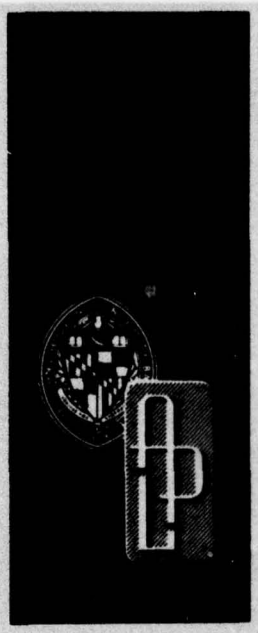
AD-A045 472 JOHNS HOPKINS UNIV LAUREL MD APPLIED PHYSICS LAB F/G 9/2
MACRO INSTRUCTIONS AND THEIR PROCESSOR FOR STRUCTURED PROGRAMMI--ETC(U)
JUN 77 S W KAHNG N00017-72-C-4401
UNCLASSIFIED APL/JHU/TG-1308 NL

| OF |
AD
AD45472

END
DATE FILMED
11-77
DDC

APL/JHU
TG 1308
JUNE 1977
Copy No. 2



Technical Memorandum

MACRO INSTRUCTIONS AND THEIR PROCESSOR FOR STRUCTURED PROGRAMMING WITH ASSEMBLY LANGUAGES

S. W. KAHNG

THE JOHNS HOPKINS UNIVERSITY ■ APPLIED PHYSICS LABORATORY

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

PLEASE FOLD BACK IF NOT NEEDED FOR BIBLIOGRAPHIC PURPOSES

REPORT DOCUMENTATION PAGE

1. REPORT NUMBER: APL/JHU TG-1308	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): MACRO INSTRUCTIONS AND THEIR PROCESSOR FOR STRUCTURED PROGRAMMING WITH ASSEMBLY LANGUAGES.	5. TYPE OF REPORT & PERIOD COVERED: Technical Memorandum
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): S. W. Kahng	8. CONTRACT OR GRANT NUMBER(s): N00017-72-C-4401
9. PERFORMING ORGANIZATION NAME & ADDRESS: The Johns Hopkins University Applied Physics Laboratory, Johns Hopkins Road, Laurel, MD 20810	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Task B411F3C
11. CONTROLLING OFFICE NAME & ADDRESS: Naval Plant Representative Office, Johns Hopkins Road, Laurel, MD 20810	12. REPORT DATE: June 1977
	13. NUMBER OF PAGES: 40
14. MONITORING AGENCY NAME & ADDRESS: Naval Plant Representative Office, Johns Hopkins Road, Laurel, MD 20810	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distribution unlimited.	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	
18. SUPPLEMENTARY NOTES	

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

macro instructions
macro processor
structured programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The description of macro instructions for structured programming and organization of their processor are presented. The macro instructions generate if-then-else, repeat-while, and for statements with the assembly language programming, and the processor translates these macro statements into the assembly codes.

DD FORM 1 JAN 73 1473 031 650

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

APL/JHU
TG 1308
JUNE 1977

Technical Memorandum

MACRO INSTRUCTIONS AND THEIR PROCESSOR FOR STRUCTURED PROGRAMMING WITH ASSEMBLY LANGUAGES

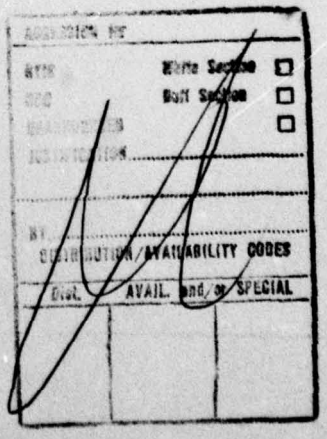
S. W. KAHNG

THE JOHNS HOPKINS UNIVERSITY ■ APPLIED PHYSICS LABORATORY
Johns Hopkins Road, Laurel, Maryland 20810
Operating under Contract N00017-72-C-4401 with the Department of the Navy

THE JOHNS HOPKINS UNIVERSITY
APPLIED PHYSICS LABORATORY
LAUREL, MARYLAND

ABSTRACT

The description of macro instructions for structured programming and the organization of their processor are presented. The macro instructions generate if-then-else, repeat-while, and for statements with the assembly language programming, and the processor translates these macro statements into the assembly codes.



CONTENTS

1. INTRODUCTION

There have been a number of papers published during the last several years on the use of macro instructions as a means of practicing structured programming with assembly languages (Refs. 1 through 5). The authors of these papers have all experienced marked improvement in program readability and productivity through the judicious practice of structured programming using macro instructions.

It is interesting to note that most of these writers developed macro instructions for small computers. Assembly languages are popular among programmers of small computers in spite of the known low productivity associated with them. This unfortunate situation results from the fact that inadequacy and inefficiency of the high-level languages and their compilers are more strongly felt for many small computer applications. More specifically:

1. Most high-level languages are for general purpose use, yet they are not general enough to meet easily the needs of the variety of tasks for small computer applications.

2. High-level languages do not let the user directly take advantage of the host computer facilities.

3. High-level language programs are frequently translated into machine codes with unacceptably inefficient core utilizations and execution speeds for many small computer applications.

Ref. 1. C. W. Barth, "STRCMACS — An Extensive Set of Macros to Aid in Structured Programming in 360/370 Assembly Language," SIGPLAN Notices, Vol. 22, No. 8, 1976, pp. 30-35.

Ref. 2. G. S. Herman-Giddens, R. B. Warren, R. C. Barr, and M. S. Spach, "BIOMAC — Block Structured Programming Using PDP-11 Assembler Language," SOFTWARE: Practice and Experiences, Vol. 5.

Ref. 3. S. W. Kahng, "Structured Programming and Debugging Aids with MACRO-11 Assembly Language for the PDP-11 Computer," APL/JHU TG 1294, April 1976.

Ref. 4. C. Pepper, "SMAL — A Structured Macro-Assembly Language for a Microprocessor," IEEE COMPCON Fall Digest of Papers, September 1974, pp. 147-151.

Ref. 5. G. E. Riens, "Structured Programming in Assembly Language," Datamation, July 1976, pp. 79-84.

To overcome existing problems, machine-oriented high-level languages (MOHLL) have been developed for several computers (PL-360 for the IBM 360 computer is the best-known MOHLL). They are high-level languages that are oriented to the host computer to take advantage of all or most of the facilities with a sacrifice in program transportability with other types of computers. MOHLL is superior to the macro instructions. However, it suffers from a few disadvantages. The most important one is the unavailability of good MOHLL for most computers. When such MOHLL are not available, macro instructions can be developed easily and can satisfy many of the advantages of MOHLL. This paper will describe examples of macro definitions that can collectively satisfy the objectives reasonably well. It is believed that they are not original contributions; the use of similar macros has already been described (Refs. 1 through 5). However, we believe this paper is still relevant for the following reasons:

1. The contents of Section 4 of this paper are not readily available to practicing programmers, and

2. It is not yet widely known that a substantial benefit can be derived from the use of these macros at an insignificant cost.

In Section 2 we discuss further the MOHLL and macro instructions. In Section 3 the constructs of three macro instructions are defined, and in Section 4 each subprogram of the processor is described. Concluding remarks are given in Section 5. A processor program for the PDP-11 computer is listed in the Appendix.

2. MOHLL AND MACRO INSTRUCTIONS

It has been claimed (Refs. 6 through 8) that the disadvantages of high-level languages as systems languages are mostly overcome through MOHLL. Productivity with MOHLL is higher than that with the assembly languages; the compiled machine codes are more efficient than those from the general purpose high-level languages. In fact, some of the MOHLL and their compiled codes are so impressive that MOHLL was proposed to replace assembly languages for programming purposes (Refs. 6 through 8).

Advantages of using MOHLL are not limited to these. Through the suitable MOHLL, a programmer can practice structured programming and produce readable codes. If, in addition, the MOHLL compiler can be easily modified by the users to add new processing capabilities, or if it has an extensive macro capability, then further programming conveniences can be gained through tailoring the MOHLL for the given task. As an example, if a programming task frequently requires double-precision integer arithmetic, it is desirable to add such arithmetic expression processing capability to the MOHLL. These capabilities need not be general purpose. A simple and efficient special purpose processor that meets the need of the specific task will be more desirable. Through this process one can modify and extend the MOHLL to suit the given task. The advantages of extending MOHLL in this way are:

1. The program becomes more readable.

2. There will be less need to modify the algorithm to fit the programming language.

3. Coding will be made, or nearly made, as the end product of the successively refined program-design process, and not as a separate effort from the design.

Ref. 6. R. Conradi, P. Holager, O1 Solberg, and G. Green, "A System for Software Development on Minicomputers," Minicomputer Software, North-Holland Publishing Co., 1976, pp. 15-29.

Ref. 7. H. Lehessaari, "A Family of Machine Oriented Higher Level Languages (MOHLL)," Minicomputer Software, North-Holland Publishing Co., 1976, pp. 231-237.

Ref. 8. R. D. Russell, "Intermediate-Level Programming Languages for On-Line Data Acquisition and Control," Computer Physics Communications, Vol. 5, 1973, pp. 89-97.

The first item is obvious. A simple case of algorithm modification (the second item) is the use of 0 as a subscript. It is not allowed in FORTRAN while it can be desirable in the description of the algorithm. The third item needs explanation. We assume that the program design is made through a language whose syntax is adapted from that of the MOHLL. If the adaptation is very loose at the top level of the design but becomes successively closer to that of the MOHLL with the progressive refinement of the design, we will end up with (or almost end up with) the program code. Such smooth transition should be possible if the MOHLL is well tailored to the programming task.

So far we have discussed many advantages of a desirable MOHLL. Unfortunately there is no language that provides all these advantages. However, we can fairly easily produce macro instructions that provide many of them.

In order to realize these advantages it is desirable to have a good macro language and its processor. Many macro languages are versatile, but most are not suitable for producing easily readable macro definitions. Those for small computers are often inadequate. On the other hand, for a special type of macro instruction (as described in Section 3), a high-level language like FORTRAN can be used, along with a few subroutines, for developing their processor with relative ease. In fact, its versatility, code readability, and processing speed surpass those of many macro processors. For these reasons and for portability it is frequently desirable to develop a special macro processor rather than to rely on the available general purpose macro processors.

The relative ease of the special purpose macro processor development enables us to add new conveniences to the host language (assembly language in this case) to suit the given software task. One example is the arithmetic formula processing capability (e.g., A = (B + C * D)/E). One can also include the program developing aids such as the tracing option (Ref. 2) or a partial program verification procedure in the processor. These may be used as necessary.

3. MACRO CONSTRUCTS

Macro instructions considered here are if, repeat-while, and for statements. Their constructs are shown below; they also show which terms of the statements can start new lines.

IF STATEMENT

<if clause> ::= <if> sd <logical operator> sd,

where sd is the source or destination of the assembly language instruction.

<if> ::= IF | IFB | IFF

<if chain> ::= <if clause> | <if clause> { AND / OR } <if chain>

<if statement> ::= <if chain> THEN <one assembly instruction>

or <if chain> THEN

<a block of instructions>

ENDIF

or <if chain> THEN

<a block of instructions>

ELSE <one assembly instruction>

or <if chain> THEN

<a block of instructions>

ELSE

<a block of instructions>

ENDIF

<logical operator> ::= GT|GE|EQ|LE|LT|NE

The distinctions between IF, IFB, and IFF are that the terms in the clause are taken as words, bytes, or floating point numbers,

respectively. A block of instructions contains one or more assembly or macro instruction — if, repeat-while, or for statements.

The terms AND, OR, and THEN can also start new lines. For example:

```
IF A NE B THEN CLR X


IF A GT B AND IFB C EQ D OR IFF E EQ F THEN
    CLR X
ELSE
    CLR Y
ENDIF


IF A GT B
AND
IFB C EQ D
OR
IFF E EQ F
THEN
    CLR X
ELSE
    CLR Y
ENDIF
```

REPEAT-WHILE STATEMENT

We divert from the ordinary while statement and use the repeat-while statement. The difference is shown in Fig. 1. The reason for using the repeat-while statement is that, unlike in a high-level language, we only allow very simple logical expression here. Consequently an expression like

```
WHILE ABS(A) LT D DO
```

(which means: while the absolute value of A is less than D, do the following) cannot be expressed easily by an ordinary while macro in a readable and simple manner. The following repeat-while statements are acceptable:

<while clause> ::= <while> sd <logical operator> sd

<while> ::= WHILE | WHILEB | WHILEF

<while chain>

::= <while clause>|<while clause> {AND / OR} <while chain>

<repeat-while statement>

::= REPEAT

<a block of instructions>

<while chain> DO <one assembly instruction>

or REPEAT

<a block of instructions>

<while chain> DO

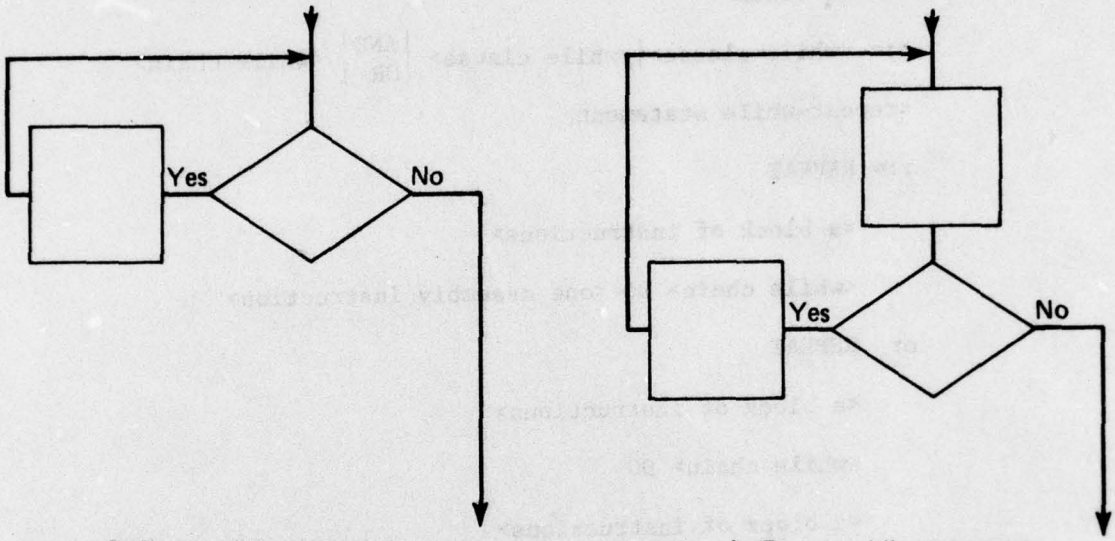
<a block of instructions>

ENDWHL

The terms AND, OR, and DO can start new lines also. For example:

```
REPEAT
WHILE A GT B DO CLR X

REPEAT
    CLR X
    IF A LT X THEN NEG A (negate A)
WHILE A LT D DO
    INC A (increment A by 1)
    CLR Y
ENDWHL
```



a. Ordinary while statement

b. Repeat-while statement

Fig. 1 Flow Diagrams

FOR STATEMENT

<for statement> ::= FOR sd FROM sd { UPTO / DOWNTO } sd { blank / BY sd } DO

<a block of instructions>

ENDFOR

The block of instructions is executed repeatedly with the value of the first sd varying from the second to the third sd, inclusive, with the increment (decrement for DOWNTO) of the fourth sd (or 1 if it is not specified) provided it does not pass over the third sd. New lines may be started only by FOR and ENDFOR. For example:

```
FOR A FROM B UPTO C DO
    CLR X
ENDFOR

FOR A FROM B DOWNTO C BY D DO
    CLR X
ENDFOR
```

4. MACRO DEFINITIONS AND THE PROCESSOR ORGANIZATION

If, repeat-while, and for statements mentioned in the previous sections are constructed through a number of macro instructions. They are as follows: IF, IFB, IFF, THEN, ELSE, ENDIF, AND, OR, REPEAT, WHILE, WHILEB, WHILEF, DO ENDWHL, FOR UPTO, DOWNTO, ENDFOR, and macros for the logical operators. The special processor mentioned earlier can be organized in a similar manner (one subroutine for each macro). In addition, the processor requires the main program and a subroutine that scans a line of code to separate the terms. We will describe the processor and use the macro names for the corresponding subroutine names also.

When a line of code is read, it is scanned and processed to determine if it is the beginning of a new conditional statement (if, repeat-while, or for statement). If it is, a unique sequence number, ISQ, is assigned to that statement and we give its nesting level, ILEV. ISQ is then stored in ISTAK(ILEV), or the ILEVth entry of the array ISTAK. Through this array the sequence number of a statement can be found through its nesting level until the statement is terminated. This sequence number plays an important role in forming the destination of the jump instructions and labels. As an example, when the new line of code starts with ELSE, we can find the sequence number ISQ for the if statement from ISTAK(ILEV) and assign a label QEDXXX, where XXX is the sequence number. The nesting level, ILEV, is incremented when a new statement is entered and is decremented when ENDIF, ENDWHL, or ENDFOR is called.

A new statement is identified by examining the new statement switch, NEWST, immediately after IF, IFB, and IFF subroutines are entered. This switch is set to yes (or 1) when the main program is entered, and after THEN, ELSE, DO, ENDIV, and ENDWHL are called. It is set to no (or 0) in IF, IFB, IFF, WHILE, WHILEB, and WHILEF subroutines. It is always a new statement when REPEAT or FOR is called.

The labels generated by the preprocessor consist of three alphabetic characters followed by numeric characters. The leading three characters are QOR, QEL, QTR, QRP, QID, QED, QFR, and QFD. In the program descriptions, the labels are denoted, for example, as QOR-IOR. (If IOR has the value 123, this is QOR123).

The descriptions of the processor subprograms follow. The subprograms are simple and not for producing "optimal" codes. codes.

1. Main Program

   Set the new statement switch NEWST to 1.

   Set the switch array IFWHSW to 0. This array is used by the subroutine THEN.

   DO subroutine sets IFWHSW(ILEV) to 1 and calls subroutine THEN. Through this switch, subroutine THEN will know if it is called as a part of if statement or repeat-while statement.

   Set the switch MODE to 0.

   MODE is set to 1 in IFB and WHILEB subroutines for the byte mode, and is set to 2 in IFF and WHILEF subroutines for the floating point number mode. It is examined in the LOPR subroutine to provide the correct comparison instruction.

   Call subroutine SCAN to separate the label, to store each term in the array DTERM, and to provide the number of terms in the line in INAR.

   If INAR is less than 1, then write the input line of codes on the output device and go back to read the next line.

   While the first term in consideration is a control term (macro name), increment the term pointer IPTR, decrement INAR, and call the appropriate subroutine to process it, and repeat the process. If the terms are not exhausted, then write the remaining terms.

   Go back to read the next line.

2. Subroutine SCAN

   Separate the label, if any, and store each term of the line in the array DTERM separately.

   Disregard the comment.

   Set the term pointer IPTR to 1.

   Set INAR to the number of terms.

   Return.

3. Subroutine IF

If the new statement switch NEWST is 1, then

ISQ = ISQ + 1 (increment the sequence number by 1)

ILEV = ILEV + 1 (increment the nesting level number)

ISTAK(ILEV) = ISQ

Set NEWST to 0

End.

Set IEL(ILV) to 0 to initialize the else-switch for this if statement. It is later set to 1 when ELSE subroutine is called. This switch is later examined by ENDIF subroutine.

Interchange the first two terms DTERM(IPT) and DTERM(IPT+1). Now DTERM(IPT) has a logical operator.

Return.

4. Subroutine IFB

Set MODE to 1 for the byte mode.

Mode will be examined in LOPR subroutine to provide the correct instruction for word compare, byte compare, or floating point number compare.

Call IF subroutine.

Return.

5. Subroutine IFF

Set MODE to 2 for the floating point number mode.

Call IF subroutine.

Return.

6. Subroutine THEN

Set NEWST to 1.

Write the conditional jump statement with the instruction code from INST1 and the destination to QTR-ISQ.

INST1 and INST2 are set by the logical operator handling subroutine LOPR. The sequence number ISQ is always found through ISTAK(ILEV).

Write the label QOR-ISQ and the unconditional jump instruction with the desitnation QEL-ISQ. The destination is the instruction immediately following the ELSE term in the statement, or ENDIF or ENDWHL if there is no ELSE in the statement.

Set IOR to IOR + 1.

Write the label QTR-ISQ.

If INAR is 0, or if there is no more remaining term in the input line of codes, then return.

Write the remaining terms.

Set INAR to 0.

If IFWHSW(ILEV) is 0, then call ENDIF.

Else call ENDWHL and set IFWHSW(ILEV) to 0.

The switch IFWHSW(ILEV) is set to nonzero if THEN is called from DO subroutine for the repeat-while statement.

Return.

7. Subroutine ELSE

Set the switch IEL(ILEV) to 1 to signify that there is an ELSE term in the if statement.

Set NEWST to 1.

Write the unconditional jump instruction with the destination QID-ISQ, or the instruction following ENDIF. This jump instruction is the last line of code in the instruction block following THEN.

Write the label QEL-ISQ.

If INAR is 0, then return.

Write the remaining term.

Call ENDIF subroutine.

Set INAR to 0.

Return.

8. Subroutine ENDIF

If IEL(ILEV) is 0, then write QEL-ISQ label. This means that ELSE was not called for the if statement, and we need to add the QEL-label.

Write QID-ISQ label.

Set INAR to 0.

Set ILEV to ILEV - 1.

The nesting level is decremented since an if statement is finished and the level is lowered.

Return.

9. Subroutine AND

Write the conditional jump instruction where the instruction code is taken from INST2 and the destination is QOR-IOR.

Return.

10. Subroutine OR

Write the conditional jump instruction where the instruction code is taken from INST1 and the destination is QTR-IOR.

Write the label QOR-ISQ.

Set IOR to IOR + 1.

Return.

11. Subroutine REPEAT

Set ISQ to ISQ + 1.

Set ILEV to ILEV + 1.

Set ISTAK(ILEV) to ISQ.

Write the label QRP-ISQ.

Return.

12. Subroutine WHILE

Set NEWST to 0.

Call IF subroutine.

Return.

13. Subroutine WHILEB

Set MODE to 1.

Call WHILE.

Return.

14. Subroutine WHILEF

Set MODE to 2.

Call WHILE.

Return.

15. Subroutine DO

Set IFWHSQ(ILEV) to 1.

Call THEN.

Return.

16. Subroutine ENDWHL

Write the unconditional jump instruction to QRP-ISQ, which is the beginning of the repeat-while statement.

Write the label QEL-ISQ.

Set ILEV to ILEV - 1.

Set NEWST to 1.

Return.

17. Subroutine FOR

(When this subroutine is entered, the terms in the array DTERM are sd1 FROM sd2 UPTO (or DOWNTO) sd3 DO.)

Set ISQ to ISQ + 1.

Set ILEV to ILEV + 1.

Set ISTAK(ILEV) to ISQ.

Write the instruction "move sd2 to sd1".

Move the fourth term, UPTO (or DOWNTO), to the third, and move the first term, sd1, to the fourth in DTERM array.

Set INAR to IWAR - 2.

Set IPTR to IPTR + 2.

Return.

Notice that DTERM(IPTR) is now UPTO (or DOWNTO).

18. Subroutine UPTO (or DOWNTO)

If INAR = 3, then set N to 1.

If INAR = 5, then set N to DTERM(IPTR + 3).

Set INAR to 0.

Write the instruction "decrement (or increment) the first argument by N".

Write the label QRF-ISQ.

Write the instruction "If the first argument is not greater (or less) than the second, then go to QFR-ISQ, else go to QFD-ISQ".

Set NEWST to 1.

Return.

19. Subroutine ENDFOR

Write the unconditional jump instruction to QFR-ISQ.

Write the label QFD-ISQ.

Set ILEV to ILEV - 1.

Set NEWST to 1.

Return.

20. Subroutine GT (or GE, EQ, NE, LE, or LT)

Set IINO to 1 (or 2, 3, 4, 5, or 6).

Call LOPR(IINO).

Return.

21. Subroutine LOPR(IINO)

Write the instruction "Compare the first two terms DTERM(IPTR) and DTERM(IPTR + 1)".

The comparison instruction can be either compare words, bytes, or floating point numbers according to the switch MODE = 0, 1, or 2, respectively.

Set MODE to 0.

Store the instruction codes in INST1 and INST2 according to the table below:

Condition	IIND	INST1	INST2
GT	1	go to if >	go to if $\leq$
GE	2	go to if $\geq$	go to if <
EQ	3	go to if =	go to if $\neq$
NE	4	go to if $\neq$	go to if =
LE	5	go to if $\leq$	go to if >
LT	6	go to if <	go to if $\geq$

5. CONCLUDING REMARKS

The macro instructions described here can substitute for machine-oriented high-level language and can help one improve the assembly language program productivity through (a) the reduced number of lines of code, (b) improved program readability, and (c) structured programming. One can implement additional macro instructions and use them to adapt the language to the given task and enjoy the above advantages even further.

If, in addition, the software development is initiated with its design using a suitable design language, its successive refinement could result in the final program code.

Development of the processor for the above-mentioned macro instructions is not difficult. Macro language processors for small computers are frequently inadequate. For these reasons the development of the special independent macro processor may be advantageous.

REFERENCES

1. C. W. Barth, "STRCMACS — An Extensive Set of Macros to Aid in Structured Programming in 360/370 Assembly Language," SIGPLAN Notices, Vol. 22, No. 8, 1976, pp. 31-35.

2. G. S. Herman-Giddens, R. B. Warren, R. C. Barr, and M. S. Spach, "BIOMAC — Block Structured Programming Using PDP-11 Assembler Language," SOFTWARE: Practice and Experiences, Vol. 5.

3. S. W. Kahng, "Structured Programming and Debugging Aids with MACRO-11 Assembly Language for the PDP-11 Computer," APL/JHU TG 1294, April 1976.

4. C. Pepper, "SMAL — A Structured Macro-Assembly Language for a Microprocessor," IEEE COMPCON Fall Digest of Papers, September 1974, pp. 147-151.

5. G. E. Riens, "Structured Programming in Assembly Language," Datamation, July 1976, pp. 79-84.

6. R. Conradi, P. Holager, O. Solberg, and G. Green, "A System for Software Development on Minicomputers," Minicomputer Software, North-Holland Publishing Co., 1976, pp. 15-29.

7. H. Lehessaari, "A Family of Machine Oriented Higher Level Languages (MOHLL)," Minicomputer Software, North-Holland Publishing Co., 1976, pp. 231-237.

8. R. D. Russell, "Intermediate-Level Programming Languages for On-Line Data Acquisition and Control," Computer Physics Communications, Vol. 5, 1973, pp. 89-97.

APPENDIX

The macro processing program for MACRO-11, the PDP-11 Computer Assembly Language, is listed here. The program performance is different from the description in Section 3 as follows:

1. IFB, IFF, WHILEB, and WHILEF are not implemented.

2. Two logical operators on bits, SETIN and VOIDIN, are added.

   SD1 SETIN SD2

   is true if the bits set in SD1 are also set in SD2; otherwise, it is false.

   SD1 VOIDIN SD2

   is true if each bit set in SD1 is not set in SD2; otherwise it is false.

```
C MACRO PREPROCESSOR FOR POP-11
C ***********
C
C
C    LOGICAL UNIT 5 FOR THE INPUT MACRO CODES
C    LOGICAL UNIT 8 FOR THE OUTPUT OF THE EXPANDED CODES
C
C     TO ADD A NEW FUNCTION
C
C       1. ADD TWO INSTRUCTIONS NEAR THE END OF MAIN PROGRAM AS FOLLOWS
C  NN CALL SUBR
C          WHERE NN IS THE LABEL NUMBER,AND SUBR IS THE SUBROUTINE NAME
C          THAT PERFORMS THE FUNCTION,E.G. IF,ENDFOR.
C
C       2. ADD A DATA CARD IN THE BLOCK DATA AS FOLLOWS
C     DATA DA(NN) /8HXXXXXXXX/
C          WHERE NN IS THE SAME AS IN 1. ABOVE,AND
C                XXXXXXXX IS THE FUNCTION NAME USED IN THE PROGRAM. IT
C                         IS A STRING OF ATMOST 8 CHARACTERS INCLUDING
C                         TRAILING BLANKS
C                ALSO INCREMENT THE DATA VALUE OF MACNO
C
C       3. ADD SUBR SUBROUTINE. THIS PERFORMS THE REQUIRED FUNCTION.
C
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISO,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
      DIMENSION IEOF(2)
      LOGICAL*1  IEOF
      DATA IEOF /1H/,1H*/
C
C
      NEWLIN=1
C
C READ A CARD
C
200    READ(5,100) (ICDIM(I),I=1,80)
100    FORMAT(80A1)
      IF(ICDIM(1) .EQ. IEOF(1) .AND. ICDIM(2) .EQ. IEOF(2)) GO TO 1000
C
C SCAN ICDIM AND SET POINTERS ISTRT & ISTOP
C ALSO TAKE CARE OF LABEL AND COMMENT. INAR= # OF ARG
C
      CALL SCAN
      IF(INAR .LE. 0)  GO TO 600
C
      IPTR=1
      INO=1
C
C GO TO 600 IF NO MORE TO PROCESS
C
300   IF(INAR .LE. 0)  GO TO 200
      IF(INO .LT. 1) GO TO 600
C
C
C SEE IF IT IS IN THE MACRO TABLE. INO GIVES THE SEQ. NO. OF THE TABLE
C
```

```
C
C     SEE IF THE TERM IS IN THE MACRO TEBLE
C
      DO 400 I=1,MACNO
      IF(DA(I) .NE. DTERM(IPTR)) GO TO 400
      INO=I
      IPTR=IPTR+1
      INAR=INAR-1
      GO TO 410
400   CONTINUE
      INO=-1
      GO TO 600
410   CONTINUE
C
C
C
      IF(LABLSW .EQ. 0)  GO TO 310
      WRITE(8,50) DLABL
50    FORMAT(1X,A8)
      LABLSW=0
310   CONTINUE
C
      IF(INO .LE.0)  GO TO 600

C    IF INO > 0  A MACRO IS FOUND. GO PROCESS IT
C
C   IPTR POINTS TO THE FIRST ARGUMENT OF THE MACRO
C   INAR GIVES THE NUMBER OF ARGUMENTS OF THE MACRO
C   INO  GIVES THE SEQ. NO. OF THE MACRO IN THE TABLE
C
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
     1 23,24,25,26),INO
C
C
1     CALL IF
      GO TO 300
2     CALL THEN
      GO TO 300
3     CALL ELSE
      GO TO 300
4     CALL ENDIF
      GO TO 300
5     CALL AND
      GO TO 300
6     CALL OR
      GO TO 300
7     CALL REPEAT
      GO TO 300
8     CALL WHILE
      GO TO 300
9     CALL DO
      GO TO 300
10    CALL ENDWHL
      GO TO 300
11    CALL FOR
      GO TO 300
12    CALL ENDFOR
      GO TO 300
13    CALL UPTO
      GO TO 300
```

```
14     CALL DOWNTO
       GO TO 300
   15  CALL GT
       GO TO 300
   16  CALL GE
       GO TO 300
   17  CALL EQ
       GO TO 300
   18  CALL NE
       GO TO 300
   19  CALL LE
       GO TO 300
   20  CALL LT
       GO TO 300
   21  CALL HI
       GO TO 300
   22  CALL HIS
       GO TO 300
   23  CALL LOS
       GO TO 300
   24  CALL LO
       GO TO 300
   25  CALL SETIN
       GO TO 300
   26  CALL DISJNT
       GO TO 300
C
C
600    IF(IPTR .EQ. 1) WRITE(8,110) (ICDIM(I),I=1,72)
110    FORMAT(9X,80A1)
       JJ=ISTRT(IPTR)
       IF(IPTR .GT. 1) WRITE(8,110) (ICDIM(I),I=JJ,72)
       GO TO 200
1000   STOP
       END
C ***,*************************
       SUBROUTINE SCAN
C SCANNS ICDIM(1-72),AND STORE EACH TERMS IN DTERM(1-50).
C SETS INAR TO # OF TERMS.
C
       COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
      1 ICDIM(80),ISTK(100),INAR,INC,
      2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
       DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
       LOGICAL*1 ICDIM
       LOGICAL*1 COMNT,LABEL,BLANK,COMMA
       DOUBLE PRECISION DITEM,DBLNK
       LOGICAL*1 ITEM
       DIMENSION ITEM(8)
       EQUIVALENCE (DITEM,ITEM(1))
       DATA DBLNK /8H        /
       DATA COMNT /1H;/
       DATA LABEL /1H&/
       DATA BLANK /1H /
       DATA COMMA /1H,/
C IBC=0 IF PREVIOUS CHARACTER WAS A SEPARATER; BLANK OR COMMA
C IBC=1 IF PREVIOUS CHARACTER WAS NOT A SEPARATER
C
       IBC=0
       JPTR=0
```

```
        INAR=0
      LABLSW=0
      DO 1000 I=1,30
      DITEM=DBLNK
99    JJ=1
100   CONTINUE
      JPTR=JPTR+1
      IF(JPTR .GE. 73) GO TO 110
      IF(ICDIM(JPTR) .NE. COMNT) GO TO 200
C COMMENT
110   IF(IBC .NE. 0) GO TO 150
      INAR=I-1
      RETURN
 150   CONTINUE
      INAR=I
      RETURN
C
C A LABEL<
200   IF(ICDIM(JPTR) .NE. LABEL) GO TO 250
C YES A LABEL
      JK=MIN0(JJ,7)
      ITEM(JK)=LABEL
      IF(ICDIM(JPTR+1) .NE. LABEL)  GO TO 160
        ITEM(JK+1)=LABEL
       JPTR=JPTR+1
  160 CONTINUE
      IBC=0
      DLABL=DITEM
      LABLSW=1
      IF(I .EQ. 1) GO TO 210
      WRITE(8,90)
90    FORMAT('  ***** LABEL IS NOT AT THE FIRST TERM')
      RETURN
210   DITEM=DBLNK
      GO TO 99
C NOT A LABEL
C A SEPARATER<
250   IF(ICDIM(JPTR) .NE. BLANK .AND. ICDIM(JPTR) .NE. COMMA) GO TO 300
C YES A SEPARATER
C PREVIOUS CHARACTER A SEPARATER<
      IF(IBC .EQ. 0) GO TO 100
C NO. END OF A TERM
      DTERM(I)=DITEM
      IBC=0
      GO TO 1000
C
C NOT A SEPARATER
300   IF(IBC .EQ. 0) ISTRT(I)=JPTR
      IF(JJ .LE. 8)  ITEM(JJ)=ICDIM(JPTR)
      JJ=JJ+1
      IBC=1
      GO TO 100
1000  CONTINUE
      WRITE(8,10)
   10  FORMAT('ERROR. THERE ARE MORE THAN 30 TERMS')
      RETURN
      END
C ****************************
      BLOCK DATA
C  NOT INCLUDED ARE BPL,BMI,BCC,BVC,BCS,BVS,...
```

```
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
      DATA MACNO /26/

      DATA  DA( 1) /8HIF      /
      DATA  DA( 2) /8HTHEN    /
      DATA  DA( 3) /8HELSE    /
      DATA  DA( 4) /8HENDIF   /
      DATA  DA( 5) /8HAND     /
      DATA  DA( 6) /8HOR      /
      DATA  DA( 7) /8HREPEAT  /
      DATA  DA( 8) /8HWHILE   /
      DATA  DA( 9) /6HDO      /
      DATA  DA(10) /8HENDWHL  /
      DATA  DA(11) /8HFOR     /
      DATA  DA(12) /8HENDFOR  /
      DATA  DA(13) /8HUPTO    /
      DATA  DA(14) /8HDOWNTO  /
      DATA  DA(15) /8HGT      /
      DATA  DA(16) /8HGE      /
      DATA  DA(17) /8HEQ      /
      DATA  DA(18) /8HNE      /
      DATA  DA(19) /8HLE      /
      DATA  DA(20) /8HLT      /
      DATA  DA(21) /8HHI      /
      DATA  DA(22) /8HHIS     /
      DATA  DA(23) /8HLOS     /
      DATA  DA(24) /8HLO      /
      DATA  DA(25) /8HSETIN   /
      DATA  DA(26) /8HDISJOINT/
C
      DATA DCND(1) /8HBGT     /
      DATA DCND(2) /8HBGE     /
      DATA DCND(3) /8HBEQ     /
      DATA DCND(4) /8HBNE     /
      DATA DCND(5) /8HBLE     /
      DATA DCND(6) /8HBLT     /
      DATA DCND(7) /8HBHI     /
      DATA DCND(8) /8HBHIS    /
      DATA DCND(9) /8HBLOS    /
      DATA DCND(10)/8HBLO     /
C
      DATA DCNDC(1) /8HBLE     /
      DATA DCNDC(2) /8HBLT     /
      DATA DCNDC(3) /8HBNE     /
      DATA DCNDC(4) /8HBEQ     /
      DATA DCNDC(5) /8HBGT     /
      DATA DCNDC(6) /8HBGE     /
      DATA DCNDC(7) /8HBLOS    /
      DATA DCNDC(8) /8HBLO     /
      DATA DCNDC(9) /8HBHI     /
      DATA DCNDC(10) /8HBHIS    /
C
      DATA IOR /100/
      DATA ISQ /100/
      DATA ILV /1/
```

```
      END
C *****************************
      SUBROUTINE GTEQLT(IINO)
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST ,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
      WRITE(8,10) DTERM(IPTR),DTERM(IPTR+1)
   10 FORMAT(1X,8X,'CMP     ',A8,',',A8)
      IPTR=IPTR+2
      INAR=INAR-2
      DINST1=DCND(IINO)
      DINST2=DCNDC(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE IF
C    IF  A B C D E
C      CMP   A,C
C      D B E F
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST
      LOGICAL*1 ICDIM
C
      COMMON /CEL/ IEL(50)
      COMMON /COM2/ IFWHSW(50)
C
      IF(NEWLIN .EQ. 0) GO TO 100
      ISQ=ISQ+1
      ILV=ILV+1
      ISTK(ILV)=ISQ
      IFWHSW(ILV)=0
      NEWLIN=0
100   CONTINUE
      IEL(ILV)=0
C INTERCHANGE DTERM(IPTR) AND DTERM(IPTR+1)
      DTERM(IPTR-1)=DTERM(IPTR+1)
      DTERM(IPTR+1)=DTERM(IPTR)
      DTERM(IPTR  )=DTERM(IPTR-1)
      RETURN
      END
C *****************************
      SUBROUTINE THEN
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST
      LOGICAL*1 ICDIM
      COMMON /COM2/ IFWHSW(50)
C
        NEWLIN=1
C       B-GL     .+6
      WRITE(8,10) DINST1
10    FORMAT(9X,A8,'.+6')
C GOR&  JMP      QEL-ISQ
      WRITE(8,20) IOR,ISTK(ILV)
20    FORMAT(1X,'GOR',I3,'&',2X,'JMP',5X,'QEL',I3)
```

```
C QTR-ISQ&
      WRITE(8,30) ISTK(ILV)
30    FORMAT(1X,'QTR',I3,'&')
      IOR=IOR+1
      IF(INAR .EQ. 0) RETURN
C REMAINIG TERMS
      JJ=ISTRT(IPTR)
      WRITE(8,40) (ICDIM(J),J=JJ,72)
      IF(IFWHSW(ILV) .EQ. 0)  CALL ENDIF
      IF(IFWHSW(ILV) .EQ. 1)  CALL ENDWHL
      IFWHSW(ILV)=0
      INAR=0
40    FORMAT(9X,72A1)
      RETURN
      END
C ****************************
      SUBROUTINE ELSE
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST
      LOGICAL*1 ICDIM
C
      COMMON /CEL/ IEL(50)
C
        NEWLIN=1
      IEL(ILV)=1
C       JMP     QID-ISQ
      WRITE(8,10) ISTK(ILV)
10    FORMAT(9X,'JMP     QID',I3)
C QEL-ISQ&
      WRITE(8,20) ISTK(ILV)
20    FORMAT(1X,'QEL',I3,'&')
      IF(INAR .EQ. 0) RETURN
C REMAINING TERMS
      JJ=ISTRT(IPTR)
      WRITE(8,30) (ICDIM(J),J=JJ,72)
30    FORMAT(9X,72A1)
      CALL  ENDIF
      RETURN
      END
C ****************************
      SUBROUTINE ENDIF
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
      COMMON /CEL/ IEL(50)
C
      IF(IEL(ILV) .EQ.0) WRITE(8,20)  ISTK(ILV)
C QEL-ISQ&
20    FORMAT(1X,'QEL',I3,'&')

C QID-ISQ&
      WRITE(8,10) ISTK(ILV)
10    FORMAT(1X,'QID',I3,'&')
      ILV=ILV-1
      INAR=0
```

```
        NEWLIN=1
      RETURN
      END
C *****************************
      SUBROUTINE AND
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
C     BAGL   QOR'IOR
      WRITE(8,10) DINST2,IOR
   10 FORMAT(1X,8X,A8,'QOR',I3)
      RETURN
      END
C *****************************
      SUBROUTINE OR
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
C      BGL      QTR-ISQ
      WRITE(8,10) DINST1,ISTK(ILV)
10    FORMAT(9X,A8,'QTR',I3)
C QOR-IOR&
      WRITE(8,20) IOR
20    FORMAT(1X,'QOR',I3,'&')
      IOR=IOR+1
      RETURN
      END
C *****************************
      SUBROUTINE REPEAT
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
      ILV=ILV+1
      ISQ=ISQ+1
      ISTK(ILV)=ISQ
      NEWLIN=0
      WRITE(8,10)  ISQ
10    FORMAT(1X,'QRP',I3,'&')
      RETURN
      END
C *****************************
      SUBROUTINE WHILE
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST
      LOGICAL*1 ICDIM
C
      NEWLIN=0
      CALL IF
      RETURN
```

```
      END
C *****************************
      SUBROUTINE  DO
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
      COMMON /COM2/ IFWHSW(50)
      IFWHSW(ILV)=1
      CALL THEN
      IFWHSW(ILV)=0
      RETURN
      END
C *****************************
      SUBROUTINE ENDWHL
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
C       JMP     QRP-ISQ
      WRITE(8,10) ISTK(ILV)
10    FORMAT(9X,'JMP',5X,'QRP',I3)
C QEL-ISQ&
      WRITE(8,20) ISTK(ILV)
20    FORMAT(1X,'QEL',I3,'&')
      ILV=ILV-1
        NEWLIN=1
      RETURN
      END
C *****************************
      SUBROUTINE FOR
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
C       FOR I FROM BEG UPDOWN END BY INC DO
      NEWLIN=0
      ILV=ILV+1
      ISQ=ISQ+1
      ISTK(ILV)=ISQ
C       MOV     BEG,I
      WRITE(8,10) DTERM(IPTR+2),DTERM(IPTR)
10    FORMAT(9X,'MOV',5X,A8,',',A8)
C       UPDOWN I END BY INC D0
C       UPDOWN I END DO
      DTERM(IPTR+2)=DTERM(IPTR+3)
      DTERM(IPTR+3)=DTERM(IPTR)
      IPTR=IPTR+2
      INAR=INAR-2
      RETURN
      END
C *****************************
      SUBROUTINE UPTO
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
```

```
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
      IF(INAR .EQ. 3) GO TO 180
      IF(INAR .EQ. 5) GO TO 200
      WRITE(8,10)
10    FORMAT('  ** FOR-STMT ERROR ***')
      INAR=0
      RETURN
180   CONTINUE
C       DEC     I
      WRITE(8,20) DTERM(IPTR)
20    FORMAT(9X,'DEC',5X,A8)
C QFR-ISQ&
      WRITE(8,30) ISTK(ILV)
30    FORMAT(1X,'QFR',I3,'&')
C       INC     I
      WRITE(8,40) DTERM(IPTR)
40    FORMAT(9X,'INC',5X,A8)
      GO TO 300
200   CONTINUE
C       SUB     INCRM,I
      WRITE(8,50) DTERM(IPTR+3),DTERM(IPTR)
50    FORMAT(9X,'SUB',5X,A8,',',A8)
C QFR-ISQ&
      WRITE(8,30) ISTK(ILV)
C       ADD     INCRM,I
      WRITE(8,60) DTERM(IPTR+3),DTERM(IPTR)
60    FORMAT(9X,'ADD',5X,A8,',',A8)
300   CONTINUE
C       CMP     I,END
      WRITE(8,70) DTERM(IPTR),DTERM(IPTR+1)
70    FORMAT(9X,'CMP',5X,A8,',',A8)
C       BLOS    .+6
      WRITE(8,80)
80    FORMAT(9X,'BLOS',4X,'.+6')
C       JMP     QFD-ISQ
      WRITE(8,90) ISTK(ILV)
90    FORMAT(9X,'JMP',5X,'QFD',I3)
      INAR=0
        NEWLIN=1
      RETURN
      END
C ****************************
      SUBROUTINE DOWNTO
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
C   DOWNTO I END DO
C   DOWNTO I END BY DEC DO
      IF(INAR .EQ. 3) GO TO 180
      IF(INAR .EQ. 5) GO TO 200
      WRITE(8,10)
10    FORMAT('  ** FOR-STMT ERROR ***')
      INAR=0
      RETURN
```

```
180   CONTINUE
C       INC     I
      WRITE(8,20) DTERM(IPTR)
20    FORMAT(9X,'INC',5X,A8)
C QFR-ISQ
      WRITE(8,30) ISTK(ILV)
30    FORMAT(1X,'QFR',I3,'&')
C       DEC     I
      WRITE(8,40) DTERM(IPTR)
40    FORMAT(9X,'DEC',5X,A8)
      GO TO 300
200   CONTINUE
C       ADD     INCRM,I
      WRITE(8,50) DTERM(IPTR+3),DTERM(IPTR)
50    FORMAT(9X,'ADD',5X,A8,',',A8)
C QFR-ISQ&
      WRITE(8,30) ISTK(ILV)
C       SUB     INCRM,I
      WRITE(8,60) DTERM(IPTR+3),DTERM(IPTR)
60    FORMAT(9X, 'SUB',5X,A8,',',A8)
300   CONTINUE
C       CMP     I,END
      WRITE(8,70) DTERM(IPTR),DTERM(IPTR+1)
70    FORMAT(9X,'CMP',5X,A8,',',A8)
C       BHIS    .+6
      WRITE(8,80)
80    FORMAT(9X,'BHIS',4X,'.+6')
C       JMP     QFD-ISQ
      WRITE(8,90) ISTK(ILV)
90    FORMAT(9X,'JMP',5X,'QFD',I3)
      INAR=0
        NEWLIN=1
      RETURN
      END
C *****************************
      SUBROUTINE ENDFOR
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
C
C       JMP     QFR-ISQ
      WRITE(8,10) ISTK(ILV)
10    FORMAT(9X,'JMP',5X,'QFR',I3)
C QFD-ISQ&
      WRITE(8,20) ISTK(ILV)
20    FORMAT(1X,'QFD',I3,'&')
      ILV=ILV-1
        NEWLIN=1
      RETURN
      END
C *****************************
      SUBROUTINE GT
      IINO=1
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE GE
```

```
      IINO=2
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE EQ
      IINO=3
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE NE
      IINO=4
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE LE
      IINO=5
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE LT
      IINO=6
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE HI
      IINO=7
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE HIS
      IINO=8
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE LOS
      IINO=9
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE LO
      IINO=10
      CALL GTEQLT(IINO)
      RETURN
      END
C *****************************
      SUBROUTINE SETIN
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
      DATA DST1 /8HBEQ     /
```

```
      DATA DST2 /8HBNE     /
        WRITE(8,10)  DTERM(IPTR+1)
        WRITE(8,11) DTERM(IPTR)
      WRITE(8,12) DTERM(IPTR+1)
   10 FORMAT(1X,8X,'MOV     ',A8,',-(SP)')
   11 FORMAT(1X,8X,'BIS     ',A8,',(SP)')
   12 FORMAT(1X,8X,'CMP     ',A8,',(SP)+')
        DINST1=DST1
      DINST2=DST2
      IPTR=IPTR+2
      INAR=INAR-2
      RETURN
      END
C *****************************
      SUBROUTINE DISJNT
      COMMON  DLABL,DTERM(50),DA(30),DCND(20),DCNDC(20),DINST1,DINST2,
     1 ICDIM(80),ISTK(100),INAR,INO,
     2 LABLSW,MACNO,NEWLIN,ISQ,ILV,IOR,ISTRT(50),IPTR
      DOUBLE PRECISION DLABL,DTERM,DA,DCND,DCNDC,DINST1,DINST2
      LOGICAL*1 ICDIM
      DATA DST1 /8HBEQ     /
      DATA DST2 /8HBNE     /
        WRITE(8,10)  DTERM(IPTR+1)
        WRITE(8,11) DTERM(IPTR)
      WRITE(8,12) DTERM(IPTR+1)
   10 FORMAT(1X,8X,'MOV     ',A8,',-(SP)')
   11 FORMAT(1X,8X,'BIC     ',A8,',(SP)' )
   12 FORMAT(1X,8X,'CMP     ',A8,',(SP)+')
        DINST1=DST1
      DINST2=DST2
      IPTR=IPTR+2
      INAR=INAR-2
      RETURN
      END
```

INITIAL DISTRIBUTION EXTERNAL TO THE APPLIED PHYSICS LABORATORY*

The work reported in TG 1308 was done under Navy Contract N00017-72-C-4401. This work is related to Task B411, which is supported by NAVSEA.

ORGANIZATION	LOCATION	ATTENTION	No. of Copies
DEPARTMENT OF DEFENSE			
DDC	Alexandria, VA		12
Department of the Navy			
NAVSEA	Washington, DC	SEA-09G3	2
NAVAIR	Washington, DC	AIR-50174	2
NAVASTROGRU	Pt. Mugu, CA	Library	1
NAVPRO	Silver Spring, MD		1
Department of the Air Force			
NORAD	Ent AFB, CO 80912	Systems Branch	1
7102 CSS Hq. USAFE, ACDYV	APO New York, NY 09012	CO	1
		D. Beilfuss	1
CONTRACTORS			
General Electric Co.	Louisville, KY 40225	Physical Sciences Dept.	1
		L. T. Nieh	1
Electronics Corp. of India, Ltd.	Hyderabad-500 040, A.P., India	Computer Group	1
		V. K. Premchand	1
UNIVERSITIES			
U. of California	Santa Barbara, CA	Computing Ctr.	1
U. of Waterloo	Waterloo, Ontario, Canada	Computing Ctr.	1
U. of New England	Armidale, NSW, 2351, Australia	W. Adamson, Dept. of Computing Science	1
U. of Texas	Galveston, TX 77550	Medical Branch	1
		P. D. Sawyer	1
U. of Regina	Regina, Sask., Canada	Dr. R. B. Maguire, Dept. of Computer Science	1
U. of Washington	Seattle, WA 98195	M. Frimer, Cardiovascular Res. and Tng. Ctr.	1
Indiana U.	Bloomington, IN 47401	J. C. Forshee, Dept. of Psychology	1
OTHER ORGANIZATIONS			
Natural Gas Pipeline Co. of America	Chicago, IL 60603	A. P. Liaugaudas	1
Aluminum Company of Canada, Ltd.	Arvida, Que., Canada	J. Menard, Process Control Dept.	1
E. R. Squibb & Sons, Inc.	Princeton, NJ 08540	P. J. Black	1
Lockheed, Palo Alto Research Laboratory	Palo Alto, CA 94304	G. Hamma, 52-33, Bldg. 205	1
Alberta Research	Edmonton, Alta., Canada	M. R. Johnson	1
Sandia Laboratories	Livermore, CA 94550	R. Y. Lee, Div. 8322	1
CSIRO Div. of Textile Physics, Wool Research Laboratories	Ryde, NSW, Australia	R. N. Caffin	1

Requests for copies of this report from DoD activities and contractors should be directed to DDC, Cameron Station, Alexandria, Virginia 22314 using DDC Form 1 and, if necessary, DDC Form 55.

*Initial distribution of this document within the Applied Physics Laboratory has been made in accordance with a list on file in the APL Technical Publications Group.