並列事前実行機構におけるループカウンタ予測

津 邑 公 暁[†] 中 島 康 彦[††,†††] 中 島 浩[†]

既存のロードモジュールを高速化する手法として，再利用技術を用いた非対称な投機的マルチスレッディングである並列事前実行を提案している．従来，投機スレッドが利用するための入力の予測は，命令区間内の全ての入力を対象としてきた．本稿では，ループカウンタなどのフレーム内変数に代表されるような，アドレス自体が固定であり，かつ値が単調変化する変数のみを予測対象とすることにより，無駄な入力予測を省き，履歴を保持する表の擾乱を防ぐ手法を提案する．SPEC95 を用いた評価では，129.compress で従来 7% に留まっていたサイクル数削減率を 12% まで向上させることができた．

Loop Counter Prediction for Parallel Early Computation

Tomoaki Tsumura,[†] Yasuhiko Nakashima[††,†††]
and Hiroshi Nakashima[†]

We have proposed parallel early computation, an asymmmetrical speculative multi-threading with region reuse technology, as a speed-up technique for existing load modules. In this system, all inputs of regions have been predicted for speculative threads. This paper proposes a way to narrow targets of prediction down to effective variables: for example local variables in frames such as loop counters. This leaves out useless input predictions and keeps inputs history table clean. We show the ratio of eliminated cycle reaches 12% against 129.compress/SPEC95 benchmark program.

1. はじめに

既存のロードモジュールを高速化する手法として，スーパスケーラがある．しかし，命令レベル並列性を追究するたけでは性能向上が見込めなくなってきており，現在では**投機的マルチスレッディング**（Speculative Multi-threading: 以下，**SpMT**）の研究が広く行われている．

さて現在の SpMT では，投機実行スレッドと通常実行スレッド間で一対一のデータ受け渡しを行っている．この際，レジスタおよび主記憶入出力の比較機構が必要となる．また，一般に再コンパイルを行ったり，静的解析に基づいて付加情報の埋め込み（バイナリアノテーション）を行うことでこれを実現している．

これに対し，我々が進めている研究では，再コンパイルやアノテーションによる補助を完全に不要とすることを目的とする．また，通常実行スレッドが，投機実行スレッドおよび通常実行スレッドの実行結果を out-of-order に利用可能とすることで，より高い性能を得ることを目的としている．このような多対一のデータ受け渡しを可能とするためには，従来の SpMT が備えているようなレジスタおよび主記憶入出力の比較機構では不十分である．そこで我々は，このための一モデルとして**再利用**を用いた SpMT である**並列事前実行**を提案している[1)]．

さて，我々の提案している並列事前実行では，CAM（Content-Addressable Memory）を入出力比較のための表として用いることを想定している．しかしこれまでの我々の手法では，CAM のスループットを効果的に利用することができなかった．一般に中容量の CAM は低速であり，これを用いて高速に入出力比較を行うためには，より効率的に CAM の検索を行う必要がある．

また，並列事前実行では，投機実行スレッドは通常実行スレッドに先がけて，予測された入力を用いて命令区間を実行する．よって命令区間の入力を予測する必要があるが，従来は当該命令区間内の全ての入力を予測対象としてきた．このため，そもそも単調変化を行わず予測が不可能であるような入力も対象となっていた．また，入力となる読み出しアドレス次第では，命令区間の実行の度に参照アドレスの個数自体が変化する場合があり，このような場合は履歴表の構造から，本来正しく予測できるはずのアドレスに対しても，予測が正しく行えない場合があった．

† 豊橋技術科学大学
Toyohashi University of Technology
†† 京都大学
Kyoto University
††† 科学技術振興機構さきがけ研究 21
PRESTO, JST

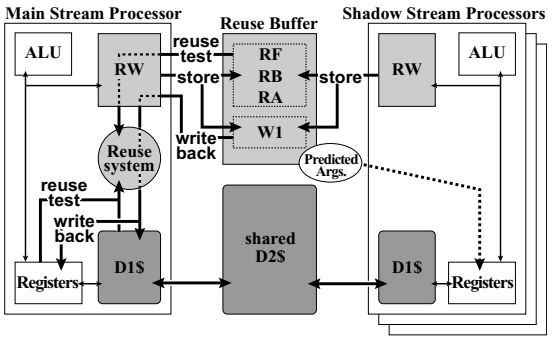

図 1 並列事前実行機構

そこで本稿では，命令区間の入力セットを互いに依存関係のないグループに分割し，それらを CAM のパイプライン機能を用いて並列検索することにより，検索オーバヘッドを削減する手法について述べる．また，並列事前実行の入力予測において，予測がヒットすることが期待できるアドレス，すなわちアドレス自体が固定であり，かつ格納値が単調変化する変数のみを予測対象とすることにより，無駄な入力予測を省略しつつ，予測対象となる参照アドレスの個数が変化しない手法を提案すし，シミュレーションによりその効果について調査した結果を述べる．

2. 並列事前実行

本章では我々が提案している，再利用を用いた非対称の SpMT である並列事前実行について述べる．

2.1 再 利 用

再利用とは，関数呼出しやループなどの命令区間において，その入力と出力の組を記憶しておき，再び同じ入力によりその命令区間が実行されようとした場合に，過去の記憶された出力を利用することで命令区間の実行自体を省略し，高速化を図る方法である．

再利用を行うためには，命令区間の入出力のペアを表に登録しておく必要がある．再び同じ命令区間を実行する必要が起こったとき，すでに同じ入力によりその命令区間が実行されている場合は，表から読み出すことで正しい出力値を求めることができる．入力セットが完全に一致していれば，実行結果は必ず同じになり，読み出した結果を検証する必要はない．

2.2 並列事前実行機構

並列事前実行機構の概要を**図 1** に示す．Main Stream Processor（以下，**MSP**）は，通常実行を行うプロセッサであり，可能である場合は命令区間の再利用を行う．一方，複数の Shadow Stream Processor（以下，**SSP**）は，MSP と並行して投機的実行を行う．演算器，レジスタ，一次キャッシュは各プロセッサごとに独立しており，再利用表，二次キャッシュ，および主記憶は全プロセッサで共有される．

MSP は，自身あるいは SSP が再利用表に登録したエントリを再利用する．一般的な SpMT との違いは，MSP は常に通常実行のみを行い，SSP は常に投機実行のみを行う点である．SSP は，MSP で実行された入力の履歴から，ストライド予測により入力を予測する．そしてその予測された入力を用いて命令区間（関数およびループ）の実行を MSP に先がけて行い，結果を再利用表に登録する．入力の予測が正しかった場合，MSP により通常実行が行われる時点では，命令区間は既に SSP により実行済であり，結果が再利用表に登録されているため，再利用により高速化を図ることができる．本機構では，図 1 中央に示した再利用表が，SSP と MSP の間の多対 1 のデータ引き継ぎを可能としている．

再利用表は，命令区間を記憶する表 **RF**，命令区間の入力値セットを記憶する表 **RB**，入力アドレスのセットを記憶する表 **RA**，そして出力値を記憶する表 **W1** から成る．また，MSP および各 SSP が命令区間の入出力を再利用表に登録する際，巨大な再利用表を毎回直接参照するとオーバヘッドが大きくなる．このため各 MSP/SSP に，作業用の小さい再利用表 **RW** を用意し，区間実行の終了時に一括して RW の内容を再利用表本体に登録する（store）．

MSP は，命令区間実行開始時に，再利用表を参照し（reuse test），入力セットが完全に一致するエントリが見つかった場合，当該エントリに対応する出力を W1 から一次キャッシュおよびレジスタに書き戻す（write back）．これにより，命令区間の実行が省略される．SSP は，再利用表のこれまでの入力セットから予測される入力セットを受けとり，投機実行を行う．投機実行の結果得られた入出力セットは，MSP と同様に命令区間の実行終了時に RW から再利用表本体に store される．

3. 入力のグループ分割

本章では，再利用対象となる命令区間の入力を，互いに依存しないグループに分割することにより，CAM の検索スループットを向上させる手法について述べる．

3.1 命令区間の入出力登録

再利用を行うためには，命令区間の入出力ペアを表に登録しておく必要がある．再び同じ命令区間を実行する必要が起こったとき，既に同じ入力によりその命令区間が実行されている場合は，表から読み出すことで正しい出力値を求めることができる．入力セットが完全に一致していれば，実行結果は必ず同じになり，読み出した結果を検証する必要はない．ここで命令区間の入力には，関数の引数に加え，関数およびループ内で参照される変数が含まれる．

さて，一般に命令区間内においては，ある入力値が参照され，その値によって次に参照すべき入力が決定することが多い．変数に主記憶アドレスが格納されている場合や，変数の値による条件分岐が発生する場合

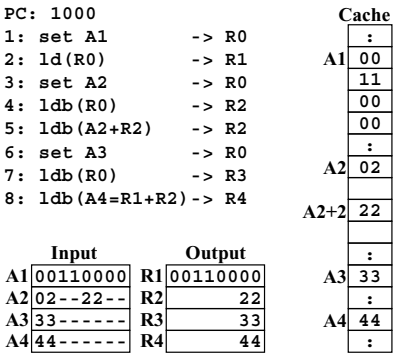

図 2 命令区間の入出力セット

などがその例である．つまり，同じ命令区間であっても，参照すべき入力セットの主記憶上アドレスは，常に同じであるとは限らない．命令区間内では，一部の入力の値によって次に参照すべき入力が決定し，またその入力の値によって次に参照すべき入力が決定する，ということが繰り返される．

図 2 に示す命令区間を実行する場合を例に説明する．いま命令区間の先頭アドレスが 1000 であるとする．第 2 の命令はアドレス A1 からロードした 4 バイトデータ 00110000 をレジスタ R1 に格納する．この場合，アドレス A1 およびデータ 00110000 が入力，レジスタ番号 R1 およびデータ 00110000 が出力となる．これらを再利用表に登録する．

命令 4 では，A2 および 02 が入力，R2 および 02 が出力となる．この際，A2 の残り 3 バイトに関しては，don't care として登録する（図中「-」部）．命令 5 では，A2+02 および 22 を入力として登録する．この際，22 は A2 の項に追加登録され，命令 2 の入力と併せて 02--22--となる．A2+01，A2+03 相当部分は，don't care のままである．

同様に登録を行っていくが，命令 8 では R1 および R2 は当該命令区間内で上書きされたレジスタであり，命令区間の入力として登録する必要がない．このため A4 および 44 のみを入力として登録する．

このように登録を行っていくと，ある一つの命令区間に対する入力セットのパターンは，**図 3** 上のようなツリー構造を形成する．ここで，ノードは参照すべき入力アドレス，エッジはその格納値であり，ルートからリーフまでの経路数が，命令区間の入力セットのパターン数となる．再利用表上では各エッジと次に参照すべきアドレスを 1 エントリとし，当該エントリの親エントリを指すインデクスと共に保存する（図 3 下）．

命令区間実行時にはルートノードから順に，記録されているアドレスを参照の上，得られた値と一致するエッジを連想検索機構を用いて選択することを繰り返し，入力セットが過去の実行時と完全に一致するか否かをテストする．そして，終端フラグが検出されると検索は終了し，対応する出力表から値を読みだして再

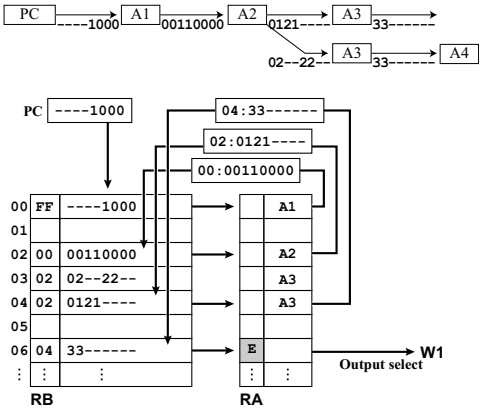

図 3 入力パターンと連想検索機構

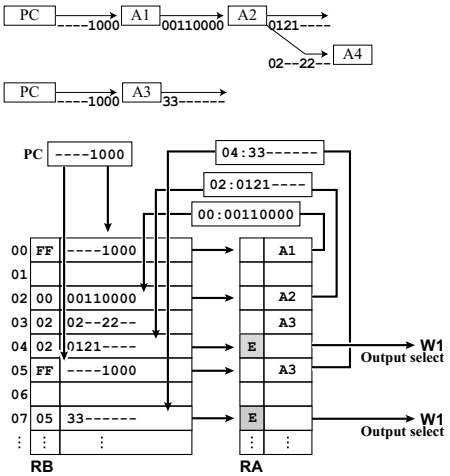

図 4 入力パターンの分割

利用が完了する．

3.2 グループ分割

上記で示した従来用いてきた検索手法は，動作は単純であるが以下のような問題がある．

- ツリー上流にあたる検索が完全に終了してから，下流の検索が行われるため，CAM のスループットを生かすことができない．
- 命令区間の入力が完全に一致しなければ，出力を再利用できない．区間内の一部の出力が，入力セットの一部にのみ依存する場合，原理的にはその部分入力セットさえ一致すれば出力の一部は再利用可能となるはずである．

これを解決するために，入力パターンを互いに依存のないグループに分割し，各グループ毎に過去の入力パターンを保持するよう木構造を構成する．そして，CAM のパイプライン機能を用いてそれぞれを並列に検索することで，命令区間の部分的再利用を実現しつつ再利用テストのオーバヘッドを削減することが可能となる（**図 4**）．

4. 主記憶値予測

本章では，従来 SSP で行ってきた入力予測の問題点を明らかにし，入力値の分類による入力値予測の効率化手法について述べる．

4.1 従来手法と問題点

前述したように，SSP は命令区間の入力値をストライド予測し，それを入力として MSP に先がけて命令区間の実行を行う．またこの際，各入力にフラグ（**C-FLAG**）を設け，再利用表登録時以降，当該アドレスに対して store が発生していないことが保証される場合など，そのアドレスを実際に読みだして比較する必要のない場合は，この C-FLAG を用いて入力値テストを省略する機構を導入している[2)]．

図 2 に示した命令列を実行した場合の，入力予測の概要を**図 5**(a) に示す．`A1` については値が変更されることがないとする．これにより再利用テスト自体を省略でき，`A1` の履歴を保持する必要はない．また，この命令区間を実行する度に，アドレス `A2` の値は，02，03，04，05 と変化していくと仮定する．このような場合，`A2` の格納値が変化するため，1 度目と 2 度目の実行では `A2+4` の位置が変化し，`A2` に対する Mask 位置も変化してしまうため，予測が不可能である．また，3 度目の実行では参照されるアドレス数そのものが変化してしまうため，表のカラムがずれ，`A2` の後続の入力 `A3`，`A4` についても予測が成功しなくなってしまう．

そもそも入力値予測が当たることが期待できるアドレスは，ループカウンタなどのフレーム内局所変数に代表されるような，まずそのアドレス自体が変化せず，かつその格納値が単調変化するもののみである．同様の変数には，ラベルにより参照される大域変数や，スタックポインタやフレームポインタをベースレジスタとして参照される局所変数がある．逆に，これら以外のアドレスは予測の対象とすべきではない．従来の手法の問題点をまとめると以下のようになる．

- 例えば配列などの場合，配列添字は単調変化するが，配列の格納値は一般に単調変化しない．また，添字をアドレスとして用いる主記憶参照は，アドレス自体が変化し，参照されるアドレス数そのものも変化する可能性がある．このような場合，再利用表の同一列の変化には規則性がなくなり，予測の的中率が激減する．
- 格納値が変化しないアドレスを予測したり，格納値の変化において過去の履歴から全く規則性が見いだせないアドレスに関しては，予測そのものを省略すべきである．また，マスク位置が変化するアドレスに関しては，マスク位置の変化まで予測するのは困難であるため，やはり予測を省略すべきである．

これらは，従来手法では登録された全アドレスを同

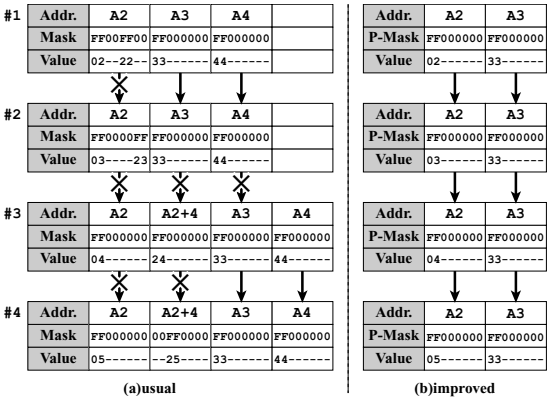

図 5 入力予測の変化

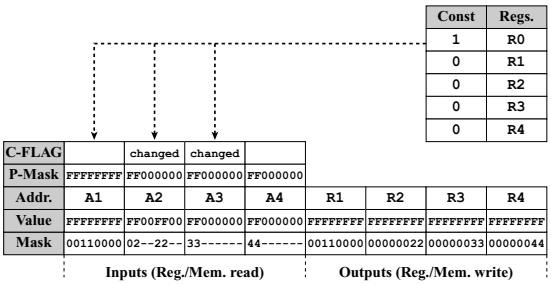

図 6 入力予測改良のための機構

等に扱っている点に原因がある．これを解決するために，予測が当たることが期待できるアドレスと，そうでないアドレスを区別する必要がある．

4.2 提 案 手 法

予測が当たる可能性のあるアドレスを区別するため，以下のような主記憶値予測機構を導入する．予測機構の概要を**図 6** に示す．

まず，アドレス自体が変化するか否かの判断のために，load 命令実行時のアドレス計算が参照するレジスタに，フラグ（**Const-FLAG**）を設ける．スタックポインタやフレームポインタなど，命令区間実行中に変化しない値に用いられるレジスタについては，Const-FLAG をセットする．またその他のレジスタについても，定数セット命令の実行時に Const-FLAG をセットする．

また，格納値の変化に規則性が見られるアドレスであるか否かの判断のために，当該アドレスを履歴保存対象とするかどうかを示す履歴マスク（**P-Mask**）を設ける．このマスクは，当該アドレスを再利用表に新たに登録する時点でリセットする．また，load 命令実行時に，当該アドレスを生成したレジスタの Const-FLAG を参照し，これがセットされている場合に，P-Mask のうち load 対象であるバイト位置をセットする．

図 2 で示した命令列を用いて具体的に説明する．まず命令 1 が実行される際，レジスタ `R0` に格納され

表 1 シミュレーション時のパラメータ

D1 Cache 容量	32 KBytes
ラインサイズ	32 Bytes
ウェイ数	4
レイテンシ	2 cycles
Cache ミス ペナルティ	10 cycles
共有 D2 Cache 容量	2 MBytes
ラインサイズ	32 Bytes
ウェイ数	4
レイテンシ	10 cycles
Cache ミス ペナルティ	100 cycles
Register Window 数	4 sets
Window ミス ペナルティ	20 cycles/set
ロードレイテンシ	2 cycles
整数乗算 〃	8 cycles
整数除算 〃	70 cycles
浮動小数点加減乗算 〃	4 cycles
単精度浮動小数点除算 〃	16 cycles
倍精度浮動小数点除算 〃	19 cycles
Read アドレス	256 word/RW
Write アドレス	256 word/RW
RF エントリ数	256
RB/RA/W1 エントリ数	4096
SSP 台数	3

るのはアドレス定数 A1 であるため，R0 に対応する Const-FLAG をセットする．次に命令 2 が実行される際，3.1 節で示した従来手法と同様に入力として A1 および 00110000，出力として R1 および 00110000 が記録される．このとき，アドレスとした用いた R0 に対応する Const-FLAG はセットされているため，A1 の P-Mask として 4 バイトデータ 00110000 に対応する FFFFFFFF がセットされる．また，R1 に対応する Const-FLAG はリセットされる．

この手法を用いた場合従来と異なるのは，命令 5 の実行の際である．アドレスとして用いた R2 に対応する Const-FLAG がリセットされているため，A2+4 に対応する P-Mask はセットされない．このように命令の実行を進めていくと，C-FLAG がセットされ，かつ P-Mask がセットされている入力は，A2 の第 1 バイトおよび A3 の第 1 バイトのみとなる．このようにして A2 および A3 のみの入力履歴を保持し，これらの値の予測を行うことによって，図 5 に示すように，従来手法では予測が正確に行えなかった A2 および A3 について正しく予測が行えるようになる．なお，アドレス A2+4 については，予測値を求めずに主記憶を直接参照することで，正確な値を得ることができる．

5. 評　　価

以上で述べた，入力のグループ分割および主記憶値予測機構を付加した，並列事前実行を行うシミュレータを開発し，評価を行った．

5.1 評価環境

並列事前実行機構を実装した単命令発行の SPARC-V8 シミュレータを用い，MSP および SSP のサイクルベースシミュレーションを行った．評価に用いたパラメータを**表 1** に示す．なお命令レイテンシは，SPARC64-III[3] を参考にしている．

ハードウェア構成に関しては，一次キャッシュを 32KB：4way とし，共有二次キャッシュは 2MB：4way とした．また，一次キャッシュ，共有二次キャッシュ，主記憶のレイテンシはそれぞれ，2cycle，10cycle，100cycle と仮定した．

再利用表については，まず RW を，32Byte 幅 ×256 エントリ ×4 セットで，一次キャッシュと同サイズの 32KB とし，レイテンシも一次キャッシュと同じと仮定した．RB についても，二次キャッシュと同サイズの 2MB とした．また，RB を構成する CAM のレイテンシとして，レジスタとの比較に 32Byte/cycle，キャッシュとの比較に 32Byte/2cycle を仮定した．

MSP が D1/D2 に対し store を行うと，SSP 内の D1 で invalidate が発生し，後続命令をそれぞれ 10cycle/100cycle だけ stall させる．以後 SSP には，D1 ミスとして観測される．また，MSP/SSP で D2 に対し load が発生した場合，それより 100cycles 後以降は MSP/SSP 内の D1 上で hit すると仮定した．

CAM の構成は，MOSAID 社の DC18288[4] を参考にしている．今回の評価では CAM の大きさを 128KB（深さ 4K），2MB（深さ 64K）と仮定したが，DC18288 は幅 288bit× 深さ 64K=2MB の大きさを持つ CAM であり，この仮定は現実的なものである．

5.2 SPEC95

SPEC95（train）を gcc-3.0.2（-msupersparc -O2）によりコンパイルし，スタティックリンクにより生成したロードモジュールを用いて評価を行った．**図 7** にその結果を示す．

グラフ中の凡例はサイクル数の内訳を示しており，exec は命令サイクル数である．test(r)，test(m) はそれぞれ，再利用表とレジスタ，再利用表とキャッシュの比較に要したサイクル数である．write は，命令区間の出力を再利用表からレジスタおよびキャッシュへ書き戻すのに要したサイクル数である．また，D1$，D2$，および window は，それぞれキャッシュミスペナルティとレジスタウィンドウミスによるペナルティを表している．

また，各ベンチマークプログラムのグラフに関しては，それぞれ左から順に

(M) SSP と再利用のいずれもなし
(R^4) 従来の並列事前実行（深さ 4K）
(R^4_L) 主記憶値予測の改良（深さ 4K）
(R^{64}) 従来の並列事前実行（深さ 64K）
(R^{64}_L) 主記憶値予測の改良（深さ 64K）

が要したサイクル数を表しており，それぞれ (M) を 1 とする正規化を行っている．

このグラフより，129.compress では従来 7% であったサイクル数削減率が 12% まで向上し，一部のプログラムで提案手法の効果が得られることが分かった．

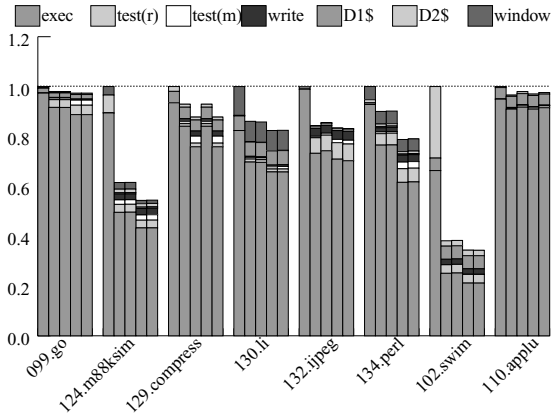

図 7 MSP の実行サイクル数（SPEC95）

132.ijpeg においても，CAM の深さを 64K とした場合には僅かに性能向上が見られた．

次に，効果が得られた 129.compress を用いて，(R^4) と (R^4_L) で命令区間の再利用状況にどのような変化があったか調査を行った．主な命令区間について，提案手法の適用前後における再利用ヒット回数の変化を**表 2** に示す．なお命令区間名が括弧で括られているものは，その関数内に存在するループであることを示している．

この結果より，関数 getbyte，関数 readbytes，および関数 readbytes 内ループにおける再利用率の向上が顕著である．これら，getbyte および readbytes 内ループでは大域配列 InBuff，readbytes 内では引数配列 buf が入力として参照されており，4 章で示したとおり，提案手法により配列格納値の予測が大幅に向上していることが分かった．

5.3 考　　察

本稿で行った評価では，129.compress 以外のベンチマークでは提案手法による有意な結果が得られなかった．表 2 において再利用率が向上している命令区間を調べると，たとえば readbyte 内ループにおいては，ループイタレーション間にデータの依存関係が存在しないことが分かる．このため提案手法の効果が明確に現れたのだと考えられる．逆にループイタレーション間にデータの依存関係が存在する場合，MSP/SSP でループ間の並列処理を行っても，有効な結果が得られないと予想される．

本稿で提案した主記憶値予測機構をより有効に機能させるには，更に入力の依存性の解析を動的に行い，依存のあるアドレスに関して MSP/SSP 間で値を待ち合わせる機構を実現することが必要となるであろう．

6. おわりに

本稿では，並列事前実行機構の入力値予測において，(1) 参照アドレスが固定で変化しないもの，かつ (2) そのアドレスの格納値が単調に変化するものに限定することによって，予測のヒット率を向上させる手法を示した．

表 2 129.compress における再利用状況

アドレス	命令区間	ヒット回数 (R^4)	(R^4_L)
0001aaec	(compress)	160923	160878
0001ace4	output	17396	17282
0001ae80	(output)	290814	290719
0001af28	(decompress)	156	3174
0001af98	(decompress)	331629	331859
0001b020	(decompress)	390537	396289
0001b0c0	getcode	310537	332422
0001b570	getbyte	63535	**442706**
0001b5a8	putbyte	825430	838862
0001b5c0	readbytes	150	**8094**
0001b600	(readbytes)	3939	**266075**
0001b8a0	(PIo2)	172010	172010
0001b8ec	(getranchar)	98550	98550
0001b910	ran2	5106	5106

まず各レジスタに対応するフラグを設けることで，当該レジスタが定数であるかを判断する機構を実現した．また，アドレスの各部分を履歴の保存対象とするか否かを示す履歴マスクを設け，前述のフラグによりマスク位置が変化することが予測される場合は，その部分を履歴マスクすることによって，入力予測の対象から外す手法を提案した．

本手法を実装したシミュレータを用い SPEC95 で評価を行い，再利用率向上が得られるプログラムが存在することを示した．従来 129.compress においては，従来サイクル数削減率が 7% に留まっていたのに対し，本手法によりこれを 12% 以上まで引き上げることができた．

今後の課題としては，MSP/SSP 間で値を待合わせる機構を実現することにより，並列実行において有効な計算結果が得られるようにする必要があると考える．

参 考 文 献

1) 津邑公暁, 笠原寛壽, 清水雄歩, 中島康彦, 五島正裕, 森眞一郎, 富田眞治: 大容量汎用 3 値 CAM を用いた並列事前実行機構の効率的実現, 先進的計算基盤システムシンポジウム SACSIS2004 論文集, 情報処理学会, pp. 251–259 (2004).
2) 津邑公暁, 中島康彦, 五島正裕, 森眞一郎, 富田眞治: 並列事前実行機構における主記憶値テストの高速化, 情報処理学会論文誌：コンピューティングシステム, Vol. 45, No. SIG 1(ACS 4), pp. 31–42 (2004).
3) HAL Computer Systems/Fujitsu: *SPARC64-III User's Guide* (1998).
4) MOSAID Technologies Inc.: *Feature Sheet: MOSAID Class-IC DC18288*, 1.3 edition (2003).