記　号　処　理　23－1
（１９８３．３．１８）

ＡＬＰＨＡ－高性能ＬＩＳＰマシン

服部　彰　　林　弘　　秋元　晴雄　　丹羽　雅司　　品川　明雄
篠木　剛　　木村　康則　　佐藤　公則
（富士通研究所）

１．はじめに

我々は，高級言語マシンアーキテクチャの研究と大容量高速なLISP処理環境の提供を目的として，Ｍシリーズ計算機のバックエンド型LISPマシン「ＡＬＰＨＡ」を開発したので報告する。

ALPHA の主な特徴としては，次のものがあげられる。

（１）Ｍシリーズ計算機のバックエンド型マシン
LISPマシンの高速性とその占有使用により，汎用機のTSS ユーザのレスポンスタイムを大幅に短縮すると共に，ホストの豊富なソフトウエア環境（大容量ファイル，ライブラリ等）を利用できる。

（２）仮想マルチスタック〔１〕
マルチプロセスのための複数の仮想スタックをサポートするハードウェアスタックを備えている。

（３）大容量記憶向きの実時間ガーベジコレクタ〔１〕
部分空間単位に不要セルの回収を行い，大容量仮想記憶でも実時間処理ができるガーベジコレクタ

（４）言語仕様は，Ｍシリーズ上のUTILISP 〔２〕とコンパティブル

以下，２節ではシステム構成，３節では言語処理系，４節では入出力処理系，５節ではハードウェア，６節では性能，７節では今後の予定について述べる。

２．システム構成

図.1にシステム構成を示す。LISPマシンALPHA は，専用のアダプタを介してM-180 ⅡのBMC (Block Multiplexer Channel）に結合されており，M-180 Ⅱをホストマシンとするバックエンド型マシンである。その入出力デバイスはM -180Ⅱ側に接続されており，M上の制御プログラムに依頼してユーザ端末，データファイル，ページングDASD等へアクセスする。

ALPHA のハードウェアは，LISP処理向きアーキテクチャをもつCPU，主記憶MSU，ALPHA を高い転送速度で結合する専用ADAPTER 〔３〕，ALPHA のIMPL，診断等を行なうSVP (Service Processor) から構成されている。

ALPHA 側にはインタプリタ，コンパイラ，標準関数，ガーベジコレクタ等の言語処理系の他，M-180 Ⅱから送られてきたキャラクタ列をLISPオブジェクトへ変換したり，その逆を行なうREAD／PRINT 関数等の入出力関数がある。

M-180 Ⅱ側には，システム共通の処理（アダプタへの入出力処理，ALPHA からの要求の解析と分配，LISPシステムファイルやページングDASDやコンソールへのアクセス）を行なうALPHA 制御システムと，各ユーザ毎の処理（ユーザ端末，データファイルへのアクセス等）を行なうLISPコマンドプロセッサがある。〔４〕

ALPHA をユーザが使うには，OSⅣ／F4のTSS セッションを開設し，LISPセッション開設要求コマンドを入力すると，上記のコマンドプロセッサが起動されてALPHA と論理的に結合される。以後，端末ユーザはALPHA 上でUTILISP を会話的に使うことができる。

入出力処理の詳しい動作については，４節で述べる。

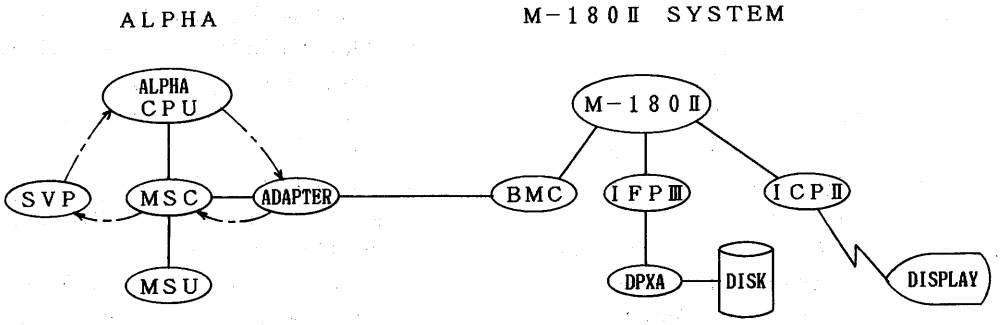


図.1　ＬＩＳＰ処理システムの構成

3．言語処理系

ALPHA は，インタプリタ，標準関数，ガーベジコレクタ等をファームウェア化し，ハードウェアスタックやタイプ判定機能を有効に使って性能向上をはかっている。

3.1 データ型とメモリマップ

ALPHA のLISP言語UTILISP のデータ型には，FIX-SHORT／LONG，FLOAT-SINGLE／DOUBLE，STREAM，CODE，REFERENCE，VECTOR，STRING，SYMBOL，LIST がある。ALPHA の主記憶のメモリマップを図.2に示す。

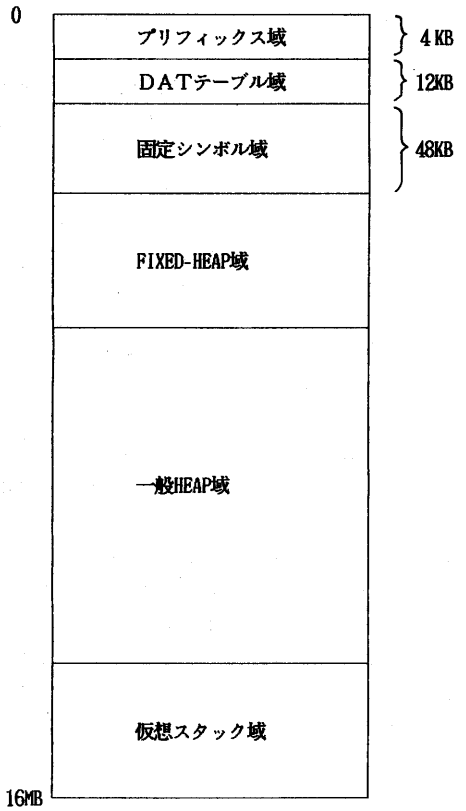


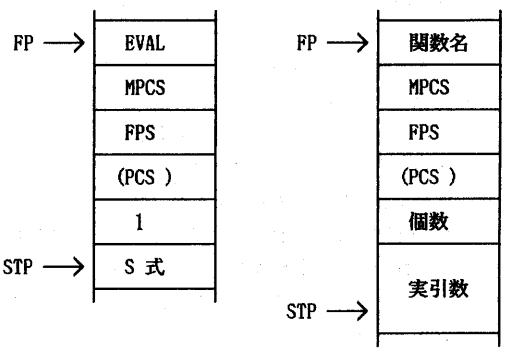
図.2　ＡＬＰＨＡのメモリマップ

プリフィックス域；ハードウェア使用域
DAT テーブル域　；仮想記憶用の変換テーブル域
固定シンボル域　；システムで使うシンボル域
FIXED-HEAP域　　；CODE，LCCW，LCCW用パラメータ，入出力バッファ
一般HEAP域　　；FIX-SHORT ／LONG，FLOAT-SINGLE／DOUBLE，LIST，REFERENCE，VECTOR，STRING，STREAM，一般シンボル
仮想スタック域　；各プロセスの仮想スタック域

現在は，移動型ガーベジコレクタを用いており，一般HEAP域がその回収対象である。

3.2 インタプリタ

インタプリタはEVAL，FUNCALL とPROG，COND等の17種の特殊形式から構成されている。　関数のタイプとしては，MSUBR，MRSUBR，SUBR，EXPR がある。図.3にスタックフレームの基本構造を示す。FP（Frame Pointer ）が示す語にはこのフレームの関数名，MPCS（Micro Program Counter Save）域にはマイクロリターンアドレス，FPS 域には上位関数フレームのFPを格納している。PCS 域にはこの関数がSUBRの場合にマクロリターンアドレスを格納している。



（a）EVALフレーム　　（b）一般のフレーム

図.3　スタックフレームの基本構造

以下，各関数タイプ及び特殊形式の処理について説明する。

（1）MSUBR は引数個数が一定で，他の関数を呼ぶことのないマイクロ関数である。インタプリタはフレームは作らず，マイクロ専用のスタックを使ってMSUBR 本体へMicro Branch&Linkする。

（2）MRSUBRは引数個数が可変か，他の関数を呼ぶことのあるマイクロ関数である。図.3（b ）のようにEVALフレームを流用してMRSUBR本体へMicro Branchする。

（3）PROG，COND等の特殊形式の場合，EVAL は引数を評価せず，そのままそれぞれの特殊形式処理ルーチンへMicro Branchする。

（4）SUBRはLAMBDA文をコンパイルしたものである。PCS 域へ現在のPC（マシン命令のProgram Counter ）を格納し，EVALフレームを流用してSUBR本体へMicro Branchする。

（5）EXPRはユーザ定義関数のことで，EVALが引数評価の他，FUNCALL が本来行なうShallow-Bind，Bodyの評価，Shallow-Unbindも行い，処理の高速化をはかっている。

インタプリタの総ステップ数は，エラー処理も含めて約1.5 K ステップ，マイクロ組込関数は現在のところ140 個約 3 Kステップである。

3.3 マシン命令

ALPHA のマシン命令は基本的にはスタックマシンタイプのものであり，命令長は2，4，6 Byteの3種類がある。表.1にマシン命令のタイプと個数を示す。ALPHA は標準関数をほとんどファーム化しているので，それらを使った方がインライン展開するより高速な場合にはコンパイルコードからもそれらを使う。そのため，各マイクロ関数に対応する専用のマシン命令を備え，一般のマシン命令と同様のディスパッチ機構でマイクロ組込関数を起動できるようにしている。一方，LAMBDA文，MACRO 文，特殊形式，及び関数引数を扱う関数はマシン命令列へコンパイルして高速化をはかっている。

命令タイプ	個数
LOAD関係	10
STORE関係	9
BRANCH関係	8
CALL／RETURN関係	8
BIND関係	5
STACK関係	5
MSUBR関数関係	73
MRSUBR関数関係	34
CATCH	1

表.1 マシン命令のタイプと個数

特徴的なマシン命令としては，TCALL，TMRCALL，CATCH がある。TCALL，TRCALL はそれぞれ，現フレームを流用してSUBR関数，MRSUBR関数を終端コールする命令で，通常のCALL命令の機能に加えて，現フレーム内の旧引数や旧ローカル変数を除去し，新引数を所定の位置へ移動する機能をもっている。そのため，次に示すLAP コードのようにフレーム作成命令が省略でき，関数コール／リターン処理が速くなる。

```
（MKP    関数名）
＜ 引数の評価code ＞      ＜ 引数の評価code ＞
（CALL   引数個数）        （TCALL  n   関数名）
```

（a）通常のコール手順　　（b）終端コール手順

CATCH は非Local-Exitに使う命令である。前述したように，特殊形式は一般にマシン命令列に展開されるが，CATCH 形式だけは専用の命令が用意されている。この命令はCATCH フレームを作成し，そのPCS 域へこのCATCH 形式に後続する命令列の先頭アドレスを格納する。その後，THROW 命令や関数THROW 又はRTN 命令でこのフレームへ制御が戻ってくる。これにより，非Local-Exit処理が完全にコンパイルされる。以下に，CATCH 形式のLAP コードの一般形を示す。

```
                          ＜ イの code ＞
＜ イ ＞                   （CATCH  label）
（CATCH  tag, args）       ＜ tagの code ＞
＜ ロ ＞                   ＜arg1～n-1 のcode＞
                           （POP   n-1）
                           ＜ argn のcode＞
                           （ RTN  ）
                    label：＜ ロの code ＞
```

（a）ソースの一般形　　（b）LAP コードの一般形

3.4 コンパイルコード

SUBRは次の要素から構成されており，図.4にメモリ上での構造を示す。

Branch-Block ；引数個数の不足をチェックするコード
Code ；コード本体
Quote-Vector ；このコードが参照するS式
Variable-Table；シンボルを直接参照している命令オペランドの位置を示すテーブル

ALPHA では高速化のため，マシン命令からQuote-Vectorを経由せずに大域変数やコールする関数名を直接参照している。そこで，ガーベジコレクタのためにこのVariable-Tableを用意している。

上記のコード本体は次のような手順になっており，図.5に示すスタックフレーム上で動作する。

（1）デフォルト引数域のスペース確保
（2）Special 宣言された正規引数のShallow-Bind
（3）デフォルト引数初期値の評価とBind
（4）Lambda-Body の評価
（5）リターン
（6）エラー処理

上記（1）は対応する実引数がなかったデフォルト引数のためのスタック領域（m＋n－j語）を確保するコードであり，（3）においてそれらの初期値を求め，Special宣言の有無によりShallow ／Local-Bindしている。（4）において，Lambda-Body が1個か，又は最後のLambda-Bodyの場合には一般（関数がMRSUBR，SUBR又はEXPRで主Lambda変数にSpecial 宣言がない場合）に前述の終端コールのコードになる。その場合には，（5）のリターン手順は不要になる。

図.5は，正規引数n個のうちk個，デフォルト引数m個のうち1個がそれぞれスペシャル宣言され，実引数がj個の場合のSUBRのスタックフレームを示す。EP（Environment-Pointer ）は最近のShallow-Bindブロック（Shallow-Bindされた変数名とその旧値の対からなるブロック）を示し，EPS は1 つ以前のSallow-Bind ブロックへのリンクである。

変数の参照については，ローカル変数はFP相対でスタックアクセスし，大域変数は絶対アドレスで直接シンボルを参照する。

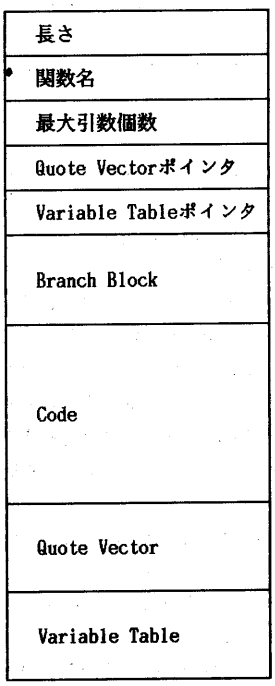


図.4 SUBRコードの構造

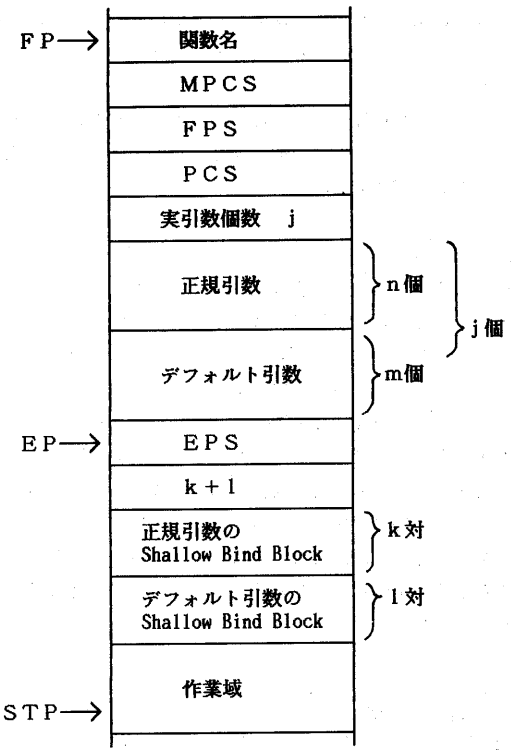


図.5 SUBRのスタックフレーム

4．入出力処理系

UTILISP では入出力処理はストリームに対して行なわれる。外部ファイルの入出力処理を例として，その手順を以下に説明する。

4.1 ＡＬＰＨＡ側の処理

（1）データセットとDD名の結合

基本的にはTSS のALLOCATEコマンドでM-180 Ⅱ側で行なうが，関数ALLOC によりALPHA 側からM-180 Ⅱへ依頼することもできる。

（2）ストリームとDD名の結合

関数STREAMにより，図.6のようなストリームを作成する。

（3）ストリームのオープン

関数INOPEN又はOUTOPEN により，図.7のようにLCCWとＩ／Ｏバッファを FIXED-HEAP 域から獲得してストリームへ結合し，M-180 ⅡへDD名データセットのオープンを依頼する。

（4）データ転送

関数TYI，TYO等により，Ｉ／Ｏバッファを介してM-180 Ⅱとデータ転送する。

（5）ストリームのクローズ

関数CLOSE により，LCCWとＩ／Ｏバッファを解放し，M-180 ⅡにDD名データセットのクローズを依頼する。

ユーザ端末の場合には，（4）のデータ転送だけでよい。

上記の処理において，TSS コマンドの実行，データセットのオープン／クローズ，データ転送等の具体的Ｉ／Ｏ処理は，LCCW（図.8参照）にその要求をセットしてStart Ｉ／Ｏを発行することによりアダプタ経由でM上の制御システムに依頼される。

バイト	
0	長さ
4	バッファ内の現在位置
8	バッファの先頭
12	バッファの終端
16	オープンのモード
20	デバイスのタイプ
24	DD名
28	最大レコード長
32	論理機番
36	LCCW キュー
60	予備

図.6 ストリームの構造

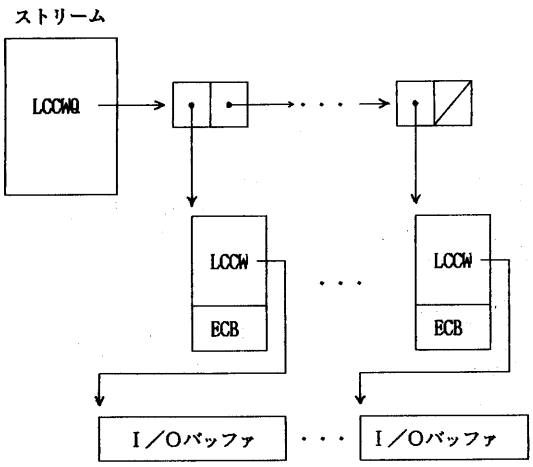


図.7 ストリーム，ＬＣＣＷ，Ｉ／Ｏバッファの結合

バイト			
0	制御フラグ		論理機番
4	送出データ長		受信データ長
8		送出データアドレス	
12		受信データアドレス	
16	実送出データ長		実受信データ長
20	終了情報		

図.8 ＬＣＣＷの構造

4.2 Ｍ－１８０Ⅱ側の処理

ALPHA 制御システムはALPHA からＩ／Ｏ要求を受取ると，そのLCCWを解析し，システム共通の処理（ページングDASDのアクセス等）は自身のサブタスクで処理するが，ユーザ固有の処理はLISPコマンドプロセッサに処理を依頼する。（図.9参照）

LISPコマンドプロセッサはLCCWに従って次のような処理を行なう。

（1）ユーザ端末への入出力処理

（2）外部ファイルへの入出力処理

（3）TSS コマンドの実行

（4）M上のプログラムの実行

これらの処理が終了すると，LISPコマンドプロセッサは終了情報をLCCWへ設定して，制御システム経由でALPHA に割込み，処理を終了する。

上記のように，Ｉ／Ｏ処理をLCCWという高いレベルで分割して処理することにより，ALPHA 側のＩ／Ｏ処理のオーバヘッドが大幅に減少し，ALPHA のLISP処理の高速性を更に生かすことができる。

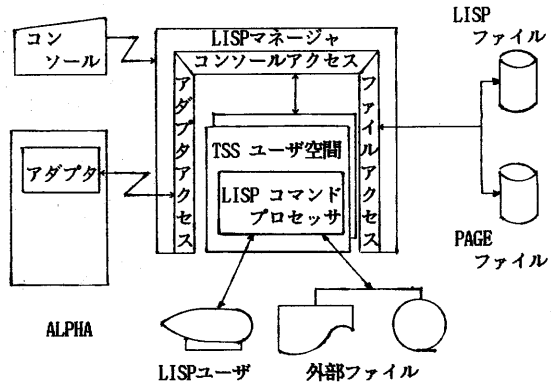


図.9 Ｍ－１８０Ⅱ上のＡＬＰＨＡ制御システム

5．ハードウェア

表.2（最後のページ）にハードウェア諸元を示す。又，図.10-a は CPUの主処理部，図.10-b は CPUのマイクロシーケンス制御部を示す。CPU の主処理部には３本のバス（A-BUS，B-BUS，Y-BUS）があり，各々32 bit幅である。又，この主処理部と乗除算器，入出力インタフェース，仮想スタックタグ，PSW，タイマー，SVP-インタフェース等の機能ブロック（図.10-a のEXTR）を結ぶ32 bit幅のEXT-BUS がある。

CPU はスタックアクセスとALU ／SHIFTER 演算を１マイクロ命令サイクルで実行することができ，この主ルートを高速化するためにスタックポインタレジスタを，PTR １個にした。そこで，複数のポインタを扱う際のオーバヘッドを減らすため，PTR 退避用レジスタPTRSを設け，主処理部の制御とは独立にPTR とPTRSのスワップ操作を可能にしている。スタックはアクセスタイム35nsの 4 Kbit メモリ素子を用いている。

図.10-b において，OPM （OP code Memory）はマシン命令実行マイクロルーチンへのディスパッチメモリである。BRTN レジスタは，MSUBR ,MRSUBR の起動と，MSUBR 以外の関数のリターンに使われる。MSUBR はマイクロスタックをポップアップすることによりリターンする。

ALPHA は，処理部でのマイクロ命令の実行とシーケンス制御部での次マイクロ命令フェッチが並行して行なわれ，その上，条件付分岐の場合でも空サイクルなしに次マイクロ命令の実行が開始される。

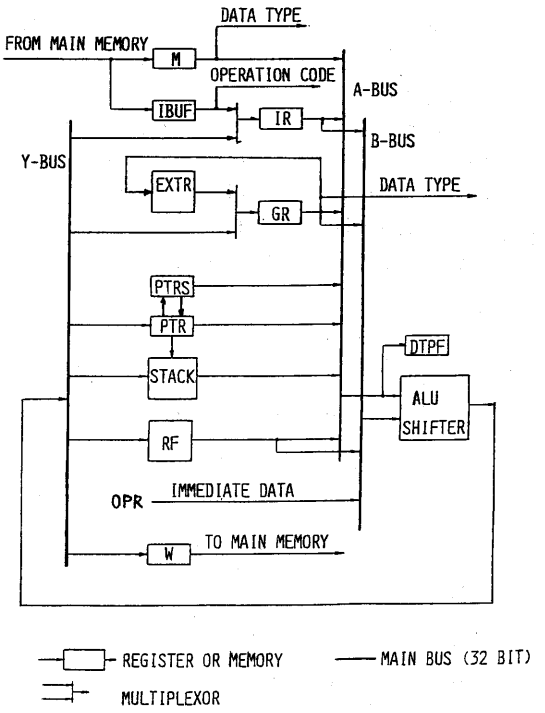


図.10-a　CPU主処理部の構成

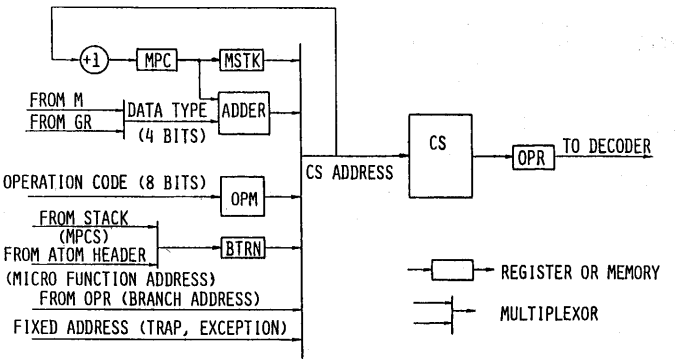


図.10-b　マイクロシーケンス制御部

主なハードウェア機能について以下に述べる。

（1）マルチプロセス向きの仮想スタック〔1〕

複数のプロセス間でのハードウェアスタックのスワッピングオーバヘッドを減らすために，ALPHA は仮想スタック方式を採用している。

図.11 にPTR 回路と論一実スタックアドレス変換回路を示す。新しい論理スタックアドレスが，PTR へ格納される時やPTR が更新される時，それを含むブロックがハードウェアスタックに存在するか否がチェックされる。その新しいアドレスの論理ブロック番号LNがPTR 中のLNと比較される。もし両者が不一致ならば，PTR の有効表示フラグVがリセットされ，これを用いてスタックアクセスするとトラップが発生する。そこでトラップハンドラはアドレス変換回路によりLNを対応する物理ブロック番号PNへ変換してPTR へ格納し，同時に有効表示フラグVがセットされる。スタックタグの各エントリはハードウェアスタックの各ブロックの論理スタック番号（プロセスに対応）LSN と論理ブロック番号LNを保持している。上記アドレス変換において，対応するブロックが不在の場合には，LRU 回路により，あまり使われていないブロックが割付られ，トラップハンドラはそのブロックのスワップ処理を行なう。

（2）データ型判定

スタック又は主記憶から読出したデータのタイプによってマルチウェイ分岐する機能をもつ。更に，A-BUSを通ったデータによりデータ型フラグをセットし，そのフラグにより条件分岐する機能ももっている。

（3）マシン命令サポート

6 Byte長のマシン命令先取りバッファを備え，マシン命令を命令レジスタへロードするのと並行して，その実行ルーチンへ分岐するディスパッチ機構をもつ。

（4）ストリング処理サポート

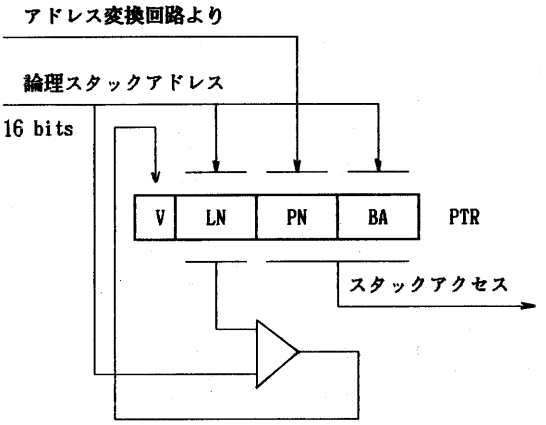
テキスト処理等にLISP言語を使うためにはストリング処理の効率も重要である。そこで，メモリ読出／書込データをアドレスバウンダリによりアラインする機構をもつ。

（5）仮想記憶サポート

MSC はDAT オペレーション機構と256 エントリのTLB をもち，CPU はTLB フォールト，ページクロスの検出機構をもっている。

（6）環境の退避，復旧の完全性

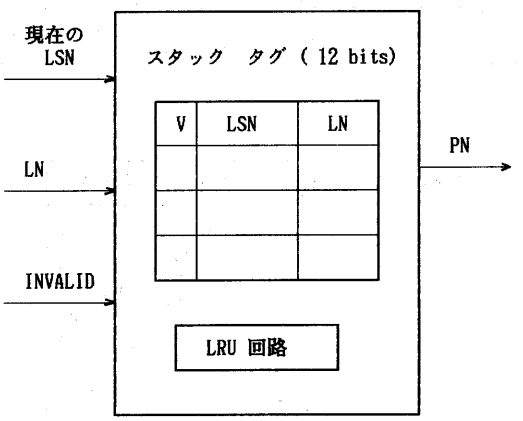
インタプリタや組込関数をファーム化した高級言語マシンではマイクロ命令レベルで割込条件が発生する。そのため，CPU 内の全てのマイクロアクセス可能レジスタとマイクロスタック（ポインタも含む）の退避，復旧が可能になっている。



ブロック番号の一致チェック

V ：有効表示フラグ　（1 bit ）
LN：論理ブロック番号（5 bits ）
PN：物理ブロック番号（2 bits ）
BA：ブロック内変位　（11 bits ）

図.11-a スタックポインタ回路



V　：有効表示ビット　（1 bit ）
LSN ：論理スタック番号（6 bits ）
LN　：論理ブロック番号（5 bits ）

図.11-b スタックアドレス変換回路

マイクロ命令	CS容量 サイクルタイム 種類 マイクロスタック	48bit ×16KW 152ns 5種類（演算，分岐，回路制御，即値，メモリアクセス） 2語
主記憶	実記憶 アクセスタイム データ幅 仮想記憶 DAT機構	4～8MB 456 ～608 ns 4Byte 16MB TLB 256 エントリ
スタック	論理スタック容量 ハードウェアスタック 語幅 スワッピング マルチプロセス機構	64KW 8KW 4Byte 2KWブロック単位
レジスタ	汎用レジスタ 命令レジスタ レジスタファイル	4Byte 4Byte 4Byte×16W
機械命令	命令長 命令の種類 命令先取りバッファ	2，4，6Byte 約 160種類 6Byte
乗除算器	単，倍精度乗除算	

表.2　ハードウェア諸元

6．性能

インタプリタによる第2回LISPコンテストの関数の実行結果を表.3に示す。いずれも,GC の時間は含んでいない。この結果では，ALPHA はM-180 ⅡのUTILISP にくらべ1.1～1.4 倍，DEC-2060のMACLISP に比べ 5.3 ～5.5 倍の性能である。

関数	実行時間	関数	実行時間
BITA -5	15	TARAI-3	44
-6	51	-4	824
-7	176	-5	22.437
-8	614	-6	824.377
BITB -5	4	TPU -1	644
-6	12	-2	2.714
-7	36	-3	1.056
-8	115	-4	1.382
-9	387	-5	189
SORT-20	62	-6	5.512
-40	228	-7	1.045
-60	364	-8	995
-80	553	-9	677
-100	709		

表.3　ＬＩＳＰコンテストの実行結果（単位：ms）

7．おわりに

我々は，Mシリーズ計算機のバックエンド型LISPマシンALPHA を開発した。ALPHA は現在インタプリタで安定に動作しており，今後，以下のことを行なっていく。

（1）性能評価
- 現在作成中のマシン命令，コンパイラの性能評価
- 現在は一括処理タイプのガーベジコレクタを用いているが，更にページ単位の回収を行なう実時間ガーベジコレクタを作成し，評価する。

（2）仮想記憶
- 上記実時間ガーベジコレクタと連係動作するページングスーパバイザの作成

（3）マルチプロセス機能
- 実時間処理や画面制御のユーザインタフェイス改善のため，仮想マルチスタック機構を使ったマルチプロセス機能を作成，評価する。

（4）直結のディスプレイ端末
- 柔軟な画面制御のため，ALPHA にディスプレイ端末を直接つなぎ，フロントエンド化をはかる。

謝辞

日頃御指導いただく棚橋情報処理研究部長に深謝いたします。又，ALPHA 制御システムの作成に多大な援助をいただいた小野氏，アダプタの設計に御援助いただいた上原ソフト1研室長，川戸氏，斎藤氏，広瀬氏に感謝します。

8．参考文献

〔1〕服部，篠木，品川，林：高速リスト処理に適したアーキテクチャについて，　情報処理学会　記号処理研究会資料18-9 (1982)．

〔2〕Chikayama,T ：“UTILISP MANUAL”，Tokyo University report METR 81-6.

〔3〕秋元，林，広瀬：LISPマシンALPHA のアダプタ，情報処理学会第26回（昭58前期）全国大会予稿．

〔4〕佐藤，秋元，品川，小野：LISPマシンALPHA の制御システム，　情報処理学会第26回（昭58前期）全国大会予稿．

〔5〕瀧，金田，前川：LISPマシンの試作，　情報処理学会論文誌 Nov.1979.

〔6〕後藤，井田，相馬：記号処理向き計算機FLATの構想，情報処理学会　記号処理研究会資料1-1 (1977)．

〔7〕日比野，渡辺，大里：LISPマシンELISの基本設計，情報処理学会　記号処理研究会資料12-15 (1980)