AIプロセサのためのツインレジスタアーキテクチャ

2W-1

的場 司, 前田 賢一, 高宮 健, 斎藤 光男

(株) 東芝

1. はじめに

我々は、Prologのバックトラックを高速化するためにツインレジスタアー キテクチャを開発した。本アーキテクチャは、バットラック処理の本質であるレジスタ内容のスタックへの退避/復帰を高速に行なうメカニズムをサポートする。これまでに我々は高速ＡＩプロセサＩＰ７０４の開発を行なってきた。ＩＰ７０４は、WAM(Warren Abstract Machine) [WAMM] が提唱する命令セットを効率良く実行するプロセサである。我々はツインレジスタアーキテクチャの有効性を評価するために、これをＩＰ７０４上に実現した。本報告ではツインレジスタアーキテクチャの概念, 実現方法, 評価結果について述べる。

2. バックトラック動作

Prolog実行過程で、同一のヘッドを持つ複数の候補節を順番に調べていく場合、ローカルスタック中にチョイスポイントを生成し、現在のレジスタの内容を退避する必要がある。これらは候補節を調べる各々の処理に対する情報となる。ある一つの候補節の同一化に失敗するとバックトラックし、チョイスポイントの情報をレジスタに復帰した後、次の候補節を調べる。チョイスポイントのサイズは最低７ワードであり、チョイスポイントのアクセス頻度はメモリアクセス全体の２０～３０%である[TICK85]。そのためバックトラック時のメモリアクセス回数を削減することはPrologの高速実行のために効果がある。ＲＩＳＣプロセサの中には、多重レジスタウインドウアーキテクチャを採るものがあるが、WAMのバックトラックサポートのためには適していない。シャドウレジスタアーキテクチャはWAMのバックトラックに向いているが、これを実現するためには大規模なハードウェアが必要となる。

3. ツインレジスタアーキテクチャ

3. 1 概念

ツインレジスタアーキテクチャの特徴を以下に示す。

(1) 仮想的な無限個のレジスタファイルの実現。
(2) caller-callee 間でレジスタ内容が変化しない。
(3) 自動的にレジスタ退避/復帰を行なうハードウェア。
(4) 比較的小規模なハードウェア。

ツインレジスタで実現しようとする仮想的な無限個のレジスタファイルの例を図１に示す。Prologにおいてある候補節の同一化を開始する動作がｃａｌｌに対応し、逆にバックトラックがｒｅｔｕｒｎに対応する。レジスタファイルが切り替わった後でも、値を更新されないレジスタは元の値を読み出すことができる。また更新されたレジスタは次にレジスタファイルが切り替わった後で自動的にメモリに退避される。

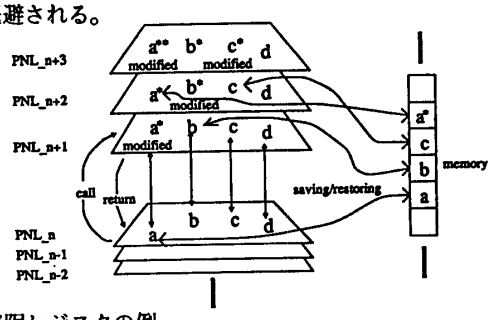

図１ 無限レジスタの例

Twin Register Architecture for AI processor
Tsukasa MATOBA, Ken-ichi MAEDA,
Takeshi TAKAMIYA, Mitsuo SAITO
Toshiba corporation

3. 2 実現方法

図２にＩＰ７０４に組み込まれたツインレジスタの構成図を示す。図中点線で囲まれたツインレジスタユニットは、１対のレジスタファイルと３種類のフラグ情報と３種類の制御レジスタによって構成される。

- レジスタファイル
  {(FR0, BR0), (FR1, BR1), ..., (FRn-1, BRn-1)}
- 選択フラグ（Ｓフラグ）, 更新フラグ（Ｍフラグ）
  {(S0, M0), (S1, M1), (S2, M2), ..., (Sn-1, Mn-1)}
- 退避/復帰モードフラグ（ＳＲＭＦ）
- 退避/復帰レジスタポインタ（ＳＲＰ）
- 退避/復帰レジスタ最大アドレス（ＳＲＬＰ）
- 退避/復帰スタックポインタ（ＳＳＰ）

Ｓフラグは、１対のレジスタのうちどちらが命令の実行に用いられるかを示す。Ｍフラグは、それに対応するレジスタの値が更新されたことを示す。ＳＲＭＦは、これが０のときレジスタ退避を、１のとき復帰を行なうことを示す１ビットのフラグである。ＳＲＰは退避/復帰すべきレジスタのアドレスを、ＳＲＬＰは退避/復帰すべきレジスタの最大アドレスを、ＳＳＰは退避/復帰を行なうローカルスタックのアドレスをそれぞれ保持するレジスタである。

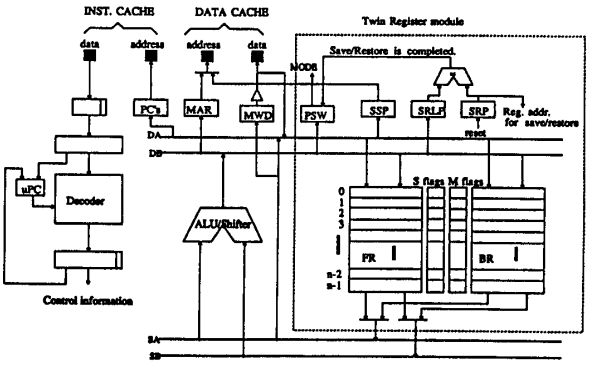

図２ ＩＰ７０４に組込まれたツインレジスタの構成

ツインレジスタの動作ルールを以下に示す。

Ⅰ. システムリセット時、チョイスポイントを生成する命令(try-me-else) の実行時にＳＲＭＦはリセットされる（退避モードとなる）。

Ⅱ. チョイスポイントを消去する命令(trust-me-else) の実行時、ＳＲＭＦはセットされる（退避モードとなる）。

Ⅲ. ユニフィケーションに失敗したとき、ＳＲＬＰより番号の低いＳフラグのうち、対応するＭフラグがセットされているものを反転する。

Ⅳ. 命令実行時にはＳフラグが指すレジスタから値が読み出される。

Ⅴ. Ⅰ, Ⅱのイベントが発生したとき、全てのＭビットはリセットされる。

Ⅵ. 退避モードにおいて、

(1) 命令実行結果をレジスタに書き込むときは以下のルールによる。

```
if (!Mi) {
    Mi = 1;
    if (!Si)        FRi = value;
    else            BRi = value;
    Si = !Si;
} else      {
    if (Si)         FRi = value;
    else            BRi = value;
}.
```

(2) レジスタ退避のために読み出す場合は以下のルールによる。

```
if (!Mi) {
    if (Si)     *(--SSP) = FRi;
    else        *(--SSP) = BRi;
} else {
    if (Si)     *(--SSP) = BRi;
    else        *(--SSP) = FRi;
}.
```

(3) 退避が完了する前に次のチョイスポイント生成命令を実行しようとすると、この実行は退避が完了するまで待たされる。

(4) 退避が完了する前にチョイスポイント消去命令を実行しようとすると、退避は打ち切られる。

Ⅶ. 復帰モードにおいて、

(1) 命令実行結果をレジスタに書き込むときは以下のルールによる。

```
if (Si)     FRi = value;
else        BRi = value;.
```

(2) レジスタ復帰のために書き込む場合は以下のルールによる。

```
if (Si)     BRi = *(SSP++);
else        FRi = *(SSP++);.
```

(3) 復帰が完了する前に次のチョイスポイント生成命令を実行しようとすると、復帰は打ち切られる。

(4) 復帰が完了する前にチョイスポイント消去命令を実行しようとすると、この実行は復帰が完了するまで待たされる。

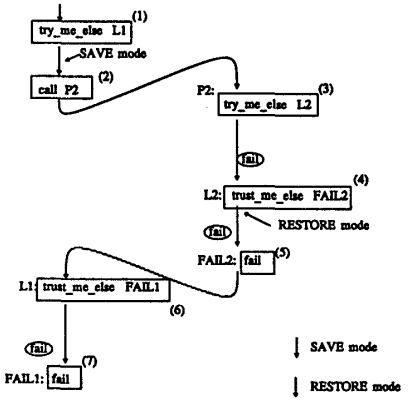

図3 Prologバックトラックの例

次に上記ルールに基づいたツインレジスタの動作について述べる。例として図3にPrologのバックトラックシーケンスを示す。初期状態において、全てのSフラグ，Mフラグはリセット、ＳＲＭＦは退避モードを示しているものとする。まず述語Ａが呼び出され、命令'try-me-else L1'が実行されると、チョイスポイントＡが生成される。このとき制御レジスタは以下のように設定される。

ＳＲＰ←０，
ＳＲＬＰ←ｒｍａｘＡ
ＳＳＰ←チョイスポイントＡの先頭アドレス

ｒｍａｘＡは、述語Ａを同一化するためのプログラムで使用する最大のレジスタのアドレスを意味する。すなわちＲ０～ＲrmaxA のレジスタが退避される。退避は命令実行の空きサイクルで自動的に実行される。それ以降のレジスタ読出し，書き込み，退避は、それぞれルールⅣ，Ⅵ-(1)，Ⅵ-(2)に従って実行される。さらに続いて述語Ｂが呼び出され、命令'try-me-else L2'が実行されると、チョイスポイントＢが生成される。このとき制御レジスタは以下のように設定される。

ＳＲＰ←０，
ＳＲＬＰ←ｒｍａｘＢ
ＳＳＰ←チョイスポイントＢの先頭アドレス

これにより述語Ｂの同一化で変更される可能性のあるレジスタＲ０～ＲrmaxB が退避される。第１候補の同一化に失敗すると、ＳＲＬＰの値より低いアドレスのＳフラグのうちＭフラグがセットされているものを全て反転する。これによりレジスタ内容はチョイスポイントＢ生成時と等しくなる。(shallow Backtracking)

図３のポイント(4) において、最後の候補のために命令'trust-me-else FAIL2'が実行されると、この命令はチョイスポイントＢを消去し、ＳＲＭＦをリセット（復帰モード）する。制御レジスタは以下のように設定される。

ＳＲＰ←０，
ＳＲＬＰ←ｒｍａｘＢ
ＳＳＰ←チョイスポイントＡの先頭アドレス

これにより、続いて起こる可能性のあるバックトラックの準備のために、ルールⅦ-(2)に従って、裏レジスタにチョイスポイントＡの情報が復帰される。

ツインレジスタアーキテクチャでは、基本的にはレジスタ退避／復帰のために余計なクロックサイクルを必要としない。退避／復帰は以下の場合自動的に実行される。

(1) メモリアクセスを伴わない命令の実行時。
(2) ”ｎｏｐ”命令の実行時。
・ブランチ命令の次のスロット。
・命令キャッシュミス時。

4．性能評価

図４に'try-me-else' 命令のマイクロプログラムの例を示す。TYPE-Aはツインレジスタでない一般的なプログラム、TYPE-Bはツインレジスタを用いたＩＰ７０４のマイクロプログラムである。TYPE-Aでは、引数レジスタが無い場合でも９ステップかかるのに対し、TYPE-Bでは、引数レジスタの数にかかわらず４ステップで実行する。

ＩＰ７０４のソフトシミュレータを用いて、８Ｑｕｅｅｎプログラムを実行したところ、シングルレジスタの場合と比較して、約１５％速度が向上した。また本アーキテクチャは手続き型のプログラムでも効果があり、Dhrystone ベンチマークでも約１０％速度が向上した。

5．おわりに

Prologの基本メカニズムであるバックトラックを高速化するためのツインレジスタアーキテクチャについて述べた。１対のレジスタファイルと小規模なハードウェアで仮想的な無限個のレジスタファイルを実現した。本アーキテクチャはPrologのみならず、手続き型プログラムの実行にも効果があることがわかった。

参考文献

[SAITO] Mitsuo Saito, "*An AI Processor, Risc Architecture with Hardware Support for AI Languages*", Denshi Tokyo N.27, 1988.

[WARR] D.H.D.Warren, "*An Abstract Prolog Instruction Set.*" Tech Note 309 Artificial Intelligence Center,SRI International,Oct.1983.

[TICK84] E.Tick and D.H.D.Warren ,"*Towards a Pipelined Prolog Processor* " 1984 International Symposium on Logic Programming, pp29-40,1984

[TICK85] E.Tick, " *Prolog Memory-Referencing Behavior.*" Technical Report 85-281, Computer Systems Laboratory, Stanford University, 1985.

[TICK86] E.Tick, " *Lisp and Prolog Memory Performance.*" Technical Report 85-291, Computer Systems Laboratory, Stanford University, 1986.

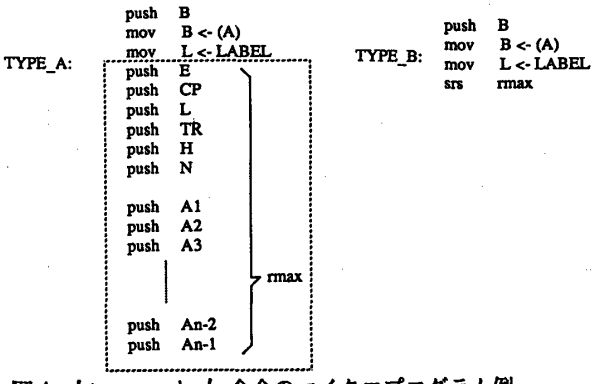

図４ 'try-me-else' 命令のマイクロプログラム例