LISP マシンの試作†

——インタプリータの構造とシステムの評価——

瀧 和男†† 金田悠紀失††† 前川禎男†††

LISP プログラムの高速計算を目的として開発した LISP マシンシステムの高速化インタプリータの構造と稼働したシステムの性能評価に関して論じている．本マシンでは高速化のためインタプリータ全体がマイクロプログラム化されているほか以下の試みを行っている．(1)関数間の制御情報の受け渡しに高速ハードウェアスタックを用いている，(2)マイクロプログラムレベルの再帰呼出が可能，(3)変数のバインディング方式をシャローバインディング方式としている，(4)インタプリータで多用される条件判断において，If then else タイプの判断を繰返すのを避け，フィールド抽出回路，マッピングメモリ出力等を直接用いた高速ジャンプ命令であるマルチウェイジャンプ命令を用いる．

稼働したマシンにおいて，第2回 LISP コンテストで出題されたプログラムを実行し性能評価を行っている．プログラムの実行時間はインタプリータモードにおいてはコンテストに参加した超大型計算機を含むいずれのシステムよりも短く，本 LISP マシンの有効性が実証された．またシステムの動特性の詳細を測定を行い，ハードウェアスタック，マッピングメモリ，各種フラグレジスタ，主記憶などがバランスよく有効に動作して本システムの持つ高速性を実現していることを示した．

1. まえがき

著者等は LISP 言語で記述されたプログラムの高速処理を目的とした LISP マシンシステムを開発したが，本論文は高速化されたインタプリータの構造と実動したシステムに対して行った測定データを中心としたシステム評価について論じている．

測定はベンチマークプログラムとして第2回 LISP コンテスト[6),8)]で出題された課題プログラムを実行して行ったもので，インタプリータによる実行速度は超大型機も含むコンテストに参加したいずれの処理系よりも高速であることが判明した．

なお本論文は文献 1) の内容が既知なものとして議論を進める．

2. 高速インタプリータの構造

開発した LISP マシンにおいてはインタプリータ全体がマイクロプログラム化されており，特に処理速度の向上を求めている．ここではインタプリータの構造とその高速化のため取入れられた試みについて論じる．

2.1 式の読込みと結果のプリント

プログラムは式の読込み，評価，結果のプリントのサイクルを繰返すことにより実行される．式の評価は LISP プロセッサが実行するが，式の読込みと結果のプリントは LSI-11 と LISP プロセッサが相互通信を行いながら実行している．LSI-11 は入力の場合は，入力文字列を括弧，ドット，アトムのストリングに分離し，アトムの登録を行い LISP プロセッサに制御を渡す．LISP プロセッサはストリングを2進木に変換し，式の評価を行う．一方出力の場合は，括弧，ドット，アトムへのポインタの形式をした LISP プロセッサの実行結果を LSI-11 は出力文字ストリングに変換しながら出力する方式をとっている．

2.2 式の評価方式と高速ハードウェアスタックの利用

本システムは高速のハードウェアスタックを実装しているが，スタック上には関数の実行に対応してフレーム (frame) という領域が作られる．フレームの構造を**図1**に示す．

フレームヘッド (frame head) には関数本体へのポインタ，プログラムカウンタ (PC)，後述するマイクロプログラムカウンタ (MPC) を保存する領域 (old PC と old μ-PC) および1つ前のフレームヘッドを指す旧フレームポインタ (old FP) が含まれている．ローカルスタックには関数本体実行時には，計算の中間結果やマイクロサブルーチンへの引数などが置かれる．FP はフレームヘッドの位置を，STP はスタックトップポインタで現在のスタックの先頭を示す．PC,

† An Experimental LISP Machine—Its Interpreter and System Evaluations—by KAZUO TAKI (Omika Works, Hitachi, Ltd.), YUKIO KANEDA and SADAO MAEKAWA (Department of Systems Engineering, Kobe University).

†† (株)日立製作所大みか工場
††† 神戸大学工学部システム学科

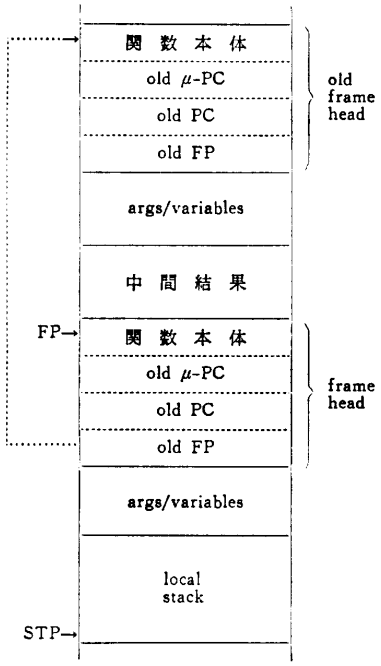

図 1 Frame の構造
Fig. 1 Structure of a frame on the stack.

MPC, FP は Am 2903 のレジスタファイル中にとられる.

プログラムカウンタは式の評価の際, ユーザプログラムの2進木リストを指して順にたどってゆくときに用いる. MPC は Am 2910 中の真のマイクロプログラムカウンタ (μ-PC) にロードするマイクロプログラムアドレス値を保持するのに用いる.

一例として, 式 (関数 1, 引数 1, 引数 2, 引数 3) を評価する場合を考えてみる. EVAL の中で式がアトムでないことが判った時点でフレームが作られ, 関数1の性質が調べられ, 引数の評価方法と評価後の関数本体の渡し方が決る. 引数を評価してはローカルスタックに積んでゆき, 完了後関数本体への制御を渡す. 制御の移動はマイクロプログラム化された関数の場合には関数本体へマイクロプログラムジャンプすることにより行われる. 関数本体の中で引数を用いて計算が終了すると結果を持って呼出しもとのフレームへもどる. もし引数に再び (関数 2, 引数 1′) の形の式があらわれた場合には新しいフレームを作って, 引数リスト (引数 2, 引数 3) と引数の評価方法, 関数本体への渡し方の情報を保存して, 引数 1′ の評価に移る. 関数評価の手順はおおよそ以上の通りである.

引数の評価では EVLIS 関数を使わずにスタックを利用して引数の受け渡しを行うのでむだな自由リストの消費がなく, ガーベージコレクションも起りにくくなっている. またマイクロプログラムの戻り番地をスタック上の old-μPC に格納すること, 引数リストをスタック上に展開することによりマイクロプログラムレベルでの再帰呼出が可能となっており, インタプリータ全体のマイクロプログラム化を可能とするとともに, 再帰呼出制御の効率化を可能としている. また変数のバインディング(binding)形式は shallow-binding で自由変数の値の取出しについて速度の向上が期待される. またバインディング処理では実引数を Apply 処理に渡す際, リストの形でなくスタックに積んだままの形で渡すこととアトムの old value とその値セルの番地をスタックに残す手法を用いることにより処理の高速化を実現している.

2.3 ハードウェアの利用によるインタプリータの高速化

インタプリータで多用される条件判断において, If then else タイプの処理を繰返すのを避け, マルチウェイジャンプ, 間接ジャンプを多用する. マルチウェイジャンプはマッピングメモリ出力の3ビットコード, またはフィールド抽出回路によって取出された4ビットコードと定数アドレスとの和の番地へ1マイクロサイクルでジャンプするもので, データ型や文字アトムの属性の種類により多分岐するときに用いる.

さらに演算結果が nil であるか, アトムか, リストか, 数値か, 文字アトムであるかといった単純な判定は専用のフラグをテストすることにより, 1ステップで済ませる. さらに条件または無条件ジャンプの際には ALU 関係の処理を並行して行い可能なかぎりマイクロプログラムのステップ数を短縮している.

2.2 で述べたように2つのポインタレジスタを持つ強力なハードウェアスタックと Am 2903 中の 16 個のレジスタを並用することにより, スタックマシンとレジスタマシンの長所を合せ持たせるよう工夫している.

3. LISP マシンシステムの評価

本章では完成したシステムで第2回 LISP コンテストに出題されたプログラムを実行したときの処理時間を示し, 大型計算機上に構成された他のシステムとの比較を行う. さらに LISP インタプリータの動的特性を測定し, 実装した各種ハードウェアの利用状態を定量化し評価を加える[3].

3.1 プログラムの実行時間

表 1 LISP コンテストのテストプログラム実行時間例 (msec)
Table 1 Execution times of the test programs of the 2nd LISP contest of IPSJ.

	……INTERPRETER……			……COMPILER……	
	FAST-LISP KOBE UNIV.	HLISP	OLISP	HLISP	OLISP
TARAI-3	55	78	197	17	36
TARAI-4	1,013	1,443	3,727	330	670
TARAI-5	27,538	39,068	100,734	8,905	18,934
TARAI-6	1,011,732	…	…	322,347	…
TPU-1	904	4,790	2,262	1,029	658
TPU-2	3,426	13,411	10,262	2,871+	2,064
TPU-3	1,365	5,684	3,932	1,227++	850
TPU-4	1,815	8,025	5,042	1,707	1,161
TPU-5	238	866	759	181	132
TPU-6	6,728	22,849	20,241	4,893+	3,617
TPU-7	1,311	5,078	4,179	1,053	776
TPU-8	1,187	3,459	3,987	724+	596
TPU-9	819	2,663	2,716	545	424

コンテストに出題されたプログラムを用いて，本システムにおける実行時間を測定したがその結果の一部を**表1**に示す．表記方法と他の処理系の実行時間は文献 6) にしたがう．測定は出題プログラムを評価するための EVAL 時間に対して行っており，READ 関数，PRINT 関数で行う前処理および後処理の時間は含まれていない．測定は LSI-11 に装備した 1 msec のタイマ割込みを用い，精度は +2, −0 msec となっている．比較対象として HITAC-8800 上に構成された HLISP と ACOS-800 上に構成された OLISP を取り上げる．これらの処理系はコンテストに参加したシステムでは最も高速な結果を示したシステムである．

3.2 インタプリータの動特性の測定

表2, 3は TARAI-3 と TPU-6 を実行させたときの総ステップ数，関数やルーチンの呼ばれた回数 (count)，それらの実行ステップの合計が全体にしめる割合である．表3ではステップ数の比率を示しているが，それはほぼそのまま，各ルーチンの実行時間が全時間に占める割合に対応する．表3から ARGEVAL ルーチンの比率が高いことが分るが，このルーチンは引数評価用として EVAL ルーチンを改良し特に高速化を計ったもので，改良の効果は大きいと考えられる．EVAL 1 は EVAL ルーチンの前処理部分を省略した高速化ルーチンである．

表4は各ルーチンや関数に含まれるハードウェア機能の利用回数を集計し，比率を表わしたものである．ここでいう比率とは，各ハードウェアオペレーションを含むステップの総和が，全実行ステップ中にしめる割合のことで，1ステップに複数のオペレーションを含むことが多く，比率の合計は 100% を超えている．

表 2 LISP プログラムの実行時間とマイクロプログラムの実行ステップ数
Table 2 Total execution time and executed micro-program steps of TARAI-3 and TPU-6.

プログラム	実行時間 (msec)	総実行ステップ数	1ステップの平均実行時間 (nsec)
TARAI-3	55	154,763	355
TPU-6	6,726	19,457,963	346

表 3 (a) TARAI-3 実行時の各関数，ルーチンの呼ばれた回数と，全実行ステップに占める割合
Table 3 Total reference counts and percent execution time of functions or routines (TARAI-3).

関数またはルーチン(*)名		回数	比率 (%)
interpreter functions /routines (85.2%)	*EVAL	1,347	} 13.9
	*EVAL 1	2,523	
	*ARGEVAL	5,047	28.0
	*APPLY	673	26.5
	*EXPR	673	7.4
	COND	673	9.2
system functions (14.8%)	GREATERP	673	9.3
	SUB1	504	5.5
defined fn.	TARAI	673	…

表5はメモリオペレーション数の合計を 100% として集計をしなおし，そのうちわけを示したものである．ただし read while write オペレーションの実行時間は，write オペレーションの中に含まれる．**表6**は直接ジャンプ（表4の条件ジャンプと無条件ジャンプの和）オペレーションを集計しなおして，うちわけを示したものである．jump with other op. とは，ジャンプしながらその他のオペレーションを並列実行することである．

表 3 (b) TPU-6 実行時の各関数，ルーチンの呼ばれた回数と，全実行ステップ数に占める割合
Table 3 (b) Total reference counts and percentage execution time of refered functions and routines.

関数またはルーチン(*)名		回 数	比率 (%)
interpreter functions /routines (43.7%)	*EVAL	39,262	
	*EVAL 1	331,900	9.3
	*ARGEVAL	390,002	13.0
	*APPLY	15,706	4.2
	*EXPR	15,706	1.2
	PROG	6,311	9.7
	GO	17,049	1.4
	RETURN	6,311	0.5
	COND	42,360	4.4
system functions (56.3%)	SUBST	4,361	17.3
	EQUAL	12,974	11.3
	SETQ	55,320	6.3
	MEMBER	6,392	4.8
	CAR	43,865	3.2
	NULL	25,743	2.1
	ATOM	19,319	1.6
	CDR	22,415	1.6
	OR	12,892	1.5
	CADR	9,343	1.2
	APPEND	1,210	0.8
	LENGTH	4,224	0.7
	CADDR	3,744	0.7
	LIST	2,936	0.5
	EQ	3,569	0.4
	CONS	3,379	0.4
	others	12,310	1.9
externally defined functions	RENAME	2,291	...
	INSIDE	1,836	...
	DISAGREE	7,559	...
	UNIFICATION	2,018	...
	DELETEV	184	...
	URESOLVE	499	...
	GUNIT	4	...
	PNSORT	4	...
	FDEPTH	0	...
	FTEST	0	...
	SUBSUME	1,293	...
	STEST	8	...
	CONTRADICT	8	...
	DTREE	1	...
	TPU	1	...

表 4 ハードウェア機能利用の割合
Table 4 Utilization of hardware facilities.

オペレーション		比率 (%) TARAI-3		比率 (%) TPU-6	
メモリアクセス	read	15.4	12.8	11.4	10.2
	write		2.6		1.2
ジャンプ	conditional	41.0	17.0	38.3	15.0
	un-cond.		13.0		13.4
	indexed		5.8		3.9
	indirect		5.3		6.1
スタックオペレーション	source	38.0	18.4	49.0	23.4
	destination		19.6		24.6
	stack to stack		1.1		3.4
	[( )−(+( )]		[13.0 12.7]		[18.9 17.8]
直接数値演算		22.2		19.1	
ALU オペレーション		20.2		20.3	
FEX オペレーション		1.3		0.6	
マッピングメモリ直接		7.0		3.9	
IDLE		1.0		0.4	

FEX: フィールド抽出回路

表 5 メモリオペレーションのうちわけ
Table 5 Classifications of memory operations.

オペレーション	比率 (%) TARAI-3	比率 (%) TPU-6
READ, car only	19.0	34.0
READ, cdr only	0.0	19.2
READ, car & cdr	64.1	36.2
WRITE CAR	16.9	7.5
WRITE CDR	0.0	2.1
WRITE BOTH	0.0	1.0
read while write	8.5	6.9

表 6 直接ジャンプのうちわけ
Table 6 Classification of the direct jumps.

直接ジャンプオペレーション	比率 (%) TARAI-3	比率 (%) TPU-6
un-conditional	43.4	47.3
MAP-flag	18.5	17.3
NTL-flag	6.5	12.1
Stack over-flag	5.1	10.3
ALU-flag	26.5	13.0
jmp with other op.	77.4	70.6

4. 測定結果によるハードウェア各部の評価

LISP 処理のため実装した各種ハードウェアの有効性の評価を実用的なプログラム例と考えられる TPU での測定結果を用いて議論する.

4.1 ハードウェアスタック

表4よりスタックオペレーションの比率がたいへん高いことが分るが，以下の2点について考察する.

第1はスタックアクセス時間で，本システムではアクセス時間は完全に1つのマイクロプログラムサイクルに組込まれる．したがってスタックアクセス時間は，スタックをデスティネーションとした場合は 0，ソースとした場合はサイクル延長分の 75 msec となる．このことはスタックオペレーションの比率からみて，十分高速化に貢献しているといえる．スタックア

クセスの時間を不要にしているのは，ハードウェアスタック，ALU，バス相互のデータ経路の選び方と，スタックポインタレジスタの使用および 70 msec の高速メモリの使用によるものである．第2はスタックオペレーションのうち自動減少，自動増加(表中( )ー，+( ) で示したもの) の利用である．スタック操作にしめる割合は共に約 70% で，もし自動減少，増加の機能がないとそのためよけいなステップを必要とする．

以上から，ハードウェアスタックを実装して強力な機能を持たせたことはおおむね成功であったといえる．

4.2 ジャンプに関するハードウェア機能

表4からジャンプオペレーションの比率もたいへん高いことが分るが，以下の3点について考察する．

第1は，もっとも比率の高い条件ジャンプに関するもので，そのうちわけは表6から分かる．マッピングメモリ関係フラグ，NIL レジスタフラグ，スタックオーバフローフラグ，ALU 関係フラグのいずれも 10% 台の利用率であるが，これらのフラグをなくした場合には，定数と比較の後 ALU フラグジャンプとなり1ないし2ステップのオーバヘッドとなる．したがって多様なフラグを用意したことは成功といえる．

第2は，表6の jump with other op. に関することである．これはジャンプしながら他に，スタックオペレーションや ALU オペレーションその他を並行して行うことで，飛び先での仕事の先まわり実行など行うことが多く，直接ジャンプのうち 70% に利用されており，マイクロ命令コードを長くとって並行実行を可能にしたことが成功であったといえる．

第3はインデックスジャンプと間接ジャンプについてで，表4より両者の比率の合計は 10%，ジャンプ中に占める割合は 26% である．これらは主にインタプリータ中で使用され，マイクロプログラムの再帰呼び出しを許したりマルチウェイジャンプを可能にしたりする．この二者を除くと，インタプリータの構造自体を変更せねばならず，効率の低下は大きい．したがってジャンプアドレスとしてYバスデータが利用できるというバス構造は必要と考えられる．

4.3 メモリオペレーションに関する機能

表4から，メモリアクセスのためのステップ比率は 11.4% で，10 ステップに1度はメモリ参照のあることが分る．本システムの特徴は第一にメモリアクセス時間が短かく待ち時間が1サイクルしか必要でないこと，第二にメモリ待ちのサイクル中に別のオペレーションが可能でありむだ時間にならないこと，第三に読出し幅が 32 ビットで，car 部と cdr 部を同時に読出し，メモリアクセス回数を減らせること，第四に read while write 機能があることである．

第二点については，表中の IDLE のところが何もしない純粋なメモリ待ちであり，TPU の場合には，全メモリ参照のうち 96%((11.4−0.4)÷11.4≒0.96) までが，メモリ待ちのサイクル中に別の仕事をしていることになる．すなわち，本システムの実行速度はメモリリミテッドになっていないといえる．

第三点の読出し幅については，表5より car, cdr の両方を利用するための READ が，メモリアクセス中 36% をしめ，比較的よく利用されていることが判る．第四点については，read while write 機能の利用は，メモリアクセス中の 6.9%，WRITE オペレーションの 65% をしめる．この機能をなくすると，読出してどこかに保存したあと書込みを行う手順が必要となり，1回当り最低3ステップのオーバヘッドを生じる．

以上から，メモリアクセス関係のハードウェア構成はほぼ成功したといえ，またメモリアクセスが LISP 処理の速度低下を招かなくなったことは大きな進歩といえる．

4.4 定数演算機能

5ステップに1度は利用されており，マイクロ命令コードを 56 ビットと長くしてこの機能を持たせたことが，有効になったと考える．

4.5 マッピングメモリとフィールド抽出回路

マッピングメモリを直接読出して利用する割合は，TPU では 3.9% であるが，すべてインタプリータルーチンのインデックスジャンプに利用されるため重要である．マッピングメモリフラグの利用率を合わせると 8.8% となる．フィールド抽出回路の利用率は，TPU では 1% を割るが，もし実装しなければ，シフトとマスクにかなりのステップを必要とする．コンパイラやタグマシンの場合にはより有効になると考えられる．

4.6 ALU オペレーション

ALU オペレーションの比較は TPU では 20.3% で，演算もデータ転送も不要のステップは IDLE と単なる直接ジャンプだけであるから，全実行ステップ中の 71% は，単にデータの移しかえだけを行っていることになる．したがって，プロセッサモジュールのバス構造を変更し ALU を経由しないデータパスを設

けることにより，マイクロ命令サイクルをより短縮できる可能性がある．なおビットスライスプロセッサ中のレジスタファイルをソースオペランドとして利用する比率は 25%，デスティネーションとしては 35%（全ステップに対して）であった．

4.7 ビットアドレッシング回路

TPU-6 の実行時に発生した1回のガーベージコレクションを例にとると，総ステップ数が 462,917，実行時間が 157 msec．回収セル数が 36,123（全セルの 88.2%）であり，マーキングに実行時間の 14.3%，回収に 84.3% を費していた．このときの各オペレーションのステップ比率は，BITSET が 1.0%，BIT-TEST が 9.9%，BIT フラグジャンプが 9.9%，メモリアクセスが 9.4%，IDEL が 12.0% であった．すなわちビットテーブルの参照が 10.9% あり，ビットテーブル操作には通常はステップ数を要することから，ビットアドレッシング回路は有効と考えられる．

5. システムの総合評価と考察

以上から，設計段階において効率向上を期待して組込んだ各種ハードウェアはほぼ期待どおりの働きをしていることが確認できたがここでは本システムの高速性についてまとめる．

5.1 プログラム実行の高速性について

プログラムの実行時間が短い理由として，各ハードウェアの高度な機能によるステップ数の減少の他に，マイクロ命令を長くとったことにより，いろいろな部分での並行処理パイプライン処理が行われて，機械語命令では不可能な短いステップ数でインタプリータを実現できたことが考えられる．実際にインタプリータ部分のステップ数は 539 ステップという小さい値である．またもう一つの見方として，本システムのメモリ構成が，インタプリータの記憶のための WCS，データとソースプログラムのための主記憶，インタプリータが制御動作を行うためのハードウェアスタックというふうに大きく3カ所に分散していて，従来のメモリリミテッドなシステムから脱していると考えられる点である．

5.2 コンパイラ実装についての考察

コンパイラを実装した場合に，どの程度の速度向上があるかを考察する．表3を見ると，システム関数の比率が 56.3%（リターン処理を含んでいる）あり，その中から関数からのリターン処理にかかるステップ比率の合計を差引いても残りは約 50% である．すなわちコンパイラによって，インタプリータが行っていた処理ステップが完全になくされても速度向上は2倍にとどまる．実際には中間言語命令のフェッチとデコードのためそれより悪化するはずである．これは TPU の場合であるが，TARAI のようにインタプリータルーチンのしめる割合が極端に高いと，速度向上の割合も高いと考えられる．試みとしてマイクロプログラムで TARAI を記述したときの実行時間は，TARAI-3 から5が各々，14，245，6,646 msec であった．しかしながら TPU の方がより実際的なプログラムであり，本システムではコンパイラを実装しても大きな速度向上は望めないと予想される．このことを逆から考えると，インタプリータによる処理速度がコンパイラの出力コードの実行速度に近づいたとも言えるわけで，これは本研究の成果の中でも興味深いものである．

6. あとがき

ハードウェアを実装する上では，高速の IC を使用するため，接地ラインの強化と配線長の短縮に特に注意を払った．その結果として高密度実装となり，強制空冷が必要となったりして，気軽に試作できる規模としては限界に近い感がある．またインタプリータをマイクロプログラム化してみると，LISP プログラムの内部表現は機械語の一種として取扱うことができるという感触を得た．システムの評価にあたってはシステム依存性の高い部分もあるが，実測データが今後の LISP マシンを始めとする斬新なアーキテクチャを持つ次代のコンピュータシステムの研究に役立てば極めて幸いである．

参考文献

1) 瀧，金田，前川：LISP マシンの試作，情報処理学会研究会資料，計算機アーキテクチャ 32-3 (1978).
2) 瀧，金田，前川：試作 LISP マシンのインタプリータのマイクロプログラム化について，情報処理学会研究会資料，計算機アーキテクチャ 33-2 (1978).
3) 瀧，金田，前川：試作 LISP マシンとその評価，情報処理学会研究会資料，記号処理 7-3 (1979).
4) 山口，島田：仮想計算機による LISP プログラムの動的特性，信学論文誌 D, J 61-D, 8 (1978).
5) 島田，山口，坂村：LISP マシンとその評価，信学論文誌 D, J 59-D, 6 (1976).
6) 竹内郁雄：LISP 処理系コンテストの結果，情報処理学会研究会資料，記号処理 5-3 (Aug. 1978).

7) 黒川利明: LISP のデータ表現, 情報処理 Vol. 17, No. 2 (Feb. 1976).
8) 竹内郁雄: 第二回 LISP コンテスト, 情報処理 Vol. 20, No. 3 (1979).
9) Teitelman, W.: INTERLISP Reference Manual, Xerox (Feb. 1974).
10) Greenblatt, R.: The LISP Machine, MIT AI Lab. Working Paper 79 (Nov. 1974).
11) Knight, Tom: CONS, MIT AI Lab. Working Paper 80 (Nov. 1974).
12) Bobrow, D.G.: A Model and Stack Implementation of Mutiple Environments. C. ACM Vol. 16, No. 10 (1973).
13) Bawden, Aian et al.: LISP Machine Progress Report, MIT AI Lab. Memo No. 444 (Aug. 1977).
14) McCarthy, J. et al.: LISP 1.5 Programmer's Manual, MIT Press (1966).

(昭和54年5月21日受付)
(昭和54年6月21日採録)