8086 用試作 Lisp の設計とその機能

井田昌之（青山学院大学）

1. はじめに

8086を搭載したパーソナルコンピュータ用のLispを設計し，その初版を試作したので，その設計と機能について報告する。このLispをLisp 05と呼ぶ。

Lisp 05は，CP/M-86をOSとして，NS200/05上で動作する。また，接続されたUX-300との通信機能を有している（図1）。

2.以下では，1）Lisp 05試作の目的，2）Lisp 05の設計，3）その処理系，4）実行性能について順に述べる。

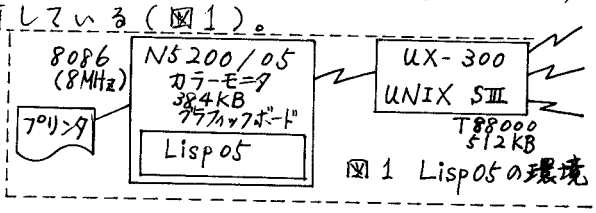


図1 Lisp 05の環境

2. Lisp 05の試作の背景と目的

Lisp 05はおよそ次のようなことを念頭に設計した。

①数式処理言語Reduce3をパーソナルコンピュータへ移植するための基礎データを得たい。そして可能ならば移植する。

②8086系のCPUへのLisp処理系の適合設計。

2.1. Reduce Execution Vehicle としてのLisp

筆者は82年度まで所属していた間野研究室において，8080をCPUに持つALPS/I［Ida79］および，ビットスライスによるALPS/II［SAT82］の試作研究を行なってきた。

特に，ALPS/IではReduce2［Hea73］を動かし，その動態についてのデータを得ることを，小林茂男，遠峰隆好，佐藤衛の修士論文および学部卒業論文を通して行ない，また，学会にても発表してきた［Ida78］。（表1）

表1 Reduce2 on ALPS/I

版	才1版 パイロット 1977~78	才2版 1978~79	才3版 1979~81
機能	約7割に削った版（行列，高エネルギー物理，文処理等を削除）	高エネルギー物理を除く全機能	全機能（Reduce 2マニュアルの機能を全てチェックした）
所要サイズ フリーストレージ（56k max）	30kセル（全体の53%）	35kセル（60%）	48Kセル（80%）
所要サイズ アソシエータ領域（2kスロット）	—	1317（65%）	1600（80%）
参考	$\frac{d^2}{dx^2}ax^2y$ に13秒．ローディングは最初紙テープ1h，後デジタルカセット20分	フロッピーロード20分	QED計算の例題を2パートに分けて50分で実行（GC 7回，2047アソシエータ）

才3版のReduceは佐藤衛の労力による所が大きいが，バグもほとんどなくなり，Reduce2の全機能を実行させることができた。合わせてマニュアルの邦訳も作成した［Man80］。この版の作業域（Reduceによって占められていないフリーストレージ）は約8Kセルと少ないにもかかわらずQEDの例題までも実行できたことは，ALPS/I試作の意義を評価できる。

Lisp 05においては，これらの成果を元に，まずReduce2を移植試験し，その状況をふまえ既にRand Corp.のHearn博士より提供をうけたReduce3［Hea83］を最終的に移植することを目的

の一つとする。Reduce 3は，Reduce 2にくらべ大幅に機能が増えており，ALPS/Iの設計そのままでは移植できない。また速度的にもALPS/Iでは2MHzの8080をCPUとしているので充分ではない。これらを解決しなければReduce 3の移植実験の意味は半減してしまう。

また，①作業領域としては，必ずしも膨大なフリーストレージを用意しなくともよい。（そのマシン環境に見合った充分な規模が実行できればよい）
②その他の領域（アソシエータ領域，束縛スタック，コントロールスタック等）もそれほど大きな領域はいらない。
③Reduceプロセッサはコンパイルしておけば，実行性能を上げられると同時に，フリーストレージの消費をおさえられる。

などの点がALPS/Iの結果から仮定できる。ALPS/Iでの方針を基本的に拡張しなければならないのは，コンパイラの採用という点だけであり，他は基本的にはALPS/Iの考え方で対処できる。また逆に，基礎データ（比較データ）を得るには，ALPS/Iからの継続性は少なくとも必要となるが，基本的な設計が等しくとれれば結果的にこのことは満たされることになる。

2.2. 8086用Lispとしての側面

8086ベースのパーソナルコンピュータは，現在かなり普及しているといえる。これらの上で，ある程度実際的な規模のLisp処理系が可能となれば，Lispの普及・教育と理解に役立つことができる。

そこで，OSの選択にあたっても かなり内部に対する見通しがよく，また使用量も多いCP/M-86をベースとして試作することを前提とした。

次に，実現するLispの規模としては，8086およびその後継のハードの機能に合わせた規模を想定することとした。すなわち，たとえばポインタを24ビット長，あるいは32ビット長に取り，仮想記憶あるいはその他の技法を用いるような，ソフトウェアオーバヘッドの大きい設計はとらないことを決めた。

これにより基本的に16ビット長のポインタ構成を前提とすることとなる。このことは規模の上限を決定することになるが，Reduceを想定しても，2.1で述べたような原則を是とすれば，16ビット空間でもかなりの仕事ができることになる。

応用が拡大し，規模の拡張が必要だとすれば，その時点でたとえば286（ないしは386）の機能を利用して24ビット化（あるいは32ビット化）すればよいという方針をとることにした。たとえば，32ビットアドレス値のロード命令（LES命令）は，8086では13クロックを要し，8086でこの命令の使用を前提とするのは高価すぎるが，80286では7クロックで済み，1クロックの時間が同じだとしても約半分の所要時間で済むことになる。現在のLisp05における対応処理部分は，ポインタ値からフリーストレージアドレスを作成する1ビットシフト命令が2クロック，アソシエータアドレスを作成する2ビットシフト命令が16クロックである。また32ビット加算は（8086：6クロック ⟶ 286：4クロック）と所要クロック数が減らせるなど，286になれば自然に32ビット化が可能となると思われる。

次に，8086用Lispとして具備する機能を考えてみる。総じて言えることは，

パーソナルワークステーションとしてのLisp05システム

としての機能充実を試みることである。具体的には，

①充分な関数の具備

②ホストに対するTSS知能端末としての動作
③CP/M-86, CCP/M-86コマンドの呼び出し実行
④Two Window 処理を含むグラフィック機能の活用
⑤数式処理を含む文書処理

を目標とする。Reduceが動けば①と⑤はおよそ満たしうる。②～④については，そのためのソフトウェアを開発し具備させることとした。このための基本的な道具立てについては[Ida 83]でふれた。図2にCPMコマンド実行の例を示す。

```
EVAL entered ...1035.4600
(cpm '"STAT *.DAT")
1036.0300

Drive A:                                    User :  0
Recs  Bytes FCBs Attributes      Name
   5     2k    1 Dir RW          A:BITA     .DAT
   4     2k    1 Dir RW          A:BITB     .DAT
   1     2k    1 Dir RW          A:FIB      .DAT
   3     2k    1 Dir RW          A:SEQ100D  .DAT
   1     2k    1 Dir RW          A:SEQ20D   .DAT
   2     2k    1 Dir RW          A:SEQ60D   .DAT
   2     2k    1 Dir RW          A:SEQ80D   .DAT
   7     2k    1 Dir RW          A:SEQUENCE.DAT
-----------------------------------------------
Total:    16k     8
A: RW, Free Space:          50k
END of EVAL,  Value is...1036.1000
NIL

EVAL entered ...1036.1100
```

図2 CPMコマンド実行例

3. Lispo5の設計と仕様

Lispo5は次のような仕様によることとした。

i) トップレベルをevalとする。
ii) 変数束縛はスタックによる。
iii) コンパイラを必須とする(初版においては未完成)。
iv) グラフィックプリミティブを内蔵する。
v) その他全体的な処理系仕様および具備する関数群は，ここに記すものを除いてALPS/IのLispに準ずる。このことにより生じる細かな設計については既発表のものと重複することになるので，ここでは省略する。
vi) データタイプとしては図3に示すものを基本とする。
vii) メモリモデルは図4に示す。
viii) ポインタの構成を図5に，データタイプの識別方法を図6に示す。

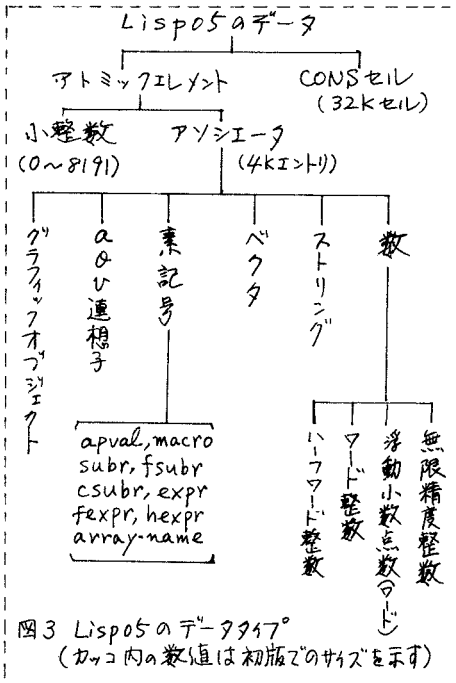


図3 Lispo5のデータタイプ
(カッコ内の数値は初版でのサイズを示す)

3.1. データタイプ

Lispo5のデータは，CONSセルとアトミックエレメントに分けられる。アトミックエレメントは小整数と，実体をもつアソシエータに分ける。グラフィックオブジェクト，aa連想子，ベクタを持つ点，csubr(コンパイルされたオブジェクト)を持つ点などが特徴的である。ベクタは，3つまでの16ビット整数値のベクタで，①グラフィック等の座標データ，②配列名と連想させ，配列の添字としてなどに用いる。ベクタは<x y z>のように表現し，値を表わす。数値のリストとベクタとの間の変換関数ltov, vtoiを用意する。

その他，基本的なデータタイプおよびその表現はALPS/Iに準ずる仕様である。

3.2. メモリモデル

メモリモデルは図4に示す。フリーストレージは2分割しCARバンクとCDRバンクに分ける。これによりポインタの識別力を2倍にする。Lisp05のシステムは環境（操作プリミティブ，コンパイラ，インタプリタ，スタック）と素バンク（アソシエータ領域），コードバンク（コンパイルされたコードがはいる），補助バンク（グラフィックプリミティブ，通信ユーティリティ等の汎用モジュールがはいる）に分ける。

8086には4つのセグメントレジスタがある。CS（コードセグメント），DS（データセグメント），SS（スタックセグメント），ES（エクストラセグメント）である。1つのセグメントは64KBであり，従って同時には256KBまでしかアクセスできない。それ以上の領域を扱う場合にはセグメントレジスタの切換が必要となる。Lispにおいて静的に参照すべきなのはフリーストレージであるので，これらは極力入れかえの必要がなく，また，機械語で直接アクセスできるように割付けることが速度の点からも望ましいことになる。

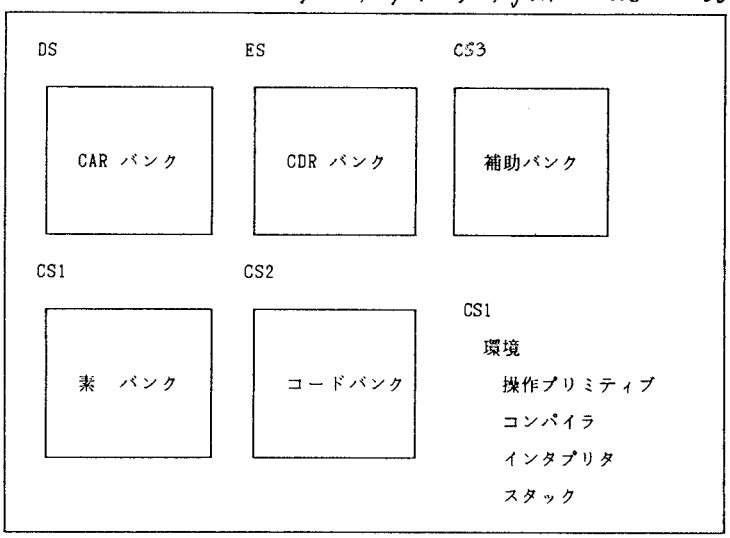
そこでCARバンクはDSに，CDRバンクはESに割りつけるように設計した。環境部と素バンクはLisp05初版では同一セグメントに割りつけ，これを論理的にCS1と呼ぶ。コードバンクはCS2と呼び，その中ではBig modelでオブジェクトを形成することにより64KBを越えても格納できるようにした。補助バンクはCS3と呼ぶ。スタックは環境部から直接アクセスもでき，大きさもそれほど大きくないので，SSレジスタはCS1と同じ値とし，同一セグメント内に含ませた。CS中のローカルデータの処理は8080モデルとして処理し，DSレジスタを可能な限り切り換えないようにした。

CP/M-86システムのいるセグメントをCS0と呼ぶ。CS0，CS1，CS3の切り換えはプログラム割込みによる[Ida83]。CS2はインタプリタからのfarcallによる。

3.3. ポインタの構成

ポインタは16ビット長とし，図5の意味づけを行なう。第0ビットが1であればそのポインタはCONSセルである。アトミックオブジェクトは第0，1ビットが00である。図6に識別手順を示す。識別のためのシフト命令は2クロックしかかからない。またそれにより残された結果は各々のアドレス値としてそのまま用いられるように設計した。実例を図7①②に示す。

この構成は8086ではかなり速い設計であろう。対象の選択をする各々のルーチ



CS0：CP/Mレジデント　CS2：コンパイルドコード
CS1：Lisp05システム　CS3：グラフィックプリミティブ，通信ユーティリティ

図4　メモリモデル

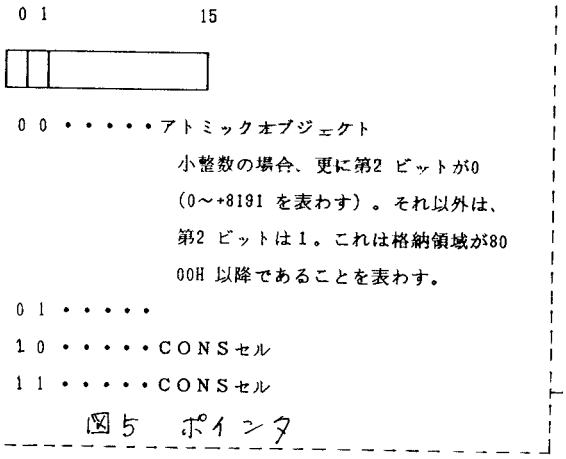


図5 ポインタ

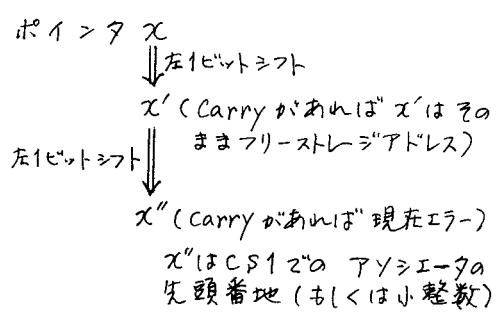
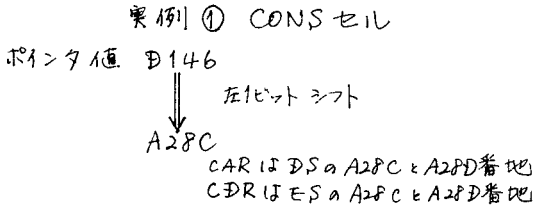


図6 ポインタからのデータタイプの識別(初版)



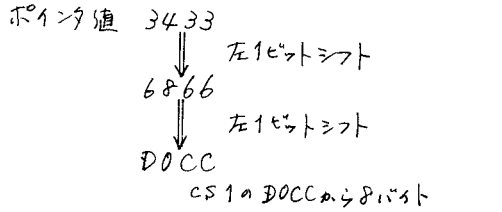


図7 ポインタの識別とアドレスの形成例

ンでは，オーバーフローフラグ等をも利用することにより，無駄な時間をかけずにタイプチェックができる．

4. Lisp05 処理系

Lisp05の処理系(CS1)のコードは現在約16kB，グラフィックプリミティブは約2kB，通信ユーティリティは20kBを占める。通信ユーティリティは日本電気ソフトウェア生産技術研究所の協力をえている。他はすべて独力による。インタプリタは2，3の改良，たとえばexprはeval内だけで処理する，を行なったが基本的にはALPS/Eと等しいのでここでは省略する。関数は現在約90を備えている。インタプリタとコンパイルコードでのレジスタ規約は次のようにした。AX：値レジスタ，BX：第1引数，SI：第2引数，DI：第3引数。

処理系に含められた特徴的な機能のいくつかを紹介する。

①グラフィックプリミティブ：N5200のもつグラフィックボードを生かした2画面処理を可能にするためのものであり，7220制御シーケンスを機能ごとにエントリを持たせ，INT216プログラム割込みをインタフェイスとした。このcallは(Gbios fuc# パラメータ1 パラメータ2)の形式をとる。"グラフィックオブジェクト"からは自動的にこのseqへ変換される。

②標準入出力の切り換え：コンソール入出力による会話をファイルに指定するリダイレクション機能を設計した(図8)。

(stdin 'ファイル名) により入力を，

(stdout 'ファイル名) により出力をファイルにとる。(ただし現在stdoutはデバッグ中)。

③CPMコマンドの実行(図2)：Lisp05に一度はいればCPMに戻らなくともすむ

ようにコマンドを実行できるようにした。このCPM subrでは次のことを行なっている。BDOS call #47(プログラムチェイン)および#0(return)をパッチし，#47を実行する。そうすると帰ってくるのでパッチを戻して終わり。ビルトインを実行させるとA>にもどりCP/M-86の実行が進む。ctrl Cをいれると Lispにもどる。(トランジェントの場合は1実行だけですぐ帰ってくる。)

④ Monitorコマンド：デバッグ用にddt86と往き来したり，アソシエータ領域のダンプをしたりするエントリをもつ関数。

5. 実行結果に見る評価 (表2, 表3)

Lisp05の各ルーチンは，ALPS/Iの対応ルーチンの半分のステップ数で記述できている。コンテストの例題の実行結果からは，ALPS/Iの約16倍の速度であることがわかる。(クロックで4倍，ソフトで4倍。) またFIB(20)の結果から，Interlisp(PDP10)程度の速さが読める。

6. まとめと今後

Lisp05は当初の予想を上まわる機能を持ちうることがわかった。[MAS80]や[GRI81]を参考にコンパイラは現在Lispで記述し，インタプリタ上でテストを行ない仕様をまとめている。これを用いてReduce 2をテスト移植し，その結果をみてReduce 3.0に挑戦する予定である。結果については改めて報告したい。

```
EVAL entered ...1048.0900
(stdin 'BITA)
1048.3200

END of EVAL,  Value is...1048.3300
BITA

EVAL entered ...1048.3300

1048.3300

END of EVAL,  Value is...1048.3300
bit

EVAL en^    《中 略》  ^^48.3400

END of EV..            .s...1048.3400
mapappend

EVAL entered ...1048.3500

EOF encountered on STDIN

EVAL entered ...1048.3500
(bit '(a b c d e f g h i))
1048.5000
 G.C. START ! G.C. START ! G.C. STAR
END of EVAL,  Value is...1049.1900
((a @ (b @ (c @ (d @ (e @ (f @ (g @
 (h @ i)))))))) (a @ (b @ (c @ (d @ (
```

図8 標準入力のリダイレクション

参考文献

[GRI81] M.L. GRISS 他：A Portable Lisp Compiler, software pract. & Exp. Vol 11 pp541-605

[HER73] A. C. HEARN：Reduce 2 User's Manual：Univ. of Utah

[HEA83] A. C. HEARN：Reduce 3 User's Manual：Rand Corp.

[Ida78] 井田 他：ALPS-REDUCEのインプリメンテーション 情処第19回全国大会 pp29-30

[Ida79] 井田，間野：An Adaptable Lisp Machine based on Microprocessors；proc. IMMCC IEEE CS pp210-215

[Ida83] 井田：8086系試作LispのOSプリミティブについて 情処第27回全国大会 pp1333-1334

表3 FIB(20)の実行時間 (WGSYM24より) 単位：秒

Lisp05	Vlisp86	lelisp68k	S3600	Interlisp
9.3 ±0.25	23.5	12.0	28.1	9.25

表2 実行時間の比較 単位：秒

	ALPS/I 8080 (1978)	Lisp05 8086 (1984)	UTLISP CDC6600 (1978)	L1.9 T5600 (1978)	L1.9 ACOS1000 (1983)	AIM1.0 MS50 (1983)
BITA-7	25.5	1.5 ±0.25	1.264	0.936	0.104	0.583
A-8	89.5	5.5 ±0.5	4.628	2.925	0.362	2.026
A-9	-	29.0 GC5回 ±1	-	15.236	-	-
BITB-7	4.95	0.3 ±0.05	0.308	0.172	0.022	0.119
B-8	16.0	1.0 ±0.2	1.009	0.549	0.070	0.380
B-9	53.0	3.0 ±1	3.937	1.832	0.231	1.267
B10	185.0	17.0 GC3回 ±1	-	-	-	-
SEQ 60	62.5	3.7 ±0.3	5.831	2.061	0.242	0.363
80	93.0	5.5 ±0.5	10.428	3.121	0.373	2.116
100	113.0	7.0 ±1.0	16.319	6.154	0.479	2.720
データ出典	[TAK78]	equal appendをsubrにすれば約70%の時間	[TAK78]	[TAK78]	[ITO83]	[ITO83]

参考文献(続き)

[ITO83] 伊藤貴康 他：マイクロプログラミング方式によるLisp処理系とその高速化；WGSYM 22-1

[MAN80] 間野浩太郎：ALPS/I-Reduce使用の手引き；科研費 B-446193

[MAS80] L.M. Masinter & L.P. Deutsch：Local Optimization in a Compiler for Stack-based Lisp Machines；Xerox PARC SSL-80-4

[SAT82] 佐藤，井田，間野：会話型人工知能専用機ALPS/IIのLisp処理機構；WGSYM 17-2

[TAK78] 竹内：LISP処理系コンテストの結果；WGSYM 5-3