パーソナルLISPマシンの開発

A PERSONAL LISP MACHINE

山口喜教
Yoshinori YAMAGUCHI
電子技術総合研究所
Electrotechnical Laboratory

1. まえがき

マイクロコンピュータの普及や高性能化に伴い計算機の捉え方に対する見方が変化しつつあるが、それはハードウェアのみならずソフトウェアあるいは計算機システム全体に大きな影響を及ぼしつつある。従来大型計算機やTSS端末を介してのみ使用可能であった比較的大規模なシステムや特殊目的のシステムが、パーソナル化しスタンドアローンの状態で使用でき得る環境が着々とととのいつつある。

LISP言語は、従来人工知能や記号処理といった特殊目的の言語として研究用に使用されてきた。しかるに自然言語処理の発達や、将来のオフィス・オートメーションの普及、知的能力を備えたワードプロセッサの要求などを考えると、LISP言語や類似の言語が広く使われる可能性も高い。このことと将来のマイクロコンピュータの普及を考え合わせると、従来、大中型機の上にインプリメントされてきたLISP言語システムをその性能を保ちつつ小型化、パーソナル化することが1つの研究課題であると考えられる[1,2][3,4]

このような背景のもとに本稿で紹介するLISP専用マシンシステムは設計、試作された。本システムで使用されたハードウェア素子は、通産省大型プロジェクト「パターン情報処理システム」において開発された16ビットマイクロプロセッサPULCE、高速スタックRAM（制御記憶）、16Kビットのダイナミック RAM（主記憶）などであり、これらにより小型化、高性能化を目指すとともに、リスト処理やスタック処理のオーバーヘッドを軽減する方式をとった。またファームウェアおよびソフトウェア的には、LISP言語に適した中間言語マクロ命令を設定し、これをマイクロプログラムで解釈実行することにより、コンパクトなプログラム表現と高速化が計られている。

2. 設計目標

本LISP専用システム開発の目的は次の3点に集約される。第1には、パーソナル化した計算機システムによりユーザーがスタンドアローンでLISPシステムを使用できるようにすること。第2には、LISP言語特有の処理に必要な機能を専用のハードウェアに持たせることにより、性能の分散をはかり、オーバーヘッドの大きな部分の処理をそのハードウェアで、それ以外の部分は汎用のマイクロプロセッサで処理するという形態をとる。そして第3には、実用規模の比較的大きなプログラムを実行可能なものとすることなどである。このために次のような基本仕様を定めた。

1). マイクロプロセッサを使用することにより、設計・製作の簡易化および装置の小型化を計る。
2). マイクロプログラムにより柔軟な制御構造を持たせる。
3). 主メモリのアドレス空間は比較的大きなものとする。実装メモリ容量も実用規模の

ものとする。

4). LISP言語処理のオーバーヘッド[5]であると考えられる。メモリアクセス処理を1つのハードウェアユニットとして独立させ豊富な機能を持たせる。

5). スタック構造を検討し、スタックアクセスの高速化を計る。

6). LISP言語とほぼ1対1に対応するマクロ命令を定める。

7). マクロ命令はスタックオリエンテッドなものとし、基本命令長は16ビットとする。

8). データ表現にはタグを付加し、データ型の判定を容易にする。

9). LISP言語の外部仕様はTOSBAC 5600上のLISPシステム[11](L1.9)に準拠し、そのサブセットとする。

10). 入出力ファイルは当面、大型機と本システムを端末モードで接続することにより、大型機のファイルシステムを代用する。

3. ハードウェア構成

LISP専用マシンのハードウェア構成を図1に示す。このシステムはデータ処理を行うPULCE、実行順序制御を行うMMC、制御記憶(MPM)、入出力バスインタフェース(IBI)、メモリインタフェースユニット(MIU)がX-バスと呼ばれる外部バスで接続された構成となっている。また主記憶はMIUから出る20ビットのアドレスバス、32ビットのデータバスに接続されている。以下各サブユニットについて概説する。

3.1 マイクロプロセッサPULCE

PULCEに関してはすでに数多くの発表があり[6,7,8]、詳しく解説がなされているので、ここでは概要を述べるにとどめる。PULCEは約7000ゲートが集積された1チップからなる16ビットの高速レジスタ用LSIである。このプロセッサの特徴は、44語という豊富なレジスタを持つこと、またマスク機能や強力なシフト機能、スタック操作性能などを備え、汎用エミュレーションに適した構成となっていることなどである。PULCEの内部処理は16ビット単位であるが、2個のインタフェースレジスタ(IFR0,1)を並列的に動作させることにより外部と32ビットの入出力を行うことが可能となっている。

3.2 MMC (Microprogram Memory Controler)

MMCはPULCEおよびMIUの動作を制御するためのシーケンス制御部であり、これは大型プロジェクトの中で開発・試作されたマイクロコンピュータPMCSとほぼ同様の構成となっている。したがって詳しくは文献[9][10]を参照されたい。本稿では概要と特殊な性能についてのみ述べる。

MMCの主な性能は次の3つである。

(1) シーケンスコントロール性能

16ビットのマイクロプログラムカウンタ(MPC)が次に実行されるマイクロ命令のアドレスを指す。マイクロ命令は32ビット長である。4レベルのスタックを持ち、マイクロプログラムレベルでサブルーチンリンクができる。分岐命令の条件としては、PULCE内のステータス信号の他に、外部信号の状態を指定する

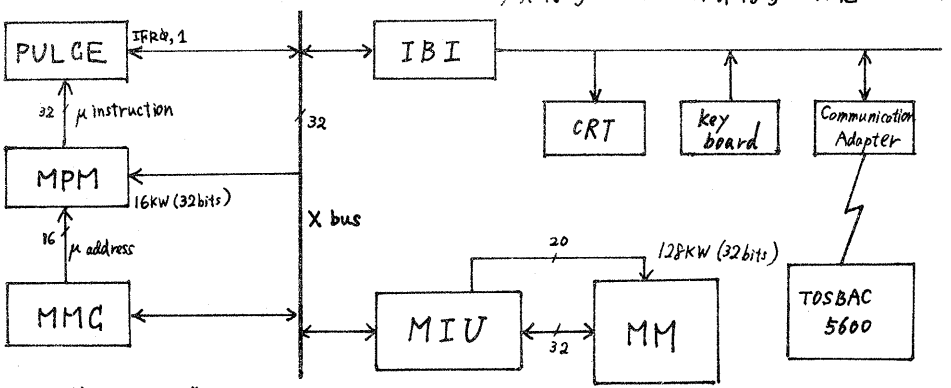


図1 LISP専用マシン構成図

ことが可能となっている。本マシンでは、これを用いLISP言語に特有なデータ型としてのNILやATOMの判定に使用している。

(2). 割込処理機能.

MMCではマイクロプログラムレベルでの割込を許しており、この割込原因は8個まで接続することが可能である。この8個の割込原因ごとに1ビットずつの割込許可ビットがある。割込原因にはPMC$と同様なイリーガル命令の実行やI/Oバスに接続された機器からの割込信号の他に、MIUのハードウェアスタックのあふれや空をによる割込み、またNILデータを操作した場合の割込みなどが付加されている。

(3). バスインタフェース制御機能.

MMCはXバスで接続されている各サブユニット間のデータ転送の制御を行う。

3.3 MIU (Memory Interface Unit)

メモリインタフェースユニットはLISP言語処理の中核を占める。リスト処理、ラムダ変数の結合処理、スタックアクセスなどを効率よく実行するために設計されたモジュールであり、その構成図を図2に示す。MIUの特徴は以下に列挙する如くである。

(1). 主メモリとの間のデータバスは32ビットであり、32ビット単位の読み書きができる。

(2). 各種のポインターや読み出したデータを一時格納するための32ビットレジスタが8コある。

(3). アドレスのインデキシングを行う機構を設ける。

(4). MDRは主メモリから読み出されたデータを格納するレジスタであり、MIRは読み出されたマクロ命令を格納するレジスタである。

(5). MIRに接続されたMPXは16ビット単位のマクロ命令を切り出すことを容易にする。

(6). 8語(32ビット)のバッファレジスタを2組設け、これをハードウェアスタックとして用いることによりスタックアクセスの高速化が計られる。

(7). NINT信号はNILフラグを持ったデータをアドレスとしてメモリアクセスを行った場合に、MMCに対する割込を生じるための信号である。

(3)および(6)については次節で詳しく述べる.

3.4 その他

制御記憶(MPM)は2K語(1語32ビット)のROMと14K語のRAMから成る。また主記憶(MM)は128K語(1語32ビット)の容量を持ち16KビットのダイナミックRAM素子を用いている。通信アダプターは大型計算機(TOSBAC 5600)やその他のシステムと端末モード(RS232C)で接続し、データを受け渡しするためのインタフェースである。

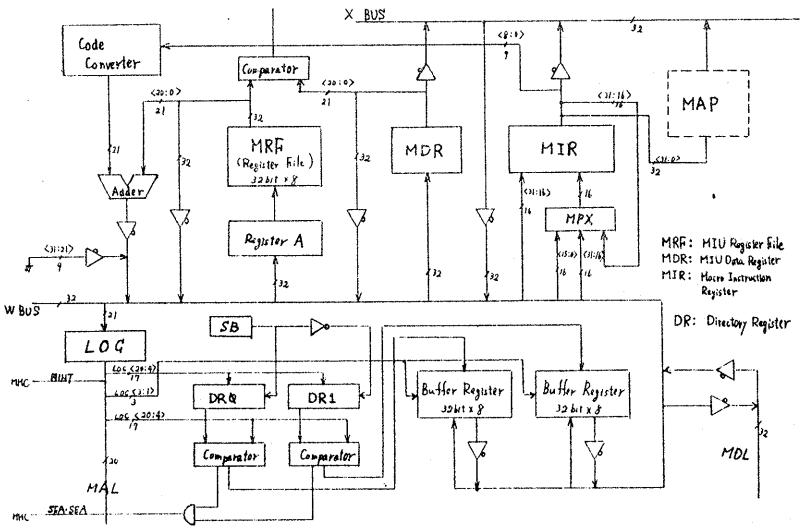


図2. MIU ブロック図

4. マイクロ命令

4.1 マイクロ命令の形式

マイクロ命令の語長は32ビットであり、そのフォーマットや意味は、PULCEを制御する命令、MMC関連の命令（分岐やマイクロプログラムメモリアクセス命令）に関してはPMCSと同様であるので文献[9],[10]を参照されたい。本稿ではMIU制御命令のみにつき概要を述べる。

MIU制御命令のフォーマットは図3に示す通りであり、ここにおけるMIC（Memory Interface Control）フィールドは2種類ある。タイプIは、ARで示される MRFレジスタの内容にAMODで示す修飾をほどこし、得られたアドレスに対してDRで示される MRFレジスタとの間に主メモリとのデータのやりとりを行うものである。（レジスタ間のデータ授受で終わってしまう場合もある。）タイプIIは、RGで示される MRFレジスタとPULCEのインタフェースレジスタとの間でのデータ転送を指定するものである。タイプIにおけるMOPの種類を表1に示す。

4.2 アドレス修飾とスタックアクセス

本マシンにおけるメモリアクセスの様子を図4に示す。AMODフィールドにはアドレス修飾の指定の他に、アドレス値を自動的にインクリメントまたはデクリメントする指定が可能となっている。これによってスタックアクセス時のポインタの増減を自動的に行うことができる。またアドレス修飾はMIRの下位6ビットによっても与えられるので、スタック内に格納されたアーギュメントの値をその相対番号によって直接に読み書きすることができる。

表1において、*Bというニモニックを持つマイクロ命令はMIU内のバッファを使用する命令であり、バッファのディレクトリにアクセスすべき番地が含まれていれば、バッファ内のデータを直接読み書きの対象とする。そうでない場合には2つのケースがあり、BIEビットが1の時には割込みが生じ、割込み処理ルーチンでバッファの内容と主メモリの内容を入れ換える操作をする。一方BIEビットが0の時には割込みは生じずに主メモリの対応番地にアク

図3. MIU制御命令のフォーマット

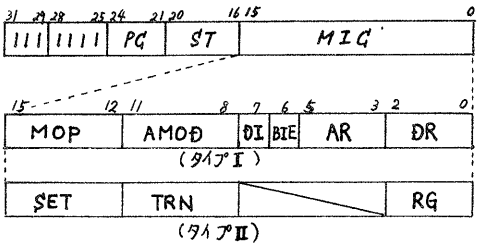
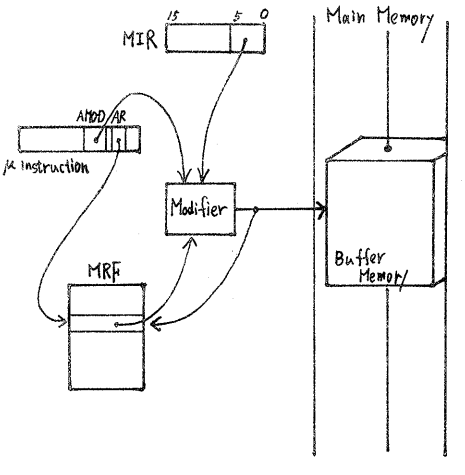


表1. MOPの種類

ニモニック	意味
MRD	Memory Read
MRDT	Memory Read with Transfer
MWT	Memory Write
MWTT	Memory Write with Transfer
MRDH	Memory Read Halfword
MRDH$	Memory Read Halfword or Shift
MRDB	Memory Read using Buffer
MRDTB	Memory Read with Transfer using Buffer
MWTB	Memory Write using Buffer
MWTTB	Memory Write with Transfer using Buffer
MOVBM	Move Buffer to Memory
MOVMB	Move Memory to Buffer
SETDN	Set Directory normally
SETDR	Set Directory reversively
CLRB	Clear Buffer
EQUAL	Equal
MOVR	Move Register

図4. メモリアクセスの方式



セスするのみであり、これは不必要なバッファの入れ換えによってスタックアクセスの実効速度が低下するのを防ぐためである。

5. マクロ命令

本システムにおけるマクロ命令はハードウェアの仕様とはほぼ独立しており、LISPシステム実行に際しての高速性やコンパクトな表現が実現可能であることを目標とした。したがってLISPソースプログラムとこれをコンパイルしたマクロ命令表現との対応関係は明確である。このマクロ命令は文献[12]で示した仮想LISPマシンの中間言語に準拠しているが、改良点としては、分岐(BRANCH)命令の種類を豊富にした点、比較的実行頻度が高いと考えられる命令の組み合わせを1つの命令にまとめたことなどである。なおマクロ命令は主記憶内に格納され、その語長は16ビットである。表2にマクロ命令の一覧を示す。詳しくは、文献[12]または資料[13]を参照されたい。

6. Middle binding 方式

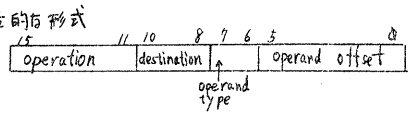
LISPにおける変数名と変数値の結合方式の選択は、LISP処理システム全体に大きな影響を及ぼす。この結合方式は大別してdeep-binding と shallow-binding がある。これをインプリメントの立場から考えてみる。LISPの変数には局所変数と自由変数の区別がある。deep-binding 方式においてLISP 1.5[14]流に、結合対をリスト上に作成すれば、変数の値を求める際に局所変数と自由変数とを区別する必要はない。一方、スタック上に結合環境をおく方式も考えられ、この場合にはコンパイルされた関数を実行する場合の引数の相対位置はあらかじめ知れているため、インデキシングにより引数(局所変数)へのアクセスを高速に行うことが可能となる。

shallow-binding においては、局所変数であるか自由変数であるかにかかわらず、変数へのアクセスは一意に行える。ただし、各束縛変数に対し、関数へのエントリ時に古い値を蓄えておき、関数のリターン時にその値を復帰させるという操作が必要となる。

通常のインプリメントにおいては shallow-binding の方が有利であると言われており、文献[12]の結果もそのことを示している。しかしながら本システムは前に述べたようにスタックアクセスを高速化するためのバッファメモリを備えており、これを一種のハードウェアスタックとして用いることが可能になっている。したがってこのバッファ上に求めるデータが存在している場合にはそのアクセスは通常のメモリアクセスに比し短時間で行える。文献[12]の測定によれば、通常のLISPプログラムにおいては局所変数に対するアクセス数の方が自由変数に対するものよりかなり多くの数になっている。コンパイルされた関数の実行においては、局所変数の位置はスタック上の相対位置として定まり、またこれらの値はスタックの上部がバッファメモリにあれば、バッファ上にある確率が高いと考えられるので deep-binding 的な処理が適していると考えられる。一方、自由変数についてはまだ未知な要素が多いが、インタプリタやコンパイルコードの混在なども考えに入れると、shallow-binding 的な処理が適しているであろう。そこで本システムでは両 binding 方式の長所を取り入れた次のような方式をインプリメントしその効果を比較することとした。これを middle binding 方式と呼ぶことにする。

表2. マクロ命令の形式と種類

代表的な形式



命令の種類	命令表現
データ移動命令	(operation operand destination)
関数コール命令	(CALL / CALL@ function destination)
分岐命令	(BRANCH condition relative-address)
データ操作命令	(operation operand)
定数転送命令	(MOVEC {NIL, T / 0, 1} destination)
変数テスト命令	(operation (ARG name) relative-address)
オペレーション命令	(operation destination)

(1). コンパイルされた関数の局所変数へのアクセスは、局所変数ポインタを局所変数のナンバーでインデクスすることにより、スタック内に直接アクセスする。
(2). 関数のエントリ時には、それまで実行中であった関数の結合環境と value cell の値とを入れ換える。
(3). 関数からのリターン時には、リターンして再開される関数の結合環境に value cell の値を復帰させる。
(4). コンパイルされた関数の自由変数へのアクセスは value cell に対して行う。(通常の shallow binding と同様)
(5). インタプリトされる関数における変数へのアクセスは局所、自由を問わず value cell に対して行う。

7. ソフトウェア・システム

7.1 システム・イメージ

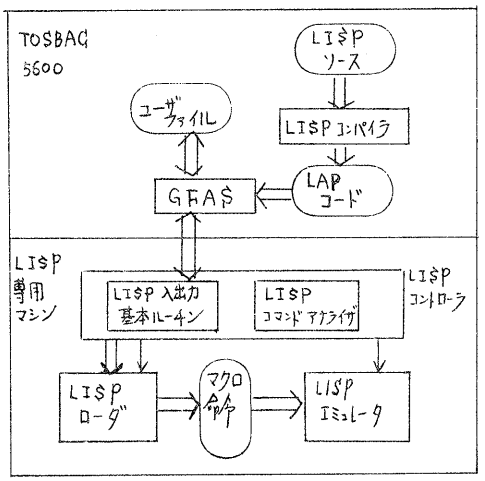
本システムは、LISPのソースをLISPコンパイラによりLAPコードと呼ばれるオブジェクトに変換し、これをローダがマクロ命令としてLISPマシン上にストアし、さらにこの中間言語命令をマイクロプログラムで実行処理するものである。コンパイラは TOSBAC-5600 上に作成され、その出力としてのLAPコードもこの大型計算機上のファイルとして作られる。したがってLISP専用マシンと大型計算機の間のソフトウェアインタフェースが必要であり、この様子を図5に示す。

LISP専用マシン上のソフトウェアシステムは次の3つのサブシステムから構成されている。

(1) LISP ローダ
LAPコードを、LIST構造およびマクロ命令に変換し、メイン・メモリに書き込む。
(2). LISP エミュレータ
マクロ命令、または S式と呼ばれるリスト構造を解読し、実行する。
(3). LISP コントローラ
上記1および2の制御やユーザからのコマンドを解析する。

なおその他にバックアップシステムとして、T56上に、本専用マシン用のマイクロプログラムのアセンブラや ELMCOM と呼ばれる、通信用およびデバッグ用ルーチンが専用マシン上にインプリメントされた。

図5. LISP システム ソフトウェア インタフェース



7.2 MIU内レジスタの意味.

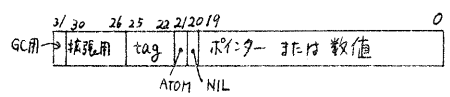
3.3 で述べたようにMIU内に8個のレジスタがあるが、これを種々のポインターとして使用することによりMIU内での処理能率を高めることができる。本システムでは以下のようなポインターやデータをレジスタに格納することにした。

- AC (Accumulator)
  LISPサブルーチンや関数の値がおかれる
- SP (stack pointer)
  スタックのトップをさすポインター
- PC (Program Counter)
  次に解読すべきマクロ命令のアドレスを指す.
- FLP (Free List Pointer)
  フリーリストの先頭を指すポインター
- CFP (Current Frame Pointer)
  現在実行中のスタック上の環境のアドレスを表わす.
- NFP (Next Frame Pointer)
  次に実行すべき環境のアドレスを表わす.
- XP (Index Pointer)
  自由変数や関数コールにおけるインデックスの先頭となるポインター

7.3 内部仕様の概要

◎ リストセルの表現

リストセルは car 部，cdr 部ともに 32 ビットで表わされる。その形式を次に示す。

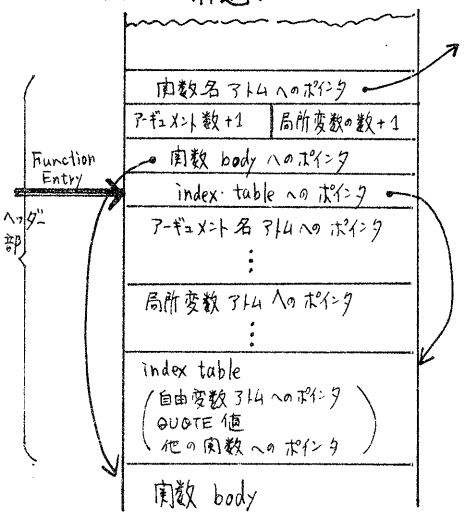


特別の ATOM ビットや NIL ビットにより，ATOM や NIL の判定がハードウェア的になされる。tag フィールドはデータタイプを指定するためのものであり，現在 16 種の内の 10 種類しか定めていない。

○ Form の表現

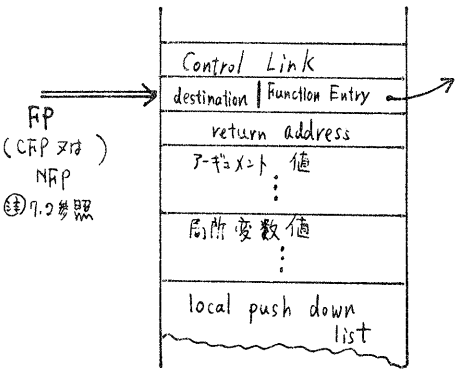
マクロ領域は主記憶上にとられ，各関数の表現単位を form と呼ぶことにするとその構造は図6のようになっている。form は引数や変数のフォーマルパラメータや関数内で使用される定数値などを表わす ヘッダー部と マクロ命令が置かれる関数 body とからなっている。form は必ず偶数番地（本マシンの番地付けは 16 ビット単位になっている）から始まらなければならない。図6における矢印で表わした Function Entry は，関数へのアクセスエントリーであり このポインターから form 内の全ての情報を相対的に求めるためのものである。

図6. form の構造.



○ Frame の表現

Frame とは form に対する環境を表わすものであり，これは表2における関数コール命令を実行した時に生成される。Frame はスタック上に作られるので，Frame の表現はスタック内部構造に外ならない。Frame の表現を以下に示すが，ここにおいて，アーギュメントや局所変数の値は 6 節で述べた middle binding の場合，現在アクティブな Frame においては，その binding の結合値を，インアクティブなものでは，value セルのバックアップとなっていることに注意されたい。



8. おわりに

以上新たに開発した LISP 専用マシンシステムの概要について述べてきたが，当初の設計目標であるパーソナル化と機能分散化への方向付けはほぼ達成できたと考えられる。ただし後者に関しては，将来 LSI の規模が拡大されれば必要がなくなるが別の意味での機能分散が考えられることになる可能性が大きい。

本システムのハードウェアは動作状態にあり，現在 ファームウェアおよびソフトウェアのコーディングを行っている段階である。ハードウェアに関して，インプリメント上の制約から当初予定した性能を省略したり，製作に簡便な手法をとったためにいくつかの問題点が指摘されるが，これらの点に関してはソフトウェアが動作状態となった時点において全体的な立場から評価するつもりである。特に ハードウェアバッファと Middle binding 方式の評価は必要であろう。

謝辞

本システム開発の機会を与えて頂いた、黒川一夫部長、西野博二部長、飯塚肇室長に感謝します。またハードウェア製作にたずさわった、東通エンジリアリングの中村治男氏、ソフトウェア製作に従事してあらゆる構造計画研究所の白井豊氏、河津隆詞氏に厚く御礼申し上げる。

参考文献

(1). Greenblatt, R, et.al. : The LISP Machine, IEEE on Computer (to appear).
(2). 瀬, 金田, 前川: LISPマシンの試作, 情報処理学会論文誌, Vol. 20, No. 6, pp. 487-493 (1979).
(3). 島田, 山口, 坂村: LISPマシンとその評価, 電子通信学会論文誌 D, J59-D, No.6 (1976).
(4). 井田, 間野: マイクロプロセッサを用いたLispマシン ALPS/I, 情報処理学会論文誌, Vol. 20, No. 2, pp. 113-121 (1979).
(5). Clark, D.W and Green, C.C. : An Empirical Study of List Structure in LISP, CACM, Vol. 20, No. 2, pp. 78-87 (1977).
(6). 飯塚, 古谷: マイクロプロセッサアーキテクチャの一設計, 電子通信学会論文誌 D, Vol. 59-D, No. 3, pp. 188-195 (1976)
(7). 電子技術総合研究所: パターン情報処理システムプロジェクト マイクロプロセッサ PULCE 解説書 (昭和51-01)
(8). Iizuka, H., Hayashi, Y., Tamaru, K. and Hara, H. : Development of a high-performance universal computing element - PULCE, AFIPS Proc. NCC, 47 (1978)
(9). 田丸, 花田他: 高性能マイクロコンピュータPMCsの開発, 電子通信学会論文誌D, Vol 62-D, No. 11, pp. 750-757, (1979)
(10). 東芝総合研究所: PMCs マイクロ命令説明書 (昭和53年5月)
(11). 電子技術総合研究所: LISP User's Manual, EPICS-5-ON-4 (1978)
(12). 山口, 島田: 仮想計算機によるLISPプログラムの動的特性, 電子通信学会論文誌, D, J61-D, No.8. (1978)
(13). 山口: ELM-1 用マクロインストラクション =1次案= (内部資料)
(14). Mc Carthy, J. et al. : LISP 1.5 Programmer's Manual, MIT Press (1966).

付録 LISP専用処理システム概観