2J−10　特定用途向きマイクロプロセッサ開発システム(2) − コンパイラ・ジェネレータの実現方法 −

博多 哲也†, 佐藤 淳†, 今井 正治†, 引地 信之††
†) 豊橋技術科学大学 情報工学系 ††)SRA

1 はじめに

筆者らはこれまで, ASICマイクロプロセッサ(ASIP : Application Specific Integrated Processor)開発システムについて研究を行なってきた[1],[2]. 本稿では, 本システムで応用プログラム開発環境の一部として用いられる, Cコンパイラ・ジェネレータの構成方法およびその試作結果について述べる.

ASIP開発システムによって生成されるCPUは, 特定の応用分野に適した命令セットを備えている. 本システムではGNU C Compiler (GCC)[3]に与えるターゲットマシン記述ファイルを自動生成することにより, ASIP用Cコンパイラの自動生成を行なう. GCCはターゲットマシンに依存する部分が少ないため様々なCPUに移植されている. 既存のCPUのためのCコンパイラを生成する場合, 従来は人間が人手でターゲットマシンに依存する部分を記述していた. 本方法では, アーキテクチャに関する情報を入力することにより, ターゲットマシン記述ファイルを自動合成し, ASIP用Cコンパイラを自動生成する.

2 ASIPの命令セット

本システムで生成されるASIPの命令セットは, GCCによるコード生成を容易にするために, GCCの中間言語RTL(Register Transfer Language)を基礎としている. ASIPの命令セットを, 次の3つのクラスに分けることにする.

(1) **Primitive Simple RTL (PRTL)**
PRTLは最小限のハードウエア(ALUおよび若干の演算器)で実現できる命令の集合であり, GCCのRTLのサブセットである. PRTLはASIPが必ず持つ命令の集合である. PRTLに含まれる命令を原始命令と呼ぶ.

(2) **Simple RTL (SRTL)**
SRTLはC言語の演算子に対応する命令の集合であり, GCCの中間言語RTLに対応する. SRTLに含まれる命令を基本命令と呼ぶ.

(3) **Extended RTL (XRTL)**
XRTLはライブラリ関数, ユーザ定義関数に対応し, これらがコプロセッサなどのハードウエアで実現された場合に定義される命令の集合である. XRTLに含まれる命令を拡張命令と呼ぶ.

表1にPRTLおよびSRTLに含まれる命令を示す.

3 GCCの概要

GCCは図1に示すように, 構文解析パス, いくつかの最適化パス, およびコード生成のパスで構成され, ほとんどがC言語で記述されている. GCCでは, ターゲットマシンに関する情報を"md"および"tm.h"と呼ばれる2つのファイルに記述することになっている. これらのファイルのうち"md"はRTLと実際に出力されるアセンブラコードとの対応関係を記述している. この命令のテンプレートの記述において, 命令の各オペランドが参照する対象(レジスタ, メモリ, 即値)を指定することができる. また, "tm.h"はターゲットマシンのアーキテクチャの仕様を記述するC言語のヘッダファイルである. "tm.h"に記述されるアーキテクチャの仕様には, メモリのパラメータ, レジスタの使い方, レジスタクラス, アドレッシングモード, スタックレイアウト, 関数のプロローグコードとエピローグコード, コンディションコードおよびアセンブラの構文が含まれる.

本システムでは"md"と"tm.h"を自動生成することによってASIP用のCコンパイラを実現する.

4 コンパイラ生成系

図1にコンパイラ生成系の構成を示す. コンパイラ生成系は"md"ファイル生成部および"tm.h"生成部からなる. 以下では各々のファイルの生成方法について述べる.

4.1 "md"ファイル生成部

"md"ファイル生成部に入力される情報は, 命令セットおよび各命令のそれぞれのオペランドの制約条件である.

4.1.1 命令の記述方法

ASIPの命令は対応するRTLに対し次のように記述する.

(1) 原始命令
原始命令は該当するRTLと1対1に対応しており, 全てを必ず記述する.

表1 PRTLおよびSRTLに含まれる命令

class	data	instruction
PRTL	integer	add sub neg and ior xor one_cmpl ashl ashr lshl lshr
SRTL	integer	mul umul div mod udiv umod cmp tst trunc extend zero_extend
	floating point	add sub neg mul div trunc extend float fix fixuns fix_trunc fixuns_trunc

---

Compiler Generator for the Application Specific Integrated Processor Design Environment
Tetsuya HAKATA†,Jun SATO†,Masaharu IMAI† and Nobuyuki HIKICHI††
†)Department of Information and Computer Sciences, Toyohashi University of Technology and ††)Software Research Associates

(2) 基本命令
基本命令も該当するRTLと1対1に対応している．ターゲットアーキテクチャが，ある基本命令を直接実行できる演算器を持たない場合，他の基本命令や原始命令を組み合わせることによって同等な動作を実現することにする．命令系列が数命令ですむ場合には，"md" ファイル中にこの命令系列を直接記述する．例えば，整数の比較演算および符号拡張命令がこれに該当する．しかし命令系列が長い場合や比較的複雑になる場合，"md" ファイル中に直接記述せずランタイムルーチンを呼び出す命令を記述することによって実現する．例えば，整数の乗算および除算命令，浮動小数点数演算命令がこれに該当する．

(3) 拡張命令
拡張命令が定義されていれば，その定義を記述する．拡張命令が定義されていない場合，あらかじめ用意しているライブラリ関数を呼び出す．

4.1.2 オペランドの制約条件の記述方法

"md" ファイル生成部への情報の中に，ある命令についてのオペランドの参照の対象が指定されていれば命令のテンプレートを記述する．ただし，転送命令と演算命令については，何も記述がない場合でも次の既定値を与える．

(1) 転送命令でのソース，デスティネーションの組合せの制約
メモリからレジスタへ，レジスタからメモリへ，レジスタからレジスタへおよび即値をレジスタへ転送の4種類である．

(2) 演算命令での制約
演算命令のオペランドはレジスタのみを対象とする．

4.2 "tm.h" ファイル生成部

"tm.h" 生成部は第3節で述べたターゲットマシンの仕様を与えることによって"tm.h" ファイルを自動生成する．また，仕様が与えられない場合でも用意している既定値の設定を記述することで対応する．

4.2.1 レジスタに関する記述

ASIPは汎用レジスタ方式を採用している．また，浮動小数点数は専用のレジスタを用いて処理を行なう．レジスタについて特に指定がなければ，汎用レジスタおよび専用レジスタの個数をそれぞれ与えればよい．このとき専用レジスタの個数は0個でもよい．汎用レジスタはインデックスレジスタとして割り当てることもできるので，スタックポインタ，フレームポインタおよび関数の引数へのポインタは汎用レジスタに割り当てる．

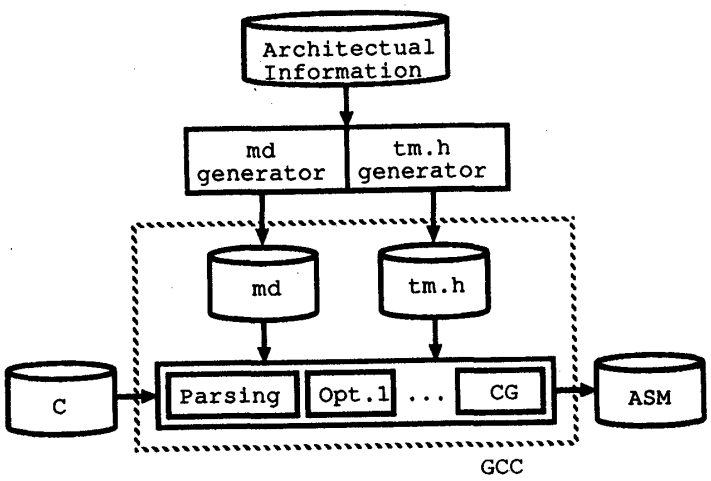

図1 コンパイラ生成系の構成

4.3 コンパイラ生成系の実現

本システムをSUN Workstation上で実現した．図2にコンパイラ・ジェネレータによって生成されたCコンパイラを用いて，応用プログラムをコンパイルした結果の一部を示す．

5 おわりに

本稿では，特定用途向きマイクロプロセッサ開発システムの一部であるコンパイラ・ジェネレータの構成方法およびこれを実現した結果について述べた．現段階において本システムは拡張命令に対応していない．また，ランタイムルーチンの記述を行なわなければならない．さらに，命令セットの変更およびアドレッシングモードの変更に対応できるアセンブラを開発する必要がある．これらは今後の課題である．

謝辞 本プロジェクトに御協力賜わるメンターグラフィックス・ジャパン(株)，富士通VLSI(株)，および討論して頂いた豊橋技術科学大学VLSI設計研究室の諸兄に深謝します．

参考文献

[1] 佐藤 淳, 福田孝一, 市田真琴, 今井正治 :"特定用途向きCPUコアの命令セットの実現方法に関する一考察," 信学技法,VLD 89-110,1990.

[2] 佐藤 淳, 博多哲也,Alaudain Y. Alomary, 今井正治, 引地信之:"特定用途向きマイクロプロセッサ開発システム (1)," 情報処理学会第42回全国大会,1991.

[3] Richard M. Stallman: "Using and Porting GNU CC,"1990.

```
#NO_APP
gcc_compiled.:
.text
        .align 1
.globl _qsort
_qsort:
        .word 0xc0
        subsi sp,$8,sp
        cmpsi 4(ap),8(ap)
        bge L1
        addsi 4(ap), 8(ap), r0
        movsi r0, r0
        tstsi r0
        bge L2
        addsi r0, $1, r0
L2:
        asrsi r0, $1, r0
        movsi _work[r0], -4(fp)
        movsi 4(ap), r6
        movsi 8(ap), r7
L3:
        cmpsi r6,r7
        bgt L4
L5:
        cmpsi _work[r6],-4(fp)
        bge L6
        addsi r6, $1, r6
        jmp L5
L6:
```

図2 生成したASIP用Cコンパイラの出力の一部