DATAMATION 1983年2月号 (邦訳:「プログラマ待望の強力なツールが登場」, 日経コンピュータ (1984年4月18日号).
12) Teitelman, W. and Masinter, L.: "The Interlisp Programming Environment", COMPUTER pp. 25-33 (1981年4月号).
[*Smalltalk*-80]
13) Xerox PARC Learning Reserch Group: "Special Issue on Smalltalk", Vol. 6, pp. 168-194 (1981).
14) Goldberg, A. and DaveRobson: "Smalltalk-80: The Language and Its Implementation", Addison-Wesley (1983).

(昭和58年11月14日受付)

製品例

Lisp マシン†

元吉　文男††

1. はじめに

Lisp マシンは, いわゆるワークステーションとは開発目的が若干異なっているが, 製品として見た場合には, ビットマップディスプレイによるマルチウインドウシステムやネットワークサポートなどの機能があり, ワークステーションの仲間に入れても差し支えがないようである.

記号処理用言語 Lisp は計算機に数値計算だけでなく, 構造を持ったデータを扱うように作成された言語であり, 人工知能等の研究に古くから使用されてきたが, メモリを大量に使用する, 処理速度が遅い等の理由で普及が遅れていた. そこで, この Lisp を高速に実行するパーソナル計算機が開発されることになり, まず CONS と呼ばれる Lisp マシンが MIT で開発された. 続いて CADR と呼ばれるマシンも同所で開発され, 商業化の動きもでてきた. このような中で LMI 社と Symbolics 社が Lisp マシンを製品化して販売することになった. どちらのマシンも CADR の流れを汲むもので, 似かよったものになっている.

近年, 知識情報処理の分野が注目を集めているが, この方面の記述には Lisp が適しており, Lisp を高速に実行するパーソナル Lisp マシンに対する需要は大きくなるものと思われる.

ここでは Symbolics 社のマシンについて詳しく見てみることにする. LMI 社の製品との大きな違いは, LMI 社のマシンは MC 68000 とのデュアル CPU 構成を前面に出しており, MC 68000 を使用した Unix が走るようになっている点である.

2. ハードウェア

全体の構成図を図-1 に示し, 以下の説明はこの図に添って行うことにする.

2.1 CPU 等本体

このマシンの CPU は Lisp を高速に実行させるために, 市販の CPU ではなく TTL を使って組み上げたマイクロプログラム制御方式である. マイクロ制御記憶は 112 ビットで 8 K 語でありマイクロマシンサイクルは 200 ns となっている.

主記憶は 1 語が 36 ビットで 256 M 語の仮想アドレス空間をもっており, 物理的には 7.5 M 語までの, エラー訂正機能付きのものをもつことができ, Lisp で扱う大きなアドレス空間をサポートしている. データの表現には 36 ビットのうち 8 ビットをデータタイプを示すタグとして使用しており, マイクロプログラムでデータタイプのチェックを行うことによって, 従来の計算機上で実行する場合に処理時間のかかっていたものが高速に処理できるようになっている.

入出力については MC 68000 を使用したフロントエンドプロセッサが使用されており, ディスク・ビットマップディスプレイ・キーボード等の管理を行っている. CPU 等が故障した場合には, このフロントエンドプロセッサが診断システムとして動作するようになっており, レジスタ・バス・主記憶・ディスク等にアクセスできる. また, ネットワークを介して, 他の Lisp マシンから診断することもできる. マイクロコードのデバッグの機能もこのフロントエンドプロセッサが持っており, ステップ実行やブレークポイントの設定が行えるようになっている.

† Lisp Machine by Fumio MOTOYOSHI (Information Science Div., Electrotechnical Laboratory).
†† 電子技術総合研究所

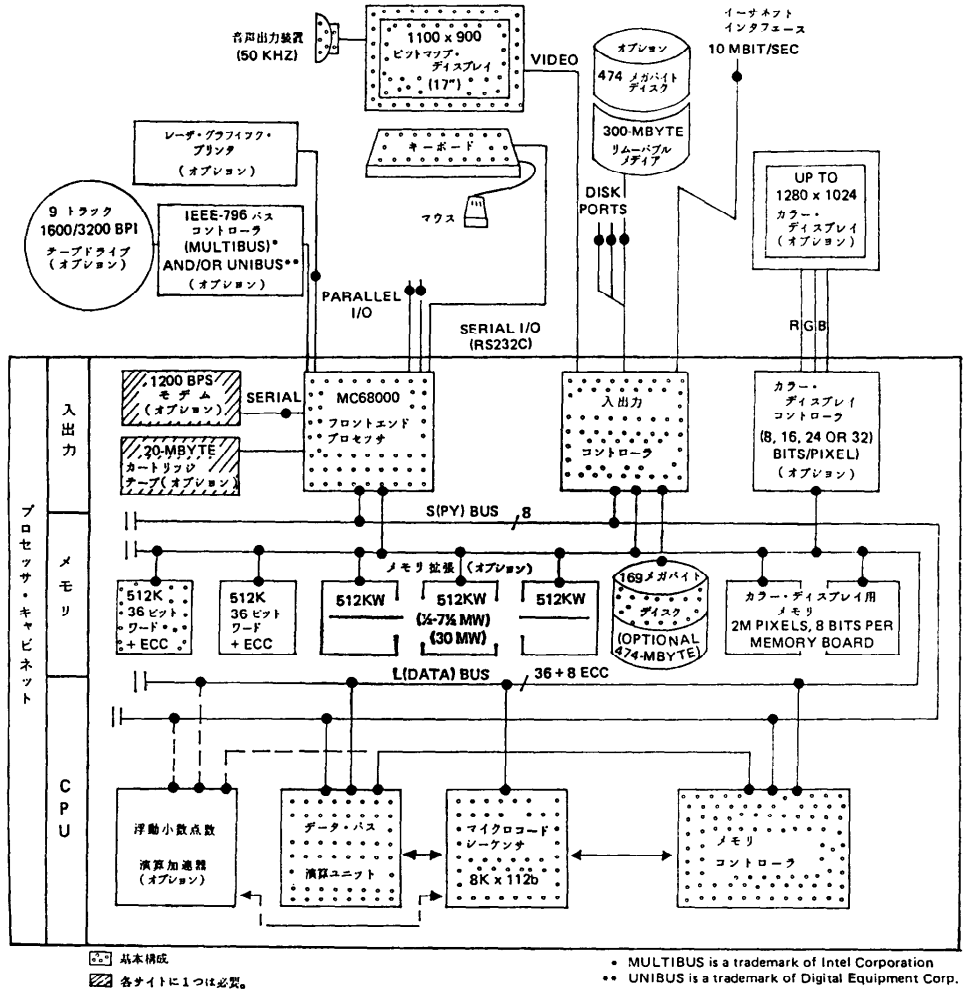

図-1 Lisp マシンのハードウェア (Symbolics 社 資料より)

2.2 ビットマップディスプレイとマウス

Lisp マシンのコンソールは高解像度モニタ，マウス，キーボードから構成されており，このモニタは，1150×900 ドットのビットマップディスプレイとなっている．

ビットマップディスプレイは様々なフォントの文字を表示できるようになっており，高速に文字表示が行えるようにマイクロコードで画面上の領域にフォントを転送している．

このディスプレイとウインドウシステムと呼ぶソフトウェアによって，画面上にメモを拡げるように何画面かを同時にオーバラップしながら表示することが可能となり，エディタと Lisp インタプリタさらにファイルシステム等を同時にマルチタスクとして実行させることもできる．

なお，Lisp マシンにはオプションではあるが，1280×1024 ドットのカラーディスプレイを接続して，1 ドット当たり 32 ビットまで使用しての多色な表示で，上に述べたような高速文字表示を行うことも可能である．もちろん，モノクロの場合もカラーの場合もグラフィックディスプレイとしての機能があり，ドット単位での制御を行うことができる．

このビットマップディスプレイを操作するためのポインティングデバイスとしてマウスがある．このマウスはメカニカルなものでボタンは 3 個ついており，ボタンの使い分けによって様々のコマンドを入力する．

Lisp マシンがこのビットマップディスプレイとマウスを持っていることが，ワークステーションの一種と考えられる要因となっている．

キーボードについては，Lisp マシン用に特別なものであり，通常のアスキ文字のほかに様々のコントロールキーが配置されている．Lisp で頻繁に使用する括弧は通常の上段にある他に下段にも配置されておりシフトをせずに括弧の入力ができるなど，マンマシンインタフェースについての配慮もなされている．

2.3 グラフィックプリンタ

Lisp マシンがハードコピー用として使用しているのはレーザグラフィックプリンタであり，これは，MC 68000 を使用して1M バイトの記憶を持ったインテリジェントタイプのものである．

フォントを前もってプリンタに転送しておき，後に文字コードを送ることによりテキストのハードコピーを好みのフォントでとるようになっている．また漢字等日本語文字の出力についても同じ方法を用いている．

画面のハードコピーもとることができるが現在はシリアルインタフェースで接続されているので，1画面のコピーに数分もかかってしまい，今後の改良が望まれる．

2.4 そ の 他

二次記憶としては 169 M バイトのハードディスクを備えており大量のデータを蓄えることができる．また 20 M バイトのカートリッジテープもオプションとして用意しており，バックアップはこのテープで行うようになっている．

また MULTIBUS* や UNIBUS** を接続してこれらのバスを使用して制御機器の操作を行うようなオプションも用意されている．

3. ソフトウェア

Lisp マシンの全体の構成を図-2 に示しておくが，ここでは特に興味を引きそうなものについて解説することにする．

3.1 Flavor と Zeta Lisp

Lisp マシンのソフトウェアは OS を含めてすべて Lisp で書かれており，しかもオブジェクトオリエンティドに書かれている．Flavor というのはこのオブジェクトのことであり Flavor に対してメッセージとパラメータを送ることによりシステムプログラムが動作している．

たとえば，ハードウェアのところで述べたマルチウインドウシステムもこの Flavor を使って書かれており，1つの Flavor が1つのウインドウを受け持って，「文字を書け」とか「画面を消去しろ」というメッセージを受け取るとそれに対応したウインドウに動作が行われるようになっている．

また，Flavor を作るときに別の Flavor をその中にとり込むことにより，とり込んだ Flavor の性質を受け継ぐことが可能となっており，階層的に抽象型データを容易に構築できるようになっている．

ウインドウを例にとって説明をすると，まず基本のウインドウとして単なるビットアレイとしての Flavor があり，その上に周りの枠をつけた Flavor を作り，文字表示をコードで実行できる Flavor をその上に作り...と次々に Flavor を積み重ねて最終的なウインドウ Flavor を構成している．ここで枠をつけるところで変化をつけて，Lisp のトップレベルで使用するウインドウとエディタで使用するウインドウの表示方法に変化をつけて見易くしたりすることも Flavor を使うと少しのプログラムで可能となっている．

Lisp マシンの Lisp は ZETALISP と呼ばれ，MIT で開発された MACLISP の改良版であり，上に述べたようなオブジェクトオリエンティドな Flavor もサポートしている．またシステムの記述にも ZETALISP を使用しているため低レベルの処理についても記述が可能であり，システム記述言語としての役割ももっている．

これらのプログラムは，マイクロプログラムがエミュレートするマクロコードの上で実行されているが，Lisp マシンにはこのコードに対するアセンブラは用意されていない．（システムプログラムもすべて Lisp で記述しコンパイルするのでアセンブラは使用されない．）また，マイクロ制御記憶も書き替え可能であるが，これに対するサポートも現在のところ一切なされていないのは残念である．

3.2 Zmacs エディタ

Lisp マシンで使用されるエディタは Zmacs と呼ばれており，これは MIT で開発されたリアルタイムのスクリーンエディタである Emacs の拡張版である．（ここでリアルタイムというのは，画面に表示されているテキストが常に現実のテキストと一致していることを意味する．）

---

* MULTIBUS は Intel Corp. の商標である．
** UNIBUS は Digital Equipment Corp. の商標である．

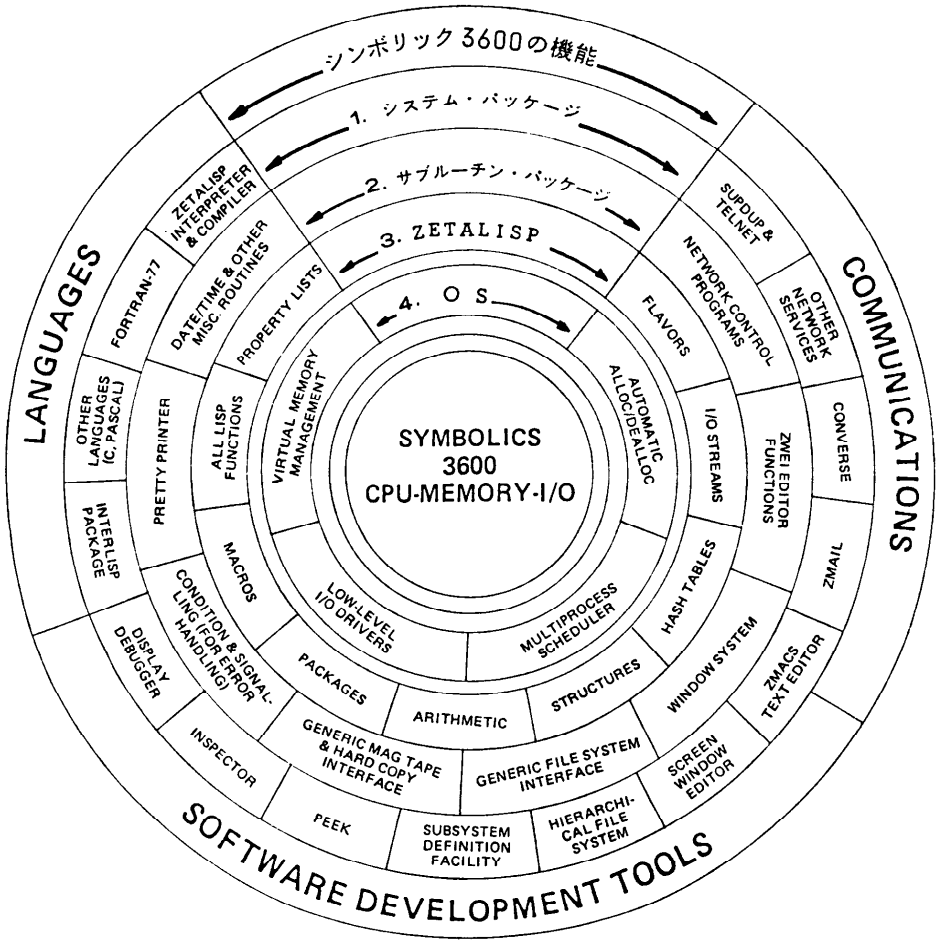

図-2 Lisp マシンのソフトウェア (Symbolics 社 資料より)

Zmacs では挿入モードとコマンドモードの区別はなく，常に1つのモードであり，通常のアスキ文字を打つとそれがそのまま挿入される．コマンドを実行させるには，別のキーと同時にコマンドを示す文字を押すことによっている．ここで別のキーにはコントロール・メタ・スーパ・ハイパなどがあり多種のコマンドの入力に使用するが，これらのキーはキーボードの両側にそれぞれ配置されており，通常の端末で入力する場合よりも入力し易い．

Zmacs が Emacs よりも拡張されている部分は，まずマウスを利用したコマンド入力である．これはカーソル移動や，テキストの範囲を指定するのに利用され，キーボードに触れずに編集作業が行える．またマウスを左端に寄せることにより，現在画面に表われている部分が全テキストのどの部分にあるかを示す棒グラフを表示する．もう1つの拡張は，エディタの中にありながら Lisp のプログラムを実行できるようにしたことである．これは，タスクを切替えずに行うのでデバッグの時には迅速に結果を見ることができる．

3.3 デバッガ

Lisp のトップレベルの入力で，入力ミスに気付いた時には，エディタと同じカーソル移動のコマンドを使用することにより，改行してしまったあとのテキストに対しても修正を行えるようになっている．またエラーが起った場合にもコントロールCと打つことにより直前に打ったテキストが表示され，それに対して修正を行うことも可能である．

エラーが起った場合には，スタックにある最初のフレームの内容を出力してデバッグモードに入り，ここでデバッグコマンドを入力するほかに，通常のS式の

評価も行えるようになっている．なお，このデバッグモードは何回でも入れ子になるようになっている．

また関数の実行の様子を知るためには，トレーサとステッパがある．このトレーサも単に関数が呼ばれた時に関数名と引数を出力するのではなく，関数が呼ばれたときに条件を調べてその時だけ出力したり，呼ばれたときに式の評価を行うことも可能となっている．ステッパはインタプリタの1ステップごとにその結果を出力し入力待ちになり，ユーザに次の対応を聞くようになっており，インタラクティブにデバッグをする場合に有効な手段である．

このほかに，データの構造を調べたり，その構造を作り変えたりする機能が備わっており，プログラムを最初から走らせずにデータを直すことも可能である．プログラムの修正もこれでも可能であるが，マルチウインドウを利用してエディタに入り，そこでソースプログラムを直してその部分だけ実行させるのが普通である．

3.4 ネットワークシステム

Lisp マシンはネットワークとしてイーサネットと同等の CHAOSnet をサポートしており，10 M ビット/秒でデータの転送を行う．主な機能としては，リモートファイル，電子メール，リモートログイン，リアルタイムメッセージ交換等がある．接続可能な計算機としては，別の Lisp マシン，VAX (VMS 又は UNIX)，DEC SYSTEM 20 等があり，これらについてはインタフェースが用意されているようである．

3.5 そ の 他

Lisp マシンでサポートされる言語は上に述べた ZETALISP の他に FORTRAN, PASCAL, C 等がある．また INTERLISP パッケージも用意されており，この Lisp を使いたい人はこれを使用するようになっている．

前にも述べたようにシステムはすべて Lisp で書かれており，そのソースプログラムも公開されているので内部を詳しく知りたい人の参考にもなるであろう．

参 考 文 献

1) Greenblatt, R.: The LISP Machine, Working paper 79, MIT AI Lab., Cambridge, Massachusetts (Nov. 1974).
2) Knight, T.: The CONS Microprocessor, Working paper 80, MIT AI Lab., Cambridge, Massachusetts (Nov. 1974).
3) Stallman, R. M.: EMACS: The Extensible, Customizable, Self-Documenting Display Editor, Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, Portland, Oregon, pp. 147-156 (June 1981).
4) Weinred, D. and Moon, D.: LISP Machine Manual, 4th Ed., Symbolics Inc., Cambridge, Massachusetts (July 1981).

(昭和58年10月18日受付)