XV. HIERARCHY SET-UP AND HIERARCHY AND CONCEPT-CONCEPT EXPANSION PROCEDURES

M. Razar and G. Shapiro

1. Introduction

The present section describes a system for setting up a concept hierarchy and for expanding document vectors using this hierarchy and the concept-concept correlations produced by link 9 of SMART.

The hierarchy consists of a tree-like structure where, in general, several related concepts appear under one item of greater generality. Provision is made for having a concept number appear in several places by allowing any concept in the hierarchy to cross-reference any other. Cross-referencing is unidirectional and implies no hierarchical relation. For a description of the considerations involved in constructing a hierarchy, the reader is referred to Sec. III of Report ISR-7.

Instead of being based on a treelike structure in core, the present version of SMART uses a hierarchy stored on tape in list format. Though this format (described in part 2) is highly redundant, it does permit hierarchies of unlimited size to be used. (Compare with the restrictions to less than one core-load effective in Sec. V of ISR-7).

If hierarchy expansion is requested, link 10 of SMART adds the appropriate entries to the document vectors of each document. There are four possible modes of expansion: by parents, brothers, sons and cross-

references, corresponding to weight parameters ROOTWT, BRANWT, LEAFWT and CROSWT. Any or all of these may be requested simultaneously. It was thought to be more efficient to do concept-concept expansion at the same time as hierarchical expansions, since this permits the execution of one pass through the sorted document vector tape for both expansions.

Parts 2 and 3 of this section describe the routines used for generating the hierarchy file on the SMART library tape. Part 4 describes the expansion programs. The production of the concept-concept correlation tape and its format are described in Sec. XIV of this report.

2. Card and Tape Formats; Size Limitations

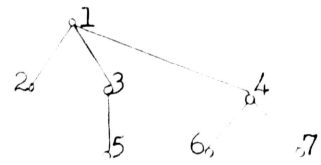
The SETUP program reads information punched on cards in a special format, constructs a tree, and writes some information pertaining to each node on tape. In particular, the following information is provided for each node: its concept number (i.e. node value) and the concept numbers of its parent, sons, brothers and cross-references. The individual nodes are not numbered as such but are referred to solely by the concept numbers assigned to them.

The card format used by SETUP is fairly simple. The first number on a card is the concept number of a node in the tree. All succeeding numbers on the card are cross-references to it. The position of the node in the tree is determined by the position of the first number on the card. The column in which the rightmost digit of this

first number is punched is referred to as the level of the node. The level of a node is compared with the levels of the nodes on preceding cards until one is found with a lower level (i.e. punched further to the left). The first such preceding node is its parent. In preparing a deck of cards to be processed by SETUP, the most esthetic approach is to punch the concept number of a card at the same level as its brothers and at a certain fixed level higher than its parent. This is not necessary, however. As long as the parent of a node is the first preceding node punched to the left of it SETUP will handle it properly. (See Fig. 1.)

The tree (a) can be punched in either of the following equivalent formats (as well as others).

(a)




```
1
  2
  3     5
  4     6
        7
```

or

```
1
        2
    3
          5
  4
          6
          7
```

Sample Card Format

Figure 1

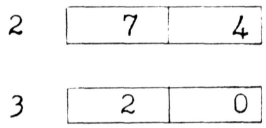
When punching extremely deep hierarchies or nodes with long lists of cross-references, one card per node may not suffice. If desired, a card may be continued to the following one by any punch in column 72. On each successive continuation card the level is

incremented by 70 so that column k on the nth card represents an effective level of 70(n - 1) + k. Column 71 is always left blank. Therefore a concept number and its cross-references should be confined to columns 1-70. A single blank card terminates a bush. Two consecutive blanks terminate the whole tree and prevent any further data from being read in. To insure proper functioning, all concept numbers should be at a higher level than the root of the bush. Roots have no parent. SETUP identifies roots by giving them the fictitious parent 0.

SETUP is designed with two criteria in mind. First, the relevant information concerning each node (i.e. parent, sons, brothers, cross-references) should be readily accessible by means of a fast and efficient search. Second, the hierarchy in the tree structure should be easy to update. Since each node is completely described by a single card (with, perhaps, some continuation cards), insertions and deletions can be made, without massive changes. To insert a node one merely adds a card in the right place. To delete a node, one removes the card and all its descendants.

To meet the requirement of accessibility, SETUP writes information on tape sorted in order by increasing concept numbers. The actual format used in preparing the tape is as follows. Each word on tape contains two numbers packed into it, the first in bits 1-17, and the second in bits 19-35. The first word of each logical record contains the concept number in its first half and the parent in its second half. The second and third words of each logical record

give information as to the number and nature of the succeeding words in the record. The second word has, in its first half, the total number of words which will contain brothers, sons and cross-references; in its second half it contains the number of words of brothers and sons. The third word contains the number of words containing brothers; its second half is filled out with zeros. Thus if there are five brothers, four sons and three cross-references we will have three words of brothers and two words each of sons and cross-references. (See Fig. 2.)



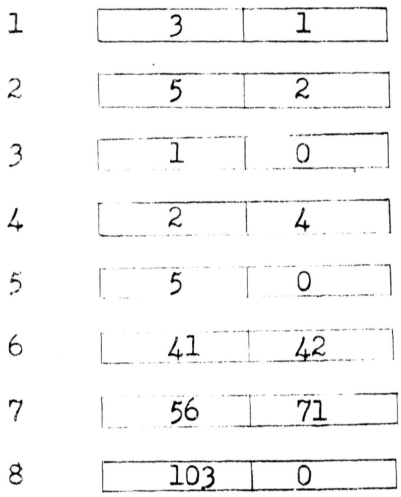
This refers to the case when there are five brothers, four sons and three cross-references.

Sample of Words 2 and 3

Figure 2

The remainder of each logical record (words 4 and after) consists of the lists of brothers, sons and cross-references packed two to a word. If n1, n2, n3 respectively represent the numbers packed into words 2 and 3, then the next n3 words (4 to n3 + 3) contain the brothers; the following n2 - n3 words (n3+ 4 to n2+ 3) the sons; the next n1 - n2 words (n2+ 4 to n1+ 3) the cross-references.

As an example, suppose that the node labeled "3" in Fig. 1 had cross-references to 41, 42, 56, 71, 103. Then it would produce the logical record indicated in Fig. 3 below.



Sample Logical Record
Figure 3

Two facts about this format should be mentioned. First, any extra half-words are simply filled out with zeros. Second, the order of the lists of brothers, sons and cross-references is determined by the order of the data cards. As no attempt is made to sort these lists, they may appear in any permutation (e.g., a list b1, b2, b3 of brothers might appear in the order b2, b1, b3).

Finally, although each logical record consists of n1+ 3 words, it is more efficient to have somewhat larger physical records on tape. A physical record will contain up to 1000 words. As many logical records as can fit within this limit will be contained in each physical record.

A word should be said about the size limitations of SETUP. No more than ten cross-references, twenty sons and twenty brothers can be

accommodated. Moreover, as each logical record is limited to 25 words, only 22 words can be used for these lists. Hence the maximum total number of sons, brothers, and cross-references must not exceed this limit. If the total number of references never exceeds 42, no trouble can arise.

Also, bushes of more than 100 nodes may cause difficulties (in practice, much larger bushes usually can be handled adequately). If more space is desired the first dimension of each of the following should be changed: IND(100), NCØN(100), LCRØ(100), NCRØ(100,10), NSØN(100,20). Neither of these limitations bothers SMART as none of the trees presently used is complex enough to require any additional space.

3. Description of Algorithms Used by SETUP

Most of the work of SETUP is done by a subroutine TRESET, which is designed to determine all the required information about the tree structure and write it on tape without sorting it or compressing it. TRESET reads the data cards with a subroutine NEWCRD which is described below.

NEWCRD has five arguments: NCØNC, LEVEL, LCRØS, NCRØS, LTEST. These are all used to return information to TRESET; they are never used to give NEWCRD information. The first argument gives the concept number and the second its level. The third and fourth arguments are the length of the cross-reference list and the list

itself. The fifth argument is a flag indicating whether or not the card just read was blank.

To determine LEVEL, NEWCRD reads a card (with A-format), character by character, into a vector CARD of dimension 72. Then it looks for nonblank characters in columns 1-70. If it finds one, it continues to look until it finds the next blank column (say, column k). Then LEVEL is set at k- 1 to indicate the position of the last digit of the concept number. If CARD (1-70) is blank, NEWCRD checks column 72. If it too is blank, then the card is considered blank and NEWCRD returns with LTEST = 0. If column 72 is not blank, it goes to the next card, adding 70(n- 1) to the level (when it is on the nth card for this node).

On the nonblank cards, once NEWCRD determines LEVEL, it checks column 72 for continuations and sets a flag to record the fact. Then it calls a subroutine REREAD which causes the current card to be read again. The numbers on the card are then read into a vector NUMB (by means of SAO G-format). NUMB(1) becomes NCØNC and the vector NCRØS of cross-references is filled from the remaining nonzero entries in NUMB. A count is kept in LCRØS. If continuations are needed, subsequent cards are read immediately into NUMB and then transferred into NCRØS. After reading all the continuation cards pertaining to the node, LTEST is set equal to 1 and NEWCRD returns.

The operation of TRESET is centered around two push-down stores. One contains the concept number read in before the one that just came back from NEWCRD. It is referred to as the current-node pushdown store or P.D.S. 1. The other contains the concept number of the last

node to qualify as a parent. It is called the current parent push-down store or simply P.D.S.2. The algorithm used by TRESET can then be outlined as follows:

(1) Read in a new card via NEWCRD. If blank, set LEVEL = 0.

(2) Is it a son of the last node before it (i.e. the top of P.D.S.1)?

  (a) If so, push down P.D.S.2 and add the top of P.D.S.1 to it. Then push down P.D.S.1 and put the new node on top. Then go back to (1).

  (b) If not go on to (3).

(3) Is it a son of the current parent (i.e. the top of P.D.S.2)?

  (a) If so, push down P.D.S.1 and put the new node on top. Go back to (1).

  (b) If not, we have the total set of sons of the current parent. Go on to (4) for processing.

(4) Set up a list of sons for the current parent. Choose nodes from this list one at a time.

(5) Process each in turn, taking the remainder of the list as its brothers. Namely, pack into a buffer the lengths of the brother, son and cross-reference lists, together with the concept number and parent of the chosen node and the actual lists. Every time the buffer is full, write it onto tape and, after clearing it, continue.

(6) When the whole list of sons has been processed, determine the status of the pointer to P.D.S.2.

(a) If it has not hit the bottom of P.D.S.2, pop up one level (i.e. lower the pointer) and return to step (3).

(b) If we are at the bottom (i.e. no parents are left in P.D.S.2), the root of the bush must be processed. To do so, its parent is set equal to zero, and after lowering the pointer beneath the bottom of P.D.S.2, one must go to step (5). (This will only occur if a blank card was read in.)

(c) If we have gone through the bottom (i.e. if the root of the bush has been processed), another card is read. If it is blank (the second consecutive one!) the buffer is written on tape and one returns to the program. If not, return is made to (1) to process a new bush.

TRESET writes tapes in the following way: 250 words make up a physical record. Each physical record is subdivided into ten 25-word logical records. Each logical record contains the information as described in part 2 of this section and is then filled out with zeros.

After calling TRESET, SETUP calls a sorting routine SPECTR, which breaks up the physical records into their logical components and sorts them in ascending order according to the value of NCØNC (which is in the first half of the first word of the logical record).

A short subroutine, called SQUASH, then removes the extra zeros at the end of each record (hence, the name) and packs them into physical records of under 1000 words. This is the final output of SETUP.

The TRESET and NEWCRD programs are described in Flowcharts 1 and 2, respectively.

4. The Expansion Programs

The main program of link 10, XPAND, performs the hierarchy and concept-concept expansions (either or both depending on the parameters CONCON and HIER). The expansion part of the program assumes that the document vector tape, the concept-concept correlation tape, and the hierarchy file of the library tape are in order, sorted by concept numbers. This is always true of the last two of the tapes mentioned. Thus expansion is essentially a two or three-way merge along with some format juggling.

The document vector tape may not have been sorted into concept order before link 10 is called. If CONCON is on, then link 9 has to sort the document vector tape in order to calculate the correlations (cf. Sec. XIV). In this case, the tape and its number are left for XPAND, unless only requests are to be expanded. In this last case the sorted tape produced by link 9 contains document vectors as well as request vectors, and thus includes much extraneous information. Therefore when this option is active, XPAND reads the request (and "LIKE") files of tape B1, and sorts them internally using the routine SORTDC; this can be done since all the requests must fit into core. This sorted information is then written out onto a tape which is used as input for the expansions proper. If only documents are to

be expanded, the requests are saved on the end of tape A4, since it would be quite time-consuming to redo the sort already done by link 9. Note that this permits tape B1 to be used as a scratch tape if necessary.

If only HIER is on, B1 will not have been sorted. In this case, XPAND used the trap-controlled tape sort, SPECTR, to sort the appropriate files of B1 (unless, again, only requests are to be expanded so that SORTDC may be used). Perhaps the following table clarifies these considerations:

To be Expanded / Method of Expansion	REQS	DØCS	ALL
CØNCØN	X B1 (reqs,docs)	X Y (docs) A4 (reqs)	X Y (reqs,docs)
HIER	B1 (reqs,docs)	B1 (reqs,docs)	B1 (reqs,docs)
BØTH	X B1 (reqs,docs)	X Y (docs) A4 (reqs)	X Y (reqs,docs)

The tapes in the boxes are those left by the previous link (except for the library tape which is not mentioned as it is always available). X stands for the concept-concept correlation tape and Y for the concept-ordered vector tape.

Before proceeding to the expansion proper, XPAND must also figure out what tape is to be used as (temporary) output tape. Because of the large number of possible combination of options, and because it is impossible to determine on which tape SPECTR (either in link 9 or 10) will leave its output, this is not a simple task. XPAND in fact attempts, during expansion, to perform the difficult feat of keeping 3 or 4 tapes (on two channels) in operation simultaneously. To facilitate what overlap is possible, XPAND picks the output tape in such a way that the tapes are divided as evenly as possible between the two channels, either two on each, or two on one and one on the other.

The following discussion assumes that both the HIER and CONCON indicators are on; if one is off, the procedure is the same except that the part relevant to the missing option is skipped. The basic procedure is to read a half-buffer load of input (from the sorted document vector tape) and then to read in as much of the hierarchy followed by the concept-concept tape as is needed to expand this buffer load. Output is accumulated in a buffer, supervised by subroutine LAUGH, and is written out each time a half-buffer is filled. All input-output is handled by the trap-controlled routine INOUT (through several interfaces). Each of the four tape operations involved uses two half-buffers, and before switching half-buffers and starting a new transmission, the IOEND entry to INOUT is called to insure that the previous transmission on this particular unit has been completed. In this way, as much overlap as is possible is effected with a minimum of bookkeeping on XPAND's part.

The item size on the output tape is two words. 500 items are packed to a physical record. The format of the item is:

S12 3 17 18 - 20 21 35

Con No.	T	WEIGHT
Doc No.		FLAG

where Con. No. is the 18 bit concept number, Doc. No. is the document number, WEIGHT is the weight of this concept in document Doc. No., and FLAG is a bit which is on if and only if this item is a document identifier. In this case, the rest of the entry is BCD and is merely copied onto the output tape for later use in identification.

During expansion, the weight of an output item is set equal to the weight of the input concept number which it is expanding. On expansion

T = 1 for a CONCON expansion
T = 2 " " parent "
T = 3 " " brother "
T = 4 " " son "
T = 5 " " cross-reference "

The input concept number is always copied onto the output tape (with a tag of zero), except if the HIER option calls for replacement of the items in the original vector by the expanded items, and CONCON is off. These tags are used by programs in link 8, when the program is called again so that it can weight each different type of item correctly according to the values of

(in order of tags) COCOWT, ROOTWT, BRANWT, LEAFWT, and CROSWT.

XPAND, at the start of the expansion code, initializes all relevant input-output and the following pointers: INQ which points to the next location in input half-buffer number INB which is to be used; IHQ which points to the next location in the hierarchy half-buffer number IHB which is to be used; ICQ which points to the next location in the concept-concept half-buffer number ICB which is to be used; and IOQ which points to the next location in the output half-buffer number IOB which is to be used. All the counts start at their maximum value and count down since FORTRAN arrays are ordered inversely with respect to tape order.

Control then passes to the hierarchy expansion section. The current element of the input buffer is compared to the current element of the hierarchy buffer. If the former is greater, subroutine HIERN is called. On return, IHQ and IHB point to the appropriate next item of the hierarchy buffer. HIERN steps IHQ (by the amount required by the pointer built into the hierarchy entry, as described in part 2) and checks to see if the buffer has been exhausted; if it has, it calls IOEND, switches IHB and resets IHQ, and starts another read.

If the two concepts match, each of the four possible hierarchy expansion option weights is examined in turn, and for each one that is nonzero, the appropriate entries are written onto the output tape. The pointer to the input buffer is then advanced as described in the next paragraph.

If the input concept number is less than the hierarchy concept number, INQ is stepped (down) and a test is made to see whether the input buffer is finished. If not, the comparison mentioned above is repeated.

When the input buffer has been exhausted by the hierarchy expansion section of code, INQ is reset and a very similar process is performed for the CONCON expansion. In this case, the operations are somewhat simplified since there is but one type of expansion to be performed, and since pointers don't have to be unpacked as they do in the hierarchy case.

When both hierarchy and concept-concept expansion have been performed for one input half-buffer, INB is switched, the next input buffer is read, and processing continues.

If either the hierarchy or concept-concept tape is exhausted before the input tape is exhausted, switches are set so that the relevant sections of code are no longer executed. At any rate, any excess on the input tape is copied onto the output tape (unless the option is HIER replace only).

After the output tape has been completely written, it must be sorted back into order according first to document number and then concept number. This sort sends the document identifier to the end of the expanded document vector, making it possible for MERVEC, in link 8, to identify the document properly. Again, some care must be taken in choosing the scratch tapes for the sort. The output tape number of the sort is left in a COMMON location for MERVEC.

The main subroutines called by XPAND are:

(1) INOUT routines through interfaces XNIBIN, XOBIN, WAIT2 and WAIT3.

(2) LAUGH(A,B) which enters A and B into the output buffer and controls output tape operation.

(3) HIERN which sets pointer IHQ and IHB and controls the hierarchy tape operations.

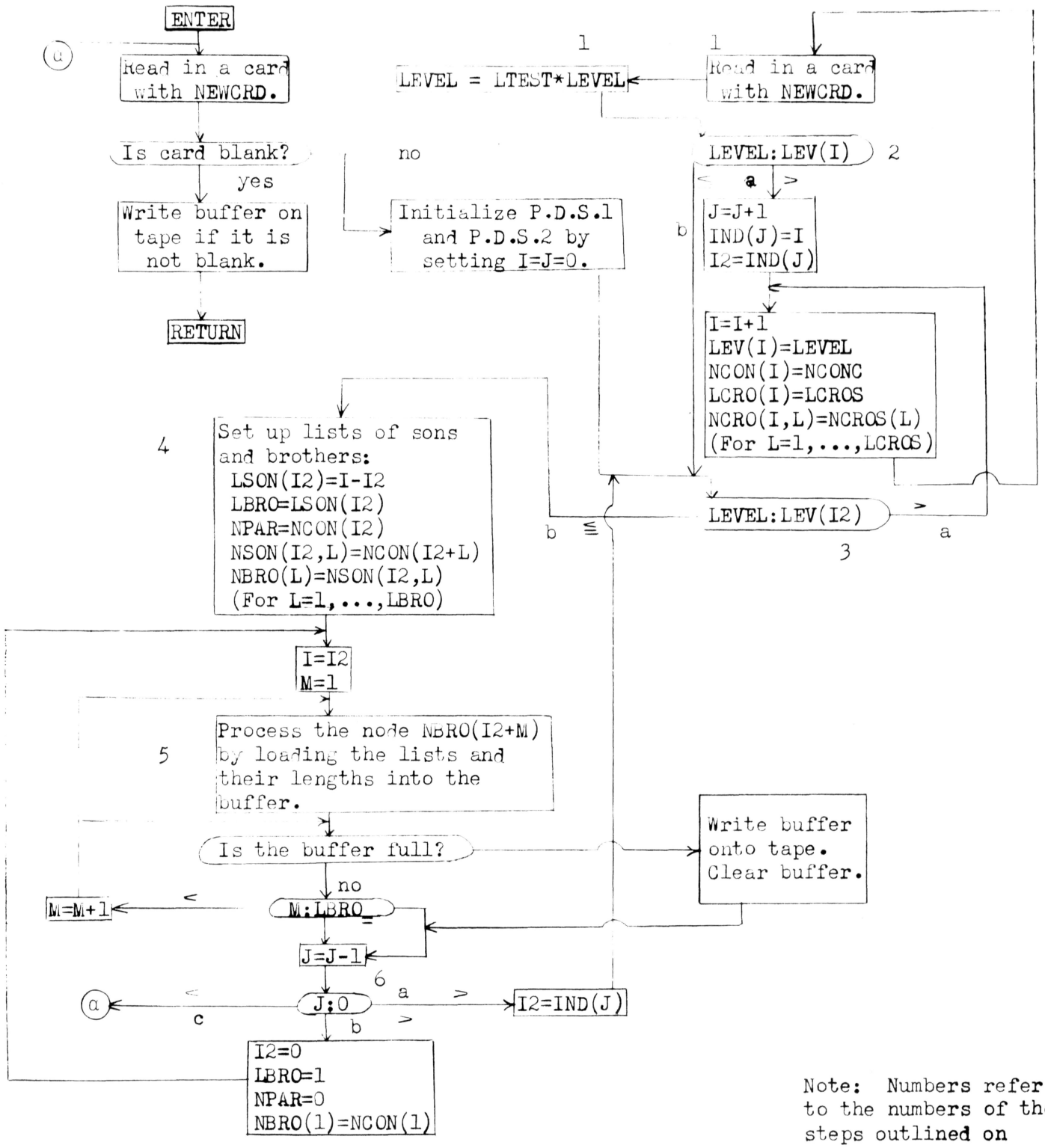
(4) CONIN which sets pointers ICQ and ICB and controls concept-concept tape operation.

(5) HUP(I,N,T) which writes onto the output tape (using LAUGH) the expansion of the current item in the input buffers by items in the hierarchy buffer, starting at index I and proceeding through N packed words, giving each output item a tag of T. This routine is used for all hierarchy expansions, except parent expansion, since the latter, in general, require a list, rather than a single item, to be added to the output tape.

Flowchart III describes XPAND. IEXPAN = 1, 2, or 3 according as requests, documents or all must be expanded. ITP is the input tape number for the expansion proper. IY is the sorted tape number left by link 9. IX is the CONCON tape number. IB5 = 10 is the number of the library tape. INP, dimensioned 2 X 500 X 2 is the input buffer; the last subscript determines which buffer half is being used.

NHIE is set to IHIER, the parameter which is 0,1,2 according as HIER is off, expand or replace. Note that, if both HIER and CONCON are on and NHIE = 1, the HIER section puts the input concept number out on tape (but CONCON doesn't since it ought not to be written out twice). If the hierarchy is exhausted before the concept-concept tape is terminated, NHIE is set to zero so that CONCON will be forced to write out these concept numbers. LCON, similarly, is set to CONCON originally. If the CONCON tape is finished before the hierarchy tape, LCON is set to zero and NHIE to 1 so that the hierarchy section will continue writing out the input concept numbers. L1 and L2 are switches corresponding to FORTRAN assigned GO's. IHCNT, ICCNT and ICCNT are set negative if and only if an end of file is read on the hierarchy, concept-concept, or input tapes, respectively. INCNT negative will, therefore terminate the expansion.

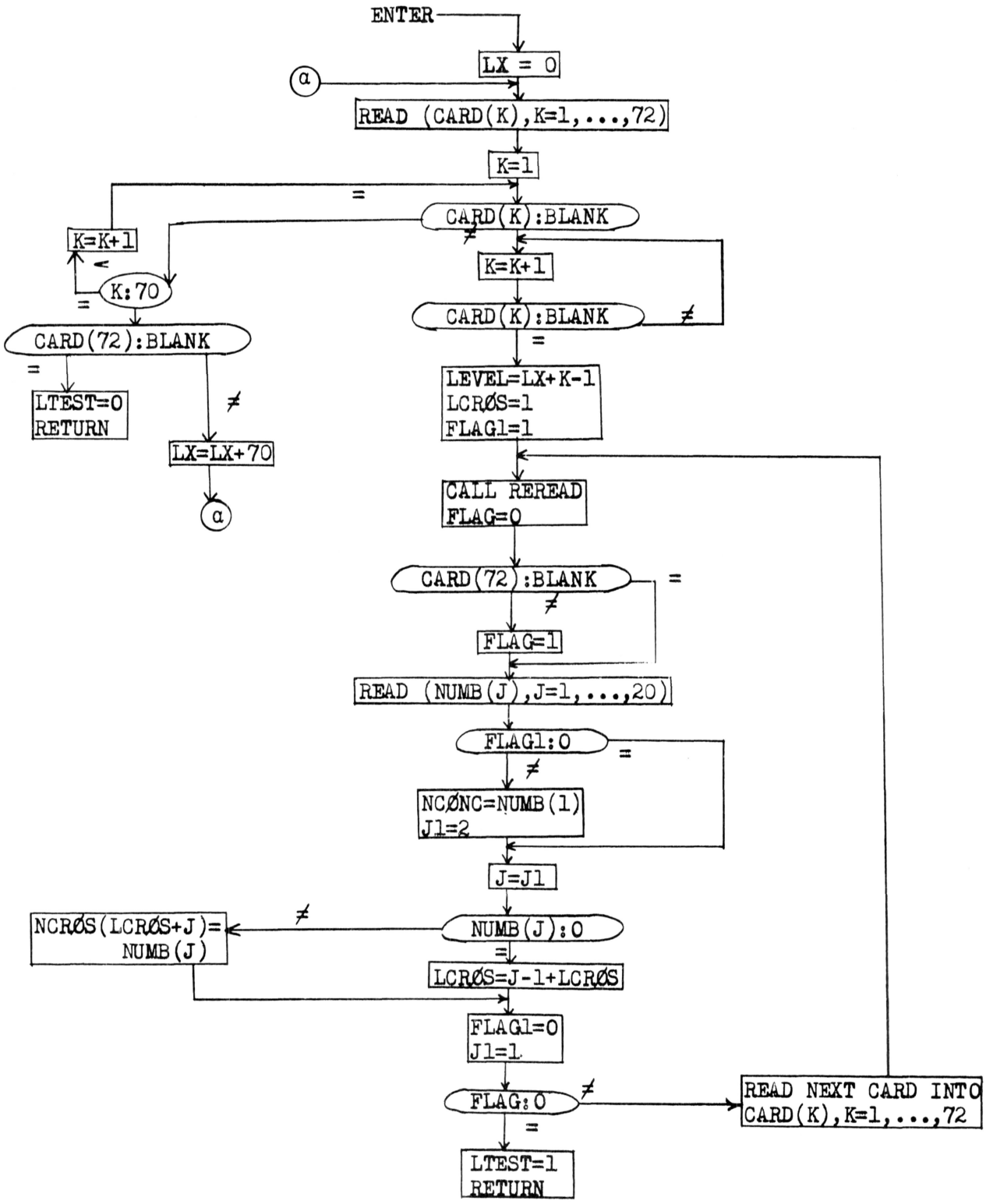
Note also that XISTF(N) masks out the low order IS bits of the argument.



Note: Numbers refer to the numbers of the steps outlined on page 9.

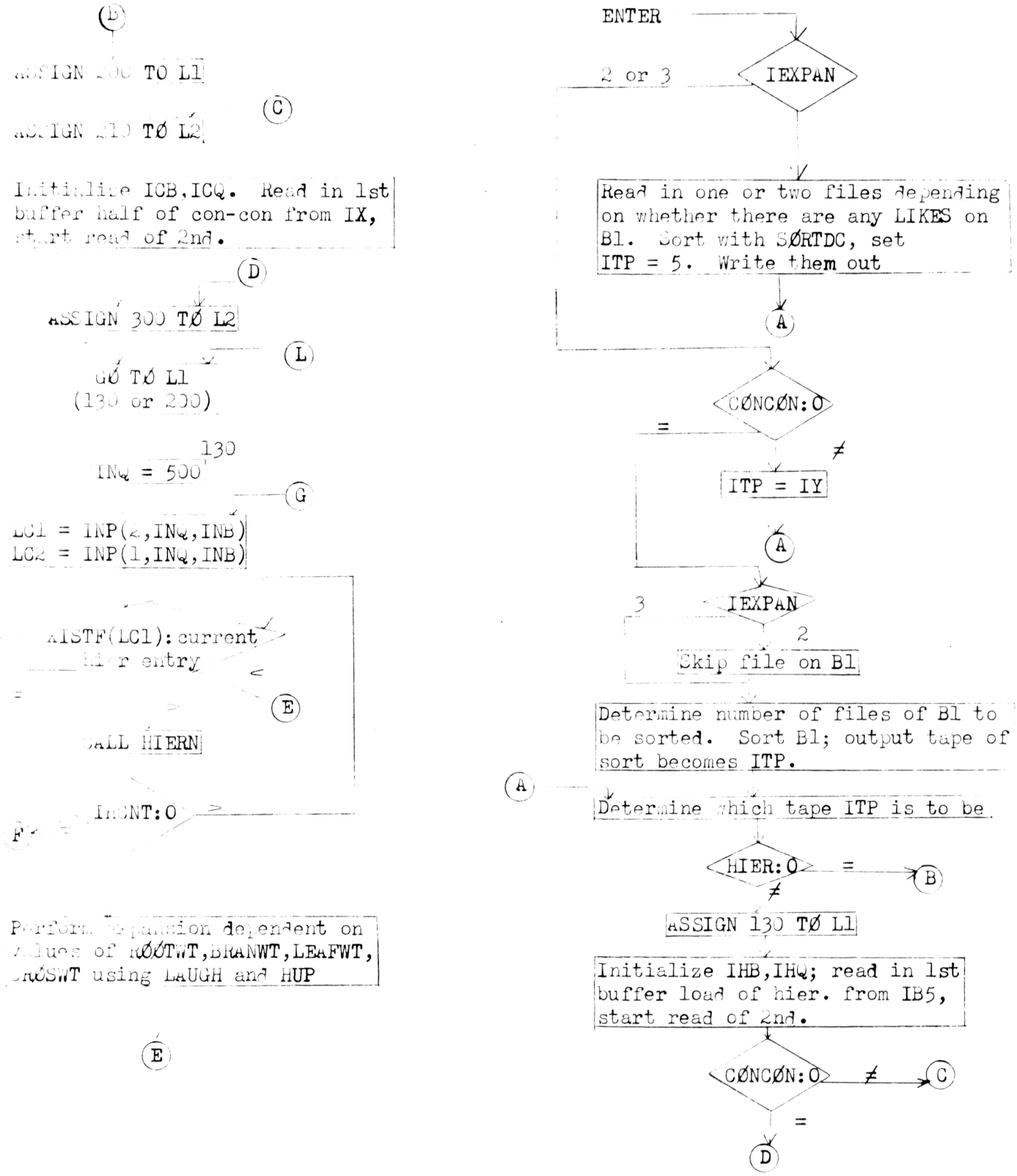
Program for TRESET

Flowchart 1

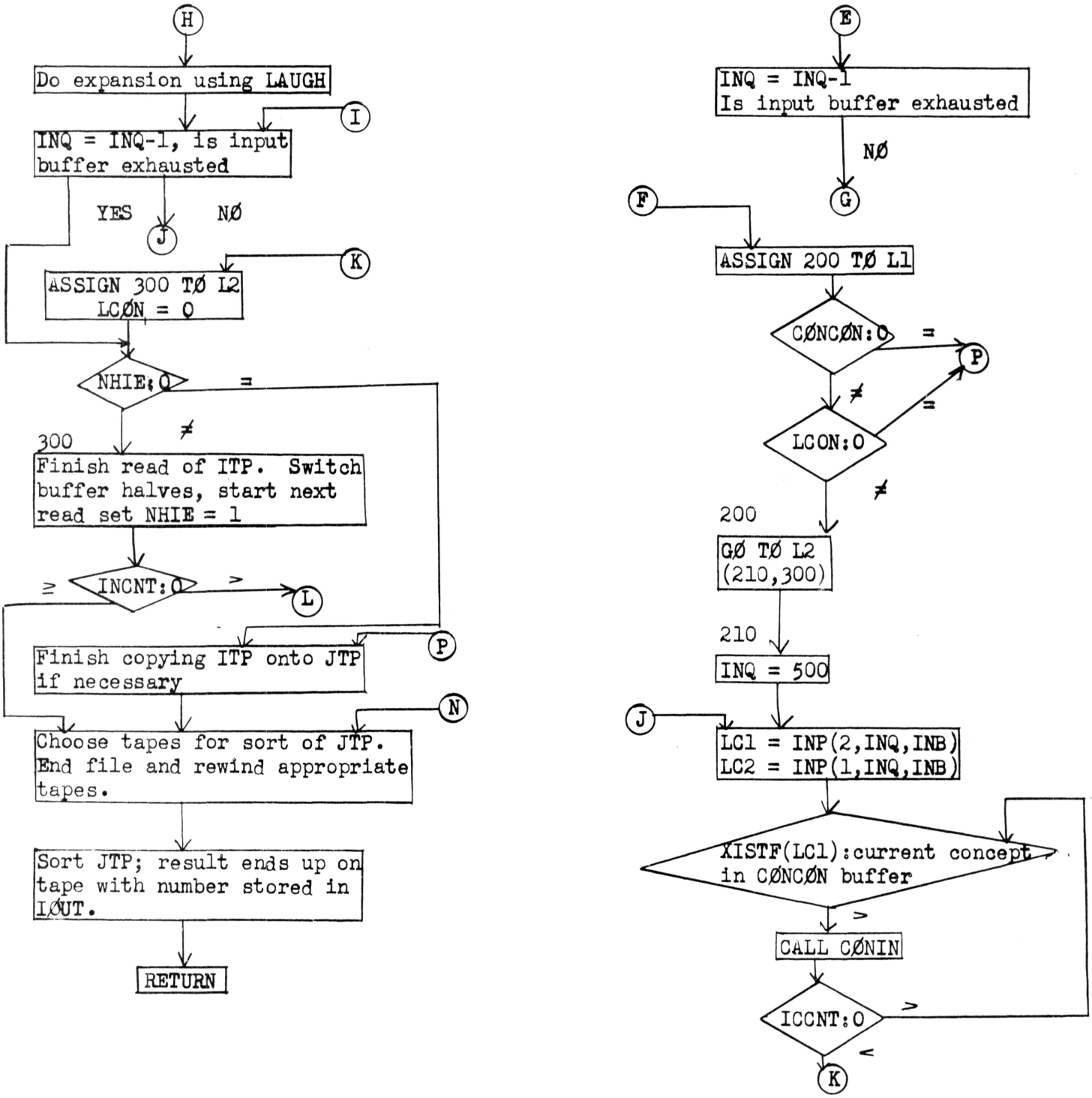


Program for NEWCRD

Flowchart 2



XPAND
Flowchart 3



Flowchart 3 (continued)