大域的同値関係解析によるコード最適化

4D－3

田中 裕久　小谷 謙介　佐山 旬子　田中 旭　湯川 博司
松下電器産業(株)　マイコン開発センター

1．はじめに

近年、組み込みマイコンにおいても、開発効率が高く、移植性の高い C 言語を利用したソフト開発が主流となっている。しかし C 言語による開発では、オブジェクトコードサイズの増大とそれに伴う実行速度の低下が問題となる。そこで、生成コードサイズの縮小化、実行速度の向上を実現する高効率コードを生成する C コンパイラが必須となっている。

今回、大域的な同値関係を解析し、その結果を利用して効率のよい最適化を行なう手法を開発したので報告する。

2．局所的同値関係解析

最初に同値関係について説明する。本稿で述べる資源とは、レジスタ、メモリなどのハードウェア資源と即値を総称したものとする。同値関係とは、ある資源とある資源が同じ値を保持しているときの資源間の関係を表す。例えば、レジスタ R とメモリ M が同じ値を保持しているとき、レジスタ R とメモリ M は同値関係にあると呼ぶ。またレジスタ R が即値 1 を保持しているとき、レジスタ R と即値 1 は同値関係にあると呼ぶ。同値関係は、転送命令によって生成される。一方、命令により資源の値が変化した時、その資源に関する同値関係が削除される。

以上のようにして生成される同値関係を、従来ではプログラムの基本ブロック内のみについて解析を行っていた。しかし、基本ブロック内の局所的な解析では、生成される情報量が少なく、効率的に最適化を行うことは困難であった。

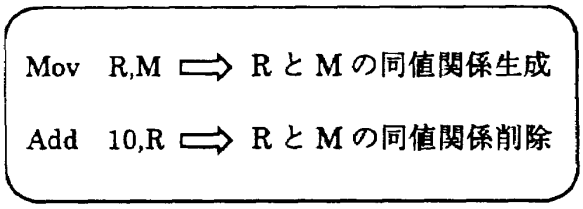


図 1

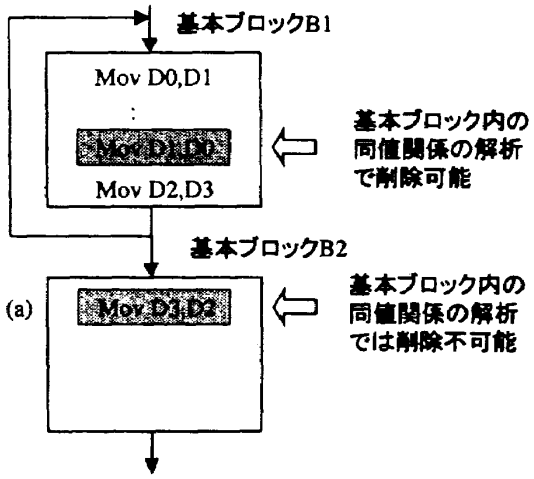


図 2

3．大域的同値関係解析

資源 R と同値関係にある資源の集合を同値関係集合と呼ぶ。また各資源に対応する同値関係集合をひとまとまりにしたものを同値集合と呼ぶ。更に基本ブロックの入口、および出口における同値集合をそれぞれ In 集合、Out 集合と呼ぶことにする。

まず、プログラムの制御の流れである制御フロー解析を行なう。次に各基本ブロックの In 集合に初期値を与え、それぞれの基本ブロックの Out 集合を計算する。これらの In 集合、Out 集合をそれぞれの基本ブロックの初期値として、以下のデータフロー方程式を解く。

ここで、r はプログラム中のある資源、B は同値関係を求める基本ブロック、P は基本ブロック B の

The Code Optimizing Method by Analyzing Global Equivalent Relations

Hirohisa Tanaka, Kensuke Odani, Junko Sayama, Akira Tanaka, Hiroshi Yukawa
Matsushita Electric Industrial Co.,Ltd.

$$EsIn[r,B] = \bigcap_{P \in pred(B)} EsOut[r,P]$$

$$EsOut[r,B] = EsGen(EsIn[r,B])$$

先行基本ブロックを表し、EsIn[r,B]は基本ブロック B における資源 r の In 集合、EsOut[r,B]は基本ブロック B における資源 r の Out 集合を表す。
全ての基本ブロックの In 集合、Out 集合の変化がなくなるまで繰り返し計算を行なう。変化がなくなったとき、各基本ブロックの入口および出口における大域的な同値関係である In 集合、Out 集合が求まる。大域的同値関係の解析の概念図を図 3 に示す。

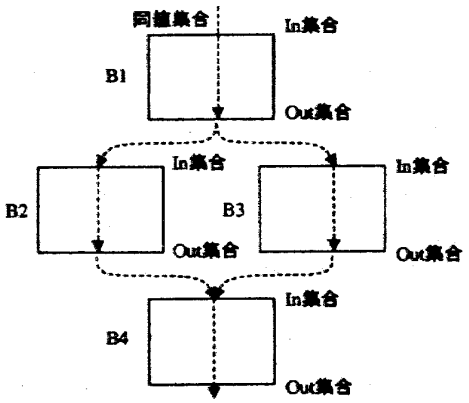


図 3

4．大域的同値関係を利用した最適化

前述の手法により解析した大域的同値関係を利用して、以下の 3 種類の最適化を施した。

- 同値資源間の転送命令の削除
- オペランド置換
- コピー伝播

以下では同値資源間の転送命令の削除の最適化を例にして説明する。

図 2 において、大域的な同値関係の解析を行うと、結果は以下のように計算される。

B1: In={…}
　Out={D0={D1}, D1={D0}, D2={D3}, D3={D2} …}
B2: In={D0={D1}, D1={D0}, D2={D3}, D3={D2} …}
　Out={…}

基本ブロックの入口の同値集合より、レジスタ D2 と D3 が同値関係にあることがわかる。従って図 2 の(a)の Mov 命令はなくてもよいことがわかり、削除することができる。

5．評価結果

一般的なベンチマークプログラムに本手法によるコード最適化を適用し、コードサイズ縮小率および実行サイクル数の改善率を調べた。その結果を図 4 および図 5 に示す。

最適化前のコードサイズ、実行サイクル数を 100 とした時の、局所的および大域的同値関係解析を行なった場合のコードサイズの割合をそれぞれ示している。

測定の結果、大域的同値関係解析による最適化では、局所的同値関係解析の場合に比べてコードサイズで最大約 2%、実行サイクル数で最大約 2.5% 改善されていることが示された。

図 4 サイズ比較

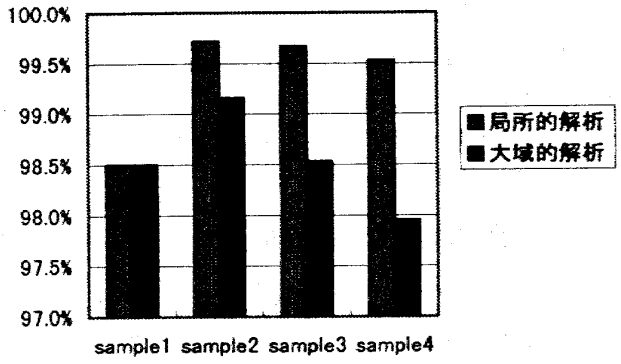
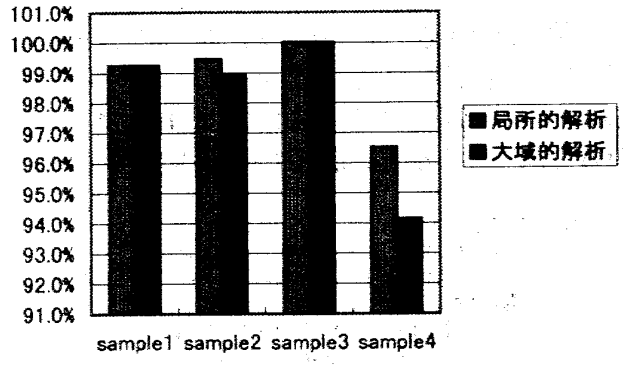


図 5 実行サイクル数比較



6．まとめ

大域的な同値関係の解析結果を利用した、効率的なコード最適化の方式を示した。今後は、資源の別名解析を行うことによって、同値関係解析の精度を向上させる予定である。