コンパイラの最適化性能テストのための リターゲッタブルアセンブリコード比較

Retargetable Assembly Comparator for Performance Testing of Compiler Optimization

村上 大喜 Daiki Murakami　北浦 幸太 Kota Kitaura　石浦 菜岐佐 Nagisa Ishiura

関西学院大学 理工学部　School of Science and Technology, Kwansei Gakuin University

1 はじめに

コンパイラには高い最適化の性能が求められるため, コンパイラが意図通りの最適化を行っているかは重要なテスト項目になる. 文献 [1] では, 異なる 2 つのコンパイラが生成するアセンブリコードの比較によって一方の最適化不足を検出している. このコード比較ツールは LLVM のバックエンドのテスト [2] 等にも利用できるが, 特定のアーキテクチャ (x86) にしか対応していない. 本稿ではこれを一般化し, 他のアーキテクチャでもコード比較ができる手法を提案する.

2 アセンブリコードの差分判定法

文献 [1] のアセンブリコード比較法は, 2 つのコード中の不一致命令抽出とその重み和の比較に基づいている. まず, 2 つのアセンブリコードで k 命令以上 (k は 7 程度) 連続して一致している部分を削除して, 命令列が一致しない区間を抽出する. この区間の中で, 順序の入れ換えや同値な命令 (演算は同じだが, オペランドのデータ長やアドレッシングモードが異なるもの) の同一化を行った上で対応がとれる命令を削除する. 残った命令に対して実行時間を考慮して定義した重みの和を求め, その比が閾値以下であればコードに差があると判定する.

3 アーキテクチャ情報の記述

本稿では, アーキテクチャ依存の処理を分離することにより, 文献 [1] の比較法を複数のアーキテクチャに対応させる. 本稿では, GNU アセンブラ (GAS) のアセンブリ記法を前提とする. このため命令のニモニックはアーキテクチャ独立に抽出できる. アーキテクチャに依存するのは, 命令の同値関係, および同値類に対する重みである. 本手法ではこれらをデータとして分離し, 比較処理自体はアーキテクチャ独立にする.

MIPS 用のアーキテクチャ記述の例を図 1 に示す. 2~6 行目は命令の同値関係を定義するリスト (各要素は, 同値な命令の集合を表す正規表現とその集合のラベルの対), 10 行目は各命令集合に属する命令のコスト (ただし, コストが 1 のものは省略する) である.

4 実験結果

提案手法を Perl5 で実装した. C コンパイラのランダムテストシステム Orange4 [3] を使用して, x86, ARM, MIPS, PowerPC 用の GCC-8.0.1 と GCC-7.3.0 の最適化性能を比較する実験を行った. 結果を表 1 に示す. “#test” は総テスト数, “#diff” はコードの差分を検出したテスト数である. いずれのターゲットでも生成コードの差異を検出できた. 図 2 は MIPS 用 GCC でコードの差分を検出したテストプログラムの一例であり, GCC-8.0.1 の最適化性能が 7.3.0 よりも劣っていることを検出している.

```
01:  my @equiv_list = (
02:      [".*mul.*", "mul"],
03:      [".*div.*", "div"],
04:      ["b.*", "b"],
05:      [".*add.*", "add"],
06:      ...
07:  );
08:
09:  weight => {
10:      mul => 5, div => 30, b => 10,
11:  }
```

図 1　アーキテクチャ情報の記述例 (MIPS)

表 1　実験結果 (GCC-8.0.1, GCC-7.3.0)

target	time [h]	#test	#diff
x86_64	2.93	2,000	132
ARM	4.21	2,000	121
MIPS	2.98	2,000	104
PowerPC	3.20	2,000	169

(Intel Core i5-6200U CPU@2.30GHz, Ubuntu 18.04 LTS)

test181.c

```
1:  int main()
2:  {
3:    volatile int x1 = 1;
4:    volatile char x2 = 1;
5:    int c1 = (char)(x2*(1-(1/x1)>=1))%1;
6:    return 0;
7:  }
```

MIPS-1.s (GCC-7.3.0)

```
1:  sw $2,12($sp)
2:  sb $2,11($sp)
3:  lw $2,12($sp)
4:
5:
6:
7:
8:
9:  move $2, $0
```

MIPS-2.s (GCC-8.0.1)

```
1:  sw $2,12($sp)
2:  sb $2,11($sp)
3:  lw $2,12($sp)
4:  li $3,-1
5:  div $0,$3,$2
6:  teq $2,$0,7
7:  mflo $2
8:  beq $2,$3,.L5
9:  move $2,$0
```

図 2　コードの差分を検出したテストプログラム

5 むすび

本稿では, アセンブリコード比較を複数ターゲットに適応させる手法を提案した. アーキテクチャ設定ファイルの記述を洗練することが今後の課題である.

参考文献

[1] 北浦, 石浦 : “アセンブリコードおよび実行時間の比較に基づく C コンパイラの最適化のランダムテスト,” DA シンポジウム 2017, pp. 39–44 (Aug. 2017).

[2] 田中, 石浦, 西村, 福井 : “LLVM バックエンドの最適化性能テストのミュータント生成,” 信学技報, VLD2017–88 (Jan. 2018).

[3] K. Nakamura and N. Ishiura: “Random testing of C compilers based on test program generation by equivalence transformation,” in *Proc. APCCAS 2016*, pp. 676–679 (Oct. 2016).