6．逆Compiler LOUPについて

門倉敏夫，山本　浩，福島啓寿，田中　賢
（早稲田大学電子計算室）

1．目　的

LOUP（Logic Of Unknown Programming）は未知のprogramを解読するprogramであり，現在の対象計算機はTOSBAC－3121である．

2．方　式

LOUPは2－pass方式で，LOUP－I，LOUP－IIに分かれている．

LOUP－Iは被test program（1+1 address方式）を忠実に実行し，その実行したprogramを1－address方式にtable（L_i tableと呼ぶ）に格納すると同時にprogramの切目を知り，Variable Connectorの存在を探知し，実行したLocationを次々印刷し，Loopに入れば，印刷を止めて，退出の時にその回数を印刷するprogramでLOUP－IIへの橋渡しを行うと同時に単独でも運転可能な動的なprogramである．

LOUP－IIはLOUP－Iで作成されたL_i表を基にしてこれをstaticに解読しFORTRAN形式に印刷するprogramである．

3．現在の能力

被test programは1,000 stepまでであつて，LOUP－IIの解読可能なstatementは，Arithmetic，IF，GO TO，Assigned GO TO，DO statementである．

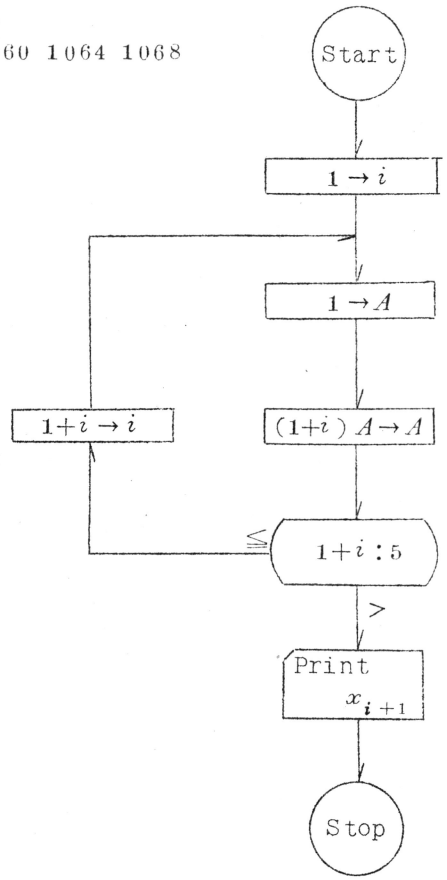
4．例　題

例題　5！

```
                LOUP I
H= 1000
T= 1056
 1000 1004 1008 1012 1016 1020 1024 1028 1032 1036 1040 1044 1060 1064 1068
 1072 1012
 1012
 1012
 loop  3    V.C.= 1044
 1048 1052 1056
 SAYONARA
 WE MEET AGAIN IN LOUP II


                LOUP II

                GS=JJ
                DW=GS
                HR=DW
4               DR=DW+JJ
                US=(GS) GL
                EH=DR
                IF(EH-ZQ) 2, 2, 1
2               GS=US
                DW=DR
                HR=DW
                GO TO  4
1               FORMAT(NUM 12)
                PRINT US
                END
```



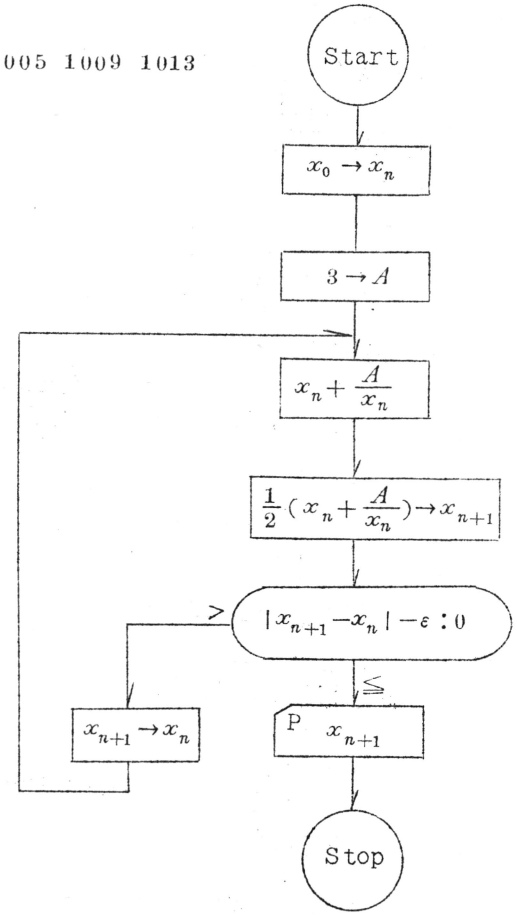
LOUP I

例題　$\sqrt{3}$

```
H= 1000
T= 1029
 1000 1004 1008 1012 1016 1020 1024 1030 1034 1038 1042 1046 1005 1009 1013
 1017 1008
 1008
 1008
 1008
 1008
 loop  5   V.C.= 1009
 1015 1025 1029
 SAYONARA
 WE MEET AGAIN IN LOUP II
```

LOUP II

```
                QP=UJ
                DG=QP
5               OE=(NC)/QP+QP
                BE=(OE)/JJ
                WC=(BE-OP)@CP-GA
                IF(WC-OO)  1,  1,  2
2               QP=BE
                DG=QP
                GO TO   5
1               FORMAT(NUM 12)
                PRINT BE
                END
```



5. LOUP—Iについて

以下にこのgeneral flow chartを示す．紙面の都合上，detailのflow chartは省略する．

被test programを所定の位置に読み込んでから，先ず部分印刷の指示I（1又は0）を与え，開始番地（H）と終了番地（T）の指示を行い，LOUP Iのみで使用する場合，便利なようにT_i tableの印刷の指示J（1又は0）を行つて，各種Tableを零として，種々のinitial setの後に①に入る．

LOUP 1は例題にも示した如く，1+1 address方式で通過Locationを明示するのがその目的の1つであるから，tracerの如く忠実に被test programを追うために各演算を実行して行くから各pseudo registerを持つている．

この対称計算機はregisterがaddressを持つているから先ずこの処置を⑦で行う．ここでは各registerについてtableを作成し，2ケの命令語で置き代えを行い結果が記録される様にして⑥に戻る．⑥→②の間では実行途中でこのtableのLocationが出現した場合の処置を行う．

②から④までの流れはLOUPに橋渡しを行つた場合に便利な様に1+1 address方式を1—address方式としてL_i表とする種々の準備をする．

すなわち

（i）通過したLocationにはcheckを入れる．

（ii）併しながら現在のLocationにcheckがなくても今まで通過して来たLocationにcheckがあれば，現在のLocationはvariable connectorかLoopのexitであるからこの処置をする．

（iii）今まで通過して来たLocationにcheckがなく，現在のLocationにcheckがあればLoopのreturnと考え，このcheckを入れ，再び同じ状態でこのcheckを通過すればLoop Counterを上げる．

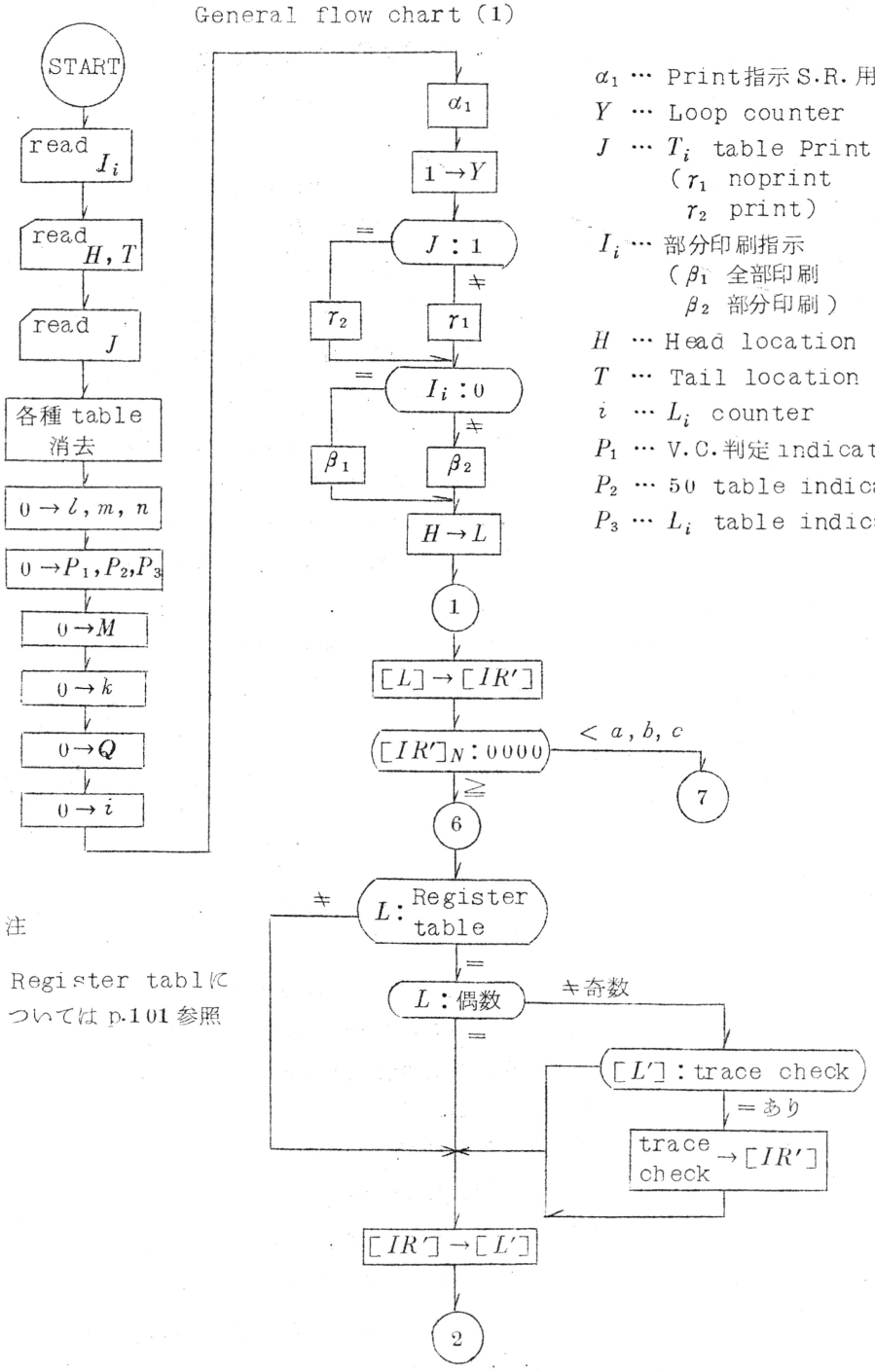
（iv）一度通過したLocationはvariable connector（testを含む）以外は印刷しない．

の動作を行う．

次に④から⑤までは先ず命令語を復元して，indexを使用した場合にはtrueのLocationに直してから，pseudo registerを相手に実行を行い，この命令語をL_i tableの1員として加え，次の実行番地を求めて①に戻る．

最終番地に到達すれば，LOUP IIへの準備のためvariable connector table完了S.R.（一番最初のvariable connectorは無視されているため）を通つてexitする．

General flow chart (1)

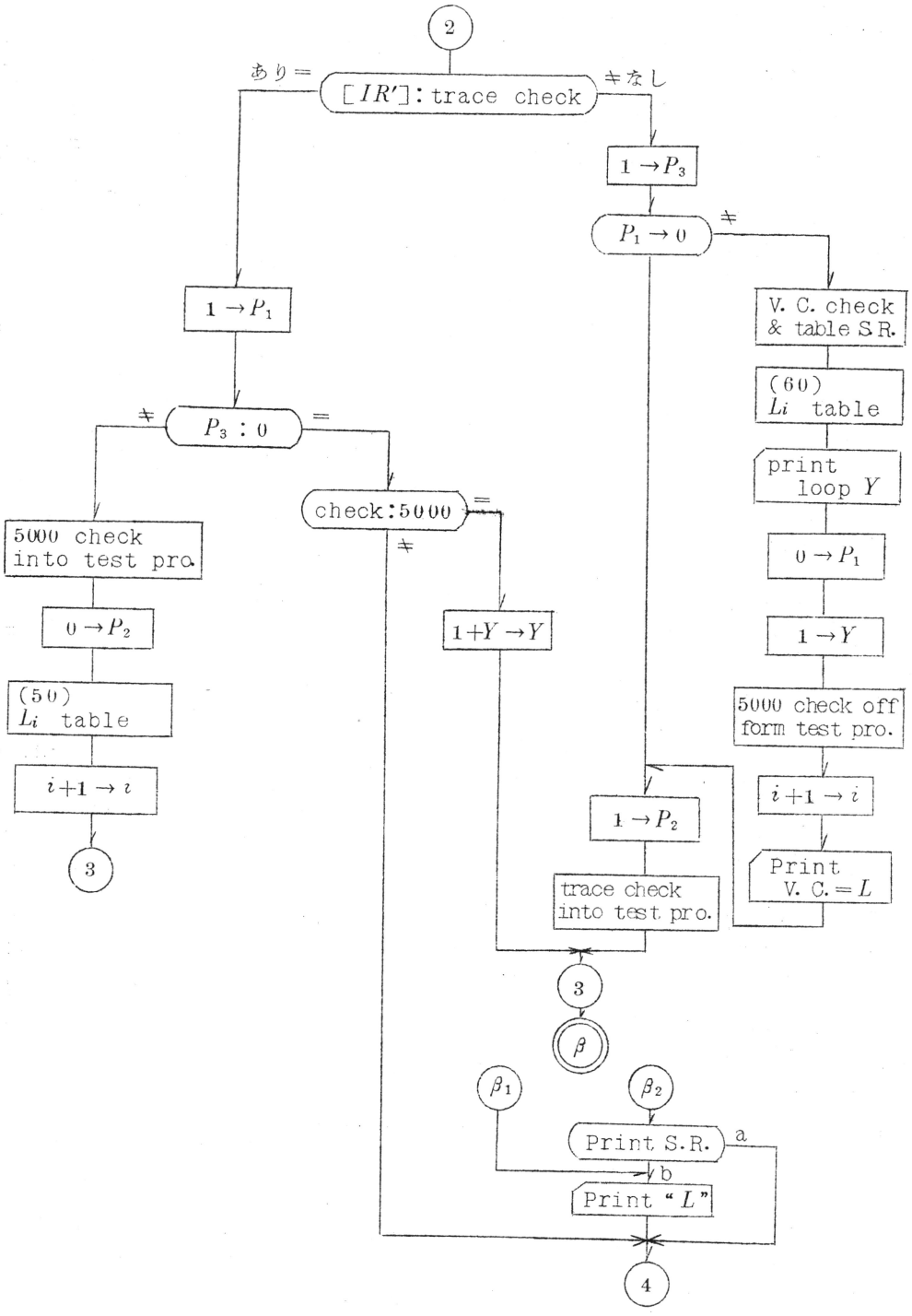


α_1 … Print指示 S.R. 用
Y … Loop counter
J … T_i table Print指示
(γ_1 noprint
γ_2 print)
I_i … 部分印刷指示
(β_1 全部印刷
β_2 部分印刷)
H … Head location
T … Tail location
i … L_i counter
P_1 … V.C.判定 indicator
P_2 … 50 table indicator
P_3 … L_i table indicator

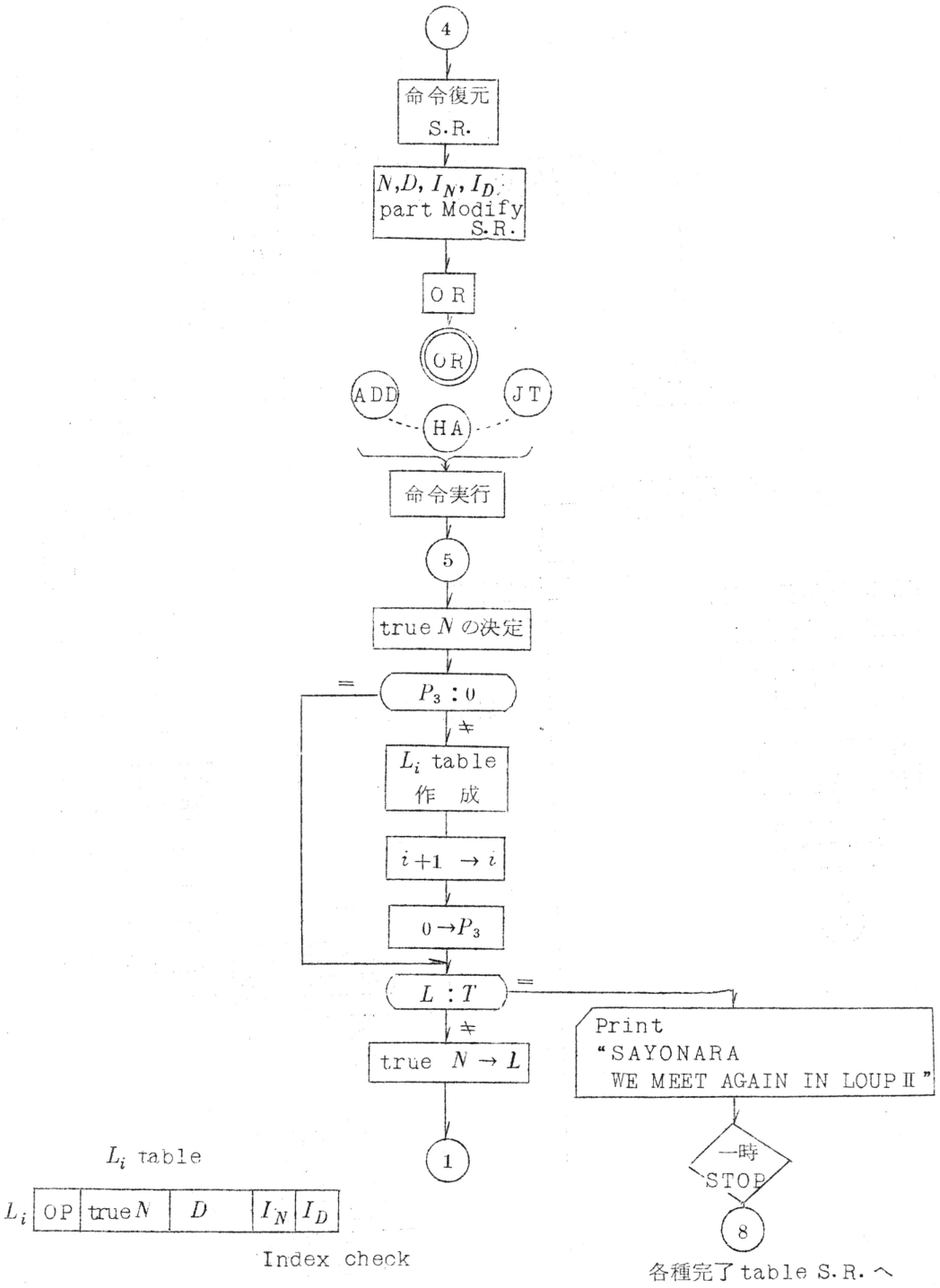
注

Register tablについては p.101 参照

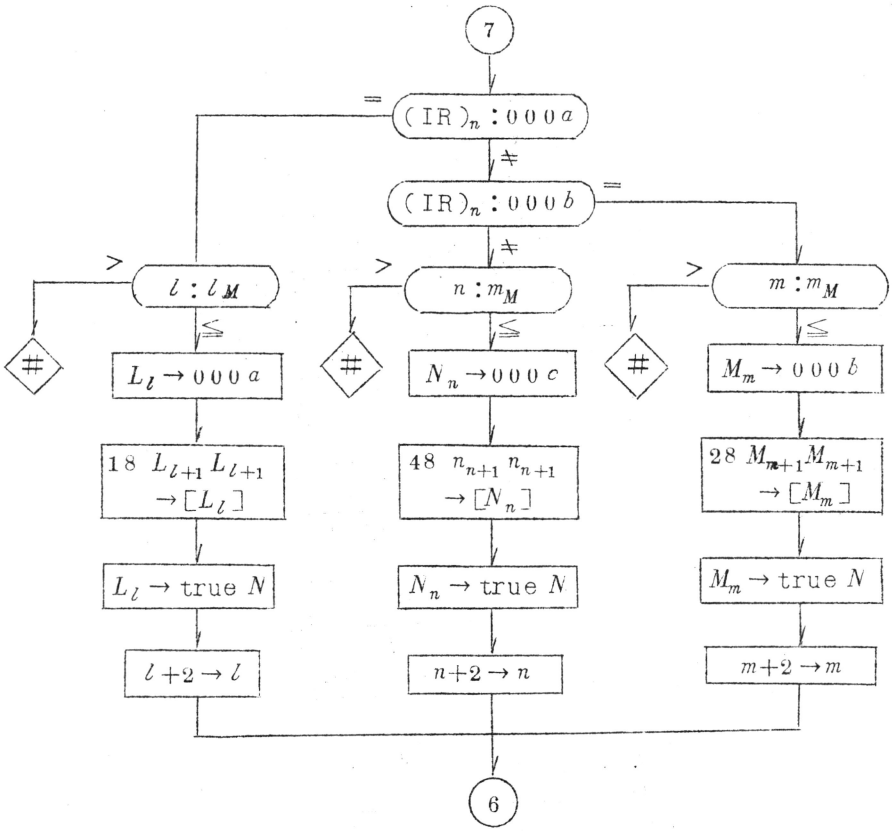
General flow chart (2)



General flow chart (3)



N Partが rA rB rC の場合

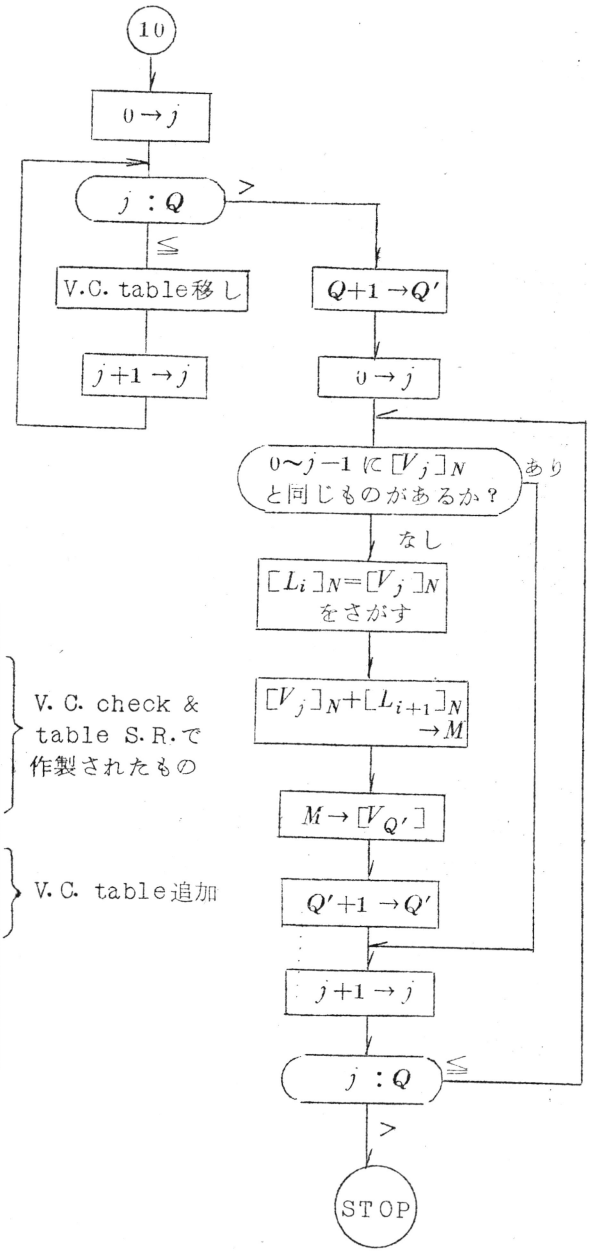


····· table overflow stop

Register Table (0340～0399)

rA:(0340～0359)	rB:(0360～0379)	rC:(0380～0399)
L_0 ; [18 L_1, L_1]	M_0 ; [28, M_1, M_1]	N_0 ; [48, N_1, N_1]
L_1 ; [rAの内容]	M_1 ; [rBの内容]	N_1 ; [rCの内容]
L_2 ; [18 L_3, L_3]	⋮	⋮
L_3 ; [rAの内容]		
⋮		
L_ℓ ; [18 $L_{\ell+1}$, $L_{\ell+1}$]	M_m ; [28, M_{m+1}, M_{m+1}]	N_n ; [48, N_{n+1}, N_{n+1}]
$L_{\ell+1}$; [rAの内容]	M_{m+1} ; [rBの内容]	N_{n+1} ; [rCの内容]

V. C. table 完了 S. R.



V. C. table

	X	N	D	
V_0		2000	2002	V. C. check & table S. R. で作製されたもの
⋮				
V_j		3000	3001	
⋮				
V_Q		2000	2003	
V_{Q+1}		2000	2001	V. C. table追加
⋮				
$V_{Q'}=Q_{max}$		3000	3002	

6. LOUP−Ⅱについて

LOUP−ⅡはLOUP−Iで作成された1−address形の L_i tableをstaticに解釈してFORTRAN型に印刷するprogramである.

併しながら未知programを解釈して行くのであるから，ConstantとVariable の区別がつかず，すべてvariableとして印刷する.勿論Fixed pointとFloating pointの判定もつかない欠点がある.

このgeneral flow chartを以下に示す.

先ずS.R. 8によつてLOUP Iで作成したV. C.（variable connector）tableを調査し，行先が3ケ以上であればV.C.と判定し，行先が2ケの場合，命令がtest 命令でない場合はV.C.と判断する．それ以外の場合は皆消し去る.

initial setの後，先ずV.C.のcheckがある場合には，V. C.であるcheckをtable用に変更して，S.R. 1に入る．ここでは与えられた10進法4桁のlocationに対し，乱数を使用して2桁の英字を発生し，tableに記入しておく.

S.R. 3では“(”のcounterを判定し(Arithmetic statement等で必要）存在すればこれを印刷する.

次にS.R. 1で求めた代名詞を印刷し，S.R. 9で A_k table（演算命令によつてArithmetic statementの原表を作成して行くtable)を全部打ち出してAssigned GO TO の印刷に入る.

先ずS.R. 4でstatement no.を印刷した後にGO TO $n(S_1, S_2, \cdots)$ と印刷して④に戻る.

次の判定はプログラムの切目のcheckであり，もしここで切れていればDO Loop の端に来た可能性があるのでDOの判定S.R. 7に入る.

ここではindexによるmodifyのみをDOと判定し通常のaddress modification，及び不定回数のiterationはIF statementで表現する.

以下の L_i tableを調査する必要があるから別にcounterを起して，この出発のLocationに戻るまで，以下のprogram中にindex判定の命令があれば一応拾い上げてtable（H_P table）を作成し，調査終了后この H_P table 内の命令語の行先をしらべ，調査範囲内に戻ればDOのnestと考えられ， 次回の同様な調査で再び拾い上げられるから，行先が外部に出たもののみを印刷する．すなわち行先のLocationのstatement no.を印刷し，（continueは不必要）③に戻る.

併しながらDO statementと判定できない場合には，ここでprogram の切目とのみ判定し，今までの A_k tableを印刷して③に戻る.

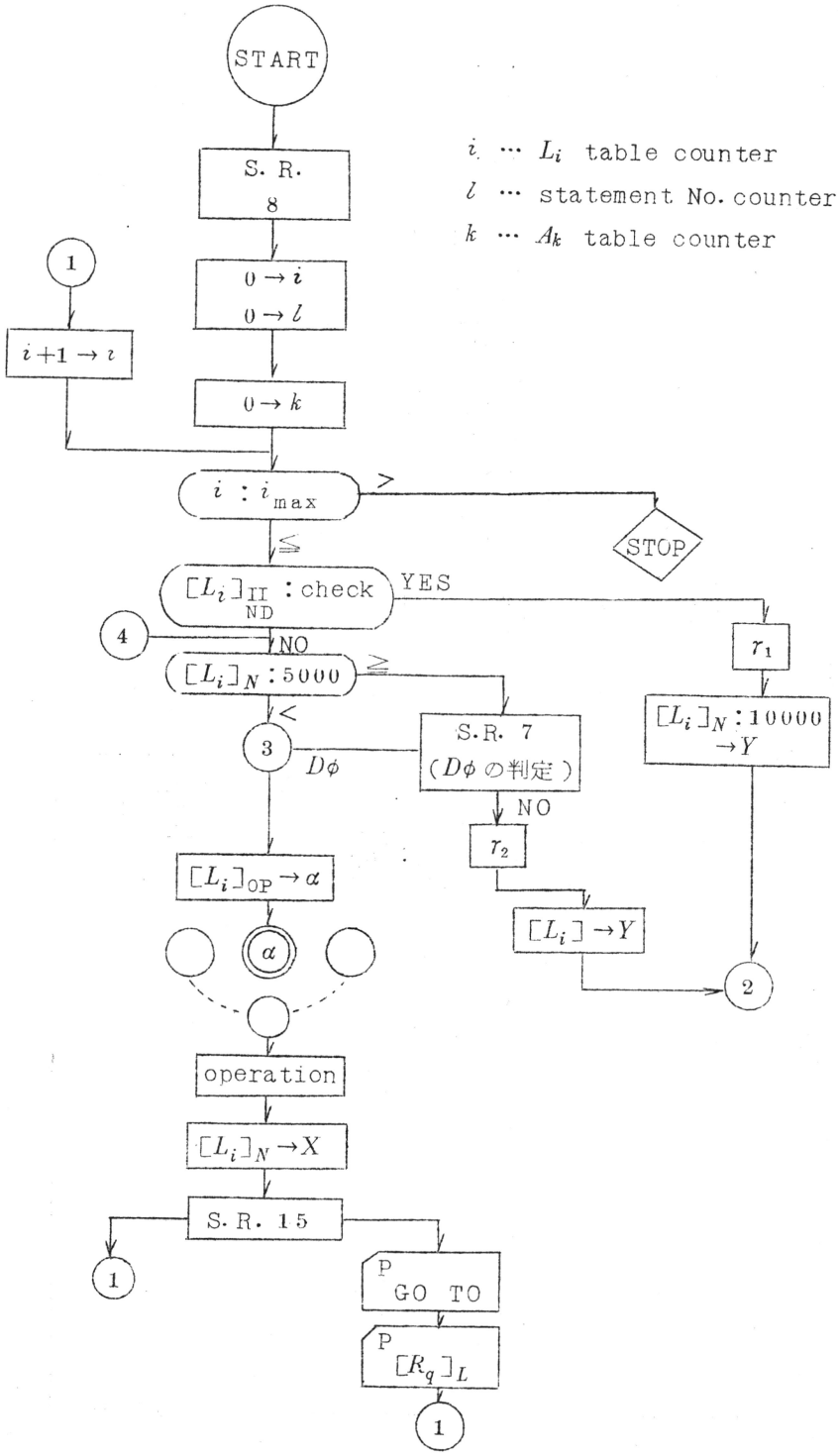
③からは各命令に従つて解読作業に入り，終了後，次のLocationを求め，S.R. 15 に入り，このLocationがstatement no. tableに登録してあればGO TO statement

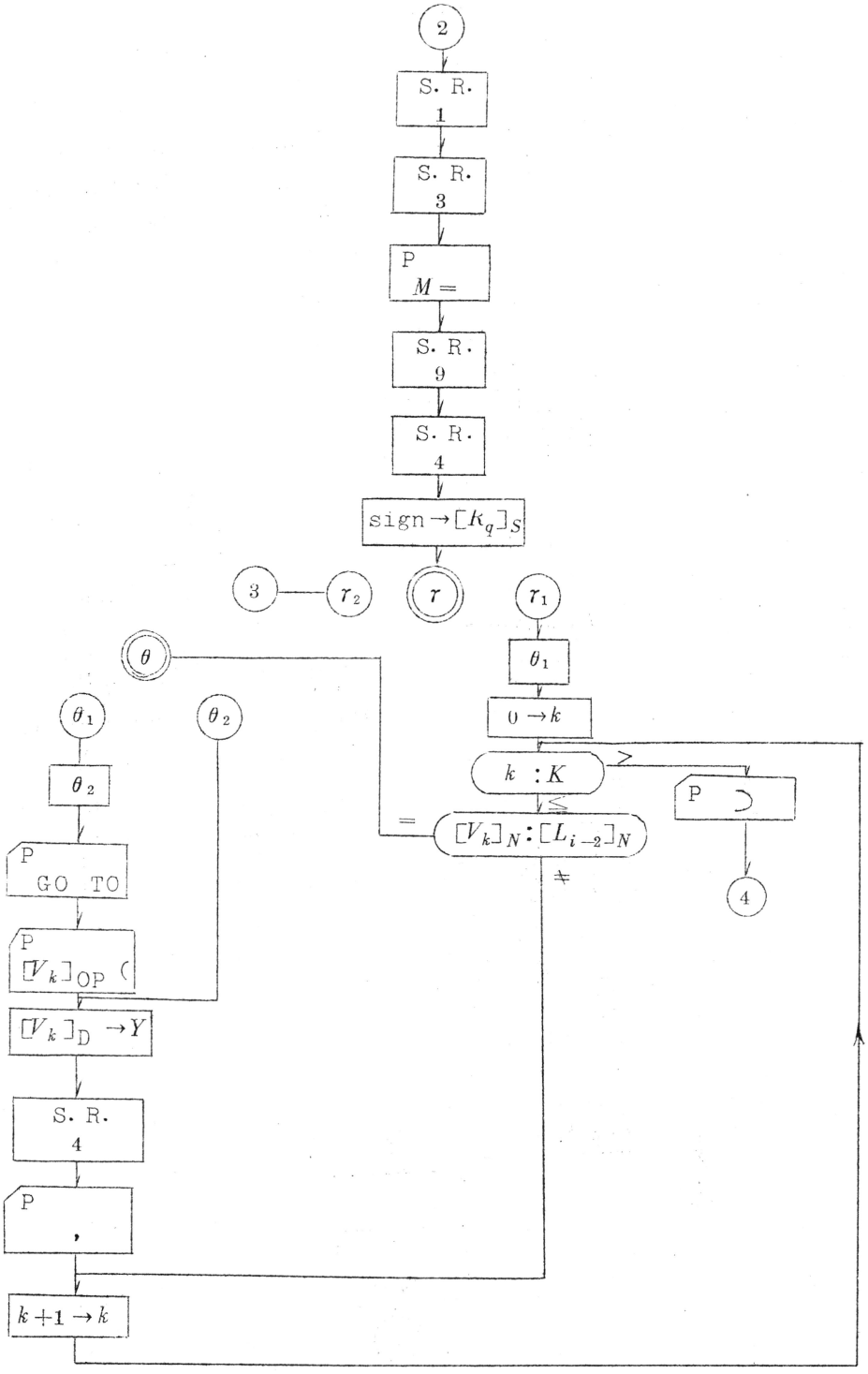
と解釈して実行し，さもなければそのまま①に戻る．

以下命令別の実行を示す．又紙数の都合で省略したS.R.の機能を説明する．

S.R.	2	A_k table作成用（S.R. 9を含む）
〃	4	statement no.発生用
〃	5	A_k table clear用（S.R. 1，2を含む）
〃	6	代名詞発生用
〃	10	IF statement印刷用（S.R. 3，6，9を含む）
〃	11	S.R. 10の補助である
〃	12	（　）の処理用（S.R. 2を含む）
〃	13	index判定用
〃	14	Assigment statement判定用

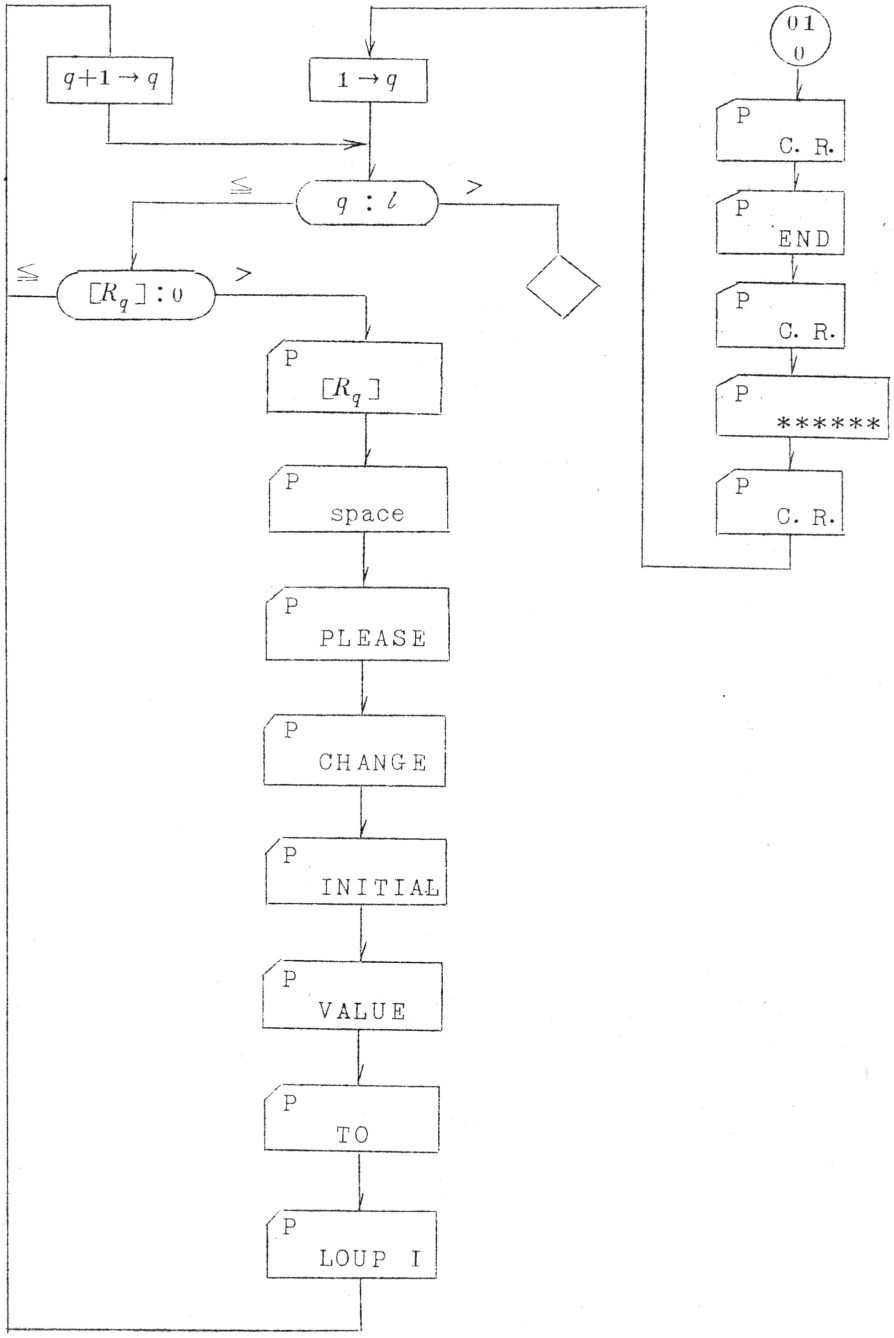
LOUP−Ⅱ の General flow chart

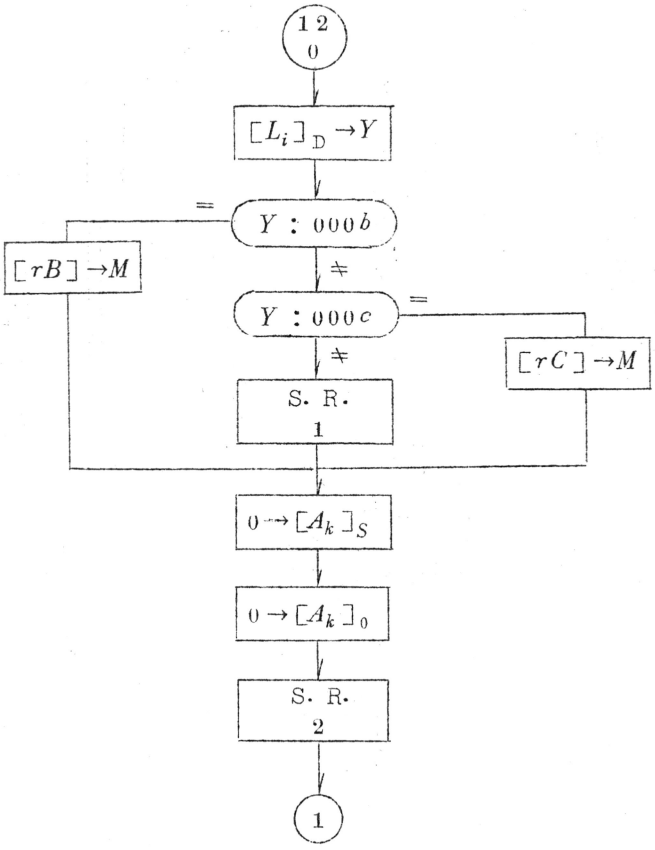


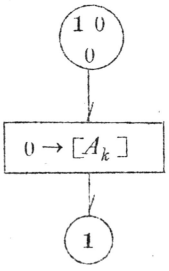

2
S. R. 1
S. R. 3
P M =
S. R. 9
S. R. 4
sign → [Kq]S
3
r2
r
r1
θ
θ1
θ2
θ1
0 → k
θ2
k : K
>
≤
=
[Vk]N : [Li−2]N
≠
P)
4
P GO TO
P [Vk]OP (
[Vk]D → Y
S. R. 4
P ,
k+1 → k

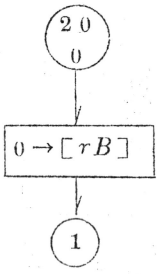
命令別による実行

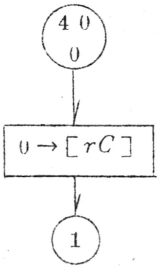
未使用の statement no.の印刷

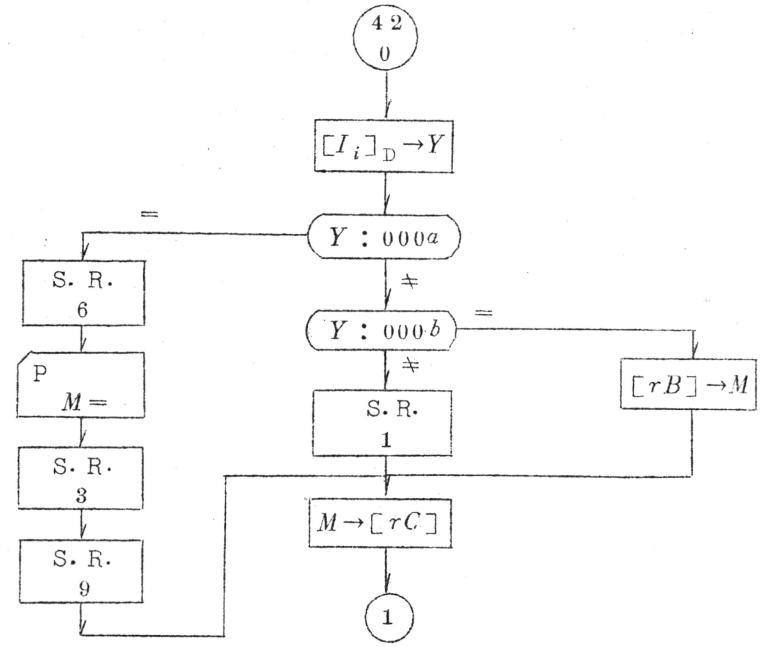


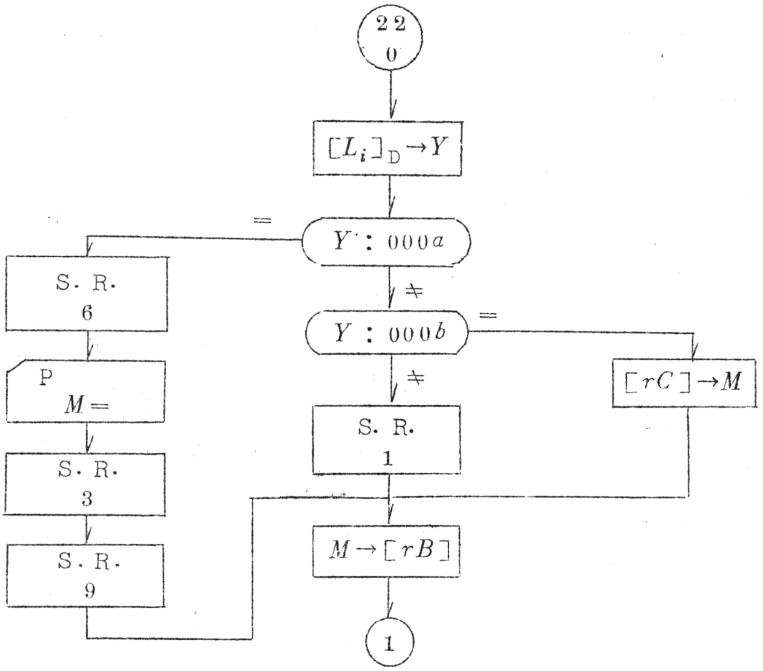

1 2
0
$[L_i]_D \rightarrow Y$
=
Y : 000b
$[rB] \rightarrow M$
≠
Y : 000c
=
≠
$[rC] \rightarrow M$
S. R.
1
$0 \rightarrow [A_k]_S$
$0 \rightarrow [A_k]_0$
S. R.
2
1

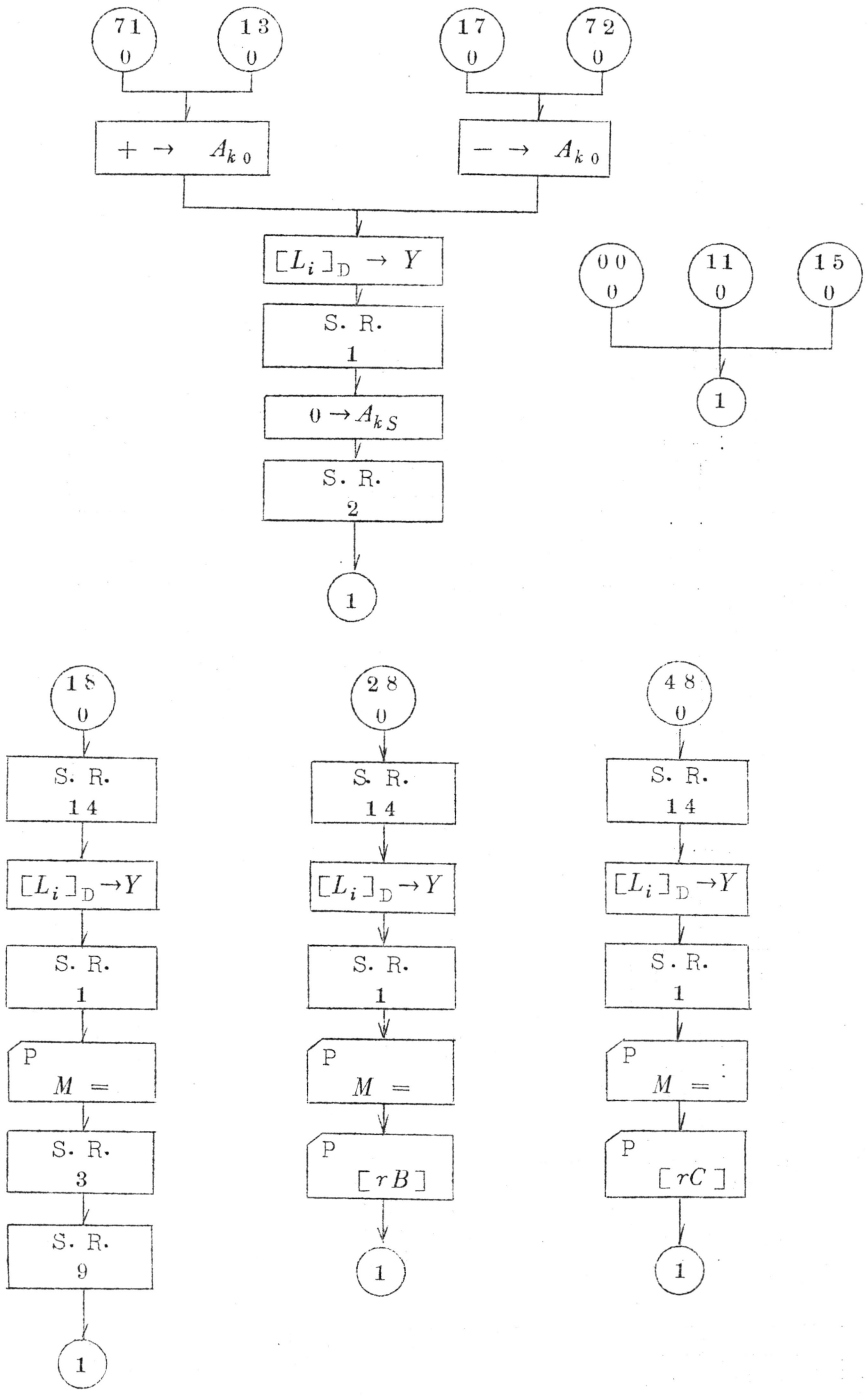

1 0
0
$0 \rightarrow [A_k]$
1

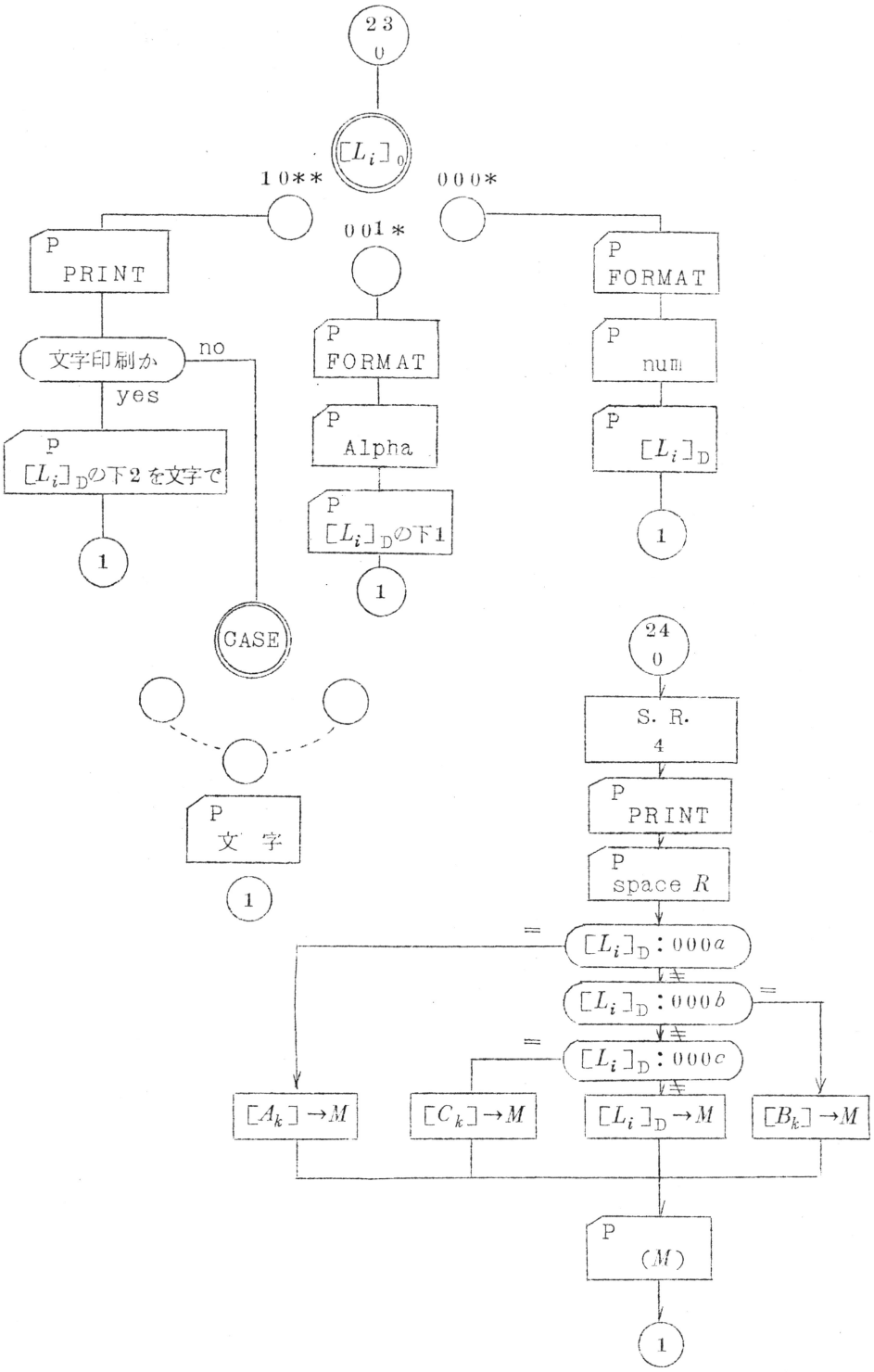

2 0
0
$0 \rightarrow [rB]$
1

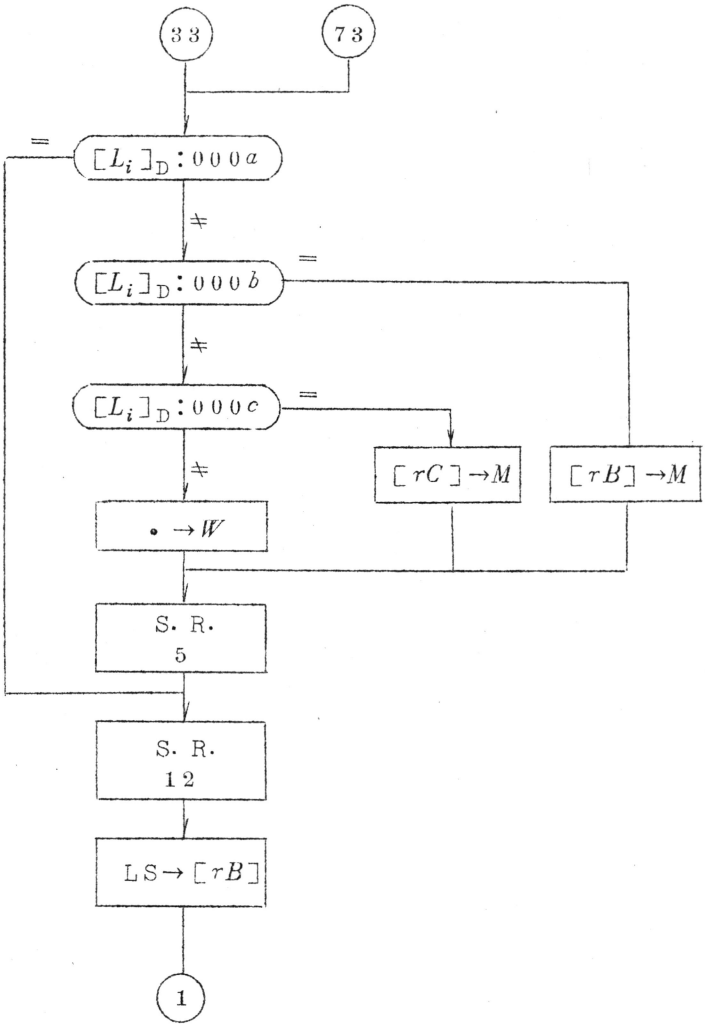

4 0
0
$0 \rightarrow [rC]$
1

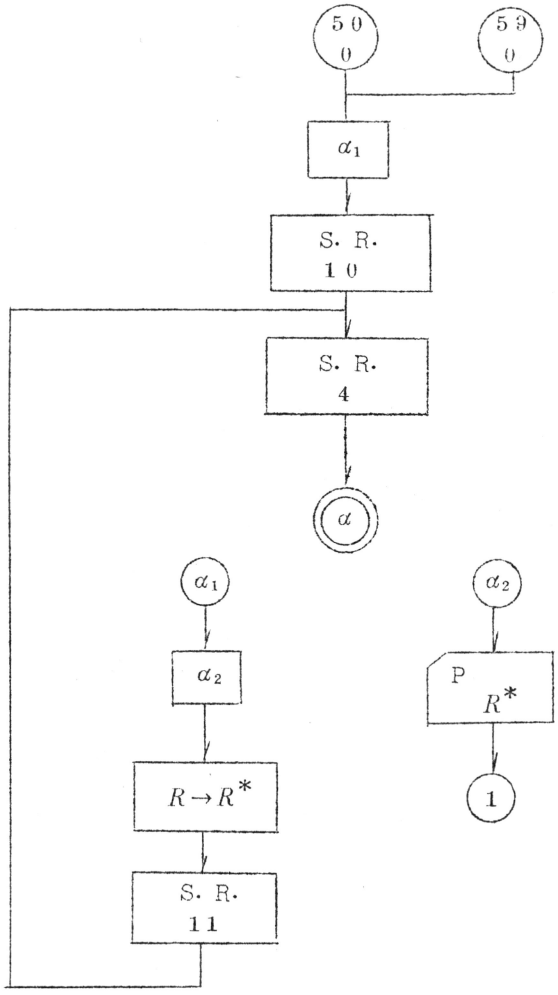

42
0
[Ii]D→Y
=
Y : 000a
≠
S. R.
6
Y : 000b
=
P
M=
≠
[rB]→M
S. R.
3
S. R.
1
S. R.
9
M→[rC]
1

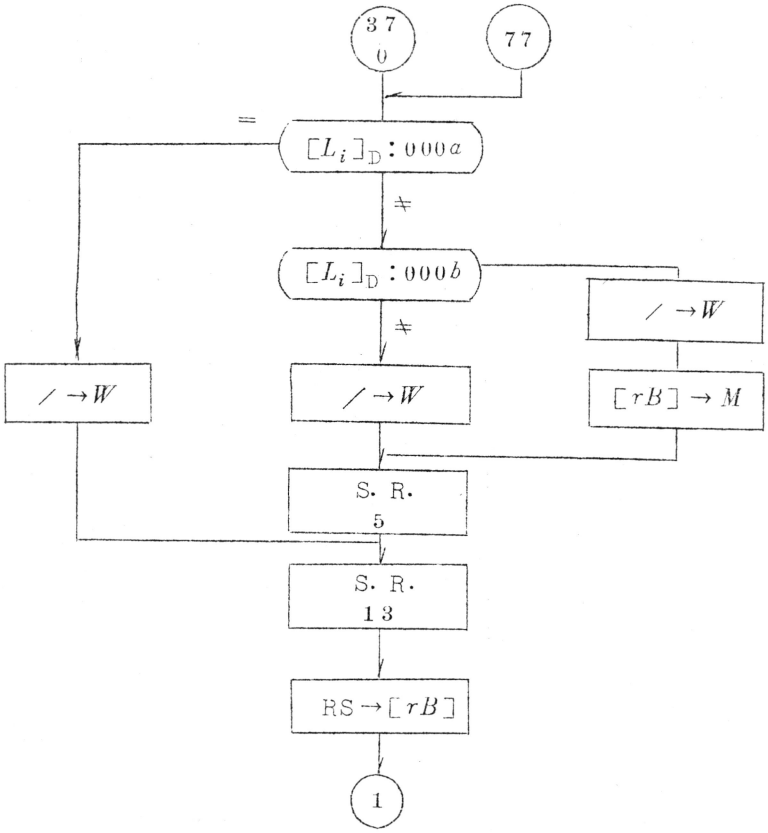

22
0
[Li]D→Y
=
Y : 000a
≠
S. R.
6
Y : 000b
=
P
M=
≠
[rC]→M
S. R.
3
S. R.
1
S. R.
9
M→[rB]
1

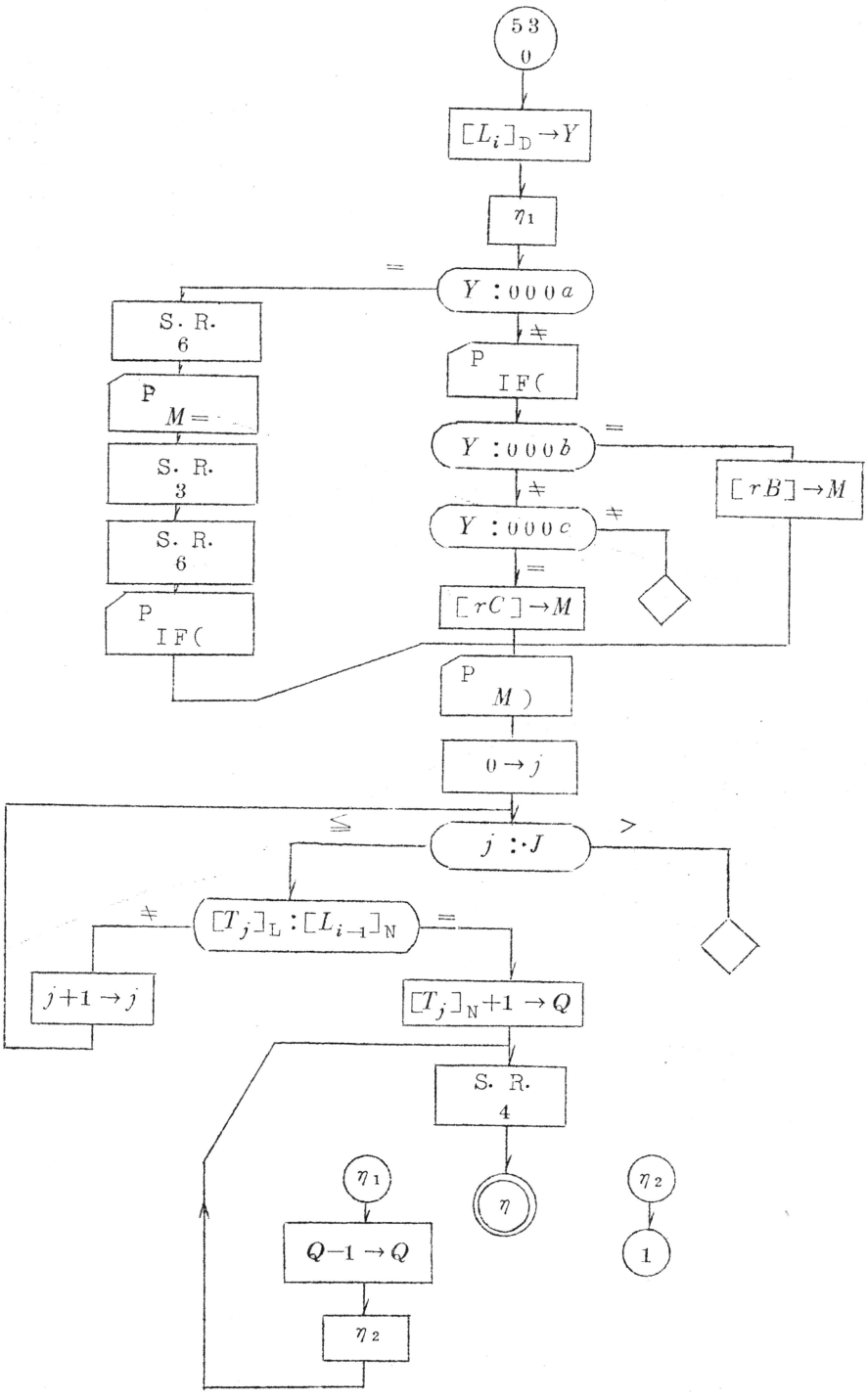

71 0
13 0
17 0
72 0
+ → A_{k0}
− → A_{k0}
$[L_i]_D \rightarrow Y$
S. R. 1
$0 \rightarrow A_{kS}$
S. R. 2
1
00 0
11 0
15 0
1
18 0
S. R. 14
$[L_i]_D \rightarrow Y$
S. R. 1
P $M =$
S. R. 3
S. R. 9
1
28 0
S. R. 14
$[L_i]_D \rightarrow Y$
S. R. 1
P $M =$
P $[rB]$
1
48 0
S. R. 14
$[L_i]_D \rightarrow Y$
S. R. 1
P $M =$
P $[rC]$
1

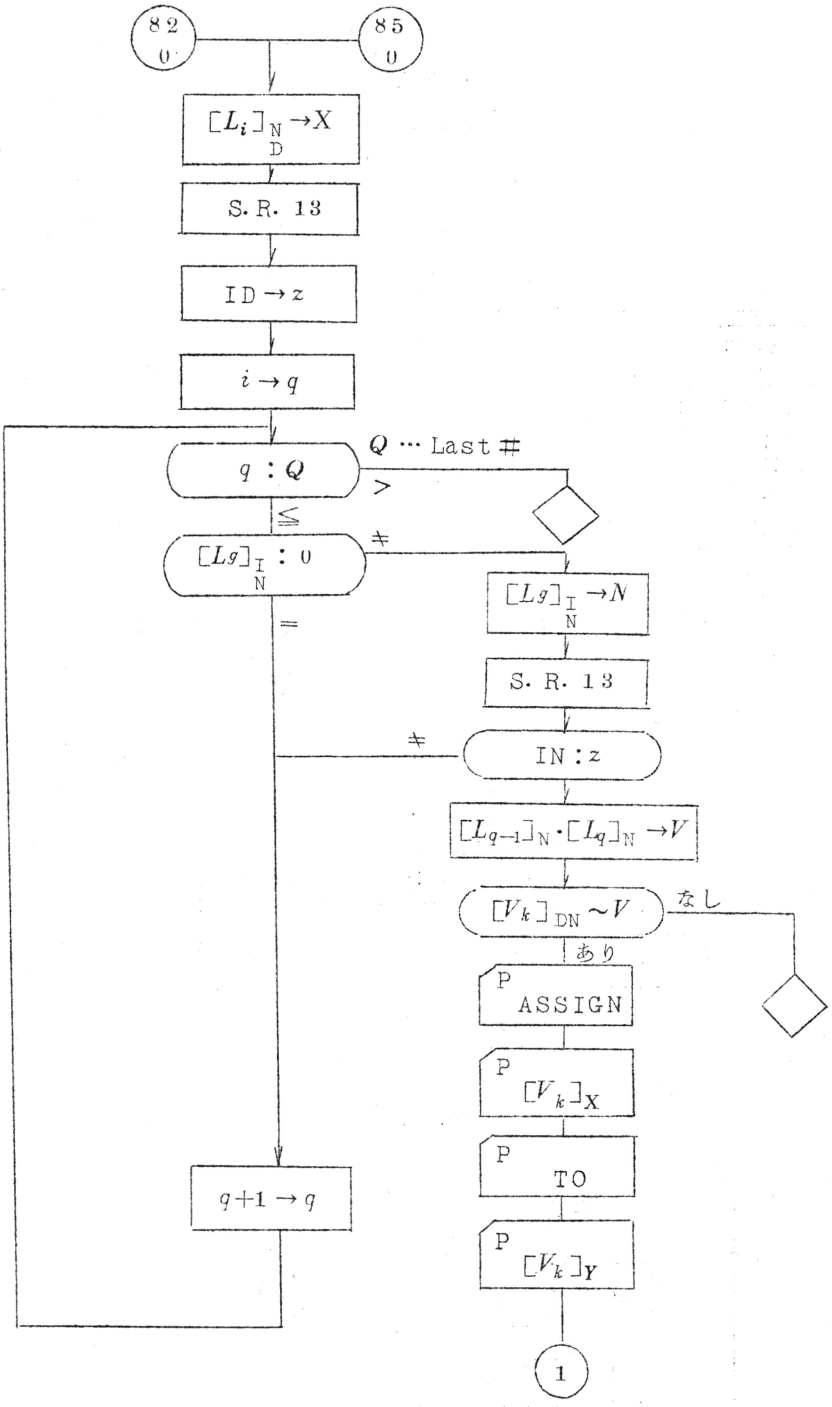

23
0
$[L_i]_0$
10**
001*
000*
P
PRINT
文字印刷か
no
yes
P
$[L_i]_D$の下2を文字で
1
P
FORMAT
P
Alpha
P
$[L_i]_D$の下1
1
P
FORMAT
P
num
P
$[L_i]_D$
1
CASE
P
文 字
1
24
0
S. R.
4
P
PRINT
P
space R
=
$[L_i]_D$: 000a
≠
$[L_i]_D$: 000b
=
≠
$[L_i]_D$: 000c
=
≠
$[A_k] \rightarrow M$
$[C_k] \rightarrow M$
$[L_i]_D \rightarrow M$
$[B_k] \rightarrow M$
P
(M)
1

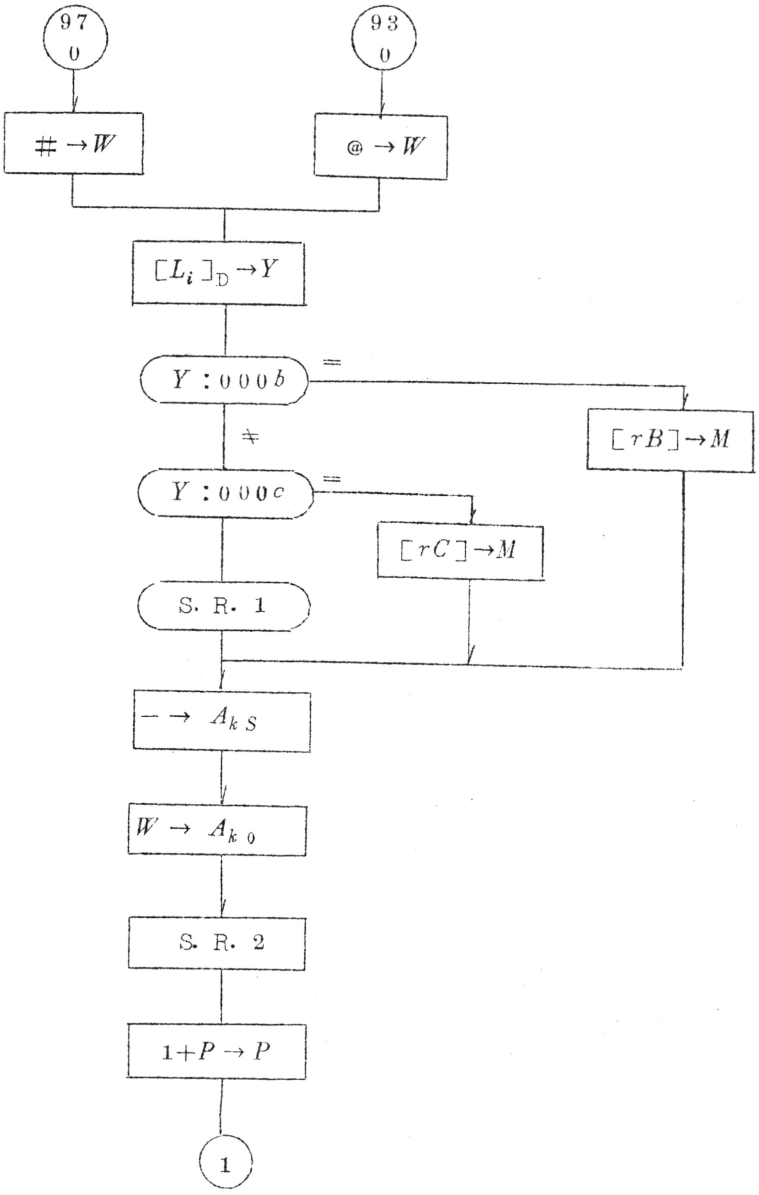

33
73
=
[Li]D : 000a
≠
[Li]D : 000b
=
≠
[Li]D : 000c
=
≠
[rC] → M
[rB] → M
• → W
S. R.
5
S. R.
12
LS → [rB]
1


5 0
0
5 9
0
α_1
S. R.
1 0
S. R.
4
α
α_1
α_2
$R \rightarrow R^*$
S. R.
1 1
α_2
P
R^*
1


37
0
77
=
$[L_i]_D : 000a$
≠
$[L_i]_D : 000b$
≠
／→W
／→W
／→W
[rB] → M
S. R.
5
S. R.
13
RS→[rB]
1


53
0
[Li]D→Y
η1
=
Y：000a
≠
P
IF(
Y：000b
=
[rB]→M
≠
Y：000c
≠
=
[rC]→M
P
M)
0→j
≦
j：J
>
≠
[Tj]L：[Li−1]N
=
j+1→j
[Tj]N+1→Q
S. R.
4
η
η1
Q−1→Q
η2
η2
1
S. R.
6
P
M=
S. R.
3
S. R.
6
P
IF(


82
0
85
0
$[L_i]_{\substack{N\\D}} \rightarrow X$
S. R. 13
ID → z
i → q
q : Q
Q … Last #
>
≦
≠
$[Lg]_{\substack{I\\N}}$: 0
$[Lg]_{\substack{I\\N}} \rightarrow N$
=
S. R. 13
≠
IN : z
$[L_{q-1}]_N \cdot [L_q]_N \rightarrow V$
$[V_k]_{DN} \sim V$
なし
あり
P ASSIGN
$P\ [V_k]_X$
P TO
$P\ [V_k]_Y$
q+1 → q
1


97
0
93
0
→ W
@ → W
[L_i]_D → Y
Y : 0 0 0 b
=
≠
[rB] → M
Y : 0 0 0 c
=
[rC] → M
S. R. 1
— → A_k S
W → A_k 0
S. R. 2
1+P → P
1

本 PDF ファイルは 1965 年発行の「第 6 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html