LOGO for the Xerox Sigma 7

Internal Specifications and System Maintenance Manual

James Hobbs, Nick Int Veldt, Robert Martin,
Linda Milburn, Donna Nagel, David Walker

TECHNICAL REPORT #34 - JUNE 1973

Department of Information and Computer Science
University of California, Irvine

ABSTRACT

THIS IS THE INTERNAL SPECIFICATIONS AND SYSTEM MAINTENANCE MANUAL FOR LOGO ON THE SIGMA 7 AT THE UNIVERSITY OF CALIFORNIA IRVINE.

I. DATA STRUCTURES AS PREVIOUSLY DEFINED IN THIS DOCUMENT.

II. STORAGE DIVISIONS

A. MAIN SECTIONS

1. SYMBOL TABLE
2. R STACK
3. S STACK
4. P STACK
5. STRING POINTER TABLE STORAGE
6. STRING STORAGE

B. STORAGE AREA

1. EACH SECTION IS ALLOTED A SET PERCENTAGE OF AVAILABLE STORAGE.

2. PRELIMINARY FRACTIONS

1). SYMBOL TABLE	1/16
2). R STACK	1/16
3). S STACK	3/16
4). P STACK	3/16
5). STRING POINTER TABLE STORAGE	3/16
6). STRING STORAGE	5/16

TABLE OF CONTENTS

ACKNOWLEDGMENTS

THIS SYSTEM WAS PRODUCED BY A GROUP OF SIX PEOPLE IN PARTIAL FULFILLMENT OF THE COURSE ICS 190 AT THE UNIVERSITY OF CALIFORNIA, IRVINE, WINTER AND SPRING 1973. THE GROUP WAS UNDER THE DIRECTION OF ROBERT J. BOBROW AND COMPUTER TIME WAS PROVIDED BY THE DEPARTMENT OF INFORMATION AND COMPUTER SCIENCE.

THE SIX PEOPLE AND THEIR DUTIES WERE AS FOLLOWS:

JAMES P. HOBBS -- INTERFACE WITH THE OPERATING SYSTEM

NICK INT VELDT -- PARSER AND COMMAND ROUTINES

ROBERT MARTIN -- EXECUTER, ARITHMETIC, SOME USER OPERATIONS

LINDA MILBURN -- DOCUMENTATION

DONNA NAGEL -- SOME OF THE USER OPERATIONS

DAVID WALKER -- STORAGE MANAGEMENT

INTRODUCTION

IN ORDER TO UNIVERSALIZE THE FORMATTING OF OUR DOCUMENTATION, THE FOLLOWING FORMAT WAS SUGGESTED FOR EACH MODULE:

I. EXTERNAL SPECIFICATIONS

   A. BRIEF FUNCTIONAL DESCRIPTION

   B. DATA STRUCTURES AND FLOWS

   C. LIST OF ENTRY POINTS AND FUNCTIONS

   D. LIST OF ROUTINES CALLED

II. INTERNAL SPECIFICATIONS

   A. INTERMODULE COMMUNICATION

   B. INTERNAL DATA STRUCTURES

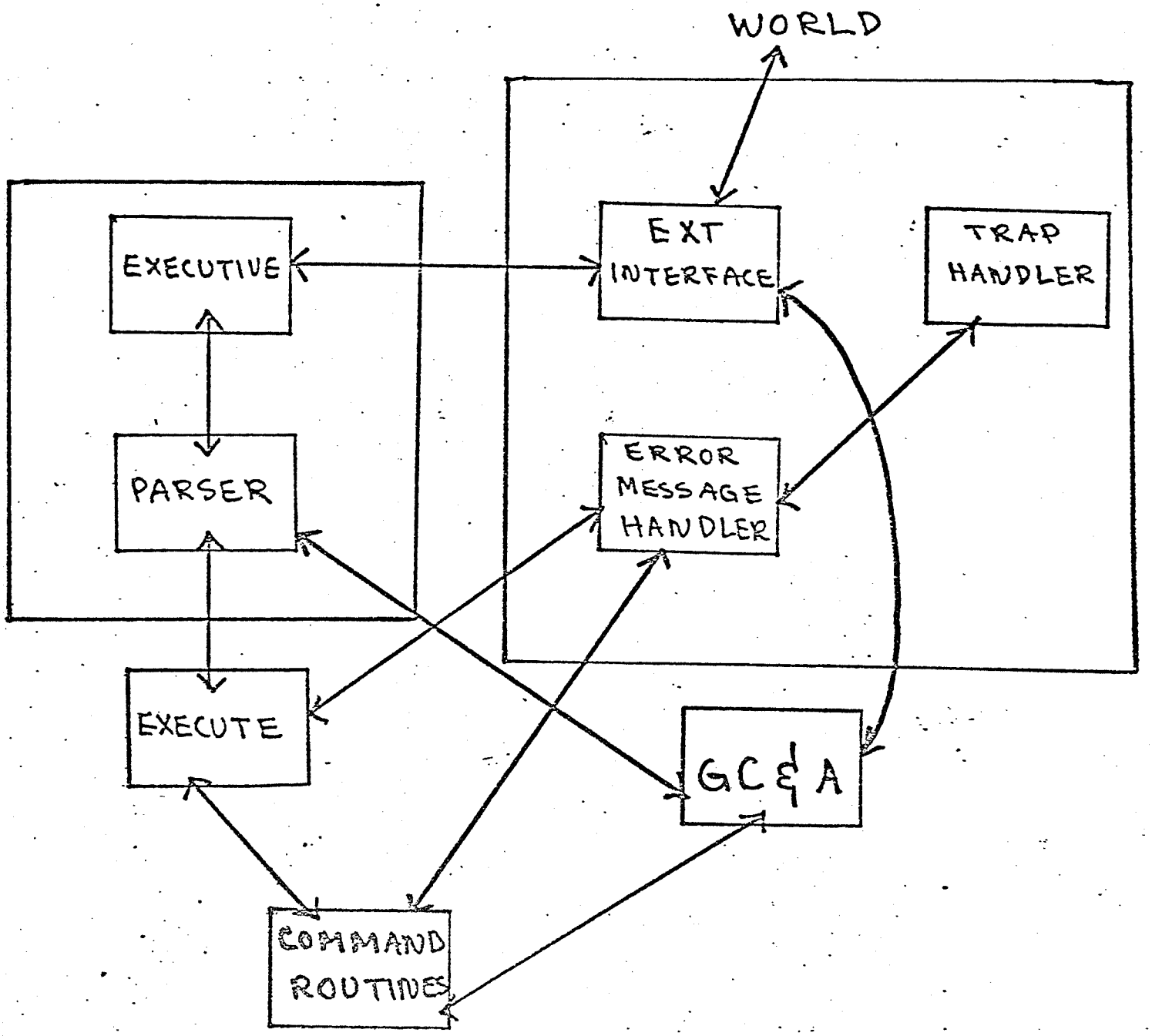
   C. INTERNAL SUBROUTINE LIST

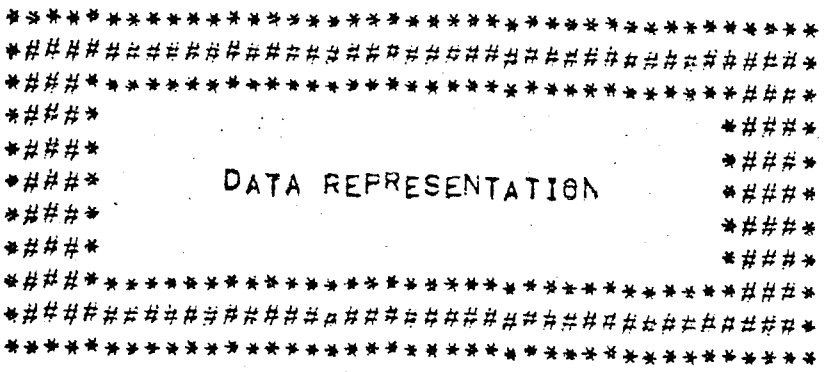
   D. DESCRIPTION OF EVERY INTERNAL MODULE

      1. PROSE DESCRIPTION

      2. FLOWCHART

THIS FORMAT WAS FOLLOWED WHERE FEASIBLE. IN EACH OF THE FOLLOWING MODULE SECTIONS THERE ARE EXTERNAL SPECIFICATIONS FOLLOWED BY INTERNAL SPECIFICATIONS. THE DATA STRUCTURES ARE PRESENTED BEFORE THE MODULE DESCRIPTIONS.


WORLD
EXECUTIVE
EXT INTERFACE
TRAP HANDLER
PARSER
ERROR MESSAGE HANDLER
EXECUTE
GC & A
COMMAND ROUTINES

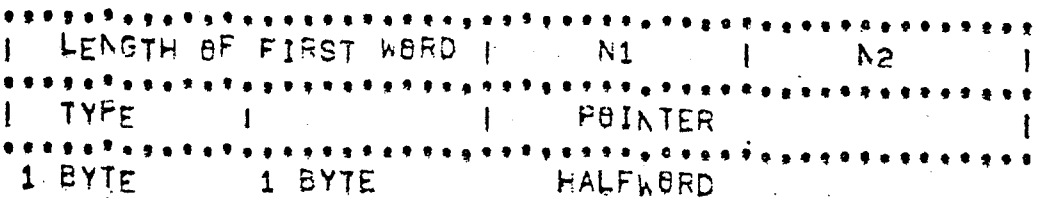

DATA REPRESENTATION

DATA REPRESENTATION

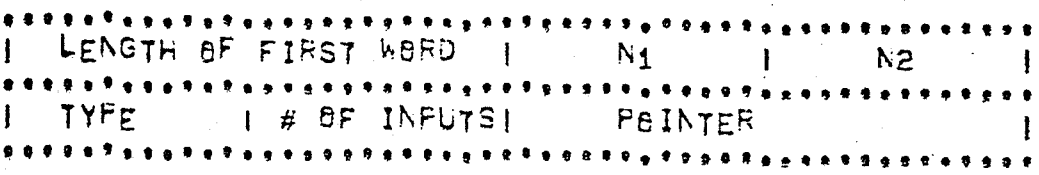
SYMBOL TABLE

THE SYMBOL TABLE WILL BE ARRANGED IN ORDER OF CREATION OF THE SYMBOLS WITH NEWEST SYMBOLS ON TOP. THIS IS SO THAT A TOP TO BOTTOM SEARCH WILL GIVE THE MOST RECENT BINDING FOR A LOCAL VARIABLE. ENTRIES IN THE SYMBOL TABLE WILL BE DOUBLEWORDS.

FORMAT:
OPERANDS, NAMES



PROCEDURES, OPERATIONS, AND COMMANDS



THE SECOND WORD OF THE DOUBLE WORD IS WHAT WILL BE COPIED TO THE P.STACK OR S.STACK. THE FIRST BYTE OF THE FIRST WORD CONTAINS A COUNT OF HOW MANY CHARACTERS (BYTES) ARE IN THE NAME. THE POINTER POINTS TO AN ENTRY IN THE SPT WHICH CONTAINS, AMONG OTHER THINGS, THE ENTIRE NAME (SINCE NAMES CAN BE OF ANY LENGTH).

TWO DIGIT HEX TYPE CODES WILL BE ASSIGNED TO THE FIRST

KINDS OF TOKENS THAT CAN APPEAR IN LOGIC AND ARE TO

BE USED AS TYPE CODES

TYPE NUMBER	NAME
1	PROCEDURE, SYSTEM
2	PROCEDURE, USER
3	LITERAL (NUM)
4	THING, SYSTEM
5	THING, USER
6	NIGHTM
7	RESERVED WORD
8	ABBREVIATION

RESERVED WORDS

PROCEDURES

NAMES

ABBREVIATION

ABBREVIATIONS

LINES

LINE

TRACE

TRACES

TITLE

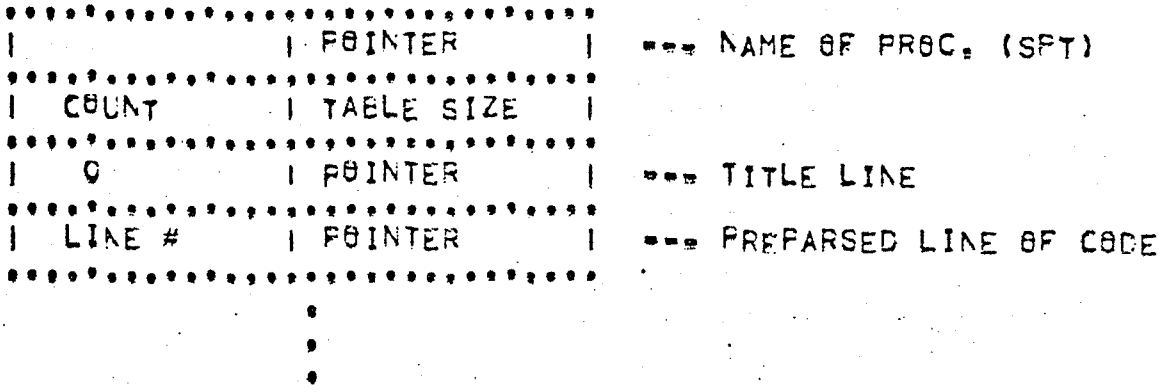
TITLES

ALL

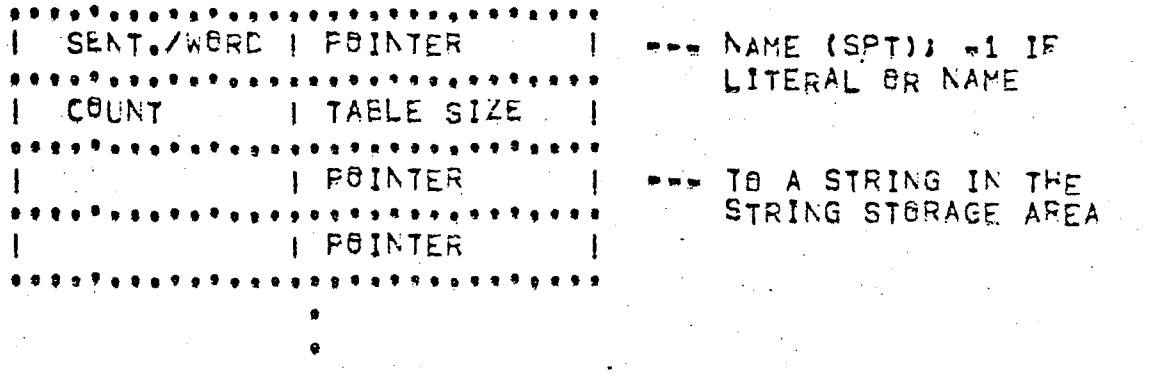
STRING POINTER TABLE

THIS TABLE CONTAINS ENTRIES FOR ALL STRINGS AND PROCEDURES.

FORMAT OF AN ENTRY CORRESPONDING TO A LOGO PROCEDURE:



FORMAT OF AN ENTRY CORRESPONDING TO A LOGO SENTENCE:



NAMES WILL BE SIMILAR TO SENTENCES EXCEPT THAT THE NAME POINTER WILL BE -1. THE COUNT WILL BE THE NUMBER OF WORDS IN THE SENTENCE (ONE IF A WORD). EACH OF THE SIGMA 7 WORDS WILL CONTAIN A POINTER TO A LOGO WORD IN STRING STORAGE. THE SENT./WORD FIELD INDICATES WHETHER THE LOGO THING IS A WORD OR A SENTENCE.

PROCEDURES WILL LOOK LIKE SENTENCES EXCEPT FOR TYPE AND THE EXISTENCE OF LINE NUMBERS IN THE LEFT HALFWORD OF THE POINTER WORDS. A 0 LINE NUMBER DESIGNATES THE TITLE LINE. THE TITLE LINE WILL RESEMBLE A LOCAL STATEMENT EXCEPT THAT ITS OPERAND COMES FROM THE S-STACK. LINE NUMBERS FROM $1-9999$ WILL BE PERMITTED.

P-STACK

THE P STACK IS USED TO HOLD THE NAMES OF COMMANDS, OPERATIONS AND PROCEDURES THAT HAVE BEEN ENCOUNTERED (BUT NOT YET EXECUTED) IN A LINE. EACH ENTRY ALSO CONTAINS THE NUMBER OF OPERANDS THAT MUST STILL BE FOUND FOR THAT ENTRY.

A STACK POINTER DOUBLEWORD (AS DEFINED BY THE SYSTEM (1)) WILL POINT TO THE TOP OF THE STACK.

FORMAT:

TYPE	# OF INPUTS	POINTER
1 BYTE	1 BYTE	HALFWORD

THE ENTRIES OF THE P-STACK WILL BE LIKE THE SECOND WORD OF THE SYMBOL TABLE EXCEPT THAT THE POINTER WILL BE TO THE COUNT FIELD RATHER THAN THE NAME FIELD OF THE ENTRY.

TYPE: A NUMERIC CODE THAT SPECIFIES WHAT KIND OF ELEMENT THIS IS, IE, USER PROCEDURE, SYSTEM PROCEDURE, ETC.

OF INPUTS: AT FIRST THIS FIELD WILL CONTAIN THE TOTAL NUMBER OF OPERANDS REQUIRED. AS OPERANDS ARE FOUND THE NUMBER WILL BE DECREMENTED.

POINTER: THIS POINTS OFF TO THE COUNT FIELD IN THE ENTRY TO THE STRING POINTER TABLE.

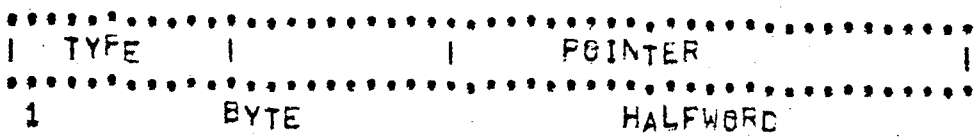
R-STACK

A STACK USED TO SAVE REGISTERS. IT IS SIMILIAR IN STRUCTURE TO THE P AND S STACKS, HOWEVER, THE CONTENTS OF AN ENTRY IS UNFORMATTED.

S-STACK

THIS STACK WILL BE LIKE THE P-STACK EXCEPT THAT ITS ENTRIES WILL BE OPERANDS RATHER THAN OPERATIONS.

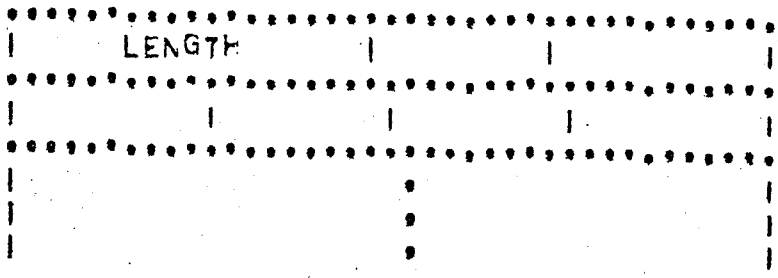
FORMAT:



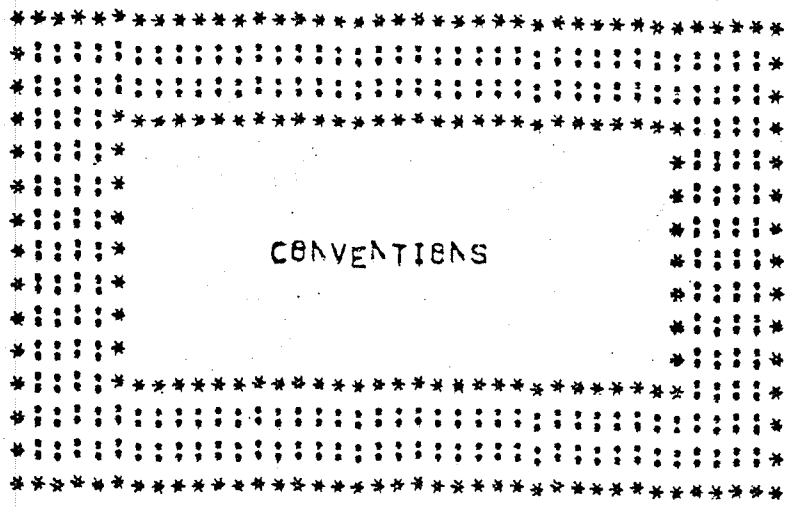
STRING STORAGE AREA

THE POINTERS IN THE SPT WILL POINT TO THE ACTUAL STRINGS.

FORMAT:



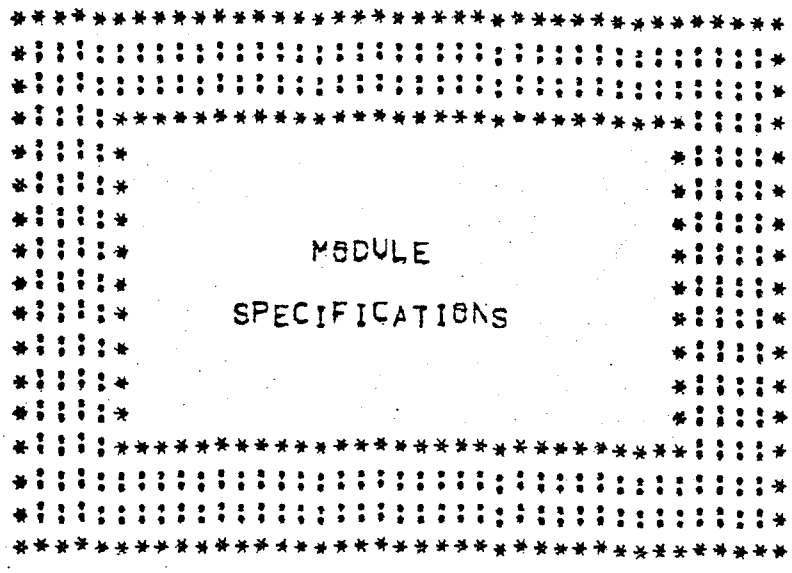
THE LENGTH FIELD CONTAINS THE NUMBER OF CHARACTERS IN THE STRING. THE ACTUAL CHARACTERS WILL BE ONE TO A BYTE FOR AS MANY BYTES AS NEEDED TO CONTAIN THE STRING.

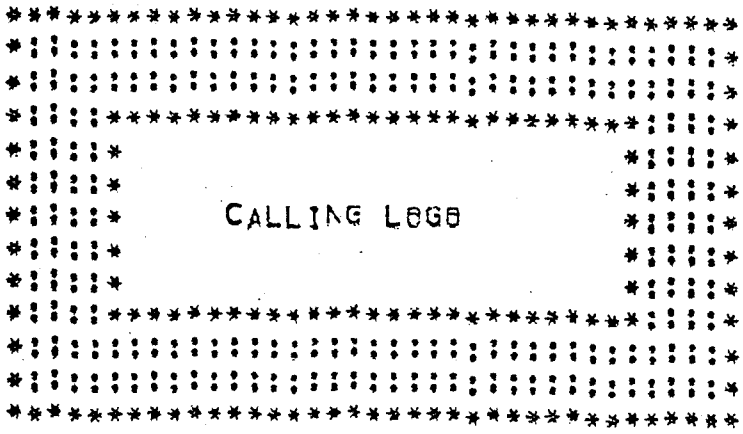

CONVENTIONS

CONVENTIONS

WHENEVER SUITABLE OUR CONVENTIONS WILL BE MUCH LIKE THOSE SPECIFIED IN THE UTS TECHNICAL MANUAL (2). PARAMETERS ARE TO BE PLACED IN REGISTERS. REGISTERS 0-4 AND 12-15 MAY BE CLOBBERED BY ANY ROUTINE. REGISTERS 5-11 ARE PRESUMED SAVED IF USED. I/O POINTERS WILL BE PLACED IN NON-VOLATILE REGISTERS. ROUTINES WILL BAL BY REGISTER 11. REGISTER 11 MAY BE CALLED LINK AND REGISTER 12 MAY BE CALLED STAT SINCE THE USER STATUS (EXECUTING, SERVING, EDITING) WILL BE KEPT IN THAT REGISTER.

FLOWCHARTS -- A CALL TO ANOTHER PROCEDURE IS REPRESENTED BY A HEXAGONAL BOX.

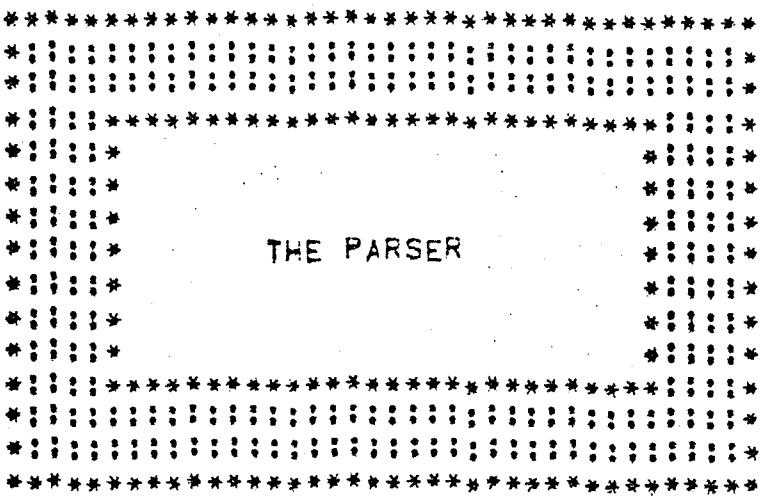

MODULE
SPECIFICATIONS


CALLING LOGO

ALL THIS DOES IS CALL THE THREE INITIALIZING ROUTINES AND THEN THE PARSER. IF ANYTHING HORRIBLE HAPPENS, SYSTEM INTERFACE WILL TRANSFER TO 'RESET' TO KILL THE STACKS AND RE-START THE PARSER.

THESE ROUTINES CAN DESTROY ALL REGISTERS, THIS DOESN'T MATTER, HOWEVER, BECAUSE THE RESET WOULD CLOBBER EVERYTHING ANYWAY.

```
LOGO:     CALL    SYSINIT   ;INITIALIZE SYSTEM
          CALL    STORINIT  ;INITIALIZE STORAGE
RESET:    CALL    STAKINIT  ;CLEAR STACKS
PARSER:    . . . .
```


THE PARSER

THE PARSER

INTERNAL SPECIFICATIONS FOR THE PARSER.

THE PARSER RUNS IN TWO DISTINCT MODES: EDIT (DEFINE) MODE AND SERVICE MODE. DETAILED PROSE PROCEDURES ARE GIVEN FOR EACH OF THESE MODES BELOW.

I. EDIT MODE.

IN THE EDIT MODE, THE PRIMARY PURPOSE OF THE PARSER IS TO STORE THE INPUT LINE IN STRING STORAGE AND CREATE AN ENTRY (LINK) FOR THIS STRING IN THE PROCEDURE'S SPT BLOCK. THIS PART OF THE MODULE REVOLVES AROUND THE LINE NUMBER: IF IT IS NOT PRESENT, THE PARSER GOES TO 'SERVICE' MODE. IF IT IS PRESENT, ITS VALIDITY IS CHECKED: IF VALID, THE APPROPRIATE CHARACTER TRANSFERS ARE MADE AND THE SPT UPDATED.

START: REQUEST AN INPUT LINE FROM THE INTERFACE MODULE

1. BUILD UP THE FIRST ELEMENT (AN ELEMENT IS A NUMBER LITERAL, NAME, COMMENT, NOISE WORD, COMMAND, PROCEDURE NAME, SYSTEM FUNCTION, OR PARENTHESIS). IF ALL THE CHARACTERS ARE DIGITS, ASSUME THE ELEMENT IS A LINE NUMBER. IF NOT, RESET TO BEGINNING OF INPUT LINE AND GO TO 'SERVICE' MODE.

2. CHECK 'EDIT MODE' FLAG. IF NOT IN EDIT MODE OUTPUT 'YOU'RE NOT EDITING ANYTHING' ERROR MESSAGE AND RETURN TO 'START'.

3. CONVERT LINE NUMBER TO BINARY. IF THE LINE NUMBER EQUALS 0, OUTPUT 'LINE NUMBER CAN'T BE 0' ERROR MESSAGE AND RETURN TO 'START'.

4. IF THE NUMBER IS GREATER THAN 9999, OUTPUT 'LINE NUMBER TOO BIG' ERROR MESSAGE AND RETURN TO 'START'.

5. LOCATE SPT BLOCK OF CURRENTLY-EDITED PROCEDURE. ATTEMPT TO FIND AN ENTRY IN THIS SPT BLOCK WITH THE SAME LINE NUMBER AS THE ONE JUST ACCUMULATED. IF FOUND, SET THIS LINE NUMBER TO -1. THE STORAGE ALLOCATOR WILL ERASE THIS ENTRY AT LINE SORT TIME (INVOKED BY 'END' AND 'LIST').

6. ALLOCATE STRING STORAGE FOR THE CURRENT LINE USING 'ALLOCSTR'. THIS WILL PRODUCE A POINTER TO STRING STORAGE IN WHICH THE REMAINDER OF THE INPUT LINE WILL BE STORED. IT ALSO ADDS AN ENTRY TO THE SPT BLOCK POINTED TO BY 'SPTPTR' (THE CURRENTLY-EDITED SPT BLOCK). MOVE THE REMAINDER OF THE INPUT LINE TO THE ALLOCATED STRING STORAGE, AND PUT THE ACCUMULATED LINE NUMBER (IN BINARY) IN THE TOP HALFWORD OF THE SPT ENTRY APPENDED TO THE CURRENT SPT BLOCK. RETURN TO 'START'.

II. SERVICE MODE.

SERVICE MODE IS GENERALLY CONCERNED WITH BREAKING UP THE INPUT LINE INTO ITS ELEMENTS AND PASSING THE RESULTING INFORMATION TO THE EXECUTER IN THE FORM OF A PARSED LIST. A POINTER TO THE BEGINNING OF THIS LIST IS ALSO PASSED.

SERVICE MODE IS ALSO RESPONSIBLE FOR:

1. DETECTING WHEN A COMMAND IS ENCOUNTERED (COMMANDS ARE 'TO', 'EDIT', 'TITLE', 'LIST', 'STORE', 'LOAD', AND 'ERASE'). BECAUSE OF THE FORMAT OF THE INPUT TO THESE COMMANDS, THE INPUTS MUST BE PROCESSED IN A SPECIAL WAY.

2. 'WEEDING OUT' NOISE WORDS AND PARENTHESES AND MAKING A CHECK FOR BALANCED PARENTHESIS. NO ATTEMPT IS MADE TO ANALYZE PARENTHESIS OR NOISE WORD CONTEXT, HOWEVER.

3. CLOSING OFF THE PARSING LIST BY MEANS OF A SPECIAL MARKER WHEN EOL IS ENCOUNTERED AND TURNING CONTROL OVER TO THE EXECUTER WITH A POINTER TO THE PARSED LIST.

THESE FUNCTIONS ARE DESCRIBED BY MEANS OF A PROSE PROCEDURE BELOW.

0. INITIALIZE TOP OF PARSING LIST, ZERO PAREN COUNT.

1. READ CHARACTERS FROM THE INPUT BUFFER UNTIL A NON-BLANK IS ENCOUNTERED.

2. IF THE NON-BLANK CHARACTER IS AN EOL GO TO 12.

3. IF THE NON-BLANK CHARACTER IS A QUOTE OR SLASH, ACCUMULATE AND COUNT CHARACTERS AND WORDS UNTIL A MATCHING QUOTE OR SLASH, OR AN EOL IS ENCOUNTERED. IF AN EOL IS ENCOUNTERED, GENERATE 'MATCHING /' OR 'MATCHING '' ' ERROR MESSAGE AND RETURN TO 'START'. IN ACCUMULATION, IGNORE SPACES AFTER THE FIRST QUOTE OR SLASH AND BEFORE THE FINAL QUOTE OR SLASH AND ALL EXCEPT THE FIRST WHEN SEVERAL OCCUR IN SUCCESSION. IF THE DELIMITERS WERE SLASHES,

THE ELEMENT IS A NAME. LOOK FOR THE NAME IN THE SYMBOL TABLE. IF FOUND, 'SPTPTR' WILL POINT TO THE SYMBOL'S SPT. IF IT IS NOT FOUND, CREATE AN SPT BLOCK FOR IT AND LINK UP THE FIRST ENTRY OF THE SPT BLOCK TO THE EMPTY THING. IF THE DELIMITERS WERE QUOTES, ALLOCATE AN SPT BLOCK FOR THE STRING WITH THE NUMBER OF ENTRIES EQUAL TO THE NUMBER OF LOGO WORDS ACCUMULATED IN THE LITERAL (WORDS ARE SEPARATED BY SPACES). FOR EACH WORD, ALLOCATE STRING STORAGE OF SIZE EQUAL TO THE NUMBER OF CHARACTERS IN THE PARTICULAR WORD; COPY EACH CHARACTER OF THE WORD TO THE STRING STORAGE AND CREATE AN ENTRY IN THE CURRENT SPT BLOCK FOR THAT WORD.

PLACE A WORD ON THE PARSING LIST IN THE FOLLOWING FORMAT:

BYTE 1 = TYPE (LITERAL OR NAME)
BYTE 2 = NOT USED
BYTES 3 & 4 = A POINTER TO THE SPT BLOCK JUST CREATED OR LOCATED.

THEN, RETURN TO 1.

NOTE -- THE REASON THAT WE HAVE DIFFERENT TYPE DESIGNATIONS FOR LITERALS AND NAMES IS THAT LITERAL STORAGE AND SPT BLOCKS ARE RELEASED AFTER USE; NAME STORAGE IS NOT.

4. IF THE NON-BLANK CHARACTER IS A SEMICOLON, IGNORE ALL CHARACTERS UNTIL ANOTHER SEMICOLON OR AN EOL IS ENCOUNTERED. IF ANOTHER SEMICOLON IS ENCOUNTERED, RETURN TO 1. IF AN EOL IS ENCOUNTERED FIRST, RETURN TO 2.

5. IF THE NON-BLANK CHARACTER IS A PARENTHESIS, INCREMENT THE PARENTHESIS COUNT. IF A LEFT PARENTHESIS, INCREMENT PAREN COUNT; IF A RIGHT PARENTHESIS, DECREMENT PAREN COUNT, AND CHECK. IF UNDERFLOW, OUTPUT ERROR MESSAGE AND RETURN TO START. OTHERWISE RETURN TO 1.

6. BUILD UP THE ELEMENT UNTIL A SPACE, PARENTHESIS, QUOTE, SLASH, SEMICOLON OR EOL IS ENCOUNTERED. ACCUMULATE CHARACTER COUNT AS WELL.

7. DETERMINE IF THE ACCUMULATED ELEMENT IS A NUMBER. THIS IS DONE BY CHECKING EACH CHARACTER, AND IF ALL ARE NUMERIC, AND THE FIRST IS EITHER NUMERIC OR + OR -, THE ELEMENT IS A NUMBER. IF SO, TREAT THE ELEMENT AS A LITERAL. THAT IS, ALLOCATE AN SPT BLOCK FOR THE NUMBER LITERAL. ALLOCATE STRING STORAGE OF SIZE EQUAL TO THE NUMBER OF CHARACTERS IN THE NUMBER LITERAL; COPY EACH CHARACTER OF THE NUMBER LITERAL TO THE STRING STORAGE AND CREATE AN ENTRY IN THE CURRENT SPT BLOCK FOR THE LITERAL.

PLACE A WORD ON THE PARSING LIST IN THE FOLLOWING FORMAT:

BYTE 1 = TYPE = LITERAL
BYTE 2 = NOT USED
BYTE 3 & 4 = A POINTER TO THE SPT JUST CREATED.

RETURN TO 1.

8. LOOK FOR THE ELEMENT IN THE SYMBOL TABLE. IF IT IS NOT THERE, GIVE 'XXX NEEDS A MEANING' ERROR MESSAGE AND RETURN TO 'START'.

9. IF THE ELEMENT IS IN THE SYMBOL TABLE, AND IS A NOISE WORD, RETURN TO 1.

10. IF THE ELEMENT IS A PROCEDURE NAME OR AN OPERATION, CREATE AN ITEM ON THE PARSING LIST OF THE FORM:

BYTE 1 = TYPE (FUNCTION OR USER PROCEDURE)
BYTE 2 = NUMBER OF INPUTS REQUIRED
BYTE 3 & 4 = POINTER TO PROCEDURE SPT OR FUNCTION HANDLER ADDRESS.

NOTE -- BYTES 1 AND 2 ARE FORMED DIRECTLY FROM THE FIRST TWO BYTES OF THE SYMBOL TABLE ENTRY FOR THE PROCEDURE OR FUNCTION. BYTES 3 AND 4 ARE RETURNED BY 'FINDTHG' IN THE 'SPTPTR' FIELD (IF THE THING FOUND IS A FUNCTION, 'SPTPTR' CONTAINS THE ADDRESS OF THE FUNCTION HANDLER).

RETURN TO 1.

11. IF WE GET HERE, WE ARE ASSURED THAT THE ELEMENT IS A COMMAND. THE COMMANDS ARE: 'TO', 'ERASE', 'LIST', 'EDIT', 'STORE', 'LOAD', 'TITLE'.

CREATE A PARSING LIST ITEM OF THE FORM:

BYTE 1 = TYPE = SYSTEM PROCEDURE
BYTE 2 = NUMBER OF INPUTS = 1
BYTE 3 & 4 = POINTER TO COMMAND HANDLER (SUPPLIED BY 'SPTPTR' AS IN 10).

NOW, PROCESS THE REST OF THE LINE AS A LITERAL. FOR THE

LITERAL ACCUMULATED, PREPARE A PARSING LIST ELEMENT OF THE FORM:

BYTE 1 = TYPE = LITERAL
BYTE 2 = NOT USED
BYTE 3 & 4 = POINTER TO SPT BLOCK CREATED FOR LITERAL.

12. CHECK PARENTHESIS COUNT. IF NON-ZERO, ISSUE 'PARENTHESIS DON'T MATCH' ERROR AND RETURN TO 'START'.

13. PLACE A TERMINATION MARKER ON THE PARSING LIST. GIVE CONTROL TO THE EXECUTER, WITH A POINTER TO THE TOP OF THE FRESHLY-CREATED PARSING LIST.

I, REGISTER ASSIGNMENTS,

NAME	NUMBER	EXPLANATION
R0	0	CONTAINS CURRENT INPUT CHARACTER
R1	1	INDEX TO INPUT BUFFER
R2	2	TEMP BUFFER CHARACTER COUNT & INDEX
R3	3	EXECUTE LIST INDEX
R4	4	SCRATCH GP OR INDEX REGISTER
R5	5	SCRATCH GP OR INDEX REGISTER
R6	6	R6,R7 AND SR1 = USED FOR 'COPY'
R7	7	SUBROUTINE; SCRATCH WHEN NOT USED
SR1	8	FOR THIS PURPOSE
SR2	9	SUBROUTINE ARGUMENT
SR3	10	SUBROUTINE ARGUMENT
SR4	11	SUBROUTINE LINK REGISTER
LINK	11	
D1	12	SYSTEM STATUS REGISTER
STAT	12	
D2	13	PARENTHESIS COUNT
D3	14	SCRATCH GP REGISTER
D4	15	SCRATCH GP REGISTER

II. MISCELLANEOUS NAMES

NAME	VALUE	EXPLANATION
EOL	X'0D'	END-OF-LINE CHARACTER
CTL:V	X'16'	CONTROL-V CHARACTER

III. ERROR MESSAGE NAMES

PE:NOMQ	8	NO MATCHING QUOTE
PE:NOMS	7	NO MATCHING SLASH
PE:RPRN	9	NO MATCHING)
PE:LPRN	10	NO MATCHING (
PE:CNTO	5	LINE NUMBER CAN'T BE ZERO
PE:LN2BG	4	LINE NUMBER TOO BIG
PE:NTED	6	YOU'RE NOT EDITING ANYTHING
PE:NDSMN	11	XXX NEEDS A MEANING
PE:NOLN	29	THAT LINE DOESN'T EXIST
PE:DEFD	30	XXX DEFINED.
PE:NOMW	31	DON'T USE EMPTY WORD FOR NAME
PE:CNTE2	12	CAN'T EDIT 2 PROCS AT ONCE
PE:ENSLH	51	INPUTS MUST BE ENCLOSED BY /'S.
PE:TOWHT	52	TO WHAT
PE:CNTBP	53	XXX CAN'T BE A PROCEDURE NAME.
PE:ALRDY	54	XXX IS ALREADY DEFINED.
PE:EDWHT	56	EDIT WHAT
PE:CNTES	57	CAN'T EDIT SYSTEM PROC
PE:CNTLS	59	CAN'T LIST A SYSTEM PROC
PE:LSWHT	60	LIST WHAT
PE:MSTBW	0	INPUT MUST BE A WORD

GB - READS A CHARACTER FROM THE INPUT BUFFER INTO CF(2). INPUT BUFFER IS INDEXED BY AF(1). IF AF(2) IS PRESENT, A JUMP IS MADE TO THIS ADDRESS IF THE CHARACTER READ WAS A BLANK.

A. PARSER MAINLINE.

READ AN INPUT RECORD FROM THE INPUT STREAM. DETERMINE IF A LINE NUMBER IS PRESENT, AND IF SO, STORE LINE AWAY IN STRING STORAGE FOR FUTURE PARSING (CHECK CONSTRAINTS FIRST). IF NO LINE NUMBER PRESENT, ASSUME THE LINE IS TO BE PARSED AND EXECUTED IMMEDIATELY; BRANCH TO P:DIRECT.

CALLED ROUTINES: GB, ALLOCSTR, COPY, P:BUILD, RTN2PROC, STAKINIT, STORINIT, SYSINIT

PiDIRECT PARSES ALL THE ELEMENTS IN THE INPUT BUFFER AND CREATES A LIST FOR THE EXECUTER. THE EXECUTER THEN TAKES THE LIST AND PUSHES ARGUMENTS ONTO THE S-STACK AND P-STACK DEPENDING ON THE TYPE OF THE ITEM ON THE LIST. EACH ITEM IS OF THE FORM:

BYTE 0 = TYPE
BYTE 1 = NO. OF INPUTS REQUIRED
BYTES 3,4 = SPT POINTER

'TYPE' REFERS TO THE TYPE OF THE ITEM UNDER CONSIDERATION; THE POSSIBLE TYPES ARE: 1=SYSTEM PROC, 2=USER PROC, 3=LITERAL, 4=SYSTEM THING, 5=USER THING. THE THINGS OF NAMES ARE LOOKED UP IN THE SYMBOL TABLE AND GIVEN TO THE EXECUTER TO BE STACKED AS LITERALS. THE EXECUTER WILL SEE THE COMMAND TYPE (NICKTYP = 6) AS A SYSTEM PROCEDURE TYPE, SINCE THE DISTINCTION MUST BE MADE ONLY AT THE PARSING LEVEL.

'NO. OF INPUTS REQUIRED' IS MEANINGFUL ONLY WHEN THE TYPE IS 'PROCEDURE' AND REPRESENTS THE NUMBER OF ARGUMENTS WHICH MUST BE ACCUMULATED ON THE S-STACK BEFORE THE PROCEDURE CAN BE EXECUTED. FOR ALL OTHER TYPE DESIGNATIONS, THIS FIELD IS ZERO.

THE 'SPT POINTER' POINTS TO THE 'COUNT' WORD OF THE SPT BLOCK ASSOCIATED WITH THE LITERAL OR PROCEDURE, RELATIVE TO A BASE ADDRESS. ABSOLUTE ADDRESSES ARE COMPUTED AS FOLLOWS:

TYPE	ABSOLUTE ADDRESS
1 = SYSTEM PROCEDURE	SPT POINTER + SYSPROCS
2 = USER PROCEDURE	SPT POINTER + SPT
3 = LITERAL	SPT POINTER + SPT
4 = SYSTEM THING	SPT POINTER + SSPT
5 = USER THING	SPT POINTER + SPT
6 = COMMAND	SPT POINTER + SYSPROCS

WHEN THE PARSING OPERATION IS FINISHED, A -1 ENTRY IS PLACED ON THE EXECUTE LIST, THE LIST IS TRANSFERED TO STRING STORAGE AND THE EXECUTER IS CALLED WITH 'CODE' POINTING TO THE FIRST ENTRY OF THE NEWLY-CREATED LIST.

C. P:CMND

P:CMND STORES THE CHARACTERS REMAINING IN SI:IBUF IN STRING STORAGE. LATER, THE ROUTINE HANDLING THE PARTICULAR COMMAND WILL RETRIEVE THIS INFORMATION FROM STRING STORAGE AND, AT COMPLETION OF EXECUTION OF THE COMMAND, WILL RETURN THE STORAGE TO FREE SPACE. P:CMND PLACES TWO ITEMS ON THE EXECUTE LIST: 1) A POINTER TO THE COMMAND INVOKED (SPTPTR) WITH TYPE = 1 (SYSTEM PROCEDURE) AND 1 INPUT REQUIRED, 2) THE INPUT REQUIRED (THE REST OF THE LINE) WITH TYPE = 3 (LITERAL) AND THE POINTER TO THE STRING CONTAINING THE CHARACTERS REPRESENTING THE REST OF THE LINE.

FINALLY, P:CMND BRANCHES TO P:CLSLST.

ROUTINES CALLED: ALLOCSTR, COPY

D. P;BUILD - BUILDS UP AN ELEMENT FROM THE INPUT BUFFER (SI;IBUF) INTO THE TEMPORARY BUFFER, P;TBUF. BUILDING CONTINUES UNTIL A DELIMITER IS ENCOUNTERED.

1). INPUT - 1) POINTER TO THE APPROPRIATE DELIMITER VECTOR IN SR2

2) FIRST CHARACTER TO BE TESTED IN R_0.

2). OUTPUT- 1) TERMINATION DELIMITER IN R_0

2) NUMBER OF CHARACTERS BUILT UP IN P;TBUF IN R2 (INDEX TO P;TBUF)

3) SR3 WILL BE ZERO IF ALL CHARACTERS BUILT UP WERE DIGITS, NON-ZERO OTHERWISE.

3). THERE ARE TWO CALLING SEQUENCES:

```
BAL,LINK    P;BUILD
BAL,LINK    P;BUILD1
```

THE FIRST SEQUENCE CLEARS R2 TO ZERO PRIOR TO BUILDING; THE 2ND DOES NOT (A PICK-UP-WHERE-YOU-LEFT-OFF SITUATION). IN THE 2ND CASE, R2 DOES NOT CONTAIN THE NUMBER OF CHARACTERS BUILT UP AT EXIT, BUT IS STRICTLY AN INDEX INTO P;TBUF.

4). ROUTINES CALLED -- GB, COPYXXX, SYSERR, P;GUOTE P;SLASH, FINDTHG, P;SEMI, P;LPAREN, P;RPAREN, ALLOCSPT, ALLOCSTR, COPY, EXECUTE

E. P:CMP - COMPARES THE CHARACTER IN RC AGAINST THE CHARACTER TABLE POINTED TO BY SR2. THE CHARACTER TABLES ARE PACKED ONE CHARACTER PER BYTE AND ARE TERMINATED BY A '0' BYTE. THEY MUST START ON A WORD BOUNDARY.

1). INPUT - 1) CHARACTER TO BE TESTED IN RC
2) POINTER TO DELIMITER TABLE IN SR2.

2). OUTPUT- NONE.

3). CALLING SEQUENCE: BAL,LINK P:CMP
CHARACTER MATCHED
CHARACTER DID NOT MATCH

4). ROUTINES CALLED: NONE

F. P;CTRL - COMPARES THE CHARACTER IN RC AGAINST THE CHARACTER TABLE POINTED TO BY SR2, AND IF A MATCH OCCURS, A BRANCH IS TAKEN TO THE LOCATION IN THE BRANCH TABLE POINTED TO BY SR3 AND INDEXED BY THE LOCATION OF THE MATCHING CHARACTER IN THE CHARACTER TABLE. AS IN P;COMP, THE CHARACTER TABLES ARE PACKED ONE CHARACTER PER BYTE AND ARE TERMINATED WITH A '0' BYTE; THEY MUST START ON A WORD BOUNDARY. THE BRANCH TABLES ARE ORGANIZED ON A WORD BASIS; EACH WORD CONTAINS A BRANCH INSTRUCTION TO THE ROUTINE APPROPRIATE TO THE CORRESPONDING CHARACTER IN THE CHARACTER TABLE.

1). INPUT - 1) POINTER TO THE CORRECT CHARACTER TABLE TABLE IN SR2

2) POINTER TO THE CORRECT BRANCH TABLE IN SR3

3) CHARACTER TO BE TESTED IN R_0.

2). OUTPUT- NONE.

3). CALLING SEQUENCE; BAL,LINK P;CTRL
CHARACTER DID NOT MATCH

4). ROUTINES CALLED; NONE

N O T E --- R_6 IS DESTROYED.

G. P:QUOTE - PROCESSES QUOTED LITERALS FROM SI:IBUF. CREATES SPT FOR LITERAL, AND MOVES EACH WORD OF LITERAL TO STRING STORAGE. IF AN ERROR OCCURS, THE SPT AND ASSOCIATED STRINGS ARE DE-ALLOCATED.

1). INPUT - NONE.

2). OUTPUT- 1) P:SPTPTR WILL POINT TO NEWLY-CREATED SPT BLOCK IF NORMAL EXIT.

3). CALLING SEQUENCE: BAL,LINK P:QUOTE
ERROR OCCURRED
NORMAL EXIT

4). ROUTINES CALLED: ALLOCSPT,GB,P:BUILD, ALLOCSTR, COPY, RTNSPT, SYSERR

h. P;SLASH - BUILDS UP A NAME FROM SI;IBUF INTO THE TEMPORARY BUFFER IN TEXTC FORMAT. USED IN PREPARATION FOR A SYMBOL TABLE SEARCH.

1). INPUT - NONE.

2). OUTPUT- 1) R2 CONTAINS THE CHAR COUNT OF NAME.

3). CALLING SEQUENCE; BAL,LINK P;SLASH
ERROR OCCURRED
NORMAL EXIT

4). ROUTINES CALLED; GB, P;BUILD, SYSERR

1. P:SEMI - PROCESSES (BYPASSES) COMMENTS. A COMMENT BEGINS WITH A SEMICOLON AND ENDS WITH EITHER A SEMICOLON OR AN EOL. NO ERROR CONDITION.

   1). INPUT - NONE.

   2). OUTPUT- 1) TERMINATION CHAR (SEMI OR EOL) IN R0.

   3). CALLING SEQUENCE: BAL,LINK P:SEMI

   4). ROUTINES CALLED: GB

J. P;LPAREN - PROCESSES LEFT PARENTHESIS, INCREMENTS PARENTHESIS COUNT IN D2.

1). INPUT - NONE,

2). OUTPUT- NONE,

3). CALLING SEQUENCE: BAL,LINK P;LPAREN

4). ROUTINES CALLED: NONE

K. P;RPAREN - PROCESSES RIGHT PARENTHESIS. DECREMENTS PARENTHESIS COUNT IN D2. IF COUNT BECOMES NEGATIVE, OUTPUTS 'UNBALANCED PARENTHESIS' ERROR MESSAGE.

1). INPUT - NONE.

2). OUTPUT- NONE,

3). CALLING SEQUENCE: BAL,LINK P;RPAREN
PAREN UNDRFLW ERROR
NORMAL EXIT

4). ROUTINES CALLED: SYSERR

L. P:BOTHER - BUILDS UP ANY ELEMENT BTHER THAN THOSE BUILT UP BY THE BTHER SUBROUTINES. STORES THE CHARACTERS IN THE ELEMENT IN TEXTC FORMAT IN P:TBUF. THERE IS NO ERROR CONDITION.

1). INPUT - 1) FIRST CHARACTER OF ELEMENT IN R_0.

2). OUTPUT- 1) R_2 CONTAINS THE NUMBER OF CHARACTERS IN THE BUILT-UP ELEMENT.

3). CALLING SEQUENCE: BAL,LINK P:BTHER

ELEMENT IS A NOISE WORD
ELEMENT IS A NUMBER
ELEMENT IS A USER PROC
ELEMENT IS A SYSTEM PROC
ELEMENT IS A COMMAND
ELEMENT IS NONE OF ABOVE

IN CASES RTN+2 THROUGH R_{TN+4}, P:SPTPTR WILL POINT TO THE SPT BLOCK OF THE FOUND ENTRY AT EXIT. THE TOP HALFWORD OF P:SPTPTR WILL CONTAIN TYPE & NO. OF INPUTS INFORMATION.

4). ROUTINES CALLED: P:BUILD, FINDTHG

M. P!RTNCD- AFTER THE EXECUTER IS DONE WITH THE PARSED LINE HANDED HIM, HE RETURNS HERE TO ACQUIRE THE NEXT LINE OF CODE. IF WE ARE AT THE TOP LEVEL AND NOT IN EDIT MODE, COMPACT IS CALLED (WHICH DOES STORAGE MANAGEMENT CLEANUP), AND WE RETURN FOR ANOTHER LINE TO BE INPUT FROM THE TERMINAL. OTHERWISE, 'LINENO' IS CHECKED. IF ZERO, THE NEXT LINE OF THE PROCEDURE IS LOCATED, THE CODE TRANSFERRED TO THE INPUT BUFFER, AND PARSING BEGINS ON THE NEW LINE. IF THERE IS NO NEXT LINE, A BRANCH IS TAKEN TO 'X$TP0', WHICH IS RESPONSIBLE FOR GENERATING A RTN2PROC CALL. IF 'LINENO' IS NON-ZERO, THIS SIGNIFIES THAT THE EXECUTER CHANGED THE COURSE OF THE PROGRAM EXECUTION (VIA A 'GOTO' STATEMENT). I SEARCH FOR THE NEW LINE IN THE PROCEDURE'S SPT. IF FOUND, I MOVE THE CODE ON THIS LINE TO THE INPUT BUFFER, AS ABOVE, AND PROCEED WITH PARSING. IF THE LINE NUMBER ISN'T FOUND, I GENERATE AN ERROR MESSAGE 'THAT LINE DOESN'T EXIST' AND BRANCH TO PARSERR.

N. COPYXXX - COPIES PROCEDURE NAME TO SI:INSM FROM P:TBUF. 45
P:TBUF IS IN TEXTC FORMAT, SI:INSM WILL BE IN
HALFWORD COUNT FORMAT. A CHECK IS MADE TO SEE
THAT THE NUMBER OF CHARACTERS COPIED DOES NOT EX-
CEED THE SIZE OF SI:INSM. R_6, R7 AND SR_1 ARE LOST.

6. PINKBIN - CONVERTS A NUMBER IN HALFWORD-COUNT TEXT FORMAT INTO ITS BINARY EQUIVALENT. IF THE NUMBER IS LESS THAN 1 OR GREATER THAN 9999, THE ERROR RETURN IS TAKEN. THE ERROR RETURN IS ALSO TAKEN IF LOGO IS NOT IN 'EDIT' MODE WHEN THIS ROUTINE IS CALLED. AN ATTEMPT IS MADE TO LOCATE AN ENTRY IN THE CURRENT SPT BLOCK WITH LINE NUMBER EQUAL TO THE NUMBER CONVERTED; DIFFERENT RETURNS ARE TAKEN DEPENDING ON THE RESULTS OF THIS TEST.

INPUT - 1) TEXT NUMBER TO BE CONVERTED IN SI:IBUF, IN HALF-WORD COUNT FORMAT.

2) POINTER TO TOP OF CURRENT SPT IN P:TOSPT.

3) R2 CONTAINS CHARACTER COUNT OF THE NUMBER IN SI:IBUF.

OUTPUT- 1) ERROR MESSAGES, AS APPROPRIATE.

2) CONVERTED NUMBER IN BINARY IN D4.

3) R4+R3 WILL YIELD POINTER TO ENTRY IN SPT WHICH HAS LINE NUMBER EQUAL TO CONVERTED NUMBER, IF A MATCH OCCURRED.

CALLING SEQUENCE: BAL,LINK P;MKBIN

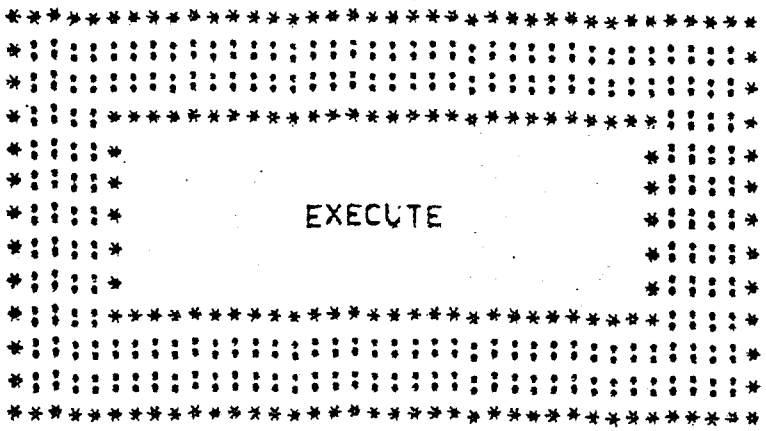
ERROR RETURN

LINE NUMBER MATCH

NO LINE NUMBER MATCH

N O T E --- R0, R1, AND R2 ARE ALTERED.

CALLED ROUTINES: GB, SYSERR


EXECUTE

EXECUTE

DESCRIPTION AND EXTERNAL SPECIFICATIONS

EXECUTE TAKES A PARSED LIST AND DOES THE COMMANDS, OPERATIONS, AND PROCEDURES. EXECUTE USES THE P-STACK TO STORE THE NAMES OF OPERATIONS, PROCEDURES, AND COMMANDS. THE S-STACK IS USED TO STORE THE POINTERS TO INPUTS OF PROCEDURES (ETC.) THAT HAVE NOT YET BEEN EXECUTED. EXECUTE SERVES AS A DISPATCHER FOR THE MODULES THAT ACTUALLY PERFORM THE OPERATIONS. A DETAILED DESCRIPTION OF EXECUTE IS CONTAINED IN THE GUIDE FOR SYSTEM PROGRAMMERS. (4)

CALLED BY THE PARSER, EXECUTE TAKES THE PARSED LINE AND TOKENS AND BUILDS THE P-STACK AND THE S-STACK. EXECUTE CAN CALL LOGO VERB ROUTINES, OR THE PARSER TO PARSE OTHER PROCEDURES. RETURN POINTS FROM LOGO VERBS WILL BE NORMAL, ERROR, TERMINATE LINE, AND TERMINATE PROCEDURE. AT THE START OF EXECUTION, THE BREAK FLAG IS QUERIED; IF IT IS SET, THE BREAK PROCEDURE IS ENTERED, IF THE FLAG IS NOT SET, EXECUTION CONTINUES.

FOR RETURNS:

```
    BAL,LINK     ROUTINE
    NORMAL RETURN -- ONE OUTPUT
 +1         NORMAL RETURN -- ZERO OUTPUTS
 +2 ERROR RETURN
 +3 CONTINUE LINE
 +4 TERMINATE PROCEDURE -- ONE OUTPUT
 +5 TERMINATE PROCEDURE -- ZERO OUTPUTS
```

AN ERROR RETURN ALSO TERMINATES THE PROCEDURE.

INTERNAL SPECIFICATIONS OF EXECUTE

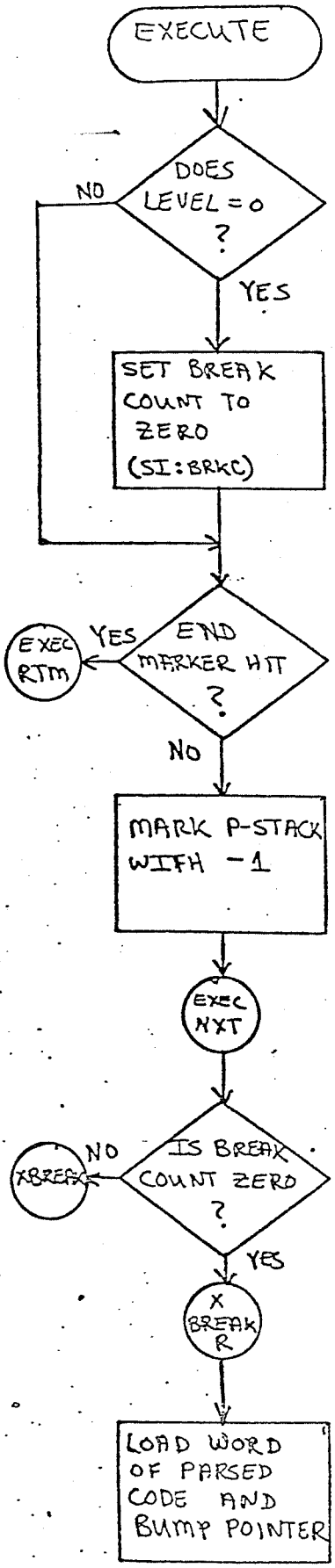
NAME: EXECUTE

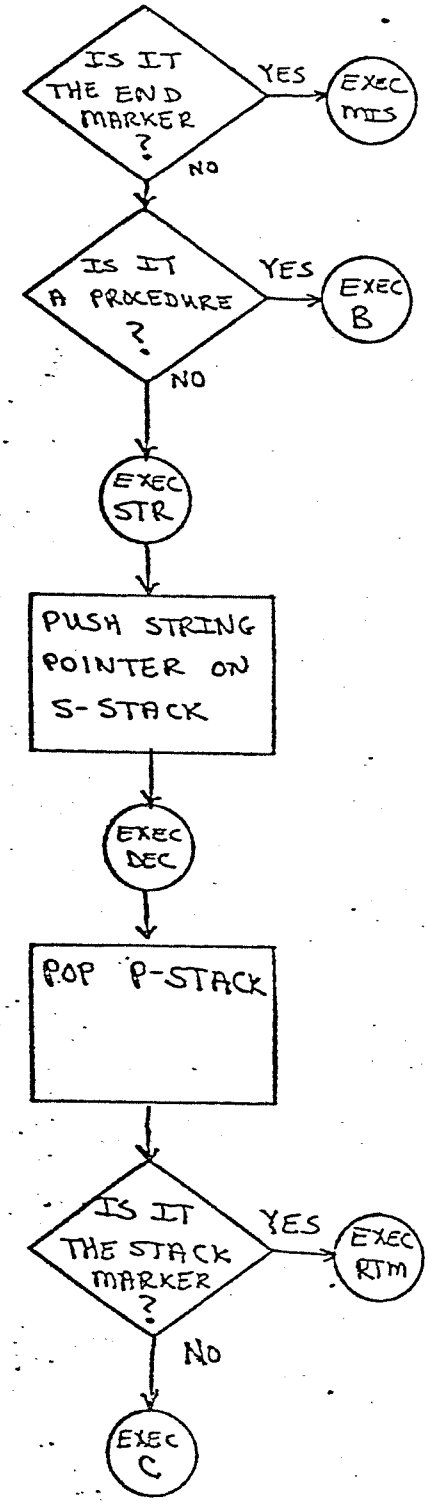
FUNCTION: EXECUTES CODE GENERATED BY PARSER

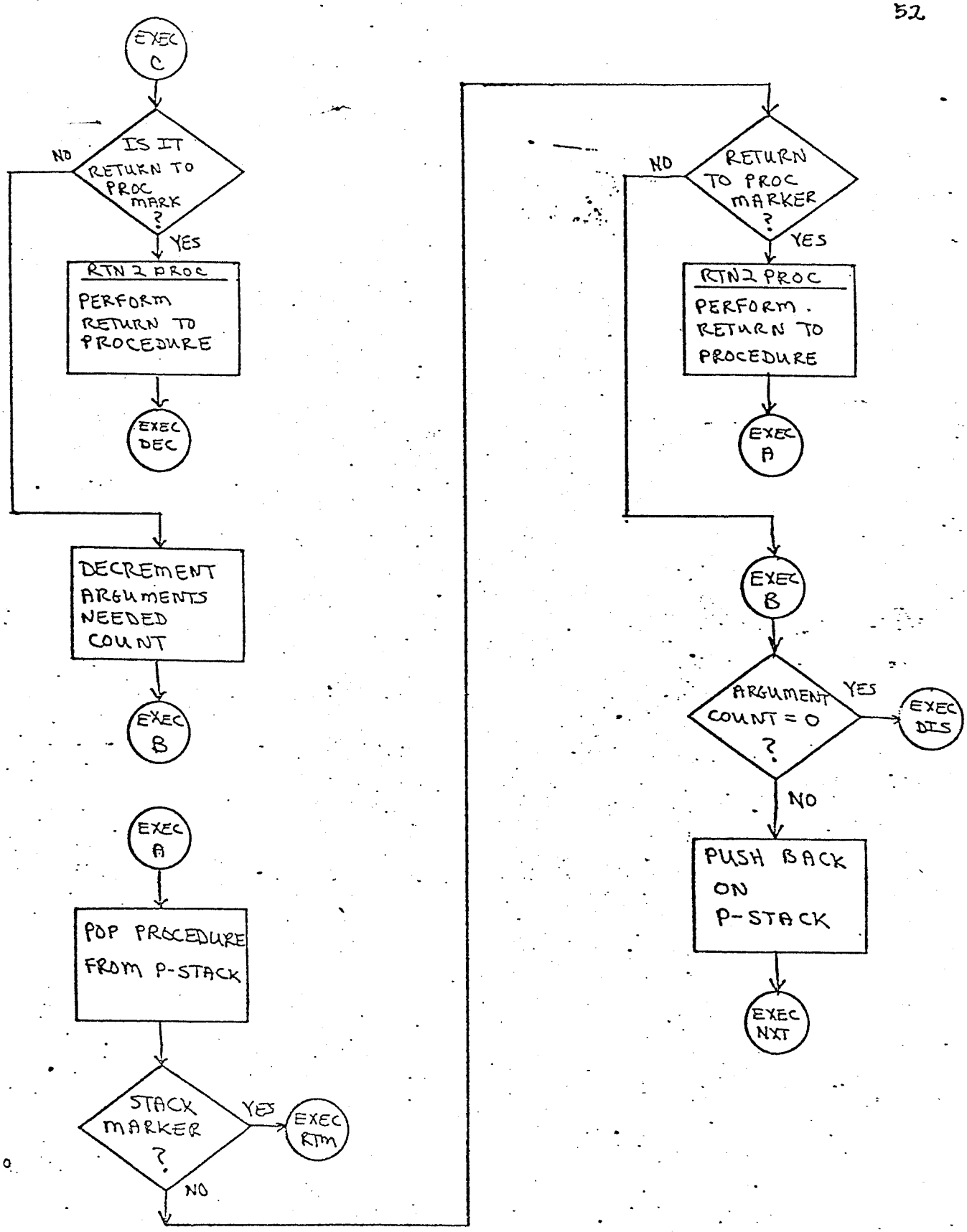
CALLED BY: PARSER

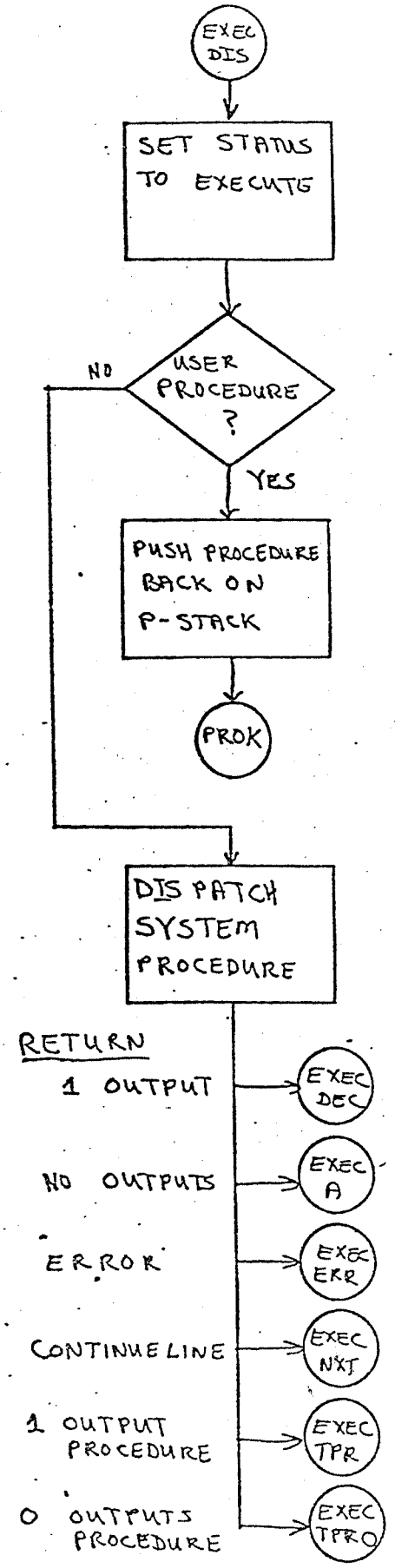
CALLED ROUTINES: ALL SYSTEM FUNCTION ROUTINES ARE CALLED BY THE EXECUTER AND THE PARSER

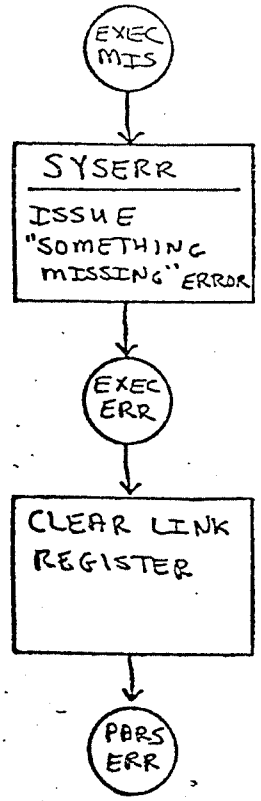
FLOWCHART ON THE FOLLOWING PAGES

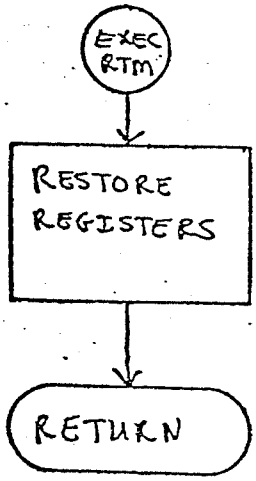

EXECUTE
DOES LEVEL = 0 ?
NO
YES
SET BREAK COUNT TO ZERO (SI:BRKC)
END MARKER HIT ?
YES
EXEC RTM
No
MARK P-STACK WITH -1
EXEC NXT
IS BREAK COUNT ZERO ?
NO
XBREAK
YES
X BREAK R
LOAD WORD OF PARSED CODE AND BUMP POINTER


IS IT THE END MARKER ?
YES
EXEC MIS
NO
IS IT A PROCEDURE ?
YES
EXEC B
NO
EXEC STR
PUSH STRING POINTER ON S-STACK
EXEC DEC
POP P-STACK
IS IT THE STACK MARKER ?
YES
EXEC RTM
No
EXEC C


EXEC C
IS IT RETURN TO PROC MARK ?
NO
YES
RTN 2 PROC
PERFORM RETURN TO PROCEDURE
EXEC DEC
DECREMENT ARGUMENTS NEEDED COUNT
EXEC B
EXEC A
POP PROCEDURE FROM P-STACK
STACK MARKER ?
YES
EXEC RTM
NO
RETURN TO PROC MARKER ?
NO
YES
RTN2 PROC
PERFORM . RETURN TO PROCEDURE
EXEC A
EXEC B
ARGUMENT COUNT = 0 ?
YES
EXEC DIS
NO
PUSH BACK ON P-STACK
EXEC NXT


EXEC DIS
SET STATUS TO EXECUTG
NO
USER PROCEDURE ?
YES
PUSH PROCEDURE BACK ON P-STACK
PROK
DISPATCH SYSTEM PROCEDURE
RETURN
1 OUTPUT
EXEC DEC
NO OUTPUTS
EXEC A
ERROR
EXEC ERR
CONTINUE LINE
EXEC NXT
1 OUTPUT PROCEDURE
EXEC TPR
0 OUTPUTS PROCEDURE
EXEC TPRO


EXEC MIS
SYSERR
ISSUE "SOMETHING MISSING" ERROR
EXEC ERR
CLEAR LINK REGISTER
PARS ERR


EXEC RTM
RESTORE REGISTERS
RETURN

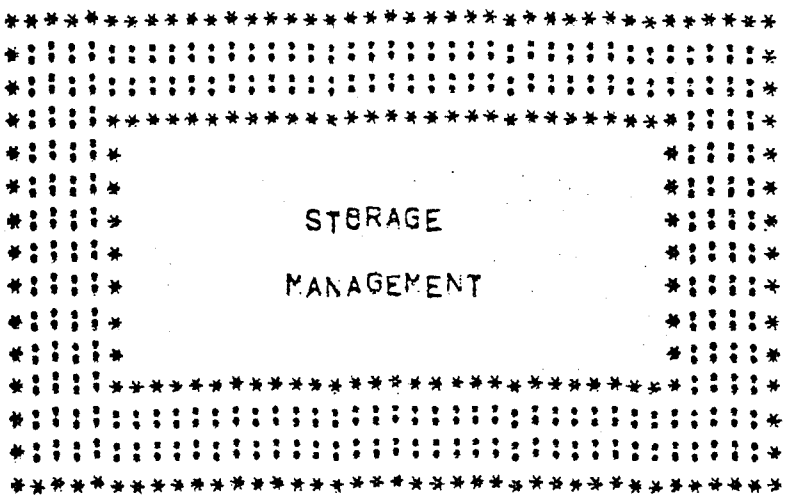
PROK - CALLED BY THE EXECUTER PRIOR TO THE EXECUTION OF A USER PROCEDURE. SAVES CURRENT OPERATING ENVIRONMENT (I.E., SAVES SUCH PARAMETERS AS TRUTH FLAG, PROCEDURE NAME POINTER AND LINE NUMBER, CODE LIST POINTERS) ON THE P-STACK. INCREMENTS RECURSION DEPTH COUNT. THEN, IT BINDS THE NAMES LOCAL TO THE PROCEDURE. FINALLY, IT DEFAULTS THE TRUTH FLAG TO 'TRUE', INITIALIZES 1ST LINE OF PROCEDURE TO BE PARSED, AND BRANCHES TO THE PARSER AT A POINT WHERE THIS LINE WILL BE PREPARED FOR PARSING.

A CHECK IS MADE TO SEE IF THE PROCEDURE'S TRACE FLAG IS SET. IF SO, A LINE IS PRINTED CONTAINING THE PROCEDURE NAME AND IT'S INPUTS. THIS MESSAGE IS INDENTED TO CORRESPOND TO THE PROCEDURE DEPTH.

RTN2PROC - COUNTERPART OF PROK. CALLED BY THE EXECUTER WHEN HE POPS THE 0 MARKER OFF THE P-STACK. RESTORES PARAMETERS SAVED BY PROK AND UNBINDS THE LOCAL NAMES OF THE PROCEDURE JUST FINISHED (DELETES THEM FROM THE SYMBOL TABLE). RETURNS TO CALLING POINT SINCE RTN2PROC IS ENTERED BY:

```
BAL,LINK    RTN2PROC
```

RTN2PROC IS ALSO CALLED FROM PARSERR TO 'UNWIND' THE P-STACK AFTER AN ERROR HAS OCCURRED AND WE ARE SEVERAL LEVELS DEEP INTO USER PROCEDURES.


STORAGE
MANAGEMENT

STORAGE MANAGEMENT-EXTERNAL SPECIFICATIONS

BRIEF FUNCTIONAL DESCRIPTION

I. INITIALIZATION OF DATA STRUCTURES IN THE USER'S VIRTUAL SPACE.

II. ROUTINES TO MANIPULATE THE STRUCTURES--(CALLED BY ANY MODULE)

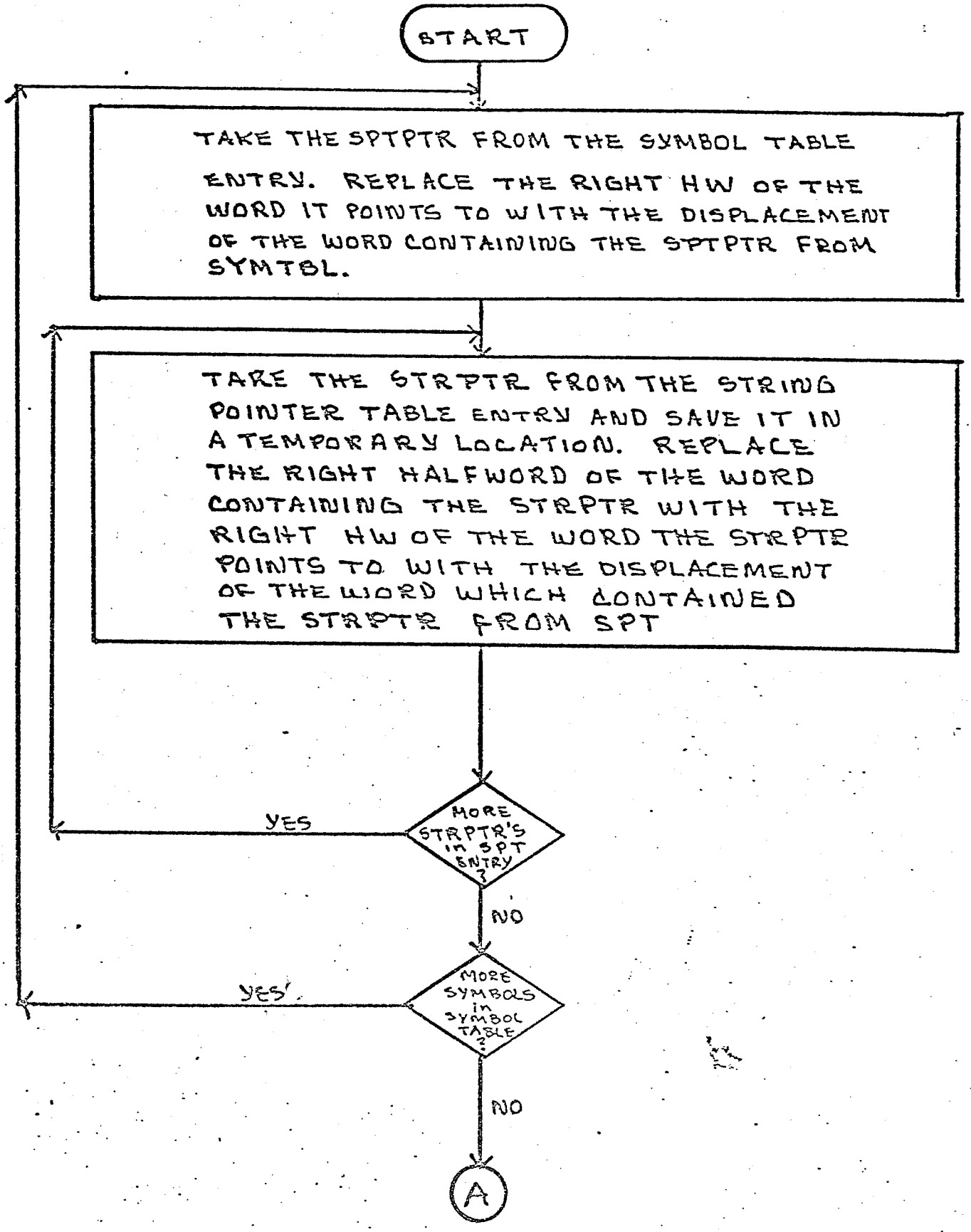
- A. ALLOCATION
- B. DEALLOCATION
- C. ROUTINES ARE CALLED BY OTHER MODULES (MACROS WILL BE WRITTEN TO FACILITATE THE CALLING PROCEDURE.)

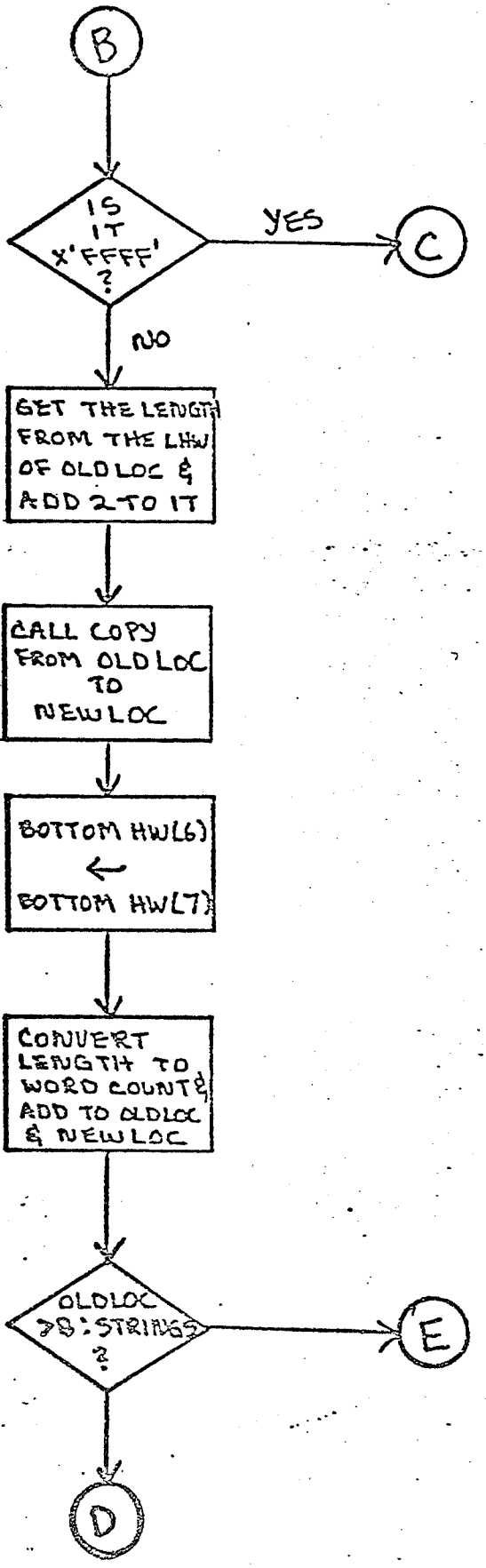
III. SCANNING AND COMPACTING

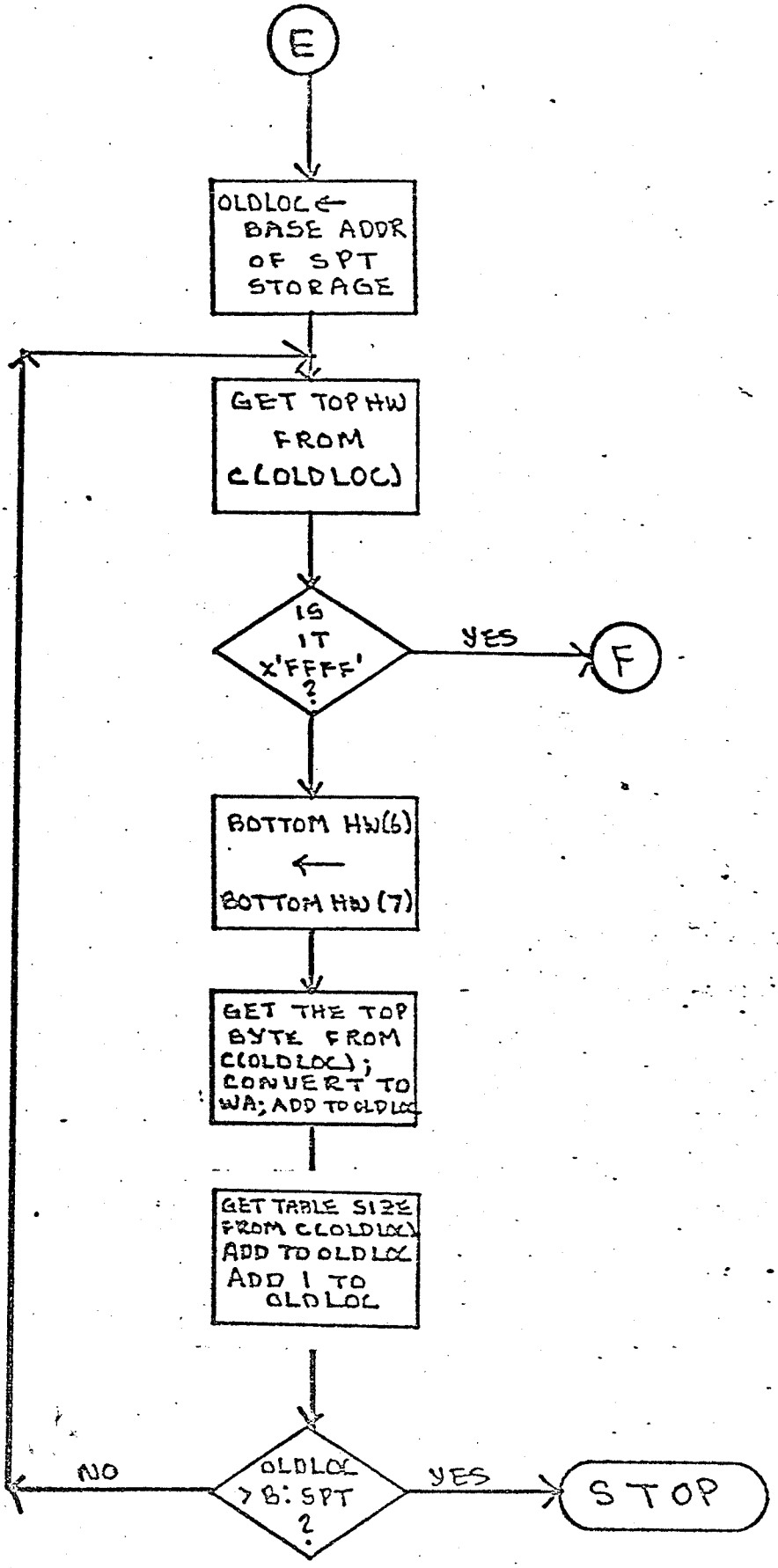
- A. CALLED BY THE EXECUTIVE
- B. COMPACTS THE INDIVIDUAL STRUCTURE SAVING ONLY USEFUL INFORMATION.
- C. REQUESTS VIRTUAL PAGES TO CORRECT THE CAUSE OF THE OVERFLOW (MAY HAVE TO PHYSICALLY MOVE OTHER STRUCTURES TO RESOLVE CONFLICTS IN ADRESSES IN THE USER'S VIRTUAL SPACE)

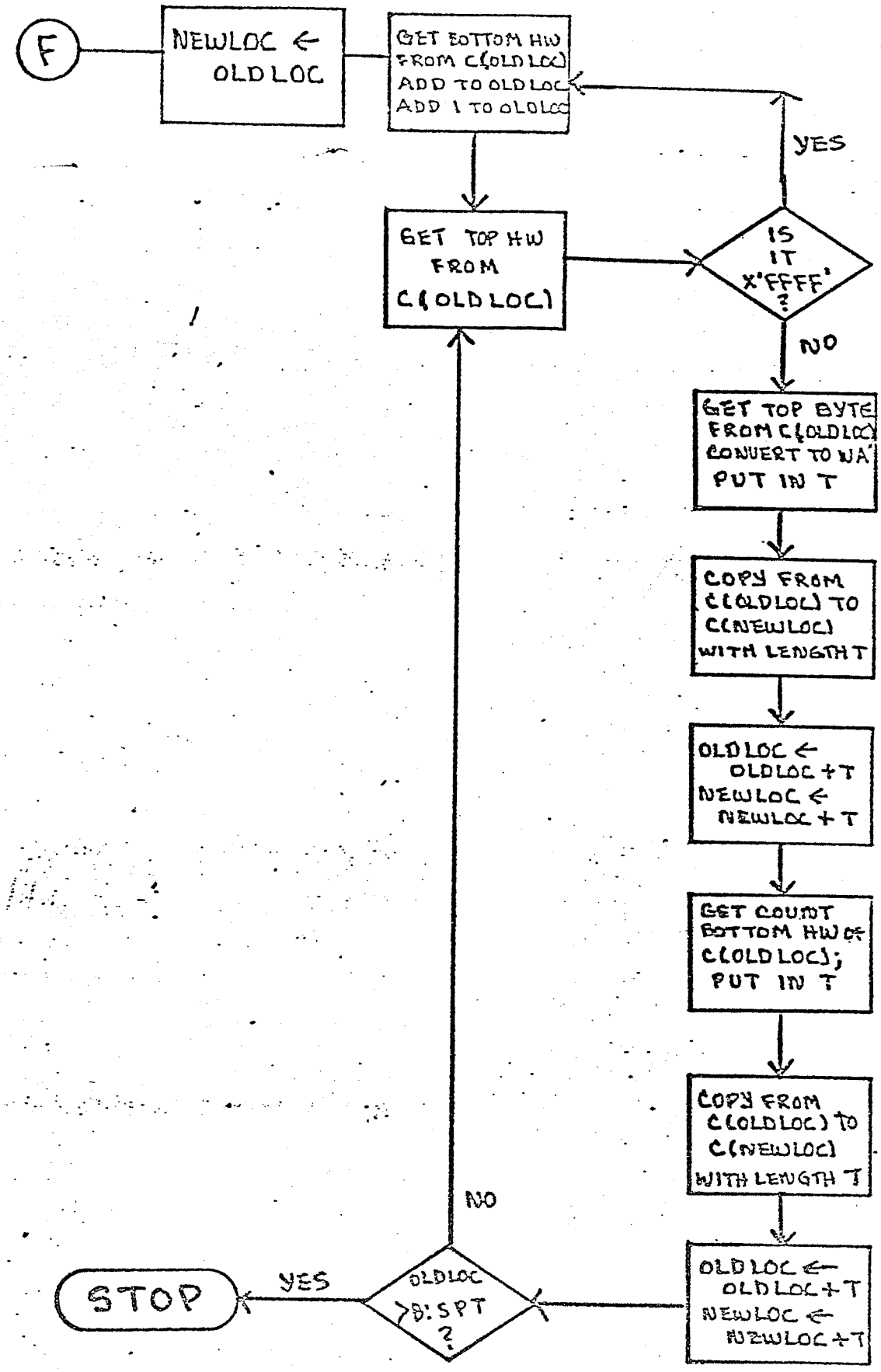
SCANNING AND COMPACTING

(GARBAGE COLLECTION)

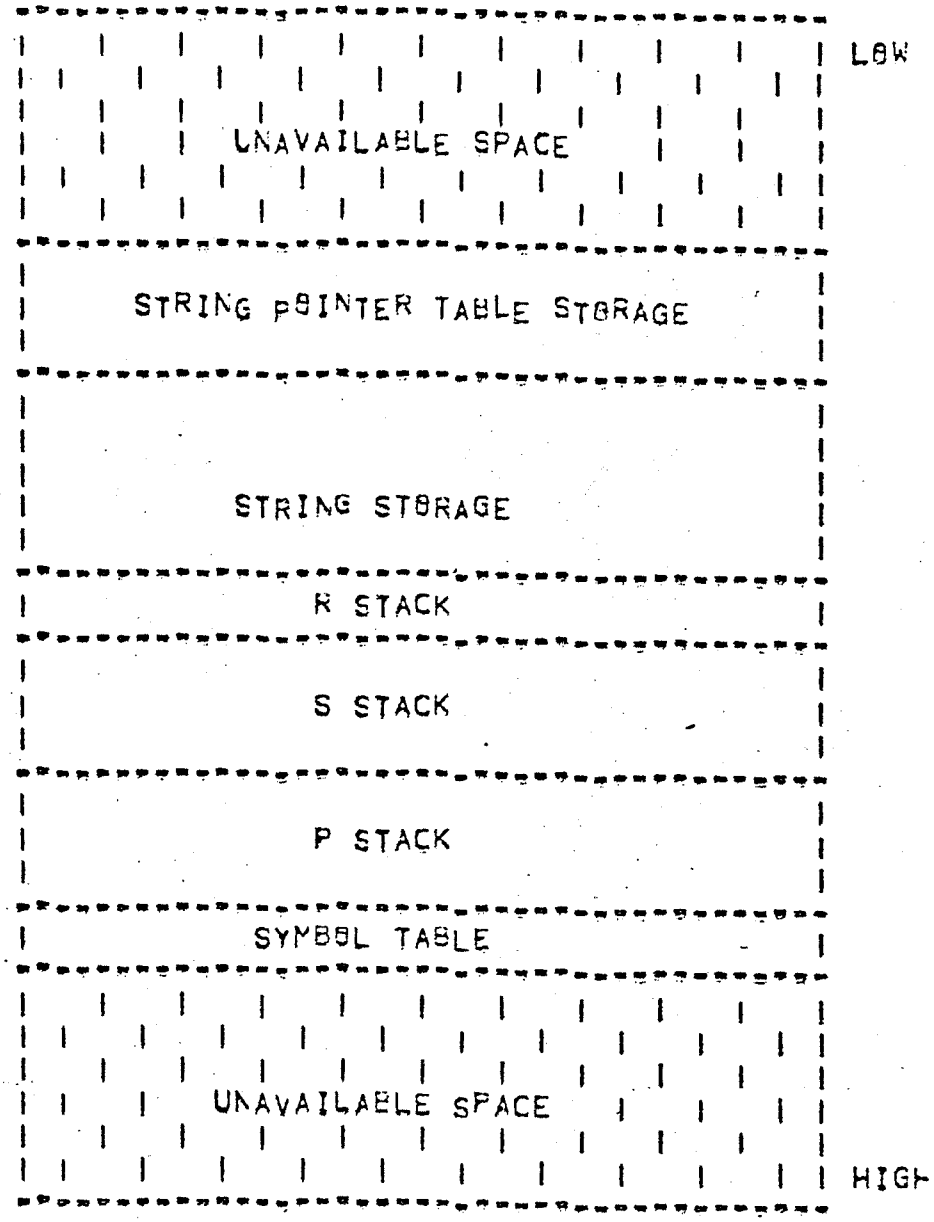



B
IS IT X'FFFF' ?
YES
C
NO
GET THE LENGTH FROM THE LHW OF OLD LOC & ADD 2 TO IT
CALL COPY FROM OLD LOC TO NEW LOC
BOTTOM HW(6) ← BOTTOM HW(7)
CONVERT LENGTH TO WORD COUNT & ADD TO OLDLOC & NEW LOC
OLDLOC >B'STRINGS' ?
E
D


E
OLDLOC ← BASE ADDR OF SPT STORAGE
GET TOP HW FROM C(OLDLOC)
IS IT X'FFFF' ?
YES
F
BOTTOM HW(6) ← BOTTOM HW(7)
GET THE TOP BYTE FROM C(OLDLOC); CONVERT TO WA; ADD TO OLDLOC
GET TABLE SIZE FROM C(OLDLOC) ADD TO OLDLOC ADD 1 TO OLDLOC
OLDLOC > B:SPT ?
NO
YES
STOP


F
NEWLOC ← OLDLOC
GET BOTTOM HW FROM C(OLDLOC) ADD TO OLDLOC ADD 1 TO OLDLOC
YES
GET TOP HW FROM C(OLDLOC)
IS IT X'FFFF'?
NO
GET TOP BYTE FROM C(OLDLOC) CONVERT TO HA PUT IN T
COPY FROM C(OLDLOC) TO C(NEWLOC) WITH LENGTH T
OLDLOC ← OLDLOC + T
NEWLOC ← NEWLOC + T
GET COUNT BOTTOM HW OF C(OLDLOC); PUT IN T
COPY FROM C(OLDLOC) TO C(NEWLOC) WITH LENGTH T
OLDLOC ← OLDLOC + T
NEWLOC ← NEWLOC + T
NO
OLDLOC > B:SPT ?
YES
STOP

3. STORAGE-DIAGRAM



A. NECESSARY PARAMETERS FOR STORAGE SECTIONS

1. BASE ADDRESSES
2. DIRECTIONS OF GROWTH
3. LENGTHS OF SPACE
4. BOUNDS OF USED SPACE

B. DEFINITIONS

1. SYMBOL TABLE (USER)

A). SYMTBL- POINTER TO BEGINNING OF SYMBOL TABLE SPACE

B). LISYMTBL- LENGTH OF SYMBOL TABLE SPACE (POINTS TO LAST WORD OF STORAGE SECTION)

C). BISYMTBL- POINTER TO END OF USED SPACE IN SYMBOL TABLE

D). TISYMBOL- THE VALUE 6. USED IN PAGES

2. SYMBOL TABLE (SYSTEM)

A). SSYMTBL- POINTER TO BEGINNING OF SYMBOL TABLE SPACE

B). BISSYMTBL- POINTER TO END OF USED SPACE IN SYMBOL TABLE

3. STRING POINTER TABLE STORAGE

A). SPT- POINTER TO BEGINNING OF STRING POINTER TABLE STORAGE SPACE

B). LISPT- LENGTH OF STORAGE SPACE

C). BISPT- POINTER TO END OF USED SPACE

D). TISPT- THE VALUE 1. USED IN PAGES

4. STRING POINTER TABLE STORAGE (SYSTEM)

SSPT- LOCATION OF THE BEGINNING OF SYSTEM STRING POINTER TABLE STORAGE SPACE.

5. STRING STORAGE (USER)

A). STRINGS- POINTER TO BEGINNING OF STRING STORAGE SPACE

B). LISTRINGS- LENGTH OF STORAGE SPACE

C). BISTRINGS- POINTER TO END OF USED SPACE

D). TISTRINGS- THE VALUE 2. USED IN PAGES

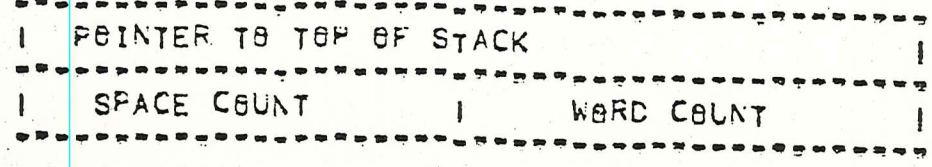
D. STRINGB- BASE ADDRESS IN BYTES OF USER STRING STORAGE

6. STRINGS (SYSTEM)

SSTRINGS- LOCATION OF THE BEGINNING OF SYSTEM STRING STORAGE SPACE

7. STACKS

A). SSTACK- STACK POINTER DOUBLEWORD (AS DEFINED IN SIGMA-7 REFERENCE MANUAL, P. 66)



1). TISSTACK- THE VALUE 4. USED IN PAGES

2). SISSTACK- A COPY OF THE ORIGINAL SPD, FOR STACK CLEAN-UP.

B). PSTACK- STACK POINTER DOUBLEWORD

1). TIPSTACK- THE VALUE 5. USED IN PAGES

2). SIPSTACK- A COPY OF THE ORIGINAL SPD, FOR STACK CLEAN-UP.

C). RSTACK- STACK POINTER DOUBLEWORD FOR

1). TIRSTACK- THE VALUE 3. USED IN PAGES

2). SIRSTACK- A COPY OF THE ORIGINAL SPD, FOR STACK CLEAN-UP.
INTERNAL STACK FOR SAVING REGISTERS

8. SYSPROCS -- POINTER TO THE BEGINNING OF THE SYSTEM PROCEDURE AND COMMAND SPACE.

C. DETERMINING UNUSED SPACE

1. THE UNUSED SPACE IN EACH STORAGE SECTION FOLLOWS THE B:SYMTBL, B:SPT, OR THE B:STRINGS; OR IS ABOVE THE TOP OF THE STACK.

2. A DEALLOCATED STORAGE SECTION'S ENTRY SPACE IS MARKED WITH ALL ONES (FF OR FFFF) IN ITS CHARACTER OR WORD COUNT BY THE RETURN ROUTINE. THIS DOES NOT APPLY TO STACKS.

3. A 256-BYTE TABLE, PAGES, WILL BE MAINTAINED TO DETERMINE WHICH PAGES HAVE BEEN REQUESTED FROM THE OPERATING SYSTEM, AND HOW THEY ARE BEING USED.

IV. STORAGE MANAGEMENT SUPPLIES OTHER MODULES WITH:

A. POINTERS TO ALLOCATED SPACE

B. POINTERS TO RESULTS OF OPERATIONS WHICH REQUIRE STORAGE MANIPULATION (I.E. WORD, FIRST, ETC.)

C. BASE ADDRESSES FOR EACH STORAGE SECTION

V. OTHER MODULES SUPPLY STORAGE MANAGEMENT WITH:

A. POINTERS TO SPACE TO BE DEALLOCATED

B. POINTERS TO INPUTS OF OPERATIONS WHICH REQUIRE STORAGE MANIPULATION

C. NAMES FOR ACCESSING THE SYMBOL TABLE

D. ELEMENTS DESCRIBING THEIR REQUEST (I.E. LENGTH, TYPE, ETC.)

ENTRY POINTS AND FUNCTIONS

I. DEFINITIONS AND COMMENTS

A. ALL ROUTINES WILL BE IMPLEMENTED AS SUBROUTINES WITH MACROS TO PUT THE ARGUMENTS INTO THE REGISTERS AND CALL THE SUBROUTINE. ALL REGISTERS WILL BE SAVED.

B. A 'TYPE' IS A BINARY WORD CONTAINING THE TYPE CODE FOR THE SYMBOL IN QUESTION.

C. A 'SYMPTR' IS A SPTPTR TO THE SENTENCE (OR WORD) WHICH IS TO BE USED AS A NAME.

D. A 'SPTPTR' IS A POINTER TO THE WORD OF AN SPT (DEFINED PREVIOUSLY) CONTAINING THE COUNTS. THE TOP HALF WORD CONTAINS THE TYPE AND NUMBER OF ARGUMENTS WHERE APPLICABLE.

E. A 'LENGTH' IS A BINARY NUMBER SPECIFYING A LENGTH IN BYTES.

F. A 'COUNT' IS A BINARY NUMBER SPECIFYING A NUMBER OF WORDS.

G. A 'STRPTR' IS A POINTER TO A STRING (PREVIOUSLY DEFINED) IN STRING STORAGE.

H. A 'POINTER' IS A DISPLACEMENT FROM THE BASE ADDRESS OF THE STRUCTURE IN QUESTION.

I. INDIRECT ADDRESSING IS ALLOWED FOR ALL ARGUMENTS.

J. THE 'CONDITION CODES' ARE AS DEFINED IN THE XEROX SIGMA-7 HARDWARE REFERENCE MANUAL.

K. BRACKETS (<>) SURROUNDING AN ARGUMENT INDICATE THAT THE ARGUMENT MAY BE OMITTED.

II. PROCEDURE DESCRIPTIONS:

A. FINDTHG SYMPTR, TYPE, SPTPTR
- SEARCH THE SYMBOL TABLES (USER FIRST) FOR THE FIRST OCCURENCE OF THE SYMBOL INDICATED BY SYMPTR, RETURN WITH SPTPTR POINTING TO THE APPROPRIATE SPT (IF FOUND) AND THE CONDITION CODES INDICATING WHETHER THE SYMBOL WAS FOUND OR NOT.

B. ADDTHG SYMPTR, SPTPTR, TYPE
- ADD THE SYMBOL INDICATED BY SYMPTR TO THE SYMBOL TAELE WITH SPTPTR INDICATING THE NECESSARY PARAMETERS (THERE IS NO FAILURE CONDITION). TYPE CODE IS IN TYPE.

C. DELTHG TYPE, SYMPTR
- REMOVE THE INDICATED SYMBOL FROM THE SYMBOL TABLE, ALSO DEALLOCATE THE ASSOCIATED SPT AND STRINGS. RETURN WITH CONDITION CODES INDICATING SUCCESS OR FAILURE, AND IN THE CASE OF FAILURE WHETHER THE SPECIFIED SYMBOL WAS A SYSTEM SYMBOL OR A USER SYMBOL.

D. ALLOCSPT <COUNT>, SPTPTR
- ALLOCATE AN SPT WITHOUT A SYMBOL NAME AND WITH A USE COUNT AS INDICATED OR OF ZERO. RETURN A POINTER TO IT IN SPTPTR.

E. RTNSPT SPTPTR
- DEALLOCATE THE SPT AND ASSOCIATED STRINGS INDICATED BY SPTPTR.

F. SRTSPT SPTPTR
- SORTS A FUNCTION SPT ACCORDING TO THE LINE LINE NUMBERS (A LINE NUMBER OF -1 INDICATES THAT THE LINE IS TO BE REMOVED).

G. ALLOCSTR LENGTH,<SPTPTR>,STRPTR

- ALLOCATE A STRING WITH THE INDICATED LENGTH AND RETURN WITH A POINTER TO THAT STRING IN STRPTR. IF SPTPTR IS SPECIFIED, ADD A POINTER TO THE STRING AT THE END OF THE SPECIFIED SPT.

H. RTNSTR STRPTR

- DEALLOCATE THE STRING INDICATED BY STRPTR.

I. IWORD SPTPTR1,SPTPTR2,SPTPTR3

- IF THE USE COUNT IN BOTH SPTPTR1 AND SPTPTR2 IS 1, RETURN IN SPTPTR3 A POINTER TO AN SPT WITH USE COUNT OF 1 WHICH POINTS TO A STRING WHICH IS THE CONCATENATION OF THE STRINGS POINTED TO BY SPTPTR1 AND SPTPTR2. IF THE USE COUNT OF SPTPTR1 AND SPTPTR2 IS NOT 1, RETURN WITH THE CONDITION CODES INDICATING THAT THIS HAPPENED.

J. ISENTENCE SPTPTR1,SPTPTR2,SPTPTR3

- RETURN IN SPTPTR3 A POINTER TO AN SPT WHICH IS THE CONCATENATION OF SPTPTR1 AND SPTPTR2.

K. IFIRST SPTPTR1,SPTPTR2

- IF THE USE COUNT IN SPTPTR1 IS 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO A STRING WHICH IS THE FIRST CHARACTER IN THE STRING INDICATED BY SPTPTR1. IF THE USE COUNT IN SPTPTR1 IS GREATER THAN 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO A STRING WHICH IS THE FIRST STRING IN SPTPTR1.

L. ILAST SPTPTR1,SPTPTR2

- IF THE USE COUNT IN SPTPTR1 IS 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO A STRING WHICH IS THE LAST CHARACTER IN THE STRING INDICATED BY SPTPTR1. IF THE USE COUNT IN SPTPTR1 IS GREATER THAN 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO A STRING WHICH IS THE LAST STRING IN SPTPTR1.

M. IBUTFIRST SPTPTR1,SPTPTR2

- IF THE USE COUNT IN SPTPTR1 IS 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO A STRING WHICH IS ALL BUT THE FIRST CHARACTER IN THE STRING INDICATED BY SPTPTR1, IF THE USE COUNT IN SPTPTR1 IS GREATER THAN 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO STRINGS WHICH ARE ALL BUT THE FIRST STRING IN SPTPTR1.

N. IBUTLAST SPTPTR1,SPTPTR2

- IF THE USE COUNT IN SPTPTR1 IS 1, RETURN IN SPTPTR2 A POINTER TO AN SPT WHICH POINTS TO A STRING WHICH IS ALL BUT THE LAST CHARACTER IN THE STRING INDICATED BY SPTPTR1, IF THE USE COUNT IN SPTPTR1 IS GREATER THAN 1, RETURN IN SPTPTR2 POINTER TO AN SPT WHICH POINTS TO STRINGS WHICH ARE ALL BUT THE FIRST STRING IN SPTPTR1.

O. INITNXT TYPE

-INITIALIZES NXTTYP TO TYPE SPECIFIED, AND INITIALIZES NXTPTR TO END OF SYMBOL TABLE. THIS ROUTINE MUST BE CALLED BEFORE ENTERING FINDNXT FOR THE FIRST TIME.

P. FINDNXT SPTPTR,SYMPTR

-SEARCHES FOR AN OCCURRENCE OF TYPE NXTTYP STARTING AT NXTPTR AND RETURNS THE CORRESPONDING SPTPTR AND SYMPTR.

Q. STORINIT
-SETS UP STORAGE SECTIONS. REQUESTS PAGES AND INITIALIZES BASE ADDRESSES, ETC.

R. STACKINIT
-INITIALIZES STACK POINTER DOUBLEWORDS AND RELEASES THE SPT'S INDICATED BY THE CONTENTS OF THE SSTACK.

S. STACKTR
-CALLS GETPAGE TO ACQUIRE A PAGE AT THE END OF THE STACK WHICH OVERFLOWED AND ADDS 512 TO THE SPACE COUNT OF THE STACK POINTER DOUBLEWORD. SYSERR IS CALLED IF STACK UNDERFLOW OCCURRED OR STACK OVERFLOW OCCURRED ON OTHER THAN PSTACK, RSTACK, OR SSTACK.

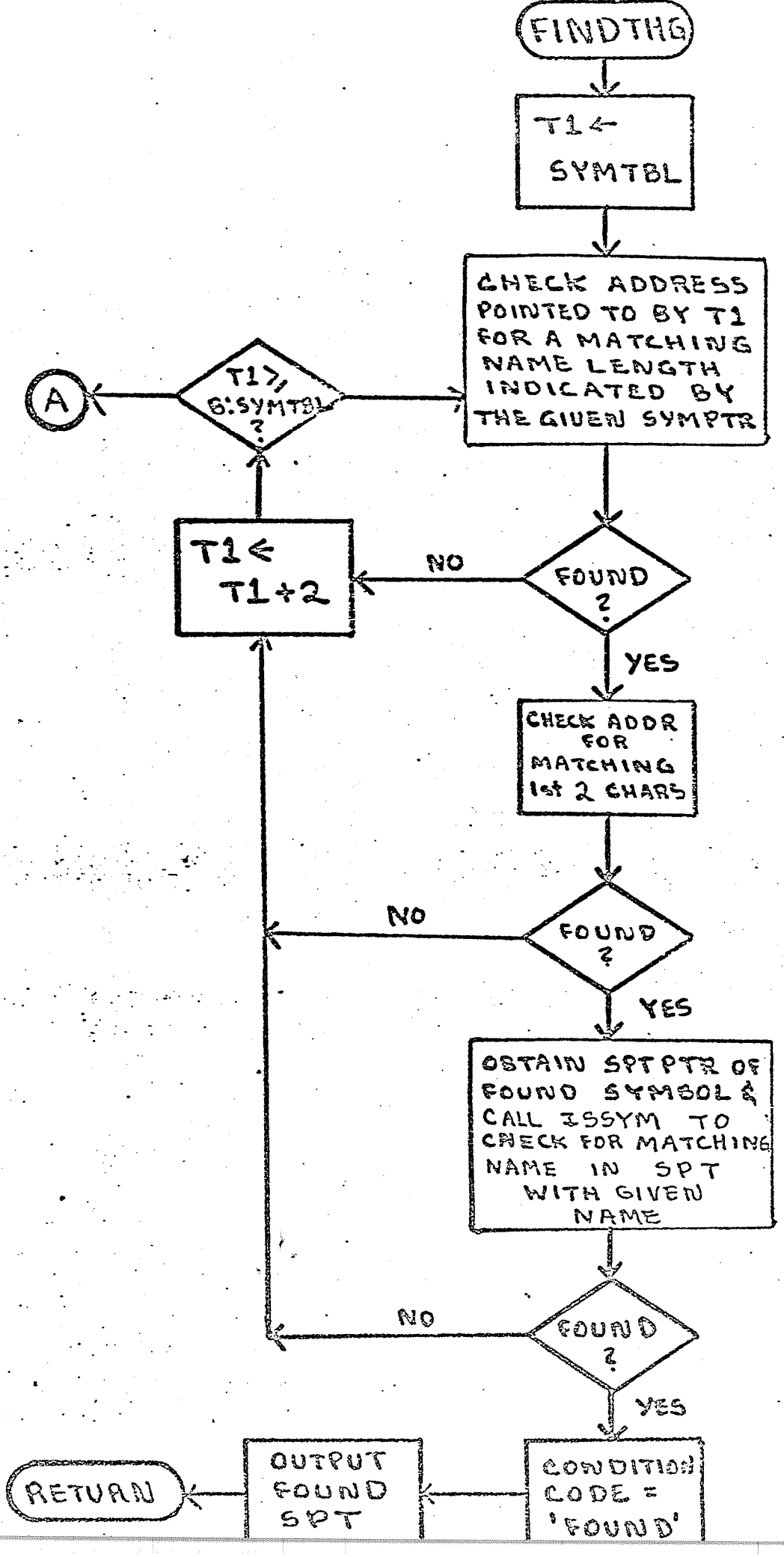
T. COMPACT
-CHECKS COMPCFLG, IF NOT SET RETURN, IF SET RETURN, IF SET COMPACTS SPT AND STRING STORAGE.

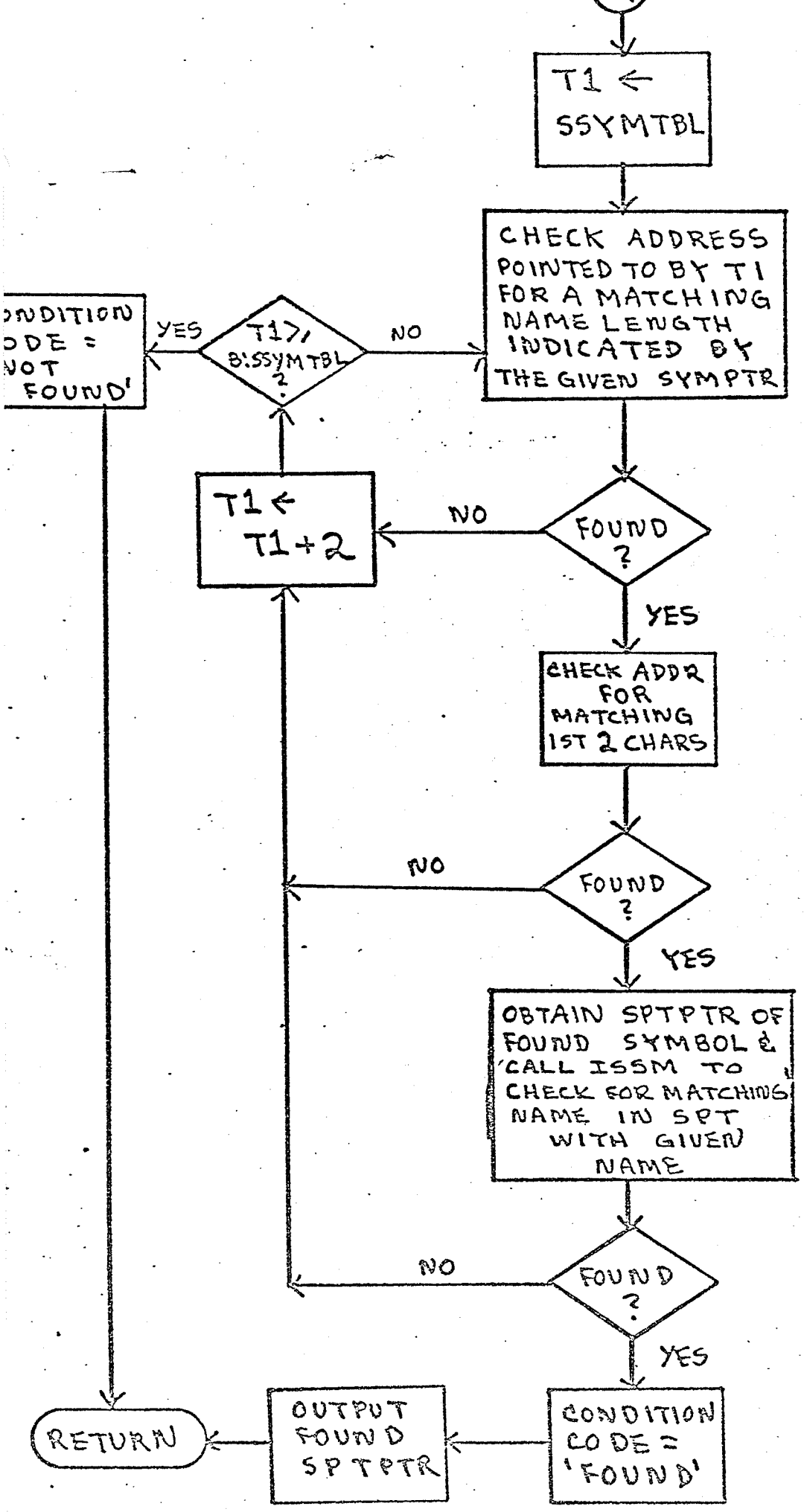
U. SYSTHG ('NAME1', . . . , 'NAMEN'), ('WORD1', . . . , 'WORDN')
-GENERATES A LOGO SENTENCE MADE UP OF THE WORDS 'WORD1', 'WORD2', . . . , 'WORDN' WITH THE NAMES 'NAME1', . . . , 'NAMEN'.

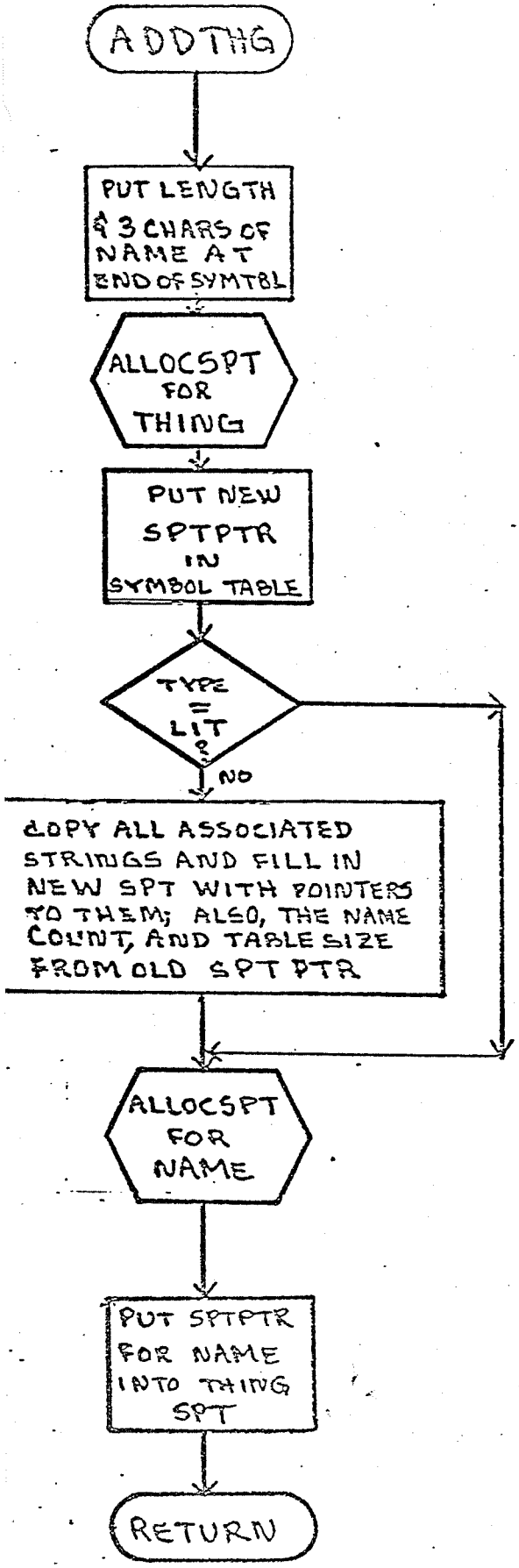
V. SYSPROC 'NAME', INPUTS, PROCEDURE
-GENERATES ENTRIES IN SSYMTBL AND SSPT FOR A SYSTEM PROCEDURE WITH NAME 'NAME', INPUTS NUMBER OF INPUTS, AND ENTRY POINT AT PROCEDURE.

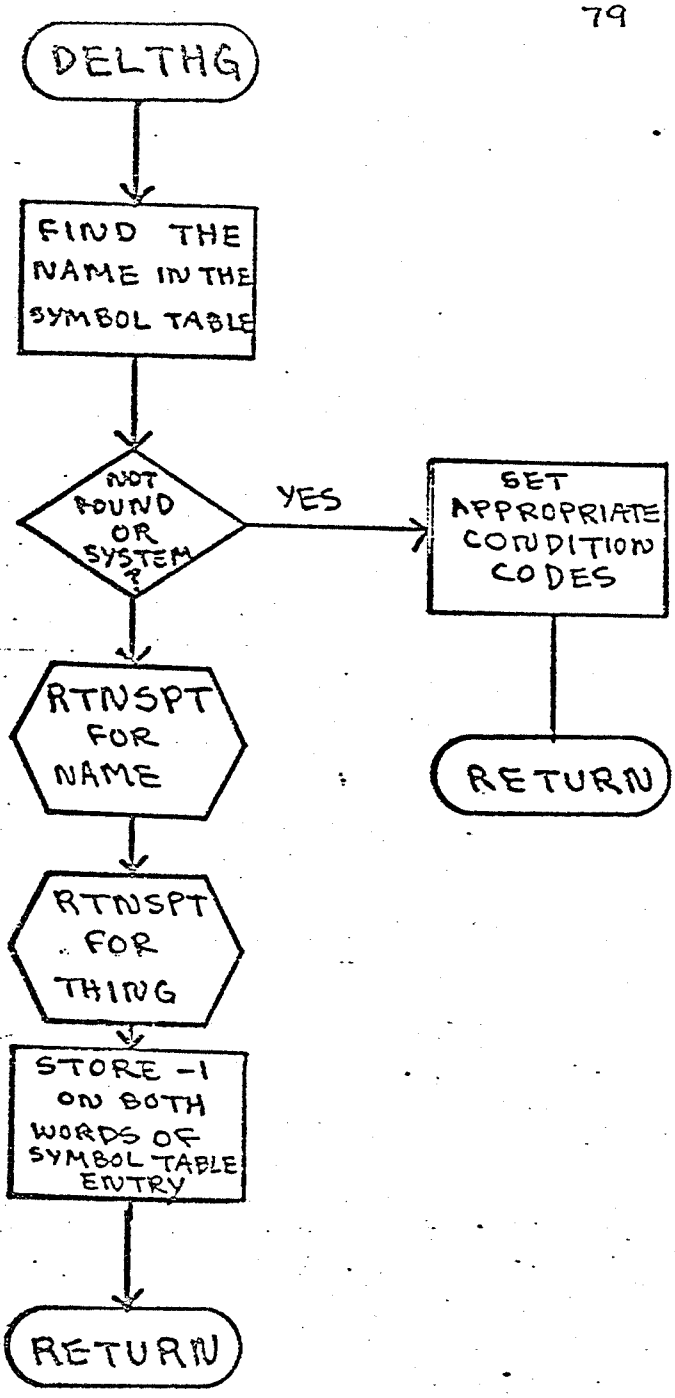
W. NIKPROC 'NAME', PROCEDURE
-GENERATES ENTRIES IN SSYMTBL AND SSPT FOR FOR A NICKTYP WITH NAME 'NAME' AND ENTRY POINT AT PROCEDURE.

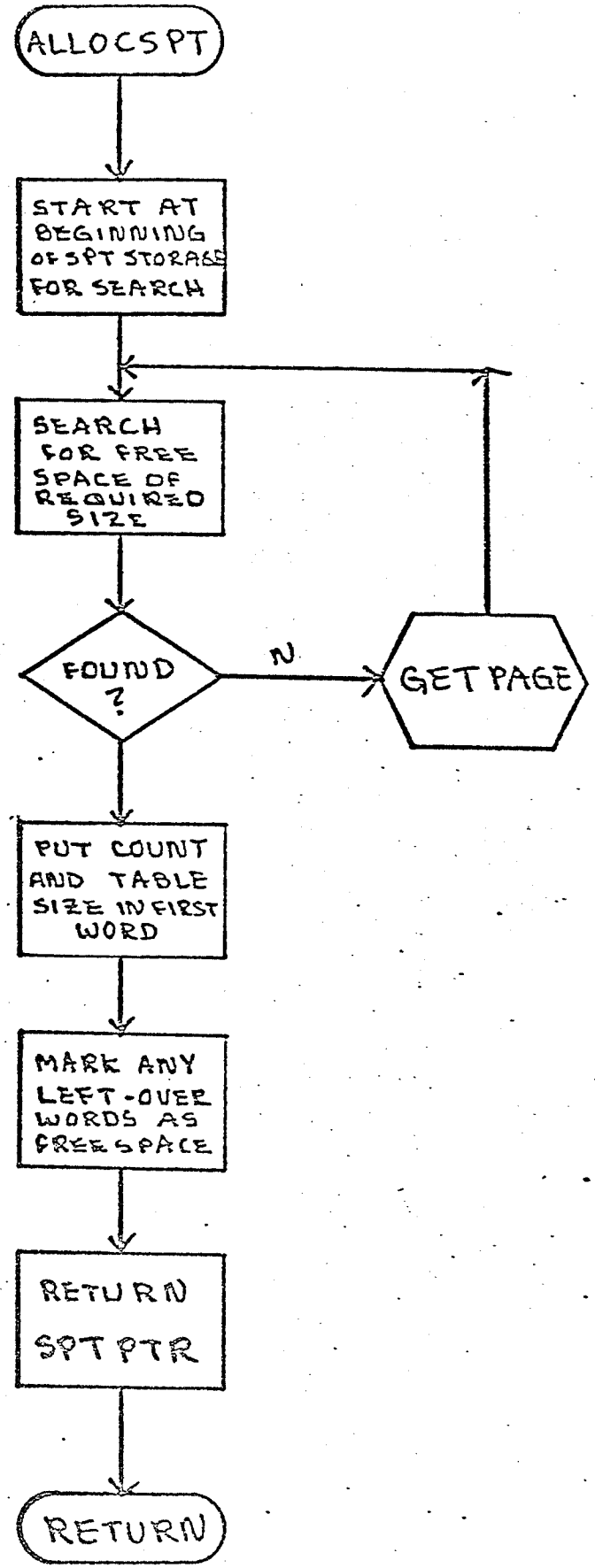
X. BINCHAR BIN, CHARPTR
-CONVERTS A BINARY NUMBER CONTAINED IN BIN TO A STRING OF CHARACTERS POINTED TO BY CHARPTR. CHARPTR IS RETURNED CONTAINING THE ADDRESS OF THE STRING PRECEEDED BY A HALFWORD CHARACTER LENGTH.

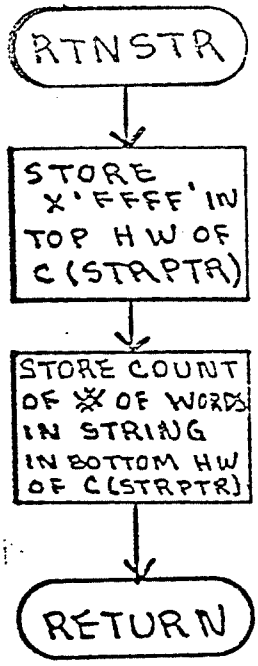

FINDTHG
T1 ← SYMTBL
CHECK ADDRESS POINTED TO BY T1 FOR A MATCHING NAME LENGTH INDICATED BY THE GIVEN SYMPTR
T1 > E.SYMTBL ?
A
T1 ← T1+2
NO
FOUND ?
YES
CHECK ADDR FOR MATCHING 1st 2 CHARS
NO
FOUND ?
YES
OBTAIN SPTPTR OF FOUND SYMBOL & CALL ISSYM TO CHECK FOR MATCHING NAME IN SPT WITH GIVEN NAME
NO
FOUND ?
YES
CONDITION CODE = 'FOUND'
OUTPUT FOUND SPT
RETURN

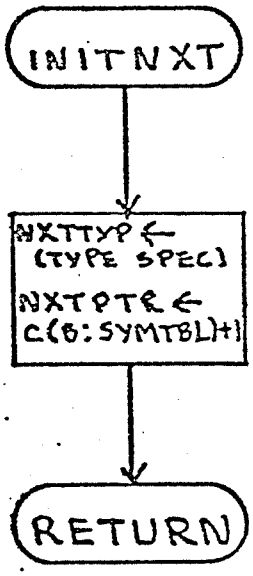

T1 ← SSYMTBL
CHECK ADDRESS POINTED TO BY T1 FOR A MATCHING NAME LENGTH INDICATED BY THE GIVEN SYMPTR
T1 > B'SSYMTBL ?
YES
NO
ONDITION ODE = NOT FOUND'
T1 ← T1+2
NO
FOUND ?
YES
CHECK ADDR FOR MATCHING 1ST 2 CHARS
NO
FOUND ?
YES
OBTAIN SPTPTR OF FOUND SYMBOL & CALL ISSM TO CHECK FOR MATCHING NAME IN SPT WITH GIVEN NAME
NO
FOUND ?
YES
CONDITION CODE = 'FOUND'
OUTPUT FOUND SPTPTR
RETURN

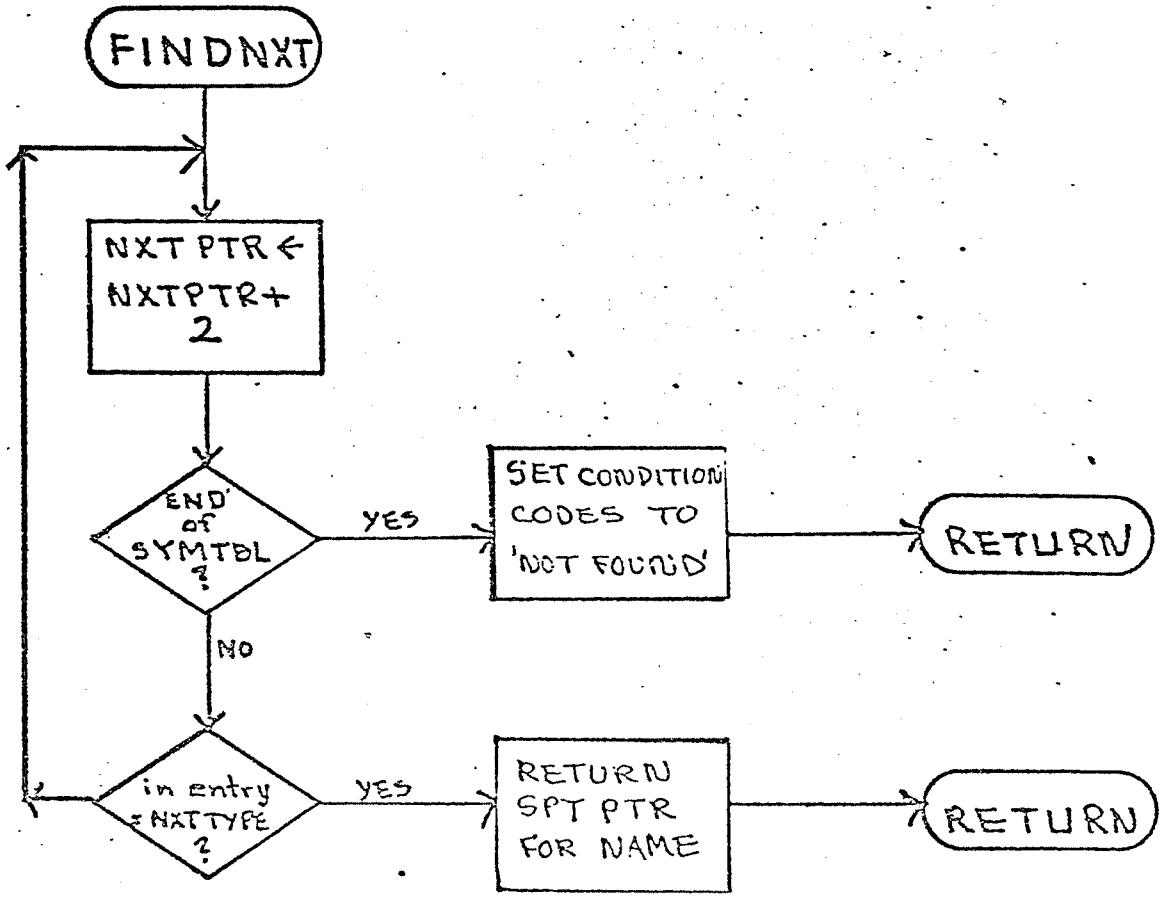

ADDTHG
PUT LENGTH & 3 CHARS OF NAME AT END OF SYMTBL
ALLOCSPT FOR THING
PUT NEW SPTPTR IN SYMBOL TABLE
TYPE = LIT ?
NO
COPY ALL ASSOCIATED STRINGS AND FILL IN NEW SPT WITH POINTERS TO THEM; ALSO, THE NAME COUNT, AND TABLE SIZE FROM OLD SPT PTR
ALLOCSPT FOR NAME
PUT SPTPTR FOR NAME INTO THING SPT
RETURN

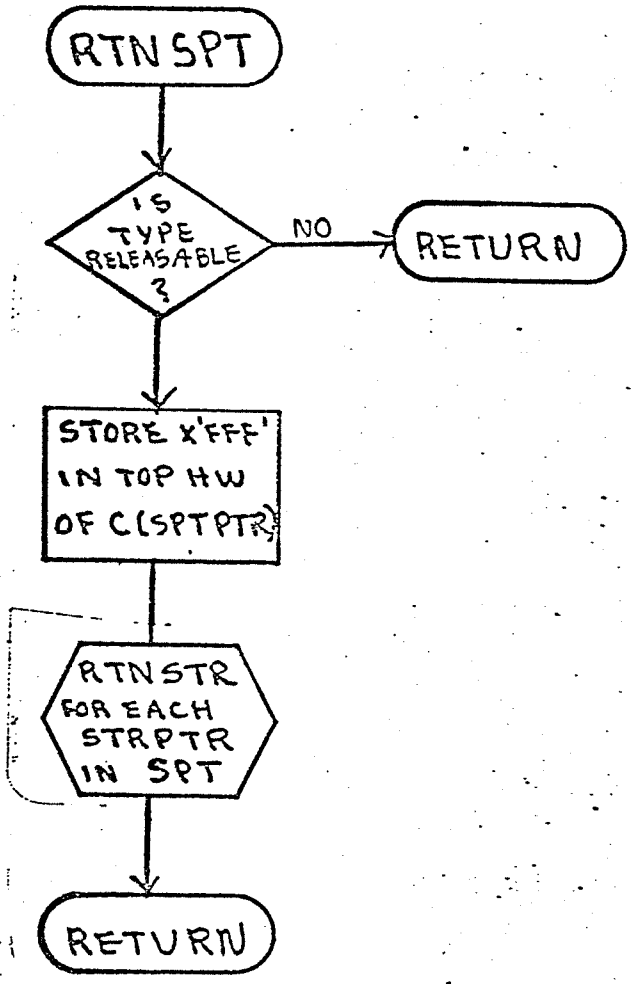

DELTHG
FIND THE NAME IN THE SYMBOL TABLE
NOT FOUND OR SYSTEM ?
YES
SET APPROPRIATE CONDITION CODES
RETURN
RTNSPT FOR NAME
RTNSPT FOR THING
STORE -1 ON BOTH WORDS OF SYMBOL TABLE ENTRY
RETURN

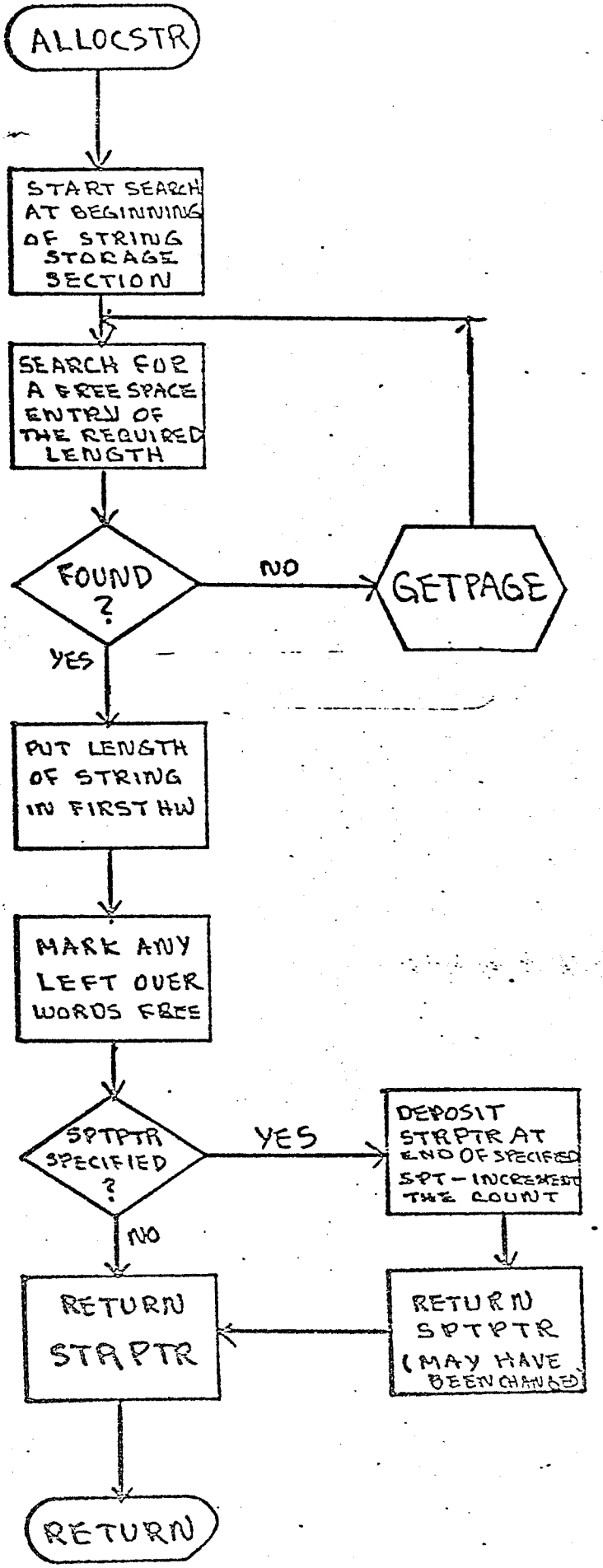

ALLOCSPT
START AT BEGINNING OF SPT STORAGE FOR SEARCH
SEARCH FOR FREE SPACE OF REQUIRED SIZE
FOUND ?
N
GET PAGE
PUT COUNT AND TABLE SIZE IN FIRST WORD
MARK ANY LEFT-OVER WORDS AS FREESPACE
RETURN SPT PTR
RETURN

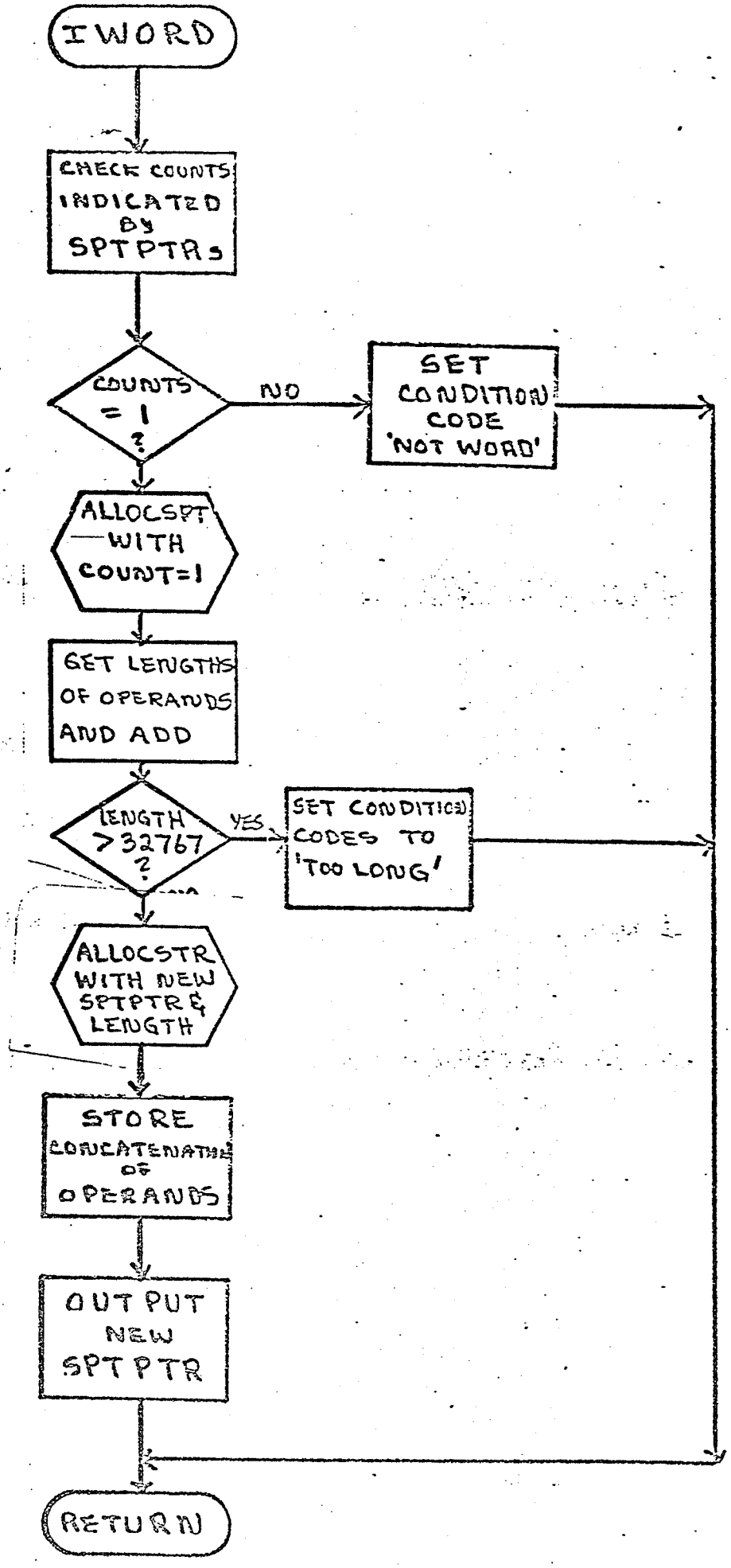

RTNSTR
STORE X'FFFF' IN TOP HW OF C(STAPTR)
STORE COUNT OF # OF WORDS IN STRING IN BOTTOM HW OF C(STRPTR)
RETURN

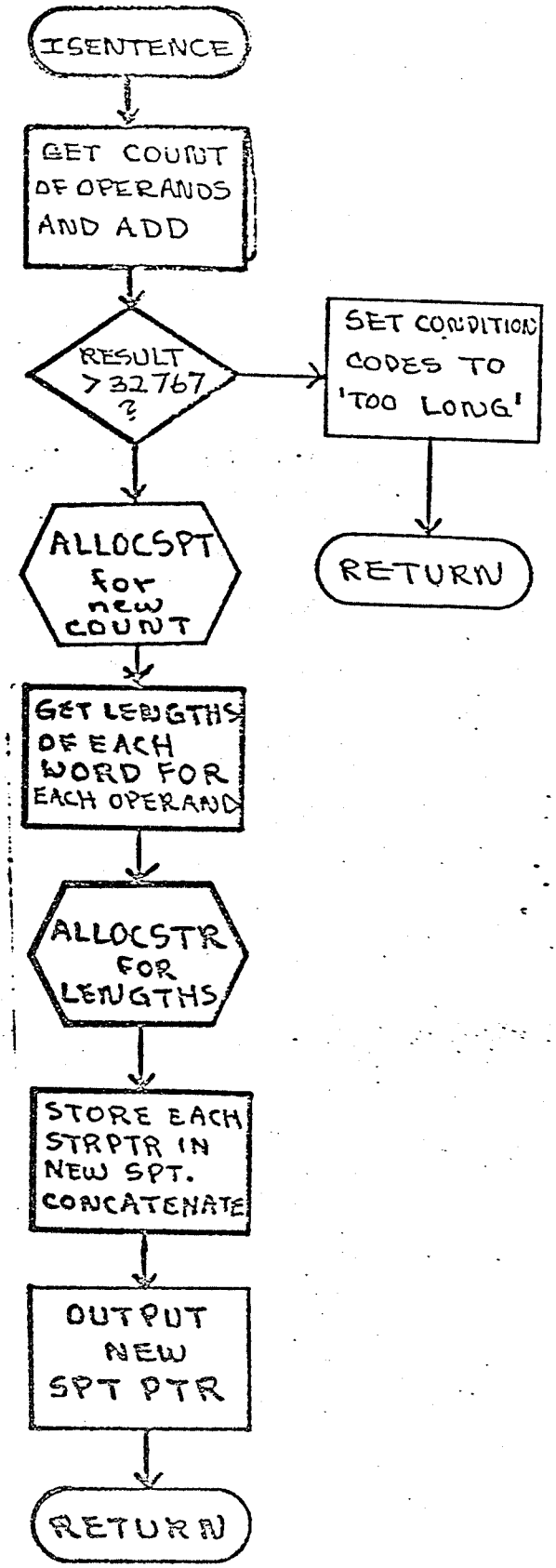

INITNXT
NXTTYP ← (TYPE SPEC)
NXTPTR ← C(6:SYMTBL)+1
RETURN

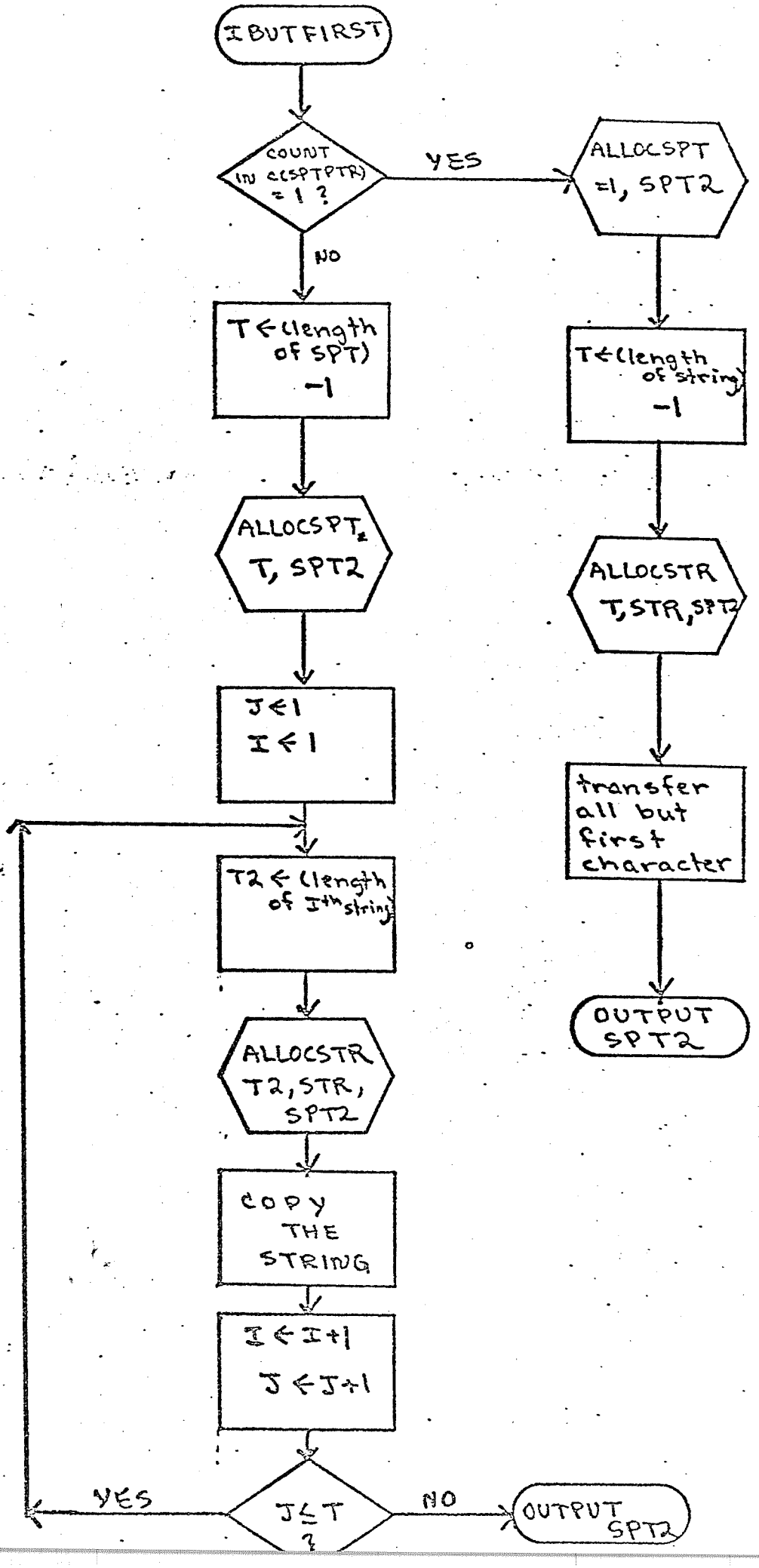

FINDNXT
NXT PTR ← NXTPTR+ 2
END of SYMTBL ?
YES
SET CONDITION CODES TO 'NOT FOUND'
RETURN
NO
in entry = NXTTYPE ?
YES
RETURN SPT PTR FOR NAME
RETURN

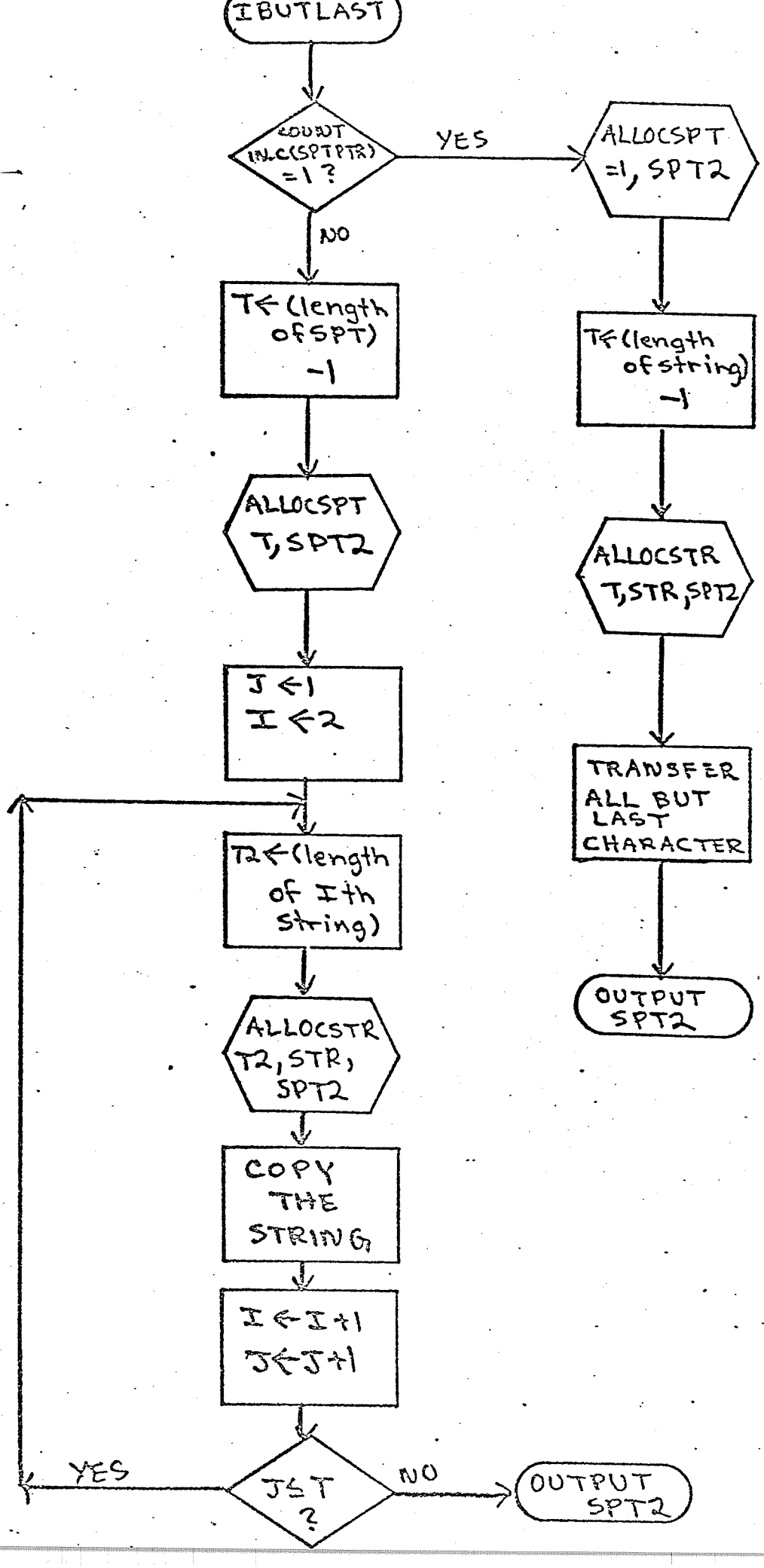

RTN SPT
IS TYPE RELEASABLE ?
NO
RETURN
STORE X'FFF' IN TOP HW OF C(SPTPTR)
RTN STR FOR EACH STRPTR IN SPT
RETURN

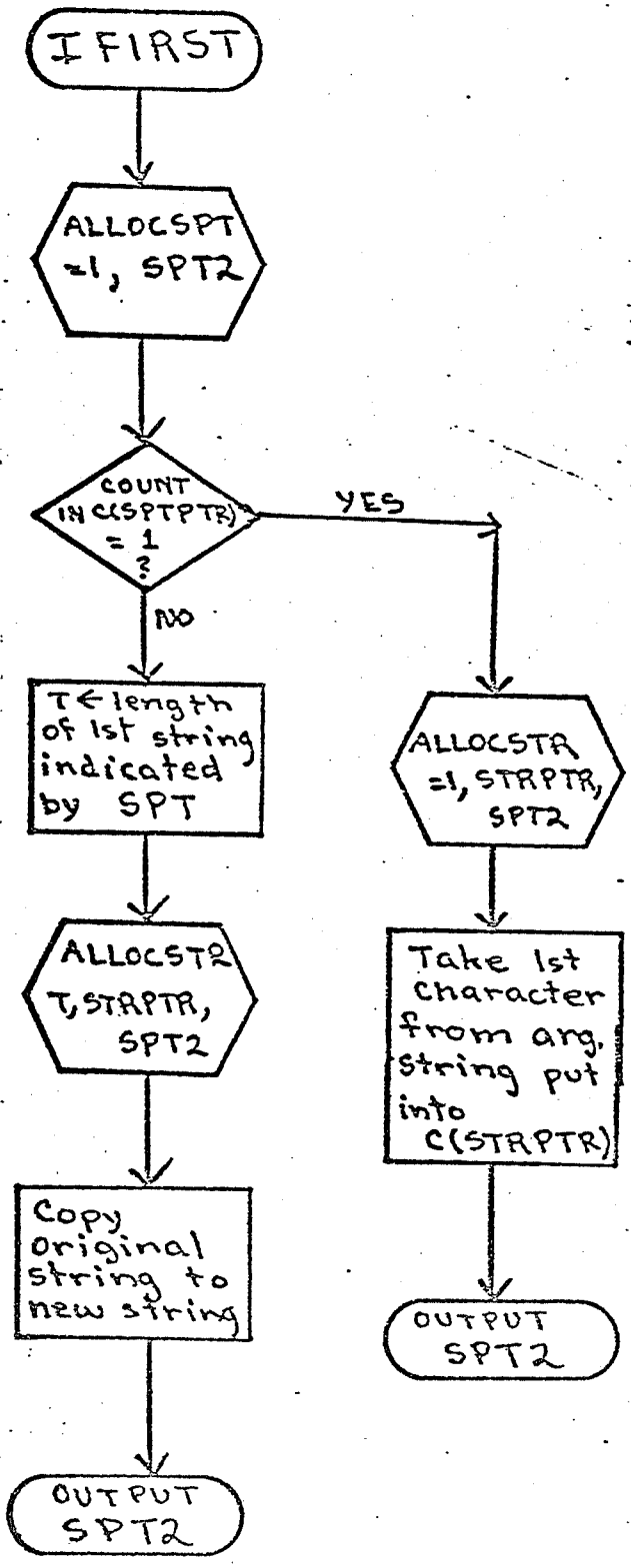

ALLOCSTR
START SEARCH AT BEGINNING OF STRING STORAGE SECTION
SEARCH FOR A FREE SPACE ENTRY OF THE REQUIRED LENGTH
FOUND ?
NO
GETPAGE
YES
PUT LENGTH OF STRING IN FIRST HW
MARK ANY LEFT OVER WORDS FREE
SPTPTR SPECIFIED ?
YES
DEPOSIT STRPTR AT END OF SPECIFIED SPT - INCREMENT THE COUNT
NO
RETURN STRPTR
RETURN SPTPTR (MAY HAVE BEEN CHANGED)
RETURN

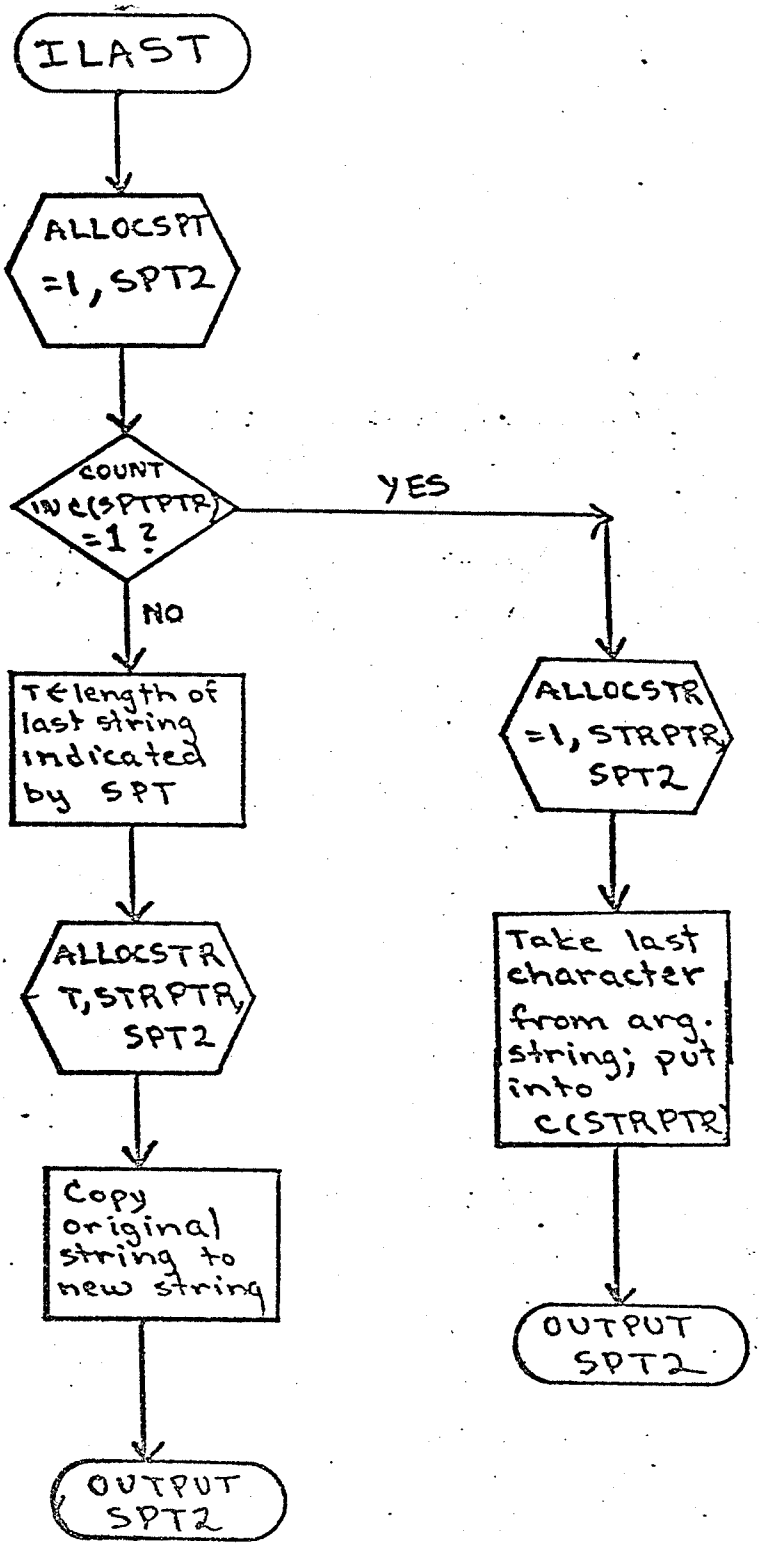

IWORD
CHECK COUNTS INDICATED BY SPTPTRs
COUNTS = 1 ?
NO
SET CONDITION CODE 'NOT WORD'
ALLOCSPT WITH COUNT=1
GET LENGTHS OF OPERANDS AND ADD
LENGTH > 32767 ?
YES
SET CONDITION CODES TO 'TOO LONG'
ALLOCSTR WITH NEW SPTPTR & LENGTH
STORE CONCATENATION OF OPERANDS
OUTPUT NEW SPTPTR
RETURN

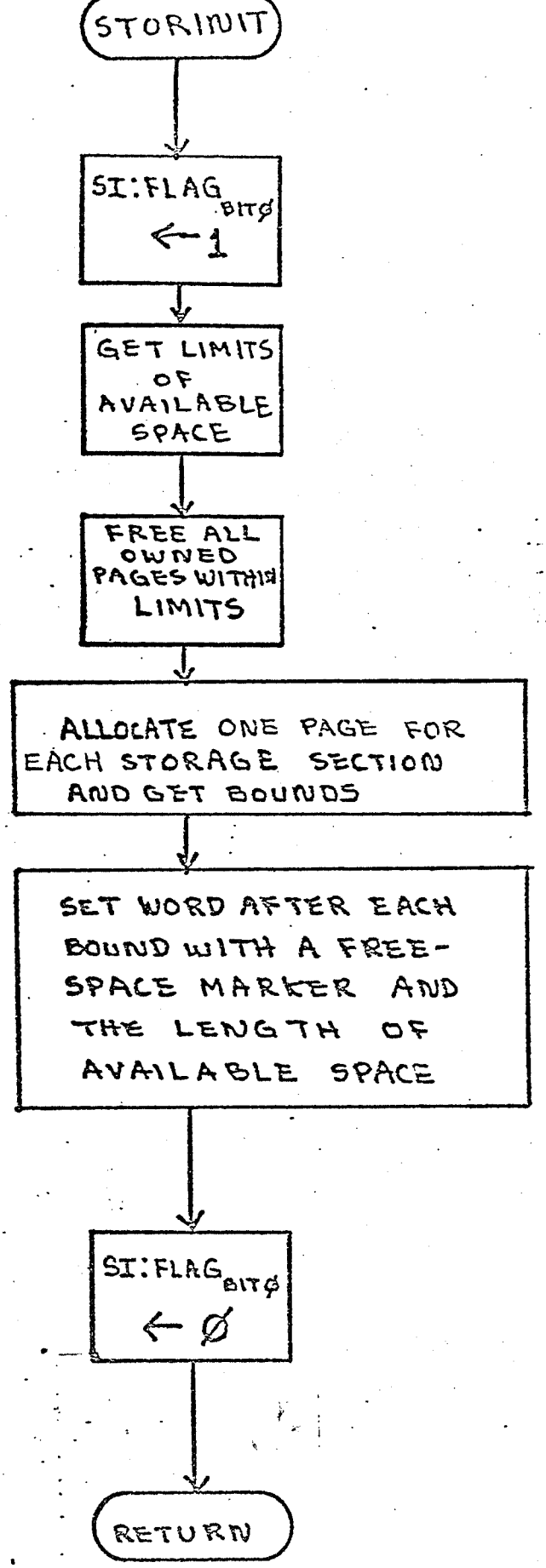

ISENTENCE
GET COUNT OF OPERANDS AND ADD
RESULT > 32767 ?
SET CONDITION CODES TO 'TOO LONG'
RETURN
ALLOCSPT for new COUNT
GET LENGTHS OF EACH WORD FOR EACH OPERAND
ALLOCSTR FOR LENGTHS
STORE EACH STRPTR IN NEW SPT. CONCATENATE
OUTPUT NEW SPT PTR
RETURN

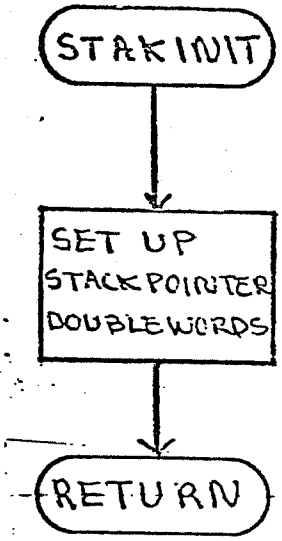

IBUTFIRST
COUNT IN C(SPTPTR) = 1 ?
YES
NO
ALLOCSPT =1, SPT2
T←(length of string) -1
ALLOCSTR T, STR, SPT2
transfer all but first character
OUTPUT SPT2
T←(length of SPT) -1
ALLOCSPT T, SPT2
J←1
I←1
T2←(length of Ith string)
ALLOCSTR T2, STR, SPT2
COPY THE STRING
I←I+1
J←J+1
J≤T ?
YES
NO
OUTPUT SPT2

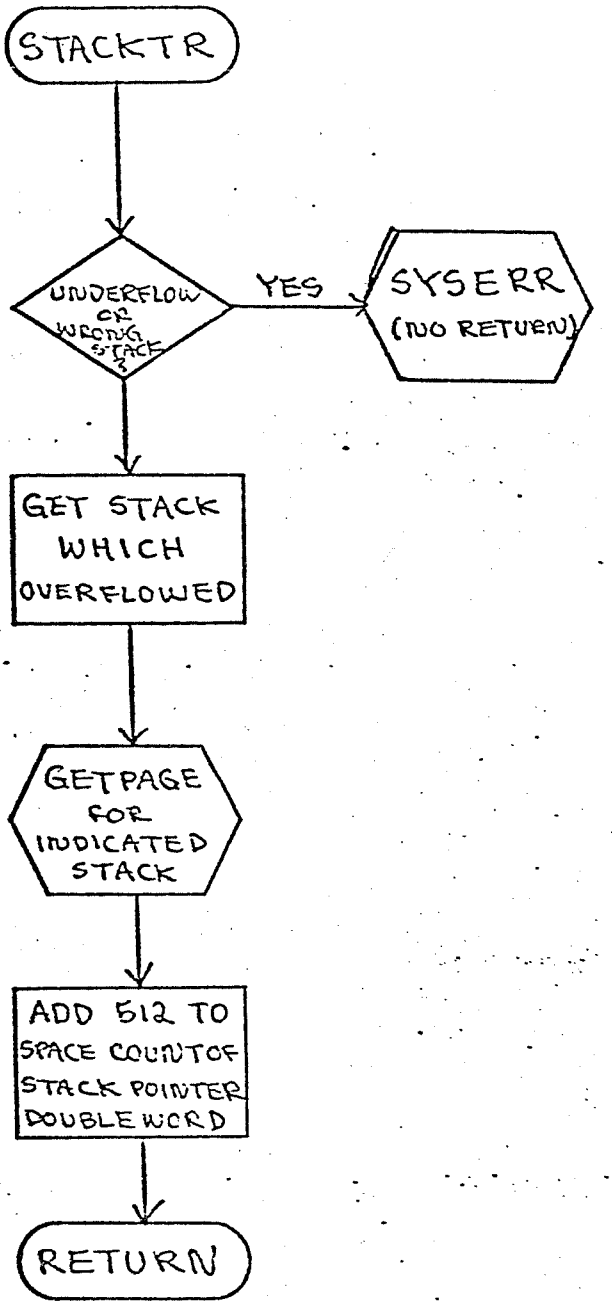

IBUTLAST
COUNT IN.C(SPTPTR) =1 ?
YES
NO
ALLOCSPT =1, SPT2
T←(length of SPT) -1
T←(length of string) -1
ALLOCSPT T, SPT2
ALLOCSTR T, STR, SPT2
J←1 I←2
TRANSFER ALL BUT LAST CHARACTER
T2←(length of Ith string)
OUTPUT SPT2
ALLOCSTR T2, STR, SPT2
COPY THE STRING
I←I+1 J←J+1
YES
J≤T ?
NO
OUTPUT SPT2


I FIRST
ALLOCSPT =1, SPT2
COUNT IN C(SPTPTR) = 1 ?
YES
NO
T←length of 1st string indicated by SPT
ALLOCSTR =1, STRPTR, SPT2
ALLOCSTR T, STRPTR, SPT2
Take 1st character from arg. string put into C(STRPTR)
Copy original string to new string
OUTPUT SPT2
OUTPUT SPT2


ILAST
ALLOCSPT
=1, SPT2
COUNT
IN C(SPTPTR)
=1 ?
YES
NO
T←length of
last string
indicated
by SPT
ALLOCSTR
T, STRPTR,
SPT2
Copy
original
string to
new string
OUTPUT
SPT2
ALLOCSTR
=1, STRPTR,
SPT2
Take last
character
from arg.
string; put
into
C(STRPTR)
OUTPUT
SPT2


STORINIT
SI:FLAG BIT∅ ← 1
GET LIMITS OF AVAILABLE SPACE
FREE ALL OWNED PAGES WITHIN LIMITS
ALLOCATE ONE PAGE FOR EACH STORAGE SECTION AND GET BOUNDS
SET WORD AFTER EACH BOUND WITH A FREE-SPACE MARKER AND THE LENGTH OF AVAILABLE SPACE
SI:FLAG BIT∅ ← ∅
RETURN


STAKINIT
SET UP STACK POINTER DOUBLEWORDS
RETURN


STACKTR
UNDERFLOW OR WRONG STACK?
YES
SYSERR (NO RETUEN)
GET STACK WHICH OVERFLOWED
GETPAGE FOR INDICATED STACK
ADD 512 TO SPACE COUNT OF STACK POINTER DOUBLEWORD
RETURN

ROUTINES CALLED

I. SYSERR- WHEN DIASTER STRIKES (I.E. NO MEMORY)

INTERMODULE COMMUNICATIONS

I. BAL ON REGISTER 11

II. USE REGISTERS 6,7,AND,8 FOR PASSING ARGUMENTS.

DATA STRUCTURES

I. DATA STRUCTURES PREVIOUSLY DEFINED IN DOCUMENT

SUBROUTINE LIST

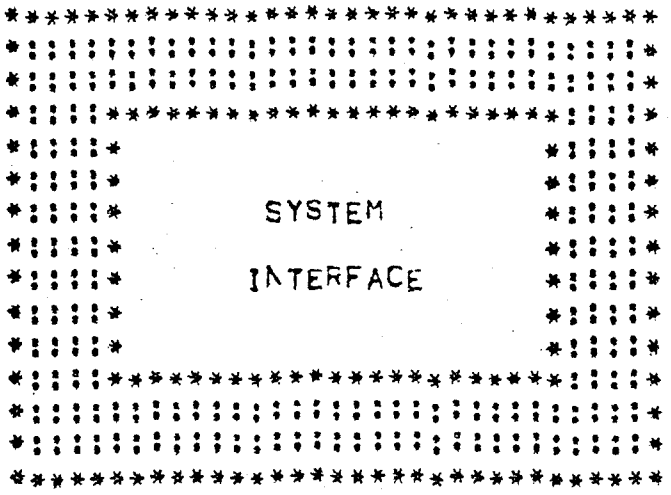
I. COPY- COPIES NUMBER OF BYTES(8) FROM BYTE ADDRESS (6) TO BYTE ADDRESS(7)

II. EXPSPT- DEPOSITS STRPTR(7) AT THE END OF THE TABLE POINTED TO BY SPTPTR(6). (WILL CREATE A NEW LARGER SPT IF NECESSARY)

III. GETPAGE- TAKES PAGE ADDRESS(6) AND REQUESTS THAT PAGE FROM THE MONITOR, IF POSSIBLE. SHUFFLES STORAGE, IF NECESSARY, TO MAKE THAT PAGE AVAILABLE.

IV. SYMCPCT- CALLED BY GETPAGE TO COMPACT THE SYMBOL TABLE. CALLS FREEPAGE TO RELEASE UNNEEDED PAGES.

V. FREEPAGE- RELEASES THE PAGE ADDRESS (6) AND ZEROS ENTRY IN PAGE TABLE.


SYSTEM
INTERFACE

1.0 FUNCTIONAL SPECIFICATION

1.1 GENERAL DESCRIPTION

THE SYSTEM INTERFACE IS RESPONSIBLE FOR THE MAJORITY OF THE TRANSACTIONS BETWEEN THE LOGO PROCESSOR AND THE UTS MONITOR. IT IS RESPONSIBLE FOR GENERAL INITIALIZATION OF THE LOGO PROCESSOR'S INTERFACE WITH THE USER, FOR ALL INPUT AND OUTPUT, AND FOR HANDLING ERROR CONDITIONS AND MONITOR TRAPS.

ITS PRINCIPAL RESPONSIBILITY IS THE LOGO FILE-HANDLING CAPABILITIES. THE INPUT AND OUTPUT DATA STREAMS ARE DESCRIBED MORE FULLY IN SECTION 1.2.

THE SYSTEM INTERFACE IS DESIGNED TO BE A SET OF SUBROUTINES CALLABLE FROM THE OTHER PORTIONS OF THE LOGO PROCESSOR. WITH THE EXCEPTION OF A TERMINAL CALL TO THE ERROR ROUTINE (WHICH NEVER RETURNS TO THE CALLING POINT) AND TRAP HANDLING (WHICH IS TREATED IN ACCORDANCE WITH MONITOR CONVENTIONS), THE MODULES OF THE SYSTEM INTERFACE ALWAYS BEHAVE IN A PREDICTABLE FASHION: CALLS ARE MADE WITH NONE OR MORE ARGUMENTS IN A REGISTER; DATA IS TRANSFERRED IMPLICITLY THROUGH DATA AREAS KNOWN TO BOTH THE CALLING PROGRAM AND THE MODULE. WITH THE EXCEPTIONS NOTED, THE CALL TO THE SYSTEM INTERFACE MAY BE VIEWED AS A SINGLE INSTRUCTION WHICH ALWAYS RETURNS TO THE NEXT INSTRUCTION IN SEQUENCE.

IT HAS ALSO BEEN DESIGNED TO FUNCTION AS NORMALLY IN BATCH AS ON LINE (WITH RESPECTS TO INPUT/OUTPUT).

1.21 LOGO INPUT/OUTPUT STREAMS.

THERE ARE FOUR INPUT/OUTPUT STREAMS USED IN THE LOGO PROCESSOR. THESE ARE THE LOAD, STORE, INPUT, AND OUTPUT STREAMS.

THE **LOAD STREAM** IS THE MEDIUM THROUGH WHICH THE PARSER RECEIVES COMMANDS FROM THE USER. UNLESS THE USER HAS ASSIGNED THE INPUT STREAM ELSEWHERE BEFORE CALLING LOGO, THE LOAD STREAM INITIALLY INPUTS FROM THE USER'S TERMINAL. AT ANY POINT THE PARSER MAY ISSUE A CALL TO A ROUTINE (SYSLSET) WHICH MAY BE USED TO DIVERT THE LOAD STREAM TO SOME OTHER INPUT DEVICE. THIS DEVICE WILL THEN BE USED UNTIL ANOTHER CALL TO SYSLSET IS MADE, OR UNTIL AN END-OF-FILE IS REACHED, AT WHICH POINT THE LOAD STREAM WILL INPUT FROM THE 'NORMAL' INPUT DEVICE (THE USER'S TERMINAL OR, FOR BATCH, THE CARD READER).

THE **STORE STREAM** IS THE MEDIUM THROUGH WHICH THE PARSER STORES USER'S PROGRAMS. IT HAS NO OTHER FUNCTION.

THE **INPUT STREAM** IS THE MEDIUM THROUGH WHICH THE EXECUTER INPUTS PROGRAM DATA. AFTER IT IS INITIALLY ASSIGNED (OR THE DEFAULT IS TAKEN) BEFORE CALLING LOGO, IT ALWAYS TAKES INPUT FROM THE SAME DEVICE.

THE **OUTPUT STREAM** IS USED BY ALL LOGO MODULES TO COMMUNICATE WITH THE USER. THE SYSTEM INTERFACE ITSELF USES THIS STREAM TO TYPE ERROR MESSAGES.

IN ADDITION TO THESE STREAMS, THREE OTHER EXTERNAL (TO LOGO) DATA PATHS ARE USED. IF THE USER HAS ASSIGNED THE OUTPUT STREAM TO A DEVICE OTHER THAN HIS TERMINAL (OR THE LINE PRINTER, IF IN BATCH), ERROR MESSAGES

ARE ISSUED TO HIM THERE AS WELL AS IN THE LISTING FILE. ANOTHER DATA PATH IS USED FOR THE ERROR LOGGING REFERRED TO BELOW. YET ANOTHER PATH IS USED TO RETRIEVE EXTENDED ERROR MESSAGES REQUESTED BY THE USER.

1.22 SYSTEM INTERFACE DATA STRUCTURES.

ALL DATA COMMUNICATION BETWEEN THE SYSTEM INTERFACE AND OTHER PORTIONS OF THE LOGO PROCESSOR IS HANDLED IN ONE OF TWO WAYS. IN THE CASE OF NUMERICAL ARGUMENTS (ERROR CODES, ETC.), A REGISTER IS PASSED CONTAINING THE NUMBER ITSELF.

DATA IS PASSED THROUGH FIXED LOCATIONS KNOWN BOTH TO THE CALLING PROGRAM AND THE SYSTEM INTERFACE MODULE. WHILE THIS IS NOT A GOOD PROGRAMMING TECHNIQUE IN GENERAL, IT IS WELL ADAPTED FOR A NON-RECURSIVE OPERATION SUCH AS I/O, BEING EASY FOR BOTH THE CALLING AND 'RECEIVING' PROGRAMS.

THESE FIXED DATA AREAS ARE CONTAINED IN SYSCTX, THE SYSTEM INTERFACE CONTEXT AREA, AND ARE DETAILED BELOW.

1.23 ACCOUNT/USER INFORMATION (SI;ACN)

THIS DATA BLOCK CONTAINS THE LOGO USER'S ACCOUNT NAME AND USER I.D. IN THE FOLLOWING FORMAT:

SI;ACN	WORD 0	FIRST FOUR CHARS, ACCT
	WORD 1	LAST FOUR CHARS, ACCT
	WORD 2	FIRST FOUR CHARS, USER
	WORD 3	NEXT FOUR CHARS, USER
	WORD 4	LAST FOUR CHARS, USER

1.24 SYSTEM INTERFACE FLAG WORD (SI:FLAG)

SI:FLAG IS A FLAG WORD IN THE FOLLOWING FORMAT

FORMAT (BITWISE):

XBDCCCCC CCCCCCCC CCCCCCCC CCCCCCCC

X = BADFLAG - TURNED ON DURING CRITICAL CODE TO AVOID RE-ENTRANT ERROR SITUATIONS.

B = BATCHFL - TURNED ON IF LOGO IS RUNNING IN BATCH, WHICH HAS A BEARING ON I/O OPERATIONS.

D = DIAGFL - TURNED ON IF THE OUTPUT STREAM (NORMALLY THE RECEPTOR OF DIAGNOSTIC MESSAGES) HAS BEEN DIVERTED, THUS PROVIDING DIAGNOSTIC MESSAGES LIVE AND DIRECT TO THE USER.

1.245 USER BREAK COUNT (SI:BRKC)

THE USER BREAK COUNT IS INCREMENTED EVERY TIME THE USER APPLIES THE BREAK KEY. IT IS KEPT IN A FULL WORD F THE CONVENIENCE OF THE EXECUTER.

1.25 INPUT BUFFER (SI:IBUF)

THIS BUFFER IS USED BY CALLS TO SYSIN AND TO SYSLOAD, I.E., BY BOTH INPUT STREAMS. SINCE THESE MAY NEVER BE ACTIVE SIMULTANEOUSLY, NO CONFLICT CAN ARISE.

SI:IBUF HALFWORD COUNT OF SIGNIFICANT DATA BYTES, FOLLOWED BY UP TO 298 BYTES OF DATA.

1.252 SI:INSM

THIS DOUBLEWORD IS DIVIDED INTO THREE HALFWORDS (THE FOURTH IS UNUSED) GIVING THE MAXIMUM BUFFER SIZE OF SI:OBUF, SI:INSM, AND SI:IBUF.

1.255 LIMITS ON BUFFER SIZE (SI:LIM)

SI:LIM CONSISTS OF A DOUBLEWORD, EACH HALFWORD OF WHICH CONTAINS THE LIMIT ON THE NUMBER OF CHARACTERS WHICH MAY BE PLACED IN ONE OF THE SYSTEM INTERFACE BUFFERS:

HALFWORD 0 LIMIT FOR SI:OBUF
1 LIMIT FOR SI:INSM
2 LIMIT FOR SI:IBUF
3 UNUSED

1.26 RECORD OF LAST ERROR MESSAGE (SI:LEM)

THIS CONTAINS THE ERROR CODE FOR THE LAST ERROR MESSAGE ISSUED TO THE USER, IT IS USED FOR DEBUGGING AND ALSO TO ALLOW LOOK-UP OF AN EXTENDED ERROR MESSAGE ON A FILE.

SI:LEM WORD 0 LAST ERROR CODE.

1.27 NAME OF LOAD FILE (SI:SNAM)

THIS BLOCK CONTAINS THE NAME OF THE FILE TO BE SET BY A SYSSSET OR SYSLSET -- 3 WORDS, TEXTC FORMAT

1.271 LOAD FILE ACCOUNT (SI:SACC)

THIS DOUBLEWORD CONTAINS THE NAME OF THE ACCOUNT TO BE SET BY SYSSSET OR SYSLSET. BLANK OR ZERO IF NONE.

1.272 LOAD FILE PASSWORD (SI:SPAS)

THIS DOUBLEWORD CONTAINS THE PASSWORD OF THE FILE TO BE SET BY SYSSSET OR SYSLSET. BLANK OR ZERO IF NONE.

1.28 OUTPUT BUFFER (SI:OBUF)

THE OUTPUT BUFFER HAS THE SAME FORMAT AS SI:IBUF, AND IS USED IN THE SAME MANNER, EXCEPT THAT IT IS USED FOR OUTPUT STREAM OPERATIONS.

1.29 TIME AND DATE (SI:TIME)

THIS BLOCK WILL CONTAIN THE CORRECT TIME AND DATE IF THE USE OF IT IS PRECEDED BY A CALL TO SYSCLOCK.

```
SI:TIME   WORD 0    !HH:M!
          WORD 1    !M XX!
          WORD 2    !X DD!
          WORD 3    !,!YY!
```

!HH:MM XXX DD,!YY! IN UTS STANDARD FORMAT (HH:MM IS THE TIME, XXX IS THE MONTH, DD THE DAY, AND YY THE LAST TWO DIGITS OF THE YEAR).

1.297 TRAP STATUS (SI:TRPS)

SI:TRPS, IN FACT, IS THE BUFFER USED BY SYSERR TO RECORD FATAL ERRORS IN A DIAGNOSTIC FILE. IT CONTAINS, AS A SUBSET, SEVERAL OF THE ITEMS MENTIONED ABOVE, IN THE ORDER INDICATED:

```
SI:TRPS   WORD 0    SI:TIME (4 WORDS)
          WORD 4    SI:ACN  (5 WORDS)
          WORD 9    SI:LEM  (1 WORD)
          WORD 10   SI:FLAG (1 WORD)
          WORD 11   SI:BRKC (1 WORD)
          WORD 12   SI:TRPX (19 WORDS)
```

THIS DATA IS WRITTEN IN THE LOGO ERROR LOG IN THE CASE OF A FATAL ERROR OR A TRAP.

1.298 MONITOR TRAP INFORMATION

SI:TRPX CONTAINS THE TRAP STATUS RETURNED FROM THE MONITOR WHEN A TRAP OCCURS:

```
WORD 0   PROGRAM STATUS DOUBLEWORD
     2   16 GENERAL REGISTERS
    18   TRAP NUMBER
```

AFTER A FATAL ERROR (NON-TRAP), THE LOCATION OF THE TRAP IS PLACED IN WORD 0, THE 16 GENERAL REGISTERS IN LOCATIONS 2-17, AND ZERO IN WORDS 1 AND 18.

1.3 MODULE FUNCTIONAL SPECIFICATIONS

ALL OF THE SYSTEM INTERFACE MODULES AVAILABLE TO THE OTHER COMPONENTS OF THE LOGO PROCESSOR ARE CALLED BY A SIMPLE 'BRANCH AND LINK' TO THE DESIRED MODULE THROUGH THE LINK REGISTER (CURRENTLY REGISTER 11). THE ARGUMENT, IF ANY, IS PASSED IN THE 'F' REGISTER (CURRENTLY REGISTER 13); ANY VALUE RETURNED IS PASSED IN REGISTER 'RET' (CURRENTLY ALSO REGISTER 13). FURTHER DETAILS ARE PROVIDED IN THE INDIVIDUAL DESCRIPTIONS.

1.31 SYSINIT-- GENERAL SYSTEM INITIALIZATION.

1.3THIS MODULE IS CALLED AS A SUBROUTINE BY THE LOGO EXECUTIVE. IT TAKES CONTROL OF ALL BREAKS AND TRAPS, SETS THE FLAGS IN SI:FLAG, AND OPENS THE INPUT, OUTPUT, AND LOAD STREAMS.

THE CALLING SEQUENCE IS OF THE FORM:

```
BAL,LINK SYSINIT
```

NO PARAMETERS ARE PASSED OR RETURNED.

1.32 SYSLSET-- DIVERT LOAD STREAM

THIS MODULE IS USED TO DIVERT THE LOAD STREAM TO THE FILE IDENTIFIED BY SI:SNAM, SI:SACC, AND SI:SPAS. THE CALL IS OF THE FORM:

```
BAL,LINK SYSLSET
```

NO PARAMETERS ARE PASSED OR RETURNED; IF THE FILE COULD NOT BE OPENED, THE STREAM IS RE-DIRECTED TO THE USER CONSOLE (ON-LINE) OR LOGO PROCESSING IS TERMINATED. IN NEITHER CASE IS AN ERROR INDICATED TO THE CALLING PROGRAM.

1.33 SYSSSET-- OPEN STORE STREAM

THIS MODULE WILL OPEN THE NORMALLY CLOSED STORE STREAM, TO THE FILE IDENTIFIED BY SI:SNAM, SI:SACC, AND SI:SPAS. THE CALL IS OF THE FORM:

```
BAL,LINK SYSSSET
CI,RET   0
BNE      NOSTORE
```

NO PARAMETERS ARE PASSED; ONE PARAMETER IS RETURNED IN LOGICAL REGISTER 'RET'; A NON-ZERO VALUE INDICATES THAT THE REQUESTED STORE CANNOT BE PERFORMED. THE EXAMPLE SHOWS A BRANCH TO NOSTORE IF THE STORE COULD NOT BE DONE.

1.335 SYSXSET

SYSXSET IS CALLED BY SYSLSET AND SYSSSET TO PROVIDE A COMMON METHOD OF CLOSING THE RESPECTIVE STREAMS AND MERGING NEW FILE INFORMATION INTO THE DCB.

1.34 SYSREAD

SYSREAD DOES THE ACTUAL READ OPERATIONS FOR SYSLOAD AND SYSIN. ALL VARIABLE PARAMETERS OF THE FPT (SEE XEROX UTS BATCH PROCESSING REFERENCE MANUAL) ARE KEPT IN REGISTERS; THEY ARE ALTERED IN SUCH A WAY THAT THE MONITOR PERFORMS ITS OPERATIONS DIRECTLY INTO SI:IBUF, THE INPUT BUFFER. THE CODE SHOULD BE STUDIED VERY CAREFULLY. THE ESSENTIAL THEORY IS THAT 69 CHARACTER READS ARE ISSUED TO THE USER (COMPLETELY TRANSPARENT TO HIM) UNTIL UNDER 100 CHARACTERS REMAIN; AT THIS POINT HE IS GIVEN A WARNING (IF HE IS AN ON-LINE USER) THAT HE IS RUNNING OUT OF SPACE. AT ANY POINT, HE MAY CONTINUE A LINE BY TYPING THE CONTINUATION CHARACTER (CONC) FOLLOWED BY A LINE TERMINATION CHARACTER (AN EOT,FF,CR,LT,SUB,FS,GS,RS, OR US). WHEN HE ISSUES A TERMINATION CHARACTER WITHOUT A CONTINUATION CHARACTER, THE INPUT IS COMPLETE.

1.341 SYSLOAD -- READ FROM THE LOAD STREAM

THIS MODULE READS ONE LOGICAL RECORD INTO SI:OBUF. THE CHARACTER (CARRIAGE-RETURN, ETC.) ENDING THE LOGICAL RECORD IS INCLUDED IN THE DATA. THE CALL IS OF THE FORM:

```
BAL,LINK SYSLOAD
```

NO PARAMETERS ARE PASSED OR RETURNED. NOTE THAT AN END-OF-FILE ON SYSLOAD WILL CAUSE THE LOAD STREAM TO BE REDIRECTED TO THE USER'S DEFAULT STREAM DEVICE. AN END-OF-FILE ON THE DEFAULT DEVICE IN BATCH WILL CAUSE LOGO TO TERMINATE.

1.342 SYSIN -- READ FROM THE INPUT STREAM

THIS ROUTINE READS AS IN SYSLOAD, EXCEPT THAT INPUT IS TAKEN FROM THE INPUT STREAM. AN END-OF-FILE ON THIS DEVICE WILL CAUSE SYSIN TO READ FROM THE DEFAULT DEVICE IN SUBSEQUENT CALLS. AN END-OF-FILE ON THE DEFAULT DEVICE IN BATCH WILL CAUSE TERMINATION OF LOGO.
THE CALL IS OF THE FORM:

```
BAL,LINK SYSIN
```

NO PARAMETERS ARE PASSED OR RETURNED.

1.36 SYSOUT -- WRITE TO THE OUTPUT STREAM

THIS MODULE WRITES THE DATA IN SI;OBUF TO THE OUTPUT STREAM, BREAKING IT INTO PHYSICAL RECORDS IF NEEDED DUE TO THE SIZE OF THE LOGICAL RECORD. CALLS ARE OF THE FORM:

```
BAL,LINK SYSOUT
```

NO PARAMETERS ARE PASSED OR RETURNED.

1.365 SYSDIAG

SYSDIAG MERELY LOADS THE DCB ADDRESS AND THE KEY ADDRESS FOR THE M;DO DCB AND CALLS SYSWRITE.

1.37 SYSSTOR -- WRITE TO THE STORE STREAM.

THIS ROUTINE WORKS AS SYSOUT, WITH THE EXCEPTION THAT IT WRITES TO THE OUTPUT STREAM, AND THAT A NULL RECORD (COUNT=0) IN SI;OBUF WILL CAUSE THE FILE TO BE CLOSED. NOTE THAT THE FILE WILL ALWAYS BE CLOSED IF LOGO TERMINATES OF ITS OWN ACCORD ANYWAY.

```
BAL,LINK SYSSTOR
CI,RET   0
BNE      NOSTORE
```

NO PARAMETERS ARE PASSED; ONE PARAMETER IS RETURNED IN LOGICAL REGISTER 'RET'; A NON-ZERO VALUE INDICATES THAT THE REQUESTED STORE CANNOT BE PERFORMED. THE EXAMPLE SHOWS A BRANCH TO NOSTORE IF THE STORE COULD NOT BE DONE.

1.375 SYSWRITE--STREAM OUTPUT ROUTINE

SYSWRITE DOES THE ACTUAL WRITE OPERATIONS FOR SYSOUT, SYSTORE, AND SYSDIAG. ALL VARIABLE PARAMETERS OF THE FPT (SEE XEROX UTS BATCH PROCESSING REFERENCE MANUAL) ARE KEPT IN REGISTERS; THEY ARE MANIPULATED IN SUCH A WAY THAT THE MONITOR PERFORMS ITS OPERATIONS DIRECTLY FROM SI:OBUF, THE OUTPUT BUFFER. THE BASIC THEORY OF OPERATION IS AS FOLLOWS: IF THE NUMBER OF CHARACTERS LEFT IN THE BUFFER IS LESS THAN 72, THE LINE IS OUTPUT AS IT IS. IF NOT, THE 70TH CHARACTER IS LOOKED AT. IF IT IS A SPACE, IT IS REPLACED BY THE CONTINUATION CHARACTER AND THE LINE IS OUTPUT TO THAT POINT. IF NOT, THE LINE IS BACKED UP UNTIL A SPACE IS FOUND OR 15 CHARACTERS HAVE BEEN INSPECTED, AT WHICH POINT THIS CHARACTER IS REPLACED AND THE LINE IS OUTPUT (THE REPLACED CHARACTER IS SAVED). THEN THE REPLACED CHARACTER IS REINSTATED AND MADE THE THEN THE REPLACED CHARACTER IS REINSTATED AND MADE THE NEW BEGINNING-OF-LINE, AND THE ABOVE PROCESS IS CONTINUED UNTIL THE ENTIRE BUFFER HAS BEEN OUTPUT.

1.38 SYSCLOCK-- UPDATE CLOCK

THIS ROUTINE REFRESHED THE CLOCK IN SI:TIME. CALLS ARE OF THE FORM:

```
BAL,LINK SYSCLOCK
```

NO PARAMETERS PASSED OR RETURNED.

1.385 SYSXMSG--EXTENDED ERROR MESSAGES

SYSXMSG TYPES AN EXTENDED ERROR MESSAGE IN RESPONSE TO A QUESTION MARK (SEE SYSLOAD). IT PICKS UP THE LAST ERROR NUMBER FROM SI:LEM AND USES IT TO GENERATE AN EDIT KEY WHICH IS USED TO READ FROM A FILE, 'LOGOMSG' THE FIRST LINE OF AN EXTENDED ERROR MESSAGE IN THIS FILE HAS THE SAME LINE NUMBER AS THE ERROR NUMBER, EACH SUCCESSIVE LINE HAS THIS NUMBER INCREMENTED BY .01.

1.39 SYSERR-- ERROR HANDLERS.

THE NORMAL ENTRY TO SYSERR IS MADE THROUGH A CALL OF THE FOLLOWING FORM:

```
SYSERR    57
```

WHICH IS A MACRO WHICH GENERATES A LOAD OF THE ERROR CODE AND A CALL TO SYSERR.
SYSERR ANALYZES THE ERROR CODE, DECIDES IF IT IS IN RANGE, AND IF SO, PRINTS THE REQUESTED ERROR MESSAGE, INSERTING TEXT FROM SI:INSM ON THE LEFT OR RIGHT IF REQUESTED. IT THEN CHECKS TO SEE IF THE USER IS EXECUTING AND, IF SO, PRINTS THE PROCEDURE NAME AND LINE NUMBER. IF THE ERROR IS FATAL, THE ERROR IS LOGGED TO AN ERROR FILE WITH THE ERROR NUMBER AND THE USER'S REGISTERS. FATAL ERRORS CAUSE A BRANCH TO 'RESET', RESTARTING THE USER, UNLESS THE 'X' OPTION WAS SPECIFIED, IN WHICH CASE THE USER EXITS. SPECIAL ERROR CODE 0 PRINTS THE 'I AM AT' MESSAGE ONLY; -1 EXITS, AND -2 LOGS THE USER OFF (IF FTLOGON IS 1).

1.396 SYSBREAK -- BREAK HANDLER

THE USER BREAK COUNT IS INCREMENTED AND A TRAP RETURN IS MADE TO THE MONITOR.

1.397 SYSTRAP -- TRAP HANDLER

1) IF THE CONDITION IS A STACK OVERFLOW, A CALL IS MADE TO STACKTR IN THE STORAGE MANAGEMENT ROUTINE. IF THIS CALL RETURNS A SUCCESSFUL CONDITION, THAT IS, IF THE STACK SIZE WAS INCREASED SUCCESSFULLY, A TRAP RETURN IS MADE TO THE MONITOR; OTHERWISE, AN APPROPRIATE ERROR CODE IS GENERATED AND A (RE-ENTRANT) CALL IS MADE TO SYSERR.

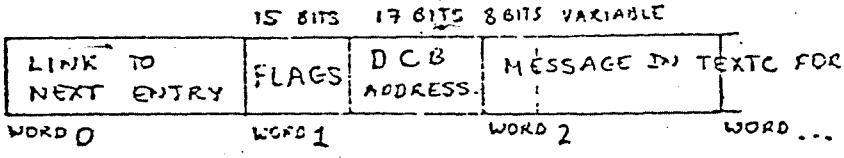
2) IF IT IS NOT A STACK OVERFLOW, A AN APPROPRIATE ERROR CODE IS GENERATED AND A CALL IS MADE TO SYSERR.

IF THE BADFLAG IS SET A TRAP MESSAGE IS GENERATED LOCALLY, AND LOGO IS ABORTED TO PREVENT RE-ENTRANT ERROR CONDITIONS.

1.398 SYSIOER -- INPUT OUTPUT ERROR HANDLER

THIS MODULE HANDLES ALL I/O ERRORS FOR LOGO. IT MAKES EVERY ATTEMPT TO RECOVER AND TO GO TO THE TELETYPE (ON-LINE ONLY) IF THE SELECTED DEVICE DOES NOT FUNCTION. IT USES M:PRINT TO HANDLE ALL MESSAGES, SINCE LOGO'S I/O, AT THE POINT THAT SYSIOER IS CALLED, IS OBVIOUSLY IN TROUBLE.

SYSIOET - SI:IOET

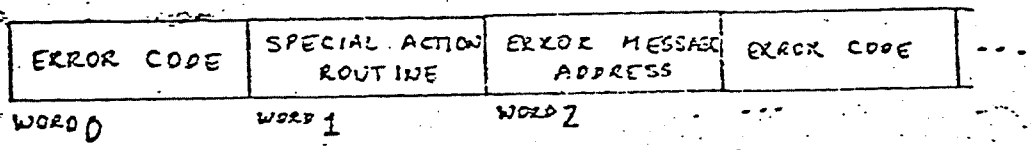


FLAGS:

BIT 0 WORD 1	XB	ABORT IF BATCH
1	TT	TRY TTY
2	XD	CANCEL DIAGNOSTICS
8	RS	RESETTABLE DCB
9	SP	SPECIAL DCB

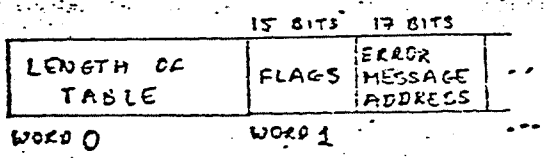
SI:IOET (1) MAKES AN ERROR MESSAGE AVAILABLE FOR EACH INDIVIDUAL DCB, AND
(2) INDICATES WHAT TYPE OF END ACTION TO PERFORM.

SYSIOET - SI:ISAT



SI:ISAT (1) MAKES AN ERROR MESSAGE AVAILABLE FOR EACH RECOGNIZED ERROR CODE, AND
(2) INDICATES WHERE TO GO TO HANDLE THE ERROR.

SYSERT - SI:ERT



FLAGS:

BIT 0	FATAL	BRANCH TO RESET, LOG ERROR
1	D	MOVE REGISTERS TO SI:TRPX BEFORE LOGGING
2	X	END LOGO EXECUTION
3 } 10	L	INSERT ON LEFT
4 } 11	R	INSERT ON RIGHT

SI:ERT (1) CONTROLS WHERE (IF AT ALL) SI:INSH WILL BE INSERTED IN ERROR MESSAGE, AND
(2) CONTROLS END ACTION.

RELEASE 2

SYSIOET 1.3	1 MY 73
SYSERT 1.5	10 MY 73

JH/UCI/31 MY 73

THE LISTING AND FLOWCHARTS EXPLAIN WHAT HAPPENS WHEN AN ERROR OCCURS ON A LOGO FILE OR DEVICE. THEY FAIL, HOWEVER TO EXPLAIN WHY.

THE PROCESS OF HANDLING A LOGO FILE ERROR IS DIVIDED INTO TWO PARTS -- THE FIRST, THE SPECIAL ACTION, IS SPECIFIC TO THE ERROR CODE PASSED BY THE MONITOR, AND THE SECOND, THE END ACTION, IS SPECIFIC TO THE LOGO DCB INVOLVED.

THE DCB'S ARE CLASSIFIED INTO THE FOLLOWING NON-MUTUALLY-EXCLUSIVE SETS:

ABORT IN BATCH

--THOSE DCB'S WHOSE MALFUNCTIONING WOULD MAKE FURTHER BATCH PROCESSING UNDESIRABLE. NOTE THAT THE END-OF-FILE ERROR CONDITION IS AN EXCEPTION TO THE PROCEDURE, AS END-OF-FILE IS THE NORMAL WAY OF TERMINATING A LOAD. THESE DCB'S ARE: M:SI (THERE IS NO PLACE TO TURN IF OUR INPUT DEVICE FAILS), M:CI (SAME, FOR DATA) AND M:LO (THERE IS NO POINT IN CONTINUING IF THE USER CANNOT GET ANY OUTPUT).

RETRY ON TELETYPE

--THOSE DCB'S WHICH SHOULD BE SWITCHED TO THE TELETYPE ON MAL-FUNCTION. CURRENTLY THESE ARE THE SAME DCB'S AS ABOVE (BATCH ABORT). NOTE THAT AN END-OF-FILE ON M:SI IN BATCH WILL NOT BE ABORTED (SEE ABOVE); IN THIS CASE, THE DCB WILL BE SWITCHED TO THE CARD READER.

KILL DIAGNOSTICS

--CURRENTLY ONLY M;DG, THE DIAGNOSTIC FLAG IS RESET, THUS ENDING THE DIAGNOSTIC OUTPUT, IF AN ERROR OCCURS. THIS IS ACCEPTABLE, SINCE IN ANY RATE THE DIAGNOSTICS ARE STILL WRITTEN TO M;LG (M;DG IS USED ONLY TO PROVIDE A COPY TO THE USER OF THE DIAGNOSTICS IF M;LG HAS BEEN DIRECTED AWAY FROM HIS TERMINAL).

RESETTABLE DCB'S

--THOSE WHICH MAY BE CHANGED UNDER PROGRAM CONTROL, I.E., M;SI AND F;STORE (LOAD AND STORE OPERATIONS). USED TO INHIBIT DCB IDENTIFICATION FOR CERTAIN ERRORS (THAT FILE DOES NOT EXIST, ETC.) WHERE PROXIMITY MAKES SUCH IDENTIFICATION UNNECESSARY, AND AESTHETICS, UNWANTED. ALSO USED TO TREAT (FOR EXAMPLE) END-OF-FILE ERRORS AS SUCH ONLY FOR FILES ON WHICH SUCH IS PERMISSABLE (IF THE DCB IS NOT IN THIS CLASS, THE ERROR IS TREATED AS A 'NON-FATAL' ERROR.

SPECIAL DCB'S

--THOSE WHOSE EXISTANCE ARE UNKNOWN TO THE USER, CURRENTLY F;MSG AND F;LOG, THUS, FOR EXAMPLE, IF THE LOG FILE DOES NOT EXIST SYSIOER DOESN'T BOTHER TELLING THE USER (SINCE HE DOESN'T CARE), OR IF THE EXTENDED ERROR MESSAGE FILE DOESN'T EXIST, THE CALLING PROGRAM MAY NOTIFY THE USER AS IT LIKES.

NOTE THAT WHILE THESE SETS PRIMARILY CONTROL END ACTION, THEY ARE USED TO MODIFY SPECIAL ACTION IN SOME CASES. MORE DETAILS ARE AVAILABLE IN THE FLOWCHARTS AND IN THE LISTING.

THOSE ERROR CODES IN THE SPECIAL ACTION TABLE WERE CHOSEN TO REFLECT ERRORS ONLY CAUSED BY PROBLEMS EXTERNAL TO LOGO. THUS, ERRORS IN I/O CAUSED BY BUGS IN LOGO ABORT THE LOGO PROCESSOR. THIS IS DONE FOR TWO REASONS:

1. TO AVOID RE-ENTRANT ERROR CONDITIONS CAUSED BY ESSENTIALLY DEFECTIVE ERROR HANDLING, AND
2. TO IMPEL THE USER TO BRING THE ERROR TO THE IMMEDIATE ATTENTION OF MAINTENANCE PERSONNEL.

IF LOGO IS DEBUGGED, THESE ERRORS SHOULD NEVER OCCUR (AND IN FACT HAVE NOT), SINCE THEY CONCERN REFERENCES TO NON-EXISTANT DCB'S AND OTHER FATAL CONDITIONS. MONITOR I/O ERRORS ARE NOT HANDLED EITHER, SINCE AT THE POINT AT WHICH ONE HAPPENS, THE USER IS ONLY SECONDS AWAY FROM DESTRUCTION IN ANY CASE.

1.4 EXTERNAL ROUTINES AND DATA USED

ONLY ONE ROUTINE EXTERNAL TO THE SYSTEM INTERFACE IS USED. THIS ROUTINE IS STACKTR AS MENTIONED IN SECTION 1.39. THE CALL TO STACKTR IS OF THE FORM:

```
        BAL,11    STACKTR
        CI,RET    0          WAS STACK INCR...
        BNE       BADSTK     IF NOT, ERROR
```

WHERE BADSTK WOULD BE AN ERROR ROUTINE AND THE CODE FOLLOWING THE CALL IN-LINE WOULD BE A MONITOR TRAP RETURN.

IN ADDITION, A BRANCH IS MADE TO RESET IN THE LOGO EXECUTIVE TO BAIL OUT OF A GRAVE ERROR.

FOUR EXTERNAL DATA REGISTERS DEFINED IN THE PARSER ARE NEEDED BY THE SYSTEM INTERFACE TO PRINT THE PROCEDURE AND LINE NUMBER FOR ERROR MESSAGES AND TO DETERMINE THE PROMPT CHARACTER:

PROCEDURE — A WORD CONTAINING A POINTER TO THE NAME OF THE CURRENTLY-EXECUTING PROCEDURE; THE PROCEDURE-NAME STRING CONTAINS A HALFWORD COUNT IN THE FIRST HALFWORD, FOLLOWED BY THE STRING.

LINE — A WORD CONTAINING THE LINE NUMBER OF THE CURRENTLY-EXECUTING LINE, (IN BINARY)

FOUR EXTERNAL DATA ARE USEFUL TO THE S.I. SYSLOAD REFERENCES 'EDFLAG' TO SEE IF LOGO IS EDITING IN ORDER TO SET THE PROPER PROMPT CHARACTER. SYSERR CHECKS 'DEPTH' FOR A NON-ZERO VALUE TO TELL IF LOGO IS EXECUTING A USER-DEFINED PROCEDURE WHEN AN ERROR OCCURS; IF SO, IT USERS 'LINE' AND 'PROCEDURE' TO INFORM THE USER WHAT AND WHERE IT WAS EXECUTING AT THE TIME OF THE ERROR.

I. LIST OF FLOWCHARTS

SYSINIT GENERAL INITIALIZATION

SYSIOER INPUT/OUTPUT ERROR HANDLER

SYSLOAD LOAD STREAM INPUT ROUTINE

SYSIN INPUT STREAM INPUT ROUTINE

SYSREAD STREAM READING MODULE

SYSOUT OUTPUT STREAM OUTPUT ROUTINE

SYSTORE STORE STREAM OUTPUT ROUTINE

SYSWRITE STREAM WRITING MODULE

SYSLSET LOAD STREAM SET ROUTINE

SYSSSET STORE STEAM SET ROUTINE

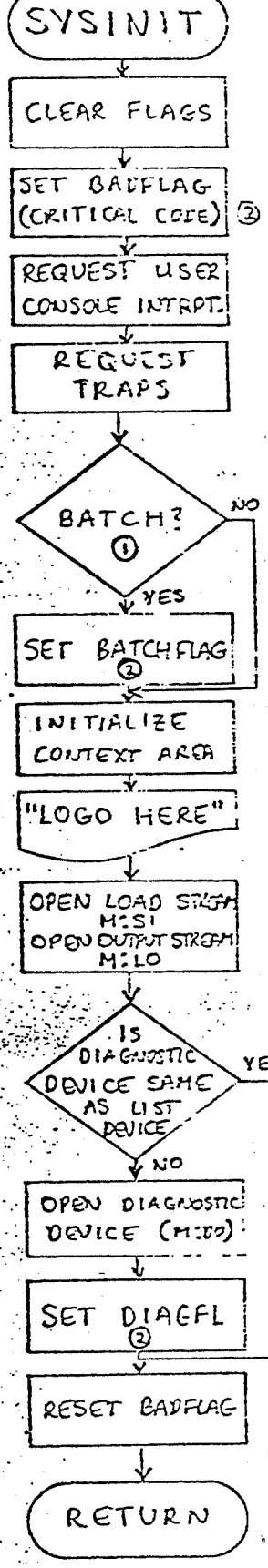
SYSERR ERROR MESSAGE HANDLER

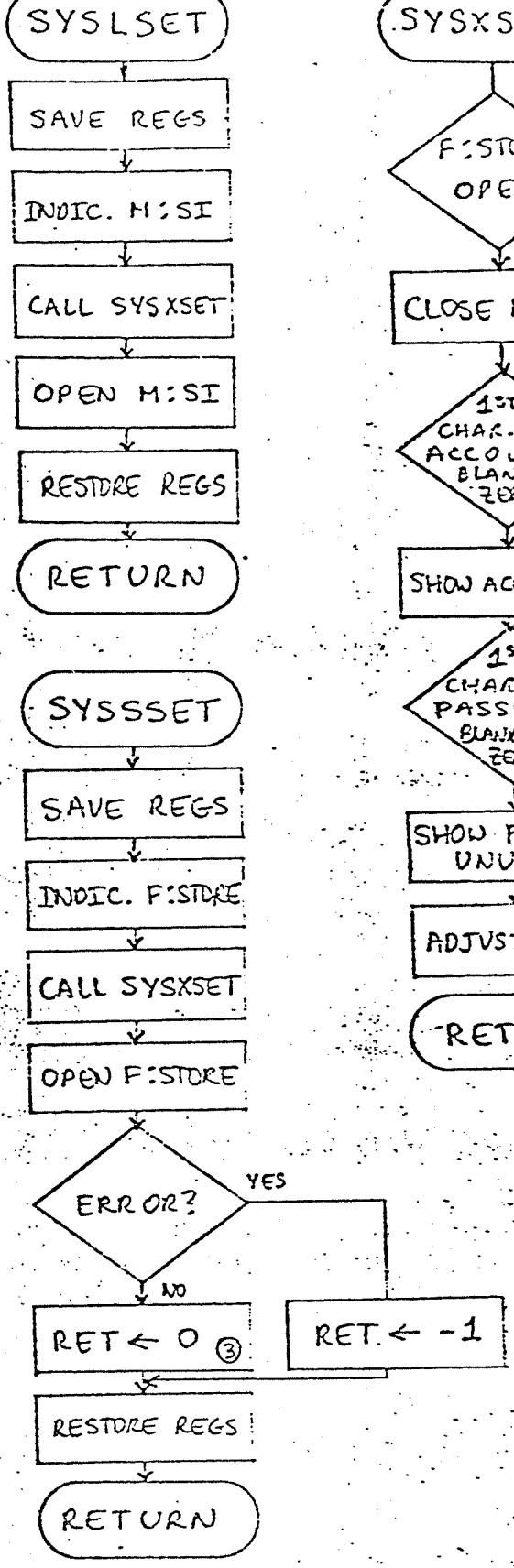
SYSTRAP TRAP HANDLER

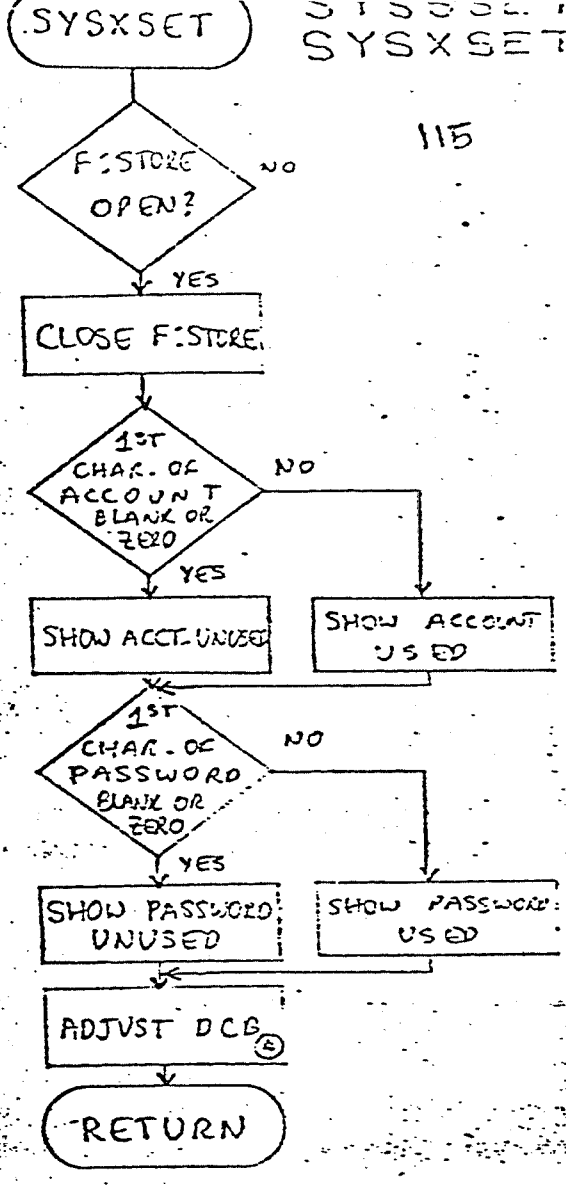
SYSBRK BREAK HANDLER

SYSCLOCK CLOCK UPDATE ROUTINE

SYSXMSG EXTENDED ERROR MESSAGE FACILITY







NOTES

① MONITOR BATCH FLAG IS HIGH-ORDER BIT OF J:JIT IN THE JOB INFORMATION TABLE.

② BADFLAG, BATCHFL, AND DIAGFL ARE IN SI:FLAG.

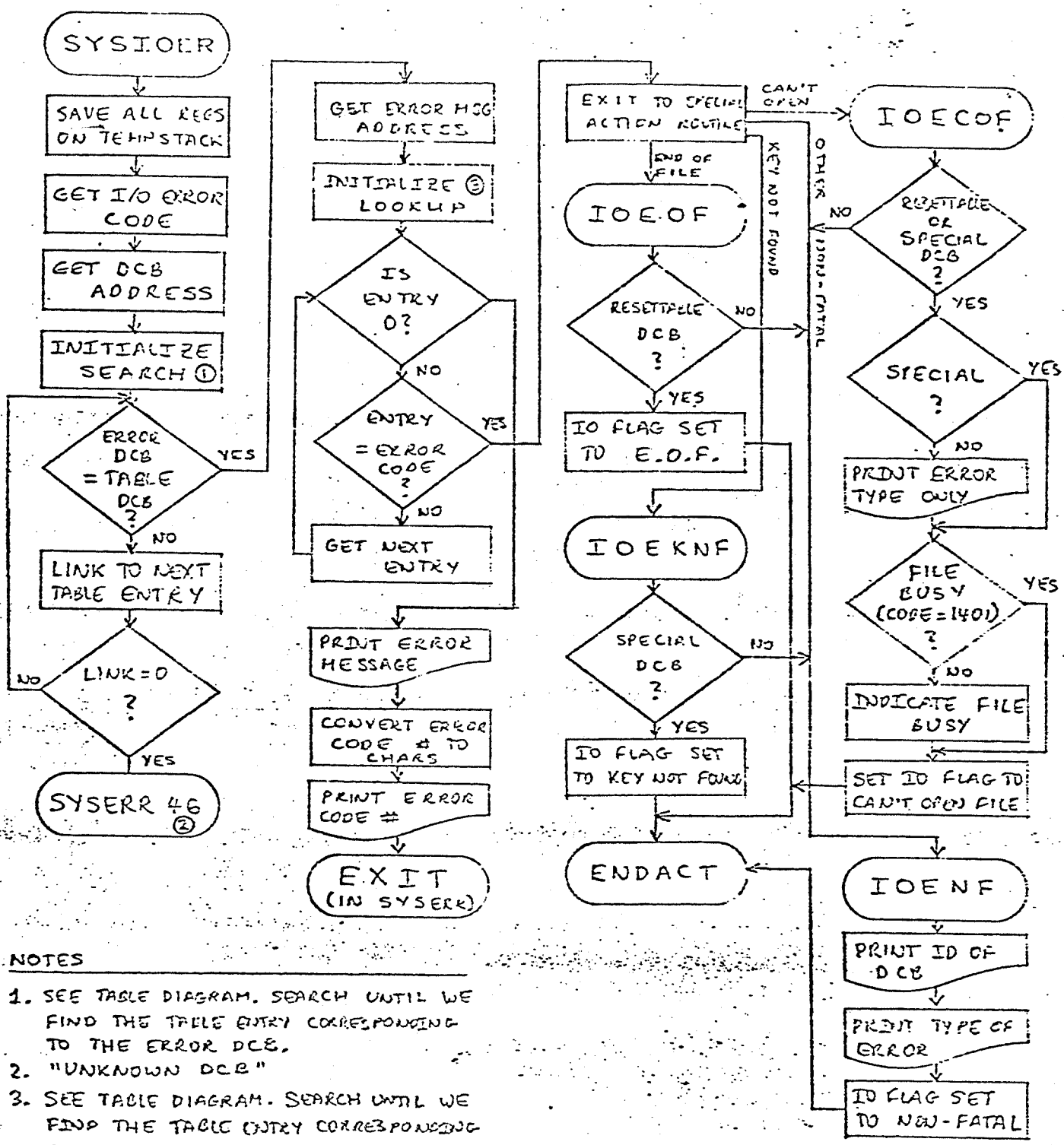
③ "RET" IS SET BY THE MODIFICATION OF THE REGISTER DIRECTLY IN THE STACK.

④ THE ADJUST DCB CALL DOES EVERYTHING THAT THE OPEN CALL DOES EXCEPT ACTUALLY OPENING THE FILE.

RELEASE 2

SYSINIT	1.7	/10 MY 73
SYSLSET	1.1	/14 MR 73
SYSSSET	1.1	/14 MR 73
SYSXSET	1.1	/14 MR 73

J. HOBBS - U.C. IRVINE - 12 MAY 1973



NOTES

1. SEE TABLE DIAGRAM. SEARCH UNTIL WE FIND THE TABLE ENTRY CORRESPONDING TO THE ERROR DCB.
2. "UNKNOWN DCB"
3. SEE TABLE DIAGRAM. SEARCH UNTIL WE FIND THE TABLE ENTRY CORRESPONDING TO THE ERROR CODE.

ALL ERROR MESSAGES FOR KNOWN DCB'S ARE PRINTED THROUGH M:PRINT TO AVOID RE-ENTRANT ERRORS.

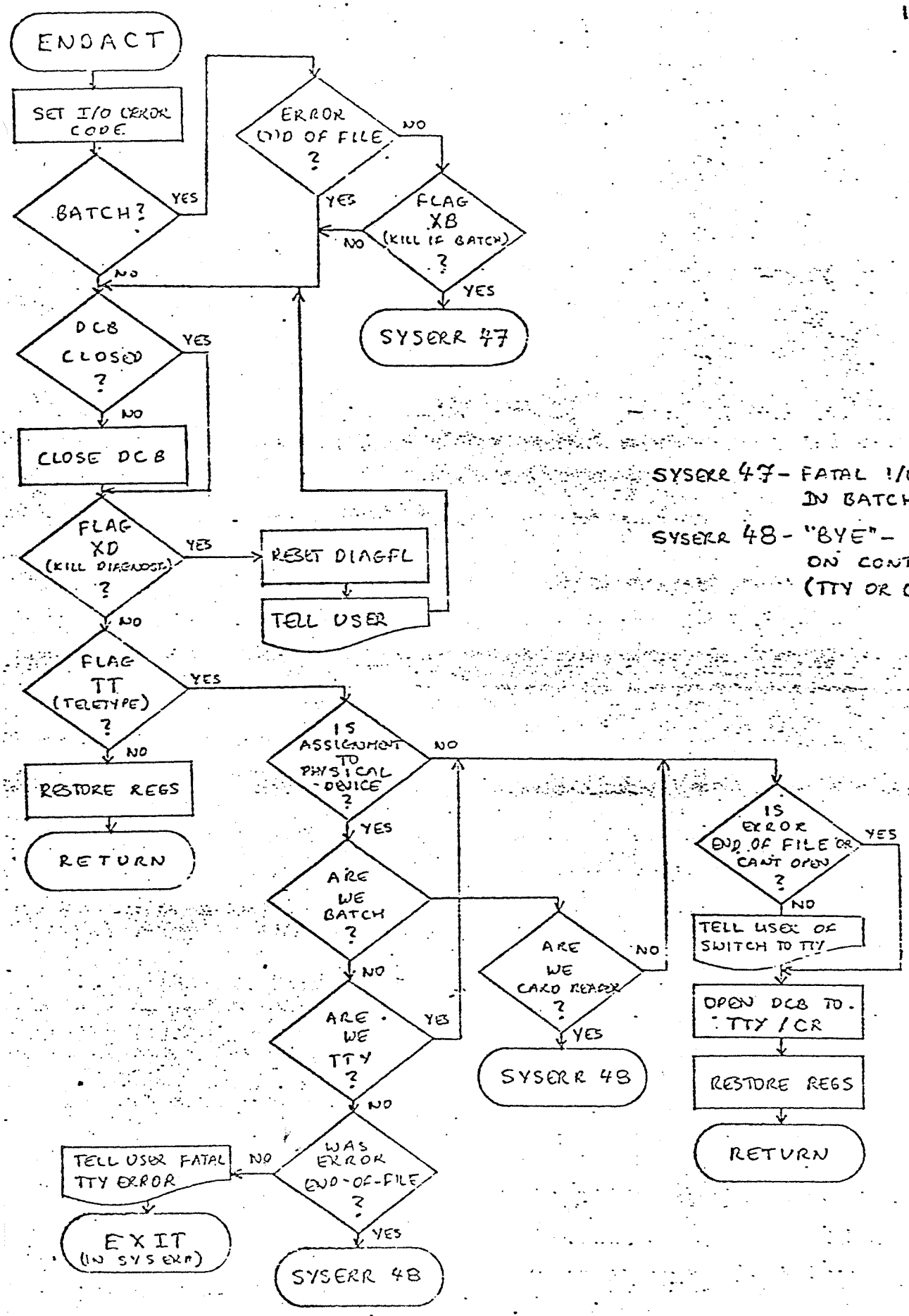
RELEASE 2

SYSIOER 2.1 8 MY 73

RESETTABLE DCB'S ARE THOSE WHOSE ASSIGNMENTS MAY BE CHANGED UNDER PROGRAM CONTROL. CURRENTLY M:SI AND F:STORE.

SPECIAL DCB'S ARE THOSE WHICH ARE INVISIBLE TO THE USER. CURRENTLY F:LOG AND F:MSG.

J. HOBBS/U.C.IRVINE/31 MAY 1973

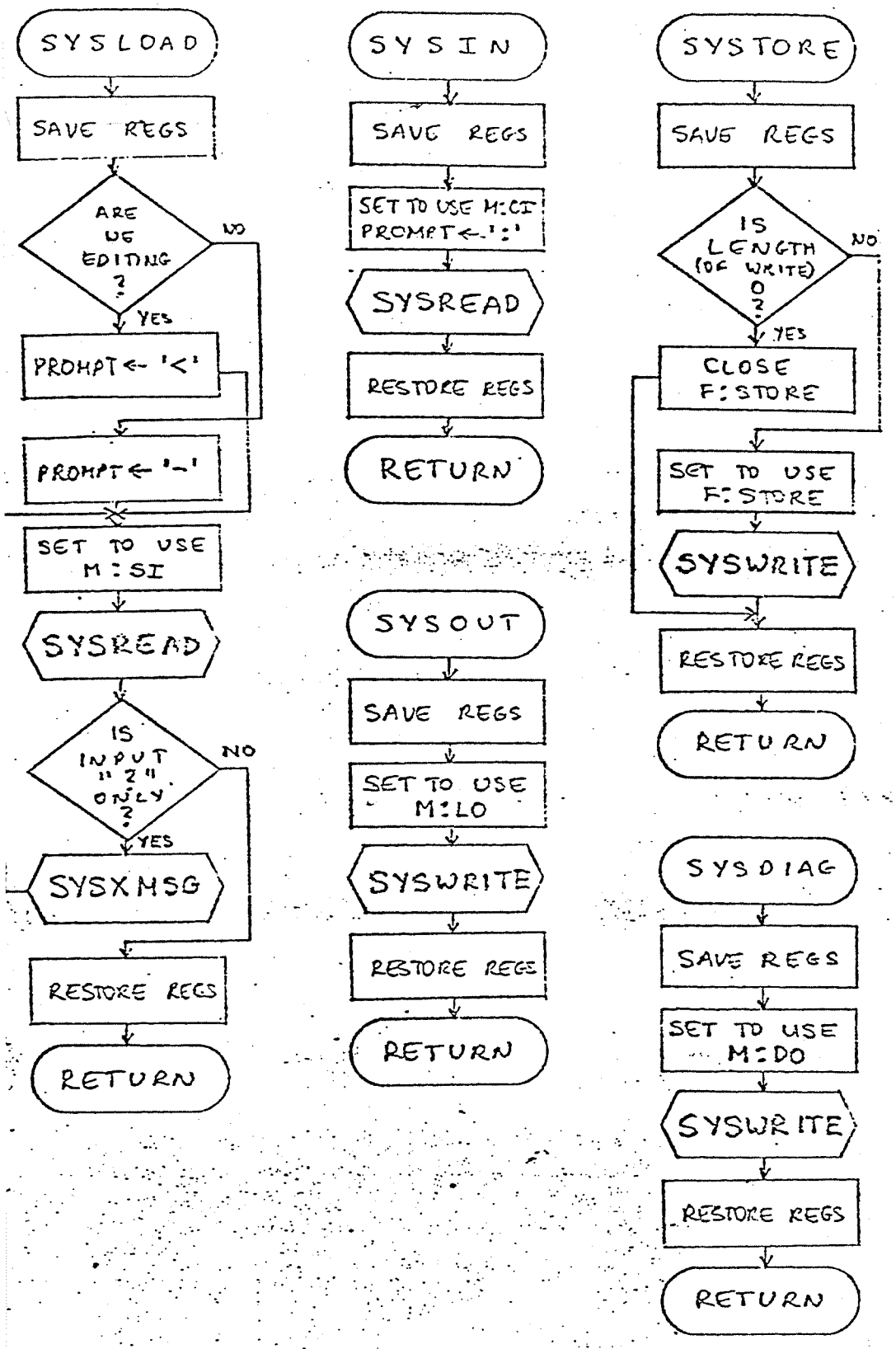


SYSERR 47 - FATAL I/O ERROR IN BATCH

SYSERR 48 - "BYE" - END-OF-FILE ON CONTROL DEVICE (TTY OR C.RDR.)

Logo System Interface

SYSLOAD
SYSIN
SYSOUT
SYSTORE
SYSDIAG

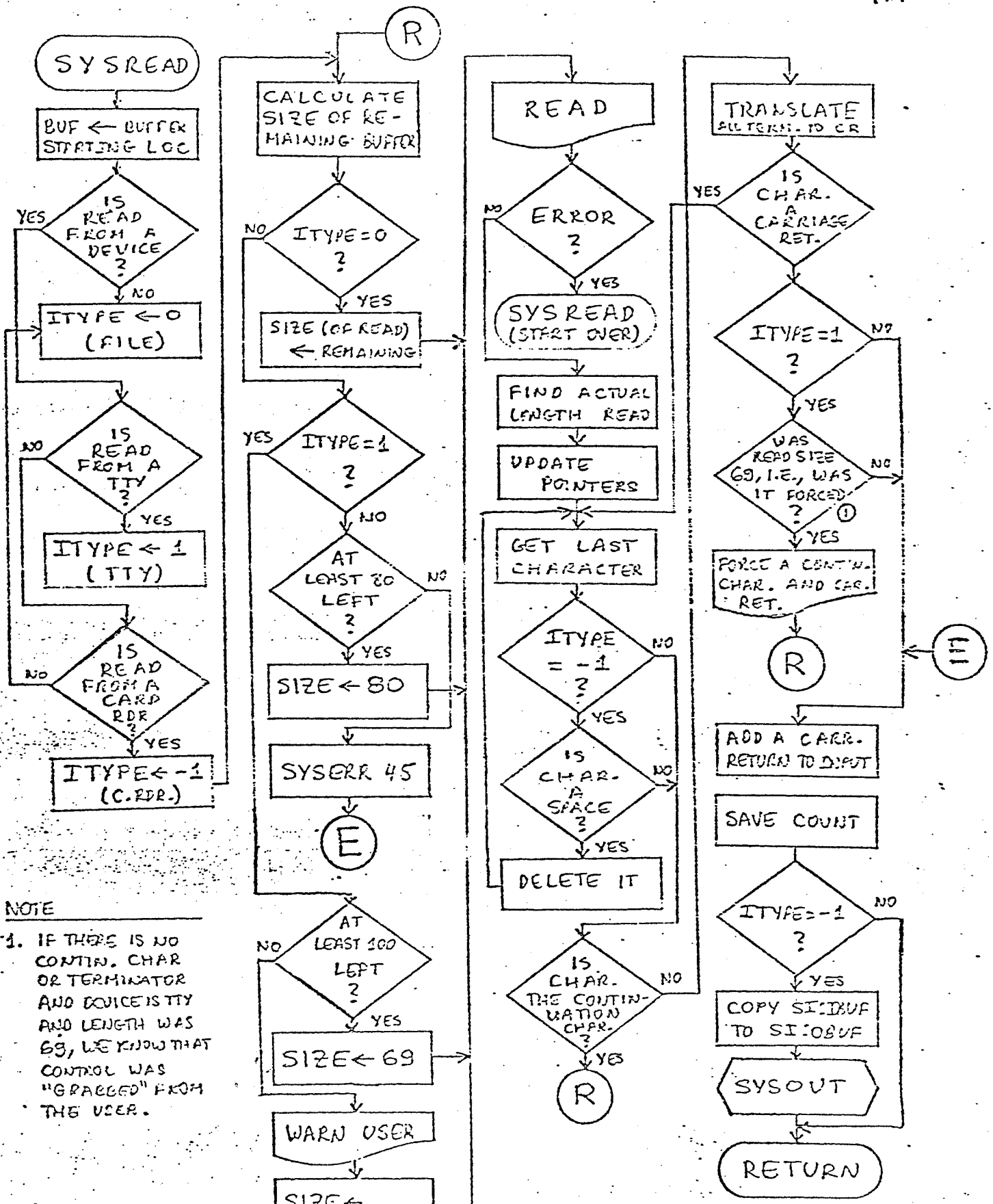


RELEASE 2

SYSLOAD	1.1	13 MR 73
SYSIN	1.1	13 MR 73
SYSOUT	1.1	14 MR 73
SYSTORE	1.1	14 MR 73
SYSDIAG	1.1	14 MR 73

JAMES P. HOBBS/UC IRVINE/1 JUNE 1973

Logo System Interface



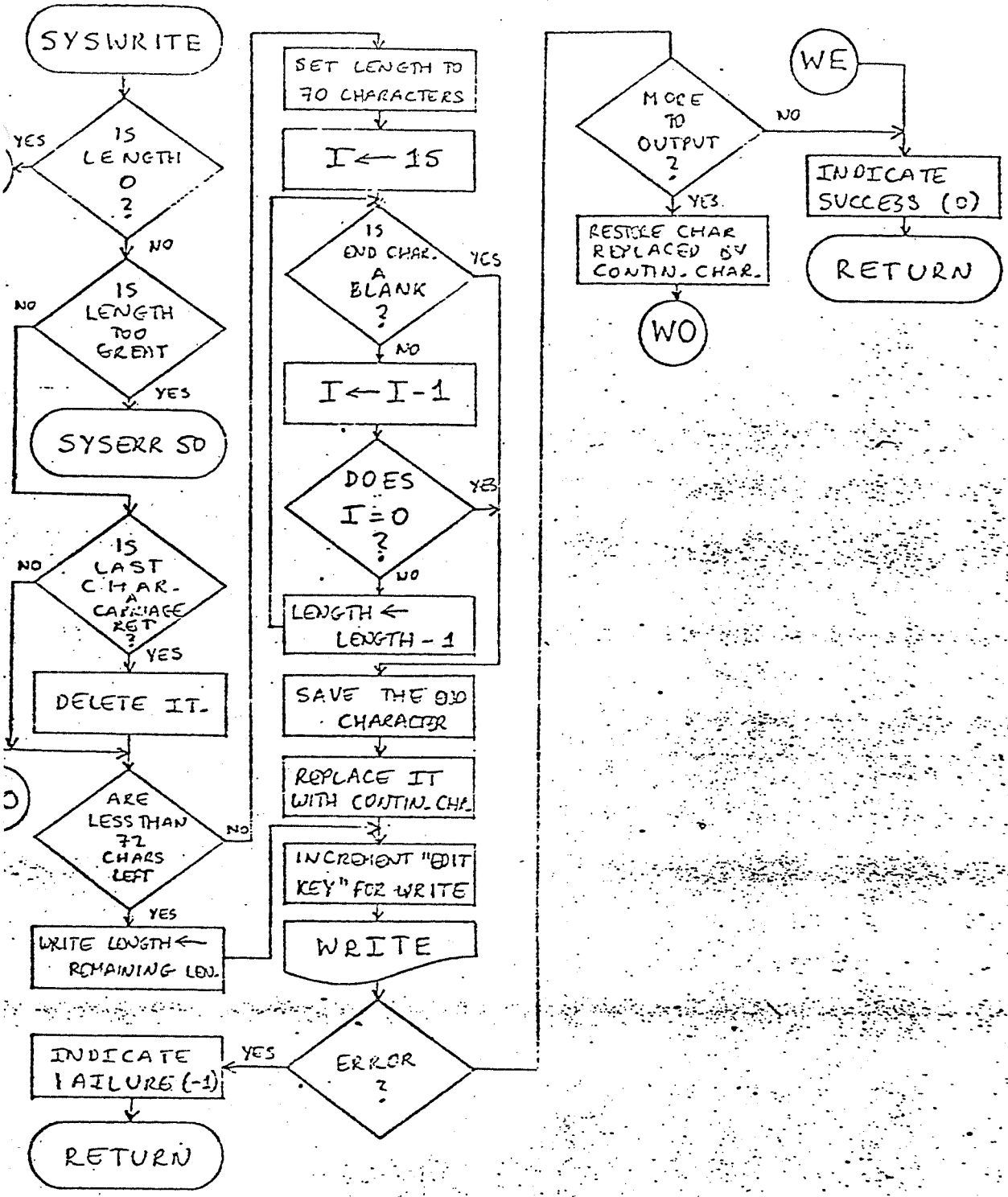
NOTE

1. IF THERE IS NO CONTIN. CHAR OR TERMINATOR AND DEVICE IS TTY AND LENGTH WAS 69, WE KNOW THAT CONTROL WAS "GRABBED" FROM THE USER.

RELEASE 2

SYSREAD 1.4 7 MY 73

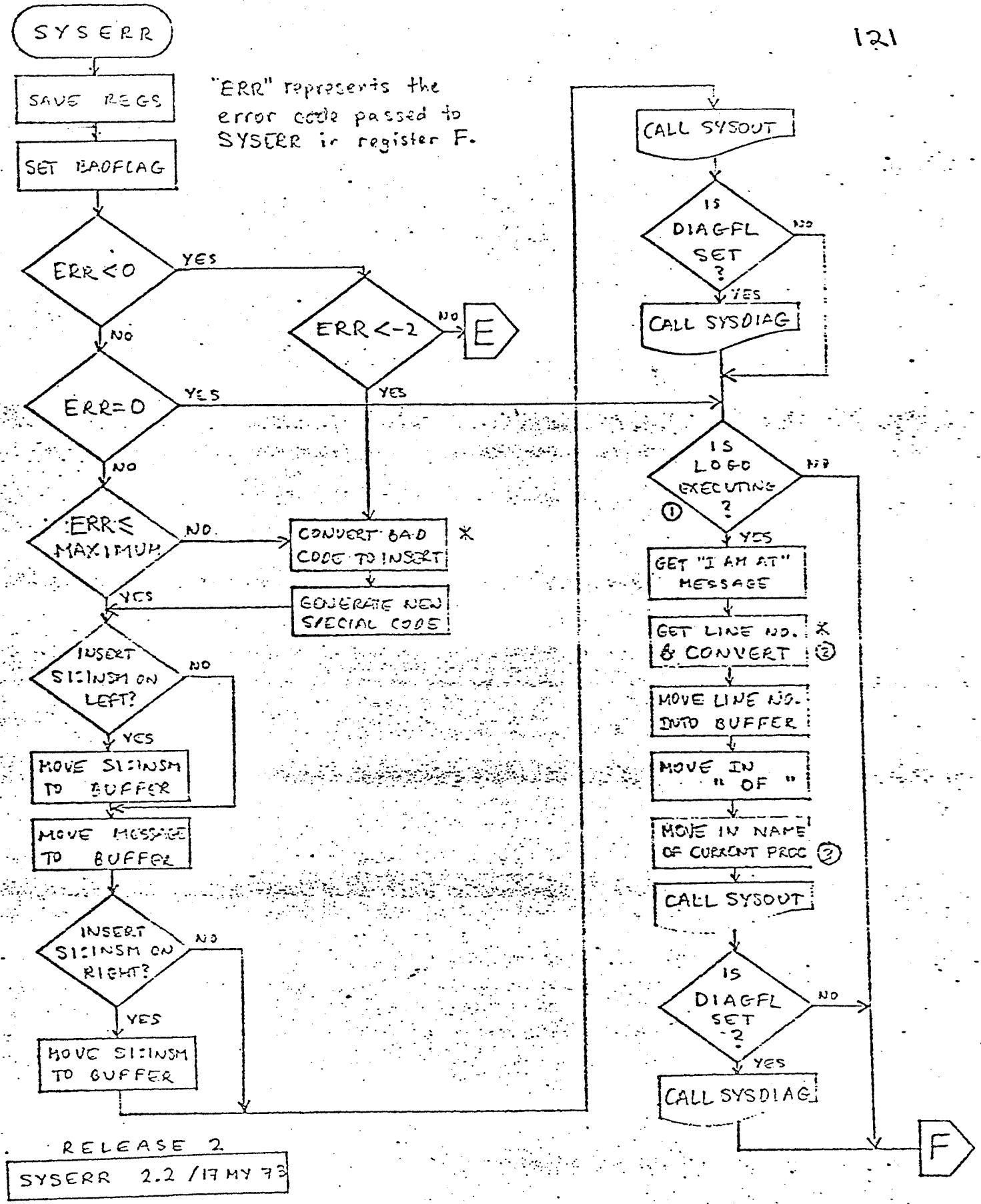
JH / UCI / 1 JUN 73



RELEASE 2

SYSWRITE 1.3 16 APR 73

J. HOBBS/U.C. IRVINE/1 JUNE 73



RELEASE 2

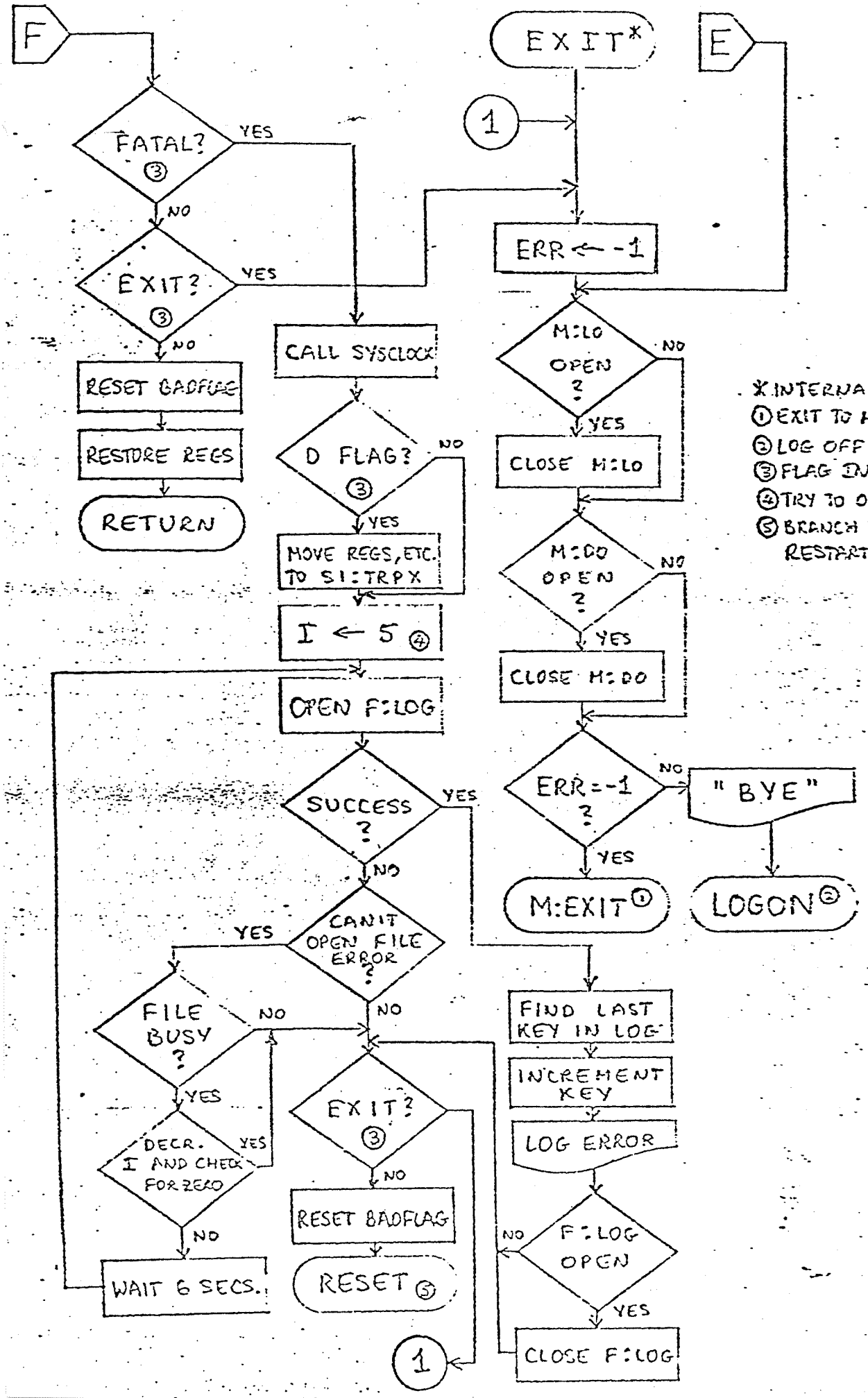
SYSERR 2.2 /17 MY 73

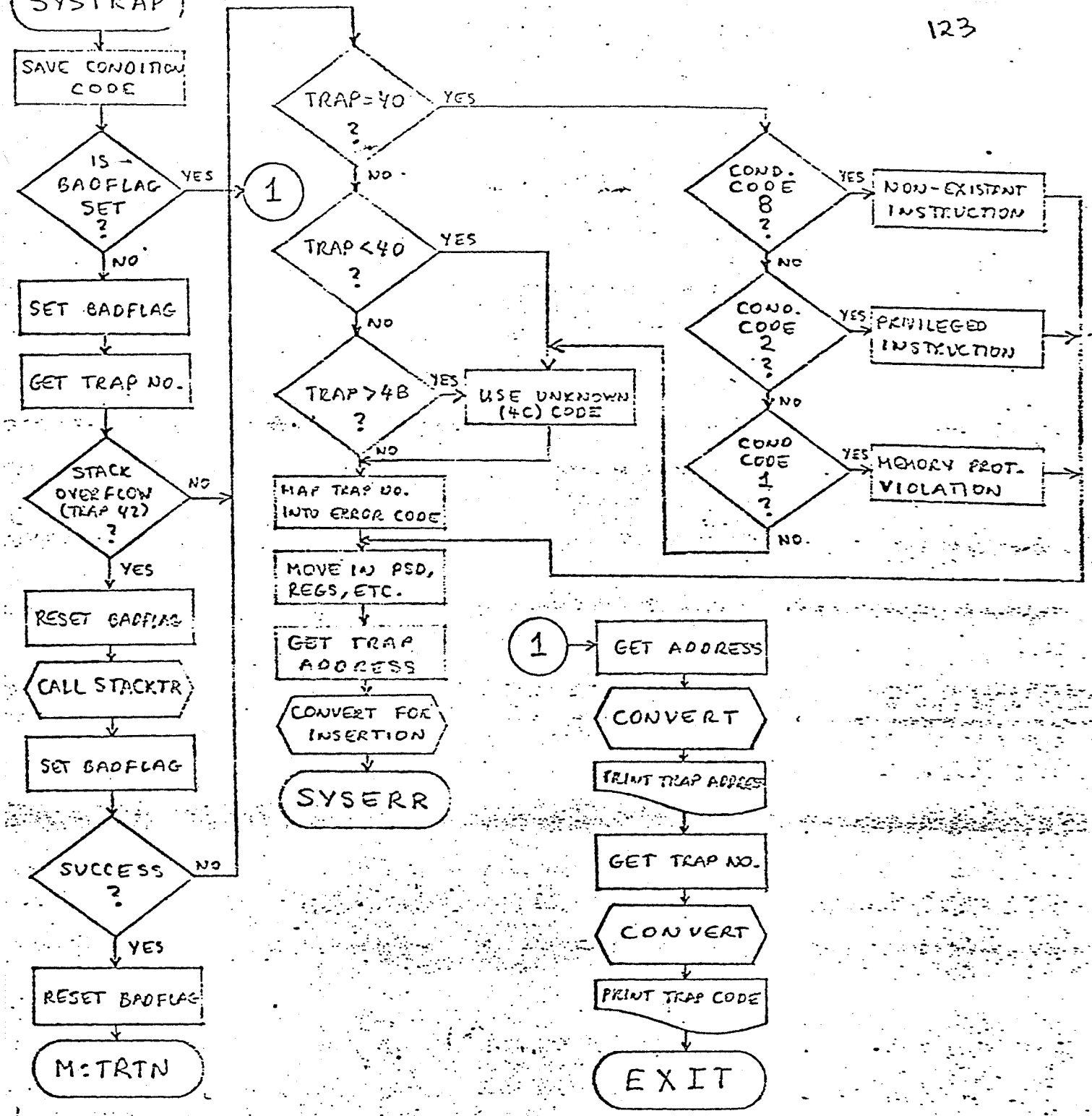
*CONVERT WITH "DECDEC"

(1) CHECK EXTERNAL REFERENCE "DEPTH" - NONZERO MEANS LOGO IS EXECUTING.

(2) EXTERNAL REFERENCE "LINE" - CURRENT EXECUTING LINE NO.

(3) EXTERNAL REFERENCE "PROCEDURE" - CURRENT EXECUTING PROCEDURE.

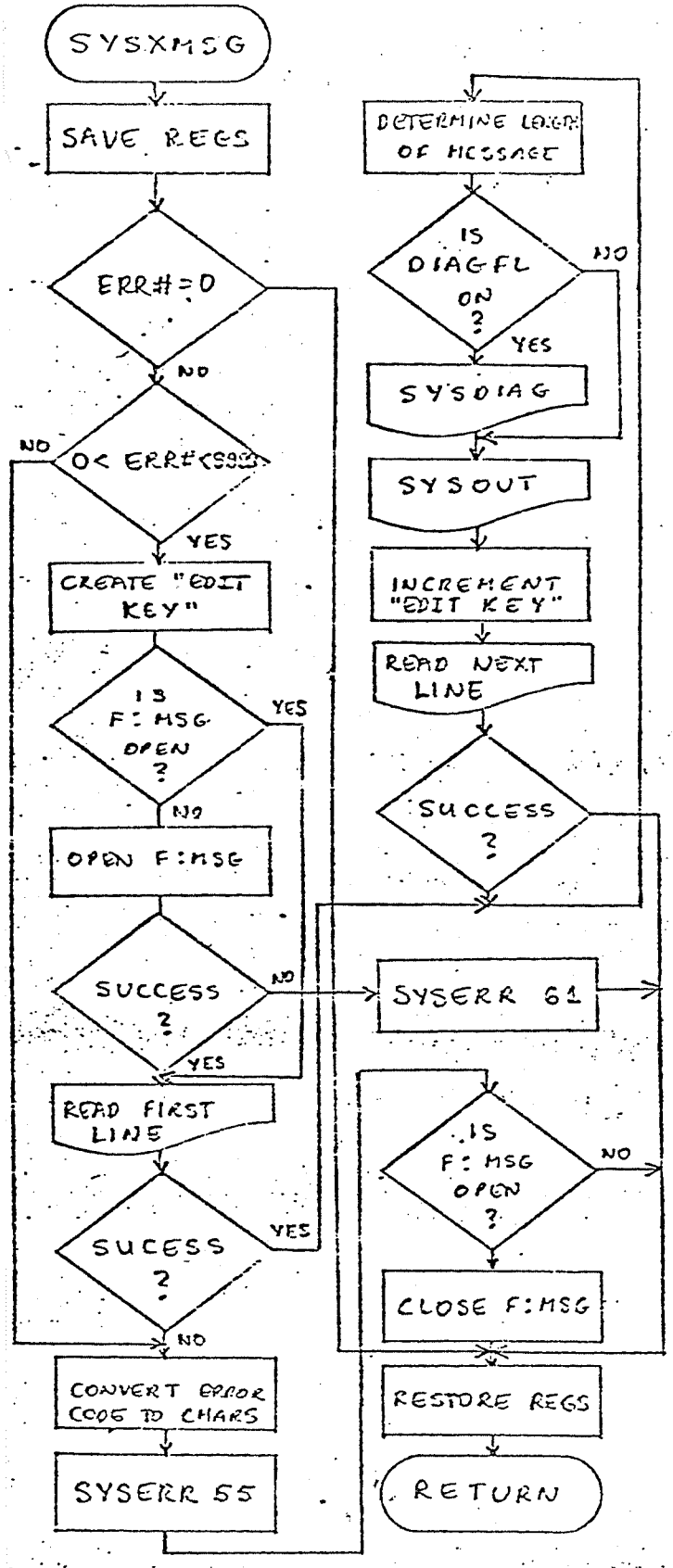

F
FATAL? ③
YES
NO
EXIT? ③
YES
NO
RESET BADFLAG
RESTORE REGS
RETURN
CALL SYSCLOCK
D FLAG? ③
NO
YES
MOVE REGS, ETC. TO S1:TRPX
I ← 5 ④
OPEN F:LOG
SUCCESS ?
YES
NO
CAN'T OPEN FILE ERROR ?
YES
NO
FILE BUSY ?
NO
YES
DECR. I AND CHECK FOR ZERO
YES
NO
WAIT 6 SECS.
EXIT? ③
NO
RESET BADFLAG
RESET ⑤
1
EXIT*
1
ERR ← -1
M:LO OPEN ?
NO
YES
CLOSE M:LO
M:DO OPEN ?
NO
YES
CLOSE M:DO
ERR=-1 ?
NO
YES
" BYE "
M:EXIT ①
LOGON ②
FIND LAST KEY IN LOG
INCREMENT KEY
LOG ERROR
F:LOG OPEN
NO
YES
CLOSE F:LOG
E
* INTERNAL TO S.I. ONLY
① EXIT TO MONITOR
② LOG OFF USER
③ FLAG IN SI:ERT
④ TRY TO OPEN 5 TIMES
⑤ BRANCH TO "RESET" TO RESTART LOGO.

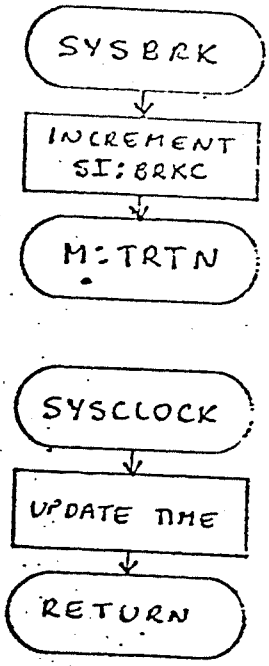


RELEASE 2

SYSTRAP 2.1 /10 MY 73

JH/UCI/17MY73





RELEASE 2

SYSERR	1.0	8 MR 73
SYSCLOCK	1.0	14 MR 73
SYSXMSG	2.1	10 MAY 73

II. LOGO GLOBAL REGISTER ASSIGNMENT

NAME	LOCATION	EXPLANATION
LINK	11	LOGO LINK REGISTER

III. SYSTEM INTERFACE WORK REGISTERS

NAME	LOCATION	EXPLANATION
F	13	FUNCTION PARAMETER FOR CALL
RET	13	RETURNS FUNCTION VALUE
TEMP	15	SCRATCH REGISTER

IV. EXTERNAL REFERENCES

A. DATA CONTROL BLOCKS

M:SI	LOAD STREAM DCB
M:CI	INPUT STREAM DCB
M:LO	OUTPUT STREAM DCB
M:DO	DIAGNOSTIC OUTPUT DEVICE DCB
F:MSG	ERROR MESSAGE FILE DCB
F:LOG	ERROR LOG FILE DCB
F:STORE	STORE STREAM DCB

B. MONITOR INFORMATION FROM JIT

J:JIT	MONITOR BATCH FLAG
J:ACCN	USER ACCOUNT AND ID
J:TCB	ADDRESS OF TEMPSTACK

C. EXTERNAL PROCEDURES AND DATA

RSTACK	REGISTER SAVE STACK
EDFLAG	FLAG SET WHILE EDITING
STACKTR	STACK TRAP HANDLER (STORAGE MGT.)
RESET	PROCEDURE TO KILL LOGO
DEPTH	CHECK FOR EXECUTION STATUS
LINE	EXECUTING LINE NUMBER
PROCEDURE	EXECUTING PROCEDURE

V. MODULE CROSS REFERENCE

IN THE TABLE THAT FOLLOWS, EACH MODULE IS LISTED ALONG WITH THOSE WHICH CALL IT AND THOSE WHICH IT CALLS. THOSE CONTAINING DATA ONLY ARE FLAGGED WITH '$', THOSE CALLED BY MODULES EXTERNAL TO SYSTEM INTERFACE ARE FLAGGED BY '*', IF THE REGISTERS ARE SAVED ON ENTRY IN THE TEMP STACK (NOT ONE OF THE NORMAL LOGO STACKS; TYPICALLY I/O ERROR AND TRAP HANDLING MODULES), THE MODULE IS FLAGGED WITH '@'; IF NO REGISTERS ARE SAVED, THE MODULE IS FLAGGED BY '+'. THOSE MODULES ENTERED IMPICITLY BY THE MONITOR ARE FLAGGED BY A '$' (TYPICALLY ERROR SERVICE ROUTINES).

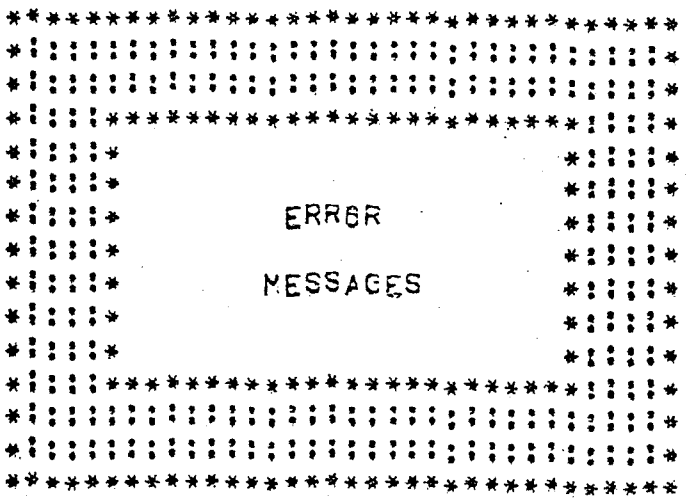
SYSINT*+	CALLED FROM:	EXTERNAL
	CALLS:	NONE
SYSLOAD*	CALLED FROM:	EXTERNAL
	CALLS:	SYSREAD, SYSXMSG
SYSIN*	CALLED FROM:	EXTERNAL
	CALLS:	SYSREAD
SYSREAD+	CALLED FROM:	SYSLOAD, SYSIN
	CALLS:	SYSERR
SYSOUT*	CALLED FROM:	EXTERNAL, SYSERR, SYSXMSG
	CALLS:	SYSWRITE
SYSTORE*	CALLED FROM:	EXTERNAL
	CALLS:	SYSWRITE
SYSDIAG	CALLED FROM:	SYSERR
	CALLS:	SYSWRITE
SYSWRITE+	CALLED FROM:	SYSOUT, SYSTORE, SYSDIAG
	CALLS:	SYSERR
SYSLSET*	CALLED FROM:	EXTERNAL
	CALLS:	SYSXSET

SYSSSET*	CALLED FROM:	EXTERNAL
	CALLS:	SYSXSET
SYSXSET+	CALLED FROM:	SYSLSET, SYSSSET
	CALLS:	NONE
SYSERR*@	CALLED BY:	EXTERNAL, SYSTRAP, SYSREAD, SYSWRITE, SYSIOER
	CALLS:	SYSOUT, SYSDIAG
SYSERT#	REFERENCED BY:	SYSERR
SYSTRAP$	CALLED BY:	MONITOR
	CALLS:	STACKTR (IN STORAGE MANAGEMENT) SYSERR
SYSBRK$	CALLED BY:	MONITOR
	CALLS:	NONE
SYSCLOCK**	CALLED BY:	EXTERNAL, SYSERR
	CALLS:	NONE
SYSXMSG	CALLED BY:	SYSLOAD
	CALLS:	SYSOUT, SYSERR
SYSIOER$@	CALLED BY:	MONITOR
	CALLS:	SYSERR
SYSIOET#	REFERENCED BY:	SYSIOER
SYSCTX*#	REFERENCED BY:	EXTERNAL, ALL
SYSDAT1#	REFERENCED BY:	ALL

LEGEND:

*	--	REFERENCED EXTERNALLY
#	--	DATA SECTION
@	--	SAVES REGISTERS ON TEMP STACK
+	--	DOES NOT SAVE REGISTERS
$	--	CALLED IMPLICITLY BY MONITOR

SPECIAL NOTE: SYSBREAK AND SYSTRAP NEED NOT SAVE REGISTERS, AS THIS IS DONE BY THE MONITOR. SYSCLOCK NEED NOT SAVE REGISTERS, AS NO REGISTERS ARE REFERENCED.

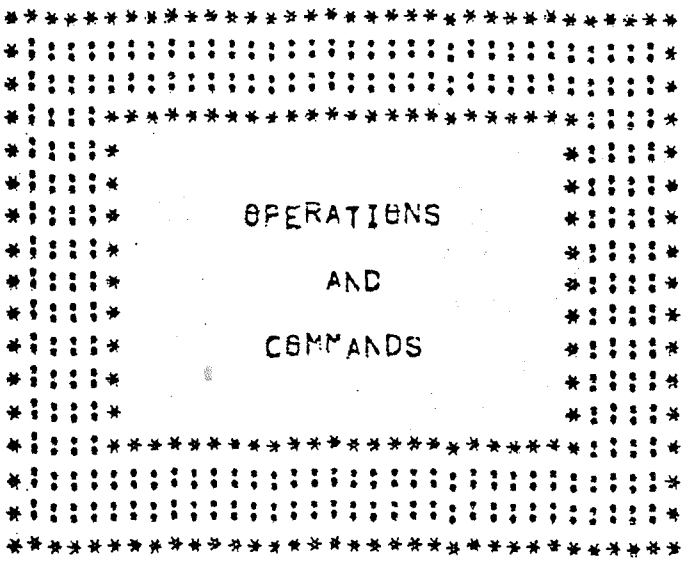

ERROR
MESSAGES

```
ERMSG1      ' IS NOT A NUMBER.'
ERMSG2      ' IS EXTRA.'
ERMSG3      'SOMETHING IS MISSING.'
ERMSG4      'LINE NUMBER IS TOO BIG.'
ERMSG5      'A LINE NUMBER CAN''T BE ZERO.'
ERMSG6      'YOU''RE NOT EDITING ANYTHING.'
ERMSG7      'I NEED A MATCHING /.'
ERMSG8      'I NEED A MATCHING  .'
ERMSG9      'I NEED A MATCHING ).'
ERMSG10     'I NEED A MATCHING (.'
ERMSG11     ' NEEDS A MEANING.'
ERMSG12     'YOU CAN''T EDIT TWO PROCEDURES AT ONCE.'
ERMSG13     ' IS ALREADY DEFINED.'
ERMSG14     ' DOES NOT EXIST.'
ERMSG15     'FILE DOES NOT EXIST.'
ERMSG16     'MY STORAGE IS FULL.'
ERMSG17     'YOU CAN''T CHANGE A SYSTEM THING.'
ERMSG18     'YOU CAN''T CHANGE A SYSTEM PROCEDURE.'
ERMSG19     'YOU CAN''T MAKE A WORD OUT OF A SENTENCE.'
ERMSG20     'INPUT MUST BE NUMERIC.'
ERMSG21     'INPUT MUST BE TRUE OR FALSE.'
ERMSG22     ' IS NOT A LINE NUMBER.'
ERMSG23     'RESULT TOO LONG.'
```

```
ERMSG24     'YOU INTERRUPTED ME.'
ERMSG25     'COMMAND VALID IN PROCEDURES ONLY.'
ERMSG26     'FOO ON YOU.'
ERMSG27     'WORD OR SENTENCE TOO LONG.'
ERMSG28     '.....OOPS  LOGO GOOFED  '
ERMSG29     'THAT LINE DOESN'T EXIST.'
ERMSG30     ' DEFINED.'
ERMSG31     'DON'T USE THE EMPTY WORD FOR A NAME.'
ERMSG32     'NOT HERE, YOU DON'T...'
ERMSG33     '--UNIMPLEMENTED INSTRUCTION (TRAP 41) AT '
ERMSG34     '--STACK OVERFLOW (TRAP 42) AT '
ERMSG35     '--FIXED POINT OVERFLOW (TRAP 43) AT '
ERMSG36     '--FLOATING POINT FAULT (TRAP 44) AT '
ERMSG37     '--DECIMAL ARITHMETIC FAULT (TRAP 45) AT '
ERMSG38     '--NONEXISTENT INSTRUCTION (TRAP 40/8) AT '
ERMSG39     '--PRIVILEGED INSTRUCTION (TRAP 40/2) AT '
ERMSG40     '--MEMORY PROTECT VIOLATION (TRAP 40/1) AT '
ERMSG41     '--CAL2 INSTRUCTION (TRAP 49) AT '
ERMSG42     '--CAL3 INSTRUCTION (TRAP 4A) AT '
ERMSG43     '--CAL4 INSTRUCTION (TRAP 4B) AT '
ERMSG44     '--UNKNOWN TRAP AT.'
ERMSG45     '--TOO MANY CONTINUATION CARDS FOR INPUT'
ERMSG46     '--ERROR IN UNKNOWN DCB: BYE'
ERMSG47     '  I/O ERROR IN BATCH JOB: BYE'
ERMSG48     '  BYE'
ERMSG49     '  FATAL I/O ERROR; BYE'
ERMSG50     '--EXCESSIVE LENGTH OUTPUT TO SYSWRITE'
```

```
ERMSG51     'YOU MUST HAVE / MARKS AROUND EACH INPUT.'
ERMSG52     'TO WHAT '
ERMSG53     ' CAN'T BE A PROCEDURE NAME.'
ERMSG54     ' IS ALREADY DEFINED.'
ERMSG55     'I CAN'T TELL YOU ANY MORE ABOUT ERROR #'
ERMSG56     'EDIT WHAT '
ERMSG57     'YOU CAN'T EDIT A SYSTEM PROCEDURE.'
ERMSG58     'YOU CAN'T DIVIDE BY ZERO.'
ERMSG59     'YOU CAN'T LIST A SYSTEM PROCEDURE.'
ERMSG60     'LIST WHAT   YOU'RE NOT EDITING ANYTHING.'
ERMSG61     'NO EXTENDED ERROR MESSAGES AVAILABLE'
ERMSG62     'A FILE NAME MUST BE A WORD.'
ERMSG63     'YOU CAN'T ERASE A SYSTEM PROCEDURE.'
ERMSG64     'ERASE WHAT   YOU'RE NOT EDITING ANYTHING.'
ERMSGXX     '--UNKNOWN ERROR CODE '
IMNSI       '  INSUFFICIENT INFORMATION TO OPEN FILE (01,CC)'
IMNSF       'THAT FILE DOES NOT EXIST'
IMCUF       'YOU CAN'T USE THAT FILE'
IMFIB       'THAT FILE IS BUSY'
IMEOD       '  END OF DATA (05,CC)'
IMEOF       '  END OF FILE (06,CC)'
IMDWL       '  DATA WAS LOST (07,CC)'
IMKNF       '  KEY NOT FOUND (43,CC)'
IMNLP       '  LINE-PRINTER USE NOT ALLOWED (48,CC)'
IMOOR       '  OUT OF FILE SPACE ON RAD/DISC (56,CC/57,CC)'
```

```
IMRE            '  READ ERROR (41,CC)'
IMPR            '  READ ERROR (41,C4)'
IMWE            '  WRITE ERROR (45,CC)'
```


OPERATIONS
AND
COMMANDS

OPERATIONS AND COMMANDS

EACH OPERATION AND COMMAND WILL BE A SEPARATE TINY MODULE THAT EXECUTE CAN CALL.

1. COPIES STRING CONTAINING REST OF INPUT LINE TO THE INPUT BUFFER (SI:IBUF) FOR PARSING AND RESTORES THE PARENTHESIS COUNT.

2. CHECKS THE EDIT FLAG. IF NOT EDIT MODE, ISSUES ERROR, RELEASES STRING CONTAINING REST OF INPUT LINE, AND RETURNS TO THE EXECUTER VIA ERROR RETURN.

3. ALLOCATES SPT BLOCK FOR THE PROCEDURE, LINKS STRING CONTAINING REST OF INPUT LINE TO SPT BLOCK WITH LINE NUMBER = 0 (TITLE LINE).

4. GETS FIRST NON-BLANK CHARACTER FROM SI:IBUF. THIS SHOULD BE THE FIRST LETTER OF THE PROCEDURE NAME OR NOISE WORD; A SEMICOLON DENOTING A COMMENT; OR A PARENTHESIS. ANYTHING ELSE FLAGS AS AN ERROR. IF NOT AN ERROR, THE REST OF THE NAME IS BUILT UP, AND A SYMBOL TABLE LOOKUP IS PERFORMED. IF THE NAME EXISTS AS A USER PROC, SYS PROC OR COMMAND (NICKTYP) IT IS AN ERROR. ELSE, THE NAME OF THE PROCEDURE IS COPIED TO STRING STORAGE AND LINKED TO THE PROC SPT WITH LINE NUMBER 20000. AN SPT ENTRY WITH LINE NUMBER = 30000 IS ADDED TO THE END OF THE PROC SPT AND LINKED TO ONE WORD OF STRING STORAGE. THE BOTTOM HALF OF THIS WORD WILL BE THE TRACE FLAG FOR THE PROCEDURE. IT IS INITIALLY CLEARED (NO TRACE).

5. THE REMAINING ITEMS ON THE LINE SHOULD BE LOCAL NAMES (ENCLOSED BY SLASHES), PARENTHESIS, COMMENTS OR NOISE WORDS. IF ANYTHING ELSE IS ENCOUNTERED, IT IS FLAGGED AS AN ERROR. FOR EACH LOCAL NAME, COPY NAME TO THE STRING STORAGE LINKED TO THE SPT BLOCK WITH LINE NUMBER = 10000.

   WHEN THE END OF THE INPUT LINE IS REACHED, THE PARENTHESIS COUNT IS CHECKED. IF NON-ZERO, AN ERROR IS ISSUED. ELSE, THE EDIT FLAG IS SET AND CONTROL TRANSFERRED BACK TO THE EXECUTER.

6. IN THE CASE OF ERRORS (EXCEPT THE ONE GENERATED BY A FALSE EDIT FLAG), THE SPT ALLOCATED FOR THE PROCEDURE AND ALL STRINGS LINKED TO THE SPT ARE DEALLOCATED AND THE EDIT FLAG CLEARED PRIOR TO EXIT.

TITLE (CHANGE PROCEDURE TITLE--NOT YET IMPLEMENTED)

1. TEST 'EDIT' FLAG. IF NOT SET, GENERATE 'YOU'RE NOT EDITING ANYTHING' ERROR MESSAGE AND BRANCH TO 'T0', PARAGRAPH 9.

2. LOCATE THE ZERO ENTRY (TITLE LINE) OF THE SPT BLOCK OF THE PROCEDURE CURRENTLY BEING EDITED. CHANGE THE LINE NUMBER TO -1.

3. CHANGE ALL OCCURRENCES OF LINE NUMBERS EQUAL TO 10000 TO -1 IN THE CURRENTLY-OPEN SPT BLOCK.

4. BRANCH TO PARAGRAPH 4 TO 'T0'.

5. RETURN TO THE PARSER.

1. COPIES REST OF INPUT LINE TO INPUT BUFFER.

2. CHECKS THE EDIT FLAG. IF SET, ISSUES 'YOU CAN'T EDIT TWO PROCEDURES AT ONCE' MESSAGE AND RETURNS TO THE EXECUTER.

3. BUILDS UP NEXT ELEMENT ON THE LINE. IF NOISE, IGNORES; IF NUMBER OR FIRST CHARACTER IS A QUOTE OR SLASH, ISSUES 'XXX CAN'T BE PROC NAME' AND RETURNS TO THE EXECUTER; IF A SYSTEM PROC OR COMMAND, ISSUES 'YOU CAN'T EDIT SYS PROCS' AND RETURNS TO THE EXECUTER; IF NOT DEFINED, ISSUES 'XXX NEEDS A MEANING' AND RETURNS TO THE EXECUTER; BYPASSES COMMENTS AND PARENTHESIS, UNLESS UNDERFLOW IS ENCOUNTERED. IF THE ELEMENT IS A USER PROCEDURE, CONTINUES WITH THE NEXT STEP.

4. SAVES PROCEDURE NAME (IN SI:OBUF) AND THE SPT POINTER (IN C:SPTPX). CHECKS THE REST OF THE ITEMS ON THE LINE. IF EACH ELEMENT IS NOT NOISE NOR PARENTHESIS NOR COMMENTS, ISSUES 'EDIT WHAT ' AND RETURNS TO THE EXECUTER.

5. CHECKS THE PARENTHESIS COUNT. IF NON-ZERO, ISSUES ERROR AND RETURNS TO THE EXECUTER.

6. COPIES THE SPT TO A NEW SPT BLOCK (POINTED TO BY P:TOSPT) AND DELETES THE NAME OF THE PROCEDURE FROM THE SYMBOL TABLE (WHICH FREES THE OLD SPT AND STRINGS). THE NAME IS RE-ADDED WHEN THE USER TYPES 'END', SETS THE EDIT FLAG AND RETURNS TO THE EXECUTER.

BEFORE RETURNING TO THE EXECUTER, THE STRING CONTAINING THE REMAINDER OF THE INPUT LINE IS FREED.

END (STOP EDITING OR DEFINING A PROCEDURE).

1. CHECKS EDIT FLAG. IF NOT IN EDIT MODE, ISSUES ERROR AND RETURNS TO THE EXECUTER.

2. RESETS THE EDIT FLAG.

3. SORTS THE SFT ENTRIES SO THAT EXECUTION OF THE PROCEDURE WILL OCCUR IN THE PROPER ORDER, AND DELETES ALL ENTRIES WITH LINE NUMBER = -1.

4. PRINTS 'XXX DEFINED'.

5. ADDS PROCEDURE NAME TO THE SYMBOL TABLE WITH NUMBER OF INPUTS REQUIRED, AND RETURNS TO THE EXECUTER.

THIS IS A TEMPORARY VERSION OF 'ERASE'. THIS VERSION WILL ERASE THE CURRENTLY-EDITED PROCEDURE OR A LINE WITHIN THE CURRENTLY-EDITED PROCEDURE, ALL PROCEDURES, OR A NAMED PROCEDURE. CHECKING IS DONE FOR THE PROPER SETTING OF THE EDIT FLAG, THE VALIDITY OF THE PROCEDURE NAMES AND LINE NUMBERS, ETC., BUT PARENTHESIS BALANCE AND GARBAGE ON THE END OF THE INPUT LINE IS NOT LOOKED AT. COMMENTS AND NOISE WORDS MAY BE EMBEDDED IN THE INPUT LINE.

THE 4 FORMATS ARE:
>ERASE

THIS FORM ERASES THE CURRENTLY-EDITED PROCEDURE AND RETURNS LOGO TO 'SERVICE' MODE.

>ERASE NNNN

THIS FORM WILL ERASE LINE NNNN OF THE CURRENTLY-EDITED PROCEDURE IF LINE NNNN EXISTS IN THAT PROCEDURE.

>ERASE ALL

ALL PROCEDURES, ABBREVIATIONS, NAMES, ARE ERASED AND LOGO IS RESTARTED.

>ERASE XXX

THIS FORM ERASES THE USER PROCEDURE XXX.

THIS IS A TEMPORARY VERSION OF 'LIST'. THIS VERSION WILL LIST THE CURRENTLY-EDITED PROCEDURE OR A LINE WITHIN THE CURRENTLY-EDITED PROCEDURE, ALL PROCEDURES, OR A NAMED PROCEDURE. CHECKING IS DONE FOR THE PROPER SETTING OF THE EDIT FLAG, THE VALIDITY OF PROCEDURE NAMES AND LINE NUMBERS, ETC., BUT PARENTHESIS BALANCE AND GARBAGE ON THE END OF THE INPUT LINE IS NOT LOOKED AT. COMMENTS AND NOISE WORDS MAY BE EMBEDDED IN THE INPUT LINE.

THE 4 FORMATS ARE:

>LIST

THIS FORM WILL LIST ALL THE LINES OF THE CURRENTLY-EDITED PROCEDURE.

>LIST NNNN

THIS FORM WILL LIST LINE NNNN OF THE CURRENTLY-EDITED PROCEDURE IF LINE NNNN EXISTS IN THAT PROCEDURE.

>LIST ALL

THIS FORM WILL LIST ALL USER PROCEDURES.

>LIST XXX

THIS FORM WILL LIST THE USER PROCEDURE XXX.

THIS IS A TEMPORARY VERSION OF 'TRACE' AND 'UNTRACE'. THIS VERSION WILL TRACE OR UNTRACE THE CURRENTLY-EDITED PROCEDURE, ALL PROCEDURES, OR A NAMED PROCEDURE. CHECKING IS DONE FOR THE PROPER SETTING OF THE EDIT FLAG, THE VALIDITY OF THE PROCEDURE NAMES, ETC., BUT PARENTHESIS BALANCE AND GARBAGE ON THE END OF THE INPUT LINE IS NOT LOOKED AT. COMMENTS AND NOISE WORDS MAY BE EMBEDDED IN THE INPUT LINE.

THE 3 FORMATS ARE:

```
>TRACE
>UNTRACE
```

THIS FORM SETS OR RESETS THE TRACE FLAG FOR THE CURRENTLY-EDITED PROCEDURE.

```
*TRACE ALL
*UNTRACE ALL
```

THIS FORM SETS OR RESETS THE TRACE FLAG FOR EACH USER PROCEDURE.

```
*TRACE XXX
*UNTRACE XXX
```

THIS FORM SETS OR RESETS THE TRACE FLAG FOR PROCEDURE XXX.

NOTE - EVENTUALLY, 'UNTRACE' WILL BE REMOVED ALTOGETHER, SINCE ERASING PROCEDURE TRACES SHOULD BE A FUNCTION OF 'ERASE'.

THESE TWO ROUTINES TAKE ONE INPUT - A WORD. THIS WORD IS THE FILE NAME IN STANDARD UTS FORMAT (FILENAME.ACCT.PASSWORD). EACH ROUTINE ALTERS THE SOURCE OF I/O - 'LOAD' CHANGES THE INPUT STREAM FROM THE TTY TO THE NAMED FILE, AND 'STORE' CHANGES THE OUTPUT STREAM FROM THE TTY TO THE NAMED FILE. 'STORE' CHECKS TO SEE THAT THERE WAS NO PROBLEM IN OPENING THE NAMED OUTPUT FILE, AND IF THERE WAS, BRANCHES TO 'XSERR', OTHERWISE, A BRANCH TO 'LISTALL' IS PERFORMED.

PLEASE NOTE THAT 'STORE' SAVES ONLY THE PROCEDURES. IT IS EASY TO MAKE AN ADDITION TO SAVE THE TRUTH FLAG, ABBREVIATIONS AND NAMES PRIOR TO BRANCHING TO 'LISTALL'.

GETREST

COPIES REMAINDER OF INPUT LINE TO THE INPUT BUFFER FOR PARSING. USED IN COMMANDS TO RETRIEVE STRING CONTAINING REMAINDER OF LINE WHICH WAS CREATED WHEN THE COMMAND WAS ORIGINALLY ENCOUNTERED, RESTORES PARENTHESIS COUNT AND RETURNS REST-OF-LINE STORAGE, ALSO.

PRINTS 'XXX CAN'T BE A PROCEDURE NAME' IF THE FIRST CHARACTER OF THE NAME IS A QUOTE OR A SLASH, OR THE NAME IS UNSIGNED NUMERIC. NUMBER LITERALS ARE HANDLED SLIGHTLY DIFFERENTLY FROM QUOTED LITERALS OR NAMES, SO R5 IS USED AS A FLAG (0 IF NOT A NUMBER LITERAL). R0 CONTAINS THE CHARACTER WHICH CAUSED ENTRY (QUOTE OR SLASH, DOES NOT APPLY IF A NUMBER LITERAL), R2 AND R4 ARE CLOBBERED.

IF A NUMBER LITERAL IS PROCESSED & IS INTENDED AS A NAME, THE MESSAGE 'NNN CAN'T BE A PROCEDURE NAME' IS OUTPUT. THIS ROUTINE SETS THE NUMBER LITERAL FLAG, SETS RC TO QUOTE, SETS UP THE INPUT INDEX AND THE INDEX TO SI:INSM IN PREPARATION FOR 'BADNAME'.

PRINTS THE TITLE LINE AND ALL THE LINES OF A PRO-
CEDURE WHOSE SPT IS POINTED TO BY C:SPTPX (UN-
BIASED).

R_0, R6, R7, SF1 AND C:SPTPX ARE ALTERED.

PRINTS THE PROCEDURE LINE WHOSE SPT ENTRY IS POINTED TO BY C:SPTPX. THE LINE NUMBER IS FIRST CONVERTED TO TEXT AND MOVED TO SI:OBUF, THEN THE SPT ENTRY POINTER IS FOLLOWED AND THE PROCEDURE LINE TEXT IS MOVED AND THE RESULT IS PRINTED USING SYSOUT. THERE ARE 2 ENTRY POINTS - C:PLINE IS THE MOST COMMON, AND IS USED WHEN THE LINE TO BE PRINTED IS NOT THE TITLE LINE; C:PPROC4 IS USED WHEN THE LINE TO BE PRINTED IS THE TITLE LINE AND THE WORD 'TO' HAS BEEN ALREADY MOVED TO THE OUTPUT BUFFER. R0 CONTAINS THE LINE NUMBER IN BINARY (IF C:PLINE IS ENTERED).

R_6, R7 AND SR_1 ARE MODIFIED.

PRINTS CARRIAGE RETURN, LINE-FEED ON OUTPUT DEVICE.
THIS IS ACCOMPLISHED BY LOADING THE OUTPUT BUFFER
WITH A SPACE AND CALLING SYSOUT. R0 IS LOST.

PRINTS 'END' ON THE OUTPUT DEVICE. USED AFTER A PROCEDURE HAS BEEN LISTED. ALTERS R6,R7 AND SR1.

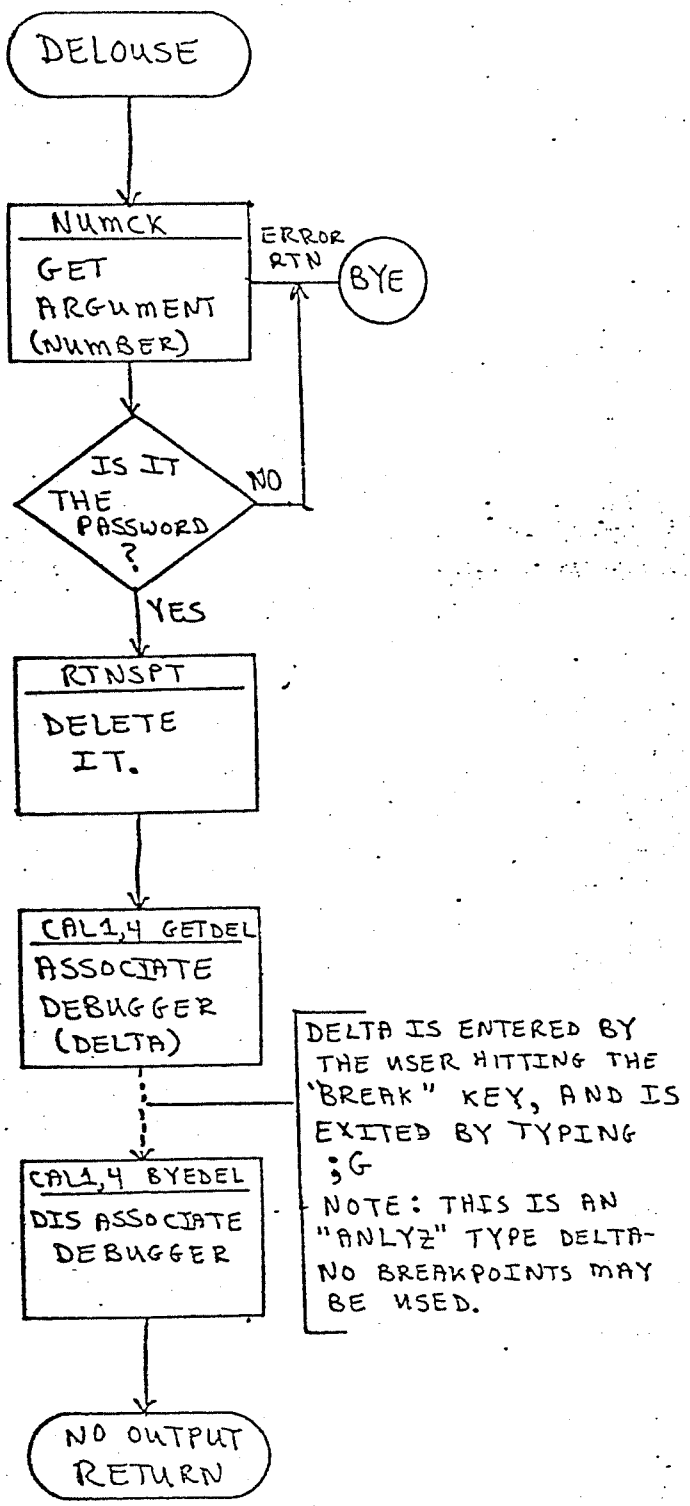
SETS OR RESETS THE TRACE FLAG FOR THE USER PRO-
CEDURE POINTED TO BY C:SPTPX, DEPENDING ON THE
CONTENTS OF D4 (C MEANS TRACE, NON-C MEANS ERASE
TRACE). R0 AND R2 ARE ALTERED.

BREAKS UP THE INPUT TO 'LOAD' AND 'STORE' INTO ITS FILE NAME COMPONENTS (NAME, ACCOUNT AND PASSWORD). IT POPS THE SPT POINTER TO THE FILE NAME OFF THE SSTACK. IF THE FILE NAME SPECIFICATION IS NOT A LEGO WORD, AN ERROR MESSAGE IS GENERATED AND RTN IS TAKEN. ELSE, IT INITIALIZES THE FILE NAME, ACCOUNT AND PASSWORD DATA AREAS TO BLANKS AND PROCEEDS. RTN+1 IS THEN TAKEN.

THE SPT POPPED OFF THE SSTACK IS RETURNED TO FREE STORAGE PRIOR TO RETURNING.

CALLS DELTA TO BEGIN DEBUGGING (OF LOGO NOT USER PROGRAMS)

CALLED BY EXECUTE -- REQUIRES PASSWORD OR USER WILL BE THROWN OFF THE SYSTEM.

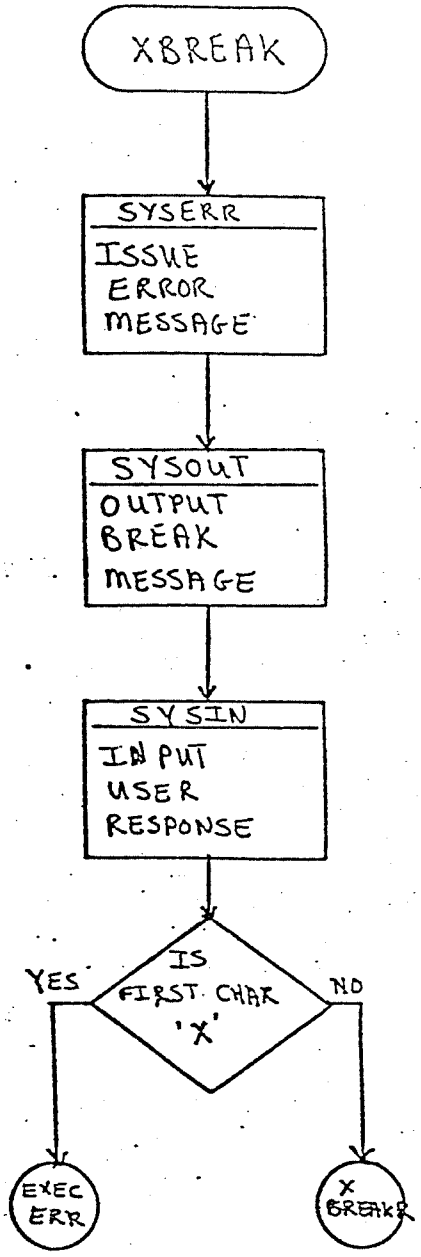

DELOUSE
NUMCK
GET ARGUMENT (NUMBER)
ERROR RTN
BYE
IS IT THE PASSWORD ?
NO
YES
RTNSPT
DELETE IT.
CAL1,4 GETDEL
ASSOCIATE DEBUGGER (DELTA)
DELTA IS ENTERED BY THE USER HITTING THE "BREAK" KEY, AND IS EXITED BY TYPING ;G
NOTE: THIS IS AN "ANLYZ" TYPE DELTA- NO BREAKPOINTS MAY BE USED.
CAL1,4 BYEDEL
DIS ASSOCIATE DEBUGGER
NO OUTPUT RETURN

BREAK KEY HANDLER

TELLS THE USER WHERE HE WAS AND GIVES HIM THE CHOICE OF EITHER CONTINUING OF STARTING OVER

CALLED BY EXECUTE

CALLED ROUTINES: SYSERR, SYSOUT, SYSIN


XBREAK
SYSERR
ISSUE
ERROR
MESSAGE
SYSOUT
OUTPUT
BREAK
MESSAGE
SYSIN
INPUT
USER
RESPONSE
IS
FIRST CHAR
'X'
YES
NO
EXEC
ERR
X
BREAK

GOODBYE

CALLED BY EXECUTER, SETS ERROR CODE TO ZERO AND CALLS ERROR MESSAGE ROUTINE TO EXIT TO SYSTEM.

INTERNAL SPECIFICATIONS

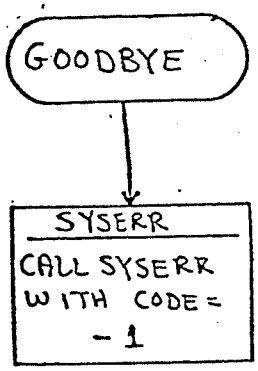
NAME: GOODBYE

FUNCTION: TERMINATE LOGO SESSION AND RETURN USER TO SYSTEM LEVEL

CALLED BY: EXECUTER

CALLING SEQUENCE: BAL,LINK GOODBYE
(NEVER RETURNS)

CALLED ROUTINES: SYSERR



BYE

CALLED BY EXECUTER, SETS UP ERROR CODE AND CALLS ERROR ROUTINE TO LOG USER OFF SYSTEM.

INTERNAL SPECIFICATIONS

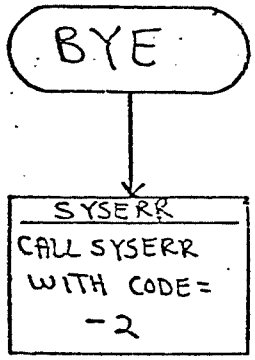
NAME: BYE

FUNCTION: TERMINATES LOGG SESSION, LOGS USER OFF UTS SYSTEM

CALLED BY: EXECUTER

CALLING SEQUENCE: BAL,LINK BYE

CALLED ROUTINES: SYSERR



CALLED BY EXECUTER, RETURNS TO PROCEDURE END POINT IN EXECUTER AFTER CLEARING ALL VALUES ON S-STACK

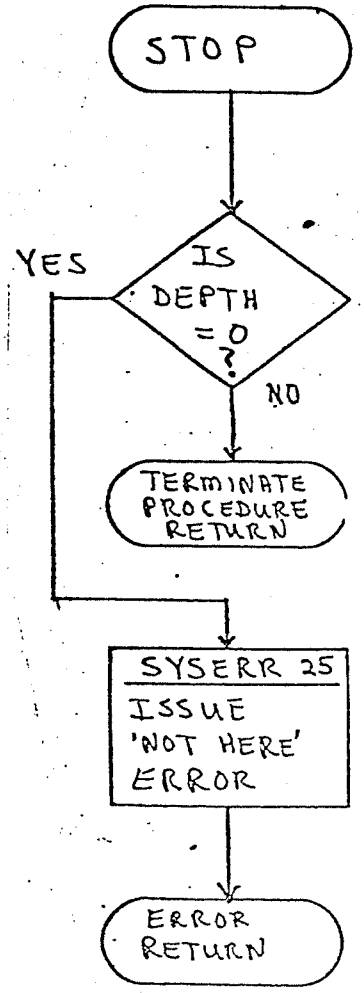
INTERNAL SPECIFICATIONS

NAME: STOP

CALLING SEQUENCE: BAL,LINK STOP

CALLED BY: EXECUTE

CALLED ROUTINES: NONE



CALLED BY EXECUTER, POPS ONE ITEM OFF S.STACK AND RETURNS TO EXECUTER

INTERNAL SPECIFICATIONS

NAME: IGNORE

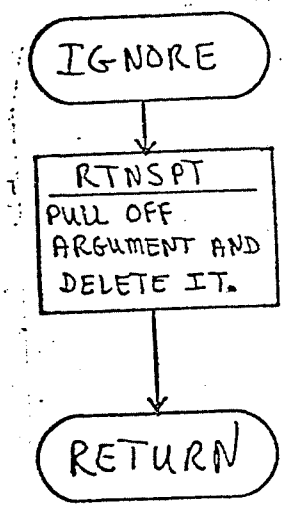
CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK IGNORE

( )

(ALWAYS RETURN HERE)

CALLED ROUTINES: RTNSTR


IGNORE
RTNSPT
PULL OFF ARGUMENT AND DELETE IT.
RETURN

OUTPUT

CALLED BY EXECUTER, RETURNS TO PROCEDURE END POINT IN EXECUTER, LEAVING S-STACK UNCHANGED

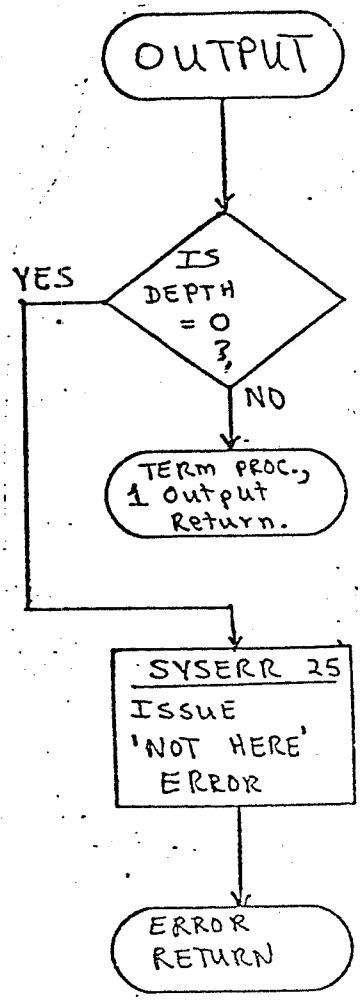
INTERNAL SPECIFICATIONS

NAME: OUTPUT

CALLING SEQUENCE: BAL,LINK OUTPUT

CALLED BY: EXECUTE

CALLED ROUTINES: NONE



PRINT

PRINTS ITS INPUT ON THE TERMINAL AND DOES A CR-LF AT THE END

TYPE

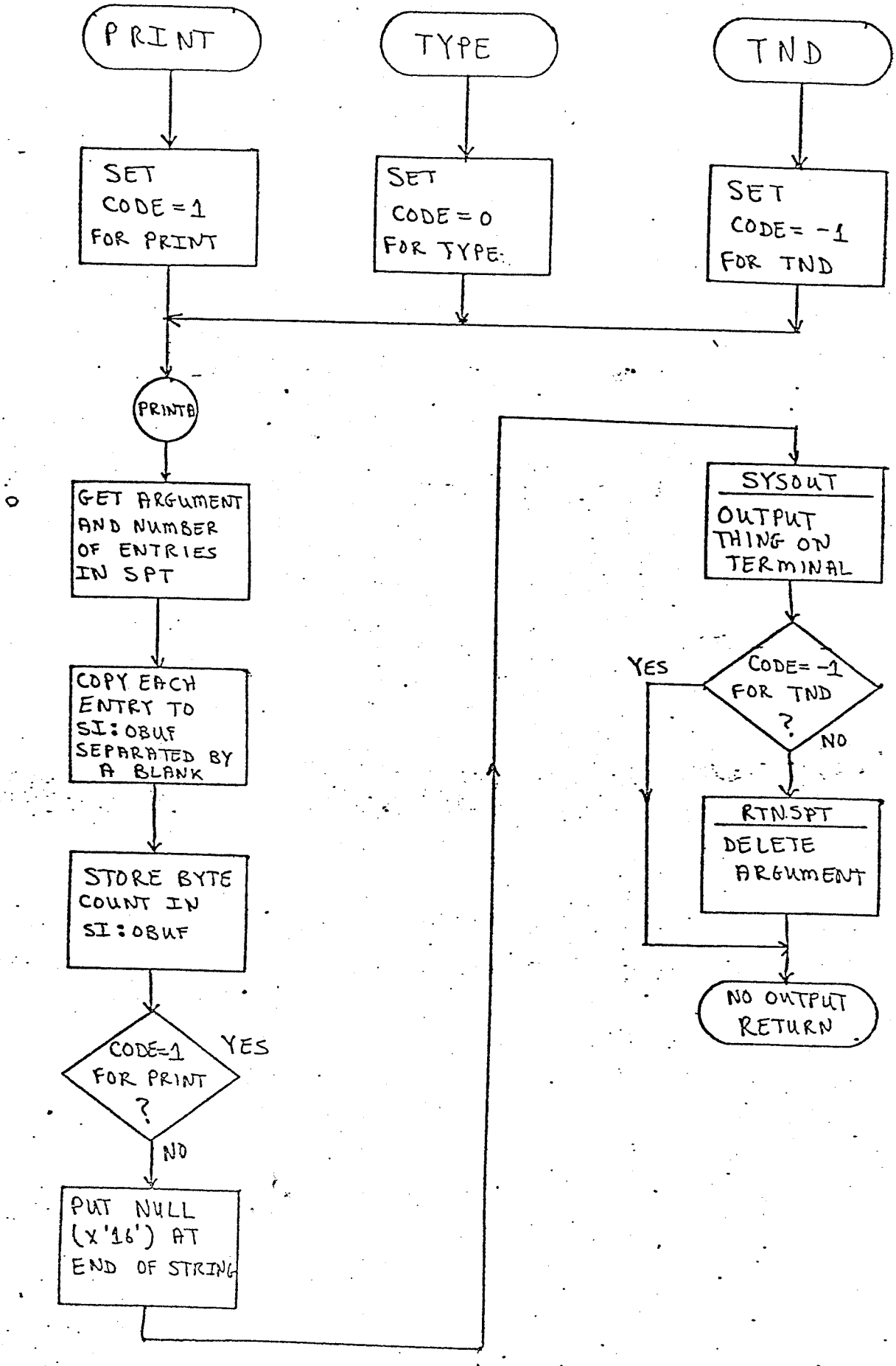
PRINTS ITS INPUT ON THE TERMINAL BUT DOES NOT DO A CR-LF

TND

TND IS AN INTERNAL ENTRY FOR TYPE BUT DON'T DELETE.

CALLED ROUTINES: COPY, SYSOUT, RTNSPT

FLOWCHART FOLLOWS


PRINT
TYPE
TND
SET CODE = 1 FOR PRINT
SET CODE = 0 FOR TYPE
SET CODE = -1 FOR TND
PRINTB
GET ARGUMENT AND NUMBER OF ENTRIES IN SPT
COPY EACH ENTRY TO SI:OBUF SEPARATED BY A BLANK
STORE BYTE COUNT IN SI:OBUF
CODE=1 FOR PRINT ?
YES
NO
PUT NULL (X'16') AT END OF STRING
SYSOUT
OUTPUT THING ON TERMINAL
CODE= -1 FOR TND ?
YES
NO
RTNSPT
DELETE ARGUMENT
NO OUTPUT RETURN

GBTB

CALLED BY EXECUTER, TAKES TOP ITEM FROM S-STACK IF IT IT IS A NUMBER, SETS THE NEXT LINE NUMBER FOR THE PARSER TO THE NUMBER AND RETURNS, OTHERWISE CALL ERROR ROUTINE AND RETURN TO EXECUTER.

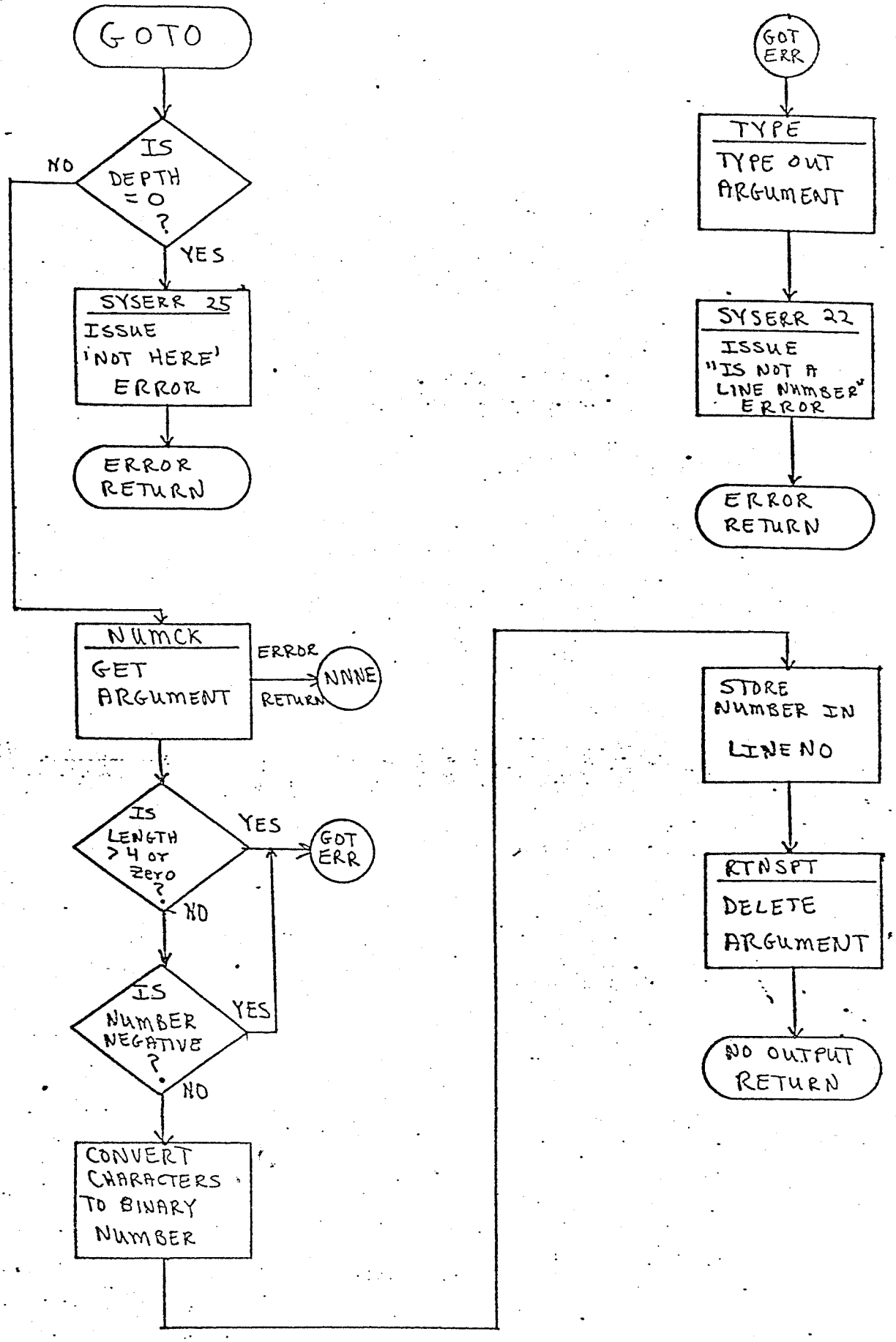
INTERNAL SPECIFICATIONS

NAME: GBTB

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK GBTB

CALLED ROUTINES: NUMCK


GOTO
IS DEPTH = 0 ?
NO
YES
SYSERR 25
ISSUE 'NOT HERE' ERROR
ERROR RETURN
GOT ERR
TYPE
TYPE OUT ARGUMENT
SYSERR 22
ISSUE "IS NOT A LINE NUMBER" ERROR
ERROR RETURN
NUMCK
GET ARGUMENT
ERROR
RETURN
NNNE
IS LENGTH > 4 or zero ?
YES
GOT ERR
NO
IS NUMBER NEGATIVE ?
YES
NO
CONVERT CHARACTERS TO BINARY NUMBER
STORE NUMBER IN LINE NO
RTNSPT
DELETE ARGUMENT
NO OUTPUT RETURN

I. EXTERNAL ABSTRACT

A. BRIEF FUNCTIONAL SPECIFICATIONS

THE ARITHMETIC PACKAGE EVALUATES THE VERBS SUM, ABS, AND DIFFERENCE; THESE ARE THE SUM OF TWO NUMBERS, THE DIFFERENCE OF TWO NUMBERS, AND THE ABSOLUTE VALUE OF A NUMBER. A NUMBER IS AN ARBITRARY LENGTH CHARACTER STRING COMPOSED OF THE CHARACTERS 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 AND OPTIONALLY PRECEDED BY A SIGN CHARACTER + OR -.

B. DATA STRUCTURE AND FLOW

EACH FUNCTION IN THE ARITHMETIC PACKAGE REMOVES THE POINTER(S) TO ITS ARGUMENT(S) FROM THE S-STACK AND PLACES A POINTER TO ITS RESULT ON THE S-STACK, IF THE FUNCTION COMPLETES NORMALLY. ALL DATA IS PASSED BY THE S-STACK; NO DATA IS NEEDED FROM OTHER MODULES.

C. ENTRY POINTS AND FUNCTIONS

ENTRY POINT	FUNCTION	# OF ARGUMENTS
NABS	ABSOLUTE VALUE	1
NSUM	ADDITION (A+B)	2
NDIFF	SUBTRACTION (A-B)	2
PRODUCT	PRODUCT (A*B)	2

D. CALLING AND CALLED ROUTINES

IT IS EXPECTED THAT THESE FUNCTIONS WILL ONLY BE CALLED BY EXECUTE; THEY CAN BE CALLED BY ANY ROUTINE NEEDING THESE SERVICES, HOWEVER. THE FOLLOWING ARE NAMES AND BRIEF FUNCTIONAL SPECIFICATIONS OF SUBROUTINES CALLED BY THEM.

FUNCTION ROUTINES:

NUMCK	CHECK TO SEE IF ARGUMENT IS A NUMBER
NADD	ADDS TWO NUMBERS (A+B)
NSUB	SUBTRACTS TWO NUMBERS (A-B)
NTRIM	REMOVES NON-SIGNIFICANT LEADING ZEROS, PLACES POINTER TO RESULT ON S-STACK.
NSWAP	SWITCH ARGUMENTS A AND B

E. ERROR CONDITIONS

THE ONLY ERROR CONDITION DETECTABLE BY THESE FUNCTIONS IS CAUSED BY AN ARGUMENT NOT BEING OF TYPE NUMBER. IF AN ARGUMENT IS NOT OF TYPE NUMBER, THE STRING IS TYPED OUT, FOLLOWED BY 'IS NOT A NUMBER' AND AN ERROR RETURN IS TAKEN TO THE CALLING ROUTINE.

II. INTERNAL SPECIFICATIONS

A. INTERMODULE COMMUNICATIONS, CALLING SEQUENCES, ETC.

SUBROUTINE NUMCK RETURNS ITS RESULTS IN REGISTERS T_0, T_1, AND T2. OTHER ROUTINES COMMUNICATE USING THE FOLLOWING DATA BLOCK, WHERE A, B, C ARE THE FIRST AND SECOND ARGUMENTS AND C IS THE RESULT OF THE OPERATION.

NUMBA POINTER FROM S-STACK TO DATA STRING
+1 BYTE OFFSET FROM STRING AREA TO DATA STRING
+2 LENGTH IN BYTES OF STRING
+3 1 IF SIGN IS MINUS

ALL SUBROUTINES ARE CALLED THROUGH THE LINK REGISTER. THE GENERAL FORMAT OF THE LINK IS:

```
L0C     BAL,LINK     SUBROUTINE
L0C+1   B            ERROR-RETURN
L0C+2     *****NORMAL RETURN*****
```

B. INTERNAL DATA STRUCTURES

THE NUMBERS OPERATED ON AND RETURNED AS RESULTS ARE OF THE FORM

NUMBER:= SIGN | DIGITS
SIGN:= '+' | '-' | NO SIGN CHARACTER
DIGITS:= DIGIT | DIGITS
DIGIT:= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | '0'

THERE IS NO SPECIFIC LIMITATION ON THE LENGTH OF NUMBERS, OTHER THAN THOSE PRESENTED BY THE AMOUNT OF STORAGE AVAILABLE.

3. INTERNAL STRUCTURE

THE FOLLOWING PAGES CONTAIN DETAILED DESCRIPTIONS AND/OR FLOWCHARTS FOR THE FOLLOWING ROUTINES:

NABS
NSUM
NDIFF
NUMCK
NADD
NSUB
NTRIM
NSWAP

(NOT NECESSARILY IN THAT ORDER)

ARITHMETIC MODULES

NAME: NSWAP

FUNCTION: SWAP PARAMETERS FOR ARGUMENTS A AND B

CALLING SEQUENCE: BAL,LINK NSWAP
(RETURN HERE ALWAYS)

HERE IS THE CODE FOR NSWAP

```
NSWAP   LCFI,2X'40'
        LM,T0     NUMBA      *LOAD THE 3 WORD PARAMETER BLOCK A
        XW,T0     NUMBB      *SWAP WORD 0 WITH BLOCK B
        XW,T1     NUMBB+1    *SWAP WORD 1
        XW,T2     NUMBB+2    *SWAP WORD 2
        XW,T3     NUMBB+3    *AND WORD 3
        LCFI,2 X'40'
        STM,T0    NUMBA      *STORE BLOCK IN A AND RETURN
        B         *LINK
```

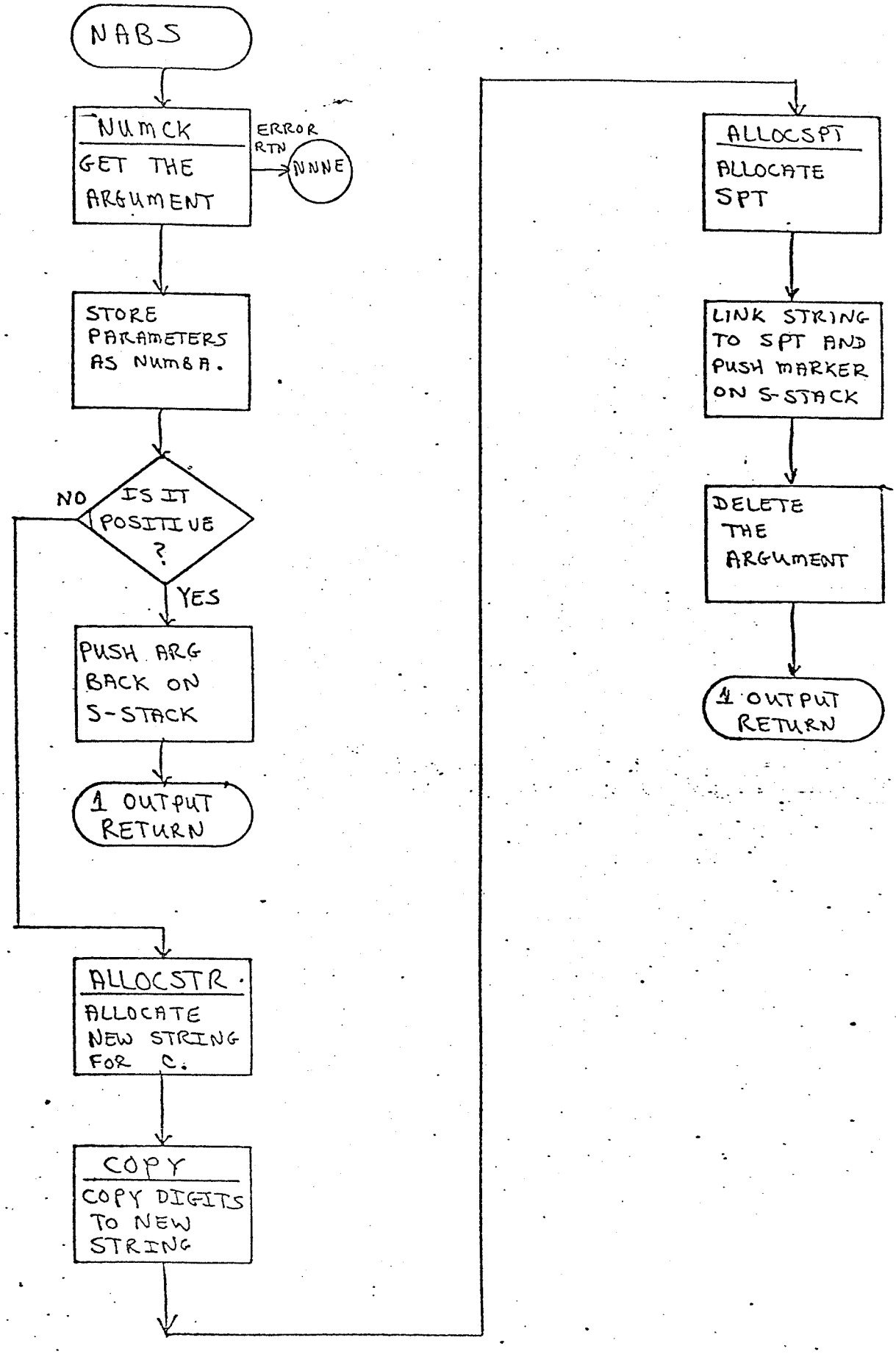
NAME: NABS

FUNCTION: RETURN THE ABSOLUTE VALUE OF A NUMBER

CALLING SEQUENCE: BAL,LINK NABS

NOTE: ERROR RETURN IS TAKEN IF ARGUMENT IS NOT A NUMBER

CALLING ROUTINES: NUMCK

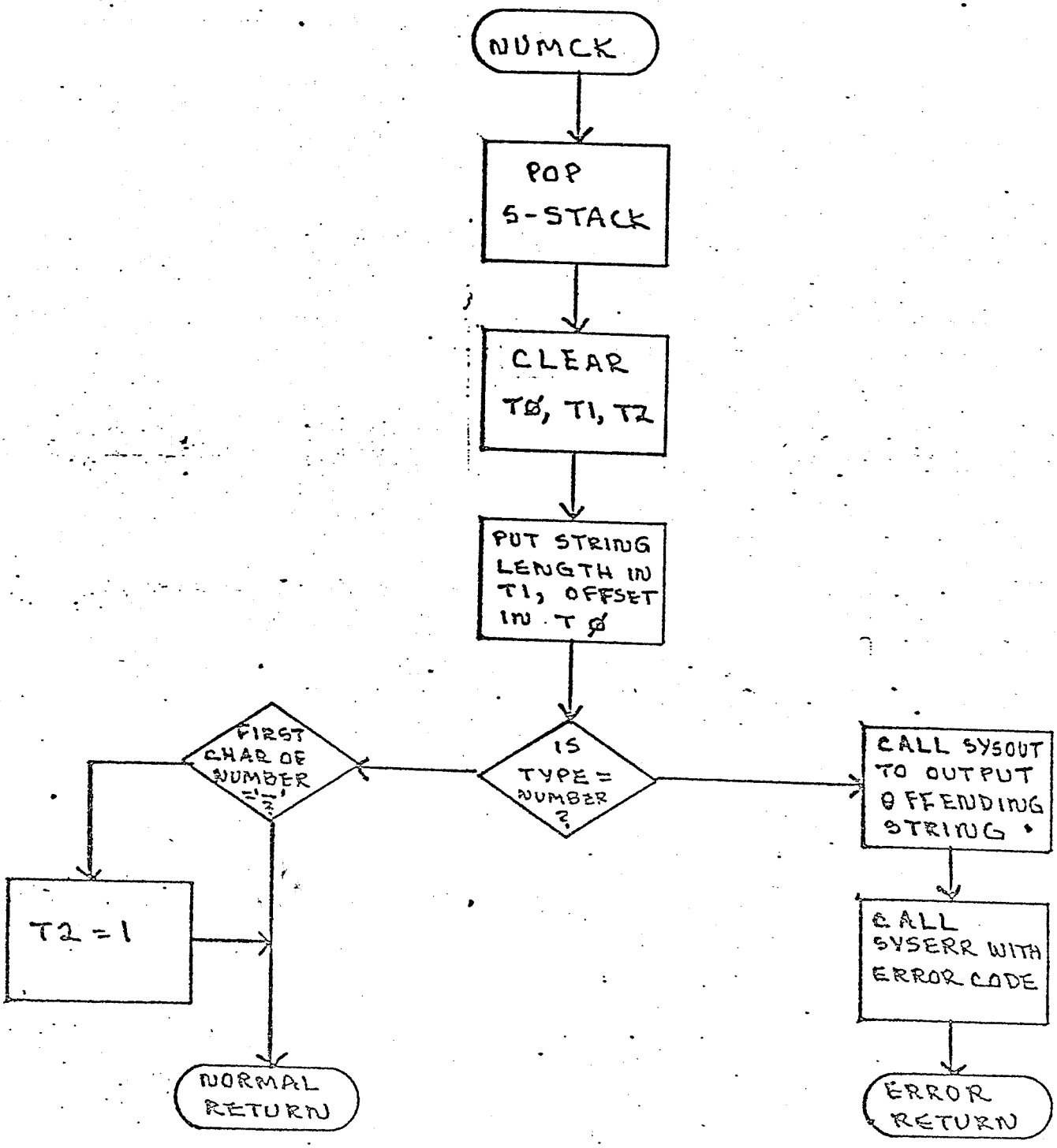

NABS
NUMCK
GET THE ARGUMENT
ERROR RTN
NNNE
STORE PARAMETERS AS NUM&A.
NO
IS IT POSITIVE ?
YES
PUSH ARG BACK ON S-STACK
1 OUTPUT RETURN
ALLOCSTR
ALLOCATE NEW STRING FOR C.
COPY
COPY DIGITS TO NEW STRING
ALLOCSPT
ALLOCATE SPT
LINK STRING TO SPT AND PUSH MARKER ON S-STACK
DELETE THE ARGUMENT
1 OUTPUT RETURN

NAME: NUMCK

FUNCTION: CHECK TO SEE IF ARGUMENT ON TOP OF S-STACK IS A NUMBER; IF SO, RETURNS BYTE DISPLACEMENT TO STRING, STRING LENGTH, AND SIGN INDICATION IN T0, T1, T2. IF STRING IS NOT A NUMBER, OUTPUTS THE STRING FOLLOWED BY THE MESSAGE 'IS NOT A NUMBER' AND TAKES ERROR RETURN.

CALLED ROUTINES: SYSOUT, SYSERR

CALLING SEQUENCE: BAL,LINK NUMCK



NAME: NSUM

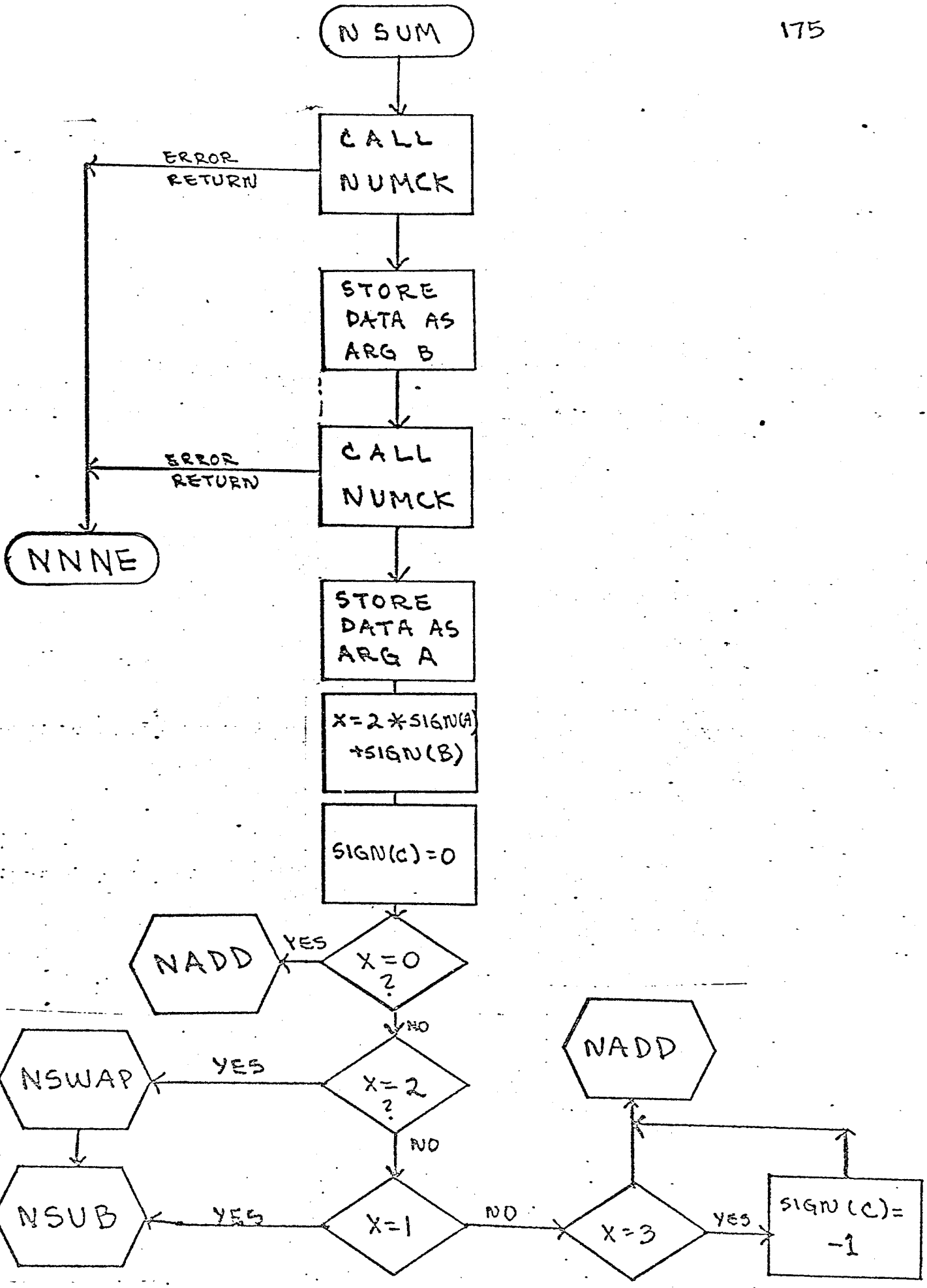
FUNCTION: SET UP ADDITION OF TWO NUMBERS, A AND B, DEPENDING ON THE SIGNS OF THE ARGUMENTS A AND B, THE FOLLOWING RESULTS ARE GENERATED:

SIGN		RESULT OBTAINED	ROUTINE CALLED
A	B		
+	+	A+B	NADD
+	-	A-B	NSUB
-	+	B-A	NSUB (SWAP ARGS)
-	-	-(A+B)	NADD

CALLING SEQUENCE:
```
BAL,LINK  NSUM
B         ERROR RETURN
(NORMAL RETURN)
```

CALLED ROUTINES: NUMCK, NADD, NSUB, NSWAP

FLOWCHART FOLLOWS ON NEXT PAGE


N SUM
CALL NUMCK
ERROR RETURN
STORE DATA AS ARG B
CALL NUMCK
ERROR RETURN
NNNE
STORE DATA AS ARG A
X=2*SIGN(A) +SIGN(B)
SIGN(C)=0
X=0 ?
YES
NADD
NO
X=2 ?
YES
NSWAP
NO
X=1
YES
NSUB
NO
X=3
YES
SIGN(C)= -1
NADD

NAME: NDIFF

FUNCTION: SET UP DIFFERENCE OF TWO NUMBERS, A AND B. DEPENDING ON THE SIGNS OF THE ARGUMENTS, THE FOLLOWING RESULTS ARE GENERATED:

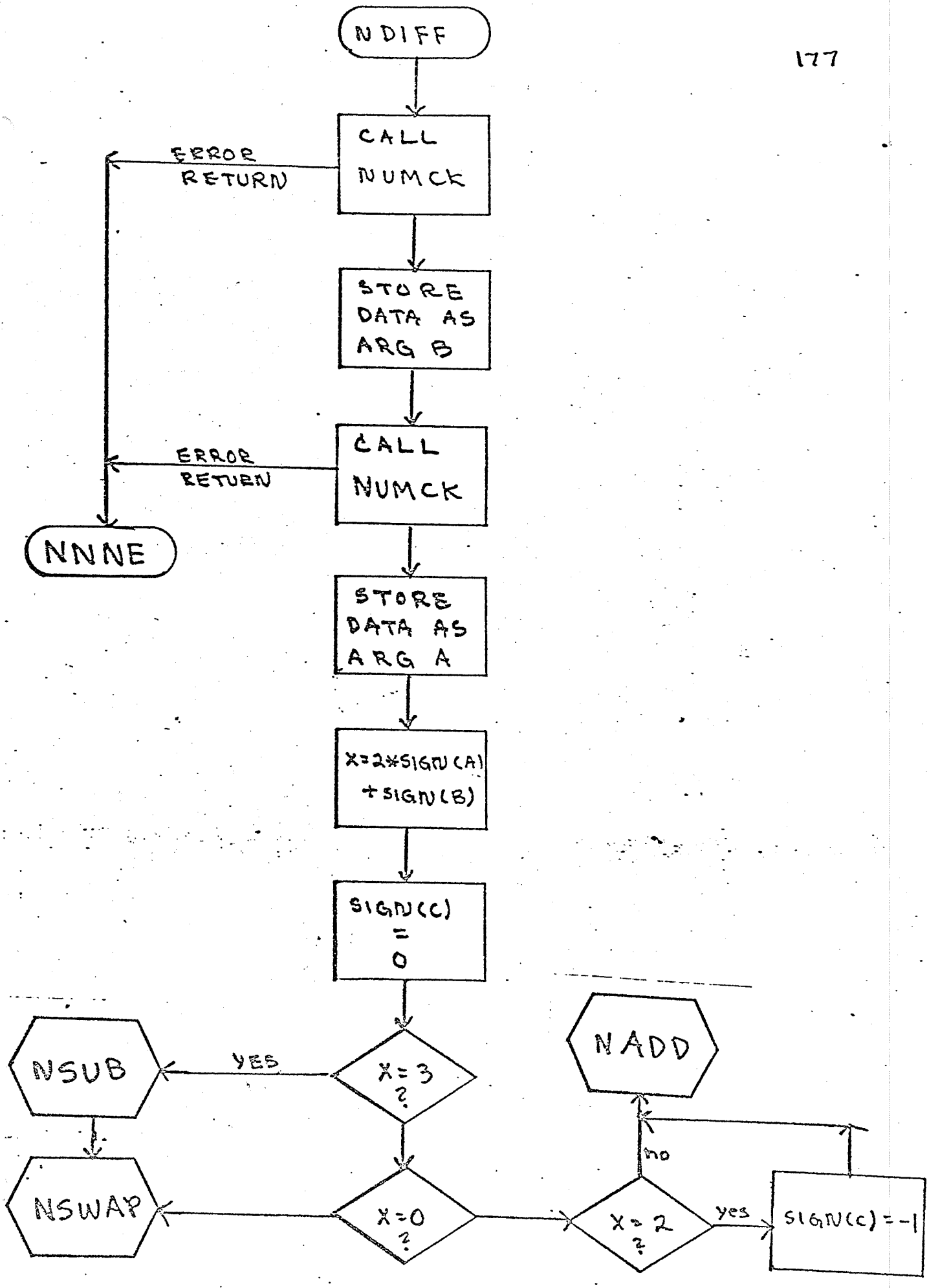
SIGN A	SIGN B	RESULTS OBTAINED	ROUTINE CALLED
+	+	A - B	NSUB
+	-	A + B	NADD
-	+	-(A+B)	NADD
-	-	B - A	NSUB (ARGS REVERSED)

CALLING SEQUENCE: BAL,LINK NSU

CALLING SEQUENCE: BAL,LINK NSUM

CALLED ROUTINES: NUMCK, NADD, NSUB, NSWAP

FLOWCHART FOLLOWS ON NEXT PAGE


NDIFF
CALL NUMCK
ERROR RETURN
STORE DATA AS ARG B
CALL NUMCK
ERROR RETUEN
NNNE
STORE DATA AS ARG A
X=2*SIGN(A) +SIGN(B)
SIGN(C) = 0
NSUB
YES
X=3 ?
NADD
NSWAP
X=0 ?
no
X=2 ?
yes
SIGN(C)=-1

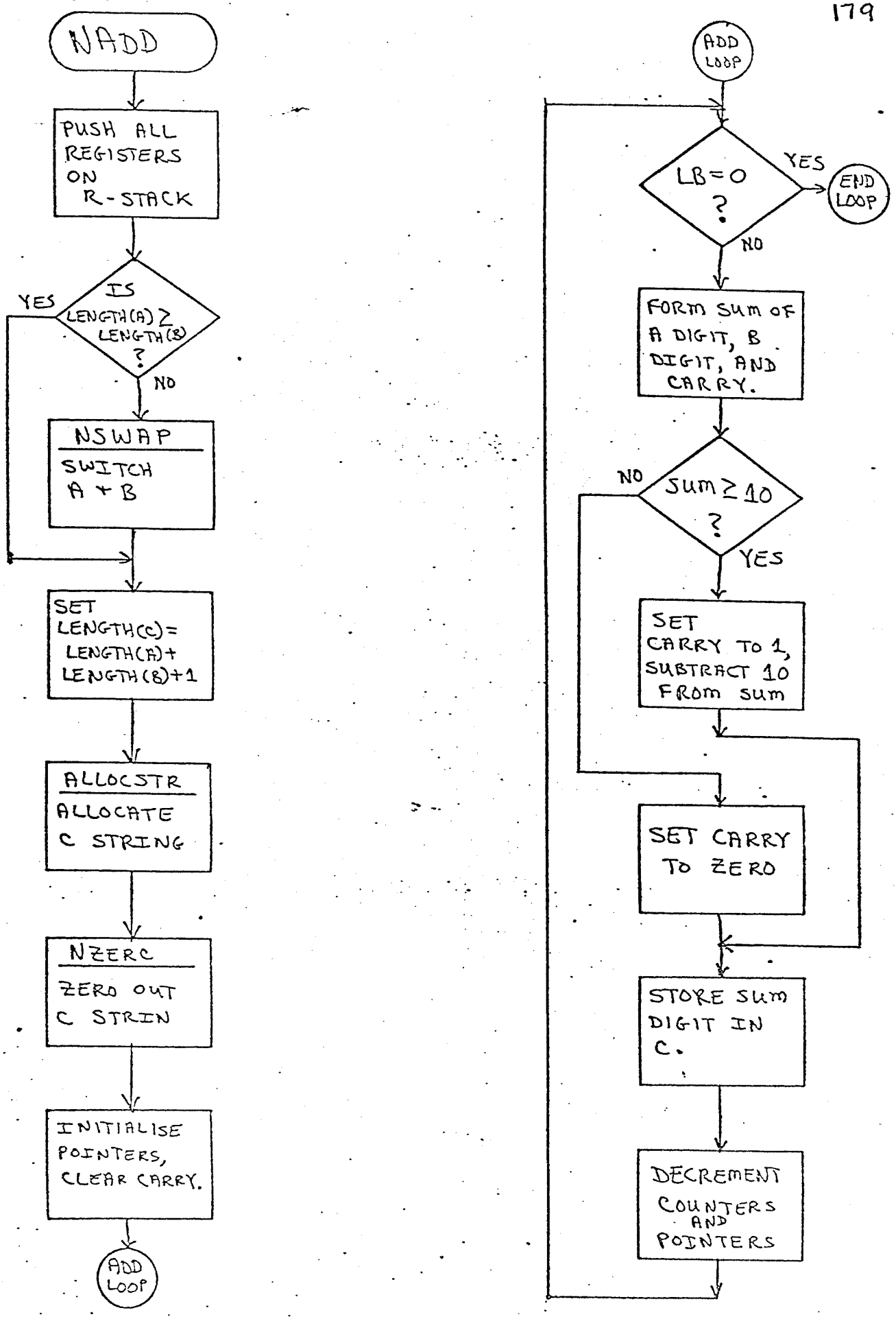
NAME: NADD

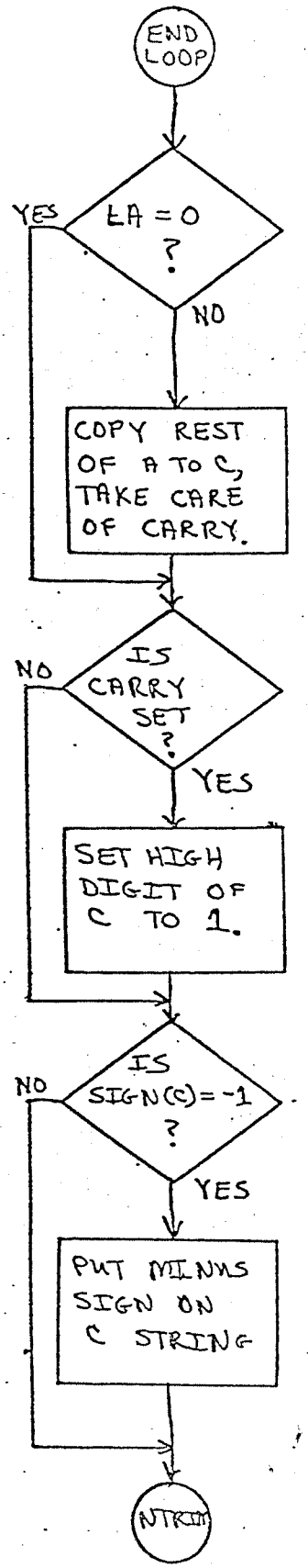
FUNCTION: ADD TWO NUMBERS, A AND B

ENTERED BY: NSUM, NDIFF

CALLED ROUTINES: ALLOCSTR, RTNSTR, ALLOCSPT

FLOWCHART ON FOLLOWING PAGES


NADD
PUSH ALL REGISTERS ON R-STACK
IS LENGTH(A) ≥ LENGTH(B) ?
YES
NO
NSWAP
SWITCH A + B
SET LENGTH(C)= LENGTH(A)+ LENGTH(B)+1
ALLOCSTR
ALLOCATE C STRING
NZERC
ZERO OUT C STRIN
INITIALISE POINTERS, CLEAR CARRY.
ADD LOOP
ADD LOOP
LB=0 ?
YES
NO
END LOOP
FORM SUM OF A DIGIT, B DIGIT, AND CARRY.
SUM ≥ 10 ?
NO
YES
SET CARRY TO 1, SUBTRACT 10 FROM SUM
SET CARRY TO ZERO
STORE SUM DIGIT IN C.
DECREMENT COUNTERS AND POINTERS

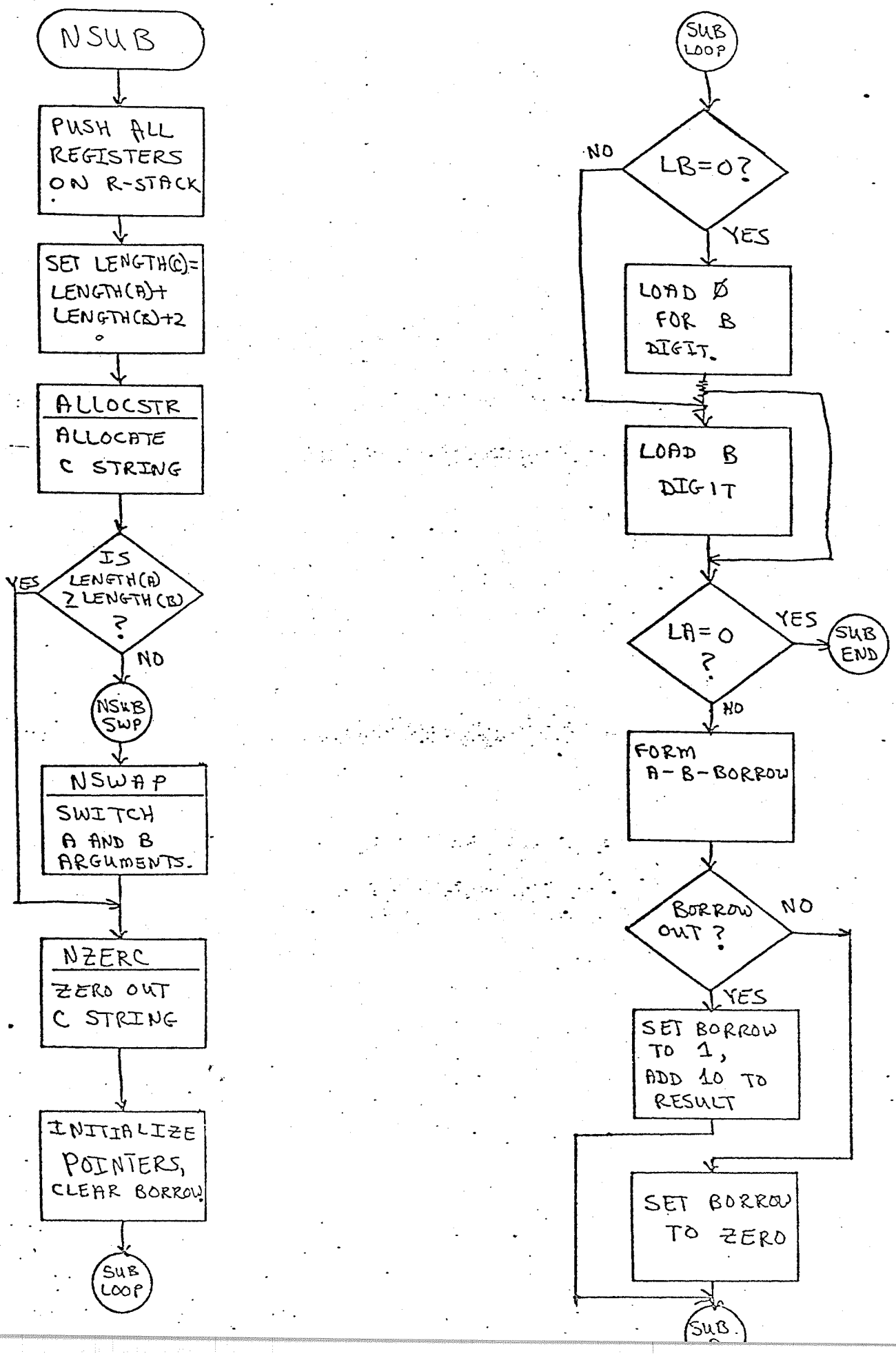

END LOOP
LA = 0 ?
YES
NO
COPY REST OF A TO C, TAKE CARE OF CARRY.
IS CARRY SET ?
NO
YES
SET HIGH DIGIT OF C TO 1.
IS SIGN(C) = -1 ?
NO
YES
PUT MINUS SIGN ON C STRING
NTRIM

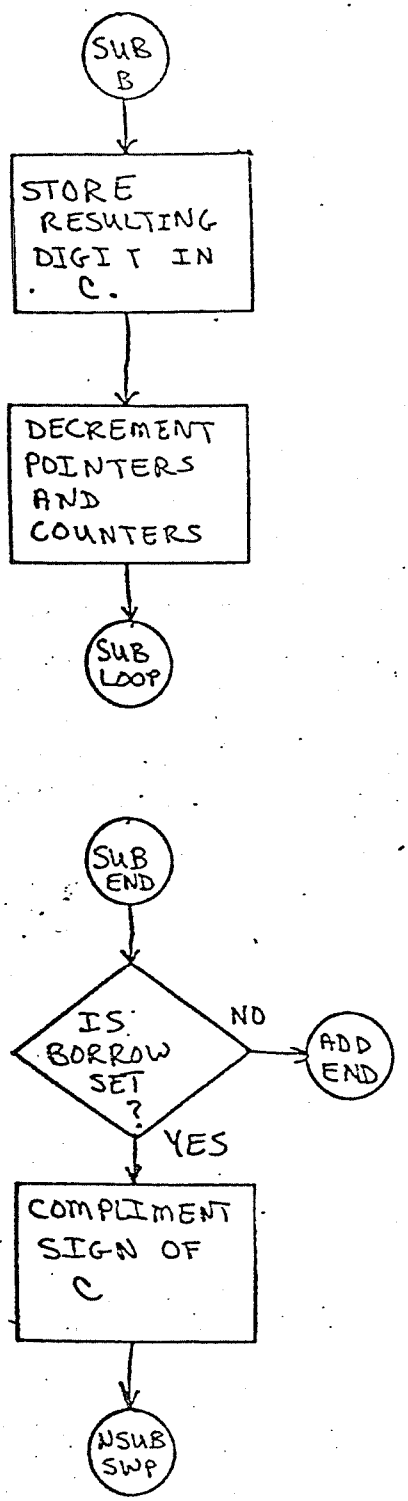
NAME: NSUB

FUNCTION: SUBTRACT TWO NUMBERS, A-B

ENTERED BY: NSUM, NDIFF

FLOWCHART ON FOLLOWING PAGES

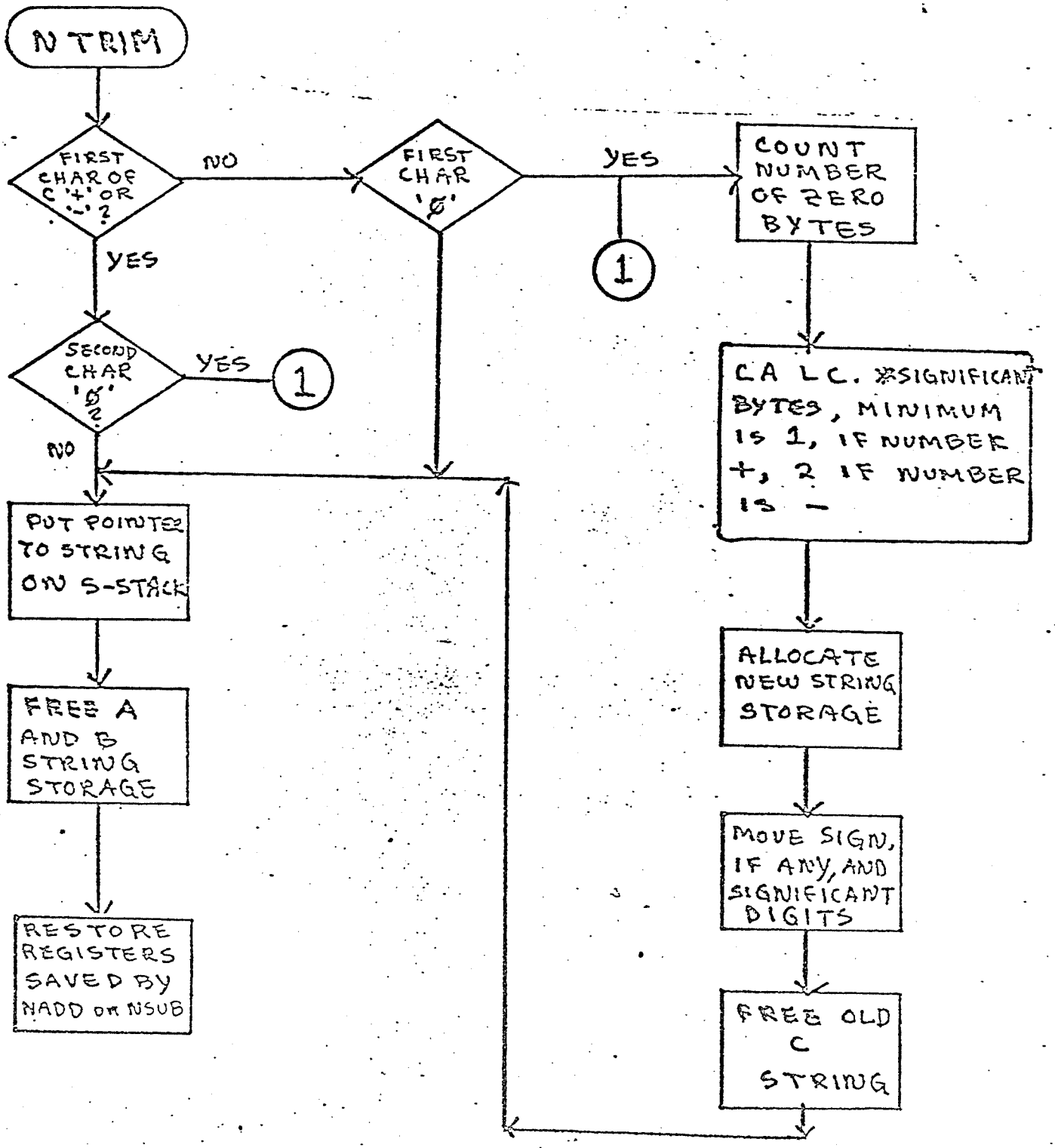

NSUB
PUSH ALL REGISTERS ON R-STACK
SET LENGTH(C)= LENGTH(A)+ LENGTH(B)+2
ALLOCSTR
ALLOCATE C STRING
IS LENGTH(A) ≥ LENGTH(B) ?
YES
NO
NSUB SWP
NSWAP
SWITCH A AND B ARGUMENTS.
NZERC
ZERO OUT C STRING
INITIALIZE POINTERS, CLEAR BORROW
SUB LOOP
SUB LOOP
LB=0?
NO
YES
LOAD Ø FOR B DIGIT.
LOAD B DIGIT
LA=0 ?
YES
SUB END
NO
FORM A-B-BORROW
BORROW OUT ?
NO
YES
SET BORROW TO 1, ADD 10 TO RESULT
SET BORROW TO ZERO
SUB


SUB B
STORE RESULTING DIGIT IN C.
DECREMENT POINTERS AND COUNTERS
SUB LOOP
SUB END
IS BORROW SET ?
NO
ADD END
YES
COMPLIMENT SIGN OF C
NSUB SWP

NAME: NTRIM

FUNCTION: REMOVE NON-SIGNIFICANT ZEROES FROM STRING, PLACE RESULT POINTER ON S-STACK

ENTERED BY: NSUB, NADD



NAME: NNNE

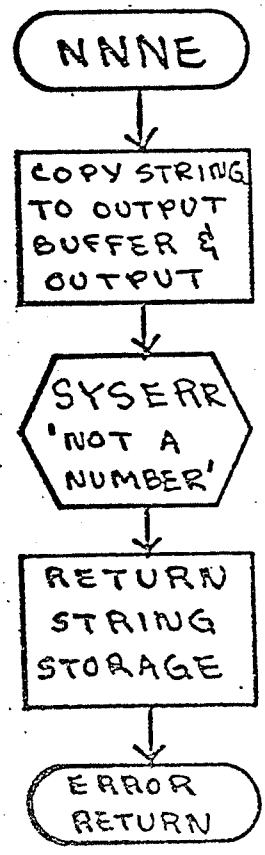
FUNCTION: ISSUE NOT A NUMBER ERROR

CALLED BY: ARITHMETIC ROUTINES

CALLING SEQUENCE: BAL,LINK NNNE
(NEVER RETURN)

EXPECTS AS ARGUMENTS:

REGISTER	CONTENTS
T0	BYTE ADDRESS OF STRING
T1	LENGTH OF STRING
T3	SPT WORD FROM S-STACK



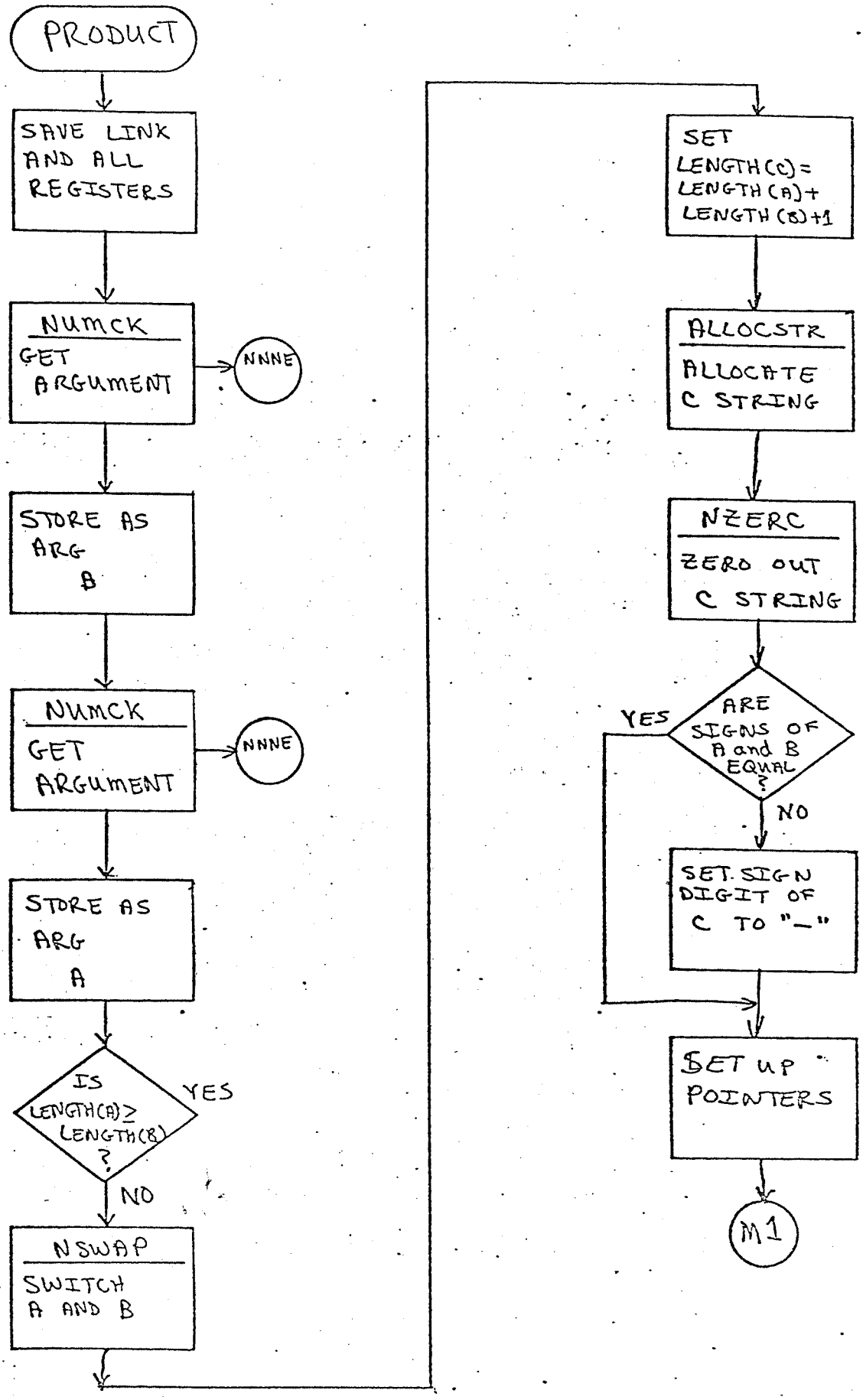
NAME: PRODUCT

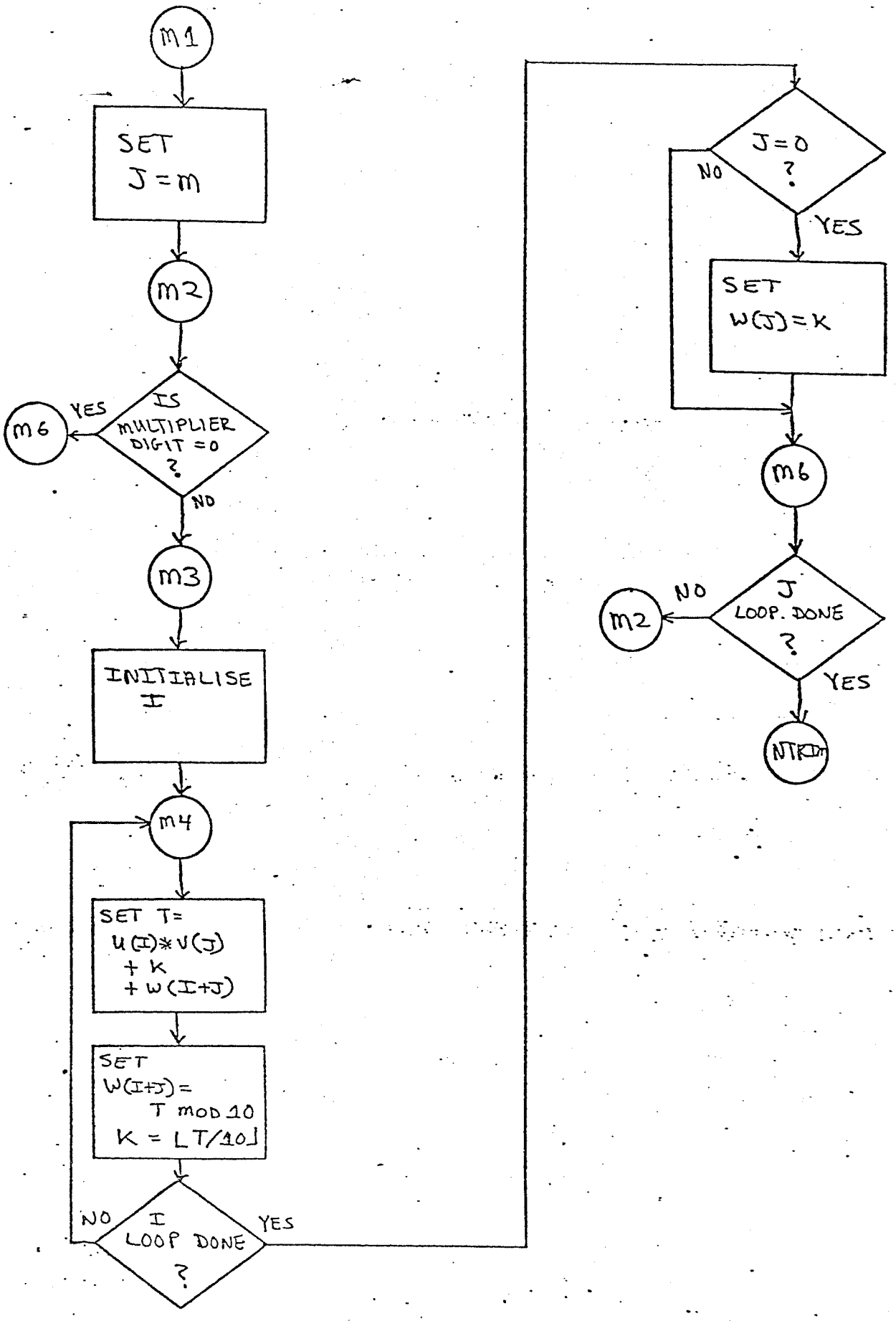
FUNCTION: RETURN THE PRODUCT OF TWO NUMBERS

CALLING SEQ: BAL, LINK PRODUCT

CALLED ROUTINES: NUMCK, NSWAP

FLOWCHART FOLLOWS


PRODUCT
SAVE LINK AND ALL REGISTERS
NUMCK
GET ARGUMENT
NNNE
STORE AS ARG B
NUMCK
GET ARGUMENT
NNNE
STORE AS ARG A
IS LENGTH(A) ≥ LENGTH(B) ?
YES
NO
NSWAP
SWITCH A AND B
SET LENGTH(C) = LENGTH(A) + LENGTH(B) + 1
ALLOCSTR
ALLOCATE C STRING
NZERC
ZERO OUT C STRING
ARE SIGNS OF A and B EQUAL ?
YES
NO
SET SIGN DIGIT OF C TO "_"
SET UP POINTERS
M1


M1
SET J = m
M2
IS MULTIPLIER DIGIT = 0 ?
YES
M6
NO
M3
INITIALISE I
M4
SET T= U(I)*V(J) + K + W(I+J)
SET W(I+J) = T MOD 10
K = LT/10J
I LOOP DONE ?
NO
YES
J = 0 ?
NO
YES
SET W(J) = K
M6
J LOOP. DONE ?
NO
M2
YES
NTRD

RETURNS THE QUOTIENT OF TWO NUMBERS. NUMBERS MUST BE 9 DIGITS OR LESS. THE RESULT IS THE FIRST INPUT DIVIDED BY THE SECOND. USES A HARDWARE DIVIDE.

CALLED BY EXECUTE

CALLED ROUTINES: SYSERR, RTNSPT, ALLOCSTR, MOVE, NUMCK

RETURNS THE REMAINDER OF THE FIRST INPUT DIVIDED BY THE SECOND. NUMBERS MUST BE NINE DIGITS OR LESS. USES A HARDWARE DIVIDE.

CALLED BY EXECUTE

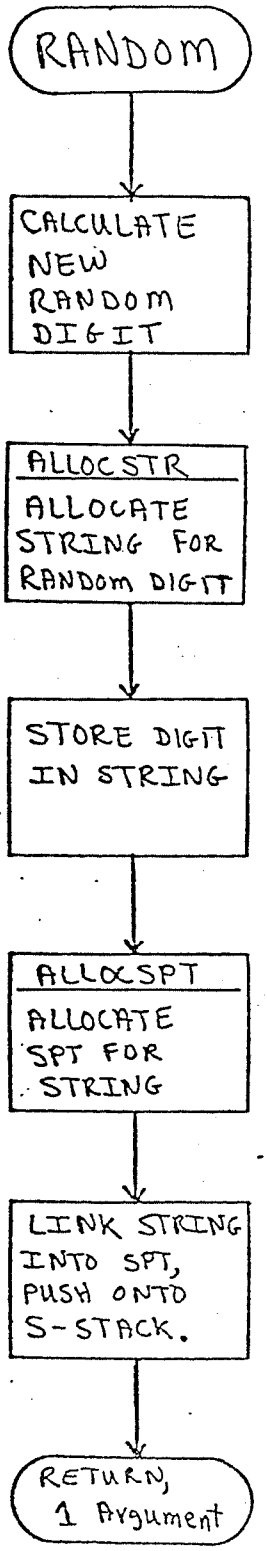
CALLED ROUTINES: ALLOCSTR, NNME, NUMCK

RANDOM

RETURNS A RANDOM DIGIT.

CALLED BY EXECUTE

CALLED ROUTINES: ALLOCSTR, ALLOCSPT


RANDOM
CALCULATE NEW RANDOM DIGIT
ALLOCSTR
ALLOCATE STRING FOR RANDOM DIGIT
STORE DIGIT IN STRING
ALLOCSPT
ALLOCATE SPT FOR STRING
LINK STRING INTO SPT, PUSH ONTO S-STACK.
RETURN, 1 Argument

MAKE

EFFECTS

1. POP 2 ARGUMENTS FROM S-STACK; FIRST IS SOURCE AND SECOND IS DEST.
2. REMOVE THE MOST RECENT OCCURRENCE OF DEST FROM THE SYMBOL TABLE. (IGNORE FAILURE CONDITIONS)
3. ADD A NEW SYMBOL TO THE SYMBOL TABLE WITH NAME INDICATED BY DEST AND VALUE INDICATED BY SOURCE.

NO OUTPUTS.

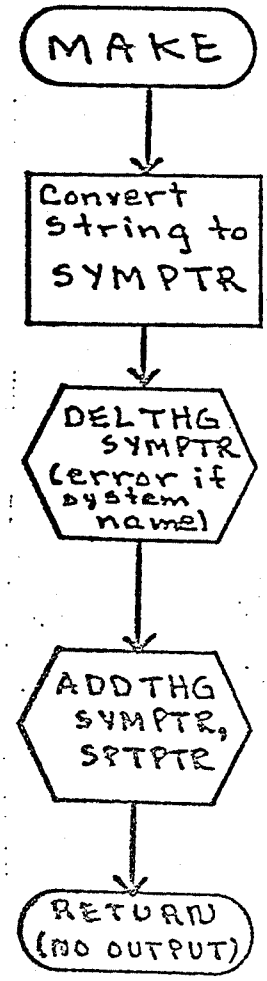
INTERNAL SPECIFICATIONS

NAME: MAKE

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK MAKE

CALLED ROUTINES: DELTHG, ADDTHG,



EFFECTS

1. POP ONE ARGUMENT FROM S-STACK
2. TAKE THE FIRST CHARACTER (WORD) FROM THE WORD (SENTENCE) INDICATED BY THE ARGUMENT AND PUSH THAT CHARACTER (WORD) ONTO THE S-STACK

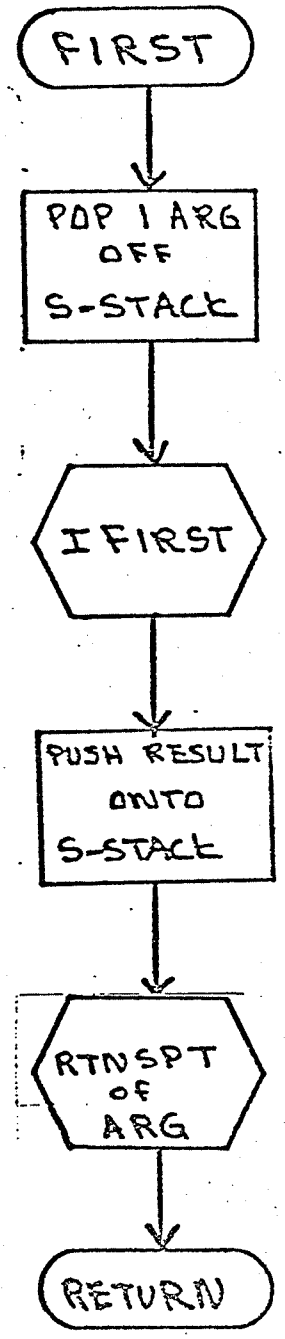
INTERNAL SPECIFICATIONS

NAME: FIRST

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK FIRST

CALLED ROUTINES: IFIRST, RTNSPT


FIRST
POP 1 ARG OFF S-STACK
IFIRST
PUSH RESULT ONTO S-STACK
RTNSPT of ARG
RETURN

EFFECTS

1. POP ONE ARGUMENT FROM THE S-STACK
2. TAKE THE LAST CHARACTER (WORD) FROM THE WORD (SENTENCE) INDICATED AND PUSH THAT CHARACTER (WORD) ONTO THE S-STACK

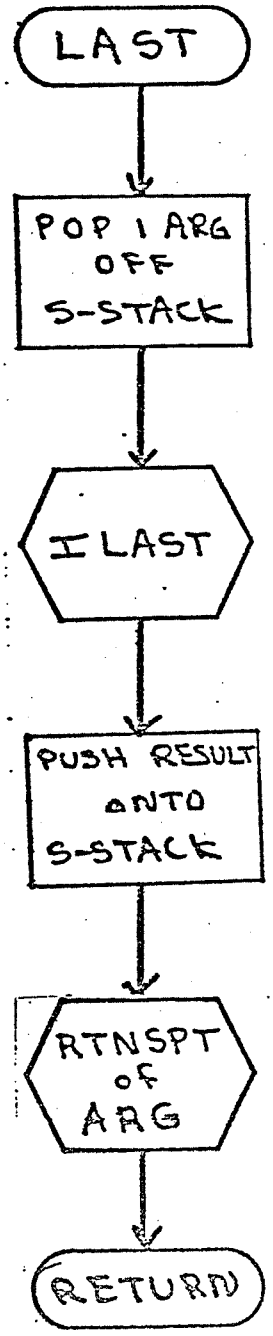
INTERNAL SPECIFICATIONS

NAME: LAST

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK LAST

CALLED ROUTINES: ILAST, RTNSPT


LAST
POP 1 ARG OFF S-STACK
I LAST
PUSH RESULT ONTO S-STACK
RTNSPT OF ARG
RETURN

EFFECTS
1. POP ONE ARGUMENT FROM THE S-STACK
2. TAKE ALL BUT THE FIRST CHARACTER (WORD) FROM THE WORD (SENTENCE) INDICATED AND PUSH THE RESULT ONTO THE S-STACK.

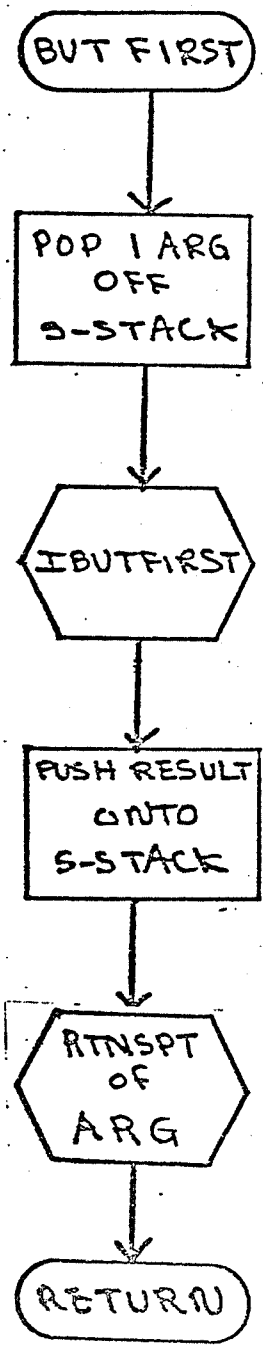
INTERNAL SPECIFICATIONS

NAME: BUTFIRST

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK BUTFIRST

CALLED ROUTINES: IBUTFIRST, RTNSPT


BUT FIRST
POP 1 ARG OFF S-STACK
IBUTFIRST
PUSH RESULT ONTO S-STACK
RTNSPT OF ARG
RETURN

BUTLAST

EFFECTS
1. POP ONE ARGUMENT FROM THE S-STACK
2. TAKE ALL BUT THE LAST CHARACTER (WORD) FROM THE ARGUMENT AND PUSH THE RESULT ONTO S-STACK

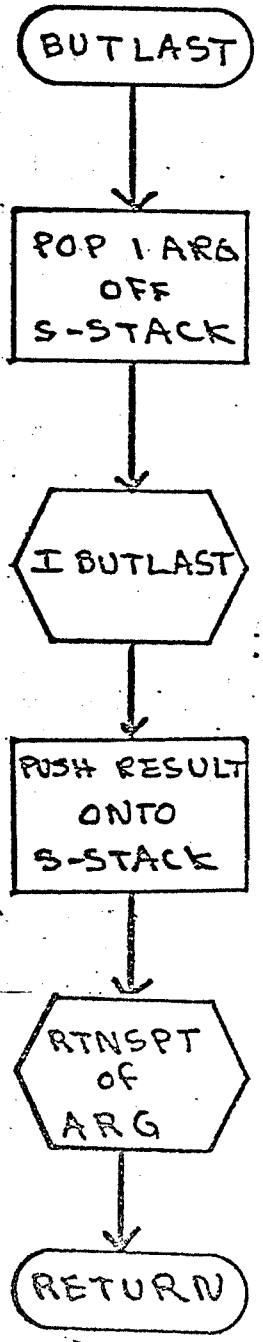
INTERNAL SPECIFICATIONS

NAME: BUTLAST

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK BUTLAST

CALLED ROUTINES: IBUTLAST, RTNSPT



WORD

PULLS TWO WORDS OFF THE S-STACK AND APPENDS THE FIRST WORD PULLED TO THE END OF THE 2ND WORD PULLED AND PUSHES THE RESULT BACK ON THE S-STACK. ERROR RETURN IF INPUTS ARE NOT WORDS OR RESULTING STRING IS TOO LONG (32,000 CHARACTERS).

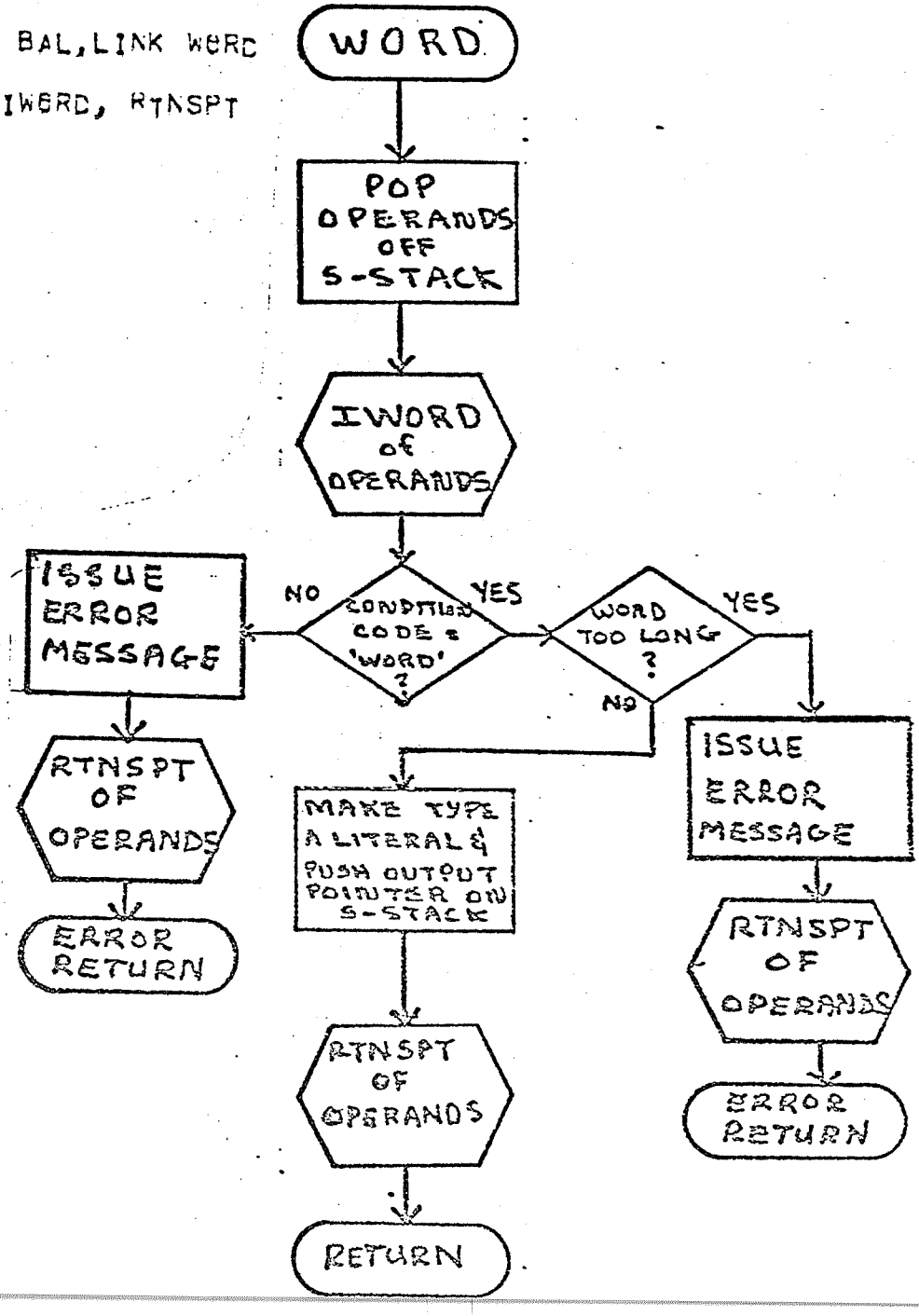
INTERNAL SPECIFICATIONS

NAME: WORD

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK WORD

CALLED ROUTINES: IWORD, RTNSPT



PULLS TWO SENTENCES OFF THE S-STACK AND APPENDS THE 1ST SENTENCE PULLED TO THE END OF THE 2ND SENTENCE PULLED AND PUSHES THE RESULT BACK ON THE S-STACK. ERROR RETURN IF THE RESULTING STRING IS TOO LONG (32,000 WORDS).

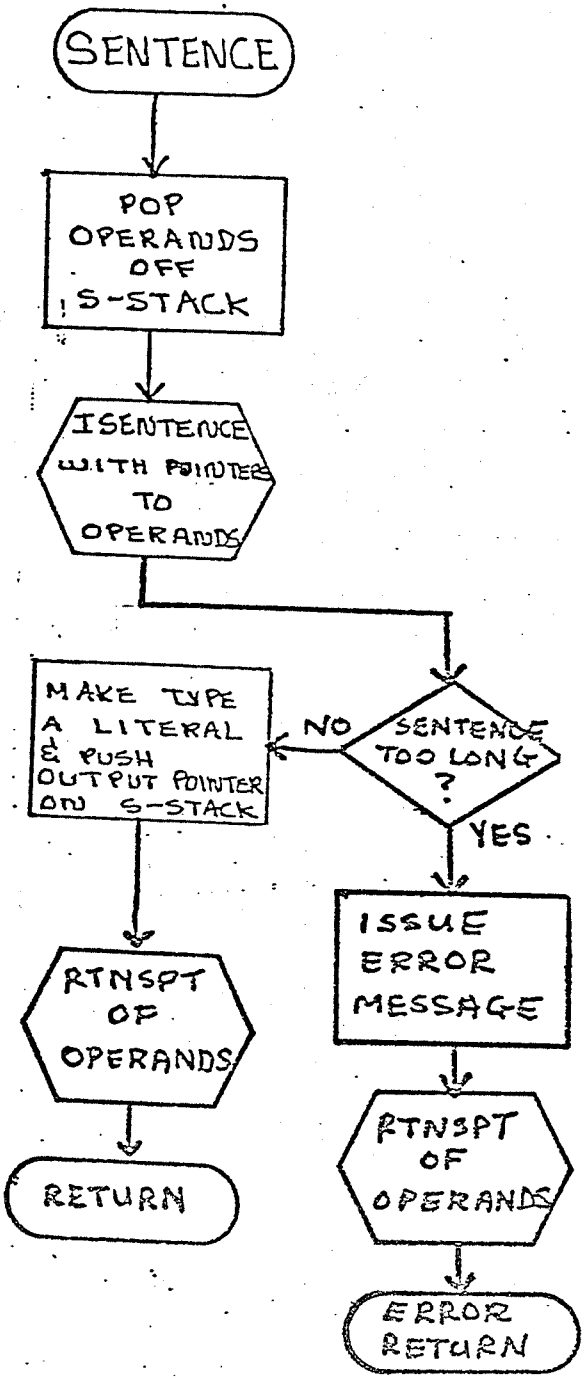
INTERNAL SPECIFICATIONS

NAME: SENTENCE

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK SENTENCE

CALLED ROUTINES: ISENTENCE,RTNSPT


SENTENCE
POP OPERANDS OFF S-STACK
ISENTENCE WITH POINTERS TO OPERANDS
MAKE TYPE A LITERAL & PUSH OUTPUT POINTER ON S-STACK
NO
SENTENCE TOO LONG ?
YES
RTNSPT OF OPERANDS
RETURN
ISSUE ERROR MESSAGE
RTNSPT OF OPERANDS
ERROR RETURN

PULLS A WORD OR SENTENCE OFF THE S-STACK. CHECKS THE WORD COUNT IN THE SPT AND IF > 1 RETURNS IT AS THE NUMBER OF WORDS IN THE SENTENCE, OTHERWISE, GOES TO THE STRING STORAGE AND RETURNS THE CHARACTER COUNT AS THE NUMBER OF CHARACTERS IN THE GIVEN WORD, PUTS THE NUMBER ON THE S-STACK.

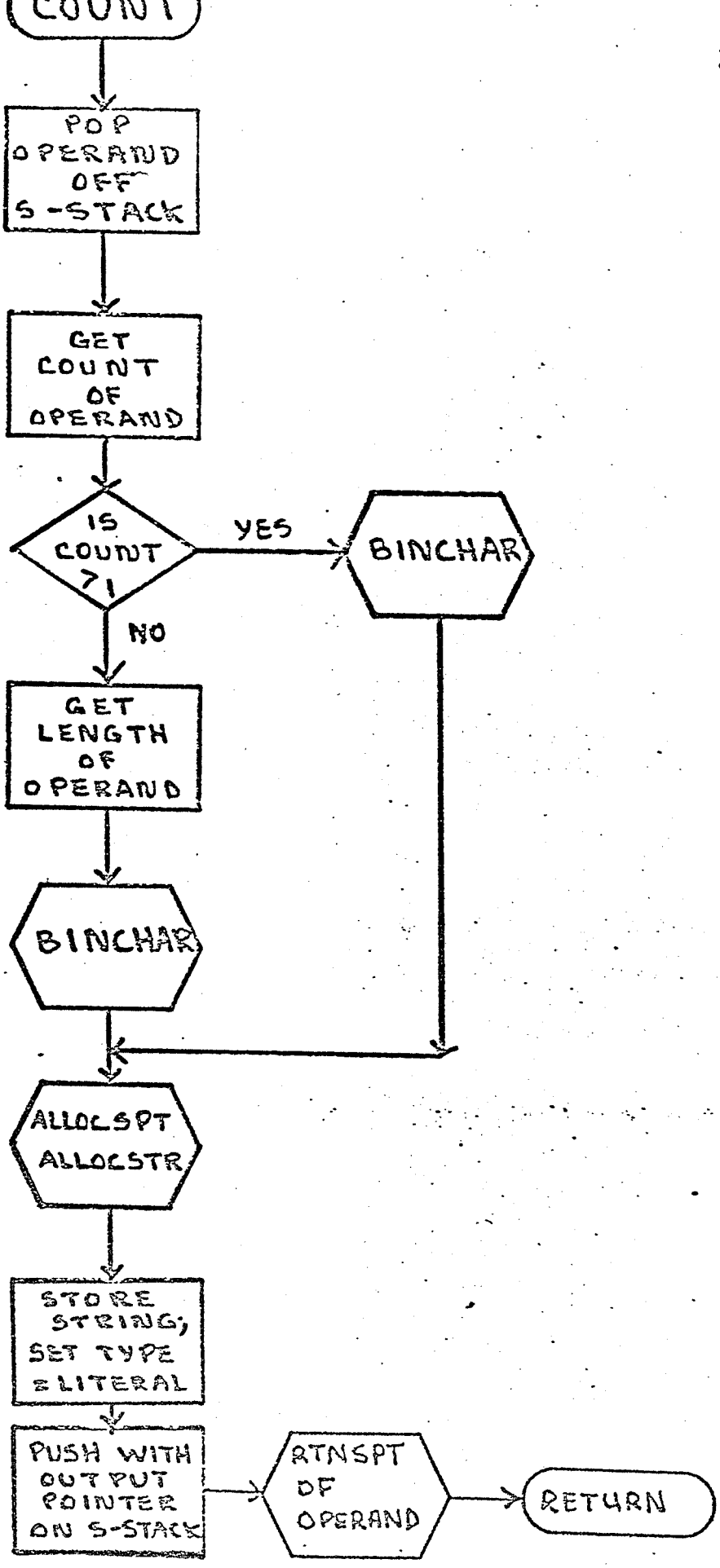
INTERNAL SPECIFICATIONS

NAME: COUNT

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK COUNT

CALLED ROUTINES: BINCHAR, ALLOCSPT, ALLOCSTR
RTNSPT, COPY


COUNT
POP OPERAND OFF S-STACK
GET COUNT OF OPERAND
IS COUNT > 1
YES
BINCHAR
NO
GET LENGTH OF OPERAND
BINCHAR
ALLOCSPT ALLOCSTR
STORE STRING; SET TYPE = LITERAL
PUSH WITH OUTPUT POINTER ON S-STACK
RTNSPT OF OPERAND
RETURN

THING

PULLS NAME OFF S-STACK, SEARCHES SYMBOL TABLE FOR GIVEN NAME AND RETURNS CORRESPONDING THING TO THE S-STACK. ERROR RETURN IF THE NAME IS EMPTY.

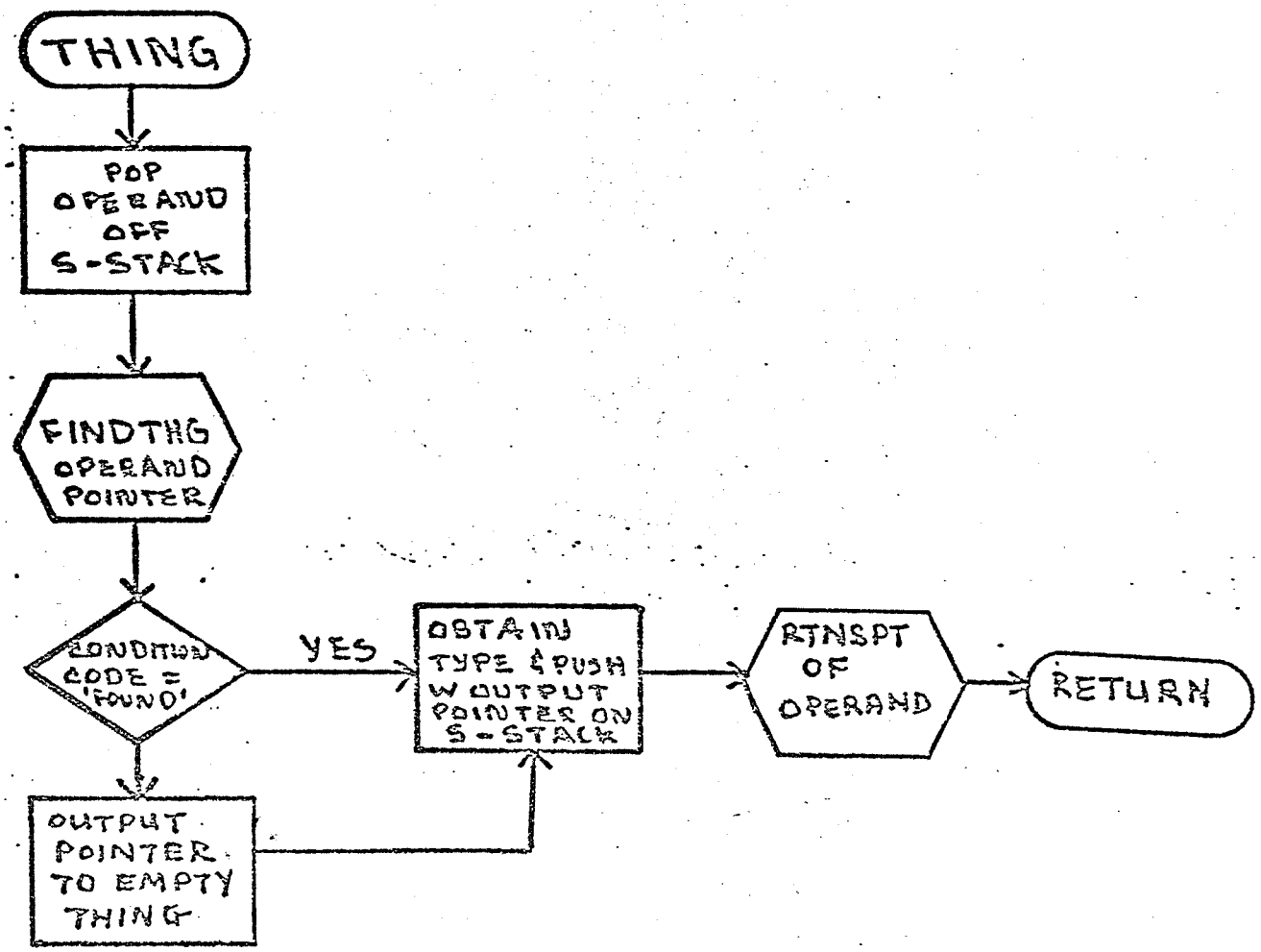
INTERNAL SPECIFICATIONS

NAME: THING

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK THING

CALLED ROUTINES: FINDTHG, RTNSPT



CHECKS THE TRUTH FLAG, 'TRUTH', IF THE VALUE IS 'TRUE' IT RETURNS TO THE EXECUTER FOR NORMAL EXECUTION OF THE REST OF THE LINE. IF THE VALUE IS 'FALSE' IT RETURNS TO THE EXECUTER FOR TERMINATION OF EXECUTION OF THAT LINE

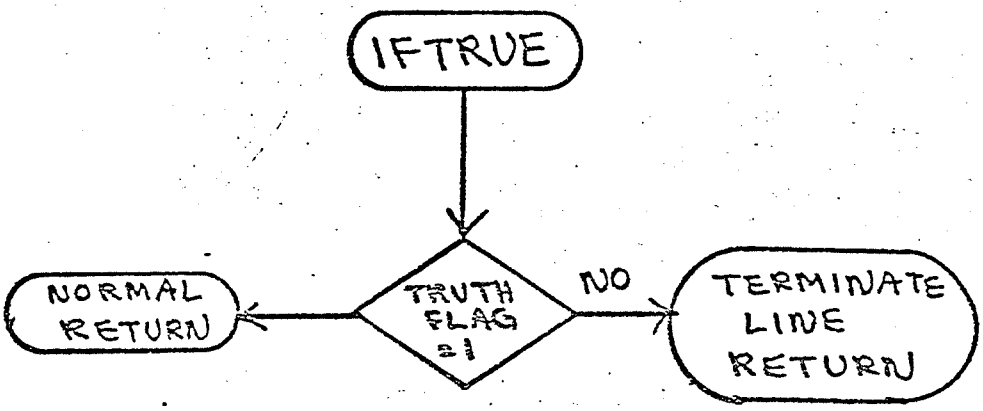
INTERNAL SPECIFICATIONS

NAME: IFTRUE

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK IFTRUE

CALLED ROUTINES: NONE



TESTS THE TRUTH FLAG, 'TRUTH', IF THE VALUE IS 'FALSE' IT RETURNS TO THE EXECUTER FOR NORMAL EXECUTION OF THE REST OF THE LINE. IF THE VALUE IS 'TRUE' IT RETURNS TO THE EXECUTER FOR TERMINATION OF EXECUTION OF THAT LINE.

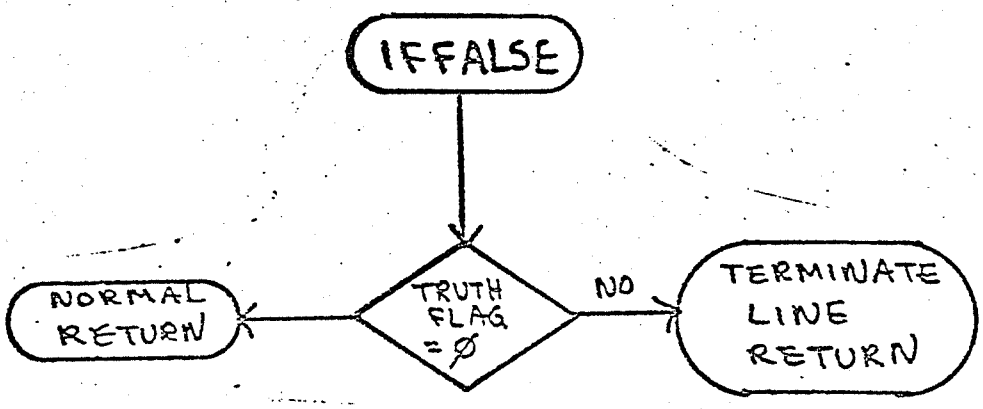
INTERNAL SPECIFICATIONS

NAME: IFFALSE

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK IFFALSE

CALLED ROUTINES: NONE



TRUE

PLACES THE STRING 'TRUE' IN STRING STORAGE AND PUSHES IT ON THE S-STACK AS THE OUTPUT OF THE CALLING ROUTINES. RETURNS FROM CALLING ROUTINE.

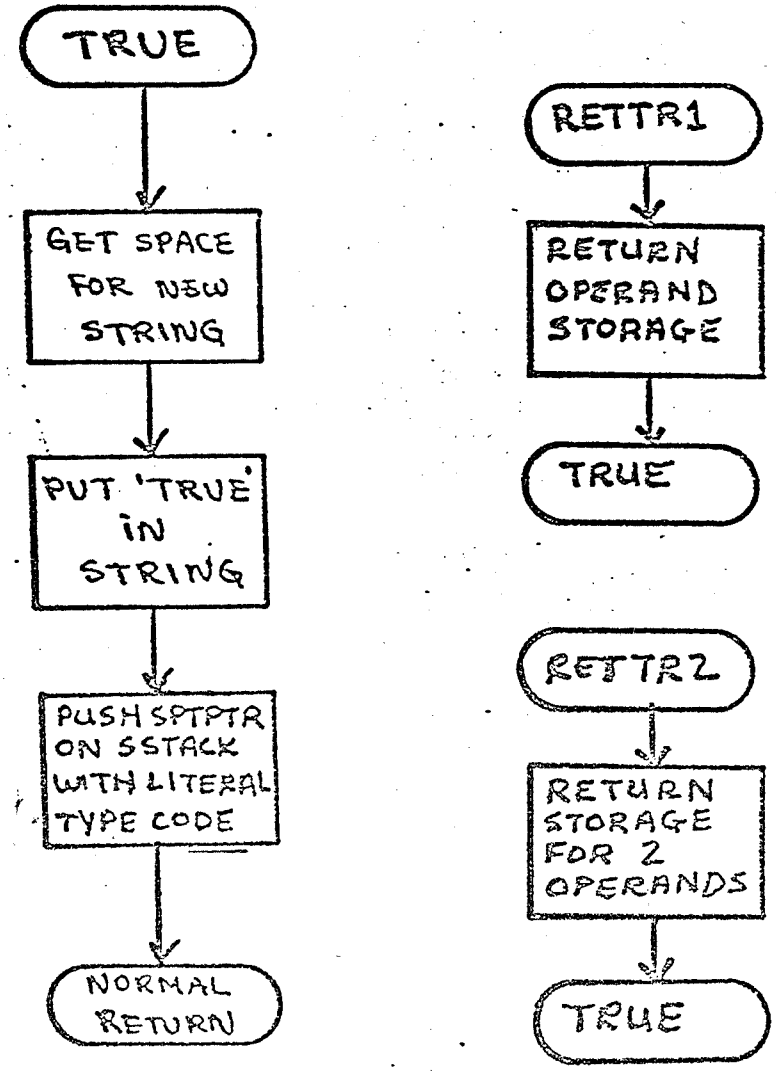
INTERNAL SPECIFICATIONS

NAME: TRUE

CALLED BY: EMPTYP, ZEROP, WORDP, SENTENCEP, NUMBERP, GREATERP, IS, BOTH, EITHER

ENTERING SEQUENCE: TRUE

CALLED ROUTINES: ALLOCSPT, ALLOCSTR, COPY



FALSE

PLACES THE STRING 'FALSE' IN STRING STORAGE AND PUSHES IT ON THE S-STACK AS THE OUTPUT OF THE CALLING ROUTINE. RETURNS FROM CALLING ROUTINE.

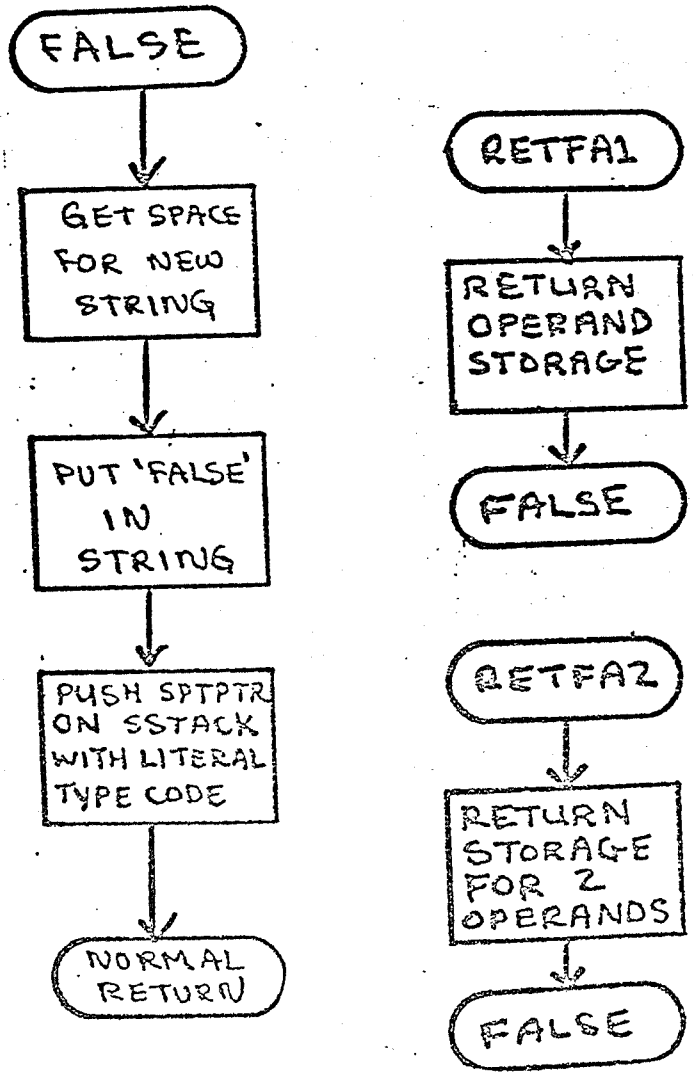
INTERNAL SPECIFICATIONS

NAME: FALSE

CALLED BY: EMPTYP, ZEROP, WORDP, SENTENCEP, NUMBERP, GREATERP, IS, BOTH, EITHER

ENTERING SEQUENCE: FALSE

CALLED ROUTINES: ALLOCSPT, ALLOCSTR, COPY



PULLS OPERAND OFF S-STACK, CHECKS IF TRUE OR FALSE AND SETS TRUTH FLAG, TRUTH, TO 1 OR 0, RESPECTIVELY. ERROR RETURN IF INPUTS ARE NOT 'TRUE' OR 'FALSE'

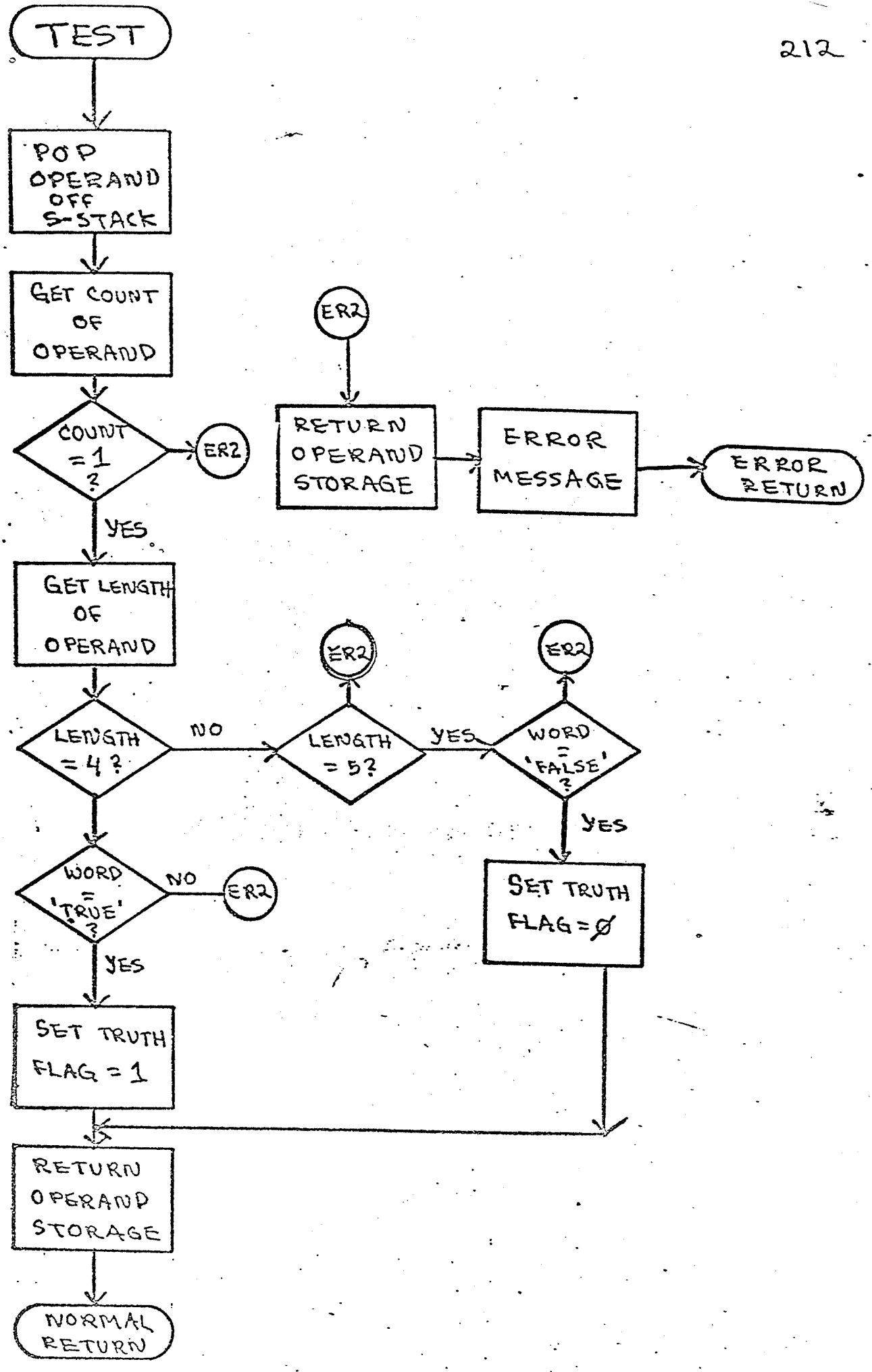
INTERNAL SPECIFICATIONS

NAME: TEST

CALLED BY: EXECUTE

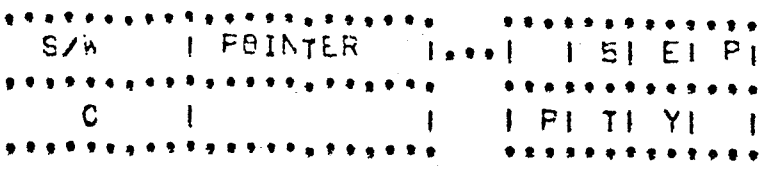
CALLING SEQUENCE: BAL,LINK TEST

CALLED ROUTINES: RTNSPT


TEST
POP OPERAND OFF S-STACK
GET COUNT OF OPERAND
COUNT = 1 ?
ER2
YES
GET LENGTH OF OPERAND
LENGTH = 4 ?
NO
LENGTH = 5?
YES
WORD = 'FALSE' ?
YES
WORD = 'TRUE' ?
NO
ER2
YES
SET TRUTH FLAG = 1
SET TRUTH FLAG = Ø
RETURN OPERAND STORAGE
NORMAL RETURN
ER2
RETURN OPERAND STORAGE
ERROR MESSAGE
ERROR RETURN
ER2
ER2

TAKES THE TOP THING OF THE S-STACK, RETURNS TRUE OR FALSE TO THE S-STACK AS THE ARGUMENT IS /EMPTY/ OR NOT AND THEN RETURNS TO THE EXECUTER.

FORMAT OF THE EMPTY THING



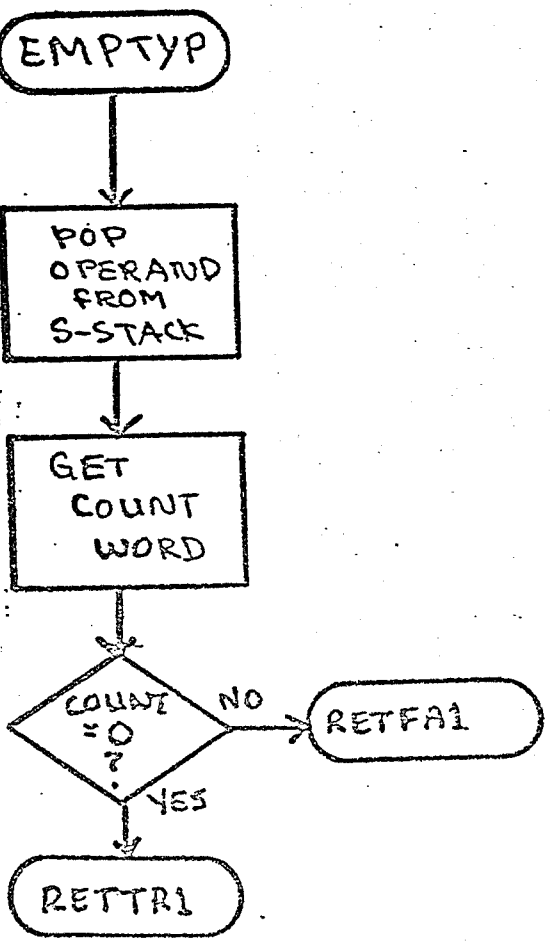
INTERNAL SPECIFICATIONS

NAME: EMPTYP

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK EMPTYP

CALLED ROUTINES: TRUE, FALSE, RTNSPT



ZERØP

TAKES THE TØP ELEMENT ØN THE S-STACK AND RETURNS TRUE ØR FALSE (AS INPUT IS ZERØ ØR NØT) TØ THE S-STACK BEFØRE RETURNING CØNTRØL TØ THE EXECUTER

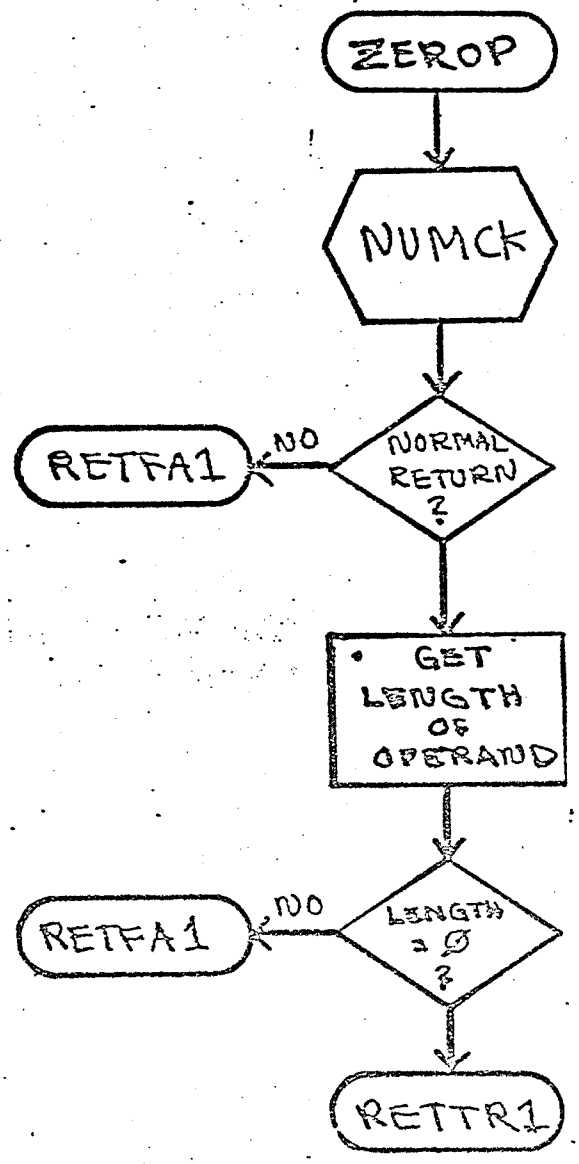
INTERNAL SPECIFICATIONS

NAME: ZERØP

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK ZERØP

CALLED RØUTINES: TRUE, FALSE, RTNSPT, NUMCK



WORDP

TAKES THE TOP ELEMENT ON THE S-STACK AND RETURNS TRUE OR FALSE (AS INPUT IS A WORD OR NOT) TO THE S-STACK BEFORE RETURNING CONTROL TO THE EXECUTER

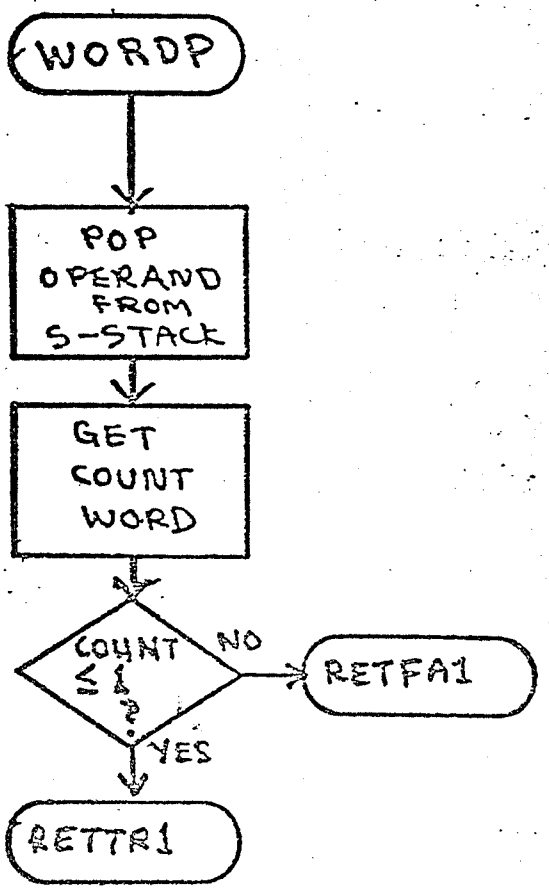
INTERNAL SPECIFICATIONS

NAME: WORDP

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK WORDP

CALLED ROUTINES: TRUE, FALSE, RTNSPT



SENTENCEP

LIKE WORDP EXCEPT TESTS IF THE INPUT IS A SENTENCE

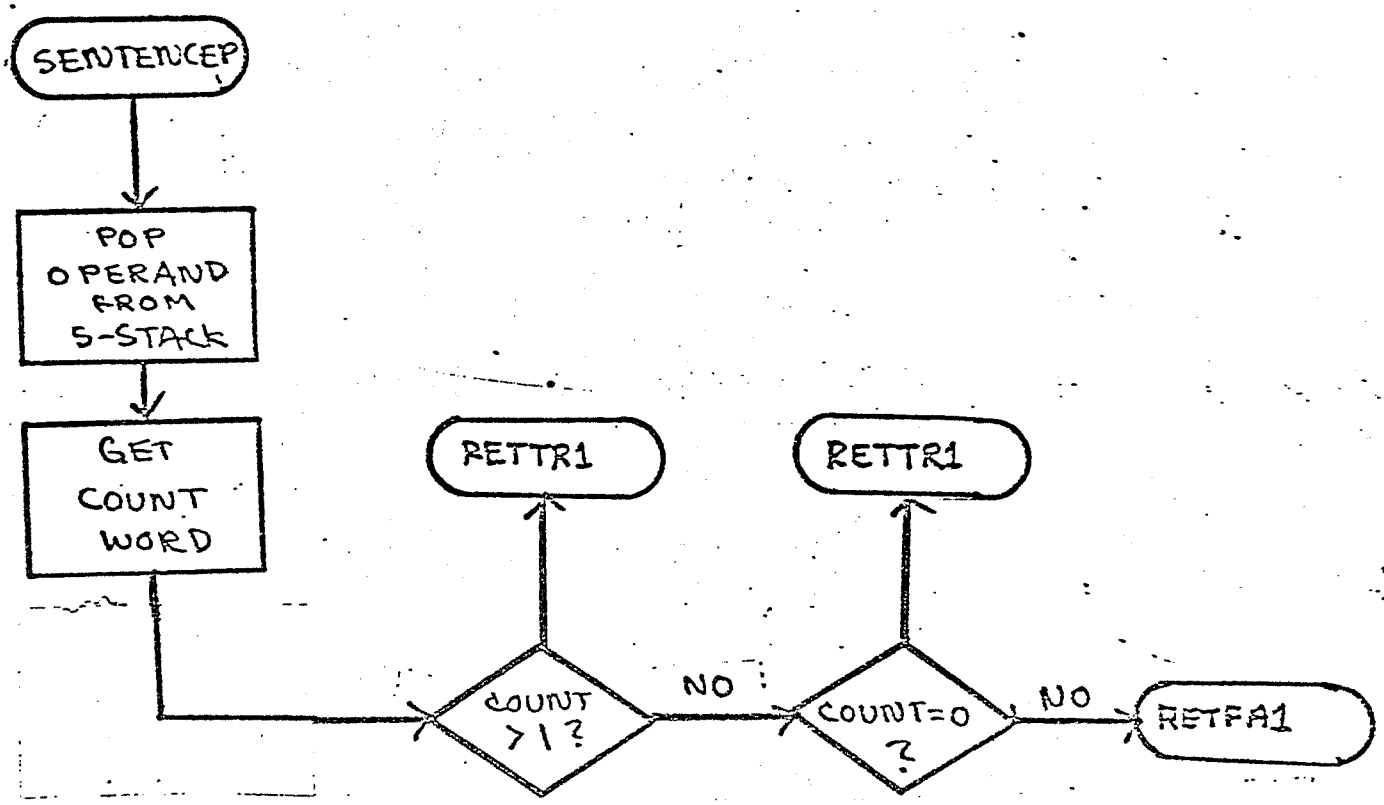
INTERNAL SPECIFICATIONS

NAME: SENTENCEP

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK SENTENCEP

CALLED ROUTINES: TRUE, FALSE, RTNSPT



NUMBERP

LIKEWISE TESTS WHETHER THE INPUT IS A NUMBER

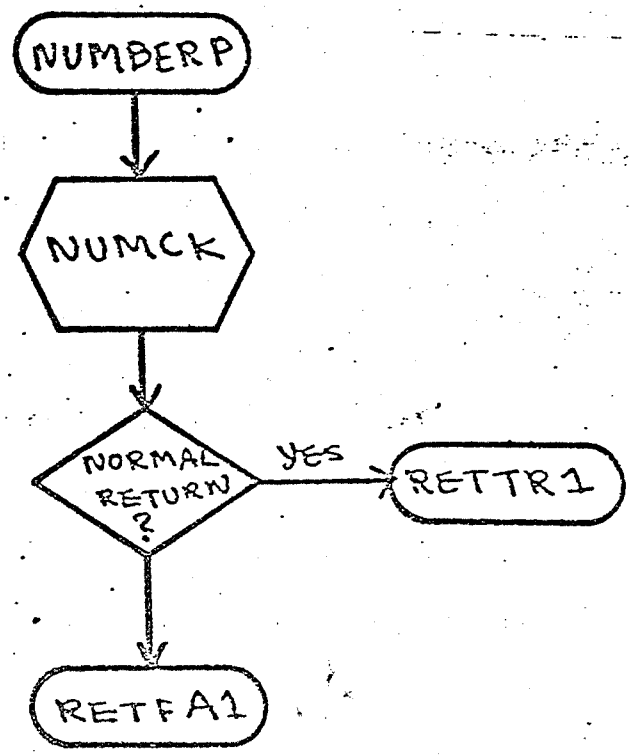
INTERNAL SPECIFICATIONS

NAME: NUMBERP

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK NUMBERP

CALLED RSUTINES: TRUE, FALSE, RTNSPT, NUMCK



GREATERP

RETURNS TRUE OR FALSE AS TO WHETHER THE FIRST INPUT (FROM THE S-STACK) IS GREATER THAN THE SECOND. ERROR RETURN IF THEY ARE NOT BOTH NUMBERS

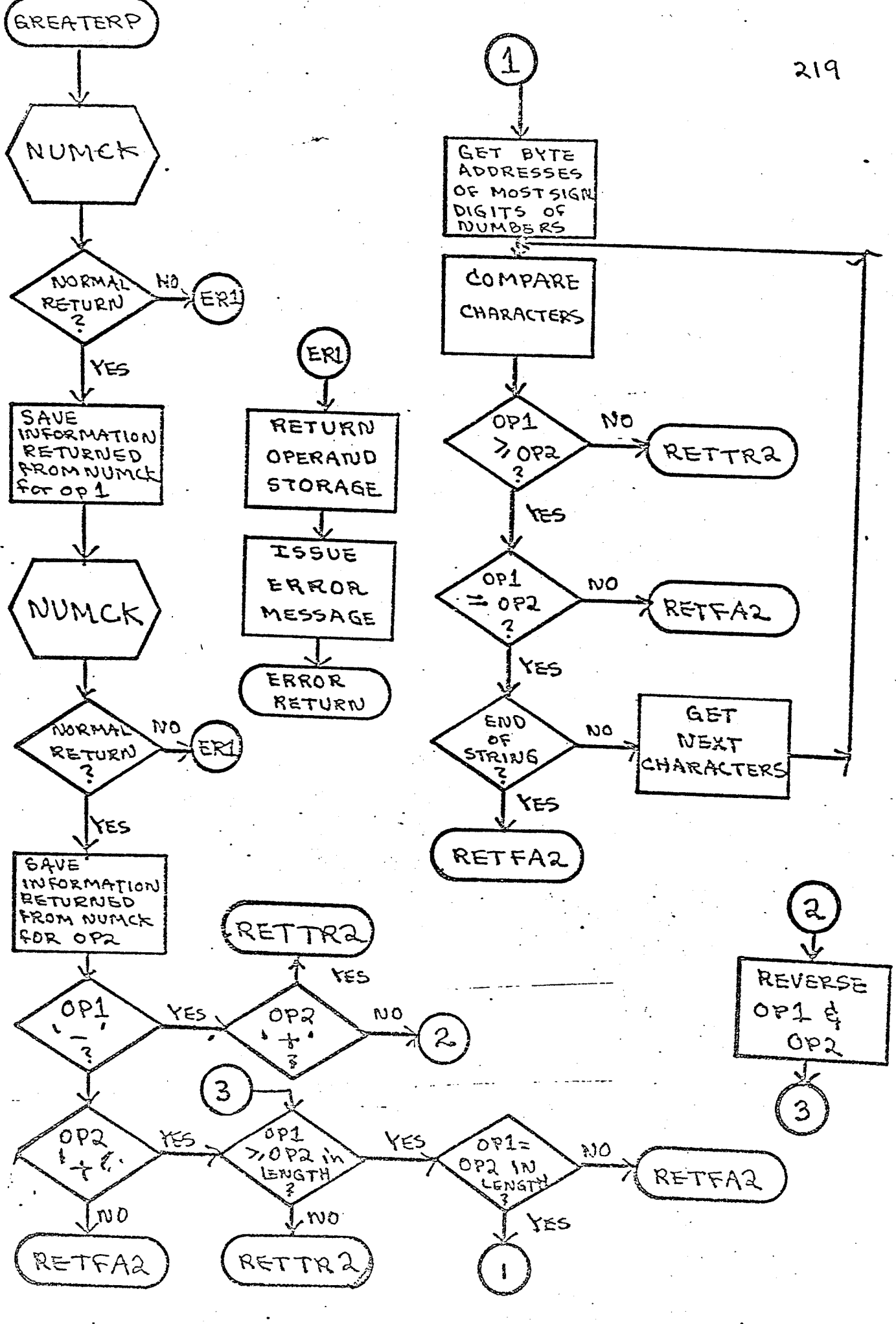
INTERNAL SPECIFICATIONS

NAME: GREATERP

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK GREATERP

CALLED ROUTINES: TRUE, FALSE, RTNSPT, NUMCK


GREATERP
NUMCK
NORMAL RETURN ?
NO
ER1
YES
SAVE INFORMATION RETURNED FROM NUMCK FOR OP1
NUMCK
NORMAL RETURN ?
NO
ER1
YES
SAVE INFORMATION RETURNED FROM NUMCK FOR OP2
ER1
RETURN OPERAND STORAGE
ISSUE ERROR MESSAGE
ERROR RETURN
1
GET BYTE ADDRESSES OF MOST SIGN DIGITS OF NUMBERS
COMPARE CHARACTERS
OP1 >= OP2 ?
NO
RETTR2
YES
OP1 = OP2 ?
NO
RETFA2
YES
END OF STRING ?
NO
GET NEXT CHARACTERS
YES
RETFA2
OP1 '-' ?
YES
OP2 '+' ?
NO
2
RETTR2
YES
OP2 '+' ?
YES
3
OP1 >= OP2 in LENGTH ?
YES
OP1 = OP2 IN LENGTH ?
NO
RETFA2
YES
1
NO
RETFA2
NO
RETTR2
2
REVERSE OP1 & OP2
3

RETURNS TRUE IF BOTH INPUTS ARE THE SAME

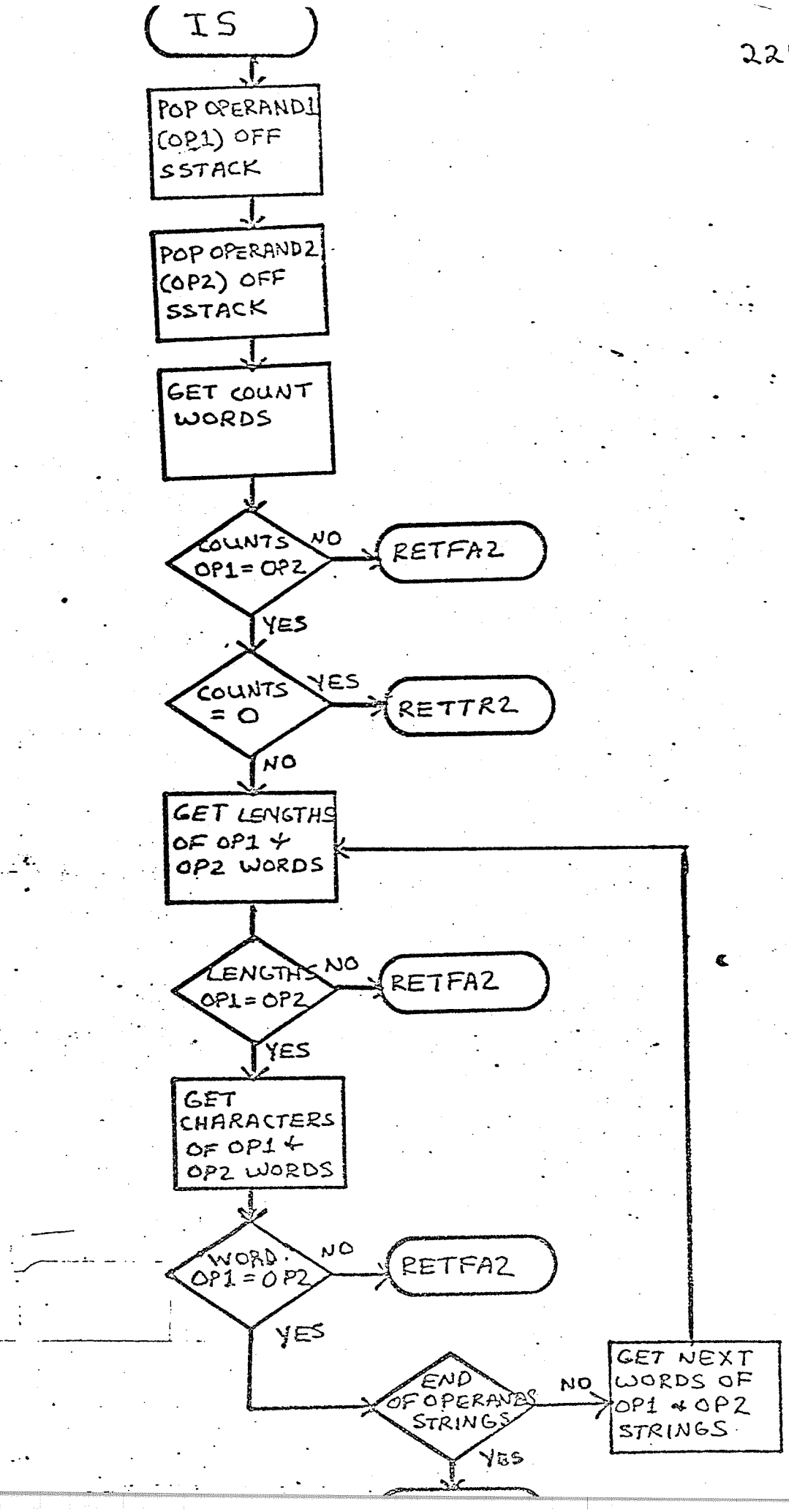
INTERNAL SPECIFICATIONS

NAME: IS

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK IS

CALLED ROUTINES: TRUE, FALSE, RTNSPT


IS
POP OPERAND1 (OP1) OFF SSTACK
POP OPERAND2 (OP2) OFF SSTACK
GET COUNT WORDS
COUNTS OP1 = OP2
NO
RETFA2
YES
COUNTS = 0
YES
RETTR2
NO
GET LENGTHS OF OP1 & OP2 WORDS
LENGTHS OP1 = OP2
NO
RETFA2
YES
GET CHARACTERS OF OP1 & OP2 WORDS
WORD OP1 = OP2
NO
RETFA2
YES
END OF OPERANDS STRINGS
NO
GET NEXT WORDS OF OP1 & OP2 STRINGS
YES

RETURNS THE LOGICAL AND OF THE VALUES OF TWO PREDICATES
ERROR RETURN IF INPUTS ARE NOT 'TRUE' OR 'FALSE'

INTERNAL SPECIFICATIONS

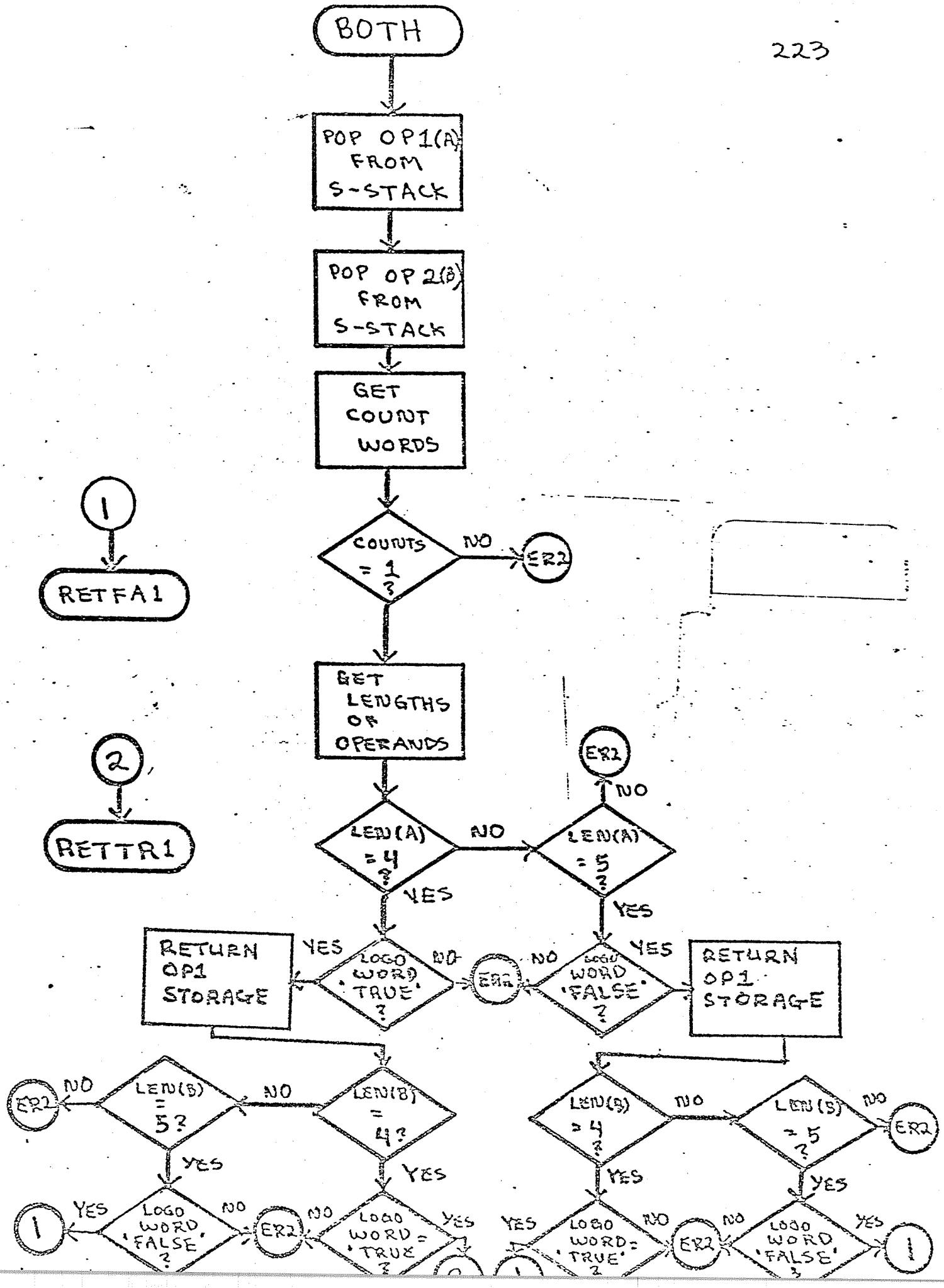
NAME: BOTH

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK BOTH

B ERROR RETURN

CALLED ROUTINES: TRUE, FALSE, RTNSPT


BOTH
POP OP1(A) FROM S-STACK
POP OP2(B) FROM S-STACK
GET COUNT WORDS
COUNTS = 1 ?
NO
ER2
1
RETFA1
GET LENGTHS OF OPERANDS
2
RETTR1
LEN(A) = 4 ?
NO
YES
LEN(A) = 5 ?
ER2
NO
YES
RETURN OP1 STORAGE
LOGO WORD 'TRUE' ?
YES
NO
ER2
NO
LOGO WORD 'FALSE' ?
YES
RETURN OP1 STORAGE
LEN(B) = 5 ?
ER2
NO
NO
LEN(B) = 4 ?
YES
YES
1
YES
LOGO WORD 'FALSE' ?
NO
ER2
NO
LOGO WORD 'TRUE' ?
YES
LEN(B) > 4 ?
NO
LEN(B) = 5 ?
NO
ER2
YES
YES
YES
LOGO WORD 'TRUE' ?
NO
ER2
NO
LOGO WORD 'FALSE' ?
YES
1

RETURNS THE LOGICAL OR OF THE VALUES OF TWO PREDICATES
ERROR RETURN IF INPUTS ARE NOT 'TRUE' OR 'FALSE'

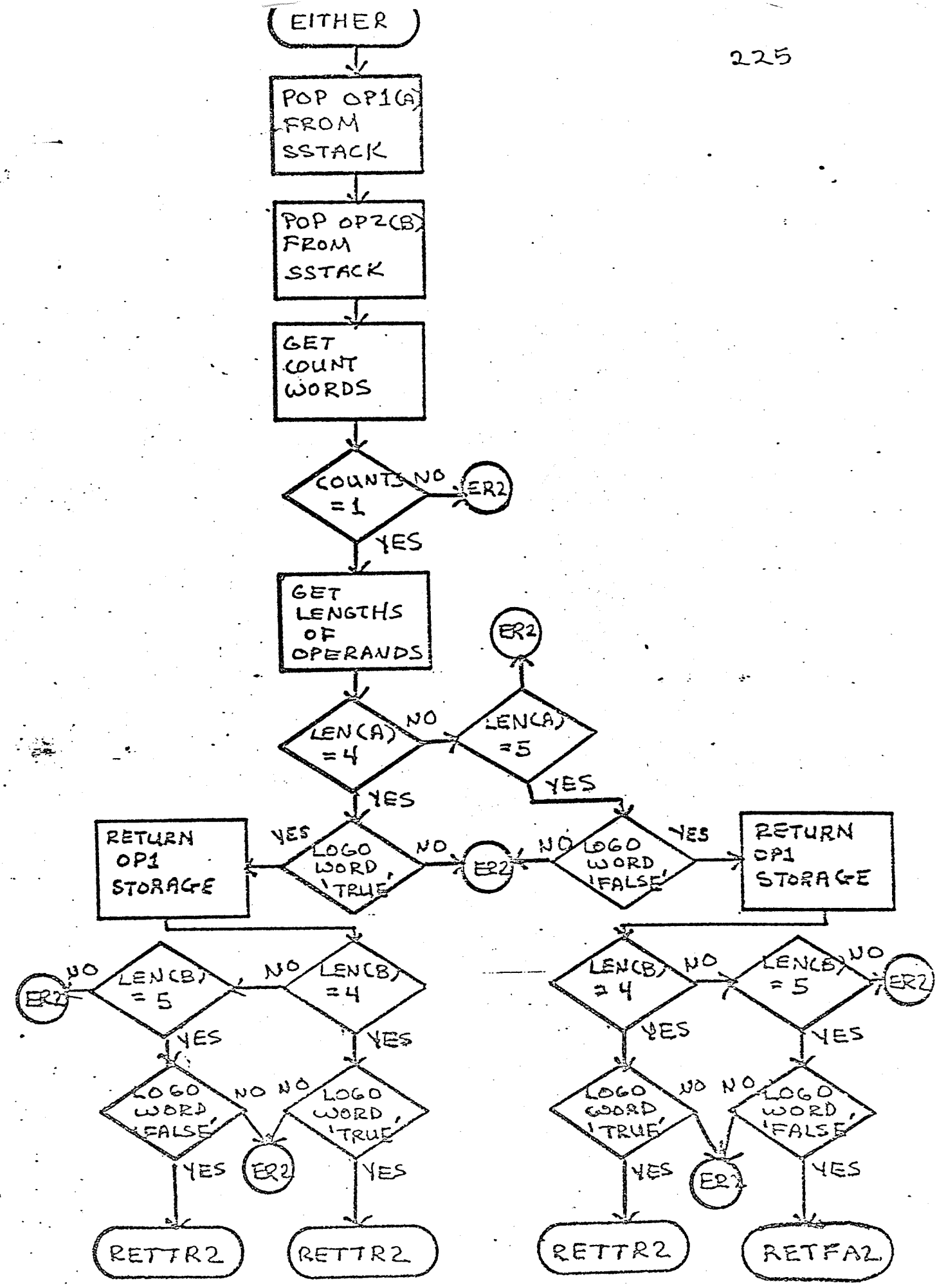
INTERNAL SPECIFICATIONS

NAME: EITHER

CALLED BY: EXECUTE

CALLING SEQUENCE: BAL,LINK EITHER

CALLED ROUTINES: TRUE, FALSE, RTNSPT


EITHER
POP OP1(A) FROM SSTACK
POP OP2(B) FROM SSTACK
GET COUNT WORDS
COUNT = 1
NO
ER2
YES
GET LENGTHS OF OPERANDS
LEN(A) = 4
NO
LEN(A) = 5
ER2
YES
YES
RETURN OP1 STORAGE
YES
LOGO WORD 'TRUE'
NO
ER2
NO
LOGO WORD 'FALSE'
YES
RETURN OP1 STORAGE
LEN(B) = 5
NO
ER2
NO
LEN(B) = 4
YES
YES
LOGO WORD 'FALSE'
NO
NO
LOGO WORD 'TRUE'
ER2
YES
YES
RETTR2
RETTR2
LEN(B) = 4
NO
LEN(B) = 5
NO
ER2
YES
YES
LOGO WORD 'TRUE'
NO
NO
LOGO WORD 'FALSE'
ER2
YES
YES
RETTR2
RETFA2

FOOTNOTES

1. XEROX UNIVERSAL TIME-SHARING SYSTEM, TIME-SHARING REFERENCE MANUAL; 90 09 07C; XEROX DATA SYSTEMS; EL SEGUNDO, CA; 1971. PP. 65-69

2. UTS 800 TECHNICAL MANUAL.

3. LOGO REFERENCE MANUAL, LOGO PROJECT NSF-C 615; BY WALLACE FEURZIG, GEORGE LUKAS, AND RICHARD GRANT; PUBLISHED BY BOLT BERANEK AND NEWMAN, INC.; CAMBRIDGE, MASS.; VOL 1, PP. 4-7.

4. IBID, PP.8-18.

BIBLIOGRAPHY

1. BOLT BERANEK AND NEWMAN, INC.; PROGRAMMING LANGUAGES AS A CONCEPTUAL FRAMEWORK FOR TEACHING MATHEMATICS, THE LOGO PROCESSOR, A GUIDE FOR SYSTEM PROGRAMMERS; REPORT NO. 2165; VOL 4; 1971.

2. BOLT BERANEK AND NEWMAN, INC; REPORT NO. 1889.

3. WALLACE FEURZIG, GEORGE LUKAS, AND RICHARD GRANT; LOGO REFERENCE MANUAL, LOGO PROJECT NSF-C 615; BOLT BERANEK AND NEWMAN, INC.; CAMBRIDGE, MASS.; VOL1, PART 3.

4. UTS ECO TECHNICAL MANUAL.

5. XEROX DATA SYSTEMS; XEROX UNIVERSAL TIME-SHARING SYSTEM, TIME-SHARING REFERENCE MANUAL; 90 09 07C; 1971.