談　話　室

靴　紐　の　話*

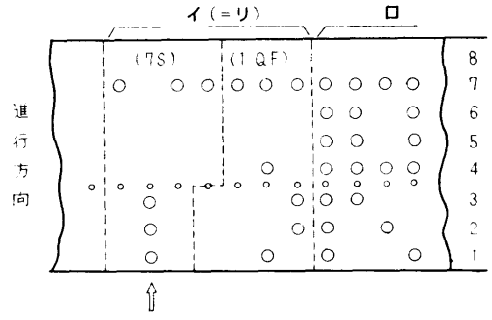
中　野　猛　夫**

われわれの所で，教育用として使っている小型計算機のような場合，完成したプログラムは，よく自己読込式紙テープとして保管される．このようなテープには，その先頭に短い「靴紐」と呼ばれる部分を持っており，テープを読取機に取り付けて，所定のスイッチを押すと，読込みと実行が数回交互に行なわれた後，靴紐は記憶装置の中で，簡単な入力用ループとなり，それに乗って2進形式で表わされた本文は，きわめて早く記憶装置に読み込まれるという仕組になっている．

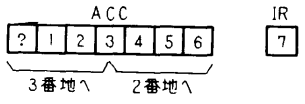
しばらく前まで，われわれの機械では，2度スイッチを押さなければならないとされていた．そのわけは以下のとおりである．これを靴紐の改良によって，何とか1度ですます方法はないかというところから，この話が始まる．

さて，われわれの用いているのは8単位のテープで，スプロケット穴をはさむ6個の穴が有効で，1字分6ビットの情報となる（**第1図**参照，第7穴は奇偶検査用）．また，1命令語は21ビット（＝3字半）で，前12ビット（＝2字分）がその番地部である．

いま，スイッチを1度押すと，$7S$：7字分，すなわち，2命令語分を ACC に読み込め，$2F$：ACC の内容を 2—3 番地移せ，という2命令が（それ自身は記憶装置にはいることなく）実行され，2番地にはいったばかりの命令から，実行する状態になる．ところが，$7S$ でテープ上の7字が全部 ACC にはいるのかというと，機構上そうはならないで，IR（入力レジスタ）にすでにはいっている不明の1字分がまずはいり，続いてテープ上の6字が送り込まれるので，最後の1字は IR に残ってしまう（**第2図**）．このため，3番地の命令は番地部が不定であるから，大低の命令では具合が悪い．そこで，従来の靴紐では，ここを停止命令にしてある．そして，もう1度スイッチを押すと，今度は IR の中味がわかっているので，最後まで連続して読み込むことができるのである．



第1図



第2図

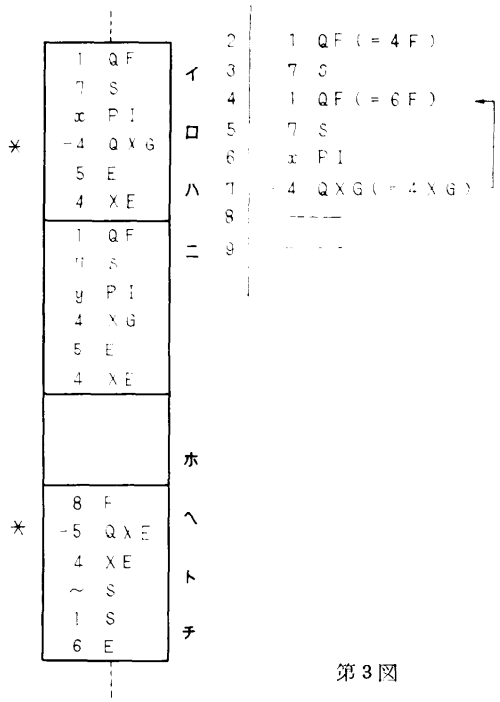
これでわかるように，われわれの目的のためには，3番地にはいる命令について，もう一工夫しなければならない．ところが，靴紐は普通のプログラムと異なり，読書きを交互に，しかも，せまい記憶範囲で行なわれなければならないので，大体コード表にあげたような命令ぐらいしか使えない．幸いにも，S がこの場合目的に合うのである．すなわち，mS によって実際に読み込まれる字数は，m が $32=2^5$ で割切れる場合を除いて，32 で割った余りに等しいように設計されている．問題の不明な1字は，番地部の前半にくるので，それが何であっても，$64-2^6$ の倍数を表わすことになり，もし，後半が7ならば，番地部は不定でも，実行の結果はいつも $7S$ と同じである．これだけのことから，問題はほとんど解けるのであって，以下に，この方式の靴紐の例を説明する．命令の内容については，コード表を参照していただきたい．

さて，テープのイの第2字目（第1図の矢印）を，読取機の穴に合わせて取り付け，スイッチを入れる．

* A Boot Strap, by Takeo Nakano (Depart-ment of Mathematics Rikkyo University)

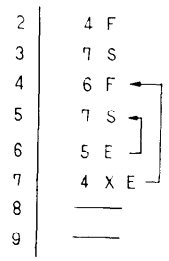
** 立教大学理学部

そうすると，すでに説明したように，命令 $1QF$ は2番地に，$7S$（または，それと同等の命令）が3番地にはいる．2番地の命令を実行するとき，SCC＝3 であるので，$1QF$ は $4F$ と同じである．このときACCには，**イ**の部分がまだ残っているから，4—5 番地には，2—3 番地と全く同じものがはいる．3番地の $7S$ で，テープの**ロ**の部分がACCにはいり，4番地の $1QF$，実は $6F$ によって，**ロ**の命令は 6—7 番地にはいる．ここまでの状態が**第3図**である．



第3図

次に，5番地の $7S$ で**ハ**がACCにはいる．6番地で本文プログラムがはいるべき先頭番地 x がインデクス2（第2インデクスレジスタ）に記録される．7番地の命令は特殊な命令で，まず，インデクス1に，そのときのSCCの値，すなわち，8を記録し，次いで，4番地へ飛び越す．インデクス1は後述でわかるように，本文の語数を数えているのであって，ここで，その初期値として8をとっているが，後に修正される．4番地でACCの内容，すなわち，**ハ**が 6—7 番地に書き込まれる．ここまでを**第4図**に示す．

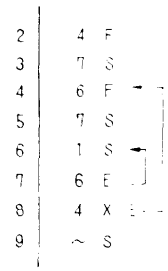


第4図

このとき 5—6 番地にできたループは，7字ずつ読み込んで，ACCの符号ビットが0である限り回り続け，1になったとき，初めて7番地へ抜け，さらに4番地へ飛び，その内容を 6—7 番地に書き入れるようになっている．これはテープの方でいうと，7字ずつの部分の第1字目第6の穴が，パンチされていなければ読み飛ばされ，パンチされているときは，その部分が 6—7 番地にはいることを意味している．各図で左に＊印をつけたものが，そのような7字分である．**ニ**から**ホ**までは**イ**～**ハ**とほとんど同じであるが，いまは読み飛ばされて**ヘ**に至って，初めてそれが 6—7 番地にはいることになる．もし，初めから**ニ**の第2字目を読取機に取り付けて出発させたものとすれば，前同様**ホ**までは，やはり無視されるが，本文の先頭番地が y になるところが前と異なる．これは本文プログラムが，いわゆる，相対番地方式であって，どこに入れてもよい場合に，単に，読取機に取り付ける位置を変えるだけで，先頭番地を容易に選びうるための工夫である．もちろん，通常はこのような自己読込テープを，全く任意の先頭番地以降に入れるためのプログラムを別に作っておくのだが，大体 4，5 とおりの先頭番地を，あらかじめ指定した上記のような靴紐で充分実用になるので，特に，アセンブラや，自己読込テープ作成用プログラムなどの場合は，まことに簡便である．

これまでの説明で，いくらか靴紐プログラムの読み方が，おわかりになったと思うので，以下では，細かい手順は省略させていただく．

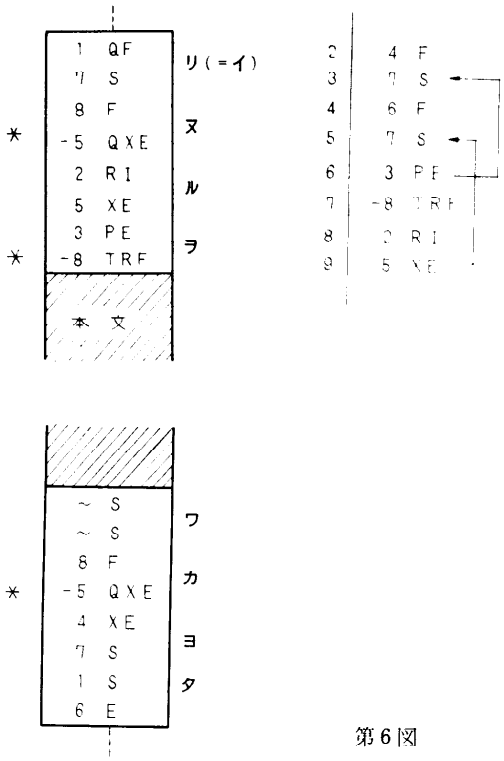
前にもどって，**ヘ**～**チ**まで読み込まれた状態が**第5図**である．このとき 6—7 番地にできたループは，以前説明したものに比べて，1字ずつ読むところが違うだけである．テープの方では，初めて第6の穴がパンチされた字が現われるまで読み飛ばし，その字を含む7字分が，6—7 番地にはいるのである．**チ**以後に続く点線部分は，このループで無視されるものなら何でもよく，また，長さも任意でよい．なぜ，このようにしてあるのかは後述でわかるが，要するに，靴紐の一区切を示しているのである．



第5図

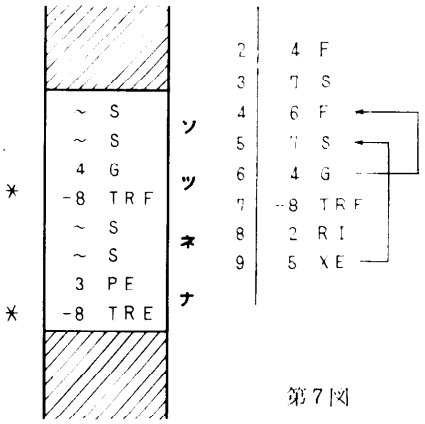
このようにして，第1図からわかるように，**リ**も読み飛ばされ，**ヌ**に至って，初めて6番地以降に**ル**～**ヲ**

がはいる．このとき**第6図**の状態になる．5—9 番地にできたループが，本文を先頭から順に読み込むのであって，7番地の命令の番地部が，—8 になっているのは，インデクス1の初期値を修正しているのに注意．



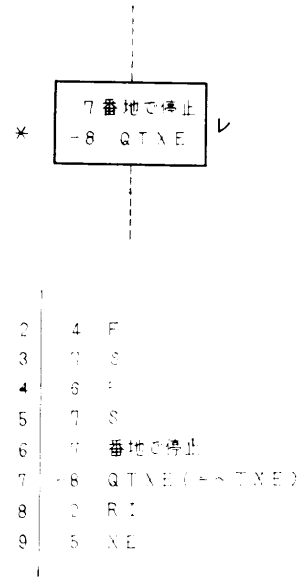
第6図

このループは，6番地の命令からわかるように，空白，すなわち，ACC のすべてのビットが0になるまで，無制限に回り続ける．これはテープ上では**ワ**のように，$\sim S$ $\sim S$ の形が読み込まれると空白になるので，もし，**ワ**が読まれると3番地に飛び，**カ～タ**の部分によって，6番地以降は再び第5図の状態になり，1字ずつ点線部分を読むことになる．すなわち，空白部分は本文読込みを一時中断させるための信号である．しかし，本文の中に空白を必要とする場合は，そんなに数は多くないのが普通であるが，全く無いわけではない．そのときは**ソ～ナ**の部分が，本文にそう入される．この働きは，一時的に読込ループをみだして，**第7図**のように6—7 番地を変え，空白**ネ**を書き込むと，すぐに再び前のループにもどるようになっている．このようにして，特定の信号によって本文読込を中断させる理由は，再び**リ**以下の別のプログラムを，接続することを可能にするためで，従来の靴紐のように，本文語数をあらかじめ読込ループの中に組み込んでしまうようなやり方では，むずかしくなるのである．



第7図

いよいよ読込みを終わらせるときがきた．すなわち，点線部分のあとに**レ**が現われると**第8図**のようになり，7番地で停止する．オペレーション・スイッチを押せば，インデクス2で定まる本文プログラムの先頭へ飛び，その実行が始まることになる．もちろん，**レ**の部分を作り変えて，停止することなく，そのまま本文の実行に移らせることも，できるわけである．



第8図

上述でおわかりになるように，この方式の靴紐は，大きく分けて，三つの部分からなる．すなわち，主として，先頭番地を指定するための**イ～チ**の部分，本文を含む**リ～ヲ**，本文，**ワ～タ**の部分，読込みを終わらせるための**レ**の部分であり，それぞれの間は，点線部分で分離，また，連結されている．したがって，テープ複写装置などを用いて，いくつかのテープを，点線部分でつなぐことができる．たとえば，本文を延長し

たり，または，異なるいくつかのプログラムを，それぞれに分けて入れたりすることが，1本のテープでできることになり，小型の計算機の場合，なかなか便利である．しかし，こうなってくると，どうも靴紐という呼名は，あまり適当ではないような気がする．

命令コード表

mS	ACC に下から *m* 字読み込め．
mF	ACC の前半分を *m*+1 番地へ，後半分を *m* 番地へ移せ．
mI	インデクス1を *m* にせよ．
mPI	インデクス2を *m* にせよ．
mE	ACC の符号ビットが0ならば *m* 番地へ飛べ．
mG	ACC の符号ビットが1ならば *m* 番地へ飛べ．
mPE	ACC が空白ならば *m* 番地へ飛べ．
mXE	*m* 番地へ飛べ．
mXG	SCC の値をインデクス1に入れ，*m* 番地へ飛べ．
T	番地部をインデクス2で修飾せよ．
R	番地部をインデクス1で修飾せよ．
Q	番地部を SCC で修飾せよ．SCC は命令の所在番地より1多くなっている．

コボル短信（1）*

西村恕彦**

コボル 65 が発表されてから3年ばかり経ち，その改定作業が，アメリカ国防総省データ組織言語協議会で，非常に活発に行なわれている．日本では，ソフトウェア研究会が，国内（および極東）における唯一の組織として，アメリカの動きに追従し，連絡を保ちながら，提案などの作業をしてきたが，研究会が解散されてしまってから，めっきり情報の風通しが悪くなり，意見の交流も低調になった．

それで，本誌の毎号の紙面によって，新しいコボルの情報をお知らせし，問題点を吟味してゆきたいと思う．読者諸氏のご批判やご意見も期待する．

本学会発行の“COBOL 1965年版”の本文は，1968年1月現在のコボルの全容を収めていて，それから7月までの変更点は，付録に項目だけあげてある．

よくご承知のことではあろうが，コボル61以後の，各コボルのおもな変更部分は，次のとおりである．

拡張コボル 61	複数個の答
	分類（SORT 命令）
	報告書機能（GENERATE 命令）
コボル 65	表操作（SEARCH 命令）
	大記憶（乱呼出し，乱処理）

そこで，これらより，さらに以後の変更点のうち，COBOL 1965年版に収められたものから解説してゆこう．詳細はこの本を参照していただきたい．

CALL（呼出し）命令が追加された．これはあるコボル語プログラムから，外部の他のプログラム（コボルであってもなくてもよい）を呼んで実行する．また，LINKAGE（連絡）セクションも追加されたので，外部の他のプログラムから，コボル語プログラムを呼ぶこともできる．

呼び出し方は

```
CALL "PROGRAM-NAME" USING PARAMETER
```

または

```
MOVE "PROGRAM-NAME" TO DATA-NAME
CALL DATA-NAME USING PARAMETER
```

である．

〔宿　題〕

次の命令群を実行したら，最終結果(COMMITTEE)の内容が空白になった．なぜか．

```
MOVE "COBOL"     TO  RESEARCH;
MOVE RESEARCH    TO  COMMITTEE;
```

次の命令群を実行したら，*A* の数値が 10 倍になった．なぜか．

```
MOVE A TO B;
MOVE B TO A;
```

次の命令群を実行したら，*C* の数値が 1/100 倍になった．なぜか．

```
MOVE C TO D;
MOVE D TO C;
```

* COBOL News and Olds (1), by Hirohiko Nisimura (ETL)

** 工業技術院電気試験所