マルチメディア通信と分散処理 31－2
（１９８６．９．２９）

ＡＩワークステーションＥＬＩＳの検討

山田康宏 大野邦夫 日比野靖 竹内郁雄
ＮＴＴ電気通信研究所

本報告は、ＡＩワークステーションＥＬＩＳに関して、基本アーキテクチャ、ハードウェア構成、ソフトウェア構成、日本語ソフト開発の支援方法等についての概要を述べる。

本装置のアーキテクチャは、当研究所が従来より開発を進めてきたＬｉｓｐマシンＥＬＩＳの基本思想を継承する。大型汎用機並の高速処理と大きな実メモリ空間をＴＳＳで共同利用することにより、大規模ソフトをグループ単位で共同開発することを可能にする。プロトタイプマシンの実績を基に、装置の小型化を図るとともに、ビットマップディスプレイを使ったウィンドウシステムを付加し、通常のワークステーションとしての機能も同時にサポートする。

日本語プログラムを作り易くするため、データタイプの共通化、漢字コードとＡＳＣＩＩコードの統一処理を行い、ＡＳＣＩＩコードを前提とした従来のプログラム上で能率良く日本語データが扱える様にする。日本語データの入力には仮名漢字変換機能が使える。

Design for AI Work Station "ELIS"

Y.YAMADA K.OHNO Y.HIBINO I.TAKEUCHI

NTT Electorical Communication Laboratories
1-2356 Take Yokosuka-City Kanagawa Japan / 3-9-11 Midori-Choh Musashino-City Tokyo Japan

This paper describes an AI workstation, and its basic architecture, hardware, software and Japanese language implementation.

The basic system architecture is almost the same as the lisp machine ELIS. Its lisp processing performance is comparable to the main frame machines. Based on the prototype machines, it enables ELIS to a small workstation. The workstation succeeds the multi-user function, which is convenient for the development of large software system by a small group. Of course, multi-window system using a bitmap display is supported. Thus this workstation can also be used as a conventional AI workstation.

In order to make the development of Japanese programs easier, datatype integration and the similar operation for both KANJI code and ASCII code were realized. Thus Japanese data can be used in the conventional ASCII programs. Useful tools are also provided for KANJI code input.

1．始めに

ＮＴＴ電気通信研究所では、従来よりＬｉｓｐ専用マシンＥＬＩＳの開発を進めている。その基本アーキテクチャについては既に報告した[1][2]。今回は、このマシンの簡素なハードウェアと高速処理性能を活かしてＡＩワークステーションとしての構想をまとめたのでここに報告する。

ハード面では、小型化、信頼性、経済性の向上を図り、静肅化により日本のオフィス環境との整合を図るとともに、独立システムとしても耐え得るだけの記憶容量と周辺装置類の接続を可能とする。ソフト面では、マルチユーザ、マルチタスクをサポートする他、核言語ＴＡＯ[3]に日本語処理機能を拡充し[4]、この上に各種ソフトウェアツール類を作成する等、日本語によるＡＩソフトを開発し易い環境を提供しようとしている。

2．開発方針

2.1 装置の特長

本装置には、以下の特長を持たせる。

（1）高速処理：　大容量のスタック、ＷＣＳ、主記憶メモリを実装し、メモリ多目的レジスタを利用してセルの先読みをする等、ＥＬＩＳアーキテクチャによる高速処理技術で汎用大型機並の性能を実現する[5]。

（2）大規模記憶：　アドレス空間は、１６Ｍセル（64bit/ｾﾙ）とする。全空間を実記憶上で使用可能とする他、仮想記憶機能も付加する。大型ソフト開発時のバージョン管理等に備え、ファイル記憶も最大１ＧＢ迄使用可能とする。

（3）利用形態：　ワークステーションとしてＬＡＮを介した各種通信機能を持つ他、汎用機並にＴＳＳ機能を持たせ、数人程度のグループ利用を可能として、計算機資源の共同利用による経済化、ファイルの共通化による大規模ソフトの開発を容易にする（図１）。

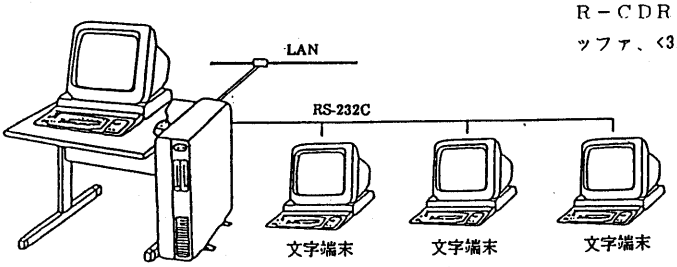


図 1　利用形態

（4）Common Lisp：　ＡＩソフト開発の共通言語の１つに成りつつある Common Lisp について、現在迄に決まっている分についてフルサポートする。

（5）日本語処理：　ＡＳＣＩＩコードと日本語コードの文字列処理を核言語ＴＡＯのレベルで統合し、従来のＡＳＣＩＩコードを基本にしたプログラムの作成環境と同様の環境で、日本語ＡＩプログラムの作成を可能にする[4]。

2.2 アーキテクチャ

上述した通り、本装置は、ＥＬＩＳの基本アーキテクチャを継承する。その要約は、以下の通りである。

（1）セル構成：　セルサイズ６４ビットの内、上位３２ビットをＣＡＲ部、下位３２ビットをＣＤＲ部とする。その各々は、８ビットのタグと２４ビットのポインタから成る（図２）。タグは、ポインタの先のセルの属性を表わす。メモリの読出／書込幅は、セル単位とするが、コンパイルされたコードは、次項で説明するレジスタをバッファにしてバイト単位に切り出す。

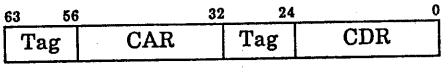


図 2 セルフォーマット

（2）メモリ多目的レジスタ：　６４ビットの汎用レジスタ（ＭＧＲ）４組を配置し、マイクロ命令のアドレスソース指定とデータ行先指定によって、<1>ＣＡＲ－ＣＤＲレジスタ、<2>コンパイルコードの命令バッファ、<3>プログラムカウンタ、<4>文字列レジスタ、等として動作させる[2]。このＭＧＲの番号及び内部のバイト位置を指定する５ビット（２ビット<ＭＧＲ番号指定>＋３ビット<バイト位置指定>）のソース先行カウンタ（ＳＤＣ）を３個設け、ＡＬＵのソースの決定と演算を並行動作させてＣＰＵの処理能率の向上を図る（図３）。

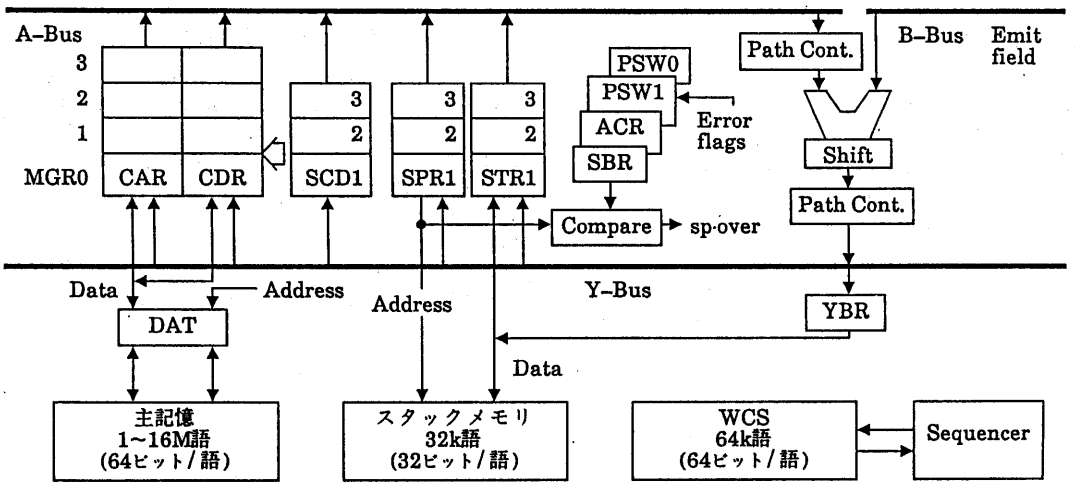


図3 CPU部ブロック図

（3）スタック構成： ３２ｋ語（３２ビット／語）構成のスタックメモリに３組のスタックポインタ（ＳＰＲ）とスタックトップレジスタ（ＳＴＲ）を設け、３本のスタックとして使う。各スタックの上限と下限は、スタック境界レジスタ（ＳＢＲ）が２ｋ語単位で管理する。

（4）マイクロ命令： マイクロ命令には３つのタイプがある（図４）。定数発生とメモリ操作とを別タイプにすることで、６４ビット長に収めている。ＡＬＵの演算は、メモリ操作、ＳＤＣの増加、定数の発生、の各々と並行処理することができる。シーケンス制御部分には、次アドレスと分岐条件を指定する[2]。

（5）シーケンス制御： マイクロ命令のフェッチと実行を１レベルのパイプラインで行うモードと、１マシンサイクル中にアドレスの決定と命令のフェッチを行なうモードがあり、プログラマが選択できる。シーケンサは、３３種類の分岐ソースを持ち、ＡＬＵ演算フラグ、タグ、スタックオーバ、ＳＤＣや内部バスの状態、外部インタフェースフラグ等に応じて分岐する。

基本マシンサイクルは、３相クロックから成る。この基本時間内に終了しない場合、例えば、スタックポインタの更新直後にスタックを演算ソースに指定した場合やメモリ操作の連続等があった場合は、１クロック単位でマシンサイクルを自動延長する。

（6）インタプリタとコンパイラの完全両立： インタプリタ、コンパイラともに関数単位で実行させて完全両立を図るとともに、全面的なマイクロプログラム制御化（ｅｖａｌと中間言語エミュレータの両方をＷＣＳ内に置く）による高速化を図る。

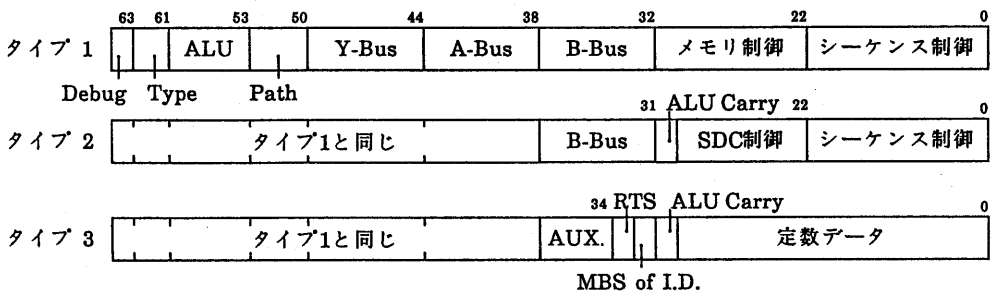


図4 マイクロ命令形式

3．装置設計

3.1 ハードウェア

（1）機能分担：　高速にリスト処理を行うＣＰＵ部分と、主として入出力処理を行うＦＥＰ部分とをＤＭＡで結合した構成とする（図5）。ＣＰＵは、６４ビット幅のローカルバス経由で主記憶及びＦＰＡ等の高速オプション回路と結び、ＦＥＰは、１６ビット幅の汎用バスを介して入出力用ハードウェアモジュール類の制御を行う。表示装置制御部やＬＡＮ制御部等の高速デバイスのソフト制御部分は、ファームウェア化する[6]。

（2）処理速度：　ＣＰＵ部の速度は速い程よいが、１処理当たりのゲート数、周辺回路素子の速度等と整合を取る必要がある。本装置のＣＰＵは、ＳＴＴＬ素子で構成していたプロトタイプマシンと同速度（ﾏｼﾝｻｲｸﾙ 180 ns）（表1）とする。

ＦＥＰ側の速度は、本装置の入出力処理速度を決める。ＦＥＰ側プロセッサには、ＭＣ６８０１０を１０ＭＨｚで使い、リアルタイムモニタで制御する。

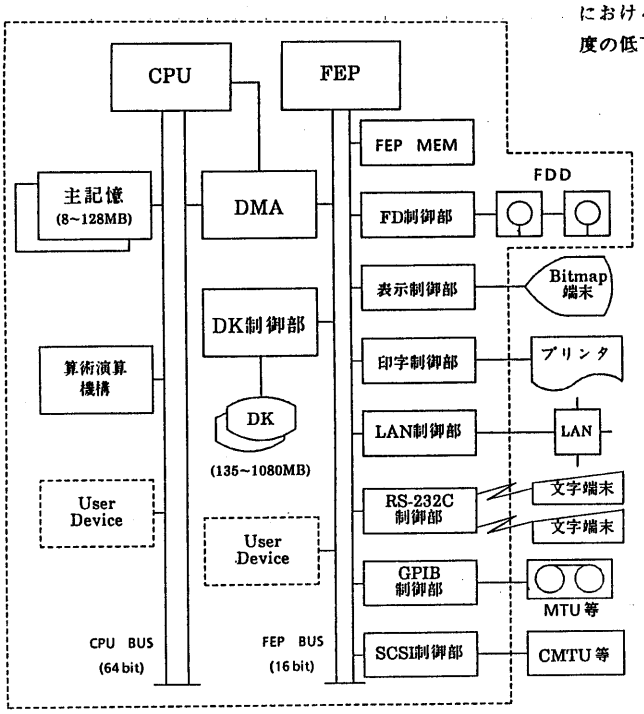


図5 ハードウェア構成

表 1　処理能力比較例

機　　種	Ｌｉｓｐ	Interpreted	Compiled
ELIS	Tao	100	(330)
Symbolics-3600	Zeta lisp	7.6	83
SIP-1100	Interlisp-D	1.6	13
VAX-8600	VAX-lisp	13	180
MV10000	KCL	8.7	88
DEC-2060	Mac lisp	37	150
3081K	LISP/VM	26	350
M-280 H	Franz lisp	60	290
M-280 H	Uti lisp	390	1400
COSMO 800/3	U lisp	11	35

注:数値は、ELISインタプリタを100とした相対値
文献[5]より Tarai-4, List-Tarai-4, String-Tarai-4, Flonum-Tarai-4, Bubble-50, Seq-100, Prl-Rev-15, BITA-5, Sort-50,TPU-1の相乗平均速度を表わす。

（3）記憶容量：　Ｌｉｓｐのメモリアクセスには偏在性が少ないので、主記憶は、１６Ｍセル（１２８ＭＢ）のアドレス空間をキャッシュを介さず直接アクセスする。全アドレス空間分の記憶容量を実装可能とし、大規模ソフト（エキスパートシステムや自動翻訳等）における仮想記憶動作（ＤＫアクセス）による処理速度の低下を防ぐ。

主記憶回路は、そのサイクル時間をＥＣＣ回路を含め、３マシンサイクル（540 ns）内に収めれば良いので、経済的なＤＲＡＭと市販ＥＣＣ回路で構成できる。

ＷＣＳは、６４ｋ語（５１２ｋＢ）すべてを常時実装しておき、マイクロ命令の充実による高速化を図る。必要ならばユーザが自己の用途に応じてマイクロ命令を追加することもできる。

3.2 ソフトウェア

（1）ソフトウェアの構成：　本装置のソフトウェアは、ＣＰＵ側とＦＥＰ側に大別できる（図6）。

ＥＬＩＳのＣＰＵ側ハードウェア上で最も効率良く動作する言語仕様は、ＴＡＯである。基本アーキテクチャをＥＬＩＳと一致させたことで、ＥＬＩＳのカーネル部、核言語ＴＡＯ、及びこの上のソフトウェア資産が継承できる。ＣＰＵ側のシステムプログラムは、核言語ＴＡＯ

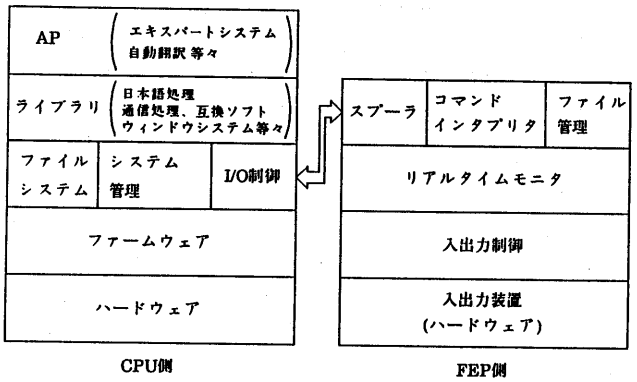


図6 ソフトウェア構成

で記述するが、特に高速処理が必要な部分はマイクロコード化する。新たに作成するソフトウェアツール類も、ＴＡＯで記述する。

ＦＥＰ側は、主としてCで記述し、高速処理が必要な部分のみアッセンブラ化する。ソフトウェアモジュール（デバイスドライバ、スプーラ、ファイルマネジャ等）を小型のリアルタイムモニタで制御する。その他、ハードウェアモジュール内にはファームウェアのマイクロプログラムがある。

（２）核言語ＴＡＯ：　本装置の核言語であるＴＡＯには、システム記述専用部分とユーザ使用可能部分とがある。ＴＡＯの基本形は、関数形のＬＩＳＰであり、Ｓ式レベルでオブジェクト指向プログラミングとロジックプログラミングのパラダイムを包含している[3]。ユーザ言語として広範な機能が使えるだけでなく、システム記述言語としてツール類を記述するにも適している。本装置のＣＰＵ側システムプログラムをＴＡＯで記述するとともに、システムを破壊する恐れのない関数はすべてユーザに開放する。

（３） Common Lisp ：　今後のＡＩの分野に於いて、共通的言語の１つになると予想される Common Lisp をフルサポートする[7]。実現方法は、以下の通りである。 <1>核言語ＴＡＯの仕様（文法、関数名、動作等）を可能な限り Common Lisp と一致させる。<2>ＴＡＯの基本設計と相容れない部分だけを互換パッケージに収容する（図７）。互換パッケージに収容した関数は、全体の約１割（５０余）であり、Common Lisp 実現に伴うオーバヘッドは殆どない。

Common Lisp は、コンパイラ重視の仕様であり、インタプリタの性能が犠牲になることはよく知られている。本装置では、関数を定義する際に前処理を施すことによって、核言語ＴＡＯの高速インタプリタとほぼ同等の高速性能を実現する。また、入出力処理関係のプログラムをマイクロコード化して、Ｓ式の入出力の高速化を図る。

（４）他言語処理：　他のマシン上で開発したプログラムを本装置上で走行を可能とし、既開発及び流通ソフト等のソフトウェア資産の活用を容易にする目的で、互換パッケージ（他の Lisp 系言語を Common Lisp と同様の手法で収容）や変換コンパイラ（Fortran、C、Prolog 系言語、等をＴＡＯに変換）を用意する。これらの言語は、本装置の上で作成した Common Lisp やＴＡＯと同一の実行環境で走行するので、ユーザプログラムのレベルで、混合して使うこともできる。

（５）エディタ：　テキストエディタは、ＥＭＡＣＳと同等の機能仕様を持たせ、日本語の入力、編集も可能とする。記述は、主にＴＡＯのオブジェクト指向部分を使ってモジュール化し、主要部をマイクロコード化して高速化を図る。

ビットマップ端末上のエディタは、文字端末上のエディタと操作の統一を図る。ビットマップ端末上では、マウスによるカーソルの移動、ポップアップメニュー、マルチフォントの使用等、ウィンドウシステムを活用した多彩な機能を利用することができる。

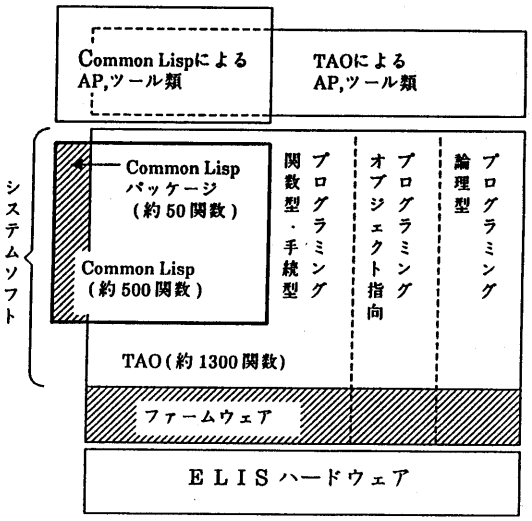


図7 Common Lispの実現

（6）ファイルシステム：　ファイルシステムの基本構造は、ＵＮＩＸと同じとする。ＤＥＣシステムに慣れたユーザの為に、ＴＯＰＳ－２０と同等の機能を持つノードを用意する。また、仮想記憶機能の実現と日本語処理における辞書ファイルやデータベース参照の能率向上のため、連続領域の割り当ても可能とする。ファイル編集のため、ファイルエディタを備える。

（7）マルチユーザ：　本装置の高速性能を活かして、ＴＳＳで数人程度の同時利用を可能にし、資源の共同利用による経済性向上と大規模ＡＩソフトの開発環境を提供することができる。Ｌｉｓｐは、メモリ消費形言語であるから、通常のＴＳＳマシンのようにユーザ毎に記憶量を固定的に割り当てると、各ユーザのメモリの制約が増大する。本装置では、セル空間を共通化して、全ユーザの使用メモリの総和が実装容量を超えない限り、仮想記憶動作による処理速度の低下がないようにする。この場合、ユーザ毎の変数の管理が必要になるので、セミグローバル変数の概念を導入する。

（8）ウインドウシステム：　ビットマップ端末を標準装備とし、マルチウィンドウとマウスを使った快適なウィンドウシステムを実現する。この端末は、１装置当たり１台とし、他ユーザは、文字端末を使用する。文字端末に替えてパソコンを疑似ビットマップ端末として使う方法も検討する。

（9）通信機能：　基本的通信手段は、ＬＡＮである(Ethernet,10Mb/s)。データリンク層（ＣＳＭＡ／ＣＤ<IEEE 802 準拠>）は、ＬＡＮ制御部のファームウェアで処理し、ＴＣＰ／ＩＰのＩＰとＩＳＯ <DIS 8473> に準拠したＩＮＰの２種類をＦＥＰ側に作る。いずれも上位プロトコルは、ＣＰＵ側にＴＡＯで記述する。

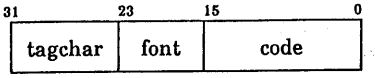
ＲＳ-２３２Ｃ制御部には、通常は複数の文字端末が接続される。ここに電話端末用変復調器を接続し、遠隔地からのＥＬＩＳ使用と Kermit を使ってのマシン相互（ＥＬＩＳ相互、ＶＡＸ－ＥＬＩＳ間等）のリモートログインやプログラム・ファイル転送を可能とする。

３.３ 文字列処理

本装置では、日本語ＡＩソフトの開発ツールとして、日本語文字とＡＳＣＩＩ文字を統合した文字列処理機能を重視する。

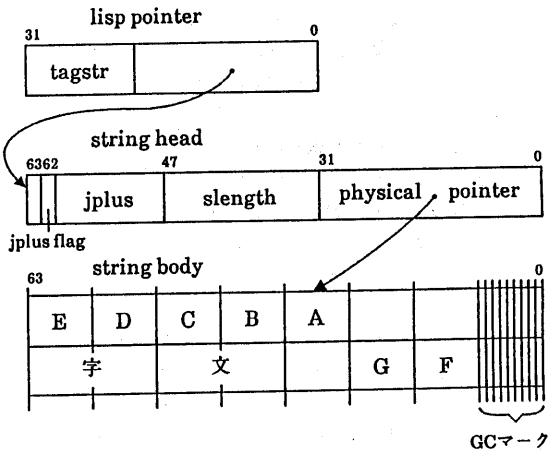
（1）文字型の統合：　日本語文字とＡＳＣＩＩ文字のデータタイプを共通化し（図８、９）、プログラマが両者を区別することなしに文字列処理ができるようにする。これにより、ＡＳＣＩＩコードを前提とした既存ＡＩプログラムの日本語化が容易になるとともに、新規作成プログラムも従来と同じ手法で開発できるようになる。

（2）メモリの節約：　日本語文字とＡＳＣＩＩ文字の混在に伴うメモリ使用量の増加を最小限にするため、マシンの内部では、ＡＳＣＩＩ文字を１バイト、日本語文字を２バイトで表わし、ＡＴＴ-ＵＮＩＸ方式に準じた区別（バイトの最上位ビットが、０ならばＡＳＣＩＩ系、１ならば漢字符号系）をする。



code は、ASCII 文字の場合 bit15-7 がすべて 0、
日本語文字の場合 bit15-0 にシフトJISが入る。
(漢字コードの各バイトの bit7 が１)

図 8 文字型の構造



string header
bit 63　gcmark　— GCのためのマークビット
bit 62　jplus flag　— 日本語文字が含まれる時に１となる
bit 61 - 48　jplus　— string の中の日本語文字の個数
bit 47 - 32　slength　— string の長さ(2 byte の日本語文字を１文字と数える)
bit 26 - 0
physical pointer　— lisp object でないものを指す pointer

図 9 文字列型の構造

（3）処理の高速化：　上記符号系の判別を水平型マイクロ命令で処理することにより、日本語文字との混在処理に伴うＡＳＣＩＩ文字処理のオーバヘッドを殆どなくすことができる。

2.2（2）で述べたメモリ多目的レジスタ（ＭＧＲ）は、文字列のキャッシュレジスタとして使うと高速化に役立つ。ＭＧＲは、６４ビット幅でメモリと読み書きし、ＡＬＵとはＳＤＣをインデックスレジスタとして、３２／１６／８ビットの任意の幅で読み書きできる。文字列処理では、３２ビット幅で自動インクリメントさせ、ＭＧＲ内の文字列を高速にたどることができる（図１０）。

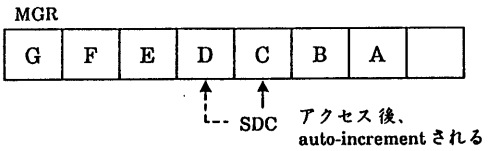


図 10 SDCを用いた文字列のアクセス

（4）速度比較：　文字列処理の速度比較をする目的で、文献［5］におけるベンチマークプログラムの内、関数呼び出しを中心にした Trai-4 と文字列処理を中心にした String-Tarai-4 の速度比較の例を表２に示した。本装置は、リスト処理同様文字列処理に関しても大型機並の性能を持つことが判る。String-Tarai-4は、日本語文字を含まないので、日本語処理能力を直接比較したことにはならないが、本方式は、日本語文字が混じっても原理的に速度の低下が殆どないので、凡その目安にはなる。

表 2　Tarai-4とString-Tarai-4の速度比較（単位はmsec）

Ｌｉｓｐ	機　種	Tarai-4	String-Tarai-4	速度比
Tao *	ELIS	628	1780	3
Zeta lisp	Symbolics-3600	119	25167	211
Interlisp-D	SIP-1100	990	64006	65
KCL	MV10000	510	8431	17
Zata lisp-plus	LAMBDA	682	25250	37
LISP/VM	3081K	137	838	6
Franz lisp	M-280 H	71	2984	42
Uti lisp	M-280 H	15	350	23
U lisp	COSMO 800/3	6399	12598	2

* はInterpreted codeの速度それ以外は、Compiled codeの速度

（5）日本語入力：　日本語データの入力には、ＥＬＩＳ上の高度な仮名漢字変換プログラムの他、パソコン上の同様の機能を使って入力することもできるようにツールソフト類の用意をする。

4．あとがき

ＥＬＩＳアーキテクチャに基づくＡＩワークステーションの所要機能、実現方法、装置構成についての構想を述べた。本装置は、従来の同種ワークステーションに比べ、小型で経済的でありながら大型汎用機並の高速処理性能を持ち、大規模ＡＩソフトの走行に適しており、ＴＳＳによる複数ユーザの同時使用もできる。また、スタンドアロン使用にも耐えられるだけの記憶容量の収容と周辺装置の接続が可能である。特に、日本語のＡＩプログラム作成環境を意識してソフトウェアの整備を行っている。

本研究を進めるにあたり、御指導、御協力を頂いている、複合通信研究所松田所長、宅内機器研究部山崎部長、小森主席研究員、入力装置研究室酒井室長、神尾、杉村、大井、杉山、岸田、基礎研究所加藤所長、情報通信基礎研究部島田部長、第二研究室渡辺、大里、奥乃、今田、村上、梅村、天海の各位に深謝致します。

［文献］

[1] 日比野,渡辺,大里 "Lispﾏｼﾝ ELISの基本設計" 情処,記処研資 12-15,1980

[2] 日比野,渡辺,大里 "Lispﾏｼﾝ ELISのｱｰｷﾃｸﾁｭｱ:ﾒﾓﾘﾚｼﾞｽﾀの汎用化とその効果" 情処,記処研資 24-3,1983

[3] 竹内,奥乃,大里 "Lispﾏｼﾝ ELIS上の新しいLisp TAO" 情処,記処研資 20-5,1982

[4] 杉村,奥乃,竹内 "TAOにおける日本語文字列処理" 情処,記処研資 36-4,1986

[5] 奥乃 "The Report of Third Lisp Contest and First Prolog Contest" 情処,記処研資 33-4,1985

[6] 荒川,徳田,鈴木,山田 "事業所分散処理装置" 通研実報 35, No.3,pp311-220,1986

[7] 竹内 "TAO/ELIS上での Common Lisp の実現" 情処,記処研資 37-7,1986