S3—9

言語PLANの分散処理向きの拡張

山内長承　田中英彦
（東京大学工学部）

1. はじめに

分散処理システムを記述するための手段として，ポリプロセッサシステムPPS-1上のミニコン向き高級言語PLANを拡張したので報告する。PPS-1のオペレーティングシステムPPS-1.OSはポリプロセッサ管理システムとしての側面と，計算機網向きオペレーティングシステムとしての側面をもっているが，いずれも分散処理システムを支援するものとなる。分散処理システムはこのオペレーティングシステムのプリミティブ（システムマクロ命令）を用いて制御されるが，プロシージャを記述する高級言語PLANとのインタフェースを取ることによって，分散処理プログラムを効率良く記述することができる。

2. PPS-1とPPS-1.OS

ポリプロセッサシステムPPS-1は3台のマイクロプログラム制御プロセッサが主記憶を共有するシステムで[1]，同時に研究室内計算機網TECNET[2]のホストでもある。PPS-1.OSはファームウェア化されたオペレーティングシステム核であり，表1のような仮想マシンを提供する。[3]

表1

入出力機能	入出力制御プロセスに対するプロセス間通信によるプロトコール・インタフェース
メモリー	磁気ディスクによる仮想記憶
プロセス制御	プロセス制御プリミティブ（生成，消滅，実行，停止）
プロセス間通信及び同期	プロセス間通信プリミティブ（相手プロセスの存在位置によらない一様な手続きによるプロセス間通信が可能）

3. 分散処理記述言語の選択とPLAN

並列処理の記述には次の2通りが考えられる。すなわち，

(1) fork/joinで記述する言語を用いる。必要に応じてコンパイラがプロセス制御/通信プリミティブに翻訳する。

(2) 通常のプログラム言語にプリミティブに対応するシンタクスを付け加え，それを用いて並列プロセスのプロシージャを1個1個記述する。プリミティブの使用はユーザが管理する。

分散処理の記述に対してどちらの方法が向いているか必ずしも明らかでないが，実験の第1段階として容易な後者を取上げる。

PPS-1上の言語としてはPLANが実装されている。PLANはALGOL-likeなミニコン向きの高級言語であり，コンパイラによって仮想機械語（PLAN-VMコード）に落とし，それをPPS-1上ではマイクロプログラムによるインタープリタによって実行する形式にしている。インタープリタはPPS-1.OSの提供する仮想マシンを利用して実現されており，仮想記憶，入出力を利用することができるが，並列処理記述を可能とするようなプロセス間の同期などの機能をユーザが使うことはできない。PLANのシンタクスは文献4)を，PLAN-VMコードは表2を参照。

0.	NOP	No Operation
1.	EX	Execution
2.		Get
3.		Put
4.		Put String
5.		Load Indirect
6.		Store Indirect
7.		Get Array
8.		Greater
9.		Less
10.		Equal
11.		Add
12.		Subtract
13.		Or
14.		Multiply
15.		Devide
16.		Module
17.		And
18.		Not
19.		Negate
20.		Shift Logical
22.		Store Immediate
23.		K-call
2.	PS	Push Stack
3.	PL	Pull Stack
4.	MSP	Modify Stack Pointer
5.	JUMP	Jump Unconditionally
6.	JFLS	Jump If False
7.	JPRC	Jump to Procedure

表2　PLAN-VMコード

4. PLANの拡張の方法

言語PLANのシンタクスから，PPS-1.OSのプリミティブを直接発行できるように拡張することによって，PLANを用いて並列処理プログラムを記述できるための基本的な手段が得られることになる。拡張は，(1) 仮想機械語（PLAN-VMコード）の拡張と，(2) PLANシンタクスの拡張の2点について行なった。

4.1 仮想機械語の拡張

仮想機械語（PLAN-VMコード）は，PPS-1のマイクロプログラムによるインタープリタによって実行される。仮想機械語上の拡張は通常用いられるSVCトラップの方式によらず，K-callと呼ぶインタフェース用の命令を設けてオペレーティングシステムのプリミティブを呼出す方式とした。実行は図1のように行なわれる。パラメータの受渡しはPLAN上のサブルーチン呼出しの形式と同様にPLAN-VMの作業空間であるスタックを介して行なう。

K-callで用意したプリミティブは表3に示すようなものである。

プリミティブ	機　能
P.CREATE	プロセス生成要求
P.START	プロセス実行要求
P.STOP	プロセス停止要求
P.REMOVE	プロセス消滅要求
P.SEND	プロセス間通信で送信要求
P.RECEIVE	プロセス間通信で受信要求
P.COMWAIT	プロセス間通信で送受信終了待ち
P.COMCANCEL	プロセス間通信で送受信要求の取消

表3　プリミティブ

4.2 PLANシンタクスの拡張

PLANのソースプログラム上で並列処理の制御を行なうためのシンタクスとしていろいろなものが考えられるが，ここではプリミティブをほとんどそのままの形で呼出す方法をとった。すなわちPLANのソースプログラム上からはシステム関数呼出しの形式で利用できるようにし，必要なパラメータは関数の引数として記述することとし，主たる復帰情報は関数値として返す形式とした。プログラム例を図2に示す。

5. 評価と今後の課題

5.1 実装したシステム

(1) インタープリタのマイクロプログラム量は750語から950語に増加した。(24ビット/語)

(2) コンパイラの増加は追加したシンタクスの処理に関数処理ルーチンを利用したため，3.75K語から4K語に増えた程度で済んだ。

(3) これらの拡張に要した仕事はわずかに10人日程度であり，極く簡便な方法であったと言え，他言語への実装も容易と考えられる。

(4) 実行速度は，一般の仮想機械語の平均17μSに比較して，K-call命令はインタプリタの平均40μSのうえにプリミティブの実行時間が加わる。

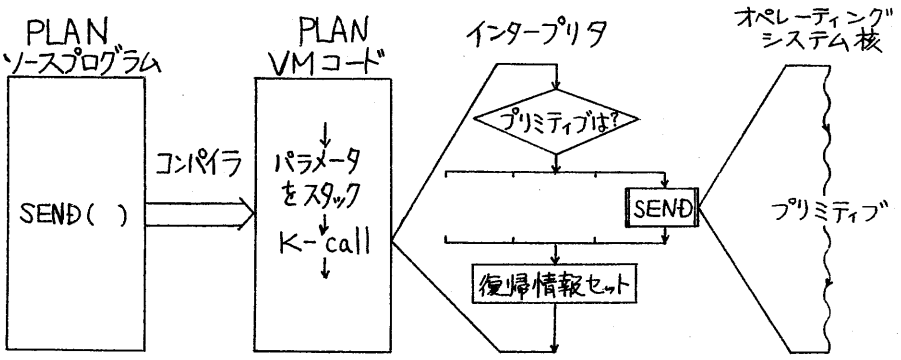

図1　拡張部の実行の様子

```
REPEAT
 ABORT:
  ERRORCODE:=0;
  REPEAT
    COMSTATUS:=RECEIVE(ANY, ANY,
          TRUE, SEGNUMBER, ARRAY BUFFER, RBLENGTH) ;
    IF COMSTATUS>#1F DO GOTO ABORT
  UNTIL COMSTATUS=0 ;
  IF BUFFER(0) AND #00FF > 2 DO ERRORCODE:=#10;
  DIRECTION:=BUFFER(0) AND #8000;
  IF DIRECTION=0 THEN BEGIN
     PRECEIVER:=HRTSEARCH(BUFFER(1));
     IF PRECEIVER=0 DO ERRORCODE:=#10;
```

図2　拡張シンタクスを用いたプログラム例

5.2 実用性

拡張したPLANを用いて，計算機網ジョブ管理システムなど並列処理を含むプログラムを記述した結果，オペレーティングシステム核の構造，機能などを良く掌握しているプログラマにとっては十分使いやすいものであった。但しプリミティブをそのままの形で使うことになるので，使い方を誤まった場合の処理はオペレーティングシステム核に任されることになり，核が十分な保護機能をもたない場合システムを混乱させる可能性がある。いずれにせよ使い易さの点において，今後他の並列プログラミング用の言語と比較する必要があろう。

6. むすび

分散処理記述用言語は分散処理システムの設計，実装上大きな影響を与える。その意味から今後並列プログラミングの経験を積むことによって，記述用言語の研究も進むものと思う。

文献

1) 元岡達・山室良夫：ポリプロセッサシステム PPS-1，情報処理 Vol.15, No.7, 1974
2) 田中英彦・元岡達：研究用電子計算機網 TECNET，電子計算機研究会資料 EC 73-57
3) 仲川明和：ポリプロセッサシステム管理用ファームウェア，東京大学工学系研究科修士論文 1976
4) 武市正人：ミニ言語のミニコンパイラ①-⑤，bit Vol.6, No.8-12 1974