5H-04　関数間レジスタプロモーションによるメモリアクセスの削減

服部 直也，　飯塚 大介，　坂井 修一，　田中 英彦
{hato, iizuka, sakai, tanaka}@mtl.t.u-tokyo.ac.jp
東京大学大学院 工学系研究科 *

1　はじめに

近年メモリの動作速度とプロセッサの動作速度は拡大を続け、メモリアクセス命令は他と比べて低速な命令になっている。また ALU などのユニットと異なり、ロード／ストアユニットは並列化が容易ではないため、今後もメモリアクセス命令がプログラム実行のボトルネックとなることが予想される。

この状況を受けて、メモリアクセス命令の数を減らす最適化であるレジスタプロモーションが提案されている。レジスタプロモーションではポインタ解析の結果を利用し、グローバル変数などのメモリ変数を一時的にレジスタ変数に昇進させることで、メモリアクセス命令を削減する。本稿では、これまで関数内で行われてきたレジスタプロモーションを、関数間で行う手法を提案する。

2　レジスタプロモーション

これまでのレジスタプロモーションの手法は大きく 2 種に分類され、(1) ループベースのプロモーション (2)Partial Redundancy Elimination を基本としたプロモーションが存在する。

前者 [1] は関数内に含まれるループのネスト構造を認識し、移動可能と判断したメモリアクセスをループの外側へ移動する (図 1)。ループ内に alias かどうかが分からないメモリアクセスや、関数呼び出しがある場合はその前後に load/store を挿入して、コードの正当性を保証する (図 2)。

後者 [2] は Partial Redundancy Elimination をメモリアクセスに対して行うもので、ループ構造がない場合でも冗長な load/store を削除する。

3　関数間レジスタプロモーションの提案

これまでの研究では、プロモーションの範囲を関数内に限定していたが、関数間でプロモーションを行えば Call By Reference をレジスタを用いて行うことが可能になる (図 3, 4, 5)。また、グローバル変数を用いた関数間通信にもレジスタを用いることができる。そこで、本稿では関数間レジスタプロモーションを提案する。本手法

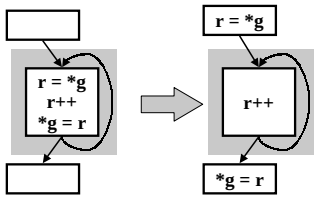

図 1: 単純なループのプロモーション
(g は Global 変数のアドレス)

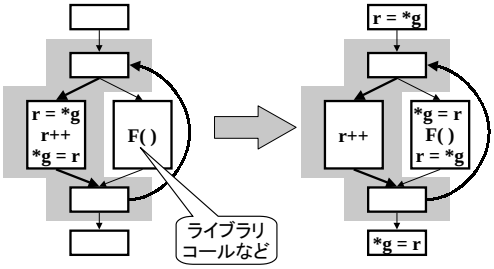

図 2: 解析不能領域を含むプロモーション

(1) ポインタ解析 (2) コード変形 (3) プロモーション区間最適化　の 3 つの処理からなる。以下にそれぞれについて説明する。

3.1　context sensitive なポインタ解析

メモリアクセスをレジスタアクセスに変換できる区間 (プロモーション区間) を調べるために、context sensitive なポインタ解析を行う。命令がアクセスするアドレスがわかれば、プロモーションが可能なため、最大プロモーション区間の終端は、ライブラリコール等の解析不能な関数呼び出しや、依存関係が曖昧な store である。(本稿中の図は、網かけ部分がプロモーション区間を示している)

3.2　コード変形

区間内のメモリアクセスはレジスタアクセスに変換する。ただし最適化前と同じ結果を保証するため、区間の入口に load、区間の出口に store を挿入する。(区間がメモリ変数の scope 終端を含む場合には、store は不要である。) また、区間内であっても依存関係が曖昧な load の前には store が必要になる。

プロモーション区間が関数呼び出しを跨いでいる場合

---

* "Memory Access Elimination by Interprocedural Register Promotion"
Naoya Hattori, Daisuke Iizuka, Shuichi Sakai, Hidehiko Tanaka
University of Tokyo, Graduate School of Engineering,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

は、最新の変数内容を関数間で伝えるために引数／返値を追加する必要がある。ただし、 callee 側以下で プロモーション対象の load がない場合は引数を追加する必要はなく、 callee 以下で対象 store がない場合は返値を追加する必要はない。図 3 → 4, 4 → 5 ではそれぞれ callee が load/store を含んでいるので、引数／返値を追加している (不要になった引数の削除も行っている)。

3.3 プロモーション区間の最適化

3.1, 3.2 までの処理でプロモーションを行うことは可能であるが、問題が生じることがある。最大区間でプロモーションを行うと、別のループの中にアクセス命令を配置して性能が低下してしまう可能性がある (図 6 左)。また、無意味に関数の引数／返値を増加させてしまう可能性もある。そのため、区間の終端を含むループや関数ではその領域をプロモーション区間に含むべきかを、プロモーション前後のアクセス数変化から判断する (図 6 右が最適な区間である)。

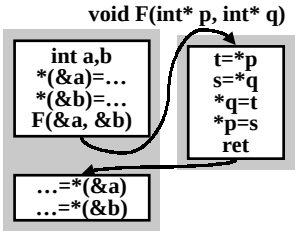

図 3: step0:
プロモーション前

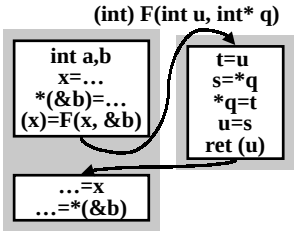

図 4: step1:
p を promotion した結果

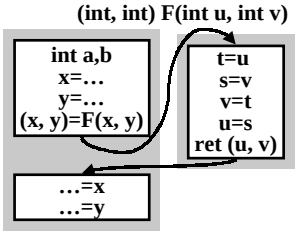

図 5: step2:
q を promotion した結果

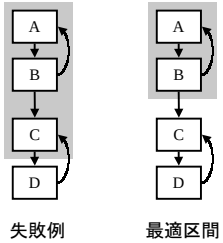

図 6: 区間最適化

4 予備評価

本稿執筆時点で、上記アルゴリズム実装は完了していない。しかし予備評価として、最大区間でプロモーションを行った場合のアクセス数の変化を調べた。

実装は富士通研究所の C コンパイラ [3] の RIC 中間コードに対して行っている。この中間コードはレジスタ無限大モデルを採用しており、引数／返値の数も制限がない。最適化前のコードでは、アドレスを取られている変数とグローバル変数全てがメモリに置かれている。このコードに対し、関数内、関数間それぞれのプロモーションを適用した。

	load の実行回数	store の実行回数
promotion 前	11,059,791,308	4,445,175,523
関数内 promotion 後の増減	-2,578,523,330	-13,145,816
関数間 promotion 後の増減	-3,204,519,361	-14,100

表 1: compress95 のプロモーション結果

入力プログラムとしては SPECint95 の compress95 を使用した。結果を表 1 に示す。

compress95 は グローバル変数を用いた計算が多いため、 load 数に関しては関数間プロモーションの効果が現れている。一方、 store の数が増えてしまったのは、最大区間でプロモーションを行っており、別ループ内にアクセスを追加してしまったためである。

5 まとめと今後の課題

本稿では、メモリアクセスを削減する最適化であるレジスタプロモーションを紹介し、これを関数間に拡張する手法について述べた。また、この手法を用いたアクセス削減量を見積もるため、 compress95 に関して予備評価を行った。

今後は、アルゴリズムの未実装部分を早急に実装し、提案手法による正確なアクセス数削減を評価することが必要である。また、現段階ではレジスタは無限に使えるという仮定を置いているが、物理レジスタ数に制限を加えた時の評価も行う必要があると考えている。

謝辞

本研究を進めるにあたり、富士通研究所の小沢年弘氏、木村康則氏にはコンパイラ及び定期的な御助言を頂きました。深く感謝致します。

参考文献

[1] A.V.S. Sastry and Roy D.C. Ju. A New Algorithm for Scalar Register Promotion Based on SSA From. *PLDI*, pp. 15–25, 1998.

[2] Raymond Lo 他. A New Algorithm for Scalar Register Promotion Based on SSA From. *PLDI*, pp. 26–37, 1998.

[3] 飯塚 大介 他. C コンパイラにおけるループ最適化の検討. *HPC 77*, pp. 65–70, 1999.