NBS TECHNICAL NOTE 849

U.S. DEPARTMENT OF COMMERCE / National Bureau of Standards

A FORTRAN Analyzer

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Institute for Computer Sciences and Technology, and the Office for Information Programs.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of a Center for Radiation Research, an Office of Measurement Services and the following divisions:

Applied Mathematics — Electricity — Mechanics — Heat — Optical Physics — Nuclear Sciences[2] — Applied Radiation[2] — Quantum Electronics[3] — Electromagnetics[3] — Time and Frequency[3] — Laboratory Astrophysics[3] — Cryogenics[3].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials and the following divisions:

Analytical Chemistry — Polymers — Metallurgy — Inorganic Materials — Reactor Radiation — Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services to promote the use of available technology and to facilitate technological innovation in industry and Government; cooperates with public and private organizations leading to the development of technological standards (including mandatory safety standards), codes and methods of test; and provides technical advice and services to Government agencies upon request. The Institute consists of a Center for Building Technology and the following divisions and offices:

Engineering and Product Standards — Weights and Measures — Invention and Innovation — Product Evaluation Technology — Electronic Technology — Technical Analysis — Measurement Engineering — Structures, Materials, and Life Safety[4] — Building Environment[4] — Technical Evaluation and Application[4] — Fire Technology.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus within the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Institute consists of the following divisions:

Computer Services — Systems and Software — Computer Systems Engineering — Information Technology.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS and other agencies of the Federal Government; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world. The Office consists of the following organizational units:

Office of Standard Reference Data — Office of Information Activities — Office of Technical Publications — Library — Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.
[2] Part of the Center for Radiation Research.
[3] Located at Boulder, Colorado 80302.
[4] Part of the Center for Building Technology.

A FORTRAN Analyzer

Gordon Lyon and
Rona B. Stillman

Institute for Computer Sciences and Technology
U.S. National Bureau of Standards
Washington, D.C. 20234

t. Technical note no. 849

Supported by

National Science Foundation
Washington, D.C. 20550

and

National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Frederick B. Dent, *Secretary*
NATIONAL BUREAU OF STANDARDS, Richard W. Roberts, *Director*

Issued October 1974

Library of Congress Catalog Card Number: 74-600144

National Bureau of Standards Technical Note 849
Nat. Bur. Stand. (U.S.), Tech. Note 849, 28 pages (Oct. 1974)
CODEN: NBTNAE

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1974

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
(Order by SD Catalog No. C13.46:849). Price 80 cents.

CONTENTS

TABLES AND FIGURES

A FORTRAN ANALYZER

Gordon Lyon and Rona B. Stillman

Details of a FORTRAN analysis package are presented. Examples illustrate a current operational level which gathers FORTRAN statement and frequency of execution statistics. Arguments support a simple technique for monitoring FORTRAN executions.

Key words: Computation and flow analysis; FORTRAN language use; programming aids; syntax analysis.

1. INTRODUCTION

A FORTRAN language analyzer was developed to lay ground work for a reasonable FORTRAN language utilization analysis package. Such a utility would provide performance records to managers and FORTRAN language development groups. Statistics would be available to measure processes of creation and maintenance of computer software. The package provides additional information to each user on his FORTRAN deck's pattern of execution; this service may be of some value in itself:

> Perhaps the most fundamental way to measure the speed of an algorithm in a reasonably machine-independent manner is to count how frequently each part...is executed [4].

The following terms are meaningful in the sequel:

Static analysis denotes a source statement examination applicable to some class of programs, and in particular, FORTRAN source programs. Static analysis does not pertain in any way to execution aspects.

Dynamic analysis denotes an analysis of program behavior during execution. No use is made of a hardware or system clock. For a good discussion of problems in elapsed-time measurements, see [1].

2. A PROGRAMMER'S AID

Among possible methods of gaining reliable information on FORTRAN usage, a program analyzer (a preprocessor) was selected as being convenient and effective [6]. A rudimentary static analyzer (based upon a FORTRAN precompiler written by Don Orser, Applied Mathematics Division, N.B.S.) has been available for several months. It collects from individual programs 118 frequency statistics on FORTRAN statement types, and accumulates these statistics over all programs analyzed. Such accumulations could be useful to language studies and FORTRAN standards committees, for system benchmark characterizations, and for studies in software portability and conversion.

Two observations motivated expansion of the static analyzer to perform dynamic analysis. First, a dynamic analyzer is a very useful programmer's tool. By identifying heavily used portions of code, it facilitates program optimization. In addition, untested segments of code are isolated, thereby promoting more thorough testing. Second, the dynamic analysis capability encourages widespread use of the whole analysis package, since static analysis is a prerequisite to dynamic analysis. This latter point is highly desirable, since meaningful statistics on FORTRAN usage can be obtained only with a broad and representative sample of programs.

A basic philosophy has been to keep the analyzer design simple. Instrumentation techniques described in Section 3 were developed in response to cumbersome methods frequently used in available commercial packages. Changes, extensions, and enhancements of the analyzer will be made as future experience dictates.

3. DYNAMIC ANALYSIS: A FORTRAN PROGRAM MONITOR

Execution activity of a FORTRAN program is treated here as a series of invocations of code segments. Each code segment is defined as a non-empty sequence $(S_k, S_{k+1},\ldots,S_{k+n})$ of statements where S_k is a unique entry statement in the sequence, and S_{k+n} the sole exit. Control flows from S_{k+j} to S_{k+j+1} for $0 \leq j < n$, i.e., is strictly sequential. A segment continues until a rule of ending or starting a new segment can apply. Certain statements always begin a new segment, e.g., labelled statements. Other statements always end a segment, e.g., unconditional GO TO statements, RETURNs, and STOPs. Some statements are themselves one or more segments. Among these are IF statements, computed GOTOs, and DO-loop terminators. The analyzer assigns segment numbers sequentially throughout the program text.

Dynamic analysis includes both computation and control activity analyses. In computation activity analysis, execution frequencies of each code segment are monitored independent of other program segments. Control activity analysis records flow from segment to segment as well.

There are many packages available which monitor FORTRAN program executions. Some are quite elaborate and expensive. Packages which analyze program structure (i.e., all except those which just interrupt execution at intervals and record the contents of the instruction counter) consist essentially of two phases. The original source code is first accepted as input to the analyzer, and an "instrumented" or augmented version of the program is produced as output. Instrumentation consists of inserting calls to a tallying function into the original source code. A second phase occurs when the augmented version is actually run, the tallying function causing the program to compute and report execution frequency statistics in addition to performing its normal function. The entire process is represented in Figure 1.

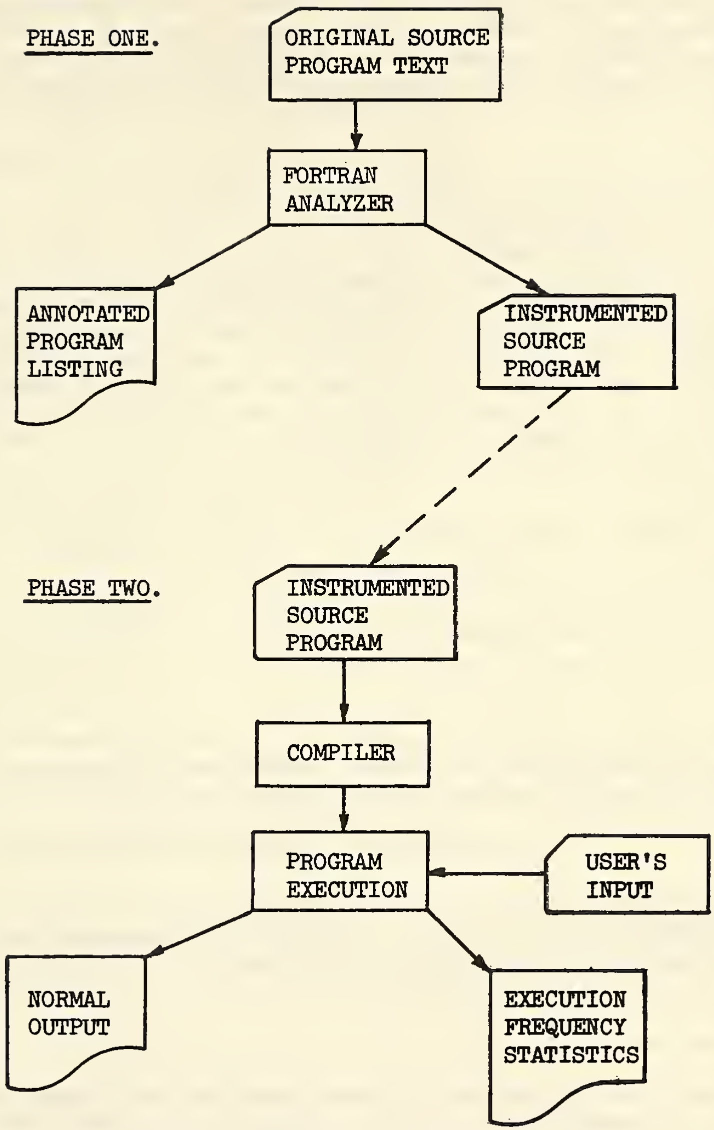


FIGURE 1. Dynamic Analysis

FORTRAN's lack of any compound-statement facility (such as BEGIN-END brackets in ALGOL) often has forced FORTRAN analyzers to be inelegant and unwieldy. Many dynamic analyzers introduce new variables and labels, and require a symbol table and (sometimes) multiple passes to avoid naming collisions. The analyzer presented here aims at simplicity, with some sacrifice in performance and analysis power (these weaknesses are detailed later). Instrumentation is a single pass operation. No new labels or variables are introduced into the source code; only one function name and a few additional statements are added.

The analyzer does change many statements into a "statistics collecting" form which calls an integer function with the reserved name IY2L9T (IY2L9T was chosen so as to avoid collision with usual FORTRAN names). The function IY2L9T, which is included as part of the instrumented program, maintains and prints execution frequency statistics. To avoid complicated interactions through COMMON and BLOCK DATA, statistics are kept in an array (SEGFRQ) which is local to IY2L9T. The dimension of SEGFRQ is fixed after a program has been analyzed, and is, of course, equal to the number of segments in the program. This approach implicitly assumes that local variables are invariant across calls to a subprogram, i.e. they have properties of ALGOL <u>own</u> variables or PL/I STATIC variables [5].

The monitoring philosophy is best explained by examples.

3.1 Simple Labelled (Executable) Statements

The labelled statement

```
xxxxx S
```

begins a new segment, say segment (i). S is any statement except an IF, a computed GOTO, a DO, or a STOP. (These latter cases are discussed below.) The instrumented version of xxxxx S is

```
xxxxx IF(IY2L9T(i,.TRUE.,0.D0,6).EQ.0)S
```

where IY2L9T returns zero and as a side effect, adds one to SEGFRQ(i), the counter for segment (i). The second and third parameters are not meaningful in this call. The fourth parameter controls the entry into IY2L9T.

3.2 Simple Logical IF

Let S be any FORTRAN statement which can occur as a consequent of a logical IF except the following: arithmetic IF, STOP, computed GOTO. Then a (possibly labelled) logical IF, [xxxxx] IF(boolean expression) S

is monitored as two segments. Segment (i) is the "decision portion" of the statement. Segment (i+1) is the consequent S. The statement following the logical IF begins segment (i+2). The execution frequency of segment (i) indicates how many times control reached the IF statement, while that of segment (i+1) indicates the number of times that the boolean expression evaluated to .TRUE. .

The instrumented version of a simple logical IF is

[xxxxx] IF(IY2L9T(i,boolean expression,0.D0,1).EQ.0) S

IY2L9T returns zero whenever boolean expression evaluates to .TRUE., and one otherwise. As a side effect, IY2L9T adds one to SEGFRQ(i) and, if boolean expression is .TRUE., increments SEGFRQ(i+1) by one as well. The third parameter has no meaning in this call. The fourth parameter effects a proper entry into IY2L9T.

3.3 Computed GOTO

The (possibly labelled) GOTO statement

[xxxxx] GOTO(n_1, n_2, ..., n_p), VAR

is monitored as (p+1) segments. Segment (i) is the "decision portion" of the statement. Segments (i+1) through (i+p) are transfers to n_1 through n_p, respectively. The instrumented statement appears below:

[xxxxx] IF(IY2L9T(i,.TRUE.,0.D0+VAR,3).EQ.0)GOTO(n_1,..,n_p),VAR

where IY2L9T returns zero. IY2L9T also increments SEGFRQ(i) and SEGFRQ(i+VAR). The second parameter is not meaningful in this call, and the last determines which parts of IY2L9T are used. The third actual argument is forced to double precision.

IY2L9T's third formal parameter is DOUBLE PRECISION, which allows FORTRAN values of greatest significance to be passed. A DOUBLE PRECISION zero (0.D0) is added to every actual argument in the third position to ensure compatibility between formal and actual arguments.

3.4 Logical IF with an Arithmetic IF as Consequent

The possibly labelled statement

[xxxxx] IF(boolean expression)IF(arithmetic
* expression)n_1,n_2,n_3

is treated as five segments: segment (i) is the decision portion of the logical IF, segment (i+1) is the decision portion of the arithmetic IF, and segments (i+2), (i+3), (i+4) are transfers to n_1 through n_3.

The instrumented statement calls IY2L9T twice:

```
[xxxxx] IF(IY2L9T(i,boolean expression,0.D0,4).EQ.0)
      * IF(IY2L9T(i+1,.TRUE.,0.D0+(arithmetic expression),
      * 2))n1,n2,n3
```

The first call to IY2L9T is similar to that described in Section 3.2. The second call returns -1, 0, or 1 determined by arithmetic expression being negative, zero, or positive. As a side effect, the second call increments SEGFRQ(i+1) and one of SEGFRQ(i+2) through SEGFRQ(i+4), as appropriate. The second parameter is not used, and the last controls entry into IY2L9T.

3.5 DO Statements

A labelled DO statement

```
xxxxx DO N INDEX=INIT,INCR,ITERM
```

is monitored as a single-statement segment, (i). The execution frequency of segment (i) indicates how many times the DO loop was entered. The statement following the DO, i.e., the first statement within the loop, always begins segment (i+1).

Because a DO statement cannot occur as the consequent of a logical IF, the DO must be monitored as a pair of statements:

```
xxxxx  DO N INDEX=INIT,INCR,ITERM
       IF(IY2L9T(i,(INDEX.EQ.INIT),0.D0,10).EQ.0)CONTINUE
```

IY2L9T always returns zero. SEGFRQ(i) is incremented whenever INDEX equals INIT. Clearly, this instrumentation is effective because neither control parameters INIT, INCR, ITERM nor control variable INDEX of a FORTRAN DO loop can be redefined within the loop; also every loop executes at least once. The third call parameter is not meaningful in this instance. The fourth effects a proper entry into IY2L9T.

3.6 DO Loop Terminators

DO statements always refer to labelled termination statements,

```
xxxxx S
```

where S is any executable statment except an arithmetic IF, RETURN, STOP, PAUSE, DO, any GOTO form, or logical IF with any of these as a consequent. Because of these restrictions, DO terminators need not be treated as a special case; they are instrumented routinely according to statement type (i.e., according to S). DO loop terminators, however, always end a segement. Thus DO loop terminators are single statement segments.

3.7 Implicit or Explicit STOP

Every possible halt statement is replaced by a call to IY2L9T which outputs the execution frequency results. For example, the instrumented version of an END statement is a pair of statements

```
IF(IY2L9T(i,.TRUE.,0.D0,9).EQ.0))STOP
END
```

where IY2L9T always returns zero and, as a side effect, prints execution frequencies. The second and third parameters are not meaningful in this call, and the fourth effects the proper entry into IY2L9T. END statements in BLOCK DATA are ignored, of course.

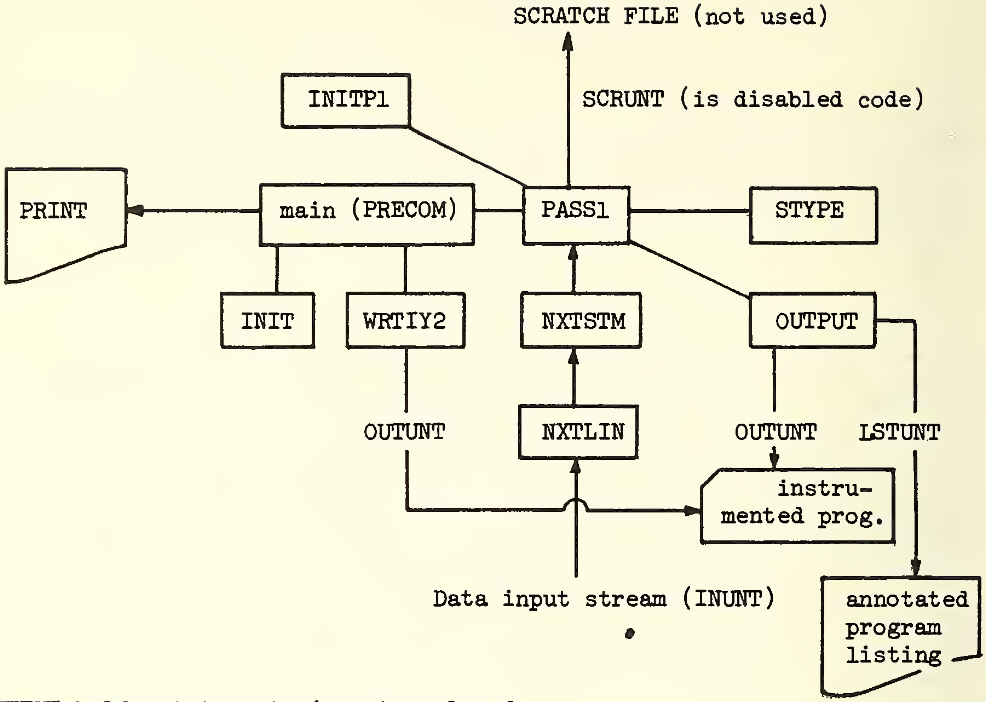
4. ANALYZER DETAILS

INIT initializes tables and logical units when it is called by main control (PRECOM--see Figure 2). PASS1 calls INITP1, and then begins processing input. Control returns to main only after the last statement has been processed. FORTRAN statements are picked up card by card through NXTLIN and NXTSTM. For each statement, nonessential blanks are removed and each character is converted to an internal code; results lie in an array ITEXT. With one character per word, represented in an internal form, some degree of machine independence can be realized. PASS1 then calls STYPE to identify the statement in ITEXT. PASS1 records the type returned by STYPE and calls OUTPUT to add the statement to the annotated program listing, and to add the instrumented version of the statement to the instrumented program file. PASS1 then initiates new calls through NXTSTM, etc.

4.1 Runs, Examples

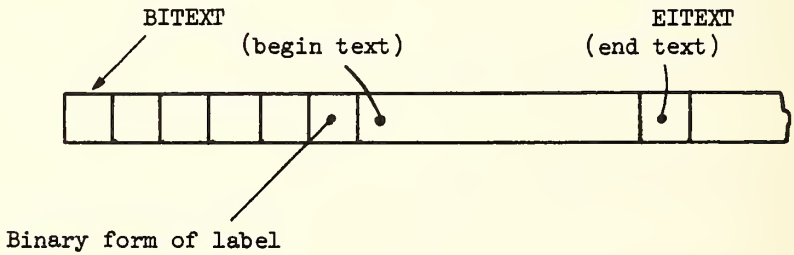
Data is read on INUNIT, output is printed on COMUNT and LSTUNT and punched on OUTUNT. (These variables are initialized in BLOCK DATA to indicate FORTRAN logical units 5, 6, 6, and 1, respectively.) Unit 10 functions as a system file to accumulate statistics on FORTRAN usage. It is read and written as one unformatted record which must be initialized to zero before attempting to accumulate meaningful statistics.

The analyzer develops and outputs an instrumented program as a stream of card images. For occasional use these images can be sent directly to a punch, but more often, as for large programs, the stream should be directed to a suitable on-line file. Details of the handling of the stream will vary considerably among different systems. For purposes of this note, a Univac 1108/EXEC-8 is assumed. In this environment, only very simple additions to subroutines OUTPUT and WRTIY2 are required to produce an on-line file rather than a card deck, and these are indicated clearly in the program text. In addition, some control cards must be added to the run stream. A typical <u>first</u> run, i.e., a run which initializes the cumulative statistics file 10, is depicted in Figure 3. Runs subsequent to the first need not initialize unit 10. For later runs, unit 10 is assigned [@ASG,A 10.] as



ETEXT holds statements in external code.
ITEXT holds internally coded statements.

ITEXT Format:



Statements in ETEXT are transformed to internal code and placed in ITEXT.

FIGURE 2.

Decks are listed below, with details.

INITP1	PASS1 initialization
NXTSTM	Get next statement
NXTLIN	Get next line
BLKDAT	BLOCK DATA
PASS1	Pass 1, static analysis statistics (no Pass 2 included), appropriate calls to OUTPUT.
STYPE	Statement classifier
PRINTF	Two copies used for output
ENCOD	Converts for symbol table (not presently used)
XERROR	Error handler, rather crude
* INTERN	Converts to internal code
POSID	Keyword checker for STYPE
INIT	Has rules for extending statement types
ITALLY	Used in recording static data types (Pass 1)
PRECOM	Program main control
* NEWPC	Text
* SPROC	Text also. This INCLUDE'd Cf. Note 1., below
TESTDAT	A bit of test data
* OUTPUT	Produces annotated program listing and instrumented program
ECONVT	Converts integer to Hollerith
* WRTIY2	Punches IY2L9T as last routine in instrumented program

---

* May need modification for machines other than 1108 with EXEC 8.

Note.

(1) INCLUDE statement is merely a copy-text command to the compiler. It is similar to a parameterless macro call, and is very easy to replace.

TABLE I.

a catalogued file.

The dynamic analysis function can be disabled by changing just one statement in subroutine PASS1. This is explained in the program source listing. An EXEC-8 setup for a static analysis run (only) is shown in Figure 4.

4.2 Results from the Analyzer: An Example

Figure 5-a is a listing of a FORTRAN program to be analyzed. The annotated source, identifying code segments, is presented in 5-b. Notice that the arithmetic IF in the middle of Figure 5-b is monitored as four segments, segment 7 being the expression evaluation, and 8 through 10 labelling the branches. Similarly, each logical IF is treated as two segments, a predicate and a consequent segment. Redundancies which appear in segmentations in Figure 5-b are a result of deliberate choices to keep low the effort in segmenting the program.

Figure 6 provides a static description of the program in 5-a. Each statement of original FORTRAN source code contributes to a collection of 118 FORTRAN statistics. Only non-zero tallies are actually printed. Observe that each logical IF is analyzed as two statements, one being the IF, and the other the consequent. Consequent statements resolve no closer than the entry CODE which is listed to the right of each statement type. Statements of similar nature share a code number.

Figure 7 is similar to 6, except that statistics are accumulative over all submissions since unit 10 was initialized.

A listing of the instrumented program, including function IY2L9T, is displayed in Figure 8. Execution of this program produced the matrix of segment execution frequencies shown in Figure 9. Each matrix row is labelled under heading "X" and specifies ten segments. The execution statistics of each segment within a decade appear in columns 0-9. (Code constituting each segment is defined via annotated listing, as in Figure 5-b.) Figure 9 says that segment 11 --the DO-loop entry-- was used once. Contrast this to ten passes for segment 12, a statement within the loop. Note also that segment 12, a decision portion of a logical IF, has frequency counts which equal the sum of those for segments 13 and 14, the potential branches from the IF.

5. Remarks and Restrictions

The analyzer was built with few frills. As a result, control activity analysis for assigned GOTOs is somewhat incomplete. Flow out of the GOTO is not recorded. In our opinion assigned GOTOs contribute little to the quality of a FORTRAN program and are, like ALTER in COBOL, best avoided.

Because no symbol table is maintained, the analyzer can not distinguish an arithmetic statement function definition

```
X(I,J,K) = I+J+K                (i)
```

from an array element assignment

```
X(I,J,K) = I+J+K               (ia)
```

Since statement function definitions should precede the first executable statement of a FORTRAN program unit, a special comment card

```
C+ *BEGIN EXECUTABLES*
```

could be used to trigger instrumentation of executable code in a unit of code. Some FORTRAN variants allow a more explicit statement function definition:

```
DEFINE X(I,J,K) = I+J+K      (ii)
```

Definitions of form (ii) are handled correctly.

Non-standard FORTRAN constructs which seemed inconvenient or difficult to handle were ignored. For example, the analyzer assumes that all calls to subprograms have standard returns; READ(...,END=...) and READ(...,ERR=...) are treated as simple READs; IF statements are not expected to contain Hollerith fields.

The analyzer expects valid American National Standard FORTRAN with mixed mode arithmetic, and with some exceptions, has very limited error handling and recovery. It is prudent, therefore, that programs be syntactically correct before submission.

No attempt has been made to minimize segmentation [4]. END statements are always monitored, even when preceded by a STOP or RETURN; decision portions of IFs and computed GOTOs are monitored as segments even when the statements are unlabelled; a DO loop terminator is always instrumented as a separate segment, even when there are no explicit transfers to it inside the loop. Whether reachable or not, STOPs are modified to call for a printout. Until more experience is gained with the analyzer, the significance of these points is uncertain. It should not be difficult to modify or extend the analyzer whenever appropriate.

Portability was an important goal in the analyzer's design. Streams of card images were chosen because they appeared most flexible. The analyzer expects pure FORTRAN source code submitted on punched cards. (The input cards or images should contain no job control language.) The instrumented program is a stream of card images. Machine independence was compromised only when common sense so dictated. COMMON declarations are INCLUDEd in subprograms, so that changes can be made uniformly by modifying COMMON text in just one place. The FORTRAN FLD [field selection] and READ(...,END=..,ERR=..,) are used as well, but sparingly.

6. CONCLUSION

Elements of an extant FORTRAN analyzer have been described. The analysis package will be useful to three distinct groups. Programmers will find the analyzer useful for improving their program's speed of execution and for finding some flow anomalies which might have been missed otherwise. Managers and systems people will know more about FORTRAN use in their installations. Finally, individuals charged with testing, certifying, or accepting software packages will have an additional tool to measure aspects of interest in reliability testing.

7. REFERENCES

[1] Gentleman, W.M., and Wichmann, B.A. Timing on computers. SIGARCH 2, (October, 1973), 20-23.

[2] Hoffmann, R.H. Automated Verification System User's Guide, TRW Note # 72-FMT-8 Project Appollo, Task MSC/TRW A-527,(January, 1972).

[3] Ingalls, D. The execution time profile as a programming tool. In R. Rustin (ed.), Compiler Optimization, 2nd Courant Computer Science Symposium (1970), Prentice-Hall, 1972, 107-128.

[4] Knuth, D.E., and Stevenson, F.R. Optimal measurement points for program frequency counts. B.I.T. 13, (1973), 313-322.

[5] Larmouth, J. Serious FORTRAN. Software-Practice and Experience 3, 2(April-June, 1973), 87-108.

[6] Lyon, G.E. Static language analysis. National Bureau of Standards (U.S.), Technical Note 797, (October, 1973).

[7] Stucki, L.G. Automatic generation of self-metric software. Proc., 1973 IEEE Symposium on Computer Software Reliability, New York, N.Y., April 30-May 2, 1973.

@RUN	Run command
@ASG,UP 10.,F///1	Assign and catalog file 10

@FOR,ISF
      INTEGER I(198)
      DATA I/198*0/
      WRITE(10) I
      REWIND 10
      STOP
      END
	Compile the FORTRAN program that follows. It will write a record of unformatted zeros on unit 10, a file. This is done only for the first run; subsequent analyzer results add to values stored in 10.

@MAP,ISF A,A	Link above compilation
@XQT A	Run above (write 10)
@ASG,A ANALYZER*FILE	Assign analyzer program file.
@ASG,T X.	Assign a temporary (card image file) named X.
@ASG,UP USER-NAMED*INSTRUFILE.	Assign a file (named by user) for the instrumented source program.
@USE Y.,USER-NAMED*INSTRUFILE.	Refer to it also as Y.
@BRKPT PUNCH$/X	Send card punch to file X.
@MAP,ISF B,B	Link analyzer routines.
IN ANALYZER*FILE.	Load object modules as indicated.
@XQT B	Run the analysis
.... subject program here	Analysis reads on unit 5, prints on 6, reads and writes on 10, writes file X via unit 1--the punch.
@EOF	End-of-file on unit 5
@ADD PRINTF @ADD PRINTF	PRINTF--need two copies for output
@BRKPT PUNCH$	Close breakpointed file X.
@ADD X.	File X. consists of card images including 1108/EXEC-8 control cards, which when ADDed write the instrumented source as a program file in Y.
@MAP,ISF C,C	Link instrumented version
IN Y.	Load object modules from Y.
@XQT C	Run instrumented version
.... test data	Execution frequencies print on unit 6
@FIN	End-of-job

FIGURE 3. Typical first run on 1108/EXEC-8

@RUN	Run command
@ASG,A 10.	Assign file 10.
@ASG,A MODIFIED*ANALYZER.	Assign modified analyzer program file
@MAP,ISF A,A	Link modified analyzer routines
IN MODIFIED*ANALYZER	Load object modules
@XQT A	Run modified analyzer
.... subject program here	Analysis reads on unit 5, prints on 6, reads and writes 10.
@EOF	
@ADD PRINTF.	
@ADD PRINTF.	Two copies of PRINTF.
@FIN	End-of-job

FIGURE 4. Typical static analysis-only run on EXEC-8

```
      INTEGER LENGTH,TABLE(100),ITEM,INDEX,TEMP
      READ(5,10)(LENGTH,(TABLE(I),I=1,LENGTH),ITEM)
   10 FORMAT(12I5)
      TEMP = LENGTH
      CALL FIND7(ITEM,INDEX,LENGTH,TABLE)
      IF(TEMP .NE. LENGTH) GOTO 100
      WRITE(6,11)(ITEM,INDEX)
   11 FORMAT(1H ,I5,8H IS THE ,I5,17HTH ENTRY IN TABLE)
      GOTO 200
  100 WRITE(6,101)ITEM
  101 FORMAT(1H ,I5,19H NOT FOUND IN TABLE)
  200 STOP
      END
      SUBROUTINE FIND7(ITEM,INDEX,LENGTH,TABLE)
C  FIND7 ATTEMPTS TO LOCATE ITEM IN TABLE,RETURNING ITS INDEX IN INDEX
C  IF SUCCESSFUL.  OTHERWISE, IT PLACES ITEM AT THE END OF TABLE,
C  INCREMENTS LENGTH, AND RETURNS THIS NEW VALUE ALSO AS INDEX.
      IMPLICIT INTEGER (A-Z)
      INTEGER TABLE(1)
      IF(LENGTH)300,200,100
  100 DO 150 I=1,LENGTH
      IF(TABLE(I) .NE. ITEM) GOTO 150
      INDEX = I
      RETURN
  150 CONTINUE
  200 LENGTH = LENGTH + 1
      IF(LENGTH .GT. 100) GOTO 400
      TABLE(LENGTH) = ITEM
      INDEX = LENGTH
      RETURN
  300 WRITE(6,301)
  301 FORMAT(1H ,32HNEGATIVE VALUE OF LENGTH ILLEGAL)
      RETURN
  400 WRITE(6,401)
  401 FORMAT(1H ,24HALLOCATED SPACE EXCEEDED)
      RETURN
      END
```

FIGURE 5a. The program to be analyzed

FORTRAN FREQUENCIES, OCT73, N1.

```
      INTEGER LENGTH,TABLE(100),ITEM,INDEX,TEMP
C*************************** SEGMENT      1 *****************************
      READ(5,10)(LENGTH,(TABLE(I),I=1,LENGTH),ITEM)
   10 FORMAT(12I5)
      TEMP = LENGTH
      CALL FIND7(ITEM,INDEX,LENGTH,TABLE)
C*********************** SEGMENTS     2 THRU     3 **********************
      IF(TEMP .NE. LENGTH) GOTO 100
C*************************** SEGMENT      4 *****************************
      WRITE(6,11)(ITEM,INDEX)
   11 FORMAT(1H ,I5,8H IS THE ,I5,17HTH ENTRY IN TABLE)
      GOTO 200
C*************************** SEGMENT      5 *****************************
  100 WRITE(6,101)ITEM
  101 FORMAT(1H ,I5,19H NOT FOUND IN TABLE)
C*************************** SEGMENT      6 *****************************
  200 STOP
      END

END PASS1, VALUE=      13
      SUBROUTINE FIND7(ITEM,INDEX,LENGTH,TABLE)
C  FIND7 ATTEMPTS TO LOCATE ITEM IN TABLE,RETURNING ITS INDEX IN INDEX
C  IF SUCCESSFUL.  OTHERWISE, IT PLACES ITEM AT THE END OF TABLE,
C  INCREMENTS LENGTH, AND RETURNS THIS NEW VALUE ALSO AS INDEX.
      IMPLICIT INTEGER (A-Z)
      INTEGER TABLE(1)
C*********************** SEGMENTS     7 THRU    10 **********************
      IF(LENGTH)300,200,100
C*************************** SEGMENT     11 *****************************
  100 DO 150 I=1,LENGTH
C*********************** SEGMENTS    12 THRU    13 **********************
      IF(TABLE(I) .NE. ITEM) GOTO 150
C*************************** SEGMENT     14 *****************************
      INDEX = I
      RETURN
C*************************** SEGMENT     15 *****************************
  150 CONTINUE
C*************************** SEGMENT     16 *****************************
  200 LENGTH = LENGTH + 1
C*********************** SEGMENTS    17 THRU    18 **********************
      IF(LENGTH .GT. 100) GOTO 400
C*************************** SEGMENT     19 *****************************
      TABLE(LENGTH) = ITEM
      INDEX = LENGTH
      RETURN
C*************************** SEGMENT     20 *****************************
  300 WRITE(6,301)
  301 FORMAT(1H ,32HNEGATIVE VALUE OF LENGTH ILLEGAL)
      RETURN
C*************************** SEGMENT     21 *****************************
  400 WRITE(6,401)
  401 FORMAT(1H ,24HALLOCATED SPACE EXCEEDED)
      RETURN
      END
```

FIGURE 5b. The annotated source listing

```
LOCAL ANALYSIS--

         COUNT                                                                      CODE
  .          4 OCCURRENCES OF      <ASSIGNMENT>  X=...                               36
  .          I OCCURRENCES OF      <ASSIGNMENT>  X(..)=..                            36
  .          1 OCCURRENCES OF      CALL X(...)                                       38
  .          1 OCCURRENCES OF      CONTINUE                                          55
  .          1 OCCURRENCES OF      DO                                                37
  .          2 OCCURRENCES OF      END                                               56
  .          5 OCCURRENCES OF      FORMAT                                            41
  .          4 OCCURRENCES OF      GOTO X                                            59
  .          3 OCCURRENCES OF      IF(<BE>)<STATEMENT>  ALSO IF(<SWITCH>)...         45
                       <STATEMENT> DATA FOR BOOLEAN IF ARE--
                               3 OCCURRENCES OF CODE 59 <STATEMENT>S.
                               0 ILL FORMED <STATEMENT>S AND
                               0 IMPROPER <STATEMENT>S.
  .          I OCCURRENCES OF      IF(<ARITH.EXPRESS.>) N1,N2,...                    49
  .          1 OCCURRENCES OF      IMPLICIT (..)                                     21
  .          I OCCURRENCES OF      INTEGER X,                                        26
  .          1 OCCURRENCES OF      INTEGER X(..)                                     26
  .          I OCCURRENCES OF      READ (..)(                                        43
  .          4 OCCURRENCES OF      RETURN                                            6I
  .          I OCCURRENCES OF      STOP                                              65
  .          I OCCURRENCES OF      SUBROUTINE X(..)                                  01
  .          I OCCURRENCES OF      WRITE  (..)(                                      42
  .          3 OCCURRENCES OF      WRITE  (..)                                       42
  .          3 OCCURRENCES OF      COMMENTS
  .         40 OCCURRENCES OF      STATEMENTS
  .         12 OCCURRENCES OF      LABELED STATEMENTS
```

FIGURE 6. Static analysis results

```
TOTAL ANALYSES--

     COUNT                                                                  CODE
         7 OCCURRENCES OF      ACCEPT X                                      46
         7 OCCURRENCES OF      ACCEPT X,                                     46
      5191 OCCURRENCES OF      <ASSIGNMENT>  X=...                           36
         7 OCCURRENCES OF      <ASSIGNMENT>  009..=..                        36
         7 OCCURRENCES OF      <ASSIGNMENT>  009..='...                      36
         7 OCCURRENCES OF      <ASSIGNMENT>  009..=16H...                    36
         7 OCCURRENCES OF      <ASSIGNMENT>  001..=A+1H,                     36
       858 OCCURRENCES OF      <ASSIGNMENT>  X(..)=..                        36
         7 OCCURRENCES OF      ASSIGN                                        51
         7 OCCURRENCES OF      BACKSPACE                                     53
       103 OCCURRENCES OF      CALL X                                        38
        20 OCCURRENCES OF      CALL X(...)                                   38
         7 OCCURRENCES OF      COMMON X                                      10
        31 OCCURRENCES OF      COMMON X,...                                  10
         7 OCCURRENCES OF      COMMON X(..),...                              10
         7 OCCURRENCES OF      COMMON X(..)                                  10
         7 OCCURRENCES OF      COMMON /../                                   10
         7 OCCURRENCES OF      COMPLEX X                                     22
         7 OCCURRENCES OF      COMPLEX X,...                                 22
         7 OCCURRENCES OF      COMPLEX X(..),...                             22
         7 OCCURRENCES OF      COMPLEX X(..)                                 22
      1046 OCCURRENCES OF      CONTINUE                                      55
       814 OCCURRENCES OF      DATA /../                                     11
         7 OCCURRENCES OF      DATA (..)/                                    11
        23 OCCURRENCES OF      DATA X,...                                    11
         7 OCCURRENCES OF      DECODE (...)(                                 17
        14 OCCURRENCES OF      DEFINE ..=..                                  16
        35 OCCURRENCES OF      DIMENSION X(..),...                           18
         7 OCCURRENCES OF      DIMENSION X(..)                               18
       445 OCCURRENCES OF      DO                                            37
         7 OCCURRENCES OF      DOUBLE COMPLEX X                              24
         7 OCCURRENCES OF      DOUBLE COMPLEX X,                             24
         7 OCCURRENCES OF      DOUBLE COMPLEX X(..),...                      24
         7 OCCURRENCES OF      DOUBLE COMPLEX X(..)                          24
         7 OCCURRENCES OF      DOUBLE PRECISION X                            23
         7 OCCURRENCES OF      DOUBLE PRECISION X,                           23
         7 OCCURRENCES OF      DOUBLE PRECISION X(..),...                    23
         7 OCCURRENCES OF      DOUBLE PRECISION X(..)                        23
         7 OCCURRENCES OF      ENCODE (..)(                                  19
        65 OCCURRENCES OF      END                                           56
         7 OCCURRENCES OF      ENTRY X                                       39
         7 OCCURRENCES OF      ENTRY X(...)                                  39
         7 OCCURRENCES OF      ENDFILE                                       58
        31 OCCURRENCES OF      EQUIVALENCE(..),...                           20
         7 OCCURRENCES OF      EQUIVALENCE(..)                               39
         7 OCCURRENCES OF      EXTERNAL X                                    25
         7 OCCURRENCES OF      EXTERNAL X,                                   25
      1033 OCCURRENCES OF      FORMAT                                        41
         7 OCCURRENCES OF      <FUNCTION> COMPLEX                            02
         7 OCCURRENCES OF      <FUNCTION> DBL PREC                           03
         7 OCCURRENCES OF      <FUNCTION> DBL COMPLX                         04
         9 OCCURRENCES OF      <FUNCTION> INTEGER                            05
        14 OCCURRENCES OF      <FUNCTION> LOGICAL                            06
         7 OCCURRENCES OF      <FUNCTION> REAL                               07
```

FIGURE 7. Cumulative static results

```
  8 OCCURRENCES OF      BLOCK DATA                                              08
 41 OCCURRENCES OF      GOTO(...),I                                             40
  7 OCCURRENCES OF      GOTO I,(...)                                            47
2274 OCCURRENCES OF     GOTO X                                                  59
2415 OCCURRENCES OF     IF(<BE>)<STATEMENT>  ALSO IF(<SWITCH>)...               45
          <STATEMENT> DATA FOR BOOLEAN IF ARE--
                 711 OCCURRENCES OF CODE 36 <STATEMENT>S.
                  36 OCCURRENCES OF CODE 38 <STATEMENT>S.
                 156 OCCURRENCES OF CODE 42 <STATEMENT>S.
                   7 OCCURRENCES OF CODE 44 <STATEMENT>S.
                1505 OCCURRENCES OF CODE 59 <STATEMENT>S.
                   0 ILL FORMED <STATEMENT>S AND
                   0 IMPROPER <STATEMENT>S.
  7 OCCURRENCES OF      IF(<ARITH.EXPRESS.>),N1,N2,...                          44
  9 OCCURRENCES OF      IF(<ARITH.EXPRESS.>) N1,N2,...                          44
  7 OCCURRENCES OF      IMPLICIT I..),...                                       21
  9 OCCURRENCES OF      IMPLICIT I..)                                           21
  2 OCCURRENCES OF      INTEGER X                                               26
 96 OCCURRENCES OF      INTEGER X,                                              26
 62 OCCURRENCES OF      INTEGER XI..),                                          26
 25 OCCURRENCES OF      INTEGER XI..)                                           26
 16 OCCURRENCES OF      LOGICAL X                                               27
 15 OCCURRENCES OF      LOGICAL X,                                              27
  7 OCCURRENCES OF      LOGICAL X(..),                                          27
  7 OCCURRENCES OF      LOGICAL XI..)                                           27
  7 OCCURRENCES OF      NAMELIST                                                12
  7 OCCURRENCES OF      PARAMETER                                               15
 14 OCCURRENCES OF      PAUSE                                                   60
  7 OCCURRENCES OF      PRINT X                                                 48
  7 OCCURRENCES OF      PRINT X,                                                48
  7 OCCURRENCES OF      PUNCH X                                                 49
  7 OCCURRENCES OF      PUNCH X,                                                49
  7 OCCURRENCES OF      READ  X                                                 43
  7 OCCURRENCES OF      READ  X,                                                43
 45 OCCURRENCES OF      READ (..)I                                              43
 55 OCCURRENCES OF      READ (..)                                               43
  7 OCCURRENCES OF      REAL X                                                  28
 19 OCCURRENCES OF      REAL X,                                                 28
  7 OCCURRENCES OF      REAL XI..I,                                             28
  7 OCCURRENCES OF      REAL X(..)                                              28
 52 OCCURRENCES OF      RETURN                                                  61
  7 OCCURRENCES OF      REWIND                                                  62
  7 OCCURRENCES OF      SKIP FILE                                               64
  7 OCCURRENCES OF      SKIP RECORD                                             63
 39 OCCURRENCES OF      STOP                                                    65
 19 OCCURRENCES OF      SUBROUTINE X                                            01
 18 OCCURRENCES OF      SUBROUTINE XI..)                                        01
  7 OCCURRENCES OF      TYPE X                                                  50
  7 OCCURRENCES OF      TYPE X,                                                 50
  7 OCCURRENCES OF      UNLOAD                                                  66
 97 OCCURRENCES OF      WRITE  (..)(                                            42
989 OCCURRENCES OF      WRITE  (..)                                             42
2365 OCCURRENCES OF     COMMENTS
18935 OCCURRENCES OF    STATEMENTS
3094 OCCURRENCES OF     LABELED STATEMENTS
 51 OCCURRENCES OF      FUNCTIONS DEFINED
  3 OCCURRENCES OF      STATEMENTS IN ERROR OR OF TYPE UNKNOWN
  7 OCCURRENCES OF      ABNORMAL X                                              31
```

FIGURE 7. (continuation)

```
.       7 OCCURRENCES OF       ABNORMAL X,                                 31
.       7 OCCURRENCES OF       ABNORMAL                                    31
.       7 OCCURRENCES OF       DELETE N                                    59
.      14 OCCURRENCES OF       DELETE N,L                                  59
.      19 OCCURRENCES OF       INCLUDE X                                   52
.      12 OCCURRENCES OF       INCLUDE X,LIST                              52
.       7 OCCURRENCES OF       DECODE( ),                                  17
.       7 OCCURRENCES OF       DECODE( )                                   17
.       7 OCCURRENCES OF       ENCODE( ),                                  19
.       7 OCCURRENCES OF       ENCODE( )                                   19

**NORMAL TERMINATION.
```

FIGURE 7. (continuation)

```
TECH*EXAMPLE.ELT00002
              SUBROUTINE FIND7(ITEM,INDEX,LENGTH,TABLE)
              INTEGER IY2L9T
              IMPLICIT INTEGER (A-Z)
              INTEGER TABLE(1)
              IF(IY2L9T(    7,.TRUE.,0.D0+(LENGTH),2))300,200,100
          100 DO 150 I=1,LENGTH
              IF(IY2L9T(   11,I.EQ.1,0.D0,10).EQ.0)CONTINUE
              IF(IY2L9T(   12,TABLE(I).NE.ITEM,0.D0, 1).EQ.0)GOTO150
              IF(IY2L9T(   14,.TRUE.,0.D0, 6).EQ.0)INDEX=I
              RETURN
          150 IF(IY2L9T(   15,.TRUE.,0.D0, 6).EQ.0)CONTINUE
          200 IF(IY2L9T(   16,.TRUE.,0.D0, 6).EQ.0)LENGTH=LENGTH+1
              IF(IY2L9T(   17,LENGTH.GT.100,0.D0, 1).EQ.0)GOTO400
              IF(IY2L9T(   19,.TRUE.,0.D0, 6).EQ.0)TABLE(LENGTH)=ITEM
              INDEX = LENGTH
              RETURN
          300 IF(IY2L9T(   20,.TRUE.,0.D0, 6).EQ.0)WRITE(6,301)
          301 FORMAT(1H ,32HNEGATIVE VALUE OF LENGTH ILLEGAL)
              RETURN
          400 IF(IY2L9T(   21,.TRUE.,0.D0, 6).EQ.0)WRITE(6,401)
          401 FORMAT(1H ,24HALLOCATED SPACE EXCEEDED)
              RETURN
              IF(IY2L9T(   21,.TRUE.,0.D0, 9).EQ.0)STOP
              END

TECH*EXAMPLE.ELT00001
              INTEGER LENGTH,TABLE(100),ITEM,INDEX,TEMP
              INTEGER IY2L9T
              IF(IY2L9T(    1,.TRUE.,0.D0, 6).EQ.0)READ(5,10)LENGTH,(TABLE(I),I=
             *1,LENGTH),ITEM
           10 FORMAT(12I5)
              TEMP = LENGTH
              CALL FIND7(ITEM,INDEX,LENGTH,TABLE)
              IF(IY2L9T(    2,TEMP.NE.LENGTH,0.D0, 1).EQ.0)GOTO100
              IF(IY2L9T(    4,.TRUE.,0.D0, 6).EQ.0)WRITE(6,11)(ITEM,INDEX)
           11 FORMAT(1H ,I5,8H IS THE ,I5,17HTH ENTRY IN TABLE)
              GOTO 200
          100 IF(IY2L9T(    5,.TRUE.,0.D0, 6).EQ.0)WRITE(6,101)ITEM
          101 FORMAT(1H ,I5,19H NOT FOUND IN TABLE)
          200 IF(IY2L9T(    6,.TRUE.,0.D0, 8).EQ.0)STOP
              IF(IY2L9T(    6,.TRUE.,0.D0, 9).EQ.0)STOP
              END
```

FIGURE 8. Additional instrumented-program elements

```
TECH*EXAMPLE.ELT IY2L9T
      INTEGER FUNCTION IY2L9T(SEGNO,BE,AE,ENT)
      INTEGER SEGNO,ENT,IAE,SEGFRQ(   21)
      DOUBLE PRECISION AE
      LOGICAL BE
      DATA SEGFRQ/   21*0/
      INTEGER NLINES,NPAGES,PTPAGE,LPERPG,MULT,ENDLIN
      DATA LPERPG/50/
      SEGFRQ(SEGNO)=SEGFRQ(SEGNO)+1
      GOTO(10,20,30,40,50,60,70,80,90,100),ENT
   10 IF(BE)GOTO 11
      IY2L9T=1
      RETURN
   11 SEGFRQ(SEGNO+1)=SEGFRQ(SEGNO+1)+1
      IY2L9T=0
      RETURN
   20 IF(AE)21,22,23
   21 SEGFRQ(SEGNO+1)=SEGFRQ(SEGNO+1)+1
      IY2L9T=-1
      RETURN
   22 SEGFRQ(SEGNO+2)=SEGFRQ(SEGNO+2)+1
      IY2L9T=0
      RETURN
   23 SEGFRQ(SEGNO+3)=SEGFRQ(SEGNO+3)+1
      IY2L9T=1
      RETURN
   30 IAE=AE+DSIGN(.2D0,AE)
      SEGFRQ(SEGNO+IAE)=SEGFRQ(SEGNO+IAE)+1
      IY2L9T=0
      RETURN
   40 IF(BE)GOTO 41
      IY2L9T=1
      RETURN
   41 IY2L9T=0
      RETURN
   50 IF(BE)GOTO 51
      IY2L9T=1
      RETURN
   51 SEGFRQ(SEGNO+1)=SEGFRQ(SEGNO+1)+1
      IAE=AE+DSIGN(.2D0,AE)
      SEGFRQ(SEGNO+1+IAE)=SEGFRQ(SEGNO+1+IAE)+1
      IY2L9T=0
      RETURN
   60 IY2L9T=0
      RETURN
   70 IF(BE)GOTO 71
      IY2L9T=1
      RETURN
   71 SEGFRQ(SEGNO+1)=SEGFRQ(SEGNO+1)+1
   80 IY2L9T=0
      NLINES=   21/10+1
      PTPAGE=NLINES-(NLINES/LPERPG)*LPERPG
      IF(PTPAGE.NE.0)PTPAGE=1
      NPAGES=(NLINES/LPERPG)+PTPAGE
      MULT=0
      DO 88 IPAGE=1,NPAGES
      WRITE( 6,81)
   81 FORMAT(1H1,40X,29HSEGMENT EXECUTION FREQUENCIES,/,/)
      WRITE( 6,82)(I,I=1,9)
   82 FORMAT(1H ,4X,1HX,5X,7H      0,9(2X,I7),/)
      DO 87 ILINE=1,LPERPG
      IF(MULT.GT.   21)GOTO 89
      IF(MULT.NE.0)GOTO 84
      WRITE( 6,83)(MULT,(SEGFRQ(I),I=1,9))
   83 FORMAT(1H ,I5,5X,7H-------,9(2X,I7))
      GOTO 86
   84 ENDLIN=MULT+9
      IF(ENDLIN.GT.   21)ENDLIN=   21
      WRITE( 6,85)(MULT,(SEGFRQ(I),I=MULT,ENDLIN))
   85 FORMAT(1H ,I5,3X,10(2X,I7))
   86 MULT=MULT+10
   87 CONTINUE
   88 CONTINUE
   89 RETURN
   90 SEGFRQ(SEGNO)=SEGFRQ(SEGNO)-1
      GOTO 80
  100 IF(.NOT.BE)SEGFRQ(SEGNO)=SEGFRQ(SEGNO)-1
      IY2L9T=0
      RETURN
      END
```

FIGURE 8. (continuation)

SEGMENT EXECUTION FREQUENCIES

X	0	1	2	3	4	5	6	7	8	9
0	-------	1	1	1	0	1	1	1	0	0
10	1	1	10	10	0	10	1	1	0	1
20	0	0								

FIGURE 9. Dynamic analysis results

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS TN-849	2. Gov't Accession No.	3. Recipient's Accession No.
4. TITLE AND SUBTITLE	A FORTRAN ANALYZER		5. Publication Date October 1974
			6. Performing Organization Code
7. AUTHOR(S)	Gordon Lyon & Rona B. Stillman		8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS	NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, D.C. 20234		10. Project/Task/Work Unit No. 640-1125
			11. Contract/Grant No. AG-350
12. Sponsoring Organization Name and Complete Address (Street, City, State, ZIP)	Institute for Computer Science and Technology National Bureau of Standards		13. Type of Report & Period Covered
			14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 74-600144

16. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)

Details of a FORTRAN analysis package are presented. Examples illustrate a current operational level which gathers FORTRAN statement and frequency-of-execution statistics. Arguments support a simple technique for monitoring FORTRAN executions.

17. KEY WORDS (six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)

Computation and flow analysis; FORTRAN language use; programming aids; syntax analysis.

18. AVAILABILITY	19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
[xx] Unlimited	UNCLASSIFIED	28
[ ] For Official Distribution. Do Not Release to NTIS		
[x] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13. 46:849	20. SECURITY CLASS (THIS PAGE)	22. Price
[ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151	UNCLASSIFIED	80 cents

USCOMM-DC 29042-P74

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH reports National Bureau of Standards research and development in physics, mathematics, and chemistry. Comprehensive scientific papers give complete details of the work, including laboratory data, experimental procedures, and theoretical and mathematical analyses. Illustrated with photographs, drawings, and charts. Includes listings of other NBS papers as issued.

Published in two sections, available separately:

- Physics and Chemistry (Section A)

Papers of interest primarily to scientists working in these fields. This section covers a broad range of physical and chemical research, with major emphasis on standards of physical measurement, fundamental constants, and properties of matter. Issued six times a year. Annual subscription: Domestic, $17.00; Foreign, $21.25.

- Mathematical Sciences (Section B)

Studies and compilations designed mainly for the mathematician and theoretical physicist. Topics in mathematical statistics, theory of experiment design, numerical analysis, theoretical physics and chemistry, logical design and programming of computers and computer systems. Short numerical tables. Issued quarterly. Annual subscription: Domestic, $9.00; Foreign, $11.25.

DIMENSIONS/NBS (formerly Technical News Bulletin)—This monthly magazine is published to inform scientists, engineers, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the work at NBS.

DIMENSIONS/NBS highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, DIMENSIONS/NBS reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing.

Annual subscription: Domestic, $6.50; Foreign, $8.25.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of high-level national and international conferences sponsored by NBS, precision measurement and calibration volumes, NBS annual reports, and other special publications appropriate to this grouping such as wall charts and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396). See also Section 1.2.3.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The purpose of the standards is to establish nationally recognized requirements for products, and to provide all concerned interests with a basis for common understanding of the characteristics of the products. The National Bureau of Standards administers the Voluntary Product Standards program as a supplement to the activities of the private sector standardizing organizations.

Federal Information Processing Standards Publications (FIPS PUBS)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The purpose of the Register is to serve as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations). FIPS PUBS will include approved Federal information processing standards information of general interest, and a complete index of relevant standards publications.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

NBS Interagency Reports—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service (Springfield, Va. 22151) in paper copy or microfiche form.

Order NBS publications (except Bibliographic Subscription Services) from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service (Publications and Reports of Interest in Cryogenics). A literature survey issued weekly. Annual subscription: Domestic, $20.00; foreign, $25.00.

Liquefied Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $20.00. Send subscription orders and remittances for the preceding bibliographic services to the U.S. Department of Commerce, National Technical Information Service, Springfield, Va. 22151.

Electromagnetic Metrology Current Awareness Service (Abstracts of Selected Articles on Measurement Techniques and Standards of Electromagnetic Quantities from D-C to Millimeter-Wave Frequencies). Issued monthly. Annual subscription: $100.00 (Special rates for multi-subscriptions). Send subscription order and remittance to the Electromagnetic Metrology Information Center, Electromagnetics Division, National Bureau of Standards, Boulder, Colo. 80302.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

U.S.MAIL

AMERICAN REVOLUTION BICENTENNIAL
1776-1976