ISS-A-037

浮動小数点を導入したスタック計算機の末尾呼び出し最適化とコンパイルの可能性について

氏家　雄司†　　塩谷　勇††
† 法政大学理工学部創生科学科　　†† 法政大学理工学研究科

1. はじめに

本報告では,末尾呼び出し最適化(Tail Call Optimization, TCO)の命令 TCO を持つスタック計算機の末尾呼び出し最適化について述べ,従来整数のみ扱うことが出来たスタック計算機への浮動小数点の導入を報告する.そして,Java 言語風の手続き型言語のコンパイルの可能性について報告する. プログラムを利用者が記述する際に繰り返し処理よりも再帰が有効である.しかし,再帰を使う際には末尾呼び出しにおけるメモリのオーバーフローを考えなければならない.本研究では,それを意識することなくプログラムを記述できる範囲について検討してきた.本報告では関数の値呼び出し(call by value)のみを考える.末尾呼び出し最適化は,return 文で関数を呼び出す際,新たにスタックフレームをプッシュダウンすることなく,現在のスタックフレームを上書きして再使用する.

```
function g(x,y){ return x+y;};
function f(x,y){ return g(x+y+1,y);};
x=f(2,3);
```

上のようなプログラムの際,図 1 のように引数の値は(2,3)から(6,3)へと上書きされる.

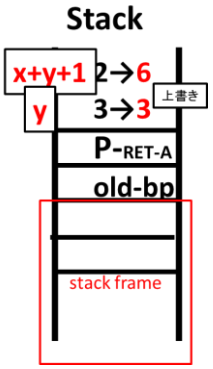

図 1.TCO の仕組みの例

古いスタックフレームは削除され,現在利用している変数の記憶域が削除されるため次の対策が考えられる.

(1) 仮引数の記憶域を利用した手法(2) 引数を追加するプログラム変換の手法(3) CPS (Continuation Passing Style) の手法(4) gcc の実装のように関数を展開する手法(5) 動的にメモリを確保して,不要になれば GC で回収

本報告では,TCO 命令を持つスタック計算機の(1),(2)について検討を行い,それらのコンパイルを行う.

2. スタック計算機 STM

スタック計算機[1]は計算機の構造とコンパイラの学習のために利用されている TCO 命令を持つシステムである.スタック計算機の TCO 命令と関数を呼ぶ命令 CALL を比較する.

CALL n :次の命令の番地をスタック S[sp]に sp←sp+1, S[sp]←bp, sp←sp+1, bp←sp, n 番地の P-CODE へ
TCO n : sp←bp, n 番地の P-CODE へ

sp はスタックポインタ,bp はベースポインタ,n は P-CODE の番地で,単に n 番地に jump するのみである.sp←bp から,スタックフレームが上書きされることがわかる.関数の戻り番地や前のスタックフレームへのリンクもそのままでよい.そして,union を用いて,その値が整数か小数か flag をつけて判断する.

3. コンパイラの実装

実装は,TCO が可能なパタンを用意し,条件を満たせば最適化を行うものである.本手法は,式,代入文,四則演算に限定し,(a)変数が仮引数か局所変数,(b)代入文の左辺か右辺,(c)return 文の式のパタンに分類してスケルトンを構成する.図 2 のようにコンパイルを行う.

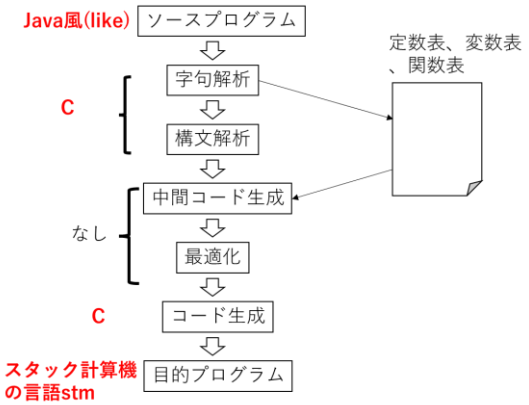

図 2.コンパイラの流れ

参考文献

[1] Isamu Shioya, Stack Machine with Tail Call Optimization, http://shio2.k.hosei.ac.jp/
[2] 氏家雄司,電気通信情報学会ポスター発表 https://www.ieice.org/~iss/jpn/Publications/issposter_2022/data/pdf/ISS-A-021.pdf