リターゲット型コンパイラ向き中間語 ArmCode の開発

7G－2

森木紀恵　渡辺坦　増石哲也
（株）日立製作所システム開発研究所

1　はじめに

近年、計算機アーキテクチャの進化の速度が速くなってきている。コンピュータの仕様が決まってからコンパイラが必要となるまでの期間が短いので、新機種に対してコンパイラを迅速に開発できる技術が望まれている。

この問題を解決するコンパイラのリターゲット技術として、ソースプログラムを機種非依存な中間語に変換した後、対象機種の機械語プログラムを生成する方式がある。[1][2]

このようなコンパイラの性能や他機種への適用容易性は、中間語の仕様に大きく左右される。

本稿では、機種別開発工数削減とオブジェクト性能のバランスを追及するリターゲット型コンパイラのために開発した、機種非依存な中間語について述べる。なお、本稿では、この中間語をArmCode [1]とよぶ。

2　ArmCode の設計方針

ArmCode を用いたリターゲット型コンパイラの構造を図1に示す。リターゲット時の開発工数とは、機械語翻訳系の開発工数のことである。

ArmCode の目的は、機械語翻訳系の開発工数が小さく、かつオブジェクト性能のよいリターゲット型コンパイラを実現することである。

一般に、機種非依存中間語を用いたコンパイラは、特定機種向けのものより性能が劣るといわれている。特定機種向けコンパイラが行なっているコンパイル上流フェーズでの機種依存な最適化ができないからである。

しかし、機種依存と考えられている最適化アルゴリズムの多くは機種間で共通である。異なるのはレジスタの個数や特性、アドレッシングモード、語長、命令の種類

Intermediate Language ArmCode Specification for Retargetable Compilers
Kie MORIKI, Tan WATANABE, Tetsuya MASUISHI
Systems Development Laboratory, Hitachi, Ltd.

[1] Abstract Regester Machine Code

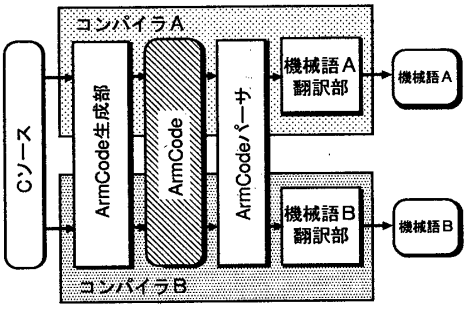

図 1: ArmCode コンパイラの構造

などである。例えば、大域的レジスタ割り付けでは、抽象的なレジスタを導入するとフロー解析や割り付け優先度の算定、使用するレジスタ数の最小化などを機種非依存に行なうことができる。ピープホール最適化についても、分岐最適化等、いくつかは機種に依存せずに処理が可能である。

前記目的のためには、機械語翻訳系が負担する処理を少なくし、アルゴリズムの簡素化と規模の縮小をはかると同時に、ArmCode 生成系で、大半の最適化を行ないオブジェクト性能の劣化を抑えればよい。

具体的には、以下の観点から ArmCode を設計した。

(1) 機械語への翻訳が容易であること
(2) データ長や語境界の違いを吸収できること
(3) レジスタ割り付けや最適化の大半を行なうことができること

3　ArmCode の特徴

3.1　RISC 型命令体系

機械語翻訳系をコンパクトにするため、ArmCode は構文木のようなプログラミング言語に近い形ではなく、機械語レベルの言語にする。

最近の開発動向から考えると、現在および近い将来出現する計算機はRISCマシンが大半である。また、RISCマシンのロードストアアーキテクチャに基づく単

```
<構造体サイズ式>::=
<メンバサイズ> {& <メンバサイズ> }
<共用体サイズ式>::=
        <メンバサイズ> { | <メンバサイズ> }
<メンバサイズ>::= <型名> * <要素数>
<構造体語境界式>::= <型名>
<共用体語境界式>::= <型名> | <型名>
<型名>::= char | short | int | long
        | float | double | quad | ptr
        | <構造体型名> | <共用体型名>
```

図 2: 構造体・共用体計算式

純な命令は、明示的にメモリアクセスがあり、レジスタ割り付けやオブジェクト最適化を機種非依存化しやすい。以上のことから、ArmCode はRISC 型の命令体系を採用することにした。

3.2 スケーラブルな命令記述

データ長や語境界は対象マシンによって異なる。ArmCode コンパイラはこれらのパラメータの値を全て機械語翻訳系で解釈するため、ArmCode ではデータ長や語境界の値を変数として表現する。

3.2.1 命令属性の型指定

各命令が対象とするデータの大きさを命令属性と呼ぶ。命令属性は、バイト数ではなく基本データ型名を用いて指定する。

例) Add.int %gi1,%gi2

3.2.2 構造体／共用体の記法

構造体型や共用体型のサイズ、語境界、メンバオフセットは、その構成メンバのサイズ、語境界、並びに依存する。構造体や共用体の型定義命中にその型のデータサイズおよび語境界のための計算式（図2）を記述する。

3.3 抽象レジスタ

ArmCode の抽象レジスタは、用途別のクラスに分類される。最近の計算機では、レジスタの用途分類はかなり類型化されている。この類型を考慮して抽象レジスタクラスを設定した。ArmCode レジスタクラスを表1に示す。

表 1: ArmCode レジスタクラス

レジスタクラス	用途
%gi	Callee-Save 整数汎用レジスタ
%ti	Caller-Save 整数汎用レジスタ
%ri	整数リターン値用レジスタ
%fpi	整数仮引数用レジスタ
%api	整数実引数用レジスタ
%gf	Callee-Save 浮動小数点数汎用レジスタ
%tf	Caller-Save 浮動小数点数汎用レジスタ
%rf	浮動小数点数リターン値用レジスタ
%fpf	浮動小数点数仮引数用レジスタ
%apf	浮動小数点数実引数用レジスタ
%sp	スタックポインタ
%ra	リターンアドレス

4 おわりに

機種別開発工数削減とオブジェクト性能のバランスを追及するリターゲット型コンパイラのために、機種非依存な中間語 ArmCode を開発した。

詳しくは別報 [3] に述べるが、ArmCode によるリターゲット型コンパイラを RISC、CISC 各 1 機種について試作した。

その結果、RISC マシンについては、少ないリターゲット工数で、特定機種向けコンパイラに比べ数％程度のオブジェクト性能劣化に抑えられることがわかった。CISC マシンでは、オブジェクト性能をあげるためには RISC マシンの場合よりリターゲット工数が大きくなる。

参考文献

[1] OSF: ANDF,Application portability and open systems,(Jun.1991)

[2] Fraser, Hanson: A Retargetable Compiler for ANSI C, ACM SIGPLAN, 26-10, pp.29-43 (Oct. 1991)

[3] 真下ほか: 機種非依存中間語 ArmCode を用いたリターゲット型コンパイラの開発と評価, 情処学会第 48 回全大論文集 (Mar. 1994)