複数命令フェッチに対する並列分岐先予測/命令フェッチ機構

中西 知嘉子[†] 安藤 秀樹[††]
原 哲也[†††] 中屋 雅夫[†]

分岐により実効的な命令供給速度は低下する．これによるマイクロプロセッサの性能低下は，スカラ・マシンに比べ，スーパスカラ・マシンでは特に著しい．高い命令フェッチ速度実現のためには，高い分岐予測精度と予測アドレスの早期送出の2つの要求を満足する必要がある．しかしこれまで，両方の要求を同時に満たす方式の研究は少なく，良い方式が見い出されていない．本論文では，これらの要求を満足する分岐先バッファを用いた命令フェッチ機構を提案する．評価の結果，本方式を用いれば，4命令フェッチのスーパスカラ・マシンにおいて，従来の方式に比べて約38%性能を改善できることを確認した．

A Parallel Target Prediction Mechanism and Instruction Fetch for Multiple-instruction Fetch

Chikako Nakanishi,[†] Hideki Ando,[††] Tetsuya Hara[†††]
and Masao Nakaya[†]

Branches reduce an effective instruction fetch rate. A low fetch rate significantly decreases performance of a superscalar machine. Accurate branch prediction and early supply of the predicted address are both required to sustain a high instruction fetch rate. Few studies, however, have been done to satisfy the both requirements, and thus no good solution has been found. This paper proposes an instruction fetch mechanism to satisfy both requirements using a branch target buffer. Our evaluation results show that our mechanism improves performance by 38% in a four-issue superscalar machine over a conventional mechanism.

1. はじめに

マイクロプロセッサでは，分岐による命令流の分断が，大きな性能低下を引き起こす．スカラ・マシンの場合に比べてスーパスカラ・マシンでは，次の2つの理由で，この問題の重大さは増している．第一に，スーパスカラ・マシンでは論理の複雑化にともないパイプラインが深くなり，分岐ペナルティが増加している．第二に，複数の命令が実行されるため，命令流分断によるパイプラインの停止サイクル数の，全実行サイクル数に占める割合が増加している．

一般に，分岐による命令流の分断を防ぐには，分岐予測を行う．最近では，分岐方向の履歴を要約する表を設け，それを分岐命令のアドレスで参照することにより予測する方法[1)]が一般的である．この表は，予測アドレスをパイプラインの上流で早く供給するために設ける**分岐先バッファ**（**BTB**: Branch Target Buffer）と1つにされ，総称してBTBと呼ばれることが多い．

BTBは分岐命令のアドレスをインデクスとするので，原理的には，最初に現在の命令アドレスに格納されている命令が分岐かどうかを知る必要がある．そのため，命令をデコードしなければBTBを参照できないことになり，予測アドレスの送出が遅れ，平均命令フェッチ速度が低下する．

しかし，BTBに登録されている情報は分岐命令に関するものだけである．よって，分岐かどうか分かる前にBTBを参照し予測を行うことができる．つまり，命令アドレスでBTBを参照し，その命令に関する予測情報がBTBに登録されていれば，そのアドレスの命令は分岐であり，予測情報は有効であると判断できる．スカラ・マシンの場合，この方法で，命令フェッチと分岐予測を並行して行うことができ，予測アドレスを早く送出できる．

† 三菱電機株式会社システムLSI事業化推進センター
System LSI Development Center, Mitsubishi Electric Corporation
†† 名古屋大学大学院工学研究科電子情報学専攻
Department of Information Electronics, School of Engineering, Nagoya University
††† 三菱電機株式会社システムLSI事業統括部
System LSI Division, Mitsubishi Electric Corporation

これに対して，スーパスカラ・マシンの場合，複数の命令を同時にフェッチするため，高い予測精度を維持し，予測アドレスを早く送出する機構を単純なハードウェアで実現することが難しい．たとえば，上記方式を適用する場合，フェッチを行う全命令のアドレスをBTBのインデクスとし参照する必要がある．したがって，同時にフェッチする命令の数に合わせて，BTBに複数の読み出しポートを設なければならず，ハードウェア・コストが大きくなってしまう．

これまで，分岐予測方式に関しては多くの研究がなされてきた（たとえば文献1)，2)）．しかし，スーパスカラ・マシンにおいて，高い予測精度と予測アドレスの早期送出の両方の要求を同時に満たす方式に関する研究は少なく[3),4)]，いまだ良い方法が見い出されていない．

本論文では，上記要求を単純なハードウェアで実現する**複数予測方式**と呼ぶ機構を提案する．以下，2章でBTBを用いた命令フェッチ機構についてまとめる．3章でスーパスカラ・マシンにおける命令フェッチの問題点について考察する．4章で複数予測方式を提案する．5章で評価結果を述べ，6章で関連研究との比較を行う．最後に7章で本論文をまとめる．

2. BTBを用いた命令フェッチ

命令フェッチ速度を分岐により低下させないためには，次の2つの要求を満足する必要がある．

(1) 分岐方向を正しく予測する．
(2) 予測アドレスを早く送出する．

分岐予測は，一般に，分岐方向の履歴を要約した表を参照することにより行われる．この表を**分岐履歴表**（**BHT**: Branch History Table）という[5)]．BHTを参照する際のインデクスとして，分岐命令のアドレスを用いる方法が一般的である．分岐方向の履歴の要約には，通常，0〜3の値をとる飽和型の2ビットのアップ・ダウン・カウンタを用いる．分岐命令実行の結果，分岐成立のときは1増加させ，不成立のときは1減少させる．カウンタ値が2以上のときは分岐成立と予測し，1以下のときは分岐不成立と予測する．この方式を**2ビット・カウンタ方式**[1)]と呼ぶ．

より高い予測精度を達成するために，**2レベル適応型方式**[2)]と呼ぶ方式も提案されている．この方式では，分岐の履歴パターンと分岐命令のアドレスを結合したものをBHTのインデクスとする．

一方，上記(2)を満たすために通常，分岐先のアドレスを記録した表を用意する．この表を分岐先バッファ（BTB）と呼ぶ．BHT，BTBともに命令アドレスをインデクスとするので，これらを1つにまとめる実現がよくとられる（たとえば，Intel Pentium[6)]）．

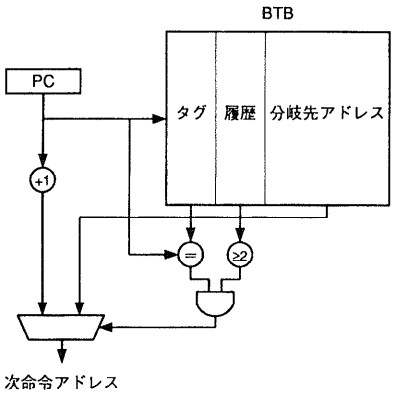

図1 スカラ・マシンにおける命令フェッチ機構
Fig. 1 Instruction fetch mechanism in a scalar machine.

本論文では，分岐方向の予測方式として2ビット・カウンタ方式を用いて議論する．本論文で提案する命令フェッチ機構は，使用する分岐方向の予測方式に依存しないので2レベル適応型方式を用いた場合でも同様に構成できる．以下の議論では，BHTとBTBを1つにまとめた表をBTBと呼ぶこととする．

図1に，スカラ・マシンにおけるBTBを用いた命令フェッチ機構を示す．BTBは，命令アドレスの一部（通常，下位の部分）で参照する．表のエントリには，当該エントリに関連する分岐の履歴を要約するカウンタと，分岐先アドレスを保持する．このほか，BTBのインデクス以外の命令アドレスの部分をタグとして持つ．これは，命令フェッチと分岐予測を同時に行うために，参照したエントリが現在のプログラム・カウンタの値（以下，PCと呼ぶ）に対応するものであることをチェックするためである．

命令フェッチは次のようにして行う．まず，PCに格納された命令アドレスを命令キャッシュに送出するのと並行して，BTBにも送出し読み出す．タグの比較とカウンタの値が2以上（上位ビットが1）かの検査を行う．タグが一致し，カウンタ値が2以上の場合，現在フェッチ中の命令は分岐であり，実行結果は分岐成立であると予測する．この場合，BTBから読み出された**分岐先アドレス**を次のサイクルの命令アドレス（以下，単に**次命令アドレス**と呼ぶ）とする．

タグが一致しないか，カウンタ値が1以下の場合は，現在読み出し中の命令は分岐でないか，または，分岐であっても分岐不成立と予測する．したがって，現在の命令アドレスの静的な次のアドレスを次命令アドレ

スとする☆.

この方式では，命令キャッシュとBTBが同時に参照されるので，分岐命令をフェッチした次のサイクルで，その分岐の動的な次の命令をフェッチできる．したがって，正しく予測された場合，分岐によって命令のフェッチが滞ることはない．

3. スーパスカラ・マシンにおける命令フェッチの問題点

命令をデコードし分岐命令を見つけた後に，その分岐に対してBTBを参照して予測を行う方式（以下，**逐次予測方式**と呼ぶ）では，予測が正しい場合でも，分岐成立時に次命令アドレスの送出が遅れ，ペナルティが生じる（たとえば，Intel Pentium[6]，MIPS R10000[7]）．

正しく分岐が予測された場合のペナルティを0にするためには，2章で述べたように，命令フェッチと並行して現在フェッチ中の命令に関する分岐の予測を行うことが要求される．2章で述べた方式をスーパスカラ・マシンへ単純に拡張するとすると，フェッチするすべての命令のアドレスをBTBのインデクスにする必要がある[8]．しかしこの方式では，同時にフェッチする命令の数だけBTBに読み出しポートが必要であり，BTBが複雑化する．すなわち，ハードウェア・コストが著しく増加し，読み出し速度が低下する．

逆に，単に，フェッチする命令グループの最初のアドレス（以下，**命令フェッチ・アドレス**と呼ぶ）をBTBのインデクスとし，エントリを1つだけ読み出す方式（以下，**単一予測方式**と呼ぶ）が考えられる．4命令フェッチの場合，下位2ビットを除く命令フェッチ・アドレスをインデクスとする．この方式はBTBを複雑化しないと考えられるが，以下のような問題がある．

(1) 複数の分岐の予測が不可能

表1に，我々が評価に用いたプログラムの概要を示す．同表に示すように，基本ブロック☆☆内の平均命令数は2.8~5.8（平均4.4）であり，4命令あるいはそれ以上を同時にフェッチするスーパスカラ・マシンでは，フェッチした命令の中に複数の分岐を含む確率は低くない．したがって，複数の分岐を同時に予測できなければ，それによる性能低下は大きいと考えられる．

☆ 本論文では，命令は1ワードと仮定し，命令アドレスはワード単位とする．

☆☆ 実行の先頭から始まり，必ず最後まで実行される一連の命令からなるブロック．ブロックの途中でブロックを出ることやブロックの途中からブロックに入ることはない．

表1 ベンチマーク・プログラム
Table 1 Benchmark programs.

プログラム	R3000のサイクル数	基本ブロック内平均命令数	機能
bubble	62.0 K	4.6	バブル・ソート
compresss	21.4 M	4.8	データの圧縮
eqntott	27.5 M	3.0	真理値表の生成
espresso	11.4 M	5.3	PLAの最適化
grep	15.7 M	2.8	文字列の検索
li	15.0 M	4.6	Lisp 翻訳
nroff	19.9 M	4.2	文書の清書
qsort	8.9 K	3.9	クイック・ソート
queens	42.7 K	5.2	8クイーン
towers	1.9 M	5.8	ハノイの塔
tree	14.9 K	4.0	2分木ノート

(2) エイリアスされる分岐数増加による予測精度低下

複数の分岐が，BTBの1つのエントリにマッピングされる場合，これらの分岐はそのエントリに**エイリアス**されているという．逐次予測方式では，エイリアスはBTBの大きさに起因するものだけである．これに対して，単一予測方式では，同一エントリにエイリアスされる分岐の数が増加する．たとえば，下位2ビットを除く命令フェッチ・アドレスをインデクスとする場合，連続する4つのアドレスに複数の分岐があれば，それらは1つのエントリにエイリアスされる．連続するいくつの命令を1つのエントリにマッピングするかを，以下，**エイリアス数**と呼ぶことにする．エイリアス数増加により，BTBのヒット率（参照したい情報がBTBに存在するかどうか）が低下する．また，エイリアスされた分岐間で頻繁に登録の入れ替えが生じると，履歴情報が蓄積されないので，予測精度が低下する．

(3) フェッチ命令の非整列による予測精度の低下

たとえば，4命令フェッチで，下位2ビットを除く命令フェッチ・アドレスをインデクスとする場合，フェッチする4命令が4ワード境界に整列していなければ，BTBで予測できない分岐が生じる可能性がある．**図2**を用いて説明する．この例は，アドレス2より4命令`i2~i5`をフェッチする場合で，`i4`を分岐命令とする．この場合，読み出されるBTBのエントリ0は`i0~i3`に対応するものであり，`i4`の分岐を予測できない．この問題は，単純にはエイリアス数を増加させれば緩和する．たとえば，下位3ビットを除いたアドレスをBTBのインデクスとすれば，BTBのエントリ0は，`i0~i7`に対応するものとなり，分岐`i4`を予測できる．しかしこの場合，上述2のエイリアスの問題が大きくなる．

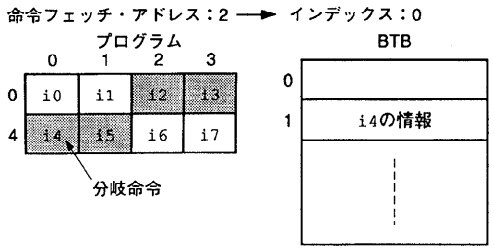

図 2 フェッチ命令非整列の問題
Fig. 2 Misalignment of fetching instructions.

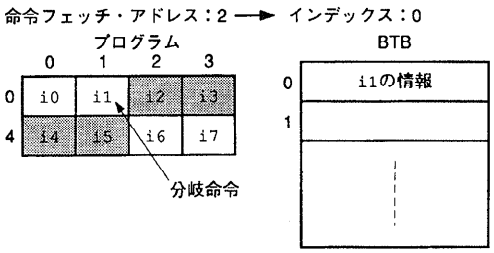

図 3 ヒット・チェック
Fig. 3 Hit check.

(4) ヒット・チェックの複雑化

BTB にヒットしたかを調べるために，タグ・チェック以外のチェックが必要である．**図 3** を用いて説明する．この例は，図 2 と同様であるが，命令 `i1` が分岐の場合である．参照する BTB のエントリ 0 に `i1` の情報が存在している場合，これを参照してはならない．

4. 複数予測方式

本章では，3 章で示した 1~4 の問題を解決する方式を提案する．以下，この方式を**複数予測方式**と呼ぶ．本方式の特長をあげる．

- BTB を複数のテーブルに分割し同時に参照することにより，複数の分岐を予測する．
- エイリアス数を大きくとりフェッチ命令非整列の問題を解決する．このとき，エイリアスの問題が悪化するが，BTB のテーブル数を増すことで対処する．
- BTB の各エントリに対応している分岐を識別するためのアドレス情報を格納し，エントリの正確なヒット・チェックを行う．

以下，具体例を用いて説明する．**図 4** に同時に最大 2 つの分岐の予測を可能とする本方式の実現例を示す．仮定しているスーパスカラ・マシンは，1 サイクルに最大 4 命令をフェッチするとする．PC は 30 ビットの命令フェッチ・アドレス（ワード・アドレス）を保持し，BTB は全部で 1024 のエントリを持つとする．

まず，BTB を 2 つのテーブルで構成する．以下，このテーブル数を**連想度**と呼ぶ．フェッチ命令非整列の問題を緩和するため，各エントリは同時フェッチ命令数 4 の倍である 8 命令に対応させる．すなわち，エイリアス数を 8 とする．これによりインデクスは $PC_{3...11}$ となる☆．各エントリは従来の BTB と同じく，タグ（$PC_{12...29}$ に対応），履歴（2 ビット・カウンタ値），分岐先アドレス（30 ビット）のほかに，そのエントリが対応している分岐命令のアドレスの下位 3 ビットを記憶する．この 3 ビットのことを以下**オフセット**と呼ぶこととする．

予測は次のようにして行う．

(1) PC を命令キャッシュに送出し読み出すのと並行して，$PC_{3...11}$ をインデクスとし BTB の 2 つのテーブルのそれぞれよりエントリを読み出す．

(2) 現在フェッチしている命令の中に分岐が存在し，かつ，読み出した各エントリがその分岐に対応するものかどうかをチェックする．これは，タグの一致比較と，オフセットと $PC_{0...2}$ の大小比較（$PC_{0...2} \leq$ オフセット $\leq PC_{0...2}+3$）によって行う．PC はフェッチする 4 命令の最初のアドレスであるから，後者のチェックが必要である．両方成立していれば，そのエントリは現在フェッチ中の命令の分岐に関する情報を保持している．これを予測に**有効**なエントリと呼ぶこととする．有効なエントリの履歴値が 2 以上の場合，分岐成立と予測する．そうでなければ不成立と予測する．無効なエントリの予測は不成立とする．

(3) 予測結果に従って次命令アドレスを決定する．2 つのエントリのうち，どちらのエントリも分岐不成立と予測した場合，現在の PC の静的に 4 だけ先のアドレス（以下，**シーケンシャル・アドレス**と呼ぶ）を次命令アドレスとする．どちらか一方のエントリが分岐成立を予測した場合，そのエントリの分岐先アドレスを次命令アドレスとする．両方のエントリが分岐成立を予測した場合，オフセットの小さい方のエントリの分岐先アドレスを次命令アドレスとする．

5. 評価結果

本章では，5.1 節で逐次予測方式と複数予測方式のハードウェアの複雑さの比較を行う．次に 5.2 節で性能の評価を行う．逐次予測方式，単一予測方式，複数予測方式を同一のスーパスカラ・マシンに適用したときの予測精度および実行サイクル数の比較を行う．

☆ $X_{m...n}$ は，X の第 m ビットから第 n ビットを選択したもの．

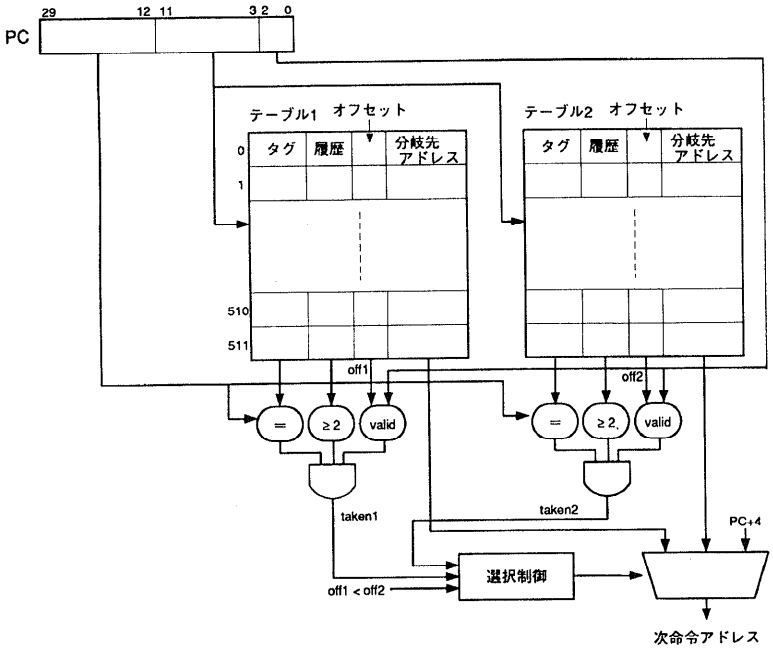

図 4 複数予測方式の構成例
Fig. 4 Organization example of the parallel prediction scheme.

5.1 ハードウェアの複雑さ

ハードウェアの複雑さを，PC 送出から次命令アドレス決定までに要する信号遅延時間を比較することによって評価した．複数予測方式は，全部で 1024 エントリの BTB を持ち，4 章で述べた構成をとるとする．比較対象の逐次予測方式は，同一エントリ数の BTB を連想度 2 で構成する．

遅延時間の見積り方法を説明する．各方式についてクリティカル・パスを見つけ，そのパスの論理を構成する．パスを構成する各ゲートの**等価ゲート段数**の和をそのパスの遅延時間とする．ここで，ある基本ゲートの等価ゲート段数とは，そのゲートのファンアウト 4 での遅延時間を同じファンアウトのインバータの遅延時間で割った値である．これはあらかじめ回路シミュレータ SPICE を用いて求めておく．たとえば，2 入力 NAND は 1.2 ゲート，4 入力 NAND は 2.0 ゲートである．また，基本ゲート以外の汎用的な部品も 1 つのゲートとして扱う．本議論では，BTB のテーブルに用いる 6.6 K バイトの RAM の読み出し時間を 13.0 ゲート [9]，タグ比較に用いる 18~21 ビットの比較器の遅延時間を 6.0 ゲートとした（当社ライブラリ）．等価ゲート段数は半導体技術に依存せずほぼ一定であることが知られており [10]，遅延時間を比較するには都合がよい．

表 2 に PC 送出から次命令アドレス決定に要する遅延時間の見積り結果を示す．次命令アドレス決定は，BTB 読み出しと，アドレス選択（「タグ比較」から「選択」まで）からなる．

表 2 次命令アドレス決定の遅延時間
Table 2 Delay time for determination of the next address.

パス	逐次予測	複数予測
BTB 読み出し	13.0	13.0
アドレス選択	15.0	16.5
タグ比較	6.0	6.0
選択制御	4.0	5.5
選択信号駆動	3.0	3.0
選択	2.0	2.0
合計	28.0	29.5

評価した逐次予測方式の構成は，図 1 において BTB を 2 つのテーブルで構成したものである．1 エントリには，21 ビットのタグ，2 ビットの履歴，30 ビットの分岐先アドレスを記憶するため，合計 53 ビット必要である．したがって，1 K エントリの BTB には約 6.6 K バイト（53/8 バイト × 1 K エントリ）の RAM が必要である．この読み出し時間は 13.0 ゲートである．アドレス選択では，最も優先度の低いシーケンシャル・アドレスを選択する論理がクリティカルであり，評価の結果，遅延時間は 15.0 ゲートであった．これにより，次命令アドレス決定に要する時間は 28.0 ゲートとなる．

複数予測方式では，アドレス選択が逐次予測方式に

表 3 複数予測方式におけるアドレス選択論理
Table 3 Address selection logic in the parallel prediction scheme.

taken1	taken2	off1 < off2	次命令アドレス
1	1	1	テーブル 1 からの分岐先
1	1	0	テーブル 2 からの分岐先
1	0	×	テーブル 1 からの分岐先
0	1	×	テーブル 2 からの分岐先
0	0	×	シーケンシャル

×：don't care

比べて複雑になる．まず，taken 信号（図 4 において，taken1 または taken2）は次の式で表される．

taken = 履歴 $\geq$ 2 &タグ一致
& ($PC_{0...2} \leq$ off $\leq PC_{0...2} + 3$)

ここで，off は BTB より読み出されたオフセット（図 4 において，off1 または off2）である．逐次予測方式の場合に比べて，BTB のヒット・チェックのために最後の項を追加する必要があり，複雑さが増している．この項を計算するために 3 ビットの加算器を含む論理が必要である．評価の結果，遅延時間は 5.7 ゲートであり，タグ比較時間（6.0 ゲート）より短いことが分かった．したがって，BTB ヒット・チェックの複雑化によって遅延時間が増加することはない．

表 3 にアドレス選択論理の真理値表を示す．同表より，

s1 = taken1 & ((off1 < off2) | !taken2)
s2 = taken2 & (!(off1 < off2) | !taken1)
s3 = !taken1 & !taken2

となる．ここで，s1 はテーブル 1 からの分岐先アドレス，s2 はテーブル 2 からの分岐先アドレス，s3 はシーケンシャル・アドレスを選択する論理である．明らかに，s1 または s2 がクリティカルである．評価の結果，タグ比較後 s1（または s2）が決定するまでの遅延時間は 5.5 ゲートであった．

以上より，アドレス選択に要する時間は 16.5 ゲートとなる．

BTB の 1 エントリは，18 ビットのタグ，2 ビットの履歴，30 ビットの分岐先アドレス，3 ビットのオフセットの合計 53 ビットで構成されるので，BTB は逐次予測方式と同一サイズである．したがって，BTB の読み出し時間は 13.0 ゲートである．

以上より次命令アドレス決定に要する時間は 29.5 ゲートとなる．この値は，逐次予測方式に比べて 1.5 ゲート長いだけであり，本方式がプロセッサのサイクル時間を悪化させる可能性はほとんどない．

5.2 性能評価

評価は，トレース駆動シミュレーションにより行った．

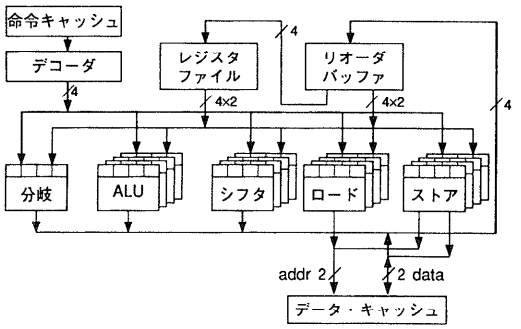

図 5 スーパスカラ・マシンのハードウェア構成
Fig. 5 Hardware organization of the superscalar machine.

性能は，MIPS R3000[11] を基本マシンとし，R3000 に対する性能比（R3000 での実行サイクル数を評価したマシンでの実行サイクル数で割った値）を用いた．以下，評価に共通のスーパスカラ・マシンのハードウェア構成，評価したモデル，評価結果について述べる．

5.2.1 スーパスカラ・マシンのハードウェア構成

図 5 にスーパスカラマシンのハードウェア構成を示す．1 サイクルに最大 4 命令をフェッチし，out-of-order で実行を行う．分岐ユニットを 1 つ，ALU，シフタ，ロード，ストアの各ユニットをそれぞれ 4 つ持つ．各機能ユニットは，4 つのエントリを持つ**リザベーション・ステーション**[12] を備える．命令のレイテンシは，ロード命令が 2 サイクルで，その他命令が 1 サイクルである．分岐予測ミス・ペナルティは 3 サイクルとした．

命令フェッチでは，最初に，命令キャッシュより命令フェッチ・アドレスに対応するライン全体を読み出し，その中よりフェッチすべき命令を選択する．フェッチしようとする 4 命令がライン境界をまたぐ場合，次のラインにある命令は読み出せない[5]．命令キャッシュは，容量 16 K バイト，ライン・サイズ 8 ワード，ラインの置き換えに 12 サイクル必要とした．命令デコーダは，フェッチされた命令から最大 4 命令をデコードし，リザベーション・ステーションへ送る．実行結果は，いったん**リオーダ・バッファ**[13] に格納され，in-order でレジスタ・ファイルへ書き込まれる．リオーダ・バッファのエントリ数は 16 とした．

5.2.2 評価モデル

評価したどのモデルも，BTB の全エントリ数を 1024 とした．このエントリ数では容量不足による衝突はほとんどなく十分な容量である．容量の違いが各予測機構に対して与える影響については，5.2.6 項で評価する．

BTB エントリの入れ替えのアルゴリズムとしては

LRU（Least Recently Used）法を用いた．すなわち，最も以前に使用したエントリを置き換える．複数予測方式の評価では，BTBの連想度をパラメータとした．逐次予測方式と単一予測方式では，連想度を2以上にしても性能差はほとんどないので，連想度2の場合の評価結果のみを示す☆．以下，各予測方式における評価モデルについて説明する．

逐次予測方式：S モデルと呼ぶ．命令は命令キャッシュに格納される際にプリデコードされており，命令フェッチ直後にデコードを行わずBTBの参照を開始できるとする．したがって，分岐成立と予測した場合に生じるペナルティは1サイクルである．1サイクルに1つの分岐しか予測できないので，フェッチした命令の中に複数の分岐があった場合は，パイプラインを停止させ順次予測を行う[7]．

単一予測方式：エイリアス数4の**U4 モデル**と，8の**U8 モデル**の2つのモデルを評価した．これらのモデルは，3章で説明した単純な単一予測方式を多少強化したもので，BTBのエントリに登録されている分岐命令のアドレス・オフセットが記録されているとした．これにより，正確なヒット・チェックが可能となる．

複数予測方式：エイリアス数4の**M4 モデル**と，8の**M8 モデル**の2つのモデルを評価した．複数予測方式では，連想度による性能差が大きいと予想されるので，M4に対して連想度2，4の2モデル，M8に対して連想度2，4，8の3モデルを評価した．モデル名の後に“-連想度”を付けこれらを区別する．たとえば，M4-2モデルは，エイリアス数4，連想度2の複数予測方式である．

5.2.3 分岐予測精度

図6に，Sモデル，U4モデル，U8モデル，M4-2モデル，M8-2モデルの分岐予測ミス率の測定結果を示す．a-meanは全ベンチマークでの算術平均である．また，**図7**に，分岐予測ミスにおける原因の分布を示す．同図で，「ベース」とは，基本的な方式（この場合，2ビット・カウンタ方式）とBTBのサイズに起因する予測ミスであり，逐次予測においても生じる予測ミスである．「エイリアス」は，3章で述べたエイリアス（複数分岐予測不能を含む）により生じた予測ミスを表す．「非整列」は，フェッチ命令非整列により生じた予測ミスを表す．

Sモデルでは，予測ミスの生じる原因は，基本的な方式とBTBのサイズによるものだけであるので，他のどのモデルより予測ミス率は小さく，平均ミス率は20.3%であった．

U4モデルでは，エイリアスにより平均で2.8%，フェッチ命令非整列により7.6%予測ミス率が増加した．この結果，予測ミス率は30.7%にもなった．

U8モデルでは，フェッチ命令非整列による予測ミスはなくなるが☆☆，エイリアスによる予測ミスが7.4%に増大した．その結果，U4モデルより改善したものの依然として27.7%という大きな予測ミス率となっている．

M4-2モデルでは，逆に，エイリアスによる予測ミスはなくなったが，フェッチ命令非整列の問題が解決されずに残っている．このため，予測ミス率は28.5%と改善は少ない．

M8-2モデルは，M4-2モデルに比べればエイリア

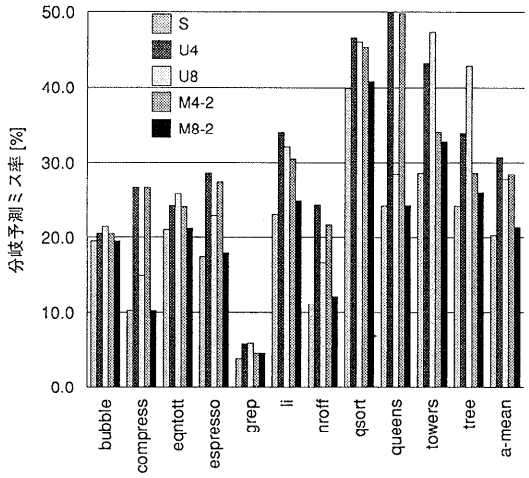

図6 分岐予測ミス率
Fig. 6 Misprediction rate.

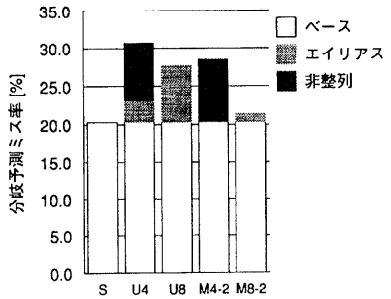

図7 分岐予測ミスの原因分布
Fig. 7 Cause distribution of misprediction.

☆ 逐次予測方式において連想度2に対する4による予測率の改善はわずか0.2%である．

☆☆ 8ワード境界を超えたアドレスにある分岐命令については，キャッシュ・ライン境界を超える命令のフェッチができないので，分岐予測の必要性が生じない．

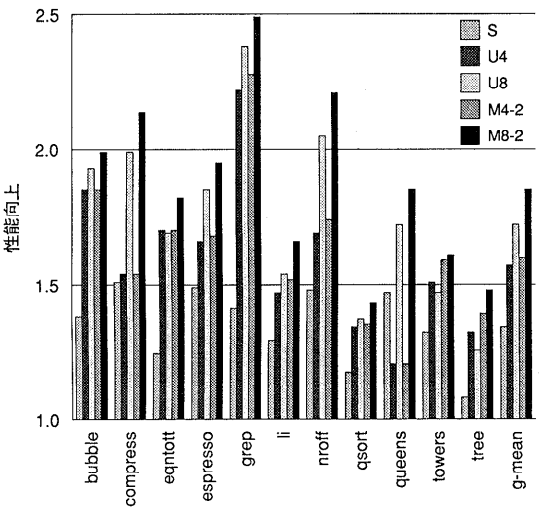

図 8 性能向上率
Fig. 8 Speedup.

スの問題が悪化するものの，フェッチ命令非整列の問題が解消される．評価の結果，エイリアスによる予測ミスはわずかに 1.0%であり，エイリアスの問題はほとんど悪化しないことが分かった．この結果，予測ミス率は 21.3%にまで改善することができた．

5.2.4 性　　能

スカラ・マシン（MIPS R3000）に対する実行サイクル数における性能向上率を**図 8** に示す．g-mean は全ベンチマークでの幾何平均である．

S モデルは，他の方式と比べて予測精度が高いにもかかわらず，最も性能が低い．最も予測精度の低い U8 モデルより 22.1%も性能が低い．このことより，分岐予測が正しい場合にも分岐成立時に生じる 1 サイクルのペナルティは，性能に非常に大きな影響を与えることが分かる．

U4，U8，M4-2，M8-2 モデルは正しく分岐を予測できた場合のペナルティはないので，これらの間の性能差は予測精度の違いだけによるものである．S モデルに比べて，U4 モデルは 17.2%，U8 モデルは 28.4%，M4-2 モデルは 19.4%，M8-2 モデルは 38.1%性能を改善している．M8-2 モデルでは，S モデルとは予測精度はほとんど変わらないので，この大きな性能改善は予測アドレスを早く送出できることによるものである．M8-2 モデルは，命令フェッチに要求される，(1) 分岐を正しく予測する，(2) 予測アドレスを早く送出する，という 2 つを同時に満足することができ，その結果，大きな性能向上を達成できた．

5.2.5 連想度による比較

複数予測方式では，エイリアスによる予測ミスを連想度を上げることにより低下させることができる．しかし，連想度を上げるとハードウェアが複雑化するので，トレードオフを考慮しなければならない．**表 4** に，M4，M8 モデルにおいて，連想度を変えたときの予測ミス率を示す．比較のため，連想度 1（M4 モデルに対しては U4 モデル，M8 モデルに対しては U8 モデルに相当）における予測ミス率も示す．

表 4 連想度対分岐予測ミス率（%）
Table 4 Associativity vs. misprediction rate (%).

モデル	連想度			
	1	2	4	8
M4	30.7	28.5	28.5	n/a
M8	27.7	21.3	20.3	20.3

M4 モデルでは，連想度を 1 から 2 に上げれば，予測ミスは 2.2%減少する．M4 では，エイリアスの生じる確率が大きくないので，連想度を上げることによる改善度は大きくない．連想度を 2 から 4 にしてもこれ以上の改善はなく，2 で十分である．ただし，M4 ではフェッチ命令非整列による予測ミスは解決できないので，S モデルでの予測精度よりかなり（8.2%）悪い．

M8 モデルでは，エイリアスの生じる確率が大きいので，連想度を上げることの効果は大きい．連想度を 1 から 2 にすれば，最大 14.6%（towers），平均 6.4%と大きく予測ミスを減少させることができた．連想度を 4 にすれば，予測ミスはわずかであるがさらに減少する（1.0%）．しかし，複数の分岐の予測が分岐成立であるとき，最もオフセットの小さいエントリの分岐先アドレスを選択する必要があり，ハードウェアが複雑化する．これがクロック速度に悪い影響を与える可能性があるので，連想度 4 にすることは得策ではない．また，連想度 8 にしても予測精度のさらなる改善はない．

以上より，エイリアス数 8，連想度 2 にするのが最適であることが分かった．

5.2.6 BTB の容量に対する影響

複数予測方式と逐次予測方式とでは，BTB のエントリへの分岐のマッピングが異なる．たとえば，総エントリ数 1024，連想度 2 の従来の BTB の第 n エントリには，512（1K エントリ/2 ウェイ）ワード境界から n だけ大きなアドレスにある分岐がマッピングされる．これに対して，同一エントリ総数の M8-2 の複数予測方式の BTB の第 n エントリは，4K（1K エントリ/2 ウェイ × 8 エイリアス）ワード境界から $8n$ だけ大きなアドレスから，これを含めてそれより連続する 8 つの分岐がマッピングされる．マッピングが異なれば，エントリで衝突する分岐が変わるので，予測

表5 BTBの総エントリ数に対する予測ミス率(%)
Table 5 Misprediction rate with the different number of BTB entries (%).

モデル	エントリ数		
	16	128	1024
S-4	27.8	21.2	20.3
M8-4	28.3	21.0	20.3

率が影響を受ける可能性がある．容量が小さいほど衝突が頻繁に起こるので，可能性は高くなる．また，エイリアス数，連想度を大きくするほど，従来のBTBのマッピングと異なるものとなり，影響を受ける可能性は高くなる．

表5に，BTBの総エントリ数を変えた場合の連想度4の逐次予測方式（S-4）と同一連想度でエイリアス数8の複数予測方式（M8-4）の分岐予測ミス率の測定結果を示す．エントリ数16では，衝突が頻繁に起こるため，予測ミス率が28%程度にまで上がっている．しかし，複数予測方式による予測率は逐次予測方式よりわずかに0.5%悪いだけである．エントリ数128では，逆に0.2%複数予測方式の方が良く，エントリ数1024ではまったく変わらない．以上より，マッピングの違いによる影響はほとんどないということができる．

6. 関連研究

Digital Alpha 21064, AMD K5, Sun UltraSPARCでは，命令キャッシュに分岐予測情報を格納している[14)~16)]（たとえば，K5では4命令に1つの分岐先アドレスを持つ）．この方法では，命令フェッチの直後に次命令アドレスを送出することが可能である．しかし，分岐を含まない命令の組に対しても，分岐予測情報を格納する領域を命令キャッシュに設けなければならいので，ハードウェアの使用効率が悪い．

文献4)には，命令フェッチと並行してBTBを参照し分岐予測を行う方式が提案されている．しかし，命令フェッチは基本ブロックを単位に行うことを仮定しており，固定の数の命令をフェッチする実際のスーパスカラ・マシンへ適用できない．

文献3)には，BTBをバンクに分割し並列に分岐を予測する方式が提案されている．たとえば，4命令発行の場合，BTBを4つのバンクに分割し，各バンクはそれぞれa, a+1, a+2, a+3（aは命令フェッチ・アドレス）をインデクスとして参照する．ある分岐命令がフェッチする命令グループの何番目にあるかは動的に変化する．そのため，この方法では，参照したバンクに目的の分岐予測情報があるとは限らず，ヒット率が低下する．

7. まとめ

本論文では，スーパスカラ・マシンにおいて，単純なハードウェアで高い命令フェッチ速度を実現する機構を提案した．本方式は，命令フェッチ・アドレスごとに分岐情報を記録するBTBを用いることにより，命令フェッチと同時に分岐の予測を行い，早期に次命令アドレスを送出することができる．命令フェッチ・アドレスについて分岐情報を記録する方式では，エイリアス，フェッチ命令非整列による予測精度の下が生じるが，本方式はこれらを単純なハードウェアで解決した．評価の結果，本方式は，分岐ごとに予測情報を参照する方式に比べ，1.0%の分岐予測ミスの増加しか見られず，早期に分岐予測を行うことによるペナルティは少ない．4命令フェッチのスーパスカラ・マシンにおいて，デコード・ステージで分岐予測を行う従来の場合に比べて，約38%性能を改善できることを確認した．

謝辞 本研究に対してご支援いただいた三菱電機（株）システムLSI事業化推進センター・角正氏に感謝いたします．

参考文献

1) Lee, J.K.F. and Smith, A.J.: Branch Prediction Strategies and Branch Target Buffer Design, *Computer*, Vol.17, No.1, pp.6–22 (1984).
2) Yeh, T-Y. and Patt, Y.N.: Alternative Implementation of Two-Level Adaptive Branch Prediction, *Proc. 19th Int. Symp. on Computer Architecture*, pp.124–134 (1992).
3) Conte, T.M., Menezes, K.N., Mills, P.M. and Patcl, B.A.: Optimization of Instruction Fetch Mechanism for High Issue Rates, *Proc. 22nd Int. Symp. on Computer Architecture*, pp.333–344 (1995).
4) Yeh, T-Y. and Patt, Y.N.: A Comprehensive Instruction Fetch Mechanism for a Processor Supporting Speculative Execution, *Proc. MICRO-19*, pp.129–139 (1992).
5) Hennessy, J.L. and Patterson, D.A.: *Computer Architecture: A Quantitative Approach*, Second Edition, Morgan Kaufmann, San Mateo, CA (1996).
6) Case, B.: Intel Reveals Pentium Implementation Details, *Microprocessor Report*, Vol.7, No.4, pp.9–17 (1993).
7) Gwennap, L.: MIPS R10000 Uses Decoupled Architecture, *Microprocessor Report*, Vol.8, No.14, pp.18–22 (1994).

8) Diep, T.A., Nelson, C. and Shen, J.P.: Performance Evaluation of the PowerPC 620 Microarchitecture, *Proc. 22nd Int. Symp. on Computer Architecture*, pp.163–174 (1995).
9) Wada, T., Rajan, S. and Przybylski, S.A.: An Analytical Access Time Model for On-Chip Cache Memories, *IEEE Journal of Solid-State Circuits*, Vol.27, No.8, pp.1147–1156 (1992).
10) Mead, C. and Conway, L.: *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA (1980).
11) Kanc, G.: *MIPS RISC Architecture*, Prentice Hall, Englewood Cliffs, New Jersey (1988).
12) Tomasulo, R.M.: An Efficient Algorithm for Exploiting Multiple Arithmetic Units, *IBM Journal of Research and Development*, Vol.11, No.1, pp.25–33 (1967).
13) Smith, J.E. and Pleszkun, A.R.: Implementation of Precise Interrupts in Pipelined Processors, *Proc. 12th Int. Symp. on Computer Architecture*, pp.36–44 (1985).
14) 浅見直樹，枝　洋樹：次世代マイクロプロセサ，スーパスカラと VLIW が融合，日経エレクトロニクス，No.626, pp.67–150 (1995).
15) Gwennap, L.: New Algorithm Improves Branch Prediction, *Microprocessor Report*, Vol.9, No.4, pp.17–21 (1995).
16) Slater, M.: AMD's K5 Designed to Outrun Pentium, *Microprocessor Report*, Vol.8, No.14, pp.1–11 (1994).

(平成 9 年 10 月 29 日受付)
(平成 10 年 4 月 3 日採録)

中西知嘉子

1988 年大阪大学基礎工学部情報工学科卒業．1988 年三菱電機（株）LSI 研究所に入社．第 5 世代コンピュータプロジェクトの推論マシン用プロセッサの設計に従事．計算機アーキテクチャ，コンパイラの研究に従事．1997 年より，信号処理プロセッサの開発に従事．現在，同社システム LSI 事業化推進センターに所属．

安藤　秀樹（正会員）

1959 年生．1981 年大阪大学工学部電子工学科卒業．1983 年同大学大学院修士課程修了．工学博士（京都大学）．1983 年三菱電機（株）LSI 研究所．1991 年 Stanford 大学客員研究員．1997 年名古屋大学大学院工学研究科電子情報学専攻・講師．計算機アーキテクチャコンパイラの研究に従事．

原　　哲也（正会員）

1966 年生．1989 年九州大学工学部情報工学科卒業．1991 年同大学大学院総合理工学研究科情報システム学専攻修士課程修了．同年三菱電機（株）LSI 研究所に入社し，細粒度並列処理アーキテクチャの研究に従事．1996 年より信号処理プロセッサの開発に従事．現在，同社システム LSI 事業統括部に所属．

中屋　雅夫（正会員）

1974 年早大学理工学部卒業．1976 年同大学大学院修士課程修了．1988 年工学博士（早稲田大学）．1976 年三菱電機（株）入社．以来，高速ゲートアレイ，MOS A/D，D/A コンバータ，三次元回路素子，通信用 LSI，並列処理プロセッサの研究開発を経て，現在，通信・ネットワーク用 LSI の開発，事業化に従事．現在，三菱電機（株）システム LSI 事業化推進センター所属．92 年度電子情報通信学会論文賞受賞．93 年度近畿地方発明表彰受賞．電子情報通信学会，IEEE 各会員．