SPM——ストリング処理の仮想機械*

浅井　清** 稲見　泰生** 斎藤　直之**

Abstract

1. Recently much of technical information is being exchanged or disseminated in the form of computer programs.

For engineering computation, most of computer programs are written in FORTRAN languages. The FORTRAN has, however, many dialects and, for this reason, it is an important problem to convert FORTRAN dialects into the standard FORTRAN or to convert a dialect into an another dialect.

2. Features of a conversion program are basically concentrated as the following;

(1) versatile features to process strings

(2) efficient set-theoretic operations for sets of which elements are strings.

The conversion program SPM is a general purpose program which executes the above mentconed features efficiently.

1. はじめ

従来からFORTRAN語を，これと類似の他のFORTRAN語に書き換える計算機プログラムの開発が行なわれてきた[1]．この種のプログラムは，変換プログラムと呼ばれている．これまでに実施されてきた方法には，それぞれ一長一短があるが，いずれの方法にも共通していることは，それらの方法に柔軟性がないことである．変換プログラムの実行効率を上げるためには，簡単な変換にも複雑な変換にも，一様に簡単な手順を適用されることが望まれる．この条件を満たすために，プログラム変換に必要とされる機能を持つ記号処理言語を使用して，変換プログラムを作成するという方向が考えられる．そこで，FORTRAN 語をこれと類似の言語へ変換するときに必要となる手順をあげてみれば，それらは文字列（以下では，文字列をストリングと呼ぶ）の転送・複写・比較・置換などのストリング処理の機能と，ストリングを要素とする集合についての集合論的な演算の機能に集約，大別されるであろう．この2種の機能の実行効率が，変換プログラムの効率を決定するといっても過言ではない．この2種の機能は，次のような基本的なものに分類することができる．

(1) 整数の算術演算.

(2) ストリングを要素とする集合についての集合演算.

(3) ストリングの名称を任意に変更して参照できる動的レイベル.

(4) 可変長のストリングの取扱い.

(5) ストリングへの早いアクセス.

(6) 文字の早い（効率のよい）認識.

(7) 文字を利用する早いブランチ.

(8) 間接的にストリングを参照できる間接番地.

これらの機能を計算機プログラムで実現する場合には，これに加えて

(9) 可変長ストリングの入出力.

(10) 記号処理言語としてプログラミングの容易なこと.

(11) 他の計算機との互換性.

などである．SPM[2]は，これらの機能を実現するために作成された，ストリング処理のための仮想機械である．

2. 必要な機能

2.1 ストリング・レイベルと添字

---

* SPM-A Pseudo Machine for String Processing by K. Asai, Y. Inami, N. Saito (Japan Atomic Energy Res. Inst.)
このプログラムは日本原子力コード委員会原子力コード整備小委員会の昭和 43 年度事業計画の一つとして，作成されたものである．

** 日本原子力研究所計算センター

ストリング処理の仮想計算機を考えるならば，ストリングの転送・複写・比較・置換などの演算は，二つのストリングを対象としているから，番地づけの方法は2番地方式が基本となる．参照されるのはストリングであるから，番地はストリング単位でつけられることでよい．この番地づけはストリングに名まえ（レイベル）を与えることによって行なう．ストリングが，転送・複写・比較・置換・検索などの演算の対象となったときに，初めてストリング内の文字の位置が問題になる．したがって，ストリング内の文字の位置は，そのストリングの最初の文字の位置を1とした相対位置（これをそのストリング・レイベルの添字と呼ぶ）を与えることによって参照できればよい．

2.2 ダイナミック・レイベル

ストリングの番地割当て・検索などの方法を抽象化した例で考えてみよう．理論的には無限に長いテープに，英数字が並んでいると仮定する．このテープを一方向に読み取りながら，文字Aが出現するたびに，その文字Aから次の文字Aが出現するまでの文字列を一つのストリングとして保存し，必要に応じて，そのストリングを参照する場合には，ストリングの長さが一定ではない無数に多くのストリングについて，そのレイベルを作り出さなければならない．このとき，作り出されたレイベルとそのストリングの間には，そのストリングの内容によって判別できる対応関係が必要である．この対応関係なしには，簡単に早くストリングを検索し，参照することはできない．

このような問題を解決するために，ダイナミック・レイベルの概念を導入する．ダイナミック・レイベル A. は，あらかじめ設定された容量を持っており，その容量の個々の要素と参照可能な番地を与えることができる．この個々の要素は，一般的に A.P なるレイベルで参照され，正の整数値をとる変数Pの値に依存して要素の番地が定められる．ストリングを集合 A. に保存するとき，そのストリングの特長を表わす部分の文字列を適当な方法で整数値Pに変換し，そのストリングを A.P なる要素に保存することによって，ストリングとレイベルの意味のある対応関係が確立される．

ダイナミック・レイベル A. は，一つの集合を表現しているとみなすことができるが，この集合に関する集合演算は，上に述べた方法によって，集合要素 A.P の持つ変数Pの算術演算に還元することができる．

2.3 間接番地

ストリング内に含まれている文字 A, B, C, ……のすべてについて，それぞれあらかじめ定められた文字の処理を実行することを考えよう．これらのひとつひとつの文字について，順次にその文字を判定してゆけば，その判定に要する時間は，無視できぬほど大きくなる．したがって，文字 A, B, C,……を認識することによって，直接に他のストリングと処理手順の参照を意味する機能が必要となる．この機能は間接番地の概念を導入することによって実現される．この間接番地の機能を使用すればレイベルXを持つストリングの内容が文字Bであるとき，レイベル（X）によって参照されるレイベルはBとなる．

これらの参照方法を組み合わせると，ストリングを要素とする集合 A. に対して，それぞれ順編成，索引つき順編成，ランダム編成番地のアクセス方法を使用することができる．

2.4 SPM で使用するレジスタ

SPM (String Processing Machine) は，二番地方式の仮想機械であり，その番地はストリング単位で与えられる．二番地方式を採用したのは，ストリングの処理が，一般に二つのストリングをその演算の対象とする比較・検索・転送などの操作から成ることが多いことによる．演算は一般には

OPC ADR, BDR, VAR

の形式で表現され，OPC は命令を，ADR, BDR, VAR はそれぞれA番地，B番地，バリアント・レジスタを示している．

以下この番地づけの方法と，関係のあるレジスタとレイベルの機能について説明する．

2.4.1 レジスタの説明

（1） 命令レジスタ (IC: Instruction Counter)
このレジスタには，現在実行されている SPM の命令の番地がはいる．

（2） A-アドレス・レジスタ (ADR: A-Address Register) このレジスタには，命令の A-アドレス部分で指定されたストリングの番地がはいる．命令の A-アドレス部分で指定されたストリングの番地が，直接にはストリングの番地を示していないこともある．そのときは，該当する番地は，アドレス・デコーダによって解析され，最終的にはストリングの番地がADRにはいる．

（3） B-アドレス・レジスタ (BDR: B-Address Register) このレジスタには，命令の B-アドレス部分で指定されたストリングの番地がはいる．機能そ

のものは，ADR と変わりない．

（4） バリアント・レジスタ (VAR: Variant Register)　このレジスタには，データの転送を行なう場合の境界を示す文字や，論理演算を行なう場合の比較の対象となる文字がはいる．

（5） A-アドレス添字レジスタ (APR: A-Address Character Pointer Register)　ADR にはいっている数値によって，その番地が示されているストリングは，SPM の命令によって，一文字単位で処理される．このとき，ストリングの何番目の文字まで処理が進んでいるかは，APR にはいっている数値によって示される．この APR には，演算が終了したとき，現在処理の対象となっていた文字の次の文字の位置が保存されている．

（6） B-アドレス添字レジスタ (BPR: B-Address Character Pointer Register)　このレジスタは，B-アドレスに対する添字のレジスタで，機能そのものは APR と変わりない．

（7） インディケイタ・レジスタ (IND: Indicator Register)　このレジスタには，論理判断の命令や特殊な命令の使用によって，定まった数値がセットされる．また，命令の誤操作によって起こる状態を示す数値も，このレジスタにセットされる．

2.5 レイベルの機能と種類

2.5.1 レイベル (label) の機能

SPM は，ストリングを一つの単位として処理する仮想機械であるから，処理されるストリングは，それぞれが他のストリングと区別できる名前を持っていなくてはならない．これがレイベルである．また，ストリングの処理は，そのストリングの最初の文字から始まるとは限らないから，ストリングに含まれる中間の文字の位置を示す機能が要求される．この機能は，添字つきレイベルを使用することによって実現される．

ストリングは，計算の前に，つまり SPM 文の翻訳の段階で定義されているものと，SPM の命令を使用して，計算途中で作り出されるものがある．前者のストリングを定義するときに使用されるレイベルを静的なレイベル，またはスタティック・レイベル (static label) と呼び，後者のストリングを定義するときに使用されるレイベルを動的なレイベル，またはダイナミック・レイベル (dynamic label) と呼ぶ．

SPM の算術命令を使用するときに現われる変数も，数値を内容とするストリングとみなされ，その変数名はレイベルとして扱われる．これは数値レイベルと呼ばれる．

SPM の命令に対してもスタティック・レイベルを与えることができる．これは主として SPM の分岐命令の行き先に対応するレイベルである．これは命令レイベルと呼ばれる．

2.5.2 レイベルの種類

上記の説明に従ってレイベルを分類すると

- レイベル
  - スタティック・レイベル
    - ストリング・レイベル
    - 命令レイベル
    - 数値レイベル
  - ダイナミック・レイベル
    - ストリング・レイベル
    - 数値レイベル

（1） 数値レイベルの定義例

```
P    DC    1
```

Define Constant なる SPM の擬似命令によって定義された，数値1を内容として持つストリングのレイベルはPで表わされる．

（2） 命令レイベルの定義例

```
MOVE    MCM     A, B,     ▼S▼
                :
        BRA     MOVE
```

上の命令の列で，MCM (MOVE CHARACTERS TO MARK OR BOUNDARY) なる命令に与えられたレイベル MOVE は，命令レイベルであって BRA (BRANCH) 命令の MOVE と対応している．

（3） ストリング・レイベルの定義例（スタティック）

```
        MCM
        :
A       DS      ▼MOVE△A△TO△B▼
```

上の命令の列で，Define String 命令によって定義されたストリングのレイベルはAである．ここで△は空白記号を表わす．

（4） ストリング・レイベルの定義例（ダイナミック）

```
B.      SIZE    100
        ADD     ONE, P
        MCM     A, B. P
                :
ONE     DC      1
A       DS      ▼MOVE△A△TO△B▼
P       DC      0
```

上の命令の列で

MCM　A, B. P

が実行されると，そのときのPの内容に依存したレイベルを持ち，その内容はストリングAの内容と同一であるストリング B. P が定義される．

2.5.2　レイベルの表現と番地の表現

（1）　スタティック・レイベルは，長さが四文字を越えない 0, 1, ～9, A, B, ～Z および特殊記号（ただし，(,)，ピリオド，カンマ，空白文字，▼ の６種を除く）を使用して定義される．

（2）　ダイナミック・レイベルは，二つのスタティック・レイベルを並べ，その中間にピリオドを一つ置くことで定義される．

（3）　上の二つのレイベルは添字を持つことができる（ただし，数値を保存しているストリングのレイベルは，添字を持つことができない）．このとき添字は，そのレイベルが示しているストリング内の文字の位置をさしている．

（4）　SPM は一重の間接番地を受けつけることができる．間接番地は，レイベルをカッコでくくることによって表現される．

間接番地の機能を簡単に述べておこう．いま，アドレス欄に (A(P)) なる間接番地が指定されているときは，アドレス・デコーダの働きは次のようになる．

(*1)　ストリングAの第P番目の位置から四文字を取ってくる．ストリングの終わりに近くて，四文字ないときは，ストリングの終わりまでの文字を取ってくる．これをスタティシク・レイベルBと呼ぼう．

(*2)　スタティック・レイベルBをスタティック・レイベル表から捜す．

(*3)　スタティック・レイベル表には，レイベル名と，その内容であるストリングの番地がはいっている．このストリングBの番地が求めるべき番地としてアドレス・レジスタに置かれる．

以上の手順からわかるとおり，間接番地の最終的な番地はスタティック・レイベルである．

（5）　次に個々のレイベルについて，以上述べてきたことをまとめて示せば

	単統番地	添字番地	間接番地
数値レイベル*	YES	NO	YES
命令レイベル**	YES	NO	NO
ストリング・レイベル	YES	YES	YES

(*)　数値レイベルは，数値化された内容を持つストリングを示しているので，添字によって文字の位置をさすことはできない．

(**)　命令レイベルは，他のストリングとは内部構造が異なっているので，添字番地・間接番地を使用できない．

（6）　データの転送命令を使用して，ダイナミック・レイベルを持つストリングへ文字の転送が行なわれる場合には，その転送されるストリングの長さに従って，ダイナミック・レイベルのストリングの長さが拡張される．これはダイナミック・レイベルの一つの特長である．

スタティック・レイベルを持つストリングは，通常は拡張されないから，そのストリングの最後の位置まで文字が転送されると，転送は自動的に終了する．

しかし，後の **3.** の命令の説明の項で述べるように，次の二つの SPM の命令 RCS (REPLACE A CHARACTER BY A STRING) と CAT (CATENATE) を実行し，その結果が B-アドレスのスタティック・レイベルで指定されたストリングの内容となるときには，この B-アドレスのストリングの長さは自動的に拡張される．

（7）　各種レイベルの使用例をあげると

（a）　単純直接，間接番地　KOKO, (KOKO)

```
        MCM   KOKO, WORK, ▼(▼
        MCM   (KOKO), SPAC
              ⋮
KOKO    DS    ▼READ (▼
WORK    DS    ▼△△△△△INPUT△TAPE▼
SPAC    DS    ▼△△△△△△OUTPUT△TAPE▼
READ    DS    ▼WRITE▼
```

上の例の最初の MCM 命令が実行されると，ストリング WORK に KOKO の内容である READ が転送され，その結果 WORK の内容は READ INPUT TAPE となる．

二番目の MCM 命令が実行されると，間接番地の使用によって，ストリング READ の内容が SPAC へ転送される．その結果，SPAC の内容は WRITE OUTPUT TAPE となる．

（b）　添字つき直接，間接番地　KOKO (P), (KOKO(P))

```
        ADD   ONE, P
        MCM   KOKO (P), WORK, ▼)▼
        MCM   (KOKO(P)), SPAC
              ⋮
KOKO    DS    ▼READ (▼
WORK    DS    ▼△△△△△INPUT△TAPE▼
SPAC    DS    ▼△△△△△△OUTPUT△TAPE▼
READ    DS    ▼WRITE▼
P       DC    0
ONE     DC    1
```

上の例で添字Pの値は１であるから，結果は（a）の例とまったく等しい．

（c）　動的レイベルの直接，間接番地　A. P,(A. P)

```
A.      SIZE  100
        ADD   ONE, P
              ⋮
```

```
        MCM     KOKO, A.P, ▼(▼
        MCM     (A.P), SPAC
          ⋮
KOKO    DS      ▼READ (▼
WORK    DS      ▼△△△△△INPUT△TAPE▼
SPAC    DS      ▼△△△△△△OUTPUT△TAPE▼
READ    DS      ▼WRITE▼
P       DC      0
ONE     DC      1
```

上の例では，最初のMCM命令によって，ダイナミック・レイベル A.P がPの値に依存し定義され，そのストリングの内容はREADである．二番目の MCM 命令によって，SPAC の内容は WRITE OUTPUT TAPE となる．

（d） 動的，添字つき直接，間接番地 A.P(Q), (A.P(Q))

```
A       SIZE    100
          ⋮
        ADD     ONE, P
        ADD     ONE, Q
        MCM     KOKO, A.P, ▼(▼
        MCM     (A.P(Q)), SPAC
          ⋮
KOKO    DS      ▼READ (▼
WORK    DS      ▼△△△△△INPUT△TAPE▼
SPAC    DS      ▼△△△△△△OUTPUT△TAPE▼
READ    DS      ▼WRITE▼
P       DC      0
ONE     DC      1
Q       DC      0
```

上の例でQの値は1に等しいから，結果は（c）の例とまったく等しい．

3. 命 令

命令を大別すると，算術演算を行なう算術制御命令，比較と分岐を行なう論理制御命令，転送を行なうデータ制御命令，入出力を行なうシステム制御命令，データの初期値を設定する擬似命令の5種の命令群に分けることができる．これらのうちから代表的なものを取り上げて説明しよう．

SPM の命令を説明するまえに，アドレスのチェイニング（番地の連鎖）について述べておこう． SPM は，SPM 語の A-アドレス，B-アドレスに対応して，それぞれ ADR（A-アドレス・レジスタ），APR (A-アドレス添字レジスタ），BDR（B-アドレス・レジスタ），BPR (B-アドレス添字レジスタ）を持っている．

これらのレジスタの内容は一度セットされると，再び SPM 語で指定されない限り，そのまま保存されている．この保存されているレジスタの内容を，他の命令のためのアドレスとして使用することをチェイニングと呼ぶ．

Table 1. The Instructions of SPM

命 令	
	算術制御 (ARITHMETIC CONTROL)
ADD	ADD ADR TO BDR
SUB	SUBTRACT ADR FROM BDR
ZADD	ZERO AND ADD TO BDR
ZSUB	ZERO AND SUBTRACT FROM BDR
MPY	MULTIPLY ADR AND BDR
DIV	DIVIDE BDR BY ADR
AND	AND ADR TO BDR
OR	OR ADR TO BDR
COMPL	STORE COMPLEMENT OF ADR IN BDR
	論理制御 (LOGICAL CONTROL)
CS	COMPARE STRINGS
CN	COMPARE NUMBERS
BCE	BRANCH IF CHARACTER EQUAL
BCT	BRANCH ON CONDITION TEST
BRA	BRANCH (UNCONDITIONAL)
NOP	NO OPERATION
	データ制御 (DATA CONTROL)
CNVRT	CONVERT STRING TO BINARY OR BINARY TO STRING
MCM	MOVE CHARACTERS TO THE MARK OR STRING BOUNDARY
MAC	MOVE A CHARACTER
MCS	MOVE CHARACTERS AND SUPPRESS VARIANT
SAR	STORE A-ADDRESS REGISTER
SBR	STORE B-ADDRESS REGISTER
SAP	STORE A-ADDRESS CHARACTER POINTER
SBP	STORE B-ADDRESS CHARACTER POINTER
STI	STORE INDICATOR
SRCH	SEARCH A STRING IGNORING VARIANT
RCS	REPLACE A CHARACTER BY A STRING
ERASE	ERASE A STRING
CAT	CATENATE ADR AND BDR SETTING VARIANT
LEN	GET LENGTH OF ADR STRING
LDA	LOAD A TO ADR
MOD	ABSOLUTE MODULO OF BDR
LAST	GET REMAINING AVAILABLE STORAGE SIZE
TON	TRACE ON
TOFF	TRACE OFF
	システム制御 (SYSTEM CONTROL)
REWIND	REWIND A UNIT
READ	READ A STRING
PUNCH	PUNCH A STRING
PRINT	PRINT A STRING
WRITE	WRITE A STRING
EXIT	EXIT FROM SPM CONTROL
	擬似命令 (PROCESSOR CONTROL)
DS	DEFINE STRING
ETC	EXTEND CARD FOR DS OPERATION
DC	DEFINE CONSTANT
SIZE	DEFINE MAX SIZE OF DYNAMIC LABEL
END	END OF SPM INSTRUCTIONS

ADR, APR, BDR, BPR を指定しなければならない命令で，これらが指定されていないときには，その

命令の直前の命令の終了した時点で，各レジスタにセットされた値が使用される．

バリアント・レジスタのチェイニングはできない．

SPM は **Table 1** で示される命令群をもつ．

Table 2 にインディケイタ・レジスタの値の一覧表を示す．

Table 2 Indicator Status

インディケイタの数値	説明
1	CN 命令で，〔ADR〕<〔BDR〕のとき
2	CN 命令で，〔ADR〕=〔BDR〕のとき
3	CN 命令で，〔ADR〕>〔BDR〕のとき
5	CS 命令で，〔ADR〕=〔BDR〕のとき
6	CS 命令で，〔ADR〕≠〔BDR〕のとき
10	BCE 命令で，BDR の添字が BDR のストリングの長さを越えて指定されたとき
12	MCM 命令で，ADR のストリングが使いつくされたとき
13	MCM 命令で，BDR がスタティック・レイベルで BDR のストリングが使いつくされたとき
15	MAC 命令で，ADR のストリングが使いつくされたとき
16	MAC 命令で，BDR がスタティック・レイベルで BDR のストリングが使いつくされたとき
18	MCS 命令で，ADR のストリングが使いつくされたとき
19	MCS 命令で，BDR がスタティック・レイベルで BDR のストリングが使いつくされたとき
21	SRCH 命令で，〔ADR〕=〔BDR〕
22	SRCH 命令で，〔ADR〕≠〔BDR〕
26	CAT 命令で，ADR の添字が ADR のストリングの長さを越えて指定されたとき
31	CNVRT 命令で，ADR の添字が ADR のストリングの長さを越えて指定されたとき (文字→整数)
32	CNVRT 命令で，BDR の添字が BDR のストリングの長さを越えて指定されたとき (整数→文字)
33	CNVRT 命令で，BDR 全部に文字が入られないうちに BDR が使いつくされたとき (整数→文字)
34	CNVRT 命令で，ADR の最初の文字が数字でないとき (文字→整数)

(**注**) 命令の説明で使用する記号を説明しておくと

ADR：A-アドレス・レジスタ

BDR：B-アドレス・レジスタ

〔ADR〕：A-アドレス・レジスタによって表示されるストリングの内容

VAR：バリアント・レジスタ

CS：COMPARE STRINGS

```
CS  ADR, BDR
CS  ADR
CS
```

この命令は，ADR, BDR ともに，指定されたレイベルは，ストリング・レイベルとして取り扱う．ストリング・レイベルは，添字を持つことができる（ただし，ストリング・レイベルに添字のないときは，添字を1と考える）．

添字で指定された位置から，ADR かまたは BDR のストリングの長さの短い方を中心にして，他方のストリングと等しいときは，インディケイタ・レジスタに数値5を，等しくないときは，インディケイタ・レジスタに数値6をおく．

ただし，ADR, BDR のストリングの長さを越えるような誤った添字を与えたときも，インディケイタ・レジスタに数値6をおく．

具体的には，ADR, BDR で指定されたストリング内の文字の添字の位置から，おのおの1文字ずつ取り出して比較し，等しくない文字が現われるか，または一方のストリングがなくなるまで比較を続ける．

ADR, BDR のストリングの文字の位置を示す APR, BPR レジスタは，常に比較し終わった次の文字をさしている．

BCT：BRANCH ON CONDITION TEST

```
BCT  ADR, BDR
```

この命令では，ADR は命令レイベル，BDR には数値そのものを書く．

BCT 命令が出現する以前に，インディケイタ・レジスタに比較命令や検索命令などでおかれた数値と，BDR に与えられた数値とを比較する．比較の結果等しい場合は，次の命令を ADR に書かれた命令レイベルに実行を移し，等しくないか，または ADR に間接番地が指定されて，スタティック・レイベル表に対応する命令レイベルがない場合は，BCT 命令の次の順番にある命令より実行が続けられる．

インディケイタ・レジスタの数値は，この BCT 命令実行後は0にされる．

ADR の命令レイベルに実行が移されたとき，ADR レジスタに BCT 命令の次の順序にある命令の番地がおかれる．これを分岐先で保存すれば，サブルーチンの働きを持った命令群を容易に作成できる．

CNVRT：CONVERT STRING TO BINARY OR BINARY TO STRING

```
CNVRT  ADR, BDR, V
```

この命令は文字を整数に，整数を文字に変換するときに使用される．バリアントで整数1が指定されているときは，ADR に指定されたストリングの文字が整

数に変換され，その結果が BDR の数値レイベルの内容となる．ADRのストリングは変化しない．

バリアントで整数2が指定されているときは，ADR に指定された数値レイベルの内容である整数が文字列に変換され，その文字列が BDR で指定されたストリングの位置に埋め込まれる．SPM の入出力命令は文字列しか取り扱うことができないので，このCNVRT 命令は入出力に際して，数字と文字を変換するために使用される．

MCM: MOVE CHARACTERS TO MARK OR STRING BOUNDARY

```
MCM  ADR, BDR, V
MCM  ADR, BDR
MCM  ADR
MCM
```

この命令は，ADRで指定されたレイベルの内容を，BDR で指定されたレイベルへ転送する働きを持っている．バリアントが指定されているときは，そのバリアントと同じ内容を持つ文字が ADR のストリング中に現われるまで転送が行なわれる．その文字自身は転送されないが，APR はその文字の位置を保存している．

バリアントの指定がない場合には，ADR で指定されたレイベルの内容が，ADR または BDR で指定されたレイベルで示されるストリングの長さに従って転送される．転送が終了したとき，ADR, BDR の添字レジスタ APR, BPR は，それぞれ次に転送を開始すべき文字の位置を保存している．バリアントの指定があったときでも，ADR のストリングにバリアントと同じ文字がなければ，この命令の働きはバリアントが指定されない場合と同じである．

SAR: STORE A-ADDRESS REGISTER

```
SAR  ADR
```

この命令は，この SAR 命令が実行される直前の A-アドレス・レジスタの内容を ADR で指定された数値レイベルへ保存する働きを持っている．この命令は分岐命令でプログラムの流れを変更した直後に，以前の分岐点を保存するために使用する．

SAP: STORE A-ADDRESS CHARACTER POINTER

```
SAP  ADR
```

A-アドレス添字レジスタ APR の内容が，A-アドレスで指定された数値レイベルの内容となる．

SRCH: SEARCH A STRING IGNORING VARIANT

```
SRCH  ADR, BDR, V
SRCH  ADR, BDR
SRCH  ADR
SRCH
```

この命令は，BDR のレイベルで指定されるストリングを，ADR で指定されるストリングの指定された文字の位置から捜し出すために使用される．

バリアントが指定されていないときは，BDR のレイベルで指定されたストリングは，ADR のレイベルで指定されたストリングの指定された文字の位置から比較される．比較は一文字単位で行なわれ，ADR のストリングと BDR のストリングの文字が一致していなければ，比較はそこで打ち切られて，インデイタに対応する数値がセットされる．

ERASE: ERASE A STRING

```
ERASE  ADR
```

この命令は，不必要となったストリングを消去するために使用される（ただしダイナミック・レイベルのみ）．消去されたメモリは，Free Storage と呼ばれる領域に保存され，他の SPM 命令で新しくメモリが必要となったときに再び使用される．

MOD: ABSOLUTE MODULO OF BDR

```
MOD  ADR, BDR, V
MOD  ADR, BDR
```

この命令は，ADR のレイベルで指定されたストリングの内容を，BDR の数値レイベルの内容で除算し，その剰余の絶対値を BDR の数値レイベルに保存するために使用される．

バリアントに整数1が指定されているときは，SPM が使用している FORTRAN 語の一語を基準にして，ADR のストリングが BDR の数値で除算される．剰余の絶対値が BDR の内容となる．バリアントに整数2が，指定されているときは，SPM が使用している FORTRAN 語の二語を基準にして，ADR のストリングが BDR の数値で除算される．

バリアントの指定がないときは，ADR も数値レイベルとみなされる．SPM では，この命令とダイナミック・レイベルを使用して，散乱格納[3),4)]の手法を利

用できる.

TON: TRACE MODE ON

```
TON
```

この命令は, 26 個のレジスタの内容を出力するために使用される. 出力は TOFF (TRACE OFF) 命令が出現するまで続く.

READ: READ A STRING

```
READ  ADR, BDR, V
READ  ADR, BDR
```

この命令は, ADR のレイベルで指定されたストリングに, 機番 V の入力装置から BDR で指定された数値レイベルの個数だけの文字を読むために使用される. バリアントには 1 から 12 までの数値を指定することができ, この数値は FORTRAN 語の論理ユニット番号と対応している.

4. 使用経験

SPM を使って FORTRAN II から FORTRAN IV への変換プログラムをいくつか作成した. その結果, 従来の変換プログラムと比較して, SPM は 2 倍から 5 倍の実行効率を持っているようである. SPM を作成するに当って, FORTRAN II の入出力文のみを変換する場合には, IBM 7090 で 800～1,000 FORTRAN 文/1 分の性能を目標の一つに置いたが, これはほぼ達成された. SPM は注釈文を除いて約 4,000 文の FORTRAN IV 文から成っていて, 32K 語以上の記憶容量を持つどのような計算機の FORTRAN IV 語でも使用できるよう設計されている. 使用できる文

付録 1.

```
$RELOAD        U07,NAME=SPM,SRCH
*SPM   UNITNO  BYTE=6,RCC

ISN        SOURCE STATEMENT                          UNIT                    PAGE  1


  1        PRINT ONE                      SKIP ONE PAGE BEFORE PRINTING
  2        MCM   ASTR,S(100)              MOVE CHARS TO MARK OR BOUNDARY
  3  R1    READ  CARD,N80                 READ A CARD
  4        BCE   CONT,CARD(6),' '          NOT A CONTINUATION CARD, CONTINUE
  5        BRA   PRINT                    CONTINUATION CARD, SKIP
  6  CONT  BCE   PRINT,CARD,'C'           SKIP IF A COMMENT CARD
  7        MCM   BLANKS,S
  8        ZADD N7,I                      CLEAR OUTPUT BUFFER
  9  FIRS  BCE   ZAD,CARD(I),' '          BRANCH TO ZAD IF CARD(I)=BLANK
 10        BCE   RED1,CARD(I),'R'         GO TO RED1 TO TEST READ
 11  PRINT PRINT CARD,N80                 PRINT N80 CHARS
 12        BRA   R1                       BRANCH TO R1
     C
 13  ZAD   ADD   ONE,I                    ADD ONE TO I
 14        BRA   FIRS                     BRANCH TO FIRS
     C
     C     TEST READ
     C
 15  RED1  SRCH  CARD(7),READ,' '         SEARCH READ IGNORING BLANKS
 16        SAP   J                        STORE A-ADDRESS CHAR. POSITION
 17        BCT   PRINT,22                 INDICATOR =22, UNEQUAL
 18        SRCH  CARD(J),LP,' '           SEARCH A LEFT PAREN. FROM CARD(J)
 19        SAP   J                        STORE A-ADDRESS CHAR POSITION
 20        BCT   PRINT,22                 GO TO PRINT IF SRCH FAILED
 21        SRCH  CARD(J),RUNITNO,' '      SKIP BLANKS AND SEARCH THE UNIT NO.
 22        SAP   J
 23        BCT   PRINT,22                 BRA IF UNEQUAL
 24        MCM   RST,S(7)                 MOVE READ(5
 25        MCM   CARD(J),S(J)             MOVE REMAINING LIST
 26        MAC   DOL,CARD(6)              MOVE A CHARACTER $ TO CARD(6)
 27        MCM   CARD,S,'$'               MOVE STATEMENT NO. TILL $ MARK
 28        PUNCH S,N80                    PUNCH N80 CHARACTERS FROM S(1)
 29        PRINT S,N120
 30        BRA   R1                       PROCESS NEXT CARD
 31  N7    DC  7                          DEFINE CONSTANT 7
 32  I     DC  0
 33  J     DC  0
 34    NC  DC  0
 35  N80   DC  80
 36  N120  DC  120
 37  ONE   DC  1
 38  READ  DS  'READ'                     DEFINE STRING READ
 39  RUNITNC DS '10'
 40  ASTR  DS  '****'
 41  RST   DS  'READ(5'
 42  DOL   DS  '$'
 43  S     DS  110X' '                    DEFINE 110 BLANK CHARACTERS
 44  LP    DS  '('
 45  CARD  DS  80X' '
 46  BLANKS DS 80X' '
 47        END

READ(3) ((A(I,J),I=1,NMAX),J=1,NN)
C     READ(10,130)((A(I,J),I=1,NMAX),J=1,NN)
      READ(5 ,130)((A(I,J),I=1,NMAX),J=1,NN)                                  ****
```

字系は EBCDIC と BCD コード系の2種であるが，これはテーブルを変更することによって，ASCII コード系なども使用できる。SPM の第1版は IBM 7044 で，第 1.5 版は IBM 7044 と IBM S/360-75 でテストされた．現在は第 1.6 版を IBM 7044 でプログラム変換に利用している．SPM の機能は COBOL よりも PL/I に近いが，長さの制限のないダイレクト・アクセスの可変長ストリングと間接番地の取扱いについては，PL/I の範囲を越えているであろう．付録 1. は入力機番 10 を5に変換する SPM のプログラム，およびその出力例である．SPM の言語としてアセンブラ言語に近いものを選んだ理由は，第一には文字の取扱いの便利なこと，第二にはコンパイラの作成の容易なことによる．

参考文献

1) 浅井 清：“FORTRAN 語のための変換プログラムの問題”，情報処理学会誌，Vol. 10, No. 4, p. 235 (昭 44-07).
2) 清井 清，斎藤直之，稲見泰生：“SPM—ストリング処理のための FORTRAN プログラム”，JAERI-memo, 3595, 日本原子力研究所.
3) Maurer, W. D.: “An Improved Hash Code for Scaller Storage, Comm. ACM”, pp. 35-38 (Jan. 1968).
4) Morris, Robert: “Scatter Storage Techniques”, Comm. ACM, pp. 38～44 (Jan. 1968).

(昭和 44 年8月4日受付)