7G－3 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

真下祐一* 渡辺坦* 増石哲也* 森木紀恵* 村田恵輔* 中嶋啓人**
* （株）日立製作所システム開発研究所 ** （株）日立製作所ソフトウェア開発本部

1 はじめに

計算機アーキテクチャの進化が速くなっているので、対象機種を容易に変えられるリターゲット型で、オブジェクト性能の良いコンパイラを開発できる技術の確立が望まれている。

これに応えるため、別報 [1] で述べる機種非依存の RISC 型疑似マシン語 Armcode [1]を中間語とするリターゲット型 C コンパイラ (ArmCode コンパイラ) を開発した。

本稿では、ArmCode コンパイラの概要と、RISC マシンと CISC マシンに対する試作結果の評価について述べる。

2 ArmCode コンパイラ概要

アセンブリ言語に似た低レベルな機種非依存の中間語 ArmCode を用いて、コンパイラの機種依存部を ArmCode 生成部より下流に局所化している。その機種依存部内でも更に、機種非依存モジュールを分離することで、機種依存モジュールを縮小している。これにより、リターゲット性の高いコンパイラを実現できる。

2.1 構成

ArmCode コンパイラの構成は対象機種によって多少異なるが、基本的な構造 (図 1) は共通である。

(1) ArmCode 生成部 (P1) … 機種非依存

(a) 構文解析木生成モジュール

(b) ArmCode 生成モジュール

(c) ArmCode 最適化モジュール

(d) ArmCode ファイル生成モジュール

(2) マシンコード生成部 (P2)

(a) ArmCode パーサ … 機種非依存

ArmCode ファイルを読み込み、メモリ上に展開する。

(b) パタン照合 … 機種半依存

ArmCode のパタンに対する機械語パタンの対応づけを行なう。機械語パタンは機種依存であるが、パタン照合モジュールは非依存である。

(c) 機械語翻訳系 … 機種依存

メモリ上の ArmCode を対象機種の機械語に変換し、アセンブラファイルを生成する。構成は機種によって多少異なるが、処理内容は類似している。

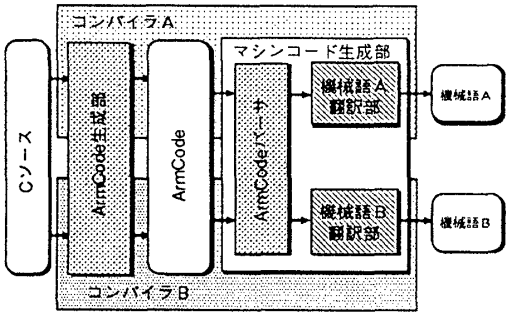

図 1: ArmCode コンパイラの構造

2.2 特徴

コンパイラが持つ処理の内、ArmCode コンパイラで特徴のあるレジスタ割り付けと最適化の概要について以

Development and Evaluation of Retargetable Compiler based on ArmCode
Yuichi MASHITA, Tan WATANABE, Tetsuya MASUISHI, Kie MORIKI, Keisuke MURATA, Hiroto NAKAJIMA
* Systems Development Laboratory, Hitachi, Ltd.
** Software Development Center, Hitachi, Ltd.
[1] Abstract Regester Machine Code

下で述べる。

(1) レジスタ割り付け

P1 では、大域的な、すなわちライブレンジ単位での抽象レジスタの割り付けと最少化を行う。割り付ける抽象レジスタ数はまだ限定しない。また P1 では、P2 のレジスタ割り付けに必要な解析情報を生成する。

P2 では、抽象レジスタを対象機種の実レジスタに変換し、実レジスタが不足する場合にはメモリへのスピルコードを生成する。例外処理として、対象機種に依存する役割の固定された実レジスタに対する局所的なレジスタ割り付けも行なう。

(2) 最適化

P1 では、大域的最適化と分岐最適化、ArmCode レベルでのピープホール最適化を機種非依存に行なう。

P2 では、機械語に変換した後に発生するピープホール最適化等を行なう。

コンパイラをこのように分割することで、従来機種依存と見做されていたレジスタ割り付けや、最適化の多くの部分を機種間で共通に行ない、性能の良いオブジェクトを生成できるようにした。

3 ArmCode コンパイラの評価

上記の ArmCode コンパイラを、RISC アーキテクチャである SPARC と、CISC アーキテクチャである i8086 を対象として試作した。

リターゲット時に、対象機種毎に開発する機械語翻訳系の規模を表 1 に示す。RISC アーキテクチャに対しては、容易にリターゲットできる可能性を得た。

表 1: 機械語翻訳系開発規模

対象機種	ステップ数 [ks]	うち共通化可能部
i8086	15.1	3.0
SPARC	6.6	2.8

dhrystone ベンチマークで測定したコンパイラの性能を各対象機種について、表 2 と表 3 で示す。表 3 の ArmCode 改良の項は、試作結果に対し、比較的容易に実現できる改良を実行したと想定した場合の性能を示す。オブジェクトサイズは、特定機種向けコンパイラに比べ両対象機種で約 16 ～ 22% の劣化になった。実行性能の面では、RISC アーキテクチャを対象とした方が性能劣化が押えられる見通しを得た。

表 2: i8086 専用コンパイラとの比率 (dhrystone)

コンパイラ	dhrystone 値比	オブジェクトサイズ比
ArmCode	0.87	1.16

i8086 専用コンパイラを 1 とする

表 3: SPARC 専用コンパイラとの比率 (dhrystone)

コンパイラ	dhrystone 値比	オブジェクトサイズ比
ArmCode	0.93	1.22
ArmCode 改良	0.98	1.04

SPARC 専用コンパイラを 1 とする

4 おわりに

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、少い工数でリターゲットできた。オブジェクト性能に関しては、試作した結果を検討し、幾つかの最適化処理の追加、および ArmCode 仕様を改良することで、特定機種向けコンパイラに遜色ない結果 (表 3) が得られる。CISC マシンに関しては、試作段階のリターゲット工数でさえ RISC の約 2.5 倍かかる。これに、オブジェクト性能を特定機種向けコンパイラと同等にするための処理を追加すると、更に多くのリターゲット工数が必要となる。

従って、機種非依存中間語 ArmCode は、CISC マシンには少し無理があるが、RISC マシンを対象としたリターゲット型コンパイラに対しては実用できると思われる。

参考文献

[1] 森木ほか: リターゲット型コンパイラ向き中間語 ArmCode の開発, 情処学会第 48 回全大論文集 (Mar. 1994)