マイクロプログラム記述の汎用化の一手法

山田昭彦　佐々木徹　高橋萬年　高橋悦男　尾藤青詩茂
青山正義　林孝雄　（日本電気 株式会社）

1. まえがき

マイクロプログラム（以下μP）制御方式を採用した各種装置開発の普及には、目を見張るものがある。これはμP制御方式の特徴である①設計対象ハードウエア（以下HW）の簡単化が期待できる②柔軟性／拡張性のある設計ができる③HWの詳細設計とμPの設計とが並行して行えるので設計期間が短縮できる，等によるものと思われる。

ビットスライスの各プロセッサはこれらの動きを推進して来ており，LSIの進歩によってマイクロコンピュータ自身にもμPを組込む動きが出て来ている。

また、装置の小型化／汎用化／機能の拡張性を目的として，マイクロコンピュータを内蔵させた各種装置が大量に出廻って来ており，それらの為のファームウエア（以下FW）の開発も膨大なものになりつつある。

更に，システムの性能の向上を目指して、従来ソフトウエア（以下SW）で処理していた部分をFW化する事も，かなり一般的になって来ている。

以上のように，HWもSWも，各々の機能をFWに移す傾向にあり，FWの作成量は増加の一途をたどっている。

FWについては，その適用範囲の拡大による作成量の問題と同時に、特にHWに密着した形のμPに於ては生産性が低く，作成後の保守性が悪いという問題がある。

アプリケーションソフトウエア	高級言語で書く，アプリケーションプログラム。	15～20行/人日
マイクロコンピュータ用ファームウエア	固定命令セットがある。	8～12行/人日
ビットスライスプロセッサのマイクロプログラム	命令セットはバラバラ。	2～5行/人日

図1. プログラムの生産性

μPの作成に汎用コンパイラが利用できれば，生産性も上り，保守性も良くなると思われる。しかし、マイクロコンピュータを内蔵した，コンピュータ周連機器用に作成するFWでさえ，従来のHWロジックに置換るSWを作成するという点で，汎用コンパイラの適用がむづかしい。

μPコンパイラの開発もいくつか試みられているが，まだ一般的にはなっていない。これは，出力されるオブジェクトμPに対する性能要求（動的／静的なμPステップ数）がきびしいという理由による——μPはHWに密着しており，タイミングがクリティカルであるという特徴を持っており，各マイクロコマンドのマイクロ命令への割付や、各マイクロ命令の各アドレスへの割付等に於てきめの細かい処理が要求されるので，性能要求を満足させるμPコンパイラの実現はむづかしい。

従って，現状では アセンブラを主体としてμPが作成されている。以前は各々専用のμPアセンブラを使用していたが、近頃はアセンブラ生成方式やテーブル参照方式等のアセンブラ汎用化手法を用いて汎用μPアセンブラを開発し，その

アセンブラを使用してμPを作成する場合が多い。

特に、テーブル参照方式の汎用アセンブラを使うと、種々のμPが一種類のアセンブラでアセンブルできるようになるので、ツールの開発面や運用面だけでなく、μPの記述面でも改善が期待できる。しかし反面、個々のμP記述に対する制約が出て来たり、細かいチェック等のサポートが受けられなくなっている。また、記述レベルがアセンブラレベルである為の生産性、保守性、信頼性等の問題は残ったままになっていた。

一方、大量に作成されるμPを見ると、内容的には、LSI化によるコストダウンや機能向上を目指した改造設計等もかなりあり、一度作成した旧機種μPを新機種μPとして有効に活用したいという要求もある。

そこで、汎用アセンブラをベースにして、その前段に汎用プリプロセッサを設ける事によって、超大型コンピュータからマイクロコンピュータ迄の各種μPの記述に対して前述の問題点等を解決し、より柔軟性を持たせてμPの設計を効率的に行えるようにしたので、汎用プリプロセッサの手法及び適用例を中心に、μP記述の汎用化の手法について報告する。

図2に、汎用プリプロセッサと汎用アセンブラとを含むMDS（MICROPROGRAMING DESIGN-SUPPORT SYSTEM）の全体図を示す。

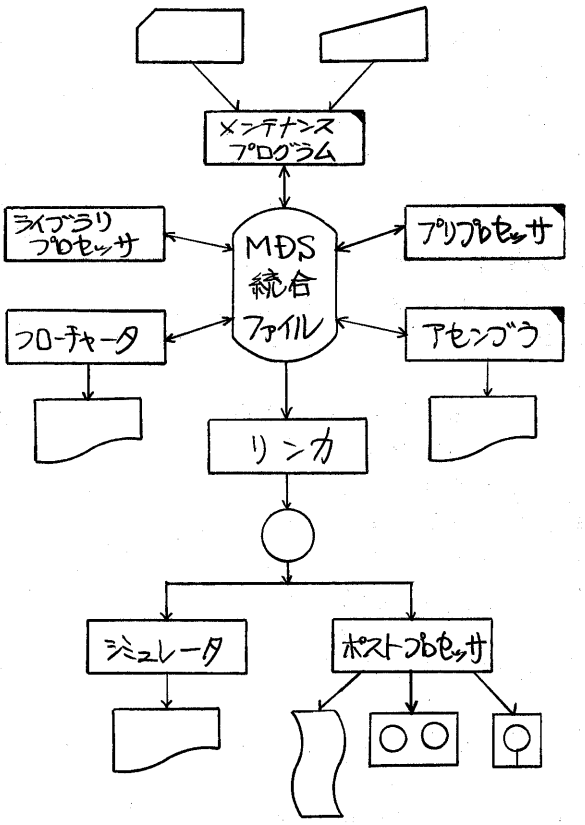

図2　MDS システム構成図

□印のプログラムは
TSSでの運用が可能。

（注1）アセンブラ生成方式．
アセンブラ生成情報とアセンブラサブルーチン群とから、アセンブラジェネレータが各々専用のアセンブラを作成する方式．

（注2）テーブル参照方式．
各々のアセンブラの命令仕様がテーブル形式に変換され、そのテーブルを参照して、汎用アセンブラがアセンブルをする方式．

2. 汎用プリプロセッサの導入.

2.1 開発目的.

(1)高級言語風記述のサポート.

各使用者が設定した様々な高級言語風記述に対して、汎用アセンブラに渡す為の前処理を行う.

高級言語風記述とは、コンパイラレベルの記述言語が持つ書き易さ，読み易さは残すが，MPを記述する人は対象HWを意識して最適なコーディングを行うような記述言語を言う。即ち，高級言語風記述では，メモリの容量やオブジェクトの性能についてはアセンブラレベルの記述言語と同じだが、生産性／保守性／信頼性についてはコンパイラレベルの記述言語並みである。

(2)マクロ機能のサポート.

特に垂直型MPの記述の簡単化を図る為に、複数のワード内にまたがるマクロ（以下縦のマクロと呼ぶ）をサポートする。

水平型のMPで用いられるような1ワード内でのマイクロコマンドの集合（以降横のマクロと呼ぶ）は、MDSでは汎用アセンブラでサポートしている。

(3)MPチェック.

信頼性のあるMPを作成する為に、マイクロコマンドの組合せチェックや静的な状態でのタイミングのチェック等を行う。また、MP記述をある語をキーにして一括修正したいとか，MPをチューンアップする時に特定の条件を満たすワードを抽出したいとかの要求をサポートする。

(4)MPの移行のサポート.

前述のように汎用MPコンパイラの開発／利用は現段階ではむずかしいので，アセンブラレベルないしは高級言語風記述でのMPの移行が行えるようにする。

これらの多くの開発目的を実現する為に、汎用プリプロセッサには，使用者が手続的に処理内容を定義したデータを解釈し実行する，インタプリタ方式を採用する事にした。このような方式をとった事によって，MP記述に一層柔軟性を持たせる事ができ，且つきめの細かいチェック等ができるようになった。

2.2 汎用プリプロセッサ概要.

図3に、汎用プリプロセッサの全体図を示す。

汎用プリプロセッサに対して使用者が指定するものは、処理定義部と実行指示部とに分けられる。処理定義部では，標準的に用意されている処理命令と作業用エリアを使用して，使用者が実行したい

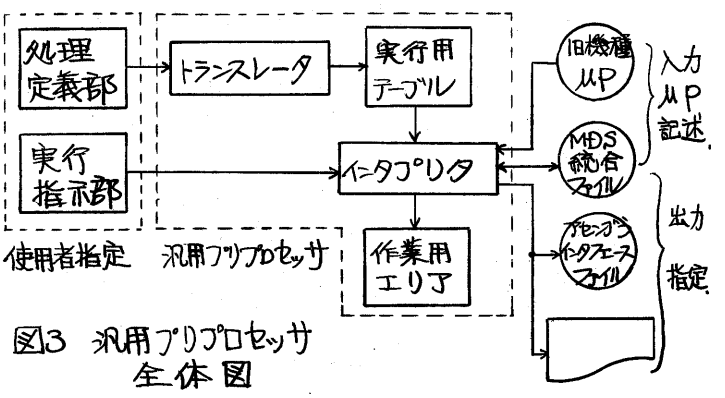

図3 汎用プリプロセッサ全体図

機能を記述する。また実行指示部では、処理の対象となるべきμP記述の指定や，処理結果の出力方法の指定等を行う。

トランスレータは、処理定義部をチェックし翻訳して，実行可能な形式に変換し実行用テーブルに格納する。

インタプリタは、実行指示部の指定に基づいて読み込んだμP記述を，実行用テーブルの内容を解釈しながら処理し，必要な出力を得る。

汎用プリプロセッサの補助的な機能として，使用者指定の一部又は全部をMDS統合ファイルから抽出する機能，実行用テーブルをMDS統合ファイルにセーブしておく機能，実行用テーブルをファイルからリストする機能等が用意されている。

2、3　処理の定義

処理定義部で処理内容を定義する為の記述言語は、以下のような特徴を持っている。

(1)全体として記述量が少なくなるように，ファイルの形式やμPの記述規則等を意識した，システムに密着した記述言語である。

(2)多種多様なμP記述を様々に処理できるようにする為に手続き型の言語になっており，文法規則も極めて簡単である。

(3)マクロ展開を行いながら，ストリング処理命令，テーブル操作命令等言語処理に適した命令で処理定義ができる。

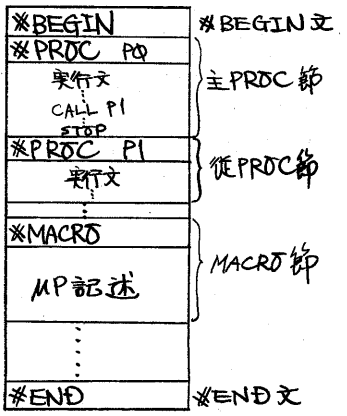

図4 処理定義部

処理定義部は、宣言文と実行文及びマクロ定義から構成される。

宣言文には、μPソースステートメント内の語を区切るデリミッタ文字や動作環境を定義する為の※BEGIN文，一連の手続きの始まりを示し以下に実行文が続く※PROC文，1個のマクロ定義の始まりを示し以下にマクロ定義の為のμP記述が続く※MACRO文，及び処理定義部の終了を示す※END文とがある。

実行文は，ラベル名とスラッシュで囲まれた条件式と任意の数の処理命令とで構成され，セミコロンで終了する。

[ラベル名:][/条件式/]処理命令[,処理命令…][ELSE 処理命令[,処理命令…]];

(1)条件式——単純条件の比較演算子としては＜，＞，＝があり，これらを＋（OR），＊(AND)，'(NOT)で組合せて複合条件とすることができる。

(2)処理命令

①入出力命令——READ, WRITE, REWIND, PRINT, MESSAGE.
②移送命令——変数:=変数または定数.
③制御命令——JUMP, CALL, RETURN, STOP.
④演算命令——ADD, SUB
⑤ストリング処理命令——GET, SET, SCAN, PUSH DOWN, POP UP, REPLACE,

⑥テーブル操作命令 ―― SET UP, STORE, SEARCH.

(3)変数と定数

変数 ―― バッファ名, スイッチ名, ポインタ名, スタック名.

定数 ―― 文字定数, 数字定数, 表意定数.

3. 汎用アセンブラ.

汎用アセンブラについては種々報告されているので, ここではMDSの汎用アセンブラの特徴についてのみ述べる。

(1)テーブル参照方式の汎用アセンブラである。

(2)モジュール化プログラミングを基本としている。

(3)μPの記述形式は、全ての使用者に共通な固定カラム形式, 使用者が記述形式を自由に定義した準固定カラム形式及び完全な自由カラム形式が可能である。

(4)オブジェクトμPの1ワードの最大ビット長は216ビットである。
また、1μP内で各マイクロ命令毎にワードのビット長を可変にできる。

(5)未使用のワードやワード内の未使用フィールドに対して任意のビットパターンを設定できる。

(6)ワード内の各フィールドの重複使用が可能である。

(7)パラメータを伴うワード内マクロ(横のマクロ)が使用できる。

(8)机上デバッグの為の豊富なリスト類(行先/来先表示, ENTRY/EXITリスト, クロスリファレンスリスト, アドレスマップ等)を出力する。

(9)μPソースステートメントとアセンブル結果, マイクロ命令の仕様定義とその解析結果のテーブル, アドレス割付情報等をMDS統合ファイルで一元的に管理している。

4. 入力記述例.

4.1 高級言語風記述の例.

従来は汎用アセンブラのみでμPを作成していたが, 今回汎用プリプロセッサと汎用アセンブラを組合せて使用する事とし, 高級言語風記述HMIP (HIGH LEVEL MICRO PROGRAMMING)を設定した。対象μPは1ワード32ビット構成の水平型μPである。

(1)記述規則.

高級言語風記述は、以下の規則に基づいて記述する。

①第1カラムによって各ステートメントの意味が異なる。

":" ―― 汎用プリプロセッサの縦のマクロ参照.

"*" ―― 汎用プリプロセッサの制御文または汎用アセンブラのコメント文.
- *MEASURE-CLOCK ---- クロック数計測の指示.
- *MSEQ-RESET ------ メモリ制御チェック用内部状態のリセット.
- *MSEQ-SET -------- メモリ制御チェック用内部状態のセット.

"※" ―― 汎用アセンブラのアセンブラ制御文.

" " ―― 汎用プリプロセッサで処理されるμPソースステートメント.

②1マイクロ命令は複数機能を含んでいる。HMIPではこれを1行に1機能とし, 複数行で記述する。1マイクロ命令の記述の先頭の行には、タイプ

```
  D             SCR=0000#
  A             IR=HW5
  B             CALL  TIADR
  B             CALL  TIPEA
  B             IF ZERO GOTO TBIERR
  B             CALL  TIMSW
  A             IR=R0
  A             LW0=ALL0
                SET STF
  A             BR=R1
  D     MPX010:SCR=1
  B             GOTO MPX01+(IR1,IR0)*2
 #MORG 8
  B     MPX01: GOTO ¥MPX03
  ZZ                        DWA     0#
  B             GOTO MPX02
  ZZ                        DWA     0#
  A             BR=SFT(BR,LEFT,N)
  B             GOTO MPX02
  A             BR=SFT(BR,LEFT,N)
  A             LW0=LW0+R1
  A     MPX02: LW0=BR+LW0
  B     MPX03: IF ZERO GOTO MPX04
  A             BR=LW0
  A             BR=R2+BR
  A             R1=BR
  A             *=ALL0+1
                SET STF
                RETURN
 *
```

図5 高級言語風記述HMIPの例.

```
#BEGIN     YYY Y      /=(,:
#PROC      MAIN                                  HMIP51 SOURCE VER.15
      {
  MMM: ADD RCN TO JP1,A¥76-4:=JP1;
     /SW0="1"/ CALL GSEQCK, JUMP PROCED;
     /RCN>4/ CALL DTAPSH, MESS ">>OVER 4 CARD ERR",
     ADD 1 TO HP1,
     JUMP PROCED;
     /A¥1="#"/ CALL GSEQCK, CALL DTAPSH, PRINT A, A¥75:="<",
       A¥80:=">", WRITE A;
     /A¥1="*"/ JUMP ASTE;
     /A¥1=":"/ CALL GSEQCK, CALL DTAPSH, PRINT A;
     /A¥1=" "/ CALL PTYPE;
     READ A;
     JUMP MMM;
   ASTE: CALL GSEQCK, CALL DTAPSH, PRINT A;
      {/A¥1-15="*MSEQ-SET"/ SW5:="3",  I¥10-2:=" ";

      /(SW5="1")'/ MESS ">>SEQUENCE NOT COMPLETE", ADD 1 TO HP1;
       CALL GCLCNT, RCN:=1, PRINT A;
******** TYPE SELECT ********
#PROC      PTYPE
      AP1:=1,FP1:=1,GP1:=1,AP2:=73,FP2:=73,GP2:=73;
       {
            /((A¥AP1<"A")+
              (A¥AP1>"Z"))*
             ((A¥AP1<"0")+
              (A¥AP1>"9"))/     RETURN ;
                JUMP LOOPD ;
#END
```

図6 高級言語風記述HMIPの処理定義データ

名を書く。タイプ名としては，演算用命令タイプ，分岐用命令タイプ，定数用タイプ等がある。タイプ名"ZZ"は汎用アセンブラへの直接入力記述である。

③μPソースステートメントとしては，代入文，IF文，GOTO文，CALL文，メモリ制御文，RETURN文，フリップフロップのセット／リセット文等がある。

(2)チェック機能．

汎用プリプロセッサでの処理は，μP記述の変換の他に，以下のチェック等を行っている。

①高級言語風記述での文法チェック．
②マイクロコマンドの組合せ禁止チェック．
③メモリ制御に関するタイミングチェック．
④クロック数のカウント．

(3)データの量等．

高級言語風記述に対して，上記の(2)で述べたチェック機能を汎用プリプロセッサで処理させる為の処理定義部のデータ量は，約1500枚で，データ作成工数は約1人月，汎用プリプロセッサの実行時間は，記述1Kステップ当り，約1.5分であった。

この高級言語風記述の設定によって，従来のアセンブラレベルでの記述に比較して，μPの作成工数が30%減少し，信頼性，保守性も大幅に向上した。

図5に高級言語風記述HMIPの記述例を，図6にはHMIPの変換，チェック等を汎用プリプロセッサに行わせる為の処理定義部のデータを示す。

4．2　2レベルμPの記述例．

2レベルμPを行う場合で，下位のマイクロ命令がシーケンス制御機能を持たず，上位のマイクロ命令の中でアクセスされるべきアドレスが指し示されるような場合には，二種類のμPの記述およびその作成に工夫が必要である。

下位のマイクロ命令の各マイクロコマンド（以下サブコマンドと呼ぶ）を，下位のμPとしてアセンブルしオブジェクトを得る。一方，上位のμPを記述する場合に，上位のマイクロ命令の各マイクロコマンド（以下コマンドと略す）を記述し，アクセスすべき下位マイクロ命令のアドレスは，下位μPの中で必要なサブコマンドをリスト上から見つけ出して，そのアドレスを上位マイクロ命令のコマンドと一緒に記述する，という方法では，アドレスの選択間違いや記述ミスが多くなる事が予想される。

そこで上位のμPの記述に於ては，必要なコマンドとサブコマンドとを記述し，コマンドについてはそのまま汎用アセンブラに渡すが，サブコマンドについては下位のμPの中でサブコマンドを比較し，対応する下位マイクロ命令のアドレ

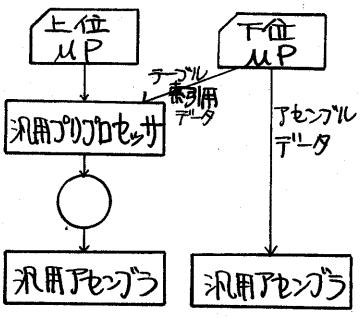

図7　2レベルμPの処理例．

スを索引して汎用アセンブラに渡す処理を汎用プリプロセッサを用いて行う。
この機能によって，μP設計者はコマンド／サブコマンドの区別を意識する事なく，必要な動作を記述すれば誤りのないμPが効率良く作成できる。
このアドレス選択機能の他に，コマンド組合せ禁止チェック及びマイクロ命令のタイプ名決定等を汎用プリプロセッサで行っているが，処理定義部のデータは500枚前後でできている。

5．3 マイクロコンピュータ用FWの記述例．

マイクロコンピュータ用FWについては，通常のマイクロコンピュータ用汎用アセンブラと同様に記述できる。図8は8080（μCOM80，インテル8080）を汎用アセンブラへ直接入力する場合の記述例である。

```
216  001A0 C9                          RET
217                          *         SUBROUTINE
218  001A1 21EF83          0 1A1#RGDSP LXI  H,ADRES + 1
219  001A4 11F483                      LXI  D,DISP
220  001A7 0604                        MVI  B,4
221  001A9 7E                          MOV  A,M
222  001AA 12                          STAX D
223  001AB 2B                          DCX  H
224  001AC 13                          INX  D
225  001AD 05                          DCR  B
226  001AE C2A901                      JNZ  * - 5
227  001B1 CDC001                      CALL SEGCG
```

図8 8080リスト例．

6．あとがき．

汎用アセンブラの前にインタプリタ方式の汎用プリプロセッサを設け，前例のような様々な記述形式／記述内容のμP記述に対する処理を，μP設計者が自由に定義できるようにした。これにより，汎用アセンブラだけの利用によるμP作成の不便さからμP設計者を解放し，生産性，保守性が大幅に向上した。また，きめの細かいチェック等も行えるようになったので信頼性も向上している。更に既存のμPの有効活用（μP変換）も行われるようになっている。

［参考文献］

(1)「計算機システム技術の最近の動向」 日本電子工業振興協会 S49.3.
(2) CELESTE S. MAGERS "Managing Software Development in Microprocessor Project" COMPUTER JUNE 1978.
(3) V. MICHAEL POWERS "Microprogram Assemblers for Bit-Slice Microprocessors" COMPUTER JULY 1978.
(4) 牧之内他 "MLTG—マイクロプログラミング ランゲージ トランスレータ ジェネレーター：LALRパーサの一つの応用例" 情処学会論文誌 1979.1

(5) 国立他 "超言語記述によるマイクロプロセッサ用汎用クロスアセンブラ" 情報処理 1978.7.
(6) 藤井他 "機械適合性を有するマイクロアセンブラ" 情報処理 1977.9.
(7) 山田他 "マイクロプログラムの設計自動化" 情処学会 設計自動化研究会 74-16
(8) 川口他 "汎用マイクロプログラムフローチャータ" S51年度 情処学会第17回全国大会.
(9) 佐々木他 "マイクロプログラム移行の一手法" S54年度 情処学会第20回全国大会.
(10) 山田他 "Microprogramming Design support System" NO.11 DA WORK SHOP.