CL 9－3
(1977. 3. 18)

SNOBOL4処理系の移植経験

東京工業大学 情報科学科
白浜律雄

概要　文字列処理用言語SNOBOL4の標準処理系を中型機FACOM230－45Sに移植した経験を述べる．

関連語　SNOBOL4，portability，macro assembler

0. はじめに

SNOBOL4は，標準的な文字列処理用語であり，応用次第では非常に強力な道具となり得る。この言語については、言語の設計と並行して，特定の計算機に依存しない形の処理系（MAINBOLと呼ぶ）が開発されており，言語処理系そのものの開発に要する努力の総和と比べれば，はるかに少ない努力で任意の機種に移し換え（移植）ができるようになっている。本稿ではMAINBOLを国産中型機FACOM230-45S（以下45Sと略す）に移植した経験と，その際得られた知見を述べる。

MAINBOLは，仮想計算機SILMのアセンブラ語とも言うべきSIL語で書かれたプログラムと，構文表とから成り，これらに各機種に即した変形を施すことによって，その機種のためのSNOBOL4処理系が作られる．我々の場合に即して言うならば，(1) SILMの131種の命令の各々について，それを実現する45Sの命令列を定め，SIL語によるMAINBOLを対応する45Sの命令列に組織的に変換する．この作業にはFASP（45Sのアセンブラ）のマクロ処理機能を用いる．(2)構文表を適当なプログラムによってFASP用のデータ定義及び附随する命令群に変換する．(3)なお，(1)のうち入出力に関しては45SのFortran用入出力ルーチンへの呼び出し列を作り出す必要がある．

1. SNOBOL4言語

SNOBOL4は，Griswold（現アリゾナ大学）によって設計された文字列処理用語である。言語の解説は本稿の目的ではないので，それについては，文献[1][2]を参照されたい。ここでは例題によって，この言語の文字列処理における力の片鱗を紹介する．次ページの図1は，文章を読み込んで，その中に現われる単語の出現回数を数えるプログラムである。尚，図2の出力結果の最初の3行が入力された文章である．

```
                LETTER = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
                WPAT   = BREAK(LETTER) SPAN(LETTER) . WORD
                COUNT  = TABLE()
READ            TEXT   = INPUT                              :F(PRINT)
                OUTPUT = TEXT
NEXTW           TEXT  WPAT =                                :F(READ)
                COUNT<WORD> = COUNT<WORD> + 1               :(NEXTW)
PRINT           RESULT = CONVERT(COUNT,'ARRAY')             :F(NONE)
                OUTPUT =
                OUTPUT = 'WORD COUNT'
                OUTPUT =
                I      = 1
PRINTC          OUTPUT = RESULT<I,1> '-' RESULT<I,2>        :F(END)
                I      = I + 1                              :(PRINTC)
NONE            OUTPUT = 'THERE ARE NO WORDS'
END
```

図 1

```
YOU CAN FOOL SOME OF THE PEOPLE ALL OF THE TIME,
AND ALL OF THE PEOPLE SOME OF THE TIME; BUT
YOU CANNOT FOOL ALL OF THE PEOPLE ALL THE TIME.

WORD COUNT

YOU-2
CAN-1
FOOL-2
SOME-2
OF-5
THE-6
PEOPLE-3
ALL-4
TIME-3
AND-1
BUT-1
CANNOT-1
```

図 2

2. MAINBOL

MAINBOLは標準的なSNOBOL4処理系で，機械に依存しない形で記述されており，Griswoldらによって言語の設計と並行して開発された。

MAINBOLは，解釈実行方式の処理系である。つまり，翻訳ルーチンは，ソースプログラムを直接に機械語に翻訳するのではなく，先ずMAINBOL独自の内部コードに変換し，次いで解釈ルーチンが，その内部コードを，逐次解釈実行していく。従って，PASCALコンパイラの移植の場合のような移植担当者が目標機種用のコード生成部を書き加える手間は必要ではない。翻訳ルーチンと解釈ルーチンは，ともにSILで記述され，他の支援ルーチン群とともに，一つのプログラムを成し，翻訳と解釈実行は，1ジョブステップのうちに行なわれる。解釈中に翻訳ルーチンが呼び出されることがあるため，オーバーレイは容易ではない。

MAINBOLの移植担当者は，ソースプログラム中のどの部分が翻訳ルーチンで，

どこが解釈ルーチンであるのかを含め，本来MAINBOLの処理論理を知る必要はないので，ここでは，これ以上深入りはしない。詳しいことは，文献[3]を参照されたい。

3. SIL 語

SIL（SNOBOL4 Implementation Language）語はMAINBOLの記述言語であり，仮想計算機（SILM）のアセンブラ語とも言うべきものである。その形式は，IBM360のアセンブラの仕様とよく似ている。1行は80字から成り，継続行というものはない。第1桁がアスタリスク（＊）である行は注釈行である。文は3つのフィールド（名札部，命令部，オペランド部）から成り，位置は固定されている。（図3参照）オペランドはコンマ（，）で区切られ，空白で終わる（この空白に続けて注釈を書くことができる）。1つのオペランドが，カッコ（'('，')'）で，くくられたリストであることもある（ただし1重まで）。名前（マクロ名を含む）の長さは6文字以下である。

名札	空白	命令	空白	オペランドおよび注釈	空白	一連番号
1－6	7	8－13	14 15	16－71	72	73－80

図3

```
*       INTERNAL GOTO
*
GOTO    PROC    ,                       INTERPRETER GOTO PROCEDURE
        INCRA   OCICL,DESCR             INCREMENT OFFSET
        GETD    OCICL,OCBSCL,OCICL      GET OFFSET
        BRANCH  RTNUL3                  RETURN
```

左の図4は，MAINBOLの一部である。

図4

FASP（45Sのアセンブラ）は，第1桁が空白以外の文字であると無視するため，注釈以外の行は第1桁から第13桁までを1文字右へずらす前処理を行なった。

4. 移植の原材料

次に挙げるものをGriswold氏から当方希望の形式（9トラック，1600bpi，EBCDICコード，80バイト/レコード，800バイト/ブロック，ラベル無し）の磁気テープで頂いた。①MAINBOLソースプログラム（3.10版） ②変更用エディタ ③①を3.11版にするための変更情報 ④360用のマクロ定義集 ⑤360用のサブルーチン群 ⑥①をSNOBOL Xにするための変更情報 ⑦①に拡張DEFINEを加える変更情報 ⑧①にTWTH機能を加える変更情報。 3.10版とは3回の大改訂，10回の小改訂を受けた版であることを示す。変更は②のエディタで行なわれる。文献[5] ②はSNOBOL4で書かれているが単純なものなので，他の言語（例えばCobol，Fortran）で容易に作成できる。注意すべきことは，変更が先頭からの行数に基づいて行なわれる点である。移植担当者がソースプログラムに変更を加えて使用した場合，配布元よりの以後の変更を自動的に吸収することが難しくなる。

①と③を②に入力して、最新版である3.11版が得られる。④及び⑤は参考として加えられたものである。⑥，⑦，⑧は特殊な機能を持たせるための変更情報である。移植の原材料は何れかの版のソースプログラムと、後に記述する構文表である。構文表は⑤の中に注釈の形で散在していた。量的には、ソースプログラムは約6,500行（そのうち 約1,800行は注釈），構文表は25種合計約200行である。

5. 移植の条件

- MAINBOLの実行には150Kバイト以上の領域が必要である。これは大まかな目安であり、機種により変わりうる。
- アセンブラにマクロ展開機能が備わっている。
- 移植担当者は、目標機種のハードウェアを熟知し、マクロ展開機能の利用法を知っている。
- Fortran の入出力ルーチン群の呼び出し法がわかっている。

これらは、移植がスムーズに行くための一応の目安であり、移植に要する手間、処理速度との兼ね合いで変わり得る。主記憶が不足している場合には、ソフトウェアによる仮想記憶化，threaded codeの利用などが考えられる。アセンブラにマクロ展開機能がなくとも、移植担当者が処理系（マクロ展開のみ，またはSILアセンブラ）を作ることは可能である。Fortranの入出力ルーチンを利用できなくとも、作成することは不可能ではない。それらの場合には、作業量が倍増することになるであろう。移植の難易に関わる要素としては、もろもろのプログラミングトゥールが挙げられる。45Sへの移植においては、キャラクタディスプレイを用いるオンライン エディタ，SNOBOL 3処理系の存在は、作業を楽にするために大きく貢献した。

6. 解説書

文献[4]が不可欠な手引書であり、SIL語の解説をしている。文献[3]は、MAINBOL の総括的解説書である。ただし、SILマクロの名前を初めとして、その用語が他の全ての文献（ソースプログラムを含む）と異なるため、旧用語と新用語の対照表が発行されている。（文献[6]）

7. FACOM 230-45S のアーキテクチャ

基本となるデータ単位(語)は16ビットである。16ビットのジェネラルレジスタ(R0～R8)を8コ持ち、R1～R3は、インデックス レジスタとしても使用できる。分岐命令を実行すると、その次の命令の番地がR0にセットされる。レジスタ間の演算命令はない。アドレス指定法には直接，間接，インデックス修飾，間接かつ インデックス修飾（間接指定が先に計算される）の4通りがある。主記憶は仮想記憶ではないが、マッピングレジスタ によって、論理アドレスから実アドレスへの変換を行なう。最大64K語までの論理的なメモリーバンクを4つ（PN0～PN3）使用できるが、ユーザはPN2とPN3 は利用できず、PN0はデータの格納，PN1 はプログラムの格納に用いる習慣になっている。多くの命令は2語から成り、全領域を直接参照す

ることができる．固定小数点演算データは1語または2語、浮動小数演算のデータは，2語または4語である．バイトを演算の対象とする命令群があり、これらにおいては、インデックス修飾を指定すると、インデックスの単位がバイトになる．詳細については文献[8]を参照して頂きたい．我々が用いた45Sの実装主記憶容量は，移植着手時には，224Kバイトであり、このうち64Kバイトはオペレーティングシステムが占有するため，ユーザ領域は最大160Kバイトしか取れず しかも、そのうちの16Kバイトはジョブ管理用領域として使われるため，MAINBOLの実行には非常に厳しい環境であった．その後，32Kバイト増設されたが，一応146Kバイトで実行できるようになっている．

8．　移植作業

移植作業とは，文献[4]に従って，SILの131種の各命令を実現する45Sの命令の組み合わせを見極め，MAINBOLを45Sの命令列に変換することである．原型と変換法を定義しておけば，組織的書き換えはFASPのマクロ展開機能が行なう．マクロの定義は，基本である3種のデータ（ディスクリプタ，スペシファイア，文字列）の表現と，それらに対する操作とを見比べつつ決定する．データ構造の決定から自然に決まるマクロも多いが，何しろ数が多い．また、なすべきことがわかっていても、アセンブラの能力不足を補う方法を考えねばならないこともあった．

8-1．　データの表現

8-1-1　ディスクリプタ

SILの基本データ単位はディスクリプタと呼ばれ，3つのフィールドから成り，マクロDESCRによって生成される．

- Aフィールド　他のディスクリプタ，スペシファイア，文字列などのアドレス，固定小数点(整数)，浮動小数点(実数)を記憶する．実数を扱うマクロはあるが，アドレスと整数は区別されず同じマクロで扱われる．
- Fフィールド　独立してセット，リセットされるフラグを格納する．MAINBOLでは5種類のフラグを使用している．2つのフラグの両方をオンにすることをSILでは、例えばTTL+MARKのように書く．TTLとMARKの定義はMAINBOL中にはなく、移植担当者がファイルPARMS中に定義するとマクロCOPYによって組み込まれる．他にファイルはMLINK，MDATAがあり、他のモジュールとの連結，作業データ定義を行うためのものである．
- Vフィールド　ディスクリプタの種別を識別する（小さい正の）整数、ディスクリプタから成る集合体の大きさ，文字列の長さを記憶する．

フィールド単位，フラグ単位の操作が短かいコードで実現できるように、フィールドの大きさ，位置を決める必要がある．45S版では，Aフィールドに2語，FとVフィールドに各々1語を割り当てる．（図5参照）　この構成であれば、1ディスクリプタの主記憶-レジスタ間転送は1命令で行なえ，フィールドの分離も容易である．

8-1-2　スペシファイア

スペシファイア は文字列へのポインタで、A.F.V.O.L の5つのフィールドから成り、2ディスクリプタ分の領域を占める。一方は A.F.V の3フィールドを持つ普通のディスクリプタ，他方は L フィールドと O フィールドに分けられる。L は文字列の長さ、O は A フィールドの指す位置から文字列の先頭までの位置のずれ（単位は1文字）である。このディスクリプタの，普通であれば F フィールドに当たる位置は常に全フラグがオフでなければならない。（図6参照）

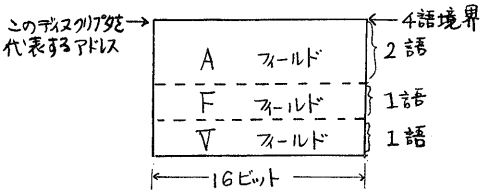


図5

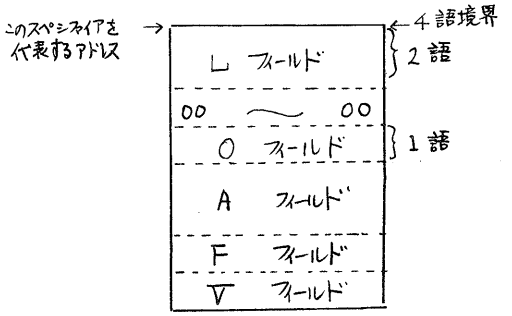


図6

8-1-3　文字列

1語に2文字ずつ詰める。

図7はスペシファイアが文字列 SNOBOL4 を指している様子である。

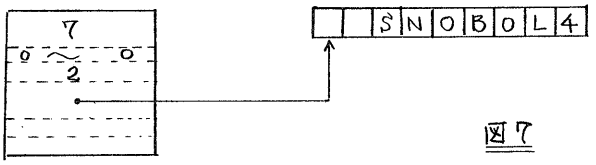


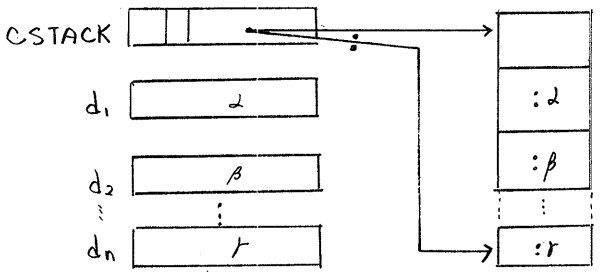
図7

8-2.　レジスタの使い方

SIL のマクロは主記憶上のデータに対する操作という形をとり，レジスタという概念はない。ただ、スタック操作を行なう一群のマクロがあり、現旧2つのスタックポインタ（CSTACK, OSTACK）というディスクリプタを介して操作の説明がなされているが、この2つはソースプログラム中に定義されているわけではなく、レジスタ上にとるか、ファイル MDATA 中に定義するかは移植担当者の自由である。45S には、インデックスレジスタが3個しかないため、CSTACK（スタック関係マクロ全てが参照）のみをレジスタR1上に置き、OSTACK はディスクリプタとして定義してある。

例として、PUSH というマクロを取り上げる。PUSH のオペランドはディスクリプタのリストであり、それら全てをスタック上に積み、CSTACK を更新する。積む順番は守らねばならない。（図8では箱はディスクリプタを示し、区切りは左から V, F, A, フィールドを示す。記法については文献[10]を参照されたい。）

形 label PUSH $(d_1, d_2, \cdots, d_n)$



(註) ':'は'になる'という意味である.

図 8

45Sでは右の図9のように定義してある.

図9

```
                MACRO     PUSH
*LABEL  PUSH    *DESCRS
          GBLA      *PB,*DB
          LCLA      *I,*L,*N
*N        SETA      N'*DESCRS
*LABEL              AXI       *N*DESCR,1
                    LI        (STLIMIT),1,2
                    BNM       OVER
.A        ANOP
*I        SETA      *I+1
                    LQ        *DESCRS(*I)
*L        SETA      *I-*N
                    STQ       -*L*DESCR,1,,*DB
          AIF       (*I LT *N).A
.MEND     MEND      PUSH
```

8-3. 構文表

構文表はマクロ CLERTB, PLUGTB, STREAMで用いられ, 表の構造には, これらだけが関与し, ディスクリプタとは異なる形でよい. 文献[4]で syntax table と呼んでいるが実体はシラブルを切り取るための表であり. STREAMが これを行なう. STREAMは;

label STREAM $S1, $S2, table, error, runout, success

という形をしている. $S1, $S2はスペシファイアで, $S2 が指す文字列をtableに従って走査して, $S1 は読まれた文字列を指し, $S2 は残りを指すようにする. error, runout, success は MAINBOL中のわれであり. 表でERROR の指定がされている文字に来るとerrorへ, STOP またはSTOSH の指定の文字を見たらSUCCESS へ, $S2 が尽きたら runout へ制御を移すための指示である. 図10は整数を読み込むための表 INTGTBである. NUMBER は0~9の数字, TERMINATOR は. ';', '>', ')', 'u', DOTは'.'からなる文字の集合である. 各FORは. その右に書かれている動作を起こすべき集合を指定する. ELSEは. どの集合にも属さない文字に対する動作の定義である. PUTは, ソースプログラム中にあるSTYPEというディスクリプタに, カッコ内の値(ソースプログラム中で定義されている)をセットする指定である. CONTINは同じ表で, GOTOはカッコ内の表で, 次の文字を読むことの指定である. STOSHがあれば. 今の文字を読まなかったことにして 動作を終了し, STOP ならば, その文字を読んだところで動作を終え. success exitから抜ける.

45Sでは STREAMはサブルーチン呼び出しに展開される. 表は次のように表現し

ている．（図11）

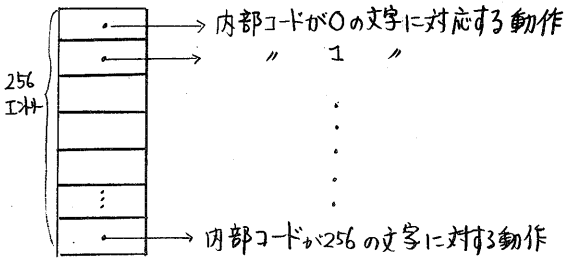


図11

```
BEGIN INTGTB
FOR(NUMBER) CONTIN
FOR(TERMINATOR) PUT(ILITYP) STOPSH
FOR(DOT) PUT(FLITYP) GOTO(FLITB)
ELSE ERROR
END INTGTB
```

図10

同じ動作は一ヶ所にまとめる．動作は；

PUT (α)	には	LI	α,, 7
CONTIN	には	B	S. CONTIN
GOTO (β)	には	LI	β,, 5
		B	S. GOTO
STOP	には	B	S. STOP
STOPSH	には	B	S. STOPSH
ERROR	には	B	S. ERROR

が対応する．

S. CONTIN等は，サブルーチン内の名札である．ポインタは45Sの性格を生かして1バイトで足りるように構成した．

SNOBOL3プログラムにより表の変形を行なった．25種の表が約3Kバイト強になった．

CLERTBは表（常にSNABTB）の全エントリをCONTIN, STOP, STOPSHまたはERRORの状態にするマクロである．

PLUGTBは表（常にSNABTB）のエントリーのうち、指定する文字に対応するエントリーのみをCONTIN, STOP, STOPSHまたはERRORにセットするマクロである．

文献[4]では，CONTINのみは0で表わすように決めてかかっているが，移植担当者の裁量でSTREAMとの間で矛盾が生じないようにすればよい．

8-4 入出力

SNOBOL4の言語そのものが，Fortranのフォーマット入出力を使うことになっており，MAINBOLでは更にメッセージの出力にもフォーマット出力を用いている．Fortranのサブルーチンを呼ぶことにするのも1つの解決であるが，引数の個数，型に応じて，何種類か作らねばならない．45S版では，Fortranの入出力ライブラリの呼び出し法を調べ，Fortranコンパイラが作り出すのと同じ形に展開している．Fortranの入出力ルーチン群は一体となって働くために，Fortranのサブモニタを組み込む必要があり，MAINBOL全体は，Fortranから呼び出される一つのサブルーチンの形に展開される（MAINBOLに制御が来る入口のマクロINIT，出口のENDEXをそのように展開する）．

9. 作業量

約6,500行のソースプログラムが展開されると、約27,000行にふくらみ、展開には45Sで約40分かかる（他のジョブは走っていない状態で。CPU使用時間は約8分。ただし、45Sのサイクルタイムは、0.7 μs/2byte, ディスクの平均待ち時間は65.2 ms）。アセンブルリストの出力は、500ページに及び、1,200行/分のラインプリンタでも印字に20分かかる。たびたび出力することはできないが、マクロ定義は、展開形の行数が少なくなるようにも気を配った。

移植作業は筆者が一人で行なった。文献[4]からSIL語を理解し、Fortranの入出力ルーチンの呼び出し方を調べるために1ヶ月弱を要した。特に構文表に関しては理解に手間どった。この間、不明箇所の解明のために360用のものを読み、またマクロ名で並べ直したリスト，クロスリファレンスリスト，名札で並べ直したリスト等を作製した。45S用MAINBOLの構想，設計、マクロ定義のコーディングに1ヶ月強を要した。構文表の45S表現の決定、展開プログラムの作成に、この期間の1/3程度を費やしている。当初45Sの主記憶量はMAINBOLの実行には足らないとみられたので、一時インライン展開ではない実現法も考慮するなど、領域量には神経を配った。テスト，デバッグには1ヶ月弱を要した。合計で約3ヶ月である。マクロは全部で約2,300行、この中から呼ばれるサブルーチン群が約500行、構文表を展開するプログラムが約100行程である。他に、マクロライブラリを作る補助として、制御文の組み込みなどを行なうFormatterをSNOBOL3で作成し利用した。

10. 困ったこと

FASPにはリテラルの機能がないため、Aフィールドに定数を加えるマクロの作成では、現われる定数を調べあげ、MDATAに定義しておき、それらを参照するようにしなければならなかった。同様に、ディスクリプタの配列を生成するマクロでは、個数が式で書かれていることもあるのに、FASPの定数定義命令では、大きさを数字で書くことしか許されず、得むなく、全式の値をあらかじめ計算しておき、マクロ定義中でその値と置き換えて、少しずつ生成するようにしてある。

11. まとめ

MAINBOLの移植は、ソースプログラムを既存のコンパイラに入力することによって終る程単純ではないが、新しく処理系を作成するよりは、はるかに楽である。このような移植が可能であるためには、処理系の記述法もさることながら、手引書の存在が重要である。文献[4]は180ページに及ぶものであるが、まだ不満な所があり、手引書のでき具合は移植の手間に大きく影響すると思われる。移行性のあるソフトウェアを開発、配布、維持していくことは並大抵のことではないというのが実感である。

謝辞

筆者の所属している研究室の長である木村泉助教授からは多大の御支援を頂いた．ここに感謝の意を表する．

参考文献

[1] Griswold, R.E., Poage, J.F., Polonsky, I.P., The SNOBOL 4 Programming Language, 2/e, Prentice-Hall, USA, 1971

[2] Griswold, R.E., Griswold, M.T., A SNOBOL 4 Primer, Prentice-Hall, USA, 1973

[3] Griswold, R.E., The Macro Implementation of SNOBOL 4, W.H. Freeman, 1972

[4] Griswold, R.E., A Guide to the Macro Implementation of SNOBOL 4 Bell Telephone Lab., 1972

[5] Griswold, R.E., Version 3.11 of the SIL Implementation of SNOBOL 4, "SNOBOL 4 Newsletter May 20, 1975" Univ. of Arizona, 1975

[6] Griswold, R.E., New Notation and Terminology for SNOBOL 4 Source Material, Bell Telephone Lab., 1971

[7] Griswold, R.E., Bibliography of Numbered SNOBOL 4 Documents May 1967 through May 1975, Univ. of Arizona, 1975

[8]「FACOM 230-45S ハードウェア総合解説編」 富士通

[9]「FACOM 230 OSII FASP解説編 I」 富士通

[10] 竹内郁雄「プログラミング画法の提案」情報処理学会第17回全国大会報告集, 1975