ＰＳＩ－ⅡにおけるＥＳＰサポート用ファームウェア

6B-5

池田　守宏、村澤　靖、吉田　裕之、近山　隆
（三菱電機）　（沖電気）（ＩＣＯＴ）

１．はじめに

第５世代コンピュータプロジェクトの一環として、マルチＰＳＩのフロントエンドプロセッサＰＳＩ－Ⅱを開発した。　このＰＳＩ－Ⅱでは、システム記述言語、及びユーザー言語としてＥＳＰ（Ｅｘｔｅｎｄｅｄ　Ｓｅｌｆ－ｃｏｎｔａｉｎｅｄ　Ｐｒｏｌｏｇ）を用いている。ＥＳＰは、Ｐｒｏｌｏｇにオブジェクト指向の機能を追加した言語である。　ＰＳＩ－Ⅱでは、このＥＳＰを効率よく実行するため、専用の機械語命令を設けてサポートしている。　本稿では、ＥＳＰオブジェクトの表現形式及び、ＥＳＰサポート用の各種命令などについて紹介する。

２．ＥＳＰのオブジェクト表現形式

ＥＳＰのプログラムはクラス定義を単位として構成される（図１）。　ＥＳＰのようなオブジェクト指向言語において、実用上問題となのは、オブジェクトへのメッセージ伝達、即ちメソッドの動的呼出しが、直接的な関数の呼出し（ＥＳＰにおいてはローカル述語の呼出し）に比べて時間がかかるという点である。　これを短縮するためには、オブジェクトの表現形式をどのようにするかがポイントである。　オブジェクトへメッセージ伝達された際に、オブジェクトのメソッドをできる限り速く見つけ出せる形式にしなければならない。

```
class クラス名 has
[nature 継承クラス定義　　]
[ クラス・スロット定義
  クラス・メソッド定義　　]
[instance
  インスタンス・スロット定義
  インスタンス・メソッド定義]
[local
  ローカル述語定義　　]
end.
```

図１　クラスの記述形式

さてＰＳＩ－Ⅱにおいて、ＥＳＰオブジェクトは内部的に図２のような形式で表わしている。　メソッド・コードアドレスの格納されているメソッド・テーブルのアクセスにはハッシュ関数を用いて行う。　そこで、メソッド呼出しの命令（詳細は後述）の引数として、ＥＳＰオブジェクト・アドレス、あるいはオブジェクト・ディスクプリタとともに索引キーが渡される。　メソッド呼出しにおける索引キーは、そのメソッド名のアトムナンバーと引数個数の和とした。　メソッド呼出しがなされた時、そのメソッドの索引は次のようになされる。　まず、索引キーとＥＳＰオブジェクト・ディスクプリタで指定されたメソッド・テーブル・マスクの論理ＡＮＤを取り、得られた値によってハッシュする。　メソッドテーブルには、リニアサーチ個数Ｎｕｍとエントリーアドレス（相対）が入っている。そして、エントリーアドレスの指している先からＮｕｍ個のキーとメソッド・コード・アドレスのペアが格納されている。　そこで、ハッシュして得られたエントリからリニアサーチして一致するキーを捜して、メソッド・コード・アドレスを得る。　リニアサーチの最大はＮｕｍ回となるが、索引キーを上記のように設定したことで１，２回でヒットする。

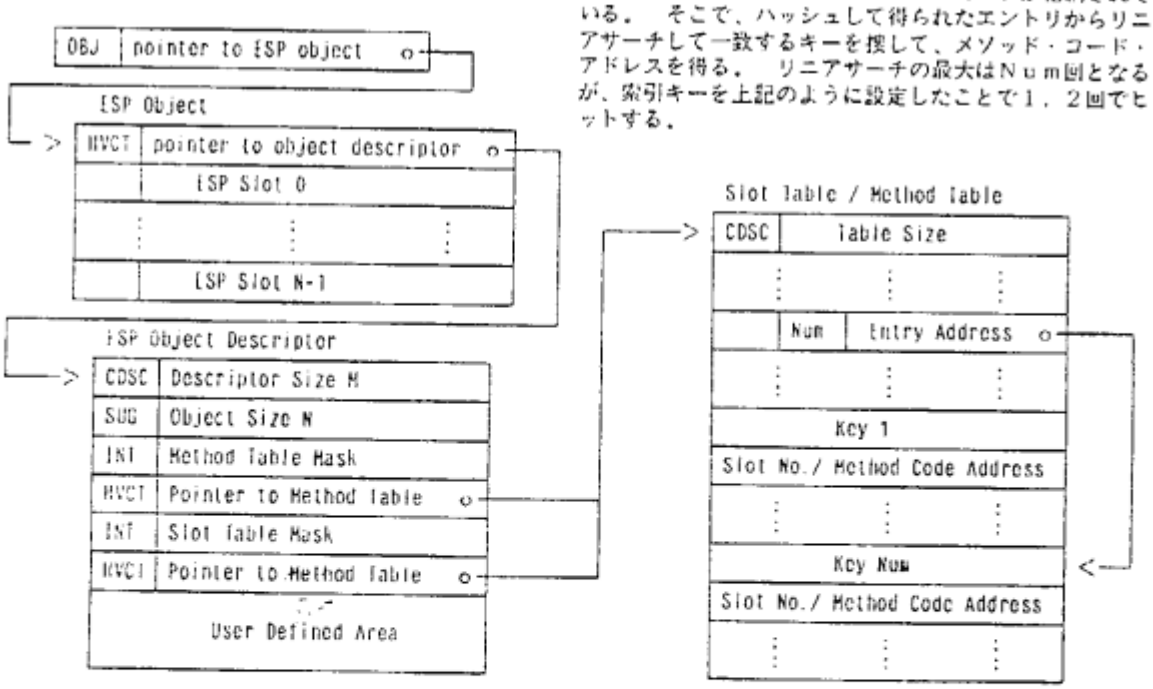

図２　ＥＳＰオブジェクトの表現形式

ESP Support Firmware on PSI-Ⅱ
M.Ikeda*1, M.Murasawa*1, H.Yoshida*2, T.Chikayama*3
*1:Mitsubishi Electric Co.,Ltd. *2:Oki Electric Industry Co.,Ltd. *3:ICOT

3．ESPサポート命令

PSI-Ⅱでは、約18種のESPサポート命令を用意しているが、ここでは上記のメソッド呼出し命令、スロットアクセス命令とデーモン・コンビネーション中のカット命令について、説明する。

（a）メソッド呼出し命令

メソッド呼出し命令には、次のものを用意している。

```
call method immediate
call method
call overridden method immediate
call overridden method
```

多くの場合、呼出すメソッド名はプログラム・ソース上に書かれている。 immediate 系はそれらのためのもので、call method immediate は、もっとも良く使用されるメソッド呼出しのために使用する。 call overridden method immediate は他のクラスで定義されているメソッドを呼出す時に使用する。 ところで、通常のメソッド呼出しでは、メソッド名や引数個数はクラス定義時に決定されていなければならない。 しかし、ESPではメソッド名や引数個数を動的に決定させることができる。 これは、:refute というメソッドで実現している。 この時、call method, call overridden method は使用する。

（b）スロットアクセス命令

スロットはオブジェクトの性質や内部状態等を保持する変数である。 スロットを参照するには、オブジェクトとスロット名の対で指定する必要がある。 スロットアクセス命令としては次のものを用意しており、各々組込述語に対応している。

```
slot
set slot
```

slotは、指定したオブジェクトのスロット値を得る命令であり、一方、set slotは、スロットに値を与える命令である。

スロットの索引法は、メソッドのそれとほぼ同じである。即ち、ハッシュ関数を用いてスロット・テーブルからスロット番号を取り出す。 この場合、索引キーとしてはスロット名のアトムナンバーを使う。 また、ハッシュ値は、索引キーとESPオブジェクト・ディスクリプタで指定されたスロット・テーブル・マスクの論理AND値を用いる。ハッシュして得られたエントリからリニアサーチして一致するキーを捜して、スロット番号を得る。 その番号より、ESPオブジェクトのスロットをアクセスする。

```
Method :- M-b, Mf-b, (M ; Mf), Mf-a, M-a ;
                      -$M
$M :- M  ;
$M :- Mf ;
M :- !... ;    <--this cut (method cut) cuts
M :- ...  ;          the alternative of $M
```

図3 メソッド・カットの例

	"append" 1ループ	ループ中の呼出し
メソッド呼出し	29	15
ローカル述語呼出し(1)	17	3
ローカル述語呼出し(2)	15	—*

*複合命令を使用しているため、他と対等な呼出しステップ数は表わせない。

表1 "append" のステップ数

```
class methodappend has
 :append(C,0,L,L):-!;
 :append(C,[X1|L1],L2,[X1|L3]):-
    :append(C,L1,L2,L3);
end.
```

図4 メソッド呼出しのappend

```
class localappend has
 :append(C,X,Y,Z):-
    append(X,Y,Z);
local
 append(0,L,L):-!;
 append([X1|L1],L2,[X1|L3]):-
    append(L1,L2,L3);
end.
```

図5 ローカル述語呼出しのappend

（c）メソッド・カット命令

ESPにおいて、継承による複数クラスのメソッドの結合は、図3のように実現され、デーモン・コンビネーションとよばれる。 この結合において注目すべき点は、主処理述語(primary predicate)間はOR結合であることである。 このため、一つの主処理述語内でのカットは、主処理述語すべてのオルタネイトもカットしなければならない。現PSIにおいて、この処理は述語呼出し深さごとに付けられたレベル番号を用い、レベル番号指定のカット（リモート・カット）で実現していた。 これに対し、PSI-Ⅱでは新たにメソッド・カット命令を用いて実現している。メソッド・カットは、カットの含まれるクローズよりも上のレベルをカットするという意味で、リモート・カットと似ている。 しかし、PSI-Ⅱではカットすべきレベルはその状況によって変化する。 例えば、トレーサーが呼ばれていて、それからメソッドが呼ばれた時は、通常の実行より深いレベルとなっている。 そこで、デーモン・コンビネーション節（図3のMethod）から、OR結合のボディを呼びだす時点のバックトラック・ポイント（BP）を、引数としてメソッド・カット命令に渡すことにした。[1] そして、メソッド・カット命令は、単にその引数として与えられたBPを新しいバックトラック・ポイントとする。引数BPは隠れた引数として、トレーサー起動中にも呼出し環境フレームに保存されるので、メソッド・カットは正しく動作する。

4．処理性能

メソッドの呼出しと、ローカル述語の呼出しの性能を比べるため、"append" をそれぞれの呼出し方法でおこなった。 表1は、それぞれの呼出しにおける "append" ループのステップ数及び、そのうちのそれぞれの呼出し命令のステップ数を表す。 ローカル述語呼出し(2)では最適化により、ローカル述語呼出しが、インデキシング命令と複合されている。 呼出しのステップ数では5倍の差はあるものの、引数のユニフィケーションを含めたループのステップ数では、複合命令を使って最適化した場合でも、約2倍前後の差となっている。

5．おわりに

本研究を行うにあたり、多くの貴重の助言をいただいたICOT及び、関連メーカーの方々に、深く感謝する。

参考文献

[1] Mats Carlsson, Compilation for Tricia and its Abstruct Machine, Upmail IR no.35, Uppsala University, 1986

[2] 池田ほか、PSI-Ⅱシステムのファームウェア、第34回情報処理学会、1987