D 4　大型機によるミニコンのプログラミング・サポート

浅井　清，稲見泰生，小沼吉男（日本原子力研究所）

1　はじめに

筆者らの勤務先でもミニコンが実験データの収集，解析装置として利用される傾向が最近とくに顕著で，実験装置，研究室に配置されるミニコンの数が増え，種類も多様になってきた。これらのミニコンを導入する側の一番大きな悩みは，導入されたミニコンのソフトウェアの作成と保守をどうするかということである。ミニコンを導入した研究室は，もともとその作業をおこなうための人手をもっていないうえに，ミニコンの導入によってむしろ省力化が期待されるのが普通である。この問題に対するひとつの解決策としてミニコン間での（ソース言語レベルでの）ソフトウェアの共有ということが考えられる。Fortran語はソフトウェアの記述という点からみると適当ではない。PL360〔1〕を変形したGPL〔2〕は，このソフトウェアの記述という目的に合っているが，そのコンパイラは大型計算機で実現されている。ミニコン単位でコンパイラを作成すると人手がかかりすぎるが，大型機のコンパイラの命令生成ルーチンに置き換えるだけですむのなら理想的である。現実には命令生成ルーチン以外も手直しが必要であるが，ソース言語の互換性について適当な妥協をすれば，手直しの工数は少ない。数カ月のうちに筆者らの勤務先でもいくつかのミニコンが大型機とオンラインとで接続され，また同時にタイム・シェアリング・サービスが実施される予定である。このときひとつのコンパイラで大型機とミニコンとをサービスできれば，プログラムの翻訳とディバッグを大型機で，実行はミニコンでおこなうことができる。また，タイム・シェアリング用タイプライタを使ってミニコンのプログラムの作成，編集，ディバック，実行が可能となり，多くのミニコンが大型機を仲介として，ソフトウェアを共有することができる。大型機のコンパイラはミニコンに，そのミニコンの相対形式プログラムを送り返す方式をとる。

2　GPL言語とそのコンパイラ

2.1　GPL言語の特徴

GPLはPL360 を変形したALGOL60－like のソフトウェア記述言語である。

GPL の特徴を列記すると，

(1)　インデックス・レジスタを陽に指定できる。

(2)　演算用レジスタの上での一連の操作を1つの文で記述できる。

(3)　機械語を使用できる。

(4)　Fortran語の共通領域文が使え，Fortran 語で書かれたプログラムとの結合が可能である。

(5)　システム・マクロ，Fortranのライブラリ・ルーチンを呼び出すことができる。

(6)　入出力文を持たない。

2.2　コンパイラの構成

Fig. 2.1はコンパイラの流れ図を示す。

表2.1はコンパイラの作り出す擬似的な命令を示す。

Fig.2.2 は擬似命令の形式を示している。

Fig.2.2 からわかるとおり擬似命令は2番地方式である。セクション・ベースはFortranの共通領域名でもよく；

またベース・レジスタ名であってもよい。したがって番地はベース・レジスタ，インデックス・レジスタ，ベース・レジスタからの変位を指定することによってきまる。

3. ミニコンに要求される機能

ＧＰＬで書かれたソース・レベルでの言語の互換性を実現するためには，対象となるミニコンに次の機能が必要である。

(1) ＡＣ（Accumulator），ＭＱ（Multiplier－Quotient）レジスタ，

(2) 整数の加減乗除算の機能，

(3) 1個以上のインデックス・レジスタ，

(4) すべての番地が直接指定可能。

上記の(1)で乗数，商が常にＭＱに在る必要はなく，ＡＣに入るミニコンであってもよい。

このような観点から筆者らの手近にあるミニコンのいくつかについてしらべてみた（表3.1）。

項目 / 機種	ＡＣ，ＭＱ	加減乗除	インデックス	直接指定可能番地	最大記憶容量	語　　長
ＥＡＩ６４０	有	可	1	２５６語	３２Ｋ語	１６ビット
ＰＤＰ－９	有	可	自動インデックス	１６Ｋ語	３２Ｋ語	１８ビット
ＵＳＣ－３	有	可	3	８Ｋ語	１６Ｋ語	２０ビット
Ｕ－２００	汎用レジスタ	可	汎用レジスタ	６４Ｋバイト	６４Ｋバイト	１６ビット
TOSBAC-40	汎用レジスタ	可	汎用レジスタ	６４Ｋバイト	６４Ｋバイト	１６ビット

表　3.1

表3.1からわかるように，ＰＤＰ－９は陽にインデックス・レジスタを指定できないので上記の条件(3)に合わない，ＥＡＩ６４０は条件(4)に合わない。ＵＳＣ－３は８Ｋ語までが直接指定可能で，その範囲なら条件(4)に合う。

4. ソース言語レベルでの互換性

ＧＰＬのソース・レベルでの互換性をさまたげる要因を列挙すると，つぎのようなものがある。

(1) 文字列，ビット列の表現方法の違い，

(2) インデックス・レジスタ数の違い，

(3) バイト番地，語番地など番地指定の違い，

(4) 算術演算におけるレジスタ使用法の違い，

(5) サブルーチンへの飛び越し方法の違い，

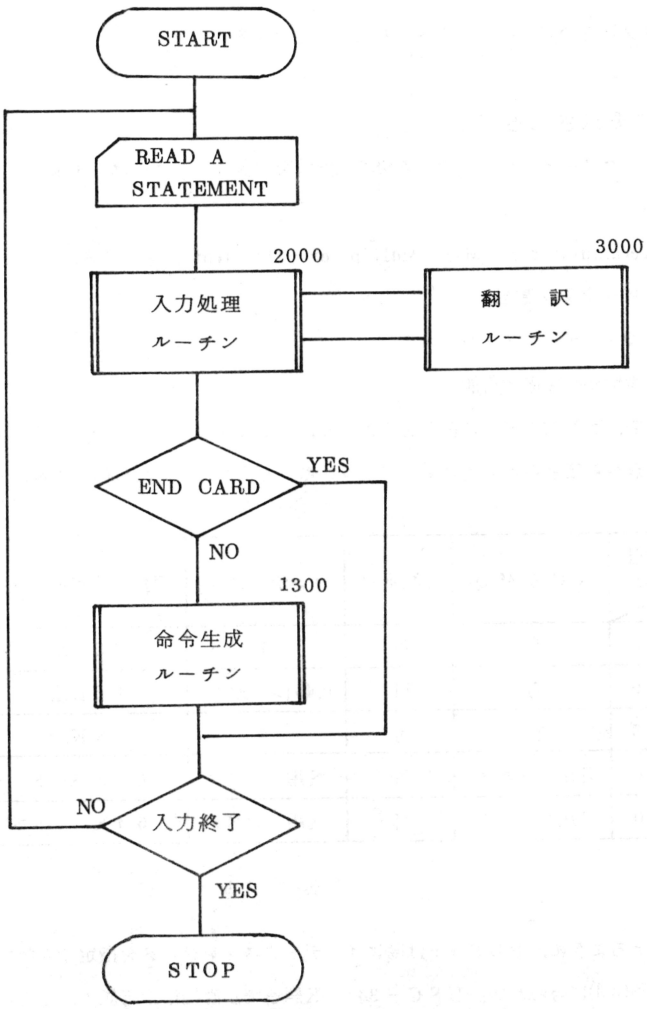
(6) ソース言語中に指定されたシステム・ルーチン，ライブラリ・ルーチンの名称の違い，

(7) ソース言語中に指定された機械語表現の違い，

(8) システム・ルーチンのパラメータ受け渡し方法の違い，

(9) ソース言語中の絶対番地の指定。

これらの問題の良い解決策は見つからないが，現在筆者らが考えている対応策はつぎのようなものである。



注）右肩数字はFORTRAN
ソース・カード枚数

fig. 2. 1

命　令	X	Y

X，Y；レジスタ番号，またはベクトルZへのポインタ

Z＝（Xi ， S.B. , DISPL, MODE）

Xi ：インデックス・レジスタ

S.B.：セクション・ベース

DISPL：セクション・ベースからの変位

MODE：演算対象となるセルの形式（整数，実数など）

fig. 2. 2

ADD	Add
SUB	Subtract
MPY	Multiply
DIV	Divide
UAD	Unnormalized add
USB	Unnormalized subtract
AND	And
OR	Or
EOR	Exclusive or
NOT	Not
ABS	Set plus
NEG	Negate
NEGABS	Set plus and negate
SHLA	Shift left arithmetically
SHRA	Shift right arithmetically
SHLL	Shift left logically
SHRL	Shift right logically
LR	Load register
LAR	Load register with address
XRR	Exchange registers
STR	Store register
ARI	Add register immediately
SRI	Subtract register immediately
JUMP	Unconditional jump
JZE	Jump if zero condition
JPL	Jump if plus condition
JMI	Jump if minus condition
JNE	Jump if nonzero condition
TOV	Test if overflow
TEQ	Test if equal
TLT	Test if less than
TGT	Test if greater than
TNE	Test if not equal
TGE	Test if greater than or equal
TLE	Test if less than or equal
SXJ	Set index and jump
SXT	Set index and transfer
LAD	Load address part
STAD	Store address part
KSTA	Masking store
NOP	No operation
PZE	Plus zero

表　2. 1

(1) について

ＧＰＬはビット列を定義できない。文字列は，１変数（１セル，通常は１語）について４文字までのものを認めている。大型機の場合は１語に４文字が普通であるが，ミニコンでは１語に２文字が多い。したがって長さ３以上の文字列が定義されているＧＰＬのプログラムは互換性がない。また文字列が数字によって表現されている場合も，そのプログラムは互換性を失なう。その数字が数値データを示しているのか，あるいはＡＳＣＩＩ，またはＥＢＣＤＩＣなどのコード系の文字を表現しているのか見当がつかないからである。倍長精度が指定されている変数については４文字までの文字列を定義できる。コンパイラは，該当するミニコンの記憶装置内での文字表現に合わせて文字列を翻訳する。

(2) について

ミニコンで使用可能なインデックス・レジスタ数をｍ，ソース言語中に出現しているインデックス・レジスタ数をｎとする。ｍ≧ｎのときはよい。ｍ＜ｎのときは，ソース言語中に出現したインデックス・レジスタx_i（ｍ≦ｉ≦ｎ）に対し，常にｍ番目のインデックス・レジスタx_mを使用する。コンパイラは変数名x_m，……，x_nを自動的に定義し，ここにインデックス・レジスタの内容を退避させるよう命令を作り出す。この操作はひとつの文（ＧＰＬ文）を単位としておこなう。このときサブルーチン・リンクに対しインデックス・レジスタx_m，……，x_nの存在を仮定したプログラムは互換性を失なう。

(3) について

ＧＰＬ文

Ｐ（x_i＋１）＝０　＄　if　x_i.ＬＴ.５　then　x_i＝Ｑ（x_j）　＄

においてＰが１語長の変数として宣言されているとしよう。これらの文が語番地の計算機で実行されるときには，Ｐの添字式（x_i＋１）と，インデックス・レジスタx_iの値は，そのまま解釈してよい。１語長が２バイト，または４バイトの計算機では，（x_i＋１）において数字１は２バイト，または４バイトの変位を示し，インデックス・レジスタx_iの値は，添字式が計算されるまえに２倍，または４倍されなければならない。ところが条件式x_i.ＬＴ.５では，x_iはもとの値を保持していなければならない。この変位とインデックス・レジスタの値の調整は，プログラマではなく，コンパイラが命令を生成することによっておこなわなければならない。インデックス・レジスタの値の調整はひとつの文を単位としておこなう。

(4) について

算術演算がレジスタＡＣ，ＭＱを使って実行される計算機ではコンパイラに変更がない。加減算，論理和，論理積などがメモリの上で実行される計算機については，これらの演算を重視して考える。この種のミニコンでは，つぎのＧＰＬ文

Ａ（x１）＝Ｂ（x２）＋Ｃ＊Ｄ　shla　５

において，式Ｂ（x２）＋Ｃまではメモリ間の演算として翻訳し，記号＊が出現したときに式Ｂ（x２）＋Ｃのために翻訳された命令群を手直しする。すなわち記号＊の出現によってＡＣ，ＭＱレジスタの存在を仮定し，すでに翻訳されたこの文についての命令群を手直しする。いくつかの汎用レジスタをもつミニコンでは，そのうちの２つをＡＣ，ＭＱレジスタとして固定して使用する。ソフトウェアの記述では加減算の頻度が大きいということから加減算を重視している。

(5) について

ＧＰＬは入出力文をもたない。入出力の操作はＦｏｒｔｒａｎの入出力の機能を借用するのが一般ユーザにとっては

もっとも簡単な方法である。このような点からもＧＰＬのサブルーチン結合法は，該当するミニコンのもつ Fortran システムのサブルーチン結合法と同じでなければならない。手近にあるいくつかのミニコン（ EAI 640，PDP-9，TOSBAC40，USC-3 ）を使って簡単な Fortran プログラムを翻訳してみた。サブルーチンへの飛び越し，復帰を間接アドレス方式でおこなうものと，レジスタを使っておこなうものがある。間接アドレスによる復帰は現在の FACOM 230-60版 GPLコンパイラの命令生成方法と合わないので，この部分はコンパイラの修正が必要である。パラメータの受け渡し方法はほとんど同じである。この部分はミニコン Fortran の翻訳方法に依存するから一定の方式で押し切ることはできないが，翻訳の方法に共通性があるようだ。

(6) について

FACOM 230-60 版 ＧＰＬではＳ．××…，Ｆ．××… と頭の2字がＳ．，またはＦ．ではじまるものをシステム・ルーチンと見なしている。これ以外の名前についてはＧＰＬの文法に宣言をつけ加えて，それがシステム・ルーチンとして翻訳されるようにする。

(7)，(8)，(9) については互換性を論じることはできない。これらが使われているプログラムはその部分を書き換えなければならない。

表 4.1 は以上の項目についてコンパイラを修正したときに必要となる工数を Fortran のスティトメント数で示している。図中で×印はソース言語レベルでも互換性が考えられないものを，また？（ｎ）はプログラムの書き方によっては（コンパイラをｎスティトメント修正することによって）互換性が保たれるものを示している。

ＴＳＳ編集とあるのは，大型機のタイム・シエアリング用タイプライタを利用してＧＰＬプログラム作成し，大型機の大記憶に保存，更新，消去するなどの作業プログラムを示す。このプログラムのための必要スティトメント数は，同じ目的のために作られた既存の Fortran プログラムから推定した。命令生成とあるのは各ミニコンのための命令生成ルーチンを示す。FACOM 230-60版 ＧＰＬ の命令生成ルーチンは約１３００スティトメントである。ミニコンの命令生成も同じ程度の手数が必要である。

項　　目	工数（枚数）
(1) 文字，ビット列	？(20)
(2) インデックス数	40
(3) 番地指定	40
(4) 算術演算	40
(5) サブルーチン・ジャンプ	20
(6) システム・ルーチン	20
(7) 機械語表現	×
(8) パラメータ授受	×
(9) 絶対番地指定	×
(10) ＴＳＳ編集	1000
(11) 命令生成	1300

表　4.1

5. おわりに

筆者らはFACOM 230-60版ＧＰＬでコンパイラなどの処理プログラム・レベルのソフトウェアを記述している。その経験では，ＧＰＬで書かれたプログラムとアセンブラ語で書かれたプログラムとの命令数，実行時間の比は1.1：1となる。1.1を1.0に近でけようとするとＧＰＬプログラム中に機械語表現が多くなり，アセンブラ語で書いたプログラムに近くなる。簡単なＧＰＬ文のいくつかを上記4.の対応策にしたがうコンパイラを想定して翻訳してみると，命令数の対アセンブラ比は1.0～1.2となる。この比が1.1程度になれば実用になる。この数カ月のうちにU200，USC-3，PDP-9などが大型機とオンラインで結ばれる予定であるが，ミニコンＧＰＬの適用可能性をみるために，現在はＵ200の命令生成ルーチンのコーディングをおこなっている。

ミニコンの場合にはアプリケィションのソフトウェアまで含めて発注される傾向が強い。そのためにミニコンＧＰＬの需要がどの程度になるかは今のところはっきりしていない。

参考文献

1) Wirth, N. ： PL 360, A Programming Language for the 360 Computers, ＪＡＣＭ, Vol. 15, No. 1, pp.37-74, Jan. 1968.

2) 浅井，富山 ： GPL - Genken Programming Language, JAERI-Mレポート 4762, 日本原子力研究所, 昭和47年2月.

3) Poole, P. C. and Waite, W. M. ： Portability and Adaptability, in Advance Course on Software Engineering, Beckman et al. (Eds.), pp. 183-277, Springer - Verlag, 1973.

4) EAI 640 Reference Handbook, Electronic Associates, Inc., April, 1971.

5) PDP-9 User Handbook F - 95, Digital Equipment Corporation.

6) ICD-507 (USC-3) 命令解説書, Tokyo Shibaura Electric Co. Ltd.

7) FACOM U-200 ハードウェア解説編, 富士通, 昭和48年1月.

8) TOSBAC-40 プログラミング説明書, OSアセンブラ編, 東京芝浦電気株式会社.

本 PDF ファイルは 1965 年発行の「第 6 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html