって検証中である．また，『新風』プロセッサは実装設計段階にあり，早期完成を目指している．

参考文献

[1] K.Murakami, et al. : SIMP (Single Instruction stream／Multiple instruction Pipelining) : A Novel High-Speed Single-Processor Architecture, Proc.16th ISCA, pp.78-85, May 1989

[2] 入江ほか：『新風』プロセッサにおける命令フェッチ機構，情処38全大論文集，5T－9（1989年3月）

[3] J.K.F.Lee and A.J.Smith : Buranch Prediction Strategies and Branch Target Buffer Design, IEEE Computer, vol.17, no.1, pp.6-22, January 1984