2レベルマイクロプログラム制御計算機MUNAPのシステム記述言語：MSDL

A System Description Language for a Two-Level Microprogrammed Computer MUNAP : MSDL

山崎勝弘[+], 橋本信行[+], 金井裕之[+]
Katsuhiro YAMAZAKI, Nobuyuki HASHIMOTO, Hiroyuki KANAI
馬場敬信[+], 奥田健三[+], 橋本和彦[++]
Takanobu BABA, Kenzo OKUDA, Kazuhiko HASHIMOTO
+宇都宮大学工学部 (Faculty of Engineering, Utsunomiya University)
++日立ソフトウェアエンジニアリング (HITACHI Software Engineering)

1. まえがき

高級言語マシン，マシンチューニングなどのダイナミックマイクロプログラミングの応用分野を拡大させるためには，それらのファームウェアを短期間に，かつ信頼性の高いシステムとして開発できる言語が必要である．従来のマイクロプログラミング言語では，実行効率の改善などの最適化に重点がおかれ，必ずしも高信頼性を指向したものではない．しかし，近年のハードウェア価格の急激な低下に伴い，信頼性が高く，人間にとって使いやすいアーキテクチャをハードウェアとファームウェアでもサポートすることが必要となってきている．

我々の研究室で開発した2レベルマイクロプログラム（μP）制御計算機MUNAPは，多重プロセッサユニット（PU）による並列処理，非数値処理用ハードウェアなどを特徴としている[1][2]．現在，MUNAPのμPは機械依存のレジスタ転送形式を有するμPアセンブリ語により記述されている．本言語では順序制御におけるテスト機能の一様化，複数PUで行われる同一演算を一文で記述可能など，μPを簡潔に記述できるよう配慮されている．また，異なるマイクロ命令間でのナノプログラム（nP）の共用化およびnP領域の最適化により，制御記憶容量の削減が図られている[3]．

一方，高級言語マシン，データベースシステムなどのMUNAPの応用分野において，大規模なシステムプログラムを短期間に，かつ信頼性の高いシステムとして開発するためには，エラー検出，デバッグ支援などを考慮したシステム記述言語が必要である．特に，MUNAPの場合には4台のPUによる並列処理を，マイクロ，ナノ2つのレベルで柔軟に制御しているために，大規模なμPをレジスタ転送形式のアセンブリ語で短期間に開発することは困難である．従って，2レベル制御と4台のPUによる並列処理を意識せずに，かつMUNAPのハードウェア機能を十分に活用できるシステム記述言語が要望される．以上の背景から，我々はMUNAP用のシステム記述言語：MSDL（MUNAP SYSTEM DESCRIPTION LANGUAGE）を設計した．

従来のシステム記述言語としては，PL/Iを基礎としてIBM360用に開発されたPL360，PDP11上で動くUNIXオペレーティングシステム用に設計されたCなどがある．Cでは豊富な演算子，近代的な制御構造，表現の簡潔性などを特徴としている．一方，MSDLの特徴として，言語仕様とハードウェアとの対応の明確化，言語仕様の簡潔性，タグ付きアーキテクチャの採用があげられる．特に，言語仕様においては，非数値処理用ハードウェアの活用を図るための演算子（シフト，交換），列関数などが導入されている．また，タグ付きアーキテクチャの採用により，種々のエラー検出，デバッグ支援などを効率よく実現できることを目的としている．

本稿ではMSDLの言語仕様，処理方式について述べ，さらに実験結果に基づいて処理系の評価を行う．

2. MUNAPの概要(1)(2)

MUNAPは，図1に示すように，4台のPUそれぞれにnPメモリ（NPM）を置き，これらを1つのμPメモリ（MPM）中のμPによって制御するという，2レベルμP制御方式を採用している．MPMとNPMは共に書き換え可能である．

μPによって直接制御される非数値処理ユニットは，シャフル交換網（SEN）とアドレスモディファイア（AM）である．SENはバス構成のかなめの位置にあって，4ビット毎の巡回シフト，鏡像交換，交換，及びブロードキャストなどの操作を行う．AMは8，16，32，64ビット単位でのアドレッシングを可能とし，しかもこれらを各8ビット長の主記憶（MM）バンク8個にインタリーブすることにより，連続した語の並列アクセスを可能としている．

nPによって制御される演算ユニットは，算術論理演算ユニット（ALU），フィールドの分割結合ユニット（DCU），及びビットのセットやテストを行うビット処理ユニット(BTU)である．このほかに，汎用レジスタ（REG），マイクロスタック（MSTK），スクラッチパッドメモリ（SPM），カウンタ（C），フラグレジスタ（FLR），ポートレジスタ（IPR，OPR）などが設けてある．

MUNAPにはコンソールプロセッサとして，ECLIPSE S/130が接続されており，MUNAPのMM，MPM，NPMなどへの読み書きはECLIPSEを介して行われる．ECLIPSE上には，μPの起動，ステップラン，ファシリティへの読み書きなどを制御するデバッガ，複数のμPの結合編集を行うローダ，及びμPを1ステップずつステップランさせて，種々の評価データを収集するエバリュエータが作成されている．

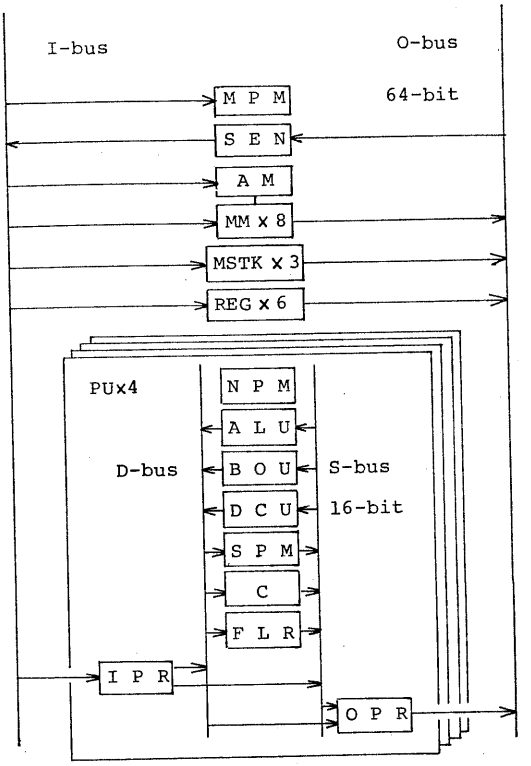

図1 MUNAPの構成

3. 言語仕様

3.1 設計方針

システム記述言語に対しては，アセンブリ語による記述と比較して，実行効率をさほど低下させることなく，記述能力，文書化能力，デバッグ支援機能などを向上させることが必要である．MSPLの設計方針は次の3点に要約される．

(1) ハードウェアの活用

MUNAPのハードウェア，特に，BTU，DCU，SENなどを十分に活用し，実行効率のよい処理系とすること．すなわち，BTU，DCUに対しては引関数を，SENに対しては シフト演算子，交換演算子などを導入する．

(2) 簡潔性

MSPLの演算子，データ長とMUNAPのマイクロアーキテクチャとの対応を明確にし，言語仕様を簡潔にする．また，処理系の作成において，MMとレジスタ間のデータ転送のように，複数のモジュールで共通に使用する機能を基本マクロ命令として定義する．この基本マクロ命令の導入により，処理系の作成および保守の効率化を図る．

(3) タグはきアーキテクチャ

MSPLの各データには26ビットのタグが付加され，データの属性，容量，オーバーフロ

ー／アンダーフローの有無，定義／参照の有無などの情報を保持している．このタグ付きアーキテクチャにより，MSDLの言語仕様とMUNAPのマイクロアーキテクチャ間のセマンティックギャップをできるだけ縮小させる．
すなわち，

(i) 種々のエラーを検出し，デバッグ支援を行う．
(ii) 記号表の参照回数を減小させ，処理速度の向上を図る．
(iii) オペランドの型変換に使用し，命令の個数を減小させる．

などの機能を実現する．

3.2 データ型，データ構造，演算子

MSDLにおけるデータ型を表1に，演算子を表2に示す．整数，実数，ビットのデータ長は，MUNAPの1語が16ビットで，4台のPUによる並列処理が可能であることに対応している．演算子型の変数には，算術，論理，シフト，交換の演算子を代入でき，演算を間接指定できる．また，データ構造として，2次元までの配列と構造体が用意されている．

演算子のうち，算術，論理はPUのハードウェア機能に，シフト，交換はSENのハードウェア機能に対応している．シフト演算子では63ビットまでの算術，論理，巡回シフトを，交換演算子ではミラー交換などを指定可能であり，これらはSENにより高速処理される．

3.3 文，列関数

プログラムは複数の外部手続きから成り，1レベルのブロック構造を有している．文に対する構文グラフを図2に示す．図2において，代入文から複合文までは，言語"C"の記法に準拠している．制御文として，IF文，WHILE文，FOR文，SWITCH文を設けている．手続き文における引数の受け渡しの方法として，値呼びと番地呼びがあり，手続きの再帰呼出しは許されない．MSDL独自の文として，フラグ変数へのセット／リセット，スタックへのプッシュ／ポップ，演算子型変数への代入，変数と変数との交換がある．

列関数を表3に示す．ビット列変数，文字列変数に対して，共に，切出し，位置検出，計数，結合の機能が用意されている．これらはBOUにおけるプライオリティエンコード，ビットカウント，DCUにおけるフィールドの分割，結合，スタック機能などに対応している．

表1 データ型

整数	SHORT(16),INTEGER(32),LONG(64)
実数	FLOAT(32),DOUBLE(64)
ビット	SBOOL(16),BOOL(32),LBOOL(64) LBOOL N (64+16K)
文字	CHAR N (8N)
フラグ	FLAG (1)
演算子	OPERATOR (8)

(注) 括弧内はビット長を表す

表2 演算子

算術	*,/,%,+,-
論理	!,&,|,^,!&,!|,!^ (注1)
シフト	SLL,SRL,SLA,SRA,SLC,SRC (注2)
交換	M8,M16,M32,M64,XX,** (注3)
関係	>,<,>=,<=,==,!=
結合	&&, ||

(注1) !(not),&(and),|(or),^(xor)

(注2) シフト数は1～63

(注3) Mi : iビットミラー交換
XX : 16ビットBADC交換
** : 16ビットDCBA交換

表3 列関数

ビット列関数	BSUBSTR(BIT,POS,N)	位置POSからのNビットの切出し
	PEC(BIT,{L/M},{0/1})	LSD/MSDからの0/1の位置検出
	BCT(BIT,{0/1})	0/1の計数
	BCON(B1,B2,..,BN)	B1,B2,..,BNの結合
文字列関数	CSUBSTR(CHAR,POS,N)	位置POSからのN文字の切出し
	INDEX(CHAR,{L/M},C)	LSD/MSDからのCの位置検出
	CCT(CHAR,C)	Cの計数
	CCON(C1,C2,..,CN)	C1,C2,..,CNの結合

4. 処理系

4.1 処理系の概要

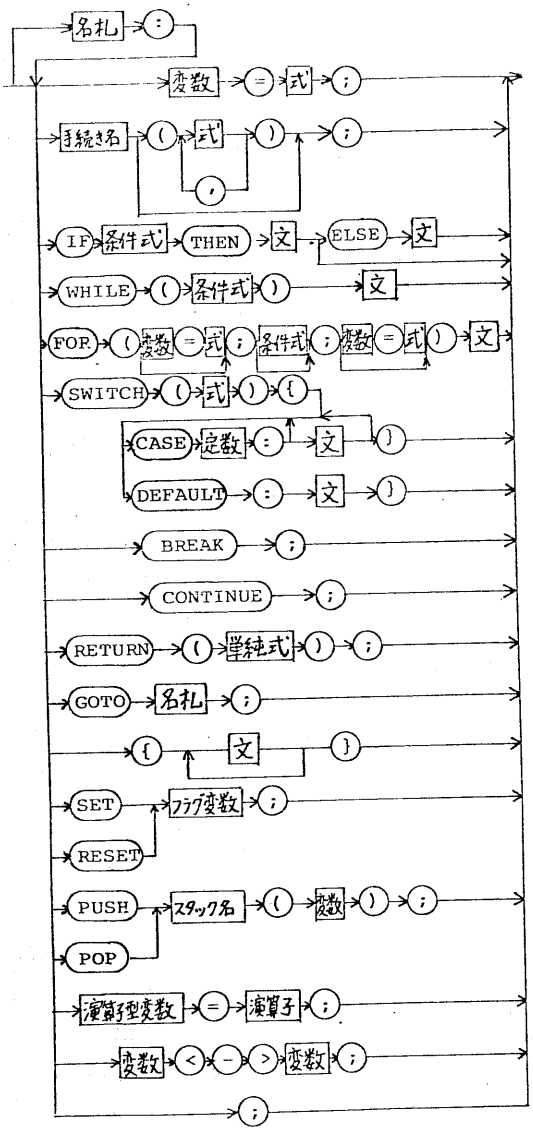

図2　文に対する構文図

MSBLで記述されたソースプログラムは，トランスレータインタプリタ方式により解釈実行される．トランスレータはソースプログラムを順次走査して，記号表，レコード表，配列表に必要な情報を格納すると共に，ユーザ定義の変数の初期化を行い，さらに中間言語（IL）系列を作成する．

インタプリタにおいて，各ILに対する処理を一様とするため，SPM内に中間言語エリア（ILA），アクセスエリア（ACA）が設けられている．インタプリタの基本サイクルは次のように実行される．

(i)　IL系列内のキーワードから次のキーワードの前までをMMからILAにフェッチする．

(ii)　IL内に記号表へのポインタが存在すれば，記号表を参照して変数の値をアクセスし，ACA内に格納する．さらに，記号表へのポインタをACAへのポインタに変更する．（これを新中間言語と呼ぶ）

(iii)　新中間言語内のキーワードを解読して，各処理ルーチンへ分岐し，解釈実行する．

トランスレータはECLIPSEのアセンブリ語により，インタプリタはMUNAPのMLアセンブリ語により記述されている．インタプリタでは，SPM4K語のうちの3K語とMM64K語のうちの30K語をユーザプログラム領域とし，残りを処理系で使用するシステム変数領域としている．

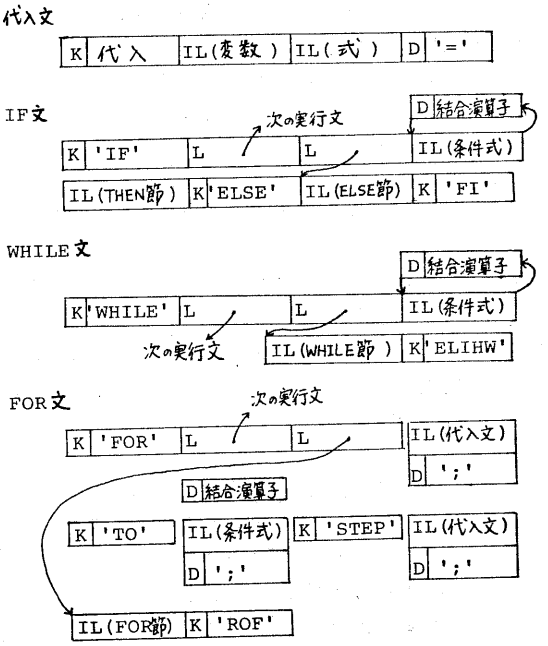

図3　代入文，IF文，WHILE文，FOR文の中間言語

4.2　中間言語仕様

代入文，IF文，WHILE文，FOR文に対するILを図3に示す．ILはKコード（キーワード），Dコード（デリミタ），Lコード

（ロケーション），Pコード（ポインタ），Sコード（引関数），及び他の非終端記号に対するILから構成されている(4)．ILの先頭のKコードにより，インタプリタにおける命令フェッチを容易に行うことができる．ILはMSDLの各文と1対1に対応しており，トランスレータの簡略化とインタプリタにおけるMUNAPのマイクロアーキテクチャの十分な活用が図られている．

4.3 基本マクロ命令

SPMへの読み／書き，SPMとMM間のデータ転送，実数型データの正規化，タグの作成，分解のように，インタプリタの複数のモジュールで共通に使用する機能を基本マクロ命令として定義した．基本マクロ命令を表4に示す．MUNAPでは4台のPUによる並列処理を，マイクロ，ナノプログラムのレベルで柔軟に制御しているために，ステップ数の大きいuPのデバッグは非常に困難である．今までのuPの作成経験から，50ステップ程度のuPであれば比較的容易に作成できることが確認されている．従って，この基本マクロ命令の導入により，インタプリタの各処理モジュールの大きさを50ステップ程度とし，インタプリタの作成，デバッグ，保守の効率化を図っている．

4.4 変数のアクセスと式の処理

IL内で変数は記号表へのポインタにより表現されている．単純変数，配列要素，構造体要素の場合には，記号表内でその変数のアドレスを求め，タグと値をACAに格納する．これらの変数が左辺に現われた場合には，さらに変数のアドレスを保持しておく．

配列名であれば，配列の先頭アドレス，タグ，上限を記号表と配列表から求め，ACAに格納する．配列名，構造体名の場合には，以降の解釈実行ルーチンで個々の要素にアクセスする．ACAに格納した後，変数に関する記号表へのポインタをACAへのポインタに変更する．

式はトランスレータにより逆ポーランド記法に変換され，インタプリタに渡される．インタプリタでは式を走査して，変数ならばMSTK1に積み，演算子ならば演算を行い，結果をMSTK1に積む．手続き呼出しの実引数が式ならば，式の値を評価し，結果を局所変数領域に格納する．さらに，仮引数のアドレス部に評価結果へのポインタを格納する．

表4 基本マクロ命令

機能		命令数
転送	システム変数 -> SPM (1語,4語)	2
	SPM -> システム変数 (1語,4語)	2
	MM -> SPM (1語,4語,任意語)	3
	SPM -> MM (1語,4語,任意語)	3
	SPM -> SPM (任意語)	1
正規化，符号検査，指数部と仮数部の分解		3
タグの作成，分解		2
タグフィールドのインクリメント		1
レジスタファイルの退避，復旧		2
列変数の先頭アドレス		1
同一データの位置の検出(1語,2語,4語)		3

4.5 演算子の処理

算術演算子のうち，加減乗除に関してはDOUBLEとLSN子に対するルーチンが，剰余に関してはLSN子に対するルーチンが用意されている．オペランドの型がこれらと一致しない場合，各演算において許される型であれば，型変換が行われる．演算実行後，実数型ルーチンでは結果の正規化が行われる．また，各演算子ルーチンでは，結果に対応した情報がタグとしては加えられる．

シフト演算子ルーチンでは，SENによる64ビットデータに対する4ビット毎の巡回シフトと，各ALUでのシフト機能が利用される．図4に64ビットデータに対するNビットの巡回シフトのための処理フローを示す．Nを4で割った商をD4，剰余をM4としたとき，まず，SENで4×D4ビットの巡回シフトを1マシンサイクルで実行する．次に，残りのM4ビットのシフトを，ALUにおけるシフト機能を用いて1ビットずつ行う．従って，SENを用いた場合，シフトのためのデータ移動の最大回数が高々4回，平均で2.5回であるのに対し，SENを使用しない場合には最大で63回，平均で31.5回であり，SENの効果が顕著である．算術シフト，論理シフトでは，巡回シフトを行

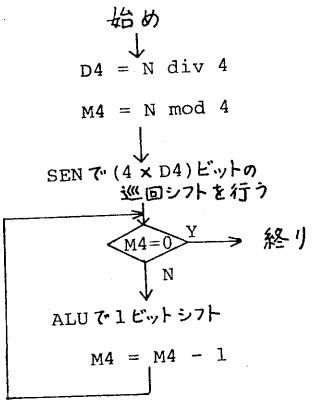

図4　64ビットデータに対する
Nビット巡回シフト

った後，符号ビット，上位/下位ビットの補正が行われる．

論理演算子と交換演算子は，各々，ALUによる論理演算機能とSENによる交換機能に1対1に対応しているので，それらの処理は非常に容易に行われる．

4.6　文の処理

代入文における式の評価結果の左辺への代入操作は，代入記号が逆ポーランド記法の中に演算子と同様に含まれているので，式の処理において実行される．

IF文，WHILE文，FOR文，SWITCH文などの制御文の処理は同様に行われるので，FOR文の処理について述べる．FOR文の記述例とそのIL系列を図5に示す．図5の文に対する処理は各Kコード毎に次のように実行される．

(i) K'FOR'：FOR文の次の実行文のアドレスとFOR節のアドレスを，各々，それらの値を格納するスタックFWSS，FWCSに積む．
FWSS←432，FWCS←424

(ii) K'代入'：Iに1を代入する．

(iii) K'TO'：条件式（I<10）とそれを格納するエリアFSCに格納する．

(iv) K'STEP'：代入文（I=I+1）とそれを格納するエリアFSAに格納する．

(v) K'代入'：文（A=A+B）を実行する．

```
FOR ( I=1 ; I<10 ; I=I+1 )
    A=A+B ;
```

400	K 'FOR'	L 432	L 424	K '代入'
408	P記1	P記10	D '='	K 'TO'
410	P記1	P記11	D '<'	K'STEP'
418	K '代入'	P記1	P記1	P記10
420	D '+'	D '='	K '代入'	P記2
428	P記2	P記3	D '+'	D '='
430	K 'ROF'			

(注) P記1, P記2, P記3, P記10, P記11は，各々，記号表内でI, A, B, 1, 10へのポインタを示す

図5　FOR文の記述例とそのIL系列

(vi) K'ROF'：FSAの内容をILAに格納し，I=I+1を実行する．FSCの内容をILAに格納し，I<10を判定する．結果が真ならば，中間言語カウンタにFWCSの値を代入し，文（A=A+B）を実行する．偽ならば，中間言語カウンタにFWSSの値を代入し，FOR文から抜けだす．

手続き文では最初に引数の受け渡しの処理を行う．仮引数が着地呼びか値呼びかを記号表で調べ，着地呼びならば仮引数のアドレス部を実引数の先頭アドレスで置換える．値呼びならば実引数の値を一旦，局所変数領域に格納した上で，仮引数のアドレス部をそこへのポインタで置換える．次に，中間言語カウンタの値を退避した上で，手続きの先頭の文に制御を移す．

4.7　列関数の処理

ビット列関数のうち，BSUBSTRとBCONは共に，論理シフト，及びBCUによる結合機能を用いて処理される．BSUBSTRに関して，64ビットの変数BITの位置POSからのNビットを取出し，結果をANSに格納するには次のように行う．

(i) BITをRR(3)に格納し，（POS−1）ビット左論理シフトを行う．

(ii) シフト数Nを16で割った商をPUNO，剰余をPLとする．

(iii) RR(3)をBCUスタック（BCUSTK）に積む．

(iv) i = PUNOとしたとき，PUiにおいて，DCUのTKのスタックトップの左から(PL+1)ビット目で0と結合し，ANSに格納する．

BIT = 00AABBBB CCCC DDEE，PUNO = 3，N = 52の場合の処理を図6に示す．PEC，BCTは各々，BSUのプライオリティエンコード，ビットカウント機能により，比較的容易に処理される．

文字列関数のうち，INDEX，CCTでは先にSENによるブロードキャスト，DCUによるフィールドの分割が有効である．例えば，CCTにおいて8文字の変数A内に，ある文字"C"がいくつ存在するかを求める場合を考えてみる．まず，"C"をブロードキャストして8文字の変数Bに格納する．次に，右PUでDCUを用いてA，Bの上位バイトを抽出した後，Aの上位バイトからBの上位バイトを減じ，0のものの合計を求める．同様に，右PUでAの下位バイトからBの下位バイトを減じ，0のものの合計を求める．また，CSUBSTR，CCONに関しては，MMのバイトアドレシング機能により，容易に処理される．

4.8 エラー処理

プログラム内で使用される各データには，図7に示すように26ビットのタグが付加されており，種々のエラー検出，デバッグ支援機能などを効率よく実現することができる．このタグ付きアーキテクチャにより，MSDLの言語仕様とMUNAPのマイクロアーキテクチャ間のセマンティックギャップをできるだけ縮小し，信頼性が高く，使いやすいアーキテクチャをユーザに提供することを目的としている．

図8にインタプリタの基本サイクルとエラー検出の適用箇所を示す．エラー検出は，モジュール入口でのエラー検出とモジュール内部でのエラー検出に大別される．ここで，モジュールとはKコードを解釈実行するルーチンを意味するので，モジュール入口でのエラー検出は，中間言語のフェッチ，オペランドアクセスの過程で行われる．

(1) モジュール入口でのエラー検出

(i) システム変数エラー：中間言語カウンタ，アクセスエリアアドレスなどのように，処理系で使用するシステム変数の値が対応する範囲内にない．

(ii) インターフェースエラー：手続き呼出しの場合に，実引数，仮引数の個数，属性，順序が一致しない．

(iii) アクセスエラー

(a) MM，SPMアドレスエラー：オペランドのアドレスがユーザ変数領域内にない．

(b) 未定義エラー：オペランドの値が定義されていない．

(c) 配列エラー：配列の添字が範囲内にない．

(2) モジュール内部でのエラー検出

各モジュール内において，処理に対応した工

(i) RF(3) = 00AA BBBB CCCC DDEE

RF(3) = AABB BBCC CCDD EE00

(ii) PUNO = 3　PL = 4

(iii) DCUSTK = | AABB | BBCC | CCDD | EE00 |

PU0 PU1 PU2 PU3

(iv) ANS = | AABB | BBCC | CCDD | E000 |

図6 BSUBSTRの処理例

4	8	2	2	10
属性	容量	オーバーフロー アンダーフロー	定義 参照	参照回数

図7 タグ

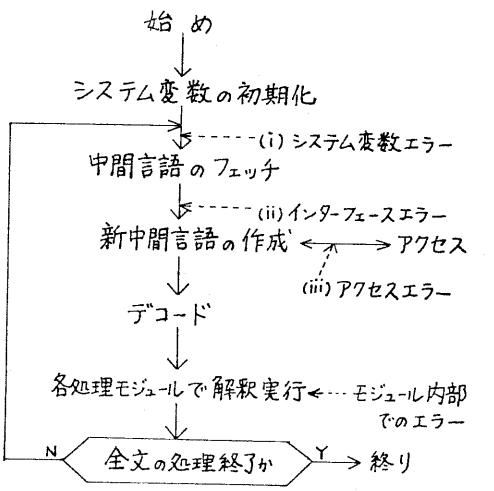

図8 インタプリタの基本サイクルとエラー検出

ラー検出を行う．例えば，四則演算ルーチンにおけるオーバーフロー，アンダーフロー，除算ルーチンにおける0による除算，列関数において抽出すべきビット数が範囲外，などがあげられる．

5. 処理系の評価

インタプリタの評価を行うために，基本マクロ命令，全体の制御，演算子，式，文，列関数に関する静的評価データと，演算子に関する動的評価データを収集した．各々の評価データを表5，表6に示す．表5，表6のMはマイクロ命令だけが実行される命令数を表し，無条件分岐，サブルーチン呼出し，復帰などが対応する．NはナノI命令だけが実行される命令数を表し，ALU，BTU，DCUでの演算やレジスタへの定数生成が対応する．MNはマイクロ命令からナノI命令を起動する命令で，Nで実行される命令の他に，PU内での演算結果による条件分岐，及びMM，SPM間のデータ転送などが対応する．MACROは基本マクロ命令の呼出し回数を表す．

(1) 静的評価

マイクロ命令数（μI）は2～66.75，平均で28.64，ナノI命令数（nI）は4～46.25，平均で17.89である．M(b)，N(c)，MN(d)，MACRO(e)の比率は各々，平均で26%，22%，47%，5%である．また，MNのうち，18%はPUでの演算結果による条件分岐命令である．すなわち，ME全体のうち，26%が無条件分岐，サブルーチン呼出し，復帰などであり，9%がPU内での演算結果による条件分岐，5%が基本マクロ命令の呼出し，そして60%がPU内の演算か，PU内のレジスタとマイクロレベルのファシリティ（MM，MSTKなど）の転送などである．表5のDECODEにおいてMの比率が大きいのは，Kコードの値を解読して，他のモジュールをサブルーチン呼出ししているからである．

平均PU使用率（R）は1.01～4.00，平均で1.98である．PU使用率は全体の制御ではばらつきが大きく，演算子，

表5　静的評価データ

モジュール名		(a) モジュール数	(b) M	(c) N	(d) MN	(e) MACRO	(f) μI数	(g) nI数	(h) APU
基本マクロ命令		23	5.74	4.78	9.43	0.52	15.70	10.74	2.13
全体の制御	MAIN	1	5	2	3	0	8	4	1.80
	INIT	1	1	20	7	0	8	26	3.77
	FETCH	1	2	13	21	1	24	25	2.22
	NEWCODE	1	10	19	41	0	51	31	2.05
	ACCESS	2	12	9.50	28.50	2.50	43	28	1.63
	DECODE	3	39.33	1.33	3	0	42.33	4.33	1.07
演算子	算術	14	6.07	8.93	15.66	2.07	21.57	13.93	2.09
	論理	1	2	5	4	2	8	8	2.55
	シフト	4	14.50	23.25	50.75	3.50	68.75	46.25	1.74
	交換	1	10	11	15	3	28	19	2.23
式		3	32	6.33	18.67	1	51.67	16.67	1.71
文	代入	3	8.67	7.33	20.67	5.33	34.67	28.00	1.77
	手続き	2	16	14.50	25.50	1.50	43	26	1.69
	IF	3	2	1.33	3.33	0.66	6	4.33	1.50
	GOTO	1	6	16	6	1	13	12	1.58
	FOR	4	6.25	12	10.50	0.50	17.25	19	2.02
列関数	ビット列	4	12	10	43.75	8.25	64	34.50	1.90
	文字列	4	9.75	10	37.25	2.25	49.25	31	2.21
平均		—	9.54	8.41	17.73	1.78	28.64	17.89	1.98

(注) M：マイクロ命令だけが実行される命令数　　N：ナノ命令だけが実行される命令数

MN：マイクロ命令からナノ命令を起動する命令数

MACRO：基本マクロ命令の呼出し回数　　APU：平均PU使用率

μI数=M+MN+MACRO　　nI数<=N+MN　（文献(3)参照）

表6 動的評価データ

モジュール名		(i) M	(j) N	(k) MN	(l) 総ステップ数	(m) 処理内容
整数	ADD	51	29	61	141	LONG,INTEGERの加減乗除
	SUB	50	30	58	138	
	MULT	70	128	100	298	
	DIV	57	69	366	492	
実数	ADD	77	77	123	277	DOUBLE,FLOATの加減乗除
	SUB	76	76	120	272	
	MULT	77	122	127	326	
	DIV	66	121	613	800	
シフト	SC	18	16	58	92	17ビットシフト
	SL	19	18	59	96	
	SA	32	48	83	163	
MOD		54	71	364	489	LONG,INTEGER, LBOOLの論理積
LOGICAL		39	20	32	91	
CHANGE		37	31	59	127	32ビットミラー変換

引用数のようにPU内での演算の比率の大きいモジュールではほぼ2前後の値を示している.

インタプリタ全体の大きさは, エラー処理ルーチンを除いて, モジュール数約100, MPM約3.2K語, NPM約1.9K語である. 従って, MUNAPの制御記憶容量(MPM, NPM共に4K語)内に納まる規模となっている. また, MSDLで円周率πを100桁まで求めるプログラムをMadlenの方法で記述した結果, EL体到に約630バイト, 記号表に約820バイト, 配列表に40バイトを要した.

(2) 動的評価

算術演算子ルーチンでは, 演算前にタグ内の属性フィールドを調べて左右オペランドの型を検査し, 必要ならば型変換を行っている. また, 演算後には結果に対応したタグを生成し, タグと結果を指定された場所に格納する. さらに, 実数型演算では, 演算前の仮数部と指数部の分解, 演算後の分解が必要である.

表6における総ステップ数(l)のうち, 上に述べた演算前後の処理を除いた実質的演算に要するステップ数は, 整数型加減乗除に対して, 各2, 10, 7, 167, 361, 実数型加減乗除に対しては, 各2, 70, 65, 119, 593である. 実質的演算以外に要するステップ数は, 整数型, 実数型に対して, 各2 131, 207であり, それらの処理の比率を表7に示す. 型検査と型変換の合計に要する比率が整数型で約75%, 実数型で約43%となっている. これらのオーバーヘッドは, 各モジュールで種々のデータ型の演算を許しているために生じたものである. 例えば, 整数型加減乗除では, データ型として整数型, ビット型, 文字型が許可されており, いずれの型もLONGに変換された後, 演算が行われる.

巡回, 論理, 算術の各シフトルーチンは, 16, 32, 64ビットデータの左右のシフトが可能で, また, 他ルーチンからも呼び出せるよう作成されている. 例えば, BSUBSTRにおいて, フィールドの切出しを行う先頭ビットがPUのMSBに来るまで, 与えられたデータを左論理シフトしている. すなわち, シフトルーチンでは, シフト演算前にシフトデータ, シフト数, データ長の設定と共に, 共用化のための処理が必要である. また, 演算後にはタグの作成と結果の格納が行われる. SCの総ステップ数のうち, 演算前に38ステップ, 演算後に45ステップ実行され, 実質的なシフトは9ステップで完了している. 従って, シフト演算において, SENが有効に利用されていることがわかる.

引用数では, 64ビット変数の9ビット目からの52ビットを切出す処理(BSUBSTR)に225ステップを要し, 平均PU使用率は2.21であった. また, 20文字変数の3番目からの10文字を切出す処理(CSUBSTR)に124ステップを要し, 平均PU使用率は1.89であった.

表6の演算子ルーチン全体において, M, N, MNの比率は平均で19%, 23%, 58%, 平均PU使用率は1.97である. 各ルーチンにおける基本マクロ命令の呼出し回数は3~8回, 平均で5.29回である. また, 式A*B+C(A, B, CはLONG型)の評価を行う

表7 演算以外の処理の比率

	型検査	型変換	仮数部と指数部の分離	正規化	タグ作成	結果格納	ステップ数合計
整数型	28.24%	46.56%	0%	0%	14.50%	10.69%	131
実数型	18.84%	24.15%	22.22%	18.36%	9.66%	6.76%	207

のに810ステップを要し，平均PU使用率は2.01であった．

(3) 基本マクロ命令の効果

基本マクロ命令23個のうち，7個は他の基本マクロ命令の呼出しを行っている．基本マクロ命令のマイクロ命令数(ナ)，ナノ命令数(ヌ)は各々，平均で15.70，10.74であり，典型的なデータに対する動的ステップ数は7~70ステップ，平均で約30ステップ程度である．

基本マクロ命令以外の各処理モジュールでは，動的に平均で5.29回基本マクロ命令を呼び出しており，μPの静的ステップ数((b)+(c)+(d)+(e))は平均で44.82である．この結果，モジュール当り約50ステップという当初の目標を達成することができた．また，基本マクロ命令の導入により，インタプリタの各モジュールのデバッグを階層的に行うこと，及び修正の可能性が生じても比較的容易に対処することが可能となっている．なお，基本マクロ命令自体の静的ステップ数の平均は20.47であり，この程度の規模のμPの作成は非常に容易であった．

6. むすび

本稿ではMUNAPのシステム記述言語：MSDLの言語仕様，処理方式について述べ，インタプリタの評価データを示した．

MUNAPの非数値処理用ハードウェアの活用を図るために導入されたシフト演算子，列関数などの処理において，SEN，BOU，DCUなどが有効であることが確認された．また，演算子ルーチンの動的評価により，タグ付きアーキテクチャに伴う操作に関して，詳細なデータを得ることができた．なお，インタプリタの容量はMPM 3.2K語，NPM 1.9K語であり，MUNAPの制御記憶容量内に納まる程度の規模となっている．

今後の課題として，(i) エラー検出，デバッグ支援機能の強化，(ii) 種々のプログラムの記述によるタグ付きアーキテクチャの有効性の評価が必要であろう．

謝辞 貴重な御意見を頂いた京都大学工学部富田眞治助教授，ならびに当学科研究室の各位に厚く感謝致します．

文献

(1)馬場，石川，奥田："2レベルマイクロプログラム制御計算機MUNAPのアーキテクチャ"，信学論(D)，J64-D，pp.518-525(昭56-06)．

(2)馬場，石川，奥田："2レベルマイクロプログラム制御計算機MUNAPにおける非数値処理"，信学論(D)，J64-D，pp.526-533(昭56-06)．

(3)馬場，橋本，山崎，奥田："2レベルマイクロプログラム制御計算機MUNAPにおけるマイクロプログラムの記述とその処理"，信学論(D)，J65-D，pp.1265-1272(昭57-10)．

(4)新実，栄山，富田，萩原："ファームウェアによる拡張言語の処理システム：FELPS"，信学技報，EC79-79(1980-2)．

(5)北村，栄山，富田，萩原："QA-1のファームウェアによる会話型手続き向き言語の処理システム"，信学技報，EC80-10(1980-5)．

(6)山崎，橋本，橋本，馬場，奥田："2レベルマイクロプログラム制御計算機MUNAPのシステム記述言語"，情報処理学会第24回全大，3D-2，p.135(昭57)．