試作ＬＩＳＰマシンの機能について

小林 康博 三上 昭弘 高岸 英之 金田 悠紀夫 前川 禎男
神戸大学工学部システム工学科

1．はじめに

ＬＩＳＰ言語の普及に伴い，その実行速度の向上等の要求から，ソフトウェアの要請を極力ファームウェア（ハードウェア化）したＬＩＳＰ処理専用システムの開発が進められている。[8]-[11] われわれの試作したＬＩＳＰマシンは，市販のＬＳＩを利用し，インタープリタの全てをマイクロプログラム化し，ＬＩＳＰ言語処理のための機能をハードウェア化したもので，きわめて高速にＬＩＳＰ言語を処理することが可能である。[1]-[4] 本報告は，試作したＬＩＳＰマシンシステムの概要を述べるとともに，ユーザに提供するソフトウェア，及びその機能（インタプリタ，コンパイラ，組込み関数）について述べる。

2． ＬＩＳＰマシンシステムの概要

2．1 ＬＩＳＰマシンシステムの構成

試作したＬＩＳＰシステムのハードウェア構成を図1に示す。ＬＩＳＰプログラム処理の中心となるプロセッサモジュールとメモリモジュールをミニコンピュータ（ＤＥＣ社ＬＳＩ－11）のバスに接続した構成である。ＬＳＩ－11はパラレルインターフェイスを通じて，マイクロコンピュータＰＤＣ－80と中型計算機システム（FACOM 230-38）にも接続されており，ＰＤＣ－80に接続された入出力機器を利用することが可能である。メモリモジュールは，32ビット×64Ｋ語から構成され，アクセスタイムは約375 n sec である。プロセッサモジュールは，処理幅が16ビット，アドレス空間も16ビットであり，メモリとのデータ転送は，16ビットはもちろん，32ビット幅でcar部，cdr部を同時に読み書きすることも可能である。

2．2 ＬＩＳＰプロセッサモジュールの構成

プロセッサモジュールのハードウェア構成を図2に示す。ビットスライスプロセッサエレメントとして4ビットスライスであるAm2903（Advanced Micro Device社）を4個使用している。このプロセッサモジュールは，ハードウェアスタック，フィールド抽出回路，マッピングメモリ回路，豊富な条件テスト回路ＮＩＬレジスタ，ビットアドレッシング回路等を有している。また制御部を構成するＣＣＵは，Am2910 マイクロプログラムシーケンサ，ＷＣＳ（56ビット×4Ｋ語），ＰＬ（パイプラインレジスタ），ＣＭＲ，フラグレジスタから成り立っている。マイクロ命令サイクルは300 n sec を基本とし，命令コードは1語56ビットで，ＡＬＵまわりとＣＣＵの制御とを並列に行える

2．3 ＬＩＳＰ言語処理向きのハードウェア

以下に示すものがＬＩＳＰ言語を高速に処理するのに役立っている。[1][3]

(1) ハードウェアスタックとバス構成 ： 70 n sec の高速ハードウェアスタックの実装により，スタック上のデータを高速にアクセスすることが可能になる。また，ＡＬＵ演算後のマイクロ命令アドレスへジャンプすることもでき，スタッ

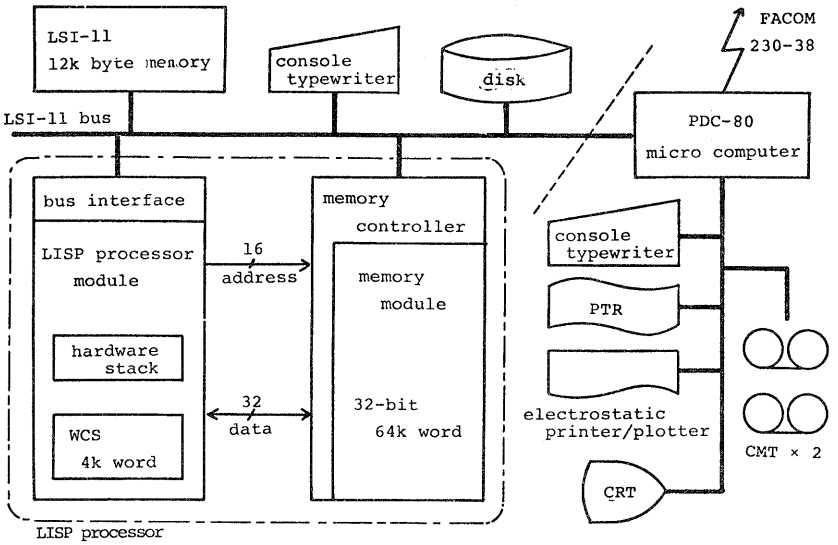


図1．ＬＩＳＰマシンシステムのハードウェア構成

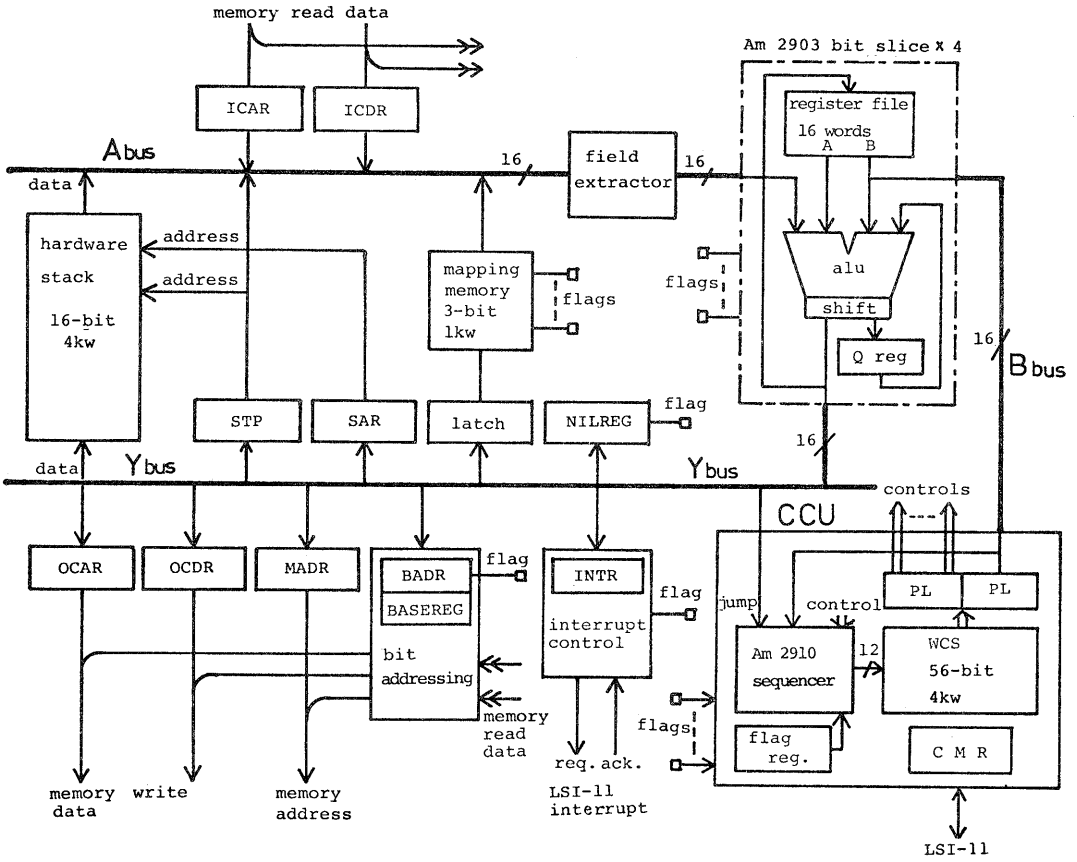


図2．プロセッサモジュールのハードウェア構成

ク上に積んでおいた戻り番地へジャンプすることも可能である。このため、マイクロプログラムレベルでの再帰呼出しが可能となり、主記憶の参照回数を大幅に減らすことができた。

(2) データ型の高速判定 ： 実行中にデータ型を判定する機会は多く、本システムでは主記憶のアドレスとマッピングメモリの内容により自動的にデータ型を判定することができる。このデータ型の結果はフラグレジスタにもセットされ、このフラグを用いてデータ型に対する条件ジャンプが1ステップで行える。

(3) 演算部と制御部の並列処理 ： ＷＣＳの命令コードを56ビットと幅広く取ったため、ＡＬＵまわりと条件ジャンプとを並列に処理できる。このためマイクロプログラムにおけるジャンプ命令に要する時間は0と考えることができる。

(4) メモリの分散によるアクセス回数の低減 ： 本システムではメモリ空間をスタック、主記憶、ＷＣＳ、ＡＬＵ内のレジスタファイルと分散させたため、これらのメモリ内容を同時に利用することが可能である。

(5) その他 ： ＮＩＬの一致検出を行うＮＩＬレジスタ、データのシフトとマスキングを同時に行うフィールド抽出回路等の使用により、実行ステップ数を減少させることが可能である。

3. ＬＩＳＰマシンシステムのソフトウェア構成

本システムのソフトウェアを大別すると2つに分けることができる。第1は、FACOM 230-38上に構成されたマイクロアセンブラと、そこで発生したマイクロコードをＰＤＣ-80に接続されたカセットＭＴ装置、およびＬＳＩ-11に接続されたフロッピーディスクまで転送するオンラインソフトウェアである。第2がＬＩＳＰプログラムの実行時に働くソフトウェア群であり、本システムの中心をなすものである。これらの構成を図3(a)、(b)に示しておく。本システムのマイクロアセンブラは、FORTRAN言語で記述されており、約1300ステップのサイズのものである。このマイクロアセンブラはＷＣＳレベルのみならず、組込み関数用のアトムヘッダ、フルワードデータ、マッピングメモリコード、システム制御用パラメータの初期値設定等のコード発生も行っているため、他のマイクロアセンブラよりも多くの機能を必要とする。第2のソフトウェア群は、インタープリタやコンパイラ等のＬＩＳＰプログラムを処理するためのソフトウェアと、LISPモニタやＬＳＩ-11のＯＳであるＲＴ-11等のＬＩＳＰシステムを管理するソフトウェアから成り立っている。

4. ＬＳＩ-11のソフトウェアについて

ＬＳＩ-11のソフトウェアは、ＬＩＳＰプログラムの実行を管理するＬＩＳＰモニタとＬＳＩ-11自身のＯＳであるＲＴ-11モニタから成る。ＬＩＳＰモニタはこのＲＴ-11のサブシステムとして構成されており、ユーザーはＲＴ-11を通じて、ＬＩＳＰモニタやＬＩＳＰプログラムを作成するためのTEXT-EDITORを呼出したりしながらＬＩＳＰシステムと会話的に処理を行っていく。

ＬＩＳＰモニタの機能を大別すると次のようになる。

(i) ＬＩＳＰプログラムの実行開始、停止、終了の管理
(ii) インタープリタからの要求による入出力処理、タイマ制御処理
(iii) マイクロプログラムのデバッグ機能

(i)については、以前のプログラムの実行により使用されたアトムヘッダ領域、フルワード領域等の初期化、コンパイラの呼出し、既存のＬＩＳＰプログラムの実行、等のＬＩＳＰプロセッサへ起動をかけるコマンドの管理、またプロセッサが起動されたのちの、プログラムの停止、終了のコマンドの管理を行う。(ii)については、ＬＩＳＰマシンからの割込み要求を処理するもので、その割込み処理は、(1) 入出力処理 (2) タイマ制御 (3) メッセージ出力処理 の３つに分れる。入出力処理においては、ＬＩＳＰ言語の入出力関数の中のアトムアナライザの部分を担当している。つまり、アトムヘッダ、ストリングヘッダの作成、および、デリミッタ、ASCIIコード等の内部コードへの変換を行うものであり、二進木リストへの変換はＬＩＳＰマシン側で行っている。またそれらに付随する入出力機器の制御も行っており、特にファイルに対しては、ファイル名で表わされる論理チャンネルからの物理チャンネルへの変換等も行っている。タイマ制御は、GC、READ、PRINT等の関数の実行時間、及びトップレベルの実行時間を計測するものである。(iii)のデバッグ機能は、本システムのインタープリタがマイクロプログラムで記述されているため、マイクロプログラムのデバッグを行う際、ＬＩＳＰプロセッサ側にデバッグ機能を盛り込む事が困難であるため、ＬＳＩ-11上に構成されたものである。これらの機能を列挙してみると、

(a) ＷＣＳ、マッピングメモリ、主記憶の各メモリセルの読み書き
(b) ＣＭＲ（コマンドレジスタ）の読み書き
(c) マイクロプログラムの１ステップ実行
(d) ブレイクポイント設定による任意アドレスでの実行停止、及び再設定と再実行
(e) マイクロプログラム実行総ステップ数の計数

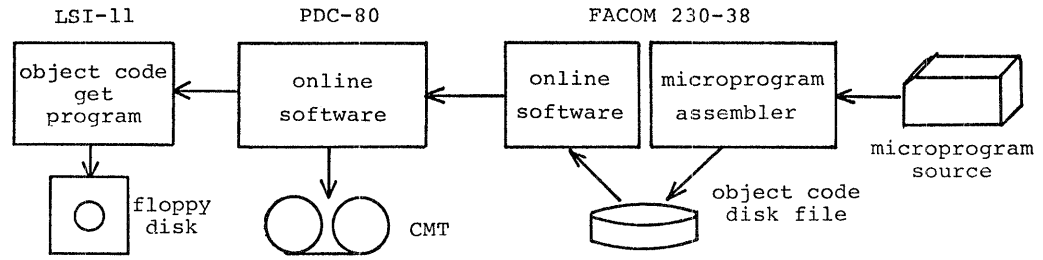
となる。(a)の機能により、マイクロプログラムコードの修正、追加が速やかに行え、また(d)の機能により、任意のアドレスでマイクロプログラムの実行を停止でき、その時のメモリ内容等を知ることができる。またその時のプロセッサモジュール内のレジスタの内容は、メインテナンスパネル側から知ることができる。

5. インタープリタと組込み関数について

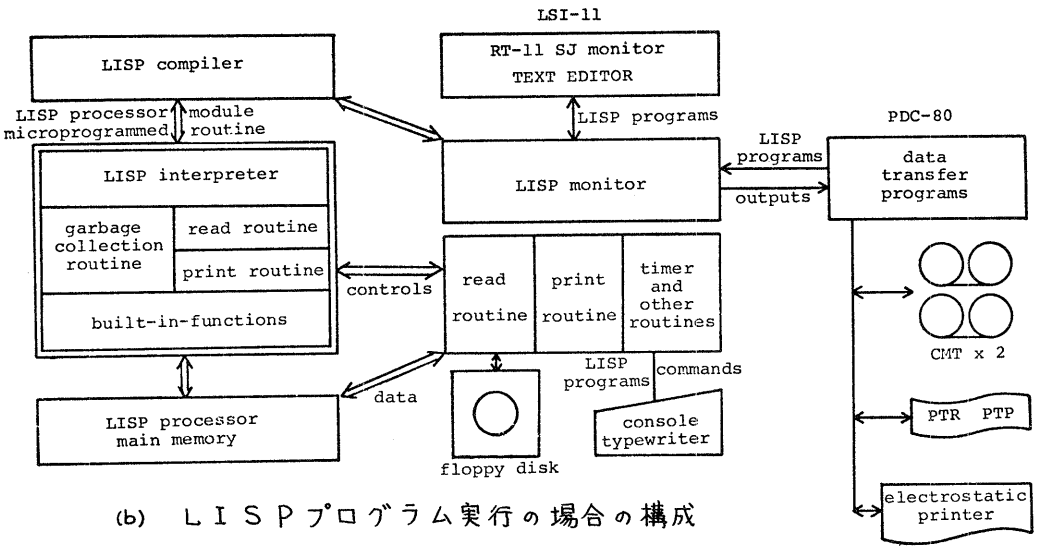
5.1 ＬＩＳＰ言語の仕様

本システムで現在用意しているデータ型としては、小整数、整数、ストリング文字アトム、リスト要素の五種類がある。これらの表現を図４に示す。文字アトムは連続した領域に情報を置くことにより、属性のチェック、属性値の取り出しを容易にした。また、変数値の結合には shallow-binding を採用し、変数値の取り出しを高速に行っている。ストリングに対しては、先頭の文字位置をデータに含め、不必要なフルワード領域の消費をおさえている。また、図５のように主記憶上の領域をデータ型によって分割し、マッピングメモリ回路により、高速に型判別を行っている。

関数型としては、ＥＸＰＲ、ＦＥＸＰＲと、システム関数用として、マイクロコードで記述された MSUBR、MRSUBR、MRFSUBR を用意した。MSUBRは、マイクロサブルーチンとして記述されたもので、再帰呼出しは許されない。SUBR、FSUBR は EXPR、FEXPRをコンパイルしたもので、中間言語命令で記述される。



(a) マイクロアセンブラとマイクロコード転送の場合の構成



(b) ＬＩＳＰプログラム実行の場合の構成

図３．ＬＩＳＰマシンシステムのソフトウェア構成

データ型	語数	表　現
小整数	0	ポインタの値自身　-8192 ≦ n ≦ 8191
整数	1	31 … 0：2の補数表現で整数エリアに格納
文字アトム	4	31 … 16 15 … 12 11 … 0：関数本体 / 属性 / 属性別飛び先；flag / 引数個数 / top-level-value；property-list / print-name；go-label-value / print-name-hash
リスト要素	1	31 … 16 15 … 0：cdr / car
ストリング	1	full wordへのpointer / 位置 / 文字数

図４．　データ型とその表現

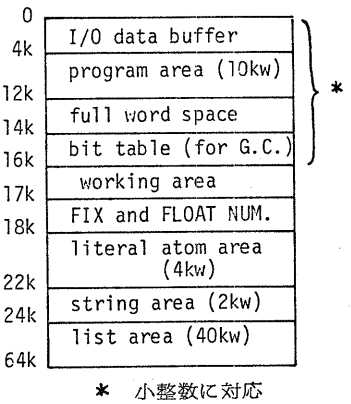


図５．メモリ割当て

各関数型に対し、主な関数を次に示す。ユーザはこれらの関数型を意識しないで利用できる。

```
MSUBR   : CAR,CDR,CONS,EQ,NULL,ATOM,ADD1, etc.
MRSUBR  : EVAL,APPLY,EVLIS,EQUAL,MAPCON,  etc.
MRFSUBR : COND,PROG,AND,OR,LIST,DE,SETQ,  etc.
```

5．2 インタープリタについて

本システムのEVAL、APPLYの外部仕様はshallow-bindingを採用していることを除き、ほぼLISP1.5と同じである。関数評価を行う際、関数の実行に先だち、スタック上にframeが作られる。このframeの構造を図6に示す。frame-headには関数本体へのポインタ、PC、MPCを保存する領域、及び1つ前のframe-headを指すポインタ（FP）が含まれる。frame-headの上に関数への実引数が積まれ、関数本体の実行に移る。評価が終るとRETURNオペレーションにより、スタックトップの値を現FP値にセットし、呼出し元へ制御が移る。

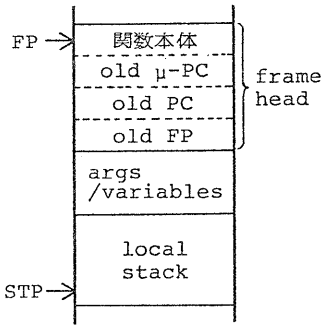


図6．frameの構造

インタープリタの内部仕様は、次の各点に注意して、高速処理を目指している。

(1) 引数評価ルーチン（ARGEVL）の作成
(2) EVLISを使用しないで、直接スタック上に実引数を積み上げる。
(3) EVALやCONDのframeは、APPLY等のframeに転用することにより、むだなframeの作成をおさえる。
(4) if-then-else型のジャンプを用いず、データ型に対するマッピングコードを用いたマルチウェイジャンプを使用する。
(5) フラグレジスタにより、データ型の判別、NILの検出を速やかに行う。
(6) プリミティブな関数はMSUBR型で記述し、他の関数でも直接これを用いる。
(7) NIL、T専用のRETURNオペレーションを作成する。

上記のように構成したインタープリタに対し、第二回LISP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。その結果を表1に示す。他の処理系の実行時間、及び表記法は文献[5]にしたがう。

比較の対象として選んだ他の処理系の特徴を以下に簡単にまとめてみると、

(i) HLISP ： HITAC-8800上に構成されたシステムで、サイクルタイムが50 n sec、アクセスタイムが100 n secの超大型計算機システムである。
(ii) OLISP ： ACOS-800上に構成されたシステムである。
(iii) TOSBAC-5600 LISP1.9 ： TSS向きのLISPシステムであり、ACOSシリーズの計算機に移植可能である。
(iv) LIPQ ： PDP11/55上に構成されたシステムで、サイクルタイムは、300 m secであり、本システムとほぼ同じ規模のシステムである。

表1より、本システムにおける実行速度は他のどの大型計算機よりも短いことがわかる。本システムのサイクルタイムが300 n secであるから、HLISPと

	-----INTERPRETER-----						-----COMPILER-----				
	FAST-LISP KOBE UNIV.	HLISP	OLISP	TOSBAC-5600 LISP1.9	LIPQ		FAST-LISP KOBE UNIV.	HLISP	OLISP	TOSBAC-5600 LISP1.9	LIPQ
BITA-4	6	6	14	24	90	*	2	2	4	7	18
BITA-5	17	22	46	76	300	*	9	9	15	22	63
BITA-6	55	71	155	249	1,010	*	27	30++	50	71	219
BITA-7	188	249	531	936	3,460	*	92	108+	176	248	775
BITA-8	652	869	1,849	2,925	---	*	320	377++	622	499	---
BITA-9	STACK OVER	3,111+	6,498++	15,236++	---	*	1,128	1,359++	2,229	2,705	---
BITB-4	3	6	5	11	45	*	3	1	1	4	9
BITB-5	6	14	13	23	104	*	6	3	3	6	25+
BITB-6	13	40	34	59	282	*	13	10++	9	14	75+
BITB-7	37	118+	104	172	870	*	28	33	28	39	258+
BITB-8	120	394+	338	549	3,070+	*	88	105++	91	126	1,080++
BITB-9	397	1,318+	1,130	1,832	---	*	292	356++	307	712	---
BITB-10	1,355(+)	4,542++	---	---	---	*	1,000	1,230++	---	---	---
SORT-20	73	106	287	505	1,400	*	26	12	18	31	23++
SORT-40	261	374	1,109	1,415	4,900	*	94	43	69	108	85++
SORT-60	415	---	2,459	2,061	7,800	*	150	---	153	176	140++
SORT-80	626	568	4,337	3,121	11,900	*	225	65	271	487+	210+
SORT-100	804	1,134	6,753	6,154	15,000	*	290	135	423	605+	270+
TARAI-3	55	78	197	227	900	*	26	17	36	87	13
TARAI-4	1,013	1,443	3,727	5,124	16,000	*	473	330	670	1,064	255
TARAI-5	27,538	39,068	100,734	138,819	437,000	*	12,849	8,905	18,934	43,609	6,900
TARAI-6	1,011,732	---	---	---	---	*	472,063	322,347	---	1,603,910	255,000
TPU-1	904	4,790	2,262	4,403+	12,000+	*	601	1,029	658	1,591	2,200+
TPU-2	3,426	13,411	10,262	21,148++	55,200+	*	1,957	2,871+	2,064	6,764+	6,100+
TPU-3	1,365	5,684	3,932	7,447+	21,200++	*	805	1,227++	850	2,798+	2,600+
TPU-4	1,815	8,024	5,042	10,958+	27,400++	*	1,096	1,707	1,161	3,250+	3,700+
TPU-5	238	866	759	1,461+	4,000++	*	125	181	132	555	400
TPU-6	6,728(+)	22,849	20,241	66,502++	118,000++	*	3,590(+)	4,893+	3,617	27,049++	10,500+
TPU-7	1,311	5,078	4,179	8,350+	21,800++	*	710	1,053	776	3,189+	2,300+
TPU-8	1,187	3,459	3,987	5,459+	21,000++	*	582	724+	596	1,163+	1,600+
TPU-9	819	2,663	2,716	4,133+	14,200++	*	411	545	424	964+	1,100+
PLISP-3	7,717	3,096	46,061	49,125	---	*	2,934	303	1,876	2,933	2,240
PLISP-3	163	132	575	871	2,150	*	163	132	575	871	2,150
RATIO	47.3	23.5	80.1	56.4	---	*	18.0	2.3	3.26	3.4	1.04
PLISP-4	170,471(+)	56,877	1,096,786	1,118,014	---	*	65,356(+)	4,998	44,910	66,722	48,500
PLISP-4	3,059(+)	2,020	11,384	18,119	44,000	*	3,059(+)	2,020	11,384	18,119	44,000
RATIO	55.7	28.1	96.3	61.7	---	*	21.4	2.5	3.95	3.7	1.1
PLISP-5	STACK OVER	---	---	---	---	*	2,107,056(+)	125,290	---	2,015,495	---
PLISP-5	83,321	46,431	---	503,999	1,249,000	*	83,321	46,431	---	503,999	1,249,000
RATIO	---	---	---	---	---	*	25.3	2.7	---	4.0	---

表1. LISPコンテストのプログラムの実行時間 (m sec)

単純に比べると6倍遅いことになるが、これはハードウェア、ソフトウェア両面での改良の効果が十分に現われたためと考えられる。

5．3　組込み関数について

現在、本システムに組込まれている関数の総数は74種であり、主としてリストを処理する関数群と入出力関数から構成されている。数値演算関数としては、整数の四則演算を行うものだけであり、まだfloatingの導入は行われていない。ストリング処理関数は現在作成中である。以上で述べてきたマイクロプログラムによるソフトウェアのサイズはおおよそ次に示すとおりである。

マイクロプログラム　：　1962総ステップ

インタープリタ　－　573ステップ　入出力関数群　－　217ステップ

ガーベージコレクション　－　122ステップ　組込み関数群　－　801ステップ

以上のようにインタープリタの総ステップ数がかなり小さいことも、本システムの特徴である。

6．コンパイラについて

6．1　中間言語命令

本システムは本質的に機械語命令を持たない。そのため機械語命令に対応した中間言語命令を設定し、それをオブジェクトコードとして生成するコンパイラを構成した。オブジェクトコードは1中間言語命令に対し、32ビットの命令コードに変換され、主記憶内のプログラム領域にロードされる。コンパイルされた関数の実行時には、マイクロプログラムで記述された命令フェッチ部と命令実行部により、命令コードが順にフェッチされ、実行される。

これら中間言語命令の設定の際、命令コードの小型化は考えず、高度な並列処理やパイプライン処理が損われない事に留意し、次のような命令体系を考案した。

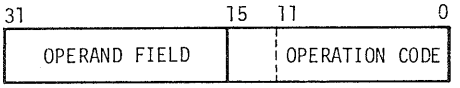


図7．命令コード

(1) CALL　：オペランドで示された関数を、その属性に応じた実行の予備操作を行い、その後関数を呼び出す命令である。

(2) PSH　：データをスタックトップに積み上げる命令である。データとしては、(i) 実行する関数の実引数、(ii) 自由変数、(iii) データそのもの、の3種類が指定でき、以下においても同様である。

(3) POP、STORE　：スタックトップの内容をオペランドで示されるフィールドにコピーする命令で、POPはスタックポインタを1つpop-upする。

(4) MKFRM　：関数の実行に先だち、frame-headを作る命令で、オペランドの指定があるとPSH命令もかねる。

(5) JMP　：条件により、label先へ分岐する命令であり、スタックポインタのpop-upも指定できる。条件としては、(i) 無条件分岐、(ii) NIL分岐 (iii) T分岐、の3種類が指定できる。

(6) RETURN　：スタックトップの内容を関数の評価結果として、現FP値にセットし、frameを消滅させ、呼出し元へリターンする命令である。

(7) SHALLOW　：コンパイル時に自由変数が宣言されていると、その自由変数が束縛を受ける際、shallow-bindingを行う命令である。

6．2　コンパイラの構造と実行時間

本システムのコンパイラはマイクロプログラム、中間言語命令では記述されておらず、ＬＩＳＰ言語で記述された１種のＥＸＰＲ関数である。このコンパイラは、インタープリタからも、ＬＩＳＰモニタからも実行が可能である。ユーザはこのコンパイラを使用する際、関数の中での自由変数、関数の属性等を宣言しなければならない。またオブジェクトコードをプログラム領域へロードする関数（ＬＡＰ）をマイクロプログラムで構成しているため、(i)　オブジェクトコードの生成、(ii)　コンパイル後即実行、の２種のコンパイル条件を指定する事ができる。

このコンパイラは73の関数から構成されており、約４Ｋ語を使用するものである。このうち分けは、コンパイル処理に18関数、ＦＳＵＢＲ処理に９関数、オブジェクトコード発生に20関数、その他の処理に26関数を使用している。

ＥＸＰＲ関数をコンパイルする事で実行速度を改善できる点は、

(1)　仮引数をローカルスタックのロケーションに割り当てる事により、重複して行われる引数評価、及び変数と変数値のbind処理を行わなくてよいこと。

(2)　再帰呼出しを行わない関数群（MSUBR）については、frameを作らなくてよい。また、リターン処理も省略できる。

(3)　COND、PROG、AND、SETQ 等のＦＳＵＢＲ関数群は引数の評価手順を示したものなので、中間言語命令列に簡単にオープンに展開する事が可能である。

(1)については、ＥＸＰＲ関数を評価する際、変数値のbinding のため、ＥＶＡＬからＡＰＰＬＹを呼び、bindingを行い、さらにＥＶＡＬ を呼び出す手順が必要であるが、ＳＵＢＲ関数では、変数がスタックのロケーションとして表わされるため、そのロケーションをアクセスするだけで済み、大幅な処理ステップの短縮が可能になる。(2)、(3)より、関数呼出しのためのframeの作成、ＲＥＴＵＲＮ処理、連続したＥＶＡＬの呼出し手続き等の省略が可能となる。上記の３点によるだけでも総実行ステップ数の50％以上の短縮が可能である。図８に関数定義体とそのコンパイルしたオブジェクトの例を示しておく。またインタープリタと同様にＬＩＳＰコンテストのプログラムをコンパイルした場合の実行時間を表１に示しておく。

```
(DE TARAI (X Y Z)
 (COND ((GREATERP X Y)
        (TARAI (TARAI (SUB1 X) Y Z)
               (TARAI (SUB1 Y) Z X)
               (TARAI (SUB1 Z) X Y) ))
       (T Y) ))
```

```
(LAP TARAI SUBR 3
     (MKFRM 1 -6)
     (PSHARG -6)
     (RCALL GREATERP)
     (JMPPOPNIL G0)
     (MKFRM 3 -14)
     (RCALL SUB1)
     (PSHARG -10)
     (PSHARG -10)
     (SCALL TARAI)
     (MKFRM 2 -14)
     (RCALL SUB1)
     (PSHARG -10)
     (PSHARG -13)
     (SCALL TARAI)
     (MKFRM 2 -14)
     (RCALL SUB1)
     (PSHARG -13)
     (PSHARG -13)
     (SCALL TARAI)
     (SCALL TARAI)
     (RETURN)
G0   (PSHARG -2)
     (RETURN))
```

図８．関数定義体とそのコンパイル例

表１において、インタープリタとコンパイラの場合を比較してみると、他のシステムでは、５～100倍の短縮が可能であるが、本システムではコンパイルしても２～３倍程度の短縮しか得られていない。（ちなみにTARAI-3を組込み関数、ＥＸＰＲ,ＳＵＢＲとした場合の実行時間はそれぞれ、14 m sec 55 m sec、26 m sec である）これはインタープリタが高速に動作しているためと考え

られるが、中間言語命令のフェッチ、及びデコード部をマイクロプログラムで構成した事によるオーバヘッド、特にフェッチによるオーバヘッドが大きいためと考えられる。そこで、このフェッチをハードウェアで構成することを考えると、現在の実行時間の10～30％の短縮が可能と思われるが、実装上の問題もあり、この早急な実現は難しいと思われる。ユーザは、メモリセルの節約、ガーベージコレクションの発生をおさえることを望む場合、コンパイラを利用するのがよいであろう。

7. 今後の応用について

本システムは昨年の1月にハードウェアとインタープリタが完成し、約一年が経過した現在、ソフトウェアの開発がほぼ完了しようとしている。しかし、まだ完成されたものではないため、特定のユーザを有していない。完成後の本システムの応用としては、数式処理、物体認識、画像処理等の分野での使用、また、MILISY、REDUCE等の応用プログラムの稼動、等が考えられる。ただ、本システムにおける使用可能なリスト要素が40K語であるため、この範囲内での使用に限定される。これからも、ＬＩＳＰ言語の使用はますます、人工知能の分野を中心にひろく普及してゆくであろうから、多種多様の応用が可能であろう。

参考文献

[1] 瀧、金田、前川：ＬＩＳＰマシンの試作－アーキテクチャとＬＩＳＰ言語の仕様－、情報処理学会論文誌、Vol.20、No.6、(Nov. 1979).

[2] 瀧、金田、前川：ＬＩＳＰマシンの試作－インタープリタの構造とシステムの評価－、情報処理学会論文誌、Vol.20、No.6、(Nov. 1979).

[3] 瀧、小林、金田、他：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、(Mar. 1979).

[4] Y.KANEDA,K.TAKI: The Experimental LISP Machine,IJCAI proceedings,(Aug. 1979).

[5] 竹内：第２回ＬＩＳＰコンテスト、情報処理、Vol.20、No.3、1979

[6] 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、J61-D、8 (Aug. 1978).

[7] 黒川：ＬＩＳＰのデータ表現、情報処理、Vol.17、No.2、1976.

[8] 長尾、中村、他：ＬＩＳＰマシンＮＫ３のアーキテクチャとそのマイクロインストラクション、信学 研、EC－77－17、(1977).

[9] J.Allen: Anatomy of LISP, McGrawHill,1978.

[10] W.Teitelman: INTERLISP Reference Manual,Xerox,(Feb. 1974).

[11] LISP User's Manual, EPICS-5-on-4,(Mar. 1978).