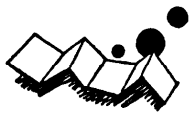
解　説



Prolog マシン†

金田悠紀夫††

1. はじめに

知識工学の分野の研究は主として LISP 言語を基本として進められてきた．MIT を始めとして多くの研究機関において開発された人工知能システムの大半は LISP 言語によって記述されている．これら LISP プログラムの開発は当初からタイムシェアリングシステムを用いて会話的に行われており，それまでのバッチ形のプログラム開発とは異なった形態をとるようになっていた．これは現在の高機能ワークステーションを用いたソフトウェア開発の先駆となっていたとも考えられる．しかしながら開発されるプログラムの機能が複雑化，高度化してくると，メモリ容量の制限や実行時間の遅さが問題となってきた．

このネックを打破するために研究開発が積極的に進められたのが LISP マシンであった[1]．内外で多くの研究が進められ成功をおさめており，特に MIT で開発された LISP マシンを基本にした Symbolics 社の LISP マシンは専用マシンとして商品化されて普及している．我が国においても商用化 LISP マシンの発表が行われている．

Prolog マシンの開発は LISP マシン開発と目的を一とするもので，マシンの高速化の要請と，新しいマシンアーキテクチャ研究とを目的として精力的に研究が進められている．Prolog マシンとして現在開発中のものは多くあるが，ここでは ICOT（新世代コンピュータ技術開発機構）の PSI マシン，神戸大学の PEK マシン，Warren と Tick 提案[4]のパイプライン形 Prolog マシンを取り上げ解説を行う．

2. Prolog マシンの要請

表-1 に D. H. D. Warren が用いている2つのベンチマーク用のプログラム[8]を用いていくつかの Prolog システムのプログラム実行時間の測定を行った結果を示す．この表は文献 7) から抜粋したものである．

DEC-2060，VAX 11/780 については内部クロックを用いて実行時間を測定しており，その他については外部クロックを用いてマニュアルで測定している．List reverse は 30 要素のリストを逆順に変換するプログラムであり，quicksort は 50 要素のリストをソートするプログラムである．まず気付くのは実行時間の遅さとそのバラツキの大きさである．このことは現在の汎用マシンのアーキテクチャが Prolog プログラムの実行機構とうまく合っていないことを示唆しており，実行時間のバラツキの大きさは設計思想の差，実現技法の洗練度に依ると考えられ今後一層の研究の余地があると考えられる．

パーソナルコンピュータの規模で高機能ワークス

表-1 Prolog プログラムの実行時間例

	List reverse (30要素) 秒	quicksort (50要素) 秒	平均の KLIPS
2060 インタプリタ	0.191	0.235	2.6
2060 コンパイラ	0.011	0.016	42
2060 コンパイラ（高速版）	0.0095	0.014	48
C-Prolog インタプリタ VAX 11/780	0.321	0.466	1.4
K-Prolog インタプリタ UX-300	1.068	1.304	0.46
Prolog KABA インタプリタ PC-9801 F (5 MHz)	0.722	0.920	0.67
Prolog 1 インタプリタ			
CP/M IF-800 (4.9 MHz)	16.7	10.2	0.045
CP/M-86 AS-100 (4.9 MHz)	17.3	11.1	0.042
Micro Prolog インタプリタ CP/M-86	8.334	7.323	0.071

† Prolog Machine by Yukio KANEDA (Department of Systems Engineering, Kobe University).

†† 神戸大学工学部システム工学科

テーションとして使え，しかも DEC 社2060コンピュータのコンパイラ版程度の処理能力を持つ専用マシンを開発するというのがアーキテクチャ研究の面からとユーザ側からとの要請となっておりマシンとして

（1） 実行速度が十分に速いこと

（2） 大規模な応用プログラムを実行できる十分なメモリ空間を持つこと

（3） ソフトウェア開発用の高機能ワークステーションとして利用できること

等の性能を持つことが要求される．

3. Prolog プログラムの実行機構

3.1 簡単なプログラムの実行例

簡単なプログラムの例として2つのリスト L1 と L2 を連結してリスト L3 を作るプログラム append を取り上げてみる．

① append([X|L 1], L 2, [X|L 3]) :-
　　append(L 1, L 2, L 3).

② append([ ], L, L).

に対して，入力

③ ?- append([a, b], [c, d], X).

を行うとプログラムは実行され，

X=[a, b, c, d]

yes

と応答してくる．

①，②，③は節と呼ばれ，それぞれ規則節，事実節，質問ゴール節を示している．①における :- 記号の左辺を頭部，右辺を本体と呼ぶ．ここで append は述語名であり（ ）内のコンマで区切られた要素列は引数で，大文字で始まる文字列は変数，小文字で始まる文字列は定数を表わしている．[…] はリストを表わしており [ ] は空リスト，[a, b, c, d] は定数 a, b, c, d, nil を線形連結したリストを示している．

[X|L 1] において X はリストの頭部の要素を示し L 1 は尾部のリストを示している．

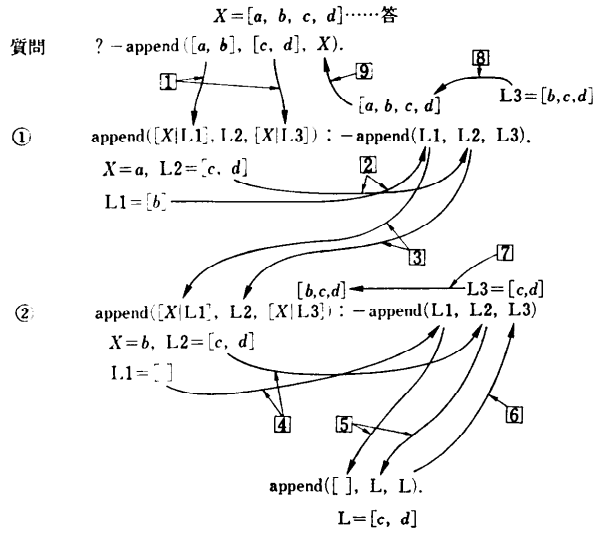
また [X|L 1] などのリストは構造体と呼ばれる表現の1つとなっており，一般に構造体は

構造体名（引数 1，引数 2，…，引数 n）

の形をしている．ここで構造体名は関数子（functor）と呼ばれる．[a, b] は LIST(a, LIST(b, nil)) の形で表現されるネスト構造を持つ構造体で LIST は関数子である．構造体の要素である変数をグローバル変数とよび，要素でない変数をローカル変数と呼びプログラム実行時点での取扱いが異なっている．

プログラム実行手順の1例を図-1に示す．

Prolog プログラムの実行の特徴としては(1)ユニフィケーションと(2)バックトラッキングがある．ある質問ゴール節を与えると，そのゴールと同一の述語名の左辺を持つ節を探し出し引数同士の一致を調べるユニフィケーションを試みる．ユニフィケーションに失敗すると同一の述語名の左辺を持つ次の節が試みられ節は上から下へ順に選択される．ユニフィケーションが成功すると，その節の本体の各項が新しいゴール列となり，左から右へ順番に実行される．



1→2→……→9 の順に値が求められていく．

図-1 append の計算過程

3.2 ユニフィケーション

同一の述語名の左辺を持つ節の集りをプロシージャと呼ぶがユニフィケーションはゴール節とそれに対応するプロシージャに含まれる節の頭部の持つ引数の一致を試みる操作で，プロシージャ中の呼ばれた節を基準にして考えると，呼ばれた節に現われる引数のタイプは次の3種のいずれかであり，ユニフィケーションの処理内容はそれに対応して決まる．ここで変数は初期値として“未定義”の値を持っている．

（1） 定数の場合はゴール側が同一の値が未定義なら成功

（2） 変数の場合はゴール側が何であっても成功

（3） 構造体の場合はゴール側も同一の構造

体か，未定義なら成功
となり，いずれか一方が未定義なら未定義変数に他方の値をセットする．両方とも未定義の場合には両者が同一の値を持つようにゴール側の変数セルのアドレスを呼ばれた側の変数セルにセットする．

3.3 バックトラッキング

Prolog ではプログラムの実行はゴール節を真にする解の探索であり，述語を真にする値を引数に代入していく過程である．ゴールとユニファイ可能な節が複数存在するときは上から順に1つずつ選択実行し，失敗すれば次の節を再試行してみるという方式をとる．

この再試行をバックトラッキングという．バックトラッキングには2つのモードがあり，同一プロシージャに属する別の節が残っている場合にはその節とのユニファイを試みる（浅いバックトラッキング）．一方同一プロシージャ内のすべての節をつくしてしまった場合には呼出しを行ったゴールそのものが失敗したことになり深いバックトラッキングの発生となる．

バックトラッキングを行うためには前試行でセットされた変数をもとに戻す undo オペレーションが必要となる．したがってバックトラッキングを行うには選択点と変数の代入環境を保持しておくことが必要となり計算実行の履歴を保存しておくことが必要となる．したがって計算の制御機構も複雑となり多くのメモリ容量が必要となる．またスタックを用いたプログラム実行制御が大切となる．ゴールがプロシージャ内の節を呼ぶときにはスタック上にフレームと呼ぶ領域を確保し，制御情報，変数の値の代入情報を格納して計算を進めるという制御方式をとっている．バックトラッキングが発生したときは，フレームをポップアップしてしまうので，フレーム上に取られた変数については自動的に消える（ローカル変数の場合）が，ユニフィケーションにより，現在のフレームより前のフレーム領域にとられた変数に対して代入が行われている場合には，該当変数を初期状態（未定義）に戻す必要がある．

この場合に変数の代入過程のトレースをスタック（トレイルスタック）に格納しておき，バックトラック時にトレイルスタックの情報を用いて変数の代入値をリセットしていくことになる．

したがってバックトラッキング処理の高速化には

① 変数セルの生成・消去を高速に行えるメモリ領域の割当機構

② 値の高速フェッチのためのメモリアクセス機構

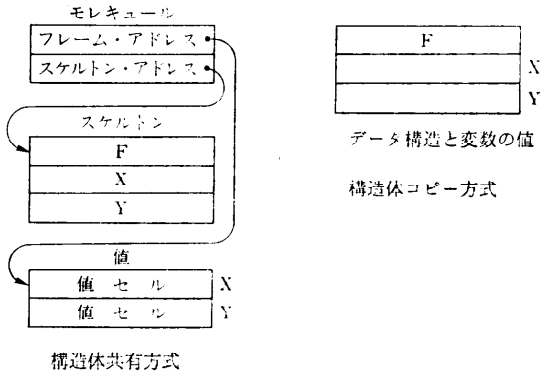


図-2 構造体 F(X, Y) の表現形式

③ 引数の型チェックの高速化のためのタグ付きデータ表現とチェック用ハードウェア回路

④ 大容量メモリスペース

が必要となる．

3.4 構造体共有方式と構造体コピー方式

構造体は複雑な入れ子構造が許されているためユニフィケーション過程におけるその取扱い方はマシン性能に大きな影響を与える．構造体コピー方式と，構造体共有方式の2つが提案されている．**図-2** に両方式での表現例を示す．

コピー方式は簡単な方式で節が呼出されるたびに引数となっている構造体自体のコピーを作りグローバルスタックと呼ぶスタックに格納していく方式である．一方，構造体共有方式はデータ構造とその値を別々に持たせるというもので，同一の構造体に対してデータ構造はひな型（スケルトン）として1つ持てばよく，構造体は値の配列と，そのスケルトンの対で表現される．したがって変数セルは2つのポインタを持ち，一方はスケルトンを指し，他方はグローバルスタック上での値の配列の先頭番地を指すことになる．この変数セルをモレキュールと呼んでいる．

ICOT の PSI マシンと神戸大学の PEK マシンは構造体共有方式を採用しており，Warren と Tick 提案のパイプライン制御形 Prolog マシンにおいては，構造体コピー方式を採用することを予定している．いずれの方式を選択するかはアーキテクチャに大きく依存する．

3.5 インタプリタ方式とコンパイル方式

Prolog プログラム実行の基本はプロシージャ内の節の呼出し，ユニフィケーション，バックトラッキングであり，スタックを中心とした計算となる．

例えば後述する ICOT の PSI システムでは，コ

ントロール，ローカル，グローバル，トレイルの4つのスタックを用いて計算が進められる．コントロールスタックはプログラムの実行環境および制御情報の格納用に用いられている．ローカルおよびグローバルスタックはローカル変数およびグローバル変数の動的な割当用に用いられている．トレイルスタックはバックトラックが発生したときの変数に対する undo 操作のための情報を格納している．

ソースプログラムを内部コードに変換した目的コードはヒープ領域に格納され，マイクロプログラムで記述されたインタプリタによりフェッチされ実行されている．実行環境は4つのスタック上に作られることになる．

このようにプログラムの実行はインタプリタがゴールと対応するプロシージャ内の節の引数を比較してユニフィケーションを行いながら実行を進めていくことになるが，コンパイルを行うとコンパイラはプロシージャ内の各節を命令のシーケンスに変換してしまい，各節の頭部はゴール側の引数と自分自身の引数がユニファイできるか否かを調べる命令列となる．

このようなコンパイル技法を用いることにより，大幅な実行速度の向上が得られ，汎用マシン上での機械語によるインプリメントの場合，インタプリタ方式に比して10倍以上の速度向上が得られている．しかしながら後述する PSI マシンや PEK マシンのような専用マシンの場合にはインタプリタがマイクロプログラムで記述されていること，インタプリタの高速化を促進するハードウェア機構が付加されていることなどにより，インタプリタ自身が高速化されているため筆者の見解ではコンパイラ方式を導入しても速度向上は2～3倍程度と考えられる．

4. システムの実例

前章までで Prolog プログラム実行機構とその高速化のための技法について概説した．

現在開発中または開発が計画されている3つのシーケンシャル実行形 Prolog マシン

- ICOT の PSI マシン
- 神戸大学の PEK マシン
- Warren, Tick 提案の Prolog マシン

を例にして実現方式について解説する．

これらのマシンの特徴を述べると，ICOT の PSI マシンは RAS 等も強化された実用機という点に力点が置かれたマシンであり，神戸大学の PEK マシンは

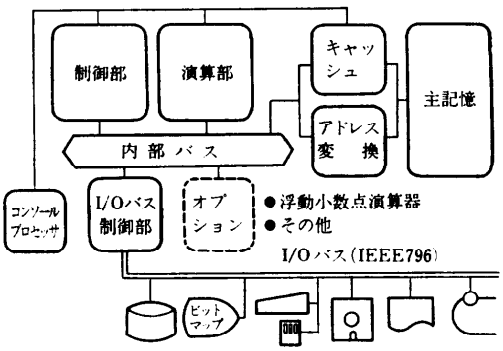


図-3 PSI のシステム構成

システムの規模を抑えつつハードウェア化を進めたマシンである．また Warren と Tick の提案したマシンは逐次実行型のマシンの性能の上限を見極めるためのマシンで，実際に実現する場合には，前二者よりハードウェア規模の大きなものになると予測される．

4.1 ICOT の PSI マシン

PSI は ICOT が開発した逐次型パーソナル推論マシンで，第五世代コンピュータシステムの研究開発プロジェクトにおいて必要となるソフトウェア開発用ツールとなることと，論理型言語向きマシンアーキテクチャの研究開発の第1ステップとなることを目指している．その性能としては実行速度 20～30 KLIPS (Logical Inference Per Second)*，最大実装記憶容量 16 M 語のパーソナル推論マシンで図-3のような構成をとっておりローカルエリアネットワークに接続することが可能となっている．マシンのハードウェアと中核となるソフトウェアは完成しており，現在ソフトウェアの拡充が計られている．端末としてはビット・マップディスプレイとマウスが付加されマルチウインドウシステムがサポートされる．ソフトウェアとしては Prolog にオブジェクト指向の機能を付加した ESP と呼ぶプログラム言語と，SIMPOS と呼ぶプログラミングシステムとオペレーティングシステムを統合したソフトウェアシステムがサポートされる．

4.1.1 PSI のハードウェアシステム

PSI は論理型言語の高速実行を行うアーキテクチャの研究と実行機構の評価にも活用できるため次の目標を設定している．

(1) ユニフィケーションの高速化のためのハードウェアを設ける

* 論理型マシンの性能を示す指標で，毎秒実行される推論計算（ユニフィケーションの成功）回数を示す．

（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

（3） 評価用データの計測・収集機構を持つ

本マシンは論理型プログラム言語 ESP 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* を効率よく実行できるように設計されている.

4.1.2 マシンアーキテクチャの概要

PSI の1語は40ビットで8ビットのタグ部と32ビットのデータ部から構成されている. タグ部の2ビットはガーベージコレクション用, 6ビットがデータタイプを表わすデータ・タグである.

KL0 をハードウェアとソフトウェアとのインタフェースに設定しており, ソース・プログラムはほぼ1対1に対応する内部コードに変換されマイクロプログラムで記述されたインタプリタにより解釈実行される.

ユニフィケーションやバックトラッキングはファームウェアのレベルで実行される.

PSI においては最大16M 語のメモリが実装できるが, メモリはアドレス変換機構により1K語のページ単位に分割されて管理されている. また実効的なメモリアクセス時間の短縮を行うため, キャッシュメモリを採用している. キャッシュへのアクセスは1マイクロ命令サイクル (200 ns) で行える. PSI では KL0 のプログラム実行用に独立に伸縮する4本のスタック (ローカル, グローバル, トレール, コントロール) と別に命令コードを置くヒープ領域が必要となる. 4本のスタック, ヒープ領域はそれぞれ独立にアドレス空間を割当て管理するようにしている.

マシンはマイクロプログラム制御により計算を進めていくが, マイクロ命令のフェッチサイクルと実行サイクルをパイプライン化して高速化しており, マイクロ命令のビット幅は64と広くとられている. またハードウェア資源の並列操作につとめている.

演算部の概念図を図-4に示す. 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であるアドレスレジスタ (AR), データレジスタ (DR), それらを相互に接続する内部バスを中心に構成されている. Prolog マシンとしての特徴は WF が1K 語と強化されていること, AR, DR が2組ずつあること, タグ操作機能があること, バレルシフタ, フィールド抽出回路など内部バス上のデータの特定フィールドの抽出回路を備えていること, ローカルスタックの先頭をキャッシングするためのバッファを設けていることなどがある.

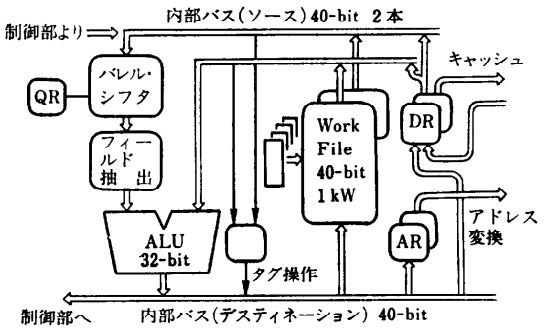


図-4 演算部の構成

4.2 神戸大学の PEK マシン

PEK は Prolog プログラムを高速に実行する計算機アーキテクチャの研究を目的として開発されている実験機である[2),6)].

マシンはマイクロプログラム制御型の逐次型マシンで, シーケンサおよび ALU はビットスライス LSI (Am 2903 A, Am 2909 A) を使用しておりインタプリタのマイクロプログラム化を行っている.

Prolog プログラムの高速実行を行うためにハードウェア化を行っているが, Prolog プログラムの高速実行に有効と考えられるハードウェア・サポートとしては多重環境の実現, ユニフィケーション実行, バックトラッキング実行を支援するハードウェアがある.

PEK はモトローラ社製の MC 68000(CP/M 68 K) をホストプロセッサにしてバックエンドプロセッサの形で, 3つのレジスタと共有メモリを介して接続されている. 図-5 に PEK のハードウェア構成を示す. 内部バスは34ビット幅で3本のバスから成り立っている. これらのバス上のデータは14ビットのフレーム部と20ビットの項部からなり, 項部は4ビットのタグ部と16ビットの値部に分けられており (モレキュールを構成している), データ転送はモレキュール単位で行われる.

マイクロ命令のサイクル時間は可変であるが平均170 nsec 程度である. 1語は96ビット長で, 16K 語の制御メモリを持っている. パイプラインレジスタを設けることにより1命令の先読みを行っている.

PEK の演算部の各ハードウェアについて概説する.

- ALU:4ビットスライスの Am 2903 A を9個

* PSI の機械語に当る言語で, Prolog のサブセットにいくつかの拡張機能を付加した論理型プログラム言語である.

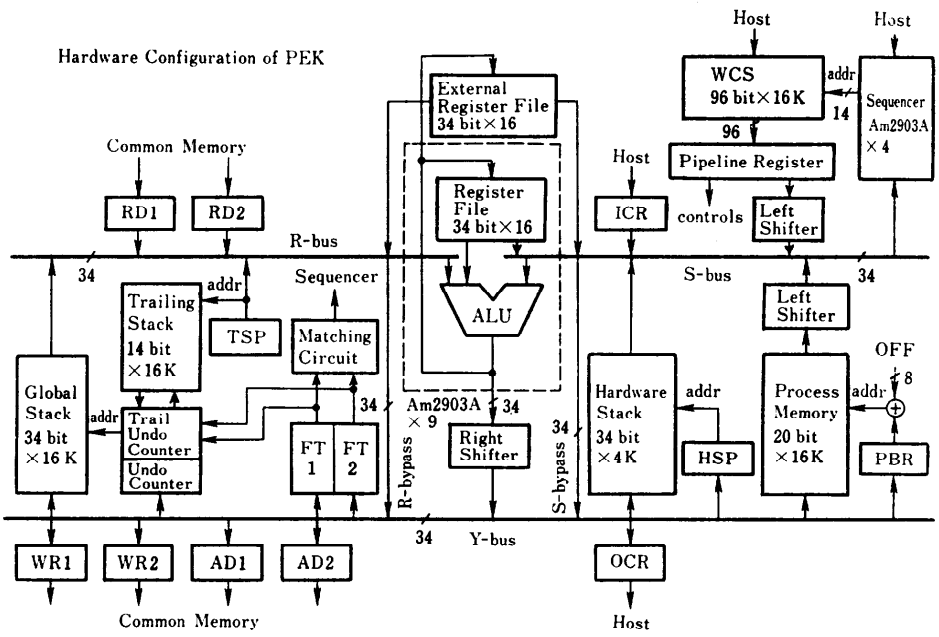


図-5 PEK のハードウェア構成

用いている.

- バイパス：PEK 内部の高速データ転送用バス.
- シフタ：内部データの特定フィールドの高速抽出用回路.
- プロセスメモリ：プロセスの管理情報格納用の高速バッファメモリ.
- ハードウェアスタック：34 ビット幅のスタックでユニフィケーション時に局所的に用いる.
- マッチング回路：モレキュール格納用の 2 つのレジスタ FT 1 と FT 2 を持ち，モレキュールのタグ部，項部の比較結果によりマイクロプログラム上での 16 通りの多重路分岐を実現する回路.
- グローバルスタック：グローバル変数の格納用の高速スタックで，ローカル変数も格納する.
- トレイルスタック：代入を行ったグローバル変数のセルアドレスを保存しておくためのスタックでトレイル作業の時用いる.
- アンドウ回路：グローバル変数の値を未定義値に戻す回路で，アンドウ動作の回数を undo カウンタに書き込むと，自動的に動作を開始し，トレイルスタックからポップされた値をグローバルスタックのアドレスとして与え，グローバルスタック中の該当要素のタグ部に未定義のフラグを書き込む回路.

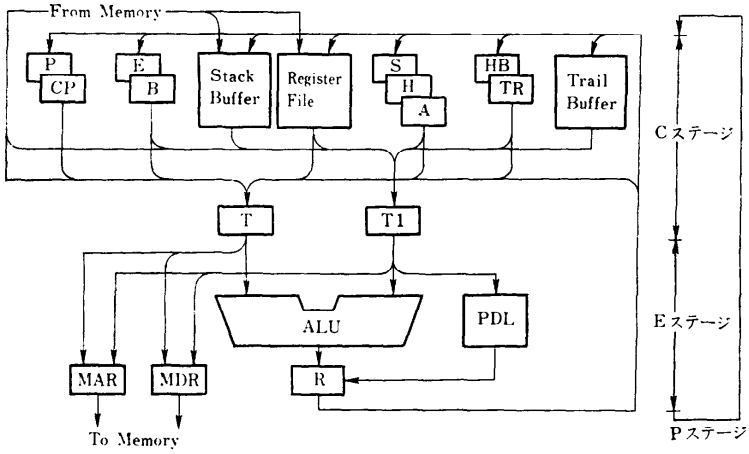
その他共有メモリへの高速アクセスを可能とするデータレジスタ，アドレスレジスタが付加されている.

Prolog プログラムはホストコンピュータ上で移動するパーザの働きにより，内部表現の形式に変換され，共有メモリに格納される．プログラムはマイクロプログラム化されたインタプリタの働きにより解釈実行されるようになっている．簡単なベンチマーク問題の実行結果ではほぼ 40 KLIPS の性能を示している.

4.3 Warren と Tick 提案の Prolog マシン

現在まだ構想段階のマシンである．考えているマシン規模は ICOT の PSI マシン，Symbolics 社の 3600 LISP マシンと同程度のものと述べられているが筆者等の判断では前述したように若干規模の大きなマシンになると考えられる[8]．マシンは逐次実行型でパイプライン制御を取り入れており，Prolog プログラムを高レベルスタックオリエンテッドな機械語命令列からなる目的プログラムにコンパイルするコンパイラの存在を前提としている.

命令ユニットによってフェッチされた機械語命令はマイクロ命令のシーケンスに拡張され，パイプライン制御形の実行ユニットにより実行される．この場合に生じる機械語命令からマイクロ命令への変換にともなうオーバヘッドは，マイクロ命令実行をパイプライン制御形実行ユニット上でのオーバラップ処理により吸



Cステージ：T, T1 へのデータの上部レジスタ，バッファ群からの読込み
Eステージ：ALU 演算 PDL, MAR, MDR, R へのデータ転送
Pステージ：Rの出力の転送
図-6 Warren の Prolog マシンの実行ユニット

収している．メモリアクセスはオーバラップ処理されメモリはインタリーブされている．本マシンはパイプライン制御方式を導入することによりシーケンシャル実行形の Prolog マシンをどこまで高速化できるかを追求している．

このマシンのアーキテクチャは Warren が SRI で研究していた改良版の Prolog 機械命令セットに基づいている[9]．本マシンでは DEC-10 prolog と異なり構造体共有方式を採用せずに構造体コピー方式を採用している．

プログラム中の項は値部とタグ部で表わされており，値部は32ビット程度，タグ部は8ビットで項の型を示している．型としては変数，構造体，リスト，定数が主なものである．

主メモリにコード領域，3つのスタック（ローカル，ヒープ（グローバルスタック），トレイル）がとられている．

本マシンのメモリシステム，命令ユニット，実行ユニットについて説明する．

メモリシステムはインタリーブされたメモリを考えており，メモリアクセスのオーバラップ処理により，メモリアクセスによるボトルネックが生じないようにしている．

実行ユニットは3つのステージのパイプラインから構成されている（**図-6**）．

- Cステージ：スタックバッファ，レジスタ，トレイルバッファ等のアクセスを行い，出力を一時レジスタ T, T1 へラッチする．
- Eステージ：ALU の実行，結果の R レジスタ，プッシュダウンリスト（PDL），メモリレジスタへの格納．
- Pステージ：Cステージのアレイユニットへの転送．

から成り立っている．

マイクロ命令のシーケンスはマイクロ命令用の2ポート読出し専用メモリに格納されており，次のアドレスの命令とマイクロ命令中の分岐先アドレスの両方に対してアクセスを同時に行えるようになっている．命令ユニットは各機械語命令（コンパイル結果の命令）ごとにその機械命令に対応しているマイクロ命令のシーケンスの先頭アドレスを出力し，実行ユニットにこれらの命令が供給される．タグによってマイクロ命令レベルでの条件分岐が行えるようになっている．

実行ユニットのデータパス上には汎用レジスタ，スタックバッファ，トレイルバッファ，PDL が設けられており，スタックバッファ，トレイルバッファは高速アクセスを可能とするスタックキャッシュの働きをしている．汎用レジスタは一時変数やプロシージャの引数を格納するのに用いられる．

命令ユニットの主な機能は実行ユニットに命令を供給することである．Prolog の機械命令には条件分岐命令はなく，プロシージャ内の複数の節の内の一つと

表-2 各種 Prolog システムの性能評価

マ シ ン	速度 (LIPS)
DEC 2060 (DEC-10 prolog)	43,000
IBM 3033	27,000
VAX 780 (C-Prolog)	1,500
PDP 11/70	1,000
Z-80 (Micro Prolog)	120
Apple-II	8
神戸大学 PEK マシン	40,000
ICOT PSI マシン	30,000
富士通 ALPHA	12,000
電電公社通研 ELIS	11,000

マッチすることのできるプロシージャコール命令だけなのでいかに高速に節のアドレスを計算するかに主な設計の着眼点がある．この目的のためインデッキシングと呼ばれるタグを利用した高速の節アクセス命令が導入されている．

5. ま と め

前章までで，Prolog プログラムの実行制御機構，高速化のための工夫，Prolog マシンの具体例について説明をしてきた．ICOT の PSI マシン，神戸大学の PEK マシンはいずれもハードウェアの開発が終っておりソフトウェアに関しても基本的な部分は完成している．

現在までに得られた性能は PSI マシン，PEK マシンとも 20〜40 KLIPS の性能をインタプリタの段階で示しており，DEC 社2060上の Prolog コンパイラの出力コードと同程度の性能を示している．

コストパフォーマンスの観点から見ると10倍以上の性能を示すことになり，十分に実用化に堪えうると考えられる．現時点で性能が実測されているマシンの性能評価のデータを表-2に示す．本データは Warren が International Workshop on High-Level Computer Architecture 84 で行った講演資料に国内で発表された文献から得られたデータを加えたものである．本データには LISP 専用マシンである ALPHA システムや ELIS システムのデータも合わせて示した．LISP マシンを用いてもかなりの高速化が可能なことが判り大変興味深い．

また Warren と Tick のマシンでは簡単なプログラムを手でコンパイルを行い，パイプライン制御方式で実行される場合の動的な動きをトレースした結果，良好な条件ではマシンサイクルを 100 ns とすると 450 KLIPS の処理能力が得られるという結果を得ている．

今後の課題としては，

1. 高機能ワークステーションとして，ハードウェア，ソフトウェアともより一層充実させること．

ソフトウェアシステムとしてはオブジェクト指向形のシステムが一つの方向と考えられる．

2. データベースやデータベースマシンとの結合を行い知識ベースマシン化すること．

3. LAN を用いたネットワークシステムの機能をサポートすること．

4. より幅の広い並列処理機能を取り入れた超高速推論マシンへの発展を試みること．

などを上げることができる．

参 考 文 献

1) 安井：LISP マシン，情報処理，Vol. 23, No. 8, pp. 757-772 (1982).
2) Kaneda, Y., Tamura, N., Wada, K. and Matsuda, H.: Sequential Prolog Machine PEK Architecture and Software System, Proc. of the International Workshop on High-Level Computer Architecture 84.
3) Tick, E.: Towards a Multiple Pipelined Prolog Processor, Proc. of the International Workshop on High-Level Computer Architecture 84.
4) Tick, E. and Warren, D. H.: Towards a Pipelined Prolog Processor, 1984 International Symposium on Logic Programming, IEEE Computer Society.
5) 瀧，横田，山本，西川，内田：パーソナル逐次型推論マシン PSI のハードウェア設計，Proc. of the Logic Programming Conf. '84.
6) 田村，和田，松田，金田，前川：PROLOG マシン PEK について，Proc. of the Logic Programming Conf. '84.
7) 溝口，片山，武田：論理型言語 Prolog の比較検討，Proc. of the Logic Programming Conf. '84.
8) Warren, D. H. D.: Implementing Prolog-Compiling Predicate Logic Programs, D. A. I. Research Report, No. 39.
9) Warren, D. H. D.: An Abstract Prolog Instruction Set, SRI International Technical Note, 309 (Oct. 1983).

(昭和59年8月20日受付)