高速 LISP マシンとリスト処理プロセッサ EVAL II†

——インタプリタによるマシンの評価——

斎藤年史†† 土井俊雄†† 西川 岳††
前川博俊†† 安井 裕††

本論文は，試作した LISP マシンのインタプリタの構成と，処理速度，動特性等によるマシンの評価について論じている．本マシンは，リスト処理専用プロセッサ EVAL II，I/O プロセッサ，メインメモリからなり，とくに EVAL II は高速処理に重点をおいて設計，製作されている．本学会記号処理研究会で用いられているベンチマークテストの結果，本マシンは超大型計算機上の LISP インタプリタに匹敵する高速性能 (TPU-6 の shallow binding 版で 2,760 msec) を達成している．分岐演算と他の演算の並列処理，CARCDR 演算機能，ハードウェアスタックへのアクセスのパイプライン化などを備えた EVAL II のアーキテクチャに対して，動特性の測定から，LISP マシンへの適合性を確認した．メインメモリのアクセス完了待ちによる，EVAL II の処理効率低下の割合を調べるため，EVAL II とメインメモリの稼動率を算出した．また，EVAL II とメインメモリのサイクルタイム比と，マシンの処理速度および両者の稼動率との関係について検討した．これらの結果をもとに，EVAL II を複数台接続したリスト並列処理システムについて稼動率を推定した．その結果，EVAL II の稼動率は，単体の場合に比べて大きく変化しない（4 台接続した場合で約 10% 減少）ことがわかった．

1. はじめに

われわれは高速処理に重点をおいて LISP マシンを設計し試作した．このマシンは，リスト処理専用プロセッサ EVAL II，I/O プロセッサ，そして，メインメモリから構成され，とくに EVAL II はリスト処理向きアーキテクチャを具備するように設計されている．本マシンの構成は，EVAL II を複数台使用したリスト並列処理マシン—EVLIS マシン—において，EVAL II が 1 台の場合に相当する．マシンのハードウェア構成ならびに EVAL II のアーキテクチャの詳細については文献 1) を参照されたい．

EVLIS マシンにおいては，当初，インタプリタシステムを考えており，本論文ではインタプリタシステムに話を限定する．以下，このマシン上にインプリメントしたインタプリタの構成，ベンチマークプログラムによるマシンの処理速度，動特性の測定によるマシンの評価，ならびに，プロセッサ EVAL II とメインメモリの稼動率について論じる．

† Fast LISP Machine and List-Evaluation Processor EVAL II—Machine Evaluation on Interpreter by TOSHIFUMI SAITO, TOSHIO DOI, TAKESHI NISHIKAWA, HIROTOSHI MAEGAWA and HIROSHI YASUI (Department of Applied Physics, Faculty of Engineering, Osaka University).

†† 大阪大学工学部応用物理学科

2. インタプリタの構成

本マシンの処理言語の仕様は LISP 1.5[2] に準拠しており，そのインタプリタはマイクロプログラムの形で EVAL II の WCS に格納される．

2.1 リストセルとアトムの構造

リストセルとアトムはそれぞれ**図 1**，**図 2** に示す構造でメインメモリ上に割り付ける．car, cdr 各部の tag の値やアトムの関数属性を表す id の値で，処理先への多方向分岐を行い，処理の高速化を図った．関数が subr あるいは fsubr 属性をもつ場合，図 2 の body は SM 内のディスパッチテーブルの先頭からの変位を示し，WCS 内にある関数の処理開始番地がこのテーブルに格納してある．

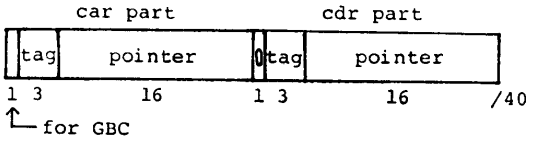
2.2 引数と変数の束縛

subr 属性をもつ関数への実引数や関数値の受け渡しは，次の理由から，汎用レジスタを用いて行う．

（1） 受け渡しの頻度が高く，アクセスの速さが全体の処理速度に大きく影響するため，高速の媒体であること．

（2） 実引数は，算術演算やポインタとしての演算など各種演算の対象となる可能性があり，これらの演算命令の三つのオペランドのすべてに直接に指定が可能であること．

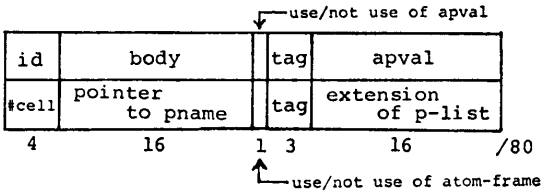
（3） 実引数の個数は実用上ある値以下に限定でき



tag	meaning of pointer
0	16 bit integer
1	pointer to a symbolic atom header
2	pointer to 39 bit integer
4	pointer to a list cell

図 1 リストセルの構造

Fig. 1 List cell structure.



id	meaning of body
1,4	pointer to def. of expr typed fn
3,6	pointer to def. of fexpr typed fn
8	inform. about entry point of subr typed fn
A	inform. about entry point of fsubr typed fn
0	—— (atom without fn property)

図 2 アトムフレームの構造

Fig. 2 Atom frame structure.

(本マシンでは4個以下とした)，一定個のレジスタのみを用いればよいこと．

変数束縛法の違いによる性能や動特性の差異をみるため，shallow binding と deep binding の二つの方式をとるインタプリタを製作した．インプリメンテーションでのそれぞれの変数束縛の内容を次に示す．

(A) shallow binding 版

・変数はアトムフレーム内の apval 部（図2）で束縛する．・同一変数名に対する旧値の保存には SM のスタックを用いる．・関数 function の実行時には，その時点で束縛を受けているすべての変数に対し a-list 形式で環境を保存する．

(B) deep binding 版

・変数は a-list 形式で束縛する．・expr 属性をもつ関数への実引数は，関数 evlis により一度リスト構造にした後，関数 pair で仮引数と対応づけを行うことにより束縛する[2]．

2.3 インプリメンテーション

本マシンに組み込んだ，subr, fsubr 属性をもつ関数は72個あり，これらのプログラムやその他のルーチンが WCS に占める命令のステップ数は**表1**のとおりである．EVAL II がメインメモリ上のデータを読み出す際には 350 nsec のアクセスタイムを要することから，マイクロプログラムのコーディング上，データの読出しアクセスを起動する命令を早く実行するように考慮した．このため，読出しの要求を起動したにもかかわらず，条件分岐により，読み出した内容を利用しない場合も起こりうるが，処理全体を通しては実行時間が短縮するものと思われる．マイクロアセンブラによるコーディングの一例として，deep binding 版での関数 apply のプログラムの一部を**図3**に示す．

表 1 インタプリタのプログラムステップ数

Table 1 Program steps of interpreter.

(unit: step)

Routine \ Binding	Shallow	Deep
Initialize	272	272
Kernel*	143	202
Various fns	650	644
Garbage col.	116	116
Others	105	105
Total	1,286	1,339

*...include apply, eval, evlis, and cond.

3. マシンの評価

処理系の性能測定にしばしば用いられた，本学会記号処理研究会における第2回 LISP コンテスト[3]の問題に対し，処理時間，動特性等の測定を行った．

3.1 処理速度

インタプリタの二つの版についての処理時間の測定結果の一部を，いくつかの他の処理系の結果とともに，**表2**に示す．これらの比較的小規模の問題に対する結果からではあるが，本マシンの処理速度は，超大型計算機上にインプリメントされた LISP システム[5),6)]に匹敵し，小型の LISP マシンとしてわれわれが目標とした高速性が達成されたものと考えられる．

3.2 各種演算出現頻度の動特性

コンテストの問題のうち，TARAI-3 と TPU-6 についての，動特性の測定結果を**表3**に示す．ここでは動特性として，命令フェッチ回数（フェッチのみで非実行の命令をも含む）と命令実行回数，および，命令フェッチ回数に対する各種命令のフェッチ回数や実行回数の割合を求めた．他の問題についても，各種命令の出現割合は表3(b)～(f)と同様の傾向を示すことを確認している．表3の各結果から次のことが考察される．

```
CM                                   Location    ALU          Mask control
address  Object code                 symbol      op.  Operands              | CARCDR    Branch

01E5  00 1D 09 0 2 FFFFF 0 0         APPLY:      LD   MAR(RSET),R1
01E6  0C 00 09 0 3 F0000 0 0         APPLY1:     EQ   R1,@NIL
01E7  04 00 09 0 3 2:0F:000 3 0                  AND  WOR,R1,@NIL                        RNE(Z)
01E8  00 0B 14 0 2 3:13:21B 5 0                  LD   R3,PCAR                            JNE(NZ)  APATOM
01E9  0C 0D 0B 4 3 10004 0 7                     XOR  R5,R3,LAMBDA            CDRSET
01EA  00 07 0F 0 2 3:13:212 4 0                  LD   SMPUSH,R7                          JMP(NZ)  APNOLAMBDA
01EB  0C 00 0A 0 3 F0000 0 3                     EQ   R2,@NIL                 CDR
01EC  0C 00 14 0 3 2:0F:20E 5 0                  EQ   PCAR,@NIL                          JNE(Z)   APLAMBDA5
01ED  00 09 16 0 2 3:0F:235 5 6                  LD   R1,RCAR                 CARSET     JNE(Z)   APERRARG
01EE  19 02 03 4 3 00001 0 0                     SUB  SAR,SPR,1
                                     ;
01EF  00 07 1A 0 2 FFFFF 0 0         APLAMBDA1:  LD   SMPUSH,ALL0
01F0  0C 00 15 0 3 F0000 0 0                     EQ   PCDR,@NIL
01F1  00 07 14 0 2 3:13:1EF 4 7                  LD   SMPUSH,PCAR             CDRSET     JMP(NZ)  APLAMBDA1
01F2  1C 0D 0D 4 3 00001 0 0                     ADD  R5,R5,1
                                     ;
01F3  00 1D 0A 0 2 FFFFF 0 0                     LD   MAR(RSET),R2
                                     ;
01F4  0C 00 15 0 3 F0000 0 0         APLAMBDA2:  EQ   PCDR,@NIL
01F5  00 05 14 0 2 3:13:1F4 4 7                  LD   SMSAR,PCAR              CDRSET     JMP(NZ)  APLAMBDA2
01F6  19 02 02 4 3 00002 0 0                     SUB  SAR,SAR,2
                                     ;
01F7  1C 0C 02 4 3 00001 0 0                     ADD  R4,SAR,1
01F8  00 1D 04 0 2 3:000 0 0                     LD   MAR(RSET),SMK(FREE)
01F9  0C 00 03 4 2 FFFFF 0 0                     EQ   SPR,R4
01FA  00 14 07 0 2 3:13:235 5 0                  LD   WCAR,SMPOP                         JNE(NZ)  APERRARG
01FB  00 16 07 0 2 FFFFF 0 0                     LD   WCDR,SMPOP
01FC  0C 00 11 4 3 00000 0 0                     EQ   MAR,0
01FD  00 1B 11 0 2 3:0F:359 7 0                  LD   MAR(WCXR),MAR                      CNE(Z)   GBCW
                                     ;
01FE  08 14 11 2 3 FFFFF 0 0                     OR   WCAR,MAR,@LIST
01FF  00 16 0F 0 2 FFFFF 0 0                     LD   WCDR,R7
0200  0C 00 15 0 3 F0000 0 0         APLAMBDA3:  EQ   PCDR,@NIL
0201  00 1B 15 0 2 3:0F:359 7 7                  LD   MAR(WCXR),PCDR          CDRSET     CNE(Z)   GBCW
0202  19 0D 0D 4 3 00001 0 0                     SUB  R5,R5,1
0203  08 0F 11 2 3 3:0F:20A 5 0                  OR   R7,MAR,@LIST                       JNE(Z)   APLAMBDA4
0204  00 14 07 0 2 FFFFF 0 0                     LD   WCAR,SMPOP
0205  00 16 07 0 2 FFFFF 0 0                     LD   WCDR,SMPOP
0206  0C 00 15 0 3 F0000 0 0                     EQ   PCDR,@NIL
0207  00 1B 15 0 2 3:0F:359 7 7                  LD   MAR(WCXR),PCDR          CDRSET     CNE(Z)   GBCW
0208  08 14 11 2 3 3:1C:200 4 0                  OR   WCAR,MAR,@LIST                     JMP      APLAMBDA3
0209  00 16 0F 0 2 FFFFF 0 0                     LD   WCDR,R7
                                     ;
020A  00 00 08 0 2 3:1C:237 6 0      APLAMBDA4:  NOP                                     CALL     EVAL
020B  00 04 15 0 2 3:000 0 0                     LD   SMK(FREE),PCDR
                                     ;
020C  00 0F 07 0 2 FFFFF 0 0                     LD   R7,SMPOP
020D  00 00 08 0 2 3:1C:000 3 0                  NOP                                     RNE
                                                 .
                                                 .
                                                 .
                                     ;
021B  00 09 0B 0 2 3:0:0:EF0 1 0     APATOM:     LD   R1,R3                   DISP(PCAR,U4)       APPLYDISP
021C  04 0D 0B 4 3 00003 0 0                     AND  R5,R3,3
                                     ;
021D  18 0D 0D 0 1 0:0F:229 5 0      APSUBR:     SB1  R5,R5,R0     MSK(B)=ALL0           JNE(Z)   APSUBR0
021E  00 1C 0A 0 2 3:0F:227 5 0                  LD   MAR(R),R2                          JNE(Z)   APSUBR1
021F  18 0D 0D 0 1 00000 0 0                     SB1  R5,R5,R0     MSK(B)=ALL0
0220  00 09 16 0 2 3:0F:225 4 0                  LD   R1,RCAR                            JMP(Z)   APSUBR2
0221  00 1C 17 0 2 FFFFF 0 0                     LD   MAR(R),RCDR
0222  00 0A 16 0 2 FFFFF 0 0                     LD   R2,RCAR
0223  00 1C 17 0 2 3:0:7:F00 1 0                 LD   MAR(R),RCDR             DISP(PCAR,L8)       SYSFUNTABLE
0224  00 0B 16 0 2 FFFFF 0 0                     LD   R3,RCAR
                                     ;
0225  00 00 08 0 2 3:0:7:F00 1 0     APSUBR2:    NOP                          DISP(PCAR,L8)       SYSFUNTABLE
0226  00 0A 16 0 2 FFFFF 0 0                     LD   R2,RCAR
                                                 .
                                                 .
                                                 .
```

Notes: (1) ALU op. -- "EQ x,y" and "NOP" are abbrs. for "XOR WOR,x,y" and "LD R0,R0", respectively.
(2) Operands -- "WOR" is a null register and "@z" is a constant valued register related to "z".
(3) Branch -- "JNE", "CNE", or "RNE" is a Jump, Call, or Return operation which inhibits the fetched next instruction from being executed when the branch is taken.

図 3 関数 apply のマイクロアセンブラによるコーディング例 (一部)

Fig. 3 Microassembler coding example of function apply (in part).

表 2 ベンチマーク問題の処理時間

Table 2 Execution times for benchmark programs—Interpreter version.

(unit: msec)

System (machine)		Experimental LISP machine		FAST-LISP[4] KOBE UNIV.	UTILISP[5] (M 200-H)	OLISP[6] (ACOS-1000)
Problem \ Var. binding		Shallow	Deep	Shallow	Shallow	Deep
TARAI	−3	24	40	55	21	34
	−4	461	755	1,013	420	638
	−5	12,559	20,556	27,538	11,210	17,344
TPU	−1	346	490	904	234	433
	−2	1,372	2,262	3,426	1,079	1,879
	−3	542	866	1,365	419	724
	−4	716	1,095	1,815	558	945
	−5	97	171	238	82	135
	−6	2,760	4,385	6,728	2,225	3,681
	−7	533	952	1,311	440	753
	−8	494	902	1,187	411	705
	−9	339	614	819	285	482

（1） 表3（c）より分岐演算を含む命令は35〜42%の高率となっている．しかも表3（b）で分岐演算のみの命令は7〜14%と低く，分岐演算の67〜79%が他の演算と同時に行われている．このことは分岐演算と他の演算の並列処理の有効性を示している．

（2） 分岐演算の結果，実際に分岐した場合，すでにフェッチ済みの命令（分岐演算命令の次の番地の命令）の実行を無効にしないためには，分岐演算命令の次に移動しても結果の変らない命令を，この位置に移すことができればよい．測定結果によれば，分岐の起こる頻度が20〜25%（表3（c））であるのに対し，フェッチのみの命令の頻度は9〜15%（表3（b））であり，分岐した場合でもその約半数はフェッチした命令を有効に実行して

表 3 問題 TARAI-3, TPU-6 での命令別出現頻度とメモリアクセス頻度

Table 3 Frequency of each instruction and memory access on some benchmark programs.

(a) Number of Fetched and Executed Instructions

Var. binding / Problem / Instruction	shallow binding		deep binding	
	TARAI-3	TPU-6	TARAI-3	TPU-6
Executed Instruction	124 158	13 076 556	174 448	17 816 250
Fetched Instruction	140 815	15 356 733	191 106	19 537 586

(unit:step)

(b) Frequency of ALU Operations

Var. binding / Problem / Instruction	shallow		deep	
	TARAI-3	TPU-6	TARAI-3	TPU-6
Data Transfer	48.3	46.7	39.9	40.5
Aritimetic	8.2	5.4	7.8	4.6
Logical	24.5	24.0	35.3	32.3
Equivalence	7.6	7.3	18.7	18.4
NOP(only Branch)	7.2	9.1	8.3	13.8
Executed Instr.	88.2	85.2	91.3	91.2
Non Executed Instr.	11.8	14.8	8.7	8.8
Fetched Instruction	100.0	100.0	100.0	100.0

(unit:%)

(c) Frequency of Branch Operations

Var. binding / Problem / Instruction	shallow		deep	
	TARAI-3	TPU-6	TARAI-3	TPU-6
Uncond. Branch	11.3	13.1	11.9	14.5
Cond. Branch	23.8	24.1	28.1	27.2
Branch Instr. Fetched	35.1	37.2	40.0	41.7
Jump	12.7	11.0	11.3	10.0
Call	4.1	4.9	3.3	3.9
Return	4.1	4.9	3.3	3.9
Dispatch	3.1	4.2	2.3	3.3
Instr. taken Branch	24.0	25.0	20.2	22.1
Fetched Instruction	100.0	100.0	100.0	100.0

(unit:%)

(d) Frequency of Access to a Scratchpad Memory

Var. binding / Problem / Instruction	shallow		deep	
	TARAI-3	TPU-6	TARAI-3	TPU-6
Read Access	20.7	19.2	14.2	20.7
Stack Pop	15.7	12.6	11.2	9.6
Write Access	16.1	14.6	12.9	11.8
Stack Push	12.3	12.7	11.2	10.1
Access Instr. to SM	36.8	33.8	27.1	32.5
Fetched Instruction	100.0	100.0	100.0	100.0

(unit:%)

(e) Frequency of Access to a Main Memory

Var. binding / Problem / Instruction	shallow		deep	
	TARAI-3	TPU-6	TARAI-3	TPU-6
Read Access	15.9	17.9	24.3	26.0
by CARCDR-op.	5.4	6.4	14.5	14.7
Write Access	2.9	2.3	3.5	2.0
Access Instr. to MM	18.8	20.2	27.8	28.0
Fetched Instruction	100.0	100.0	100.0	100.0

(unit:%)

(f) Frequency of Instructions using the K-field

Var. binding / Problem / Instruction	shallow		deep	
	TARAI-3	TPU-6	TARAI-3	TPU-6
as Const. Input Data	10.5	6.8	11.2	8.3
as Address of SM	0.0	2.5	1.4	2.7
as Inform. of Branch	35.1	37.2	40.0	41.7
Instr. using K-field	45.6	46.5	52.6	52.7
Fetched Instruction	100.0	100.0	100.0	100.0

(unit:%)

いる.

(3) 表3(d)よりSMのアクセス頻度が27～37%(その約3/4はスタックへのアクセス)の高率であり,SM へのアクセスの高速化とパイプライン化が全体の処理時間の短縮に効果を上げている.

(4) 表3(e)には,変数束縛法の違いによるメインメモリのアクセス頻度の差が現れている.また,読出しアクセスのうち CARCDR 演算機能によるものが shallow, deep の両束縛法でそれぞれ1/3, 1/2以上あり,この機能が活用されたことを示している.

CARCDR 演算機能は,リストデータへの効率よいアクセスを目的にハードウェアで備えた機能であり,メインメモリから読み出したリストセルの内容を直接アドレスとして,即時にメインメモリの読出しを行うことができる.メインメモリから読み出されたリストセルの car 部, cdr 部は,レジスタ RCAR, RCDR に格納されるが,命令のデスティネーションフィールドあるいは CARCDR 演算フィールドの指定により,レジスタ PCAR, PCDR にも格納できる.この PCAR, PCDR はアドレス兼データレジスタとして働き,命令の CARCDR 演算フィールドでそのどちらか一方のレジスタが指定されると,内容をリストセルへのアドレスポインタとして,ただちにメインメモリへのアクセスが起動される.したがって,メインメモリとこれらのレジスタの間でのデータ処理のみでリストをたぐる操作が行え,アドレスとなるデータがプロセッサ EVAL II の内部バスを経由しないため,ALU を用いる演算等と並行して CARCDR 演算が実行可能となる.

CARCDR 演算機能が高速化に与えた影響を調べるため,この機能を使用しない場合の deep binding 版の処理時間を測り,約6%の時間の増加をみた(表4).このことから,CARCDR 演算機能の効果が確かめられた.

(5) K フィールドの使用頻度は約50%で,その76～80%は分岐先に関する情報,残りが定数入力データ等に用いられている(表3(f)).アトム *T* と NIL は,Kフィールドを使用せず,BSB につながる専用レジスタに割り付けたことにより,これらのアトムを使用する演算と,分岐等でKフィールドを使用する演算とが1マイクロ命令で処理可能となる.Kフィールドの定数入力データとしての使用率は *T* と NIL を除けば 9～13% と低い.定数に対する演算と分岐演算を並列的に実行したい場合には,分岐演算命令の次に定数に対する演算命令を配することにより,Kフィールドの重複使用を避け,かつ,フェッチのみの命令をなくすことが可能である.

表4 CARCDR 演算機能の効果

Table 4 Effect of the CARCDR-operation facility.

Problem	CARCDR-operation		Ratio
	Use (ms)	Not use (ms)	
TARAI −5	20,556	21,439	1.04
TPU −5	171	183	1.07
−6	4,385	4,620	1.05
−7	952	1,018	1.07
SORT −100	520	537	1.03

3.3 EVAL II とメインメモリの稼動率

本マシンのインタプリタ方式では,リスト構造のプログラムとデータがともにメインメモリ(MM)に格納され,MM のアクセス頻度は20～30%(表3(e))に達する.MMのリードおよびライトアクセスタイムはおのおの 450 nsec, 300 nsec(クロックサイクルは 50 nsec)でありマイクロ命令の基本サイクルタイム 100 nsec よりも長い.したがって,MM に読出し要求を出した後,結果が得られる間に,EVAL II の続行できる処理がなくなる場合が生じる.この場合には,MM のアクセス完了待ちによる EVAL II の遊び時間が増大し,EVAL II の処理効率が低下する.

そこで最初に,EVAL II の遊び時間や MM のアクセス時間の割合を調べるため,コンテストの5問題について EVAL II と MM の稼動率を動特性の測定値より算出した.EVAL II の稼動率とは,EVAL II が処理中の時間(MM のアクセス完了待ちではない時間)の,全処理時間に対する割合であり,MM の稼動率とは,MM のアクセス動作中の時間(バスコントローラに対するアクセス要求から,そのアクセスの完了までの時間)の,全処理時間に対する割合である.

その結果(図4)から,EVAL II と MM の稼動率はそれぞれ 55～74%, 47～59% であり,MM のアクセス完了待ちにより EVAL II の稼動率が 26～45% 低下したといえる.このことは,MM のアクセスを高速化することにより全処理時間を最大 26～45% 短縮できる可能性があることを示している.また,MM の稼動率からも同様のことが推察できる.

次に,EVAL II と MM の速度(サイクルタイム)の変化がマシン全体の処理速度や両者の稼動率に与える影響を調べた.ここでは,問題 TARAI-3 と TPU-6 について,EVAL II と MM のクロックサイクルを

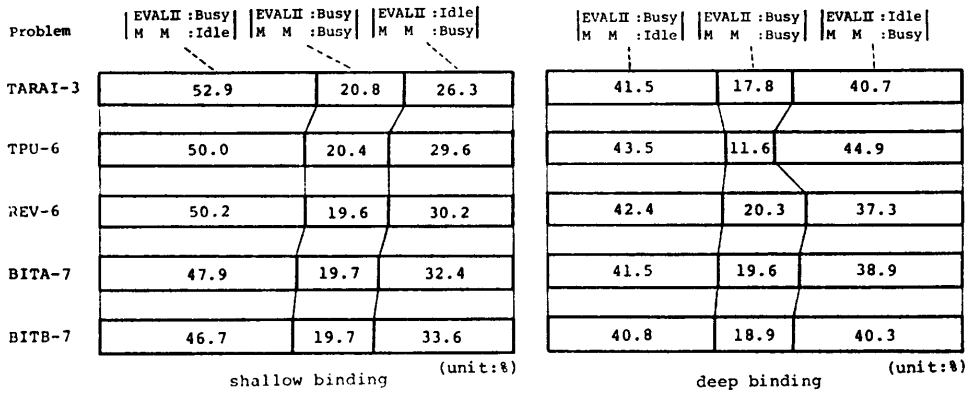


図 4 EVAL II とメインメモリの稼動率

Fig. 4 Operation ratio between the EVAL II and a main memory.

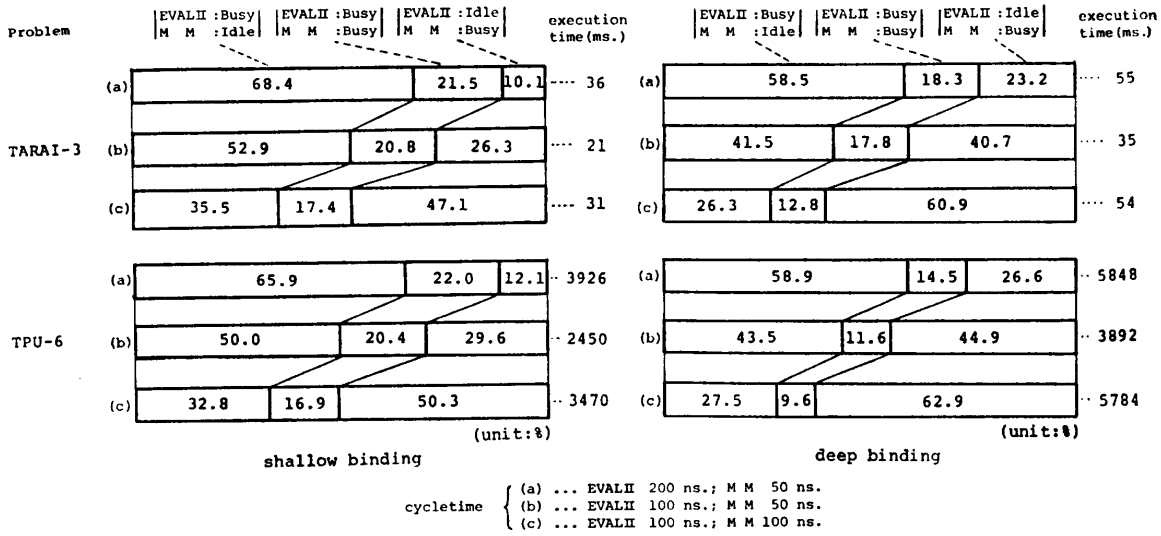


図 5 EVAL II とメインメモリのサイクルタイム比と稼動率の関係

Fig. 5 Relationships of cycletimes to operation ratio between the EVAL II and a main memory.

別々に2倍に変化させ，これらの問題の処理時間を測定し，稼動率を算出した．その結果を**図5**に示す．ここで，MMのクロックサイクルを100 nsecとしたときのリードおよびライトアクセスタイムはおのおの825，525 nsecである．図5の(b)は本マシンの現在のサイクルタイム比の場合であり，(a)と(c)は，それぞれ，EVAL IIあるいはMMのサイクルタイムを2倍にした場合である．逆にみれば，図の(a)はMMについて，図の(c)はEVAL IIについて，そのサイクルタイムを1/2に高速化した場合の稼動率とみることもできる．ただし，この場合は処理時間を図の値の1/2に換算する必要がある．

この結果から，MMとEVAL IIの速度をそれぞれ単独に2倍にした場合，全処理時間は約20%あるいは約25%短縮することがわかる．また，この場合の稼動率はそれぞれ約15〜20%変化し，EVAL IIとMMの稼動率に極端な差が出る．したがって，本マシンの現在の両者のサイクルタイム比は，両者の速度の変化がマシンの処理速度に敏感に影響を与える比となっており，マシンのより以上の高速化には，EVAL IIとMMのどちらの速度を上げることも有効であると考えられる．

最後に，上記の結果をもとに，EVLISマシンに対するEVAL IIとMMの稼動率を推定する．ここでは，リスト処理であるということを考慮して，次の仮定をおいた．

(1) EVLISマシンの各EVAL IIの稼動率とMMのアクセスの割合は，メモリ競合がなけれ

ば EVAL II が1台のシステムの場合に等しい.

(2) MM は8バンクからなり，各バンクへのアクセス頻度は一様で，バンクが異なれば同時にアクセスが可能である.

この仮定のもとに，EVAL II が1台のときの MM と EVAL II の稼動率がともに55%の場合（図5での deep binding 版の(b)の場合にほぼ等しい）をとりあげて，EVAL II を4台接続した EVLIS マシン（シミュレーション実験の結果[7]，処理時間短縮に有効と思われる EVAL II の台数は3～4台である）について稼動率を計算した.

その結果，メモリ競合により稼動率は，EVAL II が45%，MM が63%となったが，その変化は10%以内であり，単体の EVAL II での稼動率に関する上述の考察は，EVLIS マシンに対しても適用できるものと考えられる.

3.4 今後の開発に対する考察

LISP マシンとして，高速化をさらに進めるため，アーキテクチャの設計上，考慮すべき点を述べる.

(1) CARCDR 演算機能について：現在1系統の演算機能をもち，この効果についてはすでに述べた．しかし，関数 equal, pair のように2本のリストを並列に操作する場合にはその効果が発揮されない．したがって2系統以上の CARCDR 演算機能を考慮する必要がある.

(2) ハードウェアスタックについて：スタック演算の頻度は全命令の20～28%に達しており，この大部分は再帰的関数の実行に伴うものである．通常，関数呼び出し時には，スタックに中間結果を退避の後，戻り番地を保存し関数に分岐する.

スタックが1系統の EVAL II の場合，スタックへのアクセスがこの順序で行われるように命令を並べる必要があり，したがって，中間結果の退避と関数への分岐を並列に行えず，また，分岐命令の次に中間結果を退避する命令をおく（分岐の際の次命令フェッチを有効に利用する）こともできない．この点は，並列に同時にアクセス可能なスタックを2系統（コントロール用と値用）もつことにより解決される.

4. おわりに

試作した LISP マシンは，上述のごとく，所期の目標であった高速処理性能を示した．この理由として，EVAL II のすべてを論理 IC の組合せで構成することにより，既製のプロセッサのアーキテクチャに拘束されることなく，LISP マシンに適合したアーキテクチャを構築できたことや，基本命令サイクルタイム100 nsec を達成したことがあげられる.

また，LISP 特有の動特性やプロセッサとメインメモリの稼動率など，単体の EVAL II による LISP マシンのふるまいについて知見を得た．これらの結果の上にたち，現在，複数台の EVAL II を実装した EVLIS マシンを調整中であり，並列処理システムの結果については稿を改めて述べるつもりである.

本研究の一部は文部省科学研究費による.

参考文献

1) 前川，土井，西川，斎藤，安井：試作 EVLIS マシンの EVAL II と開発支援機能，情報処理学会研究会，記号処理 17-1 (1982).
2) McCarthy, J. et al.: *LISP 1.5 Programmer's Manual*, pp. 70-72, MIT Press, Cambridge, Mass. (1966).
3) 竹内郁雄：第二回 Lisp コンテスト，情報処理，Vol. 20, No. 3, pp. 192-199 (1979).
4) 瀧，金田，前川：LISP マシンの試作，情報処理学会論文誌，Vol. 20, No. 6, pp. 481-493 (1979).
5) 近山 隆：プログラミング・トゥールとしての Lisp 処理系，情報処理学会第23回全国大会論文集，pp. 225-226 (1981).
6) 高浜，日下，斎藤，安井：OLISP コンパイラのオブジェクト・プログラム最適化，情報処理学会第23回全国大会論文集，pp. 243-244 (1981).
7) 斎藤，宮崎，三石，安井：EVLIS マシンのための並列処理特性の測定と評価，情報処理学会第21回全国大会講演論文集，pp. 451-452 (1980).

(昭和57年9月27日受付)
(昭和58年4月19日採録)