解　説

論理型言語指向の推論マシン

# 4. 逐次型推論マシンのアーキテクチャ†

金田悠紀夫††　松田秀雄††

## 1. はじめに

Prolog の処理系が汎用機上で作られ始めたのは 1970 年代の初めである．当初はインタプリタしかなかったが D. H. D. Warren らによりコンパイラをもつ処理系である DEC-10 Prolog が作成され[1]，汎用機上の Lisp 処理系とほぼ同じくらいの実行速度で動くようになった．

1982 年に第 5 世代プロジェクトが開始され，その最初の 3 年間である前期に，逐次型推論マシン PSI (Personal Sequential Inference Machine)[2)~4)] が開発されたが，それには DEC-10 Prolog の方式が参考になっている．PSI はデータ型判定のためのハードウェアや動的なメモリ割当てを支援する論理アドレッシング機構などのハードウェアを備えており，実行速度はマイクロプログラムによるインタプリタの実行で，約 30 KLIPS (Kilo Logical Inferences Per Second) であった．

PSI と同時期に，やはり DEC-10 Prolog の方式を参考にした逐次型推論マシンとして神戸大の PEK[5)] が開発されている．PEK では，ユニフィケーション専用回路，自動トレイル回路，自動アンドゥ (undo) 回路を備え PSI よりもさらに専用化，ハードウェア化を押し進めており，マイクロプログラムによるインタプリタでの実行速度は約 60 KLIPS であった．これら DEC-10 Prolog の方式をベースにしたマシンは逐次型推論マシンの第 1 期といえる[6)]．

その後，Warren は VAX のアーキテクチャを参考に新しいコンパイル技法を開発した[7)]．この方式では，Prolog のプログラムを仮想マシン (Warren Abstract Machine を略して WAM と呼ばれる) の命令にコンパイルして実行する．WAM 方式の特徴は，ゴール引数や一時変数などをレジスタに置きユニフィケーション時のデータアクセスを高速化する引数レジスタ方式，節の呼出しを高速化するインデキシング機能，終端再帰の最適化 (tail recursion optimization, 略して TRO と呼ぶ) の一般化などであるが，それらに加えてコンパイラによる最適化を徹底的に行い不要なデータ転送をできるだけ取り除くことも重要な特徴である．このような最適化により実行速度は格段に向上する．たとえば，前述の PEK はこの WAM に基づいたコンパイラにより append で約 430 KLIPS という高い性能を実現している[8)]．

WAM 方式は，ハードウェアとして基本的にはデータ型の判定機構とレジスタファイルがあれば良く，汎用機でも比較的高い性能が出せることから，現在では論理型言語の処理系を実現する際の標準的な手法となった．そして，最近発表される逐次型推論マシンは，全てといって良いほど WAM をベースにしている．これらは逐次型推論マシンの第 2 期といえる．本稿ではこの第 2 期のマシンを中心に解説することにする．

## 2. 逐次型推論マシンの構成

### 2.1 逐次推論マシンの動作

WAM 方式の逐次型推論マシンでは，基本的には以下のような動作の繰返しで実行を進める．

（a） ゴール引数をレジスタへ設定
（b） 節の呼出し
（c） ユニフィケーション

たとえば，次のようなプログラムを考える．

?-$p(X, b)$.
$p([a], Y)$ :-$q(Y)$.

このプログラムは WAM 方式による実行の場合，コンパイラにより，上のゴールは，

put_variable $X1$, $A1$　　(1)
put_constant $b$, $A2$　　(2)

† Architecture of Sequential Inference Machines by Yukio KANEDA and Hideo MATSUDA (Dept. of Systems Engineering, Faculty of Engineering, Kobe University).

†† 神戸大学工学部システム工学科

execute $p$ (3)

下の節は,

$p$: get_list $A$ 1 (4)
unify_constant $a$ (5)
unify_nil (6)
get_variable $A$ 1, $A$ 2 (7)
execute $q$ (8)

といった命令列に展開される．ここで，(1)，(2)が「(a) ゴール引数をレジスタに設定」に相当し，変数 $X$，定数 $b$ を引数レジスタにセットしている．(3)，(8)は「(b) 節の呼出し」に相当し，それぞれヘッドの述語名が $p, q$ の節を呼び出している．呼出しは通常は call 命令であるが節の最後の呼出しであるので execute 命令となる．(4)，(5)，(6)，(7)は「(c) ユニフィケーション」に相当し，これには引数レジスタの値とのユニフィケーションを行う get 命令，ヘッドにリスト，構造体が来た場合さらにその要素データに対してユニフィケーションを行う unify 命令がある．

以上のほかに，ユニフィケーションが失敗したとき，

「(d) バックトラッキング」という動作が必要である．このためには，あらかじめレジスタの値などの状態をローカルスタックに退避しておき(この領域を選択点フレームと呼ぶ)，バックトラッキング時にここから元の状態を回復する．これには変数の値を元に戻すためのトレイル処理も必要であり，トレイルスタックという領域を使用する．

たとえば前の例では，ヘッドの述語名が $p$ である節がほかにも存在したとすると，節の呼出し後に選択点フレームが作成され，状態の退避が行われる．その後，ユニフィケーションで変数 $X$ が［$a$］に束縛されるが，もしゴール $q$ が失敗すると選択点フレームからの状態の回復とともに変数 $X$ を未束縛状態に戻す必要がある（この処理をアンドゥ (undo) と呼ぶ)．このため変数 $X$ の値セルのアドレスをトレイルスタックに積んでおき，アンドゥはこれにより行う．

### 2.2 パイプラインプロセッサ

逐次型推論マシンがどのようなアーキテクチャをとるかを，WAM 方式のマシンの中では最も初期に提案されたパイプラインプロセッサ (Pipelined Prolog Processor)[9] を例にあげて示す．

パイプラインプロセッサは後述するように 4 段のパイプライン制御によりマシン命令を実行する．マシン命令は WAM 命令をパイプライン制御のために一部改良した命令となっている．命令フェッチと解読処理を行う I-Unit と命令の実行を行う E-Unit の二つのユニットからなる．プロセッサを I-Unit と E-Unit とに分離したのは，プロセッサ内部のモジュール化をはかることとともに，単一の I-Unit に複数の E-Unit を組み合わせた多重パイプラインプロセッサを構成する狙いがある．

#### 2.2.1 アーキテクチャサポート

パイプラインプロセッサでは，WAM の機能を実現するため以下のようなアーキテクチャサポートを行っている．これらのサポートは以降で述べる多くの逐次型推論マシンでも採用されている．

(1) レジスタおよびスタック・キャッシュ

2.1 で述べた動作を行うためには，まず引数および実行時の状態を保持するのに十分な数のレジスタファイルが必要である．また，ヒープ，ローカルスタック，トレイルスタックといったデータ領域のアクセスは，バックトラッキングの動作が加わるため，単純なプッシュ/ポップだけではなくランダムな参照も含まれる．したがって，ハードウェアスタックはあまり有効ではなく，キャッシュやメモリのインタリーブによりアクセスの高速化をはかる必要がある．

パイプラインプロセッサは E-Unit (図-1) にこれらのハードウェアを実装している．レジスタ P, CP, E, B, S, H, A, HB, TR は WAM の同名のレジスタと同じ働きをする．さらに，ローカルスタック，トレイルスタック，PDL (構造体同士の複雑なユニフィケーション時に一時的なデータ格納領域として使われる) の各領域についてはコピーバック型のキャッシュとして働くバッファが E-Unit 内部に設けられている．これらのうちのいくつかは，図-1 のデータパスで示されているように 2 ポート構成をとっている．

(2) マイクロプログラム制御

パイプラインプロセッサはマイクロプログラムで制御されており，E-Unit の数だけマイクロプログラム・コントローラを装備している．I-Unit はマシン命令を解読し，制御記憶中に保持された

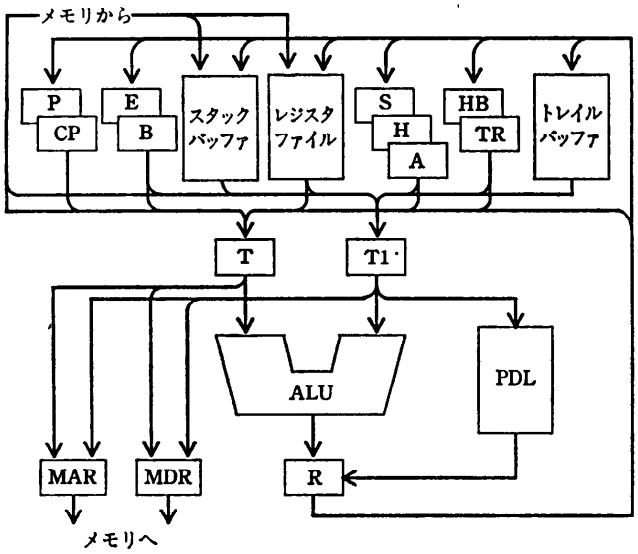


図-1　パイプラインプロセッサの E-Unit の構成

マシン命令に対応するマイクロプログラムのアドレスを決定する．マイクロプログラム・コントローラは，これにより E-Unit の実行を制御する．

（3）　タグアーキテクチャ

パイプラインプロセッサは論理型言語のデータ型に対応するタグをチェックするハードウェアをもついわゆるタグアーキテクチャを採用している．タグの処理はジャンプの条件の生成などで必要であり，その高速処理はジャンプ命令などの実行サイクルに大きく影響する．パイプラインプロセッサではこのためのハードウェアを備えている．

（4）　パイプライン処理

パイプラインプロセッサのパイプライン動作は以下の4つのステージからなる．

- Dステージ：マイクロ命令のフェッチと解読を行う．
- Cステージ：レジスタおよびバッファからデータを読み出す．その値は T または T1 にラッチされる．
- Eステージ：ALU, PDL, MAR, MDR の操作を行う．ALU と PDL の操作結果は R にラッチされる．
- Pステージ：メモリまたは R からレジスタまたはバッファにデータを書き込む．

これらのステージがどのように働くかを決定的 append の実行動作を例にとって説明する．append プログラム

```
append ([ ], L, L).
append ([X|L1], L2, [X|L3]) :-
    append (L1, L2, L3).
```

が決定的動作を行う場合，その実行は最後の1回の呼出しを除いて2番目の節に集中する．2番目の節をマシン命令にコンパイルすると，

```
prefetch append/3
get_list A1
unify_variable X4
unify_variable A1
get_list A3
unify_value X4
unify_variable A3
jump
```

となる．これの実行をサイクルごとに示したものを図-2 に示す．上の8個のマシン命令は実行時に動的に 12 個のマイクロ命令に展開される．図-2 の右に各マシン命令に対応するマイクロ命令シーケンスの開始位置を示す．マシン命令の前にマイクロ命令の番号を付けている．prefetch と jump の実行サイクルは，これらの命令が E-Unit とは並行に動作する I-Unit により実行されるので，この図には現れない．

図-2 の左にこの 12 個のマイクロ命令の動作タイムチャートを示す．縦はサイクル数，横は各マイクロ命令のパイプライン動作による進行を表している．図-2 では左の各マイクロ命令の位置がその命令の実行開始サイクルを表している．な


図-2　決定的 append のタイムチャート

お，この実行ではメモリはインタリーブされており，一つのアクセスに4サイクルかかるとしている．Pステージの前の空白の4サイクルがメモリアクセスのサイクルを表す．最初の get_list 命令(最初の二つのマイクロ命令で実行されている)では append ゴールの第1引数（レジスタ $A1$ に格納されている）がリストかどうかの判定を行うだけなのでメモリアクセスはない．

### 2.2.2 パイプラインプロセッサの性能

決定的 append のプログラムの場合，パイプラインプロセッサが1サイクル 100 nsec で動作すると仮定すると，450 KLIPS の実行速度となる．

パイプラインプロセッサは，実際に作成されていないペーパーマシンであるが，論理型言語の実行動作をフォンノイマン型アーキテクチャの上でパイプライン処理により高速実行できる可能性を示した意義は大きい．バークレイで開発された PLM[10] はこのパイプラインプロセッサをほぼそのまま実現したものであり，その性能は約 400 KLIPS とされている．

## 3. 逐次型推論マシンの実例

以下では WAM 方式の逐次型推論マシンの実例について述べる．これらのマシンではパイプラインプロセッサのところで述べた，2ポート構成のレジスタ，マイクロプログラム制御 (RISC によりプロセッサを構成するマシンもある)，タグアーキテクチャ，パイプラインまたは命令プリフェッチのためのハードウェアなどのアーキテクチャサポートを行っているものが多い．また，さらに，以下のような高速化のための工夫をしている．

- マイクロ命令コードの高機能化

マイクロプログラム制御のマシンの場合，WCS (Writable Control Storage) に多ビット長の水平型マイクロ命令コードを格納し，一つのサイクル内で多くの処理を実行する．

- 実行時のモード指定

一般に引数が入力変数なのか出力変数なのか実行時でなければ判定できないので，プロセッサモードを設け実行時のモード指定により命令の機能を修正し余分なサイクル時間の節約をしている．

- 複合命令の導入

WAM ではリストのユニフィケーションなどで特定の命令の組合せが繰返し実行されることが多い．これらの組合せをまとめて一つの新しいマシン命令にまとめ，実行時間の短縮をはかる．

- Prolog データ型のサポート

基本データ型として，変数，定数，構造体，リストの4つがタグの形で表される．マシンによってはより多くのデータ型が表現できるものもある．

- ガーベージコレクションのサポート

タグ部にガーベージコレクションのためのビットをもち，そのデータが有効かどうかなどを示すことができる．

- デリファレンスのサポート

アクセスしたデータがほかのデータへのポインタであったとき，そのポインタで指されているデータにアクセスする必要がある．この動作をデリファレンスと呼ぶ．逐次型推論マシンの中にはデリファレンスを行うハードウェアが装備されているものがある．

### 3.1 PSI-II

PSI-II[11] は PSI の改良版であるが，DEC-10 Prolog 方式から WAM 方式に変更したことによりアーキテクチャを大幅に変更している．また，PSI-II はスタンドアローンのワークステーションであるだけでなく複数台を接続して Multi-PSI の要素プロセッサとしても使うので，ハードウェア量を抑える必要があった．このため PSI と比べて LSI 化が進んでおり，基板の枚数にして 1/4 程度に抑えられている．

PSI では Prolog (正確には KL0) プログラムをそのまま変換した表形式[4] のマシン命令を採用していたが，PSI-II では WAM をベースに最適化のための命令や組込み述語用命令を拡張した命令セットになった．これを 53 ビット幅の水平型マイクロプログラムで解釈実行する方式をとっている．また，ハードウェア構成も PSI と比べてシンプルなものとなっている．さらに，主記憶容量は最大実装 64 M 語に拡張されている．

実行速度は，サイクル時間 166.7 nsec (24 MHz) で決定的 append の実行が 400 KLIPS と PSI と比べて 10 倍以上高速になっている．なお，PSI-II の後継機[12]が発表されており，サイクル時間 60 nsec の5段のパイプラインとデータ型判定やデリファレンスのハードウェアなどで約 1.3 MLIPS を達成している．

#### 3.1.1 高速化のための工夫

PSI-Ⅱ では論理型言語の高速実行のために次のような工夫をしている.

- レジスタ

2ポート構成の引数レジスタ，汎用レジスタがともに 32 個ある．また，命令フェッチ用とデータアクセス用に2組のデータ/アドレスレジスタが用意されている.

- マシン命令のプリフェッチ機構

3段のプリフェッチ機構が備わっている.

- タグアーキテクチャ

データの語形式は PSI と同様タグ部8ビット，値部 32 ビットの 40 ビット構成となっている.

- 論理アドレッシング機構

PSI と同様，メモリの動的割付け機構として論理アドレッシング機構がある．論理型言語の実行ではヒープ，ローカルスタック，トレイルスタックといった領域をアクセスするが，各領域に必要な大きさは応用プログラムの特性に依存し，あらかじめ決めることはできない．そこで，論理アドレッシング機構によりプロセスごとに複数の自由に拡張可能な論理空間を提供している．また，ページ単位で動的なメモリ割当てをサポートするためメモリ領域不足を検出するハードウェア機構を備えている.

### 3.2 CHI

日本電気の CHI (Co-operative High Performance Inference Machine)[13] は WAM 方式の逐次型推論マシンであり，CHI-Ⅰ と CHI-Ⅱ の二つがある．いずれも UNIX ワークステーションをフロントエンドとするバックエンド型マシンであり，WAM ベースのマシン命令をマイクロプログラムにより実行する.

CHI-Ⅰ は ECL で作られていたためパーソナル・ユースにはサイズが大きすぎたので，CHI-Ⅱ では CMOS ゲートアレイを使用しデスクサイドサイズに抑えている．また，より大規模なプログラムを実行するため，主記憶容量を CHI-Ⅰ で 64 M 語（1語 36 ビット）であったのが CHI-Ⅱ では 128 M 語（1語 40 ビット）と非常に大きなものとしている.

マシンのサイクル時間は，素子の変更により CHI-Ⅰ では 100 nsec であったのが CHI-Ⅱ では 170 nsec と遅くなった．しかし，推論速度そのものは以下に述べるような改良により，決定的 append が CHI-Ⅰ の 285 KLIPS から CHI-Ⅱ では 490 KLIPS と向上している.

#### 3.2.1 マイクロ命令ステップ数の短縮

CHI-Ⅱ の設計方針として，主記憶のアクセス回数に対するマイクロ命令ステップ数を可能なかぎり短縮するようにしている．タグ部の検査，値部の比較，ゴールの引数アクセス，命令プリフェッチ，ジャンプなどの処理を可能なかぎり同一のマイクロ命令ステップ内で行う.

#### 3.2.2 命令セット

CHI-Ⅱ では多くの複合命令を導入し，マイクロ命令のステップ数の短縮を行っている．これにより，決定的 append プログラムの実行で 1 LI (Logical Inference) あたりマシン命令で5個，マイクロ命令で8ステップを短縮することができる.

### 3.3 AIP

東芝で開発されたマシンで RISC 思想を取り入れた AIP (AI Processor) と呼ばれるプロセッサを用いて構成されている[14]．システムはバックエンド型のマシンとなり，UNIX ワークステーションに AIP を搭載したボードを付加することにより実現される．最高性能は 600 KLIPS 以上と言われている．AIP の中心部は IP 704 と呼ばれ WAM を想定して設計された RISC 型プロセッサである.

#### 3.3.1 IP 704 プロセッサ

IP 704 は 32 ビットマイクロプロセッサでその命令は 32 ビット固定長で形式も単純化されている．8ビットの命令コード，最大3個のレジスタ指定フィールド，直接またはディスプレースメントフィールドから構成されている.

3段のパイプライン制御方式を採用しており，命令フェッチ，解読，実行のサイクルとなっている．メモリ参照を行う命令はもう一つサイクルが必要となる．また，ジャンプ命令は命令フェッチハードウェアによりフェッチサイクルで解読されるので解読サイクルは不要となる．レジスタファイルは 32 本の 32 ビットレジスタとなっている.

#### 3.3.2 高速化のための工夫

Prolog プログラム高速実行のため次のような機能を採用している.

- 96 ビット長 4 K 語の WCS をもつ　データ

タグのチェックや挿入，多方向ジャンプや条件ジャンプ，主記憶アクセスと算術演算が1命令サイクルで実現される．

●タグは4ビットで16種のデータ型の判別ができる．

●命令キャッシュとデータキャッシュの二つを設けることにより，命令フェッチとデータアクセスを並行して行えるようになっている．

### 3.4 Pegasus

本マシンは三菱電機において開発された RISC の概念を取り入れた専用プロセッサを用いたマシンである[15]．プロセッサはトランジスタ数が約60000でサイクル時間が200 nsec，最高性能で240 KLIPS と言われている．

バックトラッキング機構実現をサポートする高速状態退避/回復機構を備えているのが特徴である．RISC 方式を取り入れているので命令は固定長1語であり1マシンサイクルで実行される．6段のパイプライン制御がなされており，命令フェッチは2サイクルごとに行われる．専用 LSI の特徴を活かしたアーキテクチャを採用している．

#### 3.4.1 バックトラッキングのサポート機能

2.1 で述べたように，バックトラッキングのためにレジスタの値の選択点フレームへの退避とそこからの回復が生じるが，これらは多くのデータ転送をともなうため負担の大きい処理となっている．

Pegasus ではカスタム LSI の特徴を活かしレジスタファイルの一部を相互にコピーできるレジスタペアとして実現している．一方を主レジスタ，他方を影レジスタと呼んでいる．選択点フレームを作るとき主レジスタの内容は影レジスタにコピーされる．逆にバックトラッキング時には影レジスタから主レジスタにコピーが行われる．これらの処理は1マシンサイクルで実現される．連続した選択点フレーム作成やバックトラッキング発生では影レジスタと主記憶間のデータ転送が発生するが，これらの転送は主記憶アクセスをともなわない命令とは並行して実行される．したがってこの処理を影オペレーションと呼んでいる(図-3)．

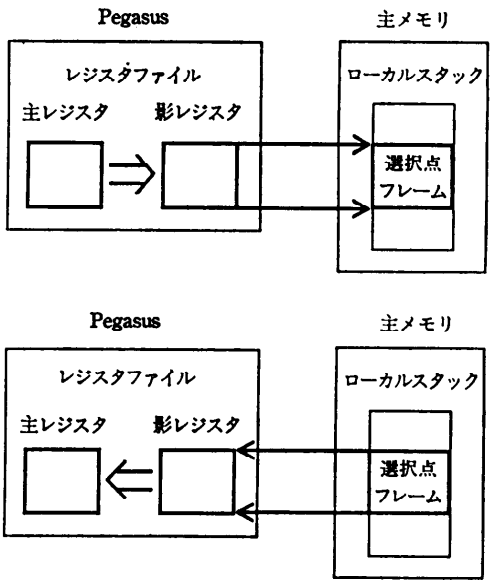


図-3 影オペレーション

#### 3.4.2 語形式

1語 40 ビットで 32 ビットがデータの値部，タグ部が上位8ビットで最上位2ビットはガーベージコレクション用，最下位2ビットで4つの基本データ型を示している．なお，RISC の特徴を活かすため命令も 40 ビット長になっている．

### 3.5 IPP

IPP (Integrated Prolog Processor)[16] は日立製作所が開発したマシンでスーパミニコンピュータに内蔵する形式のプロセッサである．内蔵型にすることにより既存のソフトウェアとの結合も容易になり，汎用機の高機能化に用いられている最新の高速 LSI，実装技術，5段の命令パイプライン制御機構などがそのまま使用できるという有利な点がある．一方語長など汎用機の制約を受け解決しなければならない問題点もある．高速化のために取り入れられた工夫を次にあげる．

#### 3.5.1 命令セット

Prolog 命令（WAM 命令に相当）とベースマシンの命令を一つの命令セットとしてまとめている．Prolog 命令はベースマシンの拡張命令コードが割り当てられている．本マシンでは高速の汎用命令と Prolog 命令とを効果的に利用しており，単純なロード・ストア命令は汎用命令で，複雑なユニフィケーション命令は専用命令で実現している．

### 3.5.2 データ形式

ベースマシンが 32 ビットマシンであるためタグ部分だけ語長をのばすことができない．そこで，基本的には1語 32 ビットでのうち4ビットをタグにし，28 ビットをデータの値部にしている．この形式ではアトム（番号）や 16 ビットの整数が表現できる．32 ビット整数，浮動小数点数，構造体，リストなどは，タグと実体を指すポインタとから構成される1語と実体から構成される．このようにして 32 ビットマシンと整合をとっている．

### 3.5.3 ハードウェアサポート

ハードウェアのサポートとしてはレジスタの補強，タグ処理ハードウェアの付加，トレイルスタック管理用とモード修飾のための回路が設けられている．また，Prolog 命令からは 16 本の汎用レジスタに加え，8本の浮動小数点レジスタを 16 本の付加汎用レジスタとしてアクセスできるようにしている．タグ処理としてはタグの切り出し，追加，タグによるデータ型判定があり汎用命令では負担の大きい処理となる．タグ格納レジスタ，タグ生成と切り出し，タグを用いたジャンプ制御のための回路を設けている．

## 3.6 KCM

KCM (Knowledge Crunching Machine)[17] は ECRC (European Computer-Industry Research Center) により開発されたマシンである．デスクトップ・ワークステーションをホストとするバックエンド型マシンとなっている．プロセッサ，主記憶，ホストインタフェースの3枚のボードから構成され，全てのボードがホストのキャビネット内に格納できる．

KCM ではホストである UNIX ワークステーションとのインタフェースを容易にするため，データの語形式をタグ部 32 ビット，値部 32 ビットとしている．タグ部 32 ビットの構成は，ガーベージコレクション用に2ビット，領域管理用（スタック，ヒープ，静的データなどの領域を区別するのに用いる）に3ビット，データ型の判定用に5ビットであり，残りのビットは使われない．またマシン命令の形式も1語 64 ビット固定の RISC 型としている．ただし，命令の解読・実行はマイクロプログラムで制御される．

1マシンサイクルは 80 nsec であり，主記憶は1ボードに 32 M バイト実装できる．性能はピーク時（決定的 append の実行）で 833 KLIPS となっている．

### 3.6.1 プロセッサの構成

プロセッサは制御ユニット，実行ユニット，プリフェッチ・ユニット，メモリ管理ユニットからなっている．ハーバードアーキテクチャを採用しており，命令用とデータ用の二つのキャッシュ（容量は共に 8K 語）が独立したバスでそれぞれプリフェッチ・ユニット，実行ユニットに接続されている．実行ユニットにはユニフィケーションとバックトラッキングをサポートするハードウェアが実装されており，以下これらについて説明する．

### 3.6.2 ユニフィケーションのサポート機能

ユニフィケーションの高速化のため MWAC (Multi-Way-Address-Calculator) がある．これは PROM で実現されており，ユニフィケーションされる二つのデータのタグを入力として4ビットの値を出力する．マイクロシーケンサはこの値により 16 方向のジャンプを行う．これにより，多様なデータのユニフィケーションを高速に実行できる．また，デリファレンスを行うハードウェアが装備されており，1サイクル当たり1段のデリファレンスが実行できる．

### 3.6.3 バックトラッキングのサポート機能

KCM では Pegasus と同様に影レジスタをもち，浅いバックトラッキング（次の候補節へのバックトラッキング）での選択点フレームの作成を抑えている．この判定のために shallow flag と choice point flag の二つのフラグがある．前者はユニフィケーションの初めに，まだ候補節があるときセットされ，実行が節のボディ部にまで進むとリセットされる．これがセットされているとき生じるのが浅いバックトラッキングである．後者は選択点フレームを作成した時点でセットされ，バックトラッキング時に影レジスタではなくスタック内の選択点フレームからの回復が必要なことを示す．

## 4. おわりに

本稿では逐次型推論マシンについて，WAM に対するアーキテクチャサポートを中心に解説した．Lisp マシンなどで従来から行われてきたマイ

クロプログラム制御，タグアーキテクチャに加えて，WAM 方式の採用によりレジスタファイル，キャッシュ，パイプライン動作など汎用機の高速化技術の導入が行われたため，最近の逐次型推論マシンは初期のマシンに比べて飛躍的に性能が向上している．

さらに最近の傾向としてマイクロプログラム制御のほかに AIP，Pegasus のように RISC によりプロセッサを構成するマシンが現れている．また，従来のソフトウェア資産の継承，ソフトウェア開発時間の短縮という点からワークステーションやミニコンピュータをフロントエンドとするバックエンド型マシンが多くなってきている．ウィンドウシステムなどユーザインタフェースのためのソフトウェアが大規模で複雑なものとなってきているため，今後はバックエンド型が主流になると考えられる．

## 参考文献


1) Warren, D. H. D.: Implementing Prolog—Compiling Predicate Logic Program, Vol. 1-2, D. A. I. Research Report, No. 39-40, Univ. of Edinburgh (1977).
2) Yokota, M. et al.: The Design and Implementation of a Personal Sequential Inference Machine: PSI, New Generation Computing, Vol. 1, No. 2 (1983).
3) Taki, K. et al.: Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI): Proc. of FGCS'84, pp. 398-409 (1984).
4) Yokota, M. et al.: A Microprogrammed Interpreter for the Personal Sequential Inference Machine, Proc. of FGCS '84, pp. 410-418 (1984).
5) 田村直之他：シーケンシャル実行型 Prolog マシン PEK，情報処理学会論文誌，Vol. 26, No. 5, pp. 855-861 (1986).
6) 横田 実：逐次型 Prolog マシン，人工知能学会誌，Vol. 2, No. 4, pp. 441-449 (1987).
7) Warren, D. H. D.: An Abstract Prolog Instruction Set, SRI Technical Note, No. 309 (1983).
8) 和田耕一他：Prolog マシン PEK における中間コードとその実行方式，情報処理学会論文誌，Vol. 30, No. 9, pp. 1231-1239 (1989).
9) Tick, E. and Warren, D. H. D.: Towards a Pipelined Prolog Processor, Proc. of Int. Symp. on Logic Programming, pp. 29-40 (1984).
10) Dobry, T. P. et al.: Performance Studies of a Prolog Machine Architecture, Proc. of 12th Int. Symp. Computer Architecture, pp. 180-190 (1985).
11) Nakashima, H. and Nakajima, K.: Hardware Architecture of the Sequential Inference Machine PSI-II, Proc. of 4th Symp. on Logic Programming, pp. 104-113 (1987).
12) 中島 浩他：PIM/*m* 要素プロセッサのアーキテクチャ，並列処理シンポジウム JSPP '90, pp. 145-151 (1990).
13) Habata, S. et al.: Co-operative High Performance Sequential Inference Machine: CHI, Proc. of Int. Conf. Computer Design, pp. 601-604 (1987).
14) 斎藤光男，岡本利夫：Prolog を 667 kLIPS で実行する RISC 風プロセッサ，日経エレクトロニクス，No. 436, pp. 225-245 (1987).
15) 瀬尾和男，横田 隆：Prolog 指向 RISC プロセッサ "Pegasus"，情報処理学会研究会報告，Vol. CA-63, No. 77, pp. 41-48 (1986).
16) Abe, S. et al.: High Performance Integrated Prolog Processor IPP, 14th Int. Symp. Computer Architecture, pp. 100-107 (1987).
17) Benker, H. et al.: KCM: A Knowledge Crunching Machine, Proc. of 16 th Int. Symp. Computer Architecture, pp. 186-194 (1989).


(平成 3 年 1 月 9 日受付)

**金田悠紀夫**（正会員）

昭和 15 年生．昭和 39 年神戸大学工学部電気工学科卒業．昭和 41 年神戸大学大学院電気工学専攻修士課程修了．昭和 41 年電気試験所（現電総研）入所．電子計算機研究に従事．昭和 51 年神戸大学工学部システム工学科，現教授．工学博士．コンピュータシステムのハードウェア，ソフトウェアの研究に従事．高級言語マシン，並列マシン，AI に興味を持っている．

**松田 秀雄**（正会員）

昭和 34 年生．昭和 57 年神戸大学理学部物理学科卒業．昭和 59 年同大学院工学研究科システム工学専攻（修士課程）修了．昭和 62 年同大学院自然科学研究科システム科学専攻（博士課程）修了．現在，神戸大学助手（システム工学科）．学術博士．論理型言語による並列処理などの研究に従事．日本ソフトウェア科学会，人工知能学会，IEEE，ACM 各会員．ICOT PIM ワーキンググループ委員．