4T－2

条件実行制御を用いたＶＬＩＷにおける大域命令スケジュール

遠藤浩太郎　境隆二　鈴木慎一郎　山田晃智　竹内陽一郎　森良哉
株式会社　東芝　情報・通信システム技術研究所

1　はじめに

ＲＩＳＣにおけるコンパイラ最適化技術のひとつに命令スケジュールがある。命令スケジュールは命令レベルの並列性を向上させる最適化であり、その効果としてパイプラインストールが減少する。またスーパスカラやＶＬＩＷなどの並列ＲＩＳＣアーキテクチャではＣＰＩも減少する。

命令スケジュールの障害としてデータ依存関係と制御依存関係がある。データ依存関係は命令間の局所的な関係であり、命令スケジュール前の最適化フェーズにおけるレジスタリネーミング、式の変形、メモリ参照情報の解析などが依存関係の解消に必要である［1］。一方、基本ブロックをまたいだ命令スケジュールでは制御依存関係も障害となる。とくにロード命令や浮動小数点演算命令などの「副作用」をもつ命令は制御依存関係のために大域スケジュールされ難い。しかしこれらの命令の大域スケジュールは性能向上のために必要である場合が多く、ループに限ればソフトウェアパイプライニングなどがあるが、我々はもっと一般的なアプローチをとった。

本発表ではまず任意の命令の大域スケジュールを一般の制御フロー上において行うアルゴリズムを示す。そして条件実行制御アーキテクチャを採用したＶＬＩＷへの応用について述べる。

2　基本ブロックのスケジュール

基本ブロックは分岐命令を含まない命令列なので、基本的にはデータフローグラフを用いてスケジュールできる（ただし実際には構造ハザードの情報も必要である）。我々が大域スケジュールしようとする命令は基本ブロック内のクリティカルパスの端になる命令である。ここでは基本ブロックを解析してクリティカルパスの始端と終端を決めるなんらかの方法があると仮定する（図１）。

3　大域スケジュールのアルゴリズム

基本ブロックのクリティカルパスの終端とデータ依存関係にない命令が後続の基本ブロックにまたがって連続しているとき、その命令を同一のブロックに配置できれば並列性が向上する（図２）。この場合の大域スケジュールの可能性は次の制御フロー上のブール値連立方程式によって表される。

$$INSOUT(i,b)=PPPLOC(i,b)\times \sum_{a\in SUCC(b)} DELIN(i,a) + \prod_{a\in SUCC(b)} DELIN(i,a)$$

$$DELIN(i,b)=TOPLOC(i,b)\times \prod_{a\in PRED(b)} INSOUT(i,a)$$

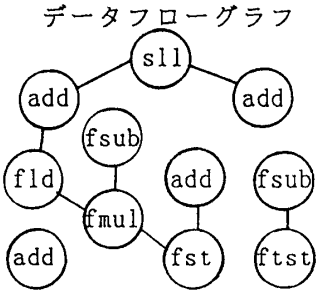
ここでPPPLOCとTOPLOCは局所性質であって、PPPLOC(i,b)は命令iが基本ブロックbのどのクリティカルパスの終端ともデータ依存関係にないことを表し、TOPLOC(i,b)は命令iが基本ブロックbのクリティカルパスの先端であることを表す。大域スケジュールを行うにはこの方程式でINSOUTとDELINについて解を求め、INSOUT(i,b)=TRUEのとき命令iを基本ブロックbに挿入してそれについてレジスタリネーミングを行い、DELIN(i,b)=TRUEのとき命令iを基本ブロックbから削除する。

4　解の縮小

この方程式の最大の解による大域スケジューリングではいくつか問題があるので解を縮小する必要がある。まず副作用のある命令の投機的な移動を考慮しなければならない。一般的にはPPPLOCのかわりにPPPLOC×ANTOUTを用いれば解が縮小されて投機的な移動は抑制される(ANTOUTについては［2］)。ちなみに整数演算などの副作用のない命令はレジスタリネーミングを行っているので投機的な移動を抑制する必要はない。

Global Instruction Scheduling Algorithm on Conditionally Executing VLIW,
K. Endo, R. Sakai, S. Suzuki, A. Yamada, Y. Takeuchi, R. Mori,
Information and Communications System Laboratory, Toshiba Corporation.

データフローグラフ

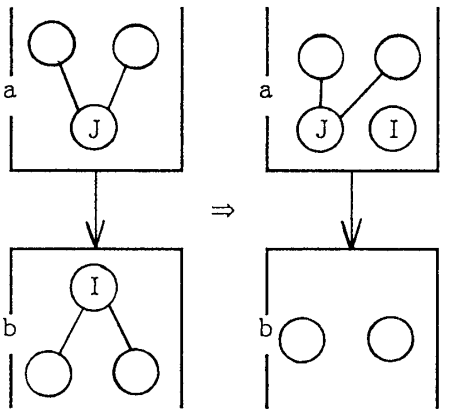


ＶＬＩＷ命令列

```
sll
add
fld   fsub
add   fsub          fmul
ftst  add    fst    add
```

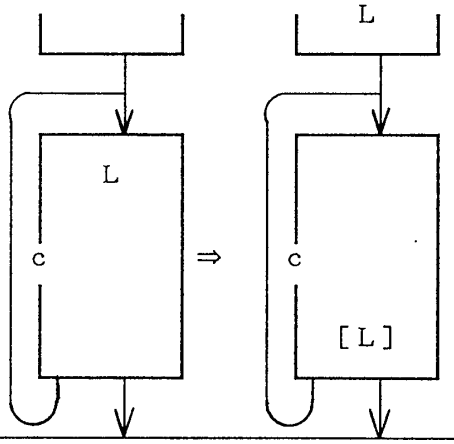
ＶＬＩＷ命令を”下向き”に詰め込んでスケジュールすれば、クリティカルパスの先端の命令が現れる。後端の命令を得るには逆に”上向き”にスケジュールする。

（図１）



クリティカルパスの後端の命令Ｊが、クリティカルパスの先端の命令Ｉとデータ依存関係にないとき、命令Ｉを基本ブロックａに移動すれば並列性が増す。ａの後続のブロックがｂだけでないときには、命令Ｉのデスティネーションレジスタをリネーミングする。

（図２）



基本ブロックｃへのロード命令Ｌの挿入は条件実行制御となる（［Ｌ］で表す）基本ブロックｃの分岐条件が早く決定される時に最も効果が大きいが、少なくともディレイドスロットには配置できる。どちらにしてもアーキテクチャでのサポートが必要である。

（図３）

もうひとつの問題は挿入した命令間での構造ハザードである。構造ハザードは順序に無関係な依存関係なのでスケジュールの障害にはならないが、並列性の低下を引き起こす。もともと命令挿入によって並列性を得ようとしているので、構造ハザードの影響がないように命令挿入を抑制して再度方程式を解く。すなわち、挿入された命令間に適当な優先順序（例えば分岐予測）を設けて基本ブロックをスケジュールし、構造ハザードのために「はみ出した」命令の挿入を抑制する。

5　条件実行制御アーキテクチャの活用

前述のように一般的には副作用のある命令の投機的な移動はできないが、条件実行制御アーキテクチャを利用すれば可能となる。条件実行制御アーキテクチャは条件フラグで命令の実行選択ができる機構であり［３］、ここでは分岐条件を指定するフラグで挿入される命令を実行選択する（図３）。

実際には方程式をPPPLOC×ANTOUTのかわりにPPPLOC×(ANTOUT＋CPPLOC)として解けばよい。ここでCPPLOCは条件フラグの決定がクリティカルパスの後端にないことを示す局所性質である。

6　まとめ

ここで示したアルゴリズムはループに対して適当なフロー変形を行えばソフトウェアパイプライニングの一般化になっている。またループでない場合にはディレイドスロット最適化の一般化にもなっている。条件実行アーキテクチャと組み合わせた場合には、ループ終了条件が単純でない場合のソフトウェアパイプライニングが可能となっている。

参考文献

[1]遠藤浩太郎、並列性向上を意識した大域最適化の方法、情処第48回全国大会5G-4
[2]E. Morel, C. Renvoise, Global Optimization by Suppression of Partial Redundancies, Comm. ACM22, 2(1979)
[3]遠藤浩太郎、多様なプログラムに適応するVLIWマシンの開発、第98回計算機アーキテクチャ研究会