さまざまなプロセッサに対応する命令レベル並列コンパイラに関する研究

5L−1

大浦 勝宏　前田 敦司　曽和 将容
電気通信大学情報システム学研究科情報ネットワーク学専攻

1　はじめに

独立して動作可能な機能ユニットによって，複数の命令を並列に実行することができるスーパースカラ技術を適用したプロセッサが現在主流となってきている．同時に実行することの出来る命令の組み合わせは，それぞれの命令が使用する機能ユニットとプロセッサの内部構造によって決定される．プロセッサの性能を十分に引き出すためには，独立して実行可能な命令を適切に組み合わせる，命令スケジューリングが重要になる．

プロセッサの内部構造は，たとえ同じ命令セット・アーキテクチャを採用しているプロセッサでもその世代ごとに内部構造が異なってくる．それに伴い同時実行の制約も異なってくるので，したがってそれぞれのプロセッサごとにコード・スケジューラを作成する必要がある．本稿ではプロセッサごとに異なる同時実行の制約をパラメータ化して，命令スケジューリングを行なう方法の概要を報告する．

2　gcc の命令スケジューリング

GNU C Compiler (gcc) はいくつかの命令セット・アーキテクチャのための目的コードを生成することができる最適化コンパイラである．gcc の最適化処理は RTL (Register Transfer Language) と呼ばれるアーキテクチャ独立の中間表現の上で行なわれる[1]．コンパイラによって出力されるマシン命令はその命令がどのような動作を行なうかを記述する RTL 式で表現され，両者ほぼ1対1に対応する．RTL 式のパターンとマシン命令との対応はマシン記述 (Machine Description) ファイルで行なわれる．マシン記述ファイルはそのほかにそのプロセッサの持つ機能ユニットの種類とその数，命令の使用する機能ユニットの種類といったマシン固有の情報が含まれる．gcc は目的マシンごとにマシン記述を作成することで，それぞれの目的マシンに最適なコードを生成することを目指している．

gcc の命令スケジューリング・パスはプロセッサ内の機能ユニットの利用をシミュレートしながら，基本ブロック内で命令スケジューリングを行なう．データ依存グラフを末端から先頭方向に辿り優先順位を計算したあと，高い優先順位を持つ命令の組ごとにデータ依存の満足された命令を順次発行していく．同じ優先度にいくつかの命令が存在する場合，機能ユニットの競合によるストールなしに実行可能な命令を選択する．その優先順位の組の中にストールなしに発行可能な命令が存在しない場合は，次に低い優先度を持つ命令の組からストールなしに発行可能な命令を選択する．ストールなしに発行可能な命令が存在しない場合は，ストールなしで発行可能な命令が存在するまでクロックを進める (ストールさせる)．現在の gcc の実装では各命令の機能ユニットの使用は，発行されてから何サイクルその機能ユニットを使用するかで表現される．機能ユニットの競合は命令の発行時のみに調べられる．このような機能ユニット競合の表現方法ではパイプライン化された複数のユニットを持つ複雑な内部構造を持つプロセッサにおける競合を表現しきれない．そこで，各クロック・サイクルにおいて使用される機能ユニットの資源を単純な表形式で表現し，競合によっ

A Study of Instruction Level Parallelism Compiler Deal with Various Processors.
Katsuhiro Ooura, Atusi Maeda, Masahiro Sowa
The Graduate School of Information Systems, University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, 182-8585, Japan

て発生するストールを見積もる方法を考える．

3　プロセッサ資源の表現

パイプライン化された機能ユニットの各パイプライン・ステージは個別の資源として表現する．それぞれの機能ユニット毎に，各命令がクロックサイクル毎にどの資源を使用するかを表現する表 (資源予約表[2]) を用意する (表1)．表の各項目はその資源を利用する場合は 1，使用しない場合は 0 として表現している．

スケジューラは現在プロセッサ内で実行中の命令がその実行を終えるまで監視する．ストールなしに発行可能な命令を検索する際，まず現在実行中の命令が使用しているすべての資源を求める．次に選択した命令が使用する機能ユニットごとに，使用する資源が空いているか(競合しないか)を調べる．同じ機能を持つユニットが複数存在する場合はそれぞれについて調べ，空いているユニットを使用する．資源競合が発生した場合は機能ユニットが空くまでのクロックサイクル数を求め，その機能ユニットに関するコストとする．選択した命令が使用する全ての機能ユニットで競合が生じない場合，その命令はストールなしで発行できる．そうでない場合は使用する各機能ユニットのコストの最大値だけストールしなければ発行できない．

このような単純な形式の表を用いると，表の列数はそのユニットのパイプライン・ステージ数と命令の実行が完了するまでの最大クロック数の積になり，浮動小数点演算のように多くのクロックサイクル数を要する命令があると列数が増大してしまう．これを表2のような形式で表現し列数の増大を防ぐ．表はその資源が使用開始されるクロック時間と，開放されるクロック時間の2つの部分からなる．表の各項目はクロック・サイクル時間を示している．

表1：単純な資源予約表

命令	clock_0				clock_1				…	clock_t			
	Res1	Res2	・・・	Resn	Res1	Res2	・・・	Resn		Res1	Res2	・・・	Resn
Insn_class_1	1					1							
Insn_class_2	1				1								1
：													
Insn_class_x													

表2：縮小表現の資源予約表

命令	開始 clock				終了 clock			
	Res1	Res2	・・・	Resn	Res1	Res2	・・・	Resn
Insn_class_1	0	1		n−1	1	2		n
Insn_class_2	0	2		t−2	2	4		t
：								
Insn_class_x								

4　おわりに

現在，考案した資源競合のモデルを使用して命令スケジューリングを行なうスケジューラを gcc 内に組み込むための作業を進めている．考案した方法では元の gcc のスケジューラよりスケジューリング速度が低下してしまう．しかし，競合の検出は単純な比較であるということから，実用上問題を起こすようなコンパイル速度の著しい低下を招かないであろうと予測する．

5　参考文献

[1]Free Software Foundation Inc : GNU C Compiler Version 2.7.2.3 Manual,1997

[2]Vicki H. Allan, Reese B. Jones, Randall M. Lee, Stephen J. Allan : Software Pipelining, ACM Computing Surveys, Vol. 27, No 3 (September 1995)