16ビット・マイクロコンピュータ68000用高級言語S－PL／Hの開発

神野俊昭　西野秀毅　加藤正道　横畑静生　渡辺 坦　尾石辰郎
（日立製作所　システム開発研究所）　（同 武蔵工場）

1. はじめに

マイクロコンピュータの応用が広範囲に拡まるにつれ，プログラム作成量も飛躍的に増大しつつある。応用システムの開発費に占めるプログラム開発費の割合は年々増加の一途にあり，プログラム開発費の低減がシステムのコスト低減を図る上での最重要課題の一つとなっている。こうしたことを背景として，プログラムの生産性・保守性及び品質の大幅向上と，プログラム開発要員の効率的な育成に対する要求が高まってきている。これらの要求に応えるためにソフトウェア工学的アプローチが必要であることは今や共通の認識となっているが，それに先立ってプログラミング言語の高級化は必須の条件と考えられる。

このような動機に基づいて，8ビット・マイクロコンピュータ6800用高級言語PL／H[6]の開発に引き続いて，このたび，16ビット・マイクロコンピュータ68000[5]用に，PL／Hと上位方向の互換性をもつ高級言語S－PL／H[3][4]（正式名称はSuper-PL/H）を開発した。

本稿では，S－PL/Hの言語仕様を概説し言語の全体像を明らかにしたあと，その処理系（コンパイラ）の方式上の特徴を述べる。

2. S－PL／Hの言語仕様

S－PL/Hは汎用高級言語PL/Iのサブセットを母体とし，これにマイクロコンピュータ特有の機能を付加したプログラミング言語である。適用対象としては主としてシステム・プログラムを想定しているが科学技術計算などのアプリケーション・プログラムへの適用にも配慮している。マイクロコンピュータの機種に依存する言語機能を除いて，PL／H（日立・8ビット系）やPL/M－86（インテル・16ビット系）と上位方向の互換性をもたせたことが特徴といえる。

以下，S－PL/Hの言語仕様について個別に，特徴的な事柄に焦点をあてて述べてゆく。

2.1 データ型

S-PL/Hのデータ型を分類して図2.1に示す。

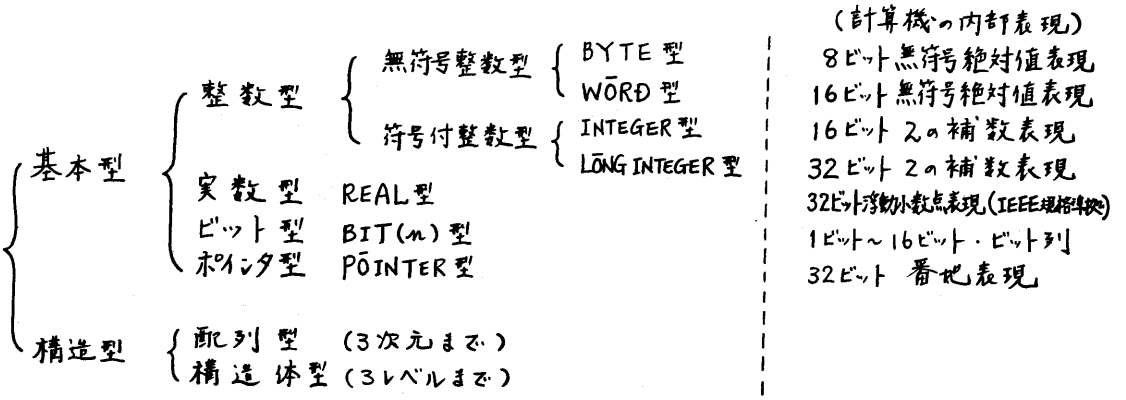

図2.1　S－PL/H のデータ型

変数記憶域のスペースを効率的に使用するために，整数型に対しては，BYTE型とWORD型，INTEGER型とLONGINTEGER型という風に「長さ」を指定することができる。BYTE型は例えばPascalにおける論理型や文字型の機能をも含んでおり，このため，比較演算の結果に対して整数データを加算するなどということが許される。これはもともと，S-PL/Hの母体となったPL/H言語が計算機の内部表現からの自然な延長としてデータ型を定めたことと深い関係があり，その意味では，S-PL/Hは"強い型"の言語よりもむしろ"型のない"言語に近い。プログラムの信頼性を高めることは言語設計の重要なファクタであるが，マイクロコンピュータ用言語の場合には計算機の機能を有効利用して記述の柔軟性を高めることも劣らず重要である。S-PL/Hのデータ型の設計にはこれらこつの要請が反映されている。

ビット型は，その長さを1ビットから16ビットまでの範囲で指定することができる。OSやプロセス制御プログラムなどで多く現われるフラグ操作の効率を上げ，かつ，記述を簡潔にする目的で導入したものである。処理系はBYTE型またはWORD型にビット長に応じて変換して目的コードを生成するため，BYTE型などに許される演算は全てビット型に対しても許される。

ポインタ型は番地の概念を抽象化したものであり，基底付変数の参照に用いられる。基底付変数は効率的にデータを参照するための強力な手段であるが，反面，使い方によって変数域の保護やプログラムの高信頼化にとって有害な影響を与えかねない。このため，ポインタ型の操作には，四則演算の適用や他の型のデータの代入などを禁止するといった強い制約を課している。

構造型は基本型を要素として段階的に新しい型を構成するための手段である。配列は3次元まで，構造体は3レベルまでというように，比較的複雑なデータ構造を表現できるようにしている。しかし実行時の効率を低下させる要因は排除したために，配列の添字の上下限値を実行時に定める動的配列は許さず，また配列の添字の下限値は常に0としてある。配列要素の型には配列型を除く全ての型が，構造体要素の型には全ての型が許される。

2.2 制御構造

S-PL/Hの制御構造を分類して図2.2に示す。

逐次実行	DO; S_1; …; S_n; END;
選択実行	IF B THEN S_1; [ELSE S_2;] DO CASE E; [C_1:] S_1; … ; [C_n:] S_n; [OTHERS: S_{n+1};] END;
反復実行	DO I = E_1 TO E_2 [BY E_3]; S_1; …; S_n; END; DO WHILE B; S_1; …; S_n; END;
GOTO文	GOTO L;
脱出文	EXIT [L];

凡例　S：文，　B,E：式
C：選択子，　I：制御変数
L：ラベル
[ ]は省略可能な部分を示す。

図2.2　S-PL/Hの制御構造

構造的プログラミング向きの機能は一とおりそなえている。図2.2の各々の意味は明らかと思われるので，拡張的な機能を2点だけ述べておく．

(1) DO-CASE文は，Eの値に応じ，Eの値を選択子に含む要素文のみを選択実行する。選択子（図の $C_1, \cdots, C_n$）は指定してもよいしなくてもよいが，指定する場合には全ての要素文に対して指定しなければならない．指定のない場合は，先頭の要素文から順に 0, 1, …, n-1 という選択子が付けられたのと同じ効果をもつ．OTHERSの付けられた要素文 S_{n+1} は，Eが $C_1, \cdots C_n$ のどの選択子とも値が一致しなかった場合にのみ実行される．この機能（いわゆるCASE文のotherwise指定）を実現したことがS-PL/Hの拡張の一つである．

(2) EXIT文は，それを含むブロック（DOブロック）の直後に分岐する。脱出するブロックは，基本的にはEXIT文を含む最小のブロックであるが，EXIT文のあとにラベルLの指定がある場合にはそのラベルの付けられたブロックである．繰り返しの中断処理をGOTO文を使わず簡潔に記述できるようにすることが導入の主なねらいである．

2.3 手続きと関数

S-PL/Hはプログラムのモジュール化を支援する手段として，手続きや関数を定義し参照するための機能をそなえている．両者の働きはほとんど同じであるが，手続きはCALL文による呼出しで実行され，関数は一次子の一つとして式の中で使われる点が異なる．関数の型にはビット型を除く基本データ型が許される．以下，「広義の」手続きという場合には関数も含むものとする．

(1) 再入可能手続き（関数）と割込み手続き

S-PL/Hでは通常の手続きの他に，宣言の中で属性を与えることによって，再入可能手続きや割込み手続きを作成することができる．再入可能属性は複数のタスク間で共有される手続きに対し，また，割込み属性は例外条件発生時の処理操作を定義する手続きに対し与える．前者は手続き宣言頭部でキーワードREENTRANTを，後者は同じくキーワードINTERRUPTと例外ベクタ番号を指定すればよい（図2.3参照）．再入可能手続きは再帰的に呼出すことができる．

```
P : PROCEDURE(X);          P:PROCEDURE(Y) REENTRANT;     P: PROCEDURE INTERRUPT 7;
  DECLARE X BYTE;            DECLARE Y INTEGER;            ....
  ....                       ....
END P;                     END P;                        END P;
```

(a)通常の手続きの宣言　(b)再入可能手続きの宣言　(c)割込み手続きの宣言

図2.3 手続き宣言の例

(2) パラメータ機構

手続きや関数にはパラメータを指定できるが，それには次の二つの種類がある．

(i) 値渡しパラメータ：呼び出し時の実パラメータの値が仮パラメータの初期値として渡される．実パラメータには一般に式が許される．

(ii) 番地渡しパラメータ：呼び出し時の実パラメータの番地（従ってその値も）が仮パラメータに渡される．実パラメータには変数のみが許される．

2.4 宣言と名前の有効範囲

変数や手続きなどのプログラム要素は，参照される前に宣言されなければならない。名前の有効範囲は通常のブロック構造型言語の場合と同様である。ただしS-PL/Hでいうブロックとは，DOブロックと手続き（広義の）ブロックとの総称である。

2.5 分割コンパイル支援機能

大規模プログラムの開発を能率よく進めてゆく場合，それをより小さなモジュールに分割し，分担して開発することが必要になってくる。そのためにはプログラムの各モジュールを単独にコンパイルでき，しかも，変数や手続きなどを互に参照し合える機能が言語や処理系に要求される。S-PL/Hではこの機能を実現するために，外部名定義属性（PUBLIC属性）と外部名参照属性（EXTERNAL属性）を変数や手続き，ラベルに対して与えられるようにしている。図2.4の例は，それぞれ独立のコンパイル単位を構成する二つのモジュールMAIN，SUBと，それらの間での変数Aと手続きPの定義-参照の関係を示している。

```
MAIN: DO ;                          SUB: DO ;
  DECLARE A REAL EXTERNAL;            DECLARE A REAL PUBLIC;
  P:PROCEDURE(X) EXTERNAL ;           P: PROCEDURE(X) PUBLIC  ;
    DECLARE X INTEGER  ;                DECLARE X INTEGER  ;
  END P ;                               /* IMPLEMENTATION OF P */
  /* MAIN PROGRAM */                    ....
  ....                                  END P ;
END MAIN ;                          END SUB ;
```

図2.4 二つのモジュール間でのプログラム要素の定義-参照.

2.6 入出力機能

S-PL/Hは機械に密着した低水準の入出力機能だけを具備する。たとえば

```
V = INPUT(8000H) ; /* 8000H番地に配置された入力レジスタから1バイト取込み変数Vに代入*/
OUTPUT(8010H)= E ; /* 8010H番地に配置された出力レジスタにEの値1バイトを設定*/
```

などと記述する。これは，言語や処理系のOS依存性をなくし，OS自体の記述をも可能にするためと，実行時ルーチンをできるだけ軽装にしたいという理由による。

2.7 マクロ機能とINCLUDE機能

マクロを定義しコンパイルに先立って展開する機能，及び，ソース・ライブラリから利用したいプログラムを選択して作成中のプログラムの中に取込む機能をそなえている。プログラムの流用性を増し標準化を促進させる上で有効と考えられる。

2.8 機械依存機能

マイクロコンピュータ用の様にシステム記述を指向する言語では，計算機の機能を有効利用して，実行効率や記述の柔軟性を高めることも重要と考えられる。この目的でS-PL/Hは以下のような，機械語に近い低水準の記述機能を具えている。

(1) 変数を絶対番地に割付ける機能.
(2) 68000ハードウェア・フラグを参照する機能.
(3) スタック制御機能

(4) 割込み制御機能

2.9 標準手続き

S-PL/Hの標準手続き（関数）を表2.1に示す。

表2.1 S-PL/Hの標準手続き（関数）

分類	標準手続き（関数）名	説明（例）
変数属性	LENGTH, LAST, SIZE	配列の要素数，添字上限，占有バイト数を返す
型変換	DOUBLE, FIX, FLOAT, HIGH, INT, LFIX, LINT, LOW, LSIGNED, SIGNED, UNSIGN, UNSBYTE	データの型を変換する。
シフト・回転	ROL, ROR, SAL, SAR, SCL, SCR, SHL, SHR	SHL(A,4)は変数Aを4ビット左にシフトした結果を返す。
ストリング操作	CMPB, CMPW, FINDB, FINDW, FINDRB, FINDRW, SKIPB, SKIPW, SKIPRB, SKIPRW, MOVB, MOVW, MOVRB, MOVRW, SETB, SETW, XLAT	FINDB(P,X1,X2)はP番地から長さX2のバイトストリングを走査しX1に合致する要素のインデクスを返す。MOVW(P1,P2,X)はP1番地から長さXのワードストリングをP2番地以降にコピーする。
入出力	INPUT, INWORD, OUTPUT, OUTWORD	2.6節参照。
その他	ABS, IABS, STACKPTR, USTACKPTR, TIME	ABS(X)は変数Xの絶対値を返す。スタックポインタの参照。TIME(X)はXで指定される時間過れば経過

3. S-PL/Hコンパイラ

S-PL/Hコンパイラは，S-PL/Hで記述されたソース・プログラムを68000機械語の目的プログラムに変換する。我々は，HITAC-Mシリーズをホスト計算機とするクロス・コンパイラと，68000システムH680SD300[12]をホスト計算機とするレジデント・コンパイラの両方を開発した。コンパイラ設計の方針として次のことを設定した。

(1) 実行効率の重視される分野に多用されることを配慮して最適化機能の強化を図る。

(2) 目標計算機が68000から他の機種へ変更されても少ない労力で対処できるようretargetabilityを高める。

(3) ホスト計算機の移行に伴なう作業量を軽減するために移植性を高める。

さらに，コンパイラ設計の基礎である目的プログラムの設計に当たっては，

(4) 再入可能かつ再帰呼出し可能な目的プログラムを生成できること，

(5) 目的プログラムを主記憶にロードしたのちでも実行前にアドレス配置を変更できるような（これを動的再配置可能[7]またはフロータブルという）目的プログラムを生成できること，

に留意した。

3.1 目的プログラムの実行方式

3.1.1 実行時の記憶域管理

実行時の記憶域の使い方を分類すると，静的記憶域方式（Fortran流）と動的記憶域方式（Pascal流）に大別されるが，S-PL/Hは両者を併用した管理方式を採用している。これは再入可能かつ再帰呼出し可能な目的プログラムを生成する必要性と，変数域の確保や変数参照などに伴なう実行時のオーバヘッドを極力軽減する必要性との折衷方式として編み出されたものである。

動的記憶域には仮パラメータ変数や再入可能手続の局所変数が割付けられ，それ以外の変数は静的記憶域に割付けられる．動的記憶域はスタック機構により実現され，手続き呼出しの度に，手続き区画（パラメータや戻り番地，再入可能手続きの場合にはさらにリンケージ・セクションや局所変数域などを一まとめにした，動的記憶域の構成単位）が確保され，戻りとともに解放される．

3.1.2 目的プログラムの動的再配置

プログラムを実行する場合，命令部とデータ部を主記憶上の特定の番地に固定する必要がある．実行時番地をいつ決定するかによって目的プログラムは次の二つのタイプに分類される．

(1) 静的再配置：リンク・エディット時又はプログラム・ロード時に決定し以後変更できない．
(2) 動的再配置：実行時に決定することができる．

S-PL/Hコンパイラは動的再配置可能な目的プログラムを生成する．動的再配置性の利点は，目的プログラムをROMに書き込んだのちこのROMをどの番地のソケットに実装してもプログラムは正しく実行される点にある．この性質によって，標準プログラムのLSI化（ソリッドステート・ソフトウェア）の実現が容易になり，ROM媒体によるプログラム供給の促進に向けて一歩近づくことになる．

動的再配置性の実現機構の要点は次のようである．

(1) ロード・モジュールのデータ部先頭番地αをプログラム実行中維持される実行管理表の中に設定し記憶する．
(2) 各プログラム・モジュールiはベース・レジスタR_Bにこのモジュールの基準番地$\alpha+d_i$を初期設定する．d_iはロード・モジュールのデータ部先頭からモジュールiのデータ部先頭までの偏位であり，リンク・エディット時に決定される相対量である．
(3) 外部変数（他のモジュールで定義された変数）は，R_Bと，参照を行なうモジュールのデータ部先頭から外部変数までの偏位（やはりリンク・エディット時に計算される相対量）で参照する．

ここで現われる絶対番地はαのみであり，しかもαは実行時に変動可能な値であることが，動的再配置性を保証しているわけである．

3.2 コンパイラ

S-PL/Hコンパイラの稼動環境を表3.1に，処理とデータの流れの概略を図3.1に示す．

表3.1 S-PL/Hコンパイラの稼動環境

	クロス・コンパイラ	レジデント・コンパイラ
本体計算機	HITAC Mシリーズ	H680SD300 (68000システム開発装置)
使用メモリ量	512 KB	128 KB
補助記憶	磁気ディスク	フロッピーディスク2台以上

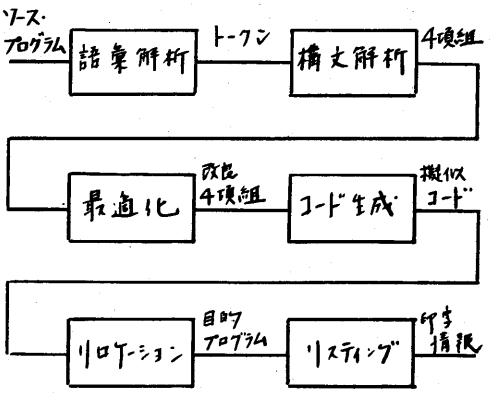

図3.1 S-PL/Hコンパイラの処理とデータの流れ．

3.2.1 最適化方式

S-PL/Hコンパイラの最適化設計にあたっては，①最適化対象の重点化，②機械独立処理と機械依存処理の分離を主な方針とした。方針①の実現のために最適化効果の事前評価を行ない，実用に耐えうる目的プログラム性能を達成するためには，少なくとも基本区間（制御の流入出が最初と最後でしか起こらぬような最大の命令列）を単位とする最適化（局所的最適化）を行なう必要があるという結果を得た。開発期間との兼ね合いからまず局所的最適化を実現することとし，フロー解析や手続き呼出しによる影響などを考慮した大域的最適化も後で容易に組込めるようインタフェース上の配慮を行なった。局所的最適化の主な項目は，①定数計算の畳み込みと伝播②冗長演算の省略③論理式の最適化④共通式の削除⑤配列要素の添字計算の最適化などである。

局所的最適化は機械に依存しないより4項組（quadruple）のレベルで施すこととした。機械に依存する最適化（合成命令の最適化や短語長の命令形式への変換など）は擬似コードのレベルで施すこととし，最適化処理部ではなく，目的プログラムを編集生成するリロケーション処理部で実現することとした。

3.2.2 コード生成方式

コード生成処理部の機能は大別すると，①4項組を定められた仕様に従って，目的コードに展開する機能と，②レジスタの割付けや管理を行なう機能とから成る。これら二つの機能もモジュールが相互に強い結合強度をもちながら混在していることが，コード生成部の処理を複雑にする要因としてあげられる。そこで我々は，コード生成部をまずレジスタ割付け部と目的コード展開部の二つのモジュールに分割し，相互の結合強度を弱めるために両者が限定されたインタフェースを介してのみ連結し合うように構成した。その上で，目的コード展開部は，retargetabilityを考慮して基本的に表駆動方式で実現することとした。

(1) レジスタ割付け部の処理.

レジスタ割付け部は4項組ごとに，その4項組から目的コードを生成するのに必要な全てのレジスタを，4項組の付随情報として予える。一般に4項組に対して割付けられるレジスタは，①オペランドをアドレシングするのに必要なレジスタ（アドレシング・レジスタ）と②演算を実行する過程で新規に必要となるレジスタ（オペレーション・レジスタ）に分類できる。アドレシング・レジスタは対象となる変数の属性（添字付か否か，ベースドか否か，記憶域が静的に割付けられるか否か等）によって一意に定まる。オペレーション・レジスタは，4項組のくオペレータ，オペランドのデータ型，オペランドの存在場所＞によって原則的には一意に定まる（効率のよい目的コードを作るために更に詳細なファクタを追加することもある）。レジスタ割付け部は各4項組に対し，アドレシング・レジスタとオペレーション・レジスタを具体的に定め，特殊中間語上に表現し，それを4項組に付随させて出力する。

(2) 目的コード展開部の処理.

一般に4項組の目的コードはレジスタにオペランド値を乗せる命令部分（基本命令部）と実際の演算を実行する命令部分（演算実行命令部）に分かれる。基本命令部は，オペランド値の存在場所によって仕様が定まる。演算実行命令部はオペレータとオペランドのデータ型によって仕様が定まる。そこで表のコンパクト化を考慮して，以下の3種類の表を用意し，これらの表をもとに展開される目的コ

ードのスケルトンと，レジスタ割付け情報とから目的コードを確定する。

① 展開パタン表：〈オペレータ，オペランドの型〉から展開コードの群組を与える表。

② 基本命令化様表：〈展開パタン，オペランド値のある場所〉から基本命令部の化様を与える表。

③ 演算実行命令化様表：〈オペレータ，オペランドの型〉から演算実行部の化様を与える表。

(3) レジスタ・ライブ指数によるレジスタ管理。

4項組に対するレジスタ割付けを行なう過程で新しいレジスタ要求があった場合，定量的な基準にもとづいて提供すべきレジスタを決定することが望ましい。そこでレジスタが保持するデータの「重要度」を定量的に表現する「ものさし」として，レジスタ・ライブ指数という概念を導入した。

変数ライブ指数　任意の変数びと任意の4項組gに対して，びのgにおける変数ライブ指数 VLIVE(び, g) とは，gの直後においてびが保持する値が，g以降の4項組で変更されることなく参照される回数に，びの性質に応じて定まる重みづけをかけ合せた値である。

レジスタ・ライブ指数　任意のレジスタγと4項組gに対して，γのgにおけるレジスタ・ライブ指数 RLIVE(γ, g) を次式で定義する。

$$RLIVE(\gamma, g) \stackrel{def}{=} \begin{cases} \sum_{v \in C(\gamma)} VLIVE(v, g) & : C(\gamma) \neq \phi \text{ の場合} \\ 0 & : \text{otherwise} \end{cases}$$

ただし $C(\gamma) = \{$変数び $|$ びの値はレジスタγにある$\}$。

提供レジスタを決定する場合は，処理中の4項組において最もレジスタ・ライブ指数の小さいものから順次必要個数分を選択することとした。

4. 結び

S-PL/Hの言語仕様を概説し，処理系の方式上の特徴を述べた。S-PL/Hは現在各種の68000応用システムの開発に適用中であり，生産性・保守性の向上のため今後さらに普及を促進してゆく計画である。

またS-PL/Hをより使い易いシステムにするために，現在，

(1) 各種ソフトウェア・ツールと接続しソフトウェア一貫生産システムの中に位置づける，

(2) より高度の最適化の実現によって目的プログラムの一層の高性能化を図る，

などの改良を加えつつある。

参考文献

1) 日立：システム開発装置 H680 SD300 概説マニュアル（1981年3月）

2) 同上：システム開発装置 H680 SD300 ユーザーズマニュアル

3) 同上：スーパPL/H言語マニュアル

4) 同上：スーパPL/Hユーザーズマニュアル

5) 同上：68000アセンブリ言語マニュアル（1981年3月）

6) 吉村外：マイクロコン用システム記述言語PL/Hコンパイラの開発思想，情報処理学会第19回全国大会講演論文集(1978年)

7) 西野外：16ビット・マイコン68000用Super-PL/Hコンパイラにおける目的語実行方式，情報処理学会第22回全国大会講演論文集(1981年)