資　料

機械適合性を有するマイクロアセンブラ*

藤井狷介** 飯塚　肇**

Abstract

This paper describes a design of the machine independent microassembler, named EGMA (ETL's General Micro-Assembler).

As implied by the name, EGMA has an ability to accept source assembly microprograms of various kinds of machines. In other words, the formats of source language statements and the corresponding object instructions can be given to EGMA as assembly time parameters.

EGMA is currently used to assemble microprograms of our experimental modular computer system, called ACE, but it is proved to be effectively used for several other types of machines. The detailes of the method to give EGMA machine adaptability is described in this paper.

1. まえがき

半導体の技術の進歩と共にハードウェアの使い易さは急速に増して来たが，それに伴うソフトウェアツールの開発ははかばかしくないように思われる．

筆者等は，開発中のマイクロプログラマブルなプロセッサユニット[1),2)]の設計変更や改訂，さらには多種のマイクロプロセッサの出現などによって新たにアセンブラを作成したり，種々なアセンブラを使用することが必要になると予想し，機械適合性を有するマイクロアセンブラ（以後，EGMA : ETL's General Micro-Assembler と呼ぶ）を作成し実用化した．

以下，EGMA の設計方針，構成，応用，評価について述べる．

2. はん用化の可能性

計算機は，普通それぞれ固有のアセンブラを備えており，これ等のアセンブラが受け入れる言語や出力する機械語はターゲット計算機の機種によって異なる．しかしながら，各々のアセンブラの本質的な論理動作は非常に似かよっており，その差は少ない．したがって，もしアセンブラのターゲット計算機に依存した部分を何等かの形で定義できれば，計算機の機種によらないはん用性を有するアセンブラが作成可能なはずである．このような，はん用性を有するアセンブラの提案は，D. E. Ferguson の論文[3)]に端を発していると思われる．

2.1 はん用化における問題点

はん用性を持たせることは，巨視的に見ると言語プロセッサの半自動作成の分野に属するものであり，アセンブラに対してもこのような半自動作成の試み[4)~7)]がいくつかなされて来ている．これ等の試みの中，アセンブラを機械語の相違によって生じる非共通部分と機械語の相違によらない共通部分とに分離し，非共通部分をパラメータ化し，変換規則として与える方法はこれまでのところ実用性に欠けていたように思われる．その主な理由は，

（1） よりはん用化しようとすると，変換規則の定義が非常に複雑になり，現実的にはん用化が困難になる．（複雑化の傾向．）

（2） 逆に，はん用化の程度を下げると一般のプログラムを記述する場合に不便であったり，支障を来たしたりする．（はん用化の程度．）

（3） 従来の情報加工を目的としたはん用計算機の場合，マイクロプロセッサのような半導体部品

---

* A Microassembler with Machine Adaptability by Kensuke FUJII and Hajime IIZUKA (Electronic Computer Division, Electrotechnical Laboratory)

** 電子技術総合研究所電子計算機部計算機方式研究室

としての性格を有さないから，何種かを同時に使用したりすることもなく，自分自身の上にサポートソフトウェアを具備できないという理由もない，したがって，アセンブラをはん用化したり，クロスアセンブるする必要性がなかった．(はん用化とクロスアセンブルの必要性．)

2.2 マイクロプログラムの特性

他方，プログラムをマイクロプログラムに限定し考えたとき，はん用化の可能性はどうであろうか．マイクロプログラムは一般のプログラムと比較して次のような特性を有していると考えられる．

(1) マイクロ命令は，機械に直結した基本的機能を実行するものである．したがって，マイクロプログラムの構造は規則性があり，一度に大規模なマイクロプログラムを記述し，アセンブルすることが少ない．(アセンブラ機能の簡略可能性．)

(2) マイクロ命令のアドレッシングは，ベースレジスタやインデックスレジスタを用いるなど複雑なことを行わない．(同上．)

(3) マイクロプログラムはアセンブルして直ちに実行することがない．(クロスアセンブルの適用可能性．)

以上の特性を考慮すると，一般のアセンブラにおける凝った記号番地や高機能なアセンブリ命令はマイクロプログラム用のアセンブラにおいて，簡略ないし省略しても役立つと判断できる．これによって，前節の“はん用化の程度を下げるとプログラムの記述に支障を来たす”という点が除かれ，実用性を有するはん用マイクロアセンブラの作成が可能になる．

2.3 はん用マイクロアセンブラの効果

一般にはん用性を有するマイクロプログラム用アセンブラを作成することによって，次の効果が期待できる．

(1) 種々なアーキテクチャを有するマイクロプログラマブルなプロセッサに対し（特にマイクロプロセッサに対し），これ等のマイクロプログラムを1つのアセンブラを用いてアセンブルできる可能性がある．

(2) マイクロプログラマブルな計算機の設計段階でアーキテクチャ上の変更が行われても，それと併行してマイクロプログラムが作成できる．

(3) はん用性を有するアセンブラを核として，より高級なマイクロプログラム用記述言語とそのプロセッサを上乗せできる．

3. 基本設計と構成

EGMA の基本的な設計方針は，

(1) はん用性を有すること．

(2) 作成が容易な（複雑すぎない）こと．

(3) 使いやすく実用性のあること．

であった．この中，はん用性と作成の容易さは互に矛盾するものであり，どの程度のはん用性を持たせるかが問題であった．

3.1 はん用化のレベル

アセンブラにおいて，機械命令（アセンブラおよびマクロ以外の命令）は機械語と1対1に対応しているためコンパイラのように高級言語を機械語によって意味づけるという処理が不必要であり，この点アセンブラのはん用化はコンパイラのはん用化より容易である．しかしながらアセンブリ命令に関しては，その意味をアセンブラで解釈してやらねばならない．また，機械命令でも単純に機械語へ変換するのみでなく，機械命令の記述形式によってはオペランドフィールドに記述されている意味をある程度アセンブラが解釈せねばならない場合もある．このことは，アセンブラもはん用化の程度を高めると意味の記述が必要になることを示している．

そこで，はん用化のレベルを2つに分けて考えてみた．

(1) 機械命令のレベルにおいてのみはん用化を許す．つまり，機械命令の記述形式と2進パターンとの対応関係の定義を許す．(機械適合性のレベル．)

このレベルにおける変換規則の定義を更に次の2つに分ける．

(1)-1 機械命令上の全フィールドを明確に記述しなければならないように変換規則を定義する．つまり，記述上のフィールドとマイクロ語上のフィールドを1対1に対応させる．

(1)-2 任意のフィールドを記述すれば，そのフィールドに関連する他のフィールドの指定がマイクロ語上で自動的に決るように変換規則を定義する．

(2) 上述の機械適合性のレベルに加えて，アセンブリ命令のはん用化も許す．つまり，アセンブリ命令の形式および意味が定義できるレベル．

上述のレベル分けから明らかなように，はん用化に

関する問題点は，命令語の記述形式から2進パターンへの変換規則およびアセンブリ命令の形式や意味を定義する超言語を，どのレベルにとどめるかという問題に帰着する．つまり，きめ細い記述形式を定義する超言語は，はん用アセンブラを複雑にし，超言語による記述自体の容易さにも欠ける．

3.2 EGMA におけるはん用化のレベル

2.2 に述べたように，筆者等はマイクロプログラムの特性から考えて，EGMA におけるはん用化の程度は，**3.1** のレベル分けにおける機械適合性のレベルにとどめ，アセンブリ命令のはん用化に対しては標準命令セットを準備することにした．さらに，機械適合性のレベルで，暗黙のフィールド表示（**3.1** の(1)-1と(1)-2 参照）を許すか否かに対し，許さない方針を取った．もし許せば，変換規則の定義が必然的にプロセジュアルとなり，アセンブラでその意味を解釈する必要が生じ，EGMA の簡略性指向に反するからである．言いかえると，わずかな便宜のためにはん用化のレベルを高め，それによって必要以上の労力を払うことを避けたかったからであり，またマイクロプログラムでは，前述した通りはん用化をそれ程高めなくても良いと判断したからである．

EGMA では，はん用化を進めることによってアセンブラが複雑化する点を次に示す手段を用いて簡略化した．

(1) アセンブラの入力形式は，原則として記述形式上の各フィールドとマイクロ語上のフィールドを1対1に対応させ，記述形式上の各フィールドは紙カード（イメージ）上の定ったカラム位置より始まるカラムセンシティブなものに限る．

(2) アセンブリ命令は，必要最少限のものを標準化し，機械によらず不変なものとする．

(3) 前処理部を設け，はん用化が困難と思われる処理を行わせる．また，場合によってはユーザに助力を求める．

(4) ラベル，アセンブリ命令，マクロ命令，マイクロ命令などの区別が容易に行え，構文解析に負担をかけないようにする．

(5) マクロ機能はマイクロプログラムにおいても有効と認められるので，単純置換型のマクロ機能を簡略化して設ける．

3.3 EGMA の構成

EGMA の言語構成は，**Fig. 1** に示す標準アセンブリ命令，マクロ定義命令 (Fig. 7 を参照)，制御命令 (Fig. 3 参照) そして制御命令によって固定される機械命令より成っている．一方，言語プロセッサの構成は **Fig. 2** に示すように，制御命令処理，マクロ処理，アセンブラの各部より成っており，その他に機械に依存した前処理部がいくつか存在する．

STAT	アセンブルの開始.
END	アセンブルの終了.
EQU	ラベルに式の値を与える.
LIST	リストのオン，オフ.
EJCT	ページの切り替え.
ENT	プログラム結合.
EXT	プログラム結合.

Fig. 1 Standard assembly instructions

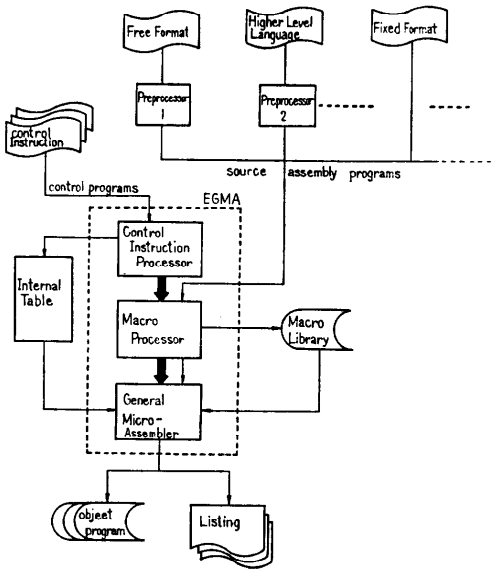


Fig. 2 Data flow in EGMA

EGMA には，FORMAT, CODE, END の3種の制御命令があり，これによって機械適合性が達成されている．以下では，制御命令処理部，アセンブラ部を中心に，EGMA の機械適合性がどのように達成されているかについて述べる．**Fig. 3**(次頁参照)に，各制御命令の形式と意味を示しておく．

3.3.1 制 御 命 令

FORMAT 命令は，命令形式ごとに記述形式上の各フィールドとマイクロ語上の各フィールドとの対応関係を定義するものである．また，記述形式上の各フィールドに許される記憶用コードのサブセット（コード

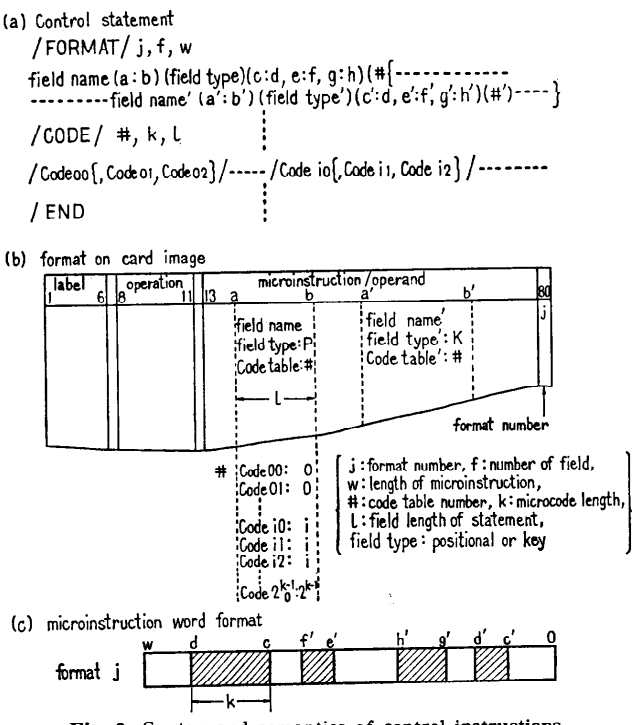


Fig. 3 Syntax and semantics of control instructions

表）を指定する．記述形式のフィールドには，一般のマクロ機能におけるポジショナル型とキイ型に似た記述を導入している．これは，全命令形式に渡って共通なフィールドをポジショナル型にし，各命令形式ごとに異なるフィールドをキイ型にすることにより記述上の容易さを与えるためである．

一般に，記述形式上の各フィールドはマイクロ語上の連続なフィールドに対応しているのであるが，ハードウェアの制約上，いくつかのサブフィールドに対応していることがある．この対応関係を指定するため，Fig. 3 における c′ : d′, e′ : f′, g′ : h′ を導入した．このように，FORMAT 命令によって定まるカラムセンシティブな記述形式を固定形式と呼ぶことにする．

CODE 命令は，FORMAT 命令で指定したコード表の番号（#）の数だけ，制御文プログラムにおいて必要である．この命令は，各フィールドで使用する記憶用コードのサブセットとマイクロコード（2進パターン）との対応づけを行う．各記憶用コードには，Fig. 3 で示すように／から／までの区切り内の記憶用コードごとに，$0, 1, 2, \cdots, 2^{l-1}$ の順序でビットパターンが割当てられる．同一のマイクロコードに対し，最大3つの記憶用コードが定義できるが，同一のフィールドに同一の記憶用コードを2つ以上定義すると，最初に定義した記憶用コードに対応するマイクロコードが有効になる．

END 命令は，一連の制御命令（制御文プログラム）の終了を示す．

3.3.2 内部テーブル

上述の制御命令を用いて定義した変換規則は，制御命令処理部によって **Fig. 4** に示すような内部テーブルとなる．FMT（形式表）は，制御文プログラム内の FORMAT 命令の数に等しい項目を有している．各項目は，その命令形式に対応する記述形式上のフィールドとマイクロ語上のフィールドとの対応関係を表わす FLT（フィールド表）を指している．FLT は，その命令形式が有するフィールド数に等しい項目を有している．各項目は，80 カラムの紙カード（イメージ）上の開始カラム番号と終了カラム番号，マイクロ語上の開始ビット番号と終了ビット番号，およびその形式上のフィールド内で使用が許される記憶用コードのセットを指すポインタ等より成っている．このポインタは，他に特殊な役割を果しており，そのフィールドが定数（リテラル）や式（記号）を記述するフィールドのときは，リザーブされた特定の数値をポインタとしてセットすることになっている．

コード表はハッシュ化してあり，各項目はそれぞれの記憶用コードに対応するマイクロコードを含んでいる．

3.3.3 形式番号

ここで問題は，入力された各文がどの命令形式に対応するのか，上述の制御命令だけでは定義できない点である．形式が判別できれば，その形式に対応する番号で FMT を引き，FMT や FLT のポインタを用いて他の表を引き，各表に定義されている変換規則に従

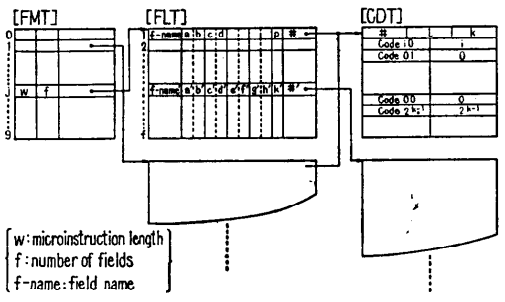


Fig. 4 Configuration of internal tables

ってアセンブルできる．

命令の形式は，機械に依存するものであり，これを一般化して変換規則の定義に持ち込むことは制御命令を複雑にする恐れがある．何故なら，形式の判別が単純に記憶用コードの相違によってのみ行われるだけでなく，任意のフィールドに記述されたストリングが示すアトリビュートによって形式を判別したい場合もあるからである．

形式判別を単純な形ではん用化することは容易であるが，EGMA では **3.2** の(**3**)の方針に従って，はん用化しにくい形式判定の部分をユーザか前処理部に行わせることにした．ユーザが行う場合は，Fig. 3 に示したように紙カード（イメージ）上の第 80 カラムに形式番号を記入し，前処理部が行う場合は，自由形式(カラムフリーな記述)から固定形式に変換されるとき，形式番号を自動的に挿入する．したがって Fig. 2 に示したように，前者は前処理を通さず，固定形式のプログラムを直接マクロ処理部やアセンブラ部に与える．

なお，EGMA の記述形式は制御命令によって全てが定義されるのではなく，Fig. 3 で示したように，ラベル，オペレーション，マイクロ命令／オペランドの各フィールドが真の意味で固定されている．したがって，制御命令によって定義できる部分は第 13 カラム以降である．これによって，マイクロ命令のメインファンクションに相当する部分がオペレーション部に記述されないため，容易にマイクロ命令とそれ以外の命令との区別ができ，構文解析が容易になっている．

4. 適　用　例

筆者等は，現在モジュール構造の複合計算機システム ACE[1),2)] を開発中であるが，そのプロセッサモジュールはマイクロプロセッサユニットを中心に構成されている．このマイクロプロセッサユニットの命令は，32 ビットの水平型を加味した垂直型マイクロ命令である．

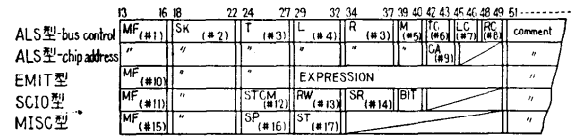
4.1 変換規則の定義

ACE プロセッサモジュール 1 号機の命令形式には，**Fig. 5** に示した 5 種類がある．各形式とも左 16 ビットが共通の形式なので，この 16 ビットに対応する記述形式上の各フィールドをポジショナル型にした．右 16 ビットは各形式によって変化するので，この 16 ビットに対応する記述形式上の各フィールドをキイ型にした．

以上の前提に基づいて，筆者等の場合は制御命令を用いて **Fig. 6** のようにプログラムし，Fig. 5 に示すような固定形式を定義している．

4.2 前 処 理 部

先に述べたように，制御命令によって定義される固定形式ではユーザが形式番号を第 80 カラムに記入せねばならない．また，記述上の各フィールドは定ったカラム位置より始まらねばならない．これ等の点を補うため，カラムフリー(自由形式)な記述を許し，形式番号も自動的に挿入する前処理部を作成した．**Fig. 7** (次頁参照)に，自由形式によるプログラム例を示す．



MF: main function. SK: skip field, T: T-bus field (Positional Type)
L: L-bus field, R: R-bus field, M: mask field, TC: T-bus Control, LC: L-bus Control
RC: R-bus Control, CA: clip address, STCM: STR/CMR select RW: MM read/write,
SR: set/reset, BIT: bit No, SP: special function, ST: stack clear (Key type)
#i: code table number

Fig. 5 Instruction formats of ACE microprocessor unit

```
                    ACE  CONTROL  LISTING

                          FORMAT  CONTROL
/FORMAT/ 0,10,32@
MF(13:16)(F)(0:5)(1)SK(18:22)(F)(6:11)(2)T(24:27)(F)(12:15)(3)@
L(0:3)(K)(16:19)(4)R(0:3)(K)(20:23)(5)M(0:1)(K)(24:25)(6)@
TC(0:1)(K)(26:28)(7)LC(0:1)(K)(29:29)(8)RC(0:1)(K)(30:31)(9)@
COM(51:71)(F)()
/FORMAT/ 1,8,32@
MF(13:16)(F)(0:5)(1)SK(18:22)(F)(6:11)(2)T(24:27)(F)(12:15)(3)@
L(0:3)(K)(16:19)(4)R(0:3)(K)(20:23)(5)M(0:1)(K)(24:25)(6)@
CA(0:2)(K)(26:31)(9)COM(51:71)(F)()
/FORMAT/ 2,5,32@
MF(13:16)(F)(0:5)(10)SP(18:22)(F)(6:11)(2)T(24:27)(F)(12:15)(3)@
ADDRES(29:49)(F)(16:31)(0)COM(51:71)(F)()
/FORMAT/ 3,7,32@
MF(13:16)(F)(0:5)(16)SK(18:22)(F)(6:11)(2)STCM(0:3)(K)(12:15)(11)@
RW(0:3)(K)(16:19)(12)SR(0:3)(K)(22:24)(13)BIT(0:1)(K)(25:31)(999)@
COM(51:71)(F)()
/FORMAT/ 4,4,32@
MF(13:16)(F)(0:5)(17)SK(18:22)(F)(6:11)(2)SP(0:3)(K)(25:27)(14)@
ST(0:3)(K)(28:31)(15)COM(51:71)(F)()

                          CODE  CONTROL
/CODE/ 1,6,4@
/ADDC/ADDZ/ADDO/ADDN/SUBC/SUBZ/SUBO/SUBN/MPY /DIV /CBD /CDB //@
///NL   /NOR /NLAR/ZERO/NAND/NR  /XOR /NRAL/NLOR/NXOR/R   /AND /@
ONE /NROL/IOR /L   /LLS /LLD /LRS /LRD /ALS /ALD /ARS /ARD /RLS /RLD /@
RRS /RRD ////////////////////
/CODE/10,6,4@
////////////////////////////////////////////////ELC /ELP ///ETR /@
ETRP/ETM //
/CODE/16,6,4@
////////////////////////////////////////////////////////SCIO////////
/CODE/17,6,4@
////////////////////////////////////////////////////////////////MISC/
/CODE/ 2,6,5@
/SK0  /RSK0 /SK1  /RSK1 /SK2  /RSK2 /SK3  /RSK3 /SK4  /RSK4 /SK5  /@
RSK5 /SK6  /RSK6 /SK7  /RSK7 /SK8  /RSK8 /SK9  /RSK9 /SK10 /RSK10/@
SK11 /RSK11/SK12 /RSK12/SK13 /RSK13/SK14 /RSK14/SK15 /RSK15/SE0  .
RSE0 /SE1  /RSE1 /SE2  /RSE2 /SE3  /RSE3 /SE4  /RSE4 /SE5  /RSE5 /@
SE6  /RSE6 /SE7  /RSE7 /SE8  /RSE8 /SE9  /RSE9 /SE10 /RSE10/SE11 /@
RSE11/SE12 /RSE12/SE13 /RSE13/SE14 /RSE14/SE15 /RSE15/
/CODE/ 3,4,4@
/GR0 ,M0L /GR1 ,M0R /GR2 ,M0T /GR3 ,M0A /GR4 ,M1L /GR5 ,M1R /@
CMR ,M1T /STR ,M1A /CDR0,M2L /CDR1,M2R /IFR0,M2T /IFR1,M2A /TSR0/TSR1/@
CTR /NPC /
```

Fig. 6 An example of control program

```
                ACE  ASSEMBLE  INPUT  LISTING                    10/13/75  PAGE    1

*      NOVA EMULATOR
*
*
*
*    MACRO DEFINITION
 MACRO
&Z READ
&Z  SCIO,,,RSW
 MEND
 MACRO
&Z WRIT
&Z  SCIO,,,WSW
 MEND
 MACRO
&Z MBRC &A,&B,&C,&D
&Z  ETR,,MPC,&A:
  ETR,,MPC,&B:
  ETR,,MPC,&C:
  ETR,,MPC,&D:
 MEND

                                    ⟆

*
*
NOVA STAT *0
*
* INSTRUCTION FETCH
IFETCH MOVE GR0,IFR0
 READ
 INC GR0,1
 MOVE IFR1,GR1,RSE1
 GOTO ARLG:
  LRS,,CIR,L=GR1,R=13,RC=1M:
 CJMP OPTYPE
OPTYPE MBRC SEQC,LDA,STA,IO
*      END INSTRUCTION FETCH
```

Fig. 7 EGMA free format

```
                ACE  ASSEMBLER  LISTING                          10/13/75  PAGE    2
EFG  LOCATION              STNO  M       SOURCE  STATEMENT                          FN

                                    ⟆

     0006  C20F 0007        75  M1              ELC       MPC  OPTYPE                2
                            76      OPTYPE MBRC SEQC,LDA,STA,IO
     0007  D20F 0020        77  M1  OPTYPE      ETR       MPC  SEQC      :           2
     0008  D20F 0035        78  M1              ETR       MPC  LDA       :           2
     0009  D20F 003A        79  M1              ETR       MPC  STA       :           2
*    000A  D20F 0000        80  M1              ETR       MPC  IO        :           2

                                    ⟆

                ACE ASSEMBLER  ERROR  LISTING                    10/13/75  PAGE    5

ERNO STNO  ERROR MESSAGE

  2    72  UNDEFINED CODE
  2    80  UNDEFINED CODE
```

Fig. 8 EGMA assembler listing

このプログラムは前処理部を通り，固定形式に変換され，マクロ処理部とアセンブラ部を通って最終的には **Fig. 8** に示すリスティングと再配置可能な目的プログラムとなる．

5. 現状と評価

EGMA は，まず昭和48年春に HITAC-8410 上にアセンブラ言語を用いてインプリメントされた[8]．その後，使用して不満足だった点を改良し，新しい EGMA を NOVA-02 上に FORTRAN-IV を用いてインプリメントした．

現在 EGMA は，ACE における，最初のプロセッサユニットと新しいアーキテクチャのプロセッサユニット双方のマイクロプログラムを記述するために利用されている．このように，プロセッサユニットのアーキテクチャが流動的な場合に機械適合性を有する EGMA を開発しておいたことは非常に有効であった．例えば，方式設計上の変更や2種類のプロセッサユニットの存在に対し，EGMA のプログラム構造は全く影響されず，非常に簡単な制御文プログラムの書きかえのみで済んでいる．

一方，単純置換型ではあるが，マクロ処理部もアーキテクチャ上の変更を吸収するのに役立っており，必要な機能だと考えられる．つまり，基本的なマクロ命令を定義しておき，プログラムをこのマクロ命令で記述することにより，アーキテクチャが変更されても，プログラムの書き換えはマクロ定義と制御文を書き換えれば良い．

なお，マクロアセンブラとしての本質的機能ではないが，次のような点を導入すれば更に使い易いものとなろう．

(1) 内部レジスタ名が CODE 命令によって半固定化されるので，プログラム時にユーザが自由に内部レジスタ名を再定義できること．

(2) 形式判定に利用するフィールドを記述の際に省略しない，およびそのフィールドに記述される記憶用コードのサブセットは他の命令形式の記憶用コードのサブセットと異なるという制限のもとで，形式番号と記憶用コードの対応づけを行う制御命令を追加する．

(3) マイクロ命令がビットステアリングを行っている場合，1ののフィールドに多重の意味を持たせる切り替え用のビット(またはフィールド)を一々記述しない．

(1)と(2)の導入は容易だと思われるが，(3)の導入はフィールドの意味を解釈し，それによって切り替え用のビット（またはフィールド）を作り出すようなはん用化が必要なため，むつかしくなるであろう．

6. むすび

機械語のプログラムを扱う従来のアセンブラでは，はん用化することが困難であった点を，マイクロプログラムに限定することによりアセンブラの簡略化を行い，解消できることを示した．また機械適合性を有するマイクロアセンブラ—EGMA—の構成と実用上の効果についても述べた．今後，ACE のソフトウェア/ファームウェア・システムの一部を，EGMA を用いて作成する予定であるが，さらに EGMA を核と見なし，この核の上にマイクロプログラムの記述が容易で，目的プログラムの効率を重視した言語（中級言語と呼んでいる）および，そのプロセッサ[9] を開発する予定でいる．

最後に，この研究の機会を与えて下さり，日頃御指導をいただく当研究所西野博二パターン情報部長，石井治ソフトウェア部長，黒川一夫電子計算機部長に厚く感謝いたします．また，御討論いただいた計算機方式研究室古谷立美，弓場敏嗣，島田俊夫の諸氏にも深く感謝いたします．

参考文献

1) H. IIZUKA : ACE-A New Modular Computer Architecture, Proceedings of Second USA-JAPAN Computer Conference, pp. 36～41 (1975)
2) 飯塚：マイクロプロセッサアーキテクチャの一設計，電子通信学会論文誌，Vol. 59-D, No. 3, pp. 188～195 (1976)
3) D. E. Ferguson : Evaluation of the Meta-Assembly Program, CACM Vol. 9, No. 3, pp. 190～196 (1966)
4) 真子：PETIT の詳細(その1)——プログラムトランスレータの開発——電子技術総合研究所彙報，第 35 巻，第 11, 12 号 (1971)
5) K. Utsunomiya : An Assembler Generator, First USA-JAPAN Computer Conference, pp. 431～435 (1972)
6) Y. Nitta : On An Efficient Assembler Building System——METAS Meta Assembler——, First USA-JAPAN Computer Conference, pp. 442～447 (1972)
7) 大駒：共通アセンブラとその変換プログラム，情報処理，Vol. 14, No. 3, pp. 165～172 (1973)
8) 藤井，飯塚：ミニコンピュータのためのはん用マイクロアセンブラ，昭和 48 年電気学会予稿 1229, pp. 1651～1652 (1973)
9) 藤井，飯塚：ACE におけるマイクロプログラム記述用中級言語，昭和 50 年電気学会全国大会予稿 1371, pp. 1816 (1975)

(昭和 50 年 10 月 27 日受付)
(昭和 51 年 12 月 8 日再受付)