B-022

GCC によるキュー・コンパイラ開発手法の提案
Suggestion about development technique of Queue Compiler with GCC.

石崎 賀昭 †　Pavel Boytchev †　吉永 努 †　曽和 将容 †

ISHIZAKI Yoshiaki　Pavel Boytchev　YOSHINAGA Tsutomu　SOWA Masahiro

1．はじめに

現在，当研究室は計算の中間結果を保存するレジスタに FIFO(First In First Out) のルールでアクセスするキュープロセッサの研究を行っている．多くのプロセッサは，ランダムアクセスでレジスタを利用するため，プロセッサの各命令が利用するレジスタを明示的に指定する必要がある．対してキュープロセッサではレジスタのアクセスを FIFO のルールでアクセスすることが決まっているため，各命令がレジスタを指定する必要がない．このため，各命令を表現するためのビットサイズを節約できることになり，プログラム全体のサイズ低減や，小さい命令サイズのまま，より多くの表現を可能とすることが期待できる．また，キュープロセッサでは構文木を幅優先で探索した命令列を利用することから，高い命令レベル並列性が期待できる [1,2,3]．本稿では，GCC(GNU Compiler Collection) を用いてキュープロセッサのひとつである QRP(Queue with Random acess register Processor) 向けのコンパイラを開発する手法について述べる．

2．キュー計算モデル

キュー計算モデルとは，命令の実行における計算の中間結果を FIFO のルールでアクセスするキューレジスタに格納するモデルをいう．図 1 を用いてキュー計算モデルにおける (a+b)*(c-d) の計算の流れを示す．まず load a でキューの末尾に a を書き込む．そして load b,load c,load d と続けてデータをキューの末尾に書き込む．次に add 命令では，キューの先頭から 2 個のデータ a と b を読み出して，その演算結果をキューの末尾に書き込む．sub,mul 命令も同様にキューの先頭から 2 個のデータを読み出して，演算結果を末尾に書き込む．すると，計算結果の (a+b) *(c-d) が得られる．キュープロセッサ用の命令列作成は，構文木を幅優先探索によって行うことにより生成される．生成された命令列は，前後の命令間のデータ依存関係が少なく，命令レベルでの高い並列性が期待できる．図 1 の場合，load a から load d の 4 命令は互いにデータ依存関係がない．また，add と sub にも互いにデータ依存関係がないため，これらの命令は並列実行が可能となる．

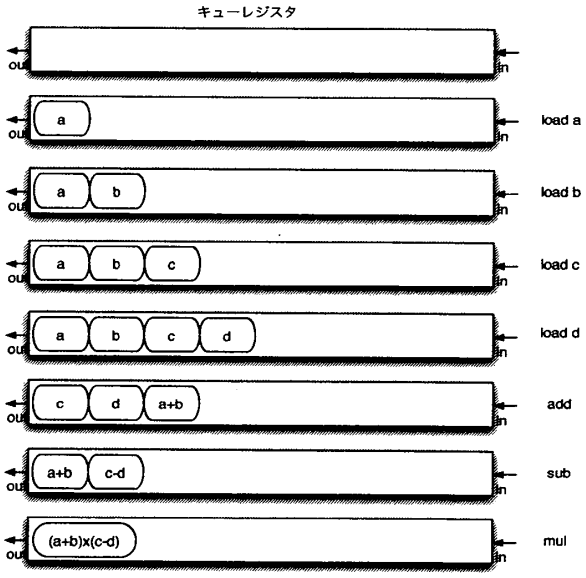

図 1：キュー計算モデルの流れ

3．GCC によるキュー・コンパイラ

GCC は一般的なランダムアクセスレジスタを装備したプロセッサ用のコンパイラ [4] であるため，キューレジスタを装備したキュープロセッサではそのまま利用することが出来ない．一般的なランダムアクセスレジスタを装備したプロセッサでは，計算のソースとなるレジスタとターゲットとなるレジスタを指定するアドレスを命令のオペランドとして与えることを前提としており，アドレスさえ与えれば，レジスタ全体の任意の位置からデータを扱うことが可能である．対して，キュープロセッサの場合は基本的には，データの挿入できる場所 (queue tail) とデータの取り出せる場所 (queue head) が決まっている．これは図 1 のとおりキューレジスタが FIFO のルールでアクセスされることに起因する．

GCC が一般的に対象としているプロセッサへポーティングを行う場合は，ターゲットマシンのサポートする命令やレジスタなどに関する定義とそのターゲットマシンに固有の最適化に関するマクロを作成することによって行う．この点においてはキュープロセッサへのポーティングにおいても同様な手法を取ることにしている．

† 電気通信大学 大学院情報システム学研究科

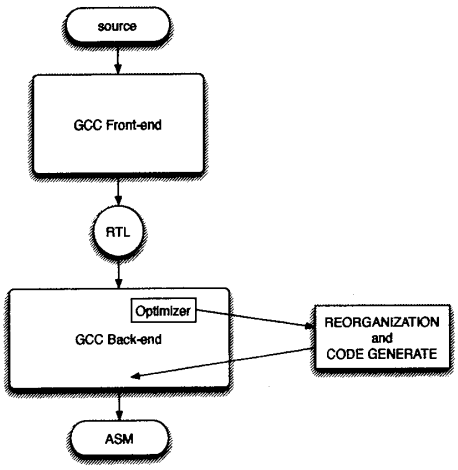

図２：GCC のブロック図

GCC にキューレジスタを一般的なランダムアクセスレジスタとして扱わせ，RTL を生成させるバックエンド部まで利用することにより GCC の資源を有効利用することを図る．このようにして生成された RTL から最終的にキューレジスタのアクセスルールに沿った命令列へ変更し出力することは，最適化部で一括して行うことにしている．これは GCC のバックエンド部が TARGET_MACHINE _DEPENDENT_REORG というマクロで指定される機種依存の最適化関数をコンパイル時に呼び出すので，その最適化関数の中でプログラムを構成する基本ブロックを単位にした，キュープロセッサ向けの命令列への組み替えと生成を同時に行う．（図２）これは同時に，キュー計算における最適化を基本ブロック内という比較的小さい範囲に限定して行われていることを示し，キュープロセッサ用のコードの出力に主眼をおいているためである．

キュープロセッサ向けの命令列への組み替えについて具体例を図３を用いて示す．まず，先に図１で示したプログラムについて，最初に出力される命令列は，load $A a, load $B b, add $C $A $B, load $D c, load $E d, sub $D $E, mul $G $C $F となる．この命令列はキュープロセッサ向けの命令列にはなっていないため，この命令列を末尾から先頭に向かってデータの依存関係を調べていき，依存の無い命令を先頭へ移動させていく．例をあげると，$G にセットする mul 命令は直前の $F にセットする sub 命令に依存するため，前方に移動させることができない．$E にセットする load 命令は前方に依存する命令が無いので一旦最上位まで移動し，その後キューレジスタの利用順位の高い $A,$B,$D が上位に配置される．このようにしてキューレジスタの並びとなった命令列から，最終的に不要なオペランドを削除して，アセンブラコードを出力するようにしている．

４．おわりに

開発している QRP-GCC を利用して，SPEC CPU 2000 から，176.GCC のソースファイル群をコンパイルしている．依存関係の都合など，複数のファイルが正常にコンパイルできていないが，コンパイルができるソースファイルを抽出し，同バージョンの

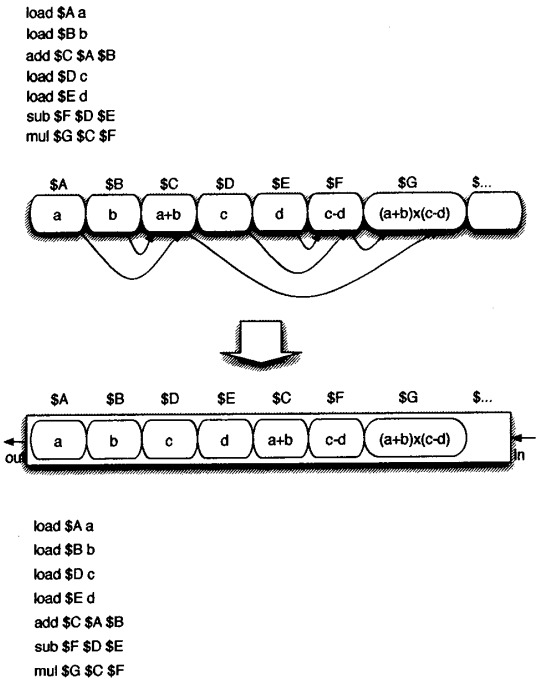

図３：命令列の組み替え

Power PC 向けの GCC を用いてコンパイルを行い，出力されたアセンブラコードファイル比較してデバッグを行っている．これは，一般的なランダムアクセスレジスタを使用しているプロセッサ向けのコンパイルと比較する評価も兼ねている．

本稿では，キュープロセッサ向けのコンパイラを一般的なコンパイラである GCC を用いて開発する手法について提案した．GCC のバックエンドでは原因の追及が困難なクラッシュが多くデバッグが難しいため、現在の方法ではバックエンドの全ては利用せず，キュー計算モデルにおける幅優先による命令列の生成と出力を，独自の最適化部で一括して行っている．今後は GCC のミドルエンド部にてキュー計算モデルの木を構築するようにして，GCC の最適化部やバックエンドを効率的に利用できるように改良していくことを考えている．

参考文献

[1] S. Okamoto, H. Suzuki, A. Maeda and M. Sowa: "Design of a Superscalar Processor Based on Queue Machine Computation Model " , IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.151-154 (1999).

[2] 中谷 陵，曽和将容，吉永 努，Ben A. Abderazek: " 生産型キュープロセッサ　PQPpf の基本設計に関する研究 " 第 67 回全国大会情報処理学会 (2005).

[3] 三好 崇之，Ben A. Abderazek, 繁田 聡一，吉永 努，曽和将容："Verilog-HDL による並列キュープロセッサのデザイン " FIT2004 (2004).

[4] Free software Foundation Inc. : "The internals of the GNU compilers" ,Free Software Foundation(2004).