論　文

ストリング処理用仮想計算機 VC/S とその上での SNOBOL 3 の実現*

国　立　勉** 吉田雄二*** 福村晃夫***

Abstract

Most usual methods to implement string manipulation system are based upon developing a high level language for string manipulations and implementing it on some actual machines.

On the other hand, there is another approach in which a string manipulation system is organized on a virtual computer designed for that purpose.

On the latter point of view, we have designed a virtual computer VC/S possessing basic operations on strings. We have examined its characteristics by implementing the interpreter of SNOBOL 3 on it, and confirmed that VC/S is useful for string manipulations.

Although the interpreter thus far constructed is not efficient, higher efficiency can be expected by providing VC/S with more convenient facilities for language processings.

1. まえがき

現在広く使用されている電子計算機は，数値処理を主な目的として設計されていて，レジスタ群の構成とそれらの取り扱い形式，メモリの構成とアクセス方式，および命令群の構成等はいずれもこの目的に合わせられている．一方，電子計算機の用途は，現在では数値処理に限らず，非数値処理の分野においても広く用いられるに至っている．非数値処理に電子計算機が応用される場合には，種々の非数値処理手順を実現するためにソフトウェア上でさまざまな工夫が行われ，現実の電子計算機上で何等かの意味で仮想的な非数値処理用の機械が構成されているのが通常である．さらに，最近マイクロ・プログラミングを始めとする電子計算機アーキテクチャの進歩により，問題向き電子計算機の実現が可能となってきた．そして，これらのことがらにもとづいていくつかの具体的な機械が構成されている[1),2)]．

本論文では，このような観点のもとに，ストリング処理向き計算機を構成する場合の問題点について考察を加え，これにもとづいて実際に一つの機械を構成し，これをソフトウェアにより実現した．この計算機は VC/S (Virtual Computer for String Manipulation の略) と呼ばれる．本論文ではさらに，代表的なストリング処理言語 SNOBOL3 の処理系を VC/S の上で実現することにより，VC/S のもつ種々の機能がどのように応用されるかを示すとともに，その動作結果にもとづいて VC/S の評価を行う．ストリング処理用仮想計算機あるいはそれに近いものはいくつか発表されているが[3),4)]，これらはいずれも，FORTRAN コンパータ，あるいは SNOBOL 処理系の実現などの特別な用途に合わせて構成されたもので，汎用のストリング処理機械として構成された VC/S とは異なる．

2. VC/S の構成

2.1 基礎的考察

ストリング（文字列）処理においては，文字列同志の比較，文字列の合成・分解・転送などは基本的な演算と考えられるので，これらの演算は十分自由な形成で行える必要がある．さらに，これらの演算が文字列内の任意の文字位置を基準として行えるならば，文字

---

* Virtual Computer VC/S for String Manipulations and the implementation of SNOBOL 3 on it, by Tsutomu KUNITACHI, (N. T. T. Yokosuka Electrical Communication Laboratory), Yuuji YOSHIDA and Teruo FUKUMURA (Faculty of Engineering, Nagoya University)

** 日本電信電話公社横須賀電気通信研究所
*** 名古屋大学工学部情報工学科

列に対する走査の回数を減らすことができて文字列処理アルゴリズムの高速化が期待されるので，文字列上の位置を指示するポインタを考え，このポインタに基づいた基本的な演算を考える．また数値処理の場合のように，特定の演算レジスタを設定して処理される文字列を，この演算レジスタへ転送した後に処理を実行することは，本来文字列の長さに依存しない処理までが文字列の長さに比例する処理時間を必要とすることになり，効率の点で好ましくない．従って，文字列に対する処理は，記憶されている文字列自身に対して直接実行されることが望ましい．さらに，文字列を構成する文字として，外部表現をもつ文字の他に，外部表現上ではある文字列に対応するような特別な文字をも含めることは，例えば構文解析における非終端記号あるいは漢字のように極めて種類の多い文字の組を扱う場合に有効であると予想される．この考えに基づき，VC/S では仮想文字の機能[5]が導入されている．

2.2 機械の構成

VC/S の構成を **Fig. 1** に示す．

$\mathcal{MR}$：記憶装置．有限個の可変長セルの集合であり，各セルは文字列1個を格納することができて，名前表の1つのエントリを通して参照される．

$\mathcal{PC}$：プログラム・カウンタ．プログラム・ストリングにおいて次に実行される命令を指示するポインタで，直接これを操作する命令はない．

$\mathcal{CR}$：カレント・レジスタ．演算対象のセルを指定する．

$\mathcal{P}$：$\mathcal{CR}$ で指定されるセル上のポインタ．文字と文字とのギャップを指示する．

$\mathcal{LR}$：ロジカル・レジスタ．論理演算の結果（真，偽）を格納する．

$\mathcal{CR}$ スタック：$\mathcal{CR}$ の内容のスタックに使われるシステム・スタック．

名前表と仮想文字表の構成については **2.4** で述べる．

2.3 記述言語の構成

VC/S 上での処理手順は，アセンブラ・レベルの言語 $\mathcal{P}_1$ のプログラム（$\mathcal{P}_1$ ストリングと呼ぶ）として記述される．プログラムの実行は，演算対象の文字列を格納する $\mathcal{MR}$ セルに $\mathcal{CR}$ をセットすることと，そのセルに対する各種の命令に対応した動作の実行との繰り返しとして行われる．命令には，文字列に対する演算，ポインタの設定，移動などがある．

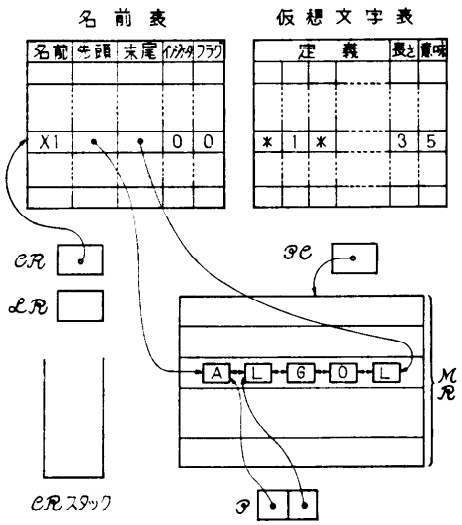

Fig. 1 Organization of VC/S.

$\mathcal{P}_1$ ストリングは **Table 1** のような構文をもつ．ここで，<変数>の¥と@は，前者が変数の値を名前とする間接参照，後者は変数の値をセル番号とする間接参照を表わす．<仮想文字>は，2.1 で述べた趣旨から，使用可能な文字セットを拡張するために導入されたもので，＜と＞との記号によって囲まれた任意の文字列である．VC/S は仮想文字をも1つの文字として扱う．

$\mathcal{P}_1$ 命令の<命令部>は，VC/S の 40 種類の基本動作を決定する．以下にそれらの名前，$\mathcal{P}_1$ の記法，及び意味を述べる．ここで，$\mathcal{CR}$ のセットされたセル（$\mathcal{CR}$ セルと呼ぶ）の上の文字列が命令の実行により α から β に変化することを，$\alpha \vdash \beta$ で表わす．特に文字列上のポインタの位置が問題となる場合には，$x\mathcal{P}y$ によりポインタ $\mathcal{P}$ の左右にそれぞれ部分文字列 x, y

Table 1 Syntax of $\mathcal{P}_1$ string

<$\mathcal{P}_1$ ストリング> ::＝<マクロ命令定義>*<$\mathcal{P}_1$ 命令列><終止符>
<マクロ命令定義> ::＝〈DEFINE〉<名前>〔<引数並び>〕〔:<局所変数並び>〕
<区切記号><定義本体>〈DEFEND〉
<定義本体> ::＝<$\mathcal{P}_1$命令列> ::＝{[<ラベル>]<$\mathcal{P}_1$命令><区切記号>}*
<$\mathcal{P}_1$ 命令> ::＝<命令部>〔<オペランド部>〕
<オペランド部> ::＝<ストリング定数>|<変数>
<ストリング定数> ::＝'<文字>*'|<数字列>
<文字> ::＝<英数字>|<特殊記号>|<仮想文字>
<変数> ::＝〔¥|@〕<名前>
<区切記号> ::＝␣|,

(注) 構文の記述には概ね BNF を使用した．但し，非終端記号は<と>を使って記述されている．〔と〕は省略可能を示し，α^* または $\{\alpha\}^*$ は α の0回を含む無限回の繰り返しを意味する．

があることを表わす．また必要な場合にはオペランド部を大文字 M,N 等で示し，小文字 a,b 等で文字列中の1文字を示す．

〔1〕 $\mathcal{CR}$ セルの文字列に対する動作

1) right 〈R〉M; $x\mathcal{P}y \vdash x\mathcal{P}$; M←y
ポインタの右側を M に格納する．M を省略した場合は右側消去に対応する．

2) left 〈L〉M; $x\mathcal{P}y \vdash \mathcal{P}y$; M←x

3) clear 〈;〉 $x \vdash \mathcal{P}$

4) catenate ⌴M $x \vdash xy$
変数 M の値 y を連接する．

5) insert 〈INSRT〉M; $x\mathcal{P}y \vdash x\mathcal{P}zy$

6) replace 〈REPLC〉M; $x\mathcal{P}ay \vdash x\mathcal{P}by$

7) encode 〈ENCD〉 $x \vdash \mathcal{P}a$
x を仮想文字化する．

8) decode 〈DECD〉 $a \vdash \mathcal{P}x$

9),10),11),12) 〈+〉,〈−〉,〈*〉,〈/〉
$\mathcal{CR}$ セルの文字列とオペランドとの算術演算の結果が $\mathcal{CR}$ セルの内容となる．

13) get cell 〈GETC〉 $x \vdash \mathcal{P}a$
x を名前としてセルを確保し，$\mathcal{CR}$ セルにはセル番号 a が残る．

〔2〕 ポインタに対する動作

14) set $\mathcal{P}$ 〈SETP〉M; $x\mathcal{P}y \vdash x'\mathcal{P}y'$
M の値に $\mathcal{P}$ をセットする．

15) move $\mathcal{P}$ 〈P〉M; $x\mathcal{P}y \vdash x'\mathcal{P}y'$
右方を正として M の値の文字数分 $\mathcal{P}$ を移動する．

16) set $\mathcal{P}$ to right bound 〈PRB〉 $x\mathcal{P}y \vdash xy\mathcal{P}$
ポインタを右端へ移動する．

17) set $\mathcal{P}$ to left bound 〈PLB〉 $x\mathcal{P}y \vdash \mathcal{P}xy$

18) store $\mathcal{P}$ 〈STRP〉M; M→$\mathcal{P}$

〔3〕 複写命令

19) store 〈=〉M; M←x
$\mathcal{CR}$ セルの文字列を複写して M の値とする．

20) store substring between pointers
〈SUB 1〉M,N,K;
$\mathcal{CR}$ セル上のポインタ位置 M,N で囲まれた部分文字列を複写して K の値とする．

21) store substring by length
〈SUB 2〉L,K;
$\mathcal{CR}$ セルの $\mathcal{P}$ の位置から右に長さ L の部分文字列を複写して K の値とする．

〔4〕 $\mathcal{CR}$ のセッティングに関する命令

22) set $\mathcal{CR}$ 〈SETCR〉M;
M のセルを $\mathcal{CR}$ セルとする．

23) stack $\mathcal{CR}$ 〈(〉
$\mathcal{CR}$ セルのセル番号を $\mathcal{CR}$ スタックに入れる．

24) reset $\mathcal{CR}$ 〈)〉
$\mathcal{CR}$ スタックからとり出したセル番号のセルを $\mathcal{CR}$ セルとする．

〔5〕 論理演算と制御命令

25) right end? 〈REND?〉
ポインタ $\mathcal{P}$ が文字列の右端にあれば $\mathcal{LR}$←T, そうでなければ $\mathcal{LR}$←F とする．

26) left end? 〈LEND?〉

27) greater than? 〈GT?〉M;
$\mathcal{CR}$ セルの文字列が M より大きいならば $\mathcal{LR}$←T, そうでなければ $\mathcal{LR}$←F とする．

28) equal? 〈EQ?〉M;

29) match? 〈MAT?〉M,N;
$\mathcal{P}$ の右側に部分文字列 M が存在するならば，$\mathcal{LR}$←T, かつ対応する右位置を N の値とする．存在しなければ $\mathcal{LR}$←F とする．

30) alpha? 〈ALPHA?〉
$\mathcal{P}$ の右1文字が英字ならば $\mathcal{LR}$←T そうでなければ $\mathcal{LR}$←F とする．

31) digit? 〈DIGIT?〉

32) virtual character? 〈VCH?〉
$\mathcal{P}$ の右1文字が仮想文字ならば $\mathcal{LR}$←T そうでなければ $\mathcal{LR}$←F とする．

33) not 〈NOT〉
$\mathcal{LR}$ の内容を否定する．

34) conditional jump 〈CJ〉L;

35) unconditional jump 〈J〉L;

36) return 〈RTRN〉
マクロ命令からのリターンの実行をする．

37) stop 〈STOP〉

38) trace 〈TRCON〉M,N,K,L…;
N 個の変数 K,L… に対してトレース・モード M をセットする．

〔6〕 入出力命令

39) input 〈IN〉%
% 記号までの文字列を読み込み $\mathcal{CR}$ セルの値とする．% 記号を省略した時はカード1枚分読み込む．

40) output 〈OUT〉
$\mathcal{CR}$ セルの文字列の出力をする．

```
1       <DEFINE> <LOAD>;
2       LT1: <IN> <MAT?>'='; <CJ>LT11
3            <MAT?>'@','*'; <CJ>LT12 <J>LT20
4          LT10: <(> <TRANS>; <)>          LT11: <P>SU1 <RTHN>
5          LT12: <OUTLT1>NO1; <J>LT1
6       LT20: <SETCH>STNO <+>SU1 <OUTLT2>NO1,STNO;
7             <SETCH>NO1 <PRB> <P>=8
8          LT201: <P>=1 <MAT?>BLNK,'*'; <CJ>LT201
9             <P>SU2 <R>; <SETCH>NO2
10      LT2: <IN> <MAT?>'=','*'; <CJ>LT10 <MAT?>'@','*'; <CJ>LT22
11           <MAT?>PRD1,'*'; <CJ>LT23 <J>LT3
12         LT22: <OUTLT1>NO2; <J>LT2
13         LT23: <OUTLT2>NO2,STNO; <P>SU1 <L>; <PRB> <P>=8
14         LT231: <P>=1 <MAT?>BLNK,'*'; <CJ>LT231 <P>SU2 <R>;
15            <(> <SETCH>NO1, NO2 <)> <J>LT2
16      LT3: <TRANS>; <SETCH>NO2 <PLB> <R>NO1; <J>LT20
17      <DEFEND>
```

Fig. 2 An Example of VC/S program.

Fig. 2 に VC/S の $\mathcal{P}_1$ プログラム例を示す．これは，SNOBOL 3 のトランスレータの上位ルーチンをなす1つのマクロ命令〈LOAD〉の定義である．

2.4 VC/S の実現

VC/S システムは FORTRAN で記述されていて，アセンブラと制御ルーチンを合わせてカード約 1,000 枚，命令実行ルーチンがやはり約 1,000枚である．$\mathcal{MR}$ セルは可変長文字列の処理とポインタの移動を容易にするため一次元両方向リストで実現されている．

システムは仮想文字表と名前表とを保持している．前者は仮想文字の番号（システムが自動的に割り当てる）に対応する位置にその外部表現を格納する．後者は各エントリが $\mathcal{MR}$ セルに対応し，そのエントリの番号がセル番号となる．各エントリは，名前，セルの先頭及び末尾の語へのポインタ，インディケータ，トレース・フラグから成る (Fig. 1 参照)．

3. VC/S による SNOBOL 3 の実現

VC/S のもつ特徴，問題点を明らかにするためには，VC/S 上で高度な文字列処理システムを実現することがもっとも有効であると考えられる．ここでは VC/S 上に SNOBOL 3 を実現した．実現の方式は，SNOBOL 3 の言語的性格から，インタプリータ方式とした．

3.1 インタプリータ・コードの形式とトランスレータ

SNOBOL 3 のステートメントは大別して3種類ある．それぞれに対応するインタプリータ・コードは，ソースプログラムに近い形であり，先頭にステートメントの種類を表わすコード（仮想文字），末尾に終端記号を持つ．以下に各ステートメントに対応するインタプリータ・コードの形式を示す．

1) 代 入 文

〈ASGNS〉 op 1 op 2 区切2
〔go to field〕 終端記号

2) マッチング文

〈MATS〉 st. ref. pat. 区切2
〔replace field〕〔go to field〕 終端記号

3) 関数 call 文

〈CALL〉 関数 table index (arg 1 arg 2…)
〔go to field〕 終端記号

下線を付した部分は非終端記号を意味する．op 1, arg 1, st. ref. 等は次の a から e までの string expression かそれらの連接であり，pat. は string expression か，f から h までの string variable か，またはそれらの連接である．

string expression

a. literal : : =〈CONST〉 文字列 区切1
b. name : : =〈EXPL〉 table index
　|¥ string expression (b, c, e のいずれかに限る)
c. 関数 call : : =〈CALL〉 関数 table index (arg 1 arg 2…)
d. arithmetic operation : : =op 1 a. op. op 2
e. parenthetical grouping : : =(string expression)

string variable (s. v. と略す)

f. arbitrary s. v. : : =〈ASV〉 name
g. fixed length s. v. : : =〈FSV〉 name length specifier
h. balanced s. v. : : =〈BSV〉 name

ここに table index とは SNOBOL プログラム中に出てくる名前に対する，名前表でのアドレスを意味するものである．これは VC/S に配列機能がなく，変数・関数テーブルとラベル・テーブルとがそれぞれ1つの $\mathcal{MR}$ セルの文字列として表わされるためで，**Fig. 3** の↓の位置で決まるポインタの値が index となる．

SNOBOL トランスレータはインタプリータ・コードが，先に述べたようにソースプログラムに近いため，簡単な逐次的変換ルーチンとなっている．Fig. 2 に示した〈LOAD〉ルーチンは，ソースプログラムを入力して1ステートメント分のストリングを取り出し（継続行の処理），〈TRANS〉ルーチンに引渡す．〈TRANS〉は実際に1ステートメントを逐次的に翻訳

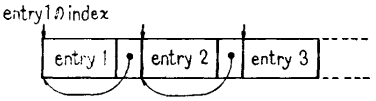

Fig. 3 Symbol table for SNOBOL 3.

する．

3.2 実行用システム

文全体の実行順序は文献 1) に従っている．

実行用システムは以下に示す7つのルーチンより構成されている．

1) INTPRT ステートメントの種類を判別し，対応した処理ルーチンを呼ぶ．

2) PREEV 5種類の string expression を前評価して，より単純な string expression または string variable の連接とするルーチン．評価すべき expression が関数 call の時は，INTPRT が再帰的に呼び出されるので，他のルーチンも再帰的呼び出し可能でなければならない．

3) 代入文処理 代入文の右辺の値を PREEV により評価し，左辺の変数に代入する．

4) マッチング文処理 string reference (st. ref.) に対して，pattern (pat.) をマッチングさせる．マッチングのアルゴリズムは **3.3** で詳しく述べる．

5) call 文処理 call 文を実行する．このルーチンは，次の関数呼び出しと go to 処理との2つのルーチンを引用する．

6) 関数の呼び出し 関数呼び出しに際して，呼び出された関数とのリンケージをとる．システム組込関数については，その値を求める．

7) go to 処理 go to field を評価して次に実行すべきステートメントを決定する．

3.3 マッチングのアルゴリズム

ここではマッチング文処理ルーチンにおける，パターン・マッチングのアルゴリズムについて述べる．初めにアルゴリズムで使用した変数の意味を次に示す．

STREF……string reference.

PAT(i)……第 i 番目のパターン・エレメント．
($1 \leqq i \leqq n$)

CRSR……一番最近にマッチングの成功した string reference 上の位置（カーソル）．

CRST……これまでに成功したマッチングでの CRSR の値を格納するスタック．

BWM……backward match のとき，値が真となる．

FATAL……現在の PAT(i) にマッチングするのに十分な長さの文字列が STREF に存在しない場合，値が真となる．

Step 1. （初期化） BWM←FATAL←偽；CRSR←CRST←c_1；i←1.

Step 2. （エレメント単位のマッチング） STREF の CRSR より右に PAT(i) をマッチングさせる．これは，PAT(i) の種類に応じたマッチング・ルーチン*により実行される．そのマッチングが成功ならば，step 3 へ，偽ならば step 4 へ．

Step 3. (forward match の進行) CRSR を CRST に入れる；$i=n$ ならば *return***，$i \neq n$ ならば，i←i+1；BWM ←偽；step 2 へ．

Step 4. （バックトラックの開始） FATAL が真ならば step 8 へ，偽ならば CRSR へ CRST よりポップ・アップする．

Step 5. （1エレメントのバックトラック） i=1 ならば step 6 へ，$i \neq 1$ ならば i←i−1；BWM←偽；step 2 へ．

Step 6. （マッチングのモードをチェック） マッチングのモードが ANCHOR ならば *return*.

Step 7. （スタート・ポイント（**Fig. 4** の c_1）の変更 CRSR が c_∞ に等しければ *return*，そうでなければ，CRSR を1文字分右へ進め，かつ CRSR を CRST に入れる；step 2 へ．

Step 8. （2つ前の s. v. へバックトラック） i=1 ならば *return*，$i \neq 1$ ならば2つ前の string variable (fixed length s. v. は除く) エレメントまでバックトラック（i を減らす），もし前に2つ以上の s. v. がなければ，step 6 へ；FATAL←偽；BWM←真；step 2 へ．

step 2 で第 m 番目のエレメントまでマッチングが成功したときの状態を **Fig. 4** に示す．

SNOBOL システムはフリー・フォーマットで記述

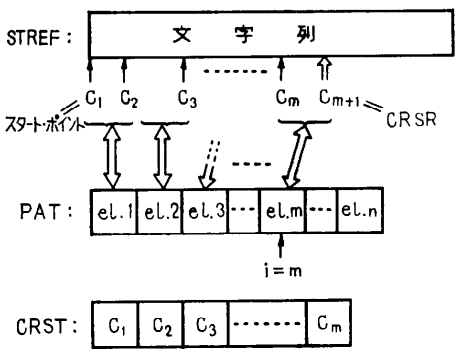

Fig. 4 Configuration of variables during matching.

* SNOBOL ソースプログラム中に現われるパターン・エレメントから派生したパターン・エレメント（最後の arbitrary s. v. と back reference される名前）に対応するマッチング・ルーチンを含めて7種類ある．

** ステートメントのマッチングが成功する場合はこの時に限る．

```
        DEFINE("NAME(X)","L0","Y")                    /(MAIN)
L0      X       "%" &ANB& "," =                       /S(L1)F(L2)
L1      Y = Y   "," ANB  "%"                          /(L0)
L2      SYSPOT = Y                                    /(RETURN)
MAIN    SYSPOT = TRIM(SYSPIT)
        NAME(SYSPOT)                                  /(END)
END
```

〔入力ストリング〕

```
%TSUTOMU,KUNITACHI%YUUJI,YOSHIDA%TERUO,FUKUMURA
```

〔出 力〕

```
%TSUTOMU,KUNITACHI%YUUJI,YOSHIDA%TERUO,FUKUMURA
'TSUTOMU''YUUJI''TERUO'
```

Fig. 5 An Example of SNOBOL 3 program.

されたカード約800枚より成り，その内240枚がトランスレータ，500 枚が実行用システム，残りは全体の制御を行うルーチンである．カード一枚には平均 4～5 の VC/S 命令が記述されている．

Fig. 5 に SNOBOL のプログラム例を示す．これは入力された姓名のリストから名前だけをとり出すプログラムである．このプログラムの翻訳・実行時間は約 54 秒であった．

4. VC/S の評価

4.1 命令構成について

実行命令の種類と実行効率に関する VC/S の評価を行うため Fig. 5 の SNOBOL プログラムを翻訳，実行した場合における $\mathcal{P}_1$ 命令の実行時間と実行回数の測定を行い，**Table 2** を得た．

ストリング処理においては，ストリングの複写，走査などのように必然的にストリングの長さに依存する演算と，ストリングの長さに全く依存しない演算とがある．Table 2 の平均命令実行時間で見ると，前者に対応する命令はいずれも実行時間が 1 ms を超えるのに対して，後者に対応する命令は，それが 1 ms 以内となっている．

一般に，ある機械の命令セットを決める際には，入出力命令や停止命令のように実行頻度に関係なく必要な命令を除けば，命令実行時間と命令実行頻度との積(命令の全実行時間)が全ての命令について一様になることが望ましいと考えられる．この観点から Table 2 を見ると，catenate, move $\mathcal{P}$, set $\mathcal{CR}$, match, conditional jump 等の命令は，他の命令に比べて著しく全実行時間が多いので，これらの実行速度を向上させることは VC/S 全体の効率に寄与すると考えられる．さらに，将来 VC/S がマイクロ・プログラム計算機を用いてハードウェア実現される場合には，これらの命令はマイクロ・プログラムにより実現すべきであると考えられる．このように，Table 2 は VC/S のハードウェア実現の方式に対する一つの指針を与えていると考えられる．

Table 2 Execution time of $\mathcal{P}_1$ instructions

命 令	全実行時間 (ms)	全実行回数	平均実行時間 (ms)
1. right	1,559	519	3.00
2. left	474	385	1.23
3. clear	391	441	0.89
4. catenate	4,624	1,260	3.67
5. insert	219	71	3.08
6. replace	16	10	1.60
7. encode	0	0	——
8. decode	0	0	——
9. add	163	53	3.08
10. subtract	178	44	4.05
11. multiply	4	1	4.00
12. divide	0	0	——
13. get cell	121	52	2.33
14. set $\mathcal{P}$	1,830	1,244	1.47
15. move $\mathcal{P}$	7,558	3,266	2.31
16. set $\mathcal{P}$ to right bound	396	587	0.67
17. set $\mathcal{P}$ to left bound	10	11	0.91
18. store $\mathcal{P}$	2,477	1,113	2.20
19. store	1,150	354	3.25
20. store substring 1	2,127	371	5.73
21. store substring 2	1,689	405	4.17
22. set $\mathcal{CR}$	7,034	3,405	2.07
23. stack $\mathcal{CR}$	963	1,377	0.70
24. reset $\mathcal{CR}$	1,001	1,377	0.73
25. right end ?	381	548	0.70
26. left end ?	229	351	0.65
27. greater than ?	46	21	2.19
28. equal ?	2,091	1,133	1.85
29. match ?	7,245	2,771	2.61
30. alpha ?	92	137	0.67
31. digit ?	28	44	0.64
32. virtual character ?	37	56	0.66
33. not	92	130	0.71
34. conditional jump	6,859	5,086	1.35
35. unconditional jump	2,592	2,150	1.21
36. return	978	1,088	0.90
37. stop	1	1	1.00
38. trace	0	0	——
39. input	343	11	31.18
40. output	455	34	18.96
合 計	61,818	30,985	2.00

4.2 データ構造について

VC/S は，ストリングを唯一のデータ構造とし，その上にかなり自由に扱えるポインタを設定することでストリングについてはかなり効率のよい処理が行えるようになっている．また，仮想文字機能が備えられていて，メタシンボルの取り扱い，インタプリータ・コードの実現等に便宜がはかられている．しかしながら，実際に VC/S 上で実現された SNOBOL 処理系の効率は 3 の最後に記されている実験結果によれば，決して良いものとはいえない．これは，言語処理系

が，単にストリング処理手順のみでなく，テーブル処理，リスト処理等を必要とするのに対して，VC/S は，直接にはそれらに対処する機能を備えていないことによると考えられる．ここで構成した処理系では，これらに対応する処理の全てをストリング上のポインタの値を利用することで間接的に行っている．Fig. 5 の SNOBOL プログラムを，その構造を保って VC/S プログラムで直接実現した場合 (SNOBOL プログラムを VC/S プログラムにコンパイルした場合と考えられる) の実行時間は，2.2 秒であった．このことからこのプログラムのアルゴリズムの実現自体については VC/S の効率は問題がないといえる．

以上述べた問題点を解決するための方法としては，

(1) ポインタ処理に関する命令の充実と高速化，

(2) 従来の構造をもつ記憶の付加，

の2つが考えられる．(1)は消極的な解決ではあるが VC/S の基本思想を乱さない範囲で行える．一方，(2)は効率に対して大幅な改善を与えると期待されるが，VC/S の設計思想の変更を必要とする．純粋に問題向き機械として考える場合には，(1)が妥当であると思われる．

5. あとがき

文字列処理を基本動作とする仮想計算機 VC/S を設計し，これを FORTRAN で記述し実現した．VC/S は文字列を記憶するための半無限長のセルの集合から成る記憶装置をもち，さらに演算対象となる文字列上の任意の位置に左右に自由に動くポインタが設定できて，これらに基づいて任意長の文字列に対する種々の演算が自由に行なわれる．また文字列を構成する文字として仮想文字が導入され，扱える文字セットを拡大することが可能である．

SNOBOL トランスレータの作成は，インタプリータ・コードの形式をソースプログラムに近いものにしたこと，仮想文字の機能を活かしたこと等によって比較的容易に行われた．また実行用システムについても，マッチング文の処理は VC/S のマッチング命令 29 の活用により簡潔に記述された．

なお，4 で述べたことがらに基づいた命令セットの改良，効率の向上，デバッグ機能の増強および VC/S 向きの文字列処理用高級言語 ($\mathcal{P}_2$) の設計等は今後の課題である．

使用計算機：名古屋大学大型計算機センター FACOM 230-60 (課題番号：昭和 48 年度 4001 CM 0366, 同 49 年度 4001 DK 1043)

謝辞 日頃熱心に御指導頂く東北大学本多波雄教授，並びに福村研究室の皆様に感謝致します．

参 考 文 献

1) R. F. Smith, et al.: SYMBOL: A large experimental system exploring major hardware replacement of software, SJCC, Vol. 39, pp. 519～538 (1971).
2) A. Hassitt, et al.: Implementation of a high level language machine, Comm. ACM, Vol. 16, No. 4 (1973).
3) 浅井他：SPM—ストリング処理の仮想機械，情報処理，Vol. 11, No. 1, pp. 11～19 (1970).
4) 三上他：新しい記号処理概念による SNOBOL インタプリタ，情報処理 Vol. 16, No. 6, pp. 476～483 (1975).
5) 吉田・福村：FORTRAN に基づく記号処理システム (COSMOS-2)，情報処理，Vol. 13, No. 4, pp. 232～238 (1972).

(昭和 50 年 3 月 3 日受付)
(昭和 50 年 5 月 28 日再受付)