大規模データパス・アーキテクチャにおけるリターンアドレス予測機構の検討

塚本泰通[†]　坂井修一[†]　田中英彦[†]

大規模データパス・アーキテクチャは最大 4 つの分岐命令を含む命令ブロックを処理単位とし、複数パス実行を行う。命令ブロックには出口が最大 5 つあるため、出る確率の高い出口を 2 つまで予測し、その予測成功率は約 90%である。本論文では、リターンアドレス予測が分岐予測機構のボトルネックであることを示し、リターンアドレススタックを用いることでリターンアドレス予測成功率が最大 35.6%向上することを示す。

Study of Return-Address Prediction Mechanism on a Very Large Data Path Architecture

Yasumichi Tsukamoto,[†] Shuichi Sakai[†] and Hidehiko Tanaka[†]

A Very Large Data Path Architecture(VLDP) supports multipath execution of instructions in a granularity called Instruction Block. An Instruction Block consists of four contiguous basic blocks, each of which can have up to 8 instructions. An instruction block has up to 5 break points. VLDP predicts two break points from which it will exit, and can achieve more than 90% branch prediction accuracy. In this paper, we show that return-address prediction is a bottleneck and by using a return-address stack, we can improve the return-address prediction accuracy by up to 35.6% compared to the case with no return-address stack.

1. はじめに

現在マイクロプロセッサは、アーキテクチャの改善と半導体プロセス技術に支えられ著しい性能向上を果たしている。しかし将来利用可能なトランジスタ数は増加が見込まれる一方、現在のアーキテクチャの主流であるスーパースカラ・アーキテクチャは、その設計の複雑さから投入したハードウェア量に見当った性能が引出せないという問題がある。そこで、大規模なハードウェアを有効に利用する大規模データパス (Very Large Data Path:VLDP)・アーキテクチャが提案されている [1)]。

VLDP では、複数パス実行による大規模な投機実行、最大 32 命令からなる命令ブロックを処理単位とする高スループット、複数の実行ユニットによる命令ブロックの並列実行、レジスタを介さないデータアクセスの明示的記述などを特徴とし、実効 IPC 8 を目指す。IPC 8 を実現するには、2 桁のフェッチ能力が必要となるため、高い精度の分岐予測と無駄のないパス展開を行う必要がある。現在 VLDP では、一番実行確率の高いパスの予測で約 82%、二番目に実行確率の高いパスの予測まで含めると約 90%の精度の分岐予測機構が提案されている [2)]。本論文では、分岐予測の中でもリターンアドレス予測の重要性を示し、その予測機構の検討を行う。

† 東京大学大学院 工学系研究科
Graduate school of Engineering, The University of Tokyo

2. VLDP のフェッチ機構

2.1 命令ブロックの導入

VLDP が実効 IPC 8 を実現するためには、毎サイクルに 2 桁命令のフェッチスループットが必要となる。そこで、VLDP では最大 4 つの分岐命令を含む命令ブロック (IB:Instruction Block) を導入 [1)3)] し、処理単位とする。命令ブロックの構成を図 1 に示す。IB は最大 8 命令からなる field を 4 つ持ち、分岐命令は各 field の最後にのみ置かれる。分岐の区切りにより命令が埋められない場合には NOP を挿入する (図 1 中の Blank Slot)。通常各 field は 1 つの基本ブロックであるが、分岐命令が 8 命令以上の間隔で出現した場合には、その基本ブロックを複数の field に分割する。VLDP では 1 サイクルに 1IB をフェッチすることで、毎サイクル最大 32 命令のフェッチスループットを確保する。また、IB の実行を行う実行ユニット (Execution Unit:EU) を複数用意し複数の IB を並列に実行することで、命令レベルの並列性に加え IB レベルの並列性の抽出も目指す。

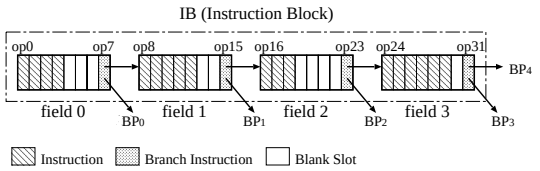

図 1 Instruction Block Structure

2.2 フェッチ機構の概要

2.2.1 フェッチ戦略

複数パス実行をする場合、すべての分岐命令の飛先を実行すると分岐予測ミスはなくなるが、実行パス数が爆発的に増加し、資源不足や管理の複雑さにつながる。そこで、実行確率の高いパスのみを選択し、そのパス上の IB のみをフェッチする必要がある。このパス選択手法をフェッチ戦略と呼び、VLDP では以下に挙げる 4 つを基本方針とする。ここで、複数のパスのうち最も実行確率の高いパスをメインパスと呼び、その他のパスをサブパスと呼ぶ。また、ある IB から見た際に、一番実行確率の高い子 IB を第一候補、二番目に実行確率の高い子 IB を第二候補と呼ぶ。

- フェッチはメインパス、サブパスから交互に行う。ただし、サブパス上の候補がない場合はメインパス上の IB を続けてフェッチする。
- サブパス上の IB の第二候補はフェッチ対象としない。
- フェッチされた IB の最大 2 つの子 IB(第一候補、第二候補) をフェッチ候補とする。
- サブパス上の IB は、フェッチ候補に挙がった順にフェッチされる。

このフェッチ戦略では、メインパス候補は常に一つで、全フェッチ候補数の爆発的増加がないため、管理が行いやすいメリットがある。

2.2.2 分岐予測機構

フェッチ戦略通りのフェッチを行うためには、各 IB の出口とその出口から始まる IB のアドレスを最大 2 つ予測する必要がある。これを分岐予測機構が行う。構造は図 2 のようになっており、それぞれ 5 つの出口に対応した IB のアドレスをインデックスとする Prediction Table(PT_0〜PT_4) を持つ。各 PT は図 3 のようになっており、テーブルは 3 種類存在する。1 つ目はメインパス候補の出口を予測するためのカウンタ (Main Path Counter:MC_0〜MC_4)、2 つ目はサブパス候補の出口を予測するためのカウンタ (Sub Path Counter:SC_0〜SC_4)、3 つ目はそれぞれの出口に対応する IB のアドレスを予測するためのテーブル (IB Target Buffer:ITB_0〜ITB_4) である。MC は飽和カウンタ、SC はリセッティングカウンタの動作をする。

PT の更新

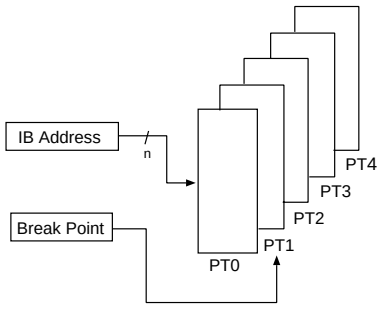

図 2 Prediction Unit の構造

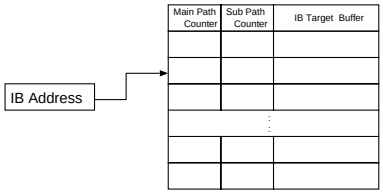

図 3 予測テーブルの構造

コミットした IB が BP_N から抜けた場合、PT_N にアクセスする。IB のアドレスをインデックスとして MC,SC,ITB それぞれを更新する。具体的には MC_N をインクリメント、SC_N を最大値にし、それ以外の SC と MC をデクリメントする。また,IB の飛先アドレスを ITB_N に格納する。

予測

フェッチした IB のアドレスをインデックスとして PT_0〜PT_4 にアクセスする。各 PT の MC のうち一番大きいものが MC_n の場合、BP_n をメインパス候補、SC_n 以外の SC_0〜SC_4 のうち一番大きいものが SC_m の場合、BP_m をサブパス候補とする。ただし、サブパス候補に関して SC_n 以外が全て 0 の場合はサブパス候補はなしとする。また各予測 BP に対応する IB のアドレスは ITB_n から得られる。

2.2.3 性　　能

文献 [2] では、2.2.2 節で述べた分岐予測機構を備え、2.2.1 節で述べたフェッチ戦略に沿って IB をフェッチ、実行するシミュレーターにおいて、分岐予測精度の評価を行っている。IB は Alpha の命令セットを元に作成している。またフェッチに注目するため、IB の実行は固定 16 サイクルかかり、命令キャッシュは 100%ヒット、PT のサイズは十分であるという条件の元、SPECint95 のベンチマーク (gcc 以外) に対して分岐予測成功率を評価している。ここで、分岐予測成功とは、ある IB がリタイアした際に分岐先の IB が既にフェッチされていることを指す。よって、分岐予測成功が次のサイクルに IB をリタイアできることにはつながらない。

提案された分岐予測機構での分岐予測成功率は約

表 1 分岐命令種別予測成功率とその割合

	P_c	P_i	P_a
予測成功率	97.5%	74.7%	90.7%
全体に占める割合	76.6%	23.4%	100%

90%である。また、単一パス実行における評価もしており、この場合の分岐予測成功率は約 83%となっている。関連研究として、文献 [4] でトレースキャッシュ用に 1 サイクルに複数の分岐予測を行う手法が提案されており、1 サイクルに 4 つの分岐予測を行う場合で約 82%の予測成功率である (単一パス実行)。この手法は木構造で分岐結果を管理するため条件分岐予測には効果的だが、レジスタ間接分岐予測を考慮していないためその点が問題である。

3. リターンアドレス予測機構の必要性

IB の実行は基本的に条件分岐命令で終わる場合とレジスタ間接分岐命令で終わる場合とそれ以外の命令で終わる場合がある。2 章の評価は、これら全てを合わせた結果である。本章ではそれぞれの場合ごとの予測成功率を見る。条件分岐命令や分岐命令以外 (条件分岐命令数に含む) で IB を抜ける場合の予測成功率を P_c、レジスタ間接分岐命令やシステムコール (レジスタ間接分岐命令数に含む) で IB を抜ける場合の予測成功率を P_i、全てを合わせた予測成功率を P_a とし、下のように定義する。

$$P_c = \frac{\text{条件分岐予測成功数}}{\text{条件分岐命令で } IB \text{ を抜けた数}}$$

$$P_i = \frac{\text{レジスタ間接分岐予測成功数}}{\text{レジスタ間接分岐命令で } IB \text{ を抜けた数}}$$

$$P_a = \frac{\text{全分岐予測成功数}}{IB \text{ 実行数}}$$

これらそれぞれの予測成功率と全体に占める割合を表 1 に示す。P_c は 97.5%と高い成功率である一方、P_i は約 74.7%と低い。また、レジスタ間接分岐命令で IB を抜ける場合は全体の約 25%を占めることから、さらに分岐予測精度を上げるためには、条件分岐予測精度の向上を目指すより、レジスタ間接分岐予測精度の向上を目指す方が賢明である。

Alpha のレジスタ間接分岐命令には JMP(Jmp)、JSR(Jump to Subroutine)、RET(Return from Subroutine) が存在する。P_i をさらに細かく分類し、各種類別予測成功率とその占める割合を表 2 に示す。ここで、RET で IB を抜ける場合のリターン値予測成功率を下のように定義する。

$$P_r = \frac{\text{リターン値予測成功数}}{\text{リターン命令で } IB \text{ を抜けた数}}$$

表 2 レジスタ間接分岐命令種別予測成功率とその割合

レジスタ間接分岐種類	JMP	JSR	RET
分岐予測成功率	59.5%	85.7%	66.1%
レジスタ間接分岐内の割合	10.3%	0.9%	88.8%

表 3 理想的なリターン予測を行った時の予測成功率

	P_a	P_i	P_r
予測成功率	96.6%	93.7%	96.3%

RET で IB を抜ける場合はレジスタ間接分岐命令で IB を抜けるうちの約 89%を占め、その予測成功率である P_r は約 66%と低い予測成功率となっている。これより、VLDP における分岐予測において、リターン値予測がボトルネックになっていることがわかる。

一方、通常の単一パス実行プロセッサでは、修復機構付リターンアドレススタックを用いることにより、ほぼ 100%に近い精度でリターン値予測が可能である。また文献 [5] では複数パス実行モデルにおけるリターンアドレススタックについて検討を行っている。1 つのリターンアドレススタックを複数のパスで共用する場合、サブルーチンコールとリターンの関係を複数のパスで破壊しあう可能性が高く性能が出ない。リターン値予測成功率は、BTB を用いた場合 57%、1 つのリターンアドレススタックを用いる場合 67%の精度となっている。しかし、パス同士の干渉を避けることを考え、1 つのリターンアドレススタックをメインパスのみで使用する場合 80%、パスごとにリターンアドレススタックを設けることで 100%に近い精度でリターン値予測が可能となる。このように、複数パス実行においても、リターンアドレススタックを用いることで、リターン値予測の改善を図ることは可能である。よって、次章以降で、VLDP におけるリターンアドレススタックを用いたリターンアドレス予測機構の検討を行う。

4. リターンアドレス予測機構の検討

4.1 理想的なリターン予測の評価

リターンアドレス予測機構の性能限界を知るため、実行トレースを元に理想的なリターン予測を行った。理想的な場合の P_a、P_i、P_r を表 3 に示す。全体の予測成功率 P_a は 96.6%と非常に高い予測成功率となり、リターンアドレス予測機構がうまく機能した場合に高い性能向上が期待できる。ここでリターン予測成功率 P_r が 100%ではないが、これは正しいリターン予測を行い正しいパス上の IB をフェッチ候補に挙げたが、フェッチ戦略によりその IB がフェッチされない場合があるためである。

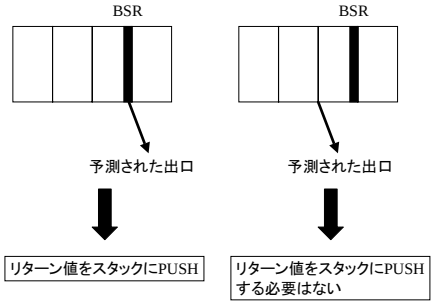

図 4 PUSH のタイミング

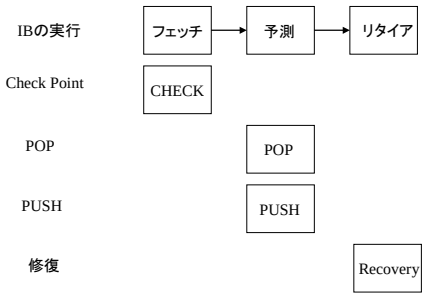

図 5 チェック、修復のタイミング

4.2 VLDP におけるリターンアドレススタック

VLDP におけるリターンアドレス予測機構として、通常のプロセッサと同様にリターンアドレススタックを導入することを考える。リターンアドレススタックは、サブルーチンコール命令 (Alpha では JSR と BSR) をフェッチした時にその命令の PC+1 の値をスタックに PUSH し、リターン命令 (Alpha では RET) をフェッチした時にスタックから値を POP し次にフェッチする命令の PC として用いる。サブルーチンコール命令とリターン命令は必ず 1 対 1 対応しているため、LIFO というスタックの性質がリターンアドレス予測に適している。しかし、投機実行、複数パス実行を行う VLDP では、以下のような問題がある。

複数パス実行
: 複数パス実行を行う場合、コントロールフローが複数あることになり、単純に考えるとスタックをコントロールフローの数分用意する必要があるためハードウェアコストがかかる上、その制御も複雑になる。

PUSH と POP のタイミング
: 通常のプロセッサでは、サブルーチンコール命令をフェッチした時に PUSH、リターン命令をフェッチした時に POP する。しかし、VLDP では IB が分岐命令を最大 4 つ含むため、フェッチした IB がサブルーチンコール命令を含む場合に、図 4 のように PUSH する場合としない場合が生じる。POP も PUSH と同様、予測出口とリターン命令の位置関係で POP する場合としない場合が生じる。また、分岐予測は 100%の精度ではないため、予測が誤っている場合はリターンアドレススタックの内容を破壊することにつながる。

投機実行 (修復機構の必要性)
: 投機実行中にスタック操作をした後その投機実行が失敗に終わる場合、スタックが破壊される。そのため、修復機構を設ける。通常のプロセッサでは、分岐命令をフェッチした時点をチェックポイントとしてスタックの内容を記憶しておき、分岐予測ミスをした際に記憶しておいたスタックの内容を読み出すことで修復を行う。この場合、チェックポイントは分岐命令をフェッチした時のみであるが、VLDP では IB のフェッチのため毎サイクル予測を行っている。そのため、VLDP では IB のフェッチをするたびにスタックの退避を行う必要がある。よって、保存するデータ量が膨大になる可能性が高い。

プロセッサ中の命令数
: VLDP ではフェッチされたがリタイアしていない命令数がスーパースカラに比べ数倍になる。これにより、分岐が確定するまでの間にスタックが操作される可能性が高くなり、スタックが破壊される可能性も高くなる。

これらの問題がある中で、複数のスタックを用意することはコストが高いと考え、一つのスタックのみを用いてリターンアドレス予測を行うことを考える。具体的には、すべてのパスで一つのスタックを共有する Unified Stack 方式と、メインパスのみで一つのスタックを用いてサブパスは ITB を用いる M-P Only(Main Path Only) 方式の評価を行った。

PUSH と POP のタイミングは両方式とも同様で、以下のように定義する。

PUSH
: IB の出口予測が行われ、その予測出口に相当する命令がサブルーチンコールである場合に、リターン値が PUSH される。

POP
: IB の出口予測が行われ、その予測出口に相当する命令がリターン命令である場合に、スタックからリターン値を POP する。

ただし各フィールドの分岐命令の種別はプリデコードにより判別可能であると仮定する。

4.2.1 Unified Stack

一つのスタックですべてのパス上のリターン値を管理する方式である。あるパス上のリターン命令が別のパス上のリターン値を POP する可能性が高くスタックが壊れやすいため、性能は期待できない。分岐予測ミス時に修復を行わない方式 (Unified Stack) と、スタックを全て無効化する方式 (Unified Stack(Clear)) の 2 つを評価した。

4.2.2 M-P Only Stack

一番実行確率が高いメインパスでのみリターンアドレススタックを用い、その他のパスでは ITB を用いる。1 つのスタックをメインパスのみで用いることで複数パス上のリターン値の混在を回避し、スタックが壊れる可能性の低下を狙った方式である。分岐予測ミス時に修復を行わない方式 (M-P Only) と、スタックを全て無効化する方式 (M-P Only(Clear)) と、修復機構を用いてスタックの修復を行う方式 (M-P Only(Repair)) の 3 つを評価した。以下で、修復機構のチェックポイントと修復方法について述べる。

チェックポイント

スタックの内容を修復用に退避するタイミングは、メインパスと予測された IB をフェッチした時とする。

修復方法

修復は分岐予測ミス時に行われる。リタイアした IB をフェッチした時点をチェックポイントとして退避したスタックデータを修復に用いる。リタイアした IB がサブルーチンコール命令やリターン命令を実行しなかった場合は、退避したスタックの内容をそのまま修復に用いる。サブルーチンコール命令を実行した場合は退避したスタックにリターン値を PUSH したデータを、リターン命令を実行した場合は退避したスタックから 1 つ POP したデータを修復に用いる。また、サブパス側に実行が確定した場合は、スタックをすべて無効化する。

4.3 評価環境

評価は、4.2 節で述べた各リターンアドレススタックを 2.2.3 節で紹介した評価環境に導入して行った。ただし、フェッチ戦略と IB のアドレス予測について変更点がある。2.2.3 節で述べたように、メインパスの予測成功率は 80%強である。そのため、メインパスとサブパスを交互にフェッチするよりメインパスを重点的に伸ばす方が賢明である。よって、メインパス上の IB4 つをフェッチ後、サブパス上の IB を 1 つフェッチするよう変更した。また、IB の分岐が確定後、図 3 の IB Target Buffer 内のアドレスがターゲットアドレスと異る場合、IB Target Buffer の更新は常に行っていた。今回は、新たに 1 ビットのフラグを各エントリに追加し、2 回連続でターゲットアドレスが異る場合のみ IB Target Buffer の値を変更するよう変更した。リターンアドレススタックのエントリ数は 32 とする。

4.4 評価

ITB を用いた場合と 4.2 節で提案した各リターンアドレススタックを用いた場合の、P_r、P_i、P_a をそれぞれ図 6、図 7、図 8 に示す。

修復機構なしの Unified Stack では、ITB よりも性能が落ちる。これは、複数パス上のリターン値の混在によりスタックが破壊され続けるためである。分岐ミス時にスタックをすべてクリアする Unified Stack(Clear) では ITB と比較し P_r が 11.7%精度が向上した。これは、分岐予測ミス時にスタックが破壊される可能性が高く、その場合壊れたスタックを使い続けるよりスタックを無効化し一からやり直した方が、さらなるリターン値の混在を回避できるためである。そして P_r の向上により、P_i は 7.0%、P_a は 1.8%精度が向上した。

修復機構なしの M-P Only では、ITB と比較し P_r が 5.6%精度が向上した。複数パス上のリターン値の混在の回避により ITB より性能向上をしたが、分岐予測ミスによる誤ったスタック操作による性能低下が影響している。一方、M-P Only(Clear) では、Unified Stack(Clear) と同様スタッククリアの効果が現れ、P_r が 14.3%向上した。これにより、P_i は 8.6%、P_a は 2.3%精度が向上した。修復機構がある M-P Only(Repair) は、最も高い性能向上を果たし P_r は 16.6% 向上した。分岐予測ミス時に M-P Only(Clear) ではスタックの積み直しとなるが、M-P Only(Repair) では分岐予測前のスタックの内容を使うことができるため、精度がより向上したと考えられる。これにより、P_i は 9.9%、P_a は 2.6%精度が向上した。

また、M-P Only 方式の P_r をベンチマーク別に見たものを図 4 に示す。li のみスタッククリアを用いた方が効果が高い。これは、li が標準ライブラリ関数である longjmp を用いるためである。longjmp でのリターン値は前もって setjmp をした時に保存した PC であり、longjmp を呼出したサブルーチンコール命令の PC+1 がリターン値とはならない。よって、longjmp を実行した場合にはスタックが破壊される。このことから、li では分岐予測ミスをしていない場合でもスタックが破壊されるため、チェックポイントで退避したスタックのデータが壊れている可能性がより高く、修復するよりスタッククリアした方が性能が高い。これを回避するためには、longjmp を呼出す時にスタックをクリアすることが考えられる。

今回、最も性能の高かった M-P Only(Repair) の場合でも、理想的なリターンアドレス予測の場合に比べて約 12%の性能差が存在する。メインパスの実行

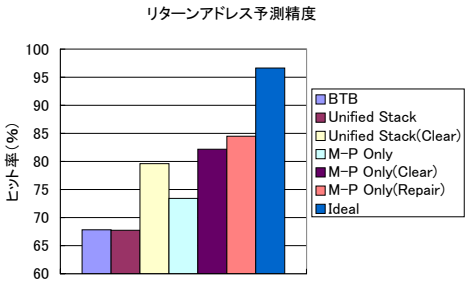

図 6 Return 予測精度 (P_r)

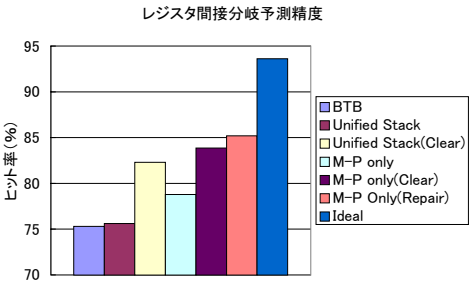

図 7 レジスタ間接分岐予測精度 (P_i)

確率は約 80%のため、平均して 5 回に 1 回予測がはずれる。これは、5 回に 1 回リターンアドレススタックが誤操作されることにつながる。よって、さらなる性能向上を図るためには、複数のリターンアドレススタックを設けパスごとにリターン値の管理を行い、制御がメインパスからサブアスに移った場合にも正しいスタックを使えるようにする必要がある。

表 4 ベンチマーク別 M-P only Stack のリターン予測精度 (P_r)

	ITB	M-P(Clear)	M-P(Repair)
go	48.5%	60.7%	69.7%
m88ksim	78.5%	91.6%	91.5%
compress	99.9%	96.7%	98.4%
li	72.6%	82.2%	80.3%
ijpeg	55.9%	86.5%	88.6%
perl	66.2%	70.7%	76.4%
vortex	53.2%	86.8%	86.5%

5. おわりに

5.1 まとめ

本論文では、VLDP の分岐予測機構において、リターンアドレス予測がボトルネックであることを示した。また、1 つのリターンアドレススタックをメインパスのみで用い、分岐予測ミス時に修復機構を用いることで、リターンアドレス予測成功率が最大 35.6%向上することを示した。

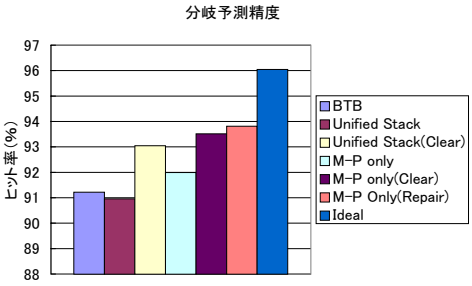

図 8 分岐予測精度 (P_a)

5.2 今後の課題

本論文で評価に用いたリターンアドレススタックの修復機構では、すべてのチェックポイントでスタックの全データを退避するため、ハードウェアコストが高い。今後は、予測確信度がある基準値以下の場合のみをチェックポイントとしてチェックポイントを減らす、またスタックの全データではなく、スタックトップからの一部のデータのみを退避し保存データ量を減らす、という 2 点から修復機構の最適化を行う。

謝辞 本研究の一部は、文部省科学研究費補助金 (基盤研究 (B) 課題番号 11480066) および (株) 半導体理工学研究センターとの共同研究によるものである。

参 考 文 献

1) 辻秀典, 安島雄一郎, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャの提案. 情報処理学会研究報告 2000-ARC-139, pp. 49–54, 2000.
2) 塚本泰通, 安島雄一郎, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャにおけるフェッチ機構. 情報処理学会研究報告 2001-ARC-143, pp. 25–30, 2001.
3) 塚本泰通, 安島雄一郎, 辻秀典, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャにおける命令ブロック構成の検討. 情報処理学会研究報告 2000-ARC-139, pp. 60–65, 2000.
4) Ryan Rakvic, Bryan Black, and John Paul Shen. Completion Time Multiple Branch Prediction for Enhancing Trace Cache Performance. *In Proc. ISCA 2000*, pp. 47–58, 2000.
5) K.Skadron, P.S. Ahuja, and M.MartonosiD.W. Clark. Improving Prediction for Procedure Returns with Return-Address-Stack Repair Mechanisms. *the 31st annual ACM/IEEE international symposium on Microarchitecture*, pp. 259–271, 1998.