汎用アセンブラのＰＳＩ上への移植

7B-7

立野 裕和 高木 茂行

(三菱電機(株)) ((財) ＩＣＯＴ)

1．はじめに

ICOTにおいて，PSI の開発用に作成された汎用アセンブラ〔1〕 を PSI上へ移植した。

移植した汎用アセンブラは，DEC 20を TSSで利用した場合，その利用者が非常に少ない状態での実質的なアセンブル速度（応答速度）注1)と同等の性能を示した。

移植に要した人工は３人月であった。

本報告では，移植で得られたPROLOGから ESP への書き換えに関する知識を中心に述べる。

注1) 応答速度とは，アセンブラを起動後アセンブルするマイクロプログラム名を入力した時刻からアセンブルが終了した時刻までの経過時間。

2．汎用アセンブラ概説

汎用アセンブラの構成を図１に示す。機械定義をハードウェアに合わせ決められた形式で記述する。この機械定義を前処理プログラムに入力することで機械定義部が得られる。機械定義部とアセンブラ核部を合わせて専用アセンブラが得られる。

汎用アセンブラのモジュールは表１に示した４種のモジュールから構成されている。

3．移植

PSIのシステム記述言語は ESP(Extended self contained prolog) 〔2〕と呼ばれ，PROLOGにオブジェク指向が付加されたものである。PSI 上の全てのプログラムはこの ESP で記述せねばならない。今回の移植作業では，ESP の特長であるオブジェクト指向性を利用することはあまり考えていない。 (デーモンの機能を少し利用した程度)

即ち，表１に示した各モジュールを１クラスとするようなコーディングとなっている。移植作業では以下の２点の扱い方が問題となった。

(1) 副作用の問題 (assert/retruct 系の組込述語)

(2) ＥＳＰでサポートされているマクロ展開機能

表1 汎用アセンブラモジュールリスト

プログラム名	モジュール名
前処理プログラム	GEN, SUB
アセンブラ核部	ASM, SUB
リンカ	LK, SUB

(1) 副作用の問題

汎用アセンブラではassert/retructが大域情報の記憶場所として活用されている。大域情報としては，アセンブルしようとするマイクロプログラムのアドレスや，ラベル名などのアトミックなデータ，及びタームの類である。これらのデータは ESPスロットやSIMPOSの提供するプールサブシステム〔3〕 の機能を利用して実現することができた。ここでは以下にターム(Stack object)の扱いについて述べる。

汎用アセンブラは，アセンブルしようとするとマイクロプログラムの表記方法を equation と呼ばれる方法でマイクロプログラマが自由に変更する機能を提供している。

```
:: ( r=0 )  equ zero_flag
      if r=0  goto  label.
```

図 2 equ命令の利用例

今，仮りにマイクロプログラムのシンタックスとして zero_flag なるものがあったとする。図 2に示したように r=0 と zero_flag を equate することで，それ以後 r=0 と zero_flag は同じ意味としてアセンブラは解釈する。

即ち，アセンブルしようとするマイクロプログラム中の図 3に示したような宣言から，a を keyとして検索すればb が得られるようなテーブルをアセンブラは作成する必要がある。

```
::  a   equ   b .
```

図 3 equ命令による定義

一般に，a,b は stack vector なのでファイルから読み込んだa,b ともに heap object に変換した後に、SIMPOSのプールサブシステムを利用してテーブルを作成した。

(2) マクロ展開

前処理プログラムは機械定義から ESP のプログラムを自動生成するプログラムである。機械定義中には，任意のオペレータ宣言がなされる。このため，これらのオペレータが ESP で規定しているマクロ展開の対象となり，生成されたプログラムが意図と異るものになる事がある。

これらを防ぐため，前処理プログラム中では，マクロ展開を抑止するために，term を必要に応じてクォートする必要があった。

The Conversion of the General Purpose Assembler to PSI

Hirokazu Tateno [1] Shigeyuki Takagi [2]

1) MITSUBISHI ELECTRIC 2) ICOT

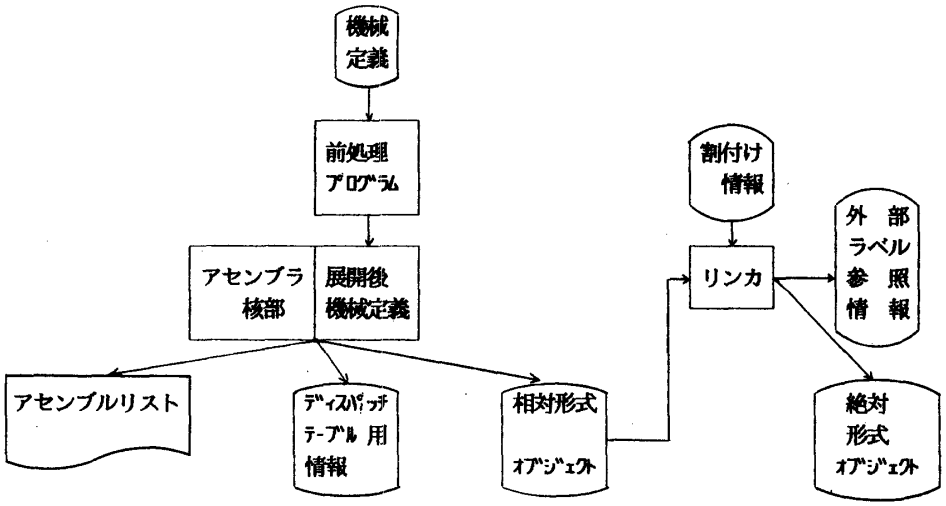

図1　汎用アセンブラの構成

4．評価と高速化

アセンブラの実行速度を測定した。測定には,PSIの組込述語 add　のマイクロプログラムソース(96steps) 及びガーベージコレクタの１モジュール gc_mark(1263steps) を利用した。

表 2にその結果を示す。DEC 20上のアセンブラと比べ約４倍の実行時間がかかり，応答速度においても約２倍の差がある。(PSI ではCPU 時間，応答速度ともほとんど同じである) 。そこで移植の頁で述べた図 2のテーブルの扱い方を変更することを考えた。

表 2　応答速度

	PSI	DEC 20 (cpu time)
add	122秒	66秒　(32秒)
gc_mark	1597秒	720秒　(620秒)

注) (　)内はcpu 時間

アセンブラはマイクロプログラムを一行読み込むごとに数回～10数回テーブルを検索する。このテーブルの検索には，図 4に示すように毎回termからstringへの変換処理が必要である。

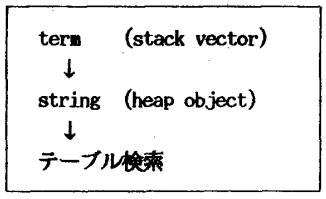

図 4テーブル検索の前処理

そこで，図 3のような一行を読むたびにそれを SIMPOS の提供する標準入出力の機能を利用し buffring しておき，全ての equation 部分が終了した時点で buffer から equation　の情報を読み出し，動的にライブラリへ登録するように変更した。

この変更により，アセンブル時間は表 3に示すように高速化された。

表 3　　高速化の効果　(応答速度)

	ＰＳＩ		DEC 20
	高速化前	高速化後	
add	122秒	70秒	66秒
gc_mark	1597秒	870秒	720秒

5．おわりに

今回の報告では，汎用アセンブラの移植において，得られた知識を紹介した。　又，移植した汎用アセンブラは，DEC 20上のそれとほぼ同等の実行性能を示し一応の目標を達成した。

現在，移植した汎用アセンブラを利用し，ＰＳＩ—II用の専用アセンブラもＰＳＩ上に準備されている。

移植作業中，ＳＩＭＰＯＳの開発メンバーから多くの有益な助言を頂いたことを感謝しております。

参考資料

〔1〕汎用型マイクロプログラム・アセンブラ
高木茂行　TR-021

〔2〕ESP Reference Manual
近山 隆　TR-044

〔3〕SIMPOS使用説明書

〔4〕KLØ 組込述語説明書