ＶＬＩＷ計算機での手続き呼出最適化

4Ｔ－1

境 隆二，遠藤 浩太郎，鈴木 慎一郎
山田 晃智，森本 展行，森 良哉
（株）東芝　情報・通信システム技術研究所

はじめに

構造化プログラミング、オブジェクト指向などの技術が進歩するにつれて、プログラムのモジュール化が進んでいる。細かくモジュール分割されたプログラムを高速に実行するには、手続き呼出を高速に実行することが不可欠である。

多くのプロセッサのリンケージ規約では、手続きで書き換えるレジスタを手続きの入り口でセーブし、出口でリストアする。この処理を高速に行うためには、セーブ・リストアするレジスタの個数を最小限にとどめるようにすることと、セーブ・リストア処理を他の処理と並行して行えばよい。

我々の開発したＶＬＩＷ計算機、ＶＬ２０００シリーズでは、手続き呼出を高速に行うための最適化を行っている。本稿では、(1)使用レジスタ数の最小化、(2)手続き呼出処理の並列化、(3)レジスタセーブ・リストアの並列化について説明し、(3)の評価結果について報告する。

使用レジスタ数の最小化

パイプライン処理も含めて、命令レベル並列処理を行う計算機の最適化コンパイラでは、命令をスケジュールすることで、プログラムの並列性を引き出す最適化を行う。一般に、命令スケジュール前のレジスタ割当は、スケジュールの自由度を低下させるが、逆にスケジュールを優先して行うとレジスタの使用量は増大する。

ＶＬ２０００では、並列アーキテクチャを十分に活用するために、命令スケジュールを優先し、レジスタ割当はその後に行っている。スケジュールを優先するため、レジスタの生存範囲がオーバーラップして、必要なレジスタ数が増大するが、グラフ採色アルゴリズム[2],[3]を２段階で適用する事により、レジスタがスピルしない場合は最小限のレジスタ使用数となるように最適化している。スピルする場合はもともと、物理レジスタ使いきりの状態なので、スピルコストが最小になるように干渉グラフを採色する（図１）。実際、ほとんどのプログラムでスピルが発生しないので、使用レジスタ数を最小に抑えることが、手続き呼出を高速化する上で有効に働いている。

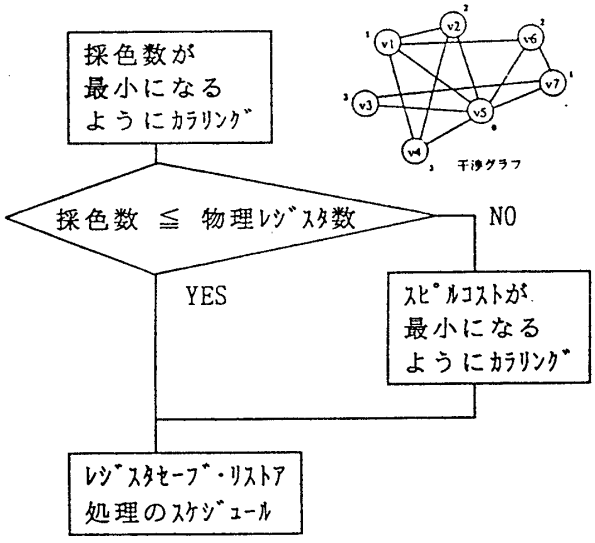

図１：２段階グラフ採色方法

手続き呼出処理の並列化

命令スケジュールをする際に、手続きを呼び出すジャンプ命令を飛び越えて命令をスケジュールすることは、手続き呼出の高速化のために重要である。

ＶＬ２０００では、手続き呼出をまたがった命令スケジュールを行うことで、呼出側のオーバー

Procedure call optimization on VLIW machine
Ryuji Sakai, Kotaro Endo, Shinichiro Suzuki
Akitomo Yamada, Nobuyuki Morimoto, Ryoya Mori
TOSHIBA Corporation

ヘッドをパラメータの設定と手続きへのジャンプ（これらは、他の処理と並列にスケジュールされる）のみに極小化している（図２）。

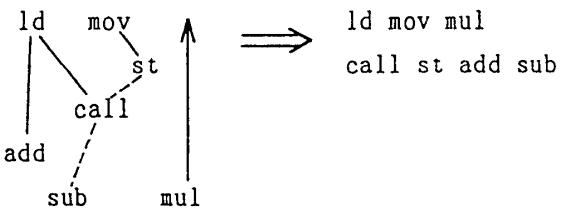

図２：手続き呼出の並列化

レジスタのセーブ・リストアの並列化

呼び出された側では、手続きで書き換えるレジスタを手続きの入り口でセーブし、出口でリストアする。これらの処理を命令レベルに展開すれば、他の処理と並列に実行可能である。この、展開された一連のレジスタのセーブ・リストア処理を手続きの入り口と出口の部分に並列にスケジュールすれば、手続き呼出を高速に行える。

ＶＬ２０００では、命令スケジュール後にレジスタ割当を行うので、物理レジスタの使用量がスケジュール後でないと確定しない。このため、レジスタ割当後、レジスタセーブ・リストア処理に対応する命令を既にスケジュールされている入り口と出口の命令列に並列実行出来るように挿入（スケジュール）する（図３）。

手続きの入り口

```
link                          link
push                          add sll push
push                          ld inc add push
push                          ld mul sub push
add sll
ld inc add
ld mul sub
```

手続きの出口

```
mov add                       pop mov add
add st                        pop add st
st                            pop st
pop                           ret
pop                           unlink
pop
ret
unlink
```

push:ﾚｼﾞｽﾀｾｰﾌﾞ命令
pop:ﾚｼﾞｽﾀﾘｽﾄｱ命令

図３：ﾚｼﾞｽﾀｾｰﾌﾞ・ﾘｽﾄｱの並列化

レジスタセーブ・リストア並列化の評価

ＶＬ２０００を使ってレジスタセーブ・リストア並列化の評価を行った（表１）。評価は、コンパイラでセーブ・リストアを並列にスケジュールしたものと、しなかったものを比較する、という方法で行った。今回の評価では、手続き呼出の頻度が多いものを対象に行った。プログラムのモジュール化が進んで、手続き呼出の頻度が高くなる一方で、手続きのインライン展開による最適化を考えてもインライン展開によってレジスタの使用量が増大したとき、レジスタのセーブ・リストア処理を並列化することは、プログラムの性能を向上させるために重要である。

評価プログラム	性能向上率
ドライストーン１．１	８．２％
ＧＣＣ	５．８％
ＶＬＩＷ最適化処理	４．３％

表１：ﾚｼﾞｽﾀｾｰﾌﾞ・ﾘｽﾄｱ並列化の効果

おわりに

今回の方式では、レジスタセーブ処理(push)は、手続きの先頭のブロックのみに並列スケジュールしている。このセーブ処理をより後方の基本ブロックで行うようにすることで、より並列度の高いコードが生成できる。しかしバックワードエッジで指されたブロックまでは後方移動できないため、より高い並列度を求めるなら、ハードウェアでの並行動作をサポートするなど対策が必要である。

参考文献

[1] K.Ebcioglu, R.D.Groves他 "VLIW Compilation Techniques in a Superscalar Enviroment" SIGPLAN vol.29, no.6, pp.36-48, Jun 1994.

[2] G.J.Chaitin, "Register allocation and spilling via coloring", SIGPLAN vol.17, no.6, pp.98-105, June, 1982.

[3] F.C.Chow and J.L.Hennessy "The priority-based coloring approach to register allocation", ACM Transaction on Programming Languages and Systems, vol.12, no.4, pp.501-536, Oct, 1990.