

AD-A107 879

SAI COMSYSTEMS CORP MCLEAN VA
SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES.(U)
JUN 77 T DENIKE, A HOLLAND, T WARD, H DESAI DAHC26-76-D-1004

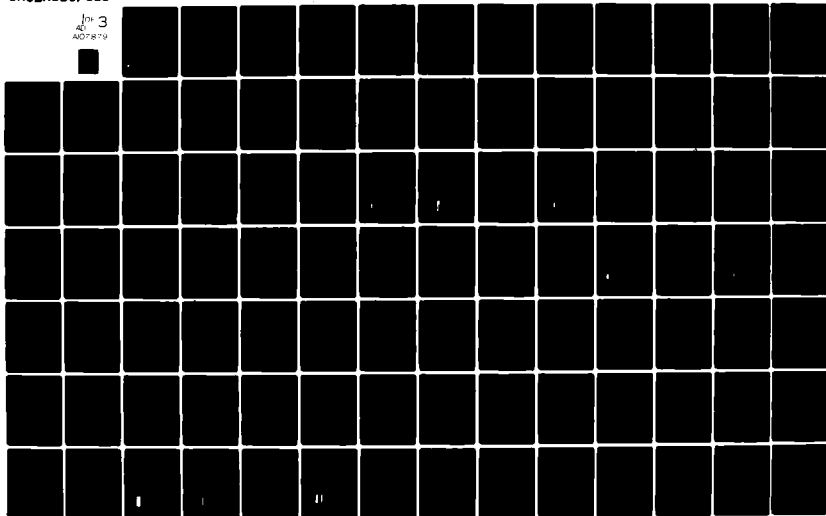
F/6 9/2

UNCLASSIFIED

USACSC-AT-77-11

NL

In 3
AD7879



AD A107879

LEVEL 1

15

TECHNICAL DOCUMENTARY REPORT

U.S. ARMY COMPUTER SYSTEMS COMMAND

18 19
USACSC-AT-77-11
6 SOFTWARE PORTABILITY STUDY
CONVERSION PROCEDURES
9 FINAL REPORT

10
Authors: T./Denike
A./Holland
T./Ward
H./Desai

11 30 JUN 1977

12 214

PREPARED FOR:

U.S. ARMY COMPUTER SYSTEMS COMMAND
FORT BELVOIR, VIRGINIA 22060

PREPARED BY:

SAI COMSYSTEMS CORPORATION
MCLEAN, VIRGINIA 22101

15
CONTRACT DAH026-76-D-1004

DTIC
SELECTED
NOV 27 1981
A

DISTRIBUTION STATEMENT:

APPROVED FOR PUBLIC RELEASE. DISTRIBUTION UNLIMITED.

DTIC FILE COPY

392818
81 11 24 05A

ABSTRACT

This document contains the results of a modification to the Software Portability Study, Delivery Order Number 9 to Contract DAHC26-D-1004. This study concentrated on determining the procedures required to convert software systems written in COBOL in accordance with USACSC standards to a portable COBOL, Florida 74. A further conversion from the Portable Standard COBOL (PSC) to a COBOL executable on the Digital Equipment Corporation (CDEC) PDP 11/70 minicomputer was studied and is presented.

FORWARD

This document was prepared under the authority of USACSC Contract Number DAHC26-76-D-1004, and was prepared by SAI Comsystems for the U.S. Army Computer System Command. This study reports the COBOL software study.

DISCLAIMER

The findings of this report are not to be considered as an official Department of the Army position unless so designated by other authorized documents.

DISPOSITION INSTRUCTIONS

Destroy this report when no longer needed. Do not return it to the originator.

Accession	
NTIS GRA&I	
DTIC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A	

TABLE OF CONTENTS

CHAPTER 1 - OVERVIEW	1-1
1.1 INTRODUCTION	1-1
1.2 PORTABILITY	1-1
1.3 PORTABILITY STUDY	1-1
1.4 DOCUMENT STRUCTURE	1-4
CHAPTER 2 - STUDY SCOPE	2-1
2.1 INTRODUCTION	2-1
2.2 TASK MODIFICATION	2-1
2.3 BACKGROUND	2-1
2.4 TRANSLATOR TRADE-OFF	2-2
CHAPTER 3 - CONVERSION PROCEDURES.	3-1
3.1 INTRODUCTION	3-1
3.2 CONVERSION SCOPE	3-1
3.3 APPENDIX DESCRIPTION	3-1
3.4 RESTRICTIONS AND LIMITATIONS	3-2
3.5 SUMMARY.	3-4
APPENDIX A - USACSC COBOL TO PSC	
APPENDIX B - PSC TO PDP 11 COBOL	

CHAPTER I

OVERVIEW

1.1 INTRODUCTION

The United States Army Computer Systems Command (USACSC) has for many years been developing and maintaining software systems for use throughout the army. The majority of these software systems have been written in COBOL and executed on IBM 360 systems. Due to the rapid advances in computer hardware, the competitive nature of the computer industry and federal government computer procurement practices, it is reasonable to expect that the present software systems will be required to be executed on hardware for which they were not developed. With this in mind, a greater emphasis is being placed on software transportability within the USACSC.

1.2 PORTABILITY

1.2.1 By definition, software portability implies the degree of executability of a high-level language program in multiple and/or varied computer environments. That is, if a program is executable in a foreign environment from which it was developed with minimal or no modification, it is considered to be portable; otherwise, the program is not portable.

1.2.2 Program portability involves many aspects of data processing.

Briefly, to cite a few:

- Compatability among computer vendors,
- Compiler compatability for a given high-level language.
- Compiler compatability within the same vendor and/or other vendors.
- Compatability of a given high-level language as used in various computer environments.
- Program application.
- Determination of the degree to which a program is or is not portable. (That is, if modification is required, how much modification is too much; how many special cases should be incorporated into considerations for portability).
- Determination of a universally executable instruction subset of a high-level language.
- Program dependency on the computer environment.
- Degree of program interdependence with Job Control Language.
- Degree of programmed system dependent device specification.
- Variability in special features such as internal sorts, internal merges, CALL's, etc.
- Compatibility between operating systems within a vendor (e.g., IBM DOS, OS, VS, HASP, MVT, etc) and with other vendors and their variations.
- Compatability in system data management procedures (within a given vendor or given vendor to another vendor.
- Compatability in the relative intelligence built into the original versus the recipient environment (the amount of programmed

information which is or is not required by the program depending on the computer environment).

1.3 PORTABILITY STUDY

1.3.1 The USACSC has tasked SAI Comsystems Corporation as Delivery Order Number 9 of Contract DAHC26-76-D-1004 to study the question of portability in the context of the command environment. The final document, Software Portability Study - Volumes I and II was delivered on April 15, 1977. In the course of this study, the following items were considered in the form of comparisons-similarities and dissimilarities.

- COBOL - Programming language study.
- Job Control Language (JCL) study.
- Executive Software Study (as applicable) in the domain of Operating Systems Environment.
- Computer Hardware Study.

This study was conducted using USACSC minimum hardware configuration requirements as the norm. All software comparisons are based on current versions of the language, JCL, and Executive Software as used by USACSC. The vendors considered include: Burroughs 3500, 3300, 6600 and 7600 Series, Honeywell 6000 Series, Univac 1100 Series, Data General Eclipse, Digital Equipment Corp. PDP 11, and Interdata 8/32.

Also included in the language (COBOL) study are USACSC COBOL and ANSI '74 COBOL.

The purpose of this extension is to take USACAC COBOL and constrain it to become portable so that the portable COBOL is usable by the PDP 11 - hardware environment.

1.4 DOCUMENT STRUCTURE

Chapter 2 of this document presents the general scope of the study. Chapter 3 presents the methodology of converting USACSC COBOL to Portable Standard COBOL (PSC) (Ref: Programming Procedures Manual USACSC 18-1-1; Optimal COBOL Subset for Software Portability DAAG29-77-G-0058) as well as the PSC to PDP 11 COBOL (Ref: Optimal COBOL Subset for Software Portability DAAG29-77-G-0058); PDP-11 COBOL User's Guide No. DEC-11-ICUGA-B-D). Appendix A consists of a heirarchy chart and the detail conversion process for USACSC COBOL to PSC. Appendix B consists of a heirarchy chart and the detail conversion process for PSC to PDP 11 COBOL.

CHAPTER 2

STUDY SCOPE

2.1 INTRODUCTION

This chapter presents the scope of the modification to the original Software Portability Study delivery order and some general understandings and concepts required to complete the task.

2.2 TASK MODIFICATION

Where the portability study included many aspects of the portability question, the modification concentrated on the methodology of achieving the conversion of systems written in COBOL. Specifically, the conversion of systems written in accordance with USAOSC standard to an intermediate COBOL language and thence to a specific hardware dependent COBOL, PDP 11, is being determined.

2.3 BACKGROUND

2.3.1 The methodology being evaluated and presented in this document is general in nature and applicable to both manual and automated modes of operation. The stress has, however, been placed on an automated mode of operation where problem areas are noted and manual intervention is required.

2.3.2 Due to the complexity of converting COBOL systems from one hardware vendor to another, it is evident that no automated mode of conversion will be self-sufficient. Too many inconsistencies occur between vendors implemented COBOL even though the vendors indicate that the language implemented is in accordance with an ANSI standard. These inconsistencies appear in

two forms:

- The entire set of ANSI COBOL is not implemented.
- Vendor extensions have been implemented to fully utilize the unique features of the vendor hardware.

2.3.3 As indicated in the portability study report, the strict adherence to programming systems as a minimal subset (i.e., COBOL statements implemented by all vendors) to achieve portability, too severely restricts the programmer in capability and makes systems thus programmed inefficient on all hardware systems on which it is executed. Therefore, although the portable COBOL in which systems are programmed and maintained may be a minimal subset, the translation from this language to any hardware dependent COBOL must attempt to transform the portable subset into the maximum vendor subset possible to obtain maximum efficiency. The trade off in this philosophy is that the greater the vendor subset attempted, then the greater the complexity of the translator.

2.4 TRANSLATOR TRADE-OFF

2.4.1 The critical point in this trade-off is when the cost of developing the translator exceeds the expected level of manual intervention in modifying the code based on error messages. The translator defined as a result of this effort has been designed with an attempt to minimize the translator development cost and manual intervention and yet enable the utilization of a maximum target COBOL subset.

2.4.2 The scope of this extension is summarized in the block diagram (Figure 2.1). The approach is to achieve portability in two stages. The first stage will accomplish the conversion of USACSC COBOL to an intermediate

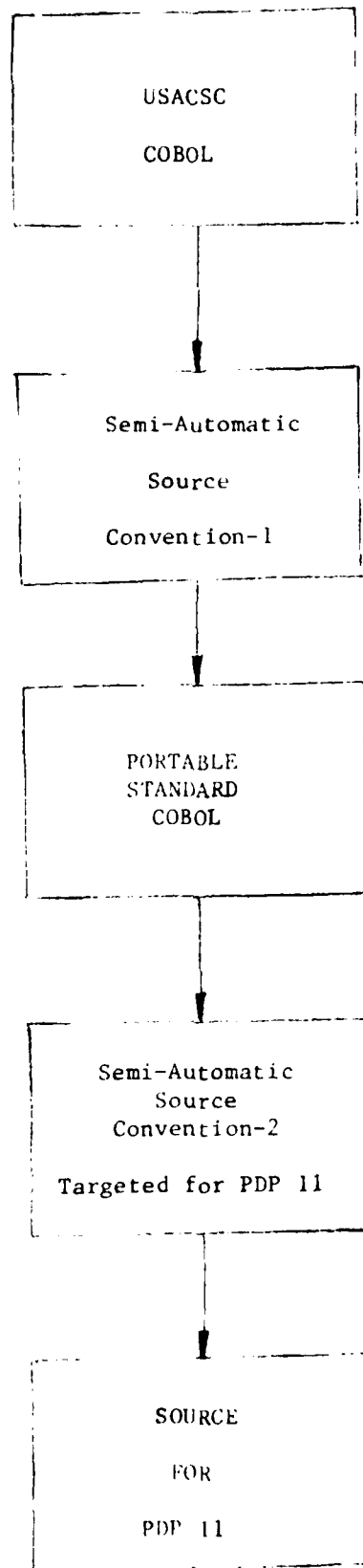


Figure 2.1 BLOCK DIAGRAM

COBOL language element set. This COBOL set, as recommended by University of Florida and agreed upon by USACSC, is a subset of ANSI '74 COBOL. Throughout the study, this intermediate COBOL will be referred to as Portable Standard COBOL (PDS). Also, PDS, being the proper subset (every element in the subset is present in the set) of ANSI '74, will be a usable COBOL on all the computers that support full ANSI '74 COBOL.

2.4.3 The second stage is to translate PDS to PDP 11 COBOL (or other desired hardware environment). This conversion has been accomplished keeping in mind the present state of the art for the COBOL available on PDP 11.

CHAPTER 3
CONVERSION PROCEDURES

3.1 INTRODUCTION

The primary function of these conversion procedures is to assist a programmer to achieve a conversion smoothly. Given a definite target COBOL environment, one can design a system that will allow a conversion. These conversion procedures serve dual purposes. It guides towards a design logic for conversion as well as lists many abnormalities that need to be resolved before automatic conversion can be achieved. This chapter discusses in detail the rationale and procedure for conversion of COBOL code maintained by USACSC to the Portable Standard COBOL (PSC) to PDP 11/70 COBOL. Also included is a discussion of Appendix A and B.

3.2 CONVERSION SCOPE

3.2.1 The conversion process is being presented strictly in terms of the COBOL language. The major consideration in this process is in terms of those capabilities of the compilers used to prepare the code for execution. JCL or Executive Software are identified only if a given function is not available in the target COBOL.

3.3 APPENDIX DESCRIPTION

3.3.1 Two appendices are provided as part of this document. Appendix A presents the conversion process for USACSC to PSC. Appendix B presents the conversion process for PSC to PDP 11/70 COBOL. Each appendix consists of a heirarchial chart and a series of IPO's required for

the conversion process.

3.3.2 The first figure in each appendix is a heirarchical representation of all the statements available for conversion from the source computer.

3.3.3 The numbering technique follows the heirarchy convention by levels, i.e.,

- level 1 - n
- level 2 - n.1; n.2, ...
- level 3 - n.1.1; n.1.2, ...
- level 4 - n.1.1.1; n.1.1.2, ...
- and so on.

3.3.4 After page 1 in each appendix, there follows a set of detailed input-process-output (IPO). On every page of this IPO, one lowest level COBOL element conversion is described. All IPO's adhere to the same format. The input of IPO is a COBOL source (element). The process part of IPO describes the methodology and/or logic required to decide the outcome. The output section of IPO describes the final result of the translation process.

3.3.5 Throughout the IPO's, all ANSI '74 COBOL syntax conventions are used. The only liberty taken is the shading of an option of the input element to indicate that the option is a problem area in conversion. Also, for an element with multiple options, if some options are not directly transferrable, the output section indicates this by showing possible translations as described in the process section. Also, for the sake of completeness, one lowest level element may appear at more than one place in the heirarchy chart; however, they all point to the same detail IPO page.

3.3.6 The usefulness of IPO's became apparent during an attempt to convert a program from USACSC COBOL to PSC. The following is an example indicative of the method of using the IPO's. The programmer performing the conversion is assumed to be familiar with both source and target COBOL systems.

1. Find the IPO corresponding to the source statement. Use heirarchical charts by division as a quick reference.
2. Follow the procedure listed in the process section of the IPO.
3. If a warning message is indicated, several situations may have occurred.
 - a) A portion of the source statement is omitted from the output as it is not required in the target COBOL. However, the omission must be accomplished in the JCL.
 - b) A portion of the source statement is omitted - not required by the targeted environment.

3.4 RESTRICTIONS AND LIMITATIONS

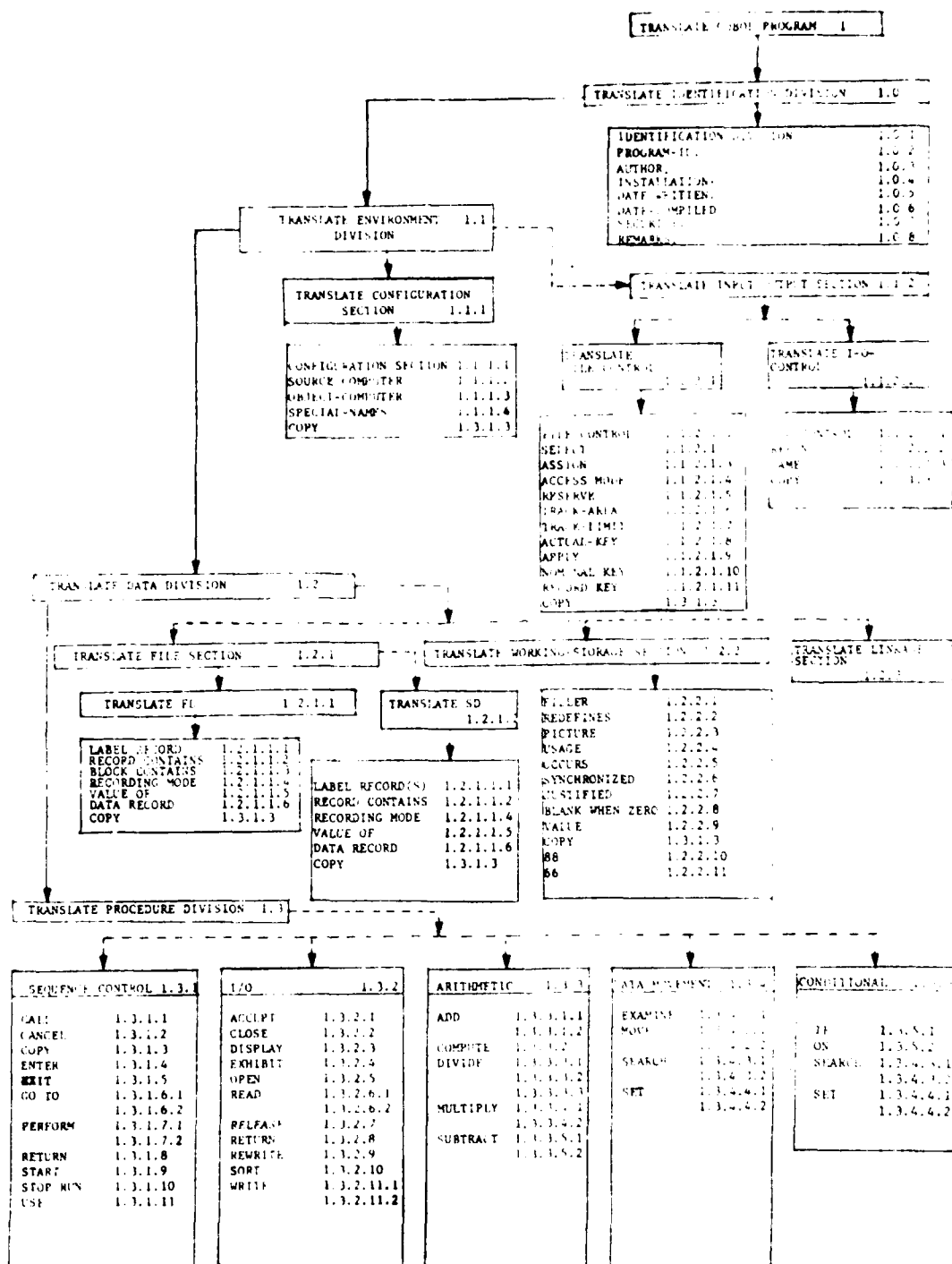
During the entire study, only one area has not been discussed. This is the restrictions and limitations of the compilers for which these conversions are being specified. A thorough check of the target compiler together with practical investigation is a must at this stage. Some of these areas are:

1. Number of GO TO's available in a GO TO --
DEPENDENT UPON clause.
2. Total number of nesting levels available
in a given compiler.
3. Naming conventions.
4. Reserved word list.
5. Number of bits to a byte.
6. word and boundary alignment.
7. Numeric storage formats.
8. Collating sequence.

4.5 SUMMARY

Even though this study covers all the aspects of COBOL conversion with respect to USACSC and PSC, this is by no means a 'total' conversion. One must also convert the data from the old environment to the new environment before a successful execution is possible. Also all the sub-routines, utilities and any macros must be converted to the new environment before program execution. Finally, JCL conversion must also be accomplished. It is also likely JCL conversion might effect source code in the new environment.

APPENDIX A
USACSC COBOL
TO
PORTABLE STANDARD COBOL (PSC)



NO: 1.0.1

INPUT

IDENTIFICATION DIVISION.

PROCESS

Read entire statement, then copy to output
area

OUTPUT

IDENTIFICATION DIVISION.

NO: 1.0.2

INPUT

PROGRAM-ID. program-name.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

PROGRAM-ID. program-name.

NO: 1.1.3

INPUT

<u>AUTHOR.</u> [comment-entry]...

PROCESS

Read entire statement, then copy to output area.
--

OUTPUT

<u>AUTHOR.</u> [comment-entry]...

NO: 1.0.4

INPUT

INSTALLATION.

[comment-entry ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

INSTALLATION.

[comment-entry ...]

NO: 1.0.5

INPUT

DATE-WRITTEN. [comment-

entry ...

PROCESS

Read entire statement, then copy to output area

OUTPUT

DATE-WRITTEN.

[comment-entry

...

NO: 1.0.6

INPUT

DATE-COMPILED.

[comment-entry ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

DATE-COMPILED.

[comment-entry ...]

NO: 1.0.7

INPUT

[SECURITY.]

[comment-entry]

...

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[SECURITY.]

[comment-entry]

...

INPUT

REMARKS:

comment-entry

PROCESS

A. Read a statement

B. Move an asterisk in column 7, and copy the entire statement.

C. If a major division header is encountered, stop process. Else

D. Move an asterisk to column 7 of every statement.

OUTPUT

*REMARKS

* comment-entry

NO: 1.1

INPUT

ENVIRONMENT DIVISION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

ENVIRONMENT DIVISION.

NO: 1.1.1.1

INPUT

CONFIGURATION SECTION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

CONFIGURATION SECTION.

NO: 1.1.1.2

INPUT

SOURCE-COMPUTER.
computer-name.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SOURCE-COMPUTER.
computer-name.

NO: 1.1.1.3

INPUT

OBJECT-COMPUTER.
computer name.

SEGMENT-LIMIT IS
segment number .

PROCESS

Read entire statement, then copy to output area.

OUTPUT

OBJECT-COMPUTER.
computer name.

SEGMENT-LIMIT IS
segment number .

NO: 1.1.1.1.4

INPUT

SPECIAL-NAMES.

implementor-name IS
mnemonic name

PROCESS

Read Entire statement, then copy to output

OUTPUT

SPECIAL-NAMES.

implementor-name IS
mnemonic name

NO: 1.1.2

INPUT

PROCESS

OUTPUT

INPUT-OUTPUT SECTION.

Read entire statement, then copy to output area.

INPUT-OUTPUT SECTION.

NO: 2.1.2.1.2

INPUT

SELECT file-name.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SELECT file-name.

INPUT

SELECT

file-name.

PROCESS

- A. Read the entire statement.
- B. If the OPTIONAL option is found, eliminate it from the statement; copy the statement to the output area, and write a warning message.
- C. Else, copy the statement to the output area.

NOTE: Warning message should indicate that program execution may be affected.

OUTPUT

SELECT file-name

warning-message

NO: 1.1.2.1.3.

INPUT

ASSIGN TO
[integer]
implementor-name-1
[implementor-name-2] ...

PROCESS

- A. Read the entire statement.
- B. If the option is found [FOR MULTIPLE
[REEL] [UNIT]], eliminate it from the state-
ment; copy the statement to the
output area, and write a warning message.
- C. Else, copy the statement to the out-
put area.

NOTE: The warning message should indicate
that program execution may be
affected.

OUTPUT

ASSIGN TO
implementor-name-1
[implementor-name-2]
[**warning-message]

NO: 1.1.2.1.4

INPUT

[ACCESS MODE IS

{ SEQUENTIAL }
RANDOM

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[ACCESS MODE IS

{ SEQUENTIAL }
RANDOM

INPUT

RESERVE

int

AREA
AREAS

PROCESS

- A. Read entire statement
- B. Move the words "RESERVE" and "int" to output area.
- C. Check for the possibility of the verb "ALTERNATE". If this verb is existent, it must be eliminated. (NOTE: Elimination of verb may effect program execution; or there may exist in the operating system some equivalence to "ALTERNATE").
- D. Move the remainder of the statement to the output area and add to "RESERVE int".

OUTPUT

RESERVE int

RESERVE int AREA
AREAS

INPUT

TRACK AREA IS

{ data-name }
 { int }

CHARACTERS

PROCESS

- A. Read the entire statement.
 - B. Since there is no-equivalence, the statement should be transferred to the output area with a warning message indicating no-equivalency.
- NOTE: There may be some effect on execution of the program by eliminating this statement.

OUTPUT

TRACK statement
 ***WARNING message

NO: 1.1.2.1.7

INPUT

TRACK-LIMIT IS

int {
TRACK
TRACKS}

PROCESS

- A. Read the entire statement.
- B. Since there is no-equivalence, this statement should be transferred to the output area with a warning message indicating no-equivalency.

NOTE: There may be some effect on the program by eliminating this statement.

OUTPUT

TRACK statement
***WARNING message

INPUT

ACTUAL KEY IS data-name

PROCESS

- A. Read the entire statement.
 - B. Since there is no-equivalent, this statement should be transferred to the output area with a warning message indicating no-equivalency.
- NOTE: The elimination of this statement may effect execution of this program, and the warning message should reflect this fact.

OUTPUT

ACTUAL statement
*** WARNING message

NO: 1.1.2.1.9

INPUT

APPLY WRITE-ONLY on

file-name-1 [file-name-2]

PROCESS

- A. Read the entire statement.
- B. Since there is no equivalence, then the statement should be transferred with a warning message of no equivalence.

Note: There might exist within the operating system or J.C.L. directives for write-only processing. This fact may be included in the warning message.

OUTPUT

APPLY statement
*** WARNING message

NO: 1.1.2.1.10

INPUT

NOMINAL KEY IS data-name.

PROCESS

- A. Read entire statement.
- B. Copy it with an asterisk, '*', in column seven.
- C. Write a warning message to indicate that the statement is not in ANSI COBOL.

NOTE: Warning message should indicate that program execution may be affected.

OUTPUT

COL. 7

* NOMINAL KEY IS data-name.

** WARNING - Message

NO: 1.1.2.1.11

INPUT

RECORD KEY IS data-name

PROCESS

Read the entire statement and copy it to the
output area.

OUTPUT

RECORD KEY IS data-name-1

NO: 1.1.2.2.1

INPUT

I-O-CONTROL.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

I-O-CONTROL.

NO: 1.1.2.2.2

INPUT

RERUN ON implementor-name
EVERY { END OF { REEL
integer RECORDS UNIT }
OF file-name } ...

PROCESS

Read entire statement, then copy to output area.

OUTPUT

RERUN ON implementor-name
EVERY { END OF { REEL
integer RECORDS UNIT }
OF file-name } ...

INPUT

SAME { RECORD } AREA FOR
 { SAME }
 file-name-1
 { file-name-2 }

PROCESS

- A. Read entire statement.
- B. If RECORD or SORT option is used, statement should be moved to output area with warning message.
- C. Otherwise copy this statement to output area.

OUTPUT

SAME statement
 ***WARNING message
 regarding RECORD or
 SORT
 SAME AREA FOR
 file-name-1
 file-name-2

NO: 1.2

INPUT

DATA DIVISION

PROCESS

Read entire statement, then copy to output area.

OUTPUT

DATA DIVISION.

NO: 1.2.1

INPUT

FILE SECTION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

FILE SECTION.

NO: 1.2.1.1

INPUT

FD file-name

PROCESS

Read entire statement, then copy to output area.

OUTPUT

FD file-name

INPUT

LABEL RECORDS ARE

{ STANDARD }
 { OMITTED }

PROCESS

Read entire statement, then copy to output area.

OUTPUT

LABEL RECORDS ARE

{ STANDARD }
 { OMITTED }

NO: 1.2.1.1.2

INPUT

[RECORD CONTAINS [integer-1
TO] integer-2 CHARACTERS]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[RECORD CONTAINS [integer-1
TO] integer-2 CHARACTERS]

INPUT

BLOCK CONTAINS

int-2 { RECORDS }
 { CHARACTERS }

PROCESS

A. Read entire statement.

B. Move "BLOCK CONTAINS" verbs to the output area.

C. Since the output code set doesn't allow a range capability, if the option "int-1 TO" must be eliminated. (NOTE: Elimination of this option may not effect execution).

D. Add the remaining attributes to the output area to complete statement.

OUTPUT

BLOCK CONTAINS

BLOCK CONTAINS int-2 { RECORDS }
 { CHARACTERS }

INPUT

RECORDING MODE IS

$\left\{ \begin{array}{c} F \\ U \\ V \end{array} \right\}$

PROCESS

- A. Read the entire statement.
- B. Move asterisk in column 7 and copy the option.
- C. Since there exists a no-equivalence condition, a warning message is printed.

NOTE: There may exist within the operating system, and/or J.C.L., a means for indicating recording functions. If not, this statement's elimination may effect the results from execution of the program.

OUTPUT

RECORDING statement.
*** WARNING message.

INPUT

VALUE OF implementor-name-1

IS
[redacted]
literal-1

implementor-name-2

[redacted]
literal-2

PROCESS

- A. Read entire statement.
- B. Move statement up to the "IS" verb to the output area.
- C. Check if a data-name or literal has been used. If a data-name has been used, the statement should be moved to the output area with a warning message indicating no-equivalency. If data-name-0 has been defined previously, substitute literal-1 with appropriate value.
- D. Else, add the literal field to the statement in the output area.
- E. If additional fields are within the statement, the same test indicated in step C (above) should be adhered to, else the additional fields should be added to the output area.

OUTPUT

VALUE OF implementor-name-1
IS

VALUE clause

***WARNING message

VALUE OF implementor-name-1
IS literal-1

VALUE OF implementor-name-1
IS literal-1

[implementor-name-2
literal-2]

NO: 1.2.1.1.6

OUTPUT

[DATA] { RECORD IS
 RECORDS ARE }

data-name-1 [,data-
name-2 ...]

PROCESS

Read entire statement, then copy to output area.

INPUT

[DATA] { RECORD IS
 RECORDS ARE }

data-name-1 [,data-
name-2 ...]

NO: 1.2.2

INPUT

WORKING-STORAGE SECTION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

WORKING-STORAGE SECTION.

NO: 1.2.1.2

INPUT

SD file-name

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SD file-name

NO: 1.2.2.1

INPUT

level-number { data-name
FILLER

PROCESS

Read entire statement, then copy to output area.

OUTPUT

level-number { data-name
FILLER

NO: 1.2.2.2

OUTPUT

REDEFINES data-name

PROCESS

Read entire statement then copy to output area.

INPUT

REDEFINES data-name

INPUT

$\left\{ \begin{array}{l} \text{PICTURE} \\ \text{PIC} \end{array} \right\}$ IS character-
string

PROCESS

Read entire statement then copy to output
area.

OUTPUT

$\left\{ \begin{array}{l} \text{PICTURE} \\ \text{PIC} \end{array} \right\}$ IS character-
string

NO: 1.2.2.4

INPUT

PROCESS

OUTPUT

USAGE IS {
 COMPUTATIONAL
 COMP
 DISPLAY
 INDEX
 COMPUTATIONAL-3
 COMP-3

- A. Read entire statement.
- B. If "COMPUTATIONAL-3" or "COMP-3" has been used, move statement to output area with warning message.
(Note: Elimination of this phrase may effect execution.)
- C. Otherwise, copy the statement to the output area.

**WARNING statement
on COMP-3, etc.

USAGE IS {
 COMPUTATIONAL
 COMP
 DISPLAY
 INDEX

NO: 1.2.2.2

INPUT

OCCURS

{ integer-1 TIMES
integer-1 to
integer-2 TIMES }

DEPENDING ON data-name }

[ASCENDING
[DESCENDING]]

KEY IS

data-name-1 [, data-
name-2 ...]

[INDEXED BY

index-name-1 [, index-
name-2 ...]

PROCESS

Read entire statement, then copy to output area

OUTPUT

OCCURS

{ integer-1 TIMES
integer-1 to
integer-2 TIMES }

DEPENDING ON data-name }

[ASCENDING
[DESCENDING]]

KEY IS

data-name-1 [, data-
name-2 ...]

[INDEXED BY

index-name-1 [, index-
name-2 ...]

NO: 1.2.2.6

INPUT

SYNCHRONIZED SYNC	LEFT RIGHT
---	----------------------------------

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SYNCHRONIZED SYNC	LEFT RIGHT
---	----------------------------------

NO: 1.2.2.7

INPUT

Diagram illustrating the INPUT stage of a process. It shows a large rectangular box containing a smaller box. Inside the smaller box, the word "JUSTIFIED" is written above a horizontal line, and the word "JUST" is written below the line. To the right of this box, the word "RIGHT" is written.

PROCESS

Diagram illustrating the PROCESS stage of a process. It shows a large rectangular box containing the text: "Read entire statement, then copy to output area."

OUTPUT

Diagram illustrating the OUTPUT stage of a process. It shows a large rectangular box containing a smaller box. Inside the smaller box, the word "JUSTIFIED" is written above a horizontal line, and the word "JUST" is written below the line. To the right of this box, the word "RIGHT" is written.

NO: 1.2.2.8

INPUT

[BLANK WHEN ZERO]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[BLANK WHEN ZERO]

NO: 1.2.2.9

INPUT

[VALUE IS literal]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[VALUE IS literal]

NO: 1.2.2.10

INPUT

[88]

condition-name

{ VALUE IS
VALUES ARE }

literal-1 [THRU literal-2]

[, literal-3 [THRU literal-4]

...

PROCESS

Read entire statement, then copy to output area.

OUTPUT

88

condition-name

{ VALUE IS
VALUES ARE }

literal-1 [THRU literal-2]

[, literal-3 [THRU literal-4]

...

INPUT

66 data-name-1
 RENAMES data-name-1
 { THROUGH } data-name-3
 { THRU }

PROCESS

Read entire statement, then copy to output area.

OUTPUT

66 data-name-1
 RENAMES data-name-2
 { THROUGH } data-name-3
 { THRU }

NO: 1.2.3

INPUT

LINKAGE SECTION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

LINKAGE SECTION.

NO: 1.3

INPUT

PROCEDURE DIVISION

[USING data-name-1

[data-name-2] ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

PROCEDURE DIVISION

[USING data-name-1

[data-name-2] ...]

NO: 1.3.1.1

INPUT

CALL literal-1
[USING data-name-1
[data-name-2] ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

CALL literal-1
[USING data-name-1
[data-name-2] ...]

NO: 1.3.1.2

INPUT

CANCEL

id-1 } id-2 }
lit-1 } lit-2 }
... }

PROCESS

- A. Read the statement until a period has been encountered.
- B. Since there exists a no-equivalence condition, the statement should be transferred to the output area with a warning message indicating no-equivalency.

NOTE: Elimination of this statement may effect the program's execution.

OUTPUT

CANCEL statement
***WARNING message

NO: 1.3.1.3

INPUT

PROCESS

OUTPUT

<u>COPY</u>	text-name
[identifier-1 literal-1 word-1 <u>REPLACING</u> BY identifier-2 literal-2 word-2 ...]	

Read entire statement, then copy to output area.

<u>COPY</u>	text-name
[identifier-1 literal-1 word-1 <u>REPLACING</u> BY identifier-2 literal-2 word-2 ...]	

NO: 1.3.1.4

INPUT

ENTER language-name

[routine-name]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

ENTER language-name

[routine-name]

NO: 1.3.1.5

INPUT

EXIT
[PROGRAM]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

EXIT
[PROGRAM]

NO: 1.3.1.6.1

INPUT

GO TO [procedure-name]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

GO TO [procedure-name-1]

NO: 1.3.1.6.2

INPUT

GO TO procedure-name-1
[procedure-name-2] ...
DEPENDING ON identifier

PROCESS

Read entire statement, then copy to output area.

OUTPUT

GO TO procedure-name-1
[procedure-name-2] ...
DEPENDING ON identifier

INPUT

<u>PERFORM</u>	procedure-name-1
<u>THRU</u>	procedure-name-2
<u>VARYING</u>	{ index-name-1 identifier-1 }
<u>FROM</u>	{ index-name-2 literal-2 identifier-2 }
<u>BY</u>	{ literal-3 identifier-3 }
<u>UNTIL</u>	condition-1
<u>AFTER</u>	{ index-name-4 identifier-4 }
<u>FROM</u>	{ index-name-5 literal-5 identifier-5 }
<u>BY</u>	{ literal-6 identifier-6 }
<u>UNTIL</u>	condition-2

PROCESS

Read entire statement, then copy to output area.

OUTPUT

<u>PERFORM</u>	procedure-name-1
<u>THRU</u>	procedure-name-2
<u>VARYING</u>	{ index-name-1 identifier-1 }
<u>FROM</u>	{ index-name-2 literal-2 identifier-2 }
<u>BY</u>	{ literal-3 identifier-3 }
<u>UNTIL</u>	condition-1
<u>AFTER</u>	{ index-name-4 identifier-4 }
<u>FROM</u>	{ index-name-5 literal-5 identifier-5 }
<u>BY</u>	{ literal-6 identifier-6 }
<u>UNTIL</u>	condition-2

PROCESS

INPUT

AFTER { index-name-7
 identifier-7 }
FROM { index-name-8
 literal-8
 identifier-8 }
BY { literal-9
 identifier-9 }
UNTIL condition-3]

OUTPUT

AFTER { index-name-7
 identifier-7 }
FROM { index-name-8
 literal-8
 identifier-8 }
BY { literal-9
 identifier-9 }
UNTIL condition-3]

NO: 1.3.1.7.2

INPUT

PROCESS

OUTPUT

PERFORM procedure-name-1

[THRU procedure-name-2]

identifier-1

{ integer-1 } TIMES

UNTIL condition-1

PERFORM procedure-name-1

[THRU procedure-name-2]

identifier-1

{ integer-1 } TIMES

UNTIL condition 1

Read entire statement, then copy to output area.

NO: 1.3.1.8

INPUT

RETURN file-name RECORD
[INTO identifier]
AT END imperative-stmt.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

RETURN file-name RECORD
[INTO identifier]
AT END imperative-stmt.

NO: 1.3.1.9

INPUT

START file-name

INVALID KEY

imperative-statement

PROCESS

Read entire statement, then copy to output area.

OUTPUT

START file-name

INVALID KEY

imperative-statement

NO: 1.3.1.10

INPUT

STOP RUN.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

STOP RUN.

Index:

SECRET

REEL
FILE
UNIT

APPENDIX I: PROCEDURE ON

```

file-name-1
file-name-2...

```

PROCESS

- A. Read the entire statement.
- B. Since there exists a no-equivalence condition, this statement should be transferred to the output area with a warning message indicating no-equivalency.

NOTE: Elimination of this statement may effect program's execution.

OUTPUT

```
USE statement
**WARNING message
```

INPUT

ACCEPT identifier FROM

DATE
DAY
TIME

PROCESS

- A. Read entire statement.
- B. Generate a SPECIAL-NAMES statement in the Data Division with a date and/or time parameter.
- C. Set-up a Procedure Division statement that would acquire the system's date and/or time.
- D. Set-up Working-Storage area to place system acquired date and/or time in appropriate format.
- E. Set-up a Procedure Division statement that would move the acquired date and/or time to Working-Storage area.
- F. Disperse generated output statements within the program in the output area.

OUTPUT

SPECIAL-NAMES.

implementor-name IS
mnemonic-name

ACCEPT identifier

WORKING-STORAGE SECTION.

01 Date or Time, etc.

MOVE identifier-1 TO
identifier-2

INPUT

CLOSE file-name-1



file-name-2

WITH NO REWIND LOCK

PROCESS

- A. Read entire statement.
- B. If the statement contains the option:
 WITH NO REWIND LOCK
 then, the remaining fields should be transferred to the output area.
 (NOTE: Even though this option is eliminated, the operating system or J.C.L. may have options for indicating no rewinding or locking.

OUTPUT

CLOSE file-name-1

REEL UNIT

file-name-2

REEL UNIT

INPUT

DISPLAY { identifier-1 literal-1 } { identifier-2 literal-2 } [REDACTED]
--

PROCESS

A. Read entire statement. B. If the "UPON" option is used, then it must be eliminated. (NOTE: there may exist within the operating system, means to simulate the device to display upon). This option should be moved to the output area with appropriate warning message. . Else, copy statement to output area.
--

OUTPUT

UPON clause ***WARNING message DISPLAY { identifier-1 literal-1 } { identifier-2 literal-2 } ...
--

OUTPUT

DISPLAB	
DISPLAB	literal identifier
DISPLAB	identifier-1 literal-1
	identifier-2 literal-2

PROCESS

1. The "display" option is used, the data must be translated as a literal, then, add the data-name as an identifier, then add result field to the "display" verb in the output area.

2. Else, add the remaining fields to the "display" verb in the output area.

EXHIBIT 1.3.1.4

1. The "display" option is used, the data must be translated as a literal, then, add the data-name as an identifier, then add result field to the "display" verb in the output area.

2. Else, add the remaining fields to the "display" verb in the output area.

NO: 1.3.2.5

INPUT

PROCESS

OUTPUT

<u>INPUT</u> file name	[REDACTED]
<u>OUTPUT</u> file name	[REDACTED]
<u>I-O</u> file name	

A. Read entire statement.

B. If the options "REVERSED" and/or "WITH NO REWIND" is used, they must be eliminated since there is a no-equivalency condition prevailing. (NOTE: There may exist in the operating system or J.C.L. options for simulating these conditions.)

C. Transfer the remaining statement to the output area.

<u>OPEN</u>	<u>INPUT</u> file-name	...
	<u>OUTPUT</u> file-name	
	<u>I-O</u> file-name	

NO: 1.3.2.6.1

INPUT

READ file RECORD

[INFO identifier]

[INVALID KEY imperative stmt]

PROCESS

Read entire statement, then output as read.

OUTPUT

READ file RECORD

[INFO identifier]

[INVALID KEY imperative stmt]

NO: 1.3.2.6.2

INPUT

PROCESS

OUTPUT

READ file [NEXT] RECORD

[INTO identifier]

[AT END imperative statement]

READ file [NEXT] RECORD

[INTO identifier]

[AT END imperative stmt.]

Read entire statement, then copy to output area

NO: 1.3.2.7

INPUT

RELEASE record name

[FROM identifier]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

RELEASE record name

[FROM identifier]

NO: 1.3.2.8

INPUT

RETURN file-name RECORD

[INTC identifier]

AT END imperative statement

PROCESS

Read entire statement, then copy to output area.

OUTPUT

RETURN file-name RECORD

[INTO identifier]

AT END imperative statement

OUTPUT

Sort file-name-1 ON

{
ASCENDING
DESCENDING
}

KEY data-
name-1

[,data-name-2] ...

{
ASCENDING
DESCENDING
}

KEY data-
name-3

[,data-name-4] ...

INPUT PROCEDURE IS section-
name-1

{
THROUGH
THRU
}

section-
name-2

USING file-name-2 [,file-
name-3]

OUTPUT PROCEDURE IS
section-name-3

{
THROUGH
THRU
}

section-
name-4

GIVING file-name-4

PROCESS

Read entire statement, then copy to
output area.

INPUT

Sort file-name-1 ON

{
ASCENDING
DESCENDING
}

KEY data-
name-1

[,data-name-2] ...

{
ASCENDING
DESCENDING
}

KEY data-
name-3

[,data-name-4] ...

INPUT PROCEDURE IS section-
name-1

{
THROUGH
THRU
}

section-
name-2

USING file-name-2 [,file-
name-3] ...

OUTPUT PROCEDURE IS
section-name-3

{
THROUGH
THRU
}

section-
name-4

GIVING file-name-4

NO: 1.3.2.11.1

INPUT

WRITE record-name

[FROM identifier-1]

[INVALID KEY imperative-
statement]

PROCESS

Read entire statement, then copy to output
area.

OUTPUT

WRITE record-name

[FROM identifier-1]

[INVALID KEY imperative-
statement]

NO: 1.3.2.9

INPUT

REWRITE record-name

[FROM identifier]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

REWRITE record-name

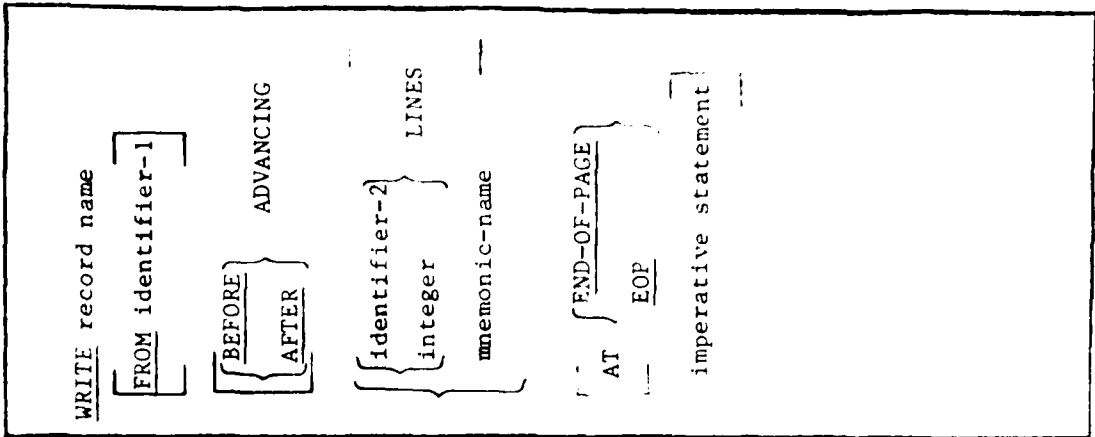
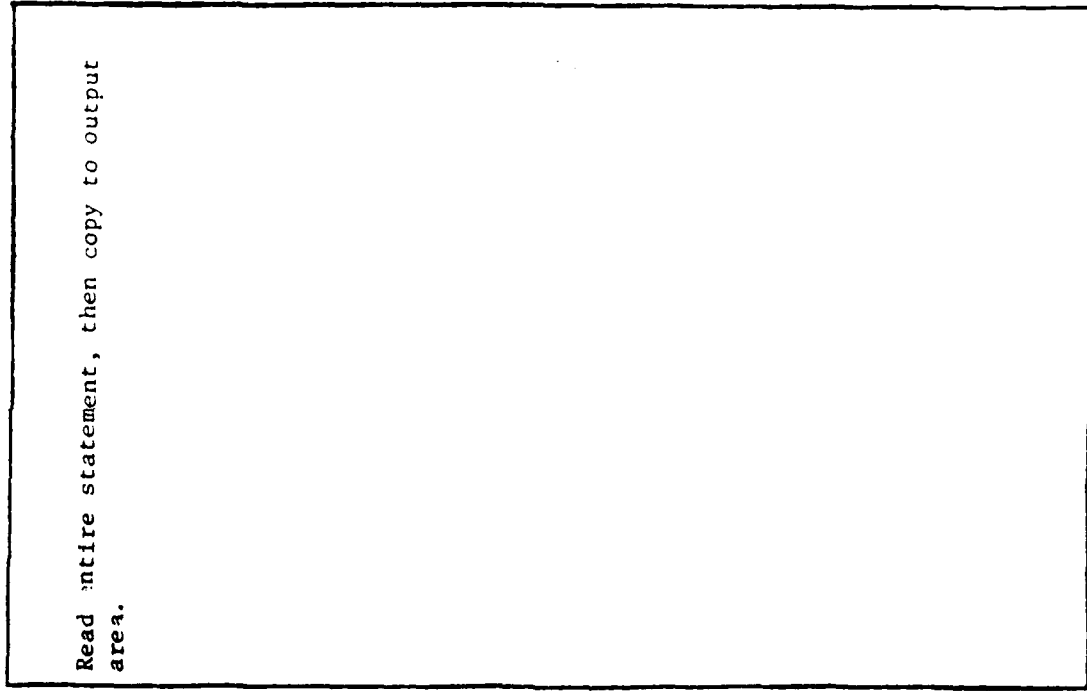
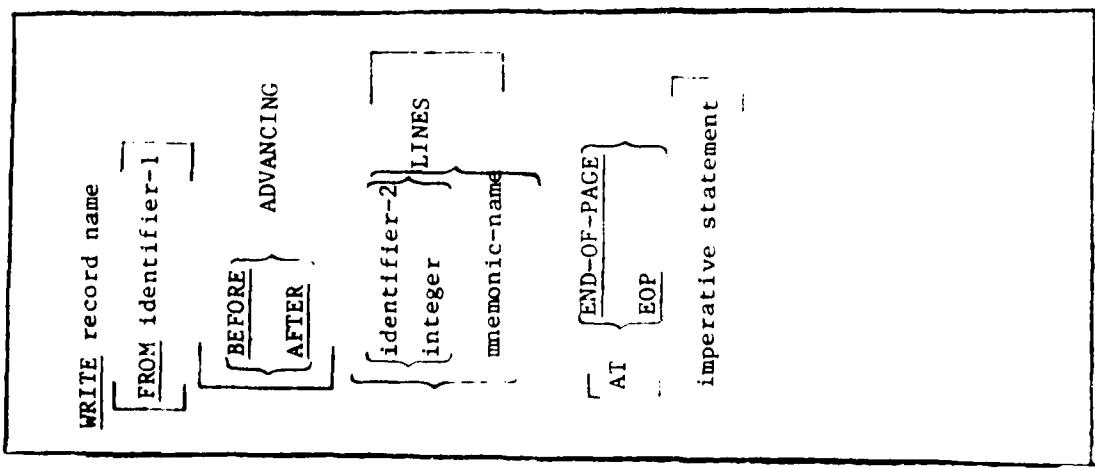
[FROM identifier]

NO: 1.3.2.11.2

INPUT

PROCESS

OUTPUT



AD-A107 879

SAI COMSYSTEMS CORP MCLEAN VA
SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES.(U)
JUN 77 T DENIKE, A HOLLAND, T WARD, M DESAI DAHC26-76-D-1004

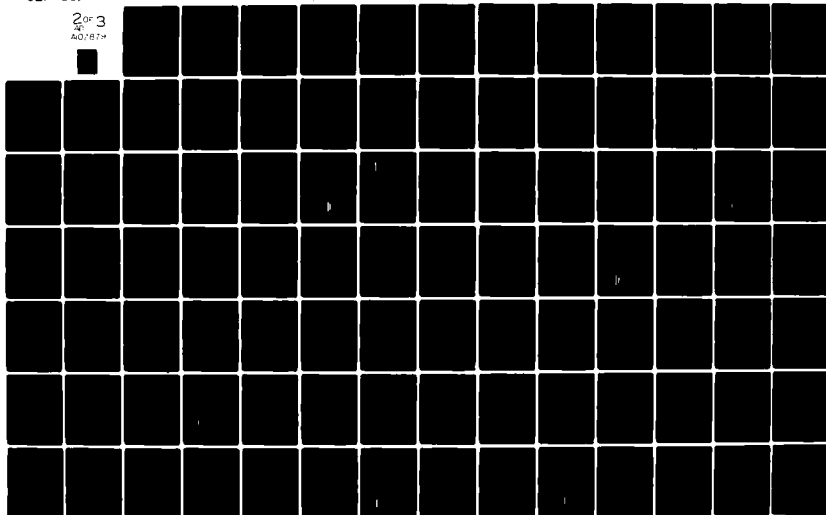
F/G 9/2

UNCLASSIFIED

USACSC-AT-77-11

NL

2 of 3
AD7874



NO: 1.3.3.1.1

INPUT

ADD
 { identifier-1 } { identifier-2 }
 literal-1 literal-2
 TO identifier-m [ROUNDED]
 identifier-n [ROUNDED] ...
 ON SIZE ERROR imperative-
 statement

PROCESS

Read entire statement then copy to output area.

OUTPUT

ADD
 identifier-1 identifier-2
 literal-1 literal-2
 TO identifier-m [ROUNDED]
 identifier-n [ROUNDED] ...
 ON SIZE ERROR imperative-
 statement

NO: 1.3.3.1.2

INPUT

PROCESS

OUTPUT

ADD

{ identifier-1 identifier-2 }
{ literal-1 literal-2 }

{ identifier-3 }
{ literal-3 }
...

GIVING identifier-m ROUNDED

identifier-n ROUNDED

ON SIZE ERROR imperative -

statement

ADD

Read entire statement, then copy to output area.

{ identifier-1 identifier-2 }
{ literal-1 literal-2 }

{ identifier-3 }
{ literal-3 }
...

GIVING identifier-m ROUNDED

identifier-n ROUNDED

ON SIZE ERROR imperative -

statement

NO: 1.3.3.2

INPUT

PROCESS

OUTPUT

COMPUTE identifier-1 ROUNDED
[identifier-2 ROUNDED] ...
= arithmetic-expression
--
[ON SIZE ERROR imperative-
statement]

Read entire statement, then copy to output area.

COMPUTE identifier-1 ROUNDED
[identifier-2 ROUNDED] ...
= arithmetic-expression
--
[ON SIZE ERROR imperative-
statement]

NO: 1.3.3.3.1

INPUT

DIVIDE { identifier-1 } INTO
 { literal-1 }

identifier-2 ROUNDED

[ON SIZE ERROR imperative-
 statement]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

DIVIDE { identifier-1 } INTO
 { literal-1 }

identifier-2 ROUNDED

[ON SIZE ERROR imperative-
 statement]

INPUT

DIVIDE { identifier-1 } { BY } { INTO }

{ identifier-2 } { literal-2 } { GIVING } { identifier-3 }

[ROUNDED] [ON SIZE ERROR] imperative statement

PROCESS

Read entire statement, then copy to output area.

OUTPUT

DIVIDE { identifier-1 } { BY } { INTO }

{ identifier-2 } { literal-2 } { GIVING } { identifier-3 }

[ROUNDED] [ON SIZE ERROR] imperative statement

NO: 1.3.3.3.3

INPUT

DIVIDE { identifier-1 } { INTO }
 { literal-1 } { BY }

{ identifier-2 } GIVING identifier-3
 { literal-2 }

[ROUNDED] [REMAINDER identifier-4]

[ON SIZE ERROR imperative-]
 [statement]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

DIVIDE { identifier-1 } { INTO }
 { literal-1 } { BY }

{ identifier-2 } GIVING identifier-3
 { literal-2 }

[ROUNDED] [REMAINDER identifier-4]

[ON SIZE ERROR imperative-]
 [statement]

NO: 1.3.3.4.1

INPUT

MULTIPLY { identifier-1 }
 { literal-1 }
BY identifier-2 [ROUNDED]
[ON SIZE ERROR imperative-
 statement]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

MULTIPLY { identifier-1 }
 { literal-1 }
BY identifier-2 [ROUNDED]
[ON SIZE ERROR imperative-
 statement]

NO: 1.3.3.4.2

INPUT

MULTIPLY { identifier-1
literal-1 }

BY { identifier-2
literal-2 }

GIVING identifier-3 ROUNDED

ON SIZE ERROR imperative-
statement

PROCESS

Read entire statement, then copy to output area.

OUTPUT

MULTIPLY { identifier-1
literal-1 }

BY { identifier-2
literal-2 }

GIVING identifier-3 ROUNDED

ON SIZE ERROR imperative-
statement

INPUT

SUBTRACT { identifier-1 }
 { literal-1 }

... FROM

{ identifier-2 }
 { literal-2 }

{ identifier-m }
 { literal-m }

identifier-n GIVING

[ROUNDED]

[ON SIZE ERROR imperative
 statement]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SUBTRACT { identifier-1 }
 { literal-1 }

... FROM

{ identifier-2 }
 { literal-2 }

{ identifier-m }
 { literal-m }

identifier-n GIVING

[ROUNDED]

[ON SIZE ERROR imperative
 statement]

NO: 1.3.3.5.2

INPUT

SUBTRACT { identifier-1
literal-1 }
[identifier-2
literal-2] ...
FROM identifier-m [ROUNDED]
[identifier-n [ROUNDED]] ...
[ON SIZE ERROR imperative-
statement]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SUBTRACT { identifier-1
literal-1 }
[identifier-2
literal-2] ...
FROM identifier-m [ROUNDED]
[identifier-n [ROUNDED]] ...
[ON SIZE ERROR imperative-
statement]

INPUT

EXAMINE id TALLYING

{	<u>UNTIL FIRST</u>	}	lit-1
	<u>ALL</u>		
	<u>LEADING</u>		

REPLACING BY lit-2EXAMINE id REPLACING

{	<u>ALL</u>	}	lit-1 BY lit-2
	<u>LEADING</u>		
	<u>FIRST</u>		
	<u>UNTIL FIRST</u>		

PROCESS

- A. Read entire statement.
- B. Transform the word EXAMINE to the word INSPECT.
- C. Carry over "id TALLYING" or "id REPLACING" fields and add to INSPECT verb.
- D. If the "UNTIL FIRST" option has been used in either format, an "if" condition must be set-up to check, within the Procedure Division.
- E. If other options are used, add to the INSPECT format and transfer entire statement(s) to the output area.

OUTPUT

INSPECT
INSPECT id TALLYING
 (or)
INSPECT id REPLACING
IF first occurrence THEN
imperative statementINSPECT id TALLYING

{	<u>ALL</u>	}	lit-1
	<u>LEADING</u>		

REPLACING BY lit-2

(or)

INSPECT id REPLACINGALLLEADING lit-1 BY lit-2FIRST

NO: 1.3.4.2.1

INPUT

PROCESS

OUTPUT

MOVE { identifier-1 }
 literal
identifier-2 [identifier-3] ...

MOVE { identifier-1 }
 literal
TO
identifier-2 [identifier-3] ...

Read entire statement, then copy to output area.

NO: 1.3.4.2.2

INPUT

PROCESS

OUTPUT

MOVE { CORRESPONDING
CORR }

Identifier-1 TO identifier-2

Read entire statement, then copy to output area.

MOVE

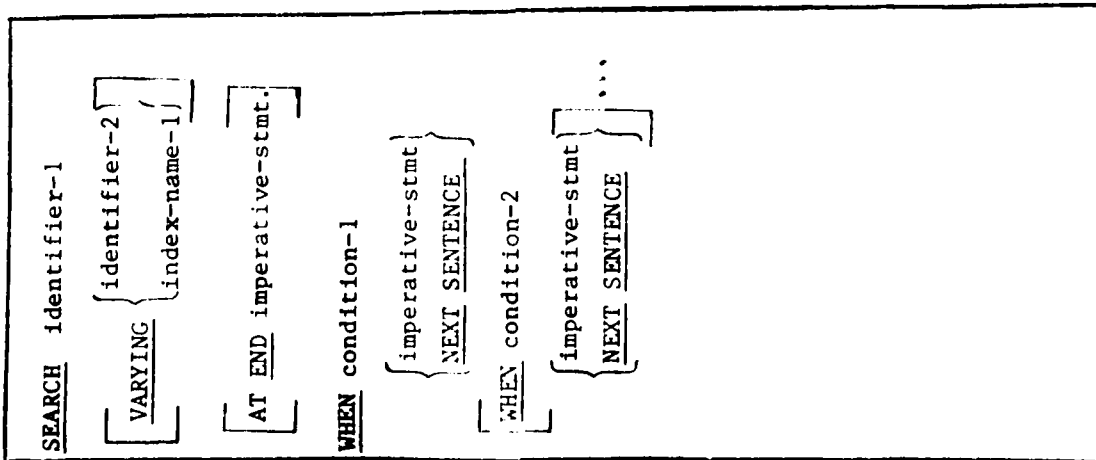
CORRESPONDING

CORR

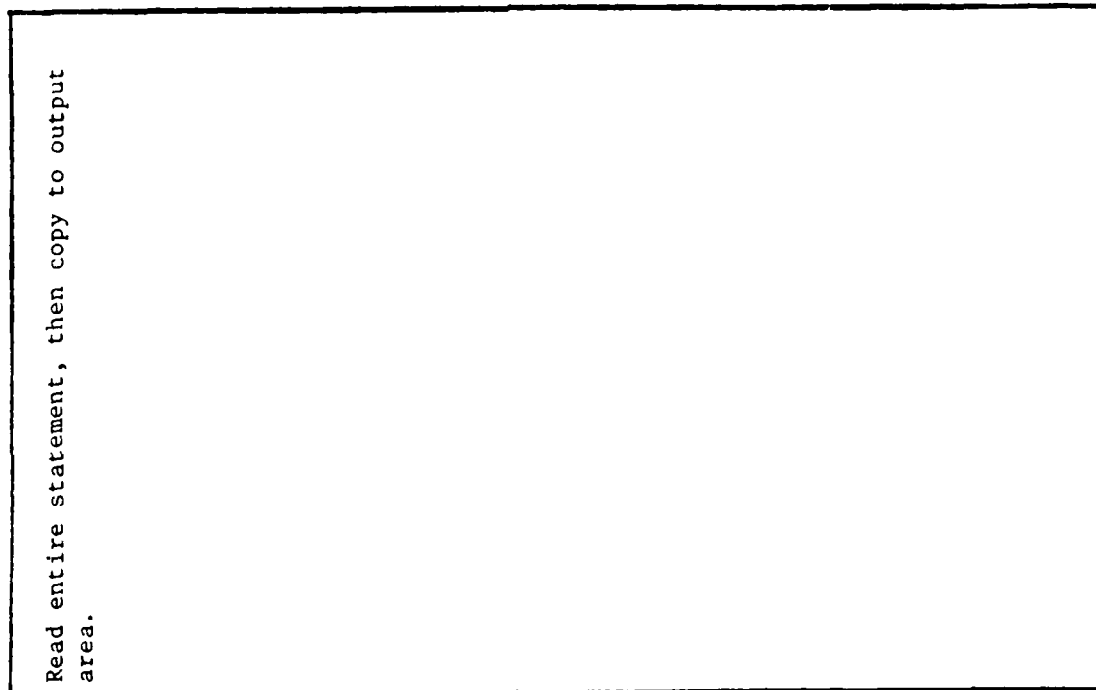
Identifier-1 TO identifier-2

NO: 1.3.4.3.1

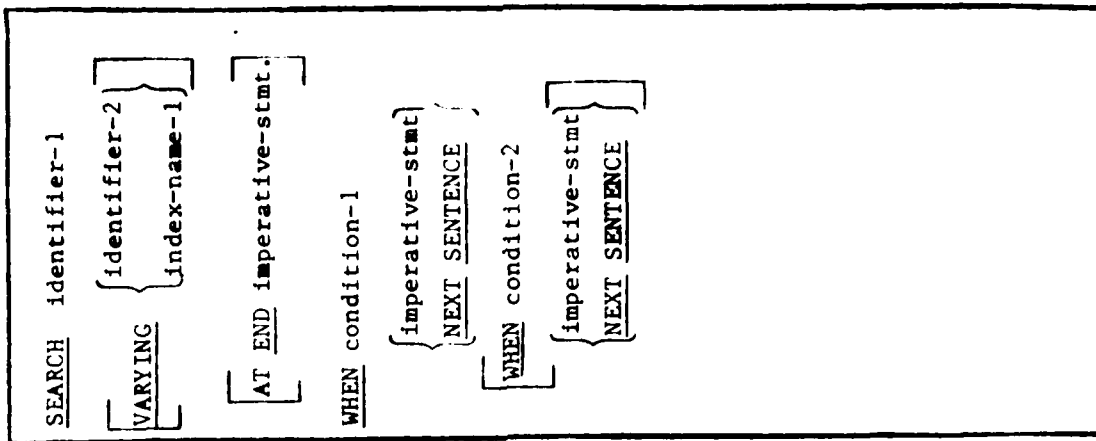
INPUT



PROCESS



OUTPUT



INPUT

PROCESS

OUTPUT

```

SEARCH ALL
identifier-1 [AT END imperative-stmt-1]
WHEN
  {data-
  name-1 { IS EQUAL TO
           IS =
  condition-name-1
  identifier-2 }
  literal-1
  arithmetic-
  expression-1 }
AND
  {data-
  name-2 { IS EQUAL TO
           IS =
  condition-name-2
  identifier-3 }
  literal-2
  arithmetic-
  expression-2 }
  ...
  {imperative-stmt-2
  NEXT SENTENCE
  }
  
```

```

SEARCH ALL
identifier-1 [AT END imperative-stmt-1]
WHEN
  {data-
  name-1 { IS EQUAL TO
           IS =
  condition-name-1
  identifier-2 }
  literal-1
  arithmetic-
  expression-1 }
AND
  {data-
  name-2 { IS EQUAL TO
           IS =
  condition-name-2
  identifier-3 }
  literal-2
  arithmetic-
  expression-2 }
  ...
  {imperative-stmt-2
  NEXT SENTENCE
  }
  
```

Read entire statement, then copy to output area.

NO: 1.3.4.4.1

INPUT

PROCESS

OUTPUT

SET

{ identifier-1
index-name-1
[identifier-2
index-name-2]
... }

TO

{ identifier-3
index-name-3
integer-1 }

SET

{ identifier-1
index-name-1
[identifier-2
index-name-2]
... }

TO

{ identifier-3
index-name-3
integer-3 }

Read entire statement, then copy to output area.

INPUT

SET
 index-name-4, index-name5, ...
 UP BY Identifier-4
 DOWN BY Integer-2

PROCESS

read entire statement, then copy to output area.

OUTPUT

SET
 index-name-4, index-name5
 { UP BY Identifier-4
 DOWN BY Integer-2 }

NO: 1.3.5.1

INPUT

IF condition

{ statement-1
NEXT SENTENCE }

ELSE { statement-2
NEXT SENTENCE }

PROCESS

Read entire statement, then copy to output area.

OUTPUT

IF condition

{ statement-1
NEXT SENTENCE }

ELSE { statement-2
NEXT SENTENCE }

INPUT

```

ON int-1 [ AND EVERY int-2 ]
  [
    UNTIL int-2
    ]
  {
    imp-stmt
    NEXT SENTENCE
  }
  {
    ELSE
    OTHERWISE
  }
  {
    statement
    NEXT SENTENCE
  }

```

PROCESS

- Read entire statement.
- Generate Working Storage statements.
- Set-up initialization statement for counter-field in Procedure Division.
- Set-up an adding statement to the counter-field for incrementing.
- Set-up "If" statement for checking incrementing process in previous step.

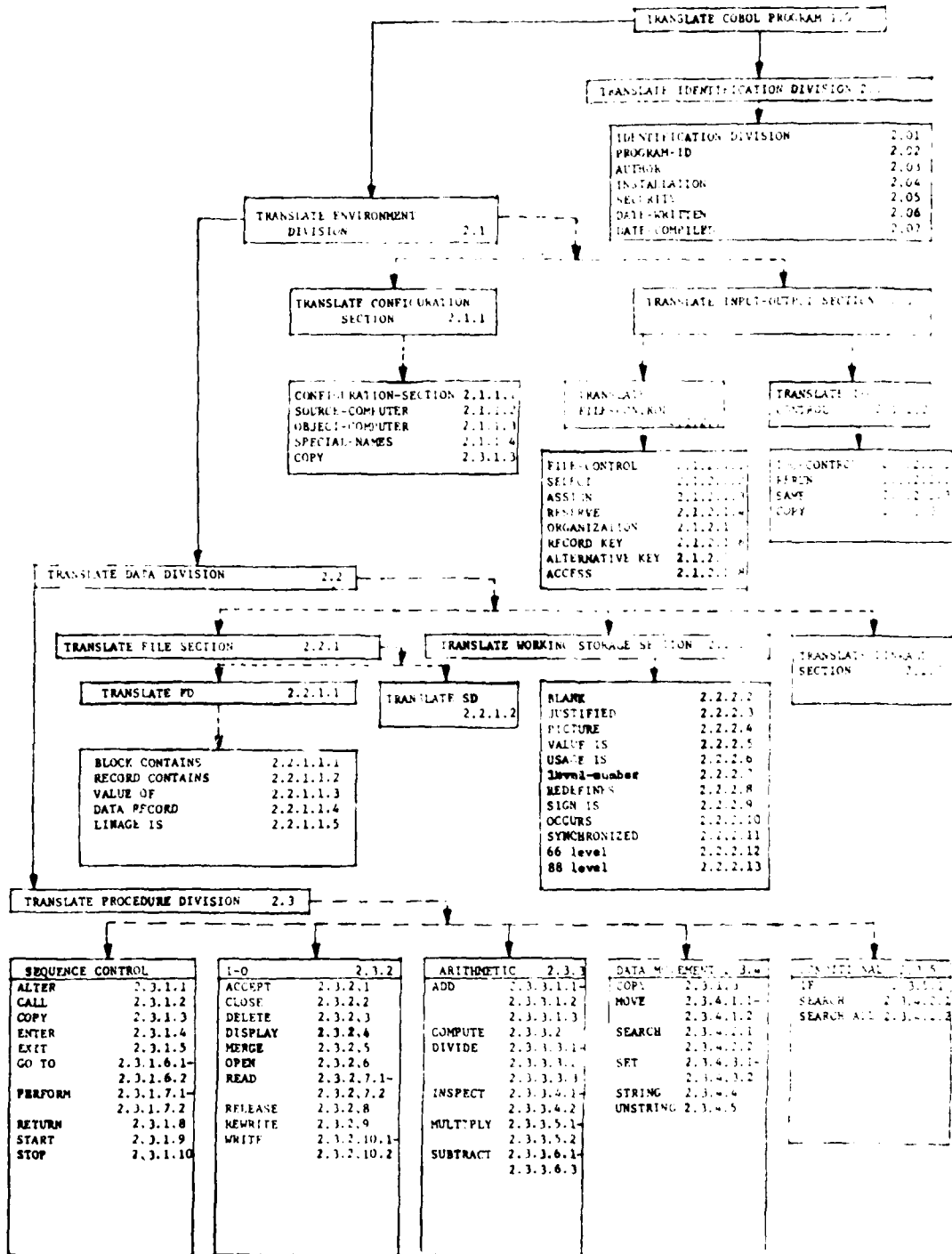
OUTPUT

```

17 argument-field PIC 9 (8)
17 counter-field PIC 9 (5)
17 NE PIC 9 VALUE 1
M VLE ZEROS TO counter-field.
ADD ONE TO counter-field.
IF counter-field = (int-1 +
(int-2 * argument field)
AND counter-field int-3
THEN
  imp-stmt
  NEXT SENTENCE
ELSE
  imp-stmt
  NEXT SENTENCE

```

APPENDIX B
PORTABLE STANDARD COBOL (PSC)
TO
PDP 11 COBOL



NO: 2.0.1

INPUT

IDENTIFICATION DIVISION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

IDENTIFICATION DIVISION.

NO: 2.0.2

INPUT

PROGRAM-ID. program-name.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

PROGRAM-ID. program-name.

NO: 2.0.3

INPUT

AUTHOR.

comment-entry ...

PROCESS

Read entire statement, then copy to output area.

OUTPUT

AUTHOR.

comment-entry

NO: 2.0.4

INPUT

INSTALLATION:

Comment entry

PROCESS

Read entire statement, then copy to output area.

OUTPUT

INSTALLATION:

Comment entry

NO: 2.0.5

INPUT

[SECURITY.]

[comment-entry] ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[SECURITY.]

[comment-entry] ...]

NO: 2.0.6

INPUT

DATE-WRITTEN.

[comment-entry ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

DATE-WRITTEN.

[comment-entry ...]

NO: 2.0.7

INPUT

[DATE-COMPILED.

[comment-entry] ...]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

[DATE-COMPILED.

[comment-entry] ...]

NO: 2.1

INPUT

ENVIRONMENT DIVISION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

ENVIRONMENT DIVISION.

NO: 2.1.1.1

INPUT

CONFIGURATION SECTION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

CONFIGURATION SECTION.

NO: 2.1.1.2

INPUT

SOURCE-COMPUTER.

computer-name

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SOURCE-COMPUTER.

computer-name

NO: 2.1.1.3

INPUT

OBJECT-COMPUTER.

computer-name

MEMORY SIZE integer

WORDS

CHARACTERS

MODULES

PROCESS

A. Read entire statement.

B. If "Segment-Limit" clause exists, move clause to output area with warning message (NOTE: elimination of this clause will not effect program execution.)

C. Else, copy statement (or remaining statement with "SEGMENT" clause eliminated) to output area.

OUTPUT

OBJECT-COMPUTER.

computer-name

MEMORY SIZE integer

WORDS

CHARACTERS

MODULES

SEGMENT clause

***warning message

NO: 2.1.1.4

INPUT

SPECIAL-NAMES.

IS implementor-name

IS mnemonic-name

ON STATUS IS cond-1

OFF STATUS IS cond-2

OFF STATUS IS cond-2

ON STATUS IS cond-1

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SPECIAL-NAMES.

IS implementor-name

IS mnemonic-name

ON STATUS IS cond-1

OFF STATUS IS cond-2

OFF STATUS IS cond-2

ON STATUS IS cond-2

NO: 2.1.2

INPUT

INPUT-OUTPUT SECTION.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

INPUT-OUTPUT SECTION.

NO: 2.1.2.1.1

INPUT

FILE-CONTROL.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

FILE-CONTROL.

NO: 2.1.2.1.1

INPUT

SELECT file-name.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

SELECT file-name.

INPUT

ASSIGN TO implementor-

name=1

[,implementor-name-2]...

PROCESS

Read entire statement, then copy to output area.

OUTPUT

ASSIGN TO implementor-

name=1

[,implementor-name-2]...

NO: 2.1.2.1.4

INPUT

PROCESS

OUTPUT

[
RESERVE integer
AREA
AREAS
]

Read entire statement, then copy to output area.

[
RESERVE integer
AREA
AREAS
]

INPUT

ORGANIZATION IS

RELATIVE
SEQUENTIAL

PROCESS

- A. Read entire statement
- B. If organization is "INDEXED", move statement and warning message to output area.
- C. If organization is "RELATIVE" or "SEQUENTIAL", move statement to output area, and copy.

OUTPUT

ORGANIZATION IS

RELATIVE
SEQUENTIAL

***Warning message. Indexed option is not allowed.

NO: 2.1.2.1.6

INPUT

RECORD KEY IS data-name

PROCESS

- A. Read entire statement
- B. Since this clause is a no-equivalent situation, it should be moved to the output area with appropriate warning message.

NOTE: There may exist within the operating system, the means to simulate this process, since elimination of this clause may effect execution.

OUTPUT

RECORD statement
***WARNING message

INPUT

ALTERNATE RECORD KEY

IS data-name

WITH DUPLICATES

PROCESS

A. Read entire statement

B. Since there is no equivalent statement within the output language, this "ALTERNATE" statement should be moved to the output area with appropriate warning message.

NOTE: There may exist means within the operating system to simulate this statement as function, because elimination of statement may effect program's execution.

OUTPUT

ALTERNATE RECORD KEY

IS data-name

WITH DUPLICATES

*** WARNING message

NO: 2.1.2.1.1.1

INPUT

ACCESS, MODE IS

SEQUENTIAL RELATIVE

KEY IS data-name-1

RANDOM
DYNAMIC

RELATIVE

KEY IS data-name-2

PROCESS

Read entire statement, then copy to output area

OUTPUT

ACCESS MODE IS

SEQUENTIAL RELATIVE

KEY IS data-name-1

RANDOM
DYNAMIC

RELATIVE

KEY IS data-name-2

NO: 2.1.2.2.1

INPUT

I-O-CONTROL.

PROCESS

Read entire statement, then copy to output area.

OUTPUT

I-O-CONTROL.

NO: 2.1.2.2.2

INPUT

PROCESS

OUTPUT

RERUN ON

[file-name-1
implementor-name]

[REEL
END OF
UNIT]

OF
file-name-2

[integer-1 RECORDS]

[integer-2 CLOCK UNITS]

[condition-name]

RERUN statement

***WARNING message

- A. Read entire statement
- B. Since there isn't an equivalent statement within the output language, this statement should be eliminated and placed in the output area with appropriate error message.

NOTE: There may exist in the operating system the means for simulating this concept, because elimination of this statement may effect simulation.

NO: 2.1.2.2.3

INPUT

PROCESS

OUTPUT

SAME AREA FOR

file-name-1

[file-name-2] ...

SAME AREA FOR

file-name-1

[file-name-2] ...

Read Entire statement then copy to output area.

NO: 2.2

OUTPUT

DATA DIVISION.

PROCESS

Read entire statement, then copy to output area

INPUT

DATA DIVISION.

NO: 2.2.1

INPUT

FILE SECTION.

PROCESS

Read entire statement, then copy to output area

OUTPUT

FILE SECTION.

NO: 2.2.1.1

INPUT

FD file-description

PROCESS

Read entire statement and copy to output area

OUTPUT

FD file-description

NO: 2.2.1.1.1

INPUT

BLOCK CONTAINS

integer-2

RECORDS
CHARACTERS

PROCESS

Read entire statement, then copy to output area

OUTPUT

BLOCK CONTAINS

integer-2

RECORDS
CHARACTERS

NO: 2.2.1.1.2

INPUT

RECORD CONTAINS

[integer-3 TO]

integer-4 CHARACTERS

PROCESS

Read entire statement, then copy to output area

OUTPUT

RECORD CONTAINS

[integer-3 TO]

integer-4 CHARACTERS

INPUT

VALUE OF
implementor-name-l

IS literal-l

[REDACTED]
[REDACTED] ...

PROCESS

- A. Read entire statement
- B. Move "VALUE OF" to output area and translate implementor-name-l to "ID" within output area.
- C. Move "IS literal-l" to output area.
- D. If the optional implementor phrase has been used, it should be moved to the output area with a warning message (NOTE: This phrase's elimination may effect program execution).

OUTPUT

VALUE OF ID
VALUE OF ID IS literal-l
implementor option phrase
***WARNING message

NO: 2.2.1.1.4

INPUT

DATA { RECORD IS
RECORDS ARE

data-name-3

[, data-name-4] ...]

PROCESS

Read entire statement, then copy to output area

OUTPUT

DATA { RECORD IS
RECORDS ARE

data-name-3

[, data-name-4] ...]

NOT: 1.1.1.1.1.1

OUTPUT

LINEAGE IS { data-name-5
integer-5 }

LINES [WITH FOOTING

AT { data-name-6
integer-6 }

, LINES AT TOP { data-name-7
integer-7 }

, LINES AT BOTTOM

{ data-name-8
integer-8 }

PROCESS

Read entire statement, then copy to output area

INPUT

LINEAGE IS { data-name-5
integer-5 }

LINES [WITH FOOTING

AT { data-name-6
integer-6 }

, LINES AT TOP { data-name-7
integer-7 }

, LINES AT BOTTOM

{ data-name-8
integer-8 }

NO: 2.2.1.2

INPUT

SD file-name

PROCESS

- A. Read entire statement, with succeeding entries.
- B. Since there exists no equivalent statement within the output language, this "SD" statement, along with succeeding entries, should be moved to the output area with warning messages.

NOTE: The operating system may contain the means to simulate this process, since eliminating these statements may effect program execution.

OUTPUT

SD statement

***WARNING message
level descriptions

***WARNING messages

NO: 2.2.2

INPUT

WORKING-STORAGE

SECTION.

PROCESS

Read entire statement, then copy to output area

OUTPUT

WORKING-STORAGE

SECTION.

NO: 2.2.2.2

INPUT

BLANK WHEN ZERO

PROCESS

Read entire statement, then copy to output area

OUTPUT

BLANK WHEN ZERO

No: 2.2.2.3

INPUT

JUSTIFIED
JUST

RIGHT

PROCESS

Read entire statement, then copy to output area

OUTPUT

JUSTIFIED
JUST

RIGHT

NO: 2.2.2.4

INPUT

PICTURE
PIC IS character string

PROCESS

Read entire statement, then copy to output area

OUTPUT

PICTURE
PIC IS character string

NO: 2.2.2.5

INPUT

VALUE IS literal

PROCESS

Read entire statement, then copy to output area

OUTPUT

VALUE IS literal

NO: 2.2.2.6

INPUT

USAGE IS

COMPUTATIONAL

COMP

DISPLAY

INDEX

PROCESS

Read entire statement, then copy to output area

OUTPUT

USAGE IS

COMPUTATIONAL

COMP

DISPLAY

INDEX

NO: 2.2.2.7

INPUT

level - number	data - name - FILLER
----------------	---------------------------

PROCESS

Read entire statement, then copy to output area

OUTPUT

level - number	data - name - FILLER
----------------	---------------------------

NO: 2.2.2.8

INPUT

[REDEFINES data-name-2]

PROCESS

Read entire statement, then copy to output area

OUTPUT

[REDEFINES data-name-2]

NO: 2.2.2.9

INPUT

[SIGN IS { LEADING TRAILING

[SEPARATE CHARACTER]

PROCESS

Read entire statement, then copy to output area

OUTPUT

[SIGN IS { LEADING TRAILING

[SEPARATE CHARACTER]

INPUT

[OCCURS
integer-1 TO integer-2 TIMES
integer-2 TIMES
DEPENDING ON data-name-3]

[ASCENDING
DESCENDING] KEY IS

data-name-4 [data-name-5] ...

[INDEXED BY index-name-1
index-name ...]

PROCESS

Read entire statement, then copy to output area

OUTPUT

[OCCURS
integer-1 TO integer-2 TIMES
integer-2 TIMES
DEPENDING ON data-name-3]

[ASCENDING
DESCENDING] KEY IS

data-name-4 [data-name-5] ...

[INDEXED BY index-name-1
index-name ...]

NO: 2.2.2.11

PROCESS

Read entire statement, then copy to the output area

INPUT

SYNCHRONIZED
SYNC

LEFT
RIGHT

OUTPUT

SYNCHRONIZED
SYNC

LEFT
RIGHT

NO: 2.2.2.12

INPUT

66 data-name-1

RENAMES data-name-2

[THROUGH
THRU]

data-name-3

PROCESS

Read entire statement, then copy to output area

OUTPUT

66 data-name-1

RENAMES data-name-2

[THROUGH
THRU]

data-name-3

2.2.2.3

OUTPUT

Condition - none

VALUE IS
VALUES ARE

THROUGH
THRU
Iteral - 2

THROUGH
THRU
Iteral - 3

Iteral - 4

PROCESS

Read file to memory, then copy to output area

Condition - none

VALUE IS
VALUES ARE
Iteral - 1

THROUGH
THRU
Iteral - 2

THROUGH
THRU
Iteral - 3

Iteral - 4

NO: 2.2.3

INPUT

LINKAGE SECTION

PROCESS

- A. Read entire statement, with succeeding entries:
- B. Since there exists no equivalent statement within the output language, this "LINKAGE" statement, along with succeeding statements, should be moved to the output area with warning messages.

NOTE: The operating system may contain the means for simulating this concept, since eliminating these statements may effect execution.

OUTPUT

LINKAGE statement with
other entries

***WARNING message

NO: 2.3.1

OUTPUT

PROCEDURE DIVISION

USING clause
***WARNING message

PROCESS

- A. Read entire statement.
- B. Move "PROCEDURE DIVISION" to the output area.
- C. If the "USING para-name-1" option is used, then it should be moved to the output area with appropriate warning message.

NOTE: Elimination of this clause may effect execution, but, the operating system may contain means to simulate the "USING" process.

PROCEDURE DIVISION

[REDACTED]

[REDACTED]

NO: 2.3.1.1

INPUT

PROCESS

OUTPUT

ALTER procedure-name-1 TO

PROCEED TO

procedure-name-2

procedure-name-3 TO

PROCEED TO

procedure-name-4] ...

ALTER procedure-name-1 TO

PROCEED TO

procedure-name-2

procedure-name-3 TO

PROCEED TO

procedure-name-4

...

NO: 2.3.1.2

INPUT

CALL literal-1

[USING data-name-1]

[, data-name-2 ...]

PROCESS

A. Read entire statement

B. Since there is no equivalent, this statement should be eliminated, and sent to the output area with appropriate warning message.

NOTE: Elimination of this statement may effect program execution. Within particular operating systems, there exists means to perform calling or linking modules.

OUTPUT

CALL statement

***WARNING message

INPUT

COPY text-nameREPLACINGliteral-1
word-1BY
literal-2
word-2
...

PROCESS

- A. Read entire statement.
- B. If statement contains "identifier" option, then move statement to output area with warning message. (NOTE: Eliminating statement may effect execution of program)
- C. Else, copy statement to output area.

OUTPUT

COPY text-name REPLACING

identifier-1 BY

identifier-2

***WARNING message

COPY text-nameREPLACINGliteral-1
word-1

BY

literal-2
word-2
...

NO: 2.3.1.4

OUTPUT

ENTER statement
***WARNING message

PROCESS

- A. Read entire statement
B. Since there is no equivalent verb for the output source, move the statement to the output area with appropriate error message.

NOTE: Elimination of this statement may effect execution, but there may exist means within the operating system to simulate this "ENTER" process.

ENTER

ENTER statement
routing message

NO: 2.3.1.5

OUTPUT

EXIT

PROCESS

Read entire statement, then copy to output area.

INPUT

EXIT

NO: 2.3.1.6.1

INPUT

GO TO [procedure - name - I]

PROCESS

Read entire statement, then copy to output area.

OUTPUT

GO TO [procedure - name - I]

NO: 2.3.1.6.2

INPUT

GO TO procedure-name-1

[, procedure-name-2] ...

, procedure-name-n

DEPENDING ON

identifier

PROCESS

Read entire statement, then copy to output area.

OUTPUT

GO TO procedure-name-1

[, procedure-name-2] ...

, procedure-name-n

DEPENDING ON

identifier

NO: 2.3.1.7.1

INPUT

PERFORM procedure-name-1

[THRU procedure-name-2

{ identifier-1
integer-1 } TIMES

UNTIL condition-1]

PROCESS

Read entire statement, then copy to output area

OUTPUT

PERFORM procedure-name-1

[THRU procedure-name-2

{ identifier-1
integer-1 } TIMES

UNTIL condition-1]

NO: 2.3.1.8

INPUT

RETURN file-name RECORD

[INTO identifier]

AT END imperative-

statement

PROCESS

A. Read entire statement

B. There exists no equivalent statement in the output language. This statement should be eliminated, and moved to the output area with appropriate warning message.

NOTE: There may exist means within the JCL or operating system, which simulates this RETURN function. Eliminating this statement may effect execution.

OUTPUT

RETURN statement

***WARNING message

INPUT

START file-name

IS EQUAL TO
 IS =
 IS GREATER THAN
 KEY { IS
 IS NOT LESS THAN
 IS NOT

data-name

PROCESS

A. Read entire statement and copy as is

OUTPUT

START file-name

IS EQUAL TO
 IS =
 IS GREATER THAN
 KEY { IS
 IS NOT LESS THAN
 IS NOT

data-name

NO: 2.3.1.10

INPUT

STOP
RUN
literal

PROCESS

A. Read entire statement and copy as is

OUTPUT

STOP
RUN
literal

NO: 2.3.2.1

INPUT

ACCEPT identifier

[FROM mnemonic-name]

PROCESS

Read entire statement then copy to output area

OUTPUT

ACCEPT identifier

[FROM mnemonic-name]

NO: 2.3.2.2

INPUT

CLOSE file-name-1

[,file-name-2]

PROCESS

Read entire statement, then copy to output area

OUTPUT

CLOSE file-name-1

[,file-name-2]

NO: 2.3.2.3

PROCESS

OUTPUT

Read entire statement, then copy to output area

DELETE file-name RECORD

INVALID KEY

INVALID KEY

imperative-statement

imperative-statement

NO: 2.3.2.4

INPUT

DISPLAY { identifier-1
literal-1 }

[, identifier-2
, literal-2] ...

PROCESS

Read entire statement, then copy to output area

OUTPUT

DISPLAY { identifier-1
literal-1 }

[, identifier-2
, literal-2] ...

INPUT

MERGE file-name-1 ON

ASCENDING }
DESCENDING } KEY

data-name-1
[, data-name-2] ...

[ON { ASCENDING } KEY
 DESCENDING }

data-name-3

[, data-name-4] ...

USING file-name-2, file-

name-3 [, file-name-4] ...

(CONTINUED ON NEXT PAGE)

PROCESS

A. Read entire statement

B. Since there is no equivalent statement, then
this statement should be moved to the output
area with a warning message.

NOTE: To simulate this merge function, a series
of statements for reading, checking and writing
may be set-up for the multiple files, but,
efficiency may be lost.

OUTPUT

MERGE statement
***WARNING message

INPUT

OUTPUT PROCEDURE IS
 section - name - 1
 GIVING file - name - 5
 THROUGH } section - name - 2
 THRU }

PROCESS

OUTPUT

READ file - name - 1
 READ file - name - 2
 IF condition - 1 THEN
 imperative - statement .
 (where the ASCENDING or
 DESCENDING condition
 is simulated and / or
 checking
 WRITE file - name - 3

INPUT

OPEN

{ INPUT file-name-1

OUTPUT file-name-3

{ I-O file-name-5

[,file-name-2] ...

[,file-name-4] ...

[,file-name-6] ...

PROCESS

Read entire statement, then copy to output area

OUTPUT

OPEN

{ INPUT file-name-1

OUTPUT file-name-3

{ I-O file-name-5

[,file-name-2] ...

[,file-name-4] ...

[,file-name-6] ...

NO: 2.3.2.7.1

INPUT

READ file-name [NEXT]
RECORD [INTO identifier]
[AT END imperative -
statement]

PROCESS

Read entire statement, then copy to output area

OUTPUT

READ file-name [NEXT]
RECORD [INTO identifier]
[AT END imperative -
statement]

INPUT

READ file-name RECORD

[INTO identifier]

[]

[INVALID KEY]

[imperative-statement]

PROCESS

A. Read entire statement

B. If "KEY IS" option is used, then it must be eliminated from the output language, by moving clause to output area with appropriate warning message. (NOTE: Elimination of this clause may effect execution.)

C. Else copy statement to output area

OUTPUT

KEY clause

**WARNING statement

READ file-name RECORD

[INTO identifier]

[INVALID KEY]

[imperative-statement]

NO: 2.3.2.8

INPUT

RELEASE record - name

[FROM identifier]

PROCESS

A. Read entire statement

B. There isn't an equivalent statement in the output language. This statement should be eliminated and moved to the output area with appropriate warning message.

NOTE: There may exist means within the operating system, which may simulate this RELEASE function. Eliminating this statement may effect execution.

OUTPUT

RELEASE statement
***WARNING message

NO: 2.3.2.9

INPUT

REWRITE record-name

[FROM identifier]

[INVALID KEY]

imperative-statement

PROCESS

Read entire statement, then copy to output area

OUTPUT

REWRITE record-name

[FROM identifier]

[INVALID KEY]

imperative-statement

INPUT

WRITE record-name

[FROM identifier-1]

[BEFORE
AFTER] ADVANCING

[identifier-2
integer] [LINE
LINES]

PROCESS

- A. Read entire statement
- B. If the option "mnemonic-name" exists within the format, it should be eliminated. The operating system has commands for simulating this option.
- C. Statement should be copied to the output error without "mnemonic-name"

OUTPUT

WRITE record-name

[FROM identifier-1]

[BEFORE
AFTER] ADVANCING

[identifier-2
integer] [LINE
LINES]

NO: 2.3.2.10.2

INPUT

PROCESS

OUTPUT

WRITE record-name

[FROM identifier]

[INVALID KEY

imperative-statement]

Read entire statement, then copy to output area

WRITE record-name

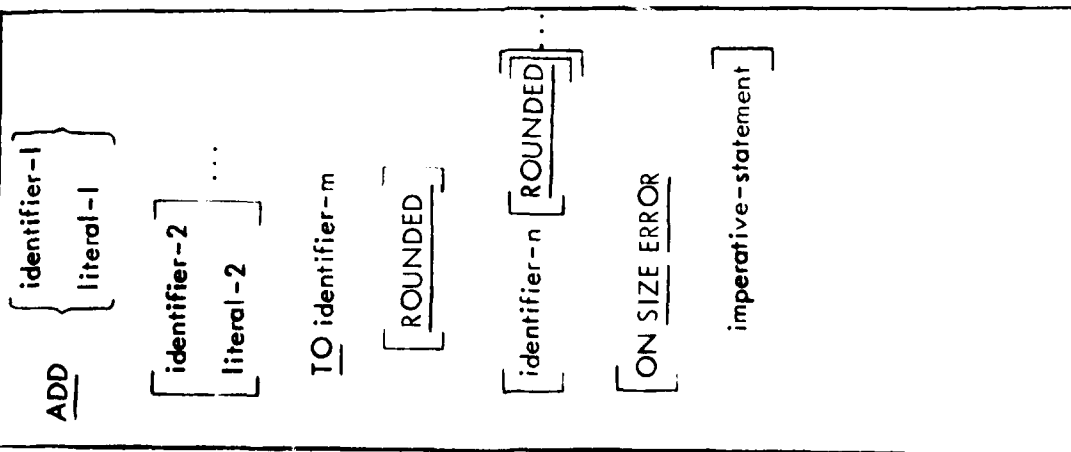
[FROM identifier]

[INVALID KEY

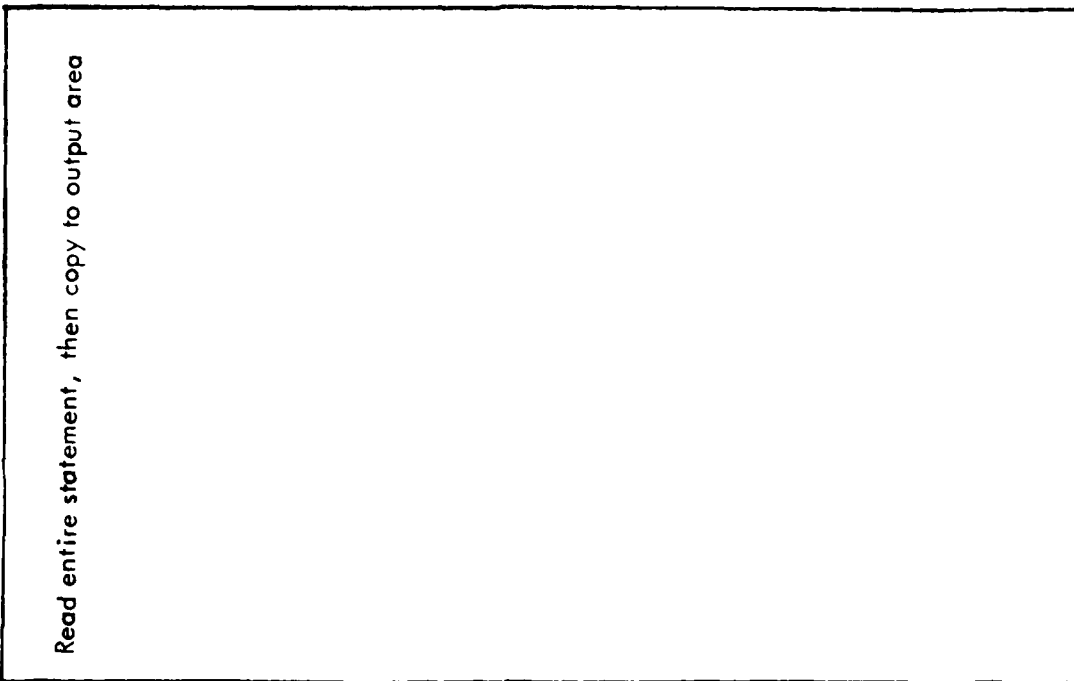
imperative-statement]

NO: 2.3.3.1.1

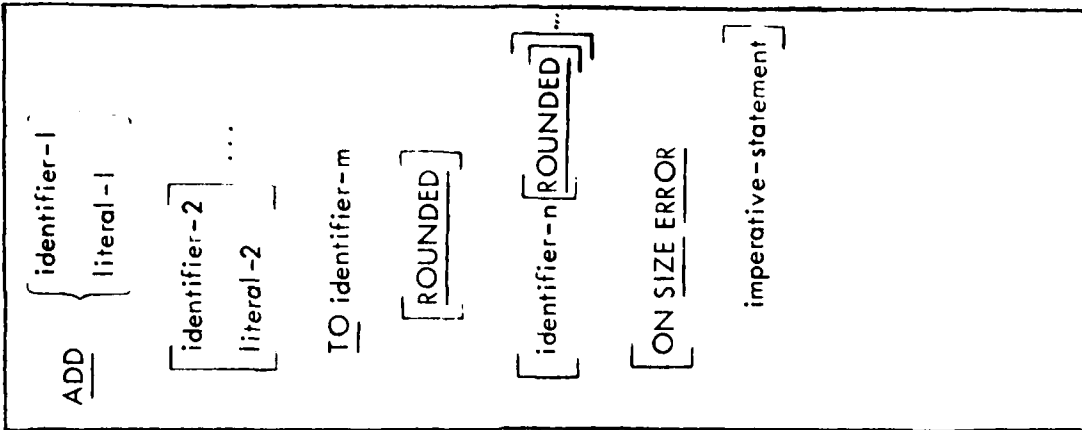
INPUT



PROCESS



OUTPUT



NO: 2.3.3.1.2

INPUT

PROCESS

OUTPUT

ADD

{ identifier-1
literal-1 }

{ identifier-2
literal-2 }

[identifier-3
literal-3] ...

GIVING identifier-m

[ROUNDED]

[ON SIZE ERROR

imperative-statement]

Read entire statement, then copy to output area

ADD

{ identifier-1
literal-1 }

{ identifier-2
literal-2 }

[identifier-3
literal-3] ...

GIVING identifier-m

[ROUNDED]

[ON SIZE ERROR

imperative-statement]

NO: 2.3.3.1.3

INPUT

PROCESS

OUTPUT

ADD { CORRESPONDING
CORR }

identifier-1

TO identifier-2 ROUNDED

[ON SIZE ERROR

imperative-statement]

Read entire statement, then copy to output area

ADD { CORRESPONDING
CORR }

identifier-1

TO identifier-2 ROUNDED

[ON SIZE ERROR

imperative-statement]

AD-A107 879

SAI COMSYSTEMS CORP MCLEAN VA
SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES, (U)
JUN 77 T DENIKE, A HOLLAND, T WARD, H DESAI DAHC26-76-D-1004
USACSC-AT-77-11 NL

F/8 9/2

UNCLASSIFIED

3 of 3
AD-A107 879



END
DATE
FILMED
4-82
DTIC

NO: 2.3.3.2

INPUT

COMPUTE identifier-1

ROUNDED

, identifier-2 ROUNDED ...

= arithmetic-expression

ON SIZE ERROR

imperative-statement

PROCESS

Read entire statement, then copy to output area

OUTPUT

COMPUTE identifier-1

ROUNDED

, identifier-2 ROUNDED ...

= arithmetic-expression

ON SIZE ERROR

imperative-statement

NO: 2.3.3.3.1

INPUT

PROCESS

OUTPUT

DIVIDE { identifier-1
literal-1 }

INTO identifier-2

[ROUNDED]

[, identifier-3

[ROUNDED] ...

[ON SIZE ERROR

] imperative-statement]

DIVIDE { identifier-1
literal-1 }

INTO identifier-2

[ROUNDED]

[, identifier-3

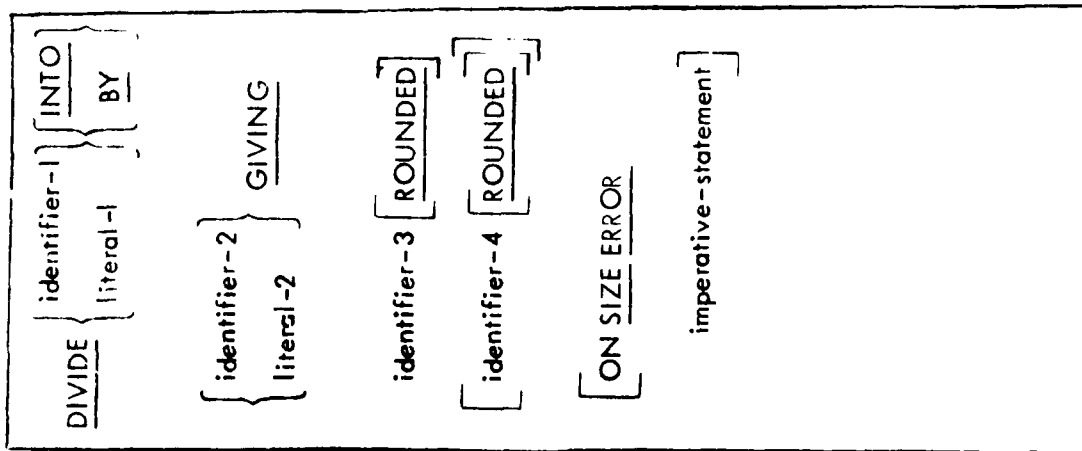
[ROUNDED] ...

[ON SIZE ERROR

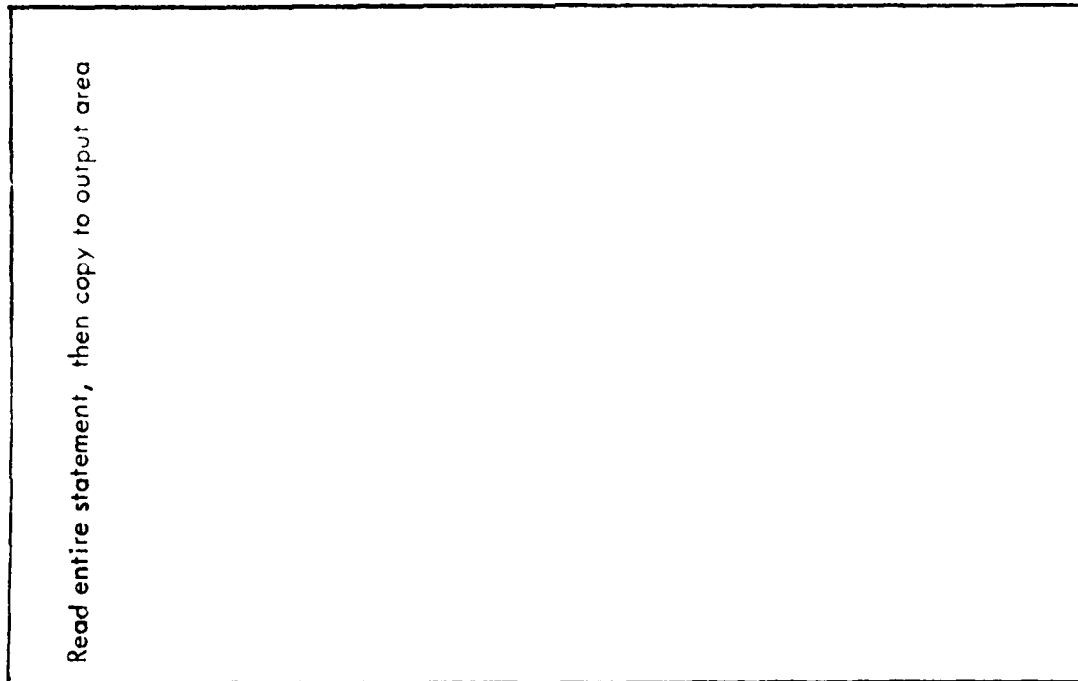
] imperative-statement]

Read entire statement, then copy to output area

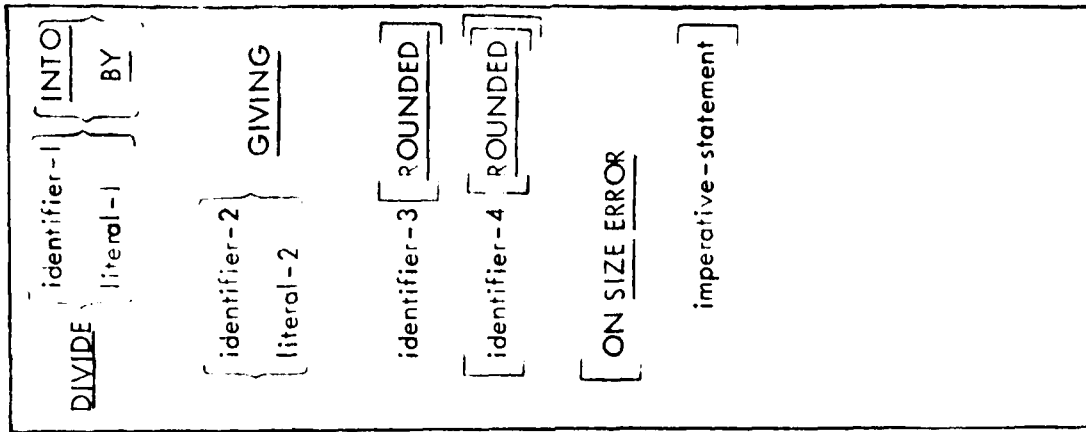
INPUT



PROCESS



OUTPUT



NO: 2.3.3.4.1

OUTPUT

INSPECT identifier-1

REPLACING

{ CHARACTERS BY { identifier-6 }
literal-4 }

{ BEFORE
AFTER } INITIAL

{ identifier-7 }
literal-5 }

{ { ALL
LEADING
FIRST } { identifier-5 }
literal-3 }

BY { identifier-6 } { BEFORE
literal-4 } { AFTER }

INITIAL { identifier-7 }
literal-5 }

PROCESS

Read entire statement, then copy to output area

INPUT

INSPECT identifier-1

REPLACING

{ CHARACTERS BY { identifier-6 }
literal-4 }

{ BEFORE
AFTER } INITIAL

{ identifier-7 }
literal-5 }

{ { ALL
LEADING
FIRST } { identifier-5 }
literal-3 }

BY { identifier-6 } { BEFORE
literal-4 } { AFTER }

INITIAL { identifier-7 }
literal-5 }

INPUT

INSPECT identifier-1

TALLYING

identifier-2 FOR

{ ALL } identifier-3
 { LEADING } literal-1

CHARACTERS

{ BEFORE }
 { AFTER }

INITIAL

identifier-4

literal-2

(CONTINUED ON NEXT PAGE)

PROCESS

Read entire statement, then copy to output area

OUTPUT

INSPECT identifier-1

TALLYING

identifier-2 FOR

{ ALL } identifier-3
 { LEADING } literal-1

CHARACTERS

{ BEFORE }
 { AFTER }

INITIAL

identifier-4

literal-2

NO: 2.3.3.3.3

INPUT

<u>DIVIDE</u>	{ identifier-1 literal-1 }	{ INTO BY }
	{ identifier-2 literal-2 }	<u>GIVING</u>
	identifier-3	<u>ROUNDED</u>
	<u>REMAINDER identifier-4</u>	
	<u>ON SIZE ERROR</u>	
	imperative-statement	

PROCESS

Read entire statement, then copy to output area

OUTPUT

<u>DIVIDE</u>	{ identifier-1 literal-1 }	{ INTO BY }
	{ identifier-2 literal-2 }	<u>GIVING</u>
	identifier-3	<u>ROUNDED</u>
	<u>REMAINDER identifier-4</u>	
	<u>ON SIZE ERROR</u>	
	imperative-statement	

INPUT

PROCESS

OUTPUT

MULTIPLY { identifier-1
literal-1 }

BY identifier-2 [ROUNDED]

[identifier-3 [ROUNDED] ...]

[ON SIZE ERROR]

imperative-statement

Read entire statement, then copy to output area

MULTIPLY { identifier-1
literal-1 }

BY identifier-2 [ROUNDED]

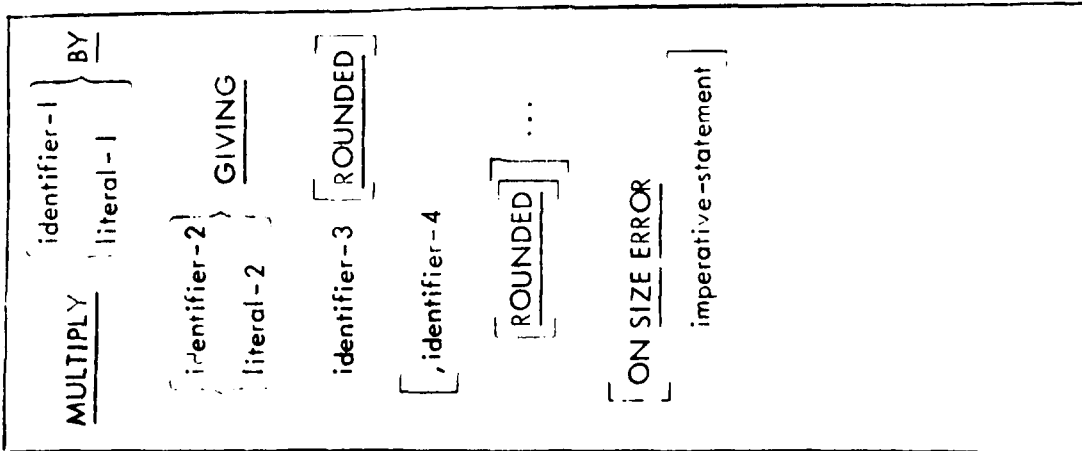
[identifier-3 [ROUNDED] ...]

[ON SIZE ERROR]

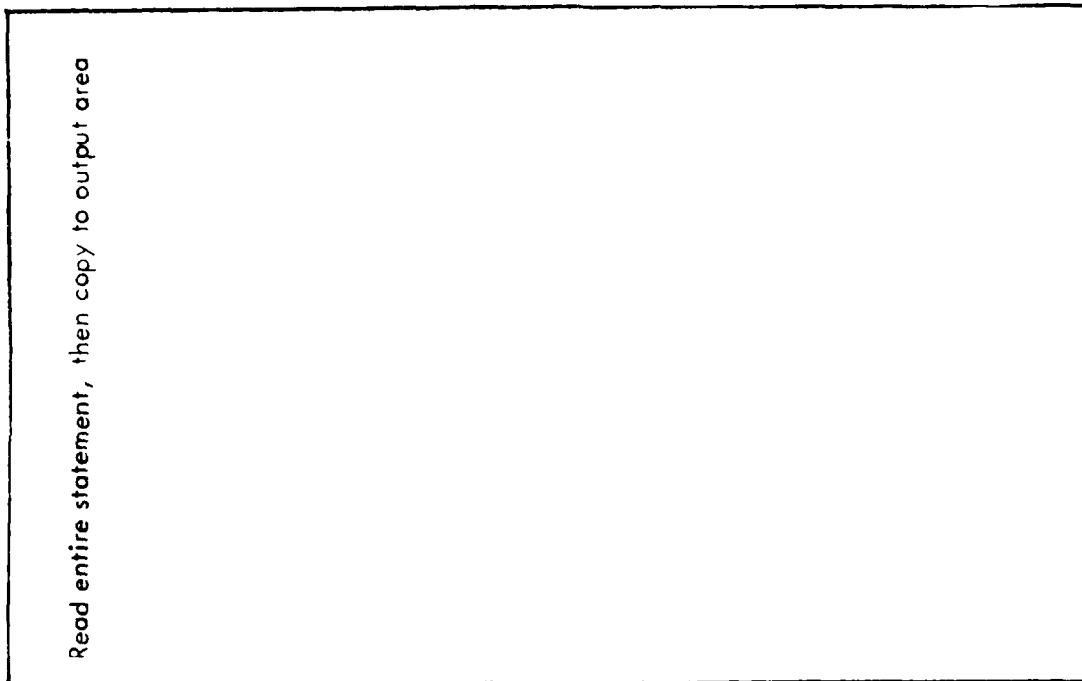
imperative-statement

NO: 2.3.3.5.2

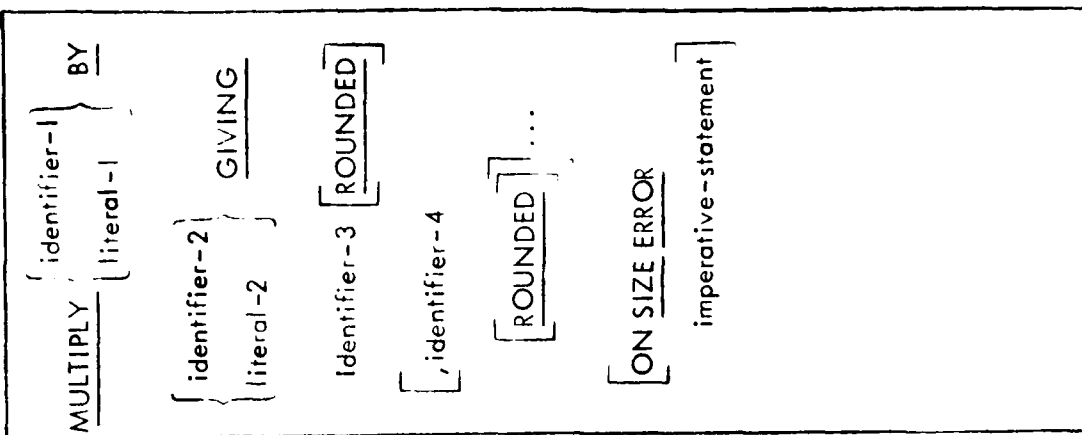
INPUT



PROCESS



OUTPUT



NO: 2.3.3.6.1

INPUT

SUBTRACT

[identifier-1
literal-1]

[, identifier-2
literal-2]

FROM identifier-m

[ROUNDED]

[, identifier-n [ROUNDED]

...

[, ON SIZE ERROR
imperative-statement]

PROCESS

Read entire statement and copy to output area

OUTPUT

SUBTRACT [identifier-1
literal-1]

[, identifier-2
literal-2]

...

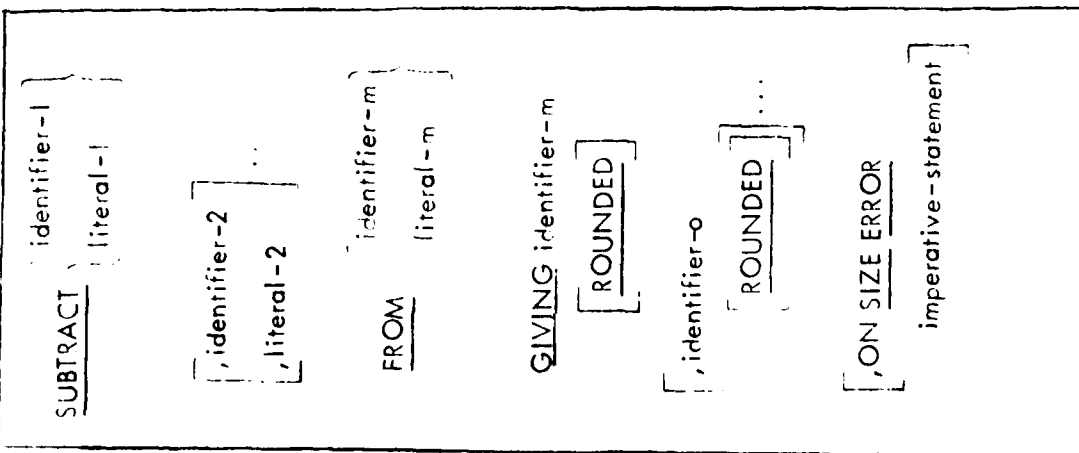
FROM identifier-m

[ROUNDED]

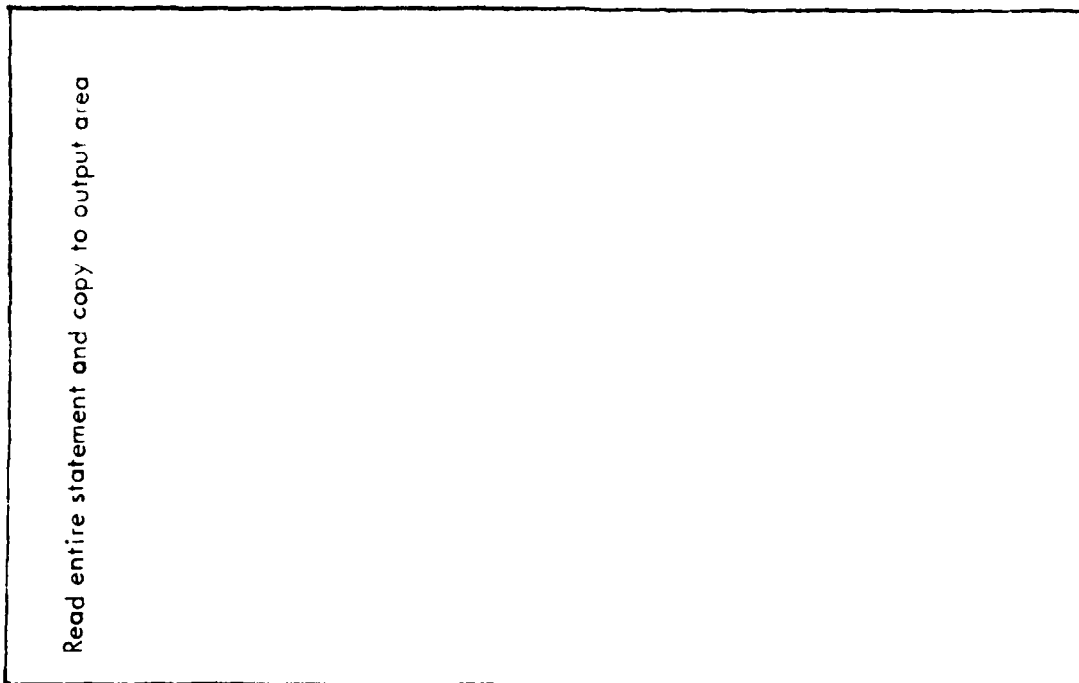
[, identifier-n [ROUNDED]

[, ON SIZE ERROR
imperative-statement]

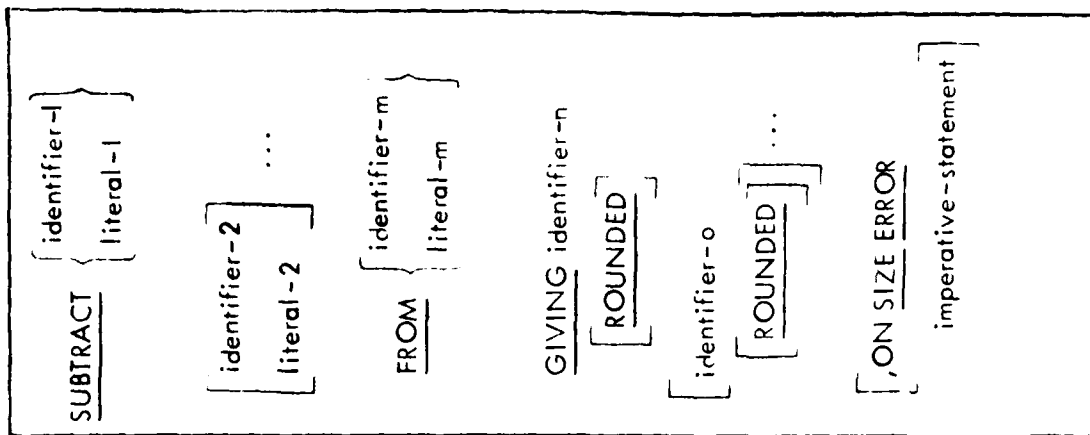
INPUT



PROCESS



OUTPUT



NO: 2.3.3.6.3

INPUT

PROCESS

OUTPUT

SUBTRACT { CORRESPONDING
CORR

identifier-1 FROM

identifier-2 [ROUNDED

[; ON SIZE ERROR
imperative-statement]

Read entire statement and copy to output area

SUBTRACT { CORRESPONDING
CORR

identifier-1 FROM

identifier-2 [ROUNDED

[ON SIZE ERROR
imperative-statement]

NO: 2.3.4.1.1

INPUT

PROCESS

OUTPUT

MOVE { identifier-1 } TO
 literal

 identifier-2

 [, identifier-3] ...

Read entire statement, then copy to output area

MOVE { identifier-1 } TO
 literal

 identifier-2

 [, identifier-3] ...

NO: 2.3.4.1.2

INPUT

PROCESS

OUTPUT

MOVE { CORRESPONDING
CORR }

identifier-1 TO
identifier-2

Read entire statement, then copy to output area

MOVE { CORRESPONDING
CORR }

identifier-1 TO
identifier-2

NO: 2.3.4.2.1

INPUT

SEARCH identifier-1

VARYING

{ identifier-2
index-name-1 }

AT END imperative -

statement-1

WHEN condition-1

{ imperative-statement
NEXT SENTENCE }

WHEN condition-2

{ imperative-statement-3
NEXT SENTENCE }

...

PROCESS

Read entire statement, then copy to output area

OUTPUT

SEARCH identifier-1

VARYING

{ identifier-2
index-name-1 }

AT END imperative -

statement-1

WHEN condition-1

{ imperative-statement
NEXT SENTENCE }

WHEN condition-2

{ imperative-statement-3
NEXT SENTENCE }

...

INPUT

```

SEARCH ALL identifier-1
[ AT END imperative-
statement-1 ]
{
  data-
  name-1 { IS EQUAL TO }
  {
    identifier-3
    { literal-1
    { arithmetic-exp-1 }
    condition-name-1
  }
}
AND
{
  data-
  name-2 { IS EQUAL TO }
  {
    identifier-4
    { literal-2
    { arithmetic-exp-2 }
    condition-name-2
  }
}
imperative-statement-2
NEXT SENTENCE
  
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```

SEARCH ALL identifier-1
[ AT END imperative-
statement-1 ]
{
  data-
  name-1 { IS EQUAL TO }
  {
    identifier-3
    { literal-1
    { arithmetic-exp-1 }
    condition-name-1
  }
}
AND
{
  data-
  name-2 { IS EQUAL TO }
  {
    identifier-4
    { literal-2
    { arithmetic-exp-2 }
    condition-name-2
  }
}
imperative-statement-2
NEXT SENTENCE
  
```

NO: 2.3.4.3.1

INPUT

SET { identifier-1
 index-name-1 }

[identifier-2] ...
[index-name-2] ...

TO

{ identifier-3
 index-name-3
 integer-1 }

PROCESS

Read entire statement and copy to output area

OUTPUT

SET { identifier-1
 index-name-1 }

[identifier-2] ...
[index-name-2] ...

TO

{ identifier-3
 index-name-3
 integer-1 }

NO: 2.3.4.3.2

INPUT

SET index-name-4

[, index-name-5] ...

{ UP BY identifier-4
DOWN BY integer-2 }

PROCESS

Read entire statements and copy to output area

OUTPUT

SET index-name-4

[index-name-5] ...

{ UP BY identifier-4
DOWN BY integer-2 }

INPUT

PROCESS

OUTPUT

STRING { identifier-1
literal-1 }
...
{ identifier-2
literal-2 }
...
DELIMITED BY { identifier-3
literal-3 }
SIZE
{ identifier-4
literal-4 } { identifier-5
literal-5 }
...
DELIMITED BY { identifier-6
literal-6 }
SIZE
INTO identifier-7
[WITH POINTER identifier-8]
[ON OVERFLOW
imperative-statement]

Read entire statements and copy to output area

STRING { identifier-1
literal-1 }
...
{ identifier-2
literal-2 }
...
DELIMITED BY { identifier-3
literal-3 }
SIZE
{ identifier-4
literal-4 } { identifier-5
literal-5 }
...
DELIMITED BY { identifier-6
literal-6 }
SIZE
...
INTO identifier-7
[WITH POINTER identifier-8]
[ON OVERFLOW
imperative-statement]

INPUT

UNSTRING identifier-1

[DELIMITED BY [ALL]

{ identifier-2
literal-1 }

[, OR [ALL] { identifier-3
literal-2 }

...] INTO identifier-4

[DELIMITER IN identifier-5]

[COUNT IN identifier-6]

[identifier-7

[DELIMITER IN identifier-8]

[COUNT IN identifier-9] ...

(CONTINUED ON NEXT PAGE)

PROCESS

Read entire statement, then copy to output area

OUTPUT

UNSTRING identifier-1

[DELIMITED BY [ALL]

{ identifier-2
literal-1 }

[, OR [ALL] { identifier-3
literal-2 }

...] INTO identifier-4

[DELIMITER IN identifier-3]

[COUNT IN identifier-6]

[identifier-7

[DELIMITER IN identifier-8]

[COUNT IN identifier-9] ...

INPUT

[WITH POINTER identifier-10]

[TALLYING IN identifier-11]

[ON OVERFLOW]

imperative-statement

PROCESS

OUTPUT

[WITH POINTER identifier-10]

[TALLYING IN identifier-11]

[ON OVERFLOW]

imperative-statement

NO: 2.3.5.1

INPUT

IF condition { statement-1
NEXT SENTENCE
ELSE statement-2
ELSE NEXT SENTENCE

PROCESS

Read entire statement, then copy to output area

OUTPUT

IF condition { statement-1
NEXT SENTENCE
ELSE statement-2
ELSE NEXT SENTENCE

