自動フローチャーティング*

山本征一郎** 山口楠雄***

Abstract

A flowchart is a most useful tool for program debugging and documentation. But it is often a tedious and cumbersome job to draw a flowchart. Thus, automatic flowcharting is very useful. This automatic flowcharting program has the following features. First, instruction sequence is divided into blocks and groups. Each block stands for one processing, and groups are used for determining the location. The location of each group is determined one after another as the drawing of the flowchart advances. The flowline consists of one vertical line which takes one of the fixed positions and two horizontal lines which can take any position.

1. まえがき

知的作業の手順，とくに情報処理の手順を示す手段としてフローチャートは最も簡潔かつ明確なものである．このフローチャートの目的とするところは，コーディングを行なう際の最終資料，ディバッグの資料，あるいはプログラムのドキュメンテーションなどであり，プログラムの作成には欠かせないものである．ところが，フローチャートをかくということは，なかなか面倒であり，また，労力のいる仕事である．とくに，プログラムを最も詳細に記述するディテールド・フローチャートのように，プログラムに直接対応しているものでは，フローチャートの量が多い上に，訂正も頻繁に行なわれるで，その面倒さと労力は並大抵のものではない．したがって，このようなフローチャートが，プログラムのリストをとるのと同様に手軽に得られれば，プログラム作成に要する労力が大幅に軽減され，大いに役立つであろう．このようなわけで，十数年以前から自動フローチャーティングに関する研究が行なわれてきた．これらには，プログラムの解析を主とするもの[1)~3)]から，実際にフロチャートを得るプログラムに関するもの[4)~6)]まである．なお，この中には，実際に一般のユーザの利用に供されているもの[6)]もある．

本論文において述べる方式は，つぎのようなものである．まず，ソースプログラムを二つの観点，すなわち各命令の果たしている機能およびコントロールの流れという二つの観点から解析し，ブロックとグループをつくる．ブロックは一つの機能を果たしている命令列をまとめたもので，一つのフローチャート・シンボルに対応するものである．一方グループは，プログラムをフローチャートとして図面上に配置していくときの単位となるものである．配置の方法には逐次方式ともいうべき方法をとる．これは一つのグループを図化し終えるたびに，つぎに図化するグループとそのかき始めの位置を決めていく方法で，配置に関するメモリが大幅に節約できる方法である．フローラインは，すべてのグループを図面にかき終えてから，それまでにつくられた分岐命令に関するデータを参照しながら引いていく．なお，フローラインは，任意の位置をとる横線2本と，あらかじめ定められた位置の中から選んで引く縦線1本から構成し，プログラムの簡単化をはかっている．

この方式で狙ったものは，見やすいフローチャートを簡単な方法で得るということである．このため出力機器は，原則として，良質の図が得られる X-Y プロッタを用いることとした．ところが従来の方式では，各ブロックの配置をあらかじめ定めてから出力するため，配置に関するデータが，かなり膨大になるし，プログラムも複雑になる．またフローチャート・シンボルの大きさを変えたりすることも容易でない．このような方式は，ラインプリンタのように，後戻りのでき

* Automatic Flowcharting, by Seiichirô YAMAMOTO and Kusuo YAMAGUCHI (Institute of Industrial Science, University of Tokyo)
** 日立製作所神奈川工場
*** 東京大学生産技術研究所

ない出力機器を用いる場合はよいが，X-Y プロッタのように，後戻りのできる出力機器を用いる場合には，あまり適当な方式ではない．そこでとったのが上に述べた遂次方式である．遂次方式では，一つブロックができるたびにそれを図にしていき，一つのグループを図化し終わったところで，つぎに図化するグループとその位置を定めていくので，配置に関するデータがごくわずかですむし，またフローチャート・シンボルの大きさも内部に入れる文字の多寡により簡単に変えることができる．なお，ラインプリンタを出力機器として用いる場合は，この遂次方式を用いてもメモリの節約はできない．

ところで，このプログラムの対象とする言語はアセンブラ言語*1とした．これは，アセンブラ言語には，利用者が限られる，あるいは機械に左右される言語であるという欠点があるものの，先に述べたフローチャートの効用が最も大であるし，またブロック化など問題点の多いものであるという理由による．なお，コンパイラ言語を対象とするときは，プログラムの解析のやり方を少し変えるだけで，基本的なところはそのまま利用できるであろう．

2. フローチャート化の方式

ディジタル・コンピュータのプログラムをフローチャートにする際重要なことは，処理の内容を適当な図形を用いてわかりやすく表現すること，一次元に配列されているプログラムを二次元の図面に配置し直して，各処理の間の関係を見やすいものとすること，それにプログラムのコントロールの流れを図にしてわかりやすくすることである．フローチャートの良否は，以上のことをどのように行なうかで決ってしまうといっても過言でない．そこでこの章では，これらの処理の方法に重点を置いて，フローチャート化の方式について述べる．

2.1 ブロック化

ブロックは，プログラムの処理の内容をフローチャートで示す際の単位となるものであるが，その示すところは，フローチャートのレベル*2によって異なってくる．

アセンブラ言語のプログラムをフローチャートにする場合，一命令ずつ図にしていったのでは，たとえ，それがディテールド・フローチャートであっても細か過ぎるので，ブロック化する必要，すなわち幾つかの命令をまとめてブロックをつくる必要がある．命令列をまとめる方法としては，ブロックが一つの処理を示すということから，ある一つの機能を果たしている命令列をまとめるということが考えられる．ところが，命令自身にはその命令の目的とするところなどは明記されていないので，命令列のどこからどこまでが一つの機能を果たしているか，コンピュータに判別させるのは一般に困難である．そこでここでは，つぎのような原則を定め，これに従ってブロック化を行なうことにした．これは，一つの機能を果たしていると明らかにわかるものだけをまとめ，まぎらわしいものはまとめずに表わすことにしたもので，フローチャートのレベルとしては，ディテールド・フローチャートに相当するものである．

2.2 ブロック化の原則

（1） ラベルのついている命令，あるいはそこに分岐してくる命令があるときは，その前の命令までを一区切りとしてブロック化する．

（2） シフト命令，ステイタス・フリップフロップのセットあるいはその状態の格納命令，さらには，コントロール命令などは一命令一ブロックとする．

（3） 演算命令は，累算器への置数命令 (LOAD) から格納命令 (STORE) までを，一つの機能を果たしている命令列としてまとめる．

（4） インデックス・レジスタへの置数，あるいはその内容を格納する命令は，同じ命令が続く限りまとめていく．

（5） 条件付分岐命令*3は，以下に続く分岐命令を分岐方向が三方向を越さない限りまとめていく．もちろん，分岐条件の範疇が異なるときはまとめない．

（6） 以上の原則が互いに矛盾するときは，上位の原則を優先させるものとする．

以上の原則に基づいてブロック化を行う方法のうち演算命令のブロック化を中心とする部分を **Fig. 1** に示す．Fig. 1 に用いられている記号のうち，BRD はブランチ・データの略号である．そこには各分岐命令に関するデータが格納されており，1 分岐命令当り 16 語が割り当てられている．その構成は付録 I (Fig. 13) に示すものである．また BLD はブロック・データの略号であり，そこにはブロック化の際に作成されるデ

*1 FACOM 270-30 FASP

*2 詳細な順に，ディテールド・フローチャート，ゼネラル・フローチャート，プロセス・フローチャートの3種に分類できる．

*3 本論文においては，コントロールが飛ぶことのある命令は，すべて分岐命令と呼ぶことにする．

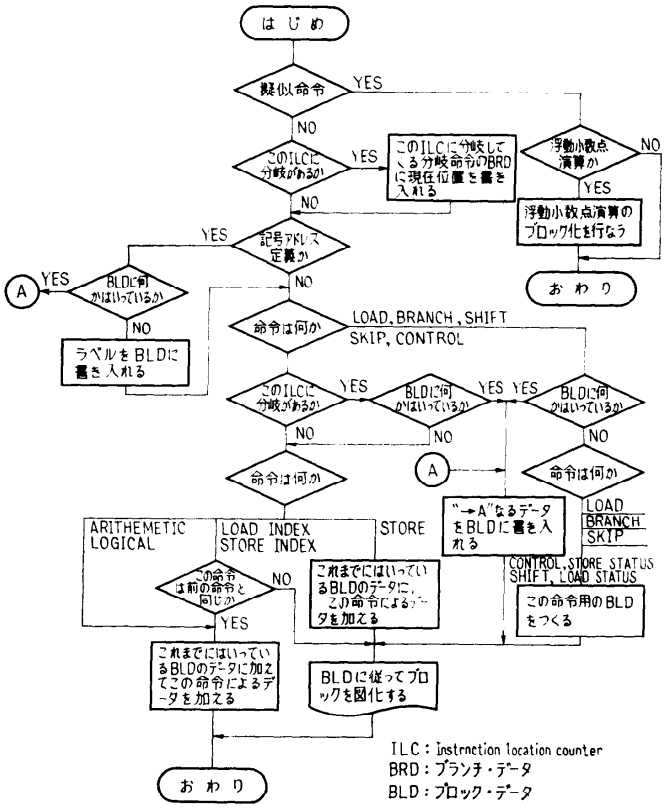


Fig. 1　ブロック化（演算命令を主とする部分）

ータが格納される．それらのデータとは，使用するフローチャート・シンボルの種類，大きさ，その書き始めの位置，現在のペンの位置，分岐の条件，シンボルの内部に入れる文字などである．なお，このメモリは，各ブロックに共通に使用され，シンボルの中に入れる文字を作成するための作業領域を含めて，約 300 語である．

この他，この方式のブロック化で特記すべきこととして，コメントの利用がある．つまり引数あるいは複数個の出口のあるサブルーチン呼出し命令では，それが標準的な形をしていないと，後に続く命令がその引数あるいはその途中の出口であるかどうかわからないことがある．そのような場合，コメントを入れることにより，それがその引数あるいは出口であることを示し，正しいフローチャートが得られるようにしてある．また分岐先が実行前には定まっていない分岐命令の場合，コメントがあれば JIS 規格に準拠したフロチャート・シンボルでかかれ，コメントがなければ分岐先のアドレスだけを示すようにしてある．

2.3　グループ化

ティジタル・コンピュータのプログラムは，その性質上，一次元に配列しなければならない．これをそのまま図にすると，**Fig. 2** のようになる．これでもフローチャートといえないことはないが，図面の持つ二次元的性質を利用していないので，人間にとって見やすいものとはなっていない．そこでこれを二次元に配置し直してみる．Fig. 2 を見れば容易に気がつくことであるが，そこには前の命令からコントロールの流

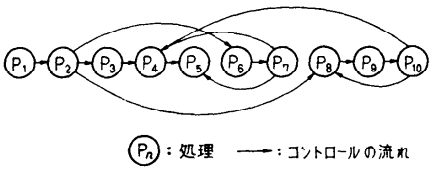


Fig. 2　一次元に配列されているプログラムをそのまま図にしたもの

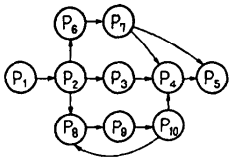


Fig. 3　Fig. 2 を二次元に配置し直したもの

れてこない命令があり，それらは必ずしもその位置になくてもよい．したがって，二次元に配置し直す際には，つぎの命令にコントロールが移らない命令によって区切られた一連の命令をひとまとめにして適当に配置し直せばよいことがわかる．このようにして配置し直したものが **Fig. 3** である．

本方式では，以上に示したような命令列のまとまりをグループと称し，配置決定の際の単位とした．このグループをつくるには，分岐命令に注目してプログラムを解析するのであるが，そのアルゴリズムはつぎのようなものである．すなわち，無条件分岐命令は，それがスキップ命令の直後にある場合，あるいはサブルーチンの途中の出口となる場合を除いてグループを区切るものとする．ただし条件付分岐命令はグループを区切らない．なお条件付分岐命令でグループを区切らないのは，プログラムとフローチャートの対応をしやすくするためである．以上のアルゴリズムを図示すると **Fig. 4** に示すようになる．

2.4　配置の方法

プログラムを図面に配置するには，グループを単位として行なうことは先に述べたとおりである．各グループを図面上にどのように配置するかということは，フローチャートのわかりやすさ，美しさを決める最も重要な要素の一つである．しかし，フローチャートのわかりやすさ，美しさとそれを実現するプログラムの簡単さということは，一般に相入れないものである．そこでここでは，できるだけ双方を満足するようつとめ，逐次方式ともいうべきつぎのような方式をとった．

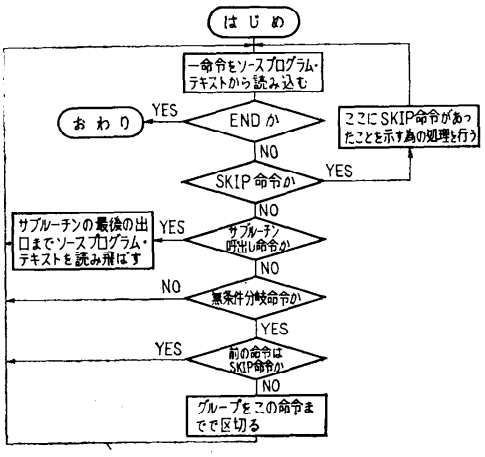


Fig. 4　グループ化のアルゴリズム

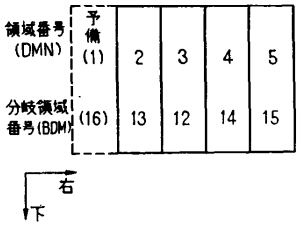


Fig. 5　図面の分割とその領域番号および分岐領域番号

（1）　図面は **Fig. 5** に示すように，四つの領域に分割し，かき始めの位置は左から2番目，すなわち領域3からかき始める．

（2）　図の流れる方向は下方向のみとする（フローラインはもちろん上方向にもいく）．

（3）　図化するグループとそのかき始める位置は，一つのグループを図化し終えるたびに逐次定めていく．

（4）　つぎに図化するグループを決めるには，グループ・データ*4に与えられている領域番号あるいは分岐領域番号*5による．

（5）　領域番号は，どの領域に図化するか決ったグループに対して与えられ，分岐領域番号は，グループの途中から分岐するグループに与えられる．

（6）　図化するグループを選ぶには，まず領域番号の与えられているグループを捜し，それがあればそれを，なければ分岐領域番号を与えられるグループを選ぶ．

（7）　分岐領域番号が複数個のグループに与えられているときは，それらのうち最も高い位置から分岐してくるものを選ぶ．

（8）　分岐領域番号は，それが与えられていても，まだ左右どちらの領域に分岐するかは決っていない．

---

*4　各グループに関するデータを格納してあり，1グループあたり 12 語を割り当ててある．その構成は付録 I（Fig. 12）に示してある．

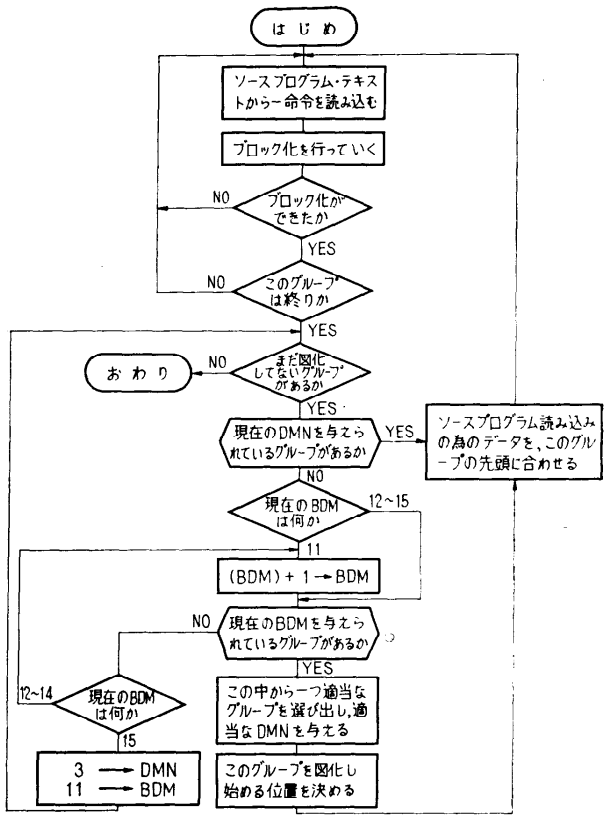
*5　これらは各領域につけられた番号であり，(Fig. 5) に示すようになっている．なお，分岐領域番号は，あるグループがどの領域から分岐してくるかを知るためのものである．

それを決めるには，そのグループの途中から他のグループに分岐するか否か，すでに図化されたグループとの位置関係，すぐ上にある条件付分岐命令から分岐したグループはどちら側にあるかなどを勘案して定める．

(9) 各グループを図化し始める位置は，それがすでに図化されたグループとオーバーラップしない限り，隣の領域にあるそのグループへの分岐点と同じ高さとする．複数個の位置から分岐してくるときは，すでに図化されたものとオーバラップしないもののうち，最も高い位置から分岐してくるものの位置とする．すべてオーバーラップするときは，オーバーラップしない位置まで下げる．

以上の方法に基づいて，各グループの配置を決めるアルゴリズムを **Fig. 6** に示す．

これを説明するとつぎのようになる．まず，最初に図化するのはプログラムの先頭にあるグループで，かき始めの位置は領域3である．このグループの図化が終わると，つぎに図化するグループとそのかき始めの位置を決めるための処理に移る．そこでは，まだ図化していないグループの中に，現在の領域番号のレジスタ（以下 DMN と略記）の領域番号を有するグループがあるかどうか調べ，そのようなグループがあれば，それを下に続けてかいていく．そうして，これと同様な処理を領域番号の与えられているグループがなくなるまで続ける．それが終わったら，つぎは現在の分岐領域番号のレジスタ（以下 BDN と略記）の値を一つ進め，その分岐領域番号を有するグループ，すなわち隣の領域に分岐するグループを捜し出す．該当するグループが二つ以上ある場合は，(7)に示した方法に従って一つグループを選び出し，それを(8)，(9)に示した方法により図化する．このグループの図化が終わ



DMN：領域番号のレジスタ
BDM：分岐領域番号のレジスタ

Fig. 6 配置決定のアルゴリズム

ったら，前と同様に，現在のDMNの領域番号が与えられているグループを捜し出し，そのようなグループがあれば下に続けて図化していき，なければ現在のBDNの分岐領域番号を与えられているグループの中から(7)に示した方法でグループを一つ選び出し，(8)，(9)で示した方法で図化していく．このようにしていって，現在のBDNの分岐領域番号を与えられているグループがなくなったら，再びBDNを一つ進め，その分岐領域番号を与えられているグループの図化に移る．以上の処理を繰り返し行なっていき，コントロールの流れが直接つながっているグループをすべて図化し終えたら，残ったグループを改めて上に述べた方法で図化していく．これを繰り返していけば，すべてのグループが図化されることになる．

2.5 フローライン

フローラインの引き方は，配置の方法と共に，フローチャートのわかりやすさ，美しさなどを決める重要な要素である．ここでとった方法は，プログラムを簡単にすることに主眼を置き，利用するデータの形なども考えてつぎのようにした．

(1) フローラインは，すべてのグループを図化し終えてから，それまでに作られたブランチ・データに従って引いていくものとする．

(2) 各領域の両側に，**Fig. 7** に示すような3本ずつのフローラインを引くための場所を確保しておき，フローラインを引くときは，このうち1本を選択して引く．したがって，フローラインは，縦線1本と横線2本とから構成することになる．このフローラインを引く場所には6～31の番号が与えられており，フローラインの縦線を引く場所の指定は，この番号，すなわちライン番号を用いて行なう．

(3) フローラインを引くのは，分岐先が同一領域内もしくは隣の領域内にある場合とする．さらに遠くの領域に分岐する場合は，頁内結合子で結合することにする．

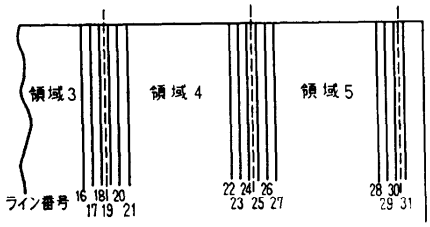


Fig. 7 ライン番号（部分図）

(4) ライン番号を与える順序は，隣の領域に分岐するものを先にし*5，同一領域内に分岐するものを後にする．また，それらの中ではループの短かいものから先*6にする．

(5) ライン番号の与え方は，分岐先の領域に従って，つぎの二つの場合のようになる．

(イ) 隣の領域に分岐先がある場合

二つの領域に囲まれたライン番号のうち，両端の2本を除く4本をとり，端から順々に，それがすでにライン番号を与えられているフローラインとオーバーラップしないかどうか調べ，オーバーラップしなければそのライン番号をそのフローラインに与える．

(ロ) 同一領域内に分岐先がある場合

その領域の両側ならびにその外側の1本ずつの計8本をとり，左右交互に内側から順々に，それがすでにライン番号を与えられているフローラインとオーバーラップせず，また同じ位置から出るフローラインと同じ方向にならないものであるかどうか調べ，そのようなライン番号があれば，それをそのフローラインに与える．

なお(イ)，(ロ)で最後まで条件を満足するライン番号がなければ，頁内結合子によって結合することにする．

3. 自動フローチャーティング・プログラム

つぎに以上述べてきた方式を実現するプログラムについて述べる．この自動フローチャーティング・プログラムは，ソースプログラムを3度読み込む3パス方式をとっている．その構成は **Fig. 8** に示すものである．アセンブラ的な処理は PASS 1，PASS 2 で行ない，フローチャートを得るための処理は，主として PASS 2，PASS 3 で行なっている．

PASS 1 は **Fig. 9** に示すようになっている．ここでは，まずカードからソースプログラムを読み込み，ラベルがあればそれをラベル・テーブルに登録する．つぎにはその命令の解読を行ない，その命令に対応する命令番号*7を与え，それを現在の ILC (Instruction Location Counter)，その命令の予測長*8 (SL) の値と

*5 同一領域に分岐するときは，左右どちら側にでも分岐できるが，隣の領域に分岐する場合は，左右どちらか一方にしか分岐できないので，前者の方が自由度が大きい．

*6 ループの短いものから順に内側のライン番号を与えていけば，フローライン同士の交差が少なくなる．

*7 以後の処理に便利なように各命令に対し与えられる番号である．

*8 各命令がフローチャート上で占めると予想される長さである．これからグループの予測長を求め，配置決定の際に用る．

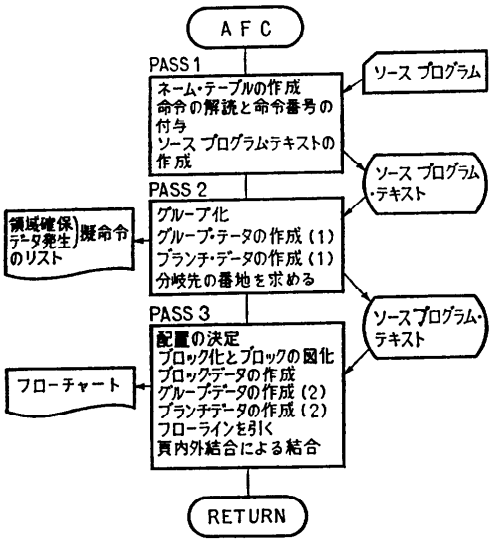


Fig. 8 フローチャート自動作成プログラム

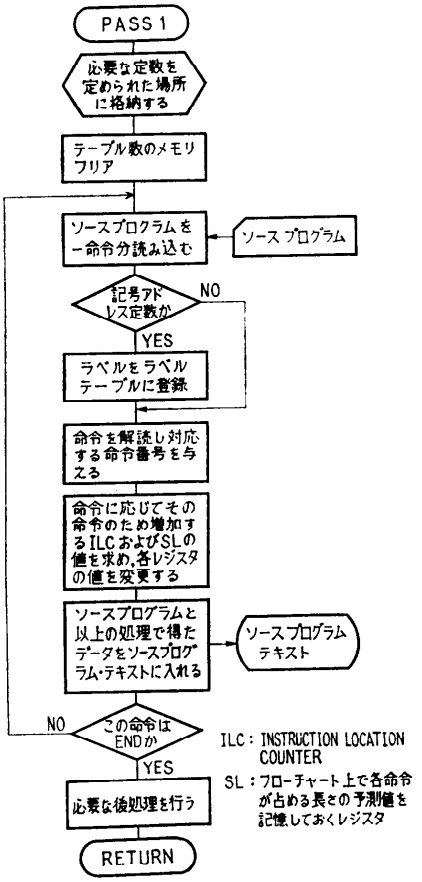


Fig. 9 PASS 1 のプログラム

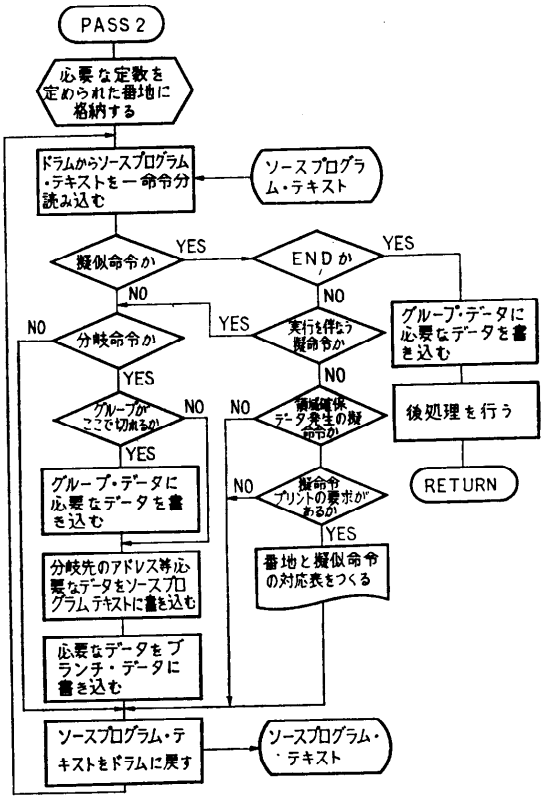


Fig. 10 PASS 2 のプログラム

共にソースプログラム・テキスト*9に書き込む．また，解読した命令に従って ILC および SL の値を増加させる．このような処理をカードを1枚読み込むたびごとに行なっていき，END がきたら PASS 1 の処理を終える．

PASS 2 は **Fig. 10** に示すようになっている．ここではまず，ソースプログラム1命令分読み込み，それが分岐命令かどうか調べる．分岐命令でなければそのままつぎの処理に移る．分岐命令であれば，そこでグループを切ることができるかどうか調べ，できればグループ化*10を行ない，できなければそのままブランチ・データの作成に移り，その分岐命令のアドレスなどのデータをそこに書き込む．この間，分岐先のアドレス，分岐の条件，この命令のブランチ・データのアドレスなどをソースプログラム・テキストに書き加え

*9 ソースプログラムとそれに付随するいくつかのデータから成るもので，その構成は付録 I (Fig. 14) に示してある．

*10 グループ化はグループ・データをつくることによって行なわれる．ここでは，グループ番号，グループの始めと終わりの ILC，その他がかき込まれる．

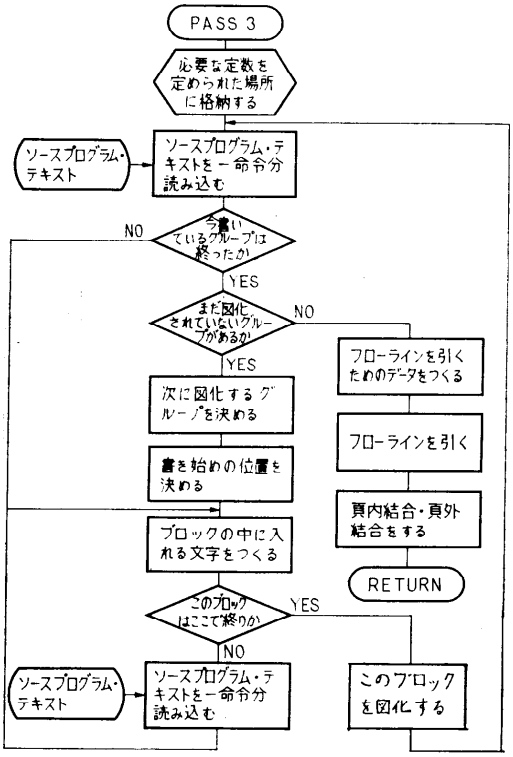


Fig. 11　PASS 3 のプログラム

ていく．このようにしていって END がきたら PASS 2 の処理を終わる．

PASS 3 は **Fig. 11** に示すようになっている．ここではまず，先頭のグループを読み込みながらブロック化を行なっていく．ブロックはそれが一つできるたびに図化される。このようにしていって最初のグループを図化したら，つぎに図化するグループとそのかき始めの位置を定め，そのグループを図化する．これを続けていき，すべてのグループを図化し終えたら，つぎはフローラインを引くための処理に移る．フローラインを引くのに必要な位置データは，ブロック化の過程で，ブランチ・データにかき込まれているので，これらを用いて，さらにライン番号，分岐方向を定め，フローライン（頁内結合子による結合も含む）を引いていく．フローラインを全部引き終わったら PASS 3 の処理は終わり，フローチャートの作成が終了する．

4. むすび

以上述べてきた自動フローチャーティング・プログラムを用いて，実際にフローチャートをかかせた例は

	グループ番号
+1	領域番号 ／ 分岐領域番号
+2	このグループの長さの予測値
+3	このグループの先頭の ILC
+4	このグループの最後の ILC
+5	予　　備
+6	このグループの先頭に分岐してくる命令の ILC (No.1)
+7	〃 (No.2)
+8	〃 (No.3)
+9	〃 (No.4)
+10	〃 (No.5)
+11	このグループの先頭の命令のソースプログラム・テキストのアドレス

Fig. 12　グループ・データ (GPD)

	ブランチ番号
+1	この分岐命令の所属するグループの番号
+2	分岐先の命令の所属するグループの番号
+3	この分岐命令の ILC
+4	分岐先の ILC
+5	グループがここで終るか否か／分岐の方向
+6	ライン番号／頁内結合子番号／頁外結合子番号
+7	分岐の条件
+8	分岐点の X 座標
+10	分岐先の X 座標
+12	分岐点の領域番号
+13	分岐先の領域番号
+14	フローラインの縦線の長さ

Fig. 13　ブランチ・データ (BRD)

付録Ⅱに示してある．この他多くのプログラムによってテストを行なったが，ほぼ期待したどおりの結果が得られた．ただ，まれではあるが，だらだらと縦方向に長いフローチャートになってしまうことがあった．これはプログラム簡単化のためとった逐次方式，あるいはプログラムとの対応をよくするために，条件付分岐命令でグループを区切らない方法をとっていることによるものである．なお，これまで問題とされてきた，複数個の出口のあるサブルーチン呼出し命令，あるいは分岐先が実行中に定まる命令などのブロック化はコメントを利用して解決したが，実行中に命令が変わるものや置数命令を用いて実行命令をつくるもの*11については未解決である．

参考までに述べると，この自動フローチャーティング・プログラムに要したメモリは，ソースプログラム・

*11 これはコメントの利用，あるいは機械語との照合により解決できるが，ここで　はまだプログラムされていない．

テキストを除いて，約24K語である．また50ステップのプログラムをフローチャートにするのに要する時間は，出力機器のX-Yレコーダを FACOM 270-30 で ON-LINE 制御すると，7分から10分くらいである．これは OFF-LINE にすれば，計算機使用時間は1分くらいとなり，十分実用になると思われる．

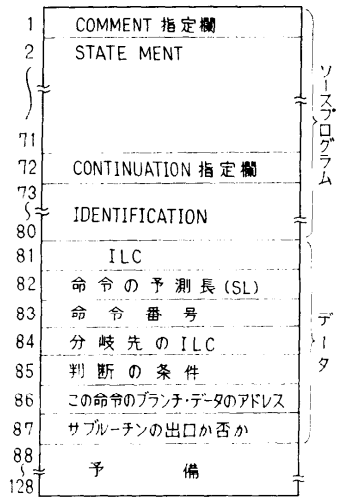


Fig. 14　ソースプログラム・テキスト

終わりにあたり，ご援助下さった東京大学生産技術研究所三部山口研究室の島田淑男氏ならびに桜井正郎氏，さらに有益なご助言を下さった渡辺研究室の各位に深甚なる謝意を表するものである．

参考文献

1) Richard M. Karp: A Note on the Application of Graph Theory to Digital Computer Programming, Information and Control, Vol. 3, No. 2 (1960), pp. 179-190
2) Edward A. Voorhees: Algebraic Formulation of Flow Diagrams, Communications of the Association for Computing Machinary, Vol. 1, No. 6 (1958), pp. 4-8
3) Lee Krider: Flow Analysis Algorithm, Journal of the Association for Computing Machi-

```
EXMPL    BSS    1                                                    EXM00010
         LX.S   *1,-10                                               EXM00020
         LX.I   *2,EX1                                               EXM00030
L21      L.L    EX2                                                  EXM00040
         A.L    EX2+1                                                EXM00050
         D.L    EX2+2                                                EXM00060
         ST.L   EX3                                                  EXM00070
         M.R    EX4                                                  EXM00080
         B.L    L22,PM                                               EXM00090
         SKP.S  P                                                    EXM00100
         B.R    L23                                                  EXM00110
         BB.L   SUBEXM (5)                                           EXM00120
         DN     3                                                    EXM00130
         B.R    L21                                                  EXM00140
         B.R    L22+2                                                EXM00150
         L.R    EX3                                                  EXM00160
         SL.S   8                                                    EXM00170
         B.I    *1,EX4 (*1) -2,L21 -4,L21+2 -6,L21+4                 EXM00180
EX1      DN     -20                                                  EXM00190
EX2      DN     5,-10,15,-20                                         EXM00200
EX3      BSS    2                                                    EXM00210
EX4      DN     -1,13                                                EXM00220
L22      AX.S   *2,2                                                 EXM00230
         B.R    L24                                                  EXM00240
         L.R    EX5                                                  EXM00250
         OR.I   EX5+1                                                EXM00260
         SR.S   4                                                    EXM00270
         B.L    L25,ZM                                               EXM00280
         SKP.S  Z                                                    EXM00281
         B.R    L26                                                  EXM00290
         STX.S  *3,-10                                               EXM00300
L27      STS.L  EX6                                                  EXM00310
         B.I    EXMPL                                                EXM00320
EX5      DN     10,100,50                                            EXM00330
EX6      BSS    1                                                    EXM00340
L24      LS.S   6                                                    EXM00350
         B.R    L27                                                  EXM00360
L26      L.L    *1,EX2+1                                             EXM00370
         S.L    *1,EX2                                               EXM00380
         ST.R   EX3                                                  EXM00390
         FL     EX7                                                  EXM00400
         FA     EX7+2                                                EXM00410
         FST    EX7+4                                                EXM00420
         B.L    L21+9                                                EXM00430
L23      AM.L   EX3,2                                                EXM00440
         B.S    4                                                    EXM00450
         L.L    *2,EX2                                               EXM00460
         S.L    EX2+1                                                EXM00470
         A.L    *2,EX2+2                                             EXM00480
         ST.L   EX2                                                  EXM00490
L25      STS.L  EX6                                                  EXM00500
         B.I    EXMPL                                                EXM00510
EX7      FLD    10.0,5.3,0.0                                         EXM00520
```

Fig. 15　Fig. 16 に示されるフローチャートのソースプログラム

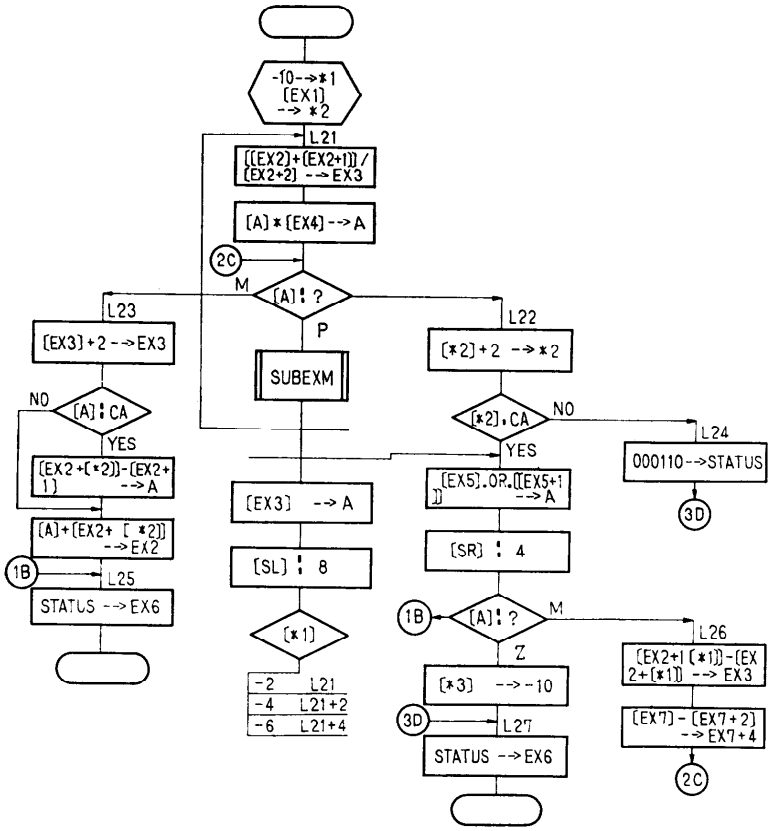


Fig. 16 自動フローチャーティング・プログラムによってかかれたフローチャート

nary, Vol. 11, No. 4 (1964), pp. 429-436
4) Lois M. Haibt: A program to Draw Multilevel Flow Charts, Proceedings of the Western Joint Computer Conference, 1959, pp. 131-137
5) 前川 守: 自動フローチャーティング, 情報処理, Vol. 9, No. 3 (1968), pp. 131-136
6) IBM Application Program, System/360 Flowchart (360 A-SE-22S) User's Manual

付 録 I

この方式の特徴の一つに，データブルをできるだけ利用して，プログラムをすっきりしたものにしているということがある．それらのテーブルのうち，本文でも説明に利用されている重要なテーブルを **Fig. 12～14** に示す．

付 録 II

実際にこの自動フローチャーティング・プログラムを用いて，フローチャートをかかせた例を **Fig. 15～16**[*12] に示す．この例は，一つのフローチャートで種種の例を示すために作られたプログラムで，実際に働いているプログラムではない．

(昭和45年7月7日受付)

*12 Fig. 16 でサブルーチン呼出し命令の下から横に出ている線は，サブルーチンに複数個の出口のある場合のその出口を示している．この場合，上から順に EXIT 1, EXIT 2 で，下方向が EXIT 3 である．なお，このような表現は一般的でないが，プログラムを簡単にするために用いた．