4M−5　生存区間分割によるレジスタ割り付け

阿部　仁　守屋康正
富士ゼロックス(株)　システム技術研究所

1. はじめに

最適化コンパイラのグローバル・レジスタ割り付けの手法としては、Chaitinら[1]やF.Chowら[2]によるグラフ彩色法を応用した手法がこれまで提案されている。しかしこれらの手法では、長い生存区間を持つ変数の実行頻度の変化が割り付け結果に反映されずに、実行効率の低下を招いた。

本稿では、予め変数の生存区間を細かく分割し、同一分割生存区間内の小片同士で優先順位をヒューリスティックに計算し、その結果に従って割り付ける手法を提案し、評価の結果に考察を加えた。

2. アルゴリズム

Rを使用可能なレジスタの集合として、次のステップを2.1~2.3の順に実行する.

2.1 生存区間解析

定義参照のデータの流れ方程式を計算し生存区間を計算する[3].その結果から
V: ソースプログラム中の生存区間
L: 生存区間の開始点の集合
D: 生存区間の終了点の集合
とする

2.2. 分割および割り付け

L∪Dで定まる「区間」の集合Sを計算する

2.2.1. 優先順位計算の省略期間

区間x($x\in V\cup S$)の開始点を$L(x)$、終了点を$D(x)$とすると、
全てのl($l\in S$)について
$L(v) \leqq_t L(l)$
$D(l) \leqq_t D(v)$
(但し、関係$\leqq_t$は左辺が右辺より遅く出現しない事を表す)
を満足する生存区間v($v\in V$)の集合N_l($N_l \subset V$)を計算し、$|N_l| \leqq |R|$のものについてN_lを割当て候補の集合C_lに加える。

2.2.2 優先順位付けおよび割り付け

$|N_l| > |R|$となるl($l\in S$)について、e($e\in N_l$)に対して優先順位を計算する。
優先順位はChow[2]の手法を参考にして

優先順位 = ロード利益 × 期間内の参照回数
　　+ ストア利益 × 期間内の代入回数
　　− 移動費用 × 移動必要回数

と決定した。eを優先順位の大きい順に$|R|$個C_lに加える。ここで移動必要回数は{0, 1, 2}であり、次の分割生存区間の候補が無ければ+1の値をとる。

2.3. 割り当て

各v($v\in V$)毎に、全てのl($l\in S$)について、$P_v = \sum_l \mathrm{isexist}(C_l, v)$を計算する。
(但しisexist(C_l, v)は$v\in C_l$の時1、$v\notin C_l$の時0を値とする関数である)
全てのv($v\in V$)について$|P_v|$を計算し、大

"Register Allocation Technique based on prior splitting of variables' live range."
Hitoshi Abe (habe@rst.fujixerox.co.jp), Yasumasa Moriya
Fuji Xerox Co., Ltd. System Technology Research Labs.

きい順にレジスタに割り当てる。
直観的には$|P_v|$は生存区間vについて、割当て可能な分割区間の数を示す。

3. 実験方法

Sun-4/110上で検証実験を行った。
最適化を指定せずにGNU Cコンパイラでコンパイルして得られるレジスタ割り付けが行われていないアセンブラ出力を、中間コードとみなし、本アルゴリズムにてレジスタ割り付けを行うレジスタアロケータを試作した。
その結果、レジスタ割り付けされたオブジェクトコードとレジスタ割り付けされていないオブジェクトコード、およびグラフ彩色法[1]を用いて手計算によりレジスタ割り付けしたオブジェクトコードのパフォーマンスとを測定・比較した。

4. 評価結果

簡単な算術演算を行う約30行のテストプログラムと、ドライストーンベンチマークを行なった結果を示す。
テストプログラムでの結果はSPARCシミュレータで得られたマシンサイクル数を示す。またシミュレータの制約上テストプログラムにはSystemコールを含まない。
この結果9.6%の高速化が達成された。
またドライストーンでは約18%の実行速度の向上が得られた。

表1テストプログラムによるベンチマーク

最適化なし	本手法	グラフ彩色
167	151	151

(単位:マシンサイクル)

表2 ドライストーンの結果

最適化なし	本手法
12483	14723

(単位: 回/sec)

5. 考察

結果として、特にグラフ彩色法と比較して同等のパフォーマンスを得る見込みがある事がわかった。特に生存区間を長くする傾向がある命令スケジューラをプリパスで併用した場合、生存区間全体の長さに依存しない本手法は高い効果が期待できる点を考慮に入れると、有効な手法であると判断する
今後は、試作したレジスタアロケータを改良し、より高性能を目指すとともに多くのベンチマークテストを行い、指針を得たいと考える。
最終的にはコンパイラに組み込み、その効果を検証していきたい。

6. 参考文献

[1]G.J. Chaitin,"Register Allocation and Spilling via Graph Coloring", *Proceedings of the ACM SIGPLAN 82 Symposium on Compiler Constructiton*, pp. 98-105.
[2]F.Chow and J.Hennesy, "Register Allocation by Priority-based Coloring", *Proceedings of the SIGPLAN 84 Symposium on Compiler Construction*, pp.222-232
[3]A.V. Aho, R. Sethi, and J.D. Ullman, "Compilers Principles, Technique, and Tools", Addison-Wesley, 1986