TR-052

パーソナル逐次型推論マシン
ＰＳＩのハードウェア設計

瀧　和夫、横田　実、山本　明
西川　宏、内田俊一

1984 3

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

パーソナル逐次型推論マシンPSI のハードウェア設計

渥　和男．横田　実．山本　明．西川　宏．内田　俊一
（（財）新世代コンピュータ技術開発機構）

1．はじめに

新しい計算機応用分野を構築するために述語論理型言語を用いる動きが活発化しているが，既存の計算機システムでは，処理能力，言語サポート，プログラム開発環境のサポート，実験・評価のための柔軟性などに関して不十分な点が多い．これらの点をカバーする強力なソフトウェア研究開発ツールを用意するために，第五世代コンピュータシステム研究開発プロジェクトの一環として，ICOTを中心に新しい述語論理型言語実行用の専用マシンであるパーソナル逐次型推論マシン（Personal Sequential Inference Machine ：以下PSI と略す）の開発を進めている［Uchida 83 ］［Yokota 83-1 ］［Nishikawa 83-1］［横田　83-2］［渥　83］［西川　83-2］［山本　83］［西川　83-3］．

PSI の開発に当り，ソフトウェア開発ツールとして十分に効率の良い述語論理型プログラムの開発環境をサポートするために，次のような目標仕様を掲げた．

(1)　PSI のシステム記述言語である論理型プログラミング言語 ESP (Extended Self-contained Prolog) ［Chikayama 83-1］［Chikayama 83-2］およびそのサブセットで機械語の機能仕様を規定するKL0 （核言語第0版）［Chikayama 84-1］を効率良く実行できること．

(2)　PSI 用に新規開発するプログラミングシステム，オペレーティングシステム(SIMPOS)［Hattori 83］を効率良くサポートするマシンアーキテクチャを有すること．

(3)　実用的規模の問題を解くために十分な記憶容量と実行速度を備えること．具体的には，DEC-2060上のProlog［Bowen 81］のコンパイラ版に比べて，記憶容量で64倍の最大16M 語，速度では同等の30K LIPS（Logical Inferenve Per Second）程度を備えること．

(4)　高度なマンマシンインタフェース機能を提供するため，ビットマップディスプレイやマウスなどの対話型入出力デバイスを備えること．

(5) PSI 相互間での通信と，資源の共通利用を可能とするローカルエリアネットワーク(LAN) システムを備えること．

(6)　個人で占有して使用できる程度の大きさで，コストパフォーマンスの点からも実用的であること．

(7)　ツールとして安定して使用していけるだけの信頼性を有すること．

(8)　早期に利用可能であること．

また開発ツールとしての役割の他に，述語論理型言語の高速実行を行なうアーキテクチャの研究用マシンおよび，述語論理型言語の実行メカニズムの実験・評価用マシンとして活用できるよう，次の目標仕様を設定した．

(1)　ユニフィケーションの高速化機構の研究成果を盛りこむこと．

(2)　柔軟性の高いマイクロプログラム制御方式と大容量の制御記憶を採用すること．

(3)　評価データの計測，収集機構を持つこと．

以上のような目標仕様を満足させるべく，アーキテクチャ設計［Yokota 83-1 ］，ハードウェア設計［西川　83-3］を進めその一部について発表を行ってきたが，以下ではまずはじめに，PSI アーキテクチャの特徴をまとめたあと，ハードウェア各部分の仕様について詳細に述べ，それらが命令実行時にどのように使用され，実行効率向上に役立っているかについて論じる．

2．マシンアーキテクチャの概要

本節では，機械語命令レベルおよびその上のユーザに見えるマシンアーキテクチャについて要約し，後ほど述べるハードウエアの動作を理解するための助けとする．

語形式：　PSI の1語はFig.1 に示すように，8ビットのタグ部と32ビットのデータ部から構成される．タグ部の2ビットはガーベージコレクション用のマークビットで，残りの6ビットがデータタイプを表すデータタグである．データタグは，データタイプの区別，データと命令コードの区別，実行制御用のコントロールデータの区別などに使用される．

命令語の設定：　PSI ではPrologのサブセットにいくつかの拡張機能［高木　83］を付加した論理型言語KL0 をハードウェア（ファームウェア）とソフトウェアのインタフェース用に設定した．システム記述言語でありユーザ言語であるESP で記述されたプログラムは，コンパイラによりKL0 に変換される．PSI では，なるべく複雑度の低いハードウェアでKL0 を効率よく実行するために，機械語命令のレベルを十分高く設定して使とんどKL0 と同じに

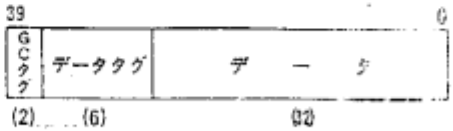


Fig.1　語 形 式

行時の命令コードのフェッチ回数を低くおさえる方針をとった．PSI の機械語命令は，Fig.2 に示すようにKL0 のソースプログラムをほぼ1対1に対応する内部コードに変換したもので，これをファームウェアのインタプリタで解釈実行する．KL0 の実行制御は基本的にはPrologと同じであり，ユニフィケーションやバックトラックはファームウェアのレベルで実現され，それを指定する命令はコード中に陽には出現しない．

KL0 の1つの節（クローズ）の命令表現は，Fig.2 のとおりクローズ・ヘッダ部，クローズ・ヘッドの引数部，複数のボディ・ゴール部からなる．ボディ・ゴール部にはユーザ定義述語（述語の命令表現へのポインタと引数）が現れる場合と組込述語が現れる場合がある．組込述語は1語中にオペレーション・コードと3個までの引数が納められたコンパクトな形式をとり，引数の各々は3ビットのタグを持つ．組込述語の実行時は，ユーザ定義述語呼出し時のようなユニフィケーションは起こらず，引数処理の後に，オペレーション・コードに対応したファームウェア・サブルーチンが呼出される．組込述語の引数には変数番号や小さな数値定数を直接置くことができ，メモリの節約と実行時間の短縮に効果がある．

KL0：　KL0 の仕様を手短かに要約すると次の3つとなる．

(a) Prologのサブセット

(b) 拡張制御機構

(c) ハードウェア制御機能

Prologのサブセットとは，主にassert，retract 等の内部データベース管理のための述語とread，write 等の入出力の述語がKL0 にはないということで，これらは(c) のプリミティブな機能を持つ組込述語で記述される．

拡張制御機構［高木　83］には，変数に値がユニファイされた時点で，あらかじめ登録しておいた手続きが起動されるバインド・フック機能，バックトラックで戻ってきたときに登録しておいた手続きが起動されるオン・バックトラック機能，述語呼出しの深さを指定して，その深さまでをカットする拡張カット機能などがある．これらの機能は言語の記述能力を高めるが，インプリメンテーションの点からみると，より多くの実行時処理（コンパイル段階では決まらない処理）を必要とする．

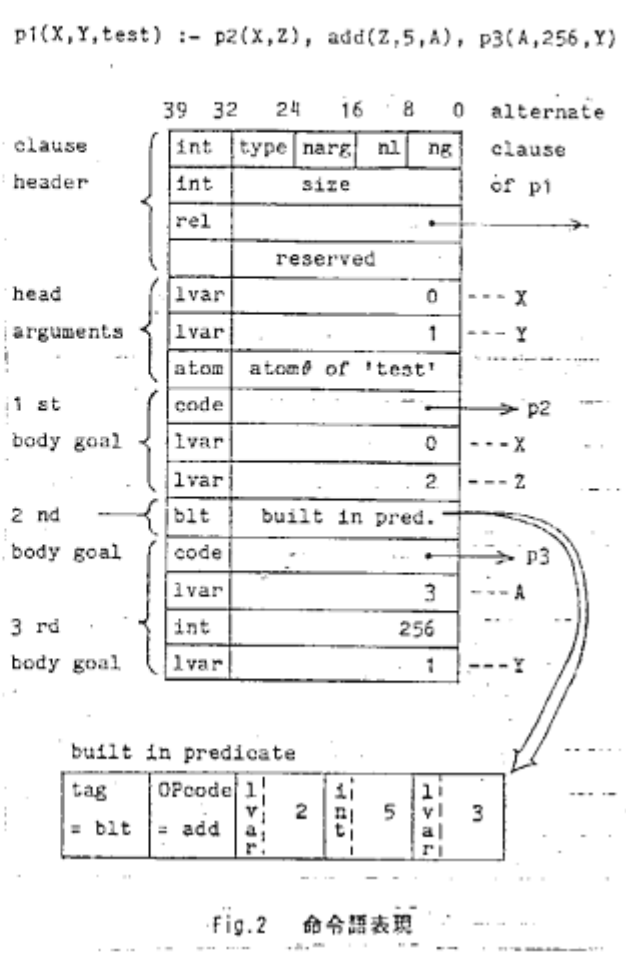


Fig.2　命令語表現

ハードウェア制御機能とは，メモリやレジスタを直接操作したり，入出力バスをアクセスしたりする低レベルの機能のことであり，OSの中で使用されるものである．

KL0 の実行環境：　KL0 は，ローカル，グローバル，コントロール，トレールの4つのスタックと，命令コードの格納されたヒープ領域を使って実行される．構造体データの表現はstructure sharing 法［Warren 77］を用い，スタックの利用方法や実行制御方式は，DEC-10 Prolog［Bowen 81］［Warren 77］と基本的に同じ方式を採用した．

Fig.3 はユニフィケーション時の

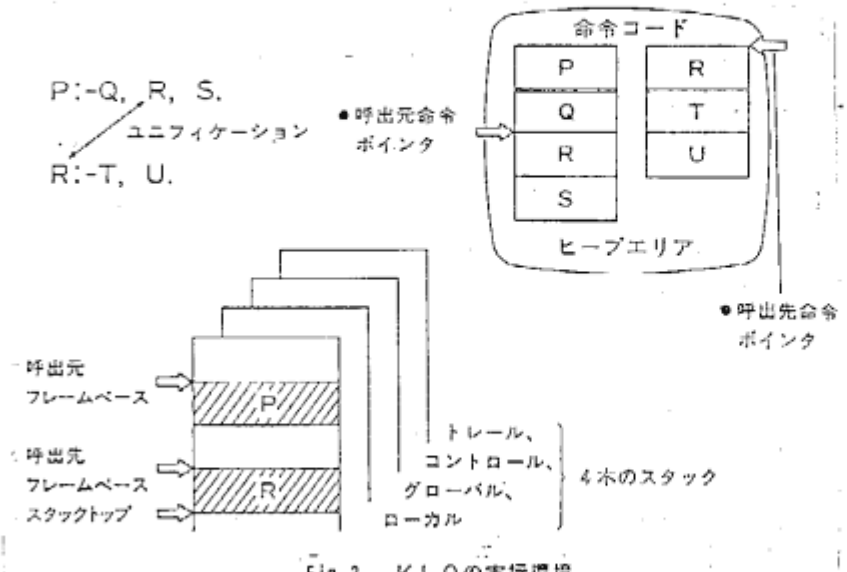


Fig.3　KL0の実行環境

KLO の実行環境を示す．ヒープ領域には呼出元と呼出先各々の命令コードが存在し，各々の命令コードを取出すための命令ポインタがある．変数の格納されるローカルスタック，構造体内の変数が格納されるグローバルスタック上には，述語呼出し毎に変数セル領域であるフレームがとられ，変数へのアクセスには各フレームのベースを指すフレームベースポインタからの相対距離が用いられる．コントロールスタックには，述語呼出しに対する戻り先情報，バックトラックの戻り先情報，戻り先で実行を継続するための各種ポインタ類からなる実行環境情報などが格納され，格納領域の起点がベースポインタで押さえられる．トレールスタックにはバックトラック時に変数の値を初期状態に戻すためのセルアドレス情報が積まれ，スタックトップを指すポインタでアクセスされる．これらの命令ポインタ，フレームベースポインタ，スタックトップポインタ類一式をまとめて，KLO の実行環境と呼んでいる．

アドレス表現：　PSI では，KLO の実行用に独立に伸縮する４本のスタックと，別に命令コードを置く領域が必要である．さらにOSやユーザからの要求により複数プロセスの並行実行をサポートし，複数プロセス間では，命令コードの共有や共通の変数領域が必要とされる．これらの要求を満たすために，４本のスタック用の領域にそれぞれ独立のアドレス空間を割当て，これをプロセスの個数分だけ持てるようにし，さらに命令コードの置かれるヒープ領域も独立のアドレス空間とした．これらの論理アドレス空間を“エリア”と呼び，それぞれに“エリア番号”を与える．PSI のアドレス表現は，Fig.4 に示すように 8ビットのエリア番号と24ビットのエリア内アドレスの合計32ビットからなり，16M 個の大きさをもつエリアが 256個まで利用できる．スタック用のエリアは，常にエリア内アドレス０番地側から使用され，ヒープ用のエリアも０番地側から使用されて途中に塵がたまれば，ガーベージコレクション時に０番地側へ詰め合わされる．

アドレス変換：　PSI には最大16M 語のメモリが実装できるが，プログラム実行中には各エリアの必要とするメモリ容量がダイナミックに変化するため，実記憶を各エリアにむだなく割当て・再配置するアドレス変換機構を設けた．実記憶は1K語のページに分割されて管理され，ページ単位に必要になった時点で割当てが行なわれ，ガーベージコレクションにより回収される．Fig.5 はアドレス変換機構の概念図で，ページ・マップ・ベースとページ・マップの２つのテーブルを用いて変換が行なわれる．

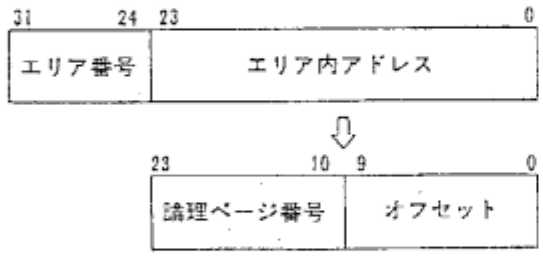
マルチプロセス：　PSI 上のプログラムは，プロセスという形をとって実行される．プロセスは実行環境として，前述のKLO の実行環境と割込許可レベルなどの若干のハードウェア制御情報を持ち，これらをーまとめにしてプロセス・コントロール・ブロック(PCB) と呼ぶ．PSI では，ユーザプログラム，OSプログラム，割込処理プログラム等は各々別のプロセスとして実行され，休止中のプロセスの PCB は定められた場所にまとめて退避されており，実行中のプロセスのPCB （カレントPCB ）はCPU のレジスタ上に分散して存在する．プロセスのスケジューリングはOSが行ない，プロセスの切替時にはカレントPCB の中身が入れ換えられる．プロセスの個数は，エリア数の制限から最大で63個であるが，OSやデバイスハンドラの記述には十分な数である．

割込み：　PSI の割込は，割込原因毎にベクターを持つ方式を採用し，ベクターにはプロセス番号を登録しておいて，割込が発生すると登録してあるプロセスへプロセス切替が起こる．割込レベルは外部２レベル，内部６レベルの８レベルを用意し，別にプログラム実行に同期して発生するエラーなどに対処するために，ノンマスカブルのトラップを設けている．

3．　システム構成

3．1　全体構成

PSI のシステム構成をFig.6 に示す．PSI はCPU 部分と入出力部分からなり，CPU 部分は制御部，演算部を中心と



1個のエリア：24ビットの論理アドレス空間

32ビットのアドレス空間：

—意に区別できる256個のエリアの集合

Fig.4　アドレス表現

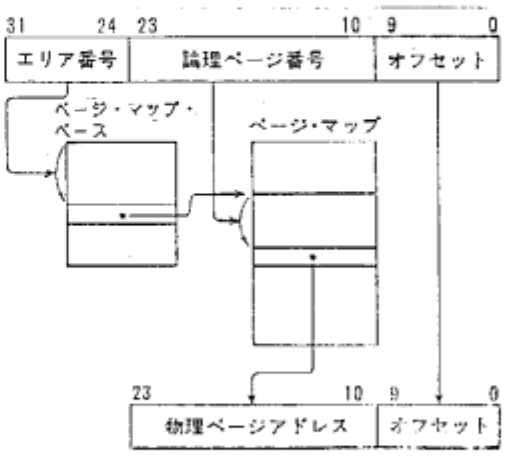


Fig.5　アドレス変換機構

し，主記憶，キャッシュメモリ，アドレス変換器を含むメモリ部，I/O バス制御部の各々が，内部バスにより相互に接続された構成をとる．入出力部分は，IEEE 796 規格の標準バスと各種の入出力デバイスから構成され，市販デバイスの接続も可能である．CPU 部分には，保守，立上げ，デバッグサポート用にコンソールプロセサが接続されており，さらに強力なデバッグのサポート用には，コンソールプロセサのかわりにミニコンピュータ(PDP11/23PLUS)が接続可能である．

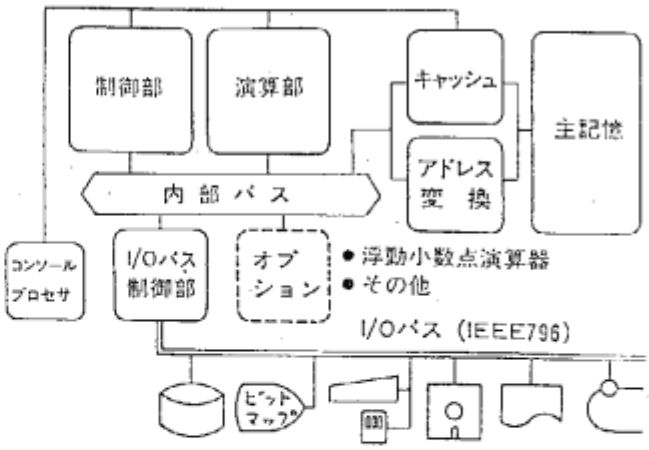


Fig.6　システム構成

3．2　基本設計方針

基本方針：　PSI のCPU は，ユニフィケーションや実行制御を高速化するハードウェア機構を取り込むが，物理サイズやコストパフォーマンスに対する要求から，ハードウェアの複雑化はなるべく避け，ファームウェアを活用する設計方針をとった．また短期開発に対する要求から，設計手法や実装技術は実績のあるものの活用に努めた．

機械語命令の設定とハードウェア構成：　2節に述べた命令語の設定により命令フェッチ頻度を低くおさえたため，命令先取り回路を省略して，cpu とメモリ間のインタフェースを1つにし，キャッシュ制御，メモリ制御回路を簡単化することができた．また命令コードのインタプリティブな実行により，命令のデコード回路も単純化できた．これにあい対する設計方式としては，機械語命令の機能を低く設定して命令デコードと実行をなるべくハードウェア化し，同時に命令の先取り機構を設け，低レベルの命令をパイプライン的に高速実行するものがある［Warren 84 ］．

後者の方が，個々の命令の実行がより決定的で（インタプリティブでなく）ハードウェアの最適化が図りやすく，高速実行には適すると考えられるが，ハードウェアの量と複雑度はそれだけ増大する．PSI では，ハードウェアの複雑化を押さえる必要があること，KL0 の説明の項で述べたPrologの拡張機能が実行時処理を多く必要とし，ファームウェアのインタプリティブな実行の方が適していることの2点から，前者の方式を採用した．

スタックアクセスの高速化：　Prolog系の言語では，バックトラックのための情報をスタックに残しながら実行を続けるため，呼出元のフレームがスタックの深いところに存在するという状態がよく起こり，スタックアクセスはスタックトップと深いところに分散する．したがって，スタックトップ部分だけを常に高速メモリ上に持ってくるようなスタックキャッシュのハードウェアでは効率が半減する．このためPSI では，より一般的な高速化機構であるキャッシュメモリを採用し，キャッシュメモリの方式に若干のスタックアクセス向きの機構を取り入れることとした（後述）．

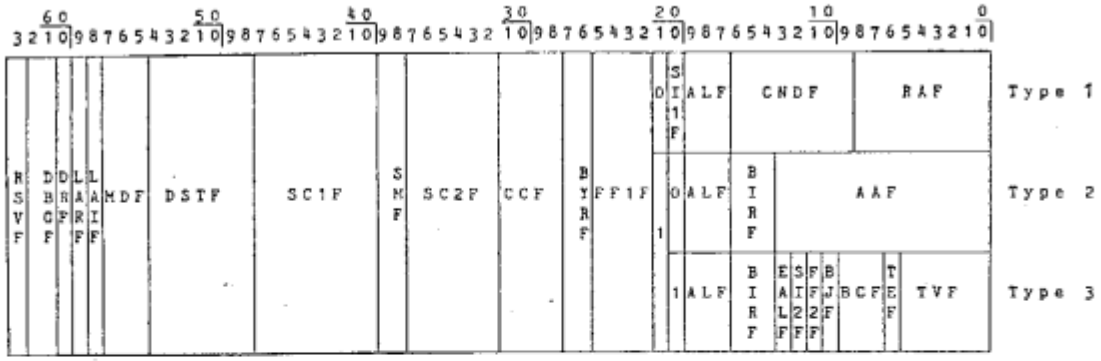
各種の高速化機構：　CPU の中で，データパスや制御タイミングなどについては単純明解であることに努めたが，Prolog系言語用の高速化機構としては，ファームウェア・インタプリタの中で多用するタグ判定，条件分岐機構の強化や，TailRecursion Optimization [Warren 80]に使用するレジスタファイルの設計などに力を注いだ．各々の詳細は5節で述べる．

4．マイクロ命令仕様

方式：　マイクロ命令の取込みと実行は，1つのマイクロ命令の実行中に次の命令を取込む最も単純なパイプライン方式を採用したが，分岐制御を工夫して1つのマイクロ命令の実行結果がすぐ次のサイクルで条件分岐やディスパッチに使えるようにした．このことは書きやすさの向上につながるとともに，分岐頻度の高いユニフィケーション処理などの高速化に役立つ．データ処理に関しては，1つのマイクロ命令によりレジスタからのデータ読出し，演算，異なるレジスタへの書込みが可能なように設計し，従来マシンのアセンブラでプログラムを書くのと同程度の手軽さでマイクロプログラミングができるように配慮した．またマイクロ命令のビット幅を許される限り広くとり，ハードウェア資源の操作の並列度を高めた．

マイクロ命令フォーマット：　Fig.7 に示すとおり，命令のビット幅は64ビットで，3種類の命令タイプがある．ビット63～22までは各タイプに共通であり主にデータ系の操作を指定し，ビット21～ 0は命令のタイプ毎にフィールドの意味が異なり，主に分岐制御とALU オペレーションを指定する．マイクロ命令の各フィールドは，マイクロプログラムの記述のし易さを考えて適度にエンコードした形式をとっている．

3つの命令タイプ：　Type 1では条件分岐やディスパッチが指定可能であるが，飛び先の範囲が正負 256語以内に制限される．またType 1では 6ビット以内の即値が使用でき，演算に関しては数値演算だけが許される．Type 2は無条件分岐の指定であり，同時に論理演算とビット・ロテ



RSVF(1) ：リザーブ・フィールド
DBGF(2) ：ブレークポイント指定，評価用カウンタ制御
DRF (1) ：キャッシュアクセス時のデータレジスタ指定
LARF(1) ：キャッシュアクセス時のアドレスレジスタ指定
LAIF(1) ：アドレスレジスタの自動加算指定
MDF (3) ：マルチデスティネーション指定；DSTFと並列的に特定レジスタへの書込を可能とする
DSTF(7) ：デスティネーション・フィールド；データ書込時の書込先指定
SC1F(8) ：ソース１・フィールド；ソースバス１側のデータ読出指定
SMF (2) ：ソースバス２のマスクパターンの指定
SC2F(6) ：ソース２・フィールド；ソースバス２側のデータ読出指定
CCF (4) ：キャッシュ制御フィールド
BYRF(2) ：ソースバス２側のバイト・ロテーション指定
FF1F(4) ：マイクロステータス・レジスタ上の識別フリップフロップや演算フラグの制御指定
SI1F(1) ：ソースバス２にSC2Fの内容を即値として出力する指定
ALF (3) ：数値演算および論理演算指定；Type 1では数値，Type 2では論理，Type 3では両方が可能
CNDF(8) ：コンディション・フィールド；分岐条件指定
RAF (9) ：条件分岐時の飛び先指定（相対アドレス）
BIRF(3) ：ビット・ロテーション指定
AAF(14) ：無条件分岐時の飛び先指定（絶対アドレス）
EALF(1) ：拡張ALF ；ALF が数値演算か論理演算かを指定
SI2F(1) ：SI1Fに同じ
FF2F(1) ：拡張FF1F；FF1Fの意味を拡張する
BJF (1) ：ジャンプ・レジスタによる間接分岐指定
BCF (3) ：入出力バス制御指定
TEF (1) ：タグデータに即値を埋め込むことの指定
IVF (6) ：タグの即値データ
注，( )内の数値はビット幅を示す

Fig.7　マイクロ命令フォーマット

ーションの指定が可能である．飛び先の制限はない．
Type 3では，分岐制御はジャンプ・レジスタを用いた間接分岐だけに制限されるが，数値・論理の演算指定，ビット・ロテーション指定，即値指定，タグ部分の即値指定，入出力バス制御指定など多くの処理が同時指定できる．

データ操作系フィールド：　ビット63~22のデータ操作系フィールドの中心を成すのが、SC1F，SC2F，DSTFの３つのフィールドで，SC1FとSC2Fで指定した２つのデータに対してALU で演算を施し，それをDSTFで指定したレジスタにしまうという，いわゆる３オペランド指定を行なう．またMDF により，DSTFで指定したレジスタ以外の特定レジスタに同時に書込を行なうマルチ・デスティネーション指定が可能である．SC2Fで指定したデータは，演算に使用される前にBYRF，SMF によりALU の入口でバイト・ロテーションとマスク操作が行なわれる．メモリアクセスの指定は上記のデータ操作とは独立してキャッシュ制御フィールド(CCF) で行なわれ，２本ずつあるアドレス・レジスタ，データ・レジスタのいずれを使用するか，またアドレスレジスタの使用後自動加算（+1)を行なうかどうかが，DRF ，LARF，LAIFで指定される．

5．　ハードウェア各部の設計

本節では，ハードウェア各部の設計と，その使用方法について，Fig.6 のシステム構成図のブロック分けにしたがって述べる．

5．1　演算部

5．1．1　演算部の構成

Fig.8 に演算部の構成の概念図を示す．ワーク・ファイル(WF)と呼ぶレジスタファイル，32ビットのALU ，メモリ部とのインタフェースを行なうアドレス・レジスタ(AR)，データ・レジスタ(DR)，それらを相互に接続する内部バスなどを中心に構成される．Prolog系言語用マシンとしての特徴は，WFが強化されていること，AR，DRが２組ずつあること，タグ操作機能を備えていることなどである．

5．1．2　バス構成

PSI の内部バスは，２本のソースバスと１本のデスティネーションバスの３本から構成され，各々のバスはタグ用に８ビット，データ用に32ビット，合計で40ビットの幅を持つ．
１マイクロ命令により，次のような基本的なデータ操作が可能である．WFまたはDRから各々のソースバスにデータを読出し，ソースバスの一方（ソースバス２）のデータに対してはシフト操作とフィールド抽出操作を行ない，その結果ともう一方のソースバス（ソースバス１）上のデータと

の間で数値演算または論理演算を行ない，演算結果をデスティネーションバス経由でWF，DR，ARなどに書込む．もちろん転送だけのときはALU やシフタ，フィールド抽出などはNOP とする．WFの２つの読出しアドレスと１つの書込みアドレスは全部異なっていて良く，またマイクロ命令のHDF の指定によりWFとAR，WFとDRへの同時書込みが可能である．先にARへ転送しておいたアドレス情報を用いて，上記のデータ操作と並行してメモリアクセスを行なうことができる．

CPU 内のほとんどすべてのレジスタの内容は読出し可能であり，その多くは２本のソースバスのどちらかに読出されて演算対象となるが，一部の読出しの遅いレジスタと演算の不要なレジスタは，直接デスティネーションバスに読出される．

実装上40ビット幅のバスを３本も通すことは難しく，ソースバス２とデスティネーションバスは，同一の信号線を時分割で使用している．

５．１．３　タグの取扱い

ALU は32ビットのものを使用しているためデータの演算時，転送時には，タグはソースバス１のタグをそのまま使用するか，タグ即値データでそっくり置きかえるかのいずれかを行なう．タグ値に演算を施すのはガーベージコレクション時以外には少なく，その場合にはタグの値を32ビットのデータ用のソースバスに読出し，ALU で演算したあとタグ用のバスに戻し，デスティネーションで指定したレジスタのタグ部の書きかえだけを行なう．タグ部の比較はしばしば必要であり，ALU 演算と並行して２つのソースバス上のタグの一致を調べフラグをセットする機能を設けている．またタグの値と即値との一致により条件分岐することができる．タグ部を持つレジスタは，WFとDRだけである．

５．１．４　バレル・シフタとフィールド抽出回路

バレル・シフタ：　バレル・シフタでは32ビットまでのロテート・シフトが可能であり，ALU の演算指定と組み合わせてQRと連結した64ビットの左右シフト（乗除算用）にも使用できる．32ビットのロテート・シフトは，バイト単位のロテーションとバイト内シフトから構成され使用頻度の高い前者はマイクロ命令の全タイプで，後者を含めた32ビット・ロテート・シフトはType 2 Type 3で使用できる．主な用途は，機械語命令コード中のオペランドの切出し，ストリングデータの位置合わせなどで，ビットストリング処理の高速化には必須である．

フィールド抽出回路：　フィールド抽出回路は３種類の決まったマスクパターンを持つマスク回路であり，オペランドの切出しやデータの特定部分の切出しに使用頻度の高い，下位５ビット，下位８ビット，下位16ビットを各々通過（上位はゼロに置きかえ）させる機能を持つ．但しNOP 指定時には全ビット通過となる．任意部分のマスクまたは抽出を行ないたい場合には本回路を使わずに，WFの定数領域（後述）内にマスクパターンを入れておいて，ALU でAND 演算を行なう．

５．１．５　ALU とスワップ回路

ALU は市販のALU LSI で構成され，LSI の機能のうち必要なものだけを使用するように，マイクロ命令のALF をエンコードしてある．算術演算には，加減算，キャリー，ボローを含めた加減算，QR制御を含む乗除算用演算の他，アドレス表現の項で述べたエリア内アドレスの計算用に24ビット演算を用意してある．マイクロ命令のFF1Fでフラグセットを許可している場合には，演算結果によりキャリー，オーバフロー，ゼロ，その他のフラグがセットされ条件分岐に使用できる．論理演算には，through ，AND ，OR，Exclusive-ORなどの他，SWAP付きのAND ，ORなどがある．SWAP付きとは，ALU の出口（Fig.8 には省略されている）にあるスワップ回路を働かせることで，ALU の演算結果がバイト単位に上下入れかえられる．すなわち，第０と第３バイト，大１と第２バイトが入れかえられデスティネーションバスに送られる．スワップ回路は，数値データのバイト並び，ストリングデータのバイト並びが逆になるのを同じにそろえたり，入出力バスへデータを送出する際にバイト順を入れかえたりするのに用いる．

５．１．６　ARとDR

アドレス・レジスタとデータ・レジスタはそれぞれ２組存在し，PLAR，CLAR，PDR ，CDR という名で呼ばれている．P とC は述語呼出時のparent，current の意味で，ユニフィケーション時にメモリから命令コードやデータを読出す際に，P は呼出元（親），C は呼出先（子）の情報をそれぞれ取扱うのに使用される．PDR ，CDR 上のデータのタグ部は，タグディスパッチ（後述）に使用され，データ部の下位５ビットは，WF上の特定領域のアドレス指定にも使用される（後述）．PLAR，CLARには，auto increment機能を備え，連続データのアクセスの効率化を図っている．

５．１．７　ワーク・ファイル

WFは演算部の中心をなす多目的のレジスタファイルで，

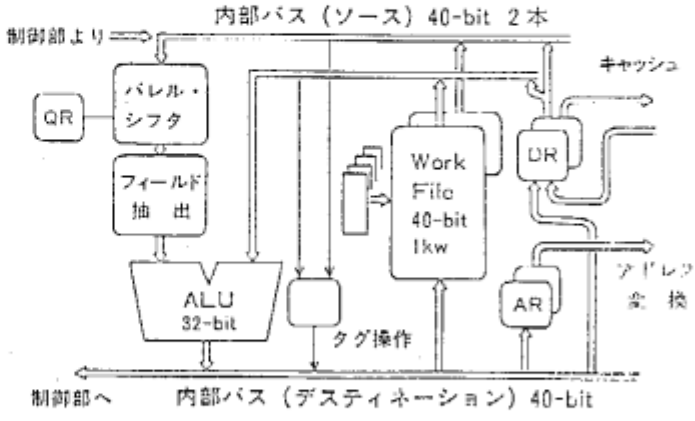


Fig.8　演算部の構成

40ビット×1K語の容量を持ち，読出し，書込みのための多くのアドレシングモードを有する．WFは1マイクロ命令で読出しと演算結果の書込みを任意のアドレスに対して行なうことができ，さらにWFの先頭部分16語はジェネラル・レジスタとして使用するためにdual-port 化されている．このことによりWF上のデータ間（一方はジェネラル・レジスタでなければならない）でのALU 演算を可能としている．WFは，Fig.9 の概念図に示すとおり，多くのアドレシングモードを有しており，以下で順に説明を加える．

(a) 直接アドレス指定

WFの先頭64語と最後の64語はマイクロ命令により直接アドレス指定ができ，使用頻度の高い情報の格納に使用する．先頭の16語は前述のとおりジェネラルレジスタに，その後の48語はKL0 の実行環境を保持するロジカルレジスタなどに使用し，最後の64語はファームウェア・インタプリタの使う定数やマスクパターンの格納に使用する．ジェネラル・レジスタは一時記憶用に使用するが，実行環境としても保存している．

(b) 間接アドレス指定

WFAR1 , WFAR2 はWF用のアドレス・レジスタであり，このいずれかで間接アドレス指定を行なうことにより，WFの任意の番地をアクセスできる．これらのレジスタは，間接後自動増加，減少の機能と，32語， 256語境界に達したことをフラグに反映する機能を有するため，WF上のある部分をスタックとして使用するのに適する．具体的には，ローカル・フレーム・バッファやトレール・バッファ（後述）のアクセスに用いている．

(c) PDR とCDR による間接指定

機能： この機能は，PDR かCDR かいずれか（マイクロ命令のDRF で指定）の下位5ビットを取り出してその上位にWFBRの内容を結合し，それをアドレスとしてWFをアクセスするものであり，WFにとられたローカル・フレーム・バッファ（以下LFB と略す）上のローカル変数のアクセスに用いる．

フレーム・バッファとＴＲＯ： LFB とは，Fig. 3 に示したスタック上のフレームのうち，呼出先(current) のローカル・フレームをスタック上ではなく一時的にWF上にとったものである．LFB を用いるユニフィケーション時には，呼出元(parent)のローカル変数のうちユニフィケーションに使われるものが一但LFB 上にコピーされ，そののち呼出先のクローズ・ヘッドの引数とユニフィケーションが行なわれ，LFB 領域はそのまま呼出先の変数領域となる．呼出先クローズのボディ・ゴールの実行に移るとき，それがユーザ定義述語であれば，LFB 内容はローカル・スタック上に掃き出され新たに呼出された述語の変数領域がLFB 上にとられる．一方ボディ・ゴールとして組込み述語が連続する間は，LFB の掃き出しは行なわずそのままcurrent の変数領域として使用されつづける．さらにユーザ定義述語呼出を行なうときにも，実行中の述語にalternative clauseがなく，かつ最後のボディ・ゴールの呼出であるときには，LFB の掃出しを行なわずに，新たな呼出先で使用するローカル・変数をLFB 上に上書きして使用する．これはすなわち，最後のボディ・

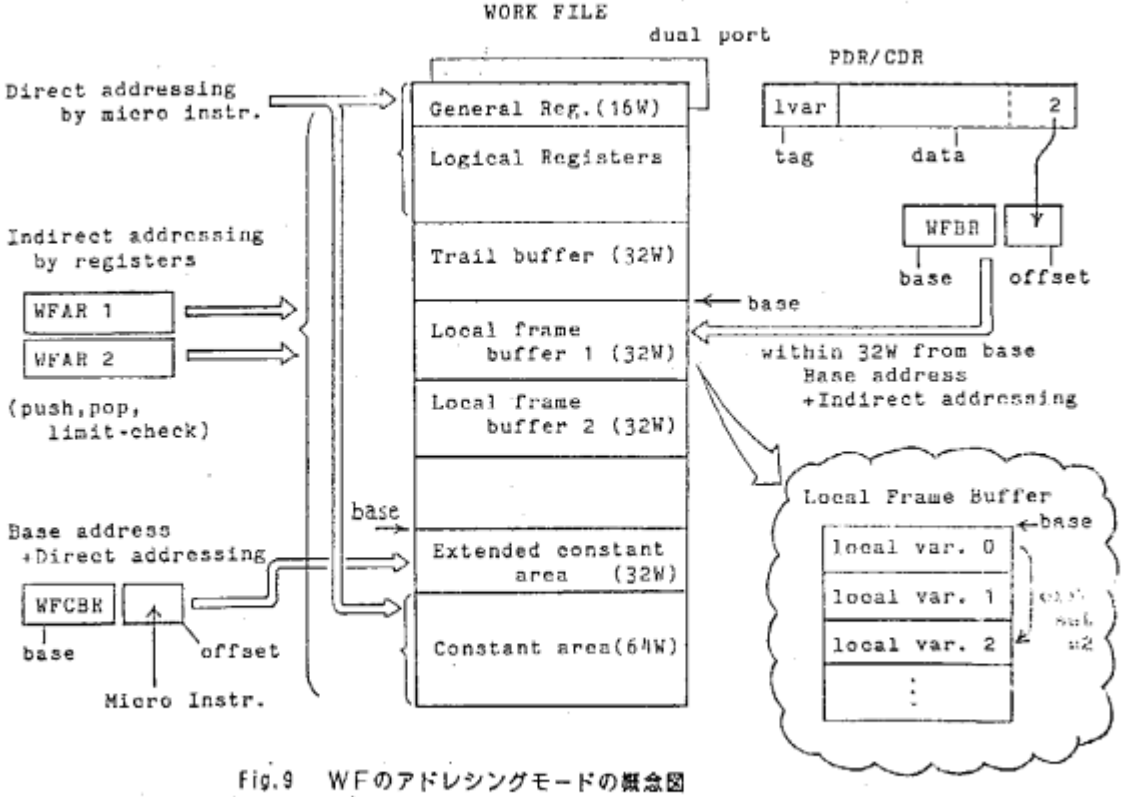


Fig.9 WFのアドレシングモードの概念図

ゴールは“call”するのでなく呼び出し先へ“jump”することに相当し，いわゆるテイル・リカージョン・オプティマイズ(TRO) [Warren 80]に当たるものである．このようにLFB を利用することにより，クローズ・ヘッドの後に組込み述語が連続するときやTRO を行なうときにはcurrent のローカル変数が常にWF内のLFB 上に存在することになり，メモリアクセスが減少し処理効率の向上に効果がある．

使用方法： PDR，CDR による間接指定の使用方法に話を戻すと，LFB の先頭をFig.9 のWFBRで押さえておき，n番目のローカル変数を表わす命令コード(lvar n)をPDR またはCDR に読み込んでくる．この状態でPDR またはCDR の間接指定でWFをアクセスすると，WFBRの指すローカル・フレーム・バッファの先頭からn番目のcurrent ローカル変数セルがアクセスできる．LFB の大きさは，32語固定としており，ファームウェアの制御で２面のLFB を交替使用している．LFB の他に，ユニフィケーション時にはトレール・スタックに積む情報も一但WF上に格納することにしており，これをトレール・バッファと読んでいる．

(d) ベース相対指定

WFCBR の内容とマイクロ命令中の５ビットを結合したものをアドレスとして，WFCBR が押さえるベースアドレスから32語以内を直接アクセスすることができる．これは(a) で述べた定数領域の拡張用やワーク用領域に使用する．

以上のように，ワーク・ファイルは，PDR，CDR とともに，ユニフィケーションやファームウェア・インタプリタ処理のデータ操作において，中心的な役割を果たすものである．

５．１．８ レジスタ・ファイルとしてのWCS

WCS はマイクロ命令サイクルの後半ではマイクロ命令の読み出しに使用されるが，サイクルの前半では内部バスからのアクセスが可能なように設計されている．これにより，ファームウェアからのWCS の書替えが可能であるが，この機能を用いてWCS の最後部1K語をプロセスの実行環境（64プロセス分）の退避領域として使用し，プロセス切替の高速化を図っている．

５．２ 制御部

５．２．１ 制御部の構成

Fig.10に制御部の構成の概念図を示す．PSI の制御部はマイクロプログラム制御方式を採用し，WCS として64ビット×16K 語の大容量のものを実装した．１つのマイクロ命令の実行中に次のマイクロ命令を読み出すもっとも単純なパイプライン方式を使用しており，各サイクルの前半は次のマイクロ命令アドレスの生成に，サイクルの後半は生成したアドレスからのマイクロ命令の読出しに使用している．マイクロ命令の生成方法として，絶対分岐，相対分岐，条件分岐（相対アドレス），継続実行，OPコード・ディスパッチ，タグ・ディスパッチ，オペランドタグによる多方向分岐，マイクロ・サブルーチン・コール，リターン，JRによる間接分岐，などが存在する．本マシンに特徴的なのは，タグディスパッチ機能，オペランドタグによる多方向分岐機能，条件分岐の豊富さなどである．

５．２．２ 継続実行と無条件分岐

あるサイクルで実行中のマイクロ命令のアドレスはMAR に保持されている．継続実行のための次の命令アドレスは

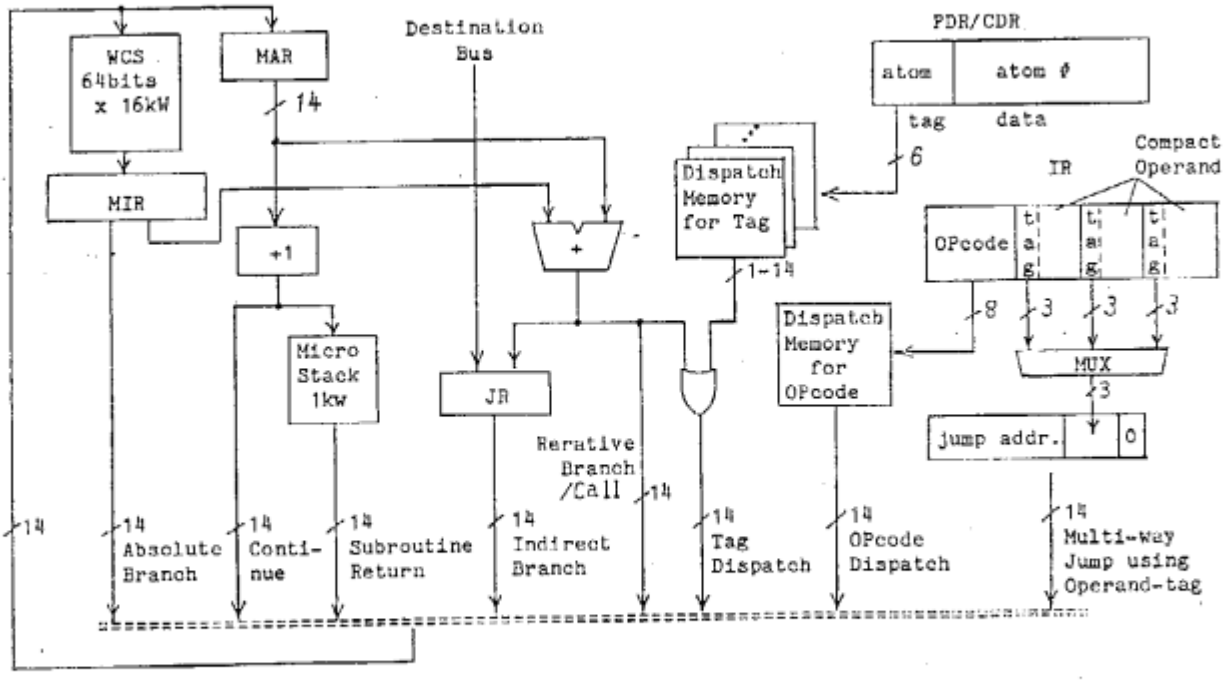


Fig.10 制御部の構成

HAR に１を加算して生成され，相対アドレスによる無条件分岐の場合の命令アドレスはHAR に相対アドレスを加算して生成される．条件分岐の場合には，上に述べた方法で生成した２通りのアドレスのいずれかを選ぶかを分岐条件により決めている．絶対アドレスによる無条件分岐では，マイクロ命令のAAF の値がそのまま分岐先アドレスとして使用される．

HAR の更新は，マイクロ命令レジスタMIR と同じタイミングで行っている．

５．２．３　OPコード・ディスパッチ

Fig.2 で示したようにKLO の命令コードの中で組込み述語は，OPコードと３つ以内のオペランドからなるコンパクトな形式を持っている．

この組込述語の命令コードは，データ・レジスタ経由でインストラクション・レジスタIRにセットされ，IRではOPコード部分の８ビットが取出されて，ディスパッチ・メモリ（OPコード用）へアドレスとして供給される．ディスパッチ・メモリは，OPコードを対応するファームウェア処理ルーチンのスタートアドレスに変換し，次のマイクロ命令アドレスとして供給する．ディスパッチ・メモリによるアドレス生成は半サイクルで終り，そのサイクル中に飛び先のマイクロ命令の読みだしが行われて分岐が完了する．

OPコード用のディスパッチ・メモリは，14ビット× 256 エントリからなり，256種までの組込み述語に対応できる．周メモリはRAM で構成され，OPコードの追加・変更が楽に行える．OPコードに対応するファームウェア処理ルーチンのスタートアドレスは，マイクロプログラム・アセンブラが自動的に収集し，編集プログラムによりディスパッチ・メモリにロードできる形に自動生成している．

５．２．４．　タグ・ディスパッチ

ファームウェア・インタプリタによるユニフィケーションや実行制御の中では，メモリから読みだした命令コードやデータのタグの判定を頻繁に行う必要がある．これを高速化するために導入したのがタグ・ディスパッチ機構であり，PDR またはCDR に読み出したデータのタグにより，１サイクルでディスパッチを行う．

タグ・ディスパッチは，前述のOPコード・ディスパッチと異なり，マイクロプログラムで指定されたアドレスをベースとした多方向分岐を行う．分りやすく説明すると，タグの種類は６４種も存在するが，タグによって異なった処理ルーチンへ分岐する分岐先の数は，実際に処理を記述してみると５種類から１６種類の範囲で足りる．そこで６ビットのタグを処理ルーチンに対応したコード（５方向分岐なら３ビット，１６方向分岐なら４ビット）に変換し，このコードをマイクロ命令で指定した分岐アドレスとコンカチネートしたものを実際の分岐アドレスとして使用し，コードによる多方向分岐（５-wayないし16-way）を実現する．タグからコードへ変換する変換テーブルはタグ用のディスパッチ・メモリに格納するが，変換パターンは複数種類必要となるので１２面用意し（内９面使用中），どの面を使ってコード変換するかをマイクロ命令で指定している．

実際には，分岐先で１ないし数ステップの処理を記述する必要があるので，ｎ方向分岐の分岐先は連続アドレスの他に２語間隔，４語間隔，８語間隔となるような変換テーブルを用意して使い分けている．

コンカチネートの処理は実際には，柔軟性を増すためにORゲートで行っており，またマイクロ命令で指定する多方向分岐のベースアドレスは，どの変換テーブルを使うかにより適当な２のベキ乗境界に合せるようにマイクロプログラム・アセンブラ上で制御している．ディスパッチ・メモリは１４ビット×１Ｋ語の高速メモリで構成されており，OPコード用とタグ用を１つの高速メモリ内に同居させている．

５．２．５　オペランド・タグによる多方向分岐

組込み述語は，Fig.2 に示した通り３個までのコンパクトなオペランドを持つことができ，各オペランドには各々３ビットのタグが付加されている．IRには，このオペランド部分のタグを取出す機構があり，取出したタグを１ビット左にシフトしたものとマイクロ命令で指定した分岐アドレスをコンカチネートして実際の分岐アドレスとし，３ビットタグによる８方向２語おきの多方向分岐を実現している．

５．２．６　条件分岐

条件分岐の多様さは，本マシンの特徴の１つである．64種の条件分岐についてそれぞれtrue branch，false branch が指定でき，また，マイクロ命令のSC2Fで指定されるレジスタのタグとタグ即値（CNDFで指定）の一致により条件分岐することが可能である．64種の分岐条件は，おおよそ以下のように分類される．(a) ～(d) までは，マイクロ・ステータス・レジスタMSTRにふくまるるフラグ類で，プロセスの実行環境として保存される．

(a) 10種類の演算系フラグ

(b) デスティネーションバスからMSTRへの書込みによりセットされる６ビットの凡用フラグ

(c) マイクロ命令のFF1Fにて個別にセット，リセットできる８ビットの凡用フラグ（ファームウェア・インタプリタがスイッチとして使用）

(d) FF1Fの指定によりデスティネーションバスのビット24から31の値がロードされる８ビットのフラグ（クローズの属性ビット保持用）

(e) マイクロ命令のSC2Fで指定されるレジスタのタグの各ビットの１／０判定

(f) 割込み要求有無のテスト２種

(g) JRの内容＝ゼロのテスト

(h) WFAR1,2 が３２語境界，２５６語境界の値に達したか否かのテスト

(i) 入出力バス状態のテスト２種

(j) メモリおよびアドレス変換器busyのテスト

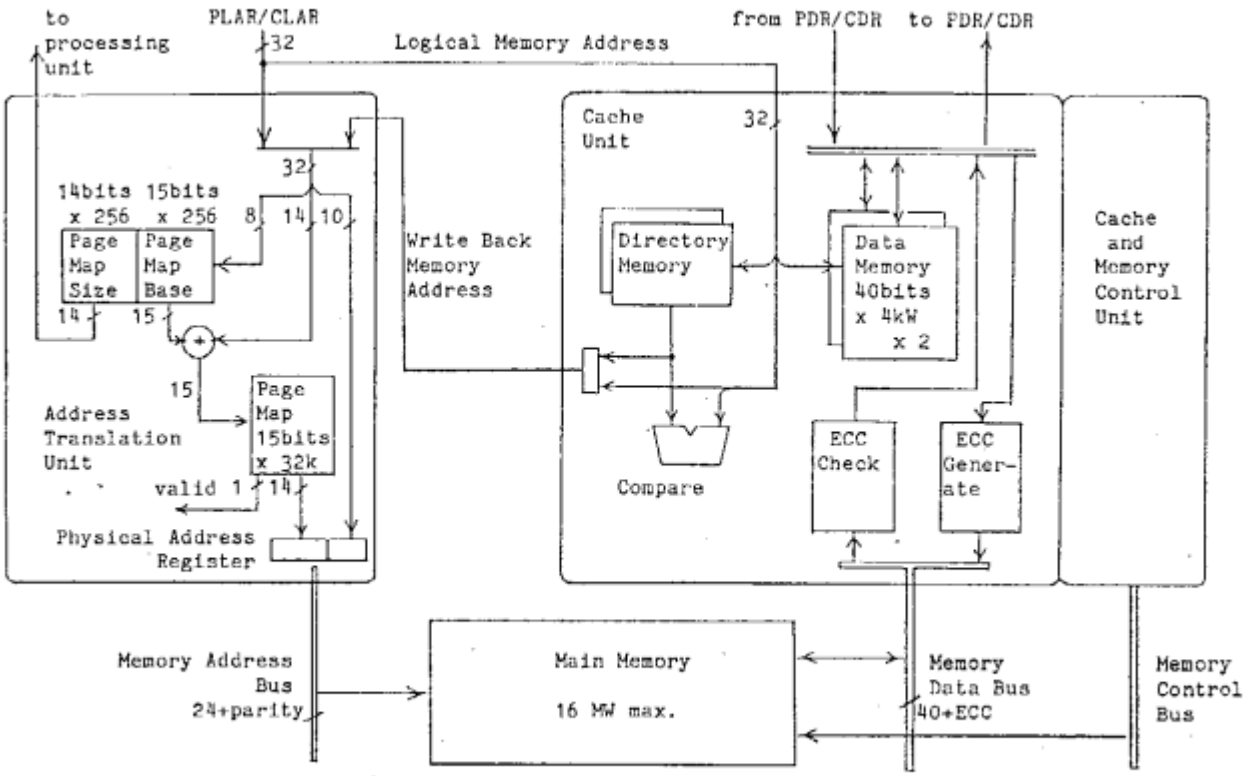


Fig.11 メモリ部の構成

これらのうち、ファームウェア・インタプリタ内で高い頻度で使用されるているのが(c) であり，ファームウェア・モジュール間のインタフェースでは，このように簡単にセット／リセット，テストのできるスイッチ類の利用価値が高いようである．

5．2．7 サブルーチン・コールとリターン

サブルーチン・コールは，次のマイクロアドレスをマイクロスタックMSTKにプッシュしながら，相対アドレスで無条件分岐を行うものであり，リターンは，MSTKをポップして読み出されたマイクロアドレスへ分岐するものである．

MSTKはデータ操作系の側から読み書きができるので，プロセス切替時等には退避復旧を行っている．MSTKは１K語の深さを持っているが現在は１６語程度しか使用していない．

5．2．8 JRによる間接分岐

JRには２とおりの方法で分岐先アドレスを格納することができる。１つはマイクロ命令のCNDFの指定により RAF で指した分岐先アドレスをロードする方法で，もう１つは，DSTFの指定でデスティネーションバスから値をロードする方法である．JRによる間接分岐は，Type1 とType3 のマイクロ命令で可能である．

JRはループカウンタとしても利用することができ，MDF により－１を指定し， 5.2.6の(g) でゼロ判定を行うことができる．

5．3 メモリ部

5．3．1 メモリ部の構成

構成： Fig.11にメモリ部の構成の概念図を示す．メモリ部は，キャッシュ・ユニット，アドレス変換ユニット，主記憶から構成される．演算部からのメモリアクセスは全て，キャッシュ制御のマイクロ命令で行われ，キャッシュがミスヒットしたときだけ，キャッシュ制御部の働きにより主記憶までアクセスがおよぶ．キャッシュ制御部は，CPU 全体の制御部とは，独立したシーケンス制御回路を持ち。アドレス変換ユニットの制御，主記憶のタイミング制御とリフレッシュ制御を同時に行う．一旦，マイクロ命令によりキャッシュアクセスのオーダが出されると，キャッシュはマイクロ命令の制御とは独立に動作を始め，キャッシュアクセスが完了するまではCPU は別の仕事をすることができる．

動作概要： キャッシュ読出しに例をとり，動作の概要を述べる．２本の論理アドレス・レジスタPLAR,CLAR のいずれかにメモリアドレスをおき，キャッシュ読出しのマイクロ命令を実行すると，キャッシュ制御部が動作を始めてまずキャッシュ・ディレクトリを検索し，ヒットかミスヒットかを判定する．ヒットの場合にはそのサイクル中に，キャッシュのデータ・メモリからデータが取り出されて，PDR とCDR のうち指定された方にデータを格納して動作を完了する．キャッシュのヒット／ミスヒットの判定を行っている間，アドレス変換ユニットでは論理アドレスから物理アドレスへの変換が行われ，ヒット時には変換結果は捨て

られる．ミスヒットの時には変換結果の物理メモリアドレスを用いて直ちに主記憶アクセスが行われ，読み出されたデータはキャッシュのデータメモリとPDR またはCDR へ格納される． アドレス変換ユニットでは，変換テーブルの使用されたエントリにバリッドビットが立っていたか，テーブル検索のためのアドレス計算時に桁あふれが無かったかをチェックしており，異常が検出されれば，CPU 内のトラップビットを立て，あとの適当な時点でファームウェア・インタプリタがトラップビットのON状態を検出してトラップ処理を行う．

5．3．2 キャッシュ・ユニット

方式： 本キャッシュ・メモリは，2セットのセットアソシアティブ方式を採用しており，おきかえのアルゴリズムとしてLRU(Least Recently Used 方式）を用いている．キャッシュ・メモリの管理単位は4語のブロックで，ミスヒット時にはブロック単位の入れかえを行う．本キャッシュ・メモリは論理アドレスでアクセスを行っており，アドレス変換のオーバヘッドを削減している．

書込みの方式としてはライトバック方式，すなわち書込み時には主記憶までデータを戻さず，ミスヒットが発生したときに始めてキャッシュ上のデータを主記憶へライトバックする方式を採用した．ライトバック方式は，ミスヒットが発生したときには，主記憶へのデータの書き戻しと新しいデータの読込みを両方行わねばならないが，スタックへのプッシュ／ポップの多いシステムでは書込み時のオーバヘッドが無く，ライトスルー方式に比べ効率が良い．

容量： 本キャッシュ・メモリは，40ビットＸ１Ｋ語のセットが２面，合計で８Ｋ語が実装されている．

速度： アクセス時間は，ヒット時には読出し，書込みとも１マイクロ命令サイクルで動作が終了する．ミスヒット時の読出し時間は，１語めが読み出されるまでに４サイクル，主記憶から４語のブロックが全部読み出されるまでに７サイクルを必要とする．

キャッシュ・オーダ： Table.1 にキャッシュ制御のマイクロ命令の一覧表を示す．CPU との同期オーダや，スタック用のライトアクセス・オーダを備えている．

5．3．3 アドレス変換ユニット

アドレス変換のメカニズムは，基本的にはFig.5 で示したとおりである．ページマップ・ベース・メモリは15ビットＸ 256語の専用メモリで，各エリア番号に対応するページマップのエントリが，何番地からはじまっているかを保持している．ページマップ・メモリは，15ビットＸ32K語の専用メモリで，論理ページ番号に対応する物理ページ番号の変換テーブルを保持している．ページマップ・メモリ上には，各エリア番号に対応する変換テーブルが，間隔を置きながら配置される．ページマップ・メモリの各エントリの最上位ビットはバリッドビットであり，ページの割当て時にセットされ，アドレス変換時にチェックされる．

本ユニットには，ページマップ・サイズ・メモリと呼ばれるもうひとつのメモリがあり，各エリアに対して割当てられている実ページ数を格納するのに使用している．

本システムでは，各エリアに対して必要になった時点で動的にページ割当てを行うため，ページ割当ての必要なことの検出を前記のページマップ・メモリ中のバリッドビットの検出で知ることもできる．しかしながら，メモリアクセスの段階になって始めてページ割当て要求がわかるのでは不便なため，アドレス計算時に１Ｋ語のページ境界を越えたか否かが容易にわかるフラグを導入して使用し，バリッドビットのほうはエラーチェックにだけ使用している．

本ユニットは、変換テーブルを専用メモリで構成しているため、１マイクロ命令サイクルで変換を終了することができる。アドレス変換作業は、キャッシュ・ミスヒット時のライトバックを行う場合にも自動的に実行される．

5．3．4 主記憶

主記憶は，１語４０ビットで最大１６メガ語が実装可能である．キャッシュ・ユニットのなかにエラー検出・訂正機構を持ち，１ビットのエラー訂正と２ビットのエラー検出を行う．１ビットエラーはマスク可能な割込みとして報告され，２ビットエラーはCPU 停止となる．

主記憶とキャッシュ間のデータ転送は，４語単位のブロック転送を行い，ダイナミックRAM のニブル・モードを利用して高速化を図っている．

Table. 1 Cache Orders

bit	operation	remarks
0000	(NOP)	no operation
0001	WAIT_PDR	wait PDR data
0010	WAIT_CDR	wait CDR data
0011	WAIT_DR	wait PDR or CDR data
0100	READ_INTERNAL(lar , dr)	read internal registers
0101	WAIT_BUSY	wait for memory busy
0110	FORCE_DIRECTORY_ERROR(lar)	diagnose
0111	CLEAR(lar)	invalidate block
1000	CLEAR_WITH_SAVE(lar)	invalidate block with write back
1001	READ(lar , dr)	read in normal mode
1010	READ_BYPASS(lar , dr)	read in cache bypass mode
1011	READ_PHYSICAL(lar , dr)	read with physical address
1100	WRITE_STACK(lar , dr)	write without read in
1101	WRITE(lar , dr)	write in normal mode
1110	WRITE_BYPASS(lar , dr)	write in cache bypass mode
1111	WRITE_PHYSICAL(lar , dr)	write with physical address

** " lar " ; PLAR / PLAR++ / CLAR / CLAR++
** " dr " ; PDR / CDR

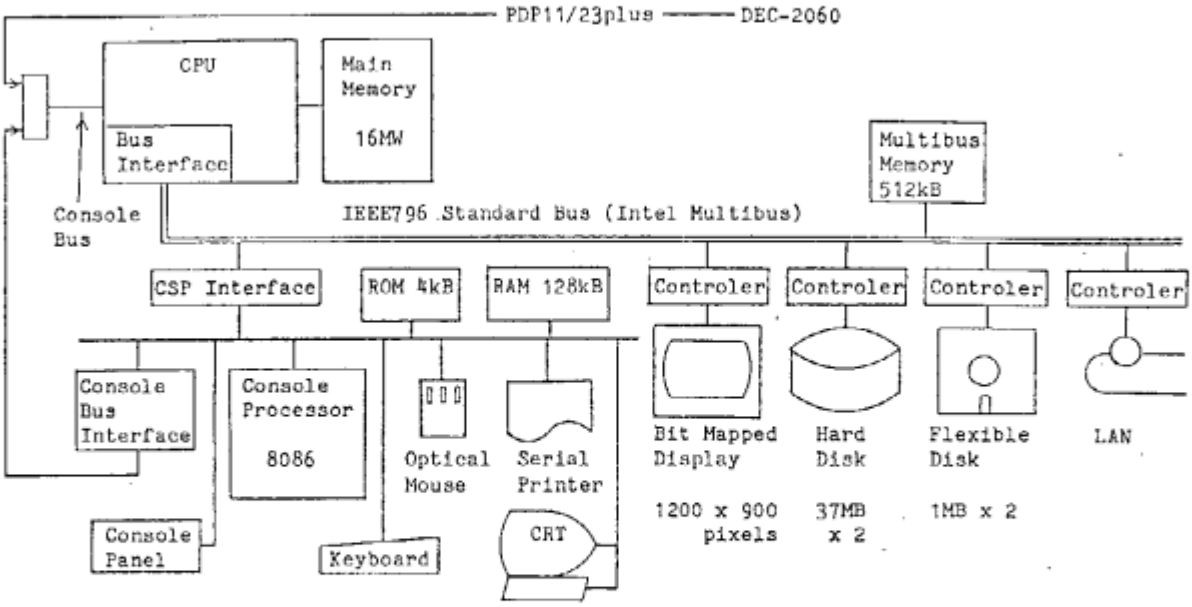


Fig.12　入出力部の構成

5．4　入出力部

5．4．1　入出力部の構成

構成：　入出力部は，入出力機器とそのコントローラ，IEE796規格の標準入出力バス（インテル・マルチバス）から構成され，入出力バスは24ビットのアドレス・レジスタと16ビットのデータ・レジスタを経由してCPU の内部バスに接続される．Fig.12にその構成を示す．Fig.6 に示したコンソールプロセサ（csp)は，実際には低速の入出力デバイスのコントローラとしても使用され，入出力バスとのインタフェースを持っている．入出力バス上には主記憶とは別に 512Kバイトのメモリがあり，ディスク装置やLAN へデータ転送する時のバッファ・メモリとして，また入出力機器のコントローラに対する制御ブロックの格納場所として利用している．

入出力バス：　入出力バスには，バスタイミングを制御する簡単な制御部があり，タイプ3のマイクロ命令のBCF により指定される読み出し，書込み，バイトアクセス，CPU との同期などのオーダを受け付ける．CPU から入出力バスを見たときのアドレス空間は，バス用のメモリ空間とI/O アドレス空間を一元化して扱えるようにハードウェアを構成してあり，16進の000000からEFFFFFまでがメモリ空間，F00000からFFFFFFまでをI/O 空間に割り当てている．入出力バスのタイムアウトやコモンバスリクエスト信号は，条件分岐命令でテストできる．

5．4．2　割込制御

PSI の割込システムには，ノンマスカブルのトラップと優先レベル判定の行われる割込とが有る．8つの割込レベルのうち5つを使用中で，内部割込としては温度・電源異常割込，タイマー割込，メモリエラー訂正報告割込がある．外部割込としては，入出力バス上の16本の割込線を任意の組合せで2つの割込レベルに割りつけることができる．トラップには，ハードウェア・エラー，ソフトウェア・エラーに関するもの，GC呼出し，デバッグ用のトレーストラップ等がある．

ハードウェアによる割込やトラップの原因は個別のフリップフロップに自動的にセットされ，ソフト的なトラップ原因も必要なものについてはフリップフロップに一旦セットしておき，割込ステータス・レジスタISTRを通して一括して参照される．この情報は割込ベクタをひくときに利用する．CPU の実行レベルは割込みマスク・レジスタIMSKR にセットされており，前述の個別フリップフロップの情報はプライオリティ・エンコーダで最も高い優先レベルの番号に変換されてIMSKR の値と比較される．そしてCPU の実行レベル（IMSKR の値）より割込み要求のレベルの方が高いときだけフラグが立つ．ファームウェア・インタプリタはKLO の命令実行の適当な隙間でそのフラグを判定し，割込処理へ移行する．

5．4．3．　入出力機器

入出力機器として次のようなものが接続されている．

(a) ビットマップディスプレイ（約1200x900ドット）
(b) 光学式マウス
(c) キーボード
(d) 固定式ディスク（37メガバイト2台）
(e) フレキシブルディスク（1メガバイト2台）
(f) ローカルエリアネットワーク（LAN）
(g) シリアルプリンタ
(h) CRT ディスプレイ（デバッグ用）

このなかで(b),(c),(g),(h) は，コンソールプロセサが機器のコントローラを兼ねており，そのほかは独自のコントローラを持っている．特に，ビットマップディスプレイ

は，画面１０数枚分のローカル・メモリを持ち，ローカル・メモリ上でデータの転送や論理演算等のできるラスタ・オペレーション機能を備えている．文字フォントやウィンドーの画面イメージは，通常ローカル・メモリ上に蓄えるようにし，入出力バスの負荷を軽減している．

５．５　保守機構とデバッグ・サポート

５．５．１　コンソールプロセサ

コンソールプロセサの機能のうち，入出力機器制御を除いた主な機能として次のものがある．

保守コンソール機能：　CPU 内のレジスタやメモリを参照，変更したり，CPU の起動，停止を行う．コマンドやデータの入出力は，CRT ディスプレイを通して行う．またCPU のエラー停止を検出しエラーログを取る．

立上げ機能：　入出力バス上のフレキシブルディスクまたは固定ディスクから，CPU にマイクロプログラムやローダのプログラムをロードし，CPU をブートストラップ・スタートさせる．

デバッグ・サポート機能：　マイクロ命令や機械語命令のステップ実行，ブレークポイントの設定／解除等を行う．またファームウェアやOSカーネル部のデバッグ用に，メモリ内の指定箇所に格納された文字列をCPU 停止時に読み出して画面表示するディスプレイ・コンソール機能をサポートする．

コンソールプロセサのCPU には，8086マイクロプロセサを使用しており，8086の実行するプログラムは，フレキシブルディスクから 128Kバイトのローカル・メモリにロードされる．コンソールプロセサは，CPU の内部バスとは独立の８ビットバス（コンソール・バス）を通してCPU にアクセスする．

５．５．２　CPU 内の保守機構とデバッグ・サポート

CPU の信頼性向上のため，主記憶には前述の通りエラー訂正機構を設け，また内部バスや全てのレジスタにはパリティ・ビットを付加して，ハードウェアエラー検出時にCPU を停止させるようにしている．CPU の停止原因，エラー部位は，コンソールプロセサから参照できる．

CPU 内のハードウェアによるデバッグ・サポート機構として，マイクロ命令アドレスのトレース・メモリ（ 256ステップ分）と，マイクロ命令中のブレークビットがある．　またファームウェアによるデバッグ・サポート機構としては，前述のディスプレイ・コンソールをサポートする組込述語や，KL0 命令用のアドレス・ブレーク，データ比較によるブレーク，アドレス・トレース，データ・トレース等を用意して，OSが立上がるまでのデバッグに活用している．

５．５．３　デバッグ用ホストCPU(PDP11)

PSI のコンソール・バスには，デバッグ用のホストCPUとしてPDP11/23PLUSを接続できるように設計してあり，コンソールプロセサと切替えて使用できる．PDP11 の主な役割は，コンソールプロセサと同様のデバッグ機能をサポートするほかに，DEC-20G0上で開発したファームウェアやKL0 のプログラムをPSI にダウンライン・ローディングすることである．PDP11 では，コマンドファイルの機能を強化してあり，レジスタやメモリの参照変更コマンドと条件分岐コマンドを組合せて，簡単なデバッグ用や評価用のプログラムを作ることができる．この機能と，PSI CPU のなかにある評価カウンタとを組合せて，CPU 評価のための計測を行う予定である．

５．６　実装

PSI は部品の調達のしやすさと小型化の点から，市販の高速ショットキーTTL ICと高集積度のMOS メモリを中心に実装を行った．CPU 部分は，IC 160個程度を搭載できるプリント基盤１２枚から構成され，主記憶部分は最大実装状態で同じサイズのプリント基盤１６枚から構成される．そのほかに，入出力機器コントローラの基盤１０枚程度と固定ディスク，フレキシブルディスクを合せて，１つの筐体に実装している．

６．　ファームウェア開発

ファームウェアの概要：　PSI のファームウェアは，KL0 命令実行用のインタプリタ核部，組込述語部，割込等のオペレーティングシステムサポート部の３つから構成され，現バージョンは全体で約１２Ｋステップの容量を持つ．

インタプリタ核部は，基本コントロールとユニフィケー

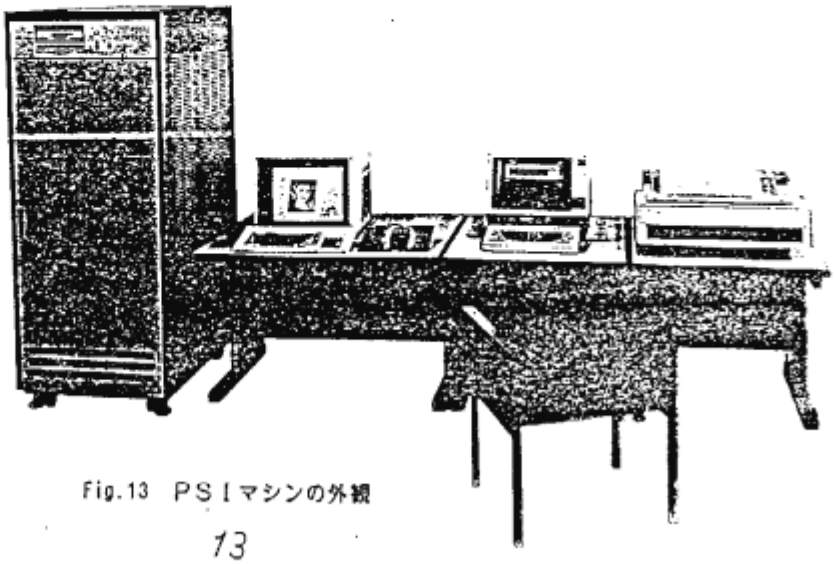


Fig.13　ＰＳＩマシンの外観

ションにほぼ同量のマイクロプログラムが必要であり，全体で約１．６Ｋステップとなっている．

組込述語部は各組込述語毎に約50から 100ステップのマイクロプログラム・ルーチンで構成される．約 150種の組込述語の記述に，ほぼ９Ｋステップを要している．

オペレーティングシステムサポート部は，割込処理，プロセス切替，メモリ管理のための各インタフェースを実現するマイクロプログラム・ルーチンよりなり，全体で約１．５Ｋステップとなっている．

開発サポートソフトウェア：　ファームウェアの開発はDEC-2060上で行い，シミュレータを用いてある程度のデバッグを終えたものを，PDP11 経由でPSI にダウンライン・ローディングし実機デバッグを行っている．

DEC-2060上のサポートソフトウェアとしては，prologで記述したマイクロプログラムアセンブラとリンカ，それにPascalで記述したマイクロシミュレータがある．マイクロアセンブラは，データ操作や分岐の指定を高級言語風のシンタクスで記述できるようにしてあり，禁止オペレーションの自動検出や，メモリアクセスに関する同期命令の自動挿入を行っている．シミュレータは，レジスタ転送レベルでPSI の動作をシミュレートするもので，高い精度でバグ検出するのに役だっている．そのほかに機械語命令用としてESP コンパイラ，KL0 コンパイラが用意されており，これらもPrologで記述されている．

７．おわりに

本論文ではICOTが中心となって開発を進めているパーソナル逐次型推論マシンPSI について，そのシステム構成，ハードウェアの設計方針，ハードウェア各部の詳細仕様に関する報告を行ない，核言語KL0 を効率良く実行するために導入した各種のハードウェアが，命令実行時にどのように使用されるかについて述べた．

PSI は，ハードウェアの試作が完了し，ファームウェアの基本部分もほぼテストを終了して，OSのブートストラッピング・システムの実機テストを開始しようとしている．

今後は，ベンチマーク・プログラムを用いてPSI の処理速度を正確に測定するとともに，ファームウェアインタプリタによる命令実行方式の長所，短所を明確にするための評価を行ない，低レベルで決定的な命令コードを持つパイプラインマシン（[Warren 84] など）とのアーキテクチャ上の比較検討を行なってゆく．またLSI 化のための再設計に向けて，ハードウェアの評価計測データを積み重ねるとともに，マルチCPU 化のためのアーキテクチャの見直しも行なってゆく予定である．

最後に，本研究の機会を与えて下さった渕一博研究所長，横井俊夫第３研究室室長，検討を通じて多くの有益な助言を下さった近山隆氏はじめICOTメンバ諸氏に深謝する．またTRO の最適化方式について助言下さった　David H. D. Warren氏には，特に記して感謝の意を表す．

参考文献

[Bowen 81]　D.L. Bowen : DEC system-10 PROLOG USER'S MANUAL, 15-DEC.-81 Department of Artificial Intelligence, University of Edinburgh

[Boyer 72]　R.S. Boyer and J.S.Moore :The Sharing of Structure in Theorem Proving Programs, Machine Intelligence Vol.1-7,Edinburgh Up (1972)

[Chikayama 83-1]　T.Chikayama : Fifth Generation Kernel Language, Proc. of the Logic Programming Conference '83,Tokyo,March 22-24 '83 pp.7.1-1～10

[Chikayama 83-2]　T.Chikayama : ESP-Extended Self-contained Prolog-as a Preliminary Kernel Language of Fifth Generation Computers, New Generation Cpmputing Vol.1 no.1 '83 Ohmsha, Ltd.

[Chikayama 84-1]　T.Chikayama : KL0 Reference Manual ICOT technical Report (to appear)

[Chikayama 84-2]　T.Chikayama : ESP Reference Manual ICOT Technical Report TR-044, Feb.3 (1984)

[Cohen 81]　J.Cohen : Garbage Collection of Linked Data Structures, Computing Serveys, 13-3(1981)

[Hattori 83]　T.Hattori et al : Basic Construct of SIM Operating System, New Generation Computing Vol.1,No.1,1983

[Horris 79]　F.L. Horris : A Time- and Space-Efficient Garbage Compaction Algorithm, CACM 22-10 (1979)

[Nishikawa 83-1]　H.Nishikawa, H.Yokota, A.Yamamoto, K.Taki, and S.Uchida : The Personal Sequential Inference Machine (PSI): Its Design and Machine Architecture, Proc. of Logic Programming Workshop, Algrave/PORTUGAL,June'83 pp.53-73

[Tick 84]　Evan Tick and David H.D.Warren : Towards a Pipelined PROLOG Processor, Proc. of the International Symposium on Logic Programming, Feb. 6-9 , 1984, pp.29-40

[Uchida 83]　S.Uchida, H.Yokota, K.Taki, and H.Nishikawa : Outline of the Personal Sequential Inference Machine : PSI, New Generation Computing Vol.1,No.1,'83 Ohmsha, ltd.

[Warren 77] David H.D.Warren : Implementing Prolog Compiling Predicate Logic Programs, Vol.1, 2, D.A.I Research Report No.39,40, Univ. of Edinburgh, 1977

[Warren 80] David.H.D.Warren : An Improved Prolog Impremention which optimises Tail Recursion, Proc. of the Logic Programming workshop, Hungary (July 1980)

[Yokota 83-1] M.Yokota, A.Yamamoto, K.Taki, H.Nishikawa, and S.Uchida : The Design and Impremention of a Personal Sequential Inference Machine:PSI, New Generation Computing Vol.1 No.2 '83 Ohmsha, ltd.

[高木 83] 高木茂行，近山隆，横田実，服部隆：拡張制御構造のPrologへの導入，情報処理学会第26回全国大会 '83年 3月 No.4D-11

[渥 83] 渥和男，西川宏，横田実，山本明，内田俊一：パーソナル逐次型推論マシンφ ── そのアーキテクチャ ──，情報処理学会第26回全国大会 '83年 3月 No.4N-2

[西川 83-2] 西川宏，横田実，山本明，渥和男，内田俊一，：逐次型パーソナル推論マシンφの設計思想とそのアーキテクチャ， Proc. of The Logic Programming Conference '83, Tokyo, March 22-24 '83 pp.7.2-1～12

[西川 83-3] 西川宏，渥和男，横田実，山本明，内田俊一：パーソナル逐次型推論マシンφ ── そのハードウェア ──，情報処理学会第27回全国大会 '83年 10月 No.5E-6

[山本 83] 山本明，横田実，西川宏，渥和男，内田俊一：パーソナル逐次型推論マシンφ ── そのマイクロインタプリタ ──，情報処理学会第27回全国大会 '83 年10月 No.5E-7

[横田 83-2] 横田実，山本明，西川宏，渥和男，内田俊一：パーソナル逐次型推論マシンφ ── その設計思想 ──，情報処理学会第26回全国大会 '83年 3月 No.4N-1