ソフトウェア・バイパス制御方式を利用したレジスタ割り当て

5N－3

新井正樹　安里彰　小沢年弘　木村康則
新情報富士通研

1　はじめに

ソフトウェア・バイパス制御方式はハードウェアを簡素化するために命令のソースオペランドの値をレジスタあるいは複数のバイパス出力のいずれから読むかをソフトウェアで指定するというプロセッサアーキテクチャである．本論文では，コンパイラによるソフトウェア・バイパス制御方式を利用したレジスタ割り当てについて述べる．

2　ソフトウェア・バイパス制御方式

ソフトウェア・バイパス制御方式をもつプロセッサでは，命令のソースオペランドの値をバイパスあるいはレジスタから読む方法を，パイプラインのタイミングを考慮してアセンブリコードで陽に指定する必要がある．ソフトウェア・バイパス制御方式を採用した VLIW プロセッサである Procyon[1] のアセンブリコードの例を図1に，その実行時のパイプラインの流れを図2に示す．図

```
(1): nop; add  %r2, %r0@R, %r1@R; nop; nop;
(2): nop; addi %r3, %r2@E, _A   ; nop; nop;
(3): nop; add  %r4, %r2@N, %r3@E; nop; nop;
```

図 1: Procyon のアセンブリコード例

1では最初のオペランドが出力レジスタを表す．整数系命令は D, E, N, W の 4 段のパイプラインをもち，E, N, W の各パイプラインステージで結果の値をバイパスに出力する．図1と図2で，VLIW 命令 (2) の%r2@E はレジスタ %r2 の値として VLIW 命令 (1) がバイパス E に出力した演算結果の値をデコードステージ D で使用することを表す．同様に，VLIW 命令 (3) の %r2@N はレジスタ %r2 の値として VLIW 命令 (1) がバイパス N に出力した演算結果の値を使用することを表す．VLIW 命令 (1) の %r0@R はレジスタ %r0 の値をバイパスではなく，レジスタから読むことを表す．

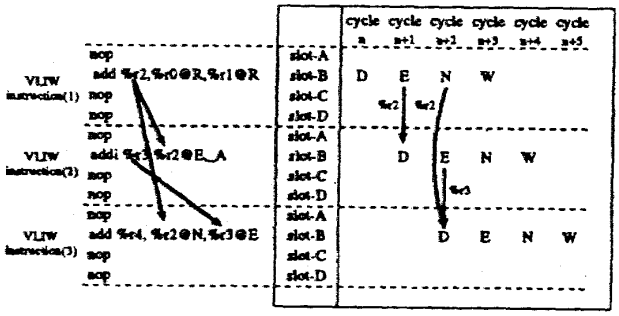

図 2: 実行時のパイプラインの流れ

3　ソフトウェア・バイパス制御方式を利用したレジスタ割り当て

ソフトウェア・バイパス制御方式をもつプロセサでは，仮想レジスタの生存区間の概念が一般のプロセサと異なる．ソフトウェア・バイパス制御方式を利用することでコンパイラのレジスタ割り当てパスは，以下の最適化 [2] を行なうことができる．

O1 仮想レジスタの生存区間の長さが演算パイプラインの長さ以下である場合には，その仮想レジスタに対してハードウェアレジスタを割り当てる必要がない．

O2 ふたつの仮想レジスタの生存区間が重なる範囲の長さが演算パイプラインの長さ以下である場合には，それらの仮想レジスタに対して同じハードウェアレジスタを割り当てることが可能である．

O3 VLIW 命令の空きスロットに仮想レジスタの生存区間を分割するためのレジスタ間 move 命令を生成し，仮想レジスタの生存区間を分割することによって，実行時の性能を低下させることなく O1 と O2 の最適化の機会を増やすことができる．

レジスタ彩色法 [3] に基づくソフトウェア・バイパス制御方式を利用したレジスタ割り当てのアルゴリズムの概要を以下に示す．

1. 各仮想レジスタの生存区間を求める．
2. 各仮想レジスタの生存区間の重なりを調べ，レジスタ干渉グラフを作成する．

Register Allocation Using Software Bypass Control
Masaki Arai, Akira Asato, Toshihiro Ozawa
and Yasunori Kimura
RWCP Multi-Processor Computing Fujitsu Laboratory
4-1-1, Kamikodanaka Nakahara-ku, Kawasaki 211, Japan

3. VLIW 命令の空きスロットにレジスタ間 move 命令を生成し，仮想レジスタの生存区間を分割することによって，**O1** と **O2** の最適化の機会を増やす (**O3**).

4. 生存区間の長さが演算パイプラインの長さ以下である仮想レジスタをレジスタ干渉グラフから削除する (**O1**).

5. 生存区間の重なる範囲の長さが演算パイプラインの長さ以下である仮想レジスタ間のエッジをレジスタ干渉グラフから削除する (**O2**).

6. レジスタ干渉グラフの各頂点にハードウェアレジスタを割り当てる.

一般のプロセッサでは，プログラムの任意の点でプロセッサ内に保持できる値の個数の最大値はハードウェアレジスタの個数に等しくなる．しかし，ソフトウェア・バイパス制御方式をもつプロセッサで，上記のレジスタ割り当てを行なった場合には，ハードウェアレジスタだけでなくバイパス上にも値を保持できるので，一般のプロセッサよりもスピルコードの量を減らすことができると考えられる.

4 評価

ソフトウェア・バイパス制御方式を利用したレジスタ割り当ての評価を行なった．SPECint92 の 4 つのプログラムに対して，レジスタ割り当て時に最適化 **O1** を適用した場合に，スピルされる仮想レジスタ数がどのように変化するかを調査した．評価用のマシンモデルとして，以下を仮定した.

- 4 並列 VLIW （スロット A, B, C, D）
  整数演算命令（スロット A, B, C, D）
  浮動小数点演算命令（スロット A, B, C, D）
  ロードストア命令（スロット C）
  分岐命令（スロット D）
- 整数演算パイプラインの長さ： 4 段
  レイテンシ 1
- 浮動小数点演算パイプラインの長さ： 4 段
  レイテンシ 2

整数レジスタと浮動小数点レジスタの数をそれぞれ 8 個ずつにした場合と 16 個ずつにした場合のレジスタ割り当ての結果を表 1 に示す．表 1 で，NORMAL は一般のレジスタ割り当てを行なった場合を，O1 は最適化 **O1** を行なった場合をそれぞれ表す．表 1 より，最適化 **O1** によって，スピルされる仮想レジスタ数を減少させることができるということがわかる．我々は，最適化 **O2** と **O3** を適用することによって，残りのスピルされた仮想レジスタの数をさらに減少させることができると考えている.

表 1: スピルされた仮想レジスタ数の変化

プログラム	使用可能なレジスタ数			
	8		16	
	NORMAL	O1	NORMAL	O1
008.espresso	1818	1598	224	210
022.li	137	113	15	15
023.eqntott	174	154	12	11
072.sc	512	467	127	123

5 おわりに

コンパイラによるソフトウェア・バイパス制御方式を利用したレジスタ割り当てについて述べた．今後はソフトウェア・バイパス制御方式を利用したレジスタ割り当てと広域命令スケジューリングの協調技法について検討して行く予定である.

評価に使用したコンパイラは、主要部を UtiLisp[4] で記述した．SunOS および Solaris 上で開発し、運用している.

参考文献

[1] 安里彰 他. ジオメトリプロセサ Procyon のアーキテクチャ，情報処理学会研究報告 97-ARC-126 (1997).

[2] 新井正樹 他. ジオメトリプロセサ Procyon の評価，情報処理学会研究報告 97-ARC-126 (1997).

[3] G. J. Chaintin. Register Allocation & Spilling via Graph Coloring, Proceedings of the ACM Symposium on Compiler Construction (1982).

[4] Wada Laboratory. UtiLisp Manual Revision 2.0, Department of Mathematical Engineering University of Tokyo (1988).