記号処理　27－4
（1984. 3. 12）

EVLISマシン上のPrologインタプリタとその動特性

大寺信行、斎藤年史、清原良三、西開地秀和、安井裕
（大阪大学・工学部）

1. はじめに

我々は、パーソナルで高速のProlog処理系の実現を目指している。そのために、既に我々の研究室で開発した高速リスト処理マシン―EVLISマシン―上に、マイクロ命令により、インタプリタをインプリメントした。各種の動特性を測定することにより、処理速度に影響を与える要因を探り、Prolog処理系に適した、マシン・アーキテクチャに関する知見を得る。

2. EVLISマシンの概要

EVLISマシン[1),2)]は、以下の各要素からなるリスト処理専用マシンである。

1) EVALⅡプロセッサ：　リスト処理向きに、設計したプロセッサであり、Prolog処理系のdirect execution machineとなっている。2－1に機能・特徴を示す。

2) I/Oプロセッサ：　入出力、EVALⅡへのマイクロプログラムのロード、EVALⅡの診断用インタフェースを介しての動特性測定のためのデータ蒐集等を行う。システム全体の管理も行う。

3) メインメモリ（MM）：　1セル40ビットで、現在32kセルを実装している。アクセスタイムは、リード時350nsec.、ライト時200nsec.（時分割バスのサイクルタイム50nsec.）である。アトム・リスト名領域、環境保存・作業領域等に使用する。

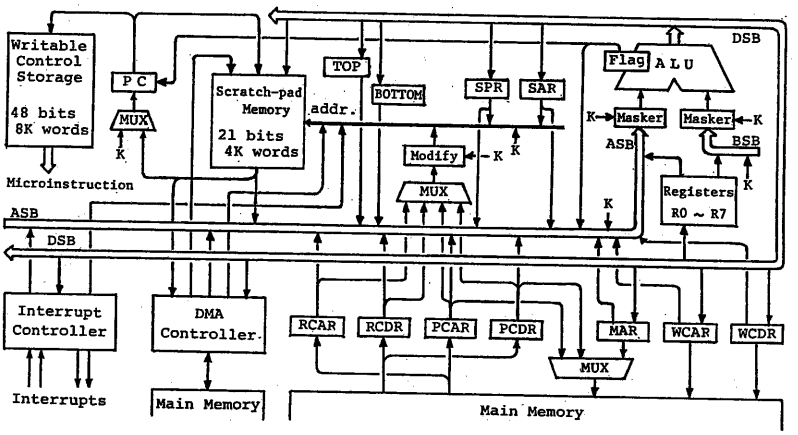
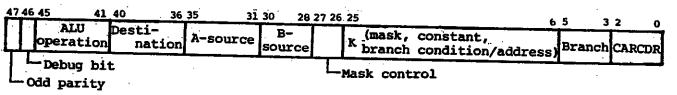


図1　EVALⅡプロセッサのブロック図

2-1. EVALⅡプロセッサ

内部構造を、図1に示す。次の機能・特徴をもつ。

1) マイクロ命令のフォーマット（図2）は、語長48ビット、8フィールドの水平



Debug bit : used to set breakpoints, to measure execution frequencies, etc.
ALU operation : specify a kind of ALU op. and flag controls
Destination, A-source, B-source : specify operands of an ALU op.
Mask control : specify masking for A- and/or B-source
K : specify a mask pattern, branch conditions and address, etc.
Branch : specify a kind of branches
CARCDR : specify a source and destinations of CARCDR-op.

図2　マイクロ命令のフォーマット

型で，ALU演算，分岐演算，マスク演算，CARCDR演算を，同時に実行可能である。マイクロプログラムは，Writable Control Storage（WCS）に格納し，その容量は8k語である。

2）マイクロ命令のフェッチと実行を，パイプライン化している。サイクルタイムは100nsec.であり，スクラッチパッドメモリへのアクセス時・加減算時は50nsec.延長される。

3）マスク演算により，データ（20ビット長）の任意の部分が抽出できる。

4）強力なディスパッチ機能がある。MMから読込んだデータは RCAR，RCDR，PCAR，PCDRにセットされ，これらのレジスタの指定部分の内容と，マイクロ命令内のベースアドレスに基づき，処理先への多方向分岐を1命令で行う（図3）。

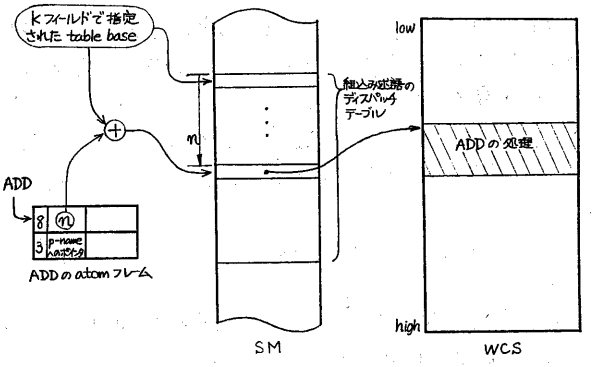


図3　ディスパッチ機能による分岐

5）スクラッチパッドメモリ（SM）は，call，return時のシステムスタック，ディスパッチ・割込み時の処理先テーブル，そして，ワークエリアとして機能する。語長20ビット，容量4k語，アクセスタイム50nsec.である。

6）診断用インタフェースは，EVALⅡの内部状態をI/Oプロセッサから直接アクセスするために設けた。マイクロ命令のデバッグビットを用いて，プログラムのブレークポイントのセットや，その計数により動特性の測定を行う。

3. Prolog処理系のインプリメント

処理系の構成を図4に示す。

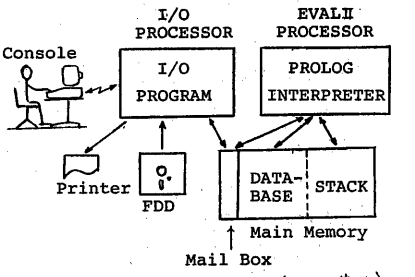


図4　Prolog処理系の構成

3-1. 処理系とマシン・アーキテクチャの整合性

EVALⅡでは，データがtag部とpointer部から構成され，マスク演算により，その個別抽出がALU演算等と同時に可能である。tag部にデータタイプ情報をもたせることにより，効率的なタイプチェックが行える。またtag部の情報を用いて，unificationの個別の処理へディスパッチ機能で分岐させることにより，高速化が期待できる。

既にLisp上にインプリメントしたインタプリタの動特性測定の結果[3]から，束縛スタックは頻繁に参照されており，高速なハードウェアスタック上に実現することが望ましい。EVALⅡのハードウェアスタック（SM上）は，システムスタックとして用いるため，束縛スタックとして利用できない。導出に用いるスタックは，容量の点も考慮して，MM上にソフトウェアスタックとする方法をとった。

環境保存にはスタックを用いる。変数束縛法の相違による動特性の差を明らかにするため、Structure-Sharing方式とStructure-Copying方式（S-S方式とS-C方式）の2種のアルゴリズムを採用した。[4]

ホーン節は、組込み述語"ENTRY"により登録され、その内部表現は、リスト構造となっている。その例を図5に示す。

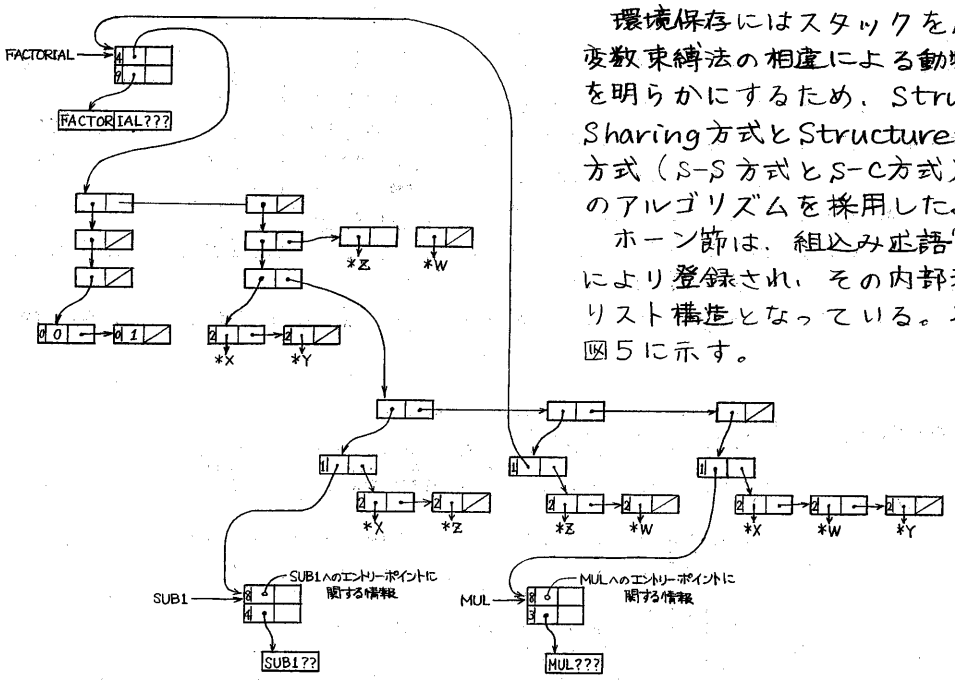
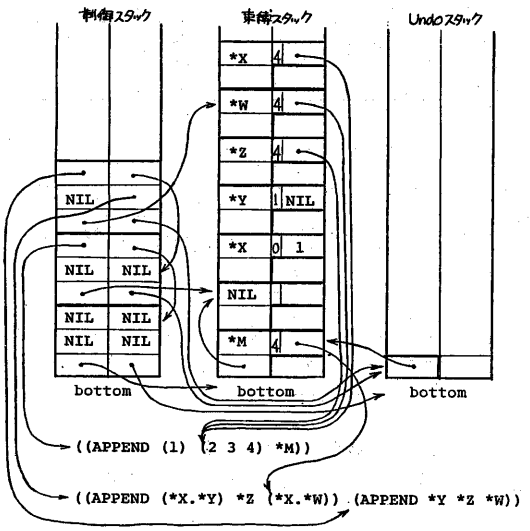
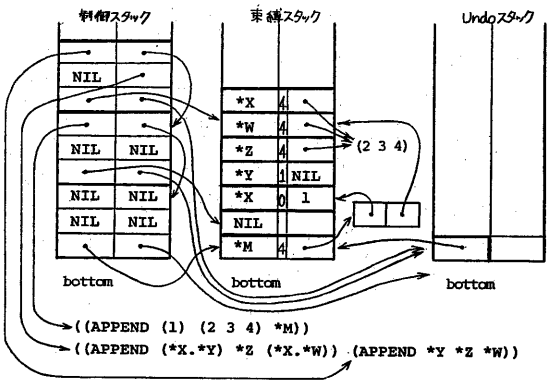


図5　ホーン節の内部表現の例

環境の保存に用いるスタックは、①制御（導出の制御を行い、他のスタックの管理を行う）、②束縛（束縛環境の保存を行う）、③undo（back track時に、束縛を解く変数束縛についての情報をもつ）の3種である。MM上のソフトウェアスタックとして実現した。各スタック間の関係の一例を図6に示す。



(a) S-S方式



(b) S-C方式

図6　リスト（1）と（2 3 4）のappendを行った後の状態

3-3 コーディングに関して

前節までに述べた、ハードウェア、ソフトウェアに関する諸点を考慮してインプリメントを行った。高速化を図るための手法として、①組込み述語の処理先への分岐、変数束縛値のデータタイプに応じた処理先への分岐に、ディスパッチ機能を用いたこと、②命令のフェッチ・実行のパイプラインを意識したコーディングを行ったこと、等が挙げられる。

表1 処理系の規模
（I/Oプログラムは除く）

束縛法	S-S方式	S-C方式
インタプリタ	477	766
unification	292	570
組込み述語	520	518
束縛値のprint・initialize・その他	647	724
合計	1644	2008

処理系の規模を命令数で表したものを表1に示す。組込み述語は、38個作成した。

4. 動特性と評価

処理系の評価のため、テストプログラムとして、①8-Queen問題、②HANOIの塔（円板8枚）を用いた。図7にプログラムを示す。

```
((HANOI *N)(MOVE *N LEFT CENTRE RIGHT))
((MOVE 0 *X *Y *Z)(CUT))
((MOVE *N *A *B *C)(SUB1 *N *M)(MOVE *M *A *C *B)
                   (INFORM *A *B)(MOVE *M *C *B *A))
((INFORM *X *Y)(WRITE MOVE A DISC FROM THE *X POLE TO THE *Y POLE))

((QUEENS *X *Y)(QUEENS1 *X NIL *Y)(WRITE *Y)(FAIL))
((QUEENS1 NIL *Y *Y))
((QUEENS1 *X *Y *Z)(SELECT *U *X *V)(CHECK *U *Y)
                   (QUEENS1 *V (*U.*Y) *Z))
((SELECT *X (*X.*Y) *Y))
((SELECT *U (*X.*Y)(*X.*V))(SELECT *U *Y *V))
((CHECK *P *Q)(CHECK1 *P *Q 1))
((CHECK1 *A NIL *B))
((CHECK1 *P (*Q.*R) *N)(NODIAG *P *Q *N)(ADD1 *N *M)(CHECK1 *P *R *M))
((NODIAG *P *Q *N)(ADD *N *P *T1)(SUB *P *N *T2)(NE *T1 *Q)(NE *T2 *Q))
```

図7 テストプログラム（HANOI-n, n-Queen）

インタプリタの動特性として、各命令毎の実行頻度を測定した。表2に、総実行ステップ数と全フェッチ命令数を示す。各問題とも、フェッチされた命令の90%は実行され、有効なコーディングができていることがわかる。

表2 フェッチ及び実行された命令数

束縛法 / 命令 / 問題	Structure Sharing		Structure Copying	
	8-Queen	HANOI-8	8-Queen	HANOI-8
実行された命令数	27,350,327	238,807	28,258,459	238,550
フェッチされた命令数	30,559,321	263,836	31,859,487	264,089
実行命令数/フェッチ命令数	0.895	0.905	0.887	0.903

EVALⅡの特徴の1つである水平型マイクロ命令の効率向上に対する寄与を表3に示す。この値は、全実行ステップ数に対し分岐演算、マスク演算が同時に実行できないと想定した場合に増加する実行ステップ数の割合を算出したものである。表3から、マスク演算による寄与は、S-S方式で約19%，S-C方式で22~25%と高く、分岐演算による寄与は、両方式とも約10%であり、水平型マイクロ命令による制御は、Prolog処理系と整合性の良いものであると言える。

表3 各種演算の並列実行の効果

束縛法 / 同時実行しない演算 / 問題	S-S方式		S-C方式	
	8-Queen	HANOI-8	8-Queen	HANOI-8
分岐演算	10.9%	10.4	9.6	11.3
マスク演算	19.4	18.8	22.6	25.0
分岐演算及びマスク演算	29.8	29.1	26.4	30.1

次に、MMへのアクセス回数をもとに、EVALⅡとMMの稼動率を算出したも

のを図8に示す。EVALⅡのサイクルタイムに比べ，MMのアクセスタイム（リード時825nsec.，ライト時525nsec.）が長いため，MMへの連続したアクセス（特にリードアクセス）は，EVALⅡのidle時間を増す。図8でEVALⅡのidle状態が，実行時間の43.5%を占めたのは，このことが原因と考えられる。そこで，インタプリタにおいて連続したリードアクセスをおこしている手続きの実行ステップ数および全実行ステップに対する比を求めたのが表4である。この手続きは，束縛スタックから変数束縛を求めるものである。表4から，この手続きは，全体の20%を占めており，EVALⅡの稼動率を低下させる原因となっていることがわかる。インタプリタを大きく5つの手続きに分けて，それぞれの実行ステップ数，および全実行ステップ数に対する比を求めたものが，表5である。束縛スタックから変数束縛を求める手続きは，unificationの処理（処理全体の60%を占める）の1/3を占めることがわかった。以上のことから，全体の処理効率を向上させるためには，束縛スタックを高速のハードウェアスタック上に実現するか，新たな変数束縛法の検討が必要と思われる。表4において，S-S方式とS-C方式は“unificationの処理”に要する実行ステップ数がほぼ同じであることがわかる。表6にS-C方式でstructureのcopyを行う手続きの実行ステップ数を示す。表4と表6から，8-Queenの問題でcopyに要した手続きのオーバーヘッドは，変数束縛を求める手続きの減少によってかなりおさえられることがわかる。したがって，S-C方式におけるcopyのオーバーヘッドが小さいアーキテクチャをもつマシン上では，S-C方式は有利であると考えられる。

また，望まれる機能として，2つのデータタイプによるディスパッチ機能が考られる。この機能によりunification時における個々の処理先への分岐を，効率的に行うことが可能となる。この機能による効率向上は，試算により，全実行ステップの約4%に相当することが予想される。

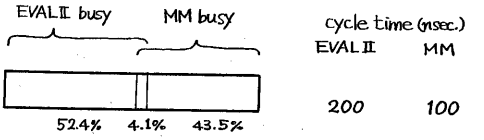


図8　EVALⅡとMMの稼動率

表4　束縛スタックから変数束縛を探索する手続きの実行ステップ数

問題		S-S方式	S-C方式
8-Queen	実行ステップ数	5,538,278	5,055,294
	比(%)	20.2	17.8
HANOI-8	実行ステップ数	49,237	49,237
	比(%)	20.6	20.6

表5　手続き毎の実行ステップ数

モジュール ＼ 問題 ／ 束縛法		8-Queen S-S方式	8-Queen S-C方式	HANOI-8 S-S方式	HANOI-8 S-C方式
算出の制御	実行ステップ数	6,580,952	6,580,952	68,273	68,273
	比(%)	24.1	23.2	28.6	28.5
Unificationの処理	実行ステップ数	16,408,391	16,818,538	154,709	154,964
	比(%)	60.0	59.2	64.8	64.8
組込み述語の処理	実行ステップ数	3,189,324	3,189,324	7,908	7,908
	比(%)	11.7	11.2	3.3	3.3
変数束縛を求す処理	実行ステップ数	641,544	1,272,058	2,805	3,060
	比(%)	2.3	4.5	1.2	1.3
候補の候補を求める処理	実行ステップ数	530,116	530,116	5,112	5,112
	比(%)	1.9	1.9	2.1	2.1
合計	総実行ステップ数	27,350,327	28,390,988	238,807	239,317
	比(%)	100	100	100	100

表6　structureをcopyする手続きの実行ステップ数

問題		S-C方式
8-Queen	実行ステップ数	1,564,228
	比(%)	5.5
HANOI-8	実行ステップ数	0
	比(%)	0

5. おわりに

インプリメントと動特性の測定により、処理効率に影響を与える要因を得ることができ、EVLISマシンのアーキテクチャを、Prolog処理系と整合させるために望まれる機能に関する知見が得られた。

なお現在、8-Queenの全解が7361.8msec.で、HANOI-8の解が76.8msec.で得られている。但し、EVALⅡのサイクルタイム100nsec.、MMのアクセスタイムがリード時600nsec.、ライト時400nsec.における値である。

参考文献

1) 前川博俊他，“高速LISPマシンとリスト処理プロセッサEVALⅡ—アーキテクチャとハードウェア構成—”，情報処理学会論文誌，Vol.24，No.5，pp683-688（1983）

2) 斎藤年史他，“高速LISPマシンとリスト処理プロセッサEVALⅡ—インタプリタによるマシンの評価—”，情報処理学会論文誌，Vol.24，No.5，pp689-695（1983）

3) 大寺信行他，“EVLISマシンにおけるPrologインタプリタの実現”，情報処理学会第27回全国大会論文集，3E-4（1983）

4) Bruynooghe,M.: The Memory Management of Prolog Impremetations, In "LOGIC PROGRAMMING", ACADEMIC PRESS (1982)