

Detailed Explanation of Processor

This document is designed for persons requiring a thorough knowledge of the 1401 Fortran processor. Each phase of the processor is described in a different section. Information enclosed in parentheses and capitalized refer to symbolic labels present in the listing.

General

The reader should be acquainted with certain key work areas addressed throughout the processor.

1. PARAMA - The hundreds position of the machine size specified on the control card when located in storage.

a. PARAMA/2 - The size of the machine

b. PARAMA/4 - The modules

c. PARAMA/6 - The mantissa and later in the processor, the mantissa plus two.

2. FAILSW - A word mark is set at this location when the processor detects an error which would make object time unrewarding.

NXTOP - The next available location at object time from the top (leftmost) part of storage. It is located at 086.

NXBTM - The next available location at object time from the bottom (right-most) part of storage. It is located at 083.

Snapshot Routine (00)

This routine exists in storage through the entire compilation and object time. It produces snapshots of storage whenever control is transferred to it. The snapshot displays each century of storage on a separate line with appropriate trimmings to assist in checking programs.

If sense switch F is on, the dump is not performed and only the message "Phase name" EXECUTED is printed. If sense switch G is up the routine halts after execution. The print area (locations 200-332) is neither printed nor saved.

This is a closed subroutine and control is transferred after execution, to the next sequential instruction after the branch to the ~~snapshot~~ routines. Snapshot uses index registers 1 and 3 for execution. These registers are saved at the beginning of execution and restored at the end.

If the index registers have zoning over the tens position, the display of the registers will be incorrect due to the logic which moved the values to the print area (HLDXT,HLD32,HLD31).

System Monitor and Parameter Card (01)

Up to 19 positions have been allowed for the control card (PRMCD).
Currently, only ¹⁵14 positions are used.

Upon completion of each phase of the compiler, control is transferred to the system monitor (MONTER). The monitor clears the previous phase (ACLEAR) and then reads the next phase from cards or tape (MONTOR). The location MONTOR is NOPed by the Loader Phase, if system input is from tape.

Prior to transferring control to the monitor, each phase initializes certain operands of the monitor. This initialization is accomplished by the FENDX macro.

TCLEAR - The highest address to be cleared by the monitor.

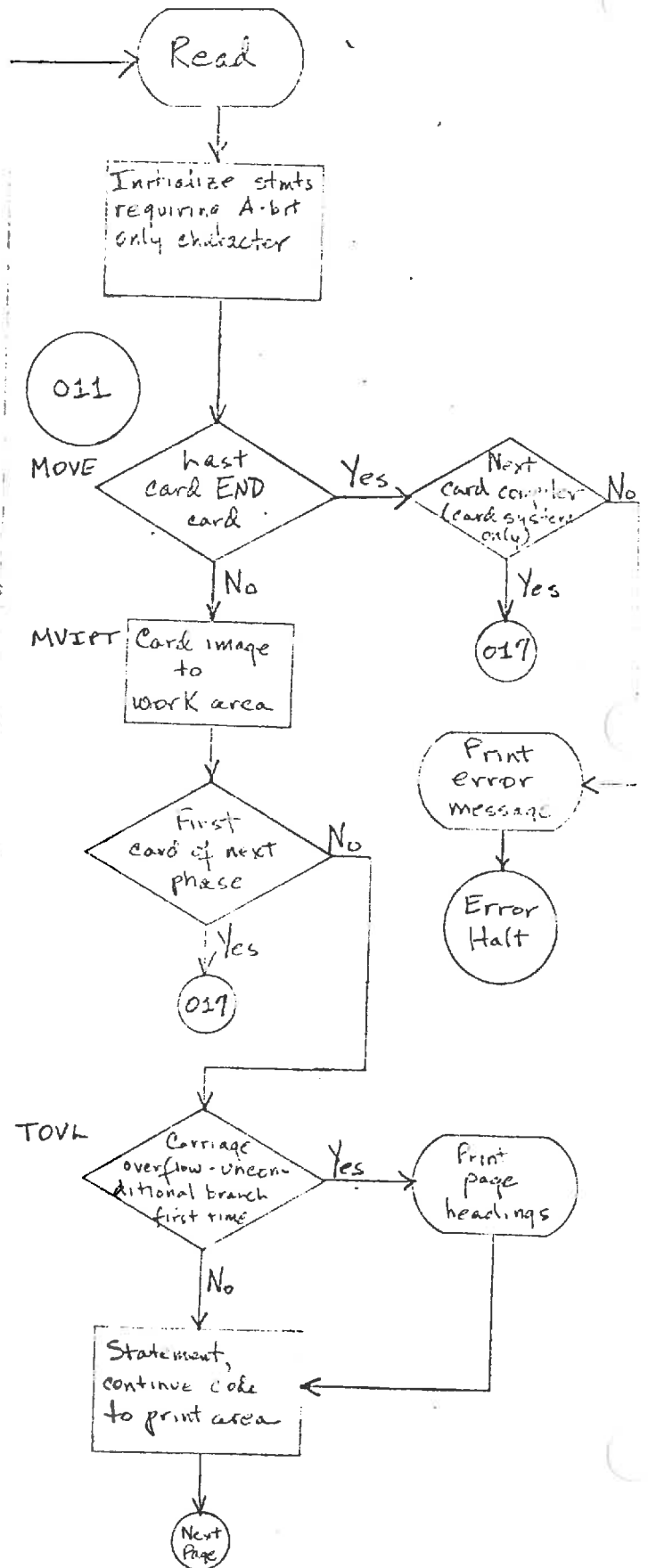
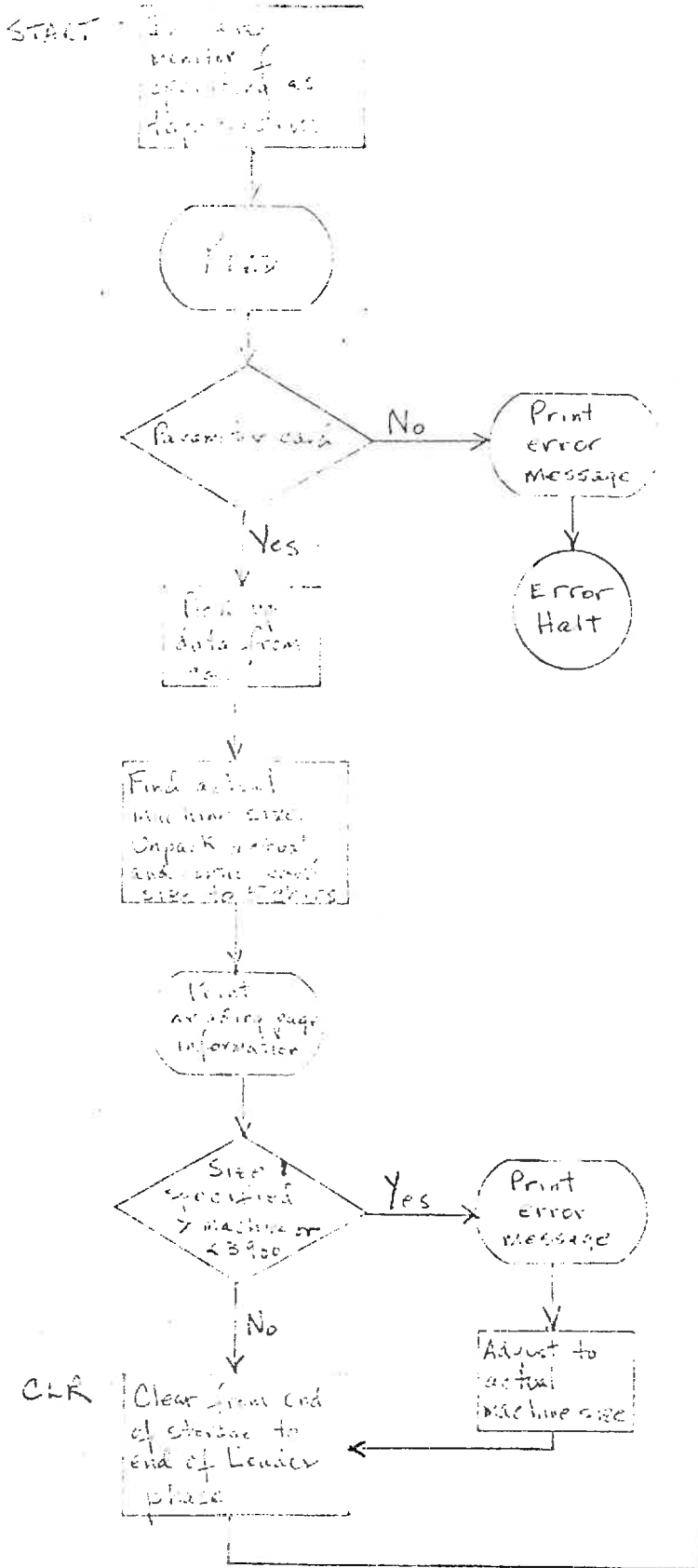
BCLEAR - The lowest address to be cleared by the monitor.

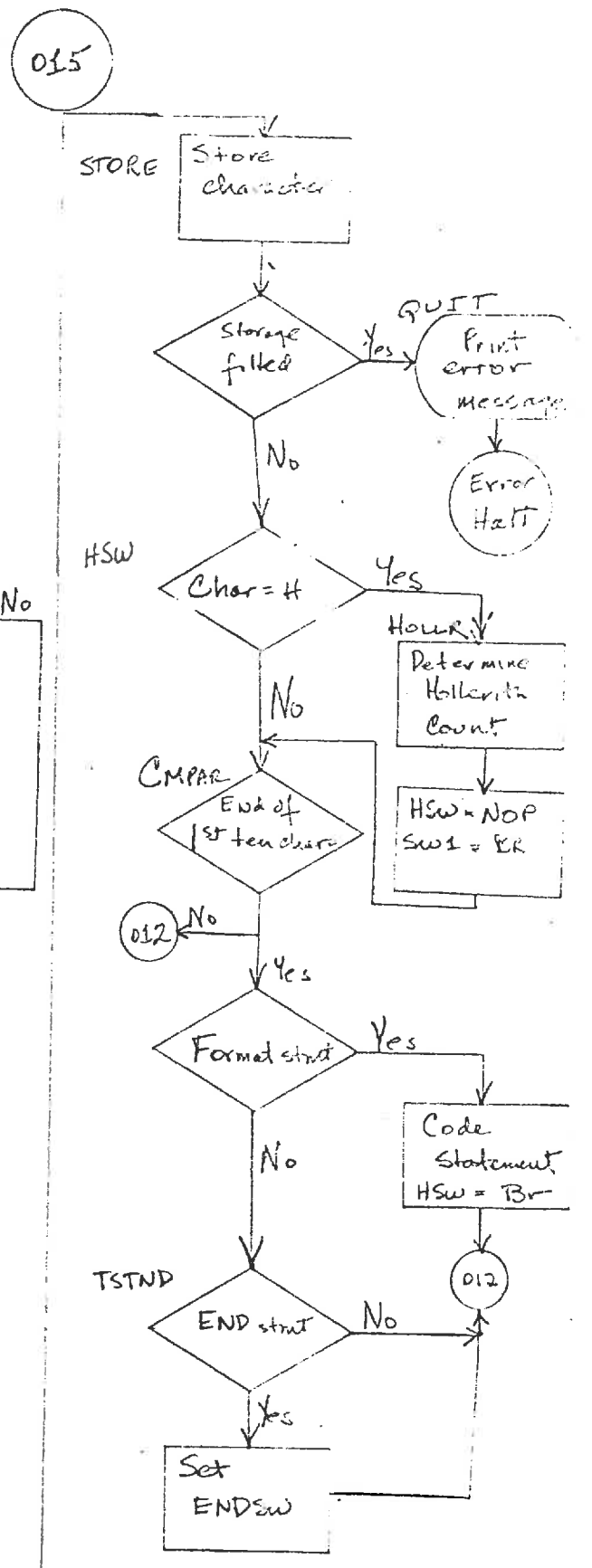
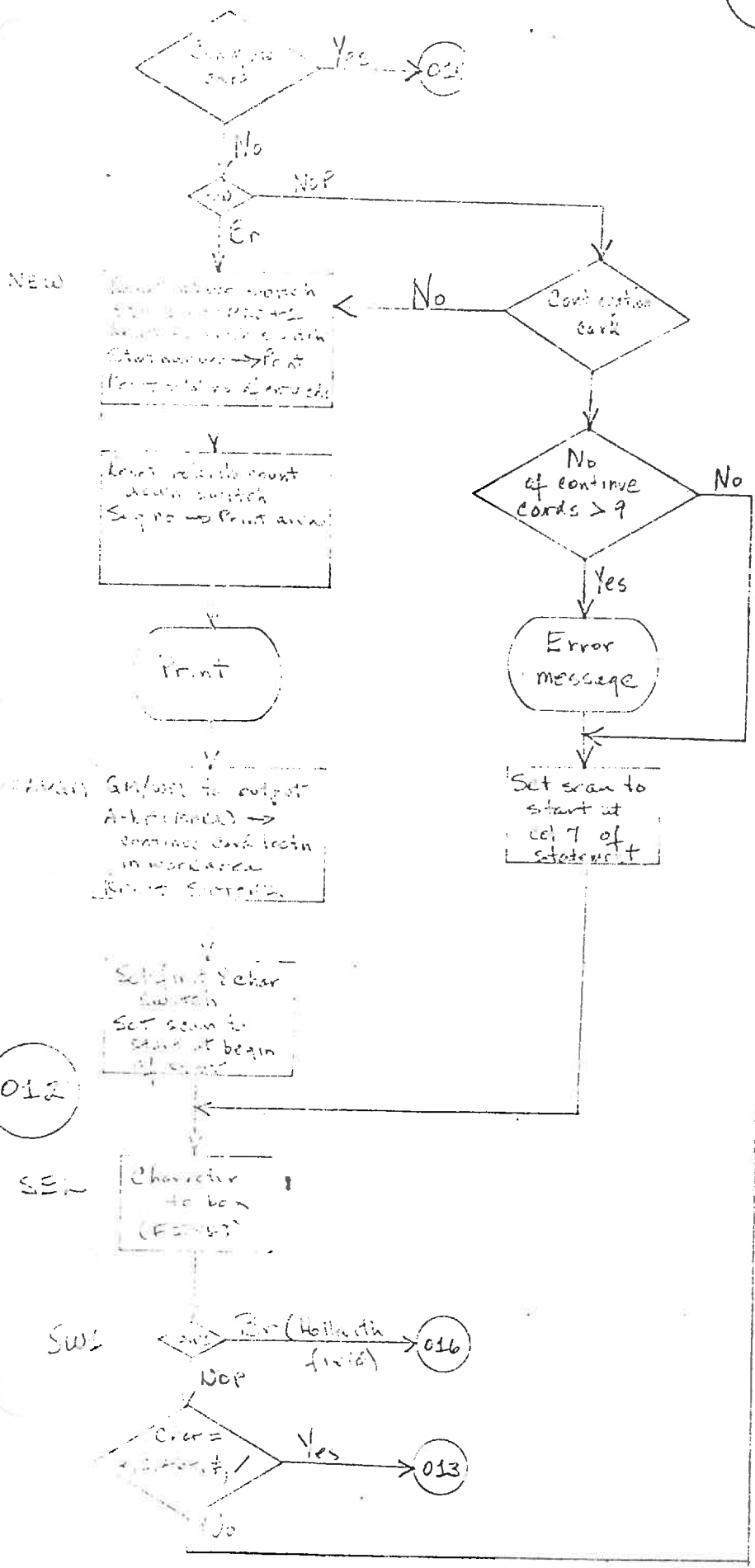
INITAP/6 - The address into which the next phase is to be read where operating as a tape system.

INITXT/3 - The address to which the monitor branches after reading the next phase. This is handled by the XFR card in the card system.

If any of these operands are the same as the previous phase, they are not reinitialized.

LOADER PHASE





015

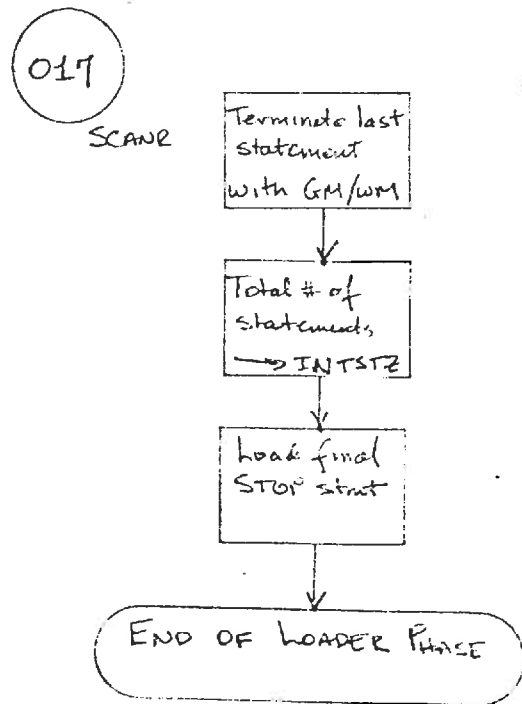
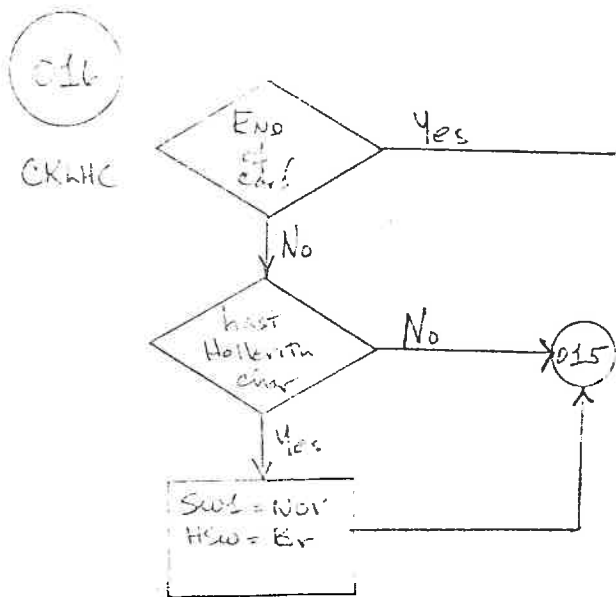
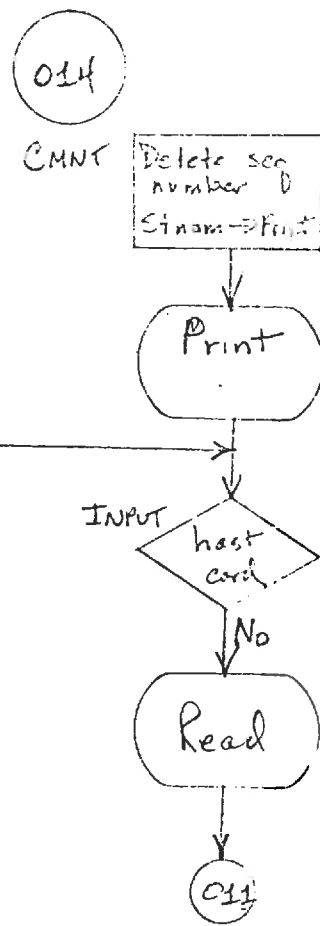
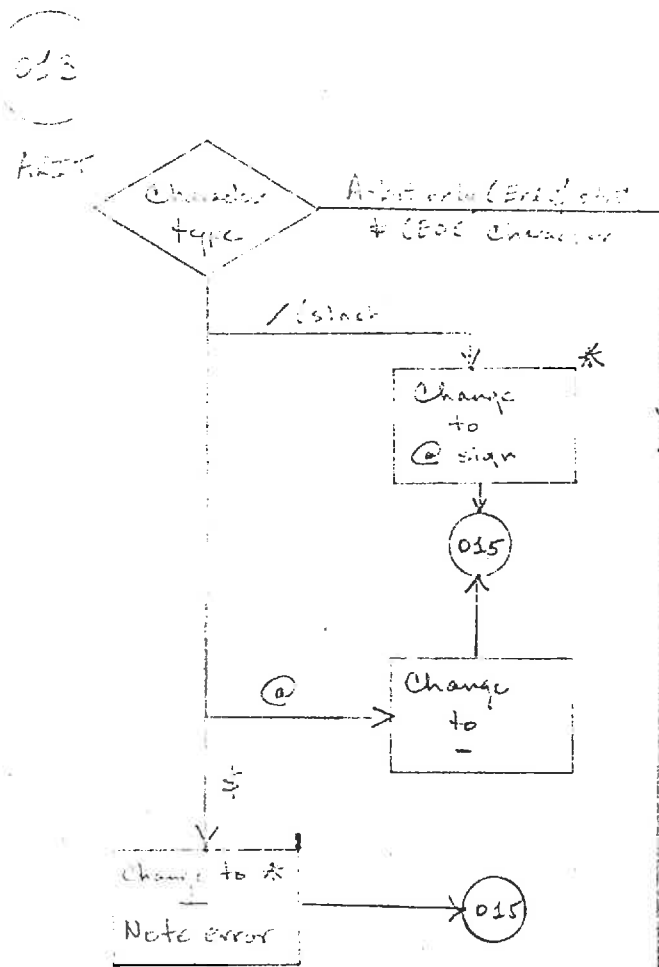
012

012

012

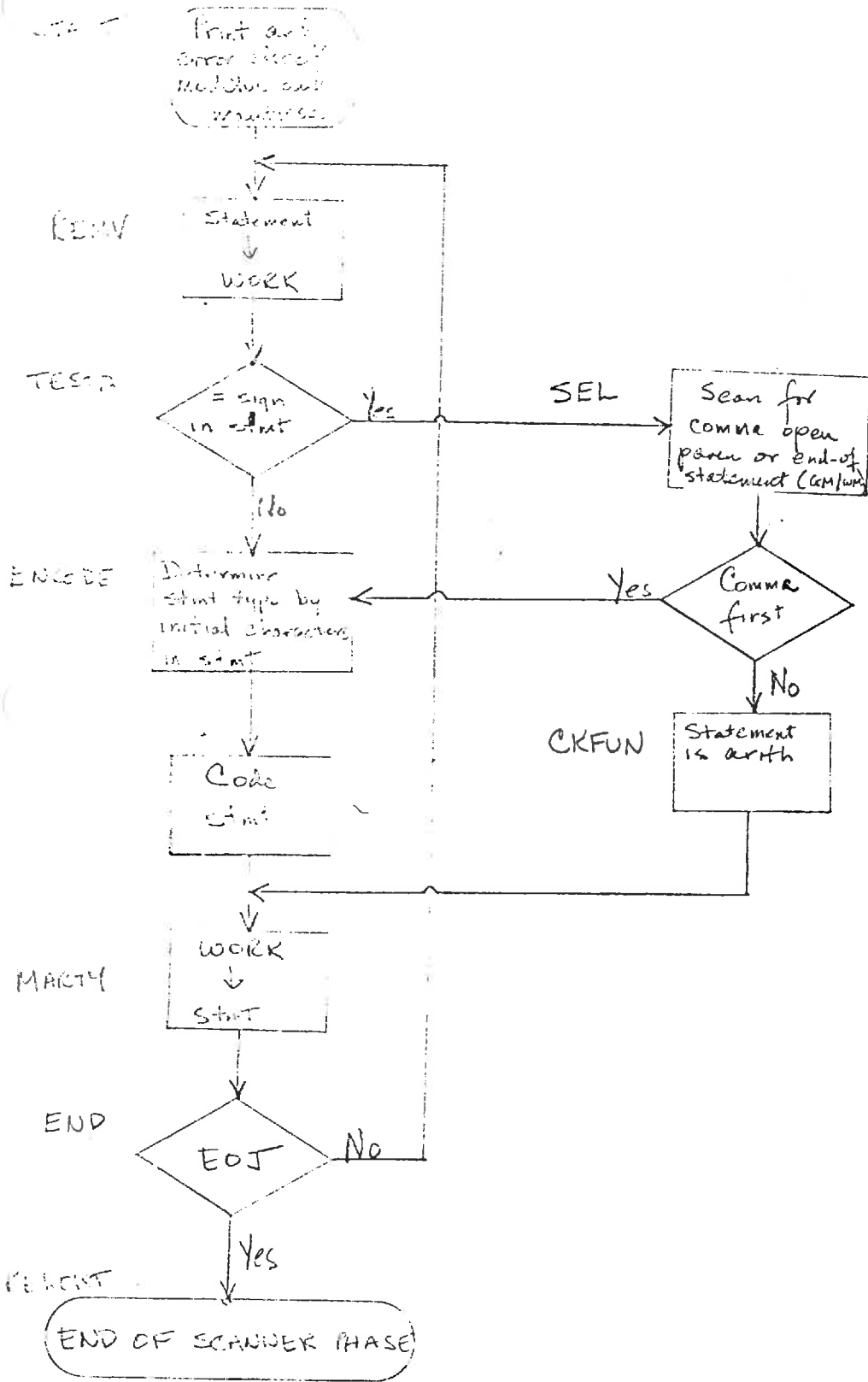
016

013

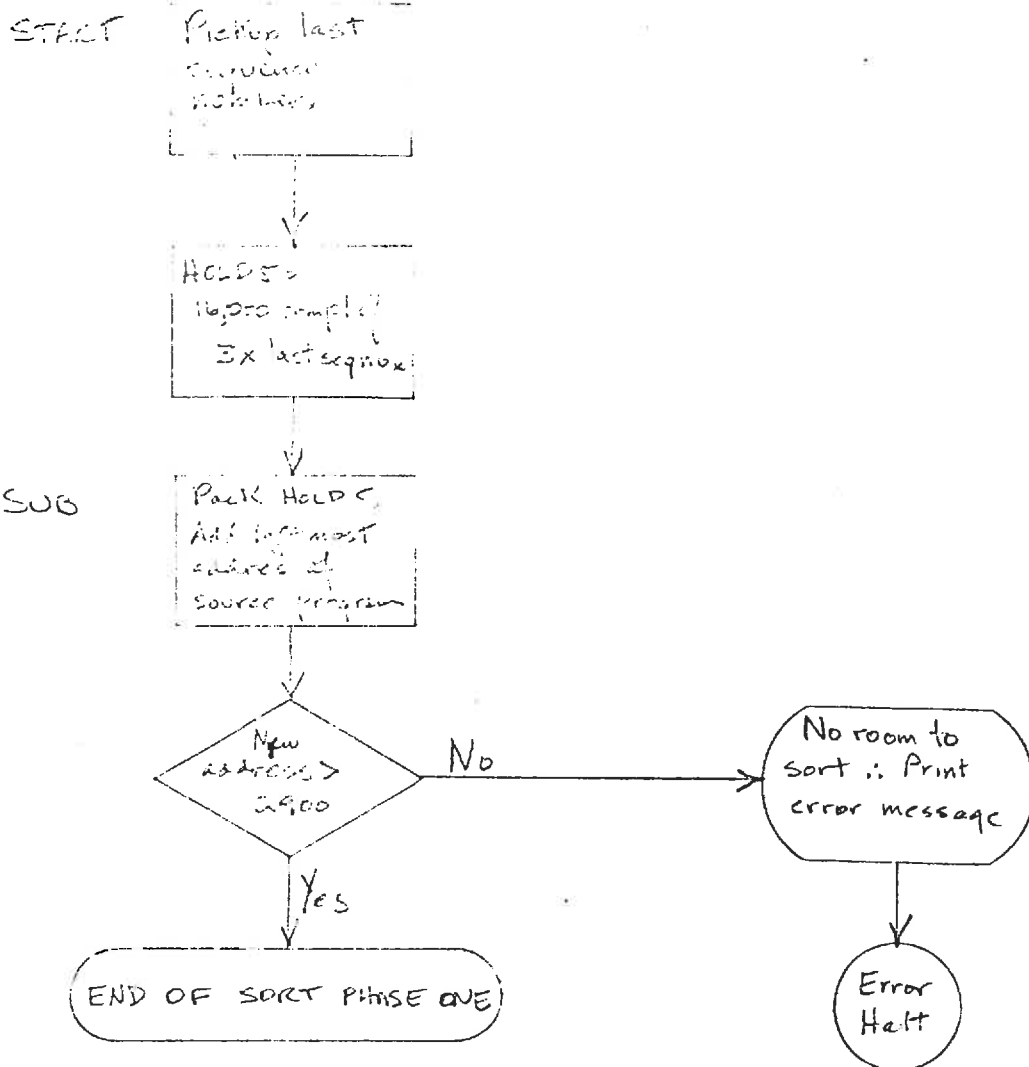


* Note: The input character / is changed to @ (4-2) because the compiler must distinguish between addresses containing / and the operator /.

SCANNER PHASE



-8-
SORT PHASE ONE



SORT PHASE TWO

NUSTM

Start → Hold-5
(to clear for
chain address)

HOLD → Output

Determine table
box for start
types. Move box
(chain address) to output

ENTU

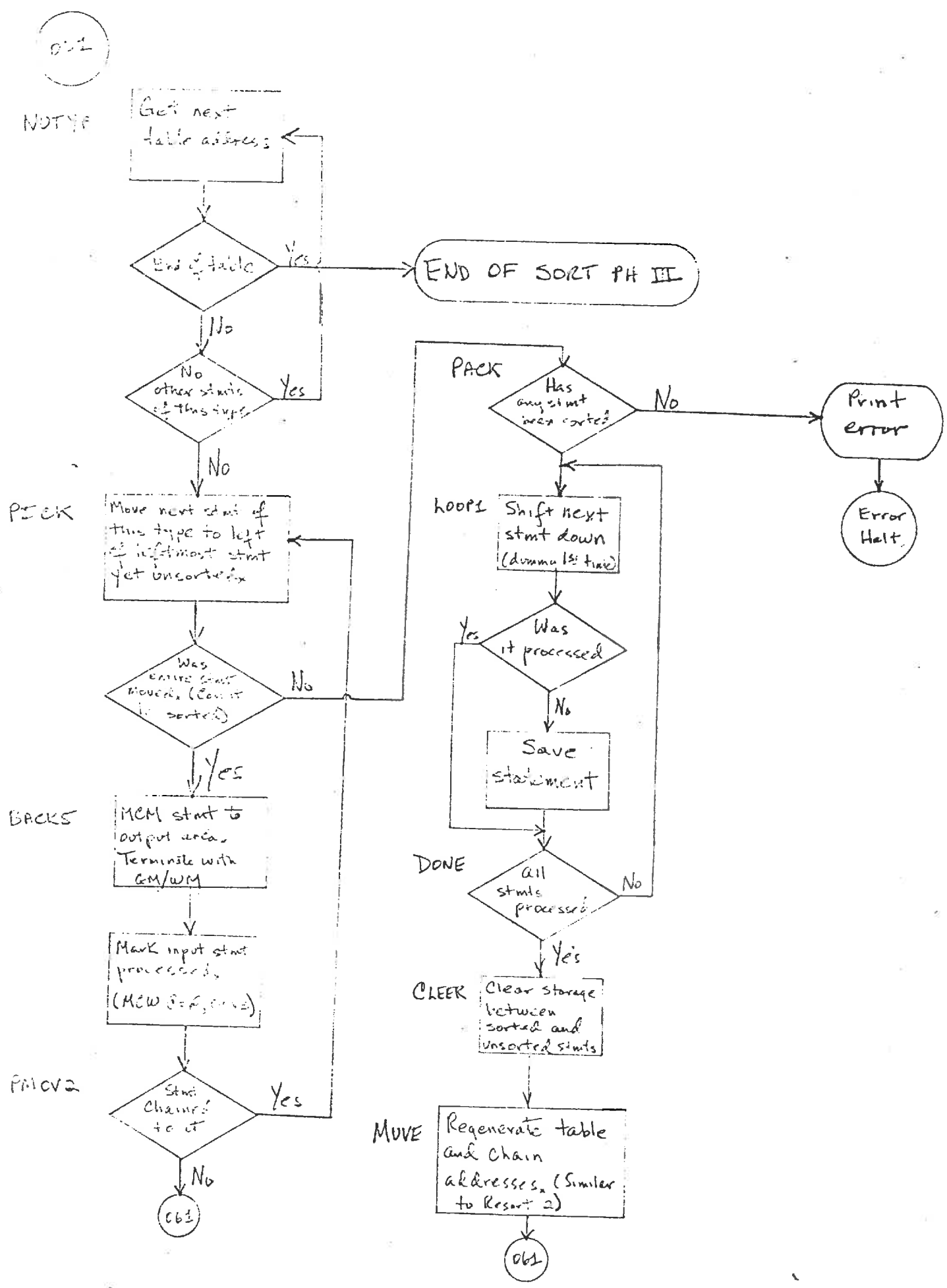
EOI
(XLZ = PAIRMA+2)

No

Yes

END OF SORT PH TWO

-10-
SORT PHASE THREE

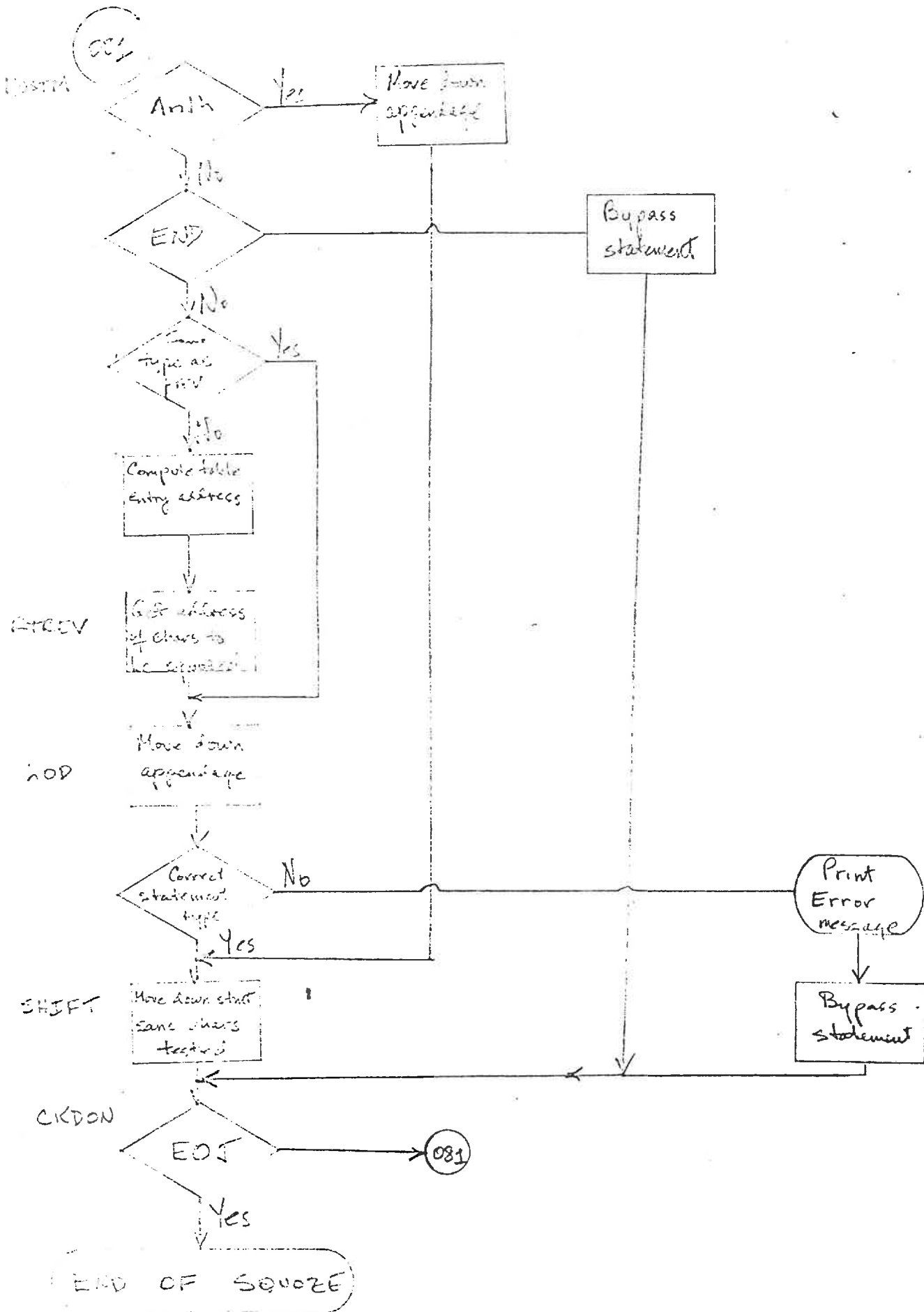


INSERT GROUP MARK PHASE

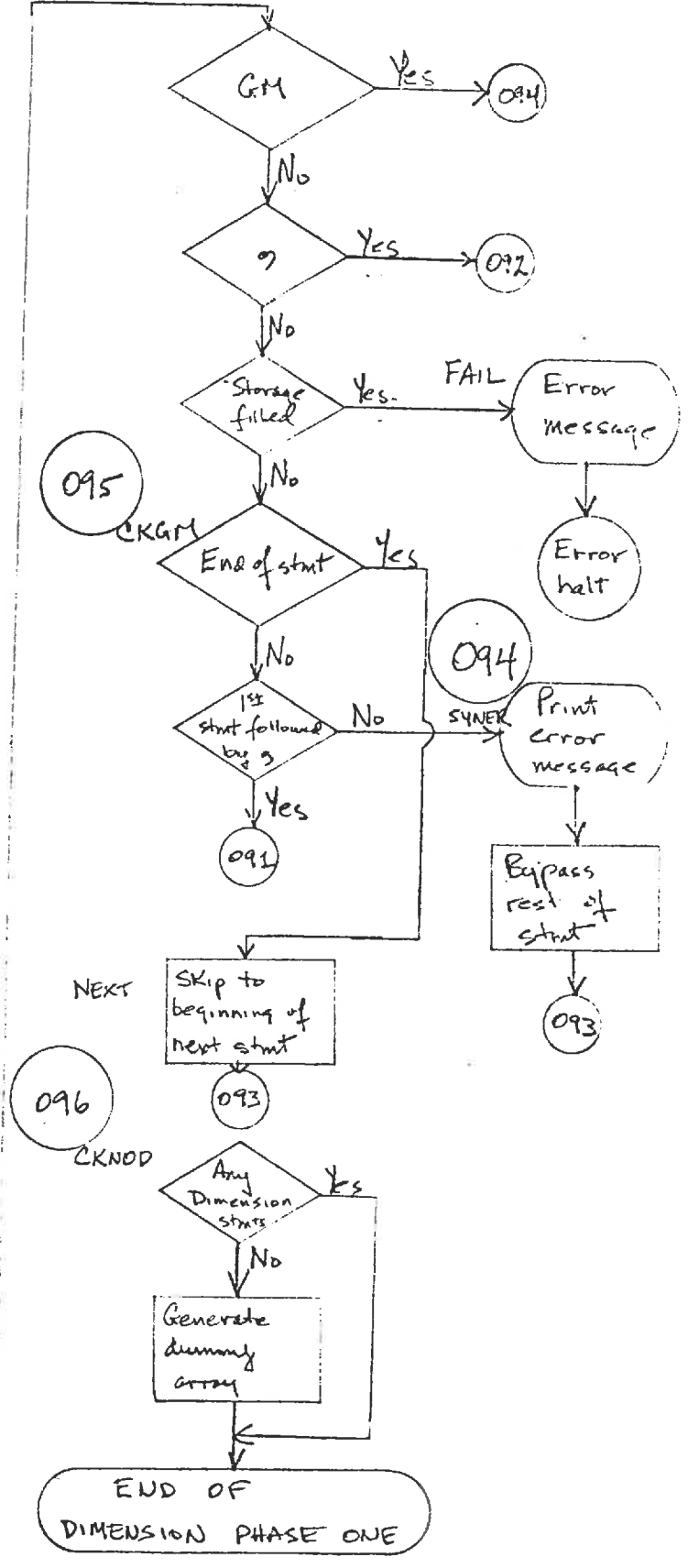
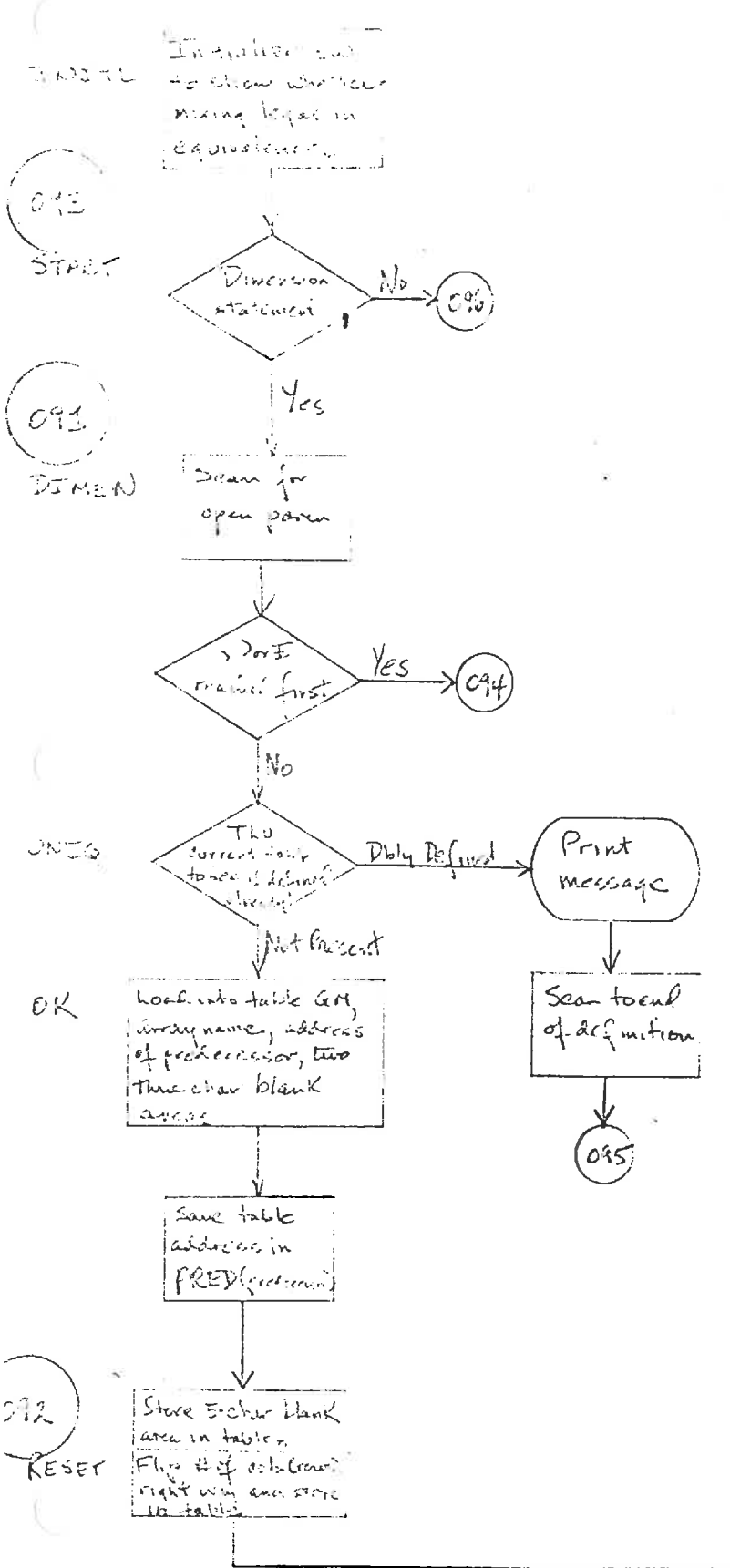
All 5-8 characters within the range of the source program are converted to group mark/word mark. This character appears between the body of the statement and its appendage.

This phase is terminated by reaching the character blank (BTEST). This test is NOPed when the statement is FORMAT (ISFMT).

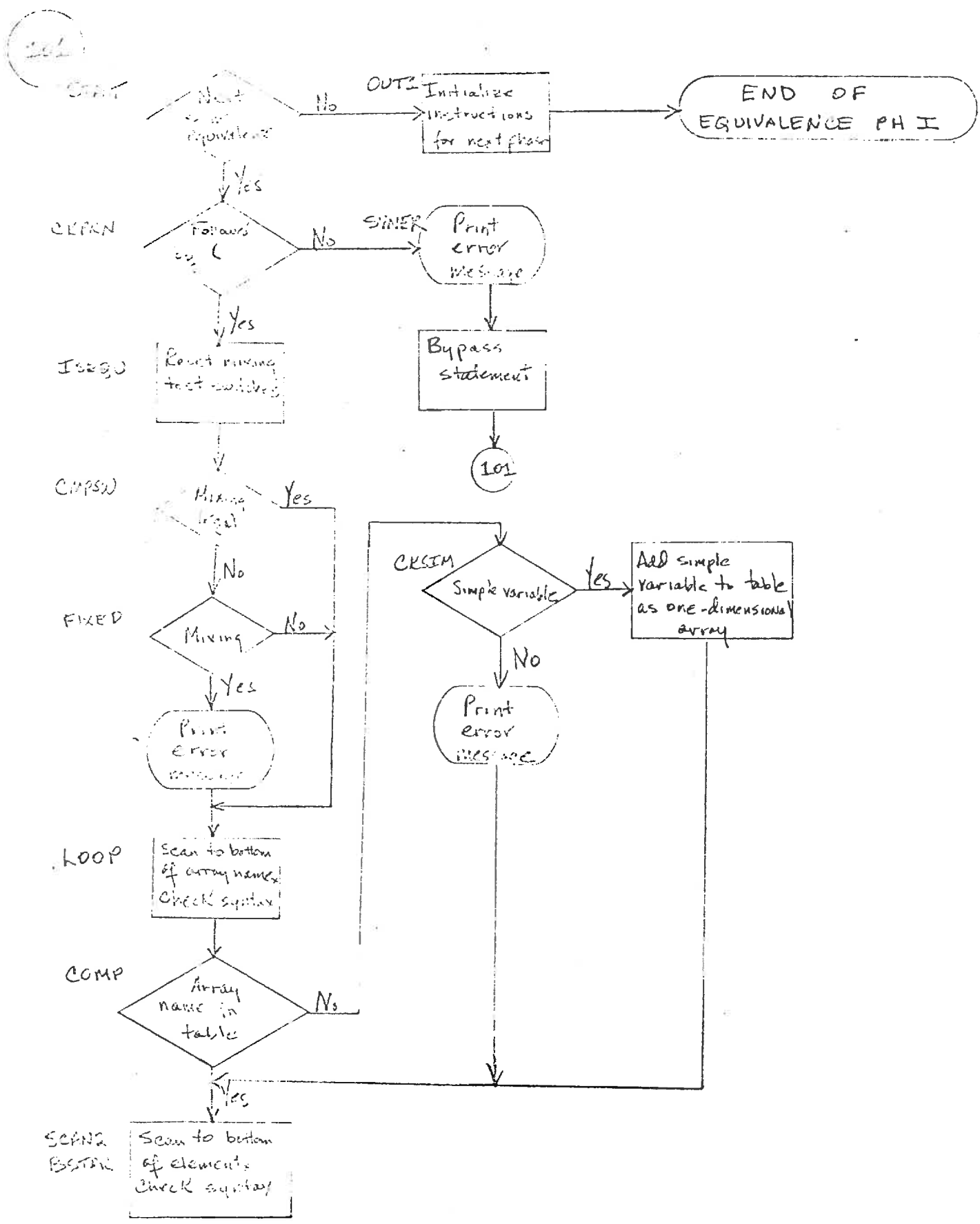
SQUEEZE PHASE



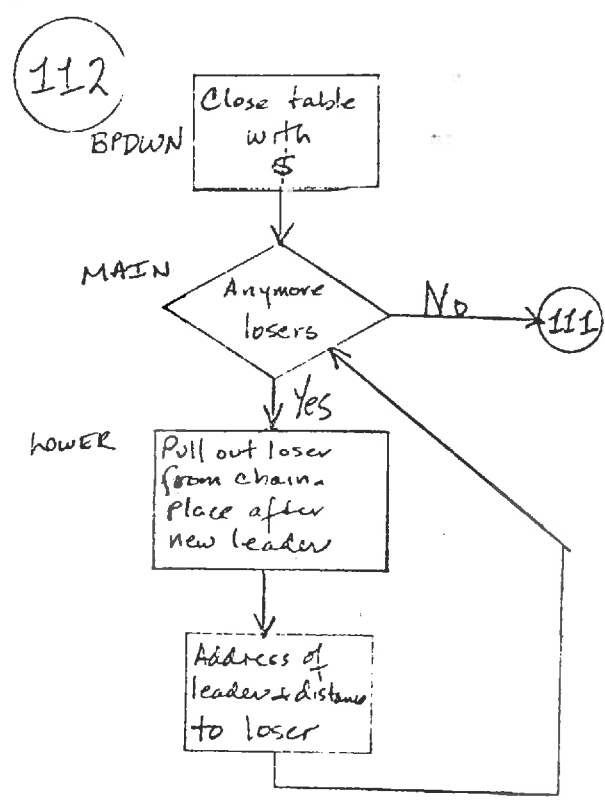
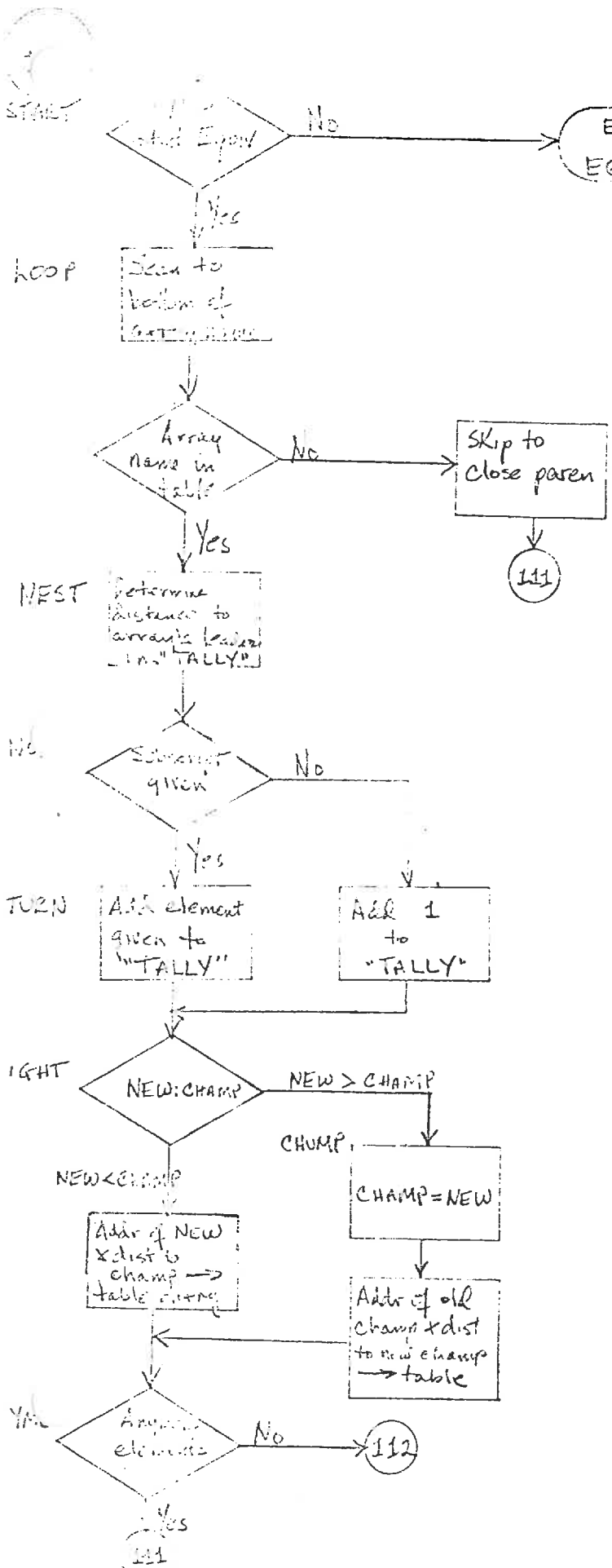
-13- DIMENSION PHASE ONE



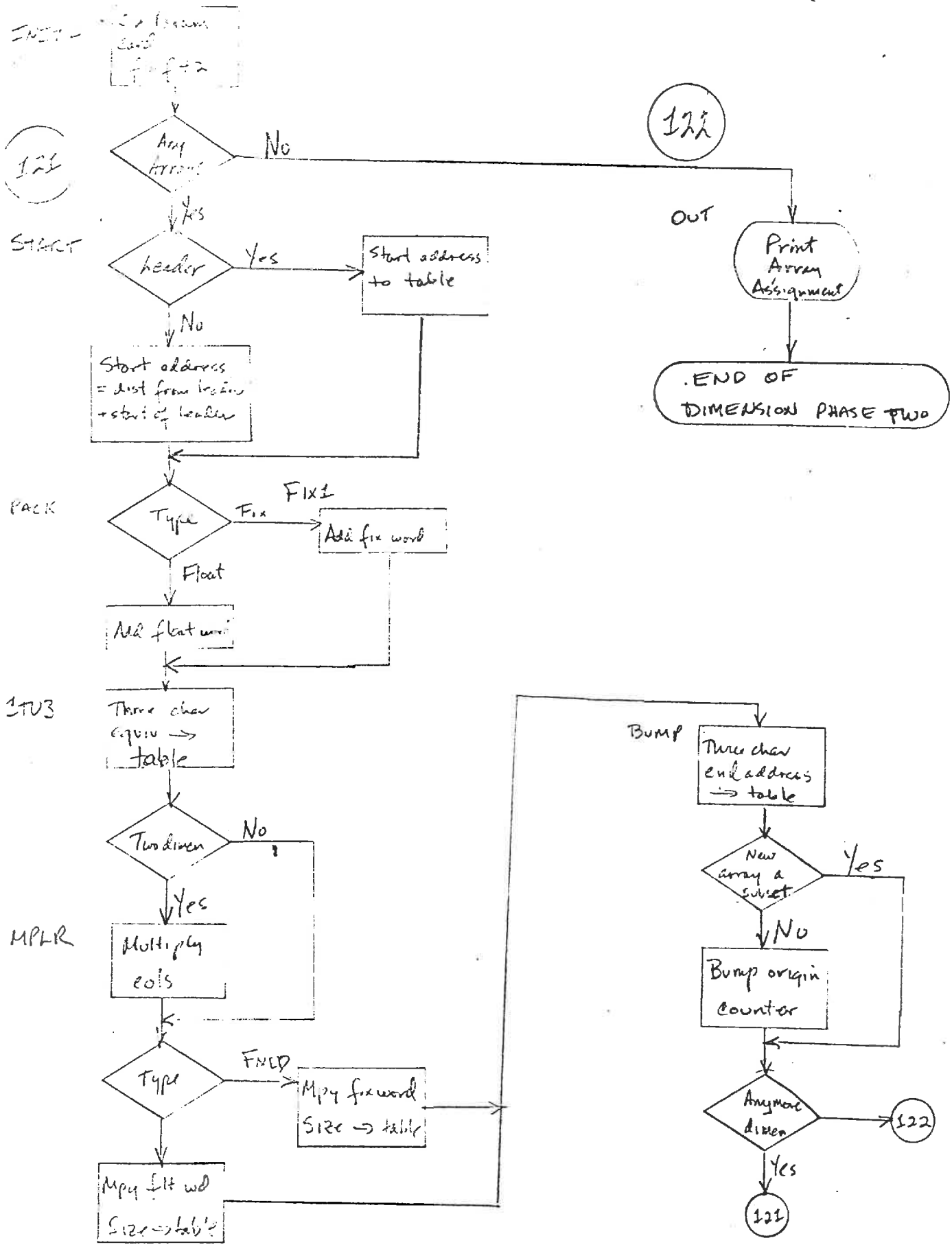
EQUIVALENCE PHASE ONE



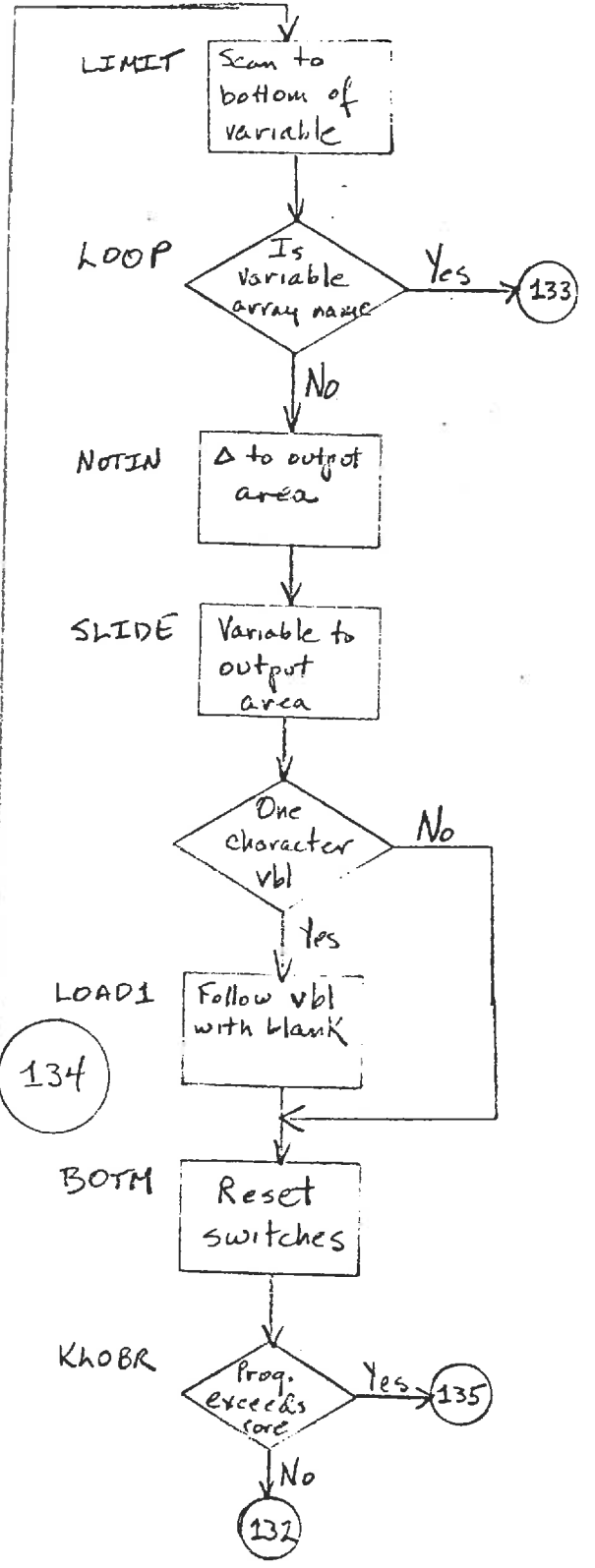
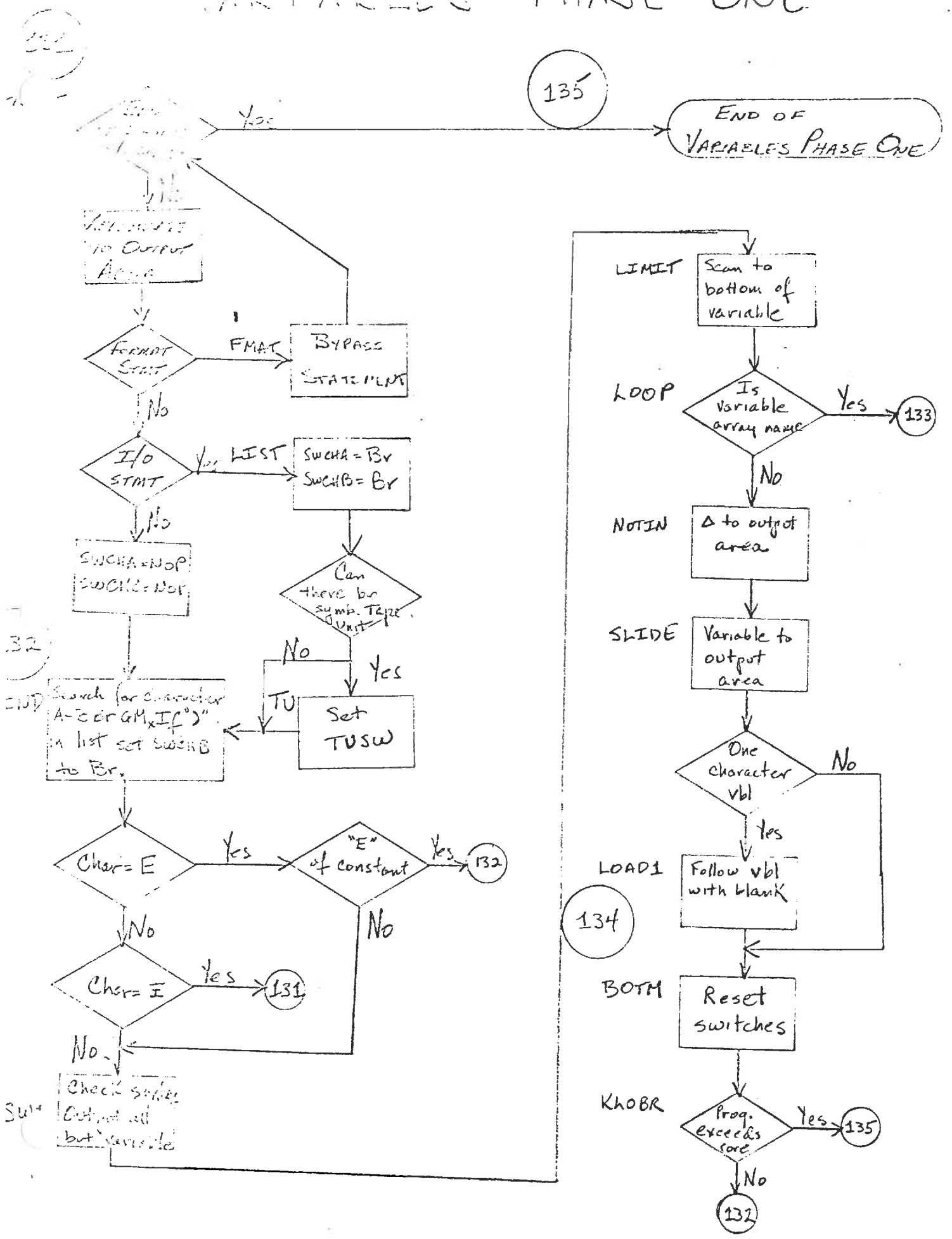
- 15 - EQUIVALENCE PHASE TWO



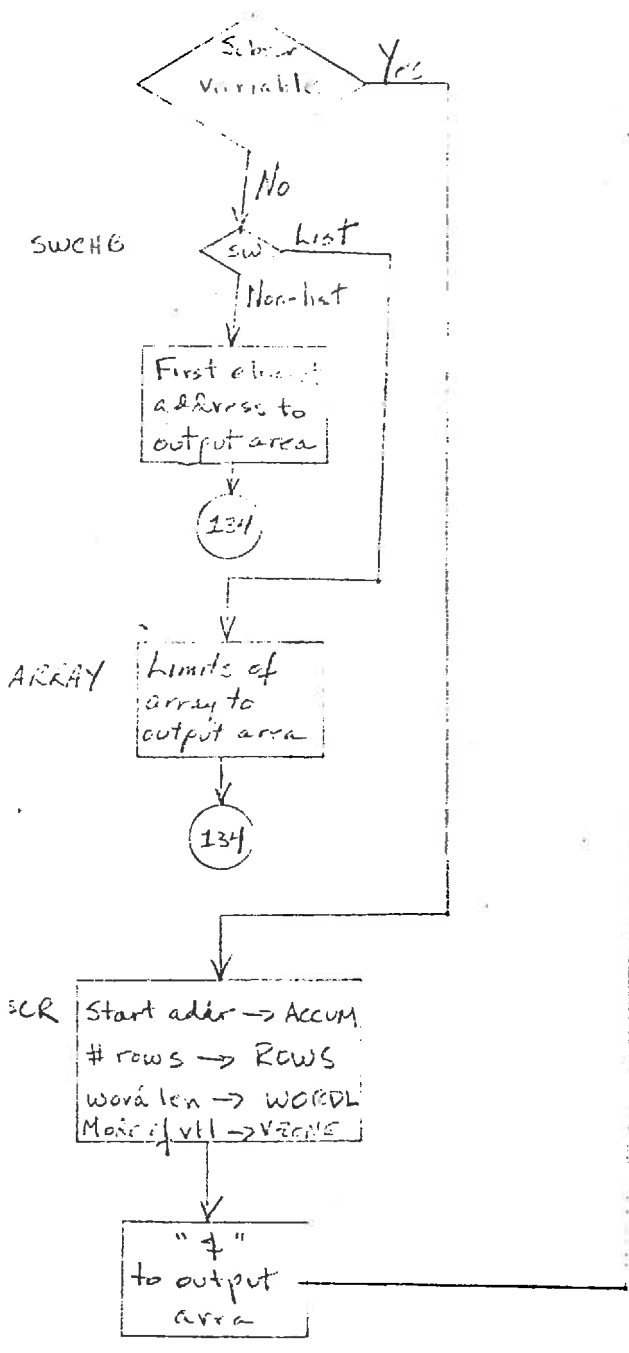
DIMENSION PHASE TWO



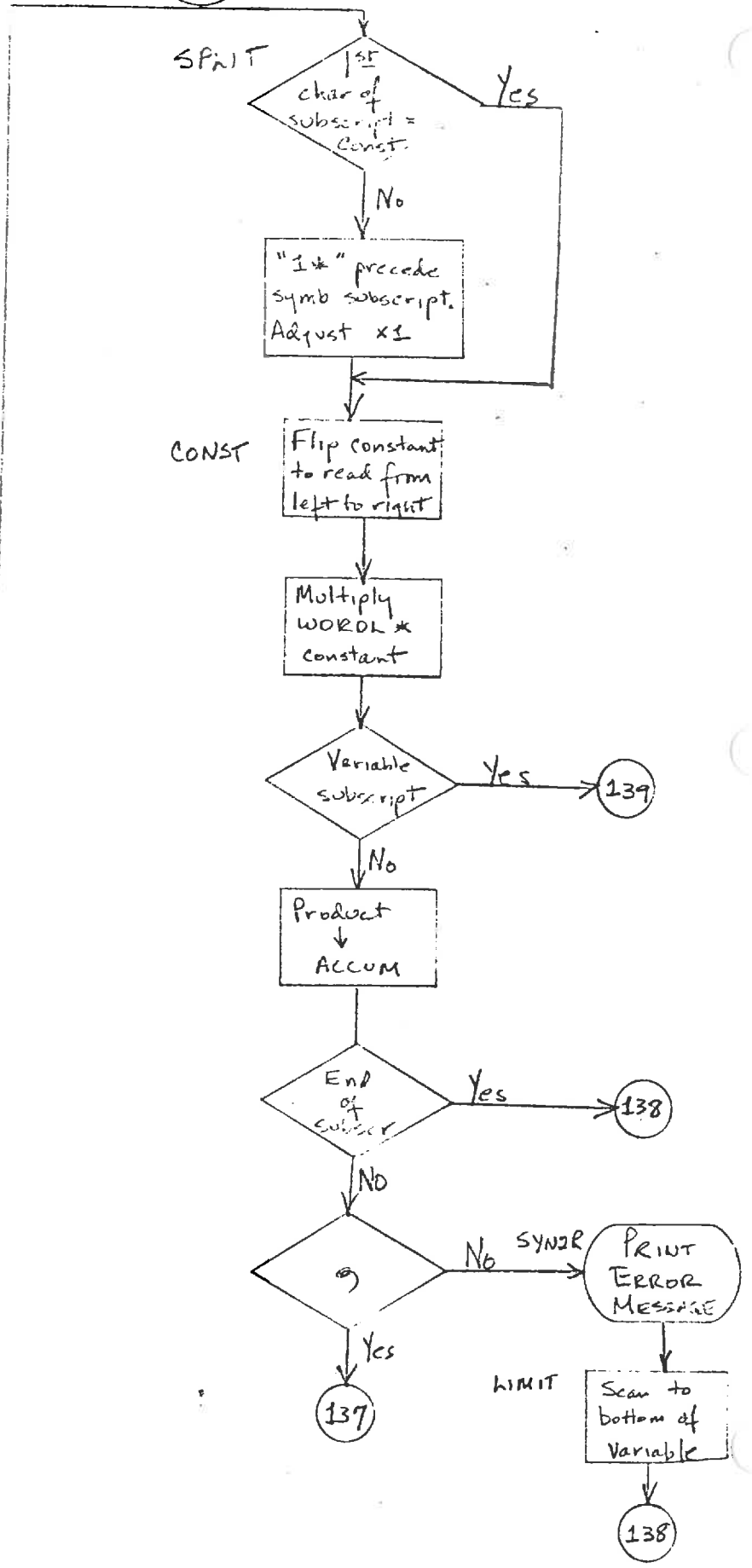
VARIABLES PHASE ONE



133



1311

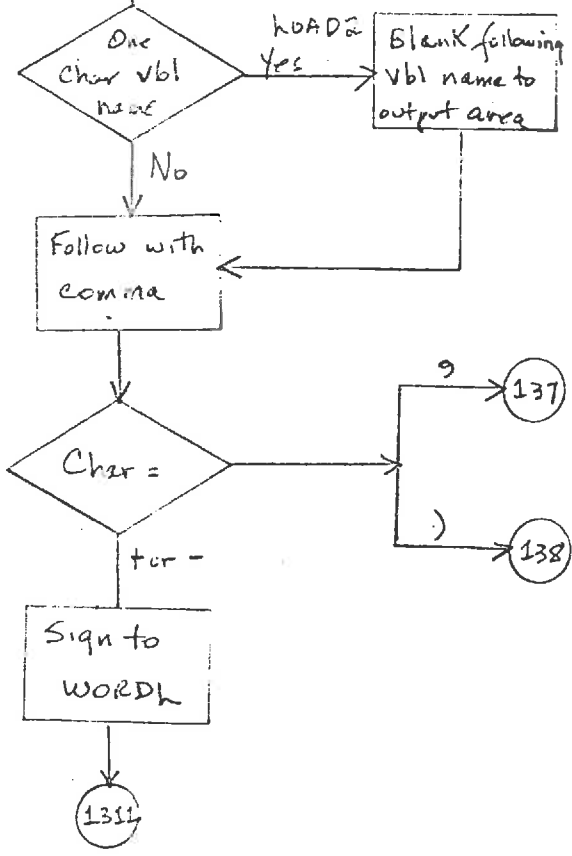


137

XC Multiply rows x word length Subtract result from ACCOM

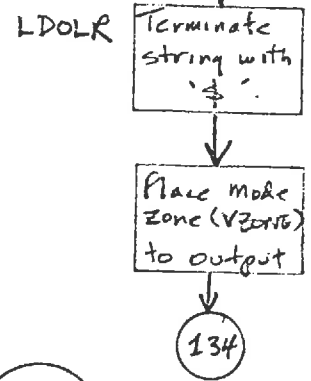
1310

ACCOM



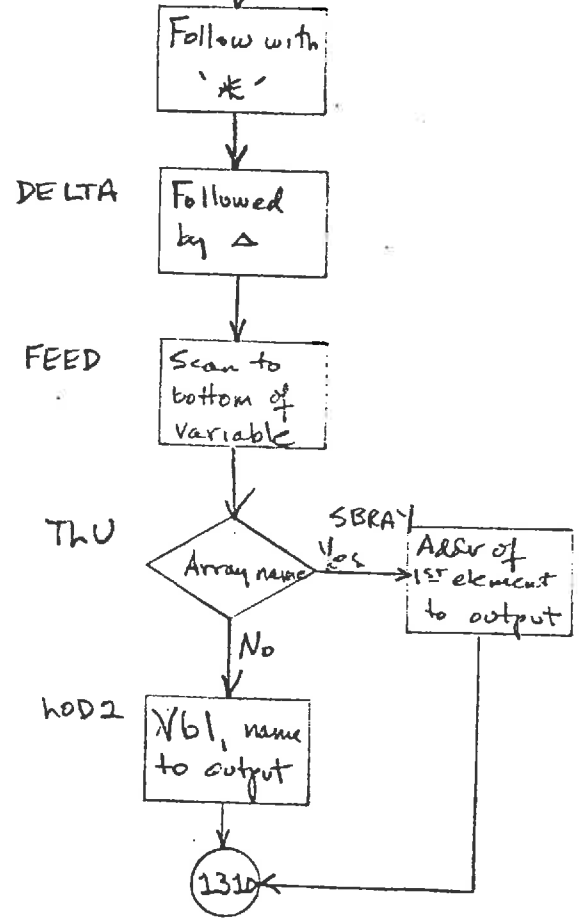
138

PHEW
NORMAL
ALCON
Get 16K modulus of constant, 3 char addr equiv to output

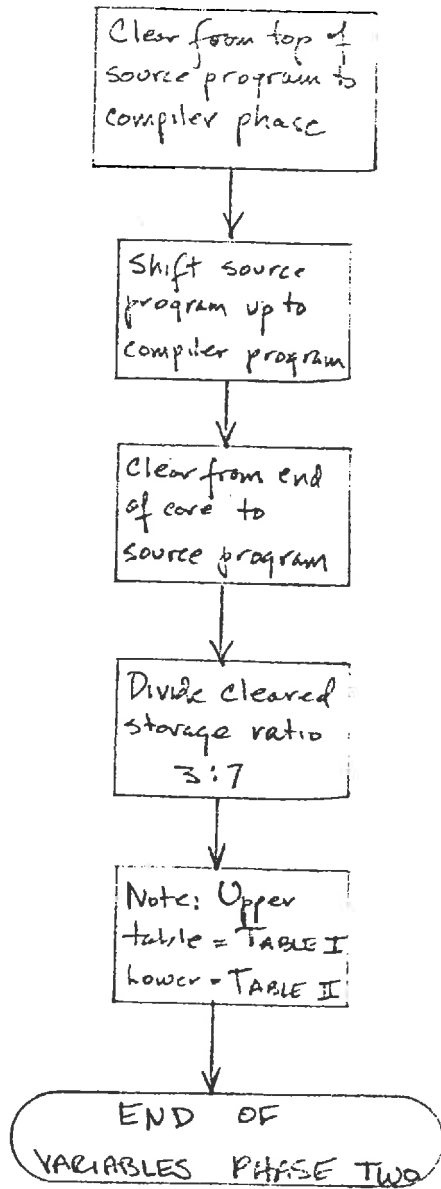


139

VARBL
NORMAL
TWIST
Get 16K modulus of constant, Place backwards in output area



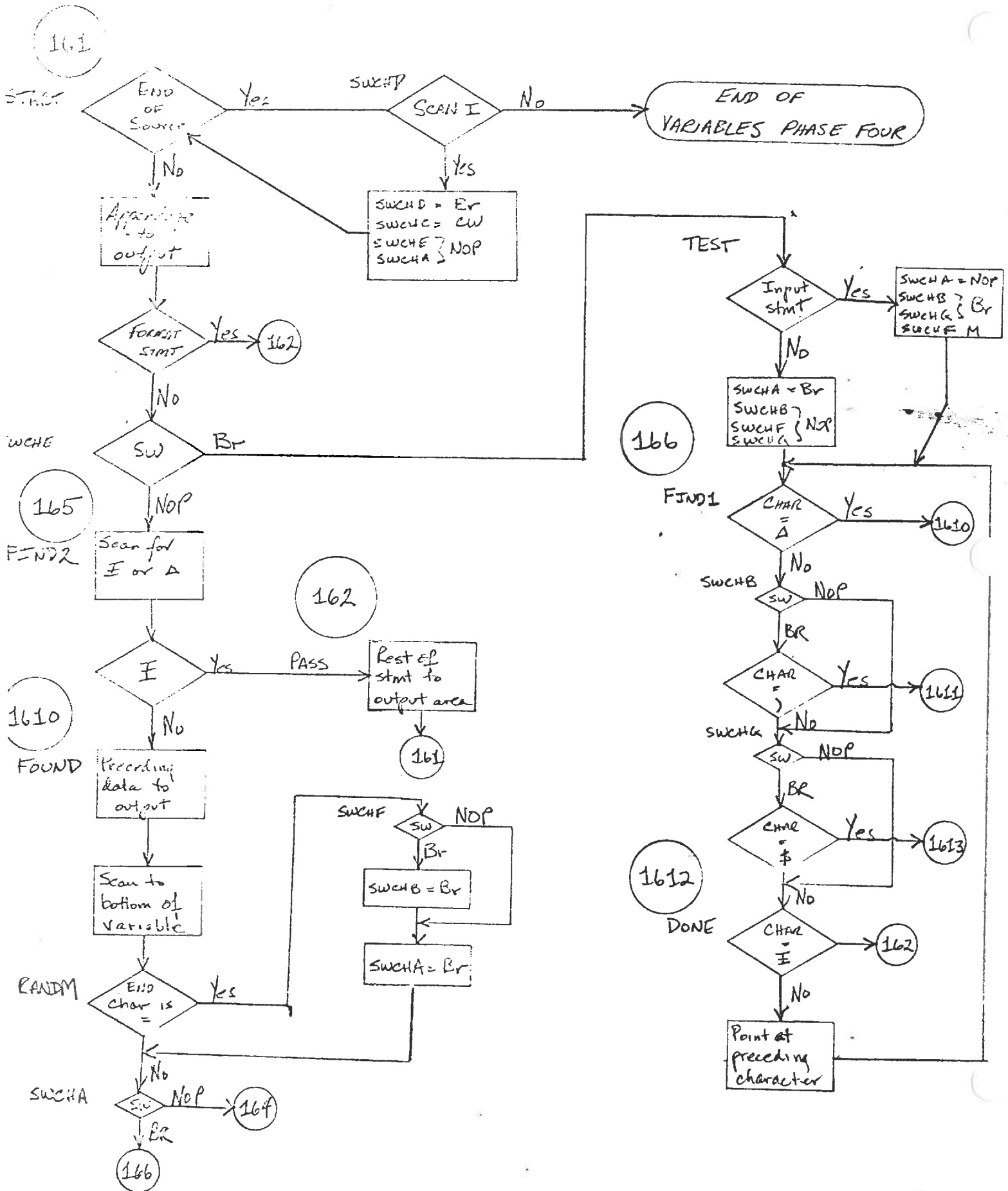
VARIABLES PHASE TWO

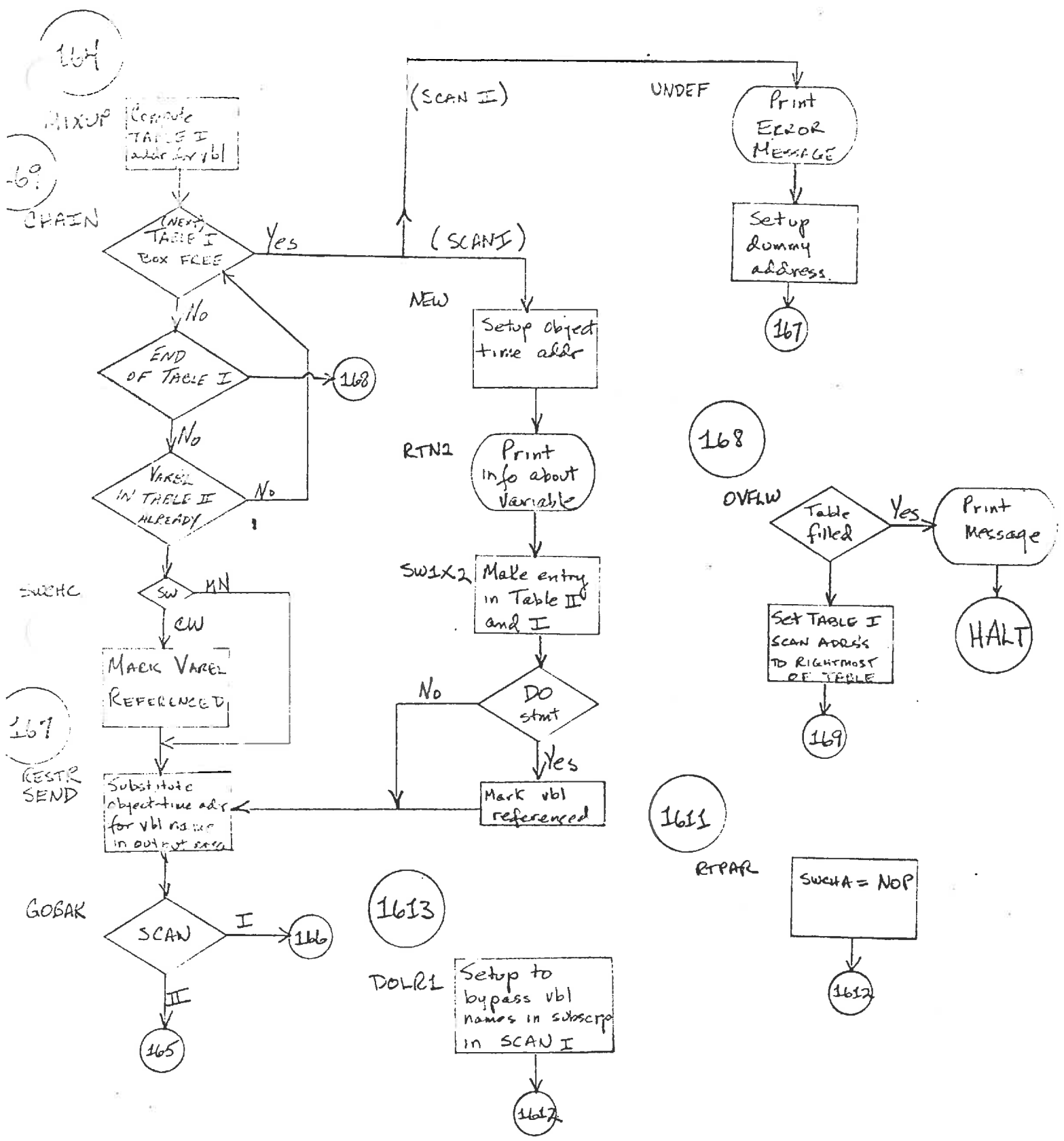


VARIABLES PHASE THREE

A housekeeping phase. The heading line "Storage Assignment - Simple
Variables" is printed. NXTOP is converted to five characters and is
stored in WORKS⁵.

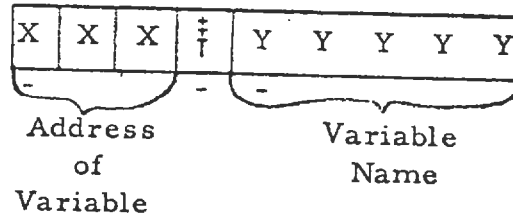
-22- VARIABLES PHASE FOUR





VARIABLES PHASE FIVE

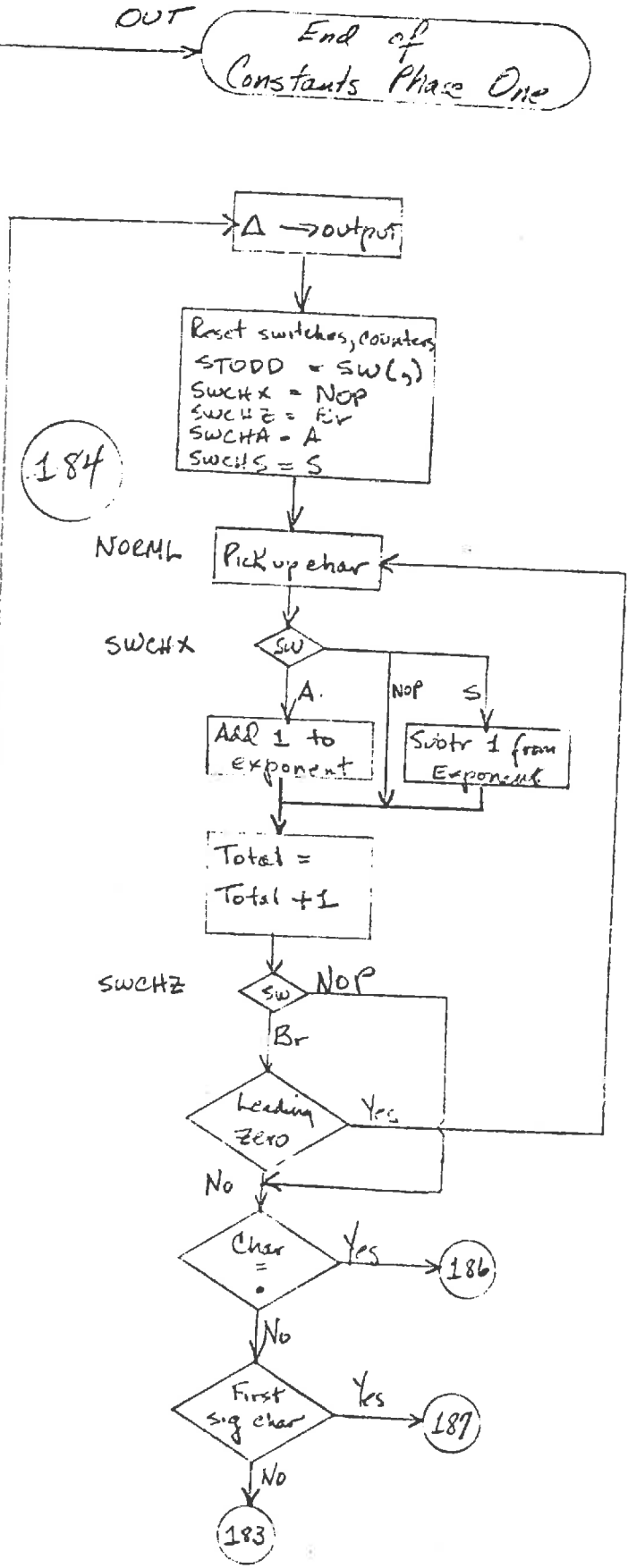
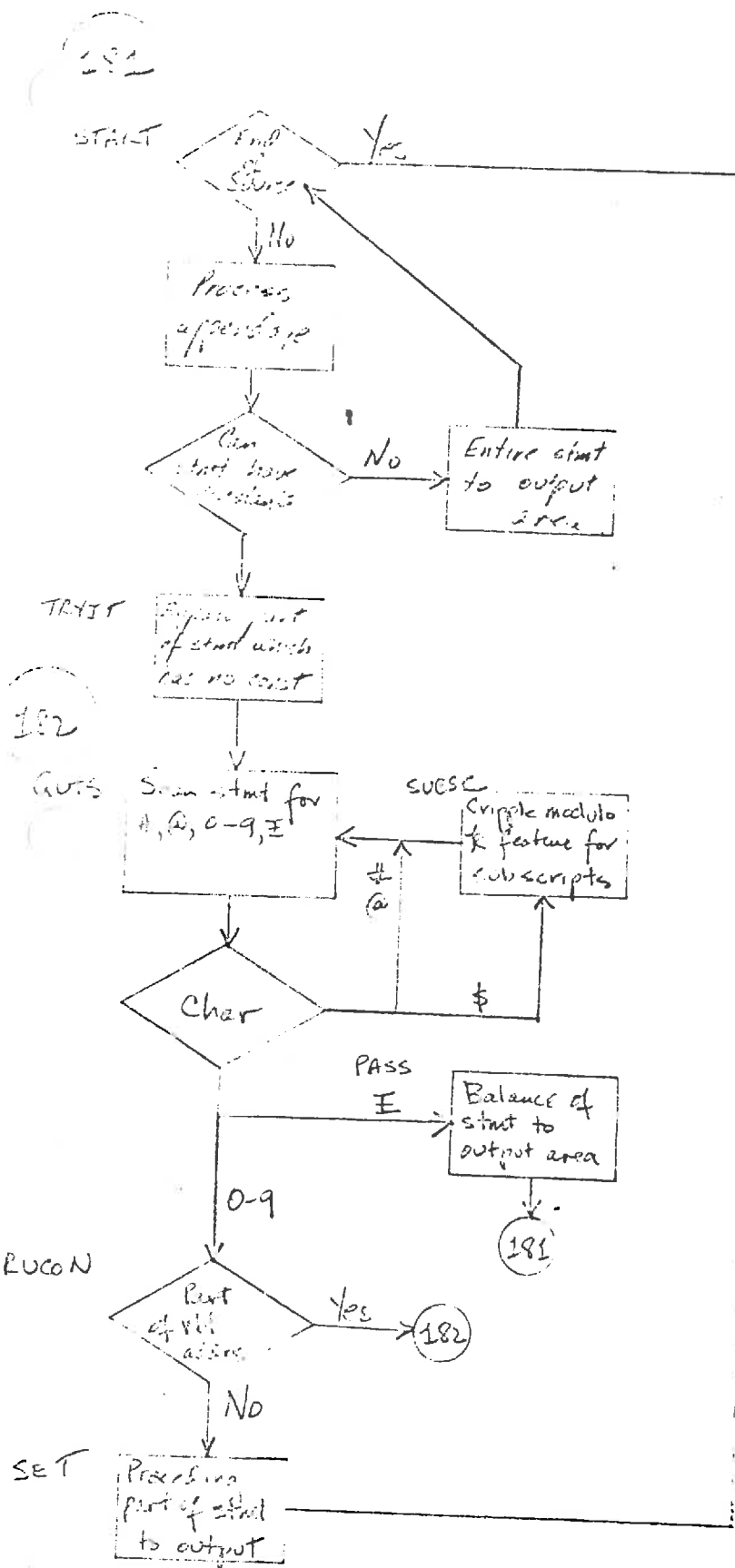
This phase scans for unreferenced variables. Each element in Variables Table II has the following format:



The word mark is cleared from the group mark in Variables Phase Four (SWCHC) when a variable is referenced.

This phase scans Table II for entries where the word mark still exists.

CONSTANTS PHASE ONE



193

Add one to count

Char =
Yes → Print error message

Char 0-9
Yes → 184

Delete start
181

Fixed + const
No

Adjust exponent count if necessary

CKTAK Process 'E' type exponent if present

Float zero
Yes → HDZER → +00 to output
185

RTLFT Delete decimal point unless it precedes first signif. char

TWIST Pick up mantissa Truncate if necessary

PHAW Delete low order bases if any

XEUNT Add exponent to constant string & store constant in output

185
BOTH
BOTH2

House Keeping

192

FIXED
Fix zero
Yes

'06' + output
185

TURN Pickup constant, Modulo if necessary except for subscr constants

Constant ↓ output

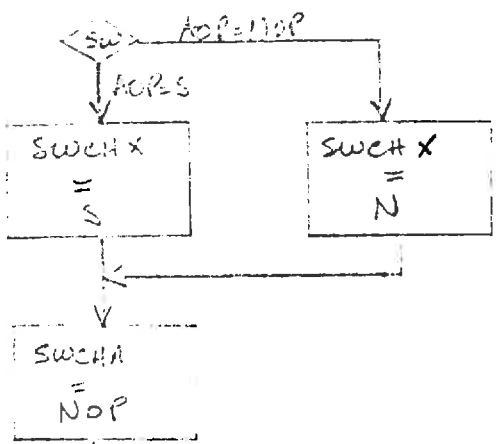
Const size = 1
Yes

Blank ↓ output

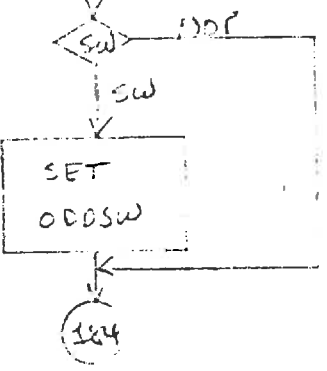
185

186

SWCHX

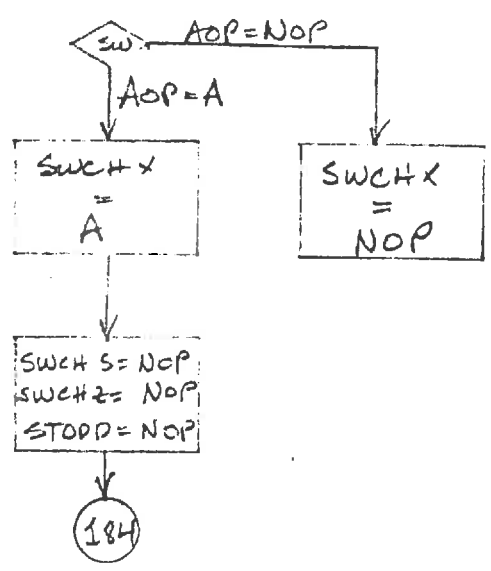


STODD

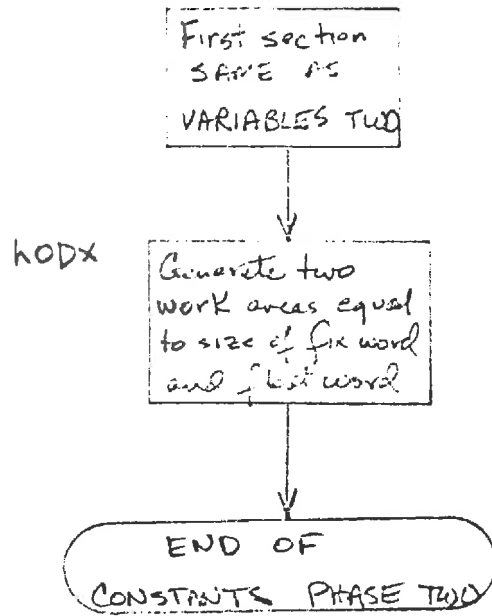


187

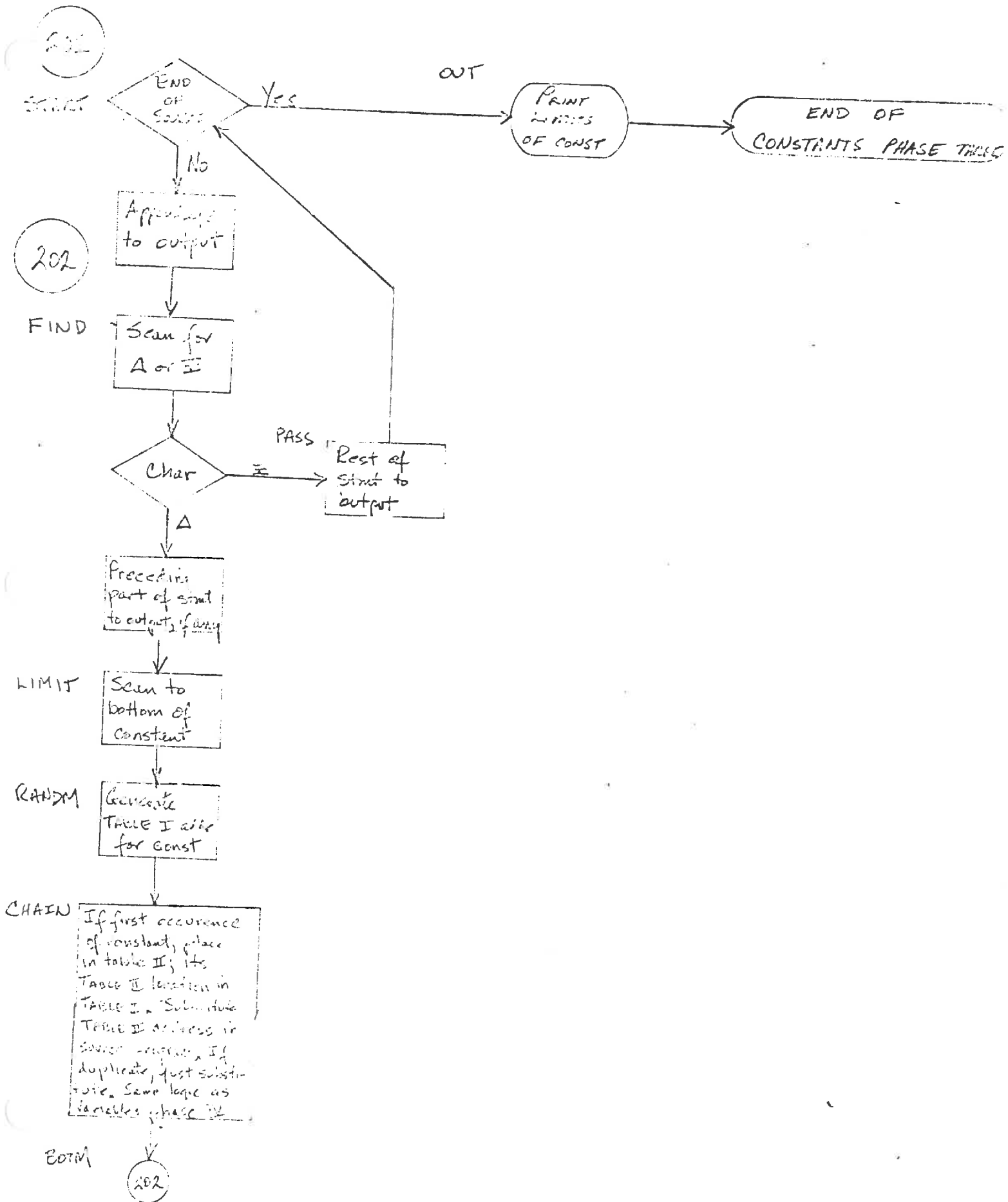
SWCHA



CONSTANTS PHASE TWO



CONSTANTS PHASE THREE



SUBSCRIPTS PHASE

This phase cleans up subscripts, eliminating the commas and asterisks necessary for proper processing of subscript constants.

The end result is the subscript parameters as they will exist at object time.

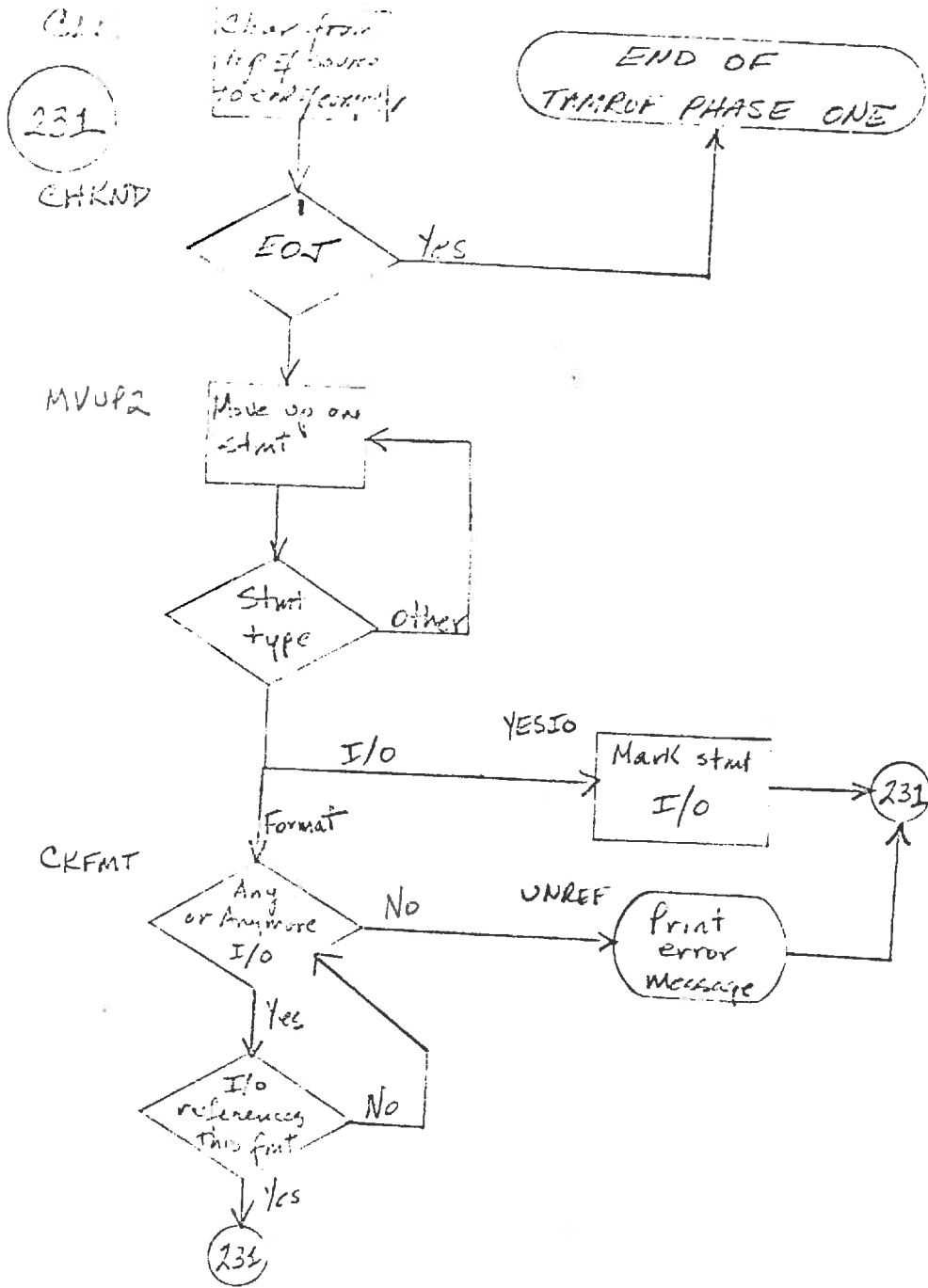
STATEMENT NUMBERS PHASE ONE

Processing in this phase is straightforward. All statement numbers are converted to unique three character representations. A table of 50 characters (TABLE) is used. The literal 50 is subtracted from the second and third, and fourth and fifth positions of each statement number. If the result is positive for the latter, one (1) is added to the first character. If the former is positive, two (2) is added to the first character. The characters in the table replace the three sections of the original statement number.

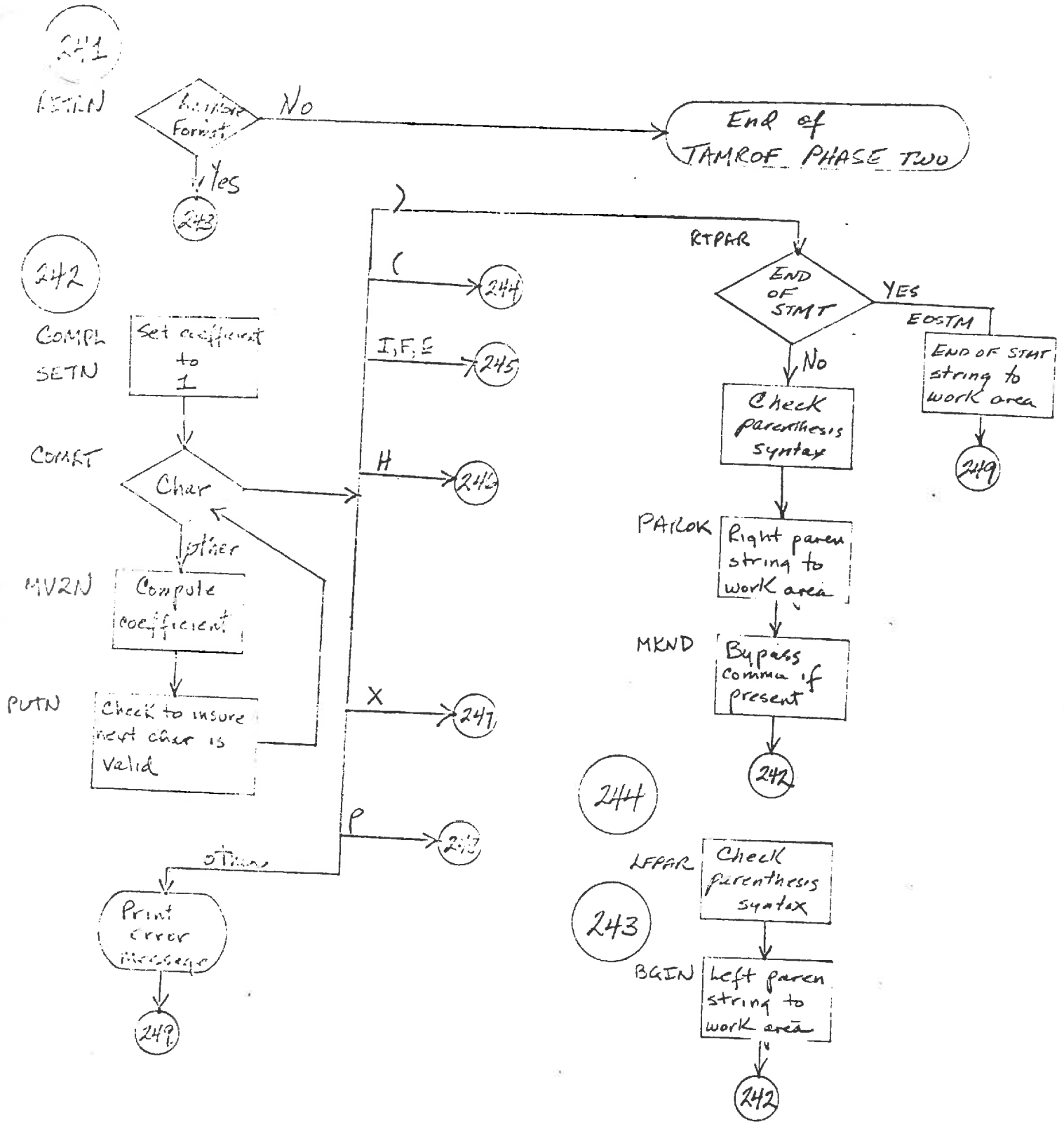
Each statement type is processed separately due to the different locations for statement numbers. The unique representations ^{for the statement numbers are} placed at the beginning (rightmost) part of the statement and terminated with a comma.

Adequate error checking is provided to check syntax and to insure that all required statement numbers are present.

TAMROF PHASE ONE



TAMROF PHASE TWO



2/15

ARST

Get branch, get char to work area

MW2N

Get width to work area

Type

Subtract 4 from width

CLRE

Make sure period follows

MV2N

Get decimal position factor

Width to work area

IEXIT

Type

Decimal factor to work area

MKN7

Bypass comma if present

242

2/16

HNOT

LINKAGE TO OBJ FORMAT TO WORK AREA

FLIP

Hollerith chars to WORK AREA

MKNP

Bypass comma if present

242

247

XNOT

Generate in-line proceed in work area

2/18

PGET

Width, 'P' linkage to work area

2/19

FINIS

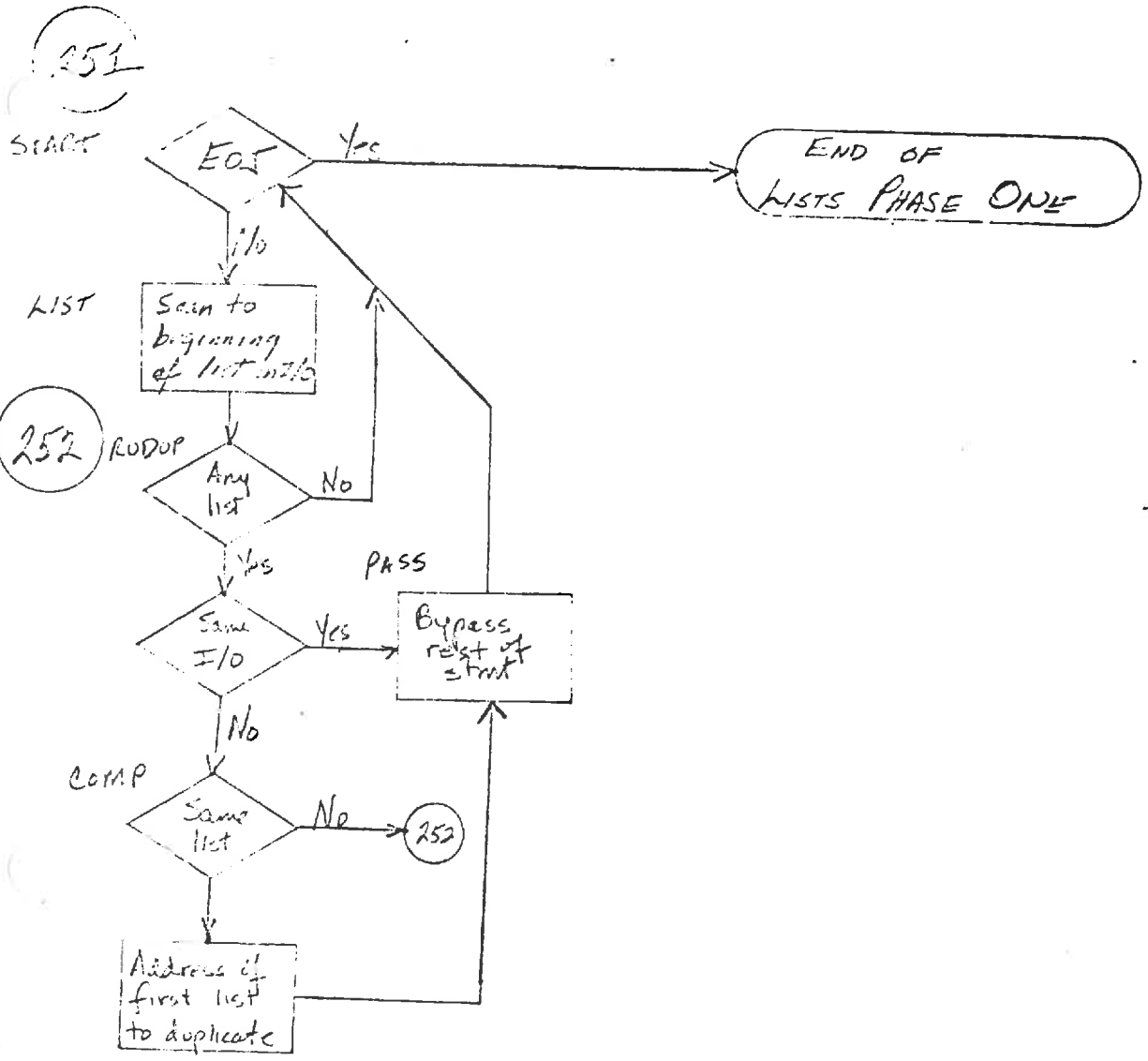
Work area to output

IOWX

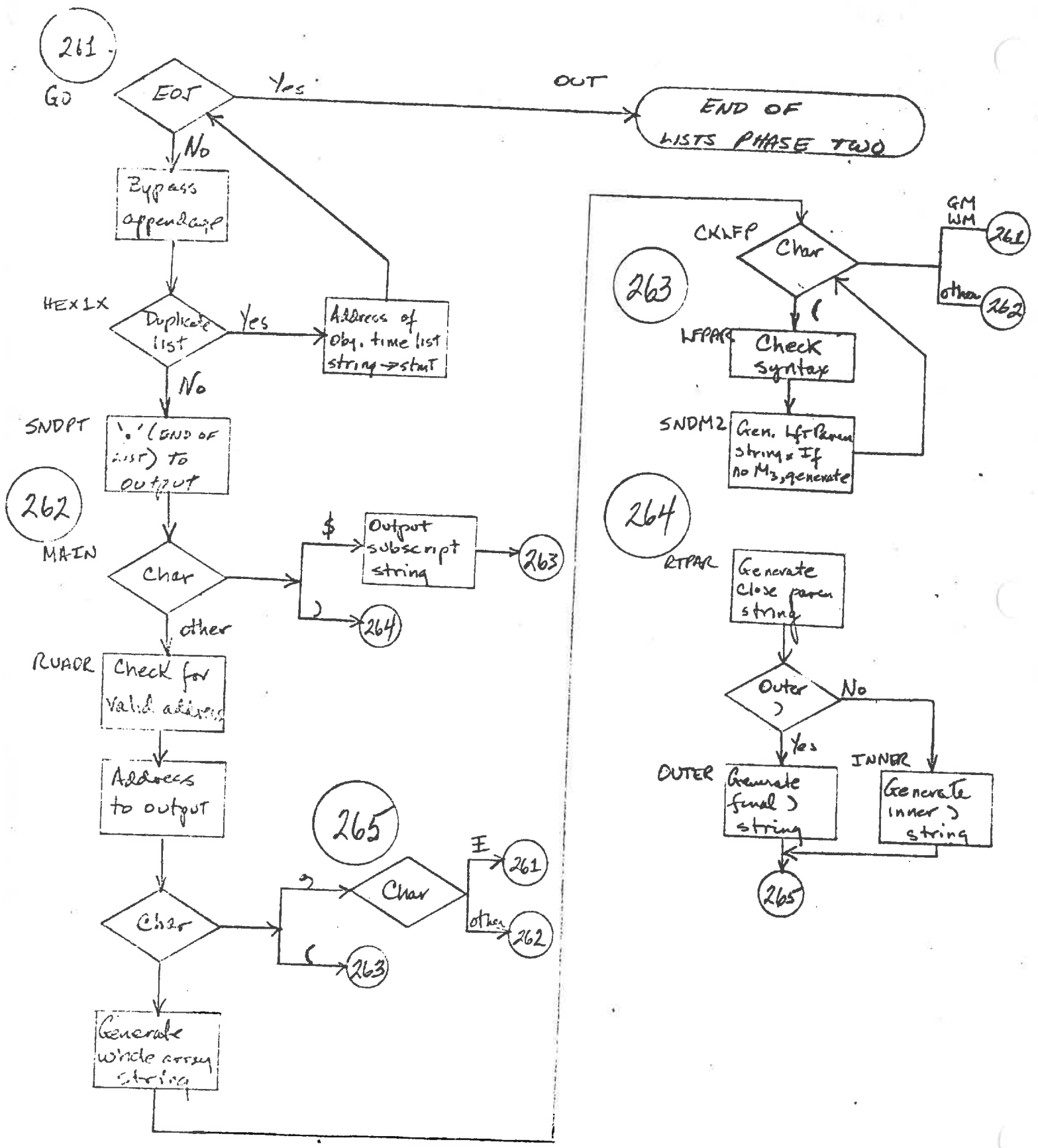
Substitute address of string in I/O stack which references it

241

LISTS PHASE ONE



LISTS PHASE TWO



LISTS PHASE THREE

This phase squeezes out extraneous characters from I/O statements, reducing each to the address of the format string (if any), the last string (a dummy address if no list) and the tape unit number (where applicable).

STATEMENT NUMBER PHASE THREE

The logic for placing in the Statement Number Table the three character equivalence for external statement numbers is identical to Variables Phase Four and Constants Phase Three.

Only the statement numbers within the body of the statement are placed in the table. The address ^{where} ~~when~~ it is placed is substituted for the three character equivalence in the body of the ^a ~~statement~~.

STATEMENT NUMBERS PHASE FOUR

The external statement numbers are matched against the Table of Statement Numbers that were present in the body of the statements. If there is an entry in the table, this entry is replaced by the internal sequence number of the statement which it references.

To illustrate the progress of statement numbers, consider these two statements:

1. GO TO 20
2. 20 STOP 1 2 3

Prior to Statement Numbers Phase One, these two statements have been reduced to:

1. 02 G 012
2. 321 025016

In Statement Numbers Phase One, the number "20" (appears as "02" above) is converted to a three-character unique representation:

1. XYZ G012
2. 321 XYZS016

In Statement Numbers Phase Three, "XYZ" is placed in a statement numbers table by virtue of the GO TO expression. The table location is substituted for XYZ in the GO TO statement.

- 71 -
1. G4Z G012
 2. 321 XYZS016
 3. XYZ
 ↑
 └── location G4Z

In Statement Numbers Phase Four, the table entry XYZ is located when processing the STOP statement. The internal sequence number of the STOP is substituted in the table

1. G4Z G012
2. 321 SG4Z
3. 016

The compiler has now established for future phases that the GO TO statement will transfer control to internal sequence number 016.

Do statements receive special treatment in the phase. The compiler requires that each Do statement have an entry in the statement number table.

If a Do statement has no external statement number, or if this number is unreferenced, an entry is placed in the table (PSUDO).

This phase also detects unreferenced and multiply defined statement numbers. Unreferenced statement numbers are those which have no table entry. Upon referencing a table entry in Table II, the three character representation is placed in Table I. If another statement occurs with the same representation, it is detected as multiply defined.

STATEMENT NUMBERS PHASE FIVE

This phase checks for Undefined Statement Numbers. This occurs when an entry in the Statement Number Table was unreferenced by the previous phase.

Note that Dimension, Equivalence and Format statements have been eliminated prior to the statement number phases. As a consequence, all references to these statements will produce an error message.

INPUT/OUTPUT PHASE ONE

All I/O statements except Rewind, Backspace and End File are reduced to ^{their} three object time procedure string.

B W 72 X A A A B B B

Where "BW72" is executed by the machine as branch to 1672, the location of the format package; X appears as the I/O type

* ~~z~~ - Print

+ - Read

- - Punch

or the numeric portion represents the tape unit number and the zone indicates the I/O type.

NZ - Read tape

A - Write tape

B - Read Input tape

AB - Write Output tape

If the tape unit numbers is symbolic, the above string is preceded by an instruction which is symbolically represented as

MN ^{NOT SPACE} III, X + 5

where III is the address of the fixed point non-subscripted variable.

ARITH PHASE TWO

This phase scans arithmetic and **If** statements for function names. Where they exist, the name is deleted and a one character code is substituted.

ARITH PHASE THREE

Key work areas and subroutines

- TRAP - value of the temporary store substring
- STOR - initially TRAP; bumped by one to create additional temporary store strings.
- NORTH - If this location contains a word mark, there are no arith statements.
- TP1SW - If no wordmark then exponentiation encountered.
- TP2SW - If no wordmark then multiplication/division encountered.
- TP3SW - If no wordmark then addition/subtraction encountered.
- STAR 2 - Starting address of mult/div string
- STAR 3 - Starting address of add/sub string
- PREV - Previous operator to mult/div string (used to force negate or invert function)

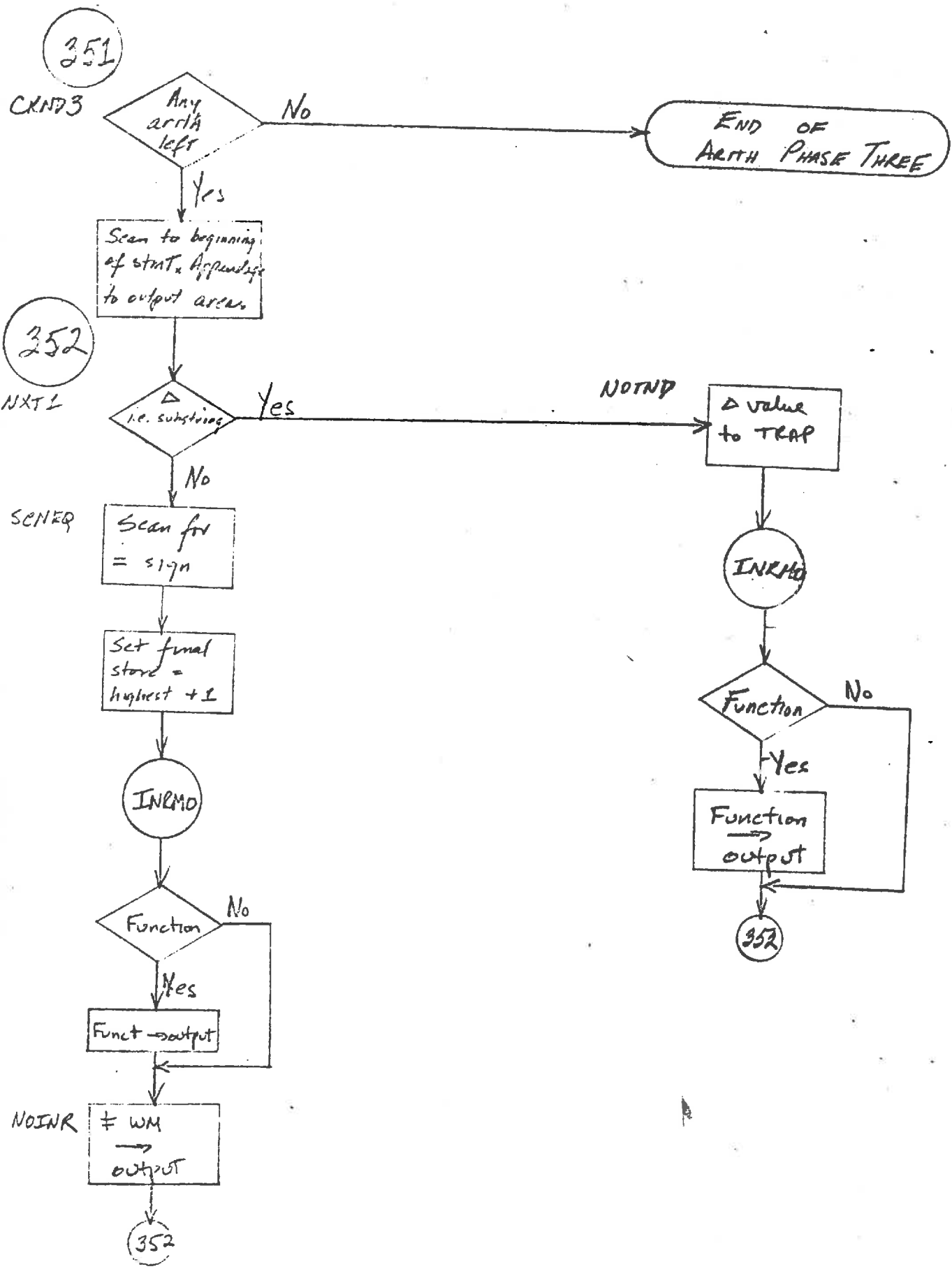
Subroutines

- INRMO - ^{This} The subroutine constitutes the main portion of this phase. It breaks down the hierarchy of execution.
- GETNX - Gets next operand and operator. This is the most used subroutine within INRMO. It places the operator preceding the operand being analysed in OP and the succeeding operator in OP2.
- GTEMP - Where additional substrings must be created due to hierarchy, this subroutine generates another temporary work area and places it in the output area.

STATEMENT NUMBER PHASE TWO

Same as Variables Phase Two.

ARITH PHASE THREE



INRMO

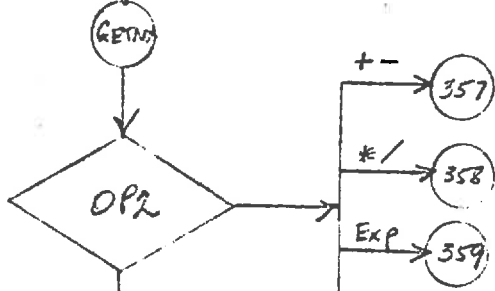
INRMO
OP3

TRAP → STOR
Reset Switches
If first op not preced
by sign, make +
preced op → OP2

353

3511

ADBB
CKHI



STOR3

Start of nest
→ STOR3
Set TP3SW

3510

YT3



355

NSWET

start of nest → X3
Operator of
nest start → OP2

356

GEN3

GENM

Set
TP3SW



354

EAK

Make new
string
Δ(stor)

353

TR13



MLAST

last A
=
First A

Subroutine
exit

DMP



TEMP

Get temp
hold area
(A)

NIX
MDET

Set TP2SW
string →
output



356

359

STAKA
Get temp hold (A)
Set TP1SW
OP → PREV; + → OP
String to output area

GETM

KILLX
Check multy upon
String to output
Kill part outputted
from input

MT2

TP2SW

NoWM → STAK2 ↓ x3

TRY2

OP2

other → 3510

DMP2

GETM

OP2

Exp

other

TP1SW

WM

NoWM

Get temp hold

Set TP3SW, TP2
String → output

OP2

other

*/

Eliminate
string from
input → 3510

358

STAKB

TP2SW

NoWM

WM

Start of nest
→ STAK2
Set TP2SW

GETM

OP2

other → 3511

357

STAKC

TP3SW

NoWM

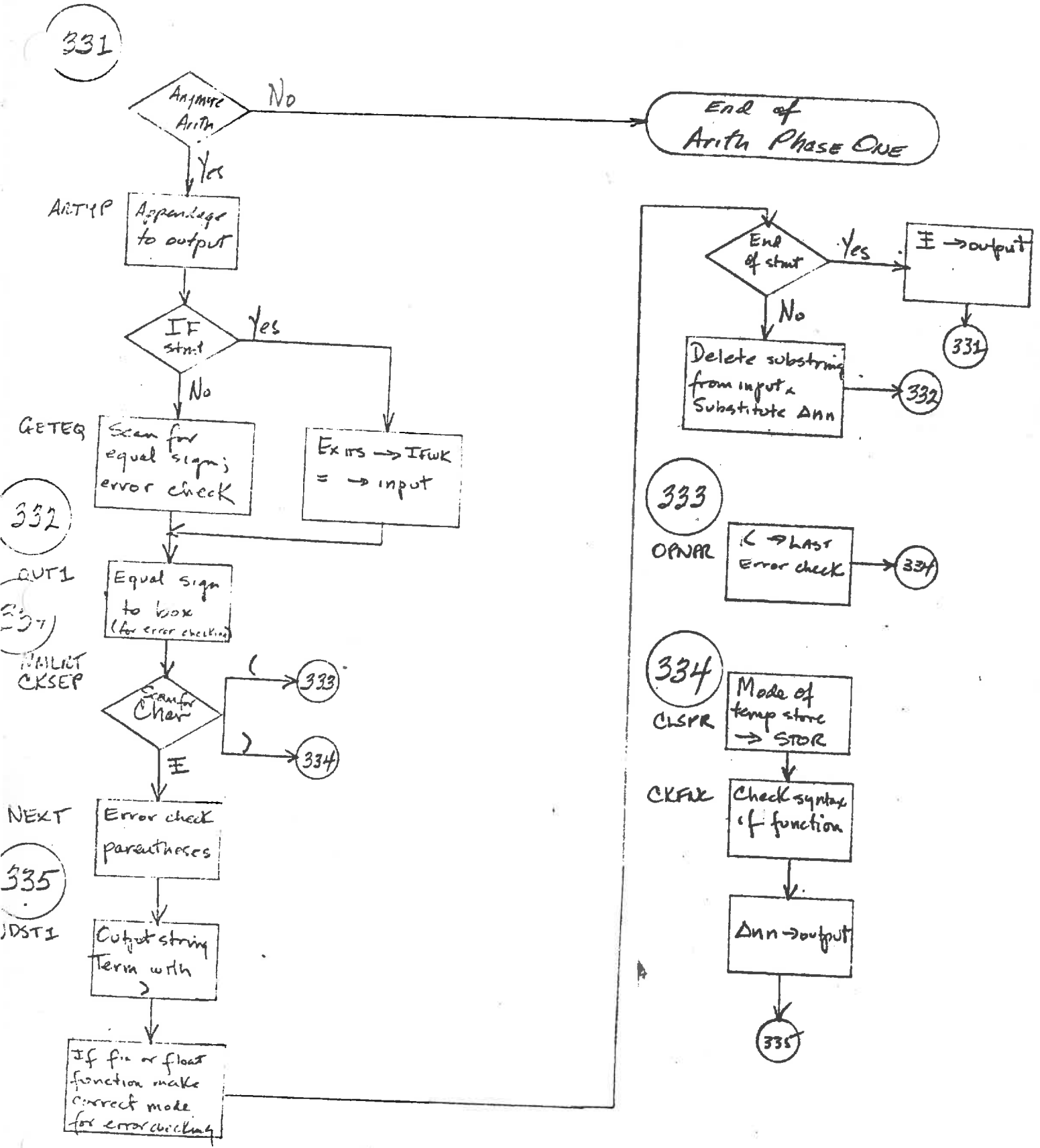
WM

Nest start
→ STAK3
Set TP3SW

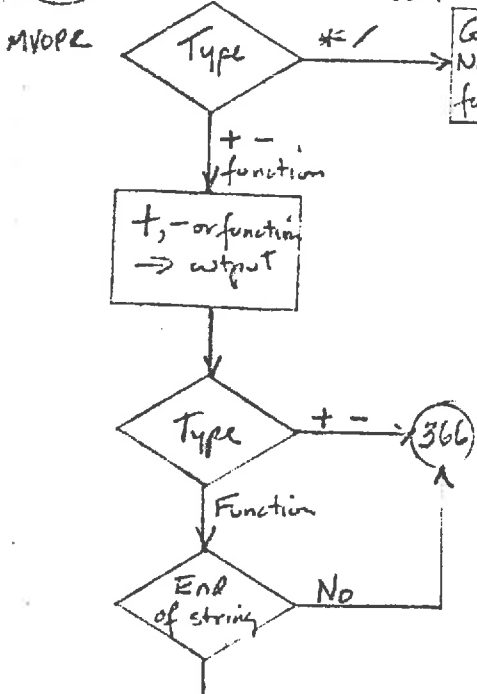
GETM

3511

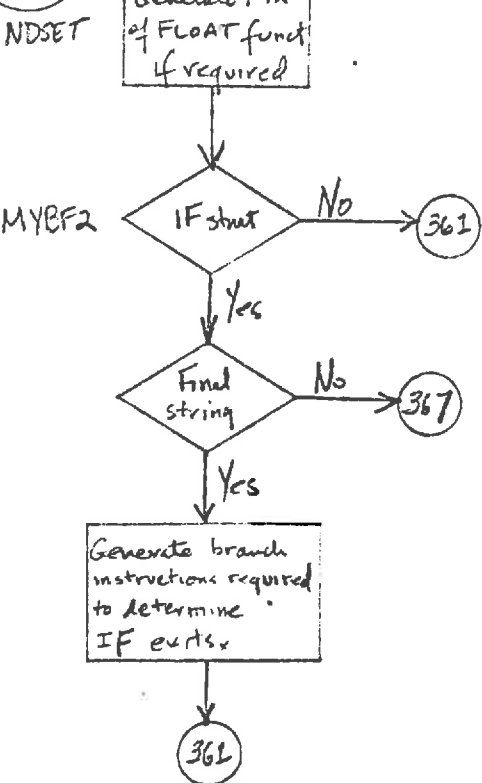
ARITH PHASE ONE



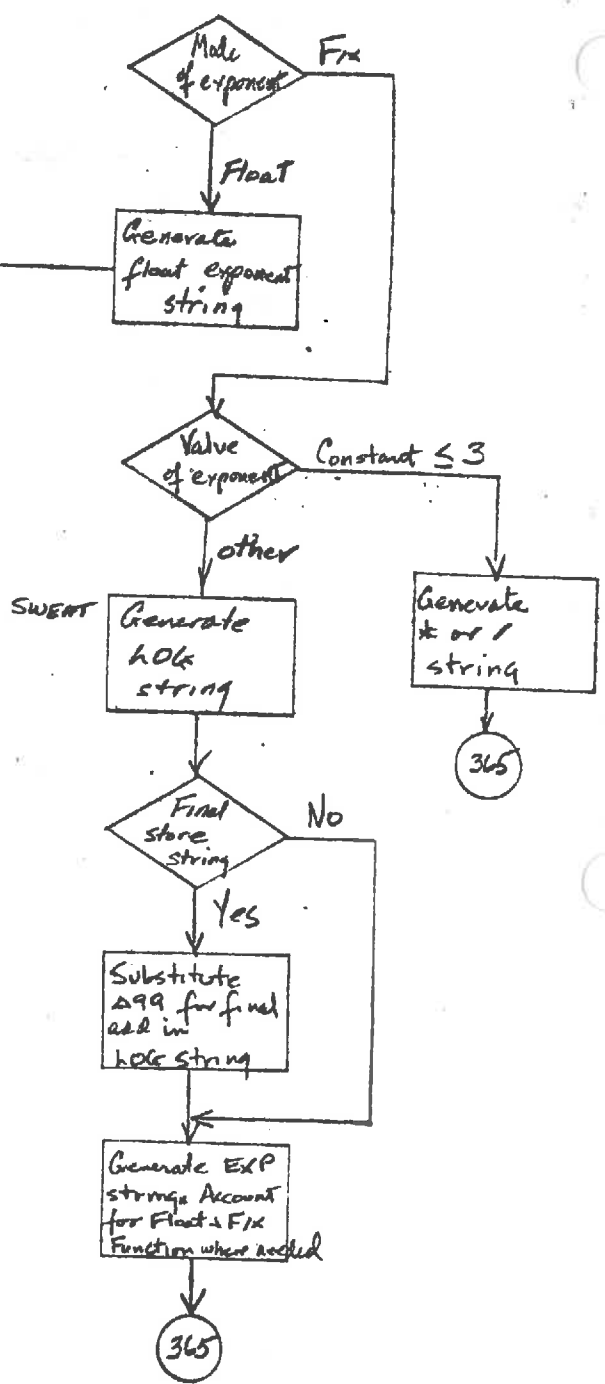
362



363



364



ARITH PHASE FIVE

PHASE 5

STARTS

KEY INIT

LOOP

WHICH

TRUE

Move down all arith
 Set SIZE = to
 $f+4$ or i^2 whichever
 is larger

EDS

Yes

No

Process
 appendage

Initialize
 TABLE to
 all 0000

Scan for
 Δ or $\frac{f}{i}$

Type

\neq

Δ
 type

left of =

Get next
 available
 temp store #

Place its #
 in nth entry
 of TABLE
 $n = A$ number

CALAD

Get temp store
 equiv from
 TABLE

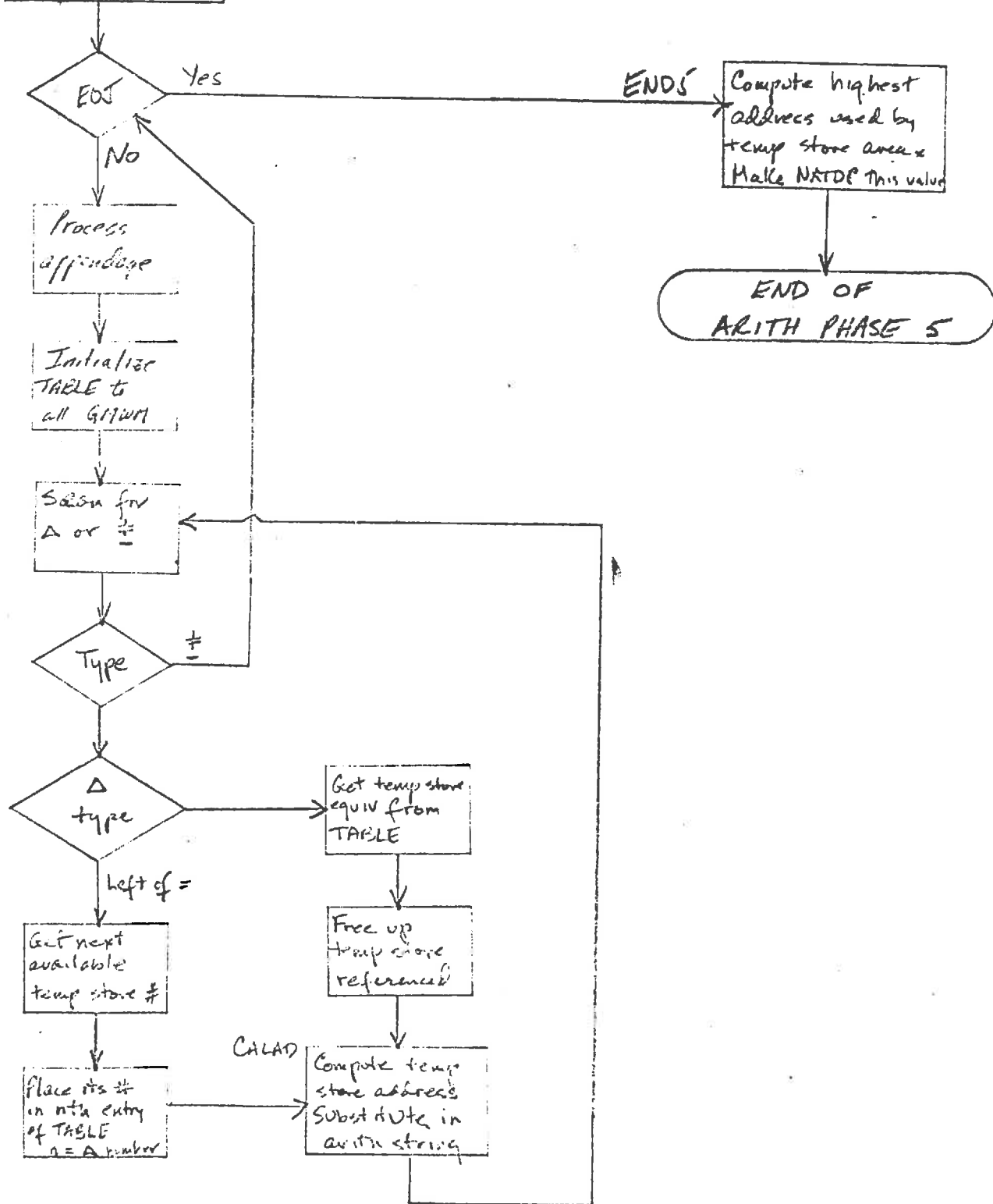
Free up
 temp store
 referenced

Compute temp
 store address
 substitute in
 arith string

ENDS

Compute highest
 address used by
 temp store areas
 Make NADP this value

END OF
 ARITH PHASE 5



ARITH PHASE SIX

This phase ~~plans~~^{Scans} arith statements for function codes and sets the switches to call ~~to~~ⁱⁿ the required function in the Function/Subroutine Loader Phase.

INPUT/OUTPUT PHASE TWO

Through

CONTINUE PHASE

These phases generate the required in-line instructions for proper execution of the various statement types involved.

← Input/Output Phase two

Rewind:

$\underline{U\%} \widehat{U_nR}$

Backspace:

$\underline{U\%} \widehat{U_nB}$

End File:

$\underline{U\%} \widehat{U_nM}$

where n is the tape unit number specified. If the tape designated is symbolic, the above instructions are preceded by an instruction which is symbolically represented as

$MN \left| \begin{matrix} \text{word equal} \\ \text{III, X + 4} \end{matrix} \right.$

Computed Go to Phase

This phase ^{generates} operates the instructions

BCE XXX, III, A

BCE XXX, III, B

H
B ~~X~~* - 8

where XXX and YYY are the exits and III is the fixed point non-subscripted variable. If the value of the variable exceeds the number of exits, the machine enters the halt loop at object time.

Go To Phase

This phase generates

B XXX

for Go To statements

Stop/Pause Phase

this phase generates

- 1. NOP nnn
- 2. H
- 3. B ~~X~~* - 8

for stop statements, where nnn is the halt number specified or 000 if no number is specified. Pause statements are identical except that instruction 3 is not generated.

Sense Light Phase

Sense lights are represented by word marks in locations 081-084. The presence of a word mark indicates the light is off.

Sense Light 0

, 082084,

Sense Light 1 - 4

X08n

when n is the ^{sense} ~~same~~ light number

If (Hardware) Phase

This phase generates the in-line coding for If Sense Light and If Sense Switch statements. For If Sense Light, the instructions generated are

VXXX03n ^{sense} ~~1~~08n BYYY

where XXX is the exit if the light is off; n is the ^{sense} ~~same~~ light number; and YYY is the exit if the sense light is on.

For If Sense Switch, the instructions are

BXXXCBYYY

where XXX is the exit if the switch is on; YYY is the exit if the switch is off; and C is a function of the sense switch number, switches 1-6 represented as B - G.

If the exit is to the next sequential statement when the sense light is on or the sense switch is off, the unconditional branch which terminates each string is omitted.

Continue Phase

Reproduces the input to the output area.

MS1013 Convert five digit number to three digit address

EXP10 Convert 3 digit address to 5 digit number

EXP10P Left to right scan stopping on 64 WH

BEGIN Set up table area, starting immediately after longest phase

THS3 Table 1 is 3 times as long as no. of strings (stores add of unsorted strings)

Generated strings (checked by do's)

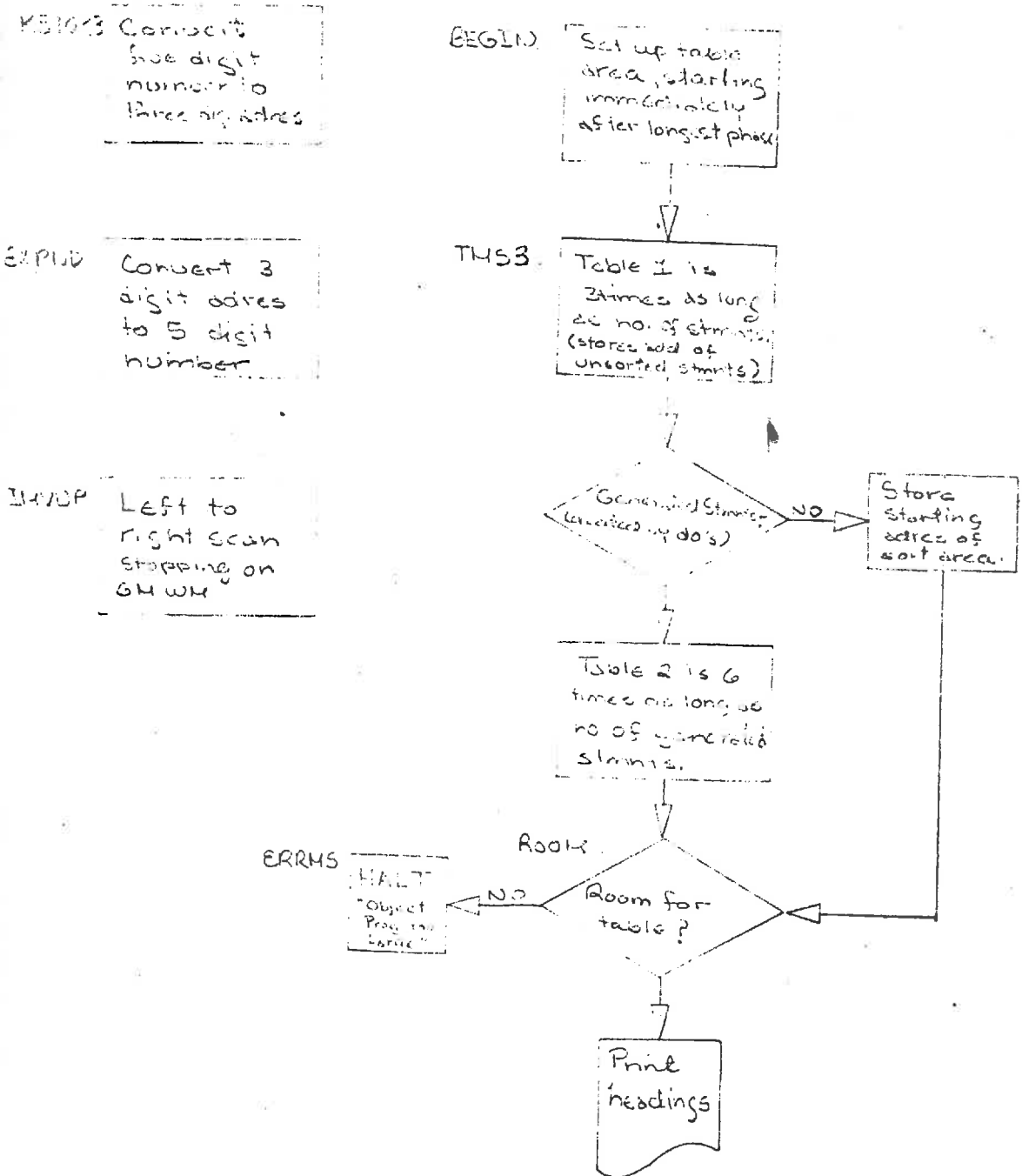
Store starting address of sort area

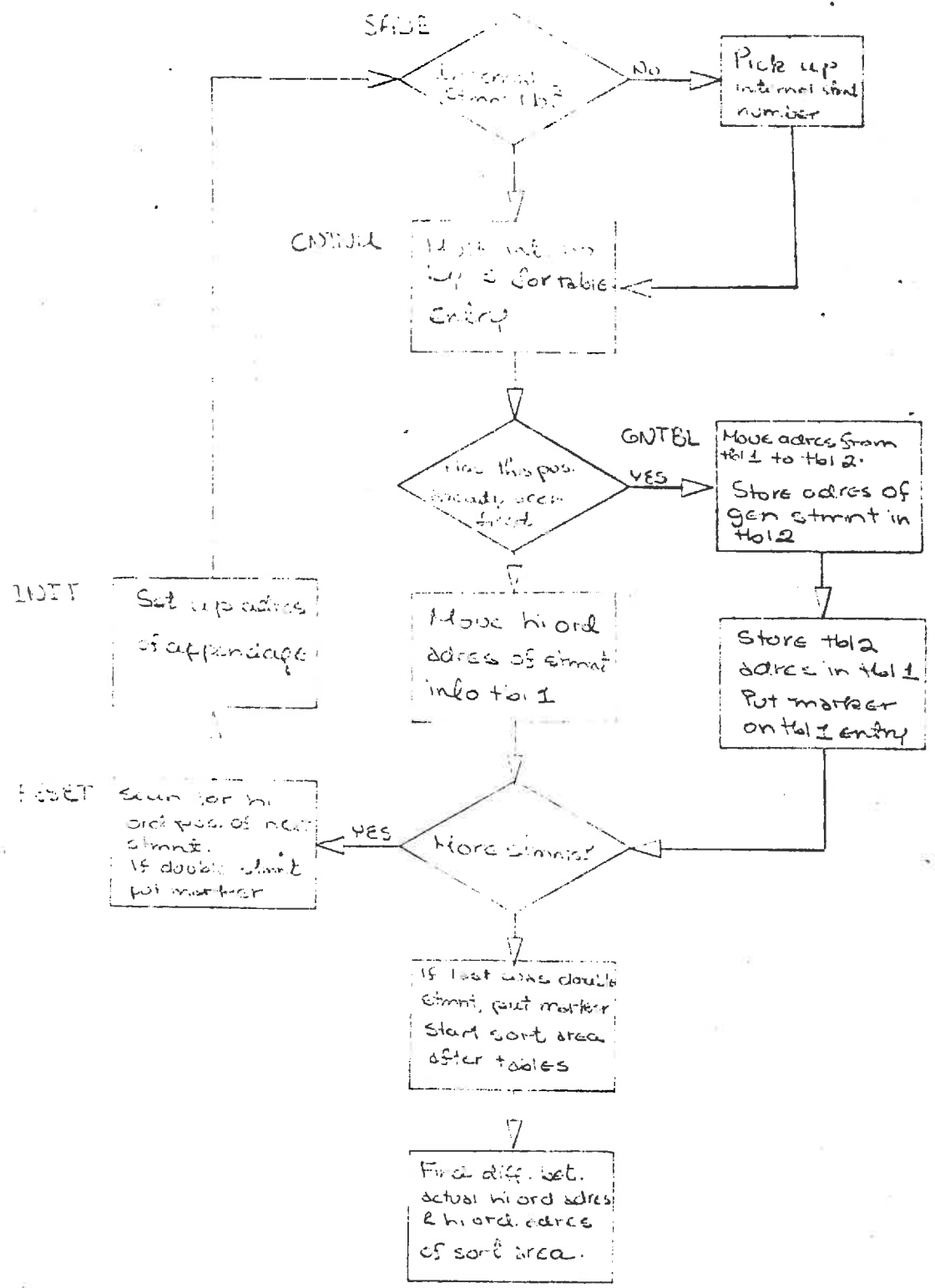
Table 2 is 6 times as long as no. of generated strings

ERRMS HALT "Object Prog. too large"

Room for table?

Print headings

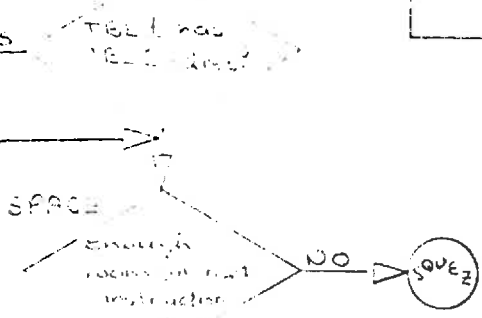
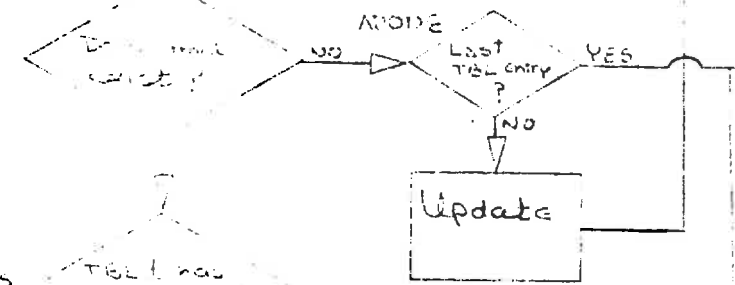




POST-PHASE 3

DATA
 10-175

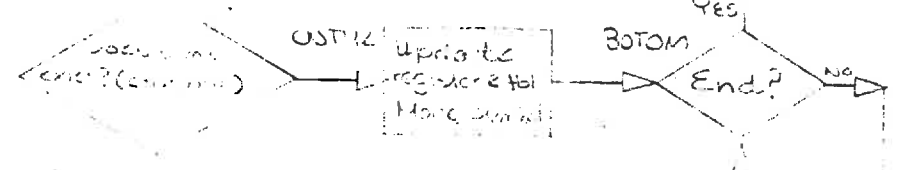
DATA
 Tab if more
 Set next
 = word (10-175)



LIST
 Updates DO
 exit

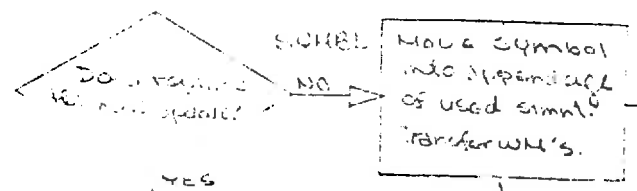
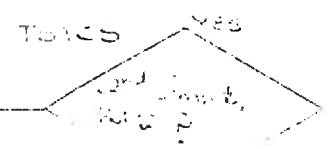
Print
 error no
 & address

LDSYM
 Put in
 error
 counter



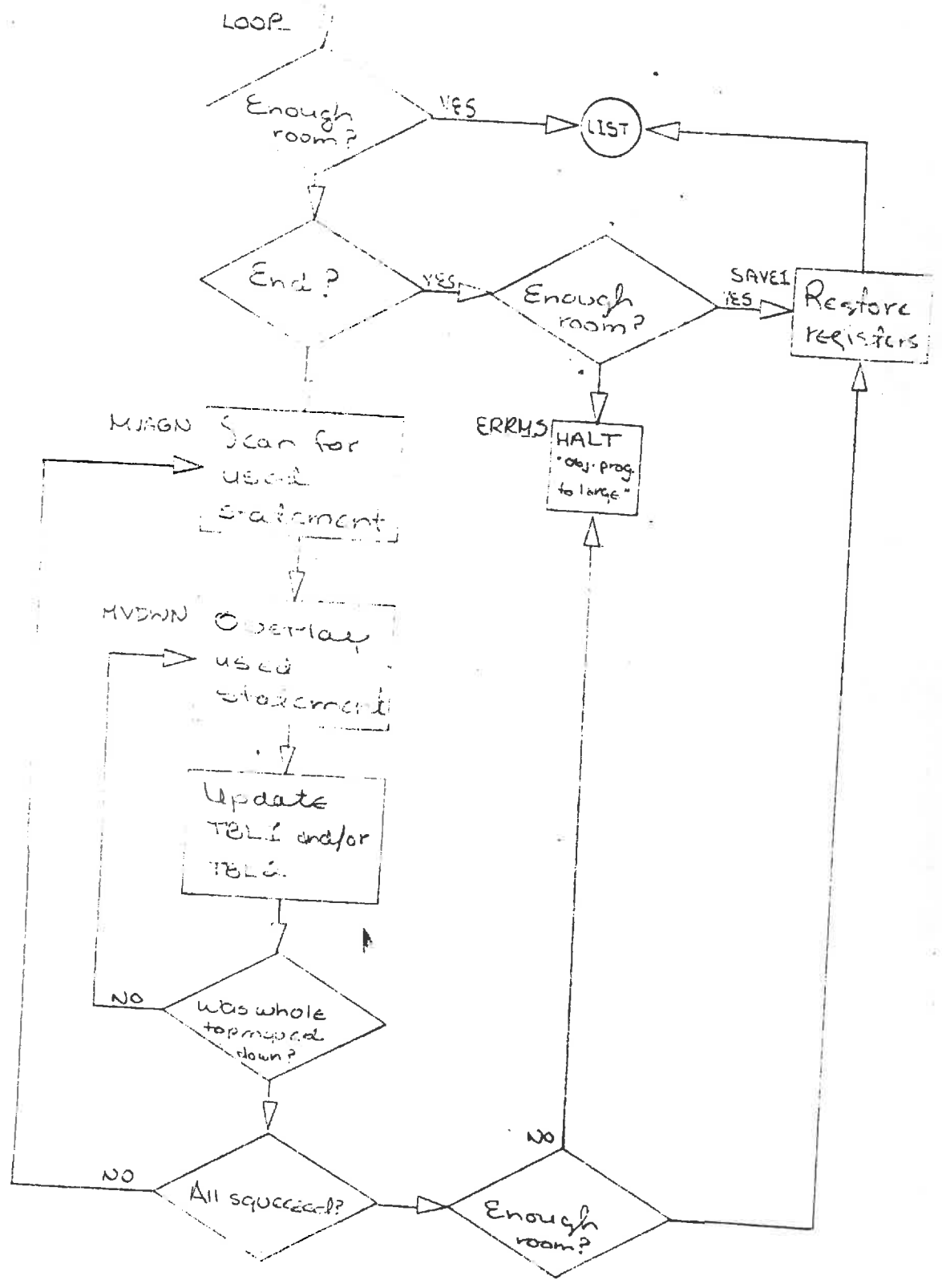
PART
 Pick up
 address from
 table

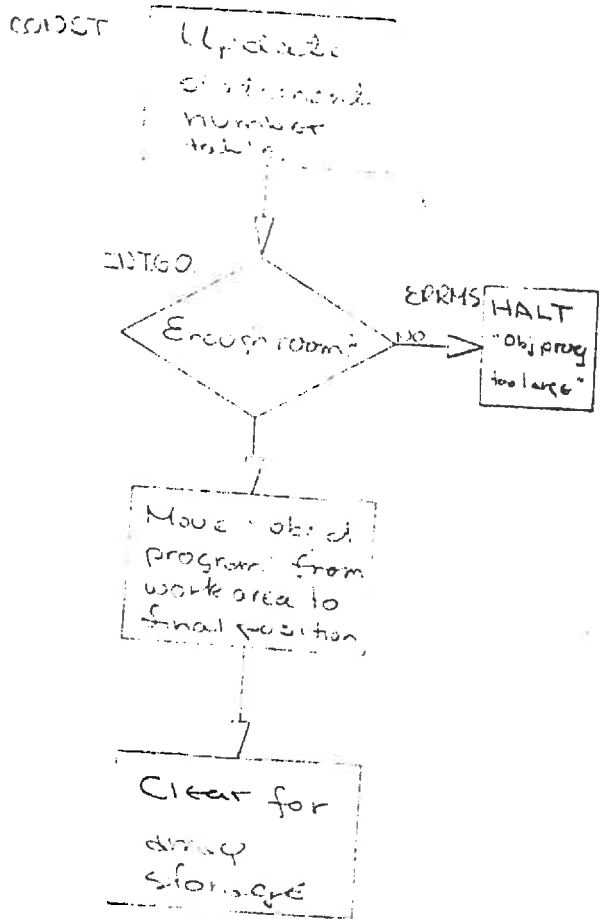
Make
 statement



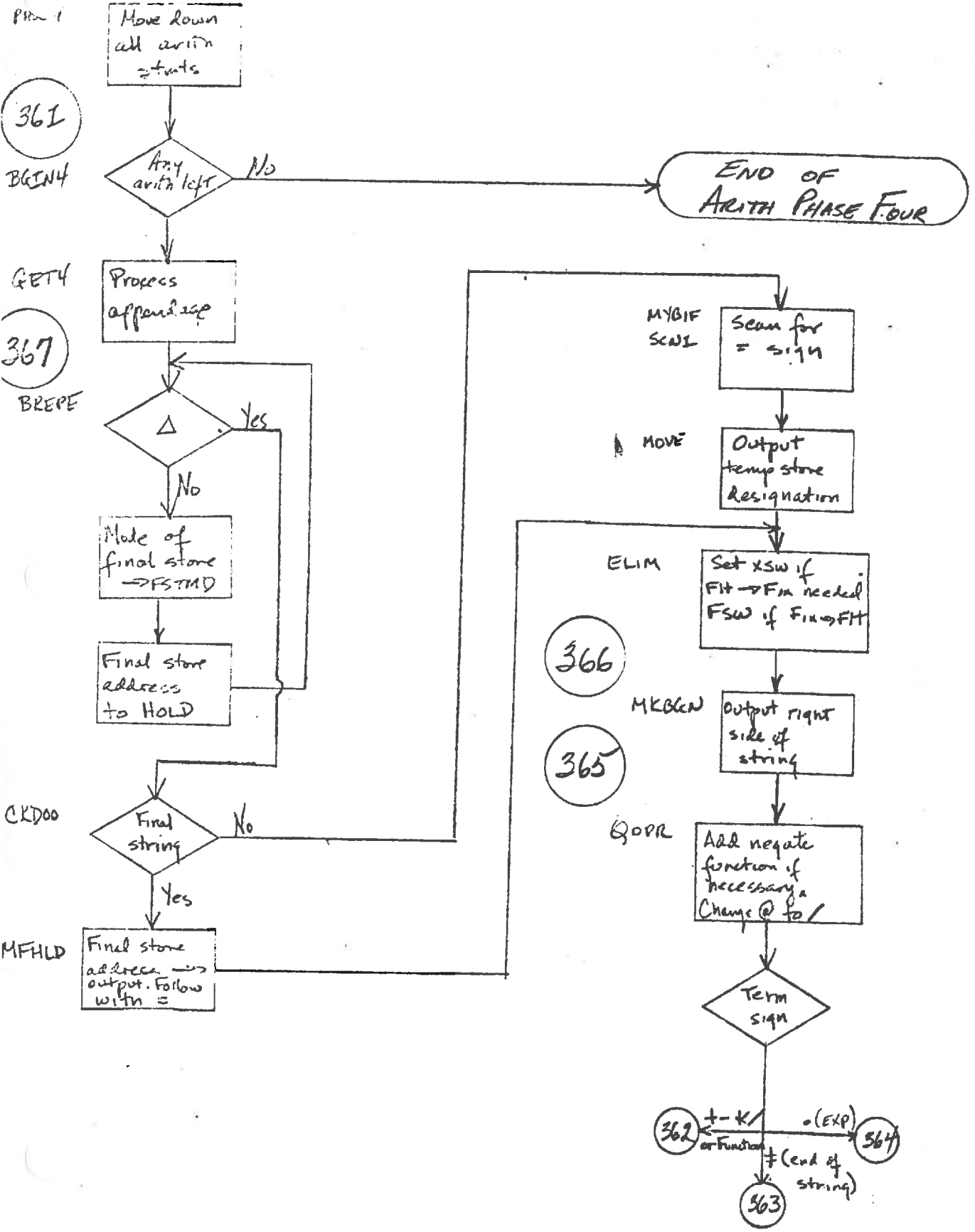
CHANGE
 Update

SQUEZ Eliminate used in order eliminate





ARITH PHASE FOUR



REPLACE PHASE ONE

This phase scans the generated source program twice. The first time it scans for subscript strings. When ^e encountered, it sets word marks in the hundredth position of each parameter and unzones the tens position of all parameters except the first.

The second scan looks at the procedure section for the following characters which have word marks: X, T, † and for operands of instructions which have AB zoning in the tens position.

† word mark

If followed by a character with a word mark (PSKIP) the statement is arith and is bypassed (SKIP).

T word mark

If the statement is a Do, the tens position of all parameters is unzoned and the exit address is generated.

X word mark

These are the instructions which terminate the range of Do statements which are not the innermost Do. The proper branch address is generated.

Operands with AB zoning

These are the "^{*}~~X~~ † nnn" type operands whose correct address is relative to itself. The ^{proper}~~progress~~ address is substituted.

DO PHASE

Each Do statement generates two output statements; the Do statement itself and an unconditional branch which follows the statement that terminates the Do. The Do string generated is

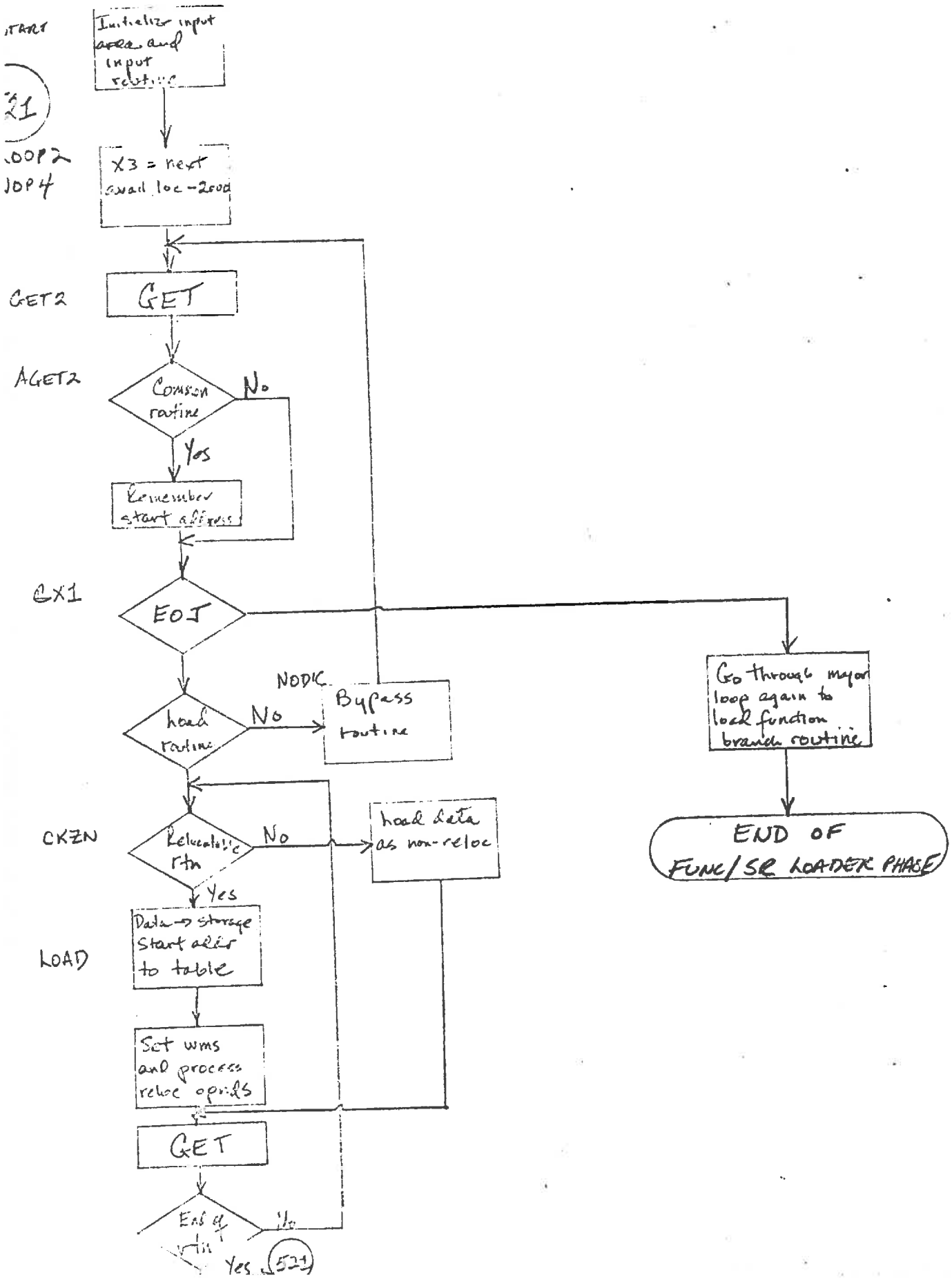
BXXXBYYYAAABBBCCCDDDEEE

where XXX is the address of the relocatable routine which initializes the index; YYY is the address of the relocatable routine which initializes the test for satisfaction of the Do; AAA is the address of M₁; BBB is M₂; CCC is M₃; DDD is the index variable; and EEE is the exit address when the Do is satisfied.

The Do Phase processes these statements backwards, i. e. the last Do first, and analyses the relationship of the Do being processed to the other Do statements. If the Do being processed is ^{the} an innermost ^{of a nest} Do, the unconditional branch generated after the last statement of the Do is a branch to the relocatable routine which tests whether ^{it} ~~the Do~~ is satisfied. Otherwise, the branch generated is to the second unconditional branch of the ~~Do~~ Do string (indicated above).

If more than one Do terminates at the same statement, the exit address EEE for the inner one ~~references~~ the second unconditional branch instruction in the string of the next outermost Do. Otherwise the exit address is to the next executable statement following the range of the Do.

FUNCTION/SUBROUTINE LOADER



- 65 -

REPLACE PHASE TWO

This phase ~~replaces~~ ^{changes} the procedure and format sections for "T" operation codes and for operands with 11-5-8 or 11-6-8 characters in their hundredths positions.

T-op codes

These instructions are those which branch to the relocatable routines. Their A/I operands reference the table generated by the Function/Subroutine Loader Phase. When encountered, the T op code is changed to B and the address of the relocated routine is substituted in the operand.

Special operands

The operands with 11-5-8 and 11-6-8 combinations represent those which reference the two work areas generated in Constants Phase Two equal to the size of fixed word (k) and float word ($f + 2$). The units position of the operand represents character adjustment of these work areas relative to their units position. This phase substitutes the appropriate address in the operands which reference them.

- 66 -

SNAPSHOT PHASE

Through

ARITH PACKAGE

Snapshot Phase

Essentially a copy of the debugging aid, this phase prints out storage starting at the beginning of variable storage when requested and if there are no input errors.

Condensed Deck Phase One

Punches the clear storage and bootstrap cards.

Condensed Deck Phase Two

Punches the cards which

1. Initialize sense lights and index registers
2. The parameter card constants required at object time.
3. The debugging ^{aid} ~~code~~ and arith package.
4. Four cards which initialize the arith package.

Condensed Deck Phase Three

Punches out storage from the first executable statement to the end of storage, bypassing unused storage.

Geaux Phase One

Prints end of job messages.

Geaux Phase Two and Arith Package

Reads in the Arith Package and initializes four operands of this routine.

The printer carriage is then restored displaying the end of job messages.