AN INTERMEDIATE CODE GENERATOR FOR HP-PASCAL

A DISSERTATION SUBMITTED IN
PARTIAL FULFILMENT OF THE REQUIREMENTS FOR
THE DEGREE OF

MASTER OF PHILOSOPHY

BY

GURMINDER SINGH

112p + Appendix

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI:110 067
1983

0

To

My Parents

DECLARATION

The work embodied in this dissertation contains the results of the research work carried out under the supervision of Professor Dilip K. Banerji, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. The work is original and has not been submitted, in part or full, to any other university for the award of any other degree or diploma.

GURMINDER SINGH
Student

Prof. Dilip K. Banerji
Dean,
School Of Computer And
Systems Sciences,
Jawaharlal Nehru University,
New Delhi:110 067

Prof. Dilip K. Banerji
Supervisor

ACKNOWLEDGEMENTS

I am indebted to my supervisor Prof. Dilip K. Banerji for his guidence and the encouragement provided in carrying out this work.

I acknowledge the invaluable help provided by Mr. Arun K. Bansal in designing some of the algorithms and programs

I express my sincere thanks to Mr. Sunil K. Dhanawat for making available his word processor software for preparing the dissertation. I am also thankful to Mr. Hawa Singh and Mr. Narender K. Bhatia for their encouragement and help.

I am thankful to Mr. Ajoy Roy for the help provided in using the computer.

Finally, I express my gratitude to the JNU for the financial assistance.

Gurminder Singh

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The basic aim of this entire project of which a part is described in this dissertation is to implement a Pascal compiler on the HP 1000/40 computer. The complete project has been divided into two logically independent parts, namely, intermediate code generation, and machine code generation.

This dissertation describes the first part which translates a program written in Pascal to a form which can be used by the second logical part to generate code executable on the HP 1000/40.

1.2 LANGUAGES AND THEIR USES

The natural way of non-verbal communication is through symbols. A set of these symbols along with certain grammatical rules governing their usage forms a language.

In the case of computers, the user communicates with

the machine through symbolic languages. A language which is directly "understood" by a computer is known as its machine language. It differs from a natural language in many ways and communication through this language is terribly tedious and fraught with opportunities for mistakes. The most serious disadvantage of a machine language is that all operations and operands must be specified in a numeric code. Further more, each machine tends to have its own (unique) machine language which makes it necessary for the user to learn different machine languages in order to use different machines.

For these reasons it is desirable to communicate with computers in a user-oriented language. The most immediate step away from machine language is the use of a symbolic assembly language, in which operations and data addresses are replaced by mnemonics or symbols.

Though symbolic assembly programs are easier to write and understand than machine language programs there still are some sever drawbacks with this approach. The programmer must know the details of how a specific computer operates, how the data is represented, the ways of addressing data in memory, and so on. In order to assist the user in avoiding these details high level languages (HLLs) are used for communication with the machine. High level languages are essentially user-oriented which helps the user in specifying the operations without worrying about

the machine details. The advantages of using HLLs can be summarized as follows.

1) It is more expressive than assembly language and, therefore, it is easier to write a program in a high level language.

2) User attention can be focussed more on the bugs in the algorithm and the flaws in design rather than the machine language bugs and tricks.

3) Being user-oriented, the language can be easily learnt by the user.

1.3 NEED OF A TRANSLATOR

Introduction of assembly and high level languages makes the programming task simpler, but it also introduces some problems. The most obvious problem is that we need to translate these into the machine language.

We define a translator as a function whose domain is a source language and range is the target language. Translators can vary in their type of translation (one to one, or one to many mapping), their scheme of translation, or in the type of language they translate. According to

their functions and specific charactristics, the translators are given different names.

1.3.1 Assembler

An assembler is a program which when executed translates source written in an assembly language into the machine language of a computer, on a one-to-one basis. The schematic diagram of an assembler is given in Fig. 1.1.

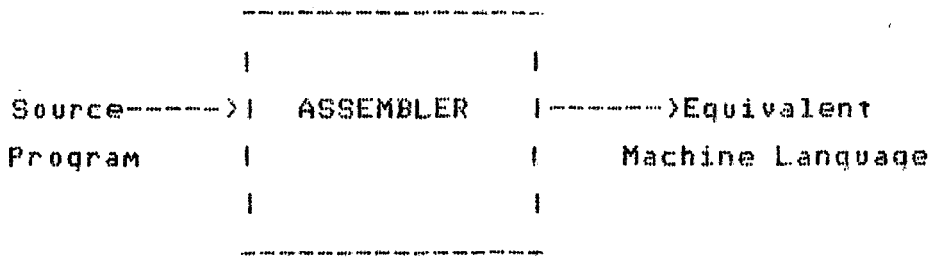


Fig. 1.1 Schematic Diagram of an Assembler

1.3.2 Compiler

If the source language is a high-level language and the target is assembly or machine language of some computer, the translator is called a compiler. The function of a compiler is shown in Fig. 1.2.

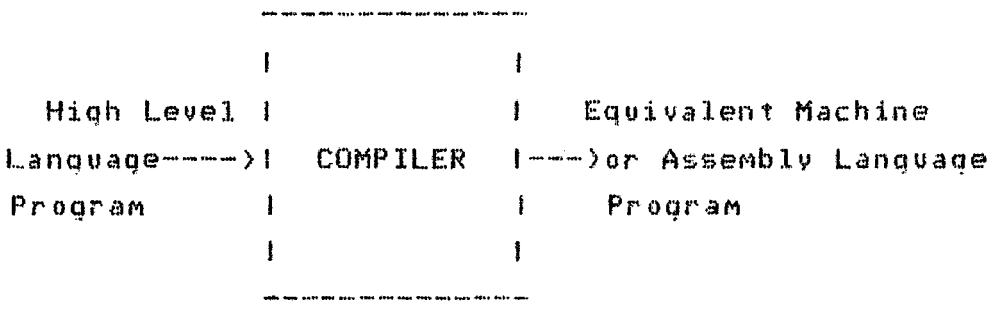


Fig. 1.2 Schematic Diagram of a Compiler

1.3.3 Interpreter

An interpreter for a language accepts a source program written in that language as input and executes it

then and there. The difference between a compiler and an interpreter is that the interpreter does not produce an object program which can be executed repeatedly.

1.4 THE STAGES OF A COMPILER

The process of compilation is a complex process and from the implementation point of view, it is important to divide it into subprocesses called phases. The different phases of a compiler and the various data structures used in these phases are shown in Fig. 1.3. The phases of a compiler can be divided into seven different categories, as follows:

1) Lexical Analysis Phase

2) Syntax Analysis Phase

3) Intermediate Code Generation Phase

4) Intermediate Code Optimization Phase

5) Storage Assignment Phase

6) Code Generation Phase

7) Assembly Phase

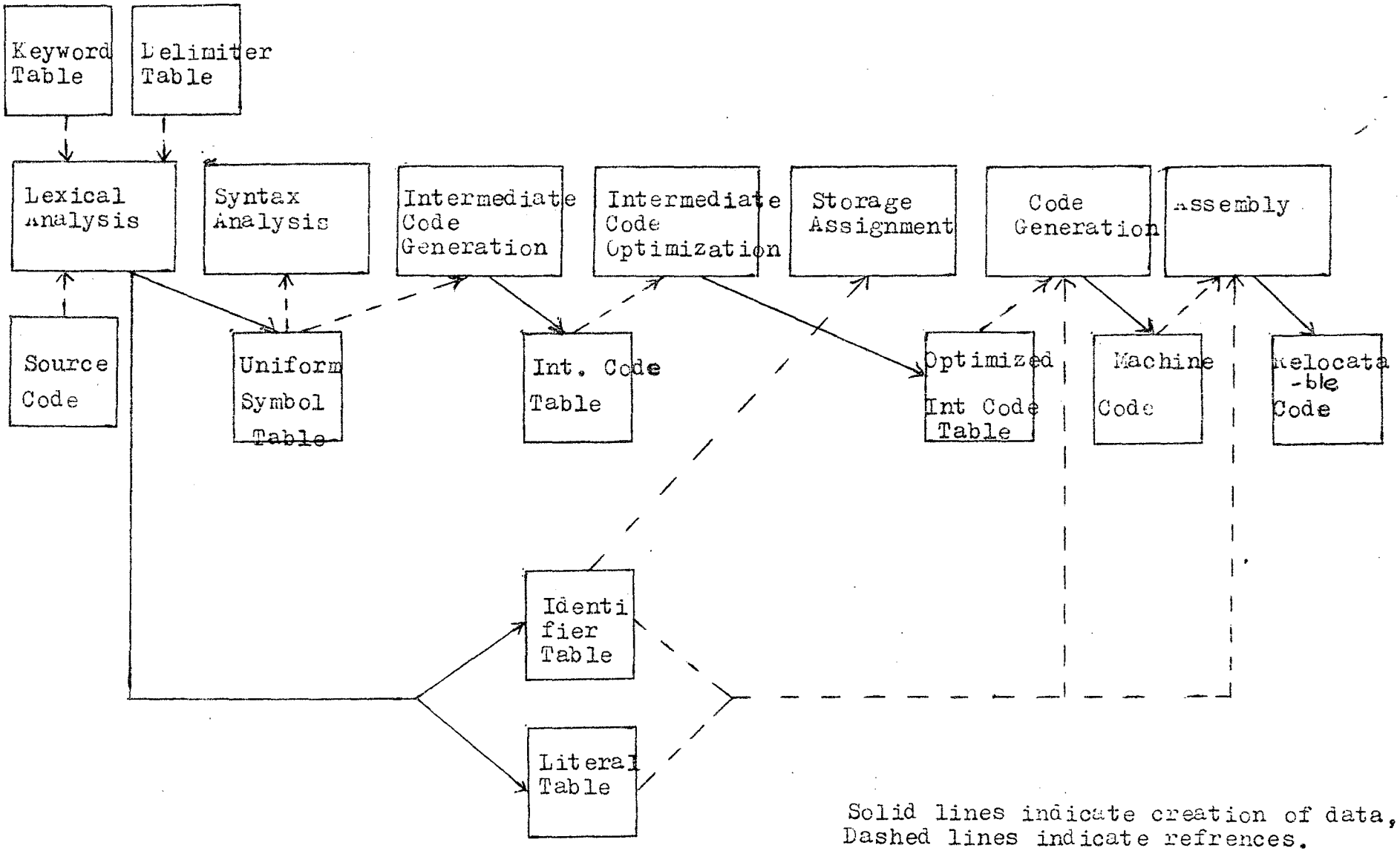


Fig. 1.3 Structure of a Compiler

Phases 1 through 4 are machine-independent and language-dependent , whereas Phases 5 through 7 are machine-dependent and language-independent. A brief description of each of these pahases is given in the following paragraphs.

1.4.1 Lexical Analysis Phase

This phase scans the characters of the input source program from left to right and separates the characters into groups that logically belong together. These groups are called tokens, atoms or symbols. During this phase, certain tables like the identifier table, literal tables, and uniform symbol table are created. This is a machine independent phase.

1.4.2 Syntax Analysis Phase (Parsing)

The input to this phase consists of the symbols generated by the Lexical Analysis Phase. These symbols are then grouped as basic syntactic constructs.

1.4.3 Intermediate Code Generation Phase

This phase generates intermediate code for syntactically correct constructs with the help of various information tables generated during the earlier phases. This code can be in the form of a tree or quadruples or triples depending on how it is to be manipulated later.

1.4.4 Intermediate Code Optimization Phase

This phase optimizes the intermediate code generated by the earlier phase. It is not mandatory for every compiler to carry out the machine independent optimization. The deciding factor for incorporating such a phase is the gain it would bring in increased efficiency against the incease in cost in terms of compilation time and complexity.

1.4.5 Storage Assignment Phase

This phase assigns storage to all variables and literals refrenced in the source program. It also assigns storage to all the temporary locations necessary for storing intermediate results.

1.4.6 Code Generation Phase

The basic function of this phase is to produce the target code (in machine or assembly language). The input to this phase is the intermediate code along with the identifier table and literal table.

1.4.7 Assembly Phase

This phase resolves the label refrences, formats the object code, and generates the appropriate information for the loader.

1.5 THE PRESENTED WORK - SCOPE AND BOUNDRIES

As mentioned earlier, the process of compilation can be split into two logically independent parts,namely

- the machine independent part
- the machine dependent part.

The machine independent part in this implementation consists of the lexical anlysis, the syntax analysis, and the intermediate code generation phase, whereas the intermediate code optimization phase has been omitted to reduce the complexity of implementation. The machine dependent part consists of the storage assignment, code

generation, and the assembly phases. The work presented in this dissertation includes the design and implementation of the machine independent part of the compiler. The schematic diagram of Fig. 1.4 shows the inputs and outputs of the present implementation.

This part of the implementation generates the Identifier Table, Literal Tables, Uniform Symbol Table, Intermediate Code Table, and the Error Table. All these tables may not be completly filled i.e. it leaves some portion of each record to be filled by the latter phases of the compiler. Detailed discussion about the structure of the tables and their information content is presented in Chapter 3.

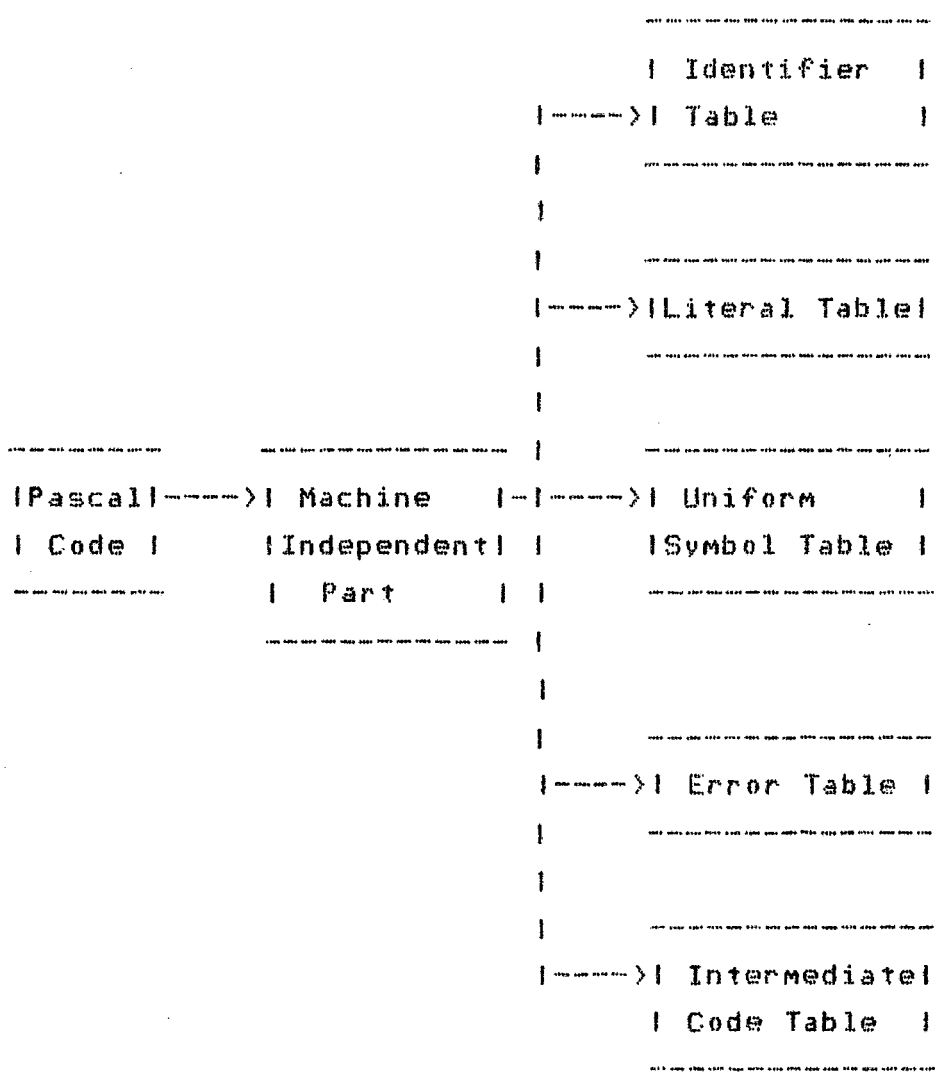


Fig. 1.4 The Boundries of the Presented Work

CHAPTER 2

THE LANGUAGE AND THE SUBSET

In this chapter we discuss why Pascal was chosen to be the language for implementation, The subset chosen and the reasons for selecting this subset.

2.1 THE CHOICE OF PASCAL

The main objective is to select a high level language which is suitable for systems programming. The HP 1000/40 machine presently supports FORTRAN-IV and the HP Assembly language only. Additionally, in an educational environment one does feel the necessity of having a language which is helpful in learning the concepts of structured programming. Keeping these objectives in mind the decision to implement a compiler for Pascal was taken.

2.1.1 Pascal : Some Historical Notes

The Pascal language was created by Professor Niklaus Wirth of Eidgenossische Technische Hechschule (ETH) in Zurich, Switzerland during 1970-1971. The language was so named in the memory of the French mathematician Blais

Pascal. The original work on Pascal began as an outgrowth of attempts by Wirth and Hoare to develop a successor to the language ALGOL-60. The programming language Pascal was designed with the following aims :

1) To make available a notation in which the fundamental concepts and structures of programming are expressible in a systamatic, precise, and appropriate manner.

2) To demonstrate that a language with a rich set of fexible data and program facilities can be implemented by a moderately-sized compiler.

3) To demonstrate that the use of a machine-independent language with felxible data and program structures for the description of a compiler leads to an increase of its readability, verifiability and consequently its reliability, and that this gain need not be offset by any loss in efficiency.

4) To gain more insight into the methods of organising large programs and managing software projects.

2.2 THE RATIONALE FOR A SUBSET

This implementation does not attempt to implement all the features of Pascal. Instead a suitable subset has been selected. This has been done since the implementation of all the features of Pascal would be beyond the scope of an M.Phil. dissertation.

2.3 THE SUBSET

The subset of Pascal selected for implementation is given in Appendix-A.

This subset includes three scalar data types, namely, integer, real, and character, and a structured data type, namely, array. The reason for selecting these data types is that they cater to most of the elementry needs for systems programming as well as general purpose programming and the other data types like boolean and subrange can be easily simulated using these.

In the control statements the subset contains "begin-end", "if-then", "if-then-else", "while-do", and "goto". These allow the basic needs for structured

programming to be met and allow the vast majority of programs to be expressed as hierarchical nesting of single-entry, single-exit blocks. These facilities contribute significantly to program clearity and understanding and it is not absolutely necessary to incorporate facilities like "repeat-until" for the purpose of this dissertation.

CHAPTER 3

THE IMPLEMENTATION

This chapter describes the design and implementation of the machine independent part of the Pascal compiler.

3.1 THE ENVIRONMENT

In this section a brief description of the hardware and software environment available for implementation is presented. The implementation has been caried out on the HP 1000/40 system available in the SCSS.

3.1.1 The Hardware Environment

The HP 1000/40 system is designed around a 16-bit microprogrammed central processing unit. The available system has 128 KB of random access memory as primary memory, and two discs of 10 MB each as secondary memory associated with it. It has one console and one bi-directional matrix printer (Recently the cofiguration has been enhanced.).

3.1.2 The Software Environment

The system has a multiprogramming operating system allowing a variable partition size (maximum partition size being 64 KB) and uesr definable job priorities. The system provides an editor for creating and editing data as well as source files. It also provides an assembler for the HP Assembly Language and a compiler for FORTRAN IV. The relocatable loader allows program overlays , the overlays being decided by the user.

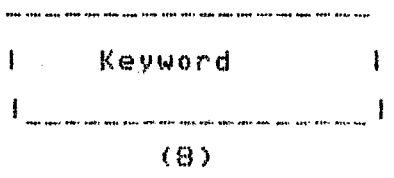
3.2 THE DESIGN

A detailed description of the implementation is presented in this section. This includes a description of the various data structures used , the intermediate code, the lexical analysis, as well as the syntax analysis and the intermediate code generation phases.

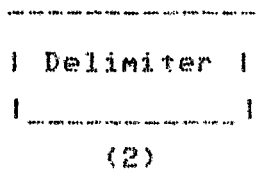
3.2.1 Data Structures

The data structures used in the implementation are described in this section. The sechematic diagrams of all the data structures used are given in Fig. 3.1. In the figure, the relative address field, as discussed in some of the data structures, is not shown because it is filled and

used by the machine dependent part of the compiler.

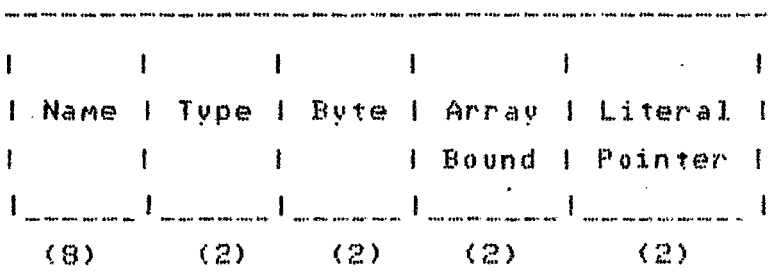


(a) Keyword Table

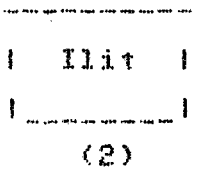


(b) Delimiter Table

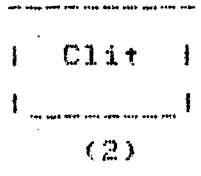
Fig. 3.1 The Data Structures Used (contd.)



(c) Identifier Table



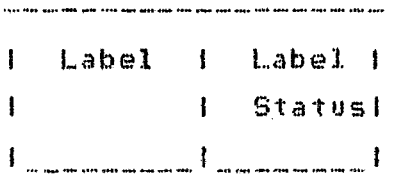
(d) Integer Literal Table



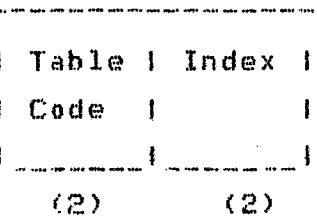
(e) Character Literal Table

Fig. 3.1 The Data Structures Used (contd.)

(f) Real Literal Table



(g) Label Table



(h) Uniform Symbol Table

Fig. 3.1 The Data Structures Used (contd.)

Line	Error Code
(2)	(2)

(4)

(i) Error Message Table

Op code	Table Code1	Index 1	Array Code1	Array Index1	Table Code2	Index 2	Array Code2	Array Index2
(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)

(18)

(j) Intermediate Code Table

Note : The number in the right most paranthesis indicate the total number of bytes for the record, whereas others indicate the number of bytes for the field.

Fig. 3.1 The Data Structures Used

a) Keyword Table

This table is available with the lexical analyser and contains all the keywords used in the language. Each entry in the table is eight bytes long and the table contains 22 such entries. A list of all the keywords stored in this table is given Appendix B.

b) Delimiter Table

This table is also available with the lexical analyser and contains all the delimiters except space. Each entry in the table is two bytes long and the table contains 14 such entries. A list of all the delimiters stored in this table is given in Appendix C.

c) Identifier Table

This table is created by the lexical analyser and consists of all the distinct variables and the program name used in the source program. A typical entry of this table consists of 16 bytes of information.Each entry in this table consists of the following fields:

1) Name : First eight bytes in the entry contain the name of the identifier. The name consists of a maximum of eight alphameric characters and the first character among the eight must be an alphabet. This field is filled up by the lexical analyser.

2) Type : This field occupies the next two bytes and denotes the type of the identifier in the entry. Type 0, 1, 2, 3 and 4 correspond to undeclared identifier, integer, real, character, and program name, respectively.

3) Byte : The number of bytes to be allocated to an identifier is given by this field. The number of bytes allocated to a program name is zero.

4) Array Bound : If the identifier is an array type then this field gives the length of the array otherwise this field contains a zero entry.

5) Literal Pointer : If the identifier has been declared as a constant , then this field provides a pointer to the literal table where the value of the constant is stored, otherwise this field contains a zero. A pointer value begining with the number 21 points to the integer literal

table; that beqining with 22 points to the character literal table, and that beqining with 23 points to the real literal table. The last three digits of a pointer value indicate the entry point within the selected literal table.

6) Ralative Address : This field shall contain the relative address of the identifier within the table.

d) Integer Literal Table

This table is generated by the lexical analyser and contains all the integer literals appearing in the source program. Each entry in the table is four bytes long, and consists of the following fields:

1) Ilit : This field contains the value of the integer literal; the length of this field is two bytes.

2) Relative address : This 2-byte field shall contain the realtive address of the literal within the table.

e) Character Literal Table

This table is generated by the lexical analyser and contains all the character literals appearing in the source program. Each entry in the table is four bytes long. The table consists of the following fields:

1) Clit : This field is two bytes long and contains the characters appearing in the character literals.

2) Relative Address : This field shall contain the relative address of the literal within the table.

f) Real Literal Table

This table is generated by the lexical analyser and contains information about the real literals appearing in the source program. Each entry in the table is six bytes long and consists of the following fields :

1) Rlit : This field is four bytes long and contains the

value of the real literal.

2) Relative Address : This field is two bytes long and shall contain the relative address of the literal within the table.

g) Label Table

This table is filled up by the syntax analyser and contains all the labels declared in the source program. Each entry of the table is four bytes long and contains the following fields :

1) Label : This field is two bytes long and contains the label declared in the program.

2) Label Status : This field is two bytes long and contains the status of the label. Status zero implies that the label has not been defined before in the program whereas status 1 implies that the lable has been defined once in the program.

h) Uniform Symbol Table

This table is generated by the lexical analyser and contains refrences to all the tokens appearing in the source program except the comments which are ignored. Each entry of the table is four bytes long and contains the following fields :

1) Table Code : This field is two bytes long and contains the number of the table in which the token is stored.

2) Index : This field is two bytes long and contains the number of the entry corresponding to the token in the table referred to by the Table Code field.

i) Error Message Table

Entries in this table are made by the lexical as well as the syntax analyser, each entry corresponding to an error detected by the lexical analyser or the syntax analyser. Each entry in the table is four bytes long and contains the following fields :

1) Line : This field is two bytes long and contains the line number of the source program in which the error has been

encountered.

2) Error Code : This field is two bytes long and contains the error code for the error detected in the line given by the Line field.

j) Intermediate Code Table

This table is generated by the syntax analyser and is to be treated as the input for the code generator part of the compiler. It provides the basic interface between the machine independent and machine dependent parts. The intermediate code consists of a triplet - an opcode, operand1, and operand2. Each entry in the table consists of eighteen bytes and contains the following fields :

1) Opcode : This field is two bytes long and contains the numeric code for the operation to be performed on the operands. A list of all the operations available, the corresponding codes, and their actions is given in Appendix D.

2) Table Code 1 : This field is two bytes long and contains the number of the table correspondins to the first operand.

3) Index 1 : Thsi field is two bytes long and contains the number of the entry in the table referred to by the Table Code 1 field.

4) Array Code 1 : This field is two bytes long and contains a 1 if operand1 happens to be an array type else it contains a zero.

5) Array Index 1 : This field is two bytes long and has significance only if the Array Code 1 field contains a 1 in which case it provides the value of the index within the array.

6) Table Code 2 : This field is two bytes long and contains the number of the table corresponding to the second operand.

7) Index 2 : This field is two bytes long and contains the number of the entry in the table referred to by Table Code 2 field of the entry.

8) Array Code 2 : This field is two bytes long and contains a 1 if operand2 happens to be an array type; it contains the number 21 if the Opcode value is 10 to 13 and the operation requires the floating point type format; otherwise this field contains a zero.

9) Array Index 2 : This is a two byte long field and contains the value of the index within the array if Array Code 2 field contains a 1, or the number of the entry in the integer literal table if the Array Code 2 contains twenty one.

3.2.2 Intermediate Code

The intermediate code consists of triplets of one operator and two operands. The code is chosen in such a way that it is easy to translate into the machine language of the HP 1000. A detailed discussion on the code structure has been presented in the earlier paragraphs. The various operators and their corresponding codes are given in Appendix D.

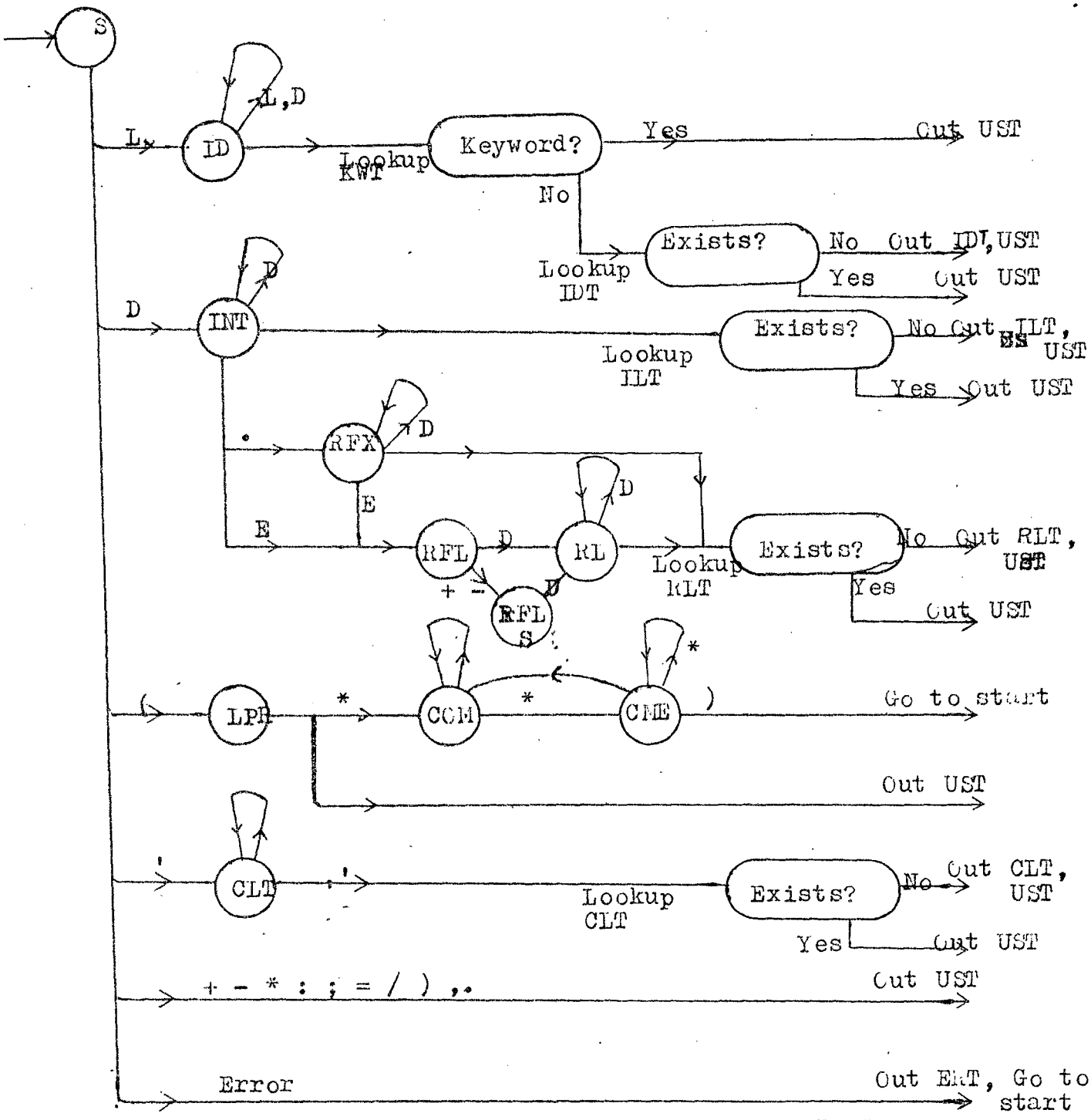
3.2.3 Lexical Analyser

As discussed previously , the function of a lexical

analyser is to read the source program (a Pascal program in our case) one character at a time, and to translate it into tokens or atoms. The lexical analysis phase has been kept seperate from syntax analysis in order to simplify the design of the compiler. In addition, since the structure of tokens can be more easily specified than the syntactic structure of the source language, the construction of a more specialised, and hence more efficient recognizer for the tokens is possible. Since a large portion of the compile time is spent in scanning the input characters, this saperation allows us to concentrate solely on reducing this time. The lexical analyser reads the complete input source program and generates the necessary information for storage in the identifier table, integer literal table, real literal table, character literal table, and the uniform symbol table for the syntax analyser to take over.

The transition diagram of Fig. 3.2 is a particularly useful tool in understanding the working of the lexical analyser because the action taken is highly dependent on what characters have been encountered recently. The circles in the diagram are called states and these are connected by arrows called edges. The lables on the various edges leaving a state indicate the input character that can appear after that state. The starting state of the transition diagram is state S; the first edge from this state indicates that the first input character must be a letter. If this is the case, we enter state ID and look at

the next input character. We continue this way, reading letters and digits, and making transitions from state ID to itself, until a delimiter for an identifier is encountered, at which point we leave the state ID (for practical reasons, the size of an identifier has been limited to a maximum of eight characters). Then a look-up is performed in the keyword table to see if the identifier is a keyword or not. If the identifier happens to be a keyword, then we make an entry in the uniform symbol table; however, if it is not a keyword, then it is entered in the identifier table (if it does not already exist there), and a corresponding refrence entry is made in uniform symbol table. Similarly, the lexical anaylser enters an integer literal in the integer literal table, a real literal (fixed point as well as floating point) in the real literal table, and a character literal in the character literal table; corresponding to each of these entries, a refrence is entered in the uniform symbol table. In the case of a valid delimiter, an entry is made in the uniform symbol table only after a look-up is performed in the delimiter table. It is to be noted that when a comment is detected, no entries are made and the scan is continued to the following symbol. In case of an error, an entry is made in the error message table and the scan is continued.



L : Letter
D : Digit
KWT : Keyword Table
IDT : Identifier Table
ILT : Integer Literal Table
RLT : Real Literal Table
CLT : Character Literal Table
UST : Uniform Symbol Table
ERT : Error Table

Fig. 3.2 Transition Diagram For Lexical Analyser

3.2.4 Syntax Analyser and Intermediate Code Generator

The syntax analyser checks that the tokens appearing in the input (which is the output of the lexical analyser) occur in patterns that are permitted by the specifications of the source language. In other words, a syntax analyser for a particular grammer takes as input a string of tokens and checks whether the string is a part of the grammer or not. If the input conforms to the grammer, then the intermediate code for such input is generated; otherwise, an appropriate error message indicating the violation of the grammer is generated. The intermediate code generator transforms the error free source statements into an intermediate language. The syntax analysis and intermediate code generation may be implemented as two different phases or both may be combined in one phase. Combining the two in one phase is normally preferred for the purpose of increasing the efficiency of the compiler. The same approach has been followed in this implementation. As soon as the syntax analyser is able to recognize any particular statement of the language, the intermediate code is generated.

The syntax analyser and intermediate code generator program reads the uniform symbol table sequentially, checks

the grammatic correctness of the input and depending on the type of the statement, updates the information present in the identifier table and/or generates the intermediate code which is filled into the intermediate code table sequentially. For grammatically incorrect input, the program fills up the appropriate error code and the corresponding line number of the source program in the error message table. The syntax analyser also maintains a table of labels, a stack of temporary variables and a stack of nestings in order to facilitate the process of syntax analysis.

The complete syntax analysis and intermediate code generation phase has ben divided into four major routines, namely, SYTAX, COMPS, ST, and EXPR. The interaction between these routines is depicted in Fig. 3.3. The first routine called SYTAX handles the porgram heading and the declaration part of the program. The second routine named COMPS parses the main-block "begin-end" and maintains the necessary control for it. The routine named ST parses the executable source statements namely the assignment, "if-then", "if-then-else", "while-do", "begin-end", "goto", "readln", "read", "writeln", and "write" statements and generates the corresponding intermediate code for them. The routine EXPR handles the simple expressions appearing in the source statements. The entry to the syntax analyser is made through the routine SYTAX which, after processing the declarations , calls the routine COMPS which inturn calls

the routine ST. Whenever necessary the routine ST calls the routine EXPR to process the simple expressions. A brief description of the four routines is given in the following paragraphs.

On entering the syntax analysis and intermediate code generation phase the routine SYTAX is called. This routine expects the first token in the input to be the keyword "program" followed by a semicolon. If there is any deviation from this input, the routine generates an appropriate error message and, if possible, makes relevant assumptions. For example, if program identifier does not exist, the routine assumes it to be Pascal and continues the analysis further. The routine then generates the appropriate intermediate code. The routine thereafter looks for the declaration part of the source program. If the declaration part exists, then it checks for the presence of label declaration, constant declaration, and variable declarations. On finding the label declarations, the routine checks for the syntactic correctness of the declarations and transfers the valid labels into the label table. For constant declarations, after the syntactic correctness has been established, the pointers to the literal table are updated in the corresponding identifier entry in the identifier table. For variable declarations, the routine first checks for the possibility of a declared identifier being an array type. If the identifier happens to be an array type, the routine then checks for the

validity of the array length, and updates this information in the corresponding identifier entry in the identifier table. The data type of the identifier, whether array type or other, still remains to be established. The data types allowed by the routine are integer, real, and character. This information is also entered in the corresponding identifier entry in the identifier table. For character type identifiers,the length of the identifier is also checked; if not specified, this length is assumed to be one. For integers, this length is assumed to be two bytes, and for reals four bytes. Before updating any information, the routine also checks the semantic correctness of the declarations, points out possible semantic errors which are reported in the error message table, and makes a recovery from the error(s). After the processing of declaration part is over the routine calls COMPS for further analysis and code generation.

The routine COMPS expects the keyword "begin" to be present at the input. On finding the required input, it calls the routine ST to process the subsequent input tokens. If COMPS does not find the keyword "begin" in the input, it assumes the main block to be preesnt, reports the message in the error table, and calls the routine ST.

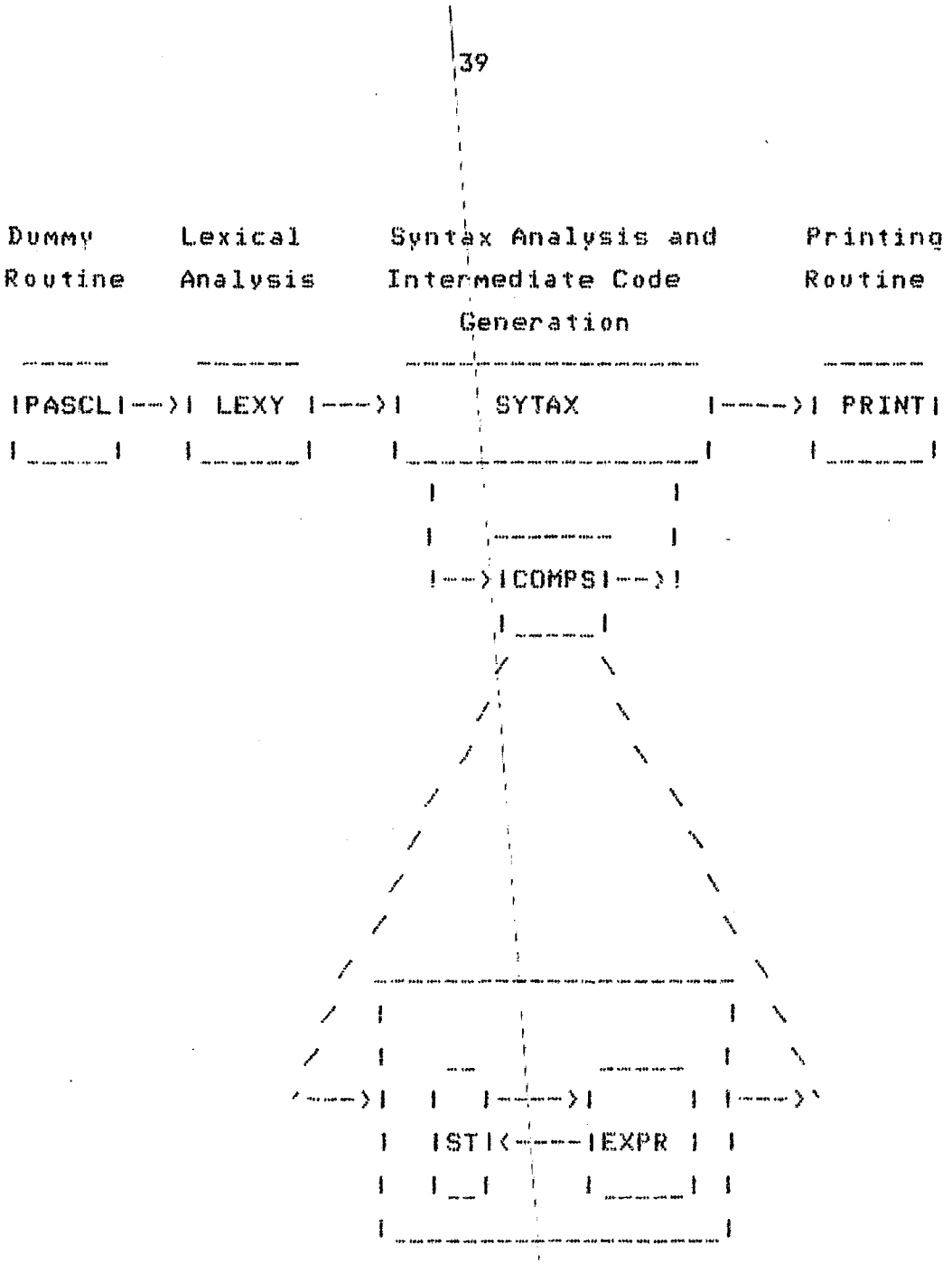


Fig. 3.3 Schematic Diagram of the Program

The routine ST first checks for the presence of a label in the input. On finding a label, it checks whether a valid declaration was made for the label or not. If the label was correctly declared, it checks whether the label has been defined earlier in the program or not. If not then it generates the intermediate code for defining the label and changes the status of the label in the label table to that of defined. If the label is undeclared, an error is reported in the error message table and the intermediate code for such a label is not generated. If the label was declared correctly but has been redefined, then the routine ignores the redefinition of the label and adds an appropriate error message in the error table.

If a label was not prseent in the input, the routine checks for the presence a conditional statement, namely, the "if" statement. On finding the keyword "if" present in the input the routine makes a call to the routine EXPR for processing the expression, and subsequently looks for keyword "then" followed by any of the allowed statements. The "if" statement may be followed by the keyword "else". If the keyword "else" is not present, the routine terminates the conditional statement, assuming it as an "if-then" type statement. But, if the keyword "else" is present, the routine looks for any of the allowed statements after "else". The matching of "if-then-else" structure is performed with the help of a stack. Along with the

verification of the syntactic correctness, the routine also generates the intermediate code for "if-then" or "if-then-else" statements. If the conditional statement was not present in the input, the routine ST checks the possibility of a repetitive statement, namely, the "while-do". In processing the "while-do" statement also, the routine has to call the routine EXPR to check the syntax of the expression; the routine EXPR also generates the code for the expression. The matching of a "do" for a "while" is performed by maintaining a stack. If the repetitive statement is not present in the input, the routine ST checks for a compound statement, namely the "begin-end". After the keyword "begin" there could be any number of the statements followed by the keyword "end". The matching of an "end" for a "begin" is performed by using a stack. If input did not denote a compound statement, the routine ST checks for the "goto" statement. The keyword "goto" should be followed by a label. A valid label has to be declared in the declaration part before its use. The routine then checks for the presence of an assignment statement statement. For processing an assignment statement, the routine call EXPR. If the input does not denote an assignment statement, the routine then checks for any of the "read", "raedln", "write", or "writeln" statements to be present. If the input does not happen to be any of these, the routine takes it as an invalid statement, and after reporting the error continues the parsing and intermediate code generation for the following

tokens.

Following is a brief description of the logic employed for generating the intermediate code for "if-then", "if-then-else", and "while-do" statements.

IF-THEN :

```
        !
        !

        IF e THEN s1;
        !
        !
```

The intermediate code generated is equivalent to

```
        !
        !

        code for expression e

        IF NOT(e) GOTO t1

        code for s1
      t1:
```

IF-THEN-ELSE :

```
        !
        !

        IF e THEN s1 Else s2;
        !
        !
```

The logic for generating the intermediate code is as follows

```
        !
        !

        code for expression e

        IF NOT(e) GOTO t1

        code for s1
        goto t2
t1:     code for s2
t2:
        !
        !
```

```
WHILE-DO :
        !
        !

       WHILE e DO s1;
        !
        !
The equivalent of the intermediate code is
        !
        !
    w1:code for e
       IF NOT(e) GOTO w2
       code for s1
       GOTO w1
    w2:
       !
       !
```

The routine EXPR parses the simple expressions and generates the intermediate code for syntactically correct simple expressions.

Due to the limitations of the memory partition (64KB), the complete work of reading the source, performing the lexical analysis and the syntax analysis, generating the intermediate code, and printing the tables has been divided into three programs which overlay each other. First, the lexical analysis program reads the source,

performs the lexical analysis, and prints the source. After it has completed its job, this program is overlayed by the syntax analysis and intermediate code generation program which, in turn, is finally overlayed by the third program which prints the various tables. The communication between these programs is established using common areas. The source listing of the programs are given in Appendix F.

3.3 AN EXAMPLE

In order to show the working of the machine-independenmt part of the compiler, an example of bubble sort has been chosen. The program written in Pascal reads ten integers, sorts them using the bubble sort algorithm, and prints the sorted integers. The example contains an error of double declaration. The program gives the corresponding error message and using the first declaration of the identifier, performs the analysis and generates the intermediate code. The Pascal program listing and various information tables along with the intermediate code are given in Appendix G.

CHAPTER 4

CONCLUSIONS

In this chapter the experience gained during the implementation, as well as the suggestions for improvement are described.

4.1 THE EXPERIENCE

It is observed that the language PL/I provides better facilities for writing a compiler than FORTRAN IV. The most serious disadvantage of FORTRAN IV is the inadequate facilities for manupulating characters and its unstructured nature.

It is also observed that the completion of the design before coding cuts down the development time considerably.

For writing compilers, the lexical analyser should take care of the conversions of data into the internal form. This helps in making the syntax analyser simpler.

4.2 SUGGESTIONS FOR IMPROVEMENTS

The subset of Pascal chosen for implementation may be extended to include extra facilities in data types as well as executable statements. The facilities like "boolean" and "subrange" may be included to enhance the data type facilities and in the executable statements another structured statement like "repeat-until" may be added.

BIBLIOGRAPHY

1. Aho, A.V. and Ullman, J.D. :'Principles of Compiler design', Addison-Wesley Publishing Co.,1979.
2. Donovan, J.J. : 'Systems Programming', McGraw-Hill Kogakusha Ltd.,1972.
3. Gries, D. : 'Compiler Construction for Digital Computers', John Wiley and Sons,Inc.,1971.
4. Gupta, R.G., et al : 'Pascal-1020 Compiler', Proceedings CSI-81.
5. Jensen, K. and Wirth, N. :'Pascal Users' Manual and Report', Springer,NY,1974.
6. Rodmay Zaks :'An Introduction to Pascal', Sybex Inc.,1980.
7. Rana R.T. :'Code Generator for Pascal-1020 Compiler', M.Phil.Dissertation,SCSS,JNU,1981.
8. Teruo, H. and Kiyoshi, I. : 'An Extended PASCAL and its implementation Using a Trunk', Report of the Computer Centre Univ. of Tokyo', Vol5,1975-1976.
9. Wirth N., 'The Design of a Pascal Compiler', 'Software-Practice and Experience', No.3,309-333.
10. Richards, James L. :'Pascal', Academic Press,1982.

APPENDIX A

THE SUBSET

```
 <program>::=<program heading><block>.
<program heading>::=PROGRAM<identifier>;
<identifier>::=<letter>{<letter or digit>}
<letter or digit>::=<letter>|<digit>
<block>::=<label declaration part>
          <constant definition part><variable
          declaration part><statement part>
<label declaration part>::=<empty>|LABEL<label>{,label>};
<label>::=<unsigned integer>
<constant definition part>::=<empty>|
                              CONST<constant definition>{;
                              <constant definition>};
<constant definition>::=<identifier>=<constant>
<constant>::=<unsigned number>|<sign><unsigned number>
<unsigned number>::=<unsigned integer>|<unsigned real>
<unsigned integer>::=<digit>{<digit>}
<unsigned real>::=<unsigned integer>.<digit>{<digit>}|
                  <unsigned integer>.<digit>{<digit>}
                     E<scale factor>|
                  <unsigned integer>E<scale factor>
<scale factor>::=<unsigned integer>|
                 <sign><unsigned integer>
```

```
<sign>::=+|-
<variable declaration part>::=<empty>|
                              VAR<variable declaration>
                              {;<variable declaration>};
<variable declaration>::=<identifier>{,<identifier>}
                         :<type>
<type>::=<standard type>|<array type>
<standard type>::=INTEGER|REAL|CHAR
<array type>::=ARRAY(1:unsigned integer)
               OF <standard type>
<statement part>::=<compound statement>
<statement>::=<unlabelled statement>|<label>:
              <unlabelled statement>
<unlabelled statement>::=<simple statement>|
                         <structured statement>
<simple statement>::=<assignment statement>|
                     <go to statement>|
                     <empty statement>
<assignment statement>::=<variable>:=<expression>
<variable>::=<identifier identifier>|<array variable>
<variable identifier>::=<identifier>
<array variable>::=<variable>
<expression>::=<simple expression>|
               <simple expression><relational
               operator><simple expression>
<relational operator>::= =|<|<=|>|>=|<>
<simple expression>::=<term>|<sign><term>|
                      <simple expression>
```

```
                        <adding operator><term>
<adding operator>::=+|-
<term>::=<factor>|<term><multiplying operator><factor>
<multiplying operator>::=*|/
<factor>::=<variable>|<unsigned constant>|(<expression>)
<unsigned constatnt>::=<unsigned number>|<string>
<go to statement>::=GOTO <label>
<empty statement>::=<empty>
<empty>::=
<structured statement>::=<compound statement>|
                         <conditional statement>|
                         <repetitive statement>
<compound statement>::=BEGIN <statement>{;<statement>}
                       END
<conditional statement>::=<if statement>
<if statement>::=IF <expression> THEN <statement>|
                 IF <expression> THEN <statement>
                 ELSE <statement>
<repetitive statement>::=<while statement>
<while statement>::=WHILE <expression> DO <statement>

 Input and Output Procedures :

<input procedure>::=READ(<input parameter>)|
                    READLN(<input parameter>)
<input parameter>::=<identifier>{,<identifier>}
<output procedure>::=WRITE(<output parameter>)|
                     WRITELN(<output parameter>)
```

<output parameter>::=<identifier>[[:<unsigned integer>]
:<unsigned integer>]

APPENDIX B

LIST OF KEYWORDS

Keyword Table Code : 3

INDEX	KEYWORD
1	PROGRAM
2	PASCAL
3	CONST
4	VAR
5	INTEGER
6	REAL
7	CHAR
8	ARRAY
9	OF
10	BEGIN
11	END
12	WHILE
13	DO
14	IF
15	THEN
16	ELSE
17	GOTO
18	READ

19	READLN
20	WRITE
21	WRITELN
22	LABEL

APPENDIX C

LIST OF DELIMITERS

Delimiter Table Code : 4

INDEX	DELIMITER
1	+
2	-
3	*
4	/
5	=
6	<
7	>
8	(
9	)
10	:
11	;
12	,
13	.
14	'

APPENDIX D

LIST OF OPERATORS

OPERATOR	OPCODE	OPERATION
Program	0	Name of the program
+	1	Add the second operand to the first
-	2	Subtract the second operand from the first
*	3	Multiply second operand with the first and put the result in the first operand
/	4	Divide the first operand by the second and the result in the first operand
Label	5	Define the label
Compare	6	Compare first and the second operand
Goto	7	Jump
Read	10	Read without line feed
Readln	11	Line feed after Read
Write	12	Write without line feed
Writeln	13	Line feed after Write
L+	50	Load the second operand in the first
L-	51	Load the negative of second operand in the first
End	99	End of the code

APPENDIX E

LIST OF ERROR CODES

CODE	MESSAGE
100	Identifier exceeds 8 characters, truncated to first 8 characters
102	Illegal character present in the input
200	Progrm not starting with keyword "PROGRAM", keyword assumed
202	Program id missing, assumed PASCAL
204	Keyword being used as program id, assumed PASCAL
206	; missing, assumed
208	Illogical end of program
210	Unidentified syntax error, statement ignored
212	Keyword being used as identifier, statement ignored
214	= expected after identifier, statement ignored
216	real or integer constant expected
218	: expected after integer, statement ignored
220	Keyword INTEGER/REAL/CHAR/ARRAY expected, statement ignored
222	(expected after identifier, statement ignored
224	Unsigned integer constant expected, statement ignored

226 Array starting range improper, assumed 1

228 Keyword INTEGER/REAL/CHAR/ARRAY expected, statement ignored

230) missing, statement ignored

232 Keyword OF missing

234 Type for declared constant declared, declaration ignored

236 Type of an identifier redeclared, declaration ignored

238 Array redeclared, declaration ignored

240 Program id cannot be used as an identifier, declaration ignored

242 Declared identifier being declared as a constant, declaration ignored

244 Constant identifier redeclared, declaration ignored

250 Main block BEGIN missing, assumed

252 Mismatching Begin present

254 Keyword END missing, assumed

256 Invalid combination of relational operators used, statement ignored

258 Invalid relational operator used, statement ignored

260 Keyword THEN missing, statement ignored

262 Keyword DO missing, statement ignored

264 Mismatching End present

266 Invalid label in GOTO statement, statement ignored

272 Label redefined, definition ignored

274 : missing after identifier, assumed

276 = missing, assumed
278 Statement not recognized, ignored
280 Mismatching ELSE present, statement ignored
282 Mismatching DO present, statement ignored
284 Extra ; present, null statement assumed
286 (missing, assumed
288) missing, assumed
290 Format error, statement ignored.

APPENDIX F

SOURCE PROGRAM LISTING

```
0001  FTN4,L
0002        BLOCK DATA
0003  C-----IDENTIFIER TABLE-----
0004        INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
0005       *RADDR(256)
0006  C-----LITERAL TABLE-----
0007        INTEGER ILIT(64),CLIT(64)
0008        REAL RLIT(64)
0009  C
0010  C-----LABLE TABLE-----
0011  C
0012        INTEGER LAB(32),LABTYP(32)
0013  C-----UNIFORM SYMBOL TABLE-----
0014        INTEGER TAB(512),INDEX(512)
0015  C-----ERROR MESSAGE TABLE-----
0016        INTEGER STMNT(32),ECODE(32)
0017  C-----KEYWORD TABLE-----
0018        INTEGER KWORD(4,22)
0019  C-----DELIMITERS-----
0020        INTEGER DEL(14)
0021        INTEGER A(80),IDVP(4),IDVU(8),C
0022        INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
0023        REAL TEMP
0024        INTEGER UN,ERRN,E,I,END,INT,ILITN,RLITN,CLITN,IDN,LABN
0025        INTEGER STM(16),NUMB(16),S,B,W,EL,TM,COP(2)
0026  C
0027  C-----INTERMEDIATE CODE TABLE-----
0028  C
0029        INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
0030       *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
0031  C
0032  C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)-----
0033  C
0034        INTEGER TVAR(16),TPREC(16),TOP(16)
0035        INTEGER PREC,OP,COUNT,T
0036        INTEGER PGM(4),LST,PAGE,LINE
0037        INTEGER IDCB(180)
0038        COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,L
0039       *,LABTYP
0040        COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALE
0041        COMMON /ERR/STMNT,ECODE,ERRN,E
0042        COMMON /UFORM/ TAB,INDEX
```

```
0043          COMMON /CARD/ C
0044          COMMON /INC1/ I
0045          COMMON /FIN/ END,UN
0046          COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
0047          COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
0048          COMMON /PRNT/ PGM,LST,PAGE,LINE
0049          COMMON/PAL/IDCB
0050          END
0051          PROGRAM PASCL,3
0052          INTEGER NAME(3),PAGE
0053          INTEGER PGM(4)
0054          COMMON/PRNT/PGM,LST,PAGE,LINE
0055          DATA ICODE,NAME/8,2HLE,2HXY,2H  /
0056          LST=9
0057          CALL EXEC(ICODE,NAME)
0058          END
0059    FTN4,L
0060          PROGRAM LEXY,5
0061    C----IDENTIFIER TABLE----
0062          INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
0063         *RADDR(256)
0064    C----LITERAL TABLE----
0065          INTEGER ILIT(64),CLIT(64)
0066          REAL RLIT(64)
0067    C
0068    C----LABLE TABLE----
0069    C
0070          INTEGER LAB(32),LABTYP(32)
0071    C----UNIFORM SYMBOL TABLE----
0072          INTEGER TAB(512),INDEX(512)
0073    C----ERROR MESSAGE TABLE----
0074          INTEGER STMNT(32),ECODE(32)
0075    C----KEYWORD TABLE----
0076          INTEGER KWORD(4,22)
0077    C----DELIMITERS----
0078          INTEGER DEL(14)
0079          INTEGER A(80),IDVP(4),IDVU(8),C
0080          INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
0081          REAL TEMP
0082          INTEGER UN,ERRN,E,I,END,INT,ILITN,RLITN,CLITN,IDN,LABN
0083          INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
0084    C
0085    C----INTERMEDIATE CODE TABLE----
0086    C
0087          INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
0088         *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
```

```
0089   C
0090   C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)-----
0091   C
0092          INTEGER TVAR(16),TPREC(16),TOP(16)
0093          INTEGER PREC,OP,COUNT,T
0094          INTEGER PGM(4),LST,PAGE,LINE
0095          INTEGER IDCB(144),NAM(3),IIBUF(40)
0096          INTEGER NAME(3)
0097          COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,L
0098         *,LABTYP
0099          COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALE
0100          COMMON /ERR/STMNT,ECODE,ERRN,E
0101          COMMON /UFORM/ TAB,INDEX
0102          COMMON /CARD/ C
0103          COMMON /INC1/ I
0104          COMMON /FIN/ END,UN
0105          COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
0106          COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
0107          COMMON /PRNT/ PGM,LST,PAGE,LINE
0108          DATA ICODE,NAME/8,2HSY,2HTA,1HX/
0109          DATA IB/2H  /
0110          DATA FMT/2H(E,1H0,1H0,2H.0,1H)/
0111          DATA DIGIT/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
0112          DATA KWORD/ 2HPR,2HOG,2HRA,2HM
0113         *,2HPA,2HSC,2HAL,2H
0114         *,2HCO,2HNS,2HT  ,2H
0115         *,2HVA,2HR  ,2H   ,2H
0116         *,2HIN,2HTE,2HGE,2HR
0117         *,2HRE,2HAL,2H   ,2H
0118         *,2HCH,2HAR,2H   ,2H
0119         *,2HAR,2HRA,2HY  ,2H
0120         *,2HOF,2H   ,2H   ,2H
0121         *,2HBE,2HGI,2HN  ,2H
0122         *,2HEN,2HD  ,2H   ,2H
0123         *,2HWH,2HIL,2HE  ,2H
0124         *,2HDO,2H   ,2H   ,2H
0125         *,2HIF,2H   ,2H   ,2H
0126          *,2HTH,2HEN,2H   ,2H
0127         *,2HEL,2HSE,2H   ,2H
0128          *,2HGO,2HTO,2H   ,2H
0129         *,2HRE,2HAD,2H   ,2H
0130         *,2HRE,2HAD,2HLN,2H
0131         *,2HWR,2HIT,2HE  ,2H
0132          *,2HWR,2HIT,2HEL,2HN
0133         *,2HLA,2HBE,2HL  ,2H  /
0134          DATA DEL/
```

```
0135          : 2H+ ,2H- ,2H* ,2H/ ,2H= ,2H( ,2H) ,2H(
0136          :,2H) ,2H: ,2H; ,2H, ,2H. ,2H' /
0137    C
0138    C
0139    C     I-COUNTER FOR NO OF CHAR IN A CARD
0140    C     J-NO OF CHARS IN AN ID
0141    C     K-NO OF DELIMITER PRESENT
0142    C     L-    "      "      "  NUMERIC LITERAL
0143    C     L1-NO OF CHARACTERS PRESENT IN LITERAL
0144    C     LPT-FLAG SET IF DECIMAL PRESENT IN NUMERIC LITERAL
0145    C     LPE FLAG SET IF EXP PRESENT IN NUMERIC LITERAL
0146    C     IDN-COUNTER FOR ID TAB
0147    C     ILITN-COUNTER FOR ILIT TAB
0148    C     CLITN-  "     "     CLIT TAB
0149    C     RLITN-  "     "     RLIT TAB
0150    C     ERRN -  "     "     ERROR TAB
0151    C     UN   -  "     "     UNIFORM SYMBOL TAB
0152    C     LABN -  "     "     LABLE TAB
0153    C     C-NO OF LINES READ
0154    C
0155    C
0156    C
0157           UN=0
0158           C=0
0159    10     WRITE(1,1)
0160    1      FORMAT(' ENTER INPUT FILE NAME :_')
0161           READ(1,2)(NAM(I),I=1,3)
0162    2      FORMAT(3A2)
0163           WRITE(1,3)
0164    3      FORMAT(' ENTER SECURITY CODE OF THE INPUT FILE :_')
0165           READ(1,*)ISEC
0166           CALL OPEN(IDCB,IER,NAM,0,ISEC)
0167           IF (IER.GT.0) GO TO 7
0168           WRITE(1,4)IER,NAM
0169    4      FORMAT('***FMGR ERROR : ',I3,' ON FILE : ',3A2)
0170           GO TO 10
0171    7      CALL HEAD1
0172    5      DO 15 I=1,80
0173    15     A(I)=IB
0174           IBK = 00040B
0175           CALL SFILL(IIBUF,1,80,IBK)
0176           CALL READF(IDCB,IER, IIBUF,40,L)
0177           IF (IER.LT.0) GO TO 370
0178           IF (L.LE.0) GO TO 370
0179           CALL CODE
0180           READ(IIBUF,37)(A(I),I=1,80)
```

```
0181  37     FORMAT(80A1)
0182  11     C=C+1
0183         LINE=LINE+1
0184         IF (LINE.GT.60) CALL HEAD1
0185         WRITE(LST,111) C,(A(I),I=1,80)
0186  111    FORMAT(4X,I4,3X,80A1)
0187         UN=UN+1
0188         TAB(UN)=99
0189         INDEX(UN)=C
0190         I=1
0191         J=0
0192  12     IF ((A(I).GE.1HA) .AND. (A(I).LE.1HZ)) GO TO 20
0193         IF (J.EQ.0) GO TO 110
0194         IF ((A(I).GE.1H0) .AND. (A(I).LE.1H9)) GO TO 20
0195         GO TO 40
0196  20     I=I+1
0197         J=J+1
0198         IF((I.GT.72) .AND. (J.GT.8))GO TO 40
0199         IF (J.GT.8) GO TO 30
0200         IF (I.GT.72) GO TO 40
0201         GO TO 12
0202  C-----ID EXCEEDS 8 CHARS----
0203  30     IF(((A(I).GE.1HA) .AND. (A(I).LE.1HZ)) .OR. ((A(I).GE.1H0) .AN
0204        -(A(I).LE.1H9))) GO TO 20
0205  35     ERRN=ERRN+1
0206         STMNT(ERRN)=C
0207         ECODE(ERRN)=100
0208  C-----OTHER THAN ID-----
0209  40     IF (J.EQ.0) GO TO 110
0210  C-----INSTAL IDENTIFIER AFTER CHECKING FOR KEYWORDS-----
0211         DO 45 IJ=1,8
0212         IDVU(IJ)=2H
0213  45     CONTINUE
0214         IJ2=I-J
0215         IF (J.GT.8) J=8
0216         DO 50 IJ=IJ2,IJ2+J-1
0217         IDVU(IJ-IJ2+  1)=A(IJ)
0218  50     CONTINUE
0219         DO 60 IJ=1,4
0220         IDVP(IJ)=IDVU(((IJ-1)*2)+1)-32+IDVU(IJ*2)/256
0221  60     CONTINUE
0222  C-----CHECK IF ID ALREADY EXISTS IN ID TAB-----
0223         DO 66 I1=1,IDN
0224         DO 64 I2=1,4
0225         IF (ID(I2,I1) .NE. IDVP(I2)) GO TO 66
0226  64     CONTINUE
```

```
0227          GO TO 68
0228    66    CONTINUE
0229          GO TO 69
0230    68    UN=UN+1
0231          INDEX(UN)=I1
0232          GO TO 102
0233    69    DO 80 IJ=1,22
0234          DO 70 IJ1=1,4
0235          IF (IDVP(IJ1).NE.KWORD(IJ1,IJ)) GO TO 80
0236    70    CONTINUE
0237    C-----INSTAL ID IN UFORM TAB-----
0238          UN=UN+1
0239          TAB(UN)=3
0240          INDEX(UN)=IJ
0241          GO TO 105
0242    80    CONTINUE
0243    C-----INSTAL ID & UNIFORM  IN ID TAB-----
0244    90    IDN=IDN+1
0245          DO 100 IJ=1,4
0246          ID(IJ,IDN)=IDVP(IJ)
0247    100   CONTINUE
0248          UN=UN+1
0249          INDEX(UN)=IDN
0250    102   TAB(UN)=1
0251    105   J=0
0252    C-----CHECK FOR LITERALS-----
0253    110   L=0
0254          L1=0
0255          LPT=0
0256          LPE=0
0257    120   IF ((A(I).GE.1H0) .AND. (A(I).LE.1H9)) GO TO 130
0258          GO TO 140
0259    130   L1=L1+1
0260          LIT(L1)=A(I)
0261          I=I+1
0262          IF (I.GT.72) GO TO 150
0263          GO TO 120
0264    140   IF  (A(I).NE.1H.) GO TO 145
0265          LPT=1
0266          GO TO 130
0267    145   IF (A(I).NE.1HE) GO TO 147
0268          LPE=1
0269          GO TO 130
0270    147   IF (.NOT.((LPE.EQ.1).AND.((A(I).EQ.1H+).OR.(A(I).EQ.1H-))))
0271         -GO TO 150
0272          GO TO 130
```

```
0273  150    IF(.NOT. ((LPT.EQ.1) .AND. (L1.EQ.1)))GO TO 155
0274         K=13
0275         GO TO 326
0276  155    IF (L1.EQ.0) GO TO 300
0277         FST=L1/10
0278         SCN=L1-(FST*10)+1
0279         FST=FST+1
0280         FMT(2)=DIGIT(FST)
0281         FMT(3)=DIGIT(SCN)
0282  C-----INITIALIZE THE BUFFER-----
0283         DO 160 L2=1,40
0284         IBUF(L2)=2H
0285  160    CONTINUE
0286  C-----FILL THE BUFFER WITH SPACES REMOVED-----
0287         L1=(L1+1)/2
0288         DO 170 L2=1,L1
0289         M=(L2-1)*2+1
0290         IBUF(L2)=LIT(M)-32+LIT(M+1)/256
0291  170    CONTINUE
0292  C-----CONVERT IN INTERNAL REP-----
0293         CALL CODE
0294         READ(IBUF,FMT)TEMP
0295  C-----INSTAL IN LITERAL & UFORM TAB-----
0296         IF((LPT.EQ.1).OR.(LPE.EQ.1))  GO TO 200
0297  C-----INSTAL INTEGER IN ILIT & UFORM TAB-----
0298         ITEMP=TEMP
0299  C-----FIND IF LITERAL ALREADY EXISTS IN ILIT-----
0300         IF((ITEMP.EQ.0) .AND. (ILITN.EQ.0))GO TO 182
0301         DO 180 L2=1,ILITN
0302         IF (ILIT(L2).EQ.ITEMP) GO TO 185
0303  180    CONTINUE
0304  182    ILITN=ILITN+1
0305         ILIT(ILITN)=ITEMP
0306         UN=UN+1
0307         INDEX(UN)=ILITN
0308         GO TO 190
0309  185    UN=UN+1
0310         INDEX(UN)=L2
0311  190    TAB(UN)=21
0312         GO TO 300
0313  C-----INSTAL REAL IN RLIT & UFORM TAB-----
0314  C-----FIND IF LITERAL ALREADY EXISTS IN RLIT-----
0315  200    DO 210 L2=1,RLITN
0316         IF (RLIT(L2).EQ.TEMP) GO TO 220
0317  210    CONTINUE
0318         RLITN=RLITN+1
```

```
0319          RLIT(RLITN)=TEMP
0320          UN=UN+1
0321          INDEX(UN)=RLITN
0322          GO TO 230
0323   220    UN=UN+1
0324          INDEX(UN)=L2
0325   230    TAB(UN)=23
0326   C-----CHECK FOR DELIMITER-----
0327   300    L=0
0328          M=0
0329   305    DO 310 K=1,14
0330          IF (A(I).EQ.DEL(K)) GO TO 320
0331   310    CONTINUE
0332          IF (A(I).EQ.1H ) GO TO 360
0333   C-----INVALID CHAR PRESENT-----
0334          ERRN=ERRN+1
0335          STMNT(ERRN)=C
0336          ECODE(ERRN)=102
0337          GO TO 360
0338   C-----INSTAL DEL IN UFORM TAB-----
0339   C-----CHECK FOR COMMENTS-----
0340   320    IF (K.NE.8) GO TO 326
0341          I=I+1
0342          IF (I.GT.72) GO TO 5
0343          IF (A(I).NE.1H*) GO TO 324
0344   322    I=I+1
0345          IF (I.GT.72) GO TO 5
0346          IF (A(I).NE.1H*) GO TO 322
0347          I=I+1
0348          IF (I.GT.72) GO TO 5
0349          IF (A(I).NE.1H)) GO TO 322
0350          I=I+1
0351          IF (I.GT.72) GO TO 5
0352          GO TO 12
0353   324    I=I-1
0354   326    UN=UN+1
0355          TAB(UN)=4
0356          INDEX(UN)=K
0357   330    I=I+1
0358          IF (I.GT.72) GO TO 5
0359          IF((K.EQ.14) .AND. (M.EQ.99))GO TO 365
0360          IF (K.NE.14) GO TO 12
0361          IF (A(I).EQ.1H') GO TO 340
0362          L=L+1
0363          IF (L.NE.2) GO TO 330
0364   C-----INSTAL CLIT-----
```

```
0365  332    IF (L.EQ.0) GO TO 12
0366         CLITN=CLITN+1
0367         IF (L.EQ.2) GO TO 334
0368         CLIT(CLITN)=A(I)
0369         GO TO 336
0370  334    CLIT(CLITN)=(A(I-1)-32)+A(I)/256
0371  336    M=M+1
0372         L=0
0373         IF (M.NE.1) GO TO 330
0374  C-----INSTAL IN UFORM TAB----
0375         UN=UN+1
0376         TAB(UN)=22
0377         INDEX(UN)=CLITN
0378         GO TO 330
0379  340    DO 350 K=1,14
0380         IF (A(I).NE.DEL(K)) GO TO 350
0381         M=99
0382         GO TO 320
0383  350    CONTINUE
0384         GO TO 12
0385  360    I=I+1
0386         IF (I.GT.72) GO TO 5
0387         GO TO 12
0388  C
0389  C-----PUT ## AT END OF CHAR LIT------
0390  C
0391  365    CLITN=CLITN+1
0392         CLIT(CLITN)=2H##
0393         GO TO 12
0394  C
0395  370    CALL CLOSE(IDCB,IER)
0396         CALL EXEC(ICODE,NAME)
0397         END
0398         SUBROUTINE HEAD1
0399  C----IDENTIFIER TABLE----
0400         INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
0401        *RADDR(256)
0402  C----LITERAL TABLE-----
0403         INTEGER ILIT(64),CLIT(64)
0404         REAL RLIT(64)
0405  C
0406  C-----LABLE TABLE-----
0407  C
0408         INTEGER LAB(32),LABTYP(32)
0409  C----UNIFORM SYMBOL TABLE----
0410         INTEGER TAB(512),INDEX(512)
```

```
0411  C----ERROR MESSAGE TABLE----
0412        INTEGER STMNT(32),ECODE(32)
0413  C----KEYWORD TABLE----
0414        INTEGER KWORD(4,22)
0415  C----DELIMITERS----
0416        INTEGER DEL(14)
0417        INTEGER A(80),IDVP(4),IDVU(8),C
0418        INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
0419        REAL TEMP
0420        INTEGER UN,ERRN,E,I,END,INT,ILITN,RLITN,CLITN,IDN,LABN
0421        INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
0422  C
0423  C----INTERMEDIATE CODE TABLE----
0424  C
0425        INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
0426       *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
0427  C
0428  C----TEMPORARY VARIABLE STACK--(TAB CODE : 10)----
0429  C
0430        INTEGER TVAR(16),TPREC(16),TOP(16)
0431        INTEGER PREC,OP,COUNT,T
0432        INTEGER PGM(4),LST,PAGE,LINE
0433        INTEGER IB(15)
0434        COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAB
0435       *,LABTYP
0436        COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
0437        COMMON /ERR/STMNT,ECODE,ERRN,E
0438        COMMON /UFORM/ TAB,INDEX
0439        COMMON /CARD/ C
0440        COMMON /INC1/ I
0441        COMMON /FIN/ END,UN
0442        COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
0443        COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
0444        COMMON /PRNT/ PGM,LST,PAGE,LINE
0445  C
0446  C
0447  C
0448  C
0449  C
0450        CALL FTIME(IB)
0451        PAGE=PAGE+1
0452  30    WRITE(LST,35)(IB(ICN),ICN=1,15),PAGE
0453  35    FORMAT(1H1,'    HP-PASCAL ',15X,15A2,10X,'
0454       *15X,'PAGE : ',I4,//)
0455  40    LINE=4
0456        END
```

```
0457  FTN4,L
0458        PROGRAM SYTAX,5
0459  C-----IDENTIFIER TABLE-----
0460        INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
0461       *RADDR(256)
0462  C-----LITERAL TABLE------
0463        INTEGER ILIT(64),CLIT(64)
0464        REAL RLIT(64)
0465  C
0466  C-----LABLE TABLE-----
0467  C
0468        INTEGER LAB(32),LABTYP(32)
0469  C-----UNIFORM SYMBOL TABLE-----
0470        INTEGER TAB(512),INDEX(512)
0471  C-----ERROR MESSAGE TABLE-----
0472        INTEGER STMNT(32),ECODE(32)
0473  C-----KEYWORD TABLE------
0474        INTEGER KWORD(4,22)
0475  C-----DELIMITERS-----
0476        INTEGER DEL(14)
0477        INTEGER A(80),IDVP(4),IDVU(8),C
0478        INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
0479        REAL TEMP
0480        INTEGER UN,ERRN,E,I,END,IDN,ILITN,RLITN,CLITN,INT,IDN,LABN
0481        INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
0482  C
0483  C-----INTERMEDIATE CODE TABLE-----
0484  C
0485        INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
0486       *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
0487  C
0488  C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)------
0489  C
0490        INTEGER TVAR(16),TPREC(16),TOP(16)
0491        INTEGER PREC,OP,COUNT,T
0492        INTEGER NAME(3)
0493        COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAB
0494       *,LABTYP
0495        COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
0496        COMMON /ERR/STMNT,ECODE,ERRN,E
0497        COMMON /UFORM/ TAB,INDEX
0498        COMMON /CARD/ C
0499        COMMON /INC1/ I
0500        COMMON /FIN/ END,UN
0501        COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
0502        COMMON/COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
```

```
0503          DATA ICODE,NAME/8,2HPR,2HIN,1HT/
0504          C=0
0505          I=0
0506          CALL INC
0507    1000  IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.1))) GO TO 1070
0508          CALL INC
0509          IF (END.EQ.1) GO TO 1110
0510    1010  IF (.NOT.((TAB(I).EQ.1).OR.(TAB(I).EQ.3))) GO TO 1080
0511          IF (.NOT.(TAB(I).EQ.1)) GO TO 1040
0512          TYPE(INDEX(I))=4
0513          INT=INT+1
0514    1020  OPC(INT)=0
0515          TAB1(INT)=TAB(I)
0516          IND1(INT)=INDEX(I)
0517          GO TO 1050
0518    1040  IF (.NOT.(INDEX(I).EQ.2)) GO TO 1090
0519          GO TO 1020
0520    1050  CALL INC
0521          IF (END.EQ.1) GO TO 1110
0522    1060  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 1100
0523          GO TO 1200
0524    C
0525    C-----PROGRAM NOT STARTING WITH KEYWORD PROGRAM, ASSUMED-----
0526    C
0527    1070  E=200
0528          CALL EFILL
0529          GO TO 1010
0530    C
0531    C-----PROGRAM ID DOES NOT EXIST ASSUMED PASCAL-----
0532    C
0533    1080  E=202
0534          CALL EFILL
0535          INT=INT+1
0536          OPC(INT)=5
0537          TAB1(INT)=3
0538          IND1(INT)=2
0539          GO TO 1060
0540    C
0541    C-----KEYWORD NOT TO BE A PROGRAM NAME, ASSUMED PASCAL-----
0542    C
0543    1090  E=204
0544          CALL EFILL
0545          INT=INT+1
0546          OPC(INT)=5
0547          TAB1(INT)=3
0548          IND1(INT)=2
```

```
0549          GO TO 1050
0550    C
0551    C-----STATEMENT TERMINATOR NOT PRESENT, ASSUMED-----
0552    C
0553    1100  E=206
0554          CALL EFILL
0555          C=C+1
0556          GO TO 1200
0557    C
0558    C-----ILLOGICAL END OF PROGRAM, JOB TERMINATED-----
0559    C
0560    1110  E=208
0561          CALL EFILL
0562          GO TO 99999
0563    C
0564    C
0565    C
0566    C
0567    1200  CALL INC
0568          IF (END.EQ.1) GO TO 1110
0569          IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.22))) GO TO 1210
0570    C
0571    C-----LABLE PRESENT-----
0572    C
0573          CALL INC
0574          IF (END.EQ.1) GO TO 1110
0575          IF (TAB(I).NE.21) GO TO 1207
0576          IA1=INDEX(I)
0577          GO TO 1204
0578    1202  CALL INC
0579          IF (END.EQ.1) GO TO 1110
0580          IF (TAB(I).NE.21) GO TO 1207
0581          IA2=INDEX(I)
0582    1204  CALL INC
0583          IF (END.EQ.1) GO TO 1110
0584          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.12))) GO TO 1205
0585          GO TO 1202
0586    C
0587    C-----CHECK IF          MISSING-----
0588    C
0589    1205  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 1208
0590          LABN=1
0591          DO 1206 N=IA1,IA2
0592          LAB(LABN)=ILIT(N)
0593          LABTYP(LABN)=0
0594          LABN=LABN+1
```

```
0595  1206  CONTINUE
0596        GO TO 1209
0597  C
0598  C----UNIDENTIFIED SYNTAX ERROR, STATEMENT IGNORED----
0599  C
0600  1207  E=210
0601        CALL EFILL
0602        CALL IGNOR
0603        GO TO 1209
0604  C
0605  C----                    MISSING, ASSUMED----
0606  C
0607  1208  E=206
0608        CALL EFILL
0609        GO TO 1210
0610  1209  CALL INC
0611        IF (END.EQ.1) GO TO 1110
0612  1210  IF (.NOT.((TAB(I).EQ.3).AND.((INDEX(I).EQ.3).OR.(INDEX(I).EQ.4
0613       -GO TO 1820
0614        IF (.NOT.(INDEX(I).EQ.3)) GO TO 1400
0615  C
0616  C----CONST PRESENT----
0617  C
0618        CALL INC
0619        IF (END.EQ.1) GO TO 1110
0620        IF (.NOT.(TAB(I).EQ.1)) GO TO 1300
0621  1250  IARR1=INDEX(I)
0622        CALL INC
0623        IF (END.EQ.1) GO TO 1110
0624        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.5))) GO TO 1320
0625        CALL INC
0626        IF (END.EQ.1) GO TO 1110
0627        IF (.NOT.((TAB(I).EQ.21) .OR.(TAB(I).EQ.23))) GO TO 1330
0628  C
0629  C----CHECK IF ID IS DEFINED AS A VARIABLE, IGNOR DEFN.----
0630  C
0631        IF (TYPE(IARR1).EQ.0) GO TO 1254
0632  C
0633  C----CHECK IF PGM NAME IS A CONSTT, IGNORE DEFN.----
0634  C
0635        IF (TYPE(IARR1).NE.4) GO TO 1252
0636        E=240
0637        GO TO 1305
0638  1252  E=242
0639         GO TO 1305
0640  C
```

```
641   C-----CHECK IF ALREADY DEFINED,IGNORE DEFN-----
642   C
643   1254   IF (LPTR(IARR1).EQ.0) GO TO 1258
644          E=244
645          GO TO 1305
646   1258   LPTR(IARR1)=TAB(I)*1000+INDEX(I)
647   1260   CALL INC
648          IF (END.EQ.1) GO TO 1110
649          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 1200
650   1270   CALL INC
651          IF (END.EQ.1) GO TO 1110
652          IF (TAB(I).EQ.1) GO TO 1250
653          IF (TAB(I).EQ.3) GO TO 1210
654          GO TO 1820
655   C
656   C----ID MISSING, STMNT IGNORED-----
657   C
658   1300   IF (TAB(I).EQ.3) GO TO 1310
659          E=210
660   1305   CALL EFILL
661          CALL IGNOR
662          IF (END.EQ.1) GO TO 1110
663          GO TO 1270
664   C
665   C-----KEYWORD CANNOT BE AN ID, STMNT IGNORED-----
666   C
667   1310   E=212
668          GO TO 1305
669   C
670   C---- = EXPECTED AFTER ID, STMNT IGNORED-----
671   C
672   1320   E=214
673          GO TO 1305
674   C
675   C-----REAL OR INTEGER COSNT EXPECTED, STMNT IGNORED-----
676   C
677   1330   E=216
678          GO TO 1305
679   C
680   C----VAR PRESENT-----
681   C
682   1400    CALL INC
683          IF (END.EQ.1) GO TO 1110
684   1420   IF (TAB(I).NE.1) GO TO  1650
685   1425   IA1=INDEX(I)
686   1430   CALL INC
```

```
0687          IF (END.EQ.1) GOTO 1110
0688          IF (TAB(I).NE.1) GO TO 1440
0689          IF (INDEX(I).GT.IA2) GO TO 1435
0690    C
0691    C-----DOUBLE VAR DDECLARATION, DEFN IGNORED-----
0692    C
0693          E=236
0694          CALL EFILL
0695          GO TO 1430
0696    1435  IA2=INDEX(I)
0697          GO TO 1430
0698    1440  IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.12)) GO TO 1430
0699          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.10))) GO TO 1670
0700          CALL INC
0701          IF (END.EQ.1) GO TO 1110
0702          IF (.NOT.((TAB(I).EQ.3).AND.((INDEX(I).GT.4).AND.(INDEX(I).L
0703         -)) GO TO 1680
0704          IF (INDEX(I).NE.8) GO TO 1600
0705    C
0706    C-----ARRAY TYPE VAR PRESENT-----
0707    C
0708    1500  ARRAY=1
0709          CALL INC
0710          IF (END.EQ.1) GO TO 1110
0711          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 1690
0712          CALL INC
0713          IF (END.EQ.1) GO TO 1110
0714          IF (TAB(I).NE.21) GO TO 1700
0715          IF (ILIT(INDEX(I)).NE.1)  GO TO 1710
0716    1520  CALL INC
0717          IF (END.EQ.1) GO TO 1110
0718          IF ((TAB(I).NE.4).AND.(INDEX(I).NE.10)) GO TO 1670
0719          CALL INC
0720          IF (END.EQ.1) GO TO 1110
0721          IF (TAB(I).NE.21) GO TO 1700
0722          VAL=ILIT(INDEX(I))
0723          CALL INC
0724          IF (END.EQ.1) GO TO 1110
0725          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.9))) GO TO 1740
0726          CALL INC
0727          IF (END.EQ.1) GO TO 1110
0728          IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.9))) GO TO 1750
0729          CALL INC
0730          IF (END.EQ.1) GO TO 1110
0731          IF (.NOT.((TAB(I).EQ.3).AND.((INDEX(I).GT.4).AND.(INDEX(I).L
0732         -))) GO TO 1720
```

```
0733  1600  TYP=INDEX(I)-4
0734        IF (TYP.EQ.1) IBYTE=2
0735        IF (TYP.EQ.2) IBYTE=4
0736        IF (TYP.NE.3) GO TO 1621
0737        IBYTE=1
0738        CALL INC
0739        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 1630
0740        CALL INC
0741        IF (END.EQ.1) GO TO 1110
0742        IF (TAB(I).NE.21) GO TO 1700
0743        IBYTE=ILIT(INDEX(I))
0744        CALL INC
0745        IF (END.EQ.1) GO TO 1110
0746        IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.9)) GO TO 1621
0747        GO TO 1740
0748  1621  CALL INC
0749        IF (END.EQ.1) GO TO 1110
0750  1630  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 1730
0751  C
0752  C----UPDATE ID TABLE----
0753  C
0754  1605  DO 1615 N=IA1 , IA2
0755        IF (LPTR(N).NE.0) GO TO 1609
0756        IF (TYPE(N).NE.0) GO TO 1611
0757        IF (VAL.EQ.0) GO TO 1607
0758        IF (LEN(N).NE.0) GO TO 1613
0759  1607  TYPE(N)=TYP
0760        BYTE(N)=IBYTE
0761        LEN(N)=VAL
0762        GO TO 1615
0763  C
0764  C----TYPE FOR CONST CANNOT BE DECLARED, DEFN. IGNORED----
0765  C
0766  1609  E=234
0767        CALL EFILL
0768        GO TO 1615
0769  C
0770  C----DOUBLE TYPE DECLARATION, DEFN IGNORED----
0771  C
0772  1611  E=236
0773        CALL EFILL
0774        GO TO 1615
0775  C
0776  C----ARRARY REDEFINED, DEFN IGNORED----
0777  C
0778  1613  E=238
```

```
0779          CALL EFILL
0780    1615  CONTINUE
0781    1620  IA1=0
0782          IA2=0
0783          VAL=0
0784          TYP=0
0785          IBYTE=0
0786          CALL INC
0787          IF (END.EQ.1) GO TO 1110
0788          IF (TAB(I).EQ.1) GO TO 1425
0789          GO TO 1210
0790    C
0791    C-----ID NOT PRESENT-----
0792    C
0793    1650  IF (TAB(I).EQ.3) GO TO 1660
0794          E=210
0795    1655  CALL EFILL
0796          CALL IGNOR
0797          IF (END.EQ.1) GO TO 1110
0798          GO TO 1620
0799    1660  E=212
0800          GO TO 1655
0801    C
0802    C----- : EXPECTED, STMNT IGNORED----
0803    C
0804    1670  E=218
0805          GO TO 1655
0806    C
0807    C-----INT,REAL,CHAR OR ARRAY EXPECTED, STMNT IGNORED-----
0808    C
0809    1680  E=220
0810          GO TO 1655
0811    C
0812    C----- ( EXPECTED AFTER ID, STMNT IGNORED-----
0813    C
0814    1690  E=222
0815          GO TO 1655
0816    C
0817    C----- +VE INTEGER EXPECTED, STMNT IGNORED-----
0818    C
0819    1700  E=224
0820          GO TO 1655
0821    C
0822    C----ARRAY STARTING RANGE NOT 1-----
0823    C
0824    1710  E=226
```

```
0825          CALL EFILL
0826          GO TO  1520
0827    C
0828    C-----VAR,REAL OR CHAR KEYWORD EXPECTED, STMNT IGNORED-----
0829    C
0830    1720  E=228
0831          GO TO 1655
0832    C
0833    C-----  ; MISSING, ASSUMED----- ASSUMED-----
0834    C
0835    1730  E=206
0836          CALL EFILL
0837          GO TO 1605
0838    C
0839    C-----  ) MISSING, STMNT IGNORED-----
0840    C
0841    1740  E=230
0842          GO TO 1655
0843    C
0844    C-----  KEYWORD OF MISSING, STMNT IGNORED-----
0845    C
0846    1750  E=232
0847          GO TO 1655
0848    1800  CONTINUE
0849    1820  CALL COMPS
0850          CALL INC
0851          IF (END.EQ.1) GO TO 1830
0852          IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.13)) GO TO 1835
0853          GO TO 1830
0854    C
0855    C-----  . MISSING, ASSUMED-----
0856    C
0857    1830  E=9999
0858          CALL EFILL
0859    1835  CALL INC
0860          IF (END.NE.1) GO TO 1840
0861          GO TO 99999
0862    C
0863    C-----ILLOGICAL END OF PROGRAM ------
0864    C
0865    1840  E=8999
0866          CALL EFILL
0867    99999 INT=INT+1
0868          OPC(INT)=99
0869          CALL EXEC(ICODE,NAME)
0870          END
```

```
0871          SUBROUTINE EXPR
0872    C----IDENTIFIER TABLE----
0873          INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
0874         *RADDR(256)
0875    C----LITERAL TABLE----
0876          INTEGER ILIT(64),CLIT(64)
0877          REAL RLIT(64)
0878    C
0879    C----LABLE TABLE----
0880    C
0881          INTEGER LAB(32),LABTYP(32)
0882    C----UNIFORM SYMBOL TABLE----
0883          INTEGER TAB(512),INDEX(512)
0884    C----ERROR MESSAGE TABLE----
0885          INTEGER STMNT(32),ECODE(32)
0886    C----KEYWORD TABLE----
0887          INTEGER KWORD(4,22)
0888    C----DELIMITERS----
0889          INTEGER DEL(14)
0890          INTEGER A(80),IDVP(4),IDVU(8),C
0891          INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
0892          REAL TEMP
0893          INTEGER UN,ERRN,E,I,END,IDN,ILITN,RLITN,CLITN,INT,LABN
0894          INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
0895    C
0896    C----INTERMEDIATE CODE TABLE----
0897    C
0898          INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
0899         *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
0900    C
0901    C----TEMPORARY VARIABLE STACK--(TAB CODE : 10)----
0902    C
0903          INTEGER TVAR(16),TPREC(16),TOP(16)
0904          INTEGER PREC,OP,COUNT,T
0905          COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAB
0906         *,LABTYP
0907          COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
0908          COMMON /ERR/STMNT,ECODE,ERRN,E
0909          COMMON /UFORM/ TAB,INDEX
0910          COMMON /CARD/ C
0911          COMMON /INC1/ I
0912          COMMON /FIN/ END,UN
0913          COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
0914          COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
0915    9000  OP=0
0916          COUNT=0
```

```
0917          PREC=1
0918    C
0919    C     INT-ENTRY NO IN INTERMEDIATE CODE TABLE
0920    C
0921    C
0922    C     T-ENTRY NO OF TEMP VAR STACK
0923    C
0924          T=0
0925    C
0926    C****FOR UNARY + & - ****
0927    C
0928          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.14))) GO TO 9200
0929          CALL INC
0930          IF (END.EQ.1) GO TO 90100
0931          IF (TAB(I).NE.22) GO TO 9320
0932          GO TO 9302
0933    9200        IF (.NOT.((TAB(I).EQ.4).AND.((INDEX(I).EQ.1).OR.
0934         -(INDEX(I).EQ.2)))) GO TO 9300
0935          T=T+1
0936          TVAR(T)=T
0937          TPREC(T)=PREC
0938          INT=INT+1
0939          IF (INDEX(I).NE.1) GO TO 9220
0940          OPC(INT)=50
0941          GO TO 9230
0942    9220  OPC(INT)=51
0943    9230  TAB1(INT)=10
0944          IND1(INT)=TVAR(T)
0945          ARR1(INT)=0
0946          CALL INC
0947          IF (END.EQ.1) GO TO 90100
0948          GO TO 9900
0949    9300  IF (.NOT.((TAB(I).EQ.1) .OR. (TAB(I).GT.20))) GO TO 9400
0950    9302  T=T+1
0951          TVAR(T)=T
0952          TPREC(T)=PREC
0953          INT=INT+1
0954          OPC(INT)=50
0955          TAB1(INT)=10
0956          IND1(INT)=TVAR(T)
0957          ARR1(INT)=0
0958    9305  TAB2(INT)=TAB(I)
0959          IND2(INT)=INDEX(I)
0960          ARR2(INT)=0
0961          IF (TAB(I).EQ.22) GO TO 93050
0962          IF (TAB(I).NE.1) GO TO 9310
```

```
0963  C
0964  C****VAR PRESENT****
0965  C
0966  C
0967  C----IF CHAR TYPE VAR PRESENT----
0968  C
0969        IF (TYPE(IND2(INT)).NE.3) GO TO 93051
0970        TAB1(INT)=11
0971        GO TO 93055
0972  C
0973  C----CHAR TYPE VAR PRESENTD----
0974  C
0975  93050 INT=INT+1
0976        OPC(INT)=50
0977        TAB1(INT)=11
0978        IND1(INT)=1
0979        TAB2(INT)=TAB(I)
0980        IND2(INT)=INDEX(I)
0981        CALL INC
0982        IF (END.EQ.1) GO TO 90100
0983        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.14))) GO TO 9320
0984        GO TO 99000
0985  93055 CALL INC
0986        IF (END.EQ.1) GO TO 90100
0987        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 99000
0988        CALL INC
0989        IF (END.EQ.1) GO TO 90100
0990        IF (TAB(I).NE.21) GO TO 93052
0991        ARR2(INT)=2
0992        ALEN2(INT)=ILIT(INDEX(I))
0993        GO TO 93053
0994  93052 IF (TAB(I).NE.1) GO TO 9320
0995        ARR2(INT)=1
0996        IND2(INT)=INDEX(I)
0997  93053 CALL INC
0998        IF (END.EQ.1) GO TO 90100
0999        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.9))) GO TO 9340
1000        CALL INC
1001        IF (END.EQ.1) GO TO 90100
1002        GO TO 99000
1003  93051 CALL INC
1004        IF (END.EQ.1) GO TO 90100
1005        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 9600
1006  C
1007  C****ARRAY TYPE VAR PRESENT****
1008  C
```

```
009          CALL INC
010          IF (END.EQ.1) GO TO 90100
011          IF (TAB(I).NE.21) GO TO 9306
012          ARR2(INT)=2
013          ALEN2(INT)=ILIT(INDEX(I))
014          GO TO 9307
015   9306   IF (TAB(I).NE.1) GO TO 9320
016          ARR2(INT)=1
017          ALEN2(INT)=INDEX(I)
018   9307   CALL INC
019          IF (END.EQ.1) GO TO 90100
020          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.9))) GO TO 9340
021   C
022   C-----CHECK IF VAR WAS ARRAY TYPE-----
023   C
024          IF (LEN(IND2(INT)).NE.0) GO TO 9310
025   C
026   C-----DIMENSION GIVEN FOR NON ARRAY TYPE VAR, STMNT IGNORED-----
027   C
028          E=112
029          GO TO 9360
030   9310   CALL INC
031          IF (END.EQ.1) GO TO 90100
032          GO TO 9600
033   C
034   C-----UNIDENTIFIED SYNTAX ERROR, STMNT IGNORED-----
035   C
036   9320   E=100
037          GO TO 9360
038   C
039   C-----  ) MISSING, STMNT IGNORED-----
040   C
041   9340   E=102
042          GO TO 9360
043   C
044   C-----CHECK IF VAR WAS ARRAY TIPE-----
045   C
046   9350   IF (LEN(IND2(INT)).EQ.0) GO TO 9600
047   C
048   C-----DIMENSION IN ARRAY VAR UNDEFINED, STMNT IGNORED-----
049   C
050          E=113
051   9360   CALL EFILL
052          CALL IGNOR
053          GO TO 90000
054   9400   IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 9500
```

```
1055  C
1056  C**** ( PRESENT IN THE BEGINING****
1057  C
1058        T=T+1
1059        TVAR(T)=T
1060        TPREC(T)=PREC
1061        PREC=PREC+1
1062        TOP(T+1)=50
1063        CALL INC
1064        IF (END.EQ.1) GO TO 90100
1065        GO TO 10200
1066  C
1067  C****NEITHER OF +, -, *, / PRESENT, STMNT IGNORED ****
1068  C
1069  9500  E=104
1070        GO TO 9360
1071  9600  IF (.NOT.((TAB(I).EQ.4).AND.((INDEX(I).GT.0).AND.(INDEX(I).L
1072       -))) GO TO 9700
1073  C
1074  C****EITHER OF +, -, *, / PRESENT****
1075  C
1076        IF (.NOT.((INDEX(I).EQ.3).OR.(INDEX(I).EQ.4))) GO TO 9610
1077  C
1078  C**** * OR / PRESENT ****
1079  C
1080        INT=INT+1
1081        OPC(INT)=INDEX(I)
1082  9605  TAB1(INT)=10
1083        IND1(INT)=TVAR(T)
1084        ARR1(INT)=0
1085        CALL INC
1086        IF (END.EQ.1) GO TO 90100
1087        GO TO 9900
1088  C
1089  C**** + OR - PRESENT****
1090  C
1091  9610  IF (T.EQ.1) GO TO 9620
1092        IF (.NOT.(TPREC(T).EQ.TPREC(T-1))) GO TO 9620
1093        INT=INT+1
1094        OPC(INT)=1
1095        TAB1(INT)=10
1096        IND1(INT)=TVAR(T-1)
1097        ARR1(INT)=0
1098        TAB2(INT)=10
1099        IND2(INT)=TVAR(T)
1100        ARR2(INT)=0
```

```
1101          T=T-1
1102   9620   T=T+1
1103          TVAR(T)=T
1104          TPREC(T)=PREC
1105          INT=INT+1
1106          OPC(INT)=50+INDEX(I)-1
1107          GO TO 9605
1108   9700   IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.9))) GO TO 9750
1109   C
1110   C**** ) PRESENT****
1111   C
1112          IF (T.EQ.1) GO TO 9720
1113          IF (.NOT.(TPREC(T).EQ.TPREC(T-1))) GO TO 9720
1114          INT=INT+1
1115          OPC(INT)=1
1116          TAB1(INT)=10
1117          IND1(INT)=TVAR(T-1)
1118          ARR1(INT)=0
1119          TAB2(INT)=10
1120          IND2(INT)=TVAR(T)
1121          ARR2(INT)=0
1122          T=T-1
1123   9720   INT=INT+1
1124          OPC(INT)=TOP(T)
1125          TAB1(INT)=10
1126          IND1(INT)=TVAR(T-1)
1127          ARR1(INT)=0
1128          TAB2(INT)=10
1129          IND2(INT)=TVAR(T)
1130          ARR2(INT)=0
1131          TOP(T)=0
1132          T=T-1
1133          PREC=PREC-1
1134          CALL INC
1135          IF (END.EQ.1) GO TO 90100
1136          GO TO 10400
1137   C
1138   C****CHECK IF COUNT OF PARANTHESIS IS ZERO OR NOT
1139   C
1140   9750   IF (COUNT.EQ.0) GO TO 9800
1141          ERRN=ERRN+1
1142          STMNT(ERRN)=C
1143          ECODE(ERRN)=106
1144          GO TO 90000
1145   9800   IF (TVAR(T).EQ.1) GOTO 99000
1146          INT=INT+1
```

```
1147          OPC(INT)=1
1148          TAB1(INT)=10
1149          IND1(INT)=TVAR(T-1)
1150          ARR1(INT)=0
1151          TAB2(INT)=10
1152          IND2(INT)=TVAR(T)
1153          ARR2(INT)=0
1154    9888  GO TO 99000
1155    9900  IF (.NOT.((TAB(I).EQ.1).OR.(TAB(I).GT.20))) GO TO 10000
1156    C
1157    C****CONST OR VAR PRESENT****
1158          GO TO 9305
1159    10000 IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 10100
1160    C
1161    C**** ( PRESENT****
1162    C
1163          TOP(T+1)=OPC(INT)
1164          INT=INT-1
1165          PREC=PREC+1
1166          CALL INC
1167          IF (END.EQ.1) GO TO 90100
1168          GO TO 10200
1169    C
1170    C****ERROR EITHER OF CONST, VAR, ( NOT PRESENT , STMNT IGNORED****
1171    C
1172    10100 E=108
1173          GO TO 9360
1174    10200 COUNT=COUNT+1
1175    10300 GO TO 9200
1176    10400 IF (COUNT.NE.0) GO TO 10500
1177    C
1178    C****ERROR NO OF ( GT ), STMNT IGNORED ****
1179    C
1180          E=110
1181          GO TO 9360
1182    10500 COUNT=COUNT-1
1183    10600 GO TO 9600
1184    90000 CONTINUE
1185    C
1186    C-----ILLOGICAL END OF PROGRAM-----
1187    C
1188    90100 E=208
1189          CALL EFILL
1190    99000 RETURN
1191          END
1192          SUBROUTINE COMPS, BEGIN-END
```

```
1193  C-----IDENTIFIER TABLE-----
1194        INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
1195       *RADDR(256)
1196  C-----LITERAL TABLE-----
1197        INTEGER ILIT(64),CLIT(64)
1198        REAL RLIT(64)
1199  C
1200  C-----LABLE TABLE-----
1201  C
1202        INTEGER LAB(32),LABTYP(32)
1203  C-----UNIFORM SYMBOL TABLE-----
1204        INTEGER TAB(512),INDEX(512)
1205  C-----ERROR MESSAGE TABLE-----
1206        INTEGER STMNT(32),ECODE(32)
1207  C-----KEYWORD TABLE-----
1208        INTEGER KWORD(4,22)
1209  C-----DELIMITERS-----
1210        INTEGER DEL(14)
1211        INTEGER A(80),IDVP(4),IDVU(8),C
1212        INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
1213        REAL TEMP
1214        INTEGER UN,ERRN,E,I,END,INT,ILITN,RLITN,CLITN,IDN,LABN
1215        INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
1216  C
1217  C-----INTERMEDIATE CODE TABLE-----
1218  C
1219        INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
1220       *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
1221  C
1222  C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)-----
1223  C
1224        INTEGER TVAR(16),TPREC(16),TOP(16)
1225        INTEGER PREC,OP,COUNT,T
1226        COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAB
1227       *,LABTYP
1228        COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
1229        COMMON /ERR/STMNT,ECODE,ERRN,E
1230        COMMON /UFORM/ TAB,INDEX
1231        COMMON /CARD/ C
1232        COMMON /INC1/ I
1233        COMMON /FIN/ END,UN
1234        COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
1235        COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
1236  C
1237  C-----INITIALIZE INTERMEDIATE CODE & STACK POINTER TO ZERO
1238  C
```

```
1239          S=0
1240          IF((TAB(I).EQ.3).AND.(INDEX(I).EQ.10))GO TO 2000
1241    C
1242    C-----KEYWORD BEGIN NOT PERESENT, ASSUMED----
1243    C
1244          E=250
1245          CALL EFILL
1246    C
1247    C-----GENERATE CODE FOR BEGIN-----
1248    C
1249    2000  S=S+1
1250          STM(S)=1HB
1251          NUMB(S)=0
1252    C
1253    C-----CALL STATEMENT LIST-----
1254    C
1255          CALL ST
1256    C
1257    C-----CHECK IF KEYWORD END EXISTS------
1258    C
1259    C     CALL INC
1260    C     IF (END.EQ.1) GO TO 2025
1261          IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.11))) GO TO 2020
1262    C
1263    C-----CHECK FOR BEGIN ON TOP OF STACK-----
1264    C
1265    2010  IF (STM(S).EQ.1HB) GO TO 2030
1266          E=252
1267          CALL EFILL
1268          GO TO 2040
1269    C
1270    C-----KEYWORD END MISSING, ASSUMED-----
1271    C
1272    2020  E=254
1273          CALL EFILL
1274          GO TO 2010
1275    2025  E=208
1276          CALL EFILL
1277          RETURN
1278    C
1279    C-----POP OFF BEGIN FROM STACK-----
1280    C
1281    2030  S=S-1
1282    2040  END
1283          SUBROUTINE ST    ,  STATEMENT
1284    C-----IDENTIFIER TABLE-----
```

```
1285          INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
1286         *RADDR(256)
1287    C-----LITERAL TABLE-----
1288          INTEGER ILIT(64),CLIT(64)
1289          REAL RLIT(64)
1290    C
1291    C-----LABLE TABLE-----
1292    C
1293          INTEGER LAB(32),LABTYP(32)
1294    C-----UNIFORM SYMBOL TABLE-----
1295          INTEGER TAB(512),INDEX(512)
1296    C-----ERROR MESSAGE TABLE-----
1297          INTEGER STMNT(32),ECODE(32)
1298    C-----KEYWORD TABLE-----
1299          INTEGER KWORD(4,22)
1300    C-----DELIMITERS-----
1301          INTEGER DEL(14)
1302          INTEGER A(80),IDVP(4),IDVU(8),C
1303          INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
1304          REAL TEMP
1305          INTEGER UN,ERRN,E,I,END,IDN,ILITN,RLITN,CLITN,INT,LABN
1306          INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
1307    C
1308    C-----INTERMEDIATE CODE TABLE-----
1309    C
1310          INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
1311         *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
1312    C
1313    C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)-----
1314    C
1315          INTEGER TVAR(16),TPREC(16),TOP(16)
1316          INTEGER PREC,OP,COUNT,T
1317          INTEGER Z
1318          INTEGER STNO(32),SP
1319          COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAB
1320         *,LABTYP
1321          COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
1322          COMMON /ERR/STMNT,ECODE,ERRN,E
1323          COMMON /UFORM/ TAB,INDEX
1324          COMMON /CARD/ C
1325          COMMON /INC1/ I
1326          COMMON /FIN/ END,UN
1327          COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
1328          COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
1329          SP=0
1330    2000  TR=0
```

```
1331  C      TR IS SET IF TERMINATOR IS PRESENT-----
1332         CALL INC
1333         IF (END.EQ.1) GO TO 8990
1334  2005   COP(1)=0
1335         COP(2)=0
1336         INDEX1=0
1337         ARR=0
1338         IND=0
1339  C
1340  C-----CHECK FOR A LABLE-----
1341  C
1342         IF (TAB(I).NE.21) GO TO 2007
1343         INDEX1=INDEX(I)
1344         CALL INC
1345         IF (END.EQ.1) GO TO 8990
1346         IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.10))) GO TO 20051
1347  C
1348  C-----CHECK IF LABLE IS DECLARED OR NOT-----
1349  C
1350         DO 20053 LC=1,LABN
1351         IF (ILIT(INDEX1).EQ.LAB(LC)) GO TO 20054
1352  20053 CONTINUE
1353  C
1354  C-----UNDECLARED LABLE, LABLE IGNORED-----
1355  C
1356         E=271
1357         CALL EFILL
1358         GO TO 20058
1359  C
1360  C-----CHECK IF LABLE HAS BEEN ALREADY USED-----
1361  C
1362  20054 IF (LABTYP(LC).NE.1) GO TO 20056
1363  C
1364  C-----LABLE REUSED, IGNORED-----
1365  C
1366         E=272
1367         CALL EFILL
1368         GO TO 20058
1369  C
1370  C-----SET THE LABLE STATUS TO USED-----
1371  C
1372  20056 LABTYP(LC)=1
1373         INT=INT+1
1374         OPC(INT)=5
1375         TAB1(INT)=24
1376         IND1(INT)=LC
```

```
20058 CALL INC
      IF (END.EQ.1) GO TO 8990
C
C-----ASSINGMENT STATEMENT------
C
2007  IF (TAB(I).NE.1) GO TO 2105
      INDEX1=INDEX(I)
      CALL INC
      IF (END.EQ.1) GO TO 8990
      IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.10)) GO TO 2010
      GO TO 5043
2010  CALL INC
      IF (END.EQ.1) GO TO 8990
      IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.5)) GO TO 5090
C
C-----UNIDENTIFIED SYNTAX ERROR, STMNT IGNORED-----
C
20051 E=210
      GO TO 4025
2100  CALL INC
      IF (END.EQ.1) GO TO 8990
C
C-----CHECK FOR KEYWORD IF-----
C
2105  IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.14))) GO TO 2277
      TR=0
      CALL INC
      IF (END.EQ.1) GO TO 8990
      CALL EXPR
      INT=INT+1
      OPC(INT)=50
      TAB1(INT)=23
      IND1(INT)=RLITN+1
      TAB2(INT)=10
      IND2(INT)=1
C
C-----CHECK FOR COMPARISON OPERATOR-----
C
      IF (.NOT.((TAB(I).EQ.4).AND.((INDEX(I).GT.4).AND.
     *(INDEX(I).LT.8)))) GO TO 2125
      COP(1)=INDEX(I)
      CALL INC
      IF (END.EQ.1) GO TO 8990
      IF (.NOT.((TAB(I).EQ.4).AND.((INDEX(I).GT.4).AND.
     *(INDEX(I).LT.8)))) GO TO 2200
      COP(2)=INDEX(I)
```

```
1423  2110   IF(((COP(1).EQ.5).AND.(COP(2).EQ.6)).OR.
1424        *((COP(1).EQ.6).AND.(COP(2).EQ.5))) GO TO 2190
1425         IF(((COP(1).EQ.6).AND.(COP(2).EQ.7)).OR.
1426        *((COP(1).EQ.7).AND.(COP(2).EQ.6))) GO TO 2190
1427         IF(((COP(1).EQ.5).AND.(COP(2).EQ.7)).OR.
1428        *((COP(1).EQ.7).AND.(COP(2).EQ.5))) GO TO 2190
1429         IF(COP(2).EQ.0) GO TO 2190
1430  C
1431  C-----INVALID COMBINATION OF COMPARISON OPERATORS, STMNT IGNORED---
1432  C
1433         E=256
1434         GO TO 2130
1435  C
1436  C-----INVALID COMPARISON OPERATOR, STMNT IGNORED-----
1437  C
1438  2125   E=258
1439  2130   CALL EFILL
1440         CALL IGNOR
1441         IF (TAB(I).EQ.4) GO TO 2272
1442         IF (TAB(I).EQ.3) GO TO 3025
1443  C
1444  C
1445  C
1446  2190   CALL INC
1447         IF (END.EQ.1) GO TO 8990
1448  2200   CALL EXPR
1449         INT=INT+1
1450         OPC(INT)=50
1451         TAB1(INT)=23
1452         IND1(INT)=RLITN+2
1453         TAB2(INT)=10
1454         IND2(INT)=1
1455  C
1456  C-----PUSH T1 IN STACK-----
1457  C
1458         S=S+1
1459         TH=TH+2
1460         STM(S)=1HT
1461         NUMB(S)=TH
1462  C
1463  C-----GENERATE INTERMEDIATE CODE-----
1464  C
1465         INT=INT+1
1466         OPC(INT)=6
1467         TAB1(INT)=23
1468         IND1(INT)=RLITN+1
```

```
1469          TAB2(INT)=23
1470          IND2(INT)=RLITN+2
1471          DO 2250 Z=1,3
1472          INT=INT+1
1473          OPC(INT)=7
1474          TAB1(INT)=1
1475          IND1(INT)=IDN+TH
1476    2250  CONTINUE
1477          IF ((COP (1).EQ.5).OR.(COP(2).EQ.5)) IND1(INT-2)=IDN+TH-1
1478          IF ((COP(1).EQ.6).OR.(COP(2).EQ.6)) IND1(INT-1)=IDN+TH-1
1479          IF ((COP(1) .EQ.7).OR.(COP(2).EQ.7)) IND1(INT  )=IDN+TH-1
1480          INT=INT+1
1481          OPC(INT)=5
1482          TAB1(INT)=1
1483          IND1(INT)=IDN+TH-1
1484    C
1485    C-----CHECK FOR KEYWORD THEN-----
1486    C
1487          IF ((TAB(I).EQ.3).AND.(INDEX(I).EQ.15)) GO TO 2260
1488    C
1489    C-----KEYWORD THEN MISSING, STMNT IGNORED-----
1490    C
1491          E=260
1492          GO TO 2130
1493    2260  SP=SP+1
1494          STNO(SP)=2270
1495          GO TO 2000
1496    C
1497    C-----POP OFF THEN FROM STACK, CHECK FOR KEYWORD ELSE-----
1498    C
1499    2270  CALL INC
1500          IF (END.EQ.1) GO TO 8990
1501    C
1502    C-----CHECK IF ; PERSENT------
1503    C
1504          TR=0
1505          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 2275
1506    2272  TR=1
1507          CALL INC
1508          IF (END.EQ.1) GO TO 8990
1509    2275  IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.16))) GO TO 2005
1510          GO TO 2278
1511    2277  IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.16))) GO TO 2400
1512    2278  IF (S.EQ.1) GO TO 9990
1513          IF (STM(S).NE.1HT) GO TO 2305
1514          T1=NUMB(S)
```

```
1515          S=S-1
1516          S=S+1
1517          EL=EL+1
1518          STM(S)=1HE
1519          NUMB(S)=EL
1520          INT=INT+1
1521          OPC(INT)=7
1522          TAB1(INT)=1
1523          IND1(INT)=IDN+EL+64
1524    2280  INT=INT+1
1525          OPC(INT)=5
1526          TAB1(INT)=1
1527          IND1(INT)=IDN+T1
1528          SP=SP+1
1529          STNO(SP)=2300
1530          GO TO 2000
1531    C     GO TO 9000
1532    2300  IF (S.EQ.1) GO TO 9990
1533          IF (STM(S).NE.1HE) GO TO 2310
1534          INT=INT+1
1535          OPC(INT)=5
1536          TAB1(INT)=1
1537          IND1(INT)=IDN+NUMB(S)
1538          IF (STM(S).EQ.1HE) IND1(INT)=IND1(INT)+64
1539          S=S-1
1540    C
1541    C----CHECK IF ; PRESENT
1542    C
1543          CALL INC
1544          IF (END.EQ.1) GO TO 8990
1545          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 2303
1546    2302  TR=1
1547          GO TO 9000
1548    2303  I=I-1
1549          GO TO 9000
1550    C
1551    C----KEYWORD IF MISSING CORRESPONDING THIS ELSE, STMNT IGNORED---
1552    C
1553    2305  E=280
1554    2308  CALL EFILL
1555          CALL IGNOR
1556          IF (TAB(I).EQ.4) GO TO 2302
1557          IF (TAB(I).EQ.3) GO TO 3025
1558    C
1559    C----UNIDENTIFIED SYNTAX ERROR----
1560    C
```

```
1561   2310  E=210
1562         GO TO 2308
1563   C2320 INT=INT+1
1564   C     OPC(INT)=5
1565   C     TAB1(INT)=1
1566   C     IND1(INT)=IDN+T1
1567   C
1568   C     I=I-1
1569   C
1570   C     GO TO 9000
1571   C
1572   C-----CHECK FOR KEYWORD WHILE-----
1573   C
1574   2400  IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.12))) GO TO 3010
1575   C
1576   C-----PUSH W3 & W1 IN STACK-----
1577   C
1578         TR=0
1579         W=W+3
1580         S=S+1
1581         STM(S)=1HW
1582         NUMB(S)=W
1583         S=S+1
1584         STM(S)=1HW
1585         NUMB(S)=W-2
1586   C
1587   C-----GENERATE LABEL W1-----
1588   C
1589         INT=INT+1
1590         OPC(INT)=5
1591         TAB1(INT)=1
1592         IND1(INT)=IDN+128+W-2
1593   C
1594   C-----CALL EXPRESSION-----
1595   C
1596         CALL INC
1597         IF (END.EQ.1) GO TO 8990
1598         CALL EXPR
1599         INT=INT+1
1600         OPC(INT)=50
1601         TAB1(INT)=23
1602         IND1(INT)=RLITN+1
1603         TAB2(INT)=10
1604         IND2(INT)=1
1605   C
1606   C-----CHECK FOR COMPARISON OP-----
```

```
1607  C
1608         IF (.NOT.((TAB(I).EQ.4).AND.((INDEX(I).GT.4).AND.
1609        *(INDEX(I).LT.8)))) GO TO 2430
1610         COP(1)=INDEX(I)
1611         CALL INC
1612         IF (END.EQ.1) GO TO 8990
1613         IF (TAB(I).NE.4) GO TO 2450
1614         IF (.NOT.((TAB(I).EQ.4).AND.((INDEX(I).GT.4).AND.
1615        *(INDEX(I).LT.8)))) GO TO 2420
1616         COP(2)=INDEX(I)
1617         GO TO 2440
1618  C
1619  C-----INVALID COMBINATION OF COMPARISON OPS, STMNT IGNORED-----
1620  C
1621  2420   E=256
1622  2425   CALL EFILL
1623         CALL IGNOR
1624         IF (TAB(I).EQ.4) GO TO 2492
1625         IF (TAB(I).EQ.3) GO TO 3025
1626  C
1627  C-----INVALID COMPARISON OP, STMNT IGNORED---
1628  C
1629  2430   E=258
1630         GO TO 2425
1631  C
1632  C-----CALL EXPRESSION-----
1633  C
1634  2440   CALL INC
1635         IF (END.EQ.1) GO TO 8990
1636  2450    CALL EXPR
1637         INT=INT+1
1638         OPC(INT)=50
1639         TAB1(INT)=23
1640         IND1(INT)=RLITN+2
1641         TAB2(INT)=10
1642         IND2(INT)=1
1643  C
1644  C-----GENERATE CODE-----
1645  C
1646         INT=INT+1
1647         OPC(INT)=6
1648         TAB1(INT)=23
1649         IND1(INT)=RLITN+1
1650         TAB2(INT)=23
1651         IND2(INT)=RLITN+2
1652         DO 2460 Z=1,3
```

```
1653          INT=INT+1
1654          OPC(INT)=7
1655          TAB1(INT)=1
1656          IND1(INT)=IDN+128+W
1657    2460  CONTINUE
1658          IF ((COP(1).EQ.5).OR.(COP(2).EQ.5)) IND1(INT-2)=IND1(INT)-1
1659          IF ((COP(1).EQ.6).OR.(COP(2).EQ.6)) IND1(INT-1)=IND1(INT)-1
1660          IF ((COP(1).EQ.7).OR.(COP(2).EQ.7)) IND1(INT  )=IND1(INT)-1
1661          INT=INT+1
1662          OPC(INT)=5
1663          TAB1(INT)=1
1664          IND1(INT)=IDN+128+W-1
1665    C
1666    C-----CHECK FOR KEYWORD DO-----
1667    C
1668          IF ((TAB(I).EQ.3).AND.(INDEX(I).EQ.13)) GO TO 2480
1669    C
1670    C-----KEYWORD DO MISSING, STMNT IGNORED-----
1671    C
1672          E=262
1673          GO TO 2425
1674    2480  SP=SP+1
1675          STNO(SP)=2490
1676          GO TO 2000
1677    C
1678    C-----CHECK FOR W ON STACK, POP OFF, GENERATE CODE-----
1679    C
1680    2490  IF (S.EQ.1) GO TO 9990
1681          IF (STM(S).NE.1HW) GO TO 2500
1682          INT=INT+1
1683          OPC(INT)=7
1684          TAB1(INT)=1
1685          IND1(INT)=IDN+128+NUMB(S)
1686          S=S-1
1687          IF (S.EQ.1) GO TO 9990
1688          INT=INT+1
1689          OPC(INT)=5
1690          TAB1(INT)=1
1691          IND1(INT)=IDN+128+NUMB(S)
1692          S=S-1
1693    C
1694    C-----CHECK IF ; PRESENT-----
1695    C
1696          CALL INC
1697          IF (END.EQ.1) GO TO 8990
1698          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 2495
```

```
1699  2492  TR=1
1700        GO TO 9000
1701  2495  I=I-1
1702        GO TO 9000
1703  C
1704  C-----WHILE MISSING CORRESPONDING TO THIS DO, STMNT IGNORED---------
1705  C
1706  2500   E=282
1707        GO TO 2425
1708  C
1709  C-----CHECK FOR KEYWORD BEGIN-----
1710  C
1711  3000  CALL INC
1712        IF (END.EQ.1) GO TO 8990
1713  3010  IF(.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.10))) GO TO 3025
1714  C
1715  C-----PUSH IN B-----
1716  C
1717        S=S+1
1718        STM(S)=1HB
1719  C
1720        SP=SP+1
1721        STNO(SP)=3020
1722        GO TO 2000
1723  C
1724  C
1725  C
1726  C-----CHECK FOR KEYWORD END-----
1727  C
1728  3020  CALL INC
1729        IF(END.EQ.1) GO TO 8990
1730  3025  IF(.NOT. ((TAB(I).EQ.3).AND.(INDEX(I).EQ.11)))GO TO 4010
1731  C
1732  C-----CHECK IF ; WAS PRESENT IN EARLIER STATEMENT-----
1733  C
1734        IF (TR.NE.1) GO TO 3030
1735  C
1736  C-----EXTRA ; PRESENT, NULL STATEMENT ASSUMED-----
1737  C
1738        E=284
1739        CALL EFILL
1740        TR=0
1741  3030  IF (STM(S).NE.1HB) GO TO 3039
1742  3035  IF (S.EQ.1) GO TO 9990
1743        S=S-1
1744  C
```

```
1745  C----CHECK IF ; PRESENT----
1746  C
1747        GO TO 5102
1748  C     CALL INC
1749  C     IF (END.EQ.1) GO TO 8990
1750  C     IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 3033
1751  C     TR=1
1752  C     GO TO 9000
1753  C3033 I=I-1
1754  C     GO TO 9000
1755  C
1756  C----KEYWORD BEGIN NOT PRESENT FOR THE PRESENT END-----
1757  C
1758  3039  IJK=1
1759  3040  IF (STM(S).NE.1HB) GO TO 3050
1760        GO TO (3035,2000),IJK
1761  3050  IF (STM(S).NE.1HT) GO TO 3060
1762        INT=INT+1
1763        OPC(INT)=5
1764        TAB1(INT)=1
1765        IND1(INT)=IDN+NUMB(S)
1766        GO TO 3070
1767  3060  IF (STM(S).NE.1HE) GO TO 3065
1768        INT=INT+1
1769        OPC(INT)=5
1770        TAB1(INT)=1
1771        IND1(INT)=IDN+64+NUMB(S)
1772        GO TO 3070
1773  3065  IF (STM(S).NE.1HW) GO TO 3045
1774        IF (S.EQ.1) GO TO 9990
1775        INT=INT+1
1776        OPC(INT)=7
1777        TAB1(INT)=1
1778        IND1(INT)=IDN+128+NUMB(S)
1779        S=S-1
1780        INT=INT+1
1781        OPC(INT)=5
1782        TAB1(INT)=1
1783        IND1(INT)=IDN+128+NUMB(S)
1784  3070  IF (S.EQ.1) GO TO 9990
1785        S=S-1
1786        SP=SP-1
1787        GO TO 3040
1788  C
1789  C----MISMATCHING END PRESENT----
1790  C
```

```
1791  3045  E=264
1792        CALL EFILL
1793        GO TO 5105
1794  C
1795  C-----CHECK FOR KEYWORD GOTO-----
1796  C
1797  4000  CALL INC
1798        IF(END.EQ.1) GO TO 8990
1799  4010  IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.17)))GO TO 5000
1800        TR=0
1801        CALL INC
1802        IF(END.EQ.1)GO TO 8990
1803        IF(TAB(I).NE.21)GO TO 4020
1804        DO 4012 LC=1,LBN
1805        IF (LAB(LC).EQ.ILIT(INDEX(I))) GO TO 4015
1806  4012  CONTINUE
1807        GO TO 4020
1808  4015  INT=INT+1
1809        OPC(INT)=7
1810        TAB1(INT)=24
1811        IND1(INT)=LC
1812        GO TO 5102
1813  C
1814  C-----CHECK IF ; PRESENT-----
1815  C
1816  C     CALL INC
1817  C     IF (END.EQ.1) GO TO 8990
1818  C     IF (.NOT.((TAB(I).EQ.1).AND.(INDEX(I).EQ.11))) GO TO 4017
1819  C     TR=1
1820  C     GO TO 2000
1821  C4017 I=I-1
1822  C     GO TO 2000
1823  C
1824  C-----INVALID LABLE IN GOTO, STMNT IGNORD-----
1825  C
1826  4020  E=266
1827  4025  CALL EFILL
1828        CALL IGNOR
1829        IF (TAB(I).EQ.3) GO TO 3025
1830        IF (TAB(I).EQ.4) GO TO 5105
1831  C
1832  C-----PROGRAM NAME & VARIABLES CANNOT BE USED AS LABLES, STMNT IGNORE
1833  C
1834  4030  E=268
1835        GO TO 4025
1836  5000  IF (TAB(I).NE.1) GO TO 6000
```

```
1837          INDEX1=INDEX(I)
1838    5040  CALL INC
1839          IF (END.EQ.1) GO TO 8990
1840    5043  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 5053
1841          CALL INC
1842          IF (END.EQ.1) GO TO 8990
1843          IF (.NOT.((TAB(I).EQ.1).OR. (TAB(I).EQ.21)))GO TO 5070
1844    5045  ARR=TAB(I)
1845          IND=INDEX(I)
1846          IF (TAB(I).NE.1) IND=ILIT(INDEX(I))
1847          CALL INC
1848          IF (END.EQ.1) GO TO 8990
1849          IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.9)) GO TO 5050
1850    C
1851    C-----  ) MISSING, STMNT IGNORED-----
1852    C
1853          E=102
1854          GO TO 5080
1855    5050  CALL INC
1856          IF (END.EQ.1) GO TO 8990
1857    5053  IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.10)) GO TO 5055
1858    C
1859    C-----  : MISSING, ASSUMED-----
1860    C
1861          E=274
1862          CALL EFILL
1863    5055  CALL INC
1864          IF (END.EQ.1) GO TO 8990
1865          IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.5)) GO TO 5090
1866    C
1867    C-----  = MISSING, ASSUMED-----
1868    C
1869          E=276
1870          CALL EFILL
1871          GO TO 5100
1872    C
1873    C-----ARRAY INDEX ILL DEFINED, STMNT IGNORED-----
1874    C
1875    5070  E=100
1876          TR=0
1877    5080  CALL EFILL
1878          CALL IGNOR
1879          IF (TAB(I).EQ.3) GO TO 3025
1880          IF (TAB(I).EQ.4) GO TO 5105
1881    C
1882    5090  CALL INC
```

```
1883          IF (END.EQ.1) GO TO 8990
1884    5100  TR=0
1885          CALL EXPR
1886          I=I-1
1887    C
1888          INT=INT+1
1889          OPC(INT)=50
1890          TAB1(INT)=1
1891          IND1(INT)=INDEX1
1892          ARR1(INT)=ARR
1893          ALEN1(INT)=IND
1894          TAB2(INT)=10
1895          IND2(INT)=1
1896    C
1897    C-----CHECK IF ; PRESENT-----
1898    C
1899    5102  CALL INC
1900          IF (END.EQ.1) GO TO 8990
1901          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 5110
1902    5105  TR=1
1903          IJK=2
1904          GO TO 3040
1905    5110  I=I-1
1906          GO TO 2000
1907    C
1908    C-----CHECK FOR READ, READLN, WRITE AND WRITELN-----
1909    C
1910    6000  IF (.NOT.((TAB(I).EQ.3).AND.((INDEX(I).GT.17).AND.(INDEX(I
1911         *.LT.22)))) GO TO 7000
1912          IOPCD=INDEX(I)-8
1913          CALL INC
1914          IF (END.EQ.1) GO TO 8990
1915          IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.8)) GO TO 6010
1916    C
1917    C-----  ( MISSING, ASSUMED-----
1918    C
1919          E=286
1920          CALL EFILL
1921    6010  CALL INC
1922          IF (END.EQ.1) GO TO 8990
1923          IF  (TAB(I).EQ.1) GO TO 6030
1924          GO TO 6020
1925    C
1926    C-----UNIDENTIFIED SYNTAX ERROR, STMNT IGNORED-----
1927    C
1928    6015  E=210
```

```
1929  6017  CALL EFILL
1930        CALL IGNOR
1931        IF (TAB(I).EQ.3) GO TO 3025
1932        IF (TAB(I).EQ.4) GO TO 5105
1933  6020  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.14))) GO TO 6015
1934        CALL INC
1935        IF (END.EQ.1) GO TO 8990
1936        IF (TAB(I).NE.22) GO TO 6050
1937  6030  INT=INT+1
1938        OPC(INT)=IOPCD
1939        IF((IOPCD.EQ.11).OR.(IOPCD.EQ.13))OPC(INT)=OPC(INT)-1
1940        TAB1(INT)=TAB(I)
1941        IND1(INT)=INDEX(I)
1942        CALL INC
1943        IF (END.EQ.1) GO TO 8990
1944        IF (TAB1(INT).NE.22) GO TO 6040
1945        IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.14))) GO TO 6050
1946        CALL INC
1947        IF (END.EQ.1) GO TO 8990
1948  6040  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.8))) GO TO 6070
1949        CALL INC
1950        IF (END.EQ.1) GO TO 8990
1951        IF (.NOT.((TAB(I).EQ.1).OR.(TAB(I).EQ.21))) GO TO 6050
1952        ARR1(INT)=TAB(I)
1953        ALEN1(INT)=INDEX(I)
1954        CALL INC
1955        IF (END.EQ.1) GO TO 8990
1956        IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.9)) GO TO 6060
1957  C
1958  C---- ) MISSING, ASSUMED----
1959  C
1960        E=288
1961        CALL EFILL
1962        GO TO 6060
1963  C
1964  C----ARRAY INDEX ILL-DEFINED, STMNT IGNORED----
1965  C
1966  6050  E=100
1967        GO TO 6017
1968  C
1969  C----CHECK FOR : ----
1970  C
1971  6060  CALL INC
1972        IF (END.EQ.1) GO TO 8990
1973  6070  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.10))) GO TO 6110
1974        CALL INC
```

```
1975          IF (TAB(I).NE.21) GO TO 6090
1976          TAB2(INT)=21
1977          IND2(INT)=ILIT(INDEX(I))
1978          CALL INC
1979          IF (END.EQ.1) GO TO 8990
1980          IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.10))) GO TO 6110
1981          CALL INC
1982          IF (TAB(I).NE.21) GO TO 6090
1983          ARR2(INT)=21
1984          ALEN2(INT)=ILIT(INDEX(I))
1985          GO TO 6100
1986    C
1987    C-----ERROR IN FORMAT, STMNT IGNORED-----
1988    C
1989    6090  E=290
1990          GO TO 6017
1991    C
1992    C-----CHECK FOR , -----
1993    C
1994    6100  CALL INC
1995          IF (END.EQ.1) GO TO 8990
1996    6110  IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.12))) GO TO 6130
1997          GO TO 6010
1998    C
1999    C-----CHECK FOR ) -----
2000    C
2001    6130  IF ((TAB(I).EQ.4).AND.(INDEX(I).EQ.9)) GO TO 6140
2002    C
2003    C-----UNIDENTIFIED SYNTAX ERROR-----
2004    C
2005          GO TO 6015
2006    6140  IF ((IOPCD.EQ.10).OR.(IOPCD.EQ.12)) GO TO 5102
2007          INT=INT+1
2008          OPC(INT)=IOPCD
2009          GO TO 5102
2010    C
2011    C-----STATEMENT NOT RECOGNISED, IGNORED-----
2012    C
2013    7000  E=278
2014          CALL EFILL
2015          CALL IGNOR
2016          IF (TAB(I).EQ.3) GO TO 5105
2017          IF (TAB(I).EQ.4) TR=1
2018          GO TO 2000
2019    8990  E=208
2020          CALL EFILL
```

```
2021          GO TO 9999
2022    9000  IF (SP.EQ.0) RETURN
2023          SP=SP-1
2024          IF (STNO(SP+1  ).EQ.2270) GO TO 2270
2025          IF (STNO(SP+1).EQ.2300) GO TO 2300
2026          IF (STNO(SP+1).EQ.2490) GO TO 2490
2027          IF (STNO(SP+1).EQ.3020) GO TO 3020
2028    C
2029    C-----POP OFF SHOULD NOT HAPPEN AS ONLY ONE BEGIN IS AVAILABLE-----
2030    C
2031    9990  CONTINUE
2032    9999  END
2033          SUBROUTINE INC,     INCREMENT U-FORM TABLE
2034          INTEGER I,UN,END
2035          INTEGER TAB(512),INDEX(512)
2036          INTEGER C
2037          COMMON /INC1/ I
2038          COMMON /FIN/ END,UN
2039          COMMON /CARD/C
2040          COMMON /UFORM/ TAB,INDEX
2041    5     I=I+1
2042          IF (I.GT.UN)END=1
2043          IF  (TAB(I).NE.99) GO TO 10
2044          C=INDEX(I)
2045          GO TO 5
2046    10    RETURN
2047          END
2048          SUBROUTINE IGNOR,    STATEMENT IGNORE
2049          INTEGER TAB(512),INDEX(512),I,C
2050          INTEGER UN,END
2051          COMMON /UFORM/ TAB,INDEX
2052          COMMON /CARD/ C
2053          COMMON /INC1/ I
2054          COMMON /FIN/ END,UN
2055    1     IF (.NOT.((TAB(I).EQ.4).AND.(INDEX(I).EQ.11))) GO TO 10
2056          IF (.NOT.((TAB(I).EQ.3).AND.(INDEX(I).EQ.11))) GO TO 10
2057    5     RETURN
2058    10    CALL INC
2059          IF (END.EQ.1) GO TO 5
2060          GO TO 1
2061          END
2062          SUBROUTINE EFILL,   FILL UP ERROR MESSAGE TABLE
2063          INTEGER STMNT(32),ECODE(32),ERRN,C,E
2064          COMMON /ERR/ STMNT,ECODE,ERRN,E
2065          COMMON /CARD/ C
2066          ERRN=ERRN+1
```

```
2067          STMNT(ERRN)=C
2068          ECODE(ERRN)=E
2069          END
2070    FTN4,L
2071          PROGRAM PRINT,5
2072    C-----IDENTIFIER TABLE-----
2073          INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
2074         *RADDR(256)
2075    C-----LITERAL TABLE-----
2076          INTEGER ILIT(64),CLIT(64)
2077          REAL RLIT(64)
2078    C
2079    C-----LABLE TABLE-----
2080    C
2081          INTEGER LAB(32),LABTYP(32)
2082    C-----UNIFORM SYMBOL TABLE-----
2083          INTEGER TAB(512),INDEX(512)
2084    C-----ERROR MESSAGE TABLE-----
2085          INTEGER STMNT(32),ECODE(32)
2086    C-----KEYWORD TABLE-----
2087          INTEGER KWORD(4,22)
2088    C-----DELIMITERS-----
2089          INTEGER DEL(14)
2090          INTEGER A(80),IDVP(4),IDVU(8),C
2091          INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
2092          REAL TEMP
2093          INTEGER UN,ERRN,E,I,END,INT,ILITN,RLITN,CLITN,IDN,LABN
2094          INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
2095    C
2096    C-----INTERMEDIATE CODE TABLE-----
2097    C
2098          INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
2099         *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
2100    C
2101    C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)-----
2102    C
2103          INTEGER TVAR(16),TPREC(16),TOP(16)
2104          INTEGER PREC,OP,COUNT,T
2105          INTEGER PGM(4),LST,PAGE,LINE
2106          COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAI
2107         *,LABTYP
2108          COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
2109          COMMON /ERR/STMNT,ECODE,ERRN,E
2110          COMMON /UFORM/ TAB,INDEX
2111          COMMON /CARD/ C
2112          COMMON /INC1/ I
```

```
      COMMON /FIN/ END,UN
      COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
      COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
      COMMON /PRNT/ PGM,LST,PAGE,LINE
C
C
C
C
      LST=9
C
C
C
      IF (TYPE(1).NE.4) GO TO 10
      DO 5 ICNT=1,4
      PGM(I)=ID(I,1)
5     CONTINUE
      GO TO 20
10    PGM(1)=2HPA
      PGM(2)=2HSC
      PGM(3)=2HAL
      PGM(4)=2H
C
C
C
20    CALL HEAD
C
C
      GO TO 110
15    LINE=LINE+3
      IF (LINE.GT.60) CALL HEAD
      WRITE(LST,30)ERRN
30    FORMAT(//,'     TOTAL ERROR(S) DETECTED : ',I2    )
      IF (ERRN.EQ.0) GO TO 9000
      LINE=LINE+6
      IF (LINE.GT.60) CALL HEAD
      WRITE(LST,40)
40    FORMAT(//,'     ERROR TABLE :',//)
      WRITE(LST,50)
50    FORMAT('     SR NO     LINE NO     ERROR     SR NO     LINE NO
     *     SR NO     LINE NO     ERROR',/)
      ICNT3=3
      ICNT2=1
      DO 90 ICNT=1,((ERRN+2)/3)
      LINE=LINE+1
      IF (LINE.GT.60) CALL HEAD
      WRITE(LST,60)((ICNT1,STMNT(ICNT1),ECODE(ICNT1)),ICNT1=ICNT2,I
```

```
 60     FORMAT(3(6X,I2,'.',7X,I4,6X,I3))
        ICNT2=ICNT3+1
        ICNT3=ICNT3+3
 90     CONTINUE
        GO TO 9000
 110    LINE=LINE+6
        IF (LINE.GT.60) CALL HEAD
        WRITE(LST,120)
 120    FORMAT(//,'     IDENTIFIER TABLE (TAB NO-1) :',//)
        LINE=LINE+2
        IF (LINE.GT.60) CALL HEAD
        WRITE(LST,130)
 130    FORMAT(44X,'ARRAY',4X,'LITERAL'                )
        WRITE(LST,140)
 140    FORMAT('     SR NO     IDENTIFIER     TYPE     BYTE      CODE
       *NTER             ',/)
        DO 150 ICNT=1,IDN
        LINE=LINE+1
        IF (LINE.GT.60)CALL HEAD
        WRITE(LST,145)(ICNT,(ID(ICNT2,ICNT),ICNT2=1,4),TYPE(ICNT)
       *BYTE(ICNT),LEN(ICNT),LPTR(ICNT)                )
 145    FORMAT(4X,I4,'.',6X,4A2,6X,I1,6X,I4,4X,I4,4X,I6       )
 150    CONTINUE
        LINE=LINE+7
        IF (LINE.GT.60) CALL HEAD
        IF (ILITN.EQ.0) GO TO 195
        WRITE(LST,160)
 160    FORMAT(//,'     INTEGER LITERAL TABLE (TAB NO-21) :',//)
        WRITE(LST,170)
 170    FORMAT(4('     SR NO              VALUE'),/)
        ICNT2=1
        ICNT3=4
        DO 190 ICNT=1,((ILITN+3)/4)
        LINE=LINE+1
        IF (LINE.GT.60) CALL HEAD
        WRITE(LST,180)((ICNT1,ILIT(ICNT1)),ICNT1=ICNT2,ICNT3)
 180    FORMAT(4(5X,I3,'.',7X,I10))
        ICNT2=ICNT3+1
        ICNT3=ICNT3+4
 190    CONTINUE
        LINE=LINE+7
        IF (LINE.GT.60) CALL HEAD
 195    IF (CLITN.EQ.0) GO TO 235
        WRITE(LST,200)
 200    FORMAT(//,'     CHARACTER LITERAL TABLE (TAB NO-22) :',//)
        WRITE(LST,210)
```

```
2205  210    FORMAT(4('     SR NO              VALUE'))
2206         ICNT2=1
2207         ICNT3=4
2208         DO 230 ICNT=1,((CLITN+3)/4)
2209         LINE=LINE+1
2210         IF (LINE.GT.60) CALL HEAD
2211         WRITE(LST,220) (ICNT1, CLIT(ICNT1),
2212        *ICNT1=ICNT2,ICNT3)
2213  220    FORMAT(4(5X,I3,'.',15X,1A2))
2214         ICNT2=ICNT3+1
2215         ICNT3=ICNT3+4
2216  230    CONTINUE
2217         LINE=LINE+7
2218         IF (LINE.GT.60) CALL HEAD
2219  235    IF (RLITN.EQ.0) GO TO 261
2220         WRITE(LST,240)
2221  240    FORMAT(//,'     REAL LITERAL TABLE (TAB NO-23) :',//)
2222         WRITE(LST,250)
2223  250    FORMAT(4('     SR NO              VALUE'))
2224         ICNT2=1
2225         ICNT3=4
2226         DO 260 ICNT=1,((RLITN+3)/4)
2227         LINE=LINE+1
2228         IF (LINE.GT.60) CALL HEAD
2229         WRITE(LST,255)(ICNT1,RLIT(ICNT1),ICNT1=ICNT2,ICNT3)
2230  255    FORMAT(4(5X,I3,'.',4X,E13.8))
2231         ICNT2=ICNT3+1
2232         ICNT3=ICNT3+4
2233  260    CONTINUE
2234         LINE=LINE+7
2235         IF (LINE.GT.60) CALL HEAD
2236  261    IF (LABN.EQ.0) GO TO 269
2237         WRITE(LST,263)
2238  263    FORMAT(//,'     LABEL TABLE (TAB NO-24) :',//)
2239         WRITE(LST,265)
2240  265    FORMAT(3('     SR NO       LABEL  STATUS'),/)
2241         ICNT2=1
2242         ICNT3=2
2243         DO 268 ICNT=1,((LABN+1)/2)
2244          WRITE(LST,267)(ICNT1,LAB(ICNT1),LABTYP(ICNT1),ICNT1=ICNT2,ICN
2245  267    FORMAT(2(5X,I3,'.',1X,I10,5X,I2))
2246         ICNT2=ICNT3+1
2247         ICNT3=ICNT3+2
2248         LINE=LINE+1
2249         IF (LINE.GT.59) CALL HEAD
2250  268    CONTINUE
```

```
2251          LINE=LINE+7
2252          IF (LINE.GT.60) CALL HEAD
2253   269    WRITE(LST,270)
2254   270    FORMAT(//,'     UNIFORM SYMBOL TABLE :',//)
2255          WRITE(LST,280)
2256   280    FORMAT(3('     SR NO     TAB NO     INDEX'),/)
2257          ICNT2=1
2258          ICNT3=3
2259          DO 300 ICNT=1,((UN+2)/3)
2260          LINE=LINE+1
2261          IF (LINE.GT.60) CALL HEAD
2262          WRITE(LST,290)(ICNT1,TAB(ICNT1),INDEX(ICNT1),ICNT1=ICNT2,ICNT
2263   290    FORMAT(3(5X,I3,'.',7X,I2,6X,I4))
2264          ICNT2=ICNT3+1
2265          ICNT3=ICNT3+3
2266   300    CONTINUE
2267          IF((LINE+10).GT.60) CALL HEAD
2268          WRITE(LST,310)
2269   310    FORMAT(//,'     INTERMEDIATE CODE TABLE :',//)
2270          WRITE(LST,315)
2271   315    FORMAT(22X,10 '-','OPERAND 1',10'-',6X,10'-','OPERAND 2',10'-'
2272          WRITE(LST,320)
2273   320    FORMAT('     SR NO     OP CD   . TAB1     IND1     ARR1     ALEN1   .
2274         *AB2     IND2     ARR2     ALEN2',/)
2275          LINE=LINE+8
2276          DO 340 ICNT=1,INT
2277          LINE=LINE+1
2278          IF (LINE.GT.60) CALL HEAD
2279          WRITE(LST,330)ICNT,OPC(ICNT),TAB1(ICNT),IND1(ICNT),ARR1(ICNT),
2280         *ALEN1(ICNT),TAB2(ICNT),IND2(ICNT),ARR2(ICNT),ALEN2(ICNT)
2281   330    FORMAT(5X,I3,'.',6X,I2,2X'.',3X,I2,5X,I4,6X,I2,6X,I2,3X,'..',
2282         *2X,I2,5X,I4,6X,I2,6X,I2)
2283   340    CONTINUE
2284          GO TO 15
2285   9000   END
2286          SUBROUTINE HEAD
2287   C-----IDENTIFIER TABLE-----
2288          INTEGER ID(4,256),TYPE(256),BYTE(256),LEN(256),LPTR(256),
2289         *RADDR(256)
2290   C-----LITERAL TABLE-----
2291          INTEGER ILIT(64),CLIT(64)
2292          REAL RLIT(64)
2293   C
2294   C-----LABLE TABLE-----
2295   C
2296          INTEGER LAB(32),LABTYP(32)
```

```
2297  C-----UNIFORM SYMBOL TABLE-----
2298        INTEGER TAB(512),INDEX(512)
2299  C-----ERROR MESSAGE TABLE-----
2300        INTEGER STMNT(32),ECODE(32)
2301  C-----KEYWORD TABLE-----
2302        INTEGER KWORD(4,22)
2303  C-----DELIMITERS-----
2304        INTEGER DEL(14)
2305        INTEGER A(80),IDVP(4),IDVU(8),C
2306        INTEGER FMT(5),LIT(80),DIGIT(10),ITEMP,FST,SCN,IBUF(40)
2307        REAL TEMP
2308        INTEGER UN,ERRN,E,I,END,INT,ILITN,RLITN,CLITN,IDN,LABN
2309        INTEGER STM(16),NUMB(16),S,B,W,EL,TH,COP(2)
2310  C
2311  C-----INTERMEDIATE CODE TABLE-----
2312  C
2313        INTEGER OPC(524),TAB1(524),IND1(524),ARR1(524),ALEN1(524)
2314       *,TAB2(524),IND2(524),ARR2(524),ALEN2(524)
2315  C
2316  C-----TEMPORARY VARIABLE STACK--(TAB CODE : 10)-----
2317  C
2318        INTEGER TVAR(16),TPREC(16),TOP(16)
2319        INTEGER PREC,OP,COUNT,T
2320        INTEGER PGM(4),LST,PAGE,LINE
2321        INTEGER IB(15)
2322        COMMON /IDLIT/ ID,TYPE,BYTE,LEN,LPTR,RADDR,ILIT,CLIT,RLIT,LAB
2323       *,LABTYP
2324        COMMON /INTCD/  OPC,TAB1,IND1,ARR1,ALEN1,TAB2,IND2,ARR2,ALEN2
2325        COMMON /ERR/STMNT,ECODE,ERRN,E
2326        COMMON /UFORM/ TAB,INDEX
2327        COMMON /CARD/ C
2328        COMMON /INC1/ I
2329        COMMON /FIN/ END,UN
2330        COMMON /STK/ STM,NUMB,S,B,W,EL,TH,COP
2331        COMMON /COUN/ IDN,ILITN,RLITN,CLITN,INT,LABN
2332        COMMON /PRNT/ PGM,LST,PAGE,LINE
2333  C
2334  C
2335  C
2336  C
2337  C
2338        CALL FTIME(IB)
2339        PAGE=PAGE+1
2340        IF (TYPE(1).NE.4) GO TO 30
2341        WRITE(LST,20)(IB(ICNT),ICNT=1,15),(ID(ICNT),ICNT=1,4),PAGE
2342  20    FORMAT(1H1,'     HP-PASCAL ',15X,15A2,10X,'PROGRAM NAME : ',4A2
```

```
2343          *15X,'PAGE : ',I4,//)
2344           GO TO 40
2345    30     WRITE(LST,35)(IB(ICNT),ICNT=1,15),PAGE
2346    35     FORMAT(1H1,'     HP-PASCAL ',15X,15A2,10X,'PROGRAM NAME : PASC
2347          *15X,'PAGE : ',I4,//)
2348    40     LINE=4
2349           END
```

APPENDIX G
AN EXAMLPE

```
 1   PROGRAM BSORT;
 2   (* THIS PROGRAM SORTS THE TEN INPUT INTEGERS IN ASCENDING ORDER*)
 3   LABEL 1234;
 4   VAR
 5      TEMP, TAG, IND, IND1 : INTEGER;
 6      TEMP : INTEGER;
 7      SER : ARRAY (1:10) OF INTEGER;
 8   BEGIN
 9      IND:=1;
10      WHILE IND(=10 DO
11                    BEGIN
12                     READ (SER(IND));
13                     IND:=IND+1
14                    END;
15   1234;
16     TAG:=0;
17     IND:=1;
18     WHILE IND( 10 DO
19                    BEGIN
20                     IND1:=IND+1;
21                     IF SER(IND) ) SER(IND1) THEN
22                                             BEGIN
23                                              TEMP:=SER(IND);
24                                              SER(IND):=SER(IND1);
25                                              SER(IND1):=TEMP;
26                                              IND:=IND+1;
27                                              TAG:=1
28                                             END
29                    END;
30       IF TAG>0 THEN GOTO 1234;
31   IND:=1;
32   WHILE IND(=10 DO
33                    BEGIN
34                      WRITE(SER(IND));
35                      IND:=IND+1
36                    END
37   END.
```

IDENTIFIER TABLE (TAB NO-1) :

SR NO	IDENTIFIER	TYPE	BYTE	ARRAY CODE	LITERAL POI NTER
1.	BSORT	4	0	0	0
2.	TEMP	1	2	0	0
3.	TAG	1	2	0	0
4.	IND	1	2	0	0
5.	IND1	1	2	0	0
6.	SER	1	2	10	0

INTEGER LITERAL TABLE (TAB NO-21) :

SR NO	VALUE	SR NO	VALUE	SR NO	VALUE	SR NO	VALUE
1.	1234	2.	1	3.	10	4.	0

LABEL TABLE (TAB NO-24) :

SR NO	LABEL	STATUS	SR NO	LABEL	STATUS	SR NO	LABEL	STATUS
1.	1234	1	2.	0	0			

UNIFORM SYMBOL TABLE :

SR NO	TAB NO	INDEX	SR NO	TAB NO	INDEX	SR NO	TAB NO	INDEX
1.	99	1	2.	3	1	3.	1	1
4.	4	11	5.	99	2	6.	99	3
7.	3	22	8.	21	1	9.	4	11
10.	99	4	11.	3	4	12.	99	5
13.	1	2	14.	4	12	15.	1	3
16.	4	12	17.	1	4	18.	4	12
19.	1	5	20.	4	10	21.	3	5
22.	4	11	23.	99	6	24.	1	2
25.	4	10	26.	3	5	27.	4	11
28.	99	7	29.	1	6	30.	4	10
31.	3	8	32.	4	8	33.	21	2
34.	4	10	35.	21	3	36.	4	9
37.	3	9	38.	3	5	39.	4	11
40.	99	8	41.	3	10	42.	99	9
43.	1	4	44.	4	10	45.	4	5
46.	21	2	47.	4	11	48.	99	10
49.	3	12	50.	1	4	51.	4	6
52.	4	5	53.	21	3	54.	3	13
55.	99	11	56.	3	10	57.	99	12

58.	3	18	59.	4	8	60.	1	6
61.	4	8	62.	1	4	63.	4	9
64.	4	9	65.	4	11	66.	99	13
67.	1	4	68.	4	10	69.	4	5
70.	1	4	71.	4	1	72.	21	2
73.	99	14	74.	3	11	75.	4	11
76.	99	15	77.	21	1	78.	4	10
79.	99	16	80.	1	3	81.	4	10
82.	4	5	83.	21	4	84.	4	11
85.	99	17	86.	1	4	87.	4	10
88.	4	5	89.	21	2	90.	4	11
91.	99	18	92.	3	12	93.	1	4
94.	4	6	95.	21	3	96.	3	13
97.	99	19	98.	3	10	99.	99	20
100.	1	5	101.	4	10	102.	4	5
103.	1	4	104.	4	1	105.	21	2
106.	4	11	107.	99	21	108.	3	14
109.	1	6	110.	4	8	111.	1	4
112.	4	9	113.	4	7	114.	1	6
115.	4	8	116.	1	5	117.	4	9
118.	3	15	119.	99	22	120.	3	10
121.	99	23	122.	1	2	123.	4	10
124.	4	5	125.	1	6	126.	4	8
127.	1	4	128.	4	9	129.	4	11
130.	99	24	131.	1	6	132.	4	8
133.	1	4	134.	4	9	135.	4	10
136.	4	5	137.	1	6	138.	4	8
139.	1	5	140.	4	9	141.	4	11
142.	99	25	143.	1	6	144.	4	8
145.	1	5	146.	4	9	147.	4	10
148.	4	5	149.	1	2	150.	4	11
151.	99	26	152.	1	4	153.	4	10
154.	4	5	155.	1	4	156.	4	1
157.	21	2	158.	4	11	159.	99	27
160.	1	3	161.	4	10	162.	4	5
163.	21	2	164.	99	28	165.	3	11
166.	99	29	167.	3	11	168.	4	11
169.	99	30	170.	3	14	171.	1	3
172.	4	7	173.	21	4	174.	3	15
175.	3	17	176.	21	1	177.	4	11
178.	99	31	179.	1	4	180.	4	10
181.	4	5	182.	21	2	183.	4	11
184.	99	32	185.	3	12	186.	1	4
187.	4	6	188.	4	5	189.	21	3
190.	3	13	191.	99	33	192.	3	10
193.	99	34	194.	3	20	195.	4	8
196.	1	6	197.	4	8	198.	1	4
199.	4	9	200.	4	9	201.	4	11
202.	99	35	203.	1	4	204.	4	10
205.	4	5	206.	1	4	207.	4	1
208.	21	2	209.	99	36	210.	3	11
211.	99	37	212.	3	11	213.	4	13

INTERMEDIATE CODE TABLE :

			OPERAND 1					OPERAND 2			
SR NO	OP CD	.	TAB1	IND1	ARR1	ALEN1	..	TAB2	IND2	ARR2	ALEN2
1.	0	.	1	1	0	0	..	0	0	0	0
2.	50	.	10	1	0	0	..	21	2	0	0
3.	50	.	1	4	0	0	..	10	1	0	0
4.	5	.	1	135	0	0	..	0	0	0	0
5.	50	.	10	1	0	0	..	1	4	0	0
6.	50	.	23	1	0	0	..	10	1	0	0
7.	50	.	10	1	0	0	..	21	3	0	0
8.	50	.	23	2	0	0	..	10	1	0	0
9.	6	.	23	1	0	0	..	23	2	0	0
10.	7	.	1	136	0	0	..	0	0	0	0
11.	7	.	1	136	0	0	..	0	0	0	0
12.	7	.	1	137	0	0	..	0	0	0	0
13.	5	.	1	136	0	0	..	0	0	0	0
14.	10	.	1	6	1	4	..	0	0	0	0
15.	50	.	10	1	0	0	..	1	4	0	0
16.	50	.	10	2	0	0	..	21	2	0	0
17.	1	.	10	1	0	0	..	10	2	0	0
18.	50	.	1	4	0	0	..	10	1	0	0
19.	7	.	1	135	0	0	..	0	0	0	0
20.	5	.	1	137	0	0	..	0	0	0	0
21.	5	.	24	1	0	0	..	0	0	0	0
22.	50	.	10	1	0	0	..	21	4	0	0
23.	50	.	1	3	0	0	..	10	1	0	0
24.	50	.	10	1	0	0	..	21	2	0	0
25.	50	.	1	4	0	0	..	10	1	0	0
26.	5	.	1	138	0	0	..	0	0	0	0
27.	50	.	10	1	0	0	..	1	4	0	0
28.	50	.	23	1	0	0	..	10	1	0	0
29.	50	.	10	1	0	0	..	21	3	0	0
30.	50	.	23	2	0	0	..	10	1	0	0
31.	6	.	23	1	0	0	..	23	2	0	0
32.	7	.	1	140	0	0	..	0	0	0	0
33.	7	.	1	139	0	0	..	0	0	0	0
34.	7	.	1	140	0	0	..	0	0	0	0
35.	5	.	1	139	0	0	..	0	0	0	0
36.	50	.	10	1	0	0	..	1	4	0	0
37.	50	.	10	2	0	0	..	21	2	0	0
38.	1	.	10	1	0	0	..	10	2	0	0
39.	50	.	1	5	0	0	..	10	1	0	0
40.	50	.	10	1	0	0	..	1	6	1	4
41.	50	.	23	1	0	0	..	10	1	0	0
42.	50	.	10	1	0	0	..	1	6	1	5
43.	50	.	23	2	0	0	..	10	1	0	0
44.	6	.	23	1	0	0	..	23	2	0	0
45.	7	.	1	8	0	0	..	0	0	0	0
46.	7	.	1	8	0	0	..	0	0	0	0
47.	7	.	1	7	0	0	..	0	0	0	0
48.	5	.	1	7	0	0	..	0	0	0	0

49.	50	.	10	1	0	0	..	1	6	1	4
50.	50	.	1	2	0	0	..	10	1	0	0
51.	50	.	10	1	0	0	..	1	6	1	5
52.	50	.	1	6	1	4	..	10	1	0	0
53.	50	.	10	1	0	0	..	1	2	0	0
54.	50	.	1	6	1	5	..	10	1	0	0
55.	50	.	10	1	0	0	..	1	4	0	0
56.	50	.	10	2	0	0	..	21	2	0	0
57.	1	.	10	1	0	0	..	10	2	0	0
58.	50	.	1	4	0	0	..	10	1	0	0
59.	50	.	10	1	0	0	..	21	2	0	0
60.	50	.	1	3	0	0	..	10	1	0	0
61.	5	.	1	8	0	0	..	0	0	0	0
62.	7	.	1	138	0	0	..	0	0	0	0
63.	5	.	1	140	0	0	..	0	0	0	0
64.	50	.	10	1	0	0	..	1	3	0	0
65.	50	.	23	1	0	0	..	10	1	0	0
66.	50	.	10	1	0	0	..	21	4	0	0
67.	50	.	23	2	0	0	..	10	1	0	0
68.	6	.	23	1	0	0	..	23	2	0	0
69.	7	.	1	10	0	0	..	0	0	0	0
70.	7	.	1	10	0	0	..	0	0	0	0
71.	7	.	1	9	0	0	..	0	0	0	0
72.	5	.	1	9	0	0	..	0	0	0	0
73.	7	.	24	1	0	0	..	0	0	0	0
74.	5	.	1	10	0	0	..	0	0	0	0
75.	50	.	10	1	0	0	..	21	2	0	0
76.	50	.	1	4	0	0	..	10	1	0	0
77.	5	.	1	141	0	0	..	0	0	0	0
78.	50	.	10	1	0	0	..	1	4	0	0
79.	50	.	23	1	0	0	..	10	1	0	0
80.	50	.	10	1	0	0	..	21	3	0	0
81.	50	.	23	2	0	0	..	10	1	0	0
82.	6	.	23	1	0	0	..	23	2	0	0
83.	7	.	1	142	0	0	..	0	0	0	0
84.	7	.	1	142	0	0	..	0	0	0	0
85.	7	.	1	143	0	0	..	0	0	0	0
86.	5	.	1	142	0	0	..	0	0	0	0
87.	12	.	1	6	1	4	..	0	0	0	0
88.	50	.	10	1	0	0	..	1	4	0	0
89.	50	.	10	2	0	0	..	21	2	0	0
90.	1	.	10	1	0	0	..	10	2	0	0
91.	50	.	1	4	0	0	..	10	1	0	0
92.	7	.	1	141	0	0	..	0	0	0	0
93.	5	.	1	143	0	0	..	0	0	0	0
94.	99	.	0	0	0	0	..	0	0	0	0

TOTAL ERROR(S) DETECTED : 1

ERROR TABLE :

SR NO	LINE NO	ERROR	SR NO	LINE NO	ERROR	SR NO	LINE NO	ERROR
1.	6	236	2.	0	0	3.	0	0