RECEIVED BY DTIC MAR 30 1970

MASTER

IS-2230

Physics (UC-34)
TID-4500

UNITED STATES ATOMIC ENERGY COMMISSION

Research and Development Report

THE KERTRAN COMPILING SYSTEM

by

C. L. Cate, W. J. Higby,
and R. L. Wagstaff

December 1969

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

leg

Ames Laboratory
at
Iowa State University of Science and Technology
Robert S. Hansen, Director
Contract W-7405 eng-82

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

IS-2230

This report is distributed according to the category Physics (UC-34) as listed in TID-4500.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy \$3.00; Microfiche \$0.65

TABLE OF CONTENTS

	Page
ABSTRACT	v
INTRODUCTION	1
COMPILER SPECIFICATIONS	2
CHARACTER MANIPULATION	3
Instruction Types	4
KERTRAN Statement Break-up	6
The Name Table and the Instruction Table	8
FUTURE DEVELOPMENT	11
KERTRAN COMPILING ROUTINES	12
Individual Program Descriptions	12
REFERENCES	20
APPENDIX I. SOURCE PROGRAM LISTING	21
APPENDIX II. FLOWCHARTS OF KERTRAN COMPILING ROUTINES	49
APPENDIX III. DEFINITION OF PROGRAM VARIABLES AND COMPUTER GENERATED INSTRUCTIONS	65
INDEX	70

iv

IS-2230

IS-2230

THE KERTRAN COMPILING SYSTEM

C. L. Cate, W. J. Higby, and R. L. Wagstaff

ABSTRACT

The paper is an analysis of the KERTRAN language translator that operates in conjunction with the on-line measuring system for bubble chamber film data at Iowa State University. Included is a discussion of programming considerations in designing a translator and of the methods by which these problems were resolved in the KERTRAN compiler. A detailed discussion of the major elements of the compiler, namely the instruction types, the name table, and the instruction table, is given; and an analysis of future improvements is made. The report is concluded with short descriptions of each of the KERTRAN routines, program listings, and flowcharts of the major routines.

INTRODUCTION

A system for on-line control and analysis of bubble chamber film measurements has been developed by the high energy physics staff at Ames Laboratory. The time sharing aspects of such a system necessarily imply a highly sophisticated control mechanism for the measuring process at any machine. The main concern in this system design was one of total flexibility. A physicist or graduate student who is doing some experiment on-line decides what functions his particular process will perform and what quality control criteria will be imposed on the measurements. He does not want to concern himself with details such as which measuring machines he will be using and what processes other machines will be performing. In addition, the description of this process must be done in a language that is relatively easy to use and is familiar to the one describing the process.

We have chosen FORTRAN as a basis for our language because most people involved in high energy physics are familiar with FORTRAN, and the statement structure is quite adequate to define a measuring process. Our extension of FORTRAN is KERTRAN, Kuber Expression TRANslator. Programs written in this language execute the various measuring machine functions such as lighting a view lamp, displaying a message to the machine operator, or reading a data point. In addition, they may call for major routines of the system to be executed. The major routines include the reconstruction programs, kinematics routines, and various other programs of the system. Other aspects of the on-line measuring

system are described in "Ames Laboratory On-Line Control System for Bubble Chamber Film Measuring," W. J. Higby and A. H. Klein, IS-2228 (December 1969), which describes the over-all system; "KUBER: The Control Program for the USAEC Ames Laboratory On-Line Measuring System," A. H. Klein and W. J. Higby, IS-2229 (December 1969), which discusses the KUBER translator; and "Physics Use of the Ames Laboratory On-Line Measuring System for Bubble Chamber Events," T. L. Schalk and W. J. Kernan, IS-2231 (December 1969), which describes the use of the on-line system for an actual experiment.

COMPILER SPECIFICATIONS

Several questions must be considered when one designs a language translator. The main questions are:

1. What language should be used?
2. Is efficiency important?
3. What are the storage limitations?

The compiler is written entirely in FORTRAN. Although character manipulation is more difficult in FORTRAN, some sacrifices are necessary if one desires a compiler that can be used by any installation.

We do not consider the execution efficiency of the compiler to be a major concern. If the system operation involved the continuous compilation of many KERTRAN routines, efficiency would be an important factor.

The question of storage limitations imposed on the compiler is a function of how one desires the compiler to operate with the system. Since we did not choose to time share the compiler with the on-line production system, the compiler is limited to our memory size of 16K words.

The conversion of a language source text to computer instruction code can be done in a variety of ways. One can compile directly to the instruction code (one-pass compilation), or one can break up the process into multiple steps. We actually use a three-pass process to arrive at the final task words that are executed by the on-line controller KUBER² (see Fig. 1). The main reason we chose a three-step system is that we already had the assist assembler. The first pass consists of the major logic of the KERTRAN translator. The output is machine independent in the sense that no particular computer instruction format has been assumed.

FIGURE 1. Steps in compilation of a KERTRAN source deck.

CHARACTER MANIPULATION

The major logic of any compiler or translator is composed of extensive character manipulation and testing. There exist many techniques for this type of logic. We have chosen to be quite crude in this respect for the following reasons. First, we are not professional compiler writers. In addition, we want to keep the KERTRAN translator as simple as possible so that one can easily make changes in the translator to meet his specific needs. Also, FORTRAN is very limited in character handling.

Instruction Types

The types of character strings we are dealing with in the input text are not easy to manipulate in their pure form. The input source language card, considered as a string of 80 characters, contains characters that do not aid in the language content of the statement. The seven basic types of KERTRAN source statements are: (1) arithmetic instructions, (2) logic instructions, (3) instructions to the measuring machines, (4) trapping interrupts, (5) subroutine calls, (6) equivalence forms, and (7) entry statements.

Arithmetic Instructions

Arithmetic expressions are instructions formed by operating on one or more variables, common or local, subscripted or non-subscripted with a +, -, /, or *. A special measuring machine instruction of the form $V_1 = V_2$ where V_1 is a variable and V_2 is a button board setting, MMX, or MMY, is also an arithmetic instruction. The button board setting refers to the value of column 1(BB1), 2(BB2), or 3(BB3) of the button board on the measuring machines. MMX and MMY are the x and y coordinates of a point on the measuring machines.¹ In KERTRAN arithmetic statements, parentheses may be used only to indicate a subscripted variable, not to denote hierarchy of operations.

Logic Instructions

Logic instructions control program flow. The statements are of the following forms:

1. IF ($V_1 \odot V_2$) LABEL1, LABEL2 where V_1 and V_2 are variables, $\odot$ is a logical operator (either EQ, LT, or GT), and LABEL1 and LABEL2 are labels of statements in the program. If the expression in parentheses is true, program flow is transferred to the statement labeled LABEL1. Otherwise control is transferred to statement LABEL2.

2. GO TO LABEL1. The program branches to statement LABEL1.

3. IF (MFAIL) LABEL1, LABEL2 Test for memory fail.

4. IF (BADDAT) LABEL1, LABEL2 Test for invalid data code.

5. IF (ENDTRK) LABEL1, LABEL2 Test for end of track.

6. IF (IØFAIL) LABEL1, LABEL2 Test for IØ parity fail.

Instructions to the Measuring Machines¹

The four messages which command the measuring machines are:

1. INDAT Data is read from the measuring machines.

2. LIGHT VIEW V_1 One of the four views is displayed to the operator.

3. DISPLAY MESSAGE V_1 The machine displays one of a set of messages to the operator.

4. DISPLAY BINA. V_1 ON N A digit (V_1) is displayed on one of 15 light panels (N) to indicate frame, event number, event comment, and event type.

Trapping Interrupts

Trapping interrupts cause a waiting state until an acknowledge (ACK) or enter (ENT) is received from the operator. The form of the instruction is

PROCEED M_1, M_2

where M_1 and M_2 are either TASK or IGNORE. If ACK is pushed, the action specified by M_1 occurs; if ENT is pushed, M_2 occurs.

Subroutine Calls

The four types of subroutine calls are: (1) CALL "name" which calls a core resident subroutine, (2) CALL LINK "n" which calls a link, (3) INVOKE "entry" which transfers control to an entry label in a KERTRAN subroutine or another TAWTA, and (4) RETURN which transfers flow back to the statement immediately following the executed INVOKE.

Entry Statements

A label to which control can be transferred in a KERTRAN subroutine is declared in the main program as an entry point with the ENTRY statement.

KERTRAN Statement Break-up

The KERTRAN program consists of a combination of the above statements. Generally speaking, each of the seven types of statements contains key words, labels, variables, constants, and delimiters. The delimiters are either essential or non-essential. For example, in the arithmetic statement:

CALC A = B(1)/RR + INC*2

the equals sign, the slash, the asterisk, and the plus sign all perform the double operation of denoting a specific operation and separating the variables. However, since the blank between CALC and A and the parentheses

around 1 are necessary only to separate variables, they perform no specific operation. (Although the parentheses may be considered to be performing a subscripting operation, we shall see later that they are unnecessary in the reduced equation.) The types of delimiters and their classes are:

Essential	Non-essential
=	b (blank)
+	(
-	)
*	.
/	,

The KERTRAN statements are read a card at a time in 80A1 format into the array IRD(80). In the subroutine STAKER, the statement is reduced to a left part containing everything on the left side of the equal sign and a right part containing everything on the right side. For example, consider the following statements:

```

LABEL    A = B(1)*2/XVAL
          CALL LINK 13

```

These statements are divided into left and right parts as indicated in Fig. 2. If there is no label on a statement, a label of length zero is indicated. Since the label length is tested in a number of routines, a length must always be assigned.

Left Right

5	1	1	1	1	1	4
L	A	B	1	2	/	X
A						V
B						A
E						L
L						

Left

0	4	4	2
	C	L	1
	A	1	3
	L	N	
	L	K	

FIGURE 2. Form of stacks generated by STAKER.

The Name Table and Instruction Table

After the statement has been broken into a left and right part, a name table and instruction table are built (see Appendix III, Tables 2 and 3).

The name table (NAMTAB) is a FORTRAN double-subscripted array containing all the variables, labels, and constants of the KERTRAN program. The second subscript indicates which entry in the table is being considered. For each variable, the name table contains the number of characters in the name, the characters of the name in A1 format, name class identification, and additional code information. The name class may be one of the following types:

- 0 General common
- 1 IICOM variable
- 2 IDEVNT variable
- 3 Local to TAWTA
- 4 Label
- 5 Label but not defined
- 6 Set by subroutine CODE to indicate
no reserve storage necessary for
this name of class 3

The first three classes are all common variables. Labels which have not been encountered as statement labels are considered not defined. The additional code information indicates the presence of a subscript

and, for common variables, the variable code number determined by an equivalence class statement.

The instruction table (INSTR) is also a FORTRAN double-subscripted array containing the pointer to the label in NAMTAB, the instruction type, the pointer to NAMTAB for the operand, the end of task control, and the number to add to the operand label (e.g., JMP ALPH + 2). The pointer is the second subscript in NAMTAB(I,J). The instruction type code numbers are listed in Appendix III, Table 4. The end of task control may be either 1 for 'NEXT' or 0 for 'TASK.'

The translator begins by taking the statement as it has been divided by STAKER and making entries in the instruction table. Each time a label variable, variable, or constant is encountered, a loop is done to find if it already exists in NAMTAB or if it must be entered in the table. One KERTRAN statement may require several entries in the instruction table. For example, consider the statement:

CALC AA = B + 2 where all variables are class 3.

The following name table and instruction table could be generated for this source statement:

NAMTAB(L,K)

ENTRY	K \ L	NO. OF CHAR. IN NAME	ALL CHARACTERS OF NAME						NAME CLASS	ADDITIONAL TYPE INFORMATION
		1	2	3	4	5	6	7	8	9
1		4	C	A	L	C	Ø	Ø	4	0
2		2	A	A	Ø	Ø	Ø	Ø	3	0
3		1	B	Ø	Ø	Ø	Ø	Ø	3	0
4		1	2	Ø	Ø	Ø	Ø	Ø	6	0

Ø = blank

		INSTR(I,J)				
INSTRUCTION NUMBER	J	POINTER TO LABEL IN NAMTAB	INSTRUCTION TYPE	POINTER TO OPERAND IN NAMTAB	END OF TASK CONTROL	NO. TO ADD TO OPERAND LABEL
	I	1	2	3	4	5
1		1	1	3	0	0
2		0	9	4	0	0
3		0	4	2	0	0

The first entry in NAMTAB, for instance, indicates a class 4 name (a defined label), four characters long, named CALC. The instructions in INSTR are a description of a step-by-step process for adding B and 2. The first entry, in mnemonic form is:

LDA B

which means load the value of variable B into the A register. Next, add 2 to the contents of register A:

ADD 2

Finally, store the result in the storage location for variable AA:

STA AA

LDA is the mnemonic for instruction type 1, ADD is instruction type 9, and STA is instruction type 4. In this case, KERTRAN generates three instructions in the instruction table for one source statement.

FUTURE DEVELOPMENT

Basically, the compiler meets our requirements for efficiency of execution, amount of space necessary, and simplicity of design and use. However, in future versions of our compiler and in any adaptations of the translator to other systems, various aspects of design could be improved.

Rather than allowing only one character per word, characters could be packed four to a word. Packing the characters would have storage advantages but could imply more computer dependence since it would involve some programming in assembler.

One improvement that should be made in the subroutines is either adding another subroutine or incorporating in PUTLAB the search for a label and the setting of the end of task control in PUTLAB. These two processes are presently duplicated in many of the routines such as ARITH, DOER, and LOGIC.

Since present searching methods for elements of a table are based on performing a DO-loop, translator efficiency is reduced. Improving the efficiency would require an efficient means of generating an organized table, a more efficient hashing technique, or both.

One other addition to the system could be the facility for indicating hierarchy of arithmetic operation through the use of parentheses. Although this facility is in no way essential, it is quite familiar to FORTRAN programmers and would make statement writing much clearer by eliminating extra variable names.

KERTRAN COMPILING ROUTINES

Individual Program Descriptions

MAIN

MAIN is the program that begins the translation and controls the entire process. After initialization (calls to INITA and INITB), a card is read from the source deck. This card may be a control card, a comment, or a KERTRAN source statement. The types of control cards and their functions are:

EOF - copy end of file on output

EOJ - end of the entire job

END - end of a routine.

If the card is a KERTRAN source statement, it must be broken down by STAKER. MAIN determines the type of statement and calls the appropriate routine to process it. The process is then begun again until the end of the job is reached.

INITA

The subroutine INITA initializes a set of variables at the beginning of each program. The variables initialized in INITA are NJ, NNT, NEXTR, NIDO, NDOP, INSTR(5,200), and ICC(46).

INITB

INITB initializes ILP(7,20) and IRP(7,20) before each card is read.

EQUIVD(IPAR)

This routine processes EQUIVALENCE(IPAR=1) and DIMENSION(IPAR=2) statements. Upon entering, the position (IB) of the desired element of ILP is determined. This value is incremented each time a name is entered in NAMTAB. The variable name is tested against the entries in NAMTAB to determine if it has previously been equivalenced. The program then determines if the equivalence variable number is valid and begins the process again.

STAKER

In the subroutine STAKER the source statement is divided into a left and right part and "stacks" of the variables are built for each source statement. The form of the stacks is described on page 7. STAKER is called by MAIN for all source statements which are not EOJ, EOF, END, or comments.

BINA V

The routine BINA V generates instructions from a bina view source statement to display a number on the selected bina view. The source statements are of the form:

```
LABEL DISPLAY BINA V F ON N
```

The instructions generated are of the following form:

```
LABEL LDA F
      LDE N
      G3 **
```

TINT

Proceed statements are processed in TINT. The statements processed are of the form:

```
LABEL PROCEED ACKCNTRL,ENTCNTRL
```

where ACKCNTRL and ENTCNTRL are acknowledge and enter commands and may be IGNORE or TASK.

ARITH

The subroutine ARITH processes arithmetic statements. The statement to be processed has already been placed in common by STAKER in the arrays ILP and IRP. After determining the end of task control, ARITH searches for a label and calls PUTLAB if one exists. The program then examines the elements of IRP and ILP one at a time and generates the appropriate instruction for each operator it encounters.

DOER(IPAR)

The routine DOER processes DO (IPAR=1) and CONTINUE (IPAR=2) statements. The continue statements are used to designate the end of a DO-loop. Too many CONTINUE statements are ignored. The form of the DO statement is:

```
LABL1 DO LABL2 I=J,K,L
      .
      .
      .
LABL2 CONTINUE
```

and the following set of instructions is generated:

```

      LABL1  LDA  J
            JMP  *+3

)DØØØN  LDA  I
        ADD  L      (or 1 if L is not specified)
        STA  I
        SUB  K
        JGZ  LABL2+1
        .
        .
        .
        JMP  )DØØØN

```

The logic to keep track of the internally generated statement labels for CONTINUE processing uses the stack IDØSTK.

CØMINS(IPAR)

CØMINS processes the following instruction types:

1. INVØKE name
2. RETURN
3. CALL SUBRØUTINE name
4. CALL LINK n
5. INPUT DATA
6. LIGHT VIEW n
7. DISPLAY n

After testing for the end of task control and the presence of a label, the program tests on IPAR which indicates which of the seven statement types is being processed. The appropriate action is then taken for that instruction type.

LOGIC

In the subroutine LOGIC, instructions of the form:

IF ($V_1 \odot_L V_2$) TRUE,FALSE

are processed. First, the program builds fetch V_1 , subtract V_2 . After testing the logical operator, LOGIC builds JLZ, JGZ, and JEZ to true for $\odot_L = LT, GT, \text{ and } EQ$, respectively. It then builds JMP to false.

GENERT(IT)

GENERT fills in the instruction table entry (INSTR(2,NJ)) with the instruction type code (IT).

GENERA(MSTAK,N)

The call list of GENERA consists of a matrix MSTAK of names and the number N of elements of MSTAK. From this matrix, GENERA builds the entries in INSTR(1,NJ). For the first instruction, it fills in the pointer to the name table (NAMTAB) for the operand. For the following instructions ($J=2,N$), it does the following:

1. The instruction label pointer is set ($=0$)
2. The instruction type is set ($=0$)
3. The pointer to the name table for the operand is filled in (call to ENAM)
4. The end of task control is set ($=IENDT$).

NEWINS

NEWINS builds a new instruction with no label and end of task control equal to IENDT.

ENLAB(ISTAK)

In ENLAB, variable names are entered in NAMTAB. First, considering the name as a label, ENLAB searches NAMTAB for the label. If the label is found, the routine sets the operand pointer in the instruction to the found address. If the label is not found, it is entered in NAMTAB as undefined and the operand pointer is set.

PUTLAB(ISTAK)

PUTLAB takes the name in ISTAK and interprets it as a statement label. It then searches NAMTAB for a duplicate label. If PUTLAB finds the label but it is designated as undefined, the designation is changed to defined. If the routine encounters a duplicate defined label, a program error (duplicate label) has occurred. If the label is not found, PUTLAB enters it in NAMTAB.

ENAM(ISTAK)

This program searches NAMTAB for the name in ISTAK. When the location of the name is found, ENAM enters the location in the instruction table (INSTR(3,NJ)). If no entry for the name is found, the name is a variable local to the TAWTA and a new entry is made in NAMTAB.

CODE

CODE forms the assist instructions from the instruction table and outputs the code to ITSYS(4). In addition, CODE performs the following functions: (1) generates entry statements; (2) generates reserve storage statements; and (3) processes call subroutine, invoke, call link, and

trap interrupt statements. Since this program is specific to the EMR 6050 instruction format, it will not be discussed in further detail.

OPCODE(IT,IOP)

OPCODE sets the three-character assist code and returns the value in IOP(1).

NAMIS2(NC,ICOD)

The parameter passed to NAMIS2 is a character stack NC where

NC(1) = number of characters in the stack

NC(2-7) = characters.

NAMIS2 determines what code to assign to ICOD by the convention

NC(2-7)	ICOD
OTHER	0
BB1	1
BB2	2
BB3	3
+	4
*	5
-	6
/	7
MMX	8
MMY	9

NAMIS3(ISTAK,ICOD)

NAMIS3 determines what name is contained in ISTAK and sets ICOD by the convention:

ISTAK	ICOD
OTHER	0
LT	1
GT	2
EQ	3
ACK	4
ENT	5
LINK	6
NEXT	7
MFAIL	8
BADDAT	9
ENDTRK	10
IØFAIL	11

NAMIS1(ISTAK, ICØD)

NAMIS1 determines what name is contained in ISTAK and sets ICØD by the convention:

ISTAK	ICØD
OTHER	0
INVOKE	1
RETURN	2
CALL	3
LINK	4
INDAT	5
LIGHT	6

DISPLA	7
IF	8
GØ	9
BINA	10
PROCEE	11
ENTRY	12
EQUIVA	13
DØ	14
CONTIN	15
DIMENS	16

ICNVT1(ICCHAR, INTEG)

This routine converts the A1 character string ICHAR to the integer INTEG. If ICHAR is not an integer, then INTEG = -1.

ICNVT2(ICCHAR, INTEG)

This program converts the integer in INTEG to an A1 character string and returns the string in ICHAR.

REFERENCES

1. W. J. Higby and A. H. Klein, "Ames Laboratory On-Line Control System for Bubble Chamber Film Measuring," IS-2228 (December 1969).
2. A. H. Klein and W. J. Higby, "KUBER: The Control Program for the USAEC Ames Laboratory On-Line Measuring System," IS-229 (December 1969).

APPENDIX 1. SOURCE PROGRAM LISTING

```

C      KERTRAN MAIN V2 L1
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOF

C      MAIN ROUTINE FOR TAWTA TRANSLATOR
C
C
C
C
C
      DIMENSION IEND(3),IEOF(3),IEOJ(4)
      IEND(1)=1HE
      IEND(2)=1HN
      IEND(3)=1HD
      IEOF(1)=1HE
      IEOF(2)=1HO
      IEOF(3)=1HF
      IEOJ(1)=1HE
      IEOJ(2)=1HO
      IEOJ(3)=1HJ
      IEOJ(4)=1H/
      IC=1HC
      IBLNK=1H
      READ(9,1) (ITSYS(I),I=1,4)
1     FORMAT(4I5)
50    CALL INITA
      INUN=ITSYS(1)
      IPRIN=ITSYS(2)
      IERR=ITSYS(3)
      IPUN=ITSYS(4)
100   CALL INITB
      IF(SENSE SWITCH 1) 105,108
105   WRITE(11,106) NNT,NJ
106   FORMAT(1X,' NNT AND NJ ARE',2I5)
108   CONTINUE
      INCD=INCD+1
      READ(INUN,120) IRD
120   FORMAT(80A1)
      IF(IPRIN) 150,150,130
130   WRITE(IPRIN,140) IRD
140   FORMAT(1X,80A1)
C      IS THIS A COMMENT CARD
150   IF(IRD(1)-IC)160,155,160
155   IF(IRD(2)-IBLNK) 160,100,160
C      IS THIS AN END CARD
160   DO 170 I=1,3
      IF(IRD(I+6)-IEND(I)) 172,170,172
170   CONTINUE
      WRITE(11,106) NNT,NJ
      CALL CODE
      GO TO 50
C      TEST FOR EOF
172   DO 174 I=1,3
      IF(IRD(I+6)-IEOF(I)) 176,174,176
174   CONTINUE
      WRITE(IPUN,120) IRD
      GO TO 50
C      TEST FOR EOJ
176   DO 178 I=1,3
      IF(IRD(I)-IEOJ(I)) 180,178,180
178   CONTINUE
      IRD(4)=IEOJ(4)
      WRITE(IPUN,120) IRD
      GO TO 9999
180   CALL STAKER
      IF(NRP) 200,200,190

```

```

C          TEST FOR DO STMT
190 CALL NAMIS1(ILP(1,2),ICOD)
    IF(ICOD-14) 194,196,194
194 CALL ARITH
    GO TO 100
196 CALL DOER(1)
    GO TO 100

C          IS COL 6 NONBLANK WHICH IMPLIES
C          AN EQUIVALENCE CONTINUATION CARD
200 IF(IRD(6)-IBLNK)210,240,210
210 CALL EQUIVD(1)
    GO TO 100
240 CALL NAMIS1(ILP(1,2),ICOD)
    IF(ICOD) 250,250,260
250 WRITE(IERR,255) INCD
255 FORMAT(1X,' ERR,INVALID STATEMENT CD. NO.',I5)
    GO TO 100
260 IF(ICOD-7) 270,270,400
270 IS=ICOD
    IF(ICOD-3) 300,280,300

C          RESOLVE CALL NAME OK CALL LINK
280 CALL NAMIS1(ILP(1,3),JCOD)
    IF(JCOD-4) 330,290,330
290 IS=4
    GO TO 330
300 IF(ICOD-7) 330,310,330

C          RESOLVE DISPLA BINA OR DISPLA MSSG
C
310 CALL NAMIS1(ILP(1,3),JCOD)
    IF(JCOD-10) 330,350,330
330 CALL COMINS(IS)
    GO TO 100

C          DISPLAY BINA
350 CALL BINAV
    GO TO 100
400 CONTINUE

C          IS IT IF OR GO TO
C
    IF(ICOD-9) 410,410,430
410 CALL LOGIC
    GO TO 100

C          IS IT PROCEE
430 IF(ICOD-11) 250,440,450
440 CALL TINT
    GO TO 100

C          IS IS ENTRY OR EQUIVA
450 IF(ICOD-13) 460,210,500
460 NNAM=NNAM+1
    DO 470 I=1,7
    NAMJOB(I,NNAM)=ILP(I,3)
470 CONTINUE
    GO TO 100

C          IS IT CONTINUE
500 IF(ICOD-15) 250,510,530
510 CALL DOER(2)
    GO TO 100
530 IF(ICOD-16) 250,540,250
540 CALL EQUIVD(2)
    GO TO 100
9999 CALL EXIT
    END
    EOF

```

```

C      INITA V2 L1
      SUBROUTINE INITA
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP
      INCD=0
      NJ=0
      NNT=0
      NEXTR=0
      NNAM=0
      NIDO=0
      NDOP=0
      DO 100 I=1,5
      DO 100 J=1,600
100 INSTR(I,J)=0
      ICC(1)=1HA
      ICC(2)=1HB
      ICC(3)=1HC
      ICC(4)=1HD
      ICC(5)=1HE
      ICC(6)=1HF
      ICC(7)=1HG
      ICC(8)=1HH
      ICC(9)=1HI
      ICC(10)=1HJ
      ICC(11)=1HK
      ICC(12)=1HL
      ICC(13)=1HM
      ICC(14)=1HN
      ICC(15)=1HO
      ICC(16)=1HP
      ICC(17)=1HQ
      ICC(18)=1HR
      ICC(19)=1HS
      ICC(20)=1HT
      ICC(21)=1HU
      ICC(22)=1HV
      ICC(23)=1HW
      ICC(24)=1HX
      ICC(25)=1HY
      ICC(26)=1HZ
      ICC(27)=1H0
      ICC(28)=1H1
      ICC(29)=1H2
      ICC(30)=1H3
      ICC(31)=1H4
      ICC(32)=1H5
      ICC(33)=1H6
      ICC(34)=1H7
      ICC(35)=1H8
      ICC(36)=1H9
      ICC(37)=1H+
      ICC(38)=1H-
      ICC(39)=1H*
      ICC(40)=1H/
      ICC(41)=1H(
      ICC(42)=1H)
      ICC(43)=1H.
      ICC(44)=1H
      ICC(45)=1H.
      ICC(46)=1H=
      DO 5 J=1,20
      ILP(1,J)=0
5 IRP(1,J)=0
      DO 10 I=2,7
      DO 10 J=1,20
      ILP(I,J)=ICC(44)
10 IRP(I,J)=ICC(44)
      RETURN
      END
      EOF

```

```

C      INITB V2 L1
      SUBROUTINE INITB
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS,IDOSTK,NIDO,NDOP
      DO 5 I=1,20
      ILP(1,I)=0
5     IRP(1,I)=0
      DO 10 I=2,7
      DO 10 J=1,20
      ILP(I,J)=ICC(44)
10    IRP(I,J)=ICC(44)
      RETURN
      END
      EOF

```



```

C      EQUIVD V2 L1
      SUBROUTINE EQUIVD(IPAR)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X      INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X      NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP

C      PROCESS EQUIVALENCE STATEMENTS BY MAKING
C      ENTRIES TO NAMTAB
C      IERR=ITSYS(3)

C
C      IBLNK=1H
      IF(IPAR-2) 30,20,20
20    IB=2
      ICLS=3
      GO TO 100

C      IS THIS A CONTINUATION CARD
C
30    IF(IRD(6)-IBLNK) 60,40,60
40    IB=4
      CALL ICNVT1(ILP(1,IB),ICLS)
      GO TO 100
60    IB=1

C      ANY VARIABLES ON THIS CARD
100   IF(IB-NLP)150,9999,9999
150   IB=IB+1

C      ARE WE FINISHED WITH CARD
C
      IF(IB-NLP) 160,9999,9999

C
C      ANY NAMES IN NAMTAB YET
C
160   IF(NNT) 260,260,180

C      YES, SO LOOK FOR DUPLICATE NAME
C
180   N=ILP(1,IB)
      DO 250 I=1,NNT
        IF(NAMTAB(1,I)-N) 250,190,250
190   DO 200 J=1,N
        IF(NAMTAB(J+1,I)-ILP(J+1,IB))250,200,250
200   CONTINUE

C      DUPLICATE NAME
C
      WRITE(IPRIN,210) (ILP(I+1,IB),I=1,N)
210   FORMAT(2X,' ERROR',1X,6A1)
      WRITE(IPRIN,215)
215   FORMAT(2X,' HAS PREV. BEEN DEFINED IN EQUIV'/2X,' THIS REF IGNOR')
      IB=IB+1
      GO TO 150
250   CONTINUE

C      NO DUP. NAME,SO ENTER IN NAMTAB
260   NNT=NNT+1
      DO 280 I=1,7
        NAMTAB(I,NNT)=ILP(I,IB)
280   CONTINUE
      NAMTAB(8,NNT)=ICLS

C      GET INTEGER EQUIV FOR ALPHA
      IB=IB+1

C
      CALL ICNVT1(ILP(1,IB),KK)
      IF(KK) 290,400,400
290   WRITE(IERR,300) INCD
300   FORMAT(1X,' ERR EQUIV VARIABLE NUMBER NOT VALID, CD NO IS',I3)
      NAMTAB(9,NNT)=0
      GO TO 150
400   CONTINUE
      NAMTAB(9,NNT)=KK
      GO TO 150
9999  CONTINUE
      RETURN
      END
      EOF

```

```

C      STAKER V2 L1
      SUBROUTINE STAKER
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP

C      PACK STATEMENT LABEL
C
C      ICL=0
      NNBLNK=0
      DO 20 I=1,5
      IF(IRD(I)-ICC(44)) 10, 20, 10
10     NNBLNK=NNBLNK+1
      ILP(NNBLNK+1,1)=IRD(I)
20     CONTINUE
      ILP(1,1)=NNBLNK
      ISW=-1
      ICH=6
      NLP=2
      NRP=1
      IL=0
30     IDUPSW=0
40     ICH=ICH+1
      ICL=0
      IF(ICH-73) 50, 200, 200
50     IC=IRD(ICH)
      DO 60 I=37,46
      IF(IC-ICC(I)) 60, 65, 60
60     CONTINUE
      GO TO 70
65     IF(I-40) 80, 80, 90
90     IF(I-45) 100, 100, 110
80     ICL=1
      IF(IDUPSW) 70, 70, 300
100    ICL=2
      IF(IDUPSW) 300, 40, 300
105    IDUPSW=0
      GO TO 40
110    ICL=3
      IF(IDUPSW) 300, 115, 300
115    NLP=NLP-1
      ISW=+1
      IR=0
      GO TO 30
200    IF(ISW) 210, 210, 220
210    NLP=NLP-1
220    NRP=NRP-1
      GO TO 500

C      PACK IC
70     CONTINUE
      IF(ISW) 230, 230, 240
230    IF(IL-6) 235,250,250
235    IL=IL+1
      ILP(IL+1,NLP)=IC
      GO TO 250
240    IF(IR-6) 245,250,250
245    IR=IR+1
      IRP(IR+1,NRP)=IC
250    IDUPSW=1
      IF(ICL-1) 40, 45, 40
45     ICL=2
      GO TO 300

C      THING DONE
300    CONTINUE
      IF(ISW) 310, 310, 320
310    ILP(1,NLP)=IL
      NLP=NLP+1
      IL=0
315    GO TO (70,105,115),ICL
320    IRP(1,NRP)=IR
      NRP=NRP+1
      IR=0
      GO TO 315
500    CONTINUE
      RETURN
      END
      EOF

```

```

C      BINAV V2 L1
      SUBROUTINE BINAV
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP
      DIMENSION M(3)
C      PROCESS DISPLAY BINAVIEW STATEMENTS
      DATA M(1),M(2),M(3) / 2,1HD,1HN/
      IERR=ITSYS(3)
      NJ=NJ+1
C      SET UP LABEL AND END OF TASK CONTROL
      CALL NAMIS3(ILP(1,NLP),ICOD)
      IF(ICOD=7) 120,110,120
110  IENDT=1
      NLP=NLP-1
      GO TO 130
120  INDT=0
130  INSTR(4,NJ)=IENDT
      IF(ILP(1,1)) 140,140,150
140  INSTR(1,NJ)=0
      GO TO 160
150  CALL PUTLAB(ILP(1,1))
C      BUILD LDA
160  IST=4
      K=0
      CALL GENERT(1)
170  K=K+1
      INDX=K+IST-1
      IF(INDX-NLP) 180,9910,9910
180  DO 190 I=1,3
      IF(ILP(I,INDX)-M(I))170,190,170
190  CONTINUE
      K=K-1
      CALL GENERA(ILP(1,IST),K)
      CALL NEWINS
C      GENERATE LDE
      CALL GENERT(2)
      IST=INDX+1
      IF(IST-NLP) 200,200,9910
200  K=NLP-INDX
      CALL GENERA(ILP(1,IST),K)
      CALL NEWINS
      CALL GENERT(29)
      GO TO 9999
9910 WRITE(IERR,9915) INCD
9915 FORMAT(1X,' ERR DISPLAY BINA STMT IN ERR CD NO',1X,15)
      GO TO 9999
9999 RETURN
      END
      EOF

```

```

C      TINT V2 L2
      SUBROUTINE TINT
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS, IDOSTK,NIDO,NDOP
C
C      ROUTINE TO PROCESS PROCEED STATEMENTS
C
C      DIMENSION M(7,3)
      DATA
X     M(1,1),M(2,1),M(3,1),M(4,1),M(5,1),M(6,1),M(7,1),
X     M(1,2),M(2,2),M(3,2),M(4,2),M(5,2),
X     M(1,3),M(2,3),M(3,3),M(4,3),M(5,3) /
X     6,1HI,1HG,1HN,1HO,1HR,1HE,
X     4,1HT,1HA,1HS,1HK,
X     4,1HN,1HE,1HX,1HT /
      IERR=ITSYS(3)
C      PROCESS LABEL AND END OF TASK CONTROL
C
      NJ=NJ+1
C      SET END OF TASK CONTROL TO TASK
      IENOT=0
130  INSTR(4,NJ)=IENOT
      IF(ILP(1,1)) 140,140,150
140  INSTR(1,NJ)=0
      GO TO 160
150  CALL PUTLAB(ILP(1,1))
C
160  IST=3
      IF(IST+1-NLP) 9910,180,9910
180  N=ILP(1,IST)
      DO 250 I=1,3
        IF(N-M(1,I)) 250,190,250
190  DO 200 J=1,N
        IF(ILP(J+1,IST)-M(J+1,I)) 250,200,250
200  CONTINUE
        IAK=I-1
        GO TO 300
250  CONTINUE
C      ERROR
      GO TO 9910
C      NOW DETERMINE ENTER INTERRUPT
C      CONTROL
C
300  IST=IST+1
      N=ILP(1,IST)
      DO 350 I=1,3
        IF(N-M(1,I)) 350,310,350
310  DO 330 J=1,N
        IF(ILP(J+1,IST)-M(J+1,I)) 350,330,350
330  CONTINUE
        IEN=I-1
        GO TO 400
350  CONTINUE
        GO TO 9910
400  ICNT=IEN*10+IAK
      CALL GENERT(30)
      INSTR(3,NJ)=ICNT
      GO TO 9999
9910 WRITE(IER,9915) INCD
9915 FORMAT(1X,' ERR PROCEED STMT IN ERROR,CARD NO',1X,15)
      GO TO 9999
9999 RETURN
      END
      EOF

```

```

SUBROUTINE ARITH
  DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
  X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
  COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
  X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOF
  CALL NAMIS3(IRP(1,NRP),ICOD)
  IF(ICOD-7) 10, 20, 10
10 IENDT=0
  GO TO 30
20 NRP=NRP-1
  IENDT=1
30 NJ=NJ+1
  INSTR(4,NJ)=IENDT
  IF(ILP(1,1)) 50,50,40
40 CALL PUTLAB(ILP(1,1))
  GO TO 60
50 INSTR(1,NJ)=0
60 CALL NAMIS2(IRP(1,1),ICOD)
  IF(ICOD-4) 80, 70, 80
70 IT=1
  GO TO 95
80 IF(ICOD-6) 100, 90, 100
90 IT=3
95 IST=1
  GO TO 110
100 IF(ICOD.EQ.1.OR.ICOD.EQ.2.OR.ICOD.EQ.3.OR.ICOD.EQ.8.OR.
  1 ICOD.EQ.9) GO TO 170
  IST=0
  IT=1
110 CALL GENERT(IT)
120 K=0
  IST=IST+1
  IPT=IST
125 IF(IST-NRP) 130,140,140
130 CALL NAMIS2(IRP(1,IST),ICOD)
  IF(ICOD.EQ.4.OR.ICOD.EQ.5.OR.ICOD.EQ.6.OR.ICOD.EQ.7) GO TO 150
  K=K+1
  IST=IST+1
  GO TO 125
150 CALL GENERA(IRP(1,IPT),K)
  CALL NEWINS
  IT=ICOD+5
  CALL GENERT(IT)
  GO TO 120
C                                     END OF STACK
140 K=K+1
  CALL GENERA(IRP(1,IPT),K)
160 CALL NEWINS
  IT=4
  CALL GENERT(IT)
  CALL GENERA(ILP(1,2),NLP-1)
  RETURN
170 IT=ICOD+5
  CALL GENERT(IT)
  INSTR(3,NJ)=0
  GO TO 160
  END
  EOF

```

```

C      DOER V2 L1
      SUBROUTINE DOER(IPAR)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NE XTR,ITSYS ,IDOSTK,NIDO,NDOP
      DIMENSION KSTAK(7),IDOLAB(7),ISTN(7)
C      ROUTINE TO COMPILE DC LOOPS
C      IPAR=1 FOR      LABL1 DO LABL2 I=J,K,L
C      IPAR=2 FOR      LABL2 CONTINUE
C
      DATA IDOLAB(2),IDOLAB(3),IDOLAB(4),IDOLAB(5),IDOLAB(6),IDOLAB(7)/
X     1H,1H0,1H0,1H0,1H0,1H0/
      NJ=NJ+1
      IERR=ITSYS(3)
      IF(ILP(1,1)) 50,50,40
40     CALL PUTLAB(ILP(1,1))
      GO TO 60
50     INSTR(1,NJ)=0
C      IS THIS DO OR CONTINUE
60     IF(IPAR-1) 500,100,500
C
C      BEGIN PROCESSING FOR DO
100    NDOP=NDOP+1
C      SET END OF TASK CONTROL
      CALL NAMIS3(IRP(1,NRP),ICOD)
      IF(ICOD-7) 110,120,110
110    IENDT=0
      GO TO 130
120    IENDT=1
      NRP=NRP-1
130    INSTR(4,NJ)=IENDT
C      GEN. LDA J
      CALL GENERT(1)
      CALL ENAM(IRP(1,1))
C      GEN JMP **3
      CALL NEWINS
      CALL GENERT(15)
      KSTAK(1)=3
      KSTAK(2)=1H*
      KSTAK(3)=1H+
      KSTAK(4)=1H3
      CALL ENLAB(KSTAK)
C      GEN. IDOON LDA I
      CALL NEWINS
      DO 150 I=2,7
      KSTAK(1)=IDOLAB(I)
150    CONTINUE
      CALL ICNVT2(ISTN,NDOP)
      N=ISTN(1)
      DO 160 I=1,N
      KSTAK(I+4)=ISTN(I+1)
160    CONTINUE
      KSTAK(1)=N+3
      NIDO=NIDO+1
      IDOSTK(NIDO)=NDOP
      CALL PUTLAB(KSTAK)
      CALL GENERT(1)
      CALL ENAM(ILP(1,4))
C      GEN. ADD L
      CALL NEWINS

```

```

      CALL GENERT(9)
C      TEST FOR NO INCREMENT SPECIFIED
      IF(NRP-2) 9910,180,170
170  IF(NRP-3) 9910,190,9910
180  KSTAK(1)=1
      KSTAK(2)=1H1
      CALL ENAM(KSTAK)
      GO TO 200
190  CALL ENAM(IRP(1,3))
C      GEN. STA I
200  CALL NEWINS
      CALL GENERT(4)
      CALL ENAM(ILP(1,4))
C      GEN. SUB K
      CALL NEWINS
      CALL GENERT(11)
      CALL ENAM(IRP(1,2))
C      GEN JGZ LABL2+1
      CALL NEWINS
      CALL GENERT(20)
      CALL ENLAB(ILP(1,3))
      INSTR(5,NJ)=1
      GO TO 9999
C
C      BEGIN PROCESSING FOR CONTINUE
C
C
500  CONTINUE
C      SET END OF TASK CONTROL
      CALL NAMIS3(ILP(1,3),ICOD)
      IF(ICOD-7) 510,520,510
510  IENDT=0
      GO TO 530
520  IENDT=1
530  INSTR(4,NJ)=IENDT
C      GEN JMP
      CALL GENERT(15)
C      PICK UP LABEL TO JMP TO FROM IDOSTK
      IF(NIDO) 9920,9920,550
550  K=IDOSTK(NIDO)
      DO 560 I=2,7
      KSTAK(I)=IDOLAB(I)
560  CONTINUE
      CALL ICNVT2(ISTN,K)
      N=ISTN(1)
      DO 570 I=1,N
      KSTAK(I+4)=ISTN(I+1)
570  CONTINUE
      KSTAK(1)=N+3
      CALL ENLAB(KSTAK)
      NIDO=NIDO-1
      GO TO 9999
9910 WRITE(IERR,9915) INCD
9915 FORMAT(1X,' ERR DO STMT CODED WRONG,CARD NO',1X,15)
      GO TO 9999
9920 WRITE(IERR,9925) INCD
9925 FORMAT(1X,' ERR TOO MANY CONTINUES, CD NO',1X,15/
      X 1X,' IS IGNORED')
      GO TO 9999
9999 RETURN
      END

```

```

C      COMINS V2 L1
      SUBROUTINE COMINS(IPAR)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS,IDOSTK,NIDO,NDOP
      NJ=NJ+1
      CALL NAMIS3(ILP(1,NLP),ICOD)
      IF(ICOD-7) 10, 20, 10
10     IENDT=0
      GO TO 30
20     NLP=NLP-1
      IENDT=1
30     INSTR(4,NJ)=IENDT
      IF(ILP(1,1)) 40,50,40
40     CALL PUTLAB(ILP(1,1))
      GO TO 60
50     INSTR(1,NJ)=0
60     GO TO (70,80,90,100,80,105,105),IPAR
70     CALL ENLAB(ILP(1,3))
      GO TO 110
80     INSTR(3,NJ)=0
      GO TO 110
90     IF(NEXTR) 95,95,91
91     N=ILP(1,3)
      DO 94 I=1,NEXTR
      IF(N-NAMEXT(1,I)) 94,92,94
92     DO 93 II=1,N
      IF(ILP(II+1,3)-NAMEXT(II+1,1)) 94,93,94
93     CONTINUE
      GO TO 97
94     CONTINUE
C      NO MATCH FOUND
95     NEXTR=NEXTR+1
      I=NEXTR
      DO 96 K=1,7
96     NAMEXT(K,NEXTR)=ILP(K,3)
97     INSTR(3,NJ)=I
      GO TO 110
100    CALL ICNVT1(ILP(1,4),INTEG)
      INSTR(3,NJ)=INTEG
      GO TO 110
105    CALL GENERT(1)
      N=NLP-3
      IST=4
      CALL GENERA(ILP(1,IST),N)
      CALL NEWINS
      INSTR(3,NJ)=0
      GO TO 110
110    ITYP=IPAR+21
      CALL GENERT(ITYP)
      RETURN
      END
      EOF

```



```

C      LOGIC V2 L3
C      SUBROUTINE LOGIC
C      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
C      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NE XTR,ITSYS
C      DIMENSION IGO(2)
C      DIMENSION KSTAK(7)

C      PROCESS STATEMENTS OF THE TYPE
C      LABEL IF(A.OP.B) TRUE,FALSE
C      PROCESS IS TO BUILD FETCH A,SUBTRACT B,
C      AND BUILD JLZ LT
C      JGZ TO TRUE FOR OP= GT
C      JEZ EQ
C      THEN BUILD JMP TO FALSE

C      IGO(1)=1HG
C      IGO(2)=1HO
C      IPRIN=11
C      NJ=NJ+1

C      IS LAST WORD ON STMT NEXT
C      CALL NAMIS3(ILP(1,NLP),ICOD)
C      IF(ICOD-7) 50,40,50
40  IENDT=1
C      NLP=NLP-1
C      GO TO 60
50  IENDT=0

C      SET STMT END OF TASK CNTRL
60  INSTR(4,NJ)=IENDT

C      ANY LABEL ON SOURCE
C      IF(ILP(1,1)) 9910,75,70
70  CALL PUTLAB(ILP(1,1))
C      GO TO 80
75  INSTR(1,NJ)=0
80  IST=3
C      K=0

C      CHECK FOR GO TO STATEMENT
C      DO 81 I=1,2
C      IF(ILP(I+1,2)-IGO(I)) 82,81,82
81  CONTINUE

C      CHECK FOR COMPUTED GO TO
C      IF(NLP-4) 220,220,810
810  CALL GENERT(1)
C      CALL ENAM(ILP(1,NLP))
C      CALL NEWINS
C      CALL GENERT(9)
C      KSTAK(1)=5
C      KSTAK(2)=1H*
C      KSTAK(3)=1H+
C      KSTAK(4)=1H2
C      KSTAK(5)=1H,
C      KSTAK(6)=1H3
C      CALL ENLAB(KSTAK)

C      FORCE END OF TASK CONTROL TO NEXT
C      ISAV=IENDT
C      IENDT=1
C      CALL NEWINS
C      CALL GENERT(4)
C      CALL ENLAB(KSTAK)
C      CALL NEWINS
C      CALL GENERT(15)
C      KSTAK(4)=1H1
C      KSTAK(1)=3
C      CALL ENLAB(KSTAK)
C      IENDT=ISAV
C      CALL NEWINS
C      CALL GENERT(0)
C      INSTR(3,NJ)=0

```

```

C      LOOP TO BUILD ALL JMPS
      IEND=NLP-1
      DO 820 I=4,IEND
      CALL NEWINS
      CALL GENERT(15)
      CALL ENLAB(ILP(1,I))
820    CONTINUE
      GO TO 9999

C      IF NAME FOLLOWING IF IS
C      A SPECIAL NAME, I.E. MFAIL,BADDAT,ENDTRK
C      GENERATE JMP INST
C
82    CALL NAMIS3(ILP(1,IST),ICOD)
      IF(ICOD-8) 89,86,83
83    IF(ICOD-11) 86,87,89
86    IT=ICOD+8
      GO TO 88
C      TYPE 31 FOR IOFAIL
87    IT=31
88    CALL GENERT(IT)
      GO TO 200
C      BUILD LDA
89    IT=1
      CALL GENERT(IT)
C      SEARCH FOR OPERATOR
90    K=K+1
      INDX=K+IST-1
      IF(K-NLP) 100,9920,9920
100    CALL NAMIS3(ILP(1,INDX),ICOD)
      IF(ICOD-1) 90,150,110
110    IF(ICOD-3) 150,150,90
C      BUILD OPERAND AND PAR INSTRUCTIONS
150    K=K-1
      CALL GENERA(ILP(1,IST),K)
C
C      FORM NEXT INSTRUCTION WITH
C      NO LABEL AND SET END OF TASK CNTRL
C      TO IENDT
C
      CALL NEWINS
C      BUILD SUB
      IT=11
      CALL GENERT(IT)
      IST=INDX+1
      K=NLP-2-INDX
      CALL GENERA(ILP(1,IST),K)
      CALL NEWINS
C      BUILD JMP LOGIC TO TRUE
      IT=ICOD+18
      CALL GENERT(IT)
C      ENTER TRUE LABEL AS OPERAND
200    CALL ENLAB(ILP(1,NLP-1))
C
C      BUILD JMP TO FALSE
C
      CALL NEWINS
220    IT=15
      CALL GENERT(IT)
      CALL ENLAB(ILP(1,NLP))
      GO TO 9999
9910  WRITE(IPRIN,9915)
9915  FORMAT(1X,' COMP ERR. NO. OF CHAR IN LABEL LT 0')
      GO TO 9999
9920  WRITE(IPRIN,9925) INCD
9925  FORMAT(1X,' STATEMENT',I3,' HAS NO LOG OPERATOR')
      GO TO 9999
9999  CONTINUE
      RETURN
      END
      EOF

```

```

C      GENERT V2 L1
      SUBROUTINE GENERT(IT)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP

```

```

C      GENERATE INST. TYPE IT
C
C
C

```

```

      INSTR(2,NJ)=IT
      RETURN
      END
      EOF

```

```

C      GENERA V2 L1
      SUBROUTINE GENERA(MSTAK,N)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP
      DIMENSION MSTAK(7,1)

```

```

C      GENERATE REST OF INSTRUCTION REFERRING TO
C      A VARIABLE ASSUMING THAT THE LABEL AND
C      END OF TASK CNTRL FOR THIS INSTRUCTION
C      HAVE BEEN SET
C
C      ALSO, BUILD ALL PAR INST. TO DEFINE INDICES
C

```

```

      IPRIN=11
C      ENTER VARIABLE NAME IN NAMTAB
      CALL ENAM (MSTAK(1,1))
      IF(N=1) 9910, 9999,100

```

```

C      BUILD PAR INST.
C

```

```

100 DO 200 J=2,N
C      FORM NEW STMNT.
      CALL NEWINS
C      GENERATE PAR
      IT=0
      CALL GENERT(IT)
C      ENTER SUBSCRIPT NAME IN NAMTAB
      CALL ENAM (MSTAK(1,J))
200 CONTINUE
      GO TO 9999
9910 WRITE(IPRIN,9915) N
9915 FORMAT(2X' COMP. ERR. N IN CALL TO GENERA IS',I3)
      GO TO 9999
9999 CONTINUE
      RETURN
      END
      EOF

```

```

C      NEWINS V2 L1
      SUBROUTINE NEWINS
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP

```

```

C      BUILD NEW INSTRUCTION
C      WITH 1. NO LABEL
C      2. END OF TASK CNTRL=IENDT
C

```

```

      NJ=NJ+1
      INSTR(1,NJ)=0
      INSTR(4,NJ)=IENDT
      RETURN
      END
      EOF

```

```

C      ENLAB V2 L2
      SUBROUTINE ENLAB(ISTAK)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP
      DIMENSION ISTAK(1)
      IPRIN=ITSYS(3)
      IF(NNT) 35,35,5
5     N=ISTAK(1)
      DO 10 I=1,NNT
      IF(N-NAMTAB(1,I)) 10, 20, 10
20    DO 30 J=1,N
      IF(ISTAK(J+1)-NAMTAB(J+1,I)) 10, 30, 10
30    CONTINUE
      IF(NAMTAB(8,I).EQ.5.OR.NAMTAB(8,I).EQ.4) GO TO 50
      GO TO 9910
10    CONTINUE
35    NNT=NNT+1
      DO 40 I=1,7
40    NAMTAB(I,NNT)=ISTAK(I)
      I=NNT
      NAMTAB(8,I)=5
50    INSTR(3,NJ)=I
      GO TO 9999
9910  WRITE(IPRIN,9915)
9915  FORMAT(' LABEL NAME USED AS VARIABLE')
9999  RETURN
      END
      EOF

```

```

C      PUTLAB V2 L2
      SUBROUTINE PUTLAB(ISTAK)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X     INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X     NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP
      DIMENSION ISTAK(1)
      IPRIN=11
      IF(NNT) 75,75,5
5     N=ISTAK(1)
      DO 70 I=1,NNT
      IF(N-NAMTAB(1,I)) 70, 10, 70
10    DO 20 J=1,N
      IF(ISTAK(J+1)-NAMTAB(J+1,I))70, 20, 70
20    CONTINUE
      IF(NAMTAB(8,I)-5) 30, 90, 30
30    IF(NAMTAB(8,I)-4) 9910, 9920, 9910
70    CONTINUE
75    NNT=NNT+1
      DO 80 I=1,7
80    NAMTAB(I,NNT)=ISTAK(I)
      I=NNT
90    NAMTAB(8,I)=4
      NAMTAB(9,I)=0
      INSTR(1,NJ)=I
      GO TO 9999
9910  WRITE(IPRIN,9915)
9915  FORMAT(' LABEL NAME ALREADY USED AS VAR NAME ')
      GO TO 9999
9920  WRITE(IPRIN,9925)
9925  FORMAT(' DUPLICATE LABEL')
9999  RETURN
      END
      EOF

```

```

C      ENAM V2 L2
      SUBROUTINE ENAM(ISTAK)
      DIMENSION IRD(80), ICC(50), ILP(7,50), IRP(7,50), NAMTAB(9,300),
X INSTR(5,600), NAMJOB(7,50), NAMEXT(7,50), ITSYS(10), IDOSTK(50)
      COMMON IRD, ICC, ILP, IRP, NLP, NRP, NAMTAB, INSTR, NNT, NJ, INCD,
X NAMJOB, NNAM, IENDT, NAMEXT, NEXTR, ITSYS, IDOSTK, NIDO, NDOP
      DIMENSION ISTAK(1)
      IPRIN=11
      IF(NNT) 95,95,5
5      N=ISTAK(1)
      DO 90 I=1,NNT
      IF(N-NAMTAB(1,I)) 90,20,90
20      DO 30 J=1,N
      IF(ISTAK(J+1)-NAMTAB(J+1,I)) 90, 30, 90
30      CONTINUE
      IF(NAMTAB(8, I)-4) 40, 9910, 40
40      GO TO 110
90      CONTINUE
95      NNT=NNT+1
      DO 100 I=1,7
100     NAMTAB(I,NNT)=ISTAK(I)
      NAMTAB(8,NNT)=3
      NAMTAB(9,NNT)=1
      I=NNT
110     INSTR(3,NJ)=I
      GO TO 9999
9910    WRITE(IPRIN,9915)
9915    FORMAT(' VARIABLE NAME IS A STATEMENT LABEL')
9999    RETURN
      END
      EOF

```

```

C      CODE V2 L1
      SUBROUTINE CODE
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS, IDOSTK,NIDO,NDOF
      DIMENSION IC(20),KCD(80),IOP(3),KSTAK(10)

C      FORM ALL ASSIST INSTRUCTIONS FROM
C      INSTRUCTION TABLE
C
      IERR=ITSYS(3)
      IPUN=ITSYS(4)
      IC(1)=1H
      IC(2)=1H*
      IC(3)=1H=
      IC(4)=1H1
      IC(5)=1H2
      IC(6)=1H3
      IC(7)=1HE
      IC(8)=1HN
      IC(9)=1HT
      IC(10)=1HR
      IC(11)=1HY
      IC(12)=1HE
      IC(13)=1HX
      IC(14)=1HT

C      GENER. ENTRY STMENTS
      DO 65 K=1,NNAM
      DO 40 I=1,80
40    KCD(I)=IC(I)
      N=NAMJOB(1,K)
      DO 50 I=1,5
50    KCD(I+6)=IC(I+6)
      DO 60 I=1,N
60    KCD(I+12)=NAMJOB(I+1,K)
      WRITE(IPUN,610) KCD
65    CONTINUE

C      GENER ALL EXTERNALS
C
      IF(NEXTR) 92,92,70
70    DO 90 K=1,NEXTR
      N=NAMEXT(1,K)
      DO 72 I=1,80
72    KCD(I)=IC(I)
      DO 74 I=1,3
74    KCD(I+6)=IC(I+11)
      DO 76 I=1,N
76    KCD(I+12)=NAMEXT(I+1,K)
      WRITE(IPUN,610) KCD
90    CONTINUE

C      GENER ALL EXT FOR INVOKE STMENTS WITH UNDEFINED LABELS
C
92    DO 99 K=1,NJ
      IF(INSTR(2,K)-22) 99,93,99
C      IS LABEL UNDEFINED
93    L=INSTR(3,K)
      ICO=NAMTAB(8,L)
      IF(ICO-5) 99,94,99
94    N=NAMTAB(1,L)
      DO 95 I=1,80
95    KCD(I)=IC(I)
      DO 96 I=1,3
96    KCD(I+6)=IC(I+11)
      DO 97 I=1,N
97    KCD(I+12)=NAMTAB(I+1,L)
      WRITE(IPUN,610) KCD
99    CONTINUE

C      GENER. ALL EQUATES
100 DO 200 K=1,NNT
C      IS THIS NAME A COMMON VAR.
      KTYP=NAMTAB(8,K)
      IF(KTYP-2) 110,110,200

C      YES IT IS,SO CONVERT COMMON
C      CODE NUMBER TO A1 FMT.

```

```

C
110 INTEG=NAMTAB(9,K)
    CALL ICNVT2(KSTAK,INTEG)
    N=KSTAK(1)
    IF(N) 9910,9910,120
120 DO 130 I=1,80
    KCD(I)=IC(1)
130 CONTINUE
    NN=NAMTAB(1,K)
    DO 140 I=1,NN
    KCD(I)=NAMTAB(I+1,K)
140 CONTINUE
    KCD(7)=1HE
    KCD(8)=1HQ
    KCD(9)=1HU
    DO 150 I=1,N
    KCD(I+12)=KSTAK(I+1)
150 CONTINUE
    WRITE(IPUN,160) KCD
160 FORMAT(80A1)
200 CONTINUE
C
                                NOW START BUILDING INSTRUCTIONS
290 DO 1000 KK=1,NJ
    DO 300 I=1,80
    KCD(I)=IC(1)
300 CONTINUE
C
                                ANY LABEL ON THIS STATEMENT
    J=INSTR(1,KK)
    IF(J) 350,350,310
310 N=NAMTAB(1,J)
    DO 320 I=1,N
    KCD(I)=NAMTAB(I+1,J)
320 CONTINUE
C
                                FORM 3 CHAR ASSIST OP CODE
C
350 CALL OPCODE(INSTR(2,KK),IOP)
    DO 370 I=1,3
    KCD(I+6)=IOP(I)
370 CONTINUE
C
                                SET INDIRECT BIT IF END OF TASK
C
                                CONTROL=NEXT
    IF(INSTR(4,KK)) 390,390,380
380 KCD(10)=IC(2)
C
                                IS THERE ANY OPERAND
390 J=INSTR(3,KK)
    IF(J) 395,395,400
395 KCD(13)=IC(2)
    KCD(14)=IC(2)
    GO TO 600
C
                                CHECK FOR SPECIAL INSTRUCTIONS
C
                                CALL SUBROUTINE
C
                                INVOKE
C
                                CALL LINK
C
                                TRAP INTERRUPT
400 ITP=INSTR(2,KK)
    IF(ITP-24) 403,401,403
401 N=NAMEXT(1,J)
    DO 402 I=1,N
    KCD(I+12)=NAMEXT(I+1,J)
402 GO TO 600
C
    IS IT INVOKE
403 IF(ITP-22) 404,403C,404
C
    IS LABEL UNDEFINED
4030 L=INSTR(3,KK)
    ICO=NAMTAB(8,L)
    IF(ICO-5) 408,4032,408
4032 N=NAMTAB(1,L)
    DO 4034 I=1,N
    KCD(I+12)=NAMTAB(I+1,L)
4034 GO TO 600
C
    TRAP INTERRUPT
404 IF((ITP.EQ.25).OR.(ITP.EQ.20)) GO TO 405
    GO TO 408

```

```

405  CALL ICNVT2(KSTAK,INSTR(3,KK))
      N=KSTAK(1)
      DO 406 I=1,N
406  KCD(I+12)=KSTAK(I+1)
      GO TO 600
408  KTOP=NAMTAB(8,J)
      IF(KTOP-5) 440,410,440
C      ERROR UNDEFINED LABEL
C      CHECK FOR OPERAND LABEL BEGINNING WITH *
410  IF(NAMTAB(2,J)-IC(2))415,412,415
412  NAMTAB(8,J)=4
      GO TO 500
415  N=NAMTAB(1,J)
      WRITE(IERR,420) (NAMTAB(I+1,J),I=1,N)
420  FORMAT(2X,' ERROR',6A1,' IS AN UNDEFINED LABEL')
      GO TO 1000
440  IF((KTOP.EQ.3).OR.(KTOP.EQ.6))GO TO 450
      GO TO 500
C
C      OPERAND NAME IS LOCAL
C      NOW CHECK TO SEE IF IT IS AN INTEGER
C
450  CALL ICNVT1(NAMTAB(1,J),INT)
      IF(INT) 500,460,460
460  KCD(13)=IC(3)
C      CHANGE OPERAND NAME CLASS
      NAMTAB(8,J)=6
      KST=13
      GO TO 520
500  KST=12
C      NOW PACK OPERAND NAME
520  N=NAMTAB(1,J)
      DO 540 I=1,N
      KST=KST+1
      KCD(KST)=NAMTAB(I+1,J)
540  CONTINUE
C      CHECK FOR NUMERIC INCREMENTS IN OPERAND
      IF(INSTR(5,KK))542,559,542
542  CALL ICNVT2(KSTAK,INSTR(5,KK))
      KST=KST+1
      KCD(KST)=1H+
      N=KSTAK(1)
      DO 544 I=1,N
      KST=KST+1
544  KCD(KST)=KSTAK(I+1)
      GO TO 600
C      IF OPERAND NAME IS A COMMON VARIABLE
C      OR LOCAL SET INDEX PART ACCORDINGLY
559  IF(KTOP-6) 5590,590,600
5590 IF(KTOP-3) 560,560,600
560  IF(KTOP) 600,600,570
570  IF(KTOP-2) 580,585,590
580  IN=IC(4)
      GO TO 595
585  IN=IC(5)
      GO TO 595
590  IN=IC(6)
595  KCD(KST+1)=1H,
      KCD(KST+2)=IN
C      WRITE OUT ASSIST STMT
600  WRITE(IPUN,610) KCD

```



```

        610 FORMAT(80A1)
    1000 CONTINUE
C
C          DONE WITH ALL CODE
C          FORM ALL NECESSARY
C          RESERVE STORAGE STATEMENTS
C
        DO 1200 J=1,NNT
            KTOP=NAMTAB(8,J)
            IF(KTOP-3) 1200,1050,1200
    1050 DO 1060 I=1,89
            KCD(I)=IC(I)
    1060 CONTINUE
            N=NAMTAB(1,J)
            DO 1070 I=1,N
            KCD(I)=NAMTAB(I+1,J)
    1070 CONTINUE
            KCD(7)=1HR
            KCD(8)=1HE
            KCD(9)=1HS
C          CONVERT NO OF STORAGE WORDS TO A1 FMT
            CALL ICNVT2(KSTAK,NAMTAB(9,J))
            N=KSTAK(1)
            DO 1090 I=1,N
    1090 KCD(I+12)=KSTAK(I+1)
            WRITE(IPUN,610) KCD
    1200 CONTINUE
C          GENERATE END,EOF AND EOJ/
C
C
        WRITE(IPUN,1250)
    1250 FORMAT(6X,3HEND)
        GO TO 9999
    9910 WRITE(IERR,9915)
    9915 FORMAT(2X,' COMP ERR IN CODE, A COMMON CODE'/
X 2X,' NUMBER IS NOT AN INTEGER')
    9999 RETURN
        END
        EOF

```

```

C      OPCOD V2 L1
      SUBROUTINE OPCOD(IT,IOP)
      DIMENSION IOP(1),M(3,32)
      DATA M(1,1),M(2,1),M(3,1),
X      M(1,2),M(2,2),M(3,2),
X      M(1,3),M(2,3),M(3,3),
X      M(1,4),M(2,4),M(3,4),
X      M(1,5),M(2,5),M(3,5),
X      M(1,6),M(2,6),M(3,6),
X      M(1,7),M(2,7),M(3,7),
X      M(1,8),M(2,8),M(3,8),
X      M(1,9),M(2,9),M(3,9),
X      M(1,10),M(2,10),M(3,10),
X      M(1,11),M(2,11),M(3,11),
X      M(1,12),M(2,12),M(3,12),
X      M(1,13),M(2,13),M(3,13),
X      M(1,14),M(2,14),M(3,14),
X      M(1,15),M(2,15),M(3,15),
X      M(1,16),M(2,16),M(3,16),
X      M(1,17),M(2,17),M(3,17),
X      M(1,18),M(2,18),M(3,18),
X      M(1,19),M(2,19),M(3,19),
X      M(1,20),M(2,20),M(3,20),
X      M(1,21),M(2,21),M(3,21),
X      M(1,22),M(2,22),M(3,22),
X      M(1,23),M(2,23),M(3,23),
X      M(1,24),M(2,24),M(3,24),
X      M(1,25),M(2,25),M(3,25),
X      M(1,26),M(2,26),M(3,26),
X      M(1,27),M(2,27),M(3,27),
X      M(1,28),M(2,28),M(3,28),
X      M(1,29),M(2,29),M(3,29),
X      M(1,30),M(2,30),M(3,30),
X      M(1,31),M(2,31),M(3,31),
X      M(1,32),M(2,32),M(3,32) /
X      1HP,1HA,1HR,
X      1HL,1HD,1HA,
X      1HL,1HD,1HE,
X      1HO,1H3,1H ,
X      1HS,1HT,1HA,
X      1HS,1HT,1HE,
X      1HO,1H6,1H ,
X      1HO,1H7,1H ,
X      1H1,1H0,1H ,
X      1HA,1HD,1HO,
X      1HM,1HP,1HY,
X      1HS,1HU,1HB,
X      1HD,1HV,1HD,
X      1H1,1H5,1H ,
X      1H1,1H6,1H ,
X      1HJ,1HM,1HP,
X      1H2,1H2,1H ,
X      1H2,1H3,1H ,
X      1H2,1H4,1H ,
X      1HJ,1HL,1HZ,
X      1HJ,1HG,1HZ,
X      1HJ,1HE,1HZ,
X      1H3,1H3,1H ,
X      1H3,1H4,1H ,
X      1H3,1H5,1H ,
X      1H3,1H6,1H ,
X      1H6,1H0,1H ,
X      1H6,1H1,1H ,
X      1H6,1H2,1H ,
X      1H6,1H3,1H ,
X      1H7,1H7,1H ,
X      1H2,1H5,1H /
      DO 20 I=1,3
      IOP(I)=M(I,IT+1)
20 CONTINUE
      RETURN
      END
      EOF

```

```

C      NAMIS2 V2 L1
      SUBROUTINE NAMIS2(NC,ICOD)
      DIMENSION NC(1)
      DIMENSION IC(15)
      IC(1)=1HB
      IC(2)=1HM
      IC(3)=1HX
      IC(4)=1HY
      IC(5)=1H1
      IC(6)=1H2
      IC(7)=1H3
      IC(8)=1H+
      IC(9)=1H*
      IC(10)=1H-
      IC(11)=1H/
      IF(NC(1)-1) 40, 10, 40
10    DO 20 I=1,4
      IF(NC(2)-IC(I+7)) 20, 30, 20
20    CONTINUE
25    ICOD=0
      RETURN
30    ICOD=I+3
      RETURN
40    IF(NC(1)-3) 25, 50, 25
50    IF(NC(2)-IC(1)) 100, 60, 100
60    IF(NC(3)-IC(1)) 25, 70, 25
70    DO 90 I=1,3
      IF(NC(4)-IC(I+4)) 90, 80, 90
90    CONTINUE
      GO TO 25
80    ICOD=I
      RETURN
100   IF(NC(2).EQ.IC(2).AND.NC(3).EQ.IC(2)) GO TO 110
      GO TO 25
110   IF(NC(4)-IC(3)) 130, 120, 130
120   ICOD=8
      RETURN
130   IF(NC(4)-IC(4)) 25, 140, 25
140   ICOD=9
      RETURN
      END
      EOF

```

```

C      NAMIS3 V2 L1
      SUBROUTINE NAMIS3(ISTAK,ICOD)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS ,IDOSTK,NIDO,NDOP
      DIMENSION ISTAK(1), M(7,11)

C      DETERMINE WHAT NAME IS CONTAINED IN
C      ISTAK AND SET ICD BY THE CONVENTION
C
C      0  OTHER.
C      1  LT
C      2  GT
C      3  EQ
C      4  ACK
C      5  ENT
C      6  LINK
C      7  NEXT
C      8  MFAIL
C      9  BADDAT
C      10 ENDTRK
C      11 IOFAIL

C      BUILD TEST NAME MATRIX
      DO 20 I=1,3
      M(1,I)=2
20  CONTINUE
      M(2,1)=ICC(12)
      M(3,1)=ICC(20)
      M(2,2)=ICC(7)
      M(3,2)=ICC(20)
      M(2,3)=ICC(5)
      M(3,3)=ICC(17)
      DO 40 I=4,5
      M(1,I)=3
40  CONTINUE
      M(2,4)=ICC(1)
      M(3,4)=ICC(3)
      M(4,4)=ICC(11)
C      ENT
      M(2,5)=ICC(5)
      M(3,5)=ICC(14)
      M(4,5)=ICC(20)
      DO 60 I=6,7
      M(1,I)=4
60  CONTINUE
C      LINK
      M(2,6)=ICC(12)
      M(3,6)=ICC(9)
      M(4,6)=ICC(14)
      M(5,6)=ICC(11)
C      NEXT
      M(2,7)=ICC(14)
      M(3,7)=ICC(5)
      M(4,7)=ICC(24)
      M(5,7)=ICC(20)
      M(1,8)=5

```

```

C      M(2,8)=ICC(13)      MFAIL
      M(3,8)=ICC(6)
      M(4,8)=ICC(1)
      M(5,8)=ICC(9)
      M(6,8)=ICC(12)
      DO 80 I=9,11
      M(1,I)=6
80    CONTINUE
C      M(2,9)=ICC(2)      BADDAT
      M(3,9)=ICC(1)
      M(4,9)=ICC(4)
      M(5,9)=ICC(4)
      M(6,9)=ICC(1)
      M(7,9)=ICC(20)
C      M(2,10)=ICC(5)      ENDTRK
      M(3,10)=ICC(14)
      M(4,10)=ICC(4)
      M(5,10)=ICC(20)
      M(6,10)=ICC(18)
      M(7,10)=ICC(11)
C      M(2,11)=1HI      IOFAIL
      M(3,11)=1HO
      M(4,11)=1HF
      M(5,11)=1HA
      M(6,11)=1HI
      M(7,11)=1HL
C      N=ISTAK(1)      BEGIN SEARCH
      DO 300 I=1,11
      IF(N-M(1,I)) 300,200,300
200   DO 250 J=1,N
      IF(ISTAK(J+1)-M(J+1,I)) 300,250,300
250   CONTINUE
      GO TO 400
300   CONTINUE
C      ICOD=0      NO MATCH FOUND
      GO TO 9999
C      400 ICOD=I      MATCH WAS FOUND
9999  RETURN
      END
      EOF

```

```

C      NAMIS1 V2 L1
SUBROUTINE NAMIS1(ISTAK,ICOD)
DIMENSION ISTAK(1),M(7,16)
DATA
X      M(1,1),M(2,1),M(3,1),M(4,1),M(5,1),M(6,1),M(7,1),
X      M(1,2),M(2,2),M(3,2),M(4,2),M(5,2),M(6,2),M(7,2),
X      M(1,3),M(2,3),M(3,3),M(4,3),M(5,3),
X      M(1,4),M(2,4),M(3,4),M(4,4),M(5,4),
X      M(1,5),M(2,5),M(3,5),M(4,5),M(5,5),M(6,5),
X      M(1,6),M(2,6),M(3,6),M(4,6),M(5,6),M(6,6),
X      M(1,7),M(2,7),M(3,7),M(4,7),M(5,7),M(6,7),M(7,7),
X      M(1,8),M(2,8),M(3,8),
X      M(1,9),M(2,9),M(3,9),
X      M(1,10),M(2,10),M(3,10),M(4,10),M(5,10),
X      M(1,11),M(2,11),M(3,11),M(4,11),M(5,11),M(6,11),M(7,11),
X      M(1,12),M(2,12),M(3,12),M(4,12),M(5,12),M(6,12),
X      M(1,13),M(2,13),M(3,13),M(4,13),M(5,13),M(6,13),M(7,13),
X      M(1,14),M(2,14),M(3,14),
X      M(1,15),M(2,15),M(3,15),M(4,15),M(5,15),M(6,15),M(7,15),
X      M(1,16),M(2,16),M(3,16),M(4,16),M(5,16),M(6,16),M(7,16)/
X      6,1HI,1HN,1HV,1HO,1HK,1HE,
X      6,1HR,1HE,1HT,1HU,1HR,1HN,
X      4,1HC,1HA,1HL,1HL,
X      4,1HL,1HI,1HN,1HK,
X      5,1HI,1HN,1HO,1HA,1HT,
X      5,1HL,1HI,1HG,1HH,1HT,
X      6,1HO,1HI,1HS,1HP,1HL,1HA,
X      2,1HI,1HF,
X      2,1HG,1HO,
X      4,1HB,1HI,1HN,1HA,
X      6,1HP,1HR,1HO,1HC,1HE,1HE,
X      5,1HE,1HN,1HT,1HR,1HY,
X      6,1HE,1HQ,1HU,1HI,1HV,1HA,
X      2,1HO,1HO,
X      6,1HC,1HO,1HN,1HT,1HI,1HN,
X      6,1HO,1HI,1HM,1HE,1HN,1HS/

C      DETERMINE WHAT NAME IS CONTAINED IN ISTAK
C      AND SET ICODE BY THE CONVENTION
C
C      0      OTHER
C      1      INVOKE
C      2      RETURN
C      3      CALL
C      4      LINK
C      5      INDAT
C      6      LIGHT
C      7      DISPLA
C      8      IF
C      9      GO
C      10     BINA
C      11     PROCEE
C      12     ENTRY
C      13     EQUIVA
C      14     DO
C      15     CONTIN
C      16     DIMENS

C      N=ISTAK(1)
C      DO 100 I=1,16
C      IF(N-M(1,I)) 100,40,100
C 40 DO 60 J=1,N
C      IF(ISTAK(J+1)-M(J+1,I)) 100,60,100
C 60 CONTINUE
C      ICODE=I
C      GO TO 200
C 100 CONTINUE
C      ICODE=0
C 200 RETURN
C      END
C      EOF

```

```

C      ICNVT1 V2 L1
      SUBROUTINE ICNVT1(ICHAR,INTEG)
      DIMENSION IRD(80),ICC(50),ILP(7,50),IRP(7,50),NAMTAB(9,300),
X INSTR(5,600),NAMJOB(7,50),NAMEXT(7,50),ITSYS(10),IDOSTK(50)
      COMMON IRD,ICC,ILP,IRP,NLP,NRP,NAMTAB,INSTR,NNT,NJ,INCD,
X NAMJOB,NNAM,IENDT,NAMEXT,NEXTR,ITSYS, IDOSTK,NIDD,NDOP
      DIMENSION ICHAR(1),INUM(10)

C      CONVERT THE CHAR STRING IN ICHAR
C      TO THE INTEGER EQUIVALENT INTEG
C
      IPRIN=11
      INUM(1)=1H0
      INUM(2)=1H1
      INUM(3)=1H2
      INUM(4)=1H3
      INUM(5)=1H4
      INUM(6)=1H5
      INUM(7)=1H6
      INUM(8)=1H7
      INUM(9)=1H8
      INUM(10)=1H9

C      N=ICCHAR(1)
      IF(N) 9910,9910,100
100  INTEG=0
      DO 200 I=1,N
      IEXP=N-I
      IAL=ICCHAR(I+1)
      DO 150 K=1,10
      IF(IAL-INUM(K)) 150,170,150
150  CONTINUE
C      ERROR,NO INTEGER FOUND
      INTEG=-1
      GO TO 9999
170  K=K-1
      INTEG=INTEG+K*10**IEXP
200  CONTINUE
      GO TO 9999
9910 WRITE(IPRIN,9915) N
9915 FORMAT(2X,' COMP ERR NO. OF CHAR INPUT IN STRING TO'/
X      2X,' ICNVT1 IS',I3)
9999 CONTINUE
      RETURN
      END
      EOF

```

```

C      ICNVT2 V2 L1
SUBROUTINE ICNVT2(ICHAR,INTEG)
DIMENSION ICHAR(1),IC(10)
IC(1)=1H0
IC(2)=1H1
IC(3)=1H2
IC(4)=1H3
IC(5)=1H4
IC(6)=1H5
IC(7)=1H6
IC(8)=1H7
IC(9)=1H8
IC(10)=1H9
ISW=-1
IVAL=INTEG
L=0
ON 100 I=1,6
K=6-I
IDIV=10**K
NUM=IVAL/IDIV
IF(NUM) 80,80,50
50  ISW=+1
60  L=L+1
    ICHAR(L+1)=IC(NUM+1)
    IVAL=IVAL-NUM*IDIV
    GO TO 100
80  IF(ISW) 100,100,90
90  NUM=0
    GO TO 60
100 CONTINUE
    ICHAR(1)=L
999 CONTINUE
    RETURN
    END
    EOF

```


APPENDIX II. FLOWCHARTS OF KERTRAN COMPILING ROUTINES

MAIN

KERTRAN COMPILING ROUTINES
MAIN

012

```

1 INVOKE
2 RETURN
3 CALL LINK
4 INDT
5 LIGHT
6 DISPLAY MSGG

```

KERTRAN COMPILING ROUTINES
EQUIV D

KERTRAN COMPILING ROUTINES
STAKEP.

KERTRAN COMPILING ROUTINES
BINAV

KERTRAN COMPILING ROUTINES
TINT

KERTRAN COMPILING ROUTINES
ARITH

KERTRAN COMPILING ROUTINES
COMINS

KERTKAN COMPILING ROUTINES
LOGIC

KERTRAN COMPILING ROUTINES
ENLAB

KERTRAN COMPILING ROUTINES
PUTLAB

KERTRAN COMPILING ROUTINES
ENAM

KERTRAN COMPILING ROUTINES
NAMIS2

APPENDIX III

TABLE 1

TRANSLATOR COMMON

Variable	Description
IRD(80)	Input source statement in A1 format
ICC(50)	A1 character array used by the system
ILP(7,50)	Left part stack matrix
IRP(7,50)	Right part stack matrix
NLP	Number of entries in ILP
NRP	Number of entries in IRP
NAMTAB(9,100)	Label and variable name table
INSTR(5,200)	Generated instruction table
NNT	Number of entries in NAMTAB
NJ	Number of entries in INSTR
INCD	Sequence number of source input card
NAMJOB(7,4)	Job names for this program
NNAM	Number of job names used
IENDT	Current end of task control from source card
NAMEXT(7,50)	Name table for subroutines called
NEXTR	Number of entries in NAMEXT
ITSYS(10)	Control variables 1 = Input device 2 = Source print device 3 = Error message print device 4 = Output code device

TABLE 1 (continued)

Variable	Description
IDØSTK(50)	Stack for DØ statement generated labels
NIDØ	Number of entries in IDØSTK
NDØP	Number of DØ statements processed
	<p data-bbox="733 688 1174 821">It is also used to generate internal labels of the form)DØ00NDØP</p>

TABLE 2

INSTR(1,J) For the Jth instruction:

INSTR(1,J)	Pointer to NAMTAB for the label (= 0 $\Rightarrow$ no label)
(2,J)	Instruction type
(3,J)	Pointer to NAMTAB for operand ^a (= 0 $\Rightarrow$ no operand)
(4,J)	End of task control, 1 = NEXT, 0 = TASK
(5,J)	Number to add to operand label, i.e., JMP ALPH + 2

TABLE 3

NAMTAB(1,J) For the Jth name:

NAMTAB(1,J)	Number of characters in the name
(2-7,J)	All characters of the name
(8,J)	Name class <ul style="list-style-type: none"> 0 General common 1 IICOM variable 2 IDEVNT variable 3 Local to TAWTA 4 Label 5 Label, not defined 6 Type 3, no reserve storage necessary
(9,J)	Additional code numbers for name <ul style="list-style-type: none"> For name class 0,1,2 this is common code number for equivalence statement For name class 3, the name is a dimensional variable and this is the dimension limit

^a There are two exceptions to this. For trapping interrupts and CALL LINK this is the number to use as the operand. For subroutine calls it is a pointer to NAMEXT for name of the called subroutine.

TABLE 4

KUBER INSTRUCTION SET

TYPE	OP CODE	MNEMONIC	DESCRIPTION
1	01	LDA	$A \leftarrow M$
2	02	LDE	$E \leftarrow M$
3	03		$A \leftarrow -M$
4	04	STA	$M \leftarrow A$
5	05	STE	$M \leftarrow E$
6	06		$A \leftarrow \text{Col. 1 of BUTTON BOARD}$
7	07		$A \leftarrow \text{Col. 2 of BUTTON BOARD}$
8	10		$A \leftarrow \text{Col. 3 of BUTTON BOARD}$
9	11	ADD	$A \leftarrow A + M$
10	12	MPY	$A \leftarrow A \times M$
11	13	SUB	$A \leftarrow A - M$
12	14	DVD	$A \leftarrow A/M$
13	15		$A \leftarrow x \text{ ENCODER SETTING}$
14	16		$A \leftarrow y \text{ ENCODER SETTING}$
15	21	JMP	UNCONDITIONAL JUMP
16	22		JUMP IF MEMORY FAIL
17~	23		JUMP IF BAD DATEX CODE
18	24		JUMP IF END OF TRACK
31	25		JUMP IF I/O PARITY FAIL
19	30	JLZ	JUMP IF $A < 0$
20	31	JGZ	JUMP IF $A > 0$
21	32	JEZ	JUMP IF $A = 0$

TABLE 4 (continued)

TYPE	OP CODE	MNEMONIC	DESCRIPTION
22	33		INVOKE
23	34		RETURN
24	35		CALL SUBROUTINE
25	36		CALL LINK
26	60		READ DATA FROM MEASURING MACHINE
27	61		LIGHT VIEW
28	62		DISPLAY MESSAGE
29	63		DISPLAY BINAVIEW
30	77		SET INTERRUPT CONTROLS, FORM OF ADDRESS PART IS IGNORE, TASK = 10_8 TASK, IGNORE = 1_8

INDEX

KERTRAN COMPILING SUBROUTINES

	Page
ARITH	14
BINAV	13
CODE	17
CØMINS	15
DOER	14
ENAM	17
ENLAB	17
EQUIVD	13
GENERA	16
GENERT	16
ICNVT1	20
ICNVT2	20
INITA	12
INITB	12
LOGIC	16
MAIN	12
NAMIS1	19
NAMIS2	18
NAMIS3	18
NEWINS	16
OPCODE	18
PUTLAB	17
STAKER	13
TINT	14