試作EVLISマシンのEVALIIと開発支援機能

前川博俊，土井俊雄，西川 岳，高木伸哉
斎藤年史，安井 裕（大阪大学 工学部）

1 はじめに

LISPの高速な処理系の実現を目指して、複数台のプロセッサによりリスト処理を並列的に行なうLISPマシンーEVLISマシンーを既に提案した。[1-6] このマシンの全体の構成を、図1に示す。EVALプロセッサについては、基本的動作の確認が完了しており、近々2台のEVALプロセッサによる並列処理マシンが稼動の予定である。

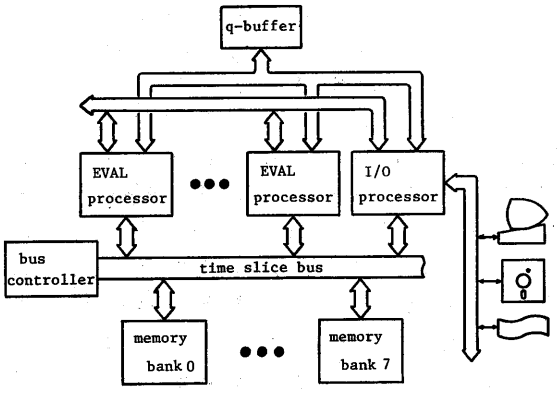


図1. EVLISマシンの構成

本稿では、リスト処理を効率よく行なうよう新たに設計したEVALプロセッサ（EVALII）のアーキテクチャを説明するとともに、マシンおよびソフトウェアの開発・改良のために準備した、プロセッサおよびメインメモリの診断用ツールについて述べる。

2 プロセッサEVALIIのアーキテクチャ

1. 設計方針 リスト処理という目的に即したアーキテクチャを備え、高速処理を実現できるハードウェアを設計するため、以下の点に留意した。

1) ALU演算は3アドレス方式とし、演算の高速性と柔軟性を考慮した。ALUの演算機能そのものは重視していない。

2) EVALII内では、メインメモリ（MM）のアドレスを、ひとつのデータとして扱うことにより、データとアドレスの結合性を高くした。

3) 分岐が多用されるため、マイクロ演算において分岐と他の演算を並列して行なえるようにした。関数のタイプやリストセルのタイプによる処理先への分岐を高速化するため、ディスパッチ機能を持たせた。

4) リストをたぐりながら演算を行なうために、マイクロ命令中にMMアクセスを独立して行なえる演算フィールド（CARCDR）を設けた。

2. マイクロ命令 プロセッサEVALIIのマイクロ命令は1語48ビットで図2のようなフォーマットになっている。ここでは、その各フィールドについて、機能、使用法等について説明する。

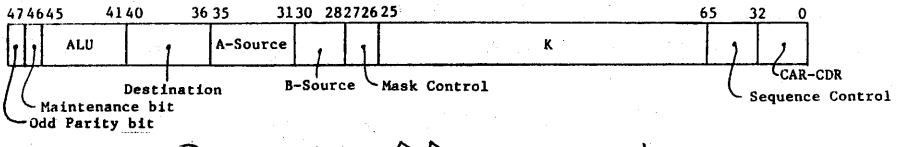


図2. マイクロ命令のフォーマット

1) Parity bit bit<47>
奇数パリティでI/Oプロセッサがプログラムをコントロールメモリ（WCS）へ

ロードする際にセットし、チェックはハードウェアで命令実行と並行して行なってあり、エラー発生時には、EVALプロセッサ内部割込みとして検出される。

2) Maintenance bit　bit<46>

Diagnostic Counterと併用し、マイクロプログラムの保守、解析に使用する。このビットがセットされていると、その命令が実行される際Diagnostic Counterに1が加えられ、このCounterがオーバーフローすると、マシンのクロックが停止するように設計されており、又、Counterは任意に初期値を設定できるようになっている。従ってこれを利用することにより、Break Pointの設定、ステップ動作などを行なわせることができ、プログラムの開発、保守に便利であるばかりでなく、プログラムの動作の解析にも有効である。

3) ALU Operation field　bit<45-41>

ALUの制御の指定を行なうフィールドで転送，算術演算（12種類），論理演算（3種類），シフト演算（4種類）の指定を行なう。

4) Destination field　bit<40-36>

演算を行なったデータの転送先を指定する。各種レジスタの他、スクラッチパッドメモリ（SM），MMへのアクセス，スタック操作，マシンステータスの設定なども行なう。

5) A-Source field　bit<35-31>

A-Source Busへ取り出す情報の選択指定を行なう。又、A-Sourceから全ビット1というパターンを出力しておいて，後述するK fieldを用いたマスクオペレーションとを組み合わせて、任意の定数をセットすることができる。

6) B-Source field　bit<30-28>

B-Source Busへ取り出す情報の選択指定を行なう。3ビットでR0～R7を指定するが、Mask Control fieldが11のときは、頻繁に用いるアトム等、いくつかの特別なパターンをB-Sourceより発生させることができる。これは、K fieldを使用していないので、例えば分岐命令のようなK fieldを用いたオペレーションとも並行して用いることができる。

7) Mask Control field　bit<27-26>

ALUに入る情報を、その手前にあるMaskerを用いてK fieldに記述してあるパターンでマスクでき，そのマスクする対象を指定する。

又、この2ビットが11のときは、前述したように、定数を発生させることができる。

8) K field　bit<25-6>

このフィールドは多目的に使用されるが、大別すると次の2つの場合がある。

(1) マスクパターンとして使用する場合

ⓐ Sourceのマスクパターンとして使用

これにより、20ビットの情報のうち必要な部分のみを取り出せる。

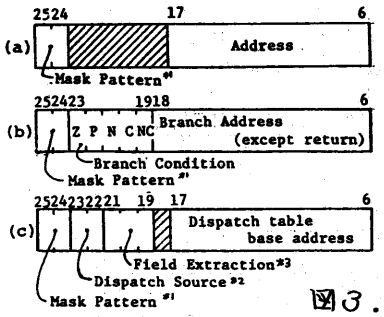
ⓑ 定数入力

Sourceから全ビットを出力しておき、このフィールドに記述してある任意のパターンでマスクすることにより任意の定数を演算に使用できる。

(2) マスクパターン以外の目的に使用する場合

ⓐ このフィールドをアドレスとしてSMをアクセスする場合（図3参照）

ⓑ Sequence Control fieldがJump，Call，Returnの場合（図3参照）



*1 Mask Pattern ---- 頻繁に使用する全ビット0，全ビット1，Tag部のみ1，Pointer部のみ1の4種類を指定

*2 Dispatch Source ---- ディスパッチの情報源とするRCAR，RCDR，PCAR，PCDRを選択指定

*3 Field Extraction ---- dispatch sourceのどの部分の情報を使用するか上位4/8ビット，下位4/8ビットより選択指定

図3．Kフィールド使用例

© Sequence Control field が dispatch の場合（図3参照）

以上のようにK fieldは他のフィールドと関連して多目的に利用される。ただし、1マイクロ命令内で異なった目的、例えば一方で定数を指定し、他方で分岐アドレスを指定するといった使用はできない。従ってアセンブラには1マイクロ命令内でK fieldが2重に使用されていないかをチェックする機能を持たせている。

9) Sequence Control field bit<5－3>

Jump，Call，Return，Dispatchの4種類があり、Dispatch以外はALUの演算結果によって、条件分岐ができる。さらにマシンの命令のフェッチと実行が、パイプライン化されているため、分岐が成功した場合、すでにフェッチしている命令を無視するか，そのまま実行するかの指定ができるようになっている。

又、Dispatch命令は、例えばセルのタグ部やアトムのID部のようなMM内の情報で，16又は128通りに分岐できる多方向分岐命令である。

10) CAR-CDR field bit<2－0>

CAR，CDRレジスタの制御を行なうフィールドで、他の操作と並行して、独立にMMに対しリードの起動をかけることができる。従って、例えばALUで演算を行なう一方で、リストをたぐる操作などが可能である。

3. EVAL IIのハードウェア（図4）

1) サイクルタイム マイクロ命令の基本サイクルタイムは100nSである。ALUで算術演算が行なわれる時とSMがリードされる時は、それぞれ50nSサイクルが延長される。PCAR，PCDR，RCAR，RCDRをリースで指定した時とディスパッチの行なわれる時、あるいはMMアクセスを起動する時に、MMがビジーであればレディーになるまでサイクルが延長される。マイクロ命令のフェッチと実行はパイプライン化している。WCSは48ビット×8K語で、I/Oプロセッサからアクセスできる。WCSはアクセスタイム35nSのスタティックメモリを使用している（45nSで動作可能である）。

2) スクラッチパッドメモリ（SM） スタックと作業領域に使用する。ディスパッチテーブル，ベクタ割り込みのためのアドレステーブルが格納される。スタックのオーバーフロー，アンダーフローは，スタックポインタの境界値レジスタT.OP，BOTTOMによって自動的に検出され、割り込みが発生する。SMへのライトは、他の演算とパイプライン化して次の命令サイクルの後半で行なう。従ってSMにライトした次の命令で同一番地をリードする場合は、ライトデータ

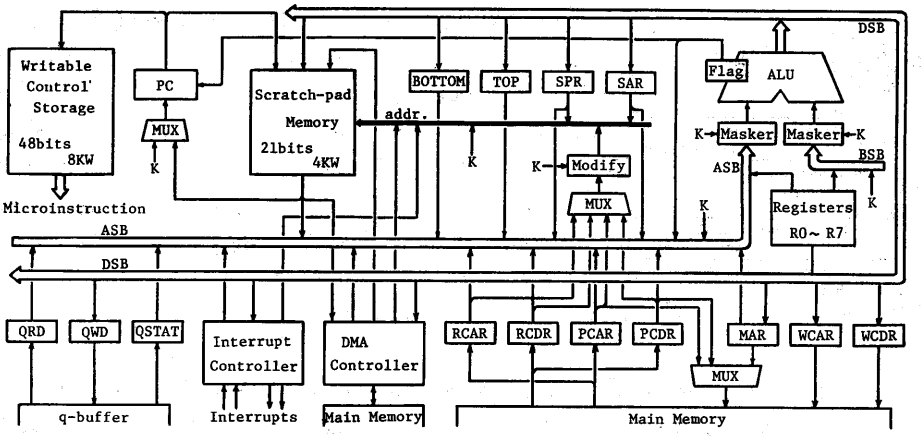


図4. プロセッサ EVALⅡのブロック図

がそのまま出力される。SMは21ビット×4K語（MSDは偶数パリティ）で、アクセスタイム35nSのスタティックメモリを使用している。

3) DMAコントローラ　SMとMMの間でDMA転送を行なう。EVALⅡ本体とは別のバスでMMと接続されているので、並列してアクセスできる。MMからは1台のプロセッサとして認識される。現在は非実装である。

4)割り込みコントローラ　プロセッサ間の割り込み、内部割り込みを処理する。

5)診断用インタフェース　I/Oプロセッサから、EVALⅡの内部状態を直接アクセスするための回路であり、次節で詳述する。

3. EVALⅡの診断用インタフェース

プロセッサEVALⅡ内には、内部状態を知るために、適当な場所にシフトレジスタをうめこみ、すべてのシフトレジスタを直列に接続した診断用インタフェースが設けられている。このシフトレジスタにデータをラッチするクロックと、データをシフトするクロックはプロセッサEVALⅡの状態には無関係にI/Oプロセッサから与えられるので、I/Oプロセッサはこの診断用インタフェースを通して随時EVALⅡプロセッサの内部状態を知ることができる。またI/Oプロセッサはこの診断用インタフェースを用いて、プロセッサEVALⅡのプログラムカウンタおよびDiagonostic Counterを任意の値に設定できる。

診断用インタフェースから読み出せる情報を表1に示す。この様に多くの内部状態を読み取ることができ、しかもI/Oプロセッサを通して見易い形式で表示できるために、ハードウェアのデバッグを効率よく行なうことができた。また診断用インタフェースはソフトウェアのデバッグおよびソフトウェアの動特性の測定に用いることができる。この他にI/Oプロセッサからは、プロセッサEVALⅡのクロックの制御（スタートおよびストップ），基本サイクルタイムの制御（100nS, 200nSおよびマニュアル）が可能である。

これらの機能を用いて、プロセッサEVALⅡのハードウェアとソフトウェアのデバッグを行なうためのプログラムを作成した。そのプログラムのコマンドとその簡単な説明を表2に示す。

表1　診断用インタフェースで読み出せる信号

A-Source bus
B-Source bus
Destination bus
スタックポインタ
SMアドレス
SMデータ
A-Source のマスク
B-Source のマスク
実行中のCMのアドレス
SMのステイタス
ディスパッチアドレス
割り込みベクトル
フラグ
MMアクセスのステイタス
Diagonostic Counter
割り込み

表2　デバッグのためのプログラムのコマンド

LOAD	CMにマイクロプログラムをロードする
MM	MMの読み出しおよび変更
CM	CMの読み出しおよび変更
SM	SMの読み出しおよび変更
GO	任意の番地から実行開始
RØ, R1 etc.	レジスタの読み出しおよび変更
DUMP	すべてのレジスタの読み出し
BPSET	Maintenance bitのセット
BPCLR	Maintenance bitのリセット
DISPLAY	診断用インタフェースの内容の表示
MCK, FCK, SCK	サイクルタイムの制御
STOPL, STRTH etc.	クロックの制御
ASM	パッチのための1パスアセンブラ
CMCLR	CMにNOPを書き込む
SMCLR	SMをゼロクリアする

4　メインメモリの共有バスと競合の計測

1. 共有バス　EVLISマシンは、各々最大8台のプロセッサとメモリバンクからなり、プロセッサのMMアクセスは、共有バスを通して行なう。プロセッサ・バンク間のアクセスは、共有バスを50nSで時分割する事により、複数のアクセスが多重化して行なわれる。バスの競合は、バンク間に優先順位を付けて制御する。この優先順位は、1クロックサイクル（50nS）毎に変化する。一つのバンクに対するプロセッサ間の優先順位は固定である。現在、この共有バスの競合状態を調べるツールを製作中である。

MMのリードアクセスタイムは375nS，リードサイクルタイムは250nSである。ライトアクセスタイムは125nS，ライトサイクルタイムは200nSである。

2. 競合測定器　プロセッサからMMへのアクセスは、MM Processor Interface（MMP.I.）を通して行なう。プロセッサからアクセス要求（RQ信号）がMMP.I.を通してBus Controller（B.C.）に送られると、アクセス許可（AK信号）がB.C.からMMP.I.に返される。アクセス終了後、MMP.I.からプロセッサにRDY信号が返される。RQ信号からAK信号までのMMのシステムクロック数を競合の度合いと考える。

この測定器は、度数分布すなわち計数されたクロック数x_iとその度数y_iを結果として出力する。x_iの計数は、一つ置きのアクセス要求に対して行なうが、アクセスが多数回行なわれるので、競合の様子を推定するには十分である。測定器1台では1つのプロセッサに関する結果が得られ、複数台のプロセッサに対しては、測定対象を順に変えながら、同一プログラムを繰り返し実行させることにより測定する。

この競合測定器のブロック図を図5に示す。RQ信号からAK信号までのクロック数 x_i を ϕ Counterで計数し、この x_i をアドレスとしてメモリに入力することにより、メモリを度数 f_i を計数するカウンタとして使っている。ϕ Counterは10ビットとし、x_i を0～1023の範囲で取れるようにした。x_i がこの範囲を越えるものは、全てひとまとめにしてOverflow Counterでその度数を計数する。f_i の範囲については、メモリを20ビットとし、そのうち19ビットを度数 f_i の計数に使用し、残りの最上位1ビットをオーバーフローの検出に用いる。1つのプログラムが終了した後、I/Oプロセッサを通じて各計測の結果を出力する。

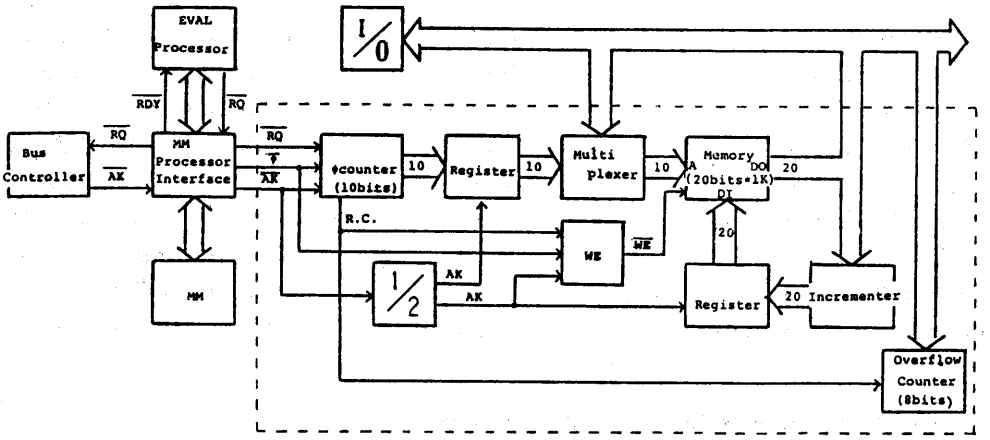


図5．競合測定器のブロック図

5 おわりに

上述のマシン構成のうち、g-butterとDMAコントローラを除いた。単一のEVALプロセッサの環境での、EVALⅡ、I/O両プロセッサの基本的な動作の確認を終えた。現在、インタプリタを組み込んでの総合的なテストを行なっており、又、2台目のプロセッサEVALⅡの製作も7割方が完了している。今後、複数台のEVALプロセッサによるリスト処理を早急に実現し、処理効率やメモリ競合の計測を行なって、並列的なリスト処理方式の問題点等を明らかにしたい。

[参考文献]

1) 安井他，"evlisの並列処理"，情報処理学会 第20回 全国大会講演論文集，8K-8，1979

2) 安井他，"LISPでの並列処理における動的特性とEVLISマシンの構成"，記号処理研究会，記号処理 10-4，1979

3) 前原他，"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 11-9，1980

4) 西川他，"EVLISマシンの並列処理アルゴリズムとそのインプリメンテーション，"情報処理学会 第23回 全国大会講演論文集 4H-7，1981

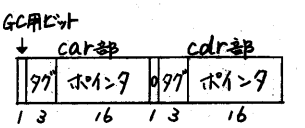
5) 土井他，"EVLISマシンの管理とI/Oプロセッサ，"情報処理学会 第23回 全国大会講演論文集，4H-8，1981

6) 前川他，"LISP並列処理システム－EVLISマシン－とプロセッサEVALⅡ，"情報処理学会 第23回 全国大会講演論文集 4H-9，1981

付　録

1. リスト・セルとアトムの構造

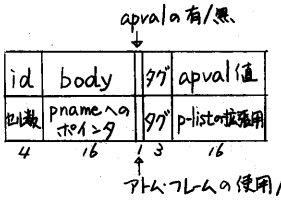
本LISPシステムでは、各EVALプロセッサから共通にアクセスされるリスト・セルやアトム等の要素をメイン・メモリに格納する。リスト・セルは、1セルが40ビットで、図A1の構造をとり、タグはcar部およびcdr部のポインタの内容を示している。このタグが'2'の場合、ポインタは並列処理のときのスタック・フレームの先頭アドレスである。タグが'1'の場合は文字アトムを表わし、ポインタは図A2に示すアトム・フレームを指す。



タグ	ポインタの意味
0	数値アトム(2の補数形式の整数)
1	文字アトム
2	フレーム
4	リスト

図 A1　リスト・セルの構造

アトム・フレーム内のidはアトムの関数属性を表現しており、これによりインタプリタでの関数型による処理先の分岐を容易にしている。またidが'8'および'A'の場合、アトムがSubr型の関数であることを表わし、このとき、bodyはSM内ディスパッチテーブルの先頭からの変位を示しており、このテーブル・エントリにCM内の関数の処理ルーチンの入口点を格納している。アトムがSubr型の関数の場合には、bodyの下2ビットがこの関数への引数の個数(0～3)を示すようにマッピングしてあり、インタプリタでの実引数の個数とのチェックを行なっている。



id	body
1又は4	expr型関数の定義式へのポインタ
3又は6	fexpr型　〃
8	subr型関数のentry pointに関する情報
A	fsubr型　〃
0	——（関数属性をもたないアトム）

図 A2　アトム・フレームの構造

2. プログラムのコーディング例

本文で述べたアーキテクチャをもつマシンに対するプログラムのコーディングの一例として、通常の単体マシンのLisp（EVALプロセッサが1台のみ）におけるインタプリタの一部を図A3に示す。

このプログラムにおいては、変数の束縛にa-listを用い、現在のa-listへのポインタはR7に格納されている。また関数に対する引数は各々、R1, R2, …を、値はR1を介して受け渡される。図3で、命令NOPはLD WOR, R0と、命令EQ x, yはXOR WOR, x, yとそれぞれ等価である。EVALプロセッサがメインメモリに対し読み出しの要求をした場合、その結果がRCAR等のレジスタに得られるまでにメモリ・アクセスの時間を要する。このことを考慮して、メモリ読み出しの命令をできるだけ早い時点で行なうようにコーディングしているため、インタプリタの実行時、場合によっては煩雑な命令の実行も行なわれるが、平均点には、実行速度が向上するものと思われる。

```
                          643 ;
                          644 ;*****  APPRY (A-LIST BINDING)  *****
                          645 ;
01E5 00 1D 09 0 2 FFFFF 0 0    646 APPLY:   LD   MAR(RSET),R1
01E6 0C 00 09 0 3 F0000 0 0    647 APPLY1:  EQ   R1,@NIL
01E7 04 00 09 0 3 2:0F:000 3 0 648          AND  WOR,R1,@NIL                    RNE(Z)
01E8 00 0B 14 0 2 3:13:21B 5 0 649          LD   R3,PCAR                        JNE(NZ) APATOM
01E9 0C 0D 0B 4 3 10004 0 7    650          XOR  R5,R3,LAMBDA          CDRSET
01EA 00 07 0F 0 2 3:13:212 4 0 651          LD   SMPUSH,R7                      JMP(NZ) APNLAM
01EB 0C 00 0A 0 3 F0000 0 3    652          EQ   R2,@NIL               CDR
01EC 0C 00 14 0 3 2:0F:20E 5 0 653          EQ   PCAR,@NIL                      JNE(Z)  APLAM5
01ED 00 09 16 0 2 3:0F:22D 5 6 654          LD   R1,RCAR               CARSET   JNE(Z)  APERAG
01EE 19 02 03 4 3 00001 0 0    655          SUB  SAR,SPR,1
                          656 ;
                                                 .
                                                 .
                                                 .
                          711 ;
021B 00 09 0B 0 2 3:0:0:EF0 1 0 712 APATOM:  LD   R1,R3          DISP(PCAR,U4) APPLYDISP
021C 04 0D 0B 4 3 00003 0 0    713          AND  R5,R3,3
                          714 ;
021D 18 0D 0D 0 1 0:0F:229 5 0 715 APSUBR:  SB1  R5,R5,R0 MSK(B)=ALL0           JNE(Z)  APSUB0
021E 00 1C 0A 0 2 3:0F:227 5 0 716          LD   MAR(R),R2                      JNE(Z)  APSUB1
021F 18 0D 0D 0 1 00000 0 0    717          SB1  R5,R5,R0 MSK(B)=ALL0
0220 00 09 16 0 2 3:0F:225 4 0 718          LD   R1,RCAR                        JMP(Z)  APSUB2
0221 00 1C 17 0 2 FFFFF 0 0    719          LD   MAR(R),RCDR
0222 00 0A 16 0 2 FFFFF 0 0    720          LD   R2,RCAR
0223 00 1C 17 0 2 3:0:7:F00 1 0 721         LD   MAR(R),RCDR    DISP(PCAR,L8) SYSFUNTABLE
0224 00 0B 16 0 2 FFFFF 0 0    722          LD   R3,RCAR
                          723 ;
0225 00 00 0B 0 2 3:0:7:F00 1 0 724 APSUB2:  NOP                 DISP(PCAR,L8) SYSFUNTABLE
0226 00 0A 16 0 2 FFFFF 0 0    725          LD   R2,RCAR
                          726 ;
0227 00 00 08 0 2 3:0:7:F00 1 0 727 APSUB1:  NOP                 DISP(PCAR,L8) SYSFUNTABLE
0228 00 09 16 0 2 FFFFF 0 0    728          LD   R1,RCAR
                          729 ;
0229 00 00 08 0 2 3:0:7:F00 1 0 730 APSUB0:  NOP                 DISP(PCAR,L8) SYSFUNTABLE
022A 00 00 08 0 2 FFFFF 0 0    731          NOP
                          732 ;
022B 00 14 09 0 2 3:1C:183 4 0 733 APERA2:  LD   WCAR,R1                        JMP     WERROR
022C 00 16 1B 0 0 00001 0 0    734          LD   WCDR,1H
                          735 ;
022D 00 14 09 0 2 3:1C:183 4 0 736 APERAG:  LD   WCAR,R1                        JMP     WERROR
022E 00 16 1B 0 0 00002 0 0    737          LD   WCDR,2H
                          738 ;
                          739 ;*****  EVAL (A-LIST BINDING)  *****
                          740 ;
022F 00 1D 09 0 2 FFFFF 0 0    741 EVAL:    LD   MAR(RSET),R1
0230 04 00 09 0 3 F0000 0 0    742          AND  WOR,R1,@NIL
0231 08 00 1A 1 2 2:13:237 5 0 743          OR   WOR,ALL0,R1 MSK(AB)=T0         JNE(NZ) EVATOM
0232 00 00 08 0 2 3:0F:000 3 2 744          NOP                        CAR      RNE(Z)
0233 04 00 14 0 3 F0000 0 0    745          AND  WOR,PCAR,@NIL
0234 00 00 08 0 2 3:0F:274 5 0 746          NOP                                 JNE(Z)  EVELSE
0235 00 00 08 0 2 3:2:0:EE0 1 0 747         NOP                 DISP(RCAR,U4) EVALDISP
0236 04 0D 16 4 3 00003 0 0    748          AND  R5,RCAR,3
                          749 ;
0237 00 0D 09 0 2 FFFFF 0 0    750 EVATOM:  LD   R5,R1
0238 08 09 15 4 3 00000 0 0    751          OR   R1,PCDR,0
0239 00 1D 0F 0 2 3:07:000 3 0 752          LD   MAR(RSET),R7                   RNE(NN)
023A 0C 00 0F 0 3 F0000 0 2    753          EQ   R7,@NIL               CAR
023B 0C 00 16 5 2 3:0F:240 5 0 754          EQ   RCAR,R5                        JNE(Z)  EVERA8
023C 0C 00 15 0 3 2:0F:000 2 0 755 EVATM1:  EQ   PCDR,@NIL                     RET(Z)
023D 00 09 17 0 2 FFFFF 0 7    756          LD   R1,RCDR               CDRSET
023E 00 00 08 0 2 3:13:23C 4 2 757          NOP                        CAR      JMP(NZ) EVATM1
023F 0C 00 16 5 2 FFFFF 0 0    758          EQ   RCAR,R5
```

location　microinstruction　ラベル　ALU演算　operand　CAR-CDR演算　分岐演算

fn がアトムのときの分岐

fn が subr型 関数のときの処理

car[form] がアトムのときの分岐

form がアトムのときの処理

図A3. プログラムのコーディング例

```
                                    759  ;
0240 00 14 0D 0 2 3:1C:183 4 0      760  EVERA8:  LD   WCAR,R5                          JMP     WERROR
0241 00 16 1B 0 0 00003 0 0         761           LD   WCDR,3H
                                    762  ;
0242 00 07 16 0 2 FFFFF 0 0         763  EVEXPR:  LD   SMPUSH,RCAR
0243 00 09 15 0 2 3:1C:278 7 0      764           LD   R1,PCDR                          CNE     EVLIS
0244 00 0A 09 0 2 3:1C:1E5 4 0      765           LD   R2,R1                            JMP     APPLY
0245 00 09 07 0 2 FFFFF 0 0         766           LD   R1,SMPOP
                                    767  ;
0246 04 09 16 4 3 5FFFF 0 0         768  EVFEXPR: AND  R1,RCAR,5FFFFH
0247 00 1C 04 0 2 3:000 0 0         769           LD   MAR(R),SMK(FREE)
0248 0C 00 11 4 3 00000 0 0         770           EQ   MAR,NIL
0249 08 17 1A 0 3 2:0F:333 7 0      771           SET  WWCDR,@NIL                       CNE(Z)  GBC
024A 00 15 0F 0 2 FFFFF 0 0         772           LD   WWCAR,R7
024B 08 16 11 2 3 FFFFF 0 0         773           OR   WCDR,MAR,@LIST
024C 00 14 15 0 2 FFFFF 0 0         774           LD   WCAR,PCDR
024D 00 1C 17 0 2 FFFFF 0 0         775           LD   MAR(R),RCDR
024E 0C 00 11 4 3 00000 0 0         776           EQ   MAR,NIL
024F 00 1B 11 0 2 3:0F:344 7 0      777           LD   MAR(WCXR),MAR                    CNE(Z)  GBCW
0250 08 0A 11 2 3 3:1C:1E5 4 0      778           OR   R2,MAR,@LIST                     JMP     APPLY
0251 00 04 17 0 2 3:000 0 0         779           LD   SMK(FREE),RCDR
                                    780  ;
0252 1B 0D 0D 0 1 0:0F:26E 5 0      781  EVSUBR:  SB1  R5,R5,R0 MSK(B)=ALL0                JNE(Z)  EVSUB0
0253 1B 0D 0D 0 1 0:0F:26A 4 3      782           SB1  R5,R5,R0 MSK(B)=ALL0    CDR      JMP(Z)  EVSUB1
0254 00 07 14 0 2 FFFFF 0 0         783           LD   SMPUSH,PCAR
0255 00 07 17 0 2 3:0F:264 5 0      784           LD   SMPUSH,RCDR                      JNE(Z)  EVSUB2
0256 00 09 16 0 2 3:1C:22F 7 0      785           LD   R1,RCAR                          CNE     EVAL
0257 00 1C 07 0 2 FFFFF 0 0         786           LD   MAR(R),SMPOP
0258 00 07 09 0 2 FFFFF 0 0         787           LD   SMPUSH,R1
0259 00 07 17 0 2 FFFFF 0 0         788           LD   SMPUSH,RCDR
025A 00 09 16 0 2 3:1C:22F 7 0      789           LD   R1,RCAR                          CNE     EVAL
025B 00 1C 07 0 2 FFFFF 0 0         790           LD   MAR(R),SMPOP
025C 00 07 09 0 2 FFFFF 0 0         791           LD   SMPUSH,R1
025D 00 09 16 0 2 3:1C:22F 7 0      792           LD   R1,RCAR                          CNE     EVAL
025E 00 0B 09 0 2 FFFFF 0 0         793           LD   R3,R1
025F 00 0A 07 0 2 FFFFF 0 0         794           LD   R2,SMPOP
0260 00 09 07 0 2 FFFFF 0 0         795           LD   R1,SMPOP
0261 00 1C 07 0 2 FFFFF 0 0         796  EVSUBE:  LD   MAR(R),SMPOP
0262 00 00 0B 0 2 3:2:7:F00 1 0     797           NOP                  DISP(RCAR,L8) SYSFUNTABLE
0263 00 00 08 0 2 FFFFF 0 0         798           NOP
                                    799  ;
0264 00 09 16 0 2 3:1C:22F 7 0      800  EVSUB2:  LD   R1,RCAR                          CNE     EVAL
0265 00 1C 07 0 2 FFFFF 0 0         801           LD   MAR(R),SMPOP
0266 00 07 09 0 2 FFFFF 0 0         802           LD   SMPUSH,R1
0267 00 09 16 0 2 3:1C:22F 7 0      803           LD   R1,RCAR                          CNE     EVAL
0268 00 0A 09 0 2 3:1C:261 4 0      804           LD   R2,R1                            JMP     EVSUBE
0269 00 09 07 0 2 FFFFF 0 0         805           LD   R1,SMPOP
                                    806  ;
026A 00 09 16 0 2 3:1C:22F 7 0      807  EVSUB1:  LD   R1,RCAR                          CNE     EVAL
026B 00 1C 07 0 2 FFFFF 0 0         808           LD   MAR(R),SMPOP
026C 00 00 0B 0 2 3:2:7:F00 1 0     809           NOP                  DISP(RCAR,L8) SYSFUNTABLE
026D 00 00 0B 0 2 FFFFF 0 0         810           NOP
                                    811  ;
026E 00 00 0B 0 2 3:2:7:F00 1 0     812  EVSUB0:  NOP                  DISP(RCAR,L8) SYSFUNTABLE
026F 00 00 08 0 2 FFFFF 0 0         813           NOP
                                    814  ;
0270 00 00 0B 0 2 3:2:7:F00 1 0     815  EVFSUBR: NOP                  DISP(RCAR,L8) SYSFUNTABLE
0271 00 09 15 0 2 FFFFF 0 0         816           LD   R1,PCDR
                                    817  ;
0272 00 14 09 0 2 3:1C:183 4 0      818  EVERA9:  LD   WCAR,R1                         JMP     WERROR
0273 00 16 1B 0 0 00004 0 0         819           LD   WCDR,4H
                                    820  ;
0274 00 07 14 0 2 FFFFF 0 0         821  EVELSE:  LD   SMPUSH,PCAR
0275 00 09 15 0 2 3:1C:278 7 0      822           LD   R1,PCDR                          CNE     EVLIS
0276 00 0A 09 0 2 3:1C:1E5 4 0      823           LD   R2,R1                            JMP     APPLY
0277 00 09 07 0 2 FFFFF 0 0         824           LD   R1,SMPOP
                                    825  ;
```

car[form]が expr型 関数のときの処理

car[form]が fexpr型 関数のときの処理

car[form]が subr型 関数のときの，引数の個数による振り分け

引数3個のとき

引数2個のとき

引数1個のとき

引数0個のとき

car[form]が fsubr型 関数のときの処理

図A3. プログラムのコーディング例（続き）