3D−1

マルチスレッド・分散処理のための Lispカーネルの設計と実装

関口 知紀　山本 強
北海道大学

1 はじめに

Lispはプロトタイピングに向いており、ユーザーがLispシステムの中で作業をする点でOSと類似する環境であり、OSや計算機環境のプロトタイピングに有用である。現在それらの土台になるマルチスレッド処理、分散処理を支援・実現するLispカーネルを開発中であるが、本稿ではそのLispカーネルの設計と、その実装に用いるLispコンパイラについて述べる。

2 Lispカーネルの設計

本研究におけるLispカーネルの設計指針は以下である。

- Lisp上でのマルチスレッド環境の提供
- 将来、Common Lispを取り込む

以下にカーネルの設計概要について述べる。

2.1 マルチスレッド機構

スレッドの生成、切替、終了、排他制御、CPUの明け渡し等の基本的なプリミティブとOSのタイマ機構を用いることにより、プリエンプティブスケジュールを実現する。

さらに、コンテクストスイッチのきっかけである入出力について、入出力のブロックに伴うスケジューリングを実現するようストリームへのアクセス手続きを拡張している。ストリームがネットワーク、端末に接続している場合有用であると考えられる。

The design and implementation of a Lisp kernel for multi-thread systems and distributed processing.
T. Sekiguchi, T.Yamamoto
Hokkaido University

2.2 オブジェクトの設計

Lispではオブジェクトに型がついているが、型は32bitの最上位の表現形式の下位3ビットのタグと、オブジェクトに附属するヘッダによるハイブリッド表現としている。Common Lispのデータ型をすべて取り込み設計した。

2.3 実行時の構成

プログラムスタック、静的環境用のデータスタック、動的環境用のスペシャルスタックの3つのスタックにより実行時環境を構成する。スレッドのコンテキストスイッチを高速化するためデータスタック領域を複数用意することを考えている。

ガーベージコレクションはstop-and-copyで行なう。

2.4 Lispカーネルの記述

Lispカーネルは扱うオブジェクトの種類・機能等により階層化されるが、各層の記述法により移植性、拡張性、保守の難易等が決まる。Lispオブジェクトを扱う部分はLispで記述するのが自然であり、拡張性・保守の点で有利であると考えられる。そこで、カーネルの大部分をLispにより記述し、カーネルコンパイラよりコンパイルすることでカーネルの実現をはかっている。

3 Lispカーネルコンパイラ

3.1 コンパイラの構成

本コンパイラは、字句・構文解析、中間コード生成、制御フロー解析、データフロー解析、コード生成部からなり、ターゲットのアセンブラを生成する。字句・構文解析系はコンパイラコン

```
(defreg sp "r31")
(definstruction call (addr)
  (("subu r31, r31, 4")
   ("bsr.n ~S" addr)
   ("st r1, sp, r0")))
(def-vmachine ld-immediate-bitlen 16)
(def-vmachine have-jump-ez t)
```

図 1: ターゲットマシンの記述例 (MC88100)

パイラにより実現した。構文解析系は let, block, tagbody, setq 等のスペシャルフォームを認識し中間コードを生成する。

3.2 最適化

構文解析系により生成された中間コードに対して制御フロー解析ではコードの並べ換え、到達しないコードの削除、データフロー解析では ud-chain、du-chain、生変数解析を行い、不要な変数・代入の削除、定数値の利用等のターゲットによらない最適化を実施している。

3.3 関数呼出のコンパイル

関数呼出ではレジスタによる引数渡しをするコードを生成し、car、cdr 等の基本的な Lisp オブジェクトに対するアクセス関数や、スタックフレームへのアクセスなどは、コンパイラがこれらを認識しインラインでアセンブラにコンパイルしている。

3.4 スペシャルフォームの記述について

本コンパイラは、Lisp のスペシャルフォームを定義・コンパイルできるよう defentry、call-entry を認識し、新たなフレームをつくらない関数呼出を実現している。これによりスペシャルフォームの記述が容易になる。

3.5 ターゲットマシンの記述

本コンパイラが多くのターゲット用のコードを生成できれば、Lisp カーネルも多くのターゲットで実現されることになる。本コンパイラでは RISC 指向の仮想アーキテクチャを設定し、ターゲットと仮想マシンとの関係をコンパイラ本体とは別の記述ファイルに記述し [1]、コンパイラ実行時にロードすることで複数のターゲットに対応している。

この記述ファイルには以下が記述される。

1. 仮想命令と実命令の対応
2. 仮想レジスタと実レジスタの対応
3. 実命令の制約と特徴

3では、実命令のオペランドに即値を渡す場合の即値の長さの制限や、比較命令がコンディションコードを変更するかなどの情報を記述する。この記述ファイルによりターゲット間のアーキテクチャの違いを吸収する。現在は SPARC、MIPS 系、MC88100 について記述できるように設計されている。

また、記述ファイルは後に Lisp 上でコンパイラを実現する場合にも容易に利用できるよう Lisp 形式とした (図 1)。Lisp 上でこれらの記述をマクロ展開すれば簡単に利用できると考えられる。

4 おわりに

今後はカーネルの実装を進め、また、分散処理を支援するために必要なプリミティブについて検討しなければならないと考えている。

参考文献

[1] R.R. Kessler and J.C. Peterson and H. Carr and G.P. Duggan and J.Knell and J.J. Krohnfeldt, "EPIC – a retargetable, highly optimizing Lisp compiler," *ACM SIGPLAN*, Vol.21, No.7, June, 1986, pp.118-130

[2] A.V. Aho, R.Sethi, J.D. Ullman, "Compilers principles, techniques, and tools," Addison-Wesley, 1986