U.D.C. 681.142.8

HITAC 201のソフトウエアシステム

Software System of HITAC 201

堂免信義* 伊藤文武* 河野勝也*
Nobuyoshi Dômen Fumitake Itô Katsuya Kôno

内　容　梗　概

小形汎用計算機 HITAC 201 のソフトウエアは翻訳プログラムとサブルーチンとから成り，ここでは翻訳プログラムを中心に解説している。HISAP 201, PRESAP 201 は汎用，SOS は科学用の翻訳プログラムである。最後に HITAC 201 ライブラリについて述べ，同ライブラリの分類，HIPAC 101 ライブラリとの比較についても述べている。

1. 緒　　言

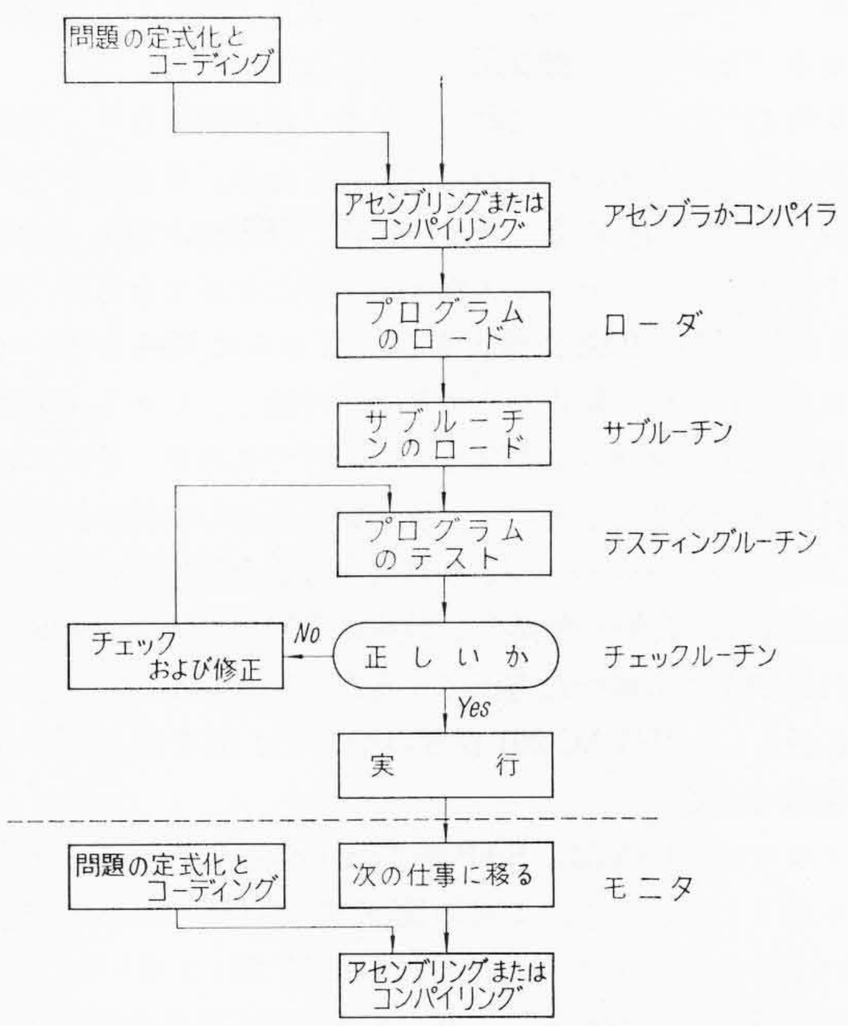
電子計算機システムの中で，電子的，電気的，機械的装置の部分を金物 (hardware) と呼ぶのに対し，本来金物で装備すべきであるのにその機能をプログラムで補うとき，それらプログラムの集まりを紙物** (software) と呼ぶ。一通りの紙物を整備するには大量の紙を消費するのでこの名前をつけたのかもしれない。(これ以降，本文ではこの言葉を用いることにする)。紙物に対するこの定義によれば，一般の関数サブルーチンの類は紙物の中に含まれないことになる。どんな設計者でも $\sin x$ とか $\tan x$ を計算するための金物を「本来つけるべきもの」とは思わないであろう。ところがユーザー(使用者)側から見れば，$\sin x$, $\tan x$ をいちいちプログラムしなければならないとすれば，使用する勇気も失(う)せるであろう。彼らにとってはこれらは金物で「本来つけるべきもの」である。

われわれはユーザー側より見て一般に必要と考えられるプログラムをすべて紙物に含めて考えることにする。もちろん特定のユーザーが特定の業務に適用するために作ったプログラムは紙物の中に含まれない。

紙物システムは，金物の性能，主たる用途，入出力機器の構成などの種々の要因を詳細に検討して作らねばならない。しかも同一本体に対し多種類の入出力機器構成が存在し，それら各構成のシステムに適合した紙物システムを作らねばならない。このことが，大形計算機システムにおいては，紙物システム設計を金物設計以上に困難で大がかりな作業にしている。

2. 紙物の種類

紙物システムはどのようなものを含むべきかを実際に即して考えてみよう。計算機を使用する場合は一般に**第1図**のような流れをたどるものである。その流れの過程に必要なプログラムが紙物に含まるべきものである。



第1図　計算機使用の流れと必要な紙物

第1図の説明

(1)　問題が与えられ定式化が終わるとコーディングにうつる。機械語によるコーディングはたいへんであるから一定の形式にしたがった記号語でコーディングし，これを計算機で機械語に翻訳させる。この翻訳ルーチンをアセンブラ (assembler) またはコンパイラ (compiler) という。コンパイラの方が高級な翻訳ルーチンである。

(2)　翻訳されたプログラム(紙テープに打ち出されている)をローダ (loader) により計算機内に入れる。サブルーチンを使用しているときはそれも入れる。

(3)　テストの結果，誤りがあればチエックルーチンを用いて誤りの個所を発見する。大形計算機の紙物システムでは相当大規模なテスティングルーチンがある。

(4)　実行が終了すると次の問題の処理に移る。高速計算機では大多数のプログラムが数分で終わってしまうので，計算から次の計算への移行をできるだけすみやかにやらないとその間の遊び時間のほうが多くなってしまう。また入出力の同時処理を行なう機械では各入出力機器の遊び時間もなくする必要がある。そのために，多重プログラムを監視したり，計算終了時に適当なメッセージを印刷してオペレータに必要な操作を指示し，オペレータの作業や判断をできるだけ少なくするシステムがある。このシステムをモニタ (monitor) という。

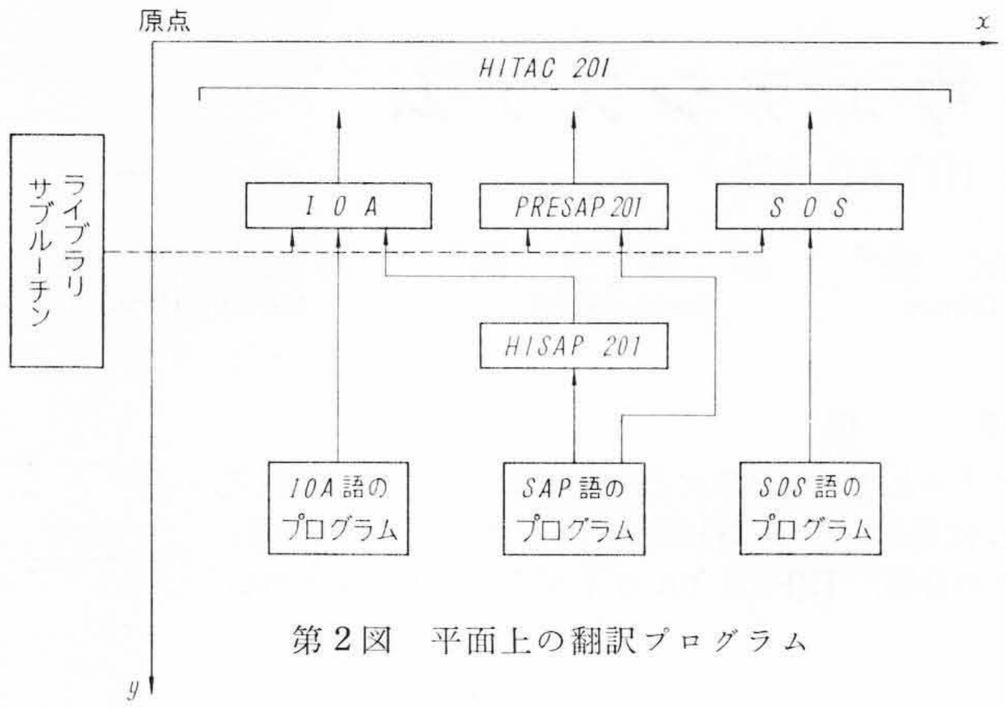
以上のシステムは高速大形機に適用されるもので，HITAC 201 紙物システムは，おもにローダ，アセンブラ，サブルーチン（チエックルーチンも含む）からなっている。

3. HITAC 201 の紙物

紙物はいわばプログラムの部品の集まりであり，プログラマがこれらの部品を適当に使って自分のプログラムを組み立てる必要がある。したがって各部品の規格はそろっていなければならない。もし紙物システムに，全体を有機的に統合する思想がなければ，いくら量があっても使いやすいものにならない。201 紙物システムのうち

* 日立製作所神奈川工場

** 小野田セメント株式会社の和田英一氏の造語であるが，適切な表現であると思うので，ここではこれを使うことにする。



第2図　平面上の翻訳プログラム

の翻訳ルーチンについては，それらの性能と手法に応じて¼平面に位置させることができ，第2図のようになる。

原点から右へ行くにつれて翻訳ルーチンが高級になり，対象とする言語が日常人間が用いている言語に近くなる。原点から下に降りるにつれて翻訳の回数が多くなる。IOA，PRESAP 201，SOS を用いれば1回の翻訳でプログラムを入力することができるが，HISAP 201 を用いれば1度 IOA に翻訳するので，さらに IOA を用いて入力しなければならない。原点からできるだけ遠く，しかも x 軸に近い位置にある翻訳プログラムが望ましいわけであるが，それには高速大形の計算機を必要とする。x，y 両軸から遠い場所に位置する翻訳プログラムもあり得る。つまり高級な言語を何段階かの翻訳を経て，順次低級な言語へ変換し，最後に SAP 語または SOS 語にする。これは 201 でも可能な方法であるが，時間がかかりすぎるので実用性に乏しい。HITAC 201 程度の能力では第2図に示す程度のものが相応である。

プログラマは，IOA 語，SAP 語または SOS 語のいずれでもプログラムを書くことができ，1回の翻訳で計算機内に入力できる。SAP 語で書かれたプログラムは特に HISAP 201 を用いれば，IOA 語へ変換することもできる。このことの利点は後述する。サブルーチンは IOA，PRESAP 201，SOS のいずれを用いても入力できる。ことに PRESAP201 または SOS を用いれば主ルーチンとの結合（後述）まで作ってくれる。これは 201 紙物システムの一特長である。

3.1　ローダとアセンブラ

ローダとアセンブラは対になって存在するものである。プログラマはアセンブラ語でプログラムを書き，それをアセンブラによりローダ言語に翻訳する。翻訳された結果は紙テープにせん孔される。この紙テープをローダを用いて入力する。ローダとアセンブラの機能をかねた翻訳プログラムもある。それを記号入力プログラムと呼び，PRESAP 201 はその一つである。記号入力プログラムのほかにローダとアセンブラのシステムが存在する理由は次のとおりである。

（1）　日常，ひん繁に使うプログラムやサブルーチンの類をいちいち記号入力プログラムで入力するのは，時間がかかって実用的でない。いったんアセンブラによりローダ言語に翻訳しておき，以後使用するときはいつもローダで入力すればよい。ローダによる入力は，記号入力プログラムによる入力よりもはるかに速い。（高速計算機では両者の差が少なくなる）。

（2）　ローダに比べて記号入力プログラムは記憶占有量が大なので不経済である。

もちろん高速大形の計算機では上記2点も問題ではなくなるが，201 クラスの機械では十分，ローダとアセンブラのシステムの存在理由になる。

ローダに要請される基本的能力は次の3点である。

第1表　IOA コード表

コード	機　能	HISAP 201 との関連
L	入力場所を指定する	制御指令 L_0
A	数値の指定	データまたは絶対アドレスの命令語
R	相対アドレスの指定	相対アドレスの命令語
E	読み込んだ命令語を実行する	作業用現場の確保
J	指定のアドレスへ飛越す	制御指令 START
H	読み込みの停止	プログラムの末尾
N	NAMEの指定	制御指令NAME

（1）　プログラムを入力する。
（2）　データを入力する。
（3）　サブルーチンを入力する。

ローダが扱うものは，ほぼ機械語に近いものであるから，プログラムとデータの区別はないものとしてもよい。二進の機械であれば両者の区別が必要であろうが，HITAC 201 は十進表示の機械である。しかしサブルーチンに関してはリロケータブル(relocatable)である必要がある。たとえばあるサブルーチンをあるときは 300 番地から，あるときは 1,000 番地からというように場合によって入力する場所を変える必要がある。HIPAC 101 または 103 のように，SCC（制御計数器）によるアドレス変更が独立に指定できる機械では，この SCC 変更子により相対アドレスのプログラムを書くことができる。ところが 201 にはその機能がないので（IR 変更をするときには SCC 変更ができない）ローダにその機能をもたせることになる。ローダによりプログラムを入力するときにアドレスの変更を行なうわけである。

（1）～（3）はローダの最小限必要な機能であり，さらにどのような機能を付加すべきかは，アセンブラの性格と密接に関係してくる。大形計算機では両者の作業区分が問題となるが，201 の場合は簡単である。201 ローダは，できるだけ入力速度が大で記憶占有量が小でなければならない。このようにして決めたローダが IOA（Initial Order A）である。そしてこの IOA と対になっているアセンブラが HISAP 201 である。

IOA 語は符号を伴った11けたの数字とその後に続く制御文字よりなる。制御文字により IOA は何をなすべきかを知る。制御文字は7種あり，その役割は第1表に示すとおりである。

IOA，HISAP 201 の対による PRESAP 201 に対する利点は下記の3点である。

（1）　サブルーチンが作りやすい。
（2）　プログラムに NAME をつけることができる。
（3）　サムチエックの機能をもっている。

サブルーチンは相対アドレスで書かねばならず，アドレシングがかなり面倒であるが，HISAP 201 を用いれば，記号アドレスをすべて相対アドレス指定にして IOA 語に翻訳する機能をもっているから，普通のプログラムを作るのと同じようにサブルーチンを作ることができる。

プログラムに NAME をつけることは 201 紙物システムで採用された独得の方式である。HISAP 201 の NAME に関する制御指令を使用することにより，オブジェクトプログラムテープの先頭に NAME をせん孔することができる。これによりプログラムテープの識別が明確になるし，サブルーチンの結合にも利用できる（後述）。

SAP 語→IOA 語の翻訳に際して，翻訳されたデータまたは命令語は1語ずつ紙テープにせん孔される。HISAP 201 はせん孔前に，それらを11けたの数値とみなして和を作っている。その和をプログラムテープの末尾にせん孔する。いっぽう IOA でこのテープを入力するとき，IOA は読み込んだデータまたは命令語の和を作り，入力

が終了したときその和をＵＡに表示する。その数とテープの末尾にせん孔してある数値を比較することにより，入力を確認することができる。

3.2 そのほかの翻訳プログラム

PRESAP 201 と SOS があり，いずれも１回の翻訳でプログラムを入力する一種の記号入力プログラムである。

3.2.1 PRESAP 201

この翻訳プログラムの文法は HISAP 201 語の文法と完全にコンパチブル(compatible) である。したがってユーザーは SAP 語でかいたプログラムを HISAP 201 により IOA 語に翻訳することも，PRESAP 201 でただちに記憶に入れることもでき，必要に応じて自由に選べる。

PRESAP 201 が

（1） プログラムをテストするとき，

（2） １度しか演算しないプログラムを作るとき

に有効なことはもちろんであるが，サブルーチンと主ルーチンの結合を作る機能があるので

（3） 主ルーチンを作るとき

にも便利である。また

（4） コンプリートプログラムを作るとき

にもこれを用い，できあがったプログラムを紙テープにダンプしておけば，後の使用に際しての入力が速くできる。

3.2.2 SOS (Science Oriented System)

科学計算用プログラムの作成の便を図った翻訳プログラムである。この言語では，算術演算はいわゆる3-アドレス方式で表現され，長い演算式はいくつかの3-アドレス表現に分解してかかなければならない。

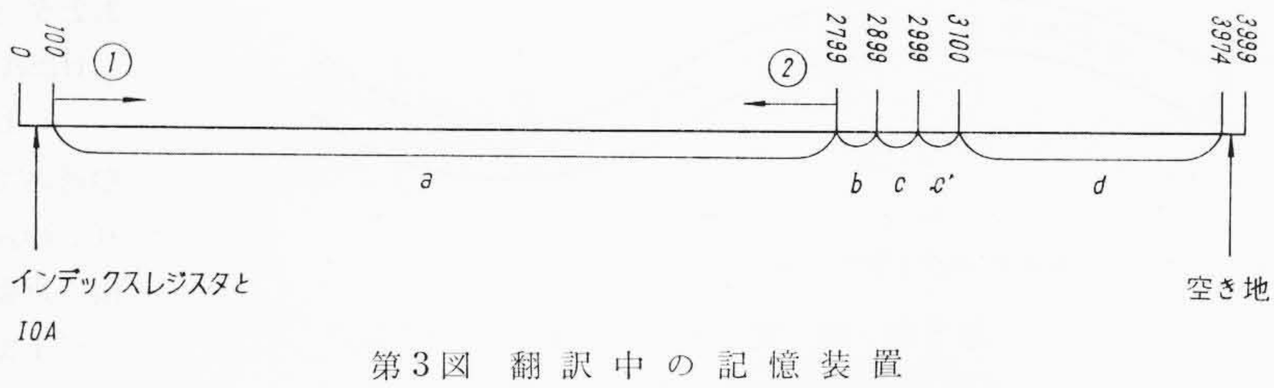
SOS 語による表現の例	説　　明
F) A＋B＝C	A＋B をＣに入れる。
SIN(X)＝Y	Ｘの正弦をＹに入れる。
E＊3.14＝L.	Ｅの3.14倍をＬに入れる。
GO　F.	Ｆという記号で示される表現に飛越す。
READ A, B.	データをよみＡ，Ｂに入れる。
PRINT S (A, B)	形式ＳにしたがってＡ，Ｂを印刷する。
−3.14＝A＝B.	−3.14をＡ，Ｂに入れる。
IF (A) L, M, N.	Ａが負，ゼロ，正であればＬ，Ｍ，Ｎに飛越す。

これらの表現の中で，算術演算を必要とするものはいずれも浮動小数点演算で計算できる。

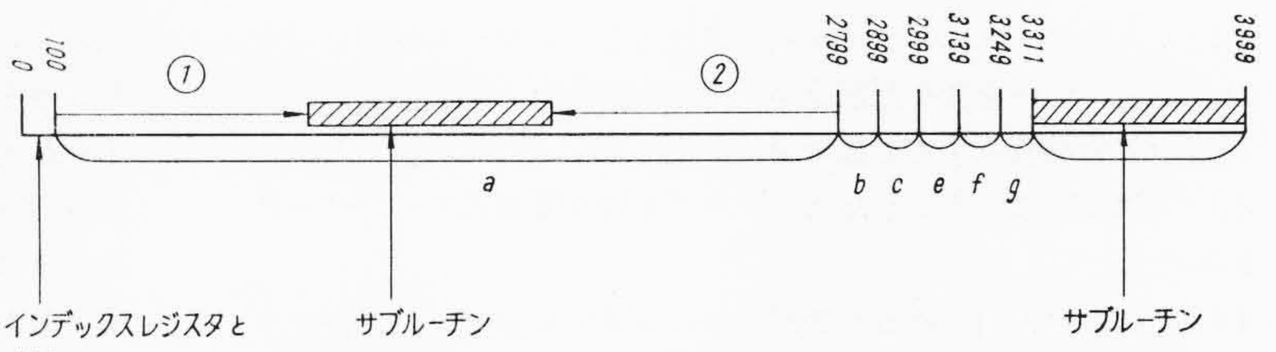
翻訳を１回ですませ，しかもオブジェクトプログラムの容量を大きくするため，システムプログラムは相当窮屈になっている。そのため使用法がやや難解になっているが，小形の計算機でコンパイラまがいのものを，あまり時間をかけずに使おうとすれば，やむを得ないであろう。SOS に採用している記憶配置 (Memory allocation) を説明して，システムの窮屈さの一端を示そう。

（1） 翻訳中の記憶配置（第３図）

a: この部分に翻訳したオブジェクトプログラムを格納する。演算部分が矢印①の向きに，プログラム中に使用するシンボルはａの部分の終点から矢印②の向きに配置される。たとえば A＋B＝C の翻訳にあたって，実際にこの計算をし，結果をＣに入れる手続きのプログラムを矢印①の部分に入れ，Ａ，Ｂ，Ｃの格納場所として矢印②の部分を使う。矢印①と矢印②が交わるような長いプログラムは翻訳できない。



第３図　翻訳中の記憶装置



第４図　実行中の記憶配置

b, c, c′: データやラベルの表として使っている。この表があふれたときは翻訳を停止する。

d: SOS 語→機械語の翻訳を行なう SOS システムプログラムがはいっている。

c′ d は翻訳が終了したら不用になる部分である。

（2） オブジェクトプログラム実行中の記憶配置（第４図）

翻訳が終了するとオブジェクトプログラムがメモリにはいる。しかしオブジェクトプログラムだけでは演算はできないのであって，浮動小数点演算や入出力用のサブルーチンを格納しなければならない。この浮動小数点演算，入出力用プログラムは SOS システムのために特に作ったものではなく，一般のサブルーチンとしてライブラリ（後述）に登録されているものである。ライブラリサブルーチンはできるだけ一般的使用に耐えるような形式に設計してある。

ａ，ｂ，ｃについては(1)で説明したとおりである。ｂ，ｃはデータまたはラベルの表でオブジェクトプログラム実行中にも残っている必要があるが，翻訳プログラムはもはや必要でない。

e: ここに浮動小数点サブルーチン（TAFL という NAME でライブラリに登録してある）を入れる。オブジェクトプログラムの浮動小数点演算はすべてこの TAFL で処理する。

f: ここに浮動小数点用入力ルーチン FIR 1 がはいる。入力の表現はこの FIR 1 ルーチンで処理する。

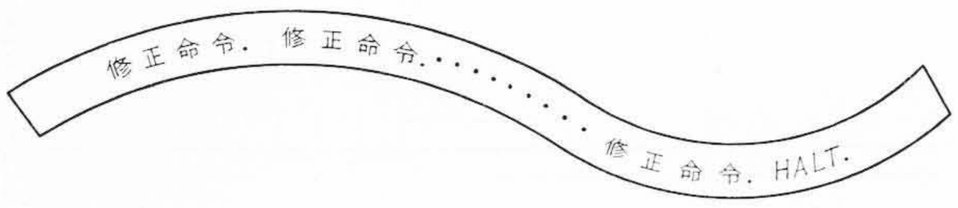
g: ここに浮動小数点用印刷ルーチン FOR 1 がはいる。印刷の表現はこの FOR 1 ルーチンで処理する。

斜線部分: ユーザーが使用するサブルーチン，たとえば sin, tan などをこの部分に入れる。サブルーチンを入力するプログラムは SOS システムプログラムの中にあるので，ユーザーは TAFL, FIR 1, FOR 1 の入力に先だってサブルーチンを入れなければならない。

3.2.3 翻訳プログラムのエラーチエック

プログラムまたはパンチャのミスにより入力テープには誤りがあるものと考えなければならない。この誤りを指摘せずにそのまま誤った翻訳を続けるようでは親切な翻訳ルーチンとは言えないし，安心して使うこともできない。ところで従来の翻訳プログラムには誤りの指摘をするだけで，事後の修正がめんどうなものが多かった。この欠点を除くため 201 システムの翻訳プログラムには簡単で実用的な新しい修正方法を採用している。それは次のような方法による。

（1） 入力テープは光電式テープリーダにかける。



最後の HALT. を忘れないようにする

第5図　修　正　テ　ー　プ

（2）　入力テープにエラーが発見されると，エラーの種類を示すコードとエラーを含んだ部分が一区切印刷され，翻訳プログラムはその部分を読みすてて翻訳を停止する。

（3）　第5図に示すような修正テープを作り機械式テープリーダにかける。

（4）　翻訳を再開させると機械式テープリーダが選ばれ，修正テープが読み込まれ，HALT（翻訳を停止せよという意味の制御指令）を読んで停止する。これで修正はすんだわけである。

（5）　再開させると光電式テープリーダの入力テープが読まれ，再び翻訳を続ける。

以上の修正手続きの間，光電式テープリーダの入力テープは動かす必要がないので操作もしやすい。この考え方はSOS, HISAP 201, PRESAP 201 のいずれにも共通している。HISAP 201 と PRESAP 201 とはエラー表示コードまで共通である。入力テープをはじめから機械式テープリーダにかけている場合でも同じような方法で修正できる。

3.2.4　2個のコード系

201本体には，H-133形と H-134形の2種類の万能入出力装置が接続できる。入力テープはこの装置で作成するわけであるが，ここにコード系が異なるという問題がある。H-133形には55キーあるのに対し，H-134形には43キーしかない。したがって翻訳プログラム用入力テープを作るには，＊，＝，I, O, (,), ♯の7個の記号および文字が不足する。また両形に採用されていてもコードの異なる記号もあるので両形に共通な翻訳プログラムを作ることは不可能である。そこで，文法だけ共通にして，各形の装置を使って作った入力テープを翻訳するプログラムを別々に作成することも考えられる。しかし H-134形の文字，記号ではまともな翻訳システムは作れないので，H-133形のユーザにまで変形文法を強いることになる。これは望ましくないので結局次の方針で進めることにした。

（1）　H-133形を対象にして翻訳プログラムを作る。

（2）　H-134形ユーザーには，H-133形のけん盤せん孔機で入力テープを作ってもらうか，または H-133形で第2表のような代用文字を採用して入力テープを作る。

このテープをプログラムにより H-133形コードのテープに変換する。変換は H-134形でできる。このようにして正式の入力テープを作る。

この方針では H-134形装置のユーザーが1回余計な手続きを経なければならないが，キー数が不足するのであるから止むを得ないと思う。

第2表　H134 形のための代用文字表

翻訳プログラムで使っている文字，記号	代用する文字	翻訳プログラムで使っている文字，記号	代用する文字
＊	メ	(	カ
＝	ニ	)	コ
I	イ	♯	キ
O	オ		

3.2.5　翻訳の速さ

HISAP 201 の翻訳プログラムは

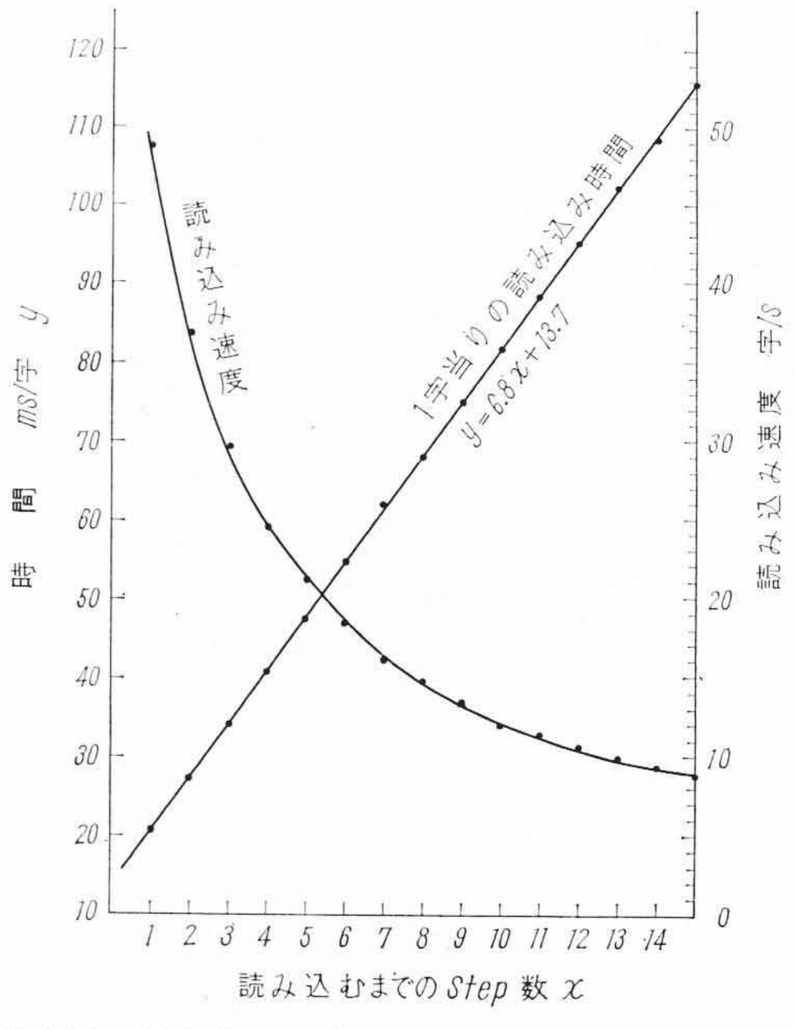
1.3～1.5 秒/命令語

の速さで翻訳する。はじめは3秒弱であったが数回の手直しによりこの速度になった。このように速度を向上することができた理由は重要な意味をもつからここで考えてみよう。はじめは，読み口が1個所しかなかった。1個所で入力し，場合に応じてプログラムの各所へ分岐するという方法を採っていた。つまり文字読み取りという作業を標準化し，1個の入力ルーチンですべてを処理しようとしていたのである。そのほうがプログラムも短縮され，手直しも簡単であろうと思っていた。ところが完成してみるとあまりの低速さに改訂の必要を痛感し，早速，全面的に作り直した。ひん度の高い場合については別処理する方法をとった。読み取りも多数個所にした。たとえば，アドレス，数値，シンボル，命令コードなどの読み込みを別々にした。こうして一挙に速度を倍加することができた。読み口が増したので1回の読み込みに対する分岐の数が少なくなり，したがって分岐の速さも増したわけである。はじめは，read 命令から次の read 命令までの時間が遅くなった結果，光電式テープリーダのスタート・ストップ時間が短くなり，いっそう翻訳速度があがったのであろうと考えた。しかし最高速度 200 字/秒に比べて命令実行速度がおそいので，スタート・ストップに対する影響はないはずである。そうすると翻訳速度の向上はまったくプログラムステップの減少によることになる。これは実験により確認された(第6図)。

第6図の説明

横軸に XRH/1. という命令を実行するまでに実行するプログラムステップ数を示す。XRH 命令(read 命令)自身もその数に入れる。縦軸にこのサイクルに要した時間と1秒間に読む字数を示す。必要時間がステップ数の一次関数になっていること，その階差が 6.7 ms (ちょうどドラム1周分の時間)であることを見ると，スタート・ストップに対する影響はまったくないことがわかる。この実験では XRH 命令以外の命令はすべて次の番地へ飛び越せ，という命令を使用している。

このようにちょっとした注意と改善により処理時間を半減することができる。逆にいえば，プログラムの効率のよい標準化がいかに困難であるかがわかる。



第6図　光電式テープリーダによる読み込みの速さ

SOSの翻訳プログラムの作成にあたっては，HISAP 201の経験を十分に生かし，始めから翻訳の速さをねらった。

3.3 サブルーチンの結合方式

主ルーチンとサブルーチンとのつなぎを結合 (linkage) という。サブルーチン側が必要とするデータは，主ルーチンの帰り番地とそのほかのパラメータである。パラメータが1個の場合を1変数サブルーチン，2個以上の場合を多変数サブルーチンと呼ぶことにする。

3.3.1 1変数サブルーチンの結合方式

変数をUAに入れ帰り番地をIR 8 (Index Register 8) にセットしてサブルーチンへ飛ぶ。この方式が最も多く，全体の50%以上を占めているので，これを標準方式と呼んでいる。たとえば正弦関数サブルーチンを用いれば次のようになる。

XA/x.	x→UA
SI 8/LC+2.	帰り番地→IR 8
J/SIN.	SIN ルーチンへ

←ここに帰ってくる。Sin x がUAにある。

3.3.2 多変数サブルーチンの結合方式

原則として変数自身ではなく変数の場所を示す表を作り，その表の先頭番地をIR 8にセットする。ただし変数がアドレス量または正整数の場合は，変数そのものを表にかくこともある。サブルーチンからは表の末尾に続く場所に帰ってくる。表の大きさが可変であるものは表の最終語にエンドマークを入れておく。

多変数サブルーチンの結合例

(1) 行列の転置の場合

SI 8/LC+2.
J/MTRAN.
m.
n.
A.
B.
←ここに帰る。

(2) 浮動小数点用入力ルーチンの場合

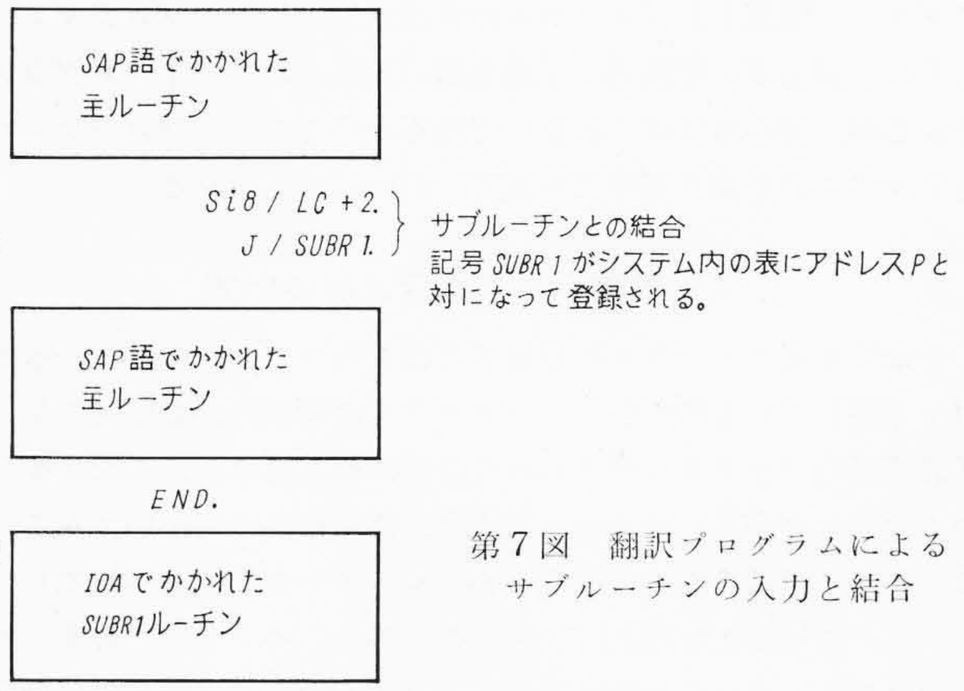
SI 8/LC+2.
J/FIR 1.
A1.
A2.
A3.
} 読み込んだ数値の格納場所
−11……1.
←ここに帰る。

3.3.3 翻訳プログラムによるサブルーチンの入力と結合

SOS と PRESAP 201 には主ルーチンとの結合を作る機能がある。PRESAP 201について説明してみよう。この例ではサブルーチン1 (NAMEはSUBR 1) を入力する。

第7図の説明

ENDによりSAP語のプログラムが終了したことがわかる。このあとにサブルーチンを入れる。このときPRESAP 201にはサブルーチンを格納すべき場所の先頭番地Qがわかっている。IOAは20～79番地にあるからこれを多少変更してコントロールをIOAにわたす。IOAはサブルーチンを読みはじめる。サブルーチンの先頭にはNAMEがせん孔してあり，IOAはこれを読んでコントロールを翻訳プログラムの方に返す。翻訳プログラムの方で，そのNAMEをキー（この場合はSUBR 1がキーである）にしてシステム内の表を検索する。一致するものがあったらそれに対し記録されているアドレス（この場合はP番地）の番地のアドレス部にQというアドレス（SUBR 1の先頭番地）を書く。以上の処理が終わると再びIOAにコントロールをわたす。IOAはサブルーチンを全部読み，最後にサブルーチン終了を示すコードを読んで翻訳プログラムの方に帰ってくる。このとき次のサブルーチンを格納する場所の先頭番地Q′がわかっている。同様な手続きで次々にサブルーチンを入力することができる。SOSについても同じように説明できる。



第7図 翻訳プログラムによるサブルーチンの入力と結合

第3表 HITAC 201 ライブラリの分類

分類	記号	例
算術演算	A	浮動小数点演算，複素数演算
チェック	C	TRACE 1, SNAP SHOT 2
微分方程式	D	Runge-Kutta-Gill 法による常微分方程式の解
関数	F	初等関数サブルーチン
インプット	I	IOA, HISAP 201
行列演算	M	行列の転置，連立一次方程式の解
アウトプット	O	データの印刷，命令の読みだし

第4表 ライブラリの比較

HITAC 201					HIPAC 101				
分類	プログラム数	割合 (%)	語数	割合 (%)	分類	プログラム数	割合 (%)	語数	割合 (%)
A	9	10.5	1,308	12.3	A	7	6.1	1,405	11.5
C	4	4.6	658	6.1	C	8	7.0	407	3.3
F	45	52.6	3,855	36.3	F	28	24.6	1,071	8.7
I	9	10.4	3,352	31.5	I	26	22.8	6,083	49.7
M	11	12.7	758	7.1	M	2	1.8	520	4.2
O	6	6.9	547	5.1	O	43	37.7	2,765	22.6
D	2	2.3	169	1.6					
SUM	86	100.0	10,647	100.0	SUM	114	100.0	12,251	100.0

3.4 HITAC 201 ライブラリ

紙物システムの各プログラムは番号をつけてライブラリに登録されている。プログラムは機能に応じ7分類されている(第3表)。

ライブラリの構成はバランスのあるものでなければならないが，金物機種の違いによりライブラリの構成も非常に異なる。HITAC 201と同じ規模であるHIPAC 101と，ライブラリの内容を比較してみよう。(第4表)

101ライブラリの方には入出力に関するプログラムが非常に多い。全体の6～7割を占めている。一方201の方は全体の2～4割にすぎない。これは101が二進表示であり，201が十進表示であることによる。

各プログラムには詳細な仕様書とSAP語で書かれたプログラムシートがあり，これらを集めてHITAC 201ライブラリの説明書ができている。仕様書，シートはルーズリーフ式に取りはずしできるようになっており，先頭に見出し表，次に機能，結合，範囲，精度，方法，規約，操作法，注意などがかいてある。

そのほかの印刷物として，プログラムの簡単な説明を集めた「ラ

イブラリ一覧表」とプログラムの手法を解説した「プログラムマニュアル」がある。経験者は「命令語の説明」だけでもプログラムができるが，初心者にはこれだけでは不十分である。プログラムマニュアルはこれを補う意味で相当ていねいにかいてある。

4. 事務用プログラムについて

少なくともライブラリに登録するほどのプログラムは，一般的使用を予想しているのだから，プログラムの標準化が必要である。科学計算用のプログラムがきわめて自然に標準化され，むだが少ないのに比べ，事務用の場合は，定式化，標準化がやっかいである。このことが小形計算機における事務用ルーチンの作成を困難にしている。しいて標準化を図れば，多くの例外や，パラメータのすべてを管理するルーチンが余分に加わり，メモリを多く占有し，処理時間が大幅に増す。たとえば磁気テープを考えてみよう。201の磁気テープは1kc(1,000けた/秒)の速さであるから1語(12けた)を読むのに12msかかる。この1語のデータを扱うにはUAに持ち込まねばならない。ところが1語をUAに持ち込むだけで12 ms程度かかる。標準化したプログラムで扱えば，磁気テープが実際に動く時間に比べて内部処理の時間が相当多くなることが予想される。このことは3. 2. 5で述べたとおり，すでにHISAP 201の翻訳プログラム作成時に経験ずみである。大形高速計算機であれば入出力速度に比べて内部処理速度が圧倒的に速いので，標準化によるムダも問題にならない。結局201程度の機械ではケースバイケースにプログラムを作る方が実用的である。また事務用のプログラムは，業務が固定しているので，いちど作っておけば後日の変更はあまり必要としないものである。したがって多少の労は払ってもアセンブラを用いて業務に最適のプログラムを作るべきである。以上のような理由によりHITAC 201ライブラリは事務用プログラムを持っていない。

5. 結　　言

201紙物システムはHITAC 102，HIPAC 103の紙物システムに比べるとささやかなものであるが，それでも1年半に近い月日を費している。われわれはこの経験を生かし，次のシステムの開発にあたっては，より良いシステムをより短い期間で作りたい。目下のところ，紙物システムの作製はまったく手工業的段階である。これを機械化することは当分の間不可能であろうが，少数のプログラマがアトリエで芸術品を作っているような現状から脱脚して，せめてマニュファクチャ（問屋制手工業）の段階に一日も早く進歩したいものだと思っている。

終わりに，システムの開発にあたって終始ご指導をいただいた中央研究所第83研究室，神奈川工場およびコンピュータ事業部企画部サービス課の関係者のかたがたに心から感謝したい。また，当初共同で作業を進めた小竹秀幸氏，上原一矩氏にシステム完了を報告できることをとてもうれしく思うものである。

参　考　文　献

201に関しては，
（1） HITAC 201 命令語の説明
（2） HITAC 201 ライブラリ
（3） HITAC 201 プログラムマニュアル
プログラミング一般に関しては，
（4） D. D. Mc Cracken : Digital Computer Programming, J. Wiley (1.957)
ソフトウエアの具体例としてはHITAC 3010のもろもろのプログラムが非常に参考になる。

特許の紹介

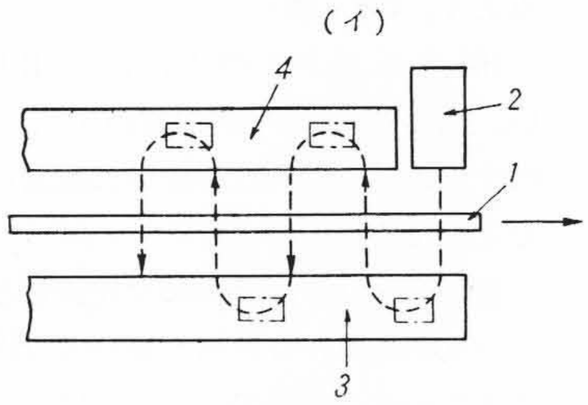
特許第277902号（特公昭35-16754）

前川明嗣

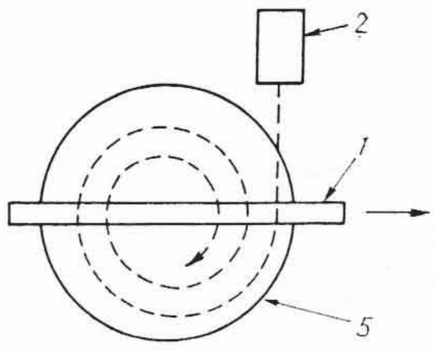
放射線の反覆照射法

最近特に高分子材料などの特性を放射線照射によって改善せしめることが有用視せられている。しかし従来一般に高エネルギーの放射線においては特に飛程が大であるために，一度被照射物体を照射透過した放射線は，なお相当のエネルギーを残在せしめているにもかかわらず無駄に棄てられている欠点があった。

この発明は従来無駄に棄てられていた放射線の残在エネルギーを有効に利用せしめるものであって，たとえば**第1図**に示すように矢印方向に移動せられるたとえばコンベア上の被照射物体1は放射線源たとえば電子線源2によって物体1の進路に対して側方より電子線を照射するようになされてある。ここでさらにまた，被照射物体1を透過した電子線は磁石3および4によって点線で示されるように弯曲蛇行せられるので，矢印方向に移動する物体は表裏両面より反覆照射せられることとなるから被照射物体の照射度は高くなるのみならず，放射線エネルギーの利用効率を著しく向上せしめることができる。なお，**第2図**は他の実施例で磁石片5が円形をなし，物体1を反覆照射せしめ得る方法である。　　（水本）



第1図



第2図