論　文

マイクロコンピュータ用言語 PL/Iμ の設計と作成*

水野忠則** 井手口哲夫** 首藤　勝** 中村敏行**

Abstract

PL/Iμ is a new high-level programming language for a microcomputer.

The PL/Iμ compiler is designed specifically for MELPS 8 (Mitsubishi Electric Co., LSI CPU, M58710S) and executed on the host computer-MELCOM 7000-as a cross-compiler.

The PL/Iμ language is similar to PL/I, with primitive operations which reflect the architecture of a microcomputer, and has ON statement for controlling the interrupt mechanism, INPUT/OUTPUT statements, GENERATE statement directly taking in machine codes and so on. It has also compile-time facilities and users may extend the PL/Iμ language with %MACRO statement.

This paper describes the basic design criteria, the PL/Iμ language specification, and the implementation technique of PL/Iμ compiler.

1. まえがき

半導体技術の成果として生れたマイクロコンピュータはその価格の低廉さ，信頼性の向上，超小型化等の利点から，(1)ワイアドロジックからの置き換え，(2)ミニコンピュータの特定機能分野への進出，(3)ワイアドロジックやミニコンピュータでは価格的に成立し得なかった新分野への進出等の過程を経て幅広く各分野で使用されてきている．

マイクロコンピュータにおいても汎用コンピュータの場合と同様にプログラムをいかに能率よく開発するかは重要な問題であり，デバッグ，保守，ドキュメンテーションなどに優れた高水準言語が必要になってくる．

我々は PL/I をベースにしたマイクロコンピュータ用高水準言語 PL/Iμ を MELPS 8 (三菱電機(株)製の8ビットマイクロプロセッサ：M 58710 S) を対象に設計し，そのコンパイラを MELCOM 7000 上で作成した．

以下，PL/Iμ の基本設計基準，言語仕様，コンパイラの内部構造およびホスト計算機における作成上の問題点について述べる．

2. 基本設計基準

コンパイラはその適応分野およびコンパイラシステムの使用環境条件等を十分考慮することにより，はじめて言語仕様やコンパイラの機能を確立することができる．そのためコンパイラの設計に際しては，目的とするコンパイラの位置付けを明確にし，その基本設計基準を確立する必要がある．

マイクロコンピュータのプログラミングにはアセンブリ言語を使用している場合が多々見うけられる．この理由としてマイクロコンピュータは数値計算やデータ処理を目的に使用されるのではなく，むしろ実時間の制御を主体に使用されることに起因しており，またこのような条件を満足するコンパイラがほとんど提供されていないことにもよる[6]．また，マイクロコンピュータは装置の1つのコンポーネントとして組込まれ，装置を構成する他のコンポーネントと密接な関係を持つといった特徴も有する．

このようにマイクロコンピュータはその適応分野および使用環境条件を十分に把握する必要があり，この点に関しては調査報告書[1),2)]も出されている．我々はコンパイラの設計に際し，より明確な位置付けを確立

* Design and Implementation of a Programming Language for a Microcomputer by Tadanori MIZUNO, Tetsuo IDEGUCHI Masaru SUDO and Toshiyuki NAKAMURA (Mitsubishi Electric Co.)

** 三菱電機(株)

するために次の項目の調査検討を行った.

(1) 言語仕様

既存のシステム記述言語6種(XPL[3], CPL1[4], PL/I*, PL/I[5], PL/M[6], PL/R[7])の言語仕様の調査および端末制御プログラム[8](約6キロバイト)のアセンブリ言語とコンパイラ言語とによる記述比較の検討

(2) コンパイラシステム

コンパイラ, アセンブラ[10]およびシミュレータ[11]の相互の関連付け, およびクロスコンパイラとしてホスト計算機上の取り扱い方の検討

この2項目から得られた結果をもとに次の事項を基本設計基準とした.

(a) 生産性, ドキュメンテーションなどに利点のある高水準言語とする.

(b) ハードウェアとのインタフェースを十分に記述可能なシステム記述言語とする.

(c) ユーザが自由に文の追加をすることができる拡張可能言語とする.

(d) 複数の機種で実行が可能なクロスコンパイラとする.

(e) アセンブラおよびシミュレータとは統一的なオブジェクト言語を採用し, お互いに結合可能とする.

(a)はマイクロコンピュータの広範な応用分野に伴ったプログラム作成量の増加によるものである. PL/Iμ は汎用言語 PL/I を基本とし, かつコンパイル時の機能の充実等により, プログラム作成効率の向上を指向している.

(b)および(c)は, (1)の検討結果によるもので, マクロアセンブラ的な記述能力が必要となる.

(d)は(2)の検討結果によるもので, コンパイラはホスト計算機に従属せず, かつ任意のホスト計算機において操作指定の一様性を保持する必要がある.

(e)はコンパイラによって得られたオブジェクトをシミュレータを用いてデバッグする場合や, コンパイラとアセンブラによってそれぞれできたオブジェクトを結合する場合等においてオブジェクト言語の統一が必要となる.

3. PL/Iμ 言語

PL/Iμ 言語は PL/I 言語を基本とし, かつマイクロコンピュータの特徴を反映させた機能を有している. PL/Iμ 言語のプログラム要素を **Table 1** に, また PL/Iμ の文とその一般形式を **Table 2** (次頁参照)に示す.

Table 1 Program Elements

項目	仕様
文字セット	55文字セット (英数字36字, 特殊文字19字)
注釈	/* */
識別子	31 文字内の英数字及び @?
予約語	44 語 (識別子として不可)
演算子	*/ MOD + − PLUS MINUS < <= <> >= > = NOT AND OR XOR
定数	2進数 8進数 10進数 61進数 文字列
配列	255 迄の1次元
構造	3レベルで配列も可
式	算術式 論理式 構造式
組込み関数	17 種類

マイクロコンピュータの機能に関連する主な項目には次のものがある.

(1) 割込み制御

ENABLE 文/DISABLE 文により, CPU に対する割込みモードを制御し, ON 文によって割込みの受付け処理を行う.

(2) 機械語によるコーディング

GENERATE 文により直接機械語を任意のビット単位でコーディングすることができる.

(3) 入出力制御

INPUT および OUTPUT の関数を用い, 入出力ポート番号をパラメータとして制御を行う.

(4) 絶対番地への割当て

特定の絶対番地にプログラムやデータを割当てる場合にはラベルに定数を用いる. 定数ラベルのついた文以降は絶対番地モードになり, 相対番地モードに変更する場合は RELOCATE 文を用いる.

(5) ROM/RAM 領域への割付け

マイクロコンピュータのメモリは通常 ROM (Read Only Memory) と RAM (Random Access Memory) からなり, PL/Iμ では ROM 領域にプログラムロジック部, RAM 領域にデータ部を割付ける. ROM 領域にデータを割付ける場合には, DECLARE 文の DATA 属性を用いる.

これらマイクロコンピュータに関連する機能の他, 4. のコンパイル時の機能および多分岐制御のための DO CASE 文がある.

PL/I から削除した主な機能には次のものがあり, マイクロコンピュータではほとんど使用されない不必要な機能である.

(1) データの型 (10進浮動小数点データ, 2進浮動小数点データ, 文字ストリング等)

(2) 入出力におけるフォーマット処理

Table 2 PL/Iμ Statements

代入文		〔ラベル：〕変数〔変数〕・・・＝式；
CALL文		〔ラベル：〕CALL 手続き名〔（実引数〔，実引数〕・・・）〕；
DECLARE文	(1)	〔ラベル：〕DECLARE { 変数名〔BASED基底変数〕 / 配列名〔BASED基底変数〕（数値） } { BINARY(i) / BIT(j) } { ALIGNED / UNALIGNED } { INTERNAL / EXTERNAL } 〔INITIAL(定数〔，定数〕・・・)〕；
	(2)	〔ラベル：〕DECLARE({ 変数名 / 配列名 } 〔BASED基底変数〕 〔，{ 変数名 / 配列名 } 〔BASED基底変数〕〕 〔(数値)〕 { BINARY(i) / BIT(j) } { ALIGNED / UNALIGNED } { INTERNAL / EXTERNAL } 〔INITIAL(定数〔，定数〕・・・)〕；
	(3)	〔ラベル：〕DECLARE 1 主構造名〔BASED基底変数〕〔(数値)〕 { INTERNAL / EXTERNAL } ， { 2 従構造名， / { 2 / 3 } 基本要素名 { BINARY(i) / BIT(j) } { ALIGNED / UNALIGNED } 〔INITIAL(定数〔，定数〕…)〕， } …；
	(4)	DECLARE ラベルLABEL〔，ラベル LABEL〕・・・；
	(5)	DECLARE 文字列定数名 DATA(定数〔，定数〕・・・)，〔文字列定数名 DATA (定数〔，定数〕・・・)〕；
DISABLE文		〔ラベル：〕DISABLE；
DO	(1)	〔ラベル：〕DO；
	(2)	〔ラベル：〕DO〔制御変数＝式 TO 式〔BY 式〕〕；
	(3)	〔ラベル：〕DO WHILE 式；
	(4)	〔ラベル：〕DO CASE 式；
ENABLE文		〔ラベル：〕ENABLE；
END文		〔ラベル：〕END〔手続き名〕；
ENTRY文		手続き名： ENTRY〔(仮引数〔，仮引数〕・・・)〕〔RETURNS({ BINARY(i) / BIT(j) })〕；
GENERATE文		〔ラベル：〕GENERATE(j，{ 定数 / 変数 })・・・；
GOTO文		〔ラベル：〕{ GOTO / GO TO } ラベル；
HALT文		〔ラベル：〕HALT；
IF文		〔ラベル：〕IF 式 THEN文〔ELSE文〕
ON文		ON k CALL 手続き名；
PROCEDURE文		〔ラベル：〕手続き名：PROCEDURE〔OPTIONS (MAIN)〕〔(仮引数〔，仮引数〕・・・)〕〔RETURNS({ BINARY(i) / BIT(j) })〕；
RELOCATE文		RELOCATE；
RETURN文		〔ラベル：〕RETURN〔(式)〕；
空文		〔ラベル：〕；
入出力文		〔ラベル：〕{ 変数＝INPUT(ℓ) / OUTPUT(ℓ)＝式 }；
マクロ引用文		〔ラベル：〕マクロ名〔実引数〕〔，実引数〕・・・；
%代入文		%〔ラベル：〕プリプロセッサ変数＝式；
%ACTIVATE文		%〔ラベル：〕ACTIVATE プリプロセッサ変数〔，プリプロセッサ変数〕・・・；
%DEACTIVATE文		%〔ラベル：〕DEACTIVATE プリプロセッサ変数〔，プリプロセッサ変数〕・・・；
%END文		%END；
%EXCLUDE文		%〔ラベル：〕EXCLUDE ファイル名；
%GOTO文		%〔ラベル：〕{ GOTO / GO TO } ラベル；
%IF文		%〔ラベル：〕IF式 %THEN プリプロセッサ文
%INCLUDE文		%〔ラベル：〕INCLUDE ファイル名；
%MACRO文		%〔ラベル：〕MACRO マクロ名〔仮引数〔仮引数〕・・・〕；
%空文		%〔ラベル：〕；

$1 \leqq i \leqq 15,\ 1 \leqq j \leqq 16,\ 0 \leqq k \leqq 7,\ 0 \leqq \ell \leqq 255$

（3） 記憶域割当て（自動記憶，被制御記憶）

（4） 多重タスク操作

4. コンパイル時の機能

PL/Iμ ではプログラムの作成をより効率的に進めるため Table 2 内のパーセント(%)記号で始まる9種類の前処理文によって次のコンパイル時の編集機能を実現している．

（1） 文字列および数値への置き換え

% 代入文および PL/Iμ 制御言語によりソースプログラム内の識別子を他の文字列および数値に置き換える．% ACTIVATE 文，% DEACTIVATE 文は数値置き換えの可否を制御する．

（2） 条件付きコンパイル

ソースプログラムの必要な部分を %GOTO 文，% IF 文によりコンパイルする。プログラムの管理において有効である.

(3) テキストの登録と挿入

%EXCLUDE 文により，テキストを指定したファイルに登録し，%INCLUDE 文によりファイルからテキストを読み出しソースプログラム中に挿入する.

(4) マクロ機能

%MACRO 文によりユーザ独自のマクロを定義し，マクロ引用文により文の種類を拡張する[9].

(5) インラインアセンブル

MELPS 8 アセンブリ言語[10]を GENERATE 文を用いてマクロで定義することによりインラインアセンブルが可能となる．このようにインラインアセンブルの記述はマクロ引用の1つの応用例である.

Fig. 1 にプリプロセッサ文を含むソースプログラムの例を示す.

```
   /* AN EXAMPLE OF PL/I-MICRO PROGRAM */

%PROC='PROCEDURE';
%INCLUDE F10;
%MACRO OPEN A;
  OUTPUT(A)=64;
%END;

%MACRO CLOSE A;
  OUTPUT(A)=8;
%END;

%MACRO ALARM A;
  OUTPUT(A)=2;
  HALT;
%END;

   CONTROL:PROC OPTIONS(MAIN);
      DECLARE I BINARY(7);
%IF SWITCH=0 %THEN %GOTO X;
   /* VARIABLE FOR DEBUG */
      DECLARE (TRACE1,TRACE2)(100) BINARY(7),
              (J,K) BINARY(7) INITIAL(0,0);
%X:;
   /* TANK1 CONTOL */
      TANK1:PROC;
            I=INPUT(10);
%IF SWITCH=0 %THEN %GOTO Y;
   /* TRACE FOR DEBUG */
            TRACE1(J)=I;
            IF J=99 THEN J=0;
%Y:;
            IF I=1 THEN OPEN 9;
            IF I=3 THEN CLOSE 9;
            IF I=0 OR I=7 THEN ALARM 9;
      END TANK1;

   /* TANK2 CONTROL */
      TANK2:PROC;
            I=INPUT(20);
%IF SWITCH=0 %THEN %GOTO Z;
   /* TRACE FOR DEBUG */
            TRACE2(K)=I;
            IF K=99 THEN K=0;
%Z:;
            IF I=1 THEN OPEN 15;
            IF I=3 THEN CLOSE 15;
            IF I=0 OR I=7 THEN ALARM 15;
      END TANK2;

   /* MAIN ROUTINE */
            ON 1 CALL TANK1; /* EXTERNAL INTERRUPT FROM 8 */
            ON 2 CALL TANK2; /* EXTERNAL INTERRUPT FROM 16 */
            ENABLE;
            DO WHILE I=1;
               /* WAITING FOR INTERRUPT */
            END;
   END CONTROL;
```

Fig. 1 An Example of PL/Iμ program

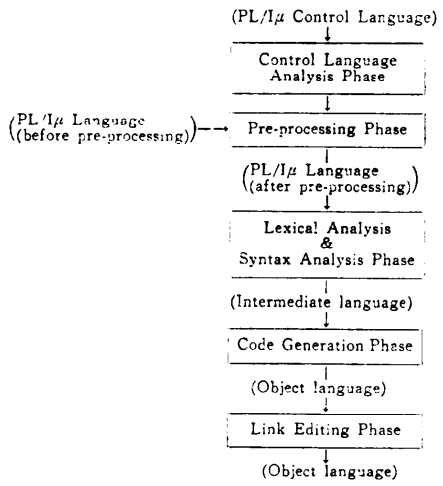

Fig. 2 A Structure of PL/Iμ Compiler

5. コンパイラの構成

PL/Iμ コンパイラは **Fig. 2** に示すように5つのフェイズから構成される.

制御言語解析フェイズはリスティングの処理，前処理変数の値設定等の指定を行う PL/Iμ 制御言語を解析し，続くフェイズの実行を制御する制御言語表を作成する.

前処理フェイズはパーセント (%) 記号で始まる前処理文の処理を行う.

字句・構文解析フェイズは前処理がなされたソースプログラムを区切り記号によって単語に分割し，各単語を内部表現形であるトークンに置き換える．このトークン列を PL/Iμ の生成規則に従い，中間言語と記号表を作成する.

コード生成フェイズは中間言語と記号表に基づき，7つのレジスタの使用方法等の最適化を行い目的とする機械語を生成する.

機械語は MELPS 8 オブジェクト言語[9] の形式に従って出力される．このオブジェクト言語は名称部，記号部，テキスト部および終了部から構成される.

結合編集フェイズは幾つかのオブジェクトプログラムに対して変数の結合とプログラムの番地割付けを行って1つのオブジェクトプログラムに結合編集する．プログラムの番地割付けは，ROM 領域（プログラムロジック部）先頭番地および RAM 領域（変数部）先頭番地の2つの相対基準番地によって行っている.

6. ホスト計算機からの独立性

PL/Iμ コンパイラをホスト計算機の性格から独立させ，幾つかのホスト計算上にクロスコンパイラとして生成するためには次の2項目を満足する必要がある．

（1） 記述言語の互換性

（2） 実行操作指定の一様性

任意のホスト計算機 H_a と H_b を考えた場合，ユーザが PL/Iμ を操作するためのジョブ制御言語をそれぞれ J_a と J_b，PL/Iμ コンパイラの記述言語をそれぞれ L_a と L_b とすると，（1）と（2）の条件を満たすためには

$$J_a=J_b \quad \text{かつ} \quad L_a=L_b$$

でなければならない．

ジョブ制御言語は一般に $J_a \neq J_b$ であるが，それぞれカタログドプロセデュア C_a と C_b とを用い，かつコンパイラ記述言語として JIS FORTRAN を用いることによって

$$C_a(J_a) \simeq C_b(J_b)$$
$$L_a=L_b$$

となる．

記述言語として FORTRAN を使用したことによってコーディング，テスト，保守，拡張，ドキュメンテーションといった高水準言語のもつ利点が得られた．

一方，FORTRAN を用いたため，次のような問題が生じた．

（1） FORTRAN では論理装置番号を通してのみ入出力操作を行っており，% EXCLUDE 文および % INCLUDE 文を用いているユーザのソースファイルおよびリンク時に指定するオブジェクトファイルをファイル名で指定できない．

（2） 開発の容易さからコンパイラを構成する各フェイズを1つのロードモジュールとして生成したが，FORTRAN はロードモジュールを呼ぶ機能が無い．

この点に関して（1）は FORTRAN の論理装置番号をファイル名と対応させ，（2）はそれぞれのフェイズをジョブ制御言語で呼ぶことによって解決した．

7. 高水準言語とアセンブリ言語

マイクロコンピュータの応用製品は，その製品サイクルが短期である場合が多く，高水準言語を利用することによってできるだけ開発期間を短かくしたいという要求がある．

一方，マイクロコンピュータのプログラムは ROM に書き込まれ，その ROM を生産台数分作成し，製品の中に組込むような使用方法が一般的である．この場合，必要となるメモリの容量は製品開発費用に対し重要な割合を占め，アセンブリ言語を利用することによってできるだけメモリの容量を小さくしたいという要求がある．

これら2つの要求の諧和がマイクロコンピュータの応用製品の開発によって特に重要な問題となる．

ここではマイクロコンピュータのプログラム開発用の記述言語として，高水準言語とアセンブリ言語のどちらを採用した方が，コスト的に有利であるかをプログラム開発費用およびメモリ費用の2つの観点からとらえる．

任意の1システムを開発する場合の全体的費用は，プログラム開発費用とメモリ費用の和からなり，

$$t=S\times P+Q\times M$$

で表わされる．ここで S と Q はそれぞれ任意の1システムを開発するために要するプログラムステップ数およびメモリ容量である．P は1ステップ当りのプログラム開発費用，M は1バイト当りのメモリ費用を示す．

このシステムを N 台生産する場合の全体的費用 T は，N 台分のメモリを必要とするため，

$$T=S\times P+N\times Q\times M \qquad (1)$$

となる．

一方，高水準言語とアセンブリ言語で生成されるメモリ容量をそれぞれ Q_c，Q_a とすると，

$$Q_c=E\times Q_a \qquad (2)$$

の関係が存在する．ここで，E はアセンブリ言語に対する高水準言語のオブジェクト容量の冗長度である．また，高水準言語およびアセンブリ言語で記述した場合のステップ数を S_c，S_a とすると，

$$S_c=Q_c/O_c \qquad (3)$$
$$S_a=Q_a/O_a \qquad (4)$$

の関係が存在する．ここで O_c，O_a はそれぞれ1ステップに対するオブジェクト生成量である．

以上の式（1），（2），（3）および（4）から高水準言語およびアセンブリ言語を用いた場合の全体的開発費用 T_c，T_a は次の式で与えられる．

$$T_c=E\times Q_a\times P/O_c+N\times E\times Q_a\times M \qquad (5)$$
$$T_a=Q_a\times P/O_a+N\times Q_a\times M \qquad (6)$$

Fig. 3 （次頁参照）に T_c と T_a の関係を示す．この図の交点の生産台数 N が高水準言語とアセンブリ言語の選択の基準となる．

たとえば，マイクロコンピュータのメモリとして最

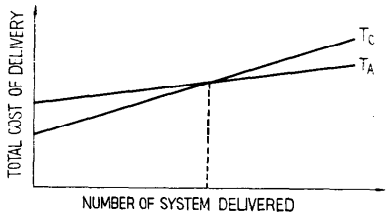

Fig. 3 Total Cost VS Number of System

も標準的な書換え不可能な ROM を用いた場合の M は,

$$M=12 \quad \text{(yen/byte)}$$

といわれている. また, O_c, O_a, E および P はプログラムの種類, 難易度によって異なるため一概には決められないが, 今次のように仮定する.

$$O_c=10 \quad \text{(byte/step)}$$
$$O_a=2 \quad \text{(byte/step)}$$
$$E=1.7$$
$$P=2000 \quad \text{(yen/step)}$$

これら値を式(5), 式(6)に代入し, Fig. 3 の交点 Nの値を求めると,

$$N \fallingdotseq 80$$

となり, 生産台数が 80 以下の計画では高水準言語を用いた方が有利である.

8. あとがき

PL/Iμ コンパイラは MELCOM 7000 上で稼動しており, MELPS 8 の応用プログラム開発のために利用されている.

マイクロコンピュータのコンパイラは特に効率の良さが問題となる. この効率を上げる一つの方法として, レジスタの一部をユーザに解放すべきか否かが問題となった. この点に関して我々は「レジスタをユーザに解放することはすでに高水準言語の域を出ており, アセンブリ言語による記述と同一である」と解釈し, PL/Iμ 言語からはずした.

しかし, きめ細かい処理も記述可能にするため, GENERATE 文および %MACRO 文を利用し, ユーザ自身で機械語レベルの文の定義を可能とした.

謝辞 PL/Iμ 開発に際し有益な御意見をいただいた三菱電機(株)計算機製作所計算機研究部および北伊丹製作所の諸氏に感謝いたします.

参考文献

1) 日本電子工業振興会: マイクロコンピュータに関する調査報告書——基礎調査編——, 50-A-89 (1975)
2) D. J. Theis: Microprocessor and Microcomputer survay, Datamation, pp. 90〜100 (1975. 12)
3) W. B. Mckeeman, J. J. Horning & D. B. Wortman: A Compiler Generator, pp. 125〜160 (1970)
4) Centre d'analyse et de programmation: CP L1 Reference Manual, GEN 1002-6 (1973)
5) IBM: System 360 Operating System PL/I Language Specifications, GY 33-6003-2 (1970)
6) INTEL: MCS-8, A Guide to PL/M Programming, MCS 127-0574-5000 (1973)
7) 小野, 池田, 清野: ミニコンのためのコンパイラ作成, 情報処理, Vol. 15, No. 8, pp. 592〜602 (1974)
8) 水野, 首藤, 大田, 山崎: マイクロコンピュータによる端末制御の一方式, 昭和 48 年度情報処理学会第 14 回大会, 講演番号 152 (1973)
9) B. N. Dickman: ETC-An extensible macro-based compiler, SJCC, pp. 529〜538 (1971)
10) 水野, 金田, 首藤: マイクロコンピュータ用クロスアセンブラ, 昭和 49 年度情報処理学会第 15 回大会, 講演番号 214 (1974)
11) 金田, 梅木, 小西: マイクロコンピュータの入出力装置を含めたシミュレーション, 昭和 50 年度電子通信学会全国大会, 講演番号 1322 (1974)

(昭和51年4月27日受付)
(昭和51年7月21日再受付)