パーソナル逐次型推論マシン PSI の評価†

——実行速度とハードウェア各部の性能について——

中島 浩†† 瀧 和男†††
中島克人††† 三石彰純††††

個人利用型 Prolog 専用マシン PSI の評価を行った．実行速度に関する評価と，Prolog の高速実行向きに導入したキャッシュメモリ，レジスタファイル，分岐ハードウェアの効果について，PSI を改良，再設計する立場から測定し評価を加えた．処理速度は，DEC-2060 上の DEC-10 Prolog（コンパイラ版）と同等であることが確認されたが，コンパイラによる最適化処理の容易なプログラムでは DEC が速く，複雑なユニフィケーションやバックトラックなどの動的処理の多いプログラムでは PSI の方が速い傾向を持つ．キャッシュメモリについては，ヒット率がオペレーティングシステムを除く応用プログラムで96%と高く，Prolog に対してキャッシュメモリが十分有効であること，ストアイン（ライトバック）方式が効率的なこと，容量を 8k 語から削減できることが判明した．またメモリアクセスの発生頻度や read と write の出現比率なども明らかとなった．レジスタファイルについては，マイクロ命令による直接アドレス指定の利用頻度が高いこと，ベース相対指定などの方式は改良の余地があること，容量を 1k 語から削減可能なことが明らかとなった．分岐機能については，タグ値にもとづく多方向分岐，条件分岐の高い出現比率が確率され，Prolog の高速実行向きハードウェアの最適化に関する指針が得られた．

1. まえがき

われわれは，Prolog を拡張した述語論理型言語の高速実行とメモリの大容量化や対話型プログラム開発環境の強化等の実用マシンとしての機能・性能の向上を目的に，日本の第5世代コンピュータプロジェクトの一環としてパーソナル逐次型推論マシン PSI を開発してきた．その設計思想，マシンアーキテクチャ，ハードウェア構成についてはすでに報告している[4)~7)]．本論文では，PSI の実行速度に関する評価と，Prolog プログラムの高速実行向きに導入したキャッシュメモリ，レジスタファイル，分岐ハードウェアの効果について，主に PSI を改良，再設計するための指針を得るという立場から測定を行い評価を加える．また動特性の測定結果は，Prolog 応用プログラムのメモリアクセス特性に関する興味深いデータを含んでおり，共に報告する．

† Evaluation of the Personal Sequential Inference Machine PSI—Execution Speed and Performance Analysis of the Hardware Components— by HIROSHI NAKASHIMA (Computer Systems Development Department, Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corporation), KAZUO TAKI, KATSUTO NAKAJIMA (Fourth Research Laboratory, Research Center, Institute for New Generation Computer Technology) and AKITOSHI MITSUISHI (Knowledge Processing Development Department, Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corporation).

†† 三菱電機(株)情報電子研究所計算機システム開発部
††† (財)新世代コンピュータ技術開発機構研究所第4研究室
†††† 三菱電機(株)情報電子研究所知識処理開発部

2. PSI のアーキテクチャとハードウェア上の特徴

2.1 命令実行方式とマシン・アーキテクチャ

PSI は Prolog を拡張した述語論理型言語 KL0 を直接実行する専用マシンである．

PSI の命令実行方式は，KL0 のマシン内部表現（命令コード）をマイクロプログラム化されたインタプリタで解釈実行する方式をとっている[7),8)]．マシンの内部表現ではプログラム中のアトム，述語名，変数などは，多くの場合それぞれ対応するタグを持った1語で表現される．なお PSI の1語は8ビットのタグ部と32ビットのデータ部から成っている．また述語の引数が短いビット幅で表現できるときには，1語中に4個までの8ビットの引数をパックした表現をとり，メモリの節約を図っている．PSI のアーキテクチャは，マイクロプログラムによるインタプリティブな実行の高速化に重点を置いた設計となっており，次節に示すようなハードウェアサポートを持つ．

マイクロプログラム化されたインタプリタによる Prolog の実行方式は，拡張された制御機能[9)]に関するものを除くと DEC-10 Prolog[10)] とほぼ同じ方式によっているが，スタックはローカル，グローバル，コントロール，トレールの4本を用い，クローズ内 OR の効率化のためにローカルスタックとコントロールスタックを分離している．ローカルスタックにはローカル

変数領域が，グローバルスタックには構造体中に現れる変数の領域がとられ，トレールスタックにはバックトラック時に変数を未束縛状態に戻すためのアドレス情報が置かれる．またコントロールスタックには，プログラムの実行制御にかかわる10語の情報が，述語呼出しに関する呼出し元コントロールフレーム，またはバックトラック用のコントロールフレームとして置かれる．これらの実行制御情報の最新のものは，ワークファイル（WF）と呼ぶレジスタファイル上に置かれ，必要に応じてコントロールスタック上に掃き出される．

4本のスタックは各々独立の論理アドレス空間に配置され，命令コードとヒープベクタ（書替えを許す構造体データ）は，ヒープ領域と呼ぶこれも独立の論理空間に置かれる．PSIではユーザプログラムと割込み処理プログラムなど，複数プログラムの並行実行をサポートしており，ヒープ領域は全プログラムで共通に使用し，スタック用の領域はプログラムごとに独立の論理空間を割り当てている．これらの論理空間のことをエリアと呼んでいる．各エリアに対して，物理メモリを必要に応じて割り当てるためのアドレス変換テーブルが，ハードウェアでサポートされている[4),5)]．

ハードウェアは制御部，演算部，メモリ部，入出力部から成り，演算部は，2本のソース・データバスと1本のデスティネーション・データバスを含む．これらのバスに，ALU，レジスタファイル，メモリインタフェース・レジスタ等が接続されており，マイクロプログラムにより制御される．マイクロ命令の実行と次マイクロ命令の読出しは並列に行われる．メモリ部には，キャッシュメモリ，アドレス変換回路，主記憶が含まれる[4)~6)]．

2.2 高速化のためのハードウェアサポート

PSIではKL0を高速に実行するためのハードウェアサポートとして，

（a）スタックおよびヒープ領域アクセスの高速化を目的としたキャッシュメモリ

（b）インタプリタの最適化のための大容量で高機能のレジスタファイル

（c）タグ値にもとづく多方向分岐機能と多様な条件分岐機能

（d）マイクロプログラムレベルの並列制御機能（水平型に近いマイクロプログラム制御：64 bit/命令）

を準備した[4),5)]．

この中でキャッシュメモリの仕様は，容量が8k語で2セットのセットアソシアティブ方式，書込方式はストアイン方式（ライトバック方式），アクセス時間はヒット時200 nsec，ミスヒット時800 nsecである．ブロックサイズは4語であり，主記憶との間の転送はブロック単位に800 nsかけて行われる．スタックへの連続書込時のキャッシュミスヒットに対して不要な読込みを行わないwrite_stackコマンドを有する．

ワークファイルと呼ぶレジスタファイルは1k語の容量を持ち，読出し，演算器による演算，結果の書込みを1マイクロ命令サイクルで実行可能である．先頭の16語はdual-port化されており，先頭部分と最後尾定数格納領域の各64語はマイクロ命令から直接アドレス指定ができる．間接指定にはWFAR 1,2という2本のアドレスレジスタが利用でき，自動増加，減少の機能を持つ．ベース相対指定には，ベースレジスタに対してデータレジスタ（PDRまたはCDR）の下位5ビットを結合するものと，WFCBRと呼ぶベースレジスタにマイクロ命令で指定する5ビットのアドレスを結合しアドレスを生成するものがある．

インタプリタによる実行時の最適化方式の1つにテイルリカージョンの最適化[11)]があるが，PSIではレジスタファイル上にフレームバッファと呼ぶローカルスタックのバッファ領域をとり，スタックへの値の掃出しが不要な場合にこの領域をくり返し利用することで実現している．この領域のアクセスにはWFAR 1による間接指定と，PDRまたはCDRによるベース相対指定を用いている．

分岐機能としては，条件分岐の種類を数多く用意したこと，タグ値にもとづく多方向分岐機能，演算やメモリアクセスを行いながら分岐のできる機能等の特徴を有す．

2.3 目標性能

PSIのCPUは市販の高速ショットキーTTL ICを主に使用し，200 nsecのマイクロ命令サイクルタイムを実現している．目標性能としては，DEC 2060上でのDEC-10 Prolog（コンパイラ版）と同等の30k LIPS (Logical Inference Per Second：1秒当りの推論回数）を設定している．

3. 実行時間の測定と性能評価

3.1 プログラムの実行時間

評価用プログラムを用いてPSIの実行時間とDEC-2060上のDEC-10 Prologの実行時間を測定し，両

者の比とともに示したのが表1である．なお DEC-10 Prolog はモード宣言と fast code 指定をしてコンパイルしたものを述語 statistics を用いて時間計測し数回の平均をとっている．PSI については CPU 内蔵の精度 1 msec のタイマを用いた．

評価用プログラムは，表中の(1)から(10)が第1回 Prolog コンテスト[12]の課題の一部であり，いずれもリスト操作を多く含む小規模のプログラムである．それに対し(11)から(19)はより規模の大きい実際的なプログラムである．BUP と LCP が自然言語処理用の各々方式の異なるパーサであり，harmonizer は音楽知識によりメロディーにハーモニーを付加する一種の専門家システムである．ソースプログラムの行数はそれぞれ，約300，約1500，約700行である．プログラムの特徴は，いずれも構造体データのユニフィケーションやバックトラックを使用したものであるが，中でも要素数8以上の比較的大きな構造体やネストした構造体データを扱うのが BUP であり，バックトラックを頻繁に行うのが harmonizer である．(11)以下のプログラムは，次節のハードウェア評価でも使用している．

表から PSI の性能は，当初目標の DEC-10 Prolog と同等を達成していることが分かる．プログラムにより DEC と PSI の優位性が逆転するのが注目されるが，その傾向としては(1)の nreverse のようにクローズインデキシングにより非決定性を除くことができ，さらにユニフィケーションに関する最適化が容易なプログラム，すなわちコンパイラによる最適化処理の容易なプログラムでは DEC-10 Prolog が高速になっている．それに対し，(11)から(16)のように構造体データどうしのユニフィケーションを多く含み頻繁なバックトラックを含むプログラム，すなわち動的処理の多いプログラムでは PSI が速くなっている．その原因としては，PSI の処理系が DEC に比べより多くの実行管理情報をスタックに積み単純なプログラムでは基本処理のオーバヘッドが目立つためと考えられるが，逆に構造体どうしのユニフィケーションやバックトラックが出現するとすべてマイクロプログラムで処理する PSI が相対的に高速になるものと考えられる．

全体的に見て PSI は動的処理を多く必要とする応用プログラムに強い傾向を持つといえる．ただ

表1 評価用プログラムの実行時間
Table 1 Execution time of benchmark programs on PSI and DEC-2060.

programs	PSI(msec)	DEC(msec)	DEC/PSI
(1) nreverse (30)	13.6	9.48	0.70
(2) quick sort (50)	15.2	14.6	0.96
(3) tree traversing	51.7	61.1	1.18
(4) lisp (tarai 3)	4024	4360	1.08
(5) lisp (fib 10)	369	402	1.09
(6) lisp (nreverse)	173	194	1.12
(7) 8 queens (1)	96.9	97.5	1.01
(8) 8 queens (all)	1570	1580	1.01
(9) reverse function	38.2	41.7	1.09
(10) slow reverse (6)	99.4	89.0	0.90
(11) BUP-1	43	52	1.21
(12) BUP-2	139	194	1.40
(13) BUP-3	309	424	1.37
(14) harmonizer-1	657	1040	1.58
(15) harmonizer-2	1879	2670	1.42
(16) harmonizer-3	24119	31390	1.30
(17) LCP-1	379	295	0.78
(18) LCP-2	1387	1071	0.77
(19) LCP-3	2130	1656	0.78

LCP だけは構造体データを扱うにもかかわらず DEC が速く，これは LCP の作者が DEC-10 Prolog 処理系の製作者の1人である F. Pereira 氏で，処理系の長所欠点（例えば構造体の要素数がある大きさを越えると急に性能が落ちる）をわきまえたコーディングをしているためと考えられるが，さらに詳しい調査解析を要する．

3.2 インタプリタの動特性

実用規模の応用プログラムにおける PSI の総合性能を評価するため，評価用プログラムを実行したときのインタプリタ各機能の動特性を測定した．表2は各機能モジュールの実行ステップ数比率を示したものである．評価用プログラムとしては PSI のオペレーティングシステムの一部をなす window システムと 8 puzzle というゲームプログラムを追加した．window は構造体のユニフィケーションやバックトラックが少

表2 ファームウェアインタプリタ各部の実行ステップ比率 (%)
Table 2 Execution step ratios of each component module of the firmware interpreter (%).

program	control	unify	trail	get_arg	cut	built
(1) window	31.1	17.1	2.0	13.6	10.0	26.2
(2) 8 puzzle	27.5	11.0	7.5	22.7	0	31.3
(3) BUP	22.3	43.0	4.7	5.2	5.6	19.2
(4) harmonizer	25.5	46.4	5.4	7.3	4.0	11.0

なく，いわば Prolog 的機能をあまり使わないプログラムである．8 puzzle は探索問題であり，バックトラックの頻度が高い．

述語の呼出し回数比率は，表中にはないが組込述語が window で 82%，より Prolog 的なプログラムといえる BUP で 65% となりユーザ述語よりかなり多い．しかしながら組込述語 (built) の実行ステップ数の比率を表より見ると，各々 26.2%，19.2% となり回数の比率に比べてずっと小さいことから，むしろ実行制御 (control＝ユーザ述語の呼出復帰処理) に多くの時間を要していることが分かる．これはすでに述べたように KL0 では実行管理情報が多く，処理にステップ数を要することの影響と考えられる．また組込述語のための引数読出し (get_arg) にも時間がかかっていることが読み取られる．これらは，処理系改良時の対象として考慮すべきものであろう．一方 unify を見ると BUP と harmonizer で著しく比率が高く，これは構造体のユニフィケーションの多さを裏付けるデータである．ユニフィケーション単体の性能については，PSI は DEC に比べて同等以上の性能値が得られており[12]，unify 部分のステップ数短縮には，コンパイラでユニフィケーション部分を最適化できるような命令体系を工夫することが必要と思われる．

4. 動特性の測定とハードウェア評価

4.1 動特性の測定

本節では主にハードウェアの改良の要否を判定する観点から，評価用プログラム実行時のハードウェアの動特性を測定し評価を加える．評価対象は，キャッシュメモリ，ワークファイル，分岐機能である．

評価用プログラムには前節と同じものを使用し，またデータ収集と評価のためにいくつかのツールを作成した．データ収集用には PSI のコンソールプロセッサ上に COLLECT と呼ぶ簡単なインタプリタシステムを実装し，CPU の起動制御とデータ収集のためのコマンド列を実行させた．具体的にはステップ実行やブレークポイントまでの実行をくり返し行わせ，CPU が停止するたびにマイクロ命令アドレス，レジスタやメモリ内容をフレキシブルディスクに貯めこむようにした．ワークファイルと分岐回路の評価には，MAP と呼ぶマイクロ命令パターンの解析ツールを用意し，COLLECT によるマイクロ命令アドレスのトレース結果を利用して，特定のマイクロ命令フィールドの特定パターンの出現度数をカウントした．キャッシュメモリの動特性解析には，PMMS と呼ぶキャッシュメモリシミュレータを用意し，同じく COLLECT により収集したキャッシュコマンドパターンとその時のメモリアドレスからキャッシュのヒット率などを求めた．

4.2 キャッシュメモリの評価

スタックおよびヒープ領域アクセスの高速化の効果を評価するため，キャッシュメモリのアクセス頻度とキャッシュヒット率に関するデータを収集した．

表 3 にキャッシュコマンド別の出現頻度を示す．全マイクロ命令ステップ中 16% から 23.1% はキャッシュコマンドを含んでおり，マイクロ命令 5 ステップに対し 1 ステップ程度の割合でメモリ参照要求のあることが分かる．read と write 系のコマンドの比率は約 3 対 1 で read コマンドが多くなっている．また write 系コマンド中の 5 割から 7 割半が write-stack コマンドであり，スタック用に導入したコマンドがよく利用されている．

表 4 はエリア別のアクセス頻度である．はじめに，heap のアクセス頻度と 4 本のスタックのアクセス頻

表 3 キャッシュコマンドの出現頻度 (%)
Table 3 Execution rate of each cache command in the total microprogram execution steps (%).

programs	read	write-stack	write	write total	total
(1) window-1	15.2	3.5	1.2	4.7	19.9
(2) window-2	15.2	3.0	1.1	4.1	19.7
(3) window-3	17.6	3.9	1.4	5.3	22.8
(4) 8 puzzle	9.9	3.2	2.8	6.1	16.0
(5) BUP	15.6	3.5	2.2	5.7	21.3
(6) harmonizer	15.3	4.6	2.2	6.8	22.1
(7) LCP	17.0	3.9	2.2	6.1	23.1

表 4 エリア別アクセス頻度 (%)
Table 4 Access frequency of each memory area (segment) (%).

programs	heap	global stack	local stack	control stack	trail stack
(1) window-1	49.6	4.6	16.5	26.7	2.6
(2) window-2	56.6	4.4	12.7	26.3	0.1
(3) window-3	52.7	6.2	12.1	28.2	0.8
(4) 8 puzzle	31.3	14.3	33.9	14.1	6.4
(5) BUP	39.0	29.9	17.3	12.0	1.8
(6) harmonizer	35.2	17.7	30.3	12.8	3.8
(7) LCP	44.7	22.3	14.1	17.4	1.4

表 5 エリア別キャッシュヒット率 (%)
Table 5 Cache hit ratios of each memory area (segment) (%).

programs	heap	global stack	local stack	control stack	trail stack	total
(1) window-1	94.1	92.8	98.9	99.4	99.6	96.4
(2) window-2	87.2	90.0	98.5	99.3	95.2	91.9
(3) window-3	84.5	92.8	97.4	98.6	98.7	90.7
(4) 8 puzzle	99.2	99.4	99.6	99.2	97.7	99.3
(5) BUP	98.2	96.8	99.0	98.2	99.7	98.0
(6) harmonizer	97.5	98.4	99.4	98.2	97.9	98.4
(7) LCP	96.6	93.8	99.2	99.1	98.6	96.2

度に分けて見ると，ヒープ領域へのアクセスが全体の3割から，5割半を占めるのが分かる．ヒープ領域へのアクセスは基本的には命令コードのフェッチであるが window のプログラムでは他にヒープベクタ型のデータが使用されており，その分アクセス頻度が上がっている．またプログラムにより命令のアクセス頻度が異なるのが分かる．

グローバル，ローカル，コントロールの各スタックへのアクセス頻度もプログラムに依存している．構造体データが多いとグローバルスタックが，引数中の(構造体に含まれない) 変数が多いとローカルスタックが，引数個数に比べ述語呼出しが多いとコントロールスタックがそれぞれ頻繁にアクセスされている．トレールスタックへのアクセスは最大でも6.4%と低くなっている．

表5はエリア別のキャッシュヒット率である．平均のヒット率は，window で90%程度のものがあるが，他はいずれも 96% 以上で高い値を示している．特に実用規模の応用プログラムでかつバックトラックやユニフィケーションを十分に利用している，(5)から(7)のプログラムで高いヒット率を得ていることから，Prolog系の言語でもメモリアクセスのローカリティは高く，PSI で採用した規模のキャッシュメモリが十分に有効であるということができる．なお window -2, -3 のヒット率が低いのは，実行中にプロセススイッチが起こっていることと，多くの ESP のクラス (PSI のシステム記述言語 ESP はオブジェクト指向機能を持っており，クラスとはオブジェクト指向にもとづくプログラムモジュールの一種) をまたがった呼出しのために命令コードのローカリティが低下しているためと考えられ，オブジェクト指向によるプログラミングがキャッシュのヒット率を低下させるか否かについては，別途検討の余地のあることを示している．次にキャッシュメモリの容量とセット数に関する評価結果を示す．これらは，window プログラムのトレースデータをキャッシュシミュレータに与え，異なる仕様のキャッシュメモリについてシミュレートした結果である．

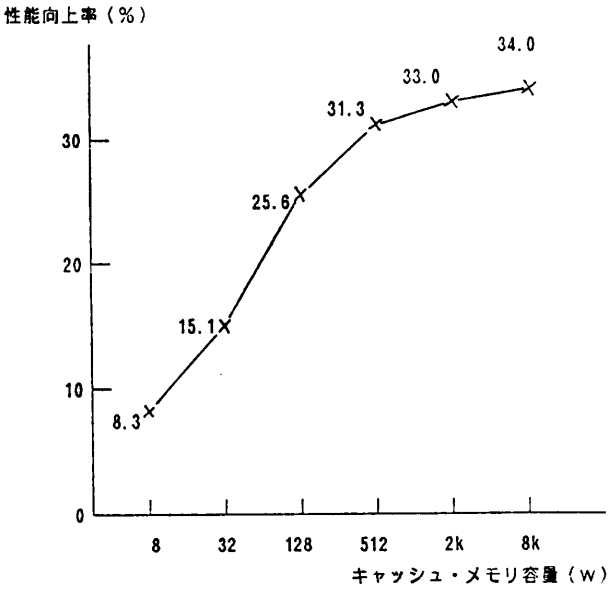

図 1 キャッシュメモリ容量と性能
Fig. 1 Performance improvement ratios against the cache memory size.

図1はキャッシュメモリ容量を 8w から 8kw まで変化させたときの性能向上率である．ただし，

性能向上率 $= \{(T_{nc}/T_c)-1\} \times 100$

T_{nc}: キャッシュメモリが無いときの実行時間

T_c: キャッシュメモリがあるときの実行時間

とした．またセット数はすべて2である．図から性能向上率が512w付近で飽和しつつあることが読み取られ，PSI のキャッシュメモリ容量 8kw は，幾分削減できる可能性のあることを示している．

一方設計の容易さとハードウェア量の面からはセット数を1にした方がより有利である．そこで 4kw 2セットの場合と 4kw 1セットの場合の性能向上率の

差を前述のキャッシュシミュレータを用いて求めてみたところ，window, 8 puzzle, BUP のいずれにおいても1セットの方が約3% 低いだけであった．これは設計・実装コストと性能の比から考えて利用可能な値ということができよう．

またストアイン方式の評価については，同様にキャッシュシミュレータによりストアスルー方式との性能向上率の差を求めた結果，ストアイン方式の方が約8% 高い値を示し，有効性を確認することができた．

4.3 ワークファイルの評価

マイクロ命令パターン解析ツール MAP により，評価用プログラムを実行したときのワークファイル (WF) 各機能の利用頻度を求めた．その中から BUP に関する測定結果を**表6**に示す．他のプログラムについても同様の傾向が見られる．

WF のアクセス頻度は表中の total の欄より，Source 1指定が 56.4%，Source 2指定が 29.1%，Destination 指定が 36.6% である．また WF を含むレジスタアクセスの命令だけを集計した場合には，WF のアクセス頻度は 50 から 80% となり，利用頻度の高さが確認された．

直接アドレス指定を行うのは，表中の(1)から(3)であるが，これらのアドレッシングモードの利用率が WF アクセス全体の 90% を越えていることが分かる．直接アドレス指定を可能とするため，レジスタアクセス用のマイクロ命令フィールドをマイクロ命令語長64ビット中の3分の1も費やしているが，有効に利用されていることが確認された．

次にベース相対指定(4)，(7)と間接指定(5)，(6)の評価を行う．(4)の PDR と CDR によるベース相対指定は利用頻度が低く，もっとも高い window でも 3.3/1.7 という値であった．PDR/CDR 以外にパックされた引数 (8 bit) を切り出してきてベース相対指定に使える機能があれば，利用率はもう少し高まると考えられる．(5)の WFAR 1 による間接指定は(4)とともに Local frame buffer のアクセスに用いられるが，9割以上が間接アクセス時の自動増加機能を利用しており，その有効性が確かめられた．(6)は Trail buffer[1),7)] のアクセスに，(7)は汎用的に利用されるがいずれも利用率が低く，PSI で行っているトレールスタックのバッファリング[1)]については再検討の余地があるといえる．

WF へのアクセスは全体の 99% 以上が，直接アドレス指定域の 128 w と Local frame buffer 2面の 64 w へのアクセスであり，このことから容量を 1 kw から大幅に削減しても性能への影響は少ないということができる．

表 6 ワークファイル各機能（アクセスモード）の利用頻度 (%)
Table 6 Dynamic frequency of the Work File (register file) access modes (%).

Access Mode	Source 1 (ALU in-1)	Source 2 (ALU in-2)	Destination (ALU out)
(1) WF00-0F	*12.2/ 6.9	*100/29.1	*33.0/12.1
(2) WF10-3F	58.5/33.0	—	63.6/23.3
(3) Constant	23.0/13.0	—	—
(4) @PDR/CDR	1.3/ 0.8	—	0.3/ 0.1
(5) @WFAR1	4.6/ 2.6	—	2.8/ 1.0
(6) @WFAR2	0.07/0.04	—	0.3/ 0.1
(7) @WFCBR	0.3/ 0.2	—	0.0/ 0.0
Total	100/56.4	100/29.1	100/36.6

* a/b……a: WF アクセス命令中の頻度
(Rate in the total Work File access counts.)
b: 全命令中の頻度
(Rate in the total microprogram execution steps.)

表 7 分岐命令の種類別出現頻度 (%)
Table 7 Dynamic frequency of the branch operations in the microprogram execution steps (%).

operation	BUP	window	8 puzzle
Type 1	73.0	73.5	72.8
(1) no operation	7.2	6.7	4.8
(2) if (cond) then	16.0	16.5	12.1
(3) if (not (cond)) then	19.2	17.0	20.3
(4) if tag (src2) then	2.7	5.2	3.1
(5) case (tag (n, P/CDR))	10.9	8.6	9.1
(6) case (irn)	2.8	4.6	4.9
(7) case (ir_opcode)	0.5	1.4	1.5
(8) goto	3.7	1.4	2.7
(9) gosub	4.0	5.7	6.5
(10) return	3.8	5.4	6.5
(11) load_jr	0.8	0.4	0.7
(12) goto @jr	1.4	0.6	0.7
Type 2	20.5	19.4	22.9
(13) no operation	9.6	7.8	7.7
(14) goto	10.9	11.7	15.2
Type 3	6.5	7.1	4.2
(15) no operation	6.5	7.0	4.2
(16) goto @jr	0.0	0.04	0.05

4.4 分岐機能の評価

マイクロ命令パターン解析ツールを用いて分岐命令の出現頻度を調べた結果を**表7**に示す．operation 欄には，マイクロ命令タイプ別の分岐命令の種類を示している．すべての分岐命令を合わせた出現頻度は，

100% から no operation の比率を差し引いて求められ，約77% から83% と非常に高い値を示している．すなわち，全マイクロ命令実行ステップ中約8割が分岐命令であったことになり，汎用マシンでは考えられない頻度といえる．分岐を行うマイクロ命令中 Destination の no operation 比率が 30% 前後であったことから，全マイクロ命令ステップ中の約3割が ALU 演算をともなわない分岐，約5割が ALU 演算を行いながらの分岐ということになる．

表中の(2)，(3)が 64 種ある条件分岐を合わせたものであり，(4)がタグ即値との比較による条件分岐である．これらの合計は，全ステップの約 35% から 39% となり，条件分岐機能の強化の有効性が確かめられた．

(5)はダグディスパッチ機能，(6)は組込述語のパックされた引数 (8 bit) に埋めこまれたダグを取り出して多方向分岐する機能であり，これらの合計の比率は約13から14% となった．すなわち，7から8ステップに1度はダグによる多方向分岐を行っていることになり，これらの機能も有効に利用されていることが示された．

(12)，(16)の JR による間接分岐[4),5)]の頻度が低いが，JR はむしろループカウンタとして使用されていることが多く，その頻度を計測すると全ステップ中の約9から 12% であったことから，専用のループカウンタを設置する方が望ましいと考えられる．

5. あとがき

PSI の処理性能と，Prolog プログラムの高速実行用に準備したハードウェアに関する評価を行った．また評価用のデータとして，プログラム実行時のメモリアクセス動特性の測定値を示した．

処理性能は，DEC-2060 上でのコンパイルコードと同等であることが確認されたが，コンパイラによる最適化処理の容易なプログラムでは DEC が速いのに対し，動的処理の大きい応用プログラムでは PSI の方が速く，これはマイクロプログラム化されたインタプリタの1つの特徴と見ることができる．

ハードウェアの評価では，キャッシュメモリのヒット率が応用プログラムで 96% と高く，Prolog に対してキャッシュメモリが十分有効であること，ストアイン方式が効率的なこと，容量を 8k 語から削減できることが判明した．メモリアクセスの動特性では，read と write コマンドの比が3対1であること，メモリアクセスの出現頻度が5マイクロ命令に1度の割であること，ヒープ領域のアクセス（主に命令コードアクセス）頻度が3割から5割強であることが示された．ワークファイルについては直接アドレス指定が9割以上利用され有効であるが，ベース相対指定などの方式には改良の余地があること，容量を 1k 語から削減可能なことが明らかとなった．また分岐機能については，分岐マイクロ命令の出現頻度が全実行ステップの約8割もあること，タグ値にもとづく多方向分岐，条件分岐とも十分に有効であることが確認され，PSI の改良に関する多くの指針が得られた．

今後はこれらの評価結果にもとづいてマシンハードウェアを再設計すること，マシン命令をコンパイラでの最適化が行いやすいように改良することなどを計画している．

謝辞 最後に本研究に当たり御指導いただいた(財)新世代コンピュータ技術開発機構研究所の内田俊一室長，ならびに研究の機会を与えてくださった同研究所の渕 一博所長に感謝の意を表する．

参考文献

1) Nakajima, K., Nakashima, H., Yokota, M., Taki, K., Uchida, S., Nishikawa, H., Yamamoto, A. and Mitsui, M.: Evaluation of PSI Microinterpreter, *Proc. of Compcon Spring 86*, pp. 173-177 (1986).
2) 西川 宏，山本 明，横田 実，中島克人，三井正樹，吉田裕之：PSI の性能評価(1)，第 30 回情報処理学会全国大会論文集，No. 1C-2 (1985).
3) 中島 浩，三石彰純，瀧 和男：PSI の性能評価(2)，第 30 回情報処理学会全国大会論文集，No. 1C-3 (1985).
4) 瀧 和男，横田 実，山本 明，西川 宏，内田俊一：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 Conference '84*, No. 8-1, Tokyo (1984).
5) Taki, K., Yokota, M., Yamamoto, A., Nishikawa, H., Uchida, S., Nakashima, H. and Mitsuishi, A.: Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI), *Proc. of the International Conference on Fifth Generation Computer Systems 1984*, pp. 398-409, Tokyo (1984).
6) Yokota, M., Yamamoto, A., Taki, K., Nishikawa, H. and Uchida, S.: The Design and Implementation of a Personal Sequential Inference Machine PSI, *New Generation Computing*, Vol. 1, No. 2, pp. 125-144, Ohmsha (1983).
7) Yokota, M., Yamamoto, A., Taki, K., Nishi-

kawa, H., Uchida, S., Nakajima, K. and Mitsui, M.: A Microprogrammed Interpreter for the Personal Sequential Inference Machine, *Proc. of the International Conference on Fifth Generation Computer Systems 1984*, pp. 410-418, Tokyo (1984).
8) 山本 明, 横田 実, 西川 宏, 瀧 和男, 内田俊一: 逐次型推論マシン ϕ のマイクロインタプリタ, 情報処理学会研究会資料, 記号処理 29-7 (1984).
9) 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, 第 26 回情報処理学会全国大会論文集, No. 4 D-11 (1983).
10) Warren, D. H. D.: Implementing Prolog—Compiling Predicate Logic Programs, Vol. 1, 2, D. A. I. Research Report No. 39, 40, Dept. of Artificial Intelligence, Univ. of Edinburgh (1977).
11) Warren, D. H. D.: An Improved Prolog Implementation Which Optimizes Tail Recursion, *Proc. of the Logic Programming Workshop*, Hungary (July 1980).
12) Okuno, H.: The Report of The Third Lisp Contest and The First Prolog Contest, 情報処理学会研究会資料, 記号処理 33-4 (1985).

(昭和 61 年 12 月 10 日受付)
(昭和 62 年 10 月 14 日採録)

中島 浩 (正会員)

昭和 31 年 5 月生. 昭和 54 年京都大学工学部情報工学科卒業. 昭和 56 年同大学院修士課程修了. 同年三菱電機(株)入社. 同社情報電子研究所において, 推論マシンのアーキテクチャの研究開発に従事. 論理型言語の実行方式および並列プロセッサのアーキテクチャに興味を持つ.

瀧 和男 (正会員)

昭和 27 年生. 昭和 51 年神戸大学工学部電子工学科卒業. 昭和 54 年同大学院修士課程システム工学修了. 工学博士. 同年(株)日立製作所入社. 同社大みか工場にて制御用計算機システムの設計に従事. 昭和 57 年(財)新世代コンピュータ技術開発機構に出向. 以来逐次型および並列型推論マシンの研究開発に従事. 並列マシンのアーキテクチャ, 並列プログラミングなどに興味を持つ. 電子情報通信学会, IEEE 各会員.

中島 克人 (正会員)

昭和 28 年生. 昭和 52 年京都大学工学部電気第 2 工学科卒業. 昭和 54 年同大学院修士課程修了. 同年三菱電機(株)入社. 以来, 情報電子研究所にて汎用・専用計算機の開発に従事. 昭和 57 年より第五世代コンピュータ・プロジェクトに参画し, 逐次型推論マシンの開発・評価に従事. 昭和 60 年より(財)新世代コンピュータ技術開発機構に出向. 現在, 逐次型推論マシンの改良, 並列型推論マシンの研究・開発に従事. VLSI 向きプロセッサ, 並列マシン・アーキテクチャ等に興味を持つ.

三石 彰純 (正会員)

昭和 53 年大阪大学工学部応用物理学科卒業. 昭和 55 年同大学院修士課程修了. 同年三菱電機(株)入社. 以来, 計算機システムの性能評価, 階層記憶システム, 専用プロセッサの研究・開発に従事. 現在, 同社情報電子研究所勤務. 電子情報通信学会会員.