7P-2 多パイプライン方式アレイ・プロセッサのレジスタわりあて方式

金田　泰　安村通晃　(日立製作所中央研究所)

1. はじめに

従来のパイプライン方式アレイ・プロセッサでは演算器やレジスタは比較的少数個だったが、さらに性能をあげるには演算器やレジスタの数をふやして多数の演算を並行しておこなう必要がある。そのような計算機においてすべてのレジスタとすべての演算器とを接続すると、その配線量は膨大になって性能を圧迫する。したがって、その接続にある種の条件をもうけたうえでそれをコンパイラが自動的にみたす方式を考案した。オブジエクト・プログラム上ではその条件はレジスタ番号の関係としてあらわされるので、これはレジスタわりあて上の問題になる。

2. ハードウエアの仮定とレジスタわりあての課題

レジスタと演算器との接続関係には種々あるが、本研究で注目したのは図1のように演算器からレジスタへのかきこみパスのうちのいくつかをまとめた場合である。かきこみパスを共有するレジスタをまとめてレジスタ族とよぶことにする。レジスタから演算器へのパスもへらさなければ全体としての配線量を十分にへらすことはできないが、それについてはここではかんがえない。

つぎに、図1のようなハードウェアのためのコンパイラのレジスタわりあてには、どのような課題があるかをかんがえる。

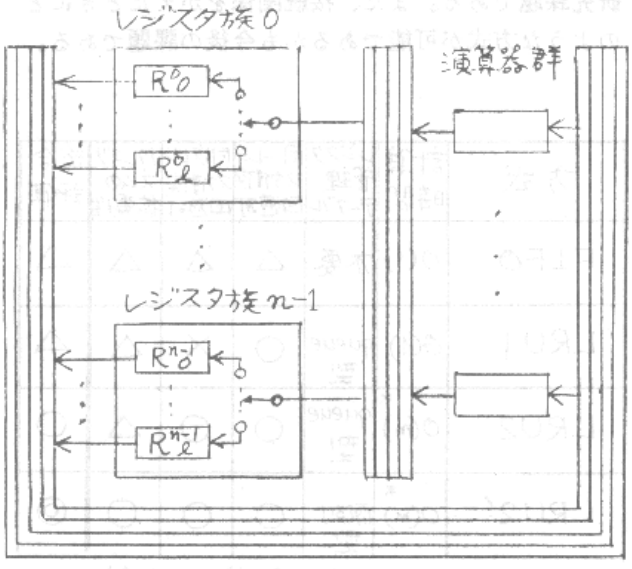

図1 ハードウェアの仮定

第1の課題は同一レジスタ衝突の回避である。並列パイプライン方式アレイ・プロセッサでは一般に複数の命令が並行して実行されうる。ところが、同一のレジスタを使用する命令が複数あるときには、先行する命令の実行が終了するまで後続の命令は実行を開始することができない場合がある。この現象を同一レジスタ衝突とよぶ。同一レジスタ衝突のうち重要なのはつぎの2つである。

(1) よみだしと後続のかきこみとの衝突

(2) よみだしどうしの衝突

(なお、かきこみと後続のよみだしとは並行しておこなえる)。このうち(2)については本報告では省略する。

第2の課題は同族レジスタの衝突の回避である。1つのレジスタ族に属するレジスタには仮定により並行してかきこむことができないので、複数のかきこみ命令があるときには先行する命令の実行はまたされる。これが同族レジスタ衝突である。この衝突をさけるため、レジスタわりあてのとき同族内のレジスタへのかきこみを極力さける必要がある。これが同族レジスタ衝突である。

3. 各種の方式とその比較

本章では、全レジスタを順にわりあてる各種の方式を比較検討する。なお、かんたんのためレジスタ不足時の処理は除外する。比較結果は図3にしめした。

(1) ＦＩＦＯ方式

全レジスタにあらかじめ順序をつけておき、命令をうえから走査しながらレジスタをその順にわりあてる。すべてのレジスタをわりあてたら、ふたたび最初からわりあてる。ただし使用中のレジスタはわりあて対象から除外する。図2にＦＩＦＯ方式によるわりあての例をしめす。(図でRをつけたのがレジスタである)。わりあてはレジスタのかきこみ点でおこなうので、この方式は「もっとも昔にかきこまれた(あるいはかきこまれたことがない)レジスタをわりあてる方式」ということができる。

(2) ＬＲＵ1方式

最後の参照がベクトル命令列上で最遠のレジ

レジスタ未わりあての命令列	命令の説明および付帯条件	FIFO方式 わりあて順	FIFO方式 わりあて内容	LRU方式 わりあて順	LRU方式 わりあて内容
Ra ← A(1:N)	A(1),…,A(N)の値をレジスタRaにいれる	1	RaにR0をわりあてる	1	RaにR0をわりあてる
Rb ← B(1:N)		2	RbにR1をわりあてる	2	RbにR1をわりあてる
Rc ← Ra + Rb	Ra,Rbの内容を加算してRcにいれる.	3	RcにR2をわりあてる	3	RcにR2をわりあてる
Rd ← abs(Rc)	} この間でRb,Rcは使用されていないとする.	4	RdにR3をわりあてる	4	RdにR3をわりあてる
⋮					
Rx ← Ra + Ru		m	RxにRm-1をわりあてる	m	RxにRm-1をわりあてる
Ry ← Rx * Rv		m+1	RyにR0をわりあてる	m+1	RyにR1をわりあてる
Rz ← C(1:N)		m+2	RzにR1をわりあてる	m+2	RzにR2をわりあてる

図2 FIFO方式とLRU1方式によるわりあての例

スタをわりあてる方式である。コンパイラは未使用のレジスタのqueueをもち、その先頭からわりあてていく。使用がおわったレジスタはqueueの最後につける。図2にはＬＲＵ１方式によるわりあての例もしめした。この方式はＦＩＦＯ方式よりはよいとかんがえられるが、同族レジスタ衝突をふせぐことができない。

(3) ＬＲＵ２方式

ＬＲＵ１方式に最小限の修正をくわえて同族レジスタ衝突をふせぐ方式である。コンパイラは未使用レジスタのqueueをもち、その先頭からわりあてていくが、わりあての際queueを走査して同族レジスタをさがしだし、それをqueueの最後にうつす。使用がおわったレジスタもqueueの最後につける。

(4) ＬＲＵ２'方式

機能的にはＬＲＵ２とほぼ同一だが、queueではなく配列型テーブルを使用する。わりあてに際してはテーブルの要素ごとに（すなわち各レジスタについて）わりあて評価関数を計算し、その値が最大のレジスタをわりあてる。レジスタの使用状況はテーブルに記録し、関数値計算時に使用する。評価関数としては、つぎのようなかたちのものを使用すればよい。

$D(R^i_j) = \min(Duse(R^i_j), Ddef(R^i_1), Ddef(R^i_2), \ldots, Ddef(R^i_{ni}))$ .....(a)

各料はつぎのような意味をもつ。

Duse(S)： レジスタSを直前によみだした命令からSをわりあてるべき命令までの距離（命令列上での命令数ではかる）。

Ddef(S)： レジスタを直前にかきこんだ命令からSをわりあてるべき命令までの距離。

$R^i_1, R^i_2, \ldots, R^i_{ni}$： レジスタ族iを構成するレジスタ。

式(a)のかわりに$D(R^i_j) = Duse(R^i_j)$とするとＬＲＵ１と機能的に同一になる。また、$D(R^i_j) = Ddef(R^i_j)$とするとＦＩＦＯ方式と同一になる。また、$D(R^i_j)$の項としてレジスタ衝突以外の条件をいれることもできる。したがってＬＲＵ２'はもっとも柔軟な方式ということができる。

4. 結論

レジスタへのかきこみパスをへらした並列パイプライン方式アレイ・プロセッサに対する各種のレジスタわりあて方式を比較検討した結果、ＬＲＵ２'方式がよいという結論をえた。それは、ＬＲＵ２'方式にはレジスタ数に比例するわりあて時間がかかるという欠点はあるものの、各種のレジスタ衝突を回避し、かつほかの条件もとりこめる柔軟性があるからである。ただし本研究で検討したのは命令列順にわりあてる方式だけであり、わりあて順を柔軟にしたときにどのような方式が可能であるかは今後の研究課題である。また、接続関係をかえたときにどのような方式が可能であるかも今後の課題である。

方式	計算時間	レジスタ管理テーブル	同一レジスタ衝突回避可	同族レジスタ衝突回避可	アルゴリズムの拡張性	総合評価
FIFO	O(1)	不要	△	△	△	△
LRU1	O(1)	queue型	○	×	△	△
LRU2	O(m)*	queue型	○	○	△	○
LRU2'	O(m)*	配列型	○	○	○	◎

図3 各わりあて方式の比較

＊ mはレジスタ数