5 E − 7

ISO規格ISLISP処理系の実装方式

各務 寛之* 山田 雅彦* 高橋 順一* 五味 弘* 新谷 義弘** 長坂 篤** 梅村 恭司*** 湯浅 太一****

*(株) 沖テクノシステムズラボラトリ ** 沖電気工業 (株)
*** 豊橋技術科学大学 **** 京都大学

1 はじめに

ISLISP は，1997 年に制定された ISO の LISP 言語規格であり，大規模な COMMON LISP に対して，言語仕様の簡潔さと処理系の実行効率の高さを目標として設計された [1], [2].

我々は，Windows95/NT および UNIX 上で動作する高速で，かつ移植性の高い ISLISP 処理系の開発を行っている [3]. 本稿では高速化のために採用した LISP オブジェクトの構造とメモリ管理方式を中心に，この ISLISP 処理系の実装方式について述べる.

2 本処理系の構成

本処理系の構成を図 1 に示す. インタプリタは，ISLISP のソースプログラムを読み込み，LISP オブジェクトに変換し，それを評価する.

コンパイラは，ソースプログラムをバイトコードプログラムに変換する. 変換されたバイトコードプログラムはバイトコードインタプリタによって解釈実行される.

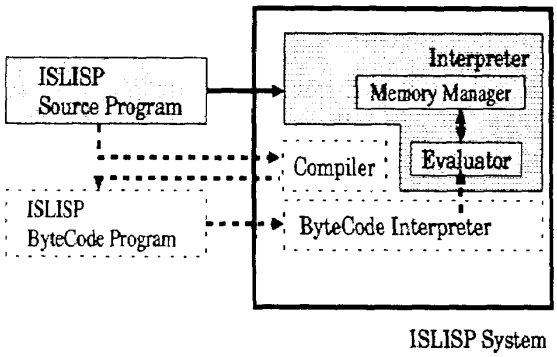

図 1: 本処理系の構成

"An implementation of ISO-standardized ISLISP system"
Hiroyuki KAGAMI*, Masahiko YAMADA*,
Junichi TAKAHASHI*, Hiroshi GOMI*,
Yoshihiro SHINTANI**, Atsushi NAGASAKA**,
Kyoji UMEMURA***, Taiichi YUASA****

*Oki Technosystems Laboratory, Inc.
**Oki Electric Industry Co., Ltd.
***Toyohashi University of Technology
****Kyoto University

3 LISP オブジェクトの表現

LISP オブジェクトは 32 ビット長のデータであり，アドレス部とタグ部からなる. タグは 8 ビットまたは 3 ビットであり，下位に割り当てられる. また，最下位ビットは GC 用に使用する (図 2).

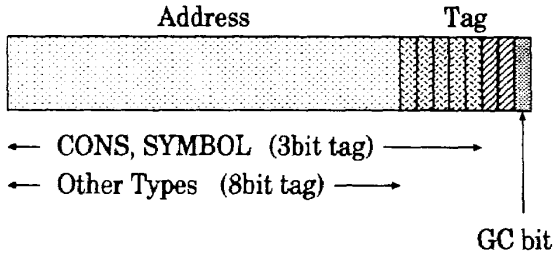

図 2: LISP オブジェクト

タグ部を下位に割り当てることにより，高速なメモリアクセスが可能となる (詳細は後述する).

広く使用されている Pentium プロセッサでは，引数がスタック渡しである. 8 ビットタグを採用することによって，スタックに積まれている LISP オブジェクトからタグ部が直接取得することができる. これにより処理を高速に行うことができる.

- CONS, SYMBOL
  これらのデータは，8 バイト境界に配置したメモリ領域に格納している. このため，アドレスの下位 3 ビットは必ず 0 になる.
  これらのタグは，3 ビット (CONS は 000, SYMBOL は 100) になっているので，LISP オブジェクト全体が実体への直接のアドレスとなり，高速にアクセスすることができる.

- 即値 (固定長整数，文字)
  固定長整数は 24 ビット整数であり，アドレス部にデータの値そのものを格納する.
  本処理系では，日本語を使用することを考慮し，すべての文字を 2 バイト (16 ビット) として扱う.

- 他のタイプ
  上記以外のタイプのデータは，アドレス部に，HEADER メモリのオフセット値を格納する.

4　メモリ管理

LISP メモリが不足したときは，ガーベッジコレクション (GC) により，LISP メモリの再利用を行い，GC を行っても不足する場合は，LISP メモリの拡張を行う．

4.1　メモリブロック

メモリブロックは，まとまった大きさで確保された LISP メモリと，その先頭に付加されたヘッダ情報から構成される．本処理系では，表 1 に示す 5 種類の LISP メモリに分け，それぞれの領域を分割して管理している．

名称	拡張	アラインメント
CONS	追加	8 バイト
SYMBOL	追加	8 バイト
HEADER	伸長	8 バイト
VECTOR	伸長	4 バイト
STACK	伸長	4 バイト

表 1: LISP メモリ

メモリの拡張を行うとき，CONS, SYMBOL はメモリブロックの追加を行うが，他のメモリは，全体が連続した領域である必要があるので，メモリの伸長を行う．メモリの伸長は，現在よりも，より大きなメモリブロックを獲得し，そこに現在の内容をコピーすることによって行う．

- CONS メモリ
  CONS データを格納する．CAR 部と CDR 部から構成され，それぞれに任意の LISP オブジェクトを格納する．
- SYMBOL メモリ
  SYMBOL データを格納する．シンボル名や，シンボルの値などを格納する．
- HEADER メモリ
  CONS および SYMBOL 以外の副構造を持つ LISP オブジェクトは，アドレス部に HEADER メモリへのオフセット値を格納する．このため，HEADER メモリ全体は常に連続な領域となっている．
  副構造物の値は，この HEADER メモリ中に格納する (浮動小数点数やストリームなど) か，または，さらに HEADER メモリを介して，他のメモリ中に格納する (ベクタや文字列など)．
- VECTOR メモリ
  VECTOR メモリには，文字列やベクタなどの可変長データを格納する．VECTOR メモリが使用される場合，その領域へのポインタは，常に HEADER メモリに格納する．
- STACK メモリ
  LISP スタックとして使用する．

4.2　ガーベッジコレクション (GC)

本処理系では，マーク/スイープ方式による一括型の GC を採用している．GC によって，CONS メモリ，SYMBOL メモリ，HEADER メモリの回収と VECTOR メモリの圧縮が行われる．

このとき，CONS メモリ，SYMBOL メモリは移動せず，また，HEADER メモリは，常に全体が連続した領域であるため，オフセット値は変化しない．即ち，GC の前後で LISP オブジェクトの値は変化しない．

よって，ある LISP オブジェクトを GC の前後で参照するとき，GC の後で，LISP オブジェクトの値が変化したかどうかのチェックをせずに済むため，高速に処理が行われる．

5　移植性

本処理系は，Windows95/NT, UNIX などの代表的な OS で動作するように，インタプリタの大部分を ANSI C で記述している．また，LISP オブジェクトの構造やアクセス方法も高速性を失うことなく各種 CPU で利用できることにより，高い移植性を実現している．

ただし，浮動小数点数演算のオーバーフローとアンダーフローをチェックする部分は一部アセンブラで記述している．

6　おわりに

現在開発中の Windows95/NT および UNIX 上で動作する ISLISP 処理系の実装について述べた．

本処理系ではタグ構成を工夫し，CONS, SYMBOL は直接実体にアクセスでき，また，メモリ管理方法の工夫により，GC によって LISP オブジェクトの値は変化しないことにより，高速化を実現している．

今後は，バイトコードコンパイラ，オブジェクトシステムの実現を行う予定であり，現在開発中である．

参考文献

[1] プログラム言語 ISLISP 作成原案, ISLISP JIS 原案作成委員会, 1997 年 12 月 25 日

[2] 伊藤 貴康: LISP 言語国際標準化と日本の貢献, 情報処理 38 巻 10 号, 1997 年 10 月

[3] Lisp の ISO 標準規格 ISLisp の処理系の研究開発, 情報処理振興事業協会 第 16 回技術発表会論文集, pp.339–343, 1997 年 10 月

[4] Nagasaka A., Shintani Y., Ito T., Gomi H. and Takahashi J.: "Tachyon Common Lisp: An Efficient and Portable Implementation of CLtL2", ACM Conference on Lisp and Functional Programming, Vol V, No. 1, pp.270–277(1992)

[5] Guy L. Steele Jr., "COMMON LISP THE LANGUAGE Second Edition", Digital Press, 1990