AAEC/E248

AAEC/E248

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

MACROS TO SIMULATE FORTRAN I/O IN ASSEMBLER PROGRAMS

by

SUSAN JOHNSON

December 1972
ISBN 0 642 99514 1

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

MACROS TO SIMULATE FORTRAN I/O IN ASSEMBLER PROGRAMS

by

SUSAN JOHNSON

ABSTRACT

The AREAD, AWRITE, AFORMAT, ABACKSP, AREWIND and AENDFILE instructions are described. They are analogous to the FORTRAN READ, WRITE, FORMAT, BACKSPACE, REWIND and END FILE statements respectively and are intended for use by FORTRAN programmers when writing in the IBM System 360 assembler language.

National Library of Australia card number and ISBN 0 642 99514 1

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA–INIS–12 (INIS: Manual for Indexing) and IAEA–INIS–13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

A CODES; FORTRAN; IBM COMPUTERS; SIMULATION

CONTENTS

1. INTRODUCTION

While the FORTRAN language is a versatile and powerful language for scientific programming it is sometimes desirable to code all or part of a program in assembler language. Of the various symbolic languagesthe IBM System 360 assembler language is the closest to machine language in form and content and so a working knowledge of the FORTRAN language is of very little benefit to an assembler novice. The macro language is an extension of the assembler language and is used to define macro instructions. During the assembly step of a program, for each occurrence of a macro instruction statement, the assembler will generate a sequence of assembler language statements determined by the particular macro definition. Thus the use of macro statements will reduce the programming load by the automatic generation of frequently used sections of code.

The macro instruction statements defined below were designed to follow the syntax of the corresponding FORTRAN statements as closely as possible. This similarity to FORTRAN provides a FORTRAN programmer with familiar I/O facilities in the assembler language.

2. GUIDE TO WRITING MACRO INSTRUCTION STATEMENTS

Each macro instruction statement has the same basic format:

NAME	OPERATION	OPERAND
a symbol or blank	mnemonic operation code	0-200 operands separated by commas

The contents of each field and the significance of the operands for each macro definition are described below.

(a) AREAD

label	AREAD	(unit,format,end,err),list

where:

label - is optional and is a symbol used to identify the statement. It must consist of eight or fewer characters, begin in the first column, and contain no blanks.

unit - is an unsigned integer constant or an F type variable name representing the data set reference number.

format - is optional and is the 'statement number' of the AFORMAT statement describing the format of the record(s) being read. It

must consist of five or less digits and/or letters.

end - is optional and is the label of the statement to which transfer is made upon encountering the end of the data set. It corresponds to the FORTRAN END=... option,

and err - is optional and is the label of the statement to which transfer is made upon encountering an error condition in reading the data. It corresponds to the FORTRAN ERR=... option.

Unlike the FORTRAN convention the 'end' subparameter, if present, must be the third item and the 'err' subparameter, if present, must be the fourth item inside the parentheses. Thus when either the 'format' or 'end' subparameter is omitted and some following subparameter is present, a comma must be coded to replace the absent subparameter. For example the following would preserve the positions of the subparameters: (unit,,end), (unit,,,err).

list - is optional and is an I/O list. The components of the list are separated by commas and are of two types (a) symbols, and (b) array symbols. An array symbol has the form (name,n) where name is a symbol defining the start of the array and n is an unsigned integer equal to the number of array elements. The array symbol is used to indicate n consecutive locations starting at name. As in FORTRAN, the elements of an array have the same length and type.

(b) AWRITE

label	AWRITE	(unit,format),list

where label, unit, format and list are the same as for (a)

(c) AFORMAT

format	AFORMAT	'(C_1,C_2,...,C_n)'

where:

format - is a 'statement number' consisting of five or less digits or letters in any combination, starting in the first column and containing no blanks,

and

C_1,C_2,...,C_n - are FORTRAN format codes (IBM, 1971).

Each single apostrophe or ampersand appearing within a format code must be

represented as a pair. For example the FORTRAN statement

```
19 FORMAT (' AMPERSAND=&')
```

would be translated as

```
19 AFORMAT '('' AMPERSAND=&&'')'.
```

Alternatively the format may be coded as

```
FMT19 DC C'('' AMPERSAND=&&'')',
```

this being the statement generated by the macro expansion (refer to section 4).

Note that because the AFORMAT instruction statement generates a DC statement and the return from IHCECOMH (see section 4) is normally via register 14 to the statement following the AREAD or AWRITE instruction statement, an AFORMAT instruction statement must not immediately follow an AREAD or AWRITE instruction statement.

(d) ABACKSP, AREWIND, AENDFILE

label	ABACKSP	unit
label	AREWIND	unit
label	AENDFILE	unit

where label and unit are as previously defined.

3. EXAMPLES OF USE

Table 1 gives a sample routine coded in FORTRAN and the same routine coded in assembler language with all I/O operations performed by macros.

4. MACRO EXPANSIONS

The object codes for the statements generated by the AREAD, AWRITE, ABACKSP, AREWIND and AENDFILE macro instruction statements are the same as that produced by the FORTRAN compilers (G and H) for the READ, WRITE, BACKSPACE, REWIND and END FILE statements respectively. Each sequence of code is a non-standard call to IHCECOMH, the FORTRAN object-time input/output processor, followed by a list of parameters. If the Extended Error Message facility has not been specified IHCECOMH is replaced by IHCFCOMH. Entry into IHCECOMH is via the IBCOM# transfer table which consists of unconditional branches to various routines within IHCECOMH. This table has the format given in Table 2.

TABLE 1 EXAMPLE OF USE OF MACROS

FORTRAN

```
      .
      .
      .
      INTEGER*2  A
      DIMENSION POINTS (250)
      IN = 1
1     READ (IN,19,END=10,ERR=15)     A,(POINTS(I),I=1,100)
        .
        .
        .
      WRITE (3,29)   A,TIME,EFFCY
      GO TO 1
10    WRITE (3,49)
        .
        .
        .
      CALL EXIT
15    WRITE (3,39) A
        .
        .
        .
      CALL EXIT
        .
        .
        .
        .
        .
19    FORMAT (5X,I3/5(E10.3,5X))
29    FORMAT (1H1,6HBATCH ,I3,8H RESULTS/1X,16HMEAN DELAY TIME ,
     $E14.7/1X,10HEFFICIENCY,6X,E14.7)
39    FORMAT   (1H1,28HINPUT ERROR IN BATCH NUMBER ,I3/1X,
     $28H****************************)
49    FORMAT    (1H1,20HINPUT DATA EXHAUSTED)
        .
        .
        .
      END
```

ASSEMBLER (I/O performed by macros)

```
          .
          .
          .
          .
          .
          .
          .
LOOP      AREAD (IN,19,ENDIO, IOERR),A,(POINTS,100)
          .
          .
          .
          AWRITE (3,OUT),A,TIME,EFFCY
          B    LOOP
ENDIO     AWRITE (3,END)
          .
          .
          .
          B   EXIT
IOERR     AWRITE (3,ERR),A
          .
          .
          .
          B   EXIT
IN        DC  F'1'
TIME      DS  E
EFFCY     DS  E
POINTS    DS  250E
A         DS  H
19        AFORMAT '(5X,I3/5(E10.3,5X))'
OUT       AFORMAT '(1H1,6HBATCH ,I3,8H RESULTS/1X,16HMEAN DELAY TIME ,  $
                 E14.7/1X,10HEFFICIENCY,6X,E14.7)'
ERR       AFORMAT '(1H1,28HINPUT ERROR IN BATCH NUMBER ,I3/1X,          $
                 28H****************************)'
END       AFORMAT '(1H1,20HINPUT DATA EXHAUSTED)'
          .
          .
          .
          END
```

TABLE 2 DISPLACEMENT TABLE

DISPLACEMENT FROM IBCOM#	FUNCTION OF ROUTINE
0	main entry, formatted READ
+4	main entry, formatted WRITE
+8	second list item, formatted
+12	second list array, formatted
+16	final entry, end of I/O list, formatted
+20	main entry, unformatted READ
+24	main entry, unformatted WRITE
+28	second list item, unformatted
+32	second list array, unformatted
+36	final entry, end of I/O list, unformatted
+40	backspace tape
+44	rewind tape
+48	write tapemark

The linkage sequence is then

```
CNOP    0,4
L       15,=V(IBCOM#)
BAL     14,N(15)
```

where

N is the displacement from IBCOM#.

The content of the parameter lists also varies with the routine (Table 3).

TABLE 3 PARAMETER LISTS

main entry for AREAD and AWRITE, both formatted and unformatted	XL.4'PI',XL.4'UI',AL3(unit) AL1(01), AL3(FMTformat) ...formatted only AL4(EOFi) ...optional AL4(ERRj) ...optional where PI - =0 if neither end nor err is specified =1 if end only is specified =2 if err only is specified =3 if both end and err are specified, UI - =0 if unit is an integer constant =1 if unit is a variable name of type F, and ERRj and EOFi are the names of address constants generated by the macro which point to the users error and end-of-file exists. The uniqueness of such names is guaranteed by the four digit integer constants i and j, provided that the programmer himself does not create a duplicate label.
entry for list item, both formatted and unformatted entry for list array, both formatted and unformatted	AL1(L'item),XL.4'T',XL.4'0',SL2(item) AL1(0),AL3(array) AL1(L'array),XL.4'T',AL.20(E) where L - is the length in bytes of the item or the array element in the I/O list, T - =7 for an E type item or array element =6 D =5 F =4 H =7 for any other type not already mentioned, and E is the number of elements in the array
final list entry, both formatted and unformatted	no parameters
ABACKSPACE, AREWIND and END FILE	XL1'UI,AL3(unit) where UI is defined as above

The AFORMAT macro instruction statement generates a DC instruction statement whose operand field is formed by preceding the operand of the macro instruction statement with the letter C, and whose name field is formed by concatenating the 'statement number' in the name field of the macro instruction statement with the letters FMT. It is the programmer's responsibility to ensure that the symbolic name thus generated (FMTformat) is unique.

The reader is directed to IBM(1970) Appendix K and IBM (1968) Appendix F for a detailed description of the functions and logic of the object-time library subprograms that are included in the user's load module by the linkage editor in response to the external reference to IBCOM#.

5. RELEASE INDEPENDENCE

The macros are release independent except for the possibility of changes being made in the calling sequences to IBCOM#. It is, however, unlikely that these sequences will alter unless a major rewrite of IHCECOMH should occur.

6. REFERENCES

1. IBM (1969) System Reference Library
IBM System 360 Operating System Assembler Language
(File No. S360-21, Form C28-6514-6).

2. IBM (1971) System Reference Library
IBM System 360 and System 370 FORTRAN IV Language
(File No. S360-25 Order No. GC28-6515-8)

3. IBM (1968) Program Logic
IBM System 360 Operating System FORTRAN IV [G] Compiler
Program Logic Manual
Program Number 360S-FO-520
(File No. S360-25(OS) Form Y28-6638-1)

4. IBM (1970) Program Logic
IBM System 360 Operating System FORTRAN IV [H] Compiler
Program Logic Manual
Program Number 360S-FO-500
(File No. S360-25 (OS) Order No. GY28-6642-4)

APPENDIX

JCL REQUIREMENTS

The macros are members of AAE.MACLIB, this library is concatenated with the system macro library SYS1.MACLIB in the catalogued procedures ASMFCLG, ASMFCL and ASMFC.

Several additional JCL cards must be included in the deck when these macros are used. The first of these is a LKED.SYSLIB DD card giving the linkage editor access to the Fortran system library, SYS1.FORTLIB, which contains the modules used during the I/O processes.

```
//LKED.SYSLIB DD   DSN=SYS1.FORTLIB,DISP=SHR
```

There must also be present appropriate DD cards to describe any data set reference number used in the program.

EXAMPLE: The JCL required to compile, link edit and execute an assembler language program with data set reference numbers 1,2,3 and 4 defined as the card reader, the card punch, the printer and a scratch tape respectively is:

```
//EXAMPLE EXEC ASMFCLG
//ASM.SYSIN DD *
        .
        .
        program
        .
        .
/*
//LKED.SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//GO.FT04F001 DD UNIT=(280,,DEFER),DISP=OLD,VOL=SER=SCRTCH,
//               DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300),LABEL=(1,NL)
//GO.FT03F001 DD SYSOUT=A
//GO.FT02F001 DD SYSOUT=B
//GO.FT01F001 DD *
     .
     .
     input data cards
     .
/*
```