THE DESIGN OF AN INTERDATA IMPLEMENTATION OF THE U. S. NAVY MINI-COBOL

BY

TERRY WAYNE ANDERSON-ROVIA

B. S., LOYOLA UNIVERSITY (CHICAGO), 1964

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1976

Approved by:

Major Professor

ACKNOWLEDGMENTS

I would like to express my deep gratitude and appreciation to Capt. Grace M. Hopper, USNR and the fine young ladies and men of the U. S. Navy Programming Languages Section without whose generosity Kansas State University would not have the MINI-COBOL compiler. I would also like to express my appreciation to Daniel Codespoti, Lee Allen, and Richard McBride for their assistance in technical matters.

TWA

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE PAGE NUMBERS CUT OFF

THIS IS AS RECEIVED FROM THE CUSTOMER

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The use of mini-computers is constantly expanding in the business world. One of the major difficulties encountered when a mini-computer is chosen for an application is the language it uses. Should a company or institution decide that one manufacturer produces a machine suitable for one application and another manufacturer produces a machine suitable for another application, the company must be aware that more than likely the two machines will use different assemblers and will necessitate the training of the company programming staff in two languages. Writing in FORTRAN is a step in the direction of reducing training expense; however, the various manufacturers have implemented specialized "subsets" of FORTRAN which tend to make portability difficult. Additionally, the present trend is to use COBOL for business applications in preference to FORTRAN.

COBOL, however, suffers basically from the same "illness" which affects FORTRAN despite the best efforts of the American National Standards Institute. Each manufacture has seen fit to add features to the basic subset as approved by the CODASYL committee generating in turn a myriad of hybrids which cause undesirable delays in the conversion effort when programs are transferred from one machine to another.

The U. S. Navy, being a large user of computers of various manufacture, long suffered from the ailments as described above until it decided to restrict the language options to only those officially sanctioned by ANSI. In their attempts to resolve the problem of portability,

the U. S. Navy conducted research and discovered that of the options offered as standard by ANSI, the majority of programmers really only utilized a relatively small portion. As a result of this study, the U. S. Navy constructed a COBOL compiler specifically designed for a mini-computer. While the special compiler is designed only for a specific mini-computer at this time, programs written in "MINI-COBOL" can be transferred with minor modifications to those statements considered machine dependent by ANSI to "MAXI" computers with a COBOL compiler of an equal or greater level. Appendix A contains a description of U. S. Navy MINI-COBOL.

While the U. S. Navy accomplished an admirable task, they did not make any attempt to design the MINI-COBOL compiler as a portable software item, i. e., the compiler itself can run on the specific machine of one manufacturer. The desire to use COBOL on other mini-computers gave birth to this project which will deal with the transfer of the U. S. Navy MINI-COBOL compiler designed for the NOVA mini-computer to the INTERDATA mini-computer. While the project may appear to be rather specific in nature, the same principles may be used as they are presented in this paper to transfer the MINI-COBOL compiler to any mini-computer which uses a micro-coded instruction set, and this feat can be effected with minor modification to the FORTRAN interpreter as designed by the Navy. Furthermore, the design presented allows for the use of a virtual system and for the use of other mini-computers as I/O device drivers. Also presented for discussion in this report is a description of the MODEL 80 INTERDATA micro-processor which when used in conjunction with the presentation of the micro-code instruction formats should give the

reader a basic framework of reference to understand the micro-code presented as emulated NOVA instructions. A description of each of the various mnemonics used in the micro-code is given to facilitate the viewing of the emulation code as presented. Formats of the NOVA instructions which will be emulated in this project are presented so that the viewer may inspect them and gain a more comprehensive view of the emulation technique. Specification of the Fetch and Decode modules found in the emulator will be presented along with the micro-code generated for each NOVA instruction is included in this report. A sample high-level algorithm along with an explanation of the abbreviations is given including a step-by-step trace table of two typical NOVA instructions with a condensed output listing in an INTERDATA-like assembly language and in the micro-code generated. A discussion on modification of the Navy supplied MINI-COBOL interpreter will be presented along with the specifications of the MINI-COBOL language, the MINI-COBOL compiler system, and the actual source code for all the modules used in the MINI-COBOL system.

Chapter 2

THE DESIGN OF AN INTERDATA IMPLEMENTATION OF MINI-COBOL

Introduction

Two alternatives were seriously considered in designing an implementation of MINI-COBOL on the INTERDATA; the first alternative would have been to code the routines for the COBOL verbs in INTERDATA micro-instruction code; the other alternative was to write a micro-instruction emulation of the NOVA instructions. Should I have chosen the first alternative, every module in the MINI-COBOL compiler would have to be micro-coded. This approach would not utilize the work already accomplished by the U. S. Navy resulting in a genuine duplication of effort on my part. Furthermore, the net result of such an effort on my part would not have, in any way, facilitated the inter-connection and compatibility of the NOVA and INTERDATA mini-computers. The only advantage that presented itself with the first alternative was speed of execution due to the fact that the actual function of any given module would be taken into consideration when being coded. Micro-code of the function would be much faster than duplication of the NOVA architecture as would be the case with the second alternative.

The second alternative, emulation of NOVA instructions on the INTERDATA, would provide the following advantages:

1) MINI-COBOL could be utilized on the INTERDATA;

2) the NOVA instruction set could be utilized on the INTERDATA mini-computer;

3) because the second alternative was "general" in its approach and because micro-code was flexible, it would allow for future expansion into the virtual machine design that is currently being worked upon by others in our department.[1,2]

There were some disadvantages to the second alternative, however, mainly that the INTERDATA, having a more powerful instruction set, could execute the same function in its own instruction set much faster than an emulated NOVA instruction set. However, I felt that the advantages of flexibility and possible growth far outweighed the disadvantage of speed in this case.

Implementation Overview - High Level

Trying to effect the second alternative, turned out to be an interesting task. Since the first two phases of the MINI-COBOL compiler are written in COBOL, it would be difficult to efficiently modify them. However, the interpreter phase seemed to lend itself naturally to modification so as to effect the second alternative. (see Appendix A - Interpreter). Basically, the scheme is as follows:

1) the user creates his COBOL source file;

2) he then invokes the COBOL compiler with the "START" instruction;

3) phase I and phase IA would create the tuples required for execution (see Appendix A - Phase I, Phase IA);

4) the user would then begin execution of his program by invoking the MINI-COBOL run time routines by using the "COBOL" "R" instructions;

5) the "COBOL" "R" instructions invoke the FORTRAN interpreter (see Appendix A - Interpreter);

6) after the interpreter allocates memory to the literals and values declared in the user program, it begins to read the execution tuples. It then examines the operation code and branches to a dispatch table (see Appendix A - figure A.4, step 5) which then gives the address of an entry point into a second dispatch table (see step 6) which contains the list of NOVA sub-routines in the order required to effect the COBOL verb. It is precisely at this point that the following modification would be inserted:

a) A "call" to the Fetch routine would be inserted, passing as parameters the list of NOVA sub-routines needed to effect the COBOL verb;

b) the Fetch routine would then read the instructions as written in NOVA assembler, one at a time, for each module listed, invoking for each instruction read, a Decoder which will translate the NOVA instruction into the Instruction Register format (see Appendix D - Instruction Register). The Decoder will then branch to the appropriate Micro-code routine for this instruction based upon the OP-CODE found in the Instruction Register.

c) the micro-coded routines will execute the specific function and return to the Decoder.

d) the Decoder will continue to construct the Instruction Register and branch to the appropriate micro sub-routine until it has completed all of the options requested by the original NOVA instruction. Upon completion, the Decoder will then return to the Fetch routine.

e) the Fetch routine will continue to read NOVA instructions until end-of-file condition is reached. It will then return to the interpreter.

f) the interpreter will then read the next execution tuple, thus closing the emulation loop.

To the user, the entire process is transparent. He is aware of only the fact that he has created a COBOL file and has invoked the execution of that file, thus anyone familiar with COBOL per se will be able to use the system with little or no training.

Summary

Thus far, I have presented the various conditions which influenced the emulator design and I have presented a high-level overview of the emulator design in light of those considerations. Succeeding chapters shall present in detail the specific design of the Fetch and Decoder modules.

Chapter 3

THE DETAILED DESIGN

Introduction

In order that the viewer gain a more complete understanding of the emulator, I will present a short section on the NOVA instruction formats. Throughout the rest of this report, the sixteen bits which contain any one of the various formats of the NOVA instruction shall be referred to as NI (0:15), wherein NI represents the abbreviation for NOVA Instruction and (0:15) represents the abbreviation for bits zero through fifteen inclusive. The Nova instructions should be considered as the input into the Fetch module which will be the second section in this chapter.

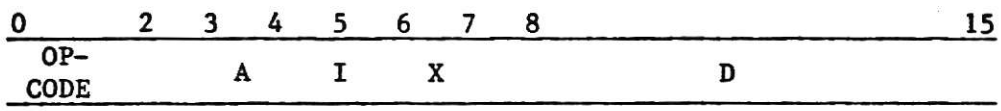
In the presentation of the Fetch module, I will use a high-level algorithmic language which will give the detailed function the module should accomplish upon invocation via the call statement from the FORTRAN interpreter (see Appendix A - Interpreter).

Since the next operation in the normal flow of the proposed emulation in the Decoder module, I will present the rules of precedence to be used in the module when constructing the Instruction Register (IR). I will also present the algorithm to be used to emulate the subset of NOVA instructions and functions I have chosen. The algorithms will be written in an INTERDATA-like assembler language for the viewer's benefit as micro-code is hardware directed and rather difficult to follow. (see Appendix E - Assembler Op-Codes, Micro-coded Emulation).

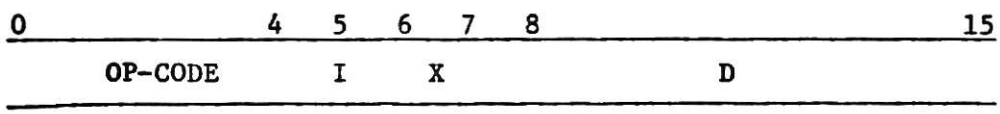
NOVA Instruction Formats

The NOVA instruction set is basically divided into three categories

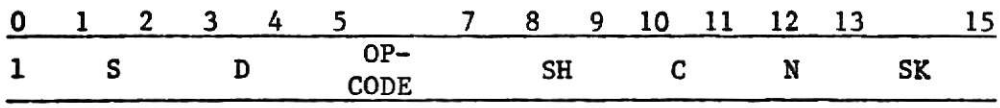
with reference to format. I shall not present any discussion with regard to I/O instructions as they are treated as a special case as described later in this report. The formats of the classes of instructions which this emulator will receive as input are presented below.[3]



MOVE DATA INSTRUCTIONS



MODIFY MEMORY/JUMP INSTRUCTIONS



ARITHMETIC AND LOGIC INSTRUCTIONS

Figure 3.1

Selected NOVA Instruction Formats

An explanation of the abbreviations used in the above figure of selected NOVA formats is given below.

OP-CODE - this is a three bit field which specifies the function requested. An exception to this rule is in the Memory/Jump instruction which has a total of five bits. Bits three through four of the Memory/Jump instruction further delineate the specific function desired.

A - this represents one of four general purpose accumulators available for use by the application.

I - this field represents the address type, whether direct or indirect.

X - this field specifies an index register, if any.

D - these eight bits represent the displacement.

1 - this is a constant indicating that the instruction is an Arithmetic and Logic instruction.

S - this is an accumulator source address.

D - in the Arithmetic and Logic instruction, this represents the accumulator destination address.

SH - this two bit field represents one of three possible shift functions.

C - this two bit field represents one of three possible operations that may be performed with respect to the Carry bit.

N - this is the No-Load function bit. If it is equal to '1'B, then the loading of the Carry bit and accumulator is inhibited.

SK - this field represents one of eight possible functions with respect to skipping the next sequential instruction.

Because of the rather limited mnemonics of the NOVA instruction set, it is quite easy to jump to the conclusion that the assembler is underpowered or limited. However, further study of the capabilities of the instruction set will reveal the fact that up to six separate operations may occur in any given Arithmetic and Logic instruction making the emulation of this phenomenon, at best, a complicated process.

The Fetch Module

With the input to the Fetch module described above, the function of the Fetch module can be presented. As was previously mentioned in preceeding chapters, the MINI-COBOL compiler utilizes a FORTRAN interpreter to read the execution tuples. The interpreter reads a tuple,

and then branches to an address which contains a list of addresses of NOVA assembler sub-routines in the order required to effect the operation requested by the COBOL verb. (see Appendix A for a complete discussion of this process).

At the point in the normal flow of operation described above wherein the FORTRAN interpreter branches to an address containing a list of addresses of NOVA sub-routines, I propose the following changes;

1) the branch to a list of address be replaced with a "call" statement to the Fetch module;

2) the list of addresses be passed as an input parameter to the Fetch module.

The effect of these proposed changes is that the Fetch will be allowed to read the first binary NOVA instruction of the routine as referenced by the parameter and it may then pass control to the Decoder, such that:

```
          (PARM1, PARM2...PARMn)---(ADR1, ADR2...ADRn)
          CALL "FETCH" USING PARM1, PARM2...PARMn
          RETURN TO READ-NEXT-TUPLE
          .
          .
          .
FETCH:    EQUATE (INPUT, PARM1, PARM2,...PARMn)
          READ (INPUT) AT END RETURN
          (NI_BUFFER)<---NI (0:15)
          CALL "DECODE" USING NI_BUFFER
          BRANCH TO FETCH
```

Figure 3.2

Fetch Algorithm

The same techniques as presented above can be applied to facilitate the virtual machine, in that, prior to using an instruction which utilizes an address in main memory, a "call" can be made to the paging routines which can resolve the address passed as a parameter and return a real address as an argument so the instruction can be processed.[2]

I/O instructions can be expedited in a manner similar to the aforementioned, i. e., upon recognition of an I/O instruction, the Decoder (in this case) will construct an SVC passing as an argument the buffers involved. The computer (of the SVC) can pass the contents of these buffers as a message to the computer acting as the I/O device handler.[2]

The Decoder Module

As can be inferred from the preceeding section, the Decoder which is the very heart of the proposed emulator, is the next topic of discussion in this report. Since the Decoder is the largest and most complex module of the emulator, I will present it as a series of discussions based upon the rules of precedence it must use. All algorithms presented in this section will be written in an INTERDATA-like assembler language for purposes of clarity. In the actual emulator, the algorithm will be micro-coded instructions.

Rules of precedence.

When the Decoder is invoked by the Fetch module, it will receive an input parameter, an entire NOVA instruction in binary format, to be decoded. The Decoder, in turn, must examine the given instruction according to the following rules of precedence:

1) Scan the instruction for the presence of indirection. If indirection is present, then call the paging routine which will resolve

the address involved into an effective address. Since one or more levels of indirection may be involved, this particular sub-function may be invoked several times until the indirect bit (NI (5)) is found to be '0'B.

2) Scan for the desired mnemonic and branch to the routine based on the specific mnemonic requested. All routines use temporary work areas identified in the corresponding code and R12 (the shifter).

3) Scan for the Carry function and based upon the particulars of the mnemonic involved from step two above, load the Carry bit into a work area identified as R15, as requested by the Carry function which is specified in R10 which represents NI (10:11).

4) Scan for the Shift function. If it is present, the Decoder will branch to the proper routine based on the specific function requested.

5) Scan for the Load function (NI 5)). If it is equal to '0'B, then branch to the routine which will load R12 to the Accumulator as specified by Ni (3:4). Should the No-Load function be requested, the Accumulators as specified by NI (1:2) and NI (3:4) will be unaffected and will retain their original contents.

6) Scan for the SKIP function and branch to the routine which shall effect the function requested by NI (13:15).

Emulation routines.

Using the rules of precedence as I have just presented and the register assignments presented below, the following section will present an INTERDATA-like assembler language algorithm for each NOVA instruction function that will be used in this emulator (see Appendix E - Assembler Op Codes and Micro coded emulation). As I have previously mentioned, I have chosen assembler for purposes of clarity but, in the actual emulator, micro-coded instructions will be generated.

R0 - R3 - used as Accumulator Source (NI (1:2)) and Accumulator Destination (NI (3:4)).

R4 - R5 - not used.

R6 - used as "Branch to" address.

R7 - used as "return" address.

R8 - used to hold number of shift positions and a general work register.

R9 - used as a general work register.

R10 - contains the Carry function (NI (10:11)).

R11 - used as a general work register.

R12 - used as the NOVA Shifter.

R13 - used as a general work register.

R14 - used as a general work register.

R15 - used as the Carry Bit of the Shifter.

Table 3.1

Emulation Register Assignments

1. Emulation of the JSR

Upon encounter of the JSR operation code (NI (0:4)), the NOVA will load the address of the next sequential instruction following the JSR instruction into Accumulator three. It will then branch to the effective address of the sub-routine as indicated by bits NI (6:15).

```
1    LH    R6,(NI 6:15)          LOAD "BRANCH TO" ADDR.
2    BALR  R3,R6                 BRANCH AND LINK REG.
```

Figure 3.3

Emulated JSR

NI (6:15) is the effective address of the routine to which the branch will occur. The Decoder must resolve the address as specified by bits 6:15 prior to loading Register 6. Register 3, emulating NOVA Accumulator 3, will have its previous contents destroyed by this instruction as it will contain the "return" address.

2. Emulation of the JMP

The NOVA will load the next sequential instructions' address into the Program Counter and branch unconditionally to the address specified by NI (6:15), when the JMP operation code is encountered in NI (0:4).

```
1    BAL   R7,(NI 6:15)          BRANCH TO EFFECTIVE ADR.
```

Figure 3.4

Emulated JMP

The address of the next sequential instruction is loaded into Register 7. The Decoder must resolve the effective address as specified by NI (6:15).

3. Emulation of the DSZ

The NOVA will subtract 1 from the contents of the location as specified by NI (6:15) and will place the result back into the location as specified by NI (6:15). If the result is equal to zero, then the next instruction in sequence will be skipped.

```
1    LH    R12,(NI 6:15)         GET AREA REFERENCED
2    SIS   R12,X'01'             SUBTRACT ONE FROM AREA
3    BZ    *+3                   IF ZERO THEN SKIP NSI
```

```
4    STH   R12,(NI 6:15)        PUT BACK RESULTS
5    B     *+3                  BRANCH OUT OF ROUTINE
6    STH   R12,(NI 6:15)        PUT BACK RESULTS
7    B     FETCH                BRANCH TO FETCH ROUTINE
```

Figure 3.5

Emulated DSZ

4. Emulation of the ISZ

The NOVA will add 1 to the contents of the location as specified by NI (6:15) and will place the results back into the location as specified by NI (6:15). If the results of the add is equal to zero, then the next instruction in sequence will be skipped.

```
1    LH    R12,(NI 6:15)        GET AREA REFERENCED
2    AIS   R12,X'01'            ADD ONE TO AREA
3    BZ    *+3                  EQUALS ZERO, SO SKIP
4    STH   R12,(NI 6:15)        PUT BACK RESULTS
5    B     *+3                  BRANCH OUT OF ROUTINE
6    STH   R12,(NI 6:15)        PUT BACK RESULTS
7    B     FETCH                BRANCH TO FETCH ROUTINE
```

Figure 3.6

Emulated ISZ

The Decoder must resolve NI (6:15) into an effective address. Once this has been accomplished, 1 will be added to the area referenced and the results compared to zero. If the result is zero, the contents are

restored to the area referenced and the next sequential instruction is fetched. Otherwise, the results are restored and the flow continues.

5. Emulation of the LDA

The NOVA will load the area of memory as referenced by NI (6:15) into the Accumulator specified by NI (3:4).

```
1    LH    (NI 3:4),(NI 6:15)      LOAD REG. FROM MEM.
```

Figure 3.7

Emulated LDA

The Decoder must translate NI (3:4) into a hexadecimal digit 0 - 3, then it must resolve the address of the area of main memory as represented by NI (6:15). The contents of the area referenced will then be loaded into the Register specified by NI (3:4).

6. Emulation of the STA

The NOVA will load the contents of the Accumulator as referenced by NI (3:4) into the area of main memory as specified by NI (6:15).

```
1    STH   (NI 3:4),(NI 6:15)      STORE ACUM.
```

Figure 3.8

Emulated STA

The Decoder must translate NI (3:4) into a hexadecimal digit 0 - 3. Then the Decoder must resolve the address as specified by NI (6:15). The contents of the Register specified will be loaded into the area referenced.

7. Emulation of the ADD

The NOVA will add the contents of the Accumulator as specified by NI (1:2) to the contents of the Accumulator as specified by NI (3:4) and will place the result in the Shifter. If the sum is equal to or greater than 2E16, then the value specified by NI (10:11) will be complemented as the Carry bit; otherwise, the value as specified by NI (10:11) will be used as the Carry bit.

```
1    LHR   R12,(NI 1:2)          GET ACCUM. SOURCE
2    LHR   R13,(NI 3:4)          GET ACCUM. DESTINATION
3    AHR   R12,R13               ADD THE ACCUMULATORS
4    CLHI  R12,X'FF'             GREATER OR EQUAL 2E16?
5    BNC   *+3                   YES
6    LH    R10,(NI 10:11)        NO, GET CARRY FUNCTION
7    B     *+3                   BRANCH OUT OF ROUTINE
8    LH    R10,(NI 10:11)        GET CARRY FUNCTION
9    XHR   R10,X'03'             COMPLEMENT CARRY FUNCTION
```

Figure 3.9

Emulated ADD

The contents of NI (1:2) and NI (3:4) must be translated into hexadecimal digits 0 - 3. The Decoder will then load the contents of these registers into R12 and R13, respectively. The registers are then added into R12 and the result checked to determine if it is equal or greater than 2E16. If it is, then the Carry function (NI (10:11)) is complemented into R10; otherwise, the Carry function is loaded into R10 unaltered.

8. Emulation of the SUB

The NOVA subtracts by adding the twos complement of the number from NI (1:2) to the number from NI (3:4) and will place the result in the Shifter. If the number in the Accumulator specified by NI (3:4) is greater or equal to the number in the Accumulator specified NI (1:2), then the complement of the value specified by Carry function (NI (10:11)) will be supplied; otherwise, the value of the Carry function as given will be supplied to the Carry bit.

```
1    LHR   R12,(NI 1:2)        GET ACS INTO R12
2    XHI   R12,X'FF'           COMPLEMENT IT
3    AHI   R12,X'01'           ADD ONE TO MAKE 2'S COMP.
4    LHR   R14,(NI 3:4)        GET ACD INTO R14
5    CHR   R14,R12             ACD G.E. ACS?
6    BC    *+4                 NO, GO LOAD CARRY
7    LH    R10,(NI 10:11)      YES, GET CARRY FUNCTION
8    XHI   R10,X'03'           COMPLEMENT IT
9    B     *+2                 GO ADD
10   LH    R10,(NI 10:11)      GET CARRY FUNCTION
11   AHR   R12,R14             ADD ACS TO ACD
```

Figure 3.10

Emulated SUB

NI (1:2) will be translated by the Decoder into hexadecimal digit 0 - 3. The contents of the register specified will be loaded into Register 12 where it will be formed into the twos complement. Register 12 is then added to the contents of R14 which holds the number specified

by NI (3:4). If R14 is greater or equal to R12, the Carry function is complemented; otherwise it is loaded as specified by NI (10:11) into R10.

9. Emulation of the AND

The NOVA will logically "and" the contents of the Accumulator specified by NI (1:2) with the contents of the Accumulator specified by NI (3:4).

```
1   LHR   R12,(NI 3:4)          GET ACD INTO R12
2   LHR   R13,(NI 1:2)          GET ACS INTO R13
3   NHR   R12,R13               AND INTO R12
4   LHR   R10,(NI 10:11)        PUT CARRY FUNCTION IN R10
```

Figure 3.11

Emulated AND

The Decoder must translate NI (1:2) and NI (3:4) into hexadecimal digits 0 - 3. The contents of the Accumulators as specified by NI (1:2) and NI (3:4) are loaded into R13 and R12 respectively. The contents are "ANDed" into R12 and the Carry function is loaded into R10.

10. Emulation of the COM

The NOVA will logically complement the contents of the Accumulator as specified by NI (1:2) into the Shifter and place the value as specified by the Carry function into the Carry bit.

```
1   LHR   R12,(NI 1:2)          LOAD ACS INTO R12
2   XHI   R12,X'FF'             LOGICALLY COMPLEMENT R12
3   LH    R10,(NI 10:11)        LOAD CARRY TO R10
```

Figure 3.12

Emulated COM

ACS as specified by NI (1:2) and translated into hexadecimal by the Decoder is loaded and complemented into R12. The Carry function is loaded into R10.

11. Emulation of the NEG

The NOVA will place the twos complement of the number contained in the Accumulator specified by NI (1:2). If the number is zero, the complement of the value specified by the Carry (NI 10:11) will be placed in the Carry bit; otherwise, the value as originally specified will be placed in the Carry bit.

```
1    LHR   R12,(NI 1:2)        LOAD ACS INTO R12
2    LHR   R11,R12             LOAD ACS INTO R11
3    XHI   R12,X'FF'           COMPLEMENT R12
4    AHI   R12,X'01'           ADD ONE
5    LHI   R13,X'00'           PUT ZERO IN R13
6    CHR   R11,R13             IS ACS EQUAL TO ZERO?
7    BNE   *+5                 NO
8    LH    R10,(NI 10:11)      PUT CARRY INTO R10
9    XHI   R10,X'03'           COMPLEMENT CARRY
10   B     *+2                 BRANCH OUT
11   LH    R10,(NI 10:11)      NOT ZERO, LOAD CARRY TO R10
```

Figure 3.13

Emulated NEG

ACS as specified by NI (1:2) is loaded into Register 12. Register 12 is complemented into the twos complement form. If ACS was zero, then

the Carry value is complemented; otherwise, the Carry function is loaded as it was originally into Register 10.

12. Emulation of the ADC.

The NOVA will add the logical complement of the number contained in the Accumulator as specified by NI (1:2) to the number contained in the Accumulator as specified by NI (3:4) and place the results in the Shifter. If the number contained in the Accumulator specified by NI (1:2) is less than the number contained in the Accumulator specified by NI (3:4), then the complement of the value as indicated by NI (10:11) will be loaded into the Carry bit; otherwise the value indicated will be loaded into the Carry bit.

```
1    LHR   R12,(NI 1:2)          LOAD ACS INTO R12
2    XHI   R12,X'FF'             COMPLEMENT IT
3    LHR   R14,(NI 3:4)          LOAD ACD INTO R14
4    AHR   R12,R14               ADD ACS TO ACD
5    LHR   R13,(NI 1:2)          GET ACS INTO R13
6    CLHR  R13,R14               IS ACD GREATER THAN ACS?
7    BC    *+3                   YES, COMPLEMENT CARRY
8    LH    R10,(NI 10:11)        NO, LOAD CARRY
9    B     *+3                   BRANCH OUT
10   LH    R10,(NI 10:11)        LOAD CARRY
11   XHI   R10,X'03'             COMPLEMENT IT
```

Figure 3.14

Emulated ADC

ACS as specified by NI (1:2) is complemented and added to ACD as specified by NI (3:4) and the results placed in the Shifter. The two Accumulators are compared and if ACD is greater than ACS the value specified by NI (10:11) is complemented into R10; otherwise, the value specified is placed into R10.

13. Emulation of the INC.

The NOVA will add one to the number specified in the Accumulator represented by NI (1:2) and will place the result in the Shifter. If ACS contained (2E16) -1, the Carry value is complemented; otherwise, the value supplied is loaded into the Carry bit.

```
1    LHR   R12,(NI 1:2)          GET ACS INTO R12
2    AHI   R12,X'01'             ADD ONE TO R12
3    LHR   R13,(NI 1:2)          GET ACS INTO R13
4    CLHI  R13,X'FE'             ACS EQUAL (2E16) -1?
5    BNE   *+4                   NO, GO LOAD CARRY
6    LH    R10,(NI 10:11)        YES, GET CARRY
7    XHI   R10,X'03'             COMPLEMENT IT
8    B     *+2                   BRANCH OUT
9    LH    R10,(NI 10:11)        LOAD CARRY
```

Figure 3.15

Emulated INC

One is added to the number contained in the Accumulator as represented by NI (1:2). If the number was equal to (2E16) -1 originally,

the Carry value is complemented; otherwise, the value specified by NI (10:11) is loaded into R10.

14. Emulation of the MOV

The NOVA will place the contents of ACS (NI (1:2)) in the shifter and place the Carry bit as specified by (NI (10:11)) in the Carry.

```
1    LHR   R12,(NI 1:2)          GET ACS INTO R12
2    LH    R10,(NI 10:11)        GET CARRY INTO R10
```

Figure 3.16

Emulated MOV

The contents of the register as specified by NI (1:2) are loaded into the Shifter (R12). The value of the Carry function as specified by NI (10:11) is loaded into R10.

15. Emulation of the Z, 0, C functions.

The NOVA will examine NI (10:11) using the following rules:

1) if NI (10:11) equals '00'B, then the current state of the Carry bit will be used;

2) if NI (10:11) equals '01'B (Z), then a zero will be supplied to the Carry bit;

3) if NI (10:11) equals '10'B (0), then one will be supplied to the Carry bit;

4) if NI (10:11) equals '11'B (C), then the current state of the Carry bit will be complemented.

```
1    CLHI  R10,X'01'           EQUAL ZERO?
2    BE    *+6                 YES
3    CLHI  R10,X'10'           EQUAL ONE?
4    BE    *+6                 YES
```

Figure 3.17

Emulated Z, 0, C

```
5    CLHI  R10,X'11'          EQUAL COMPLEMENT?
6    BE    *+6                YES
7    B     *+6                DO NOT ALTER CARRY
8    LHI   R15,X'00'          LOAD ZERO IN R15
9    B     *+4                BRANCH OUT
10   LHI   R15,X'01'          LOAD ONE IN R15
11   B     *+2                BRANCH OUT
12   XHI   R15,X'FF'          COMPLEMENT CARRY
```

Figure 3.17 (Cont.)

Emulated Z, 0, C

16. Emulation of the L, R, S functions.

The NOVA will examine NI (8:9) and will respond according to the following rules;

1) if NI (8:9) equals '00'B, then no shift will take place;

2) if NI (8:9) equals '01'B, then rotate left one place. Bit 0 is rotated into the Carry bit and the Carry bit into bit 15;

3) if NI (8:9) equals '10'B, then rotate right one place. Bit 15 is rotated into the Carry position and the Carry bit into bit 0.

4) if NI (8:9) equals '11'B, then swap half of the 16 bit result. The Carry bit is not affected.

```
1    LHR   R13,(NI 8:9)       GET SHIFT FUNCTION
2    CLHI  R13,X'01'          IS IT "L"?
3    BE    *+6                YES
4    CLHI  R13,X'02'          IS IT "R"?
```

Figure 3.18

Emulated L, R, S

```
5    BE     *+12              YES
6    CLHI   R13,X'03'         IS IT "S"?
7    BE     *+24              YES
8    B      *+40              NO SHIFT
9    SLLS   R12,X'01'         SHIFT LEFT 1
10   BC     *+4               WAS THERE A CARRY?
11   AHR    R12,R15           NO, ADD CARRY BIT
12   LHI    R15,X'00'         PUT ZERO IN CARRY BIT
13   B      *+27              BRANCH OUT
14   AHR    R12,R15           YES, ADD CARRY BIT
15   LHI    R15,X'01'         PUT ONE IN CARRY BIT
16   B      *+24              BRANCH OUT
17   SRLS   R12,X'01'         SHIFT RIGHT 1
18   BC     *+7               WAS A ONE SHIFTED?
19   CLHI   R15, X'01'        NO, IS THERE A 1 CARRY?
20   BE     *+2               YES
21   B      *+19              NO, BRANCH OUT
22   AHI    R12,X'80'         ADD THE CARRY 1
23   LHI    R15,X'00'         ZERO THE CARRY
24   B      *+16              BRANCH OUT
25   CLHI   R15,X'01'         YES, DOES CARRY EQUAL 1?
26   BE     *+3               YES
27   LHI    R15,X'01'         NO, PUT 1 IN CARRY
28   B      *+12              BRANCH OUT
29   AHI    R12,X'80'         ADD CARRY
30   B      *+10              BRANCH OUT
```

Figure 3.18 (Cont.)

Emulated L, R, S

```
31   LHI   R8,X'08'          LOAD # OF SHIFTS
32   SLLS  R12,X'01'         SHIFT LEFT 1
33   BC    *+3               WAS A 1 SHIFTED
34   LHI   R9,X'00'          NO, PUT 0 IN R9
35   B     *+3               GO DECREMENT SHIFTS
36   LHI   R9,X'01'          YES, PUT 1 IN R9
37   AHR   R12,R9            ADD SHIFT BITS
38   SHI   R8,X'01'          DECREMENT SHIFT
39   BNZ   *-7               GO SHIFT AGAIN
```

Figure 3.18 (Cont.)

Emulated L, R, S

Based on NI (8:9), a branch is made to the proper routine to expedite the requested function of Shift L, Shift R, or Swap halves.

17. Emulation of the Load/No Load.

The NOVA will examine NI (12) for a '1'B. Should '1'B be encountered, ACD (NI (3;4)) will not be loaded. If NI (12) contain '0'B, then ACD (NI (3:4)) will be loaded.

```
1    LH    R9,(NI 12)        GET LOAD/NO LOAD FUNCTION
2    LH    R11,X'00'         PUT ZERO IN R11
3    CLHR  R9,R11            EQUAL ZERO?
4    BNE   *+2               YES, BRANCH OUT (NO LOAD)
5    LHR   R12,(NI 3:4)      NO, LOAD R12 INTO ACD
```

Figure 3.19

Emulated Load/No Load

If NI (12) is equal to one then no loading of ACD as specified by NI (3:4) will occur; otherwise, ACD will be loaded with the contents of Register 12.

18. Emulation of the SKIP function.

The NOVA will examine NI (13:15) and will respond according to the following rules;

1) if NI (13:15) equal zero, then it will not Skip;

2) if NI (13:15) equal one, then it will always skip the next sequential instruction;

3) if NI (13:15) equal two, then it will skip on the Carry bit being equal to zero;

4) if NI (13:15) equal three, then it will skip the next sequential instruction if the Carry bit is nonzero;

5) if NI (13:15) equal four, then it will examine the Shifter and skip the next sequential instruction if the result is zero;

6) if NI (13:15) equal five, then it will examine the Shifter and skip the next sequential instruction if the result is nonzero;

7) if the NI (13:15) equal six, then it will examine the Shifter and the Carry bit and skip the next sequential instruction if either is equal to zero;

8) if NI (13:15) equal seven, then it will examine the Shifter and the Carry bit and skip the next sequential instruction if both are equal to nonzero.

```
1    LH     R9,(NI 13:15)        GET SKIP FUNCTION
2    CLHI   R9,X'00'             EQUAL NO SKIP?
3    BE     *+15                 YES
4    CLHI   R9,X'01'             EQUAL SKIP?
5    BE     *+14                 YES
6    CLHI   R9,X'02'             EQUAL SKIP ZERO CARRY?
7    BE     *+14                 YES
8    CLHI   R9,X'03'             EQUAL SKIP NONZERO CARRY?
9    BE     *+15                 YES
10   CLHI   R9,X'04'             EQUAL SKIP ZERO RESULTS?
11   BE     *+16                 YES
12   CLHI   R9,X'05'             EQUAL SKIP NONZERO RES.?
13   BE     *+17                 YES
14   CLHI   R9,X'06'             EQUAL SKIP EITHER ZERO?
15   BE     *+18                 YES
16   CLHI   R9,X'07'             EQUAL SKIP BOTH NONZERO?
17   BE     *+21                 YES
18   B      FETCH                GO FETCH NXT. SEQ. INST.
19   READ (INPUT)                GET NXT. SEQ. INST.
20   B      FETCH                GO FETCH NXT. SEQ. INST.
21   CLHI   R15,X'00'            CARRY EQUAL ZERO?
22   BE     *-3                  YES, GO READ, FETCH
23   B      FETCH                NO, GO FETCH NXT. SEQ. IN
24   CLHI   R15,X'01'            CARRY EQUAL 1?
```

Figure 3.20

Emulated SKIP

```
25   BE     *-6                 GO READ, FETCH
26   B      *-8                 GO FETCH
27   CLHI   R12,X'00'           RESULTS EQUAL ZERO?
28   BE     *-9                 YES, GO READ, FETCH
29   B      *-11                NO, GO FETCH
30   CLHI   R12,X'00'           RESULTS EQUAL ZERO?
31   BNE    *-12                NO, GO READ, FETCH
32   B      *-14                YES, GO FETCH
33   CLHI   R12,X'00'           RESULTS EQUAL ZERO?
34   BE     *-15                YES, GO READ, FETCH
35   CLHI   R15,X'00'           CARRY EQUAL ZERO?
36   BE     *-17                YES, GO READ, FETCH
37   B      *-19                NO, GO FETCH
38   CLHI   R12,X'00'           RESULTS EQUAL ZERO?
39   BE     *-21                YES, GO FETCH
40   CLHI   R15,X'00'           CARRY EQUAL ZERO?
41   BE     *-23                YES, GO FETCH
42   B      *-23                GO READ, FETCH
```

Figure 3.20 (Cont.)

Emulated SKIP

Based on NI (13:15), the Decoder will branch to the appropriate routine and test the results in the Shifter and Carry bit for zero/nonzero conditions and will skip or not skip the next sequential instruction based upon the conditions it was checking.

Summary

In the preceeding pages, I have presented a detailed design of the emulator I propose to effect the transfer of the U.S. Navy MINI-COBOL compiler system to the INTERDATA 80 mini-computer. The input to the emulator, the NOVA instruction, was thoroughly discussed with the intent to help the reader understand the scope and overall function of the emulator. Specifications of the Fetch module were given and an algorithm for the implementation were given. It was also noted that the techniques for implementation of the Fetch module could be easily transferred to effect a virtual machine and the use of a mini-computer as an I/O device handler.

I presented the detail design of the Decoder module, explaining the rules of precedence to be used in the implementation and the functional description of each NOVA operand, including an INTERDATA-like assembler language algorithm and explanation of the algorithm.

Chapter 4

VERIFICATION

Introduction

The most difficult task in any design project is pre-implementation verification of the design itself. The existing tools to accommodate verification are few at most and awkward at best. However, one of the more trustworthy methods is the trace table and that is the method I chose to test my detailed design as presented in the preceeding pages of this report.

Abbreviations In Sample Algorithm

The algorithm used to emulate is rather long and complex in nature; therefore, I decided to use a high level language to describe the various operations that were to be performed based upon the rules of precedence mentioned previously. I do use some rather standard assembler abbreviations to express myself in this algorithm which are as follows:

R1 - R15 - Represents General Registers 1 through 15 inclusive in the INTERDATA.

R0 - R3 - Represents AC0 - AC3 of the NOVA as emulated in the INTERDATA.

R12 - Represents the Shifter of the NOVA as emulated in the INTERDATA.

Figure 4.1

Abbreviations

R10 - Represents the Carry function (NI 10:11).

R15 - Represents the Carry bit of the NOVA.

IR - Represents the Instruction Register, i.e., the output buffer area.

PSW - Represents the Program Status Word of the INTERDATA.

NI BUFFER - NOVA instruction read-in buffer consisting of bits (0:15) of the NOVA Instruction.

NI BUFFER BEGINNING - Represents the beginning address of the read-in buffer for the NOVA instruction.

NI BUFFER ENDING - Represents the ending address of the read-in buffer for the NOVA instruction.

C, X, B-Represents "constant" which can be either alphanumeric (C 'LHI'), Hexadecimal (X'01'), or binary ('01'B).

NSI ADDRESS - Represents the Next Sequential Instruction Address.

Figure 4.1 (Cont.)

Abbreviations

Sample Algorithm

The following algorithm was designed to test two examples of NOVA instructions; the first instruction is MOVOL 1,1 which will demonstrate a complex NOVA instruction; the second instruction is ADD 1,2 which will demonstrate a simple NOVA instruction.

```
1.      FETCH:   (R4)←--(NI BUFFER BEGINNING)
2.               (R5)←--(NI BUFFER ENDING)
3.               (NI BUFFER 0:15)←--(READ--→(R4,R5))
```

Figure 4.2

Sample Algorithm

```
4.                  END BRANCH OUT
5.                  (R4)←--(NI 5:7)
*           EMULATED SCAN FOR MNEMONIC BRANCH TABLE
6.                  (R4) = ('010'B)
7.                  True; BRANCH MOV-RT
8.                  (R4) = ('110'B)
9.                  True; BRANCH ADD-RT
                    .
                    .
*           EMULATED SCAN FOR CARRY BRANCH TABLE
10.         CARRY:: (R10) = ('10'B)
11.                 True; BRANCH CARRY-RT
                    .
                    .
*           EMULATED SCAN FOR SHIFT BRANCH TABLE
12.         SHIFT:: (R13)←--(NI 8:9)
13.                 (R13) = ('01'B)
14.                 True; BRANCH SHIFT-RT
                    .
                    .
15.         LOAD::  (R9)←--(NI 12)
16.                 (R9) = ('1'B)
17.                 True; BRANCH FETCH
18.                 False; BRANCH LOAD-RT
19.         OUT::   END
*           EMULATION OF MOV
20.         MOV-RT:: (IR 0:7)←--(C'LHR')
21.                 (IR 8:11)←--(C'R12')
```

Figure 4.2 (Cont.)

Sample Algorithm

```
22.                     (IR 12:15)←--(NI 1:2)
23.                     WRITE IR
24.                     (IR 0:7)←--(C'LH')
25.                     (IR 8:11)←--(C'R10')
26.                     (IR 12:15)←--(NI 10:11)
27.                     WRITE IR
28.                     (R10)←--(NI 10:11)
29.                     BRANCH CARRY
*       EMULATION OF CARRY (O FUNCTION)
30.     CARRY-RT::      (IR 0:7)←--(C'LHI')
31.                     (IR 8:11)←--(C'R15')
32.                     (IR 12:15)←--(X'01')
33.                     WRITE IR
34.                     BRANCH SHIFT
*       EMULATION OF SHIFT (L FUNCTION)
35.     SHIFT-RT::      (IR 0:7)←--(C'SLLS')
36.                     (IR 8:11)←(C'R12')
37.                     (IR 12:15)←--(X'01')
38.                     WRITE IR
39.                     (IR 0:7)←--(C'BC')
40.                     (IR 12:15)←--(C'*+4')
41.                     WRITE IR
42.                     (IR 0:7)←--(C'AHR')
43.                     (IR 8:11)←--(C'R12')
44.                     (IR 12:15)←--(C'R15')
```

Figure 4.2 (Cont.)

Sample Algorithm

```
45.                     WRITE IR
46.                     (IR 0:7)←--(C'LHI')
47.                     (IR 8:11)←(C'R15')
48.                     (IR 12:15)←--(X'00')
49.                     WRITE IR
50.                     (IR 0:7)←(C'B')
51.                     (IR 12:15)←--(C'*+3')
52.                     WRITE IR
53.                     (IR 0:7)←--(C'AHR')
54.                     (IR 8:11)←--(C'R12')
55.                     (IR 12:15)←--(C'R15')
56.                     WRITE IR
57.                     (IR 0:7)←--(C'LHI')
58.                     (IR 8:11)←--(C'R15')
59.                     (IR 12:15)←--(X'01')
60.                     WRITE IR
61.                     BRANCH LOAD
*       EMULATION OF LOAD (NI 12 EQUALS '1'B)
62.     LOAD-RT::       (IR 0:7)←--(C'LHR')
63.                     (IR 8:11)←--(NI 3:4)
64.                     (IR 12:15)←--(C'R12')
65.                     WRITE IR
66.                     BRANCH FETCH
*       EMULATION OF ADD
67.     ADD-RT::        (IR 0:7)←--(C'LHR')
```

Figure 4.2 (Cont.)

Sample Algorithm

```
68.                    (IR 8:11)←--(C'R12')
69.                    (IR 12:15)←--(NI 1:2)
70.                    WRITE IR
71.                    (IR 0:7)←--(C'LHR')
72.                    (IR 8:11)←--(C'R13')
73.                    (IR 12:15)←--(NI 3:4)
74.                    WRITE IR
75.                    (IR 0:7)←--(C'AHR')
76.                    (IR 8:11)←--(C'R12')
77.                    (IR 12:15)←--(C'R13')
78.                    WRITE IR
79.                    (IR 0:7)←--(C'CLHI')
80.                    (IR 8:11)←--(C'R12')
81.                    (IR 12:15)←--(X'FF')
82.                    WRITE IR
83.                    (IR 0:7)←--(C'BNC')
84.                    (IR 12:15)←(C'*+3')
85.                    WRITE IR
86.                    (IR 0:7)←--(C'LH')
87.                    (IR 8:11)←--(C'R10')
88.                    (IR 12:15)←--(NI 10:11)
89.                    WRITE IR
90.                    (IR 0:7)←--(C'XHR')
91.                    (IR 8:11)←--(C'R10')
92.                    (IR 12:15)←--(X'03')
```

Figure 4.2 (Cont.)

Sample Algorithm

```
93.                    WRITE IR
94.                    (R10)←--(NI 10:11)
95.                    BRANCH CARRY
```

Figure 4.2 (Cont.)

Sample Algorithm

NI BUFFER	ACTION/ CONDITION	R4	R5	R9	R10	R13	STEP #	IR OUTPUT
ß		00					1	
			15				2	
10101010								
01100000							3	
	NOT END						4	
		010					5	
	EQUAL						6	
	TRUE:							
	MOV-RT						7	
							20	IR 0:7-'LHR'
							21	IR 8:11-'R12'
							22	IR 12:15-'R1'
	WRITE						23	LHR R12,R1
							24	IR 0:7-'LH'
							25	IR 8:11-'R10'
							26	IR 12:15-TWO
	WRITE						27	LH R10,TWO
					10		28	
	CARRY						29	
	EQUAL						10	
	CARRY-RT						11	
							30	IR 0:7-'LHI'
							31	IR 8:11-'R15'
							32	IR 12:15-X'01'
	WRITE						33	LHI R15,X'01'
	SHIFT						34	
						01	12	
	EQUAL						13	

Table 4.1

NI BUFFER	ACTION/ CONDITION	R4	R5	R9	R10	R13	STEP #	IR OUTPUT
	SHIFT-RT						14	
							35	IR 0:7-'SLLS'
							36	IR 8:11-'R12'
							37	IR 12:15-X'01'
	WRITE						38	SLLS R12,X'01'
							39	IR 0:71'BC'
							40	IR 12:15-'*+4'
	WRITE						41	BC *+4
							42	IR 0:7-'AHR'
							43	IR 8:11-'R12'
							44	IR 12:15-'R15'
	WRITE						45	AHR R12,R15
							46	IR 0:7-'LHI'
							47	IR 8:11-'R15'
							48	IR 12:15-X'00'
	WRITE						49	LHI R15,X'00'
							50	IR 0:7-'B'
							51	IR 12:15-'*+3'
	WRITE						52	B *+3
							53	IR 0:7-'AHR'
							54	IR 8:11-'R12'
							55	IR 12:15-'R15'
	WRITE						56	AHR R12,R15
							57	IR 0:7-'LHI'
							58	IR 8:11-'R15'
							59	IR 12:15-X'01'

Table 4.1 (cont.)

Sample Algorithm Trace

NI BUFFER	ACTION/ CONDITION	R4	R5	R9	R10	R13	STEP #	IR OUTPUT
	WRITE						60	LHI R15,X'01'
	LOAD						61	
				00			15	
	NOT EQUAL						16	
	LOAD-RT						17	
							62	IR 0:7-'LHR'
							63	IR 8:11-'R1'
							64	IR 12:15-'R12'
	WRITE						65	LHR R1,R12
	FETCH						66	
		00					1	
			15				2	
10110110 00000000							3	
	NOT END						4	
		110					5	
	NOT EQUAL						6	
	FALSE						7	
	EQUAL						8	
	TRUE: ADD-RT						9	
							67	IR 0:7-'LHR'
							68	IR 8:11-'R12'
							69	IR 12:15-'R1'
	WRITE						70	LHR R12,R1
							71	IR 0:7-'LHR'
							72	IR 8:11-'R13'
							73	IR 12:15-'R2'

Table 4.1 (cont.)

Sample Algorithm Trace

NI BUFFER	ACTION/ CONDITION	R4	R5	R9	R10	R13	STEP #	IR OUTPUT
	WRITE						74	LHR R13,R2
							75	IR 0:7-'AHR'
							76	IR 8:11-'R12'
							77	IR 12:15-'R13'
	WRITE						78	AHR R12,R13
							79	IR 0:7-'CLHI'
							80	IR 8:11-'R12'
							81	IR 12:15-'X'FF'
	WRITE						82	CLHI R12,X'FF'
							83	IR 0:7-'BNC'
							84	IR 12:15-'*+3'
	WRITE						85	BNC *+3
							86	IR 0:7-'LH'
							87	IR 8:11-'R10'
							88	IR 12:15-ZERO
	WRITE						89	LH R10,ZERO
							90	IR 0:7-'XHR'
							91	IR 8:11-'R10'
							92	IR 12:15-X'03'
	WRITE						93	XHR R10,X'03'
					00		94	
	CARRY						95	
	NOT EQUAL						10	
	FALSE						11	
						00	12	
	NOT EQUAL						13	

Table 4.1 (cont.)

Sample Algorithm Trace

NI BUFFER	ACTION/ CONDITION	R4	R5	R9	R10	R13	STEP #	IR OUTPUT
	FALSE						14	
				00			15	
	NOT EQUAL						16	
	FALSE						17	
	LOAD-RT						18	
							62	IR 0:7-'LHR'
							63	IR 8:11-'R2'
							64	IR 12:15-'R12'
	WRITE						65	LHR R2,R12
	FETCH						66	
		00					1	
			15				2	
------							3	
	END OUT						4	
	END						19	

Table 4.1 (Cont.)

Sample Algorithm Trace

Algorithm Output

The next step in verification was to examine the output of the above algorithm and trace its execution to insure correctness. For ease of the reader, I have condensed the output into figure 4.3 (Condensed Listing) and have provided a trace of its execution in Table 4.2 below.

```
 1    LHR   R12,R1          PUT ACS IN R12
 2    LH    R10,TWO         PUT '10'B IN R10
 3    LHI   R15,X'01'       PUT '01'B IN R15
 4    SLLS  R12,X'01'       SHIFT R12 LEFT 1 BIT
 5    BC    *+4             IS CARRY EQUAL TO 1?
 6    AHR   R12,R15         NO, ADD R15 TO R12
 7    LHI   R15,X'00'       PUT '00'B IN R15
 8    B     *+3             GO TO NEXT EMULATED INSTRUCTION
 9    AHR   R12,R15         YES, ADD R15 TO R12
10    LHI   R15,X'01'       PUT '01'B IN R15
11    LHR   R1,R12          PUT R12 INTO R1
12    LHR   R12,R1          PUT R1 INTO R12
13    LHR   R13,R2          PUT R2 INTO R13
14    AHR   R12,R13         ADD R13 INTO R12
15    CLHI  R12,X'FF'       GREATER/EQUAL 2E16?
16    BNC   *+3             NO,
17    LH    R10,ZERO        YES, GET CARRY FUNCTION
18    XHR   R10,X'03'       COMPLEMENT IT
19    LHR   R2,R12          PUT R12 INTO R2
```

Figure 4.3

Condensed Listing

STEP #	R1	R2	R10	R12	R13	R15	ACTION/ CONDITION	PSW MASK
0	00	00	00	00	00	00	----	0000
1	00	00		00				
2	00	00		00		01		
3	00	00		00		01		
4								0000
5							NO	
6	00	00		01		01		
7						00		
8							JMP +3	
11	01	00		01		00		
12	01	00		01		00		
13	01	00		00		00		
14	01	00		01		00		
15							NOT GE	
16	01	00		01		00	JMP +3	
19	01	01		01		00		

Table 4.2

Flow of Generated Code

Summary

Since few algorithms, if any, are ever accepted on an unproven basis, I have presented a sample of the proposed algorithm for this emulator and

have presented a step-by-step trace table of the logic based upon the input of two instructions, the MOVOL 1,1 which is a sample of a multi-stepped instruction, and the ADD 1,2 which is an example of a single stepped instruction. The output of these instructions was presented so that the viewer might have a more concise view of the code generated from the two input instructions, and then a trace table was provided for the generated code, in order that it might be verified as correct.

Chapter 5

CONCLUSION

In this report I presented the design of an INTERDATA implementation of the U. S. Navy MINI-COBOL compiler system. The problem of portability with regards to the present system and the alternatives of a functional rewrite of the assembler routines to effect portability versus an emulation of the instruction set of the NOVA in INTERDATA micro-code was presented. The advantages of each of the alternatives were discussed and I decided that the emulation method, while not specifically designed for machine efficiency, would provide for a system which is entirely transparent to the user and would facilitate ease of operation, being in essence, no different than the job submission scheme for most modern computers.

A presentation of the general flow of the proposed system was given wherein I discussed the philosophy of the actual design to be used at a high level, feeling that the emulation overview would aid the viewer in the following sections which presented the emulator at a more particular level.

At the detailed design presentation of the emulator, I discussed the input (the NOVA instruction formats to be considered) in detail. I then presented each NOVA function, giving an INTERDATA-like assembler language algorithm for the emulation of the function discussed. I also presented an algorithm for the emulation of the Fetch and Decoder modules which would effect the emulation in the proposed system.

After having given the detailed specifications of the Fetch and Decoder modules, the input to be utilized in the emulator, and the algo-

rithms for the actual emulation, I presented a sample verification of the design by providing a step-by-step emulation of two NOVA instructions. I then provided a step-by-step trace table of the execution of the output of the emulation to determine its soundness.

In conclusion, then, I believe the results of the report indicate that the overall design of the proposed emulator is correct and desirable; correct because all instructions were verified in function and results and included in the report was a sample of this verification technique, and desirable because the extended capabilities of the mini-computers which would be presented with COBOL and of the "general" design which would allow for expansion in the areas of virtual memory systems and mini-computer I/O device handling systems.

BIBLIOGRAPHY

1. Smith, Douglas. "HIMICS: A Virtual Memory Environment for Mini-computers and a Description of its Level 1 Processor." Masters Report, Kansas State University, (1975).

2. Bentz, Harlan. "HIMICS: A Virtual Memory Environment for Mini-computers and a Description of its Level 2 Processor." Masters Report, Kansas State University, (1975).

3. Data General Corporation. "How to Use the NOVA and the SUPERNOVA," Southboro, Massachusetts: Data General Corporation, (1969).

4. U. S. Navy. "Navy Cobol Compiling System for Mini-computers," Navy Programming Languages Section, OP-911F, Washington, DC, Unpublished Draft.

5. U. S. Navy. "MINI-COBOL Programmers Reference," Navy Programming Languages Section, OP-911F, Washington, DC, (1974), Unpublished Draft.

6. Interdata. "Model 80 Micro-instruction Reference Manual," Interdata Publication Number 29-282R01, Oceanport, New Jersey, Interdata Corporation (1973).

7. Interdata. "Interdata Model 80 Micro-program 05-04R07A13," Interdata Corporation (1973).

APPENDIX A

MINI-COBOL LANGUAGE SPECIFICATIONS

OVERVIEW OF U. S. NAVY COBOL COMPILING SYSTEM FOR MINI-COMPUTERS

Introduction

This is an overview of the design of the U. S. Navy MINI-COBOL compiler. The language does not include, as you will see, the full ANS COBOL language set, but rather a limited subset known as MINI-COBOL.

This system was developed because of the increasing use of mini-computers within the Navy and the lack of program portability these systems caused. While MINI-COBOL is not completely portable, any program written in this subset, with minor modifications, will run on any machine with an equal or higher level compiler.[4]

Overview

The Navy compiling system is essentially a two phase compiler. The first phase is subdivided into two passes, with the first pass examining the various MINI-COBOL verbs and building a tuple and the second pass resolving the references and completing the tuple. The second phase is subdivided into two separate functions. The first function will read tuples and allocate memory to declared variables, establish a constant/literal pool, and establish the various editing masks used in the application program in memory. The second function will read the tuples, examine them and then will execute them as required.

Phase I

Phase I of the U. S. Navy MINI-COBOL compiler is that portion of the system which is written in COBOL. All of the machine independent portions of the compiler are found in these programs. The inputs to

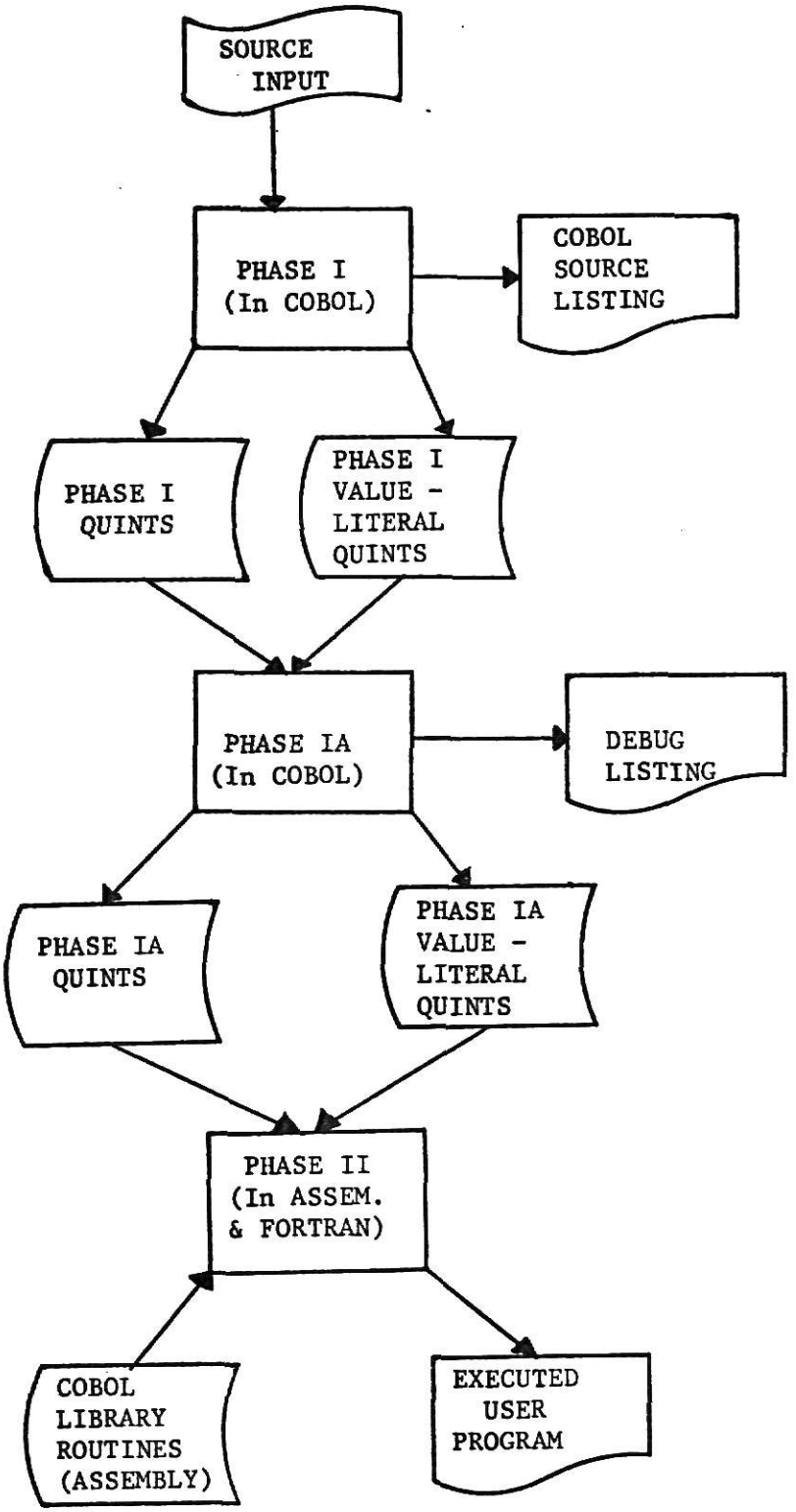


Figure A.1

Overview Flowchart

Phase I is the user's MINI-COBOL source file and the output of Phase I is the user's program as translated into tuples.

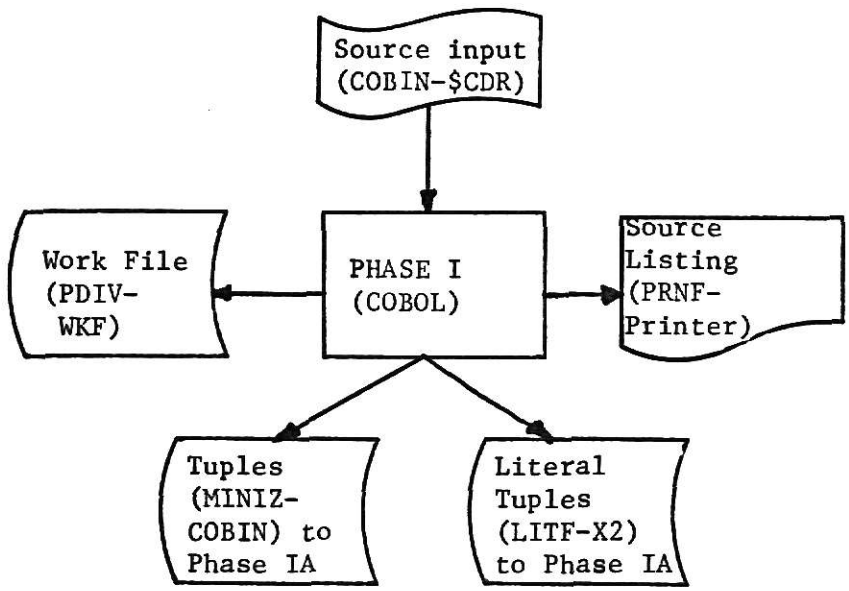


Figure A.2

Phase I File Flow

Phase I Tuples

As was previously mentioned, the input into the system is the user's source input and the output a series of tuples. The following text is a description of those tuples as generated by Phase I.

Identification Division word.

The output for the IDENTIFICATION DIVISION from the MINI-COBOL compiler consists solely of the PROGRAM-ID; a five character name which will start in position number one of the output.

POSITION	USE
1 - 5	PROGRAM-ID

Figure A.3

ID Tuple

Environment Division word.

The ENVIRONMENT DIVISION word represents the SELECT statements and has the following format.

POSITION	USE
1 - 3	Always Zero
4 - 8	Reserved
9 - 13	File-name
14 - 18	Device-name
19 - 23	Device file-name

Figure A.4

Environment Division Tuple

Data Division word.

The DATA DIVISION word has two different functions. It is used to represent the FD's (file descriptions) and it is used to represent all data items which have been declared, including all particular characteristics of the declared data-items. The DATA DIVISION word has the following format.

POSITION	USE
1 - 5	Data-name/file-name
6 - 10	Displacement from top-of-memory
11	Data type (X or 9)
12 - 15	Length of item
16 - 17	Decimal point location
18	BLANK ZERO flag
19	SIGN flag
20	USAGE flag
21	SYNC flag
22	OCCURS
23 - 24	Edit table reference
25 - 27	Value table reference

Figure A.5

Data Division Tuple

At the FD level, the first twenty-seven character word for that file is started by translating statements that follow, when present, into the total size of the file plus the number of records. The file-name is placed in the first five positions. If the BLOCK and the RECORD clauses are present, they are multiplied and the result will be placed in the twenty-seven character word as this is the length of the physical record which will be read. Should the BLOCK clause be omitted, then one record per block is assumed. Should the RECORD clause be omitted, a default of eighty characters per record will be assumed.

The record-name of the 01 level is placed in the data-name field. The top-of-memory address is passed along from the FD level (if in the FILE SECTION). If the record-name encountered is not within the FILE SECTION, then an augmented address is started for the record, i. e., an address relative to top-of-memory will be maintained on a sequential basis for each structure.

Processing continues through all the elementary data-items. The data-name for each elementary item is placed in the data-name field. The starting position (within the area of storage) is computed and placed in the displacement field. The length and data type, either X or 9, are placed into their respective fields, if present. One twenty-seven character word will be generated for each elementary data-item.

Procedure Division word.

For each PROCEDURE DIVISION entry, a twenty-three character word is generated. The PROCEDURE DIVISION word has the following format.

POSITION	USE
1 - 4	Sequence number
5	Always zero
6 - 8	Routine number
(6 - 7)	Verb
(8)	Variant of verb
9 - 13	First data-name
14 - 18	Second data-name
19 - 23	Third data-name

Figure A.6

Procedure Division Tuple

Each verb in the MINI-COBOL system has been assigned a unique verb code (see Appendix B) which is placed in position six and seven of the PROCEDURE DIVISION word. Since there are in some cases more than one variation of the verb, the system uses position eight to further delineate the operation requested.

The data-names are placed into the appropriate fields as they are encountered on the source statements. The Source sequence number is the number of the MINI-COBOL source statement from which the PROCEDURE DIVISION word is derived.

Phase IA

Although Phase IA is a separate step at this time, it is anticipated by the Navy Programming Languages Section, that this process will be incorporated into Phase I in the very near future providing a single program two step compiler.

This portion of the compiler accepts as input the tuples as generated by Phase I. These tuples are those of the PROCEDURE DIVISION statements and those generated which represent the literals/values of the DATA DIVISION and PROCEDURE DIVISION. The tuples will be manipulated and will be used to build a series of Data Tables and Procedure tuples which can be executed by the Phase II portion of the compiler.

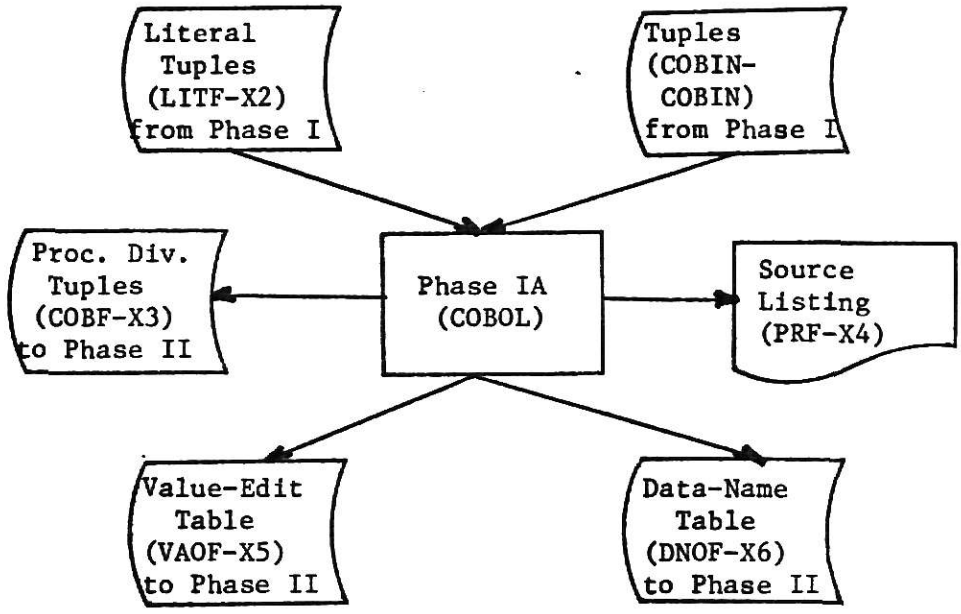


Figure A.7

Phase IA File Flow

Phase IA Tuples

As was previously mentioned, Phase IA produces a series of updated tuples and several tables the format of which is presented below.

I/O table.

X(5) - Device Name (ASCII)

X(5) - Channel Number

Figure A.8

I/O Table

The I/O table is utilized to link COBOL files to specific I/O channels. The Channel Number will be filled in by the Interpreter (Phase II) at run time.

File descriptor table.

9(5) - Raw Data Byte Pointer

9(4) - Data-name Table Entry Pointer

9(2) - I/O Table Pointer

9(4) - Total Length (Bytes)

9 - Label

9(3) - File Access Method*

9(4) - Keylen*

Figure A.9

File Descriptor Table

The file descriptor table link COBOL file name to associated I/O channels and record descriptions.

Data-name table.

9(5) - Raw Data Byte Pointer

9 - Sign Flag

9 - Data Type

9(4) - Data Length

9 - Occurs Flag

9 - Sync Flag

9 - Blank Zero Flag

9(3) - Value Table Pointer

9(3) - Data Name Edit Pointer

9(2) - Decimal Point Location

Figure A.10

Data-name Table

* File access method and keylen have not been implemented but are seen as additions for the very near future and space was reserved for them.

The Data-name table describes all data-names used in the specific COBOL program and provides all necessary information needed to utilize that data-name.

Data type may contain various values which are as listed below:

1) X - signifies the data type is alphanumeric and will be translated into the digit "2";

2) 9 - signifies numeric data and will be translated into the digit "3".

Value/literal table.

X(80) - Actual Value

Figure A.11

Value/Literal Table

The Value/Literal table (also called Memory Layout Map) sets up the initial values in memory that have been specified by the user in his COBOL program.

Edit table.

X(30) - String of ASCII Characters

Figure A.12

Edit Table

The Edit Table sets up masks to be used in editing as the user has requested in his COBOL program.

Procedure Division quints.

9(5) - Quint Sequence Number

9(3) - Operation Code

9(4) - Operand one

9(4) - Operand two

9(4) - Operand three

Figure A.13

Procedure Division Quints

The Procedure Division Quintuples are basically the same as those generated by Phase I with data-names replaced and sequence numbers added.

Phase II - Interpreter

Phase II of the MINI-COBOL compiler is written in NOVA FORTRAN and is stored in the machine in executable binary form. There are two distinct operations performed by Phase II, the first being DATA DIVISION memory allocation, and the second being COBOL quintuple examination and execution. Figure A.14 below presents the general file flow of the Interpreter.

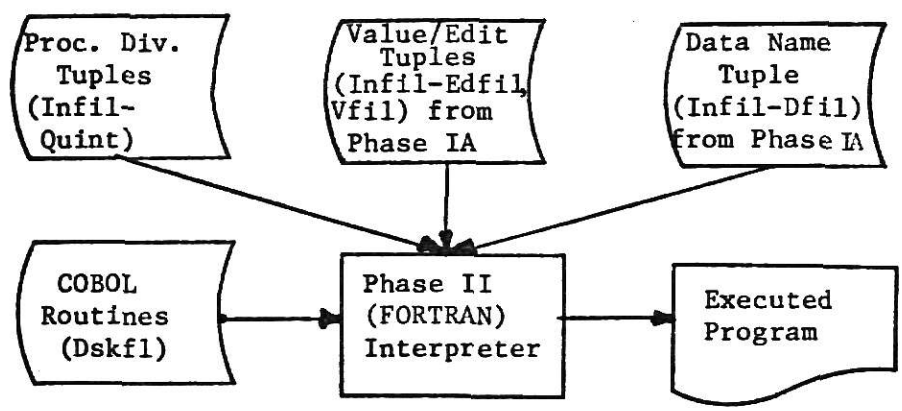


Figure A.14

Phase II File Flow

Upon execution of Phase II, the first part of the Interpreter allocates the COBOL program DATA DIVISION to memory and proceeds to map, into the proper memory locations, any initial value or literals which have been specified by the programmer. Upon completion of this task, the second part of the Interpreter begins to examine each tuples' opcode and will then pass control based on the operation code to a dispatch table. Depending upon where entry occurred in the dispatch table, control is again passed to a subroutine dispatch table. The second dispatch table contains a list of "Calls" to NOVA assembler routines in the MINI-COBOL library which are in the correct order required to process the particular operation code with its options. Upon completion of the list, control is then passed back to the Interpreter which will read the next tuple, thereby closing the loop. (see Figure A.15).

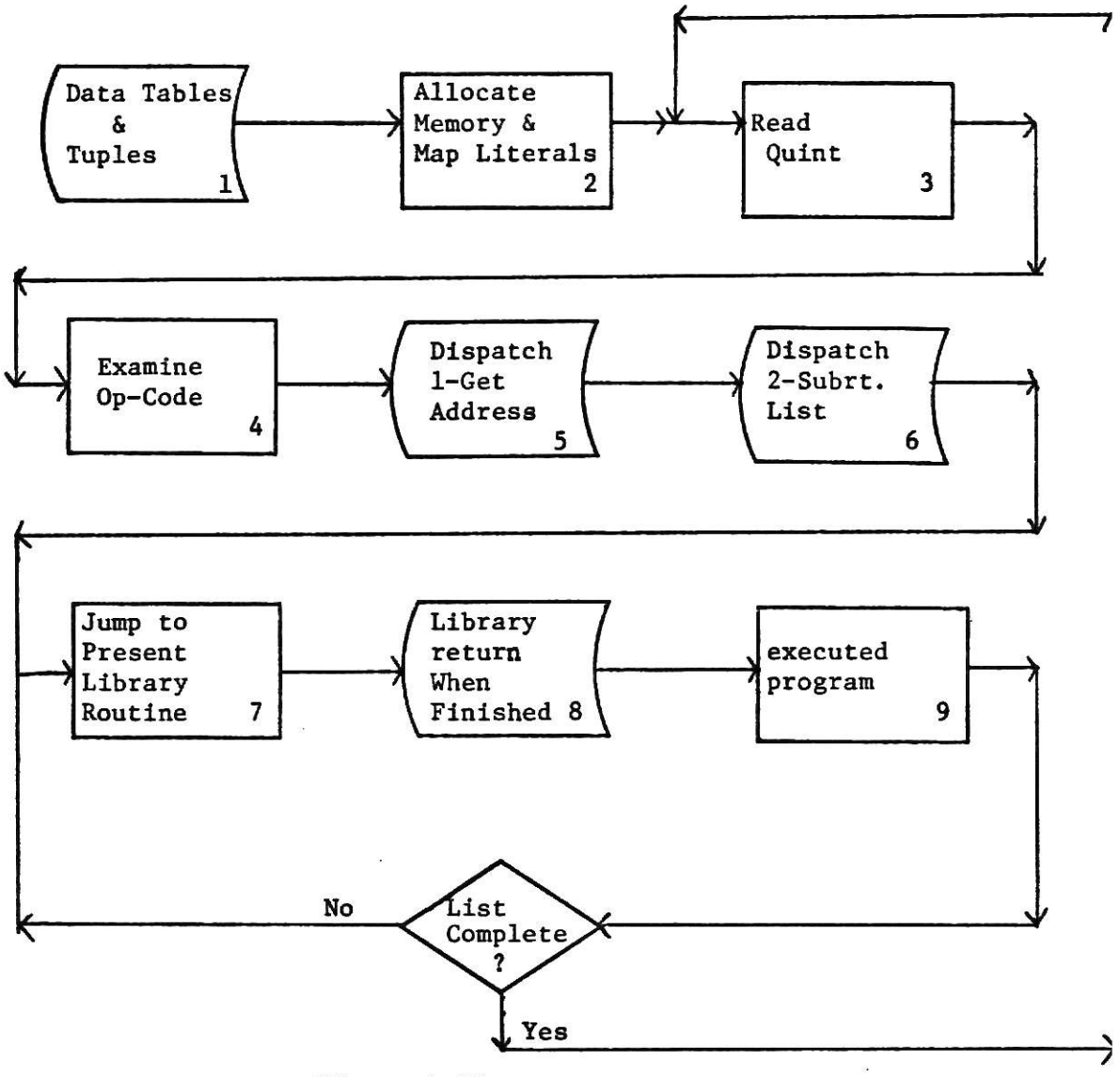


Figure A.15

Interpreter Processing Flow

Execution Summary

The first MINI-COBOL statement of importance to the compiler is the PROGRAM-ID source statement. Upon its encounter, the compiler will produce a Control card with the digit "1" in the first position. The Control Card will then be written out to the file MINIZ. Control Cards are used in Phase IA to identify the type of tuple which immediately follows. The compiler will then construct the Identification Division word as given in figure A.3 and write it to the file MINIZ.

The next source statement of significance to the compiler is the ENVIRONMENT DIVISION statement. The compiler will produce a Control Card with the digit "2" in the first position and write it out to the file MINIZ. The compiler will then construct the Environment Division word as described in figure A.4 and write it out to the file MINIZ.

When the source statement DATA DIVISION is encountered by the compiler, it will scan for the keyword FILE SECTION. When FILE SECTION is encountered it will produce a Control Card with the digit "3" in the first position and write it out to the file MINIZ. Data Division words will then be constructed as per figure A.5. When the keyword WORKING-STORAGE SECTION is encountered a Control Card with the digit "5" will be written out to the file MINIZ. Data division words will again be produced as per figure A.5 and written out to the file MINIZ. Upon completion of the construction of data division words, the compiler will produce a Control Card with the digit "5" in the first position and write it out to the file MINIZ. The edit tables and literal/value tables constructed in the program will be written out to MINIZ and the file LITF, respectively, and the compiler will then begin construction of the Procedure Division words as per figure A.6. Upon completion of construction of the last Procedure Division word, the compiler will produce a Control Card with the digits "9000" in the first four positions (sequence field) and the last sequence number will be entered into the third operand field. The compiler will then have completed Phase I after having written the above record to the file MINIZ.

Phase IA will then be invoked. The files MINIZ and LITF will be revised into a slightly different format to produce tables as per figure A.8 through A.13. When processing has been completed, Phase II will be invoked.

Phase II uses as input the save file created from Phase IA. This phase will read the save file, allocate memory and then read, examine and execute the various tuples as generated by Phase I and Phase IA.

APPENDIX B

MINI-COBOL VERBS

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  program-name.
AUTHOR.  comment-entry.
INSTALLATION.  comment-entry.
DATE-WRITTEN.  comment-entry.
SECURITY.  comment-entry.


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  computer-name-model.
DEBUGGING MODE.
OBJECT-COMPUTER.  computer-name-model.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

     SELECT file-name ASSIGN implementor-name-1.
       .
       .
       .

DATA DIVISION.
FILE SECTION.
FD  file-name
    BLOCK integer-1 RECORDS
    RECORD integer-2

    LABEL RECORD  STANDARD
                  OMITTED

    record-description . . .


WORKING-STORAGE SECTION.

     77 - Level-description-entries
     record-description-entries


FORMAT:


level-number   data-name-1
               FILLER


REDEFINES data-name-2

PIC character-string
```

Figure B.1

Language Skeleton

```
USAGE    COMP
         DISPLAY

SIGN     LEADING      SEPARATE
         TRAILING

OCCURS integer-1

SYNC     LEFT
         RIGHT

BLANK ZERO

VALUE literal-1


PROCEDURE DIVISION

paragraph-name.
     sentence .......
     .
     .
     .
```

Figure B.1 (Cont.)

Language Skeleton

```
ACCEPT  identifier-1

ADD     identifier-1   TO  identifier-2
        literal-1

ADD     identifier-1   identifier-2   TO
        literal-1      literal-2      GIVING   identifier-3

CLOSE   file-name-1   file-name-2   file-name-3

DISPLAY identifier-1   identifier-2
        literal-1      literal-2

DIVIDE  identifier-1   INTO  identifier-2
        literal-1

DIVIDE  identifier-1   INTO  identifier-2   GIVING  identifier-3
        literal-1            literal-2

ENTER   language-name   routine-name

EXIT

GO TO   procedure-name-1

GO TO   procedure-name-1   procedure-name-2....

        procedure-name-10   DEPENDING   identifier-1

IF  condition

MOVE    identifier-1   TO  identifier-2
        literal-1

MULTIPLY    identifier-1   BY  identifier-2
            literal-1

MULTIPLY    identifier-1   BY  identifier-2   GIVING  identifier-3
            literal-1          literal-2

            INPUT
OPEN        OUTPUT   file-name-1   file-name-2   file-name-3
            I-O
```

Table B.1

MINI-COBOL Verb Formats

PERFORM procedure-name-1 THRU procedure-name-2

PERFORM procedure-name-1 THRU procedure-name-2

identifier-1
literal-1 TIMES

READ file-name-1 END GO TO procedure-name-1

REWRITE record-name-1

STOP RUN
literal-1

SUBTRACT identifier-1
literal-1 FROM identifier-2

SUBTRACT identifier-1
literal-1 FROM identifier-2
literal-2 GIVING identifier-3

WRITE record-name BEFORE
AFTER integer-1 LINES
PAGE

All arithmetic verbs have the following optional clauses:

ROUNDED (only on the resultant identifier)

SIZE ERROR GO TO procedure-name-1

Table B.1 (Cont.)

MINI-COBOL Verb Formats

```
identifier-1      NOT GREATER THAN      identifier-2
literal-1         NOT LESS THAN         literal-2
                  NOT EQUAL TO
```

Figure B.2

Relational Conditions

```
identifier  NOT  NUMERIC
                 ALPHABETIC
```

Figure B.3

Class Conditions

```
data-name  (subscript)
```

Figure B.4

Subscripting/Identifiers

CARD COLUMN	USE
1 to 6	Sequence Number
7	Comment and DEBUG entries
8 to 12	Paragraph-name
12 to 23	Verb
24 to 29	Field 1
30 to 35	Field 2
36 to 41	Field 3
42 to 47	Field 4
48 to 53	Field 5
54 to 59	Field 6
60 to 65	Field 7
66 to 71	Field 8
72	Period
73 to 80	Identification

Table B.2

Coding Sheet

Because the U. S. Navy MINI-COBOL compiler expects its input to be entered in a fixed format, the coding sheet will be different than the usual coding sheet for COBOL. The above is a breakdown of the coding sheet used by this compiler.[5]

QUINT POSITIONS 6-8	VERB AND OPTION REQUESTED
010	ACCEPT ___
020	ADD ___ ___ GIVING ___
260	ADD ___ ___ TO ___
030	CLOSE ___
040	DISPLAY ___
050	DIVIDE ___ INTO ___ GIVING ___
060	ENTER FORTRAN
061	ENTER ASSEMBLY
062	ENTER FORTRAN ___
063	ENTER ASSEMBLY ___
070	GO TO ___*
080	WRITE ___ BEFORE ___
081	WRITE ___ AFTER ___
082	WRITE ___ BEFORE PAGE
083	WRITE ___ AFTER PAGE
090	MOVE ___ TO ___
100	MULTIPLY ___ BY ___ GIVING ___
110	OPEN INPUT ___
111	OPEN I-O ___
112	OPEN OUTPUT ___
120	PERFORM ___ THRU ___ ___ (TIMES)
130	READ ___ END GO TO ___
140	REWRITE ___
150	STOP RUN
280	STOP
160	SUBTRACT ___ FROM ___ GIVING ___
270	SUBTRACT ___ ___ FROM ___
170	EXIT
180	ROUNDED OPTION
190	SIZE ERROR OPTION
200	IF ___ EQUAL ___
201	IF ___ NOT EQUAL ___
210	IF ___ LESS ___
211	IF ___ NOT LESS ___
220	IF ___ GREATER ___
221	IF ___ NOT GREATER ___
230	IF ___ NUMERIC
231	IF ___ NOT NUMERIC
240	IF ___ ALPHABETIC
241	IF ___ NOT ALPHABETIC
250	DENOTES SUBSCRIPTS

* GO TO ... DEPENDING IS TREATED AS A SERIES OF IF STATEMENTS.

Figure B.5

Verb Code

APPENDIX C

MINI-COBOL SOURCE LISTING

```
        .TITL   ACCEPT
        .ENT    APT1
        .EXTD   PUT
        .NREL
APT1:   ISZ     RTST,2  ;GET NEXT RETURN POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,OP1DP,2
        STA     1,OPDP,2
        LDA     1,OP1DL,2
        STA     1,OPDL,2
        LDA     0,QMRK,2        ;LOAD QUESTION MARK IN AC0
        .SYSTEM                 ;WRITE QUESTION MARK OUT TO
        .PCHAR                  ;THE TELETYPE
        JSR     @RTER,3
        MOV     3,2
        LDA     0,BELL,2        ;LOAD BELL IN AC0
        .SYSTEM                 ;RING THE BELL ON THE TELETYPE
        .PCHAR
        JSR     @RTER,3
FETCH:  .SYSTEM                 ;GET A CHARACTER FROM THE
        .GCHAR                  ;TELETYPE
        JSR     @RTER,3
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        LDA     3,C1,2          ;LOAD CARRIAGE RETURN
        SUB#    3,0,SNR         ;COMPARE
        JMP     FNISH           ;EQUAL, JMP TO FINIS
        MOV     0,1
        LDA     3,OPDP,2    ;LOAD AC2 WITH BYTE POINTER
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT            ;PERFORM THE STORE ROUTINE
        JSR     @RTEP,2
        ISZ     OPDP,2          ;INCREMENT BYTE POINTER
        DSZ     OPDL,2          ;DECREMENT LENGTH
        JMP     FETCH           ;JUMP TO GET NEXT CHARACTER
        LDA     0,C1,2
FNISH:  .SYSTEM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        LDA     0,LF,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        JMP     @RTN,2
```

```
        .TITL   ADD
        .ENT    ADDIT,ADDPK
        .NREL
;THE CALL TO THIS ROUTINE IS (N,PAC1,PAC2,PAC3)
ADDPK:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;STORE RETURN ADDRESS
        LDA     1,PACK1,2       ;INITIALIZE ADDRESS
        STA     1,PAC1,2        ;FOR ADDPACK
        LDA     1,PACK2,2       ;ROUTINE
        STA     1,PAC2,2
        LDA     1,PACK3,2
        STA     1,PAC3,2
        LDA     1,TEN,2 ;GET AN EIGHT FOR EIGHT WORD ADD
        STA     1,N,2   ;STORE IN N
        JMP     .+3
ADDIT:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;STORE RETURN ADDRESS
        LDA     3,N,2           ;GET NUMBER OF WORDS TO BE ADDED
        STA     3,I,2           ;STORE IN I
        LDA     1,PAC1,2        ;GET FIRST FIELD TO BE ADDED
        ADD     3,1             ;GET FIRST WORD
        STA     1,PAC1,2        ;STORE BACKOUT
        LDA     1,PAC2,2        ;DO THE SAME FOR PAC2
        ADD     3,1
        STA     1,PAC2,2
        LDA     1,PAC3,2        ;DO THE SAME FOR PAC3
        ADD     3,1
        STA     1,PAC3,2
        DSZ     PAC1,2          ;TO GET PROPER ALIGNMENT
        DSZ     PAC2,2          ;FOR ADDITION AND
        DSZ     PAC3,2          ;FOR STORING
        LDA     1,@PAC1,2       ;GET FIRST ARGUMENT
        LDA     0,@PAC2,2       ;GET SECOND ARGUMENT
        ADDZ    1,0     ;ADD TOGETHER CHECK FOR
        STA     0,@PAC3,2
LP:     DSZ     I,2
        JMP     .+2
        JSR     @RTN,2
        DSZ     PAC1,2
        DSZ     PAC2,2
        DSZ     PAC3,2
        SUBCL   0,0
        LDA     1,@PAC1,2
        ADDZ    1,0
        LDA     1,@PAC2,2
        ADD     1,0
        STA     0,@PAC3,2
        JMP     LP
```

```
        .TITL   ADD
        .ENT    ADDIT,ADDPK
        .NREL
;THE CALL TO THIS ROUTINE IS (N,PAC1,PAC2,PAC3)
ADDPK:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;STORE RETURN ADDRESS
        LDA     1,PACK1,2       ;INITILIZE ADDRESS
        STA     1,PAC1,2        ;FOR ADDPACK
        LDA     1,PACK2,2       ;ROUTINE
        STA     1,PAC2,2
        LDA     1,PACK3,2
        STA     1,PAC3,2
        LDA     1,TEN,2 ;GET AN EIGHT FOR EIGHT WORD ADD
        STA     1,N,2   ;STORE IN N
        JMP     .+3
ADDIT:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;STORE RETURN ADDRESS
        LDA     3,N,2           ;GET NUMBER OF WORDS TO BE ADDED
        STA     3,I,2           ;STORE IN I
        LDA     1,PAC1,2        ;GET FIRST FIELD TO BE ADDED
        ADD     3,1             ;GET FIRST WORD
        STA     1,PAC1,2        ;STORE BACKOUT
        LDA     1,PAC2,2        ;DO THE SAME FOR PAC2
        ADD     3,1
        STA     1,PAC2,2
        LDA     1,PAC3,2        ;DO THE SAME FOR PAC3
        ADD     3,1
        STA     1,PAC3,2
        DSZ     PAC1,2          ;TO GET PROPER ALIGNMENT
        DSZ     PAC2,2          ;FOR ADDITION AND
        DSZ     PAC3,2          ;FOR STORING
        LDA     1,@PAC1,2       ;GET FIRST ARGUMENT
        LDA     0,@PAC2,2       ;GET SECOND ARGUMENT
        ADDZ    1,0       ;ADD TOGETHER CHECK FOR
        STA     0,@PAC3,2
LP:     DSZ     I,2
        JMP     .+2
        JSR     @RTN,2
        DSZ     PAC1,2
        DSZ     PAC2,2
        DSZ     PAC3,2
        SUBCL   0,0
        LDA     1,@PAC1,2
        ADDZ    1,0
        LDA     1,@PAC2,2
        ADD     1,0
        STA     0,@PAC3,2
        JMP     LP
```

```
;
;       READ TABLE PTRS AND ALLOCATE CORE
;
        .TITLE  ALCOR
        .NREL
        .TXTM   1
        .ENT    ALCOR
        .EXTN   DSKFL
;
ALCOR:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,NEWF,2
        MOV     0,0,SNR
        JMP     ALC1
        SUBZ    0,0
        STA     0,IOCNT,2
        STA     0,FDCNT,2
        STA     0,EDCNT,2
        STA     0,DNCNT,2
        STA     0,RSCNT,2
        STA     0,QNCNT,2
        JMP     ALC2
ALC1:   LDA     3,IBUFP
        STA     3,IBUF,2
        SUBZL   0,0
        STA     0,IFCN,3        ;READ
        STA     1,SVFCN,2
        STA     1,IO,3
        STA     0,STWD,3
        LDA     1,I11
        STA     1,ICT,3
        LDA     0,IOPTR,2
        STA     0,IVEC,3
        ADC     0,0
        STA     0,-10,3         ;SET PSECTR TO MINUS 1
        JSR     @DSKFP          ;READ PTRS
        SUBZ    0,0
ALC2:   LDA     1,IOCNT,2
        ADD     1,0
        LDA     1,FDCNT,2
        ADD     1,0
        LDA     1,EDCNT,2
        ADD     1,0
        LDA     1,DNCNT,2
        ADD     1,0
        LDA     1,RSCNT,2
        ADD     1,0
        LDA     1,QNCNT,2
        ADD     1,0
        LDA     1,TADD
        ADD     1,0
        STA     0,TEMP2,2       ;TOTAL # WORDS IN SAVE FILE
        STA     0,TEMP,2
        .SYSTM
        .MEM
        JMP     .+1
        MOV     3,2
        LDA     3,SNMAX,2
        MOV     3,3,SZR
        JMP     .+3
        STA     1,SNMAX,2
```

```
        JMP     ALC3
        SUB     3,1
        LDA     3,TEMP,2
        SUB     1,3
        STA     3,TEMP2,2
        LDA     1,SNMAX,2
ALC3:   SUB     1,0             ;AMOUNT OF CORE AVAILABLE
        INC     1,1
        STA     1,IOST,2        ;OLD NMAX IS START OF IOFILE
        LDA     1,TEMP,2
        SUBZ    1,0,SNC
        JSR     @RTER,2         ;SAVE FILE EXCEEDS CORE
        LDA     0,IOST,2
        LDA     1,IOCNT,2
        ADD     1,0
        LDA     1,IOAD
        ADD     1,0
        STA     0,FDST,2        ;START OF FDFILE
        LDA     1,FDCNT,2
        ADD     1,0
        LDA     1,FDAD
        ADD     1,0
        STA     0,EDST,2
        LDA     1,EDCNT,2
        ADD     1,0
        STA     0,DNST,2
        LDA     1,DNCNT,2
        ADD     1,0
        LDA     1,DNAD
        ADD     1,0
        STA     0,RSST,2
        LDA     1,RSCNT,2
        ADD     1,0
        STA     1,RSMX,2
        LDA     1,RSAD
        ADD     1,0
        STA     0,ONST,2
        LDA     0,TEMP2,2
        .SYSTM
        .MEMI
        JSR     @RTER,3
        MOV     3,2
        SUB     0,0
        LDA     3,IOST,2
ALC4:   STA     0,0,3   ;ZERO STORAGE
        INC     3,3
        DSZ     TEMP,2  ;COMPLETED?
        JMP     ALC4    ;NO
        JSR     @RTN,2
I11:    11.
TADD:    622.
IOAD:   12.
FDAD:   10.
DNAD:   50.
ONAD:   50.
RSAD:   500.
DSKFP:  DSKFL
IBUFP:  IBUFG
        .BLK    22
IBUFG:  .BLK    256.
```

```
;
;       THESE ROUTINES SET AND DELETE 20 BREAKPOINTS
;
        .TITLE  BPEAK
        .NREL
        .TXTM   1
        .ENT    BREAK,DELET,BTL
BREAK:  ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @RDAEP,2        ;READ CHAR
        JSR     @RTER,2
        JMP     .+2             ;NOT CARR RET
        JMP     BR4             ;CARR RET
        JSR     @NMNCP,2        ;READ # TO SET
        JMP     .+2
        JMP     .+3
BR1:    ISZ     CERFL,2         ;ILL COMMAND
        JMP     @RTER,2
        MOV     1,1,SNR
        JSR     BR1             ; =0
        LDA     3,BTBL
        LDA     0,BTL
        STA     0,TEMP,2
BR2:    LDA     0,0,3
        MOV     0,0,SNR         ;BREAK ALREADY SET?
        JMP     BR3             ;NO
        INC     3,3
        DSZ     TEMP,2
        JMP     BR2
        JSR     BR1
BR3:    STA     1,0,3           ;STORE QUINT #
        JSR     @RTN,2
BR4:    LDA     0,BTL           ;LIST ALL CURRENT BREAKS
        STA     0,X3T,2         ;BREAK TABLE LENGTH
        LDA     0,BTBL
        STA     0,ARGDE,2
BR5:    LDA     0,@ARGDE,2      ;QUINT #
        MOV     0,0,SNR         ;OCCUPIED?
        JMP     BR6             ;NO
        JSR     @PROCP,2        ;PRINT QUINT #
        JSR     @RTER,2
        JSR     @PRCRP,2
        JSR     @RTER,2
BR6:    ISZ     ARGDE,2
        DSZ     X3T,2
        JMP     BR5             ;NEXT
        JSR     @RTN,2          ;DONE
DELET:  ISZ     RTST,2          ;DELETE BREAKS
        STA     3,@RTST,2
        JSR     @RDAEP,2        ;READ CHAR
        JSR     @RTER,2
        JMP     .+2             ;NOT CARR RET
        JMP     DE1             ;CARR RETURN
        JSR     @NMNCP,2        ;READ # TO DELETE
        JSR     BR1             ;ILLEGAL COMMAND
        LDA     3,BTBL
        LDA     0,BTL
        STA     0,X17T,2
DE0:    LDA     0,0,3
        SUB     1,0,SNR         ;THIS BREAK ?
        JMP     DE05            ;YES
```

```
        INC     3,3
        DSZ     X17T,2          ;NO, LAST BREAK ?
        JMP     DE0             ;NO
        LDA     0,QST           ;NO BREAK
         SYSTM
        .PCHAR                  ;?
        JSR     @RTER,2
        JSR     @PRCRP,2        ;CARR RET
        JSR     @RTER,2
        JSR     @RTN,2
DE05:   SUB     0,0
        STA     0,0,3           ;RESET BREAK
        JSR     @RTN,2
DE1:    LDA     0,BTL           ;DELETE ALL
        NEG     0,0             ; - LENGTH
        LDA     3,BTBL
        SUB     1,1
DE2:    STA     1,0,3           ;RESET BREAK
        INC     3,3
        INC     0,0,SZR
        JMP     DE2
        JSR     @RTN,2
        .TXTM 0
QST:    .TXT /?/
BTL:    20.                     ;BREAK TABLE LENGTH
BTBL:   .+1
        .BLK    20.             ;BREAK TABLE
```

```
        .TITL   BREAKDOWN
        .ENT    SETED
        .EXTN   EDIT1,EDITT
        .NREL
EDIT:   EDIT1
SETED:  JSR     @EDIT           ;JMP TO BREAK DOWN
        JSR     @RTFR,2
        LDA     1,ECODE,2           ;GET EDIT CODE
        LDA     0,MSK2,2            ;GET A FIFTEEN
        SUB     0,1,SZR ;CHECK TO SEE IF EQUAL
        JMP     AAD     ;NOT EQUAL
        JMP     RTURN   ;EQUAL
AAD:    LDA     3,ECODE,2       ;GET EDIT CODE
        LDA     0,ETAB          ;GET EDIT CHAR TABLE
        ADD     0,3             ;GET EDIT CHAR DISPLACEMENT
        LDA     3,0,3           ;GET THE CHAR
        STA     3,ECHAR,2       ;STOR IN EDIT CHAR
        JMP     @.EDIT
RTURN:  JMP     @RTN,2
.EDIT:  EDITT
ETAB:   TAX
TAX:    0
        1
        2
        3
        4
        5
        6
        7
        8.
        9.
        10.
        11.
        12.
        13.
        14.
        15.
```

```
        .TITL   BYTE
        .ENT    GET,PUT
;THIS ROUTINE FETCHES BYTES AND RIGHT JUSTIFIED THEM IN AC1
;ROUTINE IS CALLED BY: JSR @GET WITH AC2 CONTAINING BYTE POINTER
        .ZREL   ;THESE ROUTINES REQUIRE 2 GLOBAL VECTORS
GET:    FETCH
PUT:    STORE
        .NREL
FETCH:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,BYPTR,2       ;GET BYTE POINTER
        MOVZR   3,3             ;CONVERT BYTE ADDRESS TO WORD ADDRESS
        LDA     1,0,3           ;FETCH WORD CONTAINING BYTE
        MOV#    0,0,SNC         ;TEST LOW ORDER BIT OF BYTE ADDRESS
        MOVS    1,1             ;SWAP BYTES IF LOW ORDER BIT IS ZERO
        LDA     0,RBMSK,2       ;FETCH BYTE MASK.2
        AND     0,1             ;CLEAR LEFT BYTE OF AC1
        JMP     @RTN,2          ;EXIT ROUTINE
;
; THIS ROUTINE STORES BYTES FROM AC1 (RIGHT JUSTIFIED) INTO
; MEMORY LOCATION POINTED TO BY AC2.  THE ADDRESS IN AC2 IS
; A BYTE POINTER.
; ROUTINE IS CALLED BY: JSR @PUT.
STORE:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,BYPTR,2       ;GET BYTE POINTER
        MOVZR   3,3             ;FORM WORD ADDRESS FROM BYTE ADDRESS
        LDA     0,0,3           ;GET WORD FROM BYTE STRING
        LDA     3,RBMSK,2       ;FETCH BYTE MASK
        ANDS    3,1,SZC         ;CLEAR LEFT BYTE OF STORE WORD,
                                ;TEST EVEN/ODD OF TARGET BYTE ADDRESS
                                ;AND SWAP BYTES OF STORE WORD
        MOVS    0,0             ;SWAP BYTES OF BYTE STRING WORD IF NECESSARY
        AND     3,0             ;CLOBBER BYTE TO BE REPLACED (LEFT BYTE)
        COM     1,1             ;PERFORM AC1 OR AC0
        AND     1,0
        ADC     1,0,SZC         ;REORIENT BYTES FOR STORE BACK
        MOVS    0,0             ;ROTATE WORD BACK IF NECESSARY
        LDA     3,BYPTR,2       ;GET THE BYTEPOINTER
        MOVZR   3,3     ;FORM WORD ADDRESS FROM BYTE ADDRESS
        STA     0,0,3           ;STORE WORD IN BYTE STRING
        JMP     @RTN,2          ;ROUTINE EXIT
```

```
;
;       THIS ROUTINE CALLS AND RETURNS FROM DEBUF
;
        .TITLE  CALDB
        .NREL
        .TXTM   1
        .ENT    CALDB,RTDBG
        .EXTN   DEBUG
CALDB:  ISZ     RTST,2
        STA     3,@RTST,2
        ISZ     RTST,2
        LDA     3,DBGP
        STA     3,@RTST,2
        JMP     @DBGP
RTDBG:  LDA     2,USP
        SUB     0,0
        STA     0,@RTST,2
        DSZ     RTST,2
        JSR     @RTN,2
DBGP:   DEBUG
```

```
      INTEGER IOFIL(6), INFIL(5,8), FWD, FCNT, FBUF, FDFIL(5,3), FDWD
      INTEGER FDCT, SVCH, BYTCT, SVBCT, FDFIL(40), BOT, DWD, DCNT, DRST
      INTEGER DISP, DFIL(9,3), VWD, VCNT, VFIL(40), DNE, OCNT, OWD, OUINT(5,4)
      INTEGER SVNAM(7), TMPNM(7), CR, INPNM(7), STWD, FDNAM(200), SDFL1
      COMMON /DAT/ IBUF(258), IBUF1(258), ICNVC(12), INFIL
      COMMON /FUDG/ BOT, CR, LP, IED, IOUT, INP, LLM, IULM, STWD
      EQUIVALENCE (INFIL,FDFIL),(INFIL,FDFIL),(INFIL,DFIL)
      EQUIVALENCE (INFIL,VFIL),(INFIL,OUINT)
      EQUIVALENCE (ICNVC(1),ICNT),(ICNVC(2),FWD),(ICNVC(3),FCNT),
     .   (ICNVC(4),FDWD),(ICNVC(5),FDCT),(ICNVC(6),DWD),(ICNVC(7),DCNT),
     .   (ICNVC(8),VWD),(ICNVC(9),VCNT),(ICNVC(11),OCNT),(ICNVC(10),OWD)
        DATA LLM/2/, IULM/7/, STWD/12/, IBUF(257)/9999/, IBUF(258)/0/
      DATA BOT/2H  /, CR/11/, LP/10/, IED/4/, IOUT/5/, INP/3/
        READ (CR,4000) INPNM
      READ (CR,4000) SVNAM
      READ (CR,4000) TMPNM
        READ (CR,1000) ICNVC
 1000   FORMAT (1215)
        IF (ICNVC(1).EQ.0) GO TO 50
        IULM=ICNVC(1)
   50   CALL OPEN (INP,INPNM,1,IER)
        IF (IER.NE.1) GO TO 890
      CALL OPEN (IOUT,SVNAM,3,IER)
      IF (IER.NE.1) GO TO 900
      CALL OPEN (IED,TMPNM,3,IER)
      IF (IER.NE.1) GO TO 910
        CALL OPEN (0,'DBG',3,IER)
        IF (IER.NE.1) GO TO 960
        CALL OPEN (1,'OLIST',3,IER)
        IF (IER.NE.1) GO TO 890
      DO 800 INDX=LLM,IULM
      GO TO (800,200,300,400,500,600,700,750), INDX
C
C       IO FILE PROCESSING SECTION
C
C        INPUT IS 80 COL CARD FORMAT.  THERE ARE 8 10 CHAR ALFA FIELDS PER CARD
C
C        THE FILE TERMINATOR IS A CARD WITH
C            COL   1-2   BLANK
C                 11-15  # ENTRIES
C
C
C       OUTPUT IS A 6 WORD ENTRY.  THE FIRST FIVE ARE THE NAME & 6 IS CHANNEL
C
C
  200 CONTINUE
      ICNT=0
      IWD=STWD
      IOFIL(6)=0
  210 READ (INP,2000) INFIL
 2000 FORMAT (40A2)
      IF (INFIL(1,1).EQ.'  ') GO TO 230
      DO 220 I=1,8
        IF (INFIL(1,I).EQ.'  ') GO TO 210
      DO 215 J=1,5
  215 IOFIL(J)=INFIL(J,I)
      CALL DSKFL (2,IBUF,IOUT,STWD,6,IOFIL,IER)
      IF (IEP.NE.1.AND.IER.NE.9) GO TO 900
      STWD=STWD+6
      ICNT=ICNT+1
  220 CONTINUE
```

```
      GO TO 210
  230 CONTINUE
      IF (ICNT.NE.NUMB(INFIL(1,2),5,0)) GO TO 920
      GO TO 800
C
C
C       FILE DESCRIPTOR PROCESSING SECTION
C
C       INPUT IS IN 80 COL CARD FORMAT.  THERE ARE 5 16 COLUMN ENTRIES PER CARD
C              THE FORMAT OF EACH ENTRY IS:
C              COL 1-4   ENTRY # OF DATA NAME
C                  5-6    ENTRY # OF IO TABLE
C                  7      LABEL  1-OMIT, 2-STD
C
C        THE FILE TERMINATOR IS A CARD WITH
C            COL   1-5   BLANK
C                 17-21  # ENTRIES
C                 33-37 # BUFFER CHAR
C
C
C       OUTPUT IS 3 WORD ENTRIES IN ORDER OF THE INPUT.
C
C
C
  300 CONTINUE
        DO 305 I=1,200
  305   FDNAM(I)=0
      FWD=STWD
      FCNT=0
      FBUF=0
  310 READ (INP,3000) FDFIL
 3000   FORMAT (3(I5,I4,I2,I4,I1,7X))
      IF (FDFIL(2,1).EQ.0) GO TO 330
      DO 320 I=1,3
      IF (FDFIL(2,I).EQ.0) GO TO 310
      CALL DSKFL (2,IBUF,IOUT,STWD,5,FDFIL(1,I),IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
      STWD=STWD+5
      FCNT=FCNT+1
        FDNAM(FCNT)=FDFIL(2,I)
  320 CONTINUE
      GO TO 310
  330 CONTINUE
      IF (FDFIL(1,2).NE.FCNT) GO TO 930
      IF (FDFIL(1,3).NE.FBUF) GO TO 930
      GO TO 800
C
C
C       EDIT FILE PROCESSING SECTION
C
C        INPUT IS IN 80 COLUMN CARD FORMAT WITH A CONTINUOUS CHARACTER STRING.
C       EACH EDIT PICTURE STRING IS TERMINATED WITH A ' AND '' ENDS THE
C       DATA ON A CARD
C       FILE TERMINATOR IS A CARD WITH
C       COL   1-2     BLANK
C            21-25    # STRINGS
C            41-45    #.CHARS
C
C       OUTPUT IS A PACKED EDIT PICTURE WITH 1 OR 2 BYTES FOR EACH CONSECUTIVE
C       STRING OF IDENTICAL CHARACTERS AND A TEMPORARY INDEX FILE WITH ONE
C       ENTRY PER STRING.  THE FORMAT OF THE 'IOUT' FILE WORD IS
```

```
C       BIT     0       EDIT PACKING. 0 SHORT FORM. 1 LONG FORM
C               1-4     EDIT CODE
C               5-15    REPITITION COUNT FOR LONG FORM
C               FOR SHORT FORM
C               5-7     REPITITION COUNT
C               8       1 IGNORE THIS BYTE. 0 2ND SHORT FORM
C               9-12    EDIT CODE
C               13-15   REPITITION COUNT
C
C       THE FORMAT FOR THE INDEX FILE ON CHANNEL IED IS ONE INTEGER
C           WORD PER STRING GIVING THE RELATIVE POINTER TO THE
C           START OF THE EDIT STRING IN THE OUTPUT FILE
C
C
  400 CONTINUE
      EDWD=STWD
        STWD=STWD+1
        IT5=EDWD
        NFLCH=0
      EDCT=1
      SVCH=-1
      BYTCT=0
      IT3=0
      IT4=0
      CALL DSKFL (2,IBUF1,IED,EDCT,1,0,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 910
  410 READ (INP,4000) EDFIL
C       WRITE(1,4100)EDFIL
 4100   FORMAT (1X,40A2)
      IF (EDFIL(1).EQ.'  ') GO TO 450
 4000 FORMAT (40A2)
      DO 430 I=1,80
      BYTCT=BYTCT+1
      IT1=IGBYT(EDFIL,I-1)
      IT3=IT3+1
        IF (SVCH.EQ.00T) GO TO 420
      IF (IT1.EQ.SVCH) GO TO 415
      IF (SVCH.EQ.-1) GO TO 415
      IEDC=IEDCD(SVCH)
      IF (IEDC.EQ.-1) GO TO 990
        ISBYC=BYTCT-1
  412   CONTINUE
C  412  WRITE (0,4200) IT4,ISBYC,IEDC,IT1,SVCH,STWD,IT3
 4200   FORMAT (12OI10)
        CALL IEDPK (IT2,IT4,ISBYC,IEDC,NFLCH)
      IF (IT4.EQ.1) GO TO 413
      CALL DSKFL (2,IBUF,IOUT,STWD,1,IT2,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
      STWD=STWD+1
      IF (IT4.EQ.2) GO TO 412
      IT4=0
  413   BYTCT=1
  415   SVCH=IT1
      GO TO 430
  420       IEDC=15
        ISBYC=1
C       WRITE (0,4200) IT4,ISBYC,IEDC,IT1,SVCH,STWD
      CALL IEDPK (IT2,IT4,ISBYC,IEDC,NFLCH)
      CALL DSKFL (2,IBUF,IOUT,STWD,1,IT2,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
      EDCT=EDCT+1
```

```
      STWD=STWD+1
      CALL DSKFL (2,IBUF1,IED,EDCT,1,STWD-EDWD,IER)
      IF (IER.NE.1) GO TO 910
        IT4=0
        CALL DSKFL (2,IBUF,IOUT,IT5,1,NFLCH,IER)
        IF (IER.NE.1.AND.IER.NE.9) GO TO 900
        IT5=STWD
        STWD=STWD+1
        NFLCH=0
        IF (IT1.EQ.OOT) GO TO 410
        SYCH=IT1
        BYTCT=1
  430   CONTINUE
C  430 WRITE (0,4200) IT3,BYTCT,ISBYC
      GO TO 410
  450 CONTINUE
        STWD=STWD-1
      EDCT=EDCT-1
        IT1=NUMB(EDFIL(11),5,0)
        IT2=NUMB(EDFIL(21),5,0)
        IF (IT1+IT2.EQ.0) GO TO 800
      IF (EDCT.NE.IT1) GO TO 940
      IF (IT3.NE.IT2) GO TO 940
      EDCT=STWD+1-EDWD
      GO TO 800
C
C
C       DATA NAME PROCESSING SECTION
C
C       INPUT IS IN 80 COL CARD FORMAT.  THERE ARE 4 20 COLUMN ENTRIES PER CARD
C               THE FORMAT FOR EACH ENTRY IS:
C               COL 1-5   DATA STARTING LOC
C                   6-7   DATA TYPE
C                   8-11  DATA LENGTH
C                  12     LEVEL
C                  13     SYNC
C                  14     BLANK
C                  15-17  OCCURS
C                  18-20  EDIT POINTER - IF DATA TYPE 4 OR 5 INPUT IS AN
C                         ENTRY # IN THE EDIT INDEX FILE (CHANNEL IED)
C                         WHICH CONTAINS THE ACTUAL POINTER
C               21-22   DIGITS TO RIGHT OF DECIMAL POINT FOR TYPES 0, 1
C
C        THE FILE TERMIANTOR IS A CARD WITH
C            COL   1-5   BLANK
C                 23-27  # ENTRIES
C                 45-49  # CHAR RAN STORAGE
C
C        OUTPUT IS A 5 WORD ENTRY OF THE FOLLOWING FORM:
C               WORD  1   DATA PTR
C                     2   BLANK B4, SYNC B5,LEVEL B6-7, SIGN B11-12, TYPE B13-15
C                     3   LENGTH
C                     4   DECIMAL PTR B7, # OCCURS
C                     5   EDIT POINTER
C
C
  500 CONTINUE
X       WRITE (12,4300) (FDNAM(I),I=1,FCNT)
X4300   FORMAT (10I10)
      DWD=STWD
      DCNT=0
```

```
      DRST=0
      DISP=0
        SDFL1=-1
        IT4=0
  510 READ (INP,5000) DFIL
C       WRITE (12,5100) DFIL
        IF (DFIL(1,1) EQ 9999) GO TO 530
      DO 520 I=1,3
        IF (DFIL(1,I) EQ 9999) GO TO 510
C       WRITE (12,4200) (DFIL(J,I),J=1,8)
        IF (DFIL(4,I) NE 1) DFIL(1,I)=DFIL(1,I)+DISP
      IT1=DFIL(2,I)/10
      IT2=DFIL(2,I)-10*IT1
      IF (IT2 LE 3 OR IT2 GE 6) GO TO 512
      CALL DSKFL (1,IBUF1,IED,DFIL(8,I),1,DFIL(8,I),IER)
C       WRITE (1,4200) IED,DFIL(8,I),I
      IF (IER NE 1) GO TO 910
  512 DFIL(2,I)=IT2+8*(32*(4*(2*DFIL(6,I)+DFIL(5,I))+DFIL(4,I))+IT1)
C       WRITE (1,4200) DFIL(2,I),DFIL(4,I),DFIL(5,I),DFIL(6,I),IT1
      IF (DFIL(4,I) NE 2 AND IT2 NE 0) GO TO 5149
        IT3=987
        IF (IT2 EQ 0) GO TO 513
        IF (SDFL1 EQ DFIL(1,I)) GO TO 5149
      IF (DRST+DISP NE DFIL(1,I)) WRITE (12,4200) IT3,DRST,DISP,DFIL(1,I)
      DRST=DRST+DFIL(3,I)
        DISP=DISP+IT4
        DFIL(1,I)=DFIL(1,I)+IT4
        IT4=0
        GO TO 514
  513   DFIL(1,I)=DRST+DISP
C******ALGORITHM
        DISP=DISP+10
  514       IF (MOD(DFIL(1,I),2) EQ 0) GO TO 5142
      DISP=DISP+1
        DFIL(1,I)=DFIL(1,I)+1
 5142   DO 5145 J=1,200
        IF (FDNAM(J) EQ 0) GOTO 5148
        IF (FDNAM(J) EQ DCNT+1) GO TO 5147
 5145   CONTINUE
        GO TO 5148
 5147   IT4=1
 5148   SDFL1=DFIL(1,I)
 5149   DCNT=DCNT+1
  515 DFIL(4,I)=DFIL(7,I)+DFIL(9,I)*2048
      DFIL(5,I)=DFIL(8,I)
      CALL DSKFL (2,IBUF,IOUT,STWD,5,DFIL(1,I),IER)
      IF (IER NE 1 AND IER NE 9) GO TO 900
      STWD=STWD+5
  520 CONTINUE
      GO TO 510
  530 CONTINUE
 5000 FORMAT (3(I5,I2,I4,3I1,2I3,I2))
 5100   FORMAT (3(I6,I3,I5,3I2,2I3,I3))
        DRST=DRST-1
      IF (DRST NE DFIL(1,3)) GO TO 950
      IF (DCNT NE DFIL(1,2)) GO TO 950
      GO TO 800
C
C
C       VALUES PROCESSING SECTION
C
```

```
C       INPUT IS IN 80 COL CARD FORMAT WITH A CONTINUOUS CHARACTER STRING
C            EACH VALUE IS TERMINATED BY A ' AND '' ALSO ENDS A DATA CARD
C            THE FILE TERMINATOR IS A CARD WITH
C            COL  1-2   BLANK
C                 21-25 # VALUES
C                 41-45  # VALUES CHARS
C            THE FIRST 4 CHAR OF EACH VALUE THE ASSOC D-N FILE ENTRY #
C
C       OUTPUT IS THE TOTAL RAM STORAGE FILE ZEROED OUT WITH THE VALUES
C            INSERTED AT THE APPOPRIATE LOCATIONS
C
600   CONTINUE
      IT3=0
      IT2='##'
      CALL DSKFL (4,IBUF,IOUT,STWD,(DRST+DISP+1)/2,IT2,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
        IT2=0
      IT1=0
      VWD=STWD
      VCNT=0
         BYTCT=0
        READ (INP,4000) VFIL
        IT5=NUMB(VFIL,5,10)
610   READ (INP,4000) VFIL
X       WRITE (12,4100) VFIL
      DO 690 I=1,80
        IF (IT1.EQ.-2) GO TO 630
        IF (IT3.GT.0) GOTO 640
      IT2=IT2+7
      IF (I.LE.74) GO TO 620
      J=I-74
      K=7-J
        IT3=0
        L=0
        IF (K.LE.4) GO TO 615
        L=K-4
        K=4
        IT3=NUMB (VFIL,L,I+3)
  615   IT4=NUMB (VFIL,K,I-1)
        IG=101
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT4,IT5,DNE,BYTCT,J,K,L
      I=80
 6000   FORMAT (13I10)
      IT1=-2
      GO TO 690
620     CONTINUE
X       WRITE (12,4100) VFIL
        IT4=NUMB (VFIL,4,I-1)
        IT3=NUMB (VFIL,3,I+3)+1
        IG=102
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT5,DNE,BYTCT
      I=I+6
      IT1=0
      GO TO 670
  630   I=J
        IF (K.EQ.4) GO TO 633
        J=J-3
        IT3=NUMB(VFIL,3,J)+1
        IT4=IT4*10**J+NUMB(VFIL,J,0)
        GO TO 636
  633   IT3=IT3*10**J+NUMB(VFIL,J,0)+1
```

```
        IG=103
  636   IT1=0
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT4,IT5,DNE,BYTCT,J,K,L
      GO TO 670
640   IT4=IGBYT(VFIL,I-1)
      IT2=IT2+1
        IG=106
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT4,IT5,DNE,BYTCT
      CALL DSKBT (2,IBUF,IOUT,BYTCT,1,IT4,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 910
        BYTCT=BYTCT+1
      GO TO 680
  670   DNE=DWD+(IT4-1)*5
        IG=107
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT4,ITK,DNE,J,K,L
        CALL    DSKFL (1,IBUF,IOUT,DNE,1,IT4,IER)
        IF (IER.NE.1) GO TO 900
        IG=104
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT4,IT5,DNE,BYTCT
        BYTCT=IT4+VWD*2
        IG=105
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT5,DNE,BYTCT
680   IT3=IT3-1
        IF (IT3.GT.0) GO TO 690
        IG=108
X       WRITE (12,6000) IG
X       WRITE (12,6000) IG,I,IT1,IT2,IT3,IT4,IT5,VCNT
X       WRITE (12,6000) IG
        VCNT=VCNT+1
        IF (VCNT.EQ.IT5) GOTO 695
690   CONTINUE
      GO TO 610
695     READ (INP,4000) VFIL
        IF (IT2+1.NE.NUMB(VFIL(21),5,0)) GO TO 960
      IF (VCNT.NE.NUMB(VFIL(11),5,0)) GO TO 960
      VCNT=(DRST+DISP+1)/2
      STWD=VWD+VCNT
      GO TO 800
C
C       QUINT PROCESSING SECTION
C
C       INPUT IS IN 80 COL CARD FORMAT  THERE ARE 4 20 COLUMN QUINTS
C               ON EACH CARD  A QUINT IS OF THE FORM:
C               COL 1-4 OPCODE
C                   5-8  OPERAND 1
C                  9-12  OPERAND 2
C                 13-16 OPERAND 3
C                 17-20 SORCE SEQUENCE #
C       THE FILE TERMINATOR IS A CARD WITH
C            COL   1-4   BLANK
C                 21-25  # QUINTS
C
C       OUTPUT IS FIVE WORD ENTRY.  THE STARTING WORD OF FILE IS WORD
C               OF THE DIRECTORY, THE NUMBER OF ENTRIS IS WORD    OF
C               DIRECTORY.  THE ENTRY FORMAT IS:
C               WORD  1  OPCODE
C                     2 OPERAND 1
C                     3  OPERAND 2
C                     4  OPERAND 3
C                     5  SEQ #
C
```

```
C
C
  700 CONTINUE
      QCNT=0
      QWD=STWD
  710 READ (INP,7000) QUINT
 7000 FORMAT (20I4)
      IF (QUINT(1,1).EQ.0) GO TO 730
      DO 720 I=1,4
        IF (QUINT(1,I).EQ.0) GO TO 710
        IT1=QUINT(2,I)/10
        IT2=QUINT(2,I)-IT1*10
        QUINT(2,I)=IT1*8+IT2
      CALL DSKFL (2,IBUF,IOUT,STWD,5,QUINT(1,I),IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
      STWD=STWD+5
      QCNT=QCNT+1
        WRITE (1,7500) QCNT,(QUINT(J,I),J=1,5),QCNT,(QUINT(J,I),J=1,5)
 7500   FORMAT (6I8,15X,6OI8)
  720 CONTINUE
      GO TO 710
  730 CONTINUE
      IF (QUINT(1,2)*10+QUINT(2,2)/1000.NE.QCNT) GO TO 970
        GO TO 800
  750   STWD=1
  760   CALL DSKFL (1,IBUF,IOUT,STWD,1,IT1,IER)
        IF (IER.NE.1.AND.IER.NE.9) GO TO 900
        WRITE (12,6500) IBUF,STWD
 6500 FORMAT (18(/1X,15OI7))
        IF (IER.EQ.9) GO TO 800
        STWD=STWD+256
        GO TO 760
800   CONTINUE
        ICNT=ICNT*6
        FCNT=FCNT*5
        EOCT=EOCT-1
        DCNT=DCNT*5
        QCNT=QCNT*5
      CALL DSKFL (2,IBUF,IOUT,1,11,ICNVC,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
      CALL DSKFL (3,IBUF,IOUT,IT1,IT1,IT1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 900
C       WRITE (0,8000) IBUF
 8000   FORMAT  (18(/1X,15OI7))
      STOP
  890   WRITE(LP ,8900) IER
 8900   FORMAT ('1DISK ERROR ON INOUT FILE: ',I6)
        STOP
  900 WRITE (LP ,9000) IER
 9000 FORMAT ('1DISK ERROR ON SAVE FILE: ',I6)
      STOP
  910 WRITE (LP ,9100) IER
 9100 FORMAT ('1DISK ERROR ON TEMP FILE: ',I6)
      STOP
  920 WRITE (LP ,9200) ICNT
 9200 FORMAT ('1IO FILE ERROR, COUNT: ',I6)
      STOP
  930 WRITE (LP ,9300) FCNT,FBUF
 9300 FORMAT ('1FD FILE ERROR, COUNT: ',I6,' BUFFER: ',I6)
      STOP
  940 WRITE (LP ,9400) EOCT,IT3,IT1,IT2
```

```
 9400 FORMAT ('1EDIT FILE ERROR, COUNT: ',I6,' CHARS: ',3I6)
      STOP
  950 WRITE (LP ,9500) DCNT,DRST,DFIL(1,3),DFIL(1,2)
 9500 FORMAT ('1DN FILE ERROR, COUNT: ',I6,' STORAGE: ',I6)
      STOP
C  955 WRITE (LP ,9550) I,DRST,DISP,DFIL(1,I)
C 9550 FORMAT ('1DN ENTRY ',I6,' BAD, LENGTH PARAMETERS: ',3I6)
C       STOP
  960 WRITE (LP ,9600) VCNT,IT2
 9600 FORMAT ('1VALUES FILE ERROR, COUNT: ',I6,' CHARS: ',I6)
      STOP
  970 WRITE (LP,9700) QCNT
 9700 FORMAT ('1QUINT FILE ERROR, COUNT: ',I6)
      STOP
  990 WRITE (LP ,9900) SVCH
 9900 FORMAT ('ILLEGAL EDIT CHARACTER: ',A2)
      STOP
      END
```

```
      SUBROUTINE DSKFL (IFCN, IBUF, IO, STWD, ICT, IVEC, IER)
      DIMENSION IBUF(258), IVEC(1)
      INTEGER STWD, CNT, SECTR, WORD, DISP
C
C
C     IFCN  FUNCTION. 1 READ, 2 WRITE, 3 SAVE, 4 WRITE CONSTANT
C     IBUF DISK BUFFER, 256 DATA WORDS, WORD 257 IS SECTOR #, WORD 258 IS
C               WRITE FLAG -1 MEANS UPDATED BUT NOT ON DISK, 0 IS CLEAN
C     IO    CHANNEL
C     STWD STARTING WORD OF DISK FILE
C     CNT  # OF WORD TO PROCESS
C     IVEC  DATA VECTOR TO PROCESS
C     IER    ERROR FLAG
C
C       IF (IFCN.EQ.2.OR.IFCN.EQ.4) WRITE (12,6000) (IVEC(I), I=1, ICT)
      CNT=ICT
      SECTR=(STWD-1)/256
      DISP=SECTR*256-STWD+1
      WORD=1-DISP
      IER=1
        IGI=1
C       WRITE (0,5000) IGI, IBUF(257), IBUF(258), LIM, WORD, CNT, DISP, IO, STWD, ICT, SEC
      GO TO (100,200,300,400), IFCN
C
C     READ
C
100   IF (SECTR.EQ.IBUF(257)) GO TO 120
      IF (IBUF(258).GE.0) GO TO 110
      CALL WRBLK (IO, IBUF(257), IBUF, 1, IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=1
X       WRITE (12,6000) IBUF, SECTR, IGI, STWD, ICT, DISP, WORD, CNT
110   CALL RDBLK (IO, SECTR, IBUF, 1, IER)
      IF (IER.NE.1) GO TO 500
120   LIM=WORD+CNT-1
      IF (LIM.LE.256) GO TO 130
      CNT=LIM-256
      LIM=256
        IGI=2
C       WRITE (0,5000) IGI, IBUF(257), IBUF(258), LIM, WORD, CNT, DISP, IO, STWD, ICT, SEC
      GO TO 140
130   CNT=0
        IGI=3
C       WRITE (0,5000) IGI, IBUF(257), IBUF(258), LIM, WORD, CNT, DISP, IO, STWD, ICT, SEC
140   DO 150 I=WORD, LIM
150   IVEC(I+DISP)=IBUF(I)
      IF (CNT.EQ.0) GO TO 180
      IF (DISP.GT.0) GO TO 170
      DISP=LIM-WORD+1
160   WORD=1
      SECTR=SECTR+1
      CALL RDBLK (IO, SECTR, IBUF, 1, IER)
      IF (IER.NE.1) GO TO 500
      IF (CNT.LE.256) GO TO 120
      LIM=256
      CNT=CNT-256
        IGI=4
C       WRITE (0,5000) IGI, IBUF(257), IBUF(258), LIM, WORD, CNT, DISP, IO, STWD, ICT, SEC
      GO TO 140
170   DISP=DISP+256
      GO TO 160
```

```
180     IBUF(257)=SECTR
        IBUF(258)=0
      GO TO 600
C
C     WRITE
C
200   IF (SECTR.EQ.IBUF(257)) GO TO 220
      IF (IBUF(258).GE.0) GO TO 210
      CALL WRBLK (IO,IBUF(257),IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=2
X       WRITE (12,6000) IBUF,SECTR,IGI,STWD,ICT,DISP,WORD,CNT
        IGI=5
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
  210 CALL RDBLK (IO,SECTR,IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=6
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
220   LIM=WORD+CNT-1
      IF (LIM.LE.256) GO TO 230
      CNT=LIM-256
      LIM=256
        IGI=7
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
      GO TO 240
230   CNT=0
        IGI=8
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
240   DO 250 I=WORD,LIM
250   IBUF(I)=IVEC(I+DISP)
      IF (CNT.EQ.0) GO TO 280
      IF (DISP.GT.0) GO TO 270
      DISP=LIM-WORD+1
260   WORD=1
      CALL WRBLK (IO,SECTR,IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=3
X       WRITE (12,6000) IBUF,SECTR,IGI,STWD,ICT,DISP,WORD,CNT
        IGI=9
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
      SECTR=SECTR+1
      CALL RDBLK (IO,SECTR,IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
      IF (CNT.LE.256) GO TO 220
      LIM=256
      CNT=CNT-256
        IGI=10
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
      GO TO 240
270   DISP=DISP+256
      GO TO 260
280   IBUF(257)=SECTR
        IBUF(258)=-1
      GO TO 600
C
C     SAVE
C
300   CALL WRBLK(IO,IABS(IBUF(257)),IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=11
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
```

```
        IGI=4
X       WRITE (12,6000) IBUF,SECTR,IGI,STWD,ICT,DISP,WORD,CNT
        IBUF(258)=0
      GO TO 600
C
C     ZERO
C
400   IV=IVEC(1)
      IF (SECTR EQ IBUF(257)) GO TO 420
      IF (IBUF(258) GE 0) GO TO 410
      CALL WRBLK (IO,IBUF(257),IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=5
X       WRITE (12,6000) IBUF,SECTR,IGI,STWD,ICT,DISP,WORD,CNT
  410 CALL RDBLK (IO,SECTR,IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
420   LIM=WORD+CNT-1
      IF (LIM.LE.256) GO TO 430
      CNT=LIM-256
      LIM=256
        IGI=12
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
      GO TO 440
430   CNT=0
        IGI=13
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
440   DO 450 I=WORD,LIM
450   IBUF(I)=IV
      IF (CNT.EQ.0) GO TO 480
      IF (DISP.GT.0) GO TO 470
      DISP=LIM-WORD+1
460   WORD=1
      CALL WRBLK (IO,SECTR,IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
        IGI=6
X       WRITE (12,6000) IBUF,SECTR,IGI,STWD,ICT,DISP,WORD,CNT
      SECTR=SECTR+1
      CALL RDBLK (IO,SECTR,IBUF,1,IER)
      IF (IER.NE.1.AND.IER.NE.9) GO TO 500
      IF (CNT.LE.256) GO TO 420
      LIM=256
      CNT=CNT-256
        IGI=14
C       WRITE (0,5000) IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
      GO TO 440
470   DISP=DISP+256
      GO TO 460
480   IBUF(257)=SECTR
        IBUF(258)=-1
        IGI=15
C       WRITE (0,5000),IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
      GO TO 600
C
C     ERROR
C
500   IBUF(257)=9999
        IBUF(258)=0
        IGI=16
C       WRITE (0,5000),IGI,IBUF(257),IBUF(258),LIM,WORD,CNT,DISP,IO,STWD,ICT,SEC
C600    IF (IFCN EQ 1) WRITE (12,6000) (IVEC(I),I=1,ICT)
  600   CONTINUE
```

```
5000    FORMAT (13I10)
6000    FORMAT (18(/1X,15OI7))
        RETURN
        END
```

```
      SUBROUTINE IEDPK (I,J,K,L,M)
C
C       INPUT IS:
C         J     BYTE PTR FOR OUTPUT WORD
C         K     REPETITION COUNT
C         L     EDIT CODE
C       M       TOTAL # FILL CHAR (Z,9,X) -1 ($,*)
C
C       OUTPUT IS PACKED EDIT CODE IN I.  IF J IS 0 OUTPUT IS IN LEFT BYTE
C         IF J IS 1 LEFT BYTE HAS BEEN PACKED WITH SHORT FORM AND OUTPUT
C         IS ADDED INTO RIGHT BYTE
C         IK K > 8 OUTPUT IS A FULL WORD
C
      CONTINUE
A       LDA     2,@T.+3,3
A       ADDZL   2,2
A       MOVZL   2,2             ;SHIFT LEFT 3 TO POSITION CODE
A       LDA     0,@T.+2,3
A       ADC     1,1
A       ADD     1,0             ;DECRIMENT REPETITION COUNT
A       MOVZR   0,1
A       MOVZR   1,1
A       MOVZR   1,1,SZR         ;LONG FORM?
A       JMP     IED2            ;YES
A       ADD     2,0             ;COMBINE EDIT CODE & REPETITION COUNT
A       LDA     1,@T.+1,3
A       MOVR    1,1,SZC         ;LEFT BYTE PACKED?
A       JMP     IED1            ;YES
A       ISZ     @T.+1,3
A       MOVS    0,0
A       JMP     IED5
AIED1:  LDA     1,@T.+0,3
A       ADD     1,0             ;ADD TO LAST CHAR
A       JMP     IED4
AIED2:  LDA     1,@T.+1,3
A       MOVR    1,1,SNC         ;LEFT BYTE PACKED?
A       JMP     IED3            ;NO
A       ISZ     @T.+1,3       ;SET FLAG TO 2 SO LEFT BYTE WILL BE STORED
A       SUBZR   1,1
A       MOVS    1,1
A       LDA     0,@T.+0,3
A       ADD     1,0             ;SET BIT 8 TO 1 TO INDICATE LONG FORM NEXT
A       JMP     IED5
AIED3:  MOVS    2,2
A       ADD     2,0             ;ADD EDIT CODE TO REPETITION COUNT
A       SUBZR   1,1
A       ADD     1,0             ;SET BIT 0 TO 1 FOR LONG FORM
AIED4:  SUB     1,1
A       STA     1,@T.+1,3
AIED5:  STA     0,@T.+0,3       ;STORE PACKED WORD
C
        IF(L.GT.13) RETURN
        IF(L.LT.11.AND.L.NE.8) RETURN
        M=M+K
        IF (L.EQ.8.OR.L.EQ.11) M=M-1
      RETURN
      END
```

```
      FUNCTION IGBYT (ARRAY,BYTPTR)
C
C       INPUT IS AN ARRAY AND POINTER TO THE BYTE WANTED (0 IS 1ST BYTE)
C
C       OUTPUT IS LEFT JUSTIFIED BYTE WITH RIGHT BLANK (A1 FORMAT)
C
      CONTINUE
C
A       LDA     2,T.+1,3
A       LDA     1,@T.+2,3
A       MOVZR   1,1
A       ADD     1,2
A       LDA     0,0,2
A       MOV     0,0,SZC
A       MOVS    0,0
A       LDA     1,MSK
A       AND     0,1
A       LDA     0,I40
A       ADD     1,0
A       STA     0,@T.+0,3
A       JMP     .+3
AMSK:   177400
AI40:   40
C
      RETURN
      END
```

```
      FUNCTION NUMB (ARRAY,NCHAR,BYTPTR)
C
C       INPUT IS AN ARRAY, THE NUMBER OF ASCII CHAR TO BE CONVERTED TO A
C       BINARY NUMBER AND A BYTE POINTER TO 1ST CHAR (0 - 1 ST BYTE OF APR)
C
C       OUTPUT IS BINARY #, OVER FLOW, VALIDITY OF DIGITS NOT CHECKED
C
      CONTINUE
C
A       LDA     0,@T.+2,3       ;2ND ARG VAL
A       STA     0,CTR
A       LDA     0,@T.+3,3       ;3RD ARG VAL
A       LDA     1,T.+1,3        ;1ST ARG ADDR
A       MOVZL   1,1
A       ADD     0,1             ;CORE BYTE PTR 1ST CHAR
A       STA     1,BYTPT
A       SUB     0,0
A       STA     0,@T.+0,3       ;ZERO NUMB
ANMB1:  LDA     2,BYTPT
A       LDA     1,@T.+0,3
A       MOVZL   1,0
A       MOVZL   0,0
A       ADDZL   0,1             ;NUMB * 10
A       MOVZR   2,2             ;ADDR CURRENT BYTE
A       LDA     0,0,2
A       LDA     2,MSK
A       MOV     2,2,SNC
A       MOVS    0,0             ;SWAP LEFT BYTE TO RIGHT
A       AND     0,2
A       ADD     2,1             ;ADD NEW DIGIT
A       STA     1,@T.+0,3       ;STORE IN NUMB
A       ISZ     BYTPT
A       DSZ     CTR
A       JMP     NMB1
A       JMP     .+4
AMSK:   17
ACTR:   .BLK    1
ABYTPT: .BLK    1
C
      RETURN
      END
```

```
      FUNCTION IEDCD (ICHR)
C
C       THE EDIT CODE FORMAT IS
C               CODE    CHAR
C                  0    NULL
C                  1    B
C                  2    /
C                  3    0
C                  4    ,
C                  5    D (OF DB)
C                  6    C (OF CR)
C                  7    .
C                  8    $
C                  9    -
C                 10    +
C                 11    *
C                 12    Z
C                 13    9 OR X
C                14       V
C                 15    EOS (NOT IN INPUT STRING)
C
      INTEGER ICOR(16)
        COMMON/DTFUD/ICOR
      DATA ICOR/32,'B ','/ ','0 ',', ','D ','C ','. ','$ ','- ',
     .   '+ ','* ','Z ','9 ','X ','V '/
      DO  10 I=1,16
      IF (ICHR.EQ.ICOR(I)) GO TO 20
   10 CONTINUE
      IEDCD=-1
      RETURN
   20 IEDCD=I-1
      IF (IEDCD.GE.14) IEDCD=IEDCD-1
      RETURN
      END
```

```
      SUBROUTINE PUTBT (ARRAY,CHAR,BYTPTR)
C
C       INPUT IS A LEFT JSUTIFIED BYTE AND POINTER TO THE BYTE WANTED (0-1ST)
C
C       OUTPUT IS THE BYTE PACKED IN THE ARRAY/CHARACTER STRING AT BYTPTR
C
      CONTINUE
C
A       LDA     2,T.+0,3
A       LDA     1,@T.+2,3
A       MOVZR   1,1
A       ADD     1,2
A       LDA     1,@T.+1,3
A       LDA     0,MSK
A       AND     0,1
A       STA     1,SVCHR
A       MOV     1,1,SZC
A       JMP     PUT1
A       MOVS    0,0
A       LDA     1,0,2
A       AND     0,1
A       LDA     0,SVCHR
A       ADD     0,1
A       JMP     PUT2
APUT1:  LDA     1,0,2
A       AND     0,1
A       LDA     0,SVCHR
A       MOVS    0,0
A       ADD     0,1
APUT2:  STA     1,0,2
A       JMP     .+3
AMSK:   177400
ASVCHR: .BLK    1
C
        RETURN
        END
```

```
      SUBROUTINE DSKBT (IFCN, IBUF, IO, BTPTR, BTCT, BYTE, IER)
      DIMENSION IBUF(257)
      INTEGER STWD, BTCT, BTPTR, BYTE
      IF (BTCT.EQ.1.AND.IFCN.EQ.2) GO TO 200
      IER=-1
      RETURN
  200 STWD=(BTPTR)/2
X     WRITE (12,1000) BTPTR, BTCT, BYTE, STWD
      CALL DSKFL (1, IBUF, IO, STWD, BTCT, IT1, IER)
      IF (IER.NE.1.AND.IER.NE.9) RETURN
      IT2=MOD(BTPTR,2)
      IG=108
X     WRITE(12,1000) IG, BTPTR, BYTE, IT1, IT2, STWD
 1000 FORMAT (10OI10)
      CALL PUTBT (IT1, BYTE, IT2)
      IG=109
X     WRITE (12,1000) IG, BTPTR, BYTE, IT1, IT2, STWD
      CALL DSKFL (2, IBUF, IO, STWD, BTCT, IT1, IER)
      RETURN
      END
```

```
        .TITL   CLOSE
        .ENT    CLOS
        .NREL
CLOS:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     0,OP1DP,2       ;LOAD THE BYTE POINTER
        LDA     2,CHANL,2       ;LOAD THE CHANNEL NUMBER
        .SYSTEM
        .CLOSE  77
        JMP     @RTER,2 ;BAD CLOSE JUMP TO ERROR
        MOV     3,2
        JMP     @RTN,2          ;PROCESSING COMPLETE, RETURN
```

```
        .TITLE  COBOL
        .NREL
        .TXTM   1
        .EXTN   CLG,ENTER,NEW,RUN,KILL,BREAK,CONT,LIST,STEP,INSNS
        .ENT    COBOL,ERROR,PRERR
        .EXTN   DATA,CALOB,DELET
COBOL:  LDA     2,DATAP
        STA     2,USP
        LDA     0,TTOP,2
        JSR     @OPAVP,2        ;OPEN AVAILABLE CHANNEL TO TTO
        JSR     ERROR
        MOV     1,2
        LDA     0,CMSG
        .SYSTM
        .WRL    77              ;WRITE COBOL
        JSR     ERROR
        LDA     2,CHNUM,3
        .SYSTM
        .CLOSE  77              ;CLOSE TTO
        JSR     ERROR
        MOV     3,2
CB1:    LDA     0,ASTRK
        .SYSTM
        .PCHAR                  ;PROMPT
        JSR     ERROR
CB2:    JSR     @GCHAP,2
        JSR     CB2
        .SYSTEM
        .PCHAR
        JSR     ERROR
        MOV     3,2
        SUB     3,3
        STA     3,CERFL,2
        JSR     CLI             ;INTERPRET COMMAND CHAR
        JMP     .+3
        JMP     CB2
        JMP     CB1
        LDA     3,CERFL,2
        MOV     3,3,SNR
        JSR     ERROR
        JSR     PRERR           ;ILLEGAL COMMAND
PRERR:  LDA     0,TTOP,2
        JSR     @OPAVP,2        ;OPEN TTO
        JSR     ERROR
        MOV     1,2
        LDA     0,ILMSG
        .SYSTM
        .WRL    77              ;WRITE ILLEGAL COMMAND
        JSR     ERROR
        LDA     2,CHNUM,3
        .SYSTM
        .CLOSE  77              ;CLOSE TTO
        JSR     ERROR
        LDA     2,USP   ;RESTORE AC2
        JMP     CB1             ;NEXT CHARACTER
CLI:    ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @VALDP,2        ;DIGIT?
        JMP     ALFA
        JSR     @IMINP
        JSR     @RTER,2
```

```
        ISZ     @RTST,2
        JSR     @RTN,2
CMSG:   .+1*2
        .TXT    /COBOL VERSION 1.0 4-1-75<15>/
ILMSG:  .+1*2
        .TXT    /ILLEGAL COMMAND<15>/
ALFA:   LDA     3,CDTA
ALFAI:  LDA     1,0,3   ;COMMANDALFA:CHAR
        MOV     1,1,SNR ;ENDTOTABLE?
        JMP     ALFA2
        SUB     0,1,SNR ;THIS CHAR?
        JMP     .+3
        INC     3,3     ;NO
        JMP     ALFAI
        JSR     @DISPL,3        ;EXECUTE
        JSR     @RTER,2
        ISZ     @RTST,2
        JSR     @RTN,2
ALFA2:  ISZ     CERFL,2
        JSR     @RTER,2
        .TXTM   0
ASTRK:  .TXT    /*/
CDTA:   .+1
        .TXTM   0
        .TXT    /G/
        .TXT    /E/
        .TXT    /N/
        .TXT    /R/
        .TXT    /K/
        .TXT    /B/
        .TXT    /D/
        .TXT    /C/
        .TXT    /L/
        .TXT    /S/
        .TXT    /*/     ;CONTROL D
        0
        DISPL=  14
        CLG
        ENTER
        NEW
        RUN
        KILL
        BREAK
        DELET
        CONT
        LIST
        STEP
        CALDB
IMINP:      IMINS
DATAP:  DATA
        .TXTM   1
ERROR:  LDA     0,TTOP,2
        JSR     @OPAVP,2
        JSR     @RTER,2
        MOV     1,2
        LDA     0,MESS
        .SYSTEM
        .WRL    77
        JSR     @RTER,2
        MOV     3,2
        LDA     0,QC,2  ;GET QUINT COUNT
```

```
        JSR     @PROCP,2
        JSR     @RTER,2         ;WRITE QUINT COUNT
        LDA     0,FIVE,2
        STA     0,X4T,2
AA:     LDA     3,QP,2          ;GET QUINT POINTER
        LDA     0,0,3           ;GET A WORD
        JSR     @PROCP,2        ;PUT THE WORD OUT TO TTI
        JSR     @RTER,2
        ISZ     QP,2
        DSZ     X4T,2
        JMP     AA
        JSR     @PRCRP,2        ;PUT OUT A CARRAGE RETURN AND LINE FEED
        JSR     @RTER,2
        JSR     PRERR
MESS:   .+1*2
        .TXT    /ERROR RETURN, ON QC# /
```

```
          .TITL   DATABASE
          .ENT    IBUF,MS256,ONE87,TWO56,DNPT1,DNPT3,DNPT2,CR,LF,FF,BELL,EOFN
          .ENT    QMRK,ECHT,BTCNT,BLK,C60,ONE,TWO,THREE,FOUR,FIVE,SIX,NINE,TEN
          .ENT    TWLVE,NFLAG,CHKWD,DOL,AST,PLUS,MINUS,C1,DB,MM,COL,BEERI,SKPIT
          .ENT    ERCDE,SFLAG,CNT2,SIGN,X3T,X4T,X11T,X17T,SAVE1,NUM,I,N,NT,BYPTR
          .ENT    PNT2,ECHAR,RDCHK,TAD,POS,INDIC,PNT1,RCNT,ECNT,CNT1,FFLAG
          .ENT    SAV,RDGIT,TYP,SAV1,PACTM,ECODE,DIGIT,BEERR,CHANI,OP,OC,SP,RTST
          .ENT    NDPT,CNT,SAVE,RTPT,OP1DL,OP1DP,OP1DT,OP1DF,OP1DX,OP2DL
          .ENT    OP2DP,OP2DF,OP2DX,OP2DT,OP3DL,OP3DP,OP3DF,OP3DX,OP3DT,RTRN1
          .ENT    PACK1,PACK2,PACK3,PACT,PAC1,PAC2,PAC3,PACT1,PACT2,PACT3,M,SAV2
          .ENT    PACTL,SAV3,PAC2,OPDT,OPDL,OPDP,OPDF,DEC1,DEC2,DEC3,NULT,C147
          .ENT    EMSK5,EMSK4,EMSK3,EMSK2,EMSK1,SHRT,STAV,NEGNO,SLSH,PERD,C100
          .ENT    RBMSK,TOC,MPFLG,ARGDF,CHAR1,CHAR2,TEMP,ONST,PSCNT,RSST,DNCNT
          .ENT    DNST,EDCNT,EDST,FDCNT,FDST,IOCNT,IOST,ONCNT,IOPTR,SVFCN
          .ENT    ELEVN,SVFLN,DATA,MSK,MSK2,C71,PSMX,OPFLG,DATAB,RTRN,CERFL
          .ENT    TTOP,CHNUM,VALDP,RTN,RTEP,NEWF,SNMAX,OPDSP,RTRNP,DEL,PNTA,CNTA
          .ENT    TEMP2,ALCRP,PRERP,IFCN,IO,STND,ICT,IVEC,IER,IOSTN,FDSTN,FDSTN
          .ENT    NMNCP,NMWCP,EXONP,OPAVP,DNSTN,RSSTN,ONSTN,GCHAP,STPFL
          .ENT    SUBT1,ZERA,ADDT,DV10,MMOV,MTEN,C31,MSK4,MSK3
          .EXTN   SUBT,ZERDA,ADDIT,DIV10,MOVA,MUL10
          .EXTN   VALDG,OPAVC,ALCOR,PREPP,RETRN,ERTRN,NMNCH,NMNCH,EXONT,OPDST,GCHA
          .EXTN   GDNST,GRSST,RDAEC,PROCT,PRCRL,GTONT,PRONT
          .ENT    SBF1,SBF2,SBF3,DNSTP,RSSTP,RDAEP,PROCP,PRCRP,GTONP,PRONP,OP1LV,O
          .NREL
          .TXTM   1
MSK4=     -123    ;VALUE 3777
MSK3=     -122    ;VALUE 174
OP1LV=    -121    ;LEVEL FOR OPERAND ONE
OP2LV=    -120    ;LEVEL FOR OPERAND TWO
SBF1=     -117
SBF2=     -116
SBF3=     -115
DNSTP=    -114
RSSTP=    -113
RDAEP=    -112
PROCP=    -111
PRCRP=    -110
GTONP=    -107
PRONP=    -106
C31=      -105
SVFLN=    -104
SUBT1=    -103
ZERA=     -102
ADDT=     -101
DV10=     -100
MMOV=     -77
MTEN=     -76
GCHAP=    -75
RTRNP=    -74
DEL=      -73
OPDSP=    -72
NMNCP=    -71
NMWCP=    -70
EXONP=    -67
RTER=     -66
RTN=      -65
TTOP=     -64
VALDP=    -63
OPAVP=    -62
ALCRP=    -61
PREPP=    -60
```

```
PACTM=  -57
PACT=   -56
PACK3=  -55
PACK2=  -54
PACK1=  -53
MSK2=   -52
C71=    -51
MSK=    -50
ELEVN=  -47     ;VALUE IS 11
RBMSK=  -46     ;VALUE IS 377
C100=   -45     ;VALUE IS 100
PERD=   -44     ;VALUE IS ".
SLSH=   -43     ;VALUE IS "/
EMSK1=  -42     ;VALUE IS 3000
EMSK2=  -41     ;VALUE IS 7400
EMSK3=  -40     ;VALUE IS 7
EMSK4=  -37     ;  VALUE IS 170
EMSK5=  -36     ;VALUE IS 200
C147=   -35     ;VALUE IS 147
NULT=   -34     ;VALUE IS 000
COL=    -33     ;VALUE IS ",
DB=     -32     ;VALUES IS DB
CR=     -31     ;VALUES IS CR
MINUS=  -30     ;VALUE IS "-
PLUS=   -27     ;VALUE IS "+
AST=    -26     ;VALUE IS "*
DOL=    -25     ;VALUE IS "$
TWLVE=  -24     ;VALUE IS 12
TEN=    -23     ;VALUE IS 10
NINE=   -22     ;VALUE IS 11
BLK=    -21     ;VALUE IS 40
SIX=    -20     ;VALUE IS 6
FIVE=   -17     ;VALUE IS 5
FOUR=   -16     ;VALUE IS 4
THREE=  -15     ;VALUE IS THREE
TWO=    -14     ;VALUE IS 2
ONE=    -13     ;VALUE IS 1
C60=    -12     ;VALUE IS 60
QMRK=   -11     ;VALUE IS "?
EOFM=   -10     ;VALUE IS 6
BELL=   -7      ;VALUE IS 7
FF=     -6      ;VALUE IS 14
LF=     -5      ;VALUE IS 12
C1=     -4      ;VALUE IS 15
TWO56=  -3      ;VALUE IS 256
ONE87=  -2      ;VALUE IS 400
MS256=  -1      ;VALUE IS 177400
DATAB:  .
PCTM:   .BLK    10
PCT:    .BLK    10      ;SECOND PACK FIELD
PCK3:   .BLK    10      ;THIRD PACK FIELD
PCK2:   .BLK    10      ;TEMPORARY PACK FIELD
PCK1:   .BLK    10      ;SECOND TEMPORARY PACK FIELD
TTOTP:  .TXT    /*TTO/
        3777
        174
        0
        0
        0
        0
        0
```

```
GDNST
GPSST
RDAEC
PROCT
PRCRL
GTONT
PRONT
31
PCTM*2
SUBT
ZEROA
ADDIT
DIV10
MOVA
MUL10
GCHAR
RTRN-1
177
OPDST
NMNCH
NMNCH
EXONT
ERTRN
RETRN
TTOTP*2
VALDG
OPAVC
ALCOR
PRERR
PCTM
PCT
PCK3
PCK2
PCK1
17
71
177
11
377
100
"
"/
3000
74000
7
170
200
147
000
",
41104
51103
"-
"+
"*
"$
12
10
11
40
6
```

```
        5
        4
        3
        2
        1
        60
        "?
        6
        7
        14
        12
        15
        256.
        400
        177400
DATA:   .BLK    25
        RTRN-1
        .BLK    40
        NDPTI-1
        CNTI-1
        SAVEI-1
        RTPTI-1
        .BLK    46
        .+2
        12.
        .BLK    36
RTRN:   .BLK    20
NDPTI:  .BLK    20
CNTI:   .BLK    20
SAVEI:  .BLK    20
RTPTI:  .BLK    20
IBUF=   0       ;POINTER TO INPUT BUFFER
DNPT1=  1       ;DATA NAME TABLE POINTERS
DNPT2=  2
DNPT3=  3
ECHT=   4       ;EDIT CHARACTER TEMPORARY
BTCNT=  5       ;BYTE COUNT IF ROUTINE
NFLAG=  6       ;NOT FLAG IF ROUTINE
CHKWD=  7       ;CHECK WORD
MM=     10      ;COUNTER MATH PACKAGE
BEFRL=  11      ;BEFORE LINE FLAG
SKPIT=  12      ;FLAG WITHIN WRITE ROUTINE
ERCDE=  13      ;ERROR CODE SAVE AREA
SFLAG=  14      ;SIZE ERROR FLAG
CNT2=   15      ;TOTAL LENGTH COUNTER
SIGN=   16      ;SIGN CLAUSE
X3T=    17      ;TEMPORARY COUNTERS
X4T=    20
X11T=   21
X17T=   22
SAVE1=  23      ;SAVE AREAS FOR RETURN ADDRESS
BYPTR=   24
RTST=   25
SAV1=   26      ;COUNTER
SAV=    27              ;COUNTER
RTRN1=  30              ;COUNTER
SAV2=   31              ;COUNTER
SAV3=   32              ;COUNTER
STAV=   34
NUM=    35      ;COUNTER
I=      37      ;WORD COUNTER MATH PACKAGE
```

```
N=      40
NT=     41
PNT1=   42      ;TEMPORARY BYTE POINTER'S FOR MOVE ROUTINE
PNT2=   43
ECHAR=  44      ;EDIT CHARACTER
RQCHK=  45      ;REQUEST CHECK IF ROUTINE
TAD=    46      ;TRUTH TABLE DISPLACEMENT
POS=    47
INDIC=  50      ;TYPE OF MOVE OPERATION
RCNT=   51      ;REPITITION COUNT
ECNT=   52      ;EDIT COUNT FLAG
CNT1=   53      ;COUNTER FOR SENDING FIELD LENGTH
EFLAG=  54      ;EDIT FLAG
RDGIT=  55      ;RECEIVING DIGITS
TYP=    56      ;TYPE OF SENDING FIELD
ECODE=  57      ;EDIT CODE
DIGIT=  60      ;NUMBER OF DIGITS TO RIGHT OF DECIMAL POINT
BEFRP=  61      ;BEFORE PAGE FLAG
CHANL=  62      ;CHANNEL NUMBER
QP=     63      ;QUINT POINTER
QC=     64      ;QUINT COUNT
SP=     65      ;STATE TABLE POINTER
NDPT=   66      ;POINTER TO EXIT PARAGRAPH
CNT=    67      ;NUMBER OF TIMES OF PERFORM
SAVE=   70      ;START PERFORM
RTPT=   71      ;RETURN POINTER FORM PERFORM
OP1DL=  72      ;OPERAND ONE LENGTH
OP1DP=  73      ;BYTE POINTER
OP1DE=  74      ;EDIT PICTURE POINTER
OP1DX=  75      ;OCCURS
OP1DT=  76      ;TYPE
DEC1=   77      ;DECIMAL POSITION
OP2DL=  100     ;OPERAND TWO POINTERS
OP2DP=  101
OP2DE=  102
OP2DX=  103
OP2DT=  104
DEC2=   105
OP3DL=  106     ;OPERAND THREE POINTERS
OP3DP=  107
OP3DE=  110
OP3DX=  111
OP3DT=  112
DEC3=   113
PAC1=   114     ;TEMPORARY BUFFER POINTERS
PAC2=   115
PAC3=   116
PACT1=  117
PACT2=  120
PACT3=  121
M=      122     ;WORD COUNTER
PACZ=   123     ;TEMPROARY BUFFER POINTER
PACTL=  124
OPDL=   125     ;RENAME AREAS FOR ARGUMENT POINTERS
OPDP=   126
OPDE=   127
OPDT=   130
SHRT=   131     ;SHORT FORMAT FLAG
TQC=    132     ;TEMPORARY QUINT COUNT
MPFLG=  133     ;MATH PACKAGE FLAG FOR GOTO
ARGDE=  134     ;TEMPORARY DECIMAL STORAGE AREA
```

```
CHAR1=  135     ;FIRST CHARACTER FROM ARGUMENT ONE
CHAR2=  136     ;FIRST CHARACTER FROM ARGUMENT TWO
TEMP=   137
IOPTR=  140
IOSTW=  141
IOCNT=  142
FDSTW=143
FDCNT=  144
EDSTW=  145
EDCNT=  146
DNSTW=  147
DNCNT=  150
RSSTW=  151
RSCNT=  152
QNSTW=  153
QNCNT=  154
SVFCN=  155
CERFL=  156
NEGNO=  160     ;NEGATIVE NUMBER FLAG
CHNUM=  161
NEWF=   162
SNMAX=  163
TEMP2=  164
OPFLG=  165
RSMX=   166
PNT0=   167     ;FLAG FOR ZERO OR BLANK FILL
CNT0=   170     ;NUMBER OF FILL CHARACTERS
IOST=   171
FDST=   172
EDST=   173
DNST=   174
RSST=   175
QNST=   176
STPFL=  177
 ;IBUF ARGUMENTS
IFCN=   -1
IO=     -2
STWD=   -3
ICT=    -4
IVEC=   -6
IER=    -7
        .END
```

```
        .TITLE  DECIMAL
        .ENT    FLDEC
        .NREL
FLDEC:  ISZ     RTST,2
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,DEC1,2        ;GET NUMBER OF DECIMAL POSITION
        MOV     1,1,SNR
        JSR     @RTN,2
        STA     1,SAV,2         ;SAVE IT
        LDA     1,PACK2,2
        STA     1,PACTL,2
        JSR     @MTEN,2
        JSR     @RTER,2
        DSZ     SAV,2
        JMP     .-5
        JSR     @RTN,2
```

```
        .TITL   DISPLAY
        .ENT    DSPLY
        .EXTD   GET
        .NREL
DSPLY:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;STORE RETURN ADDRESS
        LDA     1,OP1DP,2
        STA     1,OPDP,2
        LDA     1,OP1DL,2
        STA     1,OPDL,2
        LDA     3,OPDP,2    ;LOAD OPERAND BYTE POINTER
        STA     3,BYPTR,2       ;SAVE BYTE POINTER
        JSR     @GET        ;GET A CHARACTER
        JMP     @RTER,2
        MOV     1,0          ;MOVE CHARACTER TO AC0
        .SYSTEM
        .PCHAR
        JMP     @RTER,2
        MOV     3,2     ;RESTORE AC2
        ISZ     OPDP,2      ;INCREMENT OPERAND BYTE POINTER
        DSZ     OPDL,2      ;DECREMENT OPERAND LENGTH COUNTER
        JMP     DSPLY+6      ;RETURN TO PUT OUT NEXT CHARACTER
        LDA     0,C1,2       ;LOAD CARRIAGE RETURN
        .SYSTEM
        .PCHAR
        JMP     @RTER,2
        MOV     3,2     ;RESTORE AC2
        LDA     0,LF,2       ;LOAD LINE FEED CHARACTER
        .SYSTEM
        .PCHAR
        JMP     @RTER,2
        MOV     3,2     ;RESTORE AC2
        JMP     @RTN,2      ;PROCESSING COMPLETED
```

```
        .TITL   DIVIDE
        .ENT    DVDE,DVDPK
        .EXTN   MOZLE,MOLE
        .NREL
;CALL TO THIS ROUTINE IS (N,PACT1,PACT2,PACT3)
DVDPK:  LDA     1,TEN,2 ;GET AN EIGHT FOR EIGHT WORD DIVIDE
        STA     1,N,2   ;STORE IN N
        LDA     1,PACK1,2       ;INITILIZE ADDRESS'S FOR
        STA     1,PACT1,2       ;DIVIDE ROUTINE
        LDA     1,PACK2,2
        STA     1,PACT2,2
        LDA     1,PACK3,2
        STA     1,PACT3,2
DVDE:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,FOUR,2        ;GET A FOUR
        STA     1,X4T,2         ;INITILIZE A COUNTER
        LDA     1,N,2           ;GET NUMBER OF WORDS LONG
        MOVZL   1,1             ;MULTIPLY THE WORD
        DSZ     X4T,2           ;BY 16 TO GET PROPER
        JMP     .-2             ;VALUE FOR DIVISION
        STA     1,M,2           ;STORE FINAL VALUE IN M
        LDA     1,PACT3,2       ;ZERO OUT RECEIVING FIELD
        STA     1,PACZ,2
        JSR     @ZERA,2         ;ZERO OUT RECEIVING FIELD
        JSR     @RTER,2
        LDA     1,PACTM,2       ;ZERO OUT TEMP FIELD
        STA     1,PACZ,2
        JSR     @ZERA,2 ;GO TO ZERO-OUT ROUTINE
        JSR     @RTER,2
        LDA     1,PACT2,2   ;GET SECOND ARGUMENT
        STA     1,PAC1,2    ;FOR PROPER ADDRESS
        STA     1,PAC2,2    ;ALIGNMENT ON MOV LEFT ROUTINE
        JSR     @MOVLL      ;GO TO SHIFTLEFT ROUTINE
        JSR     @RTER,2
DVDE1:  LDA     1,PACT3,2   ;INITILIZE PACK3
        STA     1,PAC1,2    ;FOR PROPER SHIFT
        STA     1,PAC2,2    ;LEFT
        JSR     @MOVL1      ;JMP TO SHIFT LEFT ROUTINE
        JSR     @RTER,2
        SUBC    0,0
        MOVR    0,0         ;GENERATE A ONE
        STA     0,X4T,2     ;STORE IT
        LDA     1,PACT1,2   ;GET FIRST ARGUMENT
        STA     1,PAC1,2    ;FOR SUBTRACT ROUTINE
        LDA     1,PACT3,2   ;GET SECOND ARGUMENT
        STA     1,PAC2,2    ;FOR SUBTRACT ROUTINE
        LDA     1,PACTM,2
        STA     1,PAC3,2
        JSR     @SUBT1,2    ;GO TO SUBTRACT ROUTINE
        JSR     @RTER,2
        JSR     AA
        JSR     @RTER,2
        MOV     1,1,SZC     ;CHECK FOR OVERFLOW
        JMP     DVDE3       ;CARRY
        LDA     1,PACT1,2   ;GET FIRST ARGUMENT
        STA     1,PAC1,2    ;FOR ADD ROUTINE
        LDA     1,PACT3,2   ;SECOND ARGUMENT
        STA     1,PAC2,2
        LDA     1,PACTM,2
        STA     1,PAC3,2
```

```
        JSR     @ADDT,2         ;GO TO ADD ROUTINE
        JSR     @RTER,2
        JSR     AA
        JSR     @RTER,2
        LDA     1,X4T,2         ;GET FLAG FOR CARRY SET
        MOVL    1,1       ;STORE IT SINCE IT SHOULD BE A SUB#
DVDE3:  LDA     1,PACT2,2       ;GET SECOND ARGUMENT
        STA     1,PAC1,2        ;FOR PROPER ADDRESS
        STA     1,PAC2,2        ;ALIGNMENT ON MOVE LEFT ROUTINE
        JSR     @MOVL1          ;GO TO SHIFTLEFT ROUTINE
        JSR     @RTER,2
        DSZ     M,2
        JMP     DVDE1
        SUB     1,1
        STA     1,SFLAG,2
        JMP     @RTN,2       ;EXIT ROUTINE
AA:     LDA     1,PACTM,2       ;GET RECEIVING FIELD
        STA     1,PAC1,2        ;SET ADDRESS FOR
        LDA     1,PACT3,2       ;MATH-MOVE
        STA     1,PAC2,2        ;ROUTINE
        ISZ     RTST,2             ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        JSR     @MMOV,2         ;GO TO MATH-MOVE ROUTINE
        JSR     @RTER,2
        JMP     @RTN,2        ;RETURN
MOVLL:  MOZLF
MOVL1:  MOLF
```

```
        .TITL   DIVIDE10
        .ENT    DIV10
        .NREL
;THIS ROUTINE USES ONE PACK FIEL ANSWER COMES BACK IN WORD FIVE
;THE PACK FIELD TO BE USE BY THIS ROUTINE WILL BE PACTL
DIV10:  ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
;
;       * (1/2**4)
;
        LDA     1,PACTL,2       ;GET FIELD TO BE DIVIDED
        LDA     0,N,2           ;GET N - 1
        NEG     0,0
        COM     0,0
        STA     0,X4T,2
        ADD     0,1             ;TO GET FIFTH WORD
        STA     1,PAC3,2        ;STORE IT IN PAC3
        STA     1,X17T,2        ;PAC3+N-1
        STA     0,M,2          ;STORE IN COUNTER
        LDA     1,@PAC3,2       ;GET FIFTH WORD INFORMATION
        MOVZR   1,1
        MOVZR   1,1
        MOVZR   1,1
        MOVZR   1,1             ; / 16
AA:     DSZ     PAC3,2          ;TO GET FOURTH WORD
        LDA     0,@PAC3,2       ;LOAD CONTENTS
        ISZ     PAC3,2          ;PUT PAC3 TO FIFTH WORD
        SUBO    3,3             ;CLEA AC3
        MOVR    0,0             ;MOVE RIGHT FOUR TIMES
        MOVR    3,3
        MOVR    0,0
        MOVR    3,3
        MOVR    0,0             ;MOVE AC3 RIGHT FOUR TIMES
        MOVR    3,3
        MOVR    0,0             ; / 16
        MOVR    3,3             ;FRACTIONAL REMAINDER
        ADDZ    3,1,SZC         ; "TWO WORD" / 16
        INC     0,0
        STA     1,@PAC3,2       ;STORE IT BACKOUT
        DSZ     PAC3,2          ;GET NEXT ADDRESS
        MOV     0,1             ;STORE AC0 INTO AC1
        DSZ     M,2            ;DECREMENT COUNTER
        JMP     AA              ;DO AGAIN
        STA     1,@PAC3,2
;
;       * (1+1/2)
;
        LDA     1,X4T,2           ;GET N - 1
        LDA     0,X17T,2        ;GET PACK3+ N - 1
        STA     0,PAC3,2
        STA     1,M,2
        LDA     1,@PAC3,2       ;GET DATA
        SUB     3,3
AB:     DSZ     PAC3,2
        LDA     0,@PAC3,2       ;GET DATA
        ISZ     PAC3,2
        MOVZR   0,0
        MOVR    1,1             ; "TWO WORD" / 2
        ADDZ    3,1             ;ADD CARRY
        LDA     3,@PAC3,2       ;GET DATA
        ADD     3,1
```

```
        STA     1,@PAC3,2
        SUBCL   3,3             ;CARRY IN BIT 15
        DSZ     PAC3,2
        LDA     1,@PAC3,2
        DSZ     M,2             ;DECREMENT COUNTER
        JMP     AB
        STA     3,TEMP,2
        MOVZR   1,0
        ADDZ    0,1,SZC
        ISZ     TEMP,2
        STA     1,@PAC3,2
;
;       * (1+1/2**4)
;
        LDA     1,X4T,2           ;GET NUMBER OF NRODS
        STA     1,M,2
        LDA     0,X17T,2
        STA     0,PAC3,2
        LDA     1,@PAC3,2       ;GET DATA
        MOVZR    1,1             ;SHIFT RIGHT
        MOVZR    1,1
        MOVZR    1,1
        MOVZR    1,1             ; / 16
AC:     DSZ     PAC3,2
        LDA     0,@PAC3,2       ;GET DATA
        ISZ     PAC3,2
        SUBO    3,3             ;CLEAR AC3
        MOVR    0,0             ;SHIFT RIGHT AC0 THEN AC3
        MOVR    3,3
        MOVR    0,0
        MOVR    3,3
        MOVR    0,0
        MOVR    3,3
        MOVR    0,0             ; / 16
        MOVR    3,3             ;FRACTIONAL REMAINDER
        ADDZ    3,1,SZC
        INC     0,0
        LDA     3,@PAC3,2
        ADDZ    3,1,SZC
        INC     0,0
        STA     1,@PAC3,2
        DSZ     PAC3,2
        MOV     0,1
        DSZ     M,2             ;DECREMENT COUNTER
        JMP     AC
        LDA     3,@PAC3,2
        ADDZ    3,1,SZC
        ISZ     TEMP,2
        STA     1,@PAC3,2
;
;       * (1+1/2**8)
;
        LDA     0,X4T,2
        STA     0,M,2
        LDA     0,X17T,2
        STA     0,PAC3,2
        LDA     1,@PAC3,2
        MOVS    1,3
        LDA     0,RBMSK,2         ;MASK OF 377
        AND     3,0             ;SAVE RIGHT BYTE, HALF OF / 256
        SUB     3,3
```

```
        STA     3,TEMP,2
AD:     ADDZ    0,1,SZC
        ISZ     TEMP,2
        DSZ     PAC3,2
        LDA     0,@PAC3,2       ;GET WORD BEFORE
        ISZ     PAC3,2
        ADDZ    3,1,SZC
        ISZ     TEMP,2
        MOVS    0,0
        LDA     3,MS256,2       ;GET MASK 177400
        AND     0,3             ;SAVE LEFT BYTE. HALF OF / 256 FROM PREV WORD
        ADDZ    3,1,SZC
        ISZ     TEMP,2
        STA     1,@PAC3,2
        LDA     3,TEMP,2
        SUB     1,1
        STA     1,TEMP,2
        LDA     1,RBMSK,2       ;GET MASK OF 377
        AND     1,0             ; HALF OF / 256
        DSZ     PAC3,2
        LDA     1,@PAC3,2
        DSZ     M,2
        JMP     AD              ;DO AGAIN
        MOV     3,3,SZR
        ISZ     TEMP,2
        ADDZ    0,1,SZC
        ISZ     TEMP,2
        STA     1,@PAC3,2
;
;       * (1+1/2**16)(1+1/2**32)(1+1/2**64)
;
        LDA     1,X4T,2
        LDA     0,X17T,2
        STA     0,PAC3,2
        STA     1,M,2           ;INITIALIZE COUNTER
        LDA     3,TEN,2
AG:     LDA     1,M,2
        INC     1,1
        STA     1,NUM,2
        LDA     0,PAC3,2
        STA     0,X17T,2
        MOV     3,1
        SUB     3,3
AH:     LDA     0,@X17T,2
        ADDZ    0,1,SZC
        INC     3,3
        DSZ     X17T,2
        DSZ     NUM,2
        JMP     AH
        STA     1,@PAC3,2
        DSZ     PAC3,2
        DSZ     M,2
        JMP     AG              ;DO AGAIN
        LDA     0,TEMP,2
        SUB     1,1
        ADDZ    0,3,SZR
        SUBZL   1,1
        MOVR    1,1
        JSR     @RTN,2          ;RETURN
```

```
;
;       DUMMYENTRIES
;
        .NREL
        .ENT    CLG,LIST
        .EXTN   PPERP
CLG:    JSR     @PRERP
LIST:   JSR     @PRERP
PRERP:  PRERR
```

```
        .TITLE  DSKFL
        .ENT    DSKFL
        .NREL
;
;       INPUT ARGUMENTS IN IBUF (+USP)
;
;      0-255                    IO BUFFER
;       IFCN=-1                 ;1 READ, 2 WRITE, 3 SAVE
;       IO=-2                   ;CHANNEL #
;       STWD=-3                 ;STARTING BYTE IN DISK FILE
;       ICT=-4                  ;NUMBER OF BYTES TO PROCESS
;       IVEC=-6                 ;ADDR OF DATA VECTOR TO PROCESS
;       IER=-7                  ;ERROR FLAG
;
;       TEMPORARY DATA STORAGE
;
        PSECT=-10               ;PREVIOUS SECTOR # OF BUFFER
        WFLG=-11                ;0-CLEAN, 1-HAS BEEN UPDATED BUT NOT WRITTEN
        SECTR=-12               ;SECTOR # OF CURRENT DATA
        DISP=-13                ;DISP OF DATA WORD FROM LOOP COUNTER
        WORD=-14                ;WORD TO START LOOP ON
        LIM=-15                 ;WORD TO END LOOP ON
        CTR=-20
        FPTR=-21
        TPTR=-22
DSKFL:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     3,IBUF,2
        LDA     0,STWD,3
        NEG     0,0
        COM     0,0
        LDA     1,MS256,2
        AND     0,1             ;SECTR*256
        INC     0,0
        SUB     1,0
        NEG     0,0
        INC     0,0
        STA     0,DISP,3        ;SECTR*256-STWD+1
        MOVS    1,1
        STA     1,SECTR,3       ;(STWD-1)/256
        SUBZL   1,1
        STA     1,IER,3         ;IER=1 IF NO ERROR
        SUB     0,1
        STA     1,WORD,3        ;1-DISP
        LDA     0,TWO,2
        LDA     1,IFCN,3
        SUBZ    0,1,SNR         ;GO TO (100,200,300), IFCN
        JMP     D200            ;IFCN=2
        MOV     1,1,SNC
        JMP     D100            ;IFCN=1
        JMP     @D300P          ;IFCN=3
;
;       READ
;
D100:   LDA     0,SECTR,3
        LDA     1,PSECT,3
        SUB     0,1,SNR
        JMP     D120            ;CURRENT SECTOR
        LDA     0,WFLG,3
        MOV     0,0,SNR
        JMP     D110            ;PREV SECTOR WRITTEN
```

```
        LDA     1,PSECT,3
        JSR     @WRITP          ;WRITE PREV SECTOR
        JSR     @D500P
        LDA     3,IBUF,2
D110:   JSR     @READP
        JSR     @D500P
        LDA     3,IBUF,2
D120:   LDA     1,ICT,3
        LDA     0,WORD,3
        ADD     0,1
        ADC     0,0
        ADD     1,0
        STA     0,LIM,3         ;WORD+ICT-1
        LDA     1,TWO56,2
        SUBZ#   0,1,SZC         ;LIM LE 256?
        JMP     D130            ;YES
        SUB     1,0
        STA     0,ICT,3         ;LIM-256
        STA     1,LIM,3         ;256
        JMP     D140
D130:   SUB     0,0
        STA     0,ICT,3
D140:   LDA     0,WORD,3
        ADC     1,1
        ADD     1,0             ;WORD-1
        MOV     3,1
        ADD     0,1             ;IBUF(1)
        STA     1,FPTR,3
        LDA     1,IVEC,3
        ADD     0,1
        LDA     2,DISP,3
        ADD     1,2             ;IVEC(1+DISP)
        STA     2,TPTR,3
        LDA     2,USP
        LDA     1,LIM,3
        SUB     0,1             ;CTR FOR # TIMES THROUGH LOOP
        STA     1,CTR,3
D150:   LDA     0,@FPTR,3
        STA     0,@TPTR,3       ;IVEC(I+DISP)=IBUF(I)
        ISZ     FPTR,3
        ISZ     TPTR,3
        DSZ     CTR,3
        JMP     D150
        LDA     0,ICT,3
        MOV     0,0,SNR
        JMP     D180
        LDA     0,DISP,3
        MOV     0,0,SNR
        JMP     .+3
        MOVZL   0,0,SNC
        JMP     D170            ;DISP GT 0
        LDA     0,LIM,3
        LDA     1,WORD,3
        SUB     1,0
        INC     0,0
        STA     0,DISP,3        ;LIM-WORD/1
D160:   SUBZL   0,0
        STA     0,WORD,3        ;1
        ISZ     SECTR,3
        JSR     READ
        JSR     @D500P
```

```
        LDA     3,IBUF,2
        LDA     0,ICT,3
        LDA     1,TWO56,2
        SUBZ#   0,1,SZC         ;LIM LE 256?
        JMP     D120            ;YES
        STA     1,LIM,3
        SUB     1,0
        STA     0,ICT,3         ;ICT-256
        JMP     D140
D170:   LDA     0,DISP,3
        LDA     1,TWO56,2
        ADD     1,0
        STA     0,DISP,3        ;DISP+256
        JMP     D160
D180:   LDA     0,SECTR,3
        LDA     1,PSECT,3
        SUB     0,1,SNR
        JSR     @RTN,2
        STA     0,PSECT,3
        SUB     1,1
        STA     1,WFLG,3
        JSR     @RTN,2
D300P:  D300
D500P:  D500
WRITP:  WRITE
READP:  READ
;
;       WRITE
;
D200:   LDA     0,SECTR,3
        LDA     1,PSECT,3
        SUB     0,1,SNR
        JMP     D220            ;CURRENT SECTR
        LDA     0,WFLG,3
        MOV     0,0,SNR
        JMP     D210            ;PREV SECTOR WRITTEN
        LDA     1,PSECT,3
        JSR     WRITE           ;WRITE PREV SECTOR
        JSR     @D500P
        LDA     3,IBUF,2
D210:   JSR     READ
        JSR     @D500P
        LDA     3,IBUF,2
D220:   LDA     1,ICT,3
        LDA     0,WORD,3
        ADD     0,1
        ADC     0,0
        ADD     1,0
        STA     0,LIM,3         ;WORD+ICT-1
        LDA     1,TWO56,2
        SUBZ#   0,1,SZC         ;LIM LE 256?
        JMP     D130            ;YES
        SUB     1,0
        STA     0,ICT,3         ;LIM-256
        STA     1,LIM,3         ;256
        JMP     D240
D230:   SUB     0,0
        STA     0,ICT,3
D240:   LDA     0,WORD,3
        ADC     1,1
        ADD     1,0             ;WORD-1
```

```
        MOV     3,1
        ADD     0,1             ;IBUF(1)
        STA     1,TPTR,3
        LDA     1,IVEC,3
        ADD     0,1
        LDA     2,DISP,3
        ADD     1,2             ;IVEC(1+DISP)
        STA     2,FPTR,3
        LDA     2,USP
        LDA     1,LIM,3
        SUB     0,1             ;CTR FOR # TIMES THROUGH LOOP
        STA     1,CTR,3
D250:   LDA     0,@FPTR,3
        STA     0,@TPTR,3
        ISZ     FPTR,3
        ISZ     TPTR,3
        DSZ     CTR,3
        JMP     D250
        LDA     0,ICT,3
        MOV     0,0,SNR
        JMP     D280
        LDA     0,DISP,3
        MOV     0,0,SNR
        JMP     .+3
        MOVZL   0,0,SNC
        JMP     D270
        LDA     0,LIM,3
        LDA     1,WORD,3
        SUB     1,0
        INC     0,0
        STA     0,DISP,3        ;LIM-WORD\1
D260:   SUBZL   0,0
        STA     0,WORD,3        ;1
        LDA     1,SECTR,3
        JSR     WRITE           ;WRITE CURRENT SECTOR
        JSR     @D500P
        LDA     3,IBUF,2
        ISZ     SECTR,3
        JSR     READ            ;READ NEXT SECTOR
        JSR     @D500P
        LDA     3,IBUF,2
        LDA     0,ICT,3
        LDA     1,TWO56,2
        SUBZ#   0,1,SZC         ;LIM LE 256?
        JMP     D220            ;YES
        STA     1,LIM,3
        SUB     1,0
        STA     0,ICT,3         ;ICT-256
        JMP     D240
D270:   LDA     0,DISP,3
        LDA     1,TWO56,2
        ADD     1,0
        STA     0,DISP,3        ;DISP+256
        JMP     D260
D280:   LDA     0,SECTR,3
        STA     0,PSECT,3
        SUB     0,0
        STA     0,WFLG,3
        JSR     @RTN,2
D300:   LDA     1,PSECT,3
        JSR     WRITE           ;WRITE PREVIOUS SECTOR
```

```
        JSR     D500
        LDA     3,IBUF,2
        SUB     0,0
        STA     0,WFLG,3
        JSR     @RTN,2
READ:   ISZ     RTST,2
        STA     3,@RTST,2
        LDA     3,IBUF,2
        MOV     3,0
        LDA     1,SECTP,3
        LDA     2,ONEB7,2       ;1 BLOCK (LFT BYTE)
        LDA     3,IO,3          ;CHANL RT BYTE
        ADD     3,2
        .SYSTM
        .RDB    77              ;READ 1 BLOCK
        JSR     @RTER,3
        LDA     2,USP
        JSR     @RTN,2
WRITE:  ISZ     RTST,2          ;AC1 = SECTOR #
        STA     3,@RTST,2
        LDA     3,IBUF,2
        MOV     3,0
        LDA     2,ONEB7,2       ;1 BLOCK LFT BYTE
        LDA     3,IO,3          ;CHANL RT BYTE
        ADD     3,2
        .SYSTM
        .WRB    77              ;WRITE 1 BLOCK
        JSR     @RTER,3
        LDA     2,USP
        JSR     @RTN,2
D500:   LDA     3,IBUF,2
        STA     0,IER,3
        SUB     0,0
        STA     0,WFLG,3        ;ZERO WFLG & SECTR TO
        STA     0,SECTR,3       ;INVALIDATE THE DATA
        JSR     @RTER,2
```

```
;
;       INITIALIZE SAVE FILE AND CORE
;
        .TITLE  ENTER
        .NREL
        .TXTM   1
        .ENT    ENTER
        .EXTN   RDSVF,INSVF
ENTER:   ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GCHAP,2
        JSR     @RTER,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        LDA     1,C1,2
        SUB     0,1,SZR         ;CARRIAGE RETURN?
        JMP     ENT1            ;NO
        LDA     0,LF,2
        .SYSTM
        .PCHAR                  ;PRINT LINE FEED
        JSR     @RTER,3
ENT1:   JSR     @RDSVP          ;GET THE SAVE FILE NAME TO EXECUTE
        JSR     @RTER,2
        LDA     0,SVFLN,2       ;BYTE PTR TO SAVE FILE NAME (AC3 IS USP)
        JSR     @OPAVP,2        ;OPEN CHANNEL
        JSR     @RTER,2
        STA     1,SVFCN,2       ;SAVE CHANNEL #
        JSR     @ALCRP,2          ;ALLOCATE CORE
        JSR     @RTER,2
        JSR     @INSVP          ;INITIALIZE THE SIX SAVE FILES
        JSR     @RTER,2
        JSR     @RTN,2
RDSVP:  RDSVF
INSVP:  INSVF
```

```
        .TITL   EXIT
        .ENT    EXIT1
        .NREL
EXIT1:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,@NDPT,2       ;LOAD AC1 WITH THE END STATEMENT
        LDA     3,QC,2          ;LOAD AC3 WITH THE QC
        SUB     3,1,SZR         ;COMPARE
        JSR     @RTN,2          ;NOT EQUAL, RETURN
EXIT2:  DSZ     @CNT,2  ;DECREMENT COUNTER
        JMP     EXIT3           ;NOT ZERO, JUMP TO EXIT3
        SUB     3,3             ;CLEAR AC3 TO ZERO
        STA     3,@NDPT,2       ;STORE 0 IN NDPT
        STA     3,@SAVE,2       ;STORE 0 IN SAVE
        STA     3,@CNT,2
        LDA     1,@RTPT,2       ;LOAD AC1 WITH RETURN STATEMENT
        STA     1,TQC,2         ;STORE RETURN STATEMENT IN TQC
        STA     3,@RTPT,2       ;STORE 0 IN RTPT
        DSZ     CNT,2
        DSZ     NDPT,2
        DSZ     SAVE,2
        DSZ     RTPT,2
        JSR     @RTN,2
EXIT3:  LDA     3,@SAVE,2       ;LOAD AC3 WITH BEGINNING PT.
        STA     3,TQC,2         ;STORE BEGINNING PT. IN TQC
        JSR     @RTN,2          ;PROCESSING COMPLETE, RETURN
```

```
;
;       EXECUTE QUINT LOCATED IN OP
;
        .TITLE  EXECUTEQUINT

        .NREL
        .TXTM   1
        .ENT    EXQNT,PRQNT
        .EXTN   BTL

EXQNT:  ISZ     RTST,2
        STA     3,@RTST,2
EXQ1:   LDA     3,@SP,2
        MOVL#   3,3,SZC         ;ADDRESS?
        JMP     EXQ2            ;YES
        JSR     0,3             ;EXECUTE STATE ROUTINE
        JSR     @RTER,2
        ISZ     SP,2            ;GET NEXT STATE
        JMP     EXQ1
EXQ2:   MOVZL   3,3,SZR         ;BRANCH?
        JMP     EXQ3            ;YES
        LDA     0,TQC,2         ;GET NEXT QUINT
        MOV     0,0,SZR
        JMP     EXQ4
        ISZ     QC,2
        JMP     EXQ4+1
EXQ3:   MOVZR   3,3     ;STRIP OFF B0
        STA     3,SP,2          ;NEW STATE POINTER
        JMP     EXQ1
EXQ4:   STA     0,QC,2
        SUB     0,0
        STA     0,TQC,2
        LDA     3,BTLP
        LDA     0,0,3
        STA     0,TEMP,2        ;# BREAK POINTS
        INC     3,3             ;ADDR BREAK TABLE
        INC     3,3
        LDA     0,QC,2          ;QUINT #
EXQ5:   LDA     1,-1,3           ;BREAK QUINT
        MOV     1,1,SNR
        JMP     .+3
        SUB     0,1,SNR
        JMP     EXQ6
        INC     3,3
        DSZ     TEMP,2
        JMP     EXQ5
        JSR     @RTN,2          ;NOT BREAK POINT
EXQ6:   ISZ     STPFL,2         ;BREAK POINT
        JSR     @PRQNP,2
        JSR     @RTER,2
        JSR     @RTN,2
PRQNT:  ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GTQNP,2
        JSR     @RTER,2
        LDA     0,0
        .SYSTM
        .PCHAR
        JSR     @RTER,2         ;PRINT 0
        LDA     0,AAA           ;     #
        .SYSTM
```

```
        .PCHAR
        JSR     @RTER,2
        LDA     0,QC,2
        JSR     @PROCP,2        ;PRINT QUINT #
        JSR     @RTER,2
        LDA     0,FIVE,2
        STA     0,SAV3,2
EX07:   LDA     3,QP,2
        LDA     0,0,3
        JSR     @PROCP,2        ;PRINT QUINT
        JSR     @RTER,2
        ISZ     QP,2
        DSZ     SAV3,2          ;DONE ?
        JMP     EX07            ;NO
        JSR     @PRCRP,2        ;PRINT CR,LF
        JSR     @RTER,2
        JSR     @RTN,2
BTLP:   BTL
        .TXTM   0
Q:      .TXT    /Q/
AAA:    .TXT    /#/
```

```
. EXTD    IBUF, MS256, ONE87, TWO56, DNPT1, DNPT3, DNPT2, CR, LF, FF, BELL, EOFM
. EXTD    QMRK, ECHT, RTCNT, BLK, C60, ONE, TWO, THREE, FOUR, FIVE, SIX, NINE, TEN
. EXTD    TWLVE, NFLAG, CHKWD, DOL, AST, PLUS, MINUS, C1, DB, MM, COL, BEERL, SKPIT
. EXTD    ERCDE, SFLAG, CNT2, SIGN, X3T, X4T, X11T, X17T, SAVE1, NUM, I, N, NT, BYPTR
. EXTD    PNT2, ECHAR, RDCHK, TAD, POS, INDIC, PNT1, RCNT, ECNT, CNT1, EFLAG
. EXTD    SAV, RDGIT, TYP, SAV1, PACTM, ECODE, DIGIT, PERRP, CHANL, OP, QC, SP, RTST
. EXTD    NDPT, CNT, SAVE, RTPT, OP1DL, OP1DP, OP1DT, OP1DE, OP1DX, OP2DL
. EXTD    OP2DP, OP2DE, OP2DX, OP2DT, OP3DL, OP3DP, OP3DE, OP3DX, OP3DT, RTRN1
. EXTD    PACK1, PACK2, PACK3, PACT, PAC1, PAC2, PAC3, PACT1, PACT2, PACT3, M, SAV2
. EXTD    PACTL, SAV3, PAC2, OPDT, OPDL, OPDP, OPDE, DEC1, DEC2, DEC3, NULT, C147
. EXTD    EMSK5, EMSK4, EMSK3, EMSK2, EMSK1, SHRT, STAV, NEGNO, SLSH, PERD, C100
. EXTD    RBMSK, TOC, MPFLG, ARGDE, CHAR1, CHAR2, TEMP, ONST, RSCNT, RSST, DNCNT
. EXTD    DNST, EDCNT, EDST, EDCNT, EDST, IOCNT, IOST, ONCNT, IOPTR, SVECN
. EXTD    ELEVN, SVELN, MSK, MSK2, C71, RSMX, OPFLG, DATAB, RTRN, CERFL
. EXTD    TTOP, CHNUM, VALDP, RTN, RTEP, NENF, SNMAX, OPDSP, RTRNP, DEL, PNT0, CNT0
. EXTD    TEMP2, ALCRP, PRERP, IECN, IO, STWD, ICT, IVEC, IER, IOSTN, EDSTN, EDSTW
. EXTD    NMNCP, NMNCP, EXONP, OPAVP, DNSTN, PSSTN, ONSTN, GCHAP, STPFL
. EXTD    SUBT1, ZERA, ADDT, DV10, MMOV, MTEN, C31, MSK4, MSK3
. EXTN    DATA
. EXTD    SBF1, SBF2, SBF3, DNSTP, RSSTP, RDAEP, PROCP, PRCRP, GTONP, PRONP, OP1LV, O
. END
```

```
        .TITL   GETARGUMENTS
        .ENT    OP1D,OP2D,OP3D,OP4D,OP5D,OP6D
        .EXTN   OP1DS,OP2DS,OP3DS,OP1DR,OP2DR,OP3DR
        .EXTN   RETRN,GDNST,GFDST,GIOST,ERTRN
        .NREL
OP1D:   ISZ     RTST,2  ;SAVE RETURN ADDRESS
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,OP,2          ;LOAD QUINT POINTER
        LDA     0,2,3           ;GET FIRST ARGUMENT
        LDA     1,SBF1,2        ;LOAD SUBSCRIPT FLAG
        MOV     1,1,SZR         ;IS FLAG = 0
        JMP     NEW1            ;NO
        MOV     0,0,SZR         ;YES, CHECK IF OPER. = 0
        JMP     .+11            ;NO
        LDA     1,1,3           ;YES, LOAD OPCODE
        MOVZR   1,1
        MOVZR   1,1
        MOVZR   1,1
        LDA     0,C31,2         ;LOAD OPCODE 25
        SUB     1,0,SZR         ;CHECK IF EQUAL
        JSR     @RTER,2         ;NO, ERROR RETURN
        JSR     @RTN,2          ;YES, NORMAL RETURN
        JSR     @DNSTP,2
        JSR     @RTER,2
        MOV     1,3
        LDA     0,0,3           ;LOAD AC3 TO AC0
        STA     0,SAV1,2
        JSR     @RSSTP,2
        JSR     @RTER,2
        STA     0,OP1DP,2       ;GET BYTE POINTER
        MOV     1,3
        LDA     0,1,3
        STA     0,OP1DT,2       ;GET TYPE
        LDA     0,2,3
        STA     0,OP1DL,2       ;GET LENGTH
        LDA     0,3,3
        LDA     1,MSK4,2        ;LOAD MASK 377
        AND     1,0             ;SAVE RIGHT BYTE
        STA     0,OP1DX,2       ;GET NO. OF OCCUR.
        LDA     0,3,3
        MOVS    0,0             ;SWAP BYTES
        LDA     1,MSK3,2
        AND     1,0             ;SAVE RIGHT BYTE
        MOVZR   0,0
        MOVZR   0,0
        STA     0,DEC1,2        ;GET NO. DECIMAL POSITIONS
        JMP     @RTN,2      ;PROCESSING COMPLETE. RETURN
NEW1:   ISZ     SP,2            ;INCREMENT STATE POINTER
        LDA     1,SP,2          ;LOAD STATE PTR.
        SUBZR   0,0             ;SET BIT0
        ADD     0,1             ;SET BIT0
        STA     1,@DP1  ;STORE IN OP1DR
        LDA     1,DS1           ;LOAD DUMMY STATE TABLE
        STA     1,SP,2          ;STORE IN STATE PTR.
        JSR     @RTN,2          ;RETURN
OP2D:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,OP,2          ;LOAD QUINT POINTER
        LDA     0,3,3           ;GET SECOND ARGUMENT
        LDA     1,SBF2,2        ;LOAD SUBSCRIPT FLAG
        MOV     1,1,SZR         ;IS FLAG = 0
```

```
        JMP     NEW2    ;NO
        MOV     0,0,SZR         ;YES, CHECK IF OPER. = 0
        JMP     .+11            ;NO
        LDA     1,1,3           ;LOAD OPCODE
        MOVZR   1,1
        MOVZR   1,1
        MOVZR   1,1
        LDA     0,C31,2 ;LOAD OPCODE 25
        SUB     1,0,SZR         ;CHECK IF EQUAL
        JSR     @RTER,2         ;NO, ERROR RETURN
        JSR     @RTN,2          ;YES, NORMAL RETURN
        JSR     @DNSTP,2
        JSR     @RTER,2
        MOV     1,3
        LDA     0,0,3         ;LOAD AC3 TO AC0
        STA     0,SAV2,2
        JSR     @RSSTP,2
        JSR     @RTER,2
        STA     0,OP2DP,2     ;GET BYTE POINTER
        MOV     1,3
        LDA     0,1,3
        STA     0,OP2DT,2     ;GET TYPE
        LDA     0,2,3
        STA     0,OP2DL,2     ;GET LENGTH
        LDA     0,3,3
        LDA     1,MSK4,2        ;LOAD MASK 377
        AND     1,0             ;SAVE RIGHT BYTE
        STA     0,OP2DX,2     ;GET NO. OF OCCUR.
        LDA     0,3,3
        MOVS    0,0             ;SWAP BYTES
        LDA     1,MSK3,2
        AND     1,0             ;SAVE RIGHT BYTE
        MOVZR   0,0
        MOVZR   0,0
        STA     0,DEC2,2        ;GET NO. DECIMAL POSITIONS
        LDA     0,4,3
        LDA     1,EDST,2        ;GET BEGINNING OF EDIT TABLE
        ADD     1,0      ;ADD DISPLACEMENT
        MOVZL   0,0      ;CONVERT TO BYTE POINTER
        STA     0,OP2DE,2     ;GET EDIT POINTER
        JMP     @RTN,2      ;PROCESSING COMPLETE, RETURN
NEW2:   ISZ     SP,2            ;INCREMENT STATE PTR.
        LDA     1,SP,2          ;LOAD STATE PTR.
        SUBZR   0,0             ;SET BIT0
        ADD     0,1             ;SET BIT0
        STA     1,@DP2  ;STORE IN OP2DP
        LDA     1,DS2           ;LOAD DUMMY STATE TABLE
        STA     1,SP,2          ;STORE IN SP
        JSR     @RTN,2          ;RETURN
OP3D:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2     ;SAVE RETURN ADDRESS
        LDA     3,DP,2        ;LOAD QUINT POINTER
        LDA     0,4,3         ;GET THIRD ARGUMENT
        LDA     1,SBF3,2        ;LOAD SUBSCRIPT FLAG
        MOV     1,1,SZR
        JMP     NEW3
        MOV     0,0,SZR
        JMP     .+11
        LDA     1,1,3
        MOVZR   1,1
        MOVZR   1,1
```

```
        MOVZR   1,1
        LDA     0,C31,2
        SUB     1,0,SZR
        JSR     @RTER,2
        JSR     @RTN,2
        JSR     @DNSTP,2
        JSR     @RTER,2
        MOV     1,3
        LDA     0,0,3           ;LOAD AC3 TO AC0
        STA     0,SAV3,2
        JSR     @RSSTP,2
        JSR     @RTER,2
        STA     0,OP3DP,2       ;GET BYTE POINTER
        MOV     1,3
        LDA     0,1,3
        STA     0,OP3DT,2       ;GET TYPE
        LDA     0,2,3
        STA     0,OP3DL,2       ;GET LENGTH
        LDA     0,3,3
        LDA     1,MSK4,2            ;LOAD MASK 377
        AND     1,0                 ;SAVE RIGHT BYTE
        STA     0,OP3DX,2       ;GET NO. OF OCCUR.
        LDA     0,3,3
        MOVS    0,0                 ;SWAP BYTES
        LDA     1,MSK3,2
        AND     1,0                 ;SAVE RIGHT BYTE
        MOVZR   0,0
        MOVZR   0,0
        STA     0,DEC3,2            ;GET NO. DECIMAL POSITIONS
        JMP     @RTN,2      ;PROCESSING COMPLETE, RETURN
NEW3:   ISZ     SP,2
        LDA     1,SP,2
        SUBZR   0,0
        ADD     0,1
        STA     1,@DP3
        LDA     1,DS3
        STA     1,SP,2
        JSR     @RTN,2
DS1:    .GADD   OP1DS,-1
DS2:    .GADD   OP2DS,-1
DS3:    .GADD   OP3DS,-1
DP1:    OP1DR
DP2:    OP2DR
DP3:    OP3DR
OP4D:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,QP,2          ;LOAD QUINT POINTER
        LDA     1,4,3           ;GET THE THIRD ARGUMENT
        STA     1,TOC,2         ;STORE IN TEMP  QUINT CTR
        JMP     @RTN,2      ;PROCESSING COMPLETE, RETURN
OP5D:   ISZ     RTST,2  ;GET NEXT STORE POSTION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,QP,2          ;LOAD QUINT POINTER
        LDA     0,2,3           ;GET ARGUMENT 1
        JSR     @FDST1
        JSR     @RTER,2
        MOV     1,3
        LDA     0,1,3           ;LOAD AC3 TO AC0
        STA     3,PNT1,2
        JSR     @DNSTP,2
        JSR     @RTER,2
```

```
        MOV     1,3
        LDA     0,0,3
        JSR     @RSSTP,2
        JSR     @RTER,2
        STA     0,OP1DP,2       ;GET BYTE POINTER
        MOV     1,3
        LDA     0,2,3    ;GET THE LENGTH
        STA     0,OP1DL,2
        LDA     3,PNT1,2
        LDA 0,2,3
        JSR     @IOST1
        JSR     @RTER,2
        MOV     1,3
        LDA     0,5,3           ;PICK-UP CHANNEL NO.
        STA     0,CHANL,2       ;STORE IN DATA-BASE
        JMP     @RTN,2       ;PROCESSING COMPLETE, RETURN
OP6D:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,OP,2          ;LOAD QUINT POINTER
        LDA     0,2,3           ;GET FIRST ARGUMENT
        JSR     @FDST1
        JSR     @RTER,2
        MOV     1,3
        LDA     0,2,3
        JSR     @IOST1
        JSR     @RTER,2
        MOV     1,3
        LDA     0,5,3           ;PICK-UP CHANNEL NO.
        STA     0,CHANL,2       ;STORE IN DATA-BASE
        MOVZL   3,3
        STA     3,OP1DP,2       ;GET I/O BYTE POINTER
        JMP     @RTN,2       ;PROCESSING COMPLETE, RETURN
FDST1:  GFDST
IOST1:  GIOST
```

```
        .TITL   GETEDIT
        .ENT    EDIT1
        .EXTD   GET
        .NREL
EDIT1:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,SHRT,2        ;CHECK TO SEE WHICH FORMAT TO GO TO
        MOVR    1,1,SZC ;NOT SET TEST TO SEE WHICH FORMAT
        JMP     SHORT   ;LAST EDIT CODE WAS IN SHORT FORMAT
        LDA     3,OP2DE,2       ;GET EDIT FORMAT
        MOVZR   3,3
        STA     3,OPDE,2        ;STORE IN TEMP FIELD
        LDA     3,@OPDE,2       ;GET EDIT FORMAT
        MOVL#   3,3,SNC ;TEST TO SEE IF FORMAT IS LONG OR SHORT
        JMP     SHORT   ;NOT ON SHORT FORMAT
        LDA     3,@OPDE,2       ;GET EDIT FORMAT
        MOV     3,1     ;MOVE TO AC1
        LDA     0,EMSK1,2.      ;GET EDIT MASK 3000
        AND     0,1     ;TO GET REPITION COUNT
        STA     1,RCNT,2        ;STORE REPITION COUNT
        LDA     1,EMSK2,2       ;GET EDIT MASK 74000
        AND     1,3     ;TO GET EDIT CODE
        LDA     1,NINE,2        ;GET A ELEVEN
        STA     1,X11T,2        ;STORE IN COUNTER
        MOVR    3,3     ;SHIFT EDIT CODE TO RIGHT MOST POSITION
        DSZ     X11T,2  ;DECREMENT COUNTER
        JMP     .-2     ;DO TILL DONE
        STA     3,ECODE,2       ;STORE IN EDIT CODE
        ISZ     OP2DE,2
        JMP     @RTN,2
SHORT:  LDA     3,OP2DE,2       ;GET EDIT FORMAT
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET    ;GET A BYTE ROUTINE
        JMP     @RTER,2
        MOV     1,3     ;MOVE EDIT FORMAT TO AC3
        LDA     0,EMSK5,2       ;GET EDIT MASK 200
        AND     0,1     ;TO CHECK WHAT FORMAT THIS IS
        LDA     0,BELL,2        ;GET A SEVEN
        STA     0,X11T,2        ;STORE IN A COUNTER
        MOVR    1,1     ;SHIFT TO RIGHT MOST POSITION
        DSZ     X11T,2  ;DECREMENT COUNTER
        JMP     .-2
        MOVR    1,1,SNC ;TEST TO SEE IT ON
        JMP     .+3     ;NOT ON
        ISZ     OP2DE,2 ;INCREMENT EDIT FORMAT BYTE POINTER
        JMP     EDIT1+5 ;GET NEXT EDIT FORMAT
        MOV     3,1     ;GET THE EDIT FORMAT
        LDA     0,EMSK3,2       ;GET EDIT MASK 7
        AND     0,1     ;TO GET REPITION COUNT
        STA     1,RCNT,2        ;STORE IN REPITION COUNT
        ISZ     RCNT,2
        LDA     1,EMSK4,2       ;GET EDIT MASK 170
        AND     1,3     ;TO GET EDIT CODE
        LDA     1,THREE,2       ;GET A THREE FOR A COUNTER
        STA     1,X3T,2 ;STORE IN COUNTER
        MOVR    3,3     ;SHIFT EDIT CODE TO RIGHT MOST POSITION
        DSZ     X3T,2   ;DECREMENT COUNTER
        JMP     .-2
        STA     3,ECODE,2       ;STORE IN EDIT CODE
        SUBZL   1,1     ;GENERATE A ONE
        STA     1,SHRT,2        ;STORE IN SHORT FORMAT CODE TO RETURN TO HERE
```

```
ISZ     OP2DE,2
JMP     @RTN,2  ;RETURN
```

```
        .TITLE  GETTABLES
        .ENT    GDNST,GFDST,GIOST,GRSST
        .NREL
GDNST:  ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        NEG     0,0
        COM     0,0
        MOV     0,1     ;MULTIPLY BY FIVE
        ADDZL   0,0
        ADD     1,0
        LDA     1,DNST,2        ;LOAD DATA NAME TABLE PTR.
        ADD     0,1             ;GET DATA NAME ENTRY
        JSR     @RTN,2
GFDST:  ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        NEG     0,0
        COM     0,0
        MOV     0,1     ;MULTIPLY BY FIVE
        ADDZL   0,0
        ADD     1,0
        LDA     1,FDST,2        ;LOAD FD TABLE PTR.
        ADD     0,1             ;GET FD ENTRY
        JSR     @RTN,2
GIOST:  ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        NEG     0,0
        COM     0,0
        MOVZL   0,1     ;MULTIPLY BY SIX
        ADDZL   1,0
        LDA     1,IOST,2        ;LOAD I/O TABLE PTR.
        ADD     0,1             ;GET I/O ENTRY
        JSR     @RTN,2
GRSST:  ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,RSST,2        ;LOAD RAW STORAGE WORD PTR.
        MOVZL   3,3             ;CONVERT TO BYTE PTR.
        ADD     3,0             ;ADD DATA-NAME DISPLACEMENT
        JSR     @RTN,2
```

```
        .TITL   GOTO
        .ENT    GOTO1
        .NREL
GOTO1:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,SFLAG,2       ;LOAD SIZE ERROR FLAG
        MOVL    1,1,SZC         ;SEE IF SET
        JMP     GOTO2           ;YES
        LDA     1,MPFLG,2       ;NO, LOAD MATH-PACK FLAG
        MOVR    1,1,SZC         ;SEE IF SET
        JMP     GOTO3           ;YES
        JMP     @RTN,2          ;NO
GOTO2:  LDA     1,MPFLG,2       ;LOAD IN MATH-PACK FLAG
        MOVR    1,1,SZC         ;CHECK TO SEE IF SET
        JMP     @RTN,2          ;YES, DO THE GOTO
GOTO3:  SUB     1,1             ;CLEAR AC1
        STA     1,TQC,2         ;STORE IN TEMP. QUINT CTR.
        JMP     @RTN,2          ;PROCESSING COMPLETE, RETURN
```

```
        .TITL   IF
        .ENT    ALPHA,NUMRC,LESS,GREAT,EQUAL
        .EXTN   PLAT1
        .EXTD   GET
        .NREL
LESS:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,ONE,2         ;LOAD A ONE
        STA     1,ROCHK,2       ;STORE IT IN ROCHK
        LDA     1,OPFLG,2       ;LOAD OPCODE FLAG
        MOVR    1,1,SZC         ;CHECK STATUS
        JMP     FLG01           ;ON, NLESS
        JMP     FLG00           ;SET NFLAG TO 0
GREAT:  ISZ     RTST,2  ;GET NXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,TWO,2         ;LOAD A TWO
        STA     1,ROCHK,2       ;STORE IT IN ROCHK
        LDA     1,OPFLG,2.      ;LOAD OPCODE FLAG
        MOVR    1,1,SZC ;CHECK STATUS
        JMP     FLG01           ;ON, NGREATER
        JMP     FLG00           ;SET NFLAG TO 0
EQUAL:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,THREE,2       ;LOAD A THREE
        STA     1,ROCHK,2       ;STORE IT IN ROCHK
        LDA     1,OPFLG,2       ;LOAD OPCODE FLAG
        MOVR    1,1,SZC         ;CHECK STATUS
        JMP     FLG01           ;ON, NEQUAL
        JMP     FLG00           ;SET NFLAG TO 0
FLG01:  LDA     1,ONE,2         ;LOAD A ONE
        STA     1,NFLAG,2       ;STORE IT IN NFLAG
        JMP     @RELAT          ;PROCESS RELATIONAL TEST
FLG00:  SUB     1,1     ;CLEAR AC1 TO ZERO
        STA     1,NFLAG,2       ;STORE IT IN NFLAG
        JMP     @RELAT          ;PROCESS RELATIONAL TEST
ALPHA:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,TWO,2         ;LOAD A TWO
        STA     1,ROCHK,2       ;STORE IT IN ROCHK
        LDA     1,OPFLG,2       ;LOAD OPCODE FLAG
        MOVR    1,1,SNC         ;CHECK STATUS
        JMP     FLAG0           ;ON, NALPHA
        JMP     FLAG1           ;SET NFLAG TO -1
NUMRC:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,THREE,2       ;LOAD A THREE
        STA     1,ROCHK,2       ;STORE IT IN ROCHK
        LDA     1,OPFLG,2       ;LOAD OPCODE FLAG
        MOVR    1,1,SNC         ;CHECK STATUS
        JMP     FLAG0           ;ON, NNUMERIC
        JMP     FLAG1           ;SET NFLAG TO -1
FLAG0:  SUB     1,1             ;CLEAR AC1 TO ZERO
        STA     1,NFLAG,2       ;STORE IT IN NFLAG
        JMP     CLASS           ;PROCESS CLASS TEST
FLAG1:  SUBZL   1,1
        STA     1,NFLAG,2       ;STORE IT IN NFLAG
CLASS:  LDA     1,OP1DT,2       ;LOAD OPERAND 1 TYPE
        MOVZR#  1,1,SZR         ;CHECK TYPE <=1 (2 OR GREATER)
        JMP     HSKPG           ;2 OR GREATER (NO)
        LDA     1,THREE,2       ;LOAD A THREE
        STA     1,CHKWD,2       ;STORE IT IN CHKWD
```

```
        LDA     1,ONE,2         ;LOAD A ONE
        STA     1,BTCNT,2       ;STORE IT IN BTCNT
        JMP     A2              ;JUMP TO A2
HSKPG:  LDA     1,OP1DL,2       ;LOAD OPERAND 1 LENGTH
        STA     1,BTCNT,2       ;STORE IT IN BTCNT
        LDA     1,OP1DP,2       ;LOAD OPERAND 1 BYTE PTR
        STA     1,OPDP,2        ;STORE IN OPDP
A1:     SUB     1,1             ;CLEAR AC1 TO ZERO
        STA     1,CHKWD,2       ;STORE IT IN CHKWD
        LDA     3,OPDP,2        ;LOAD OPER. 1 BYTE PTR
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET            ;JUMP TO GET A CHARACTER
        JMP     @RTER,2
        LDA     3,MINUS,2       ;TEST FOR MINUS
        SUB#    3,1,SNR
        JMP     AB
        LDA     3,PLUS,2        ;LOAD A PLUS SIGN
        SUB#    3,1,SNR         ;CHECK IF EQUAL
        JMP     AB              ;EQUAL TO PLUS SIGN
        LDA     0,C60,2         ;LOAD AN ASCII ZERO
        LDA     3,C71,2         ;LOAD AN ASCII NINE
        ADCZ#   3,1,SNC         ;SKIPS IF > 9
        ADCZ#   1,0,SZC         ;SKIPS IF > OR = 0
        JMP     .+2             ;NOT EQUAL TO A DIGIT
        JMP     INC3            ;EQUAL TO A DIGIT
        LDA     3,THREE,2       ;LOAD A THREE
        LDA     0,RQCHK,2       ;LOAD REQUEST CHECK CODE
        SUB#    3,0,SNR         ;CHECK IF EQUAL
        JMP     AA              ;EQUAL
A3:     LDA     3,BLK,2
        SUB#    3,1,SNR
        JMP     INC2
        MOVS    1,3             ;SWAP BYTES OF WORD
        MOVL    3,3             ;MOVE BIT 0 INTO CARRY
        MOVL    3,3,SNC         ;MOVE NEXT BIT TO CARRY
        JMP     INC1            ;CARRY EQUAL TO 0
        JMP     INC2            ;CARRY EQUAL TO 1
INC3:   ISZ     CHKWD,2         ;INCREMENT CHKWD (WILL END UP A 3)
INC2:   ISZ     CHKWD,2         ;INCREMENT CHKWD (WILL END UP A 2)
INC1:   ISZ     CHKWD,2         ;INCREMENT CHKWD (WILL END UP A 1)
A2:     LDA     3,RQCHK,2       ;LOAD REQUEST CHECK
        LDA     1,CHKWD,2       ;LOAD CHECK WORD
        SUB     3,1,SNR         ;CHKWD - RQCHK
        JMP     .+7
AA:     LDA     3,NFLAG,2       ;LOAD NFLAG
        MOV     3,3,SZR         ;(RESULT OF CHKWD-RQCHK) - NFLAG
        JSR     @RTN,2
        SUB     1,1             ;CLEAR AC1 TO ZERO
        STA     1,TQC,2         ;STORE IN TEMP. QUINT CTR.
        JMP     @RTN,2
        ISZ     OPDP,2  ;INCREMENT BYTE POINTER
        DSZ     BTCNT,2 ;DECREMENT LENGTH
        JMP     A1      ;NOT EQUAL TO ZERO, GET NEXT CHARACTER
        LDA     1,NFLAG,2
        MOV     1,1,SNR ;TEST TO SEE IF NFLAG IS SET
        JSR     @RTN,2  ;NO
        SUB     1,1     ;YES
        STA     1,TQC,2
        JSR     @RTN,2
AB:     LDA     1,OP1DT,2
        LDA     0,FOUR,2
```

```
        SUB     1,0,SZR ;TEST TO SEE IF A SIGN FIELD
        JMP     AA      ;NO
        JMP     INC3
RELAT:  RLAT1
```

```
;
;       IMMEDIATE/INSERT COMMAND
;
        .TITLE  IMINS
        .NREL
        .TXTM   0
        .ENT    IMINS,VALDG,NMNCH,NMNCH
IMINS:  ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @NMNCP,2        ;READ NUMBER
        JMP     .+3
        ISZ     CERFL,2
        JSR     @RTER,2         ;CARR RET
        STA     1,TEMP,2        ;NUMBER
        LDA     3,CMDCT
IMIN1:  LDA     1,0,3           ;COMMAND CHARACTER
        MOV     1,1,SNR         ;END OF TABLE?
        JSR     @RTER,2         ;YES
        SUB     0,1,SNR         ;THIS CHAR?
        JMP     @DISPL,3            ;NO
        INC     3,3
        JMP     IMIN1       ;EXECUTE
CMDCT:  .+1                     ;COMMAND CHARACTER & DISPATCH TABLE
        .TXT    /I/             ;IO FILE INSERT
        .TXT    /F/             ;FD FILE INSERT
        .TXT    /D/             ;DATA NAME FILE INSERT
        .TXT    /Q/             ;QUINT FILE INSERT
        .TXT    / /             ;IMMEDIATE
        .TXT    /,/             ;IMMEDIATE
        0
        .TXTM   1
        DISPL=7                 ;DISPLACEMENT TO DISPATCH TABLE
        IOIN
        FDIN
        DNIN
        QNIN
        IMMED
        IMMED
IOIN:   JSR     @PRERP,2
FDIN:   JSR     @PRERP,2
QNIN:   JSR     @PRERP,2
;
;       INSERT DATA NAME ENTRY
;
DNIN:   DSZ     TEMP,2
        LDA     0,TEMP,2        ;DATA NAME ENTRY -1
        JSR     @DNSTP,2
        JSR     @RTER,2
         STA     1,TEMP,2        ;STARTING ADDRESS OF ENTRY
        LDA     1,FOUR,2
        STA     1,TEMP2,2
DN1:    JSR     @NMNCP,2        ;READ NUMBER
        JMP     .+2
        JMP     DN2
        ISZ     TEMP,2
        STA     1,@TEMP,2
        DSZ     TEMP2,2
        JMP     DN1
        ISZ     CERFL,2
        JSR     @RTER,2
DN2:    DSZ     TEMP2,2
```

```
        JMP     .+2
        JMP     .+3
        ISZ     CERFL,2
        JSR     @RTER,2
        ISZ     TEMP,2
        STA     1,@TEMP,2
        LDA     3,TEMP,2
        LDA     0,-3,3
        LDA     1,ONEB7,2       ;BRING IN A 400
        MOVZL   1,1
        AND     0,1,SNR
        JMP     @RTN,2
        LDA     0,RSMX,2
        LDA     1,-2,3
        MOVZR#  0,0,SZC
        INC     0,0             ;START 77 ON LEFT BYTE
        STA     0,-4,3          ;RAW STORAGE BYTE PTR
        ADD     1,0
        STA     0,RSMX,2        ;NEW MAX
        JSR     @RTN,2
IMMED:  LDA     1,TEMP,2
        LDA     3,IMONT
        INC     3,3
        STA     1,0,3
        INC     3,3
        STA     3,TEMP,2
        LDA     1,THREE,2
        STA     1,TEMP2,2
IMM1:   JSR     @NMNCP,2
        JMP     .+2
        JMP     IMM2
        STA     1,@TEMP,2
        ISZ     TEMP,2
        DSZ     TEMP2,2
        JMP     IMM1
        JSR     @PRERP,2
IMM2:   DSZ     TEMP2,2
        JSR     @PPERP,2
        STA     1,@TEMP,2
        LDA     3,IMONT
        STA     3,OP,2
        LDA     3,1,3
        LDA     1,BELL,2
        AND     3,1
        STA     1,OPFLG,2
        MOVZR   3,3
        MOVZR   3,3
        MOVZR   3,3
        LDA     1,OPDSP,2
        ADD     1,3
        LDA     1,SP,2
        STA     1,OP1DE,2       ;STORE OLD SP
        LDA     3,0,3
        LDA     1,0,3
        STA     1,SP,2
        JSR     @EXONP,2
        JSR     @RTER,2
        LDA     1,OP1DE,2
        STA     1,SP,2          ;RESTORE OLD SP
        DSZ     QC,2
        JMP     .+1
```

```
        JSR     @RTN,2
IMONT:  .+1
        .BLK    5
;
;       CHECKS FOR VALID DIGIT
;
VALDG:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     3,C71,2             ;ASCII 9
        ADCZ#   3,0,SZC
        JSR     @RTER,2             ;AC0 > 9
        LDA     3,C60,2
        ADCZ#   0,3,SZC
        JSR     @RTER,2             ;AC0 < 0
        JSR     @RTN,2              ;AC0 DIGIT
;
;       READS 5 DIGITS, ACCUMULATES IN AC1
;
NMNCH:  ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GCHAP,2
        JSR     @RTER,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        JMP     NM1
;
;       READS 4 DIGITS, ACCUMULATES IN AC1, EXPECTS RT ADJ ASCII CHAR IN AC0
;
NMNCH:  ISZ     RTST,2
        STA     3,@RTST,2
NM1:    SUB     1,1
        JSR     VALDG               ;DIGIT?
        JSR     NM3                 ;NO
        MOV     0,1
        LDA     0,C60,2
        SUB     0,1
        LDA     3,FOUR,2
        STA     3,X17T,2
NM2:    JSR     @GCHAP,2            ;NEXT DIGIT
        JSR     NM3
        .SYSTM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2
        JSR     VALDG               ;DIGIT?
        JSR     NM3                 ;NO
        MOVZL   1,3
        MOVZL   3,3
        ADDZL   3,1                 ; *10
        LDA     3,C60,2
        SUB     3,0                 ;BINARY
        ADD     0,1                 ;ACCUMULATE
        DSZ     X17T,2
        JMP     NM2
        JSR     @GCHAP,2
        JSR     NM3
        .SYSTEM
        .PCHAR
        JSR     .+1
NM3:    LDA     3,C1,2
```

```
SUB#    0,3,SZR ;CARRIAGE RETURN?
JSR     @RTER,2 ;NO
LDA     0,LF,2
.SYSTEM
.PCHAR
JSR     @RTER,3
MOV     3,2
JSR     @RTN,2
```

```
;
;       INITIALIZE SAVE FILES
;
        .TITLE  INSVF
        .NREL
        .TXTM   1
        .ENT    INSVF
        .EXTN   DSKFL
INSVF:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     3,IBUF,2
        LDA     0,SVFCN,2
        STA     0,IO,3
        SUBZL     0,0
        STA     0,IFCN,3
        LDA     0,IOSTW,2
        STA     0,STWD,3
        LDA     0,IOCNT,2
        STA     0,ICT,3
        LDA     0,IOST,2
        STA     0,IVEC,3
        JSR     @DSKFP
        JSR     @RTEP,2
        LDA     3,IBUF,2
        LDA     0,FDSTW,2
        STA     0,STWD,3
        LDA     0,FDCNT,2
        STA     0,ICT,3
        LDA     0,FDST,2
        STA     0,IVEC,3
        JSR     @DSKFP
        JSR     @RTER,2
        LDA     3,IBUF,2
        LDA     0,EDSTW,2
        STA     0,STWD,3
        LDA     0,EDCNT,2
        STA     0,ICT,3
        LDA     0,EDST,2
        STA     0,IVEC,3
        JSR     @DSKFP
        JSR     @RTER,2
        LDA     3,IBUF,2
        LDA     0,DNSTW,2
        STA     0,STWD,3
        LDA     0,DNCNT,2
        STA     0,ICT,3
        LDA     0,DNST,2
        STA     0,IVEC,3
        JSR     @DSKFP
        JSR     @RTEP,2
        LDA     3,IBUF,2
        LDA     0,RSSTW,2
        STA     0,STWD,3
        LDA     0,RSCNT,2
        STA     0,ICT,3
        LDA     0,RSST,2
        STA     0,IVEC,3
        JSR     @DSKFP
        JSR     @RTER,2
        LDA     3,IBUF,2
        LDA     0,ONSTW,2
```

```
        STA     0,STHD,3
        LDA     0,QNCNT,2
        STA     0,ICT,3
        LDA     0,QNST,2
        STA     0,IVEC,3
        JSR     @DSKFP
        JSR     @RTER,2
        JSR     @RTN,2
DSKFP:  DSKFL
```

```
        .TITL   LEVEL
        .ENT    AG1,AG2
        .NREL
AG1:    ISZ     RTST,2
        STA     3,@RTST,2
        LDA     1,OP1DT,2       ;GET DATA TYPE
        MOV     1,3
        LDA     0,RBMSK,2       ;GET MASK OF 3MM
        AND     0,1             ;SAVE RIGHT BYTE
        STA     1,OP1DT,2
        MOVS    3,3             ;SWAP BYTES
        AND     0,3             ;SAVE RIGHT BYTE
        STA     3,OP1LV,2       ;STORE
        JSR     @RTN,2
AG2:    ISZ     RTST,2
        STA     3,@RTST,2
        LDA     1,OP2DT,2
        MOV     1,3
        LDA     0,RBMSK,2
        AND     0,1
        STA     1,OP2DT,2
        MOVS    3,3
        AND     0,3
        STA     3,OP2LV,2
        JSR     @RTN,2
```

```
        .TITL   LOADPACK
        .ENT    LDPK1,LDPK2,LDPK3,SFLDP
        .EXTD   GET,PUT
        .NREL
SFLDP:  SUBZL   1,1
        STA     1,SAV,2
LDPK1:  LDA     1,PACK1,2       ;INITILIZE LOADPACK ROUTINE
        STA     1,PACTL,2       ;FOR PACK1
        LDA     1,OP1DT,2
        STA     1,OPDT,2
        LDA     1,OP1DL,2
        STA     1,OPDL,2
        LDA     1,OP1DP,2
        STA     1,OPDP,2
        LDA     1,DEC1,2
        STA     1,OPDE,2
        JMP     PAKIT
LDPK2:  LDA     1,PACK2,2       ;INITILIZE LOADPACK ROUTINE
        STA     1,PACTL,2       ;FOR PACK2
        LDA     1,OP2DT,2
        STA     1,OPDT,2
        LDA     1,OP2DL,2
        STA     1,OPDL,2
        LDA     1,OP2DP,2
        STA     1,OPDP,2
        LDA     1,DEC2,2
        STA     1,OPDE,2
        JMP     PAKIT
LDPK3:  LDA     1,PACK2,2       ;INITILIZE LOADPACK ROUTINE
        STA     1,PACTL,2       ;FOR OPERAND THREE
        LDA     1,OP3DT,2
        STA     1,OPDT,2
        LDA     1,OP3DL,2
        STA     1,OPDL,2
        LDA     1,OP3DP,2
        STA     1,OPDP,2
        LDA     1,DEC3,2
        STA     1,OPDE,2
PAKIT:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,OPDT,2    ;GET ARGUMENT 1 TYPE
        SUB     0,0         ;GENERATE A ZERO
        SUB#    3,0,SNR     ;CHECK TO SEE IF EQUAL TO ZERO
        JMP     COMP        ;YES A COMP TO COMP
        LDA     0,TEN,2     ;GET A TEN
        STA     0,N,2       ;STORE IN N
        LDA     1,PACTL,2   ;ZERO-OUT PACK1
        STA     1,PACZ,2
        JSR     @ZERA,2       ;NO, ZERO OUT PAC1
        JSR     @RTER,2
        LDA     0,OPDL,2    ;GET LENGTH OF FIRST ARGUMENT
        STA     0,MM,2        ;STORE IN M
        LDA     1,FOUR,2
        STA     1,N,2
        LDA     1,PACT,2
        STA     1,PACZ,2
        JSR     @ZERA,2
        JSR     @RTER,2
AA:     LDA     3,OPDP,2    ;GET BYTE POINTER
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET        ;GET 2 BYTE
```

```
        JSR     @RTER,2
        LDA     0,MSK2,2        ;GET MASK 17
        AND     0,1             ;SAVE CHAR DESTROY ACII
        ISZ     PACT,2          ;TO GET PROPER POSITION
        ISZ     PACT,2          ;FOR STORING OF CHARACTER
        ISZ     PACT,2
        STA     1,@PACT,2       ;STORE THE CHARACTER
        DSZ     PACT,2          ;DECREMENT RECEIVING
        DSZ     PACT,2          ;FIELD TO GET THE
        DSZ     PACT,2          ;BEGINNING OF IT
        LDA     0,PACT,2        ;GET BEGINNING OF TEMP
        STA     0,PAC1,2        ;INITILIZE
        LDA     0,PACTL,2       ;THESE FOR THE ADD ROUTINE
        STA     0,PAC2,2
        LDA     0,PACK3,2
        STA     0,PAC3,2
        JSR     @ADDT,2
        JSR     @RTER,2
        LDA     1,PACK3,2
        STA     1,PAC1,2
        LDA     1,PACTL,2
        STA     1,PAC2,2
        JSR     @MNOV,2
        JSR     @RTER,2
        DSZ     MM,2
        JMP     .+2             ;NOT EQUAL TO ZERO
        JMP     EXIT1           ;EQUAL TO ZERO
        JSR     @MTEN,2
        JSR     @RTER,2
        ISZ     OPDP,2
        JMP     AA
EXIT1:  LDA     1,SAV,2
        MOVR    1,1,SZC ;TEST TO SEE IF SET
        JMP     MMAV            ;SET
        LDA     0,OPDE,2        ;GET DECIMAL PTR
        SUBZ    1,1             ;CLEAR AC1
        SUB#    1,0,SNR         ;TEST TO SEE IF ZERO
        JMP     @RTN,2          ;YES RETURN
        LDA     1,TEN,2         ;NO SET N TO EIGHT
        STA     1,N,2
        STA     0,MM,2          ;SET M TO NUMBER OF DEC POS.
        JSR     @DV10,2
        JSR     @RTER,2
        DSZ     MM,2            ;DECREMENT DECIMAL POINT
        JMP     EXIT1+7         ;DO AGAIN TILL ZERO
        JMP     @RTN,2          ;RETURN
COMP:   LDA     1,FIVE,2        ;GET A FIVE
        STA     1,N,2           ;STORE IN N
        STA     1,MM,2  ;INITILIZE A COUNTER
        LDA     1,PACTL,2       ;ZERO OUT RECEIVING PACK AREA
        STA     1,PACZ,2
        JSR     @ZERA,2
        JSR     @RTER,2
        LDA     1,PACTL,2       ;GET RECEIVING PACK FIELD
        LDA     0,OPDE,2        ;GET DECIMAL POSITION
        MOV#    0,0,SNR ;CHECK TO SEE IT EQUAL TO ZERO
        JMP     .+5     ;YES
        ADD     0,1
        STA     1,PAC1,2        ;STORE THE ADDRESS IN PAC1
        DSZ     PAC1,2
        JMP     .+2
```

```
        STA     1,PAC1,2
        LDA     0,OPDP,2        ;GET BYTE POINTER
        MOVZR   0,0     ;CHANGE TO WORD POINTER
        STA     0,OPDP,2        ;STORE BACK OUT
        LDA     0,@OPDP,2       ;GET FIRST BINARY WORD
        STA     0,@PAC1,2       ;STORE IT
        DSZ     OPDL,2  ;DECREMENT LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @RTN,2  ;ZERO
        ISZ     OPDP,2  ;GET NEXT WORD
        ISZ     PAC1,2  ;GET NEXT STORE POSITION
        DSZ     MM,2    ;DECREMENT COUNTER
        JMP     .-10    ;DO TILL ZERO
        JMP     @RTN,2      ;RETURN TO INTERPUTER
MMAV:   SUB     1,1
        STA     1,SAV,2
        LDA     1,PACTL,2
        STA     1,PAC1,2
        LDA     0,FOUR,2
        STA     0,N,2
        ADD     1,0
        STA     0,PAC2,2
        JSR     @MM0V,2
        JSR     @RTER,2
        LDA     1,PACTL,2
        STA     1,PAC2,2
        JSR     @ZERA,2
        JSR     @RTER,2
        JSR     @RTN,2
```

```
        .TITL   MATHMOVE
        .ENT    MOVA,HERE,M3TO1,M2TO3
        .NREL
;CALL TO THIS ROUTINE IS (N,PAC1,PAC2)
M3TO1:  LDA     1,TEN,2 ;GET A TEN FOR THE NUMBER OF WORD TO BE MOVED
        STA     1,NT,2  ;INITILIZE A COUNTER
        LDA     1,PACK3,2       ;GET RECEIVING PACK AREA
        STA     1,PAC1,2        ;STORE IN PAC1,2
        LDA     1,PACK1,2
        STA     1,PAC2,2
        JMP     HERE
M2TO3:  LDA     1,TEN,2 ;TO INITILIZE A COUNTER WITH A TEN
        STA     1,NT,2
        LDA     1,PACK2,2       ;MOVE PACK2 TO PACK3
        STA     1,PAC1,2
        LDA     1,PACK3,2
        STA     1,PAC2,2
        JMP     HERE
MOVA:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,N,2           ;GET NUMBER OF WORDS LONG
        STA     3,NT,2          ;STORE IN A COUNTER
        JMP     .+3
HERE:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,@PAC1,2       ;GET  IRST ARGUMENT TO BE MOVED
        STA     1,@PAC2,2       ;AND STORE IT IN RECEIVING FIELD
        ISZ     PAC1,2          ;GET NEXT WORD
        ISZ     PAC2,2          ;GET NEXT STORE POSITION
        DSZ     NT,2
        JMP     HERE+2          ;NOT ZERO GO TO HERE
        JMP     @RTN,2          ;RETURN
```

```
          .TITL   MATHPACKIF
          .ENT    MATHP
          .EXTN   LDPK1,LDPK2,SUBPK
          .NREL
MATHP:    SUB     1,1             ;CLEAR AC1 TO ZERO
          STA     1,CHKWD,2       ;STORE IN CHECK WORD
          JSR     @ARG1
          JSR     @RTER,2
          JSR     @ARG2
          JSR     @RTER,2
          JSR     @CPARE
          JSR     @RTER,2
          LDA     1,TEN,2         ;LOAD AN EIGHT
          STA     1,X4T,2         ;STORE IN TEMP. CNTR.
          LDA     1,@PAC3,2       ;LOAD WORD FROM PAC3
          MOV     1,1,SZR         ;CHECK IF ZERO
                  JMP     .+6             ;NOT ZERO
          DSZ     X4T,2           ;ZERO, DECREMENT COUNTER
          JMP     .+2             ;NOT ZERO
          JMP     INC3            ;ZERO (ARG1 = ARG2)
          ISZ     PAC3,2          ;GET NEXT WORD
          JMP     .-7             ;DO NEXT WORD
          LDA     1,@PAC3,2       ;LOAD FIRST WORD OF ANSWER
          MOVL    1,1,SZC         ;SHIFT BIT0 INTO CARRY
          JMP     INC2            ;YES (ARG1 > ARG2)
          JMP     INC1            ;NO (ARG1 < ARG2)
INC3:     ISZ     CHKWD,2         ;INCREMENT CHKWD (WILL END UP 3)
INC2:     ISZ     CHKWD,2         ;INCREENT CHKWD (WILL END UP 2)
INC1:     ISZ     CHKWD,2         ;INCREMANT CHKWD (WILL END UP 1)
          LDA     3,RQCHK,2       ;LOAD REQUEST CHECK
          LDA     1,CHKWD,2       ;LOAD CHECK WORD
          SUB     3,1,SNR         ;CHKWD - RQCHK
          JMP     C11+3           ;RESULT = ZERO
          LDA     1,NFLAG,2       ;LOAD NOT FLAG
          MOV     1,1,SZR         ;SEE IF FLAG IS SET
          JMP     B1              ;FLAG IS SET
C11:      SUB     1,1             ;CLEAR AC1
          STA     1,TQC,2         ;CLEAR TQC
          JMP     @RTN,2          ;RETURN, BAD COMPARE
          LDA     1,NFLAG,2       ;LOAD NOT FLAG
          MOV     1,1,SZR         ;SEE IF FLAG IS SET
          JMP     C11             ;FLAG IS SET
B1:       JMP     @RTN,2          ;RETURN, GOOD COMPARE
ARG1:     LDPK1
ARG2:     LDPK2
CPARE:    SUBPK
```

```
        .TITL   MOVE
        .EXTN   VALID,TYPE1,SETED,EDITT
        .EXTD   GET,PUT
        .ENT    MOVE,B1,AAB
        .NREL
MOVE:   ISZ     RTST,2  ;GET NEXT STORE POSITION
         STA     3,@RTST,2      ;SAVE RETURN ADDRESS
        LDA     3,OP1DP,2       ;GET OP1 BYTE PTR
        STA     3,PNT1,2        ;STORE IN PNT1
        LDA     3,OP1DL,2       ;GET LENGTH FOR OP1
        STA     3,CNT1,2        ;STORE CNT1
        LDA     3,OP2DP,2       ;GET RAW DATA BYTE PTR FOR OP2
        STA     3,PNT2,2        ;STORE IN PNT2
        LDA     3,OP2DL,2          ;GET LENGTH OF RECEIVING FIELD
        STA     3,CNT2,2           ;STORE IN CNT2
        JSR     @.VALD  ;GO TO VALIDITY ROUTINE
        JSR     @RTER,2
        SUB     0,0     ;CLEAR AC0
        STA     0,EFLAG,2
        JSR     @TPYE   ;GO TO VALID TYPE ROUTINE
        JSR     @RTER,2
        LDA     1,EFLAG,2          ;GET EDIT FLAG
        MOVR    1,1,SZC ;CHECK TO SEE IF SET
B1:     JMP     @.SETD  ;JMP T SET EDIT ROUTINE
PT:     LDA     3,PNT0,2           ;GET BYTE POINTER
        MOV#    3,3,SNR ;TEST TO SEE IF EQUAL TO ZERO
        JMP     PT1             ;NOJMP TO PT1
        LDA     1,C60,2         ;YES LOAD IN ASC11 ZERO
        LDA     3,PNT2,2        ;LOAD IN BYTE PTR
        STA     3,BYPTR,2          ;STORE IN BYTE POINTER
        JSR     @PUT            ;GO TO STORE ROUTINE
        JMP     @RTER,2
        ISZ     PNT2,2          ;ISZ STORE BYTE PTR
        DSZ     CNT0,2          ;DECREMENT CNT1
        JMP     .+7             ;NOT ZERO
        DSZ     CNT2,2
        JMP     .+2
        JSR     @RTN,2
        SUB     1,1     ;CLEAR AC1
        STA     1,PNT0,2           ;CLEAR ZERO FLAG BYTE POINTER
        JMP     PT1             ;CNT1 IS ZERO
        DSZ     CNT2,2          ;DECREMENT CNT2
        JMP     PT              ;NOT ZERO
        JMP     @RTN,2         ;ZERO RUN TIME ERROR
PT1:    LDA     3,PNT1,2        ;GET BYTE PTR SENDING FIELD
        STA     3,BYPTR,2          ;STORE IN BYTE POINTER
        JSR     @GET            ;GO TO GET A BYTE ROUTINE
        JMP     @RTER,2
        LDA     3,PNT2,2        ;LOAD IN RECEIVING BYTE PTR
        STA     3,BYPTR,2          ;STORE IN BYTE POINTER
        JSR     @PUT
        JMP     @RTER,2
        ISZ     PNT1,2
        ISZ     PNT2,2
        DSZ     CNT1,2          ;DSZ LENGTH OP1
        JMP     .+2
        JMP     PT2             ;CHECK FOR PNT1  # 0
        DSZ     CNT2,2          ;DSZ LENGTH OP2
        JMP     PT1
        JMP     @RTN,2         ;JMP RUN TIME ERROR
PT2:    DSZ     CNT2,2
```

```
          JMP     .+2
          JMP     @RTN,2  ;RETURN TO INTERPRETER
          LDA     1,OP1DT,2       ;GET TYPE
          LDA     0,BELL,2        ;GET A SEVEN
          SUB#    1,0,SNR
          JMP     A0
          LDA     0,SIX,2
          SUB#    1,0,SNR
          JMP     AA      ;ZERO FILL
          LDA     1,INDIC,2       ;GET INDICATOR
          LDA     0,FOUR,2        ;GET A FOUR FOR A TEST
          SUBZ    1,0,SNC         ;TEST TO SEE IF INDICATOR IS  # = 4
          JMP     AA              ;NUMERIC FILL WITH ZEROS
A0:       LDA     1,BLK,2         ;BLANK FILL
          LDA     3,PNT2,2        ;GET BYTE PTR FOR SECOND OPERAND
          STA     3,BYPTR,2          ;STORE IN BYTE POINTER
          JSR     @PUT            ;STORE BLANK
          JMP     @RTER,2
          ISZ     PNT2,2
          DSZ     CNT2,2          ;DSZ COUNT TWO
          JMP     .-7
          JMP     @RTN,2          ;EXIT ROUTINE
AAB:      DSZ     CNT2,2
          JMP     .+2
          JMP     B1      ;TOTAL LENGTH WENT TO ZERO GET END OF EDIT CODE
AA:       LDA     1,C60,2
          LDA     3,PNT2,2        ;ZERO FILL BYTE PTR FOR 2ND OP.
          STA     3,BYPTR,2          ;STORE BYTE POINTER
          JSR     @PUT            ;STORE ZERO FILL
          JMP     @RTER,2
          ISZ     PNT2,2
          DSZ     CNT2,2          ;DSZ COUNT TWO
          JMP     AA
          JMP     @RTN,2
.VALD:    VALID
TPYE:     TYPE1
.SETD:    SETED
.EDIT:    EDITT
```

```
000000 IDENTIFICATION DIVISION                                          .00000100
000010 PROGRAM-ID.      ABCTP                                           .00000200
000860 ENVIRONMENT DIVISION                                             .00000300
000870 CONFIGURATION SECTION                                            .00000400
000880 SOURCE-COMPUTER                                                  .00000500
000890     UNIVAC-1108                                                  .00000600
000900 OBJECT-COMPUTER                                                  .00000700
000910     UNIVAC-1108                                                  .00000800
000920 INPUT-OUTPUT SECTION                                             .00000900
000930 FILE-CONTROL.                                                    .00001000
000940     SELECT      COBIN ASSIGN      $CDR                          .
           SELECT      PDIV  ASSIGN      NKF                           .
           SELECT      MINIZ ASSIGN      COBIN                         .
           SELECT      LITF  ASSIGN      X2                            .
           SELECT      PRNF  ASSIGN      PRINTER                       .00001500
       DATA DIVISION.                                                   .00001600
       FILE SECTION.                                                    .00001700
000980 FD  COBIN                                                        .00001800
000990     LABEL RECORDS OMITTED                                        .00001900
001000*SIZE IS          80                                             .00002000
001010 01  SOR                                                         .00002100
001020     02  FILLER  PIC   9X6<                                      .00002200
001030     02  CONT    PIC   X                                         .00002300
001040     02  PARAG                                                   .00002400
001050     03  PARA    PIC   X%4<                                      .00002500
001060     03  INSTR                                                   .00002600
001070     04  INSA    PIC   X%2<                                      .00002700
001080     04  FILLER  PIC   X%2<                                      .00002800
001090     04  INSB    PIC   X%8<                                      .00002900
001100     02  DATA1   PIC   X%6<                                      .00003000
001110     02  IGN-1   PIC   X%6<                                      .00003100
001120     02  CODF                                                    .00003200
001130     03  DATA2   PIC   X%6<                                      .00003300
001140     03  DATA3   PIC   X%6<                                      .00003400
001150     03  DATA4   PIC   X%6<                                      .00003500
001160     03  DATA5   PIC   X%6<                                      .00003600
001170     03  DATA6   PIC   X%6<                                      .00003700
001180     03  DATA7   PIC   X%6<                                      .00003800
001190     02  STEND   PIC   X                                         .00003900
           02  PFILL   PIC   X%8<                                      .00004000
001210 FD  MINIZ                                                       .00004100
           RECORD      30                                            .
           LABEL       RECORD       OMITTED                          .
001230*SIZE IS          80                                             .00004400
001240 01  MINI                                                        .00004500
001250     02  MIASM   PIC   X%30<                                     .00004600
       FD  PDIV                                                        .00004700
           RECORD      23                                            .00004800
           LABEL       RECORD       OMITTED                          .00004900
       01  POUTS       PIC   X%23<                                     .00005000
       FD  LITF                                                        .00005100
           RECORD      60                                            .00005200
           LABEL       RECORD       OMITTED                          .00005300
       01  LITR        PIC   X%60<                                     .00005400
       FD  PRNF                                                        .00005500
           RECORD      86                                            .00005600
           LABEL       RECORD       OMITTED                          .00005700
       01  PRNR                                                       .00005800
           02  PRN     PIC   9%4<                                      .00005900
           02  FILLER  PIC   X%2<                                      .00006000
           02  PRR     PIC   X%80<                                     .00006100
```

```
001270 WORKING-STORAGE SECTION                                          .00006200
       77  DEFG          PIC   9      VALUE ZERO                       .00006300
001280 77  FLONG         PIC   X                                       .00006400
       77  BKLOC         PIC   9%5<   VALUE ZERO                       .00006500
       77  CORLC         PIC   9%5<   VALUE 0                          .00006600
001300*SIZE IS            26                                           00006700
001310 01  DWORD                                                       .00006800
001320     02  IDLBL      PIC   X%5<                                   .00006900
001330     02  ADR        PIC   9%5<                                   .00007000
001340     02  IDPIC      PIC   X                                      .00007100
           02  IDLNG      PIC   9%4<                                   .00007200
001360     02  IDDEC      PIC   9%2<                                   .00007300
001370     02  BAK        PIC   9                                      .00007400
001380     02  SVGN       PIC   9                                      .00007500
001390     02  USGE       PIC   9                                      .00007600
001400     02  SINC       PIC   9                                      .00007700
001410     02  OCP        PIC   9                                      .00007800
001420     02  EDR        PIC   9%2<                                   .00007900
001430     02  VLR        PIC   9%3<                                   .00008000
001440*SIZE IS            23                                           00008100
001450 01  PWORD                                                       .00008200
           02  CSEQ       PIC   9%4<                                   .00008300
           02  FLOP       PIC   9                                      .00008400
           02  RTN01      PIC   9%3<                                   .00008500
           02  PBDY                                                    .00008600
           03  FRST1      PIC   X%5<                                   .00008700
           03  SECD1      PIC   X%5<                                   .00008800
           03  LAST1      PIC   X%5<                                   .00008900
001560*SIZE IS            3000                                         00009000
001570 01  EDITT                                                       .00009100
           02  EDT        OCCURS      20    TIMES                      00009200
           03  FILLER     PIC   X%18<                                  .00009300
           03  BB         PIC   X%6<                                   .00009400
           03  AA         PIC   X%6<                                   .00009500
001620*SIZE IS            2000                                         00009600
001630 01  JMPTA                                                       .00009700
           02  JMPTB      OCCURS      250                            .00009800
           03  JMPS0      PIC   9%5<                                   .00009900
           03  JMPNA      PIC   X%5<                                   .00010000
001660*SIZE IS            60                                           00010100
001670 01  TAB1                                                        .00010200
           02  T1         PIC   X     OCCURS      62    TIMES          .00010300
     *SIZE IS              119                                        00010400
001710 01  TAB2                                                        .00010500
           02  FILLER     PIC   X%11<                                  .00010600
001730     02  TA                                                      .00010700
           03  T2         PIC   X%6<  OCCURS      18                 .00010800
001750*SIZE IS            6                                            00010900
001760 01  TAB3                                                        .00011000
001770     02  T3         PIC   X     OCCURS      6     TIMES          .00011100
001780*SIZE IS            676                                          00011200
001790 01  ONET                                                        .00011300
           02  ONEDE      OCCURS      50    TIMES                      .00011400
001810     03  NA         PIC   X%5<                                   .00011500
           03  FILLER     PIC   X%5<                                   .00011600
           03  IPICT      PIC   X                                      .00011700
           03  IDLN       PIC   9%4<                                   .00011800
001840     03  FILLER     PIC   X%12<                                  .00011900
001850*SIZE IS            6                                            00012000
001860 01  IGN-2                                                       .00012100
001870     02  COL-1      PIC   X                                      .00012200
```

```
001880      02  COL-2    PIC   X                                          .00012300
001890      02  COL-3    PIC   X                                          .00012400
001900      02  FILLER   PIC   X(3)                                       .00012500
001910*SIZE IS           4                                                00012600
001920 01   PPNAM                                                        .00012700
001930      02  PPCOD    PIC   X                                          .00012800
001940      02  PPNO     PIC   9(3)                                       .00012900
001950*SIZE IS           4                                                00013000
001960 01   VNAM                                                         .00013100
001970      02  TST      PIC   X                                          .00013200
            02  LITCN    PIC   9(4)  VALUE ZERO                           .00013300
001990*SIZE IS           34                                               00013400
002000 01   SUBS1                                                        .00013500
002010      02  SUC                                                      .00013600
002020      03  S1       PIC   9                                          .00013700
002030      03  S2       PIC   9                                          .00013800
002040      03  S3       PIC   9                                          .00013900
002050      02  RDFLG    PIC   9                                          .00014000
            02  SUBR     PIC   9(4)                                       .00014100
            02  SBS      PIC   9(4)                                       .00014200
            02  SUBZ     PIC   9(4)                                       .00014300
            02  OCLN     PIC   9(4)                                       .00014400
            02  SUB      PIC   9(4)                                       .00014500
            02  SUA      PIC   9(4)                                       .00014600
002110      02  FLGNX    PIC   9                                          .00014700
002120      02  LEVL     PIC   9(2)                                       .00014800
002130      02  FOCC     PIC   9(2)                                       .00014900
002140      02  GF       PIC   9(2)                                       .00015000
002150      02  UF       PIC   9                                          .00015100
002160      02  SF       PIC   9                                          .00015200
002170      02  G2       PIC   9(2)                                       .00015300
002180      02  G3       PIC   9(2)                                       .00015400
002190      02  G4       PIC   9(2)                                       .00015500
001510      02  SEQ      PIC   9(5)                                       .00015600
            02  TNU      PIC   9(4)                                       .00015700
            02  G5       PIC   9(2)                                       .00015800
            02  G6       PIC   9(2)                                       .00015900
            02  G7       PIC   9(2)                                       .00016000
            02  G8       PIC   9(2)                                       .00016100
            02  G9       PIC   9(2)                                       .00016200
            02  CG5      PIC   9(4)                                       .00016300
            02  CG6      PIC   9(4)                                       .00016400
            02  CG7      PIC   9(4)                                       .00016500
            02  CG8      PIC   9(4)                                       .00016600
            02  CG9      PIC   9(4)                                       .00016700
            02  QDFLG    PIC   9                                          .00016800
            02  SUFLG    PIC   9                                          .00016900
        01  IOTB1                                                        .00017000
            02  IOTB2    OCCURS      15    TIMES                         .00017100
            03  RELF     PIC   9(3)                                       .00017200
            03  KEYN     PIC   X(5)                                       .00017300
            03  FNAM     PIC   X(5)                                       .00017400
            03  DTYPE    PIC   X(5)                                       .00017500
            03  DFNAM    PIC   X(5)                                       .00017600
        01  JMP2T                                                        .00017700
            02  JS02     PIC   9(5)  OCCURS      400                     .00017800
        01  QSUB                                                         .00017900
            02  XYZS     PIC   X(50) OCCURS      3                       .00018000
002200 PROCEDURE DIVISION                                                .00018100
002210*************************************************************************00018200
002220*        OPENS THE INPUT AND OUTPUT FILES                           .00018300
```

```
002240*****************************************************************************00018400
002250 A                                                                          .00018500
002260       OPEN          INPUT COBIN                                            .00018600
002270       OPEN          OUTPUT       MINIZ                                     .00018700
             OPEN          OUTPUT       PRNF                                      .00018800
             OPEN          OUTPUT       LITF                                      .00018900
002300       MOVE          SPACE TO     MINI                                      .00019000
             MOVE          ZERO  TO     CSEQ                                      .00019100
002310*****************************************************************************00019200
002320*         MAIN CONTROL FOR WHOLE PROGRAM                                    00019300
002330*****************************************************************************00019400
002340 CONTRL                                                                     .00019500
002350       PERFORM       RDCOB THRU   CORDX                                     .00019600
             IF            PARAG EQUAL @IDENTIFICATION D@                         00019700
002370       PERFORM       IDDIV THRU   IDXIT                                     .00019800
             IF            PARAG EQUAL @ENVIRONMENT DIVI@                         00019900
002390       PERFORM       ASSGN THRU   ASXIT                                     .00020000
002400       IF            PARAG EQUAL @FILE SECTION.    @                        00020100
002410       PERFORM       DDNON THRU   DDXIT                                     .00020200
             IF            PARAG EQUAL @PROCEDURE DIVISI@                         00020300
002430       PERFORM       PROC  THRU   PRXIT                                     .00020400
002440       GO TO         CONTRL                                                 .00020500
      *****************************************************************************00020600
002460*         IDENTIFICATION DIVISION CONTROL                                   00020700
002470*****************************************************************************00020800
002480 IDDIV                                                                      .00020900
             MOVE          @ 1@  TO     MIASM                                     .00021000
002500       PERFORM       WRIT  THRU   WRXIT                                     .00021100
       IDLOP                                                                      .00021200
             PERFORM       RDCOB THRU   CORDX                                     .00021300
             IF            PARAG EQUAL @ENVIRONMENT DIVI@                         00021400
             GO TO         IDXIT                                                  .00021500
             IF            PARAG NOT    EQUAL @PROGRAM-ID.       @                00021600
             GO TO         IDLOP                                                  .00021700
002510       IF            DATA1 EQUAL SPACE                                      00021800
002520       PERFORM       RDCOB THRU   CORDX                                     00021900
002530       MOVE          INSTR TO     MIASM                                     00022000
002540       PERFORM       WRIT  THRU   WRXIT                                     00022100
002550       GO TO         IDXIT                                                  .00022200
002560       MOVE          DATA1 TO     MIASM                                     .00022300
002570       PERFORM       WRIT  THRU   WRXIT                                     .00022400
002580 IDXIT                                                                      .00022500
002590       EXIT                                                                .00022600
002600*****************************************************************************00022700
002610*         ENVIRONMENT DIVISION CONTROL                                      00022800
002620*****************************************************************************00022900
002630 ASSGN                                                                      .00023000
             MOVE          @ 2@  TO     MIASM                                     .00023100
002650       PERFORM       WRIT  THRU   WRXIT                                     .00023200
             MOVE          SPACE TO     IOTB1                                     .00023300
             MOVE          ZERO  TO     SUB                                       .00023400
       ASSG0                                                                      .00023500
             PERFORM       RDCOB THRU   CORDX                                     .00023600
             IF            PARAG EQUAL @FILE-CONTROL.    @                        00023700
             GO TO         ASSG1                                                  .00023800
             IF            INSTR EQUAL @DEBUGGING    @                          00023900
             MOVE          1     TO     DEFG                                      .00024000
             GO TO         ASSG0                                                  .00024100
002660 ASSG1                                                                      .00024200
002670       PERFORM       RDCOB THRU   CORDX                                     .00024300
             IF            INSTR EQUAL @SELECT       @                          00024400
```

```
           ADD          1     TO    SUB                                    00024500
           GO TO        SELG                                               00024600
           IF           INSTR EQUAL @ORGANIZATION@                       00024700
           GO TO        ORASG                                              00024800
           IF           INSTR EQUAL @ACCESS       @                      00024900
           GO TO        ACASG                                              00025000
           IF           PARA  EQUAL @DATA@                               00025100
           MOVE         ZERO  TO    SUB                                  00025200
           GO TO        ASPRT                                              00025300
       SELG                                                               00025400
           MOVE         ZERO  TO    RELF  %SUB<                          00025500
           MOVE         DATA1 TO    FNAM  %SUB<                          00025600
           MOVE         DATA3 TO    DTYPE %SUB<                          00025700
           MOVE         SPACE TO    DFNAM %SUB<
           IF           DATA4 NOT   EQUAL SPACE                          00025900
           MOVE         DATA4 TO    DFNAM %SUB<                          00026000
           GO TO        ASSG1                                              00026100
       ORASG                                                              00026200
           IF           DATA1 EQUAL @RELATI@                             00026300
           MOVE         100   TO    RELF  %SUB<                          00026400
           GO TO        ASSG1                                              00026500
       ACASG                                                              00026600
           IF           DATA2 EQUAL @RANDOM@                             00026700
           ADD          10    TO    RELF  %SUB<                          00026800
           IF           DATA2 EQUAL SPACE                                00026900
           GO TO        ASSG1                                              00027000
           IF           DATA4 EQUAL SPACE                                00027100
           GO TO        ASSG1                                              00027200
           MOVE         DATA4 TO    KEYN  %SUB<                          00027300
           ADD          1     TO    RELF  %SUB<                          00027400
           GO TO        ASSG1                                              00027500
       ASPRT                                                              00027600
           ADD          1     TO    SUB                                  00027700
           IF           IOTB2 %SUB< EQUAL SPACE                          00027800
           GO TO        ASXIT                                              00027900
           MOVE         IOTB2 %SUB< TO    MIASM                          00028000
           PERFORM      WRIT  THRU  WRXIT                                00028100
           GO TO        ASPRT                                              00028200
002800 ASXIT                                                              00028300
002810     EXIT                                                           00028400
002820*****************************************************************00028500
002830*         DATA DIVISION CONTROL                                     00028600
002840*****************************************************************00028700
002850 DDNOW                                                              00028800
           MOVE         @ 3@  TO    MIASM                                00028900
002870     PERFORM      WRIT  THRU  WRXIT                                00029000
002880     MOVE         SPACE TO    EDITT                                00029100
002888 DDRD                                                               00029200
002890     PERFORM      RDCOB THRU  CORDX                                00029300
002900 D77                                                                00029400
002910     IF           PARA  EQUAL @77   @                              00029500
002920     MOVE         @01   @     TO    PARA                           00029600
002930     IF           PARA  EQUAL @FD   @                              00029700
002940     PERFORM      FDD   THRU  FDDX                                 00029800
002950     PERFORM      WRIT  THRU  WRXIT                                00029900
002960     IF           PARA  EQUAL @01   @                              00030000
002970     PERFORM      ONE   THRU  ONEX                                 00030100
002980     GO TO        D77                                              00030200
002990     IF           PARAG EQUAL @WORKING-STORAGE @                   00030300
           MOVE         @ 4@  TO    IDLBL                                00030400
           MOVE         BKLOC TO    ADR                                  00030500
```

```
           MOVE        DWORD TO    MIASM                                    00030600
003020     PERFORM     WRIT  THRU  WRXIT                                    00030700
003030     MOVE        00001 TO    LAST1                                    00030800
003040     GO TO       DDRD                                                 .00030900
003050     IF          PARAG NOT   EQUAL @PROCEDURE DIVISI@                 00031000
003060     GO TO       DDRD                                                 .00031100
           MOVE        @ 5@  TO    IDLBL                                    .00031200
           MOVE        CORLC TO    ADR                                      .00031300
           MOVE        DWORD TO    MIASM                                    .00031400
003080     PERFORM     WRIT  THRU  WRXIT                                    .00031500
003090     MOVE        1     TO    SUB                                      .00031600
003100 EVDPA                                                                .00031700
003110     IF          EDT   %SUB< NOT   EQUAL SPACE                        00031800
003120     MOVE        EDT   %SUB< TO    MIASM                              00031900
003130     PERFORM     WRIT  THRU  WRXIT                                    00032000
003140     ADD         1     TO    SUB                                      00032100
003150     GO TO       EVDPA                                                .00032200
003250 DDXIT                                                                .00032300
003260     EXIT                                                             .00032400
      ***************************************************************************00032500
003280*         PROCEDURE DIVISION CONTROL                                   00032600
003290***************************************************************************00032700
003300 PROC                                                                 .00032800
           MOVE        SPACE TO    JMP2T                                    .00032900
           MOVE        SPACE TO    EDITT                                    .00033000
003310     MOVE        1     TO    PPNO                                     .00033100
           MOVE        ZERO  TO    FLOP                                     .00033200
           MOVE        SPACE TO    TAB2                                     .00033300
           MOVE        @-@   TO    TST                                      .00033400
003330     MOVE        @*@   TO    PPCOD                                    .00033500
003340     MOVE        ZERO  TO    SUBS1                                    .00033600
003350     MOVE        1     TO    SUBR                                     .00033700
           MOVE        ZERO  TO    RTN01                                    .00033800
           MOVE        ZERO  TO    PRDY                                     .00033900
           OPEN        OUTPUT      PDIV                                     .00034000
003390 PROC1                                                                .00034100
003400     IF          RDFLG NOT   EQUAL 1                                  00034200
003410     PERFORM     RDCOB THRU  CORDX                                    .00034300
003510     MOVE        ZERO  TO    RDFLG                                    .00034400
003420     ADD         1     TO    SEQ                                      .00034500
           IF          PARA  NOT   EQUAL SPACE                              00034600
           MOVE        PARAG TO    JMPNA %SUBR<                             00034700
           MOVE        SEQ   TO    JMPSQ %SUBR<                             00034800
           SUBTRACT    1     FROM  SEQ                                      00034900
           ADD         1     TO    SUBR                                     00035000
003580     GO TO       PROC1                                                .00035100
           IF          BAK   EQUAL ZERO                                     00035200
003530     GO TO       VRBPR                                                .00035300
           IF          STEND EQUAL SPACE                                    00035400
           GO TO       VRBPR                                                .00035500
           MOVE        ZERO  TO    S2                                       .00035600
           MOVE        IGN-1 TO    IGN-2                                    .00035700
           IF          COL-1 EQUAL @X@                                      00035800
           ADD         1     TO    SEQ                                      00035900
           MOVE        1     TO    S2                                       00036000
           GO TO       PLOPD                                                .00036100
           IF          CODF  EQUAL SPACE                                    00036200
           GO TO       PLOPD                                                .00036300
           MOVE        CODF  TO    TAB1                                     .00036400
           MOVE        1     TO    SUB                                      .00036500
      PLOOP                                                                 .00036600
```

```
           IF          T1    %SUB< EQUAL @X@                                  00036700
           MOVE        1     TO    S2                                         00036800
           ADD         1     TO    SEQ                                        00036900
           GO TO       PLOPD                                                  .00037000
           IF          SUB   NOT   GREATER      THAN  30                      00037100
           ADD         6     TO    SUB                                        00037200
           GO TO       PLOOP                                                  .00037300
       PLOPD                                                                  .00037400
           ADD         1     TO    SEQ                                        .00037500
           MOVE        SEQ   TO    JS02  %PPNO<                               .00037600
003560     SUBTRACT    1     FROM  SEQ                                        .00037700
           IF          S2    EQUAL 1                                          00037800
           SUBTRACT    1     FROM  SEQ                                        .00037900
           MOVE        ZERO  TO    BAK                                        .00038000
           ADD         1     TO    PPNO                                       .00038100
003590 VRBPR                                                                  .00038200
003600     PERFORM     FCVRB THRU  VRBXT                                      .00038300
           MOVE        PWORD TO    POUTS                                      .00038400
           PERFORM     WMS   THRU  WMXT                                       00038500
           MOVE        SPACE TO    TAB2                                       .00038600
           MOVE        SPACE TO    EDITT                                      .00038700
003630     MOVE        ZERO  TO    PBDY                                       .00038800
003640     GO TO       PROC1                                                  .00038900
003650 PRXIT                                                                  .00039000
003660     EXIT                                                               .00039100
003670*****************************************************************************00039200
003680*         RDCOB THRU CORDX READS ONE %1< CARD IMAGE                      00039300
003690*                        AND                                             00039400
       *         FINIS PRINTS OUT THE VALUE TABLE AND IS THE                  *00039500
       *         LOGICAL END OF THIS PROGRAM                                  *00039600
003720*****************************************************************************00039700
003730 RDCOB                                                                  .00039800
           READ        COBIN END   GO TO FINIS                                .00039900
           MOVE        SPACE TO    PRNR                                       .00040000
           MOVE        CSEQ  TO    PRN                                        .00040100
           MOVE        SOR   TO    PRR                                        .00040200
           WRITE       PRNR                                                   .00040300
           ADD         1     TO    CSEQ                                       .00040400
003750     IF          CONT  EQUAL @*@                                        00040500
003760     GO TO       RDCOB                                                  .00040600
           MOVE        SPACE TO    PFILL                                      .00040700
           IF          CONT  NOT   EQUAL @D@                                  00040800
           GO TO       CORDX                                                  .00040900
           IF          DEFG  EQUAL 0                                          00041000
           GO TO       RDCOB                                                  .00041100
003770     GO TO       CORDX                                                  .00041200
003780 FINIS                                                                  .00041300
           CLOSE       LITF                                                   .00041400
003790     CLOSE       COBIN                                                  .00041500
           CLOSE       PDIV                                                   .00041600
           CLOSE       PRNF                                                   .00041700
           OPEN        INPUT PDIV                                             .00041800
           MOVE        @ 8@  TO    MIASM                                      .00041900
           PERFORM     WRIT  THRU  WRXIT                                      .00042000
           PERFORM     P     THRU  PEND                                       .00042100
           MOVE        ZERO  TO    PWORD                                      .00042200
           MOVE        9000  TO    CSEQ                                       .00042300
           MOVE        SEQ   TO    LAST1                                      .00042400
           MOVE        PWORD TO    MIASM                                      .00042500
           PERFORM     WRIT  THRU  WRXIT                                      .00042600
003910     CLOSE       MINIZ                                                  .00042700
```

```
003920      STOP        RUN                                                  .00042800
003930 CORDX                                                                 .00042900
003940      EXIT                                                             .00043000
003950*************************************************************************00043100
003960*         WRIT THRU WRXIT WRITES ONE LINE                               00043200
003970*************************************************************************00043300
003980 WRIT                                                                  .00043400
            WRITE       MINI                                                .00043500
004000      MOVE        SPACE TO    MINI                                    .00043600
004010 WRXIT                                                                 .00043700
004020      EXIT                                                             .00043800
       WMS                                                                  .00043900
            WRITE       POUTS                                               .00044000
            MOVE        SPACE TO    POUTS                                   .00044100
       WMXT                                                                 .00044200
            EXIT                                                            .00044300
       P                                                                    .00044400
            READ        PDIV  END   GO TO PEND                              .00044500
            MOVE        POUTS TO    PWORD                                   .00044600
            IF          RTN01 EQUAL 070                                      00044700
            GO TO       PROCE                                               .00044800
            IF          RTN01 EQUAL 130                                      00044900
            GO TO       PROCE                                               .00045000
            IF          RTN01 EQUAL 120                                      00045100
            GO TO       PROCB                                               .00045200
            IF          RTN01 EQUAL 190                                      00045300
            GO TO       PROCE                                               .00045400
            IF          RTN01 EQUAL 301                                      00045500
            GO TO       PROCE                                               .00045600
            IF          RTN01 GREATER      241                               00045700
            GO TO       P1                                                  .00045800
            IF          RTN01 LESS  THAN   200                               00045900
            GO TO       P1                                                  .00046000
            GO TO       PROCE                                               .00046100
       P1                                                                   .00046200
            MOVE        PWORD TO    MIASM                                   .00046300
            PERFORM     WRIT  THRU  WRXIT                                   .00046400
            GO TO       P                                                   .00046500
       PROCE                                                                .00046600
            MOVE        1     TO    SUB                                     .00046700
            MOVE        LAST1 TO    IGN-2                                   .00046800
            IF          COL-1 EQUAL @*@                                      00046900
            MOVE        LAST1 TO    PPNAM                                    00047000
            MOVE        JSQ2  %PPNO<       TO    LAST1                       00047100
            MOVE        PWORD TO    MIASM                                    00047200
            PERFORM     WRIT  THRU  WRXIT                                    00047300
            GO TO       P                                                   .00047400
            PERFORM     SEQFD THRU  SFQD                                    .00047500
            MOVE        JMPSQ %SUB< TO    LAST1                             .00047600
            MOVE        PWORD TO    MIASM                                   .00047700
            PERFORM     WRIT  THRU  WRXIT                                   .00047800
            GO TO       P                                                   .00047900
       PROCB                                                                .00048000
            MOVE        1     TO    SUB                                     .00048100
            MOVE        FRST1 TO    IGN-2                                   .00048200
            PERFORM     SEQFD THRU  SFQD                                    .00048300
            MOVE        JMPSQ %SUB< TO    FRST1                             .00048400
            MOVE        SECD1 TO    IGN-2                                    00048500
            PERFORM     SEQFD THRU  SFQD                                    .00048600
            MOVE        JMPSQ %SUB< TO    SECD1                             .00048700
            MOVE        PWORD TO    MIASM                                   .00048800
```

```
            PERFORM     WRIT  THRU  WRXIT                                    .00048900
            GO TO       P                                                    .00049000
        PEND                                                                 .00049100
            EXIT                                                             .00049200
        SEQFD                                                                .00049300
            IF          JMPNA %SUB< EQUAL SPACE                              00049400
            MOVE        @PARA REFERENCED NOT FOUND@   TO    MIASM            00049500
            PERFORM     WRIT  THRU  WRXIT                                    00049600
            GO TO       SEQD                                                 .00049700
            IF          JMPNA %SUB< EQUAL IGN-2                              00049800
            GO TO       SEQD                                                 .00049900
            ADD         1     TO    SUB                                      .00050000
            GO TO       SEQFD                                                .00050100
        SEQD                                                                 .00050200
            EXIT                                                             .00050300
004030*****************************************************************************00050400
004040*****           FD PROCESSING                                       *******00050500
004050*****************************************************************************00050600
004060 FDD                                                                    .00050700
            MOVE        1     TO    PPNO                                     .00050800
            MOVE        80    TO    RTN01                                    .00050900
004090      MOVE        ZERO  TO    DWORD                                    .00051000
004100 FDA                                                                    .00051100
004110      IF          PARA  EQUAL @FD  @                                   00051200
004120      MOVE        INSTR TO    IDLBL                                    00051300
004130      GO TO       FDRD                                                 .00051400
            IF          INSTR EQUAL @BLOCK       @                           00051500
004150      MOVE        DATA1 TO    IGN-2                                    00051600
            PERFORM     OOCR  THRU  OCRXT                                    00051700
            MOVE        SUC   TO    PPNO                                     00051800
004180      GO TO       FDRD                                                 .00051900
004190      IF          INSTR EQUAL @RECORD      @                           00052000
004200      MOVE        DATA1 TO    IGN-2                                    00052100
            PERFORM     OOCR  THRU  OCRXT                                    00052200
            MOVE        SUC   TO    RTN01                                    00052300
004230      GO TO       FDRD                                                 .00052400
004240      IF          DATA2 EQUAL @OMITTE@                                 00052500
004250      GO TO       FDRD                                                 .00052600
004260      IF          DATA2 EQUAL @STANDA@                                 00052700
004270      MOVE        01    TO    IDDEC                                    00052800
004280      GO TO       FDRD                                                 .00052900
004290      IF          PARA  NOT   EQUAL @01  @                             00053000
004300      GO TO       FDRD                                                 .00053100
            IF          PPNO  GREATER     THAN  1                            00053200
            MULTIPLY    RTN01 BY    PPNO  GIVING      IDLNG                  00053300
            MOVE        BKLOC TO    ADR                                      00053400
            ADD         IDLNG TO    BKLOC                                    .00053500
            MOVE        8     TO    IDPIC                                    .00053600
004340      MOVE        DWORD TO    MINI                                     .00053700
004350      GO TO       FDDX                                                 .00053800
004360 FDRD                                                                   .00053900
004370      PERFORM     RDCOB THRU  CORDX                                    .00054000
004380      GO TO       FDA                                                  .00054100
004390 FDDX                                                                   .00054200
004400      EXIT                                                             .00054300
010920*****************************************************************************00054400
010930*****      HANDLES 01@S, 02@S, 03@S, 04@S, 05@S AND 77@S       *********00054500
010940*****************************************************************************00054600
010950 ONE                                                                    .00054700
010960      MOVE        ZERO  TO    DWORD                                    .00054800
010970      IF          DATA1 NOT   EQUAL @PIC   @                           00054900
```

```
010980      GO TO       ONEAA                                          .00055000
010990      PERFORM     PI     THRU  PIX                               00055100
011040      MOVE        INSTR  TO    IDLBL                           .00055200
011000      IF          CODF   NOT   EQUAL SPACE                       00055300
011010      PERFORM     COD36  THRU  CODX                            .00055400
011020      MOVE        CORLC  TO    ADR                               00055500
011030      ADD         IDLNG  TO    CORLC                             00055600
            MOVE        9      TO    OCR                           .00055700
011050      MOVE        DWORD  TO    MINI                              00055800
011060      PERFORM     WRIT   THRU  WRXIT                             00055900
011070      PERFORM     RDCOB  THRU  CORDX                           .00056000
            IF          DATA1  EQUAL @REDEFI@                          00056100
            SUBTRACT    IDLNG  FROM  CORLC                           .00056200
011080      IF          LAST1  EQUAL 00001                             00056300
011090      GO TO       ONEX                                         .00056400
011100      IF          PARA   EQUAL @01  @                            00056500
011110      SUBTRACT    IDLNG  FROM  CORLC                             00056600
011120      GO TO       ONE                                          .00056700
011130      GO TO       ONEX                                         .00056800
011140 ONEAA                                                         .00056900
011150      MOVE        ZERO   TO    ONET                            .00057000
011160      MOVE        ZERO   TO    SUBS1                           .00057100
011180      MOVE        1      TO    SUB                             .00057200
011190      MOVE        INSTR  TO    IDLBL                           .00057300
011200      MOVE        CORLC  TO    ADR                             .00057400
011210      IF          CODF   NOT   EQUAL SPACE                       00057500
            PERFORM     CODG   THRU  CODGX                           .00057600
            MOVE        9      TO    OCR                           .00057700
011230 ONEA                                                          .00057800
011240      MOVE        DWORD  TO    ONEDE XSUB<                       00057900
011250      ADD         1      TO    SUB                           .00058000
            IF          SUB    EQUAL 51                                00058100
011270      MOVE        @01TOBIG@    TO    MIASM                       00058200
011280      PERFORM     WRIT   THRU  WRXIT                             00058300
011290      GO TO       ONEX                                         .00058400
011300      PERFORM     RDCOB  THRU  CORDX                           .00058500
011310      MOVE        ZERO   TO    DWORD                           .00058600
011320      IF          PARA   EQUAL @WORK@                            00058700
011330      GO TO       ONEJ                                         .00058800
011340      IF          PARA   EQUAL @PROC@                            00058900
011350      GO TO       ONEJ                                         .00059000
011360      IF          PARA   EQUAL @FD  @                            00059100
011370      GO TO       ONEJ                                         .00059200
011380      IF          PARA   EQUAL @01  @                            00059300
011400      GO TO       ONEJ                                         .00059400
011410      MOVE        INSB   TO    IDLBL                           .00059500
            IF          INSB   EQUAL @FILLER  @                        00059600
            MOVE        SPACE  TO    IDLBL                           .00059700
011420      IF          DATA1  NOT   EQUAL @PIC   @                    00059800
            PERFORM     PIS    THRU  PIEXT                             00059900
011440      GO TO       ONEA                                         .00060000
011450      PERFORM     PI     THRU  PIX                             .00060100
011460      MOVE        INSA   TO    LEVL                            .00060200
011470      IF          GF     NOT   EQUAL ZERO                        00060300
011480      PERFORM     ONEGF  THRU  ONGFX                           .00060400
011490      IF          CODF   NOT   EQUAL SPACE                       00060500
011500      PERFORM     COD36  THRU  CODX                            .00060600
011510      IF          FOCC   EQUAL ZERO                              00060700
011520      GO TO       ONEOR                                        .00060800
011530      IF          FOCC   LESS  THAN  LEVL                        00060900
            MOVE        OCLN   TO    VLR                             00061000
```

```
           ADD          IDLNG TO    OCLN                                00061100
011540     MULTIPLY     IDLNG BY    PPNO  GIVING      IDLNG             00061200
011550     MOVE         1     TO    OCR                                 00061300
011560     GO TO        ONEOR                                           . 00061400
           MOVE         70    TO    FOCC                              . 00061500
           MOVE         ZERO  TO    OCLN                              . 00061600
011580 ONEOR                                                           . 00061700
011590     IF           G2    EQUAL ZERO                                00061800
           GO TO        ONE3                                            00061900
011610     IF           02    LESS  THAN  LEVL                          00062000
011620     ADD          IDLNG TO    SUBR                              . 00062100
           GO TO        ONE3                                            00062200
011640     MOVE         SUBR  TO    IDLN  XG2<                        . 00062300
011650     MOVE         ZERO  TO    SUBR                              . 00062400
011660     MOVE         ZERO  TO    G2                                . 00062500
       ONE3                                                            . 00062600
011680     IF           G3    EQUAL ZERO                                00062700
           GO TO        ONE4                                            00062800
011700     IF           03    LESS  THAN  LEVL                          00062900
011710     ADD          IDLNG TO    SUA                               . 00063000
           GO TO        ONE4                                            00063100
011730     MOVE         SUA   TO    IDLN  XG3<                        . 00063200
011740     MOVE         ZERO  TO    SUA                               . 00063300
011750     MOVE         ZERO  TO    G3                                . 00063400
       ONE4                                                            . 00063500
011770     IF           G4    EQUAL ZERO                                00063600
           GO TO        ONE5                                            00063700
011790     IF           04    LESS  THAN  LEVL                          00063800
011800     ADD          IDLNG TO    TNU                               . 00063900
           GO TO        ONE5                                            00064000
011820     MOVE         TNU   TO    IDLN  XG4<                        . 00064100
011830     MOVE         ZERO  TO    TNU                               . 00064200
011840     MOVE         ZERO  TO    G4                                . 00064300
       ONE5                                                            . 00064400
           IF           G5    EQUAL ZERO                                00064500
           GO TO        ONE6                                            00064600
           IF           05    LESS  THAN  LEVL                          00064700
           ADD          IDLNG TO    CG5                               . 00064800
           GO TO        ONE6                                            00064900
           MOVE         CG5   TO    IDLN  XG5<                        . 00065000
           MOVE         ZERO  TO    CG5                               . 00065100
           MOVE         ZERO  TO    G5                                . 00065200
       ONE6                                                            . 00065300
           IF           G6    EQUAL ZERO                                00065400
           GO TO        ONE7                                            00065500
           IF           06    LESS  THAN  LEVL                          00065600
           ADD          IDLNG TO    CG6                               . 00065700
           GO TO        ONE7                                            00065800
           MOVE         CG6   TO    IDLN  XG6<                        . 00065900
           MOVE         ZERO  TO    CG6                               . 00066000
           MOVE         ZERO  TO    G6                                . 00066100
       ONE7                                                            . 00066200
           IF           G7    EQUAL ZERO                                00066300
           GO TO        ONE8                                            00066400
           IF           07    LESS  THAN  LEVL                          00066500
           ADD          IDLNG TO    CG7                               . 00066600
           GO TO        ONE8                                            00066700
           MOVE         CG7   TO    IDLN  XG7<                        . 00066800
           MOVE         ZERO  TO    CG7                               . 00066900
           MOVE         ZERO  TO    G7                                . 00067000
       ONE8                                                            . 00067100
```

```
                IF          G8    EQUAL ZERO                          00067200
                GO TO       ONE9                                      00067300
                IF          08    LESS  THAN  LEVL                    00067400
                ADD         IDLNG TO    CG8                           00067500
                GO TO       ONE9                                      00067600
                MOVE        CG8   TO    IDLN  %G8<                    00067700
                MOVE        ZERO  TO    CG8                           00067800
                MOVE        ZERO  TO    G8                            00067900
            ONE9                                                      00068000
                IF          G9    EQUAL ZERO                          00068100
                GO TO       ONEE                                      00068200
                IF          09    LESS  THAN  LEVL                    00068300
                ADD         IDLNG TO    CG9                           00068400
                GO TO       ONEE                                      00068500
                MOVE        CG9   TO    IDLN  %G9<                    00068600
                MOVE        ZERO  TO    CG9                           00068700
                MOVE        ZERO  TO    G9                            00068800
011850      ONEE                                                      00068900
011860          ADD         IDLNG TO    SUBZ                          00069000
011870          MOVE        CORLC TO    ADR                           00069100
011880          ADD         IDLNG TO    CORLC                         00069200
                IF          FOCC  EQUAL 70                            00069300
                MOVE        ZERO  TO    FOCC                          00069400
                PERFORM     OCGR  THRU  OCGRX                         00069500
011890          GO TO       ONEA                                      00069600
011900      ONEJ                                                      00069700
011910          MOVE        SUBZ  TO    IDLN  %1<                     00069800
011920          IF          G2    NOT   EQUAL ZERO                    00069900
011930          MOVE        SUBR  TO    IDLN  %G2<                    00070000
011940          IF          G3    NOT   EQUAL ZERO                    00070100
011950          MOVE        SUA   TO    IDLN  %G3<                    00070200
011960          IF          G4    NOT   EQUAL ZERO                    00070300
011970          MOVE        TNU   TO    IDLN  %G4<                    00070400
                IF          G5    NOT   EQUAL ZERO                    00070500
                MOVE        CG5   TO    IDLN  %G5<                    00070600
                IF          G6    NOT   EQUAL ZERO                    00070700
                MOVE        CG6   TO    IDLN  %G6<                    00070800
                IF          G7    NOT   EQUAL ZERO                    00070900
                MOVE        CG7   TO    IDLN  %G7<                    00071000
                IF          G8    NOT   EQUAL ZERO                    00071100
                MOVE        CG8   TO    IDLN  %G8<                    00071200
                IF          G9    NOT   EQUAL ZERO                    00071300
                MOVE        CG9   TO    IDLN  %G9<                    00071400
                IF          FOCC  NOT   EQUAL ZERO                    00071500
                PERFORM     OCGR  THRU  OCGRX                         00071600
011980          MOVE        1     TO    SUB                           00071700
011990      ONEK                                                      00071800
012000          MOVE        ONEDE %SUB< TO    MIASM                   00071900
012010          PERFORM     WRIT  THRU  WRXIT                         00072000
012020          ADD         1     TO    SUB                           00072100
012030          IF          ONEDE %SUB< NOT   EQUAL ZERO              00072200
012040          GO TO       ONEK                                      00072300
                IF          DATA1 EQUAL @RFDEFI@                      00072400
                SUBTRACT    IDLN  %1<   FROM  CORLC                   00072500
012050          IF          LAST1 EQUAL 00001                         00072600
012060          GO TO       ONEX                                      00072700
                IF          PARA  EQUAL @01   @                       00072800
012080          SUBTRACT    IDLN  %1<   FROM  CORLC                   00072900
012090          GO TO       ONE                                       00073000
012100      ONEX                                                      00073100
012110          EXIT                                                  00073200
```

```
004410*****************************************************************************00073300
      *****          VERB PROCESSING                                          *****00073400
004470*****************************************************************************00073500
004480 FCVRB                                                                    .00073600
004490     IF          INSTR EQUAL TO    @MOVE         @                        00073700
004500     PERFORM     MOVE1 THRU  MVXIT                                        00073800
004510     GO TO       VRBXT                                                    .00073900
004520     IF          INSTR EQUAL TO    @IF           @                        00074000
004530     PERFORM     IF-1  THRU  IFXIT                                        00074100
004540     GO TO       VRBXT                                                    .00074200
004550     IF          INSTR EQUAL TO    @GO TO        @                        00074300
004560     PERFORM     JMP-1 THRU  JPXIT                                        00074400
004570     GO TO       VRBXT                                                    .00074500
004580     IF          INSTR EQUAL @GO           @                            00074600
004590     PERFORM     JMP-1 THRU  JPXIT                                        00074700
004600     GO TO       VRBXT                                                    .00074800
004610     IF          INSTR EQUAL TO    @PERFORM      @                        00074900
004620     PERFORM     PERF1 THRU  PFXIT                                        00075000
004630     GO TO       VRBXT                                                    .00075100
004640     IF          INSTR EQUAL TO    @ADD          @                        00075200
004650     PERFORM     RT-0  THRU  OPTXT                                        00075300
004660     GO TO       VRBXT                                                    .00075400
004670     IF          INSTR EQUAL TO    @SUBTRACT     @                        00075500
004680     PERFORM     RT-0  THRU  OPTXT                                        00075600
004690     GO TO       VRBXT                                                    .00075700
004700     IF          INSTR EQUAL @EXIT         @                            00075800
004710     PERFORM     EXEX  THRU  EXTIT                                        00075900
004720     GO TO       VRBXT                                                    .00076000
004730     IF          INSTR EQUAL TO    @MULTIPLY     @                        00076100
004740     PERFORM     RT-0  THRU  OPTXT                                        00076200
004750     GO TO       VRBXT                                                    .00076300
004 60     IF          INSTR EQUAL TO    @DIVIDE       @                        00076400
004770     PERFORM     RT-0  THRU  OPTXT                                        00076500
004780     GO TO       VRBXT                                                    .00076600
004790     IF          INSTR EQUAL TO    @READ         @                        00076700
004800     PERFORM     READ1 THRU  RDXIT                                        00076800
004810     GO TO       VRBXT                                                    .00076900
004820     IF          INSTR EQUAL TO    @WRITE        @                        00077000
004830     PERFORM     RIT1  THRU  RIXIT                                        00077100
004840     GO TO       VRBXT                                                    .00077200
004850     IF          INSTR EQUAL TO    @OPEN         @                        00077300
004860     PERFORM     OPEN1 THRU  OPXIT                                        00077400
004870     GO TO       VRBXT                                                    .00077500
004880     IF          INSTR EQUAL TO    @CLOSE        @                        00077600
004890     PERFORM     CLOS1 THRU  CLXIT                                        00077700
004900     GO TO       VRBXT                                                    .00077800
004910     IF          INSTR EQUAL TO    @STOP         @                        00077900
004920     PERFORM     STOP1 THRU  STXIT                                        00078000
004930     GO TO       VRBXT                                                    .00078100
004940     IF          INSTR EQUAL TO    @DISPLAY      @                        00078200
004950     PERFORM     DPLAY THRU  DPXIT                                        00078300
004960     GO TO       VRBXT                                                    .00078400
004970     IF          INSTR EQUAL TO    @ACCEPT       @                        00078500
004980     PERFORM     ACPAR THRU  ACXIT                                        00078600
004990     GO TO       VRBXT                                                    .00078700
005000     IF          INSTR EQUAL TO    @ENTER        @                        00078800
005010     PERFORM     ENTR1 THRU  ENXIT                                        00078900
005020     GO TO       VRBXT                                                    .00079000
           IF          INSTR EQUAL @DELETE       @                            00079100
           PERFORM     DELT1 THRU  DELTX                                        00079200
           GO TO       VRBXT                                                    .00079300
```

```
           IF          INSTR EQUAL @CALL         @                      00079400
           PERFORM     CALL1 THRU  CALLX                                00079500
           GO TO       VRBXT                                            00079600
005030     IF          INSTR EQUAL TO    @REWRITE      @                00079700
005040     PERFORM     RTPAR THRU  RTXIT                                00079800
005050 VRBXT                                                            00079900
005060     EXIT                                                         00080000
      *                                                                 *00080100
      *        ALL MODULES BELOW HERE ARE @SUB-ROUTINES@                *00080200
      *                                                                 *00080300
005070*****************************************************************00080400
005090*                          ACCEPT   VERB                         *00080500
005100*****************************************************************00080600
005110 ACPAR                                                            00080700
           IF          IGN-1 EQUAL SPACE                              00080800
           MOVE        DATA1 TO    FRST1                              00080900
           MOVE        010   TO    RTN01                              00081000
           GO TO       ACXIT                                          00081100
           MOVE        4     TO    SBS                                00081200
005230     MOVE        SOR   TO    TAB2                               00081300
005240     PERFORM     SUBED THRU  SUBEX                              00081400
005250     MOVE        250   TO    RTN01                              00081500
005260     MOVE        TAB1  TO    FRST1                              00081600
           MOVE        PWORD TO    POUTS                              00081700
           PERFORM     WMS   THRU  WMXT                               00081800
           ADD         1     TO    SE0                                00081900
005300     MOVE        010   TO    RTN01                              00082000
005360     MOVE        DATA1 TO    FRST1                              00082100
005370 ACXIT                                                            00082200
005380     EXIT                                                         00082300
005390*****************************************************************00082400
005410*                          CLOSE    VERB                         *00082500
005420*****************************************************************00082600
005430 CLOS1                                                            00082700
005440     MOVE        030   TO    RTN01                              00082800
005450     MOVE        DATA1 TO    FRST1                              00082900
005580 CLXIT                                                            00083000
005590     EXIT                                                         00083100
005600*****************************************************************00083200
005620*                          DISPLAY  VERB                         *00083300
005630*****************************************************************00083400
005640 DPLAY                                                            00083500
005680     PERFORM     BRKDN THRU  BRKXT                              00083600
           IF          BB    %1<   NOT   EQUAL SPACE                  00083700
           MOVE        250   TO    RTN01                              00083800
           MOVE        BB    %1<   TO    FRST1                        00083900
           MOVE        PWORD TO    POUTS                              00084000
           PERFORM     WMS   THRU  WMXT                               00084100
           ADD         1     TO    SE0                                00084200
           MOVE        AA    %1<   TO    FRST1                        00084300
           MOVE        040   TO    RTN01                              00084400
           IF          AA    %2<   EQUAL SPACE                        00084500
           GO TO       DPXIT                                          00084600
           MOVE        PWORD TO    POUTS                              00084700
           PERFORM     WMS   THRU  WMXT                               00084800
           ADD         1     TO    SE0                                00084900
           IF          BB    %2<   NOT   EQUAL SPACE                  00085000
           MOVE        250   TO    RTN01                              00085100
           MOVE        BB    %2<   TO    FRST1                        00085200
           MOVE        PWORD TO    POUTS                              00085300
           PERFORM     WMS   THRU  WMXT                               00085400
```

```
           ADD          1      TO    SEQ                                    .00085500
           MOVE         AA     X2<   TO    FRST1                            .00085600
           MOVE         040    TO    RTN01                                  .00085700
006010 DPXIT                                                                .00085800
006020     EXIT                                                             .00085900
006030*****************************************************************************00086000
006050*                          ENTER    VERB                             *00086100
006060*****************************************************************************00086200
006070 ENTR1                                                                .00086300
006080     MOVE         060    TO    RTN01                                  .00086400
006100     IF           DATA1  EQUAL @FORTRA@                               00086500
006110     GO TO        FOREN                                               .00086600
           IF           DATA1  EQUAL @ASSEMB@                               00086700
           ADD          1      TO    RTN01                                  00086800
           GO TO        FOREN                                               .00086900
           MOVE         DATA1  TO    FRST1                                  00087000
006140     GO TO        ENXIT                                               .00087100
006150 FOREN                                                                .00087200
006160     IF           DATA2  EQUAL SPACE                                  00087300
006170     GO TO        ENXIT                                               .00087400
           MOVE         DATA2  TO    FRST1                                  .00087500
           ADD          2      TO    RTN01                                  .00087600
006270 ENXIT                                                                .00087700
006280     EXIT                                                             .00087800
006290*****************************************************************************00087900
006310*                          GO TO    VERB                             *00088000
006320*****************************************************************************00088100
006330 JMP-1                                                                .00088200
006340     IF           IGN-1  EQUAL SPACE                                  00088300
006350     MOVE         070    TO    RTN01                                  00088400
           MOVE         DATA1  TO    LAST1                                  00088500
006370     GO TO        JPXIT                                               .00088600
           MOVE         SPACE  TO    IDLBL                                  .00088700
           PERFORM      BRKDN  THRU  BRKXT                                  .00088800
           MOVE         1      TO    SBS                                    .00088900
      JW                                                                    .00089000
           ADD          1      TO    SBS                                    .00089100
           IF           AA     %SBS< NOT   EQUAL @DEPEND@                   00089200
           GO TO        JW                                                  .00089300
           ADD          2      TO    SBS                                    .00089400
           IF           BB     %SBS< NOT   EQUAL SPACE                     00089500
           MOVE         BB     %SBS< TO    IDLBL                          .00089600
           MOVE         AA     %SBS< TO    IGN-2                          .00089700
           MOVE         1      TO    SBS                                    .00089800
      JWR                                                                   .00089900
           IF           IDLBL  NOT   EQUAL SPACE                           00090000
           MOVE         250    TO    RTN01                                  00090100
           MOVE         IDLBL  TO    FRST1                                  00090200
           MOVE         PWORD  TO    POUTS                                  00090300
           PERFORM      WMS    THRU  WMXT                                   00090400
           ADD          1      TO    SEQ                                    .00090500
           MOVE         200    TO    RTN01                                  .00090600
           MOVE         IGN-2  TO    FRST1                                  .00090700
           MOVE         AA     %SBS< TO    LAST1                          .00090800
           MOVE         SBS    TO    TAB1                                   .00090900
           PERFORM      VALK   THRU  06X                                    .00091000
           MOVE         VNAM   TO    SECD1                                  .00091100
           ADD          1      TO    SBS                                    .00091200
           IF           AA     %SBS< EQUAL @DEPEND@                        00091300
           GO TO        JPXIT                                               .00091400
           MOVE         PWORD  TO    POUTS                                  .00091500
```

```
           PERFORM     WMS    THRU  WMXT                                    .00091600
           ADD         1      TO    SE0                                     .00091700
           GO TO       JWR                                                  .00091800
006950 JPXIT                                                                .00091900
006960     EXIT                                                             .00092000
007210*************************************************************************00092100
007230*                          OPEN      VERB                             *00092200
007240*************************************************************************00092300
007250 OPEN1                                                                .00092400
           IF          DATA1 NOT   EQUAL @OUTPUT@                           00092500
           GO TO       OI-IO                                              .00092600
           MOVE        112   TO    RTN01                                  .00092700
           MOVE        DATA2 TO    FRST1                                  .00092800
           GO TO       OPXIT                                              .00092900
       OI-IO                                                              .00093000
           IF          DATA1 EQUAL @INPUT @                                 00093100
           MOVE        110   TO    RTN01                                  .00093200
           IF          DATA1 EQUAL @I-O   @                                 00093300
           MOVE        111   TO    RTN01                                  .00093400
           MOVE        IGN-1 TO    FRST1                                  .00093500
007360 OPXIT                                                               .00093600
007370     EXIT                                                             .00093700
007720*************************************************************************00093800
007740*                          STOP      VERB                             *00093900
007750*************************************************************************00094000
007760 STOP1                                                                .00094100
007770     MOVE        150   TO    RTN01                                  .00094200
007780     IF          DATA1 EQUAL @RUN   @                                 00094300
007790     GO TO       STXIT                                              .00094400
           MOVE        3     TO    SBS                                    .00094500
007810     MOVE        SOR   TO    TAB2                                     00094600
007820     PERFORM     021   THRU  06X                                    .00094700
           MOVE        LITCN TO    TNU                                    .00094800
007850     MOVE        VNAM  TO    FRST1                                  .00094900
           MOVE        280   TO    RTN01                                  .00095000
007910 STXIT                                                               .00095100
007920     EXIT                                                             .00095200
008010*************************************************************************00095300
008030*                          MOVE      VERB                             *00095400
008040*************************************************************************00095500
008050 MOVE1                                                                .00095600
008090     PERFORM     BRKDN THRU  BRKXT                                    .00095700
008100     MOVE        BB    %1<   TO    FRST1                            .00095800
008110     MOVE        BB    %3<   TO    SECD1                            .00095900
008120     MOVE        SPACE TO    LAST1                                  .00096000
008130     IF          PBDY  EQUAL SPACE                                    00096100
008140     GO TO       MV1                                                .00096200
008150     MOVE        250   TO    RTN01                                  .00096300
           MOVE        PWORD TO    POUTS                                  .00096400
           PERFORM     WMS   THRU  WMXT                                     00096500
008190     ADD         1     TO    SE0                                    .00096600
008200 MV1                                                                  .00096700
008210     MOVE        AA    %1<   TO    FRST1                            .00096800
008220     MOVE        AA    %3<   TO    SECD1                            .00096900
           MOVE        ZERO  TO    LAST1                                  .00097000
008230     MOVE        090   TO    RTN01                                  .00097100
008240 MVXIT                                                               .00097200
008250     EXIT                                                             .00097300
008300*************************************************************************00097400
008310*                          IF        VERB                             *00097500
008290*************************************************************************00097600
```

```
008320 IF-1                                                          .00097700
008360      PERFORM     BRKDN THRU  BRKXT                            .00097800
008370      MOVE        BB    %1<   TO    FRST1                      .00097900
008390      MOVE        SPACE TO    LAST1                            .00098000
            MOVE        BB    %3<   TO    SECD1                      .00098100
            IF          BB    %6<   NOT   EQUAL SPACE                00098200
            MOVE        BB    %6<   TO    SECD1                      .00098300
            IF          BB    %5<   NOT   EQUAL SPACE                00098400
            MOVE        BB    %5<   TO    SECD1                      .00098500
            IF          BB    %4<   NOT   EQUAL SPACE                00098600
            MOVE        BB    %4<   TO    SECD1                      .00098700
008400      IF          PBDY  EQUAL SPACE                            00098800
008410      GO TO       IF-1A                                        .00098900
008430      MOVE        250   TO    RTN01                            .00099000
            MOVE        PWORD TO    POUTS                            .00099100
            PERFORM     WMS   THRU  WMXT                             00099200
            MOVE        ZERO  TO    PBDY                           .00099300
008460      ADD         1     TO    SE0                            .00099400
            MOVE        ZERO  TO    PBDY                           .00099500
008470 IF-1A                                                         .00099600
008480      MOVE        200   TO    RTN01                            .00099700
008490      MOVE        AA    %1<   TO    FRST1                      .00099800
008500      MOVE        2     TO    SUB                            .00099900
008510      IF          AA    %SUB< EQUAL @NOT    @                00100000
008520      ADD         1     TO    SUB                            00100100
008530      ADD         1     TO    RTN01                          .00100200
008540      IF          AA    %SUB< EQUAL @EQUAL  @                00100300
008550      GO TO       IF-2                                         .00100400
008560      IF          AA    %SUB< EQUAL @LESS   @                00100500
008570      ADD         10    TO    RTN01                          00100600
008580      GO TO       IF-2                                         .00100700
008590      IF          AA    %SUB< EQUAL @GREATE@                 00100800
008600      ADD         20    TO    RTN01                          00100900
008610      ADD         1     TO    SUB                            00101000
008620      GO TO       IF-2                                         .00101100
            IF          AA    %SUB< EQUAL @NUMERI@                 00101200
            ADD         30    TO    RTN01                          00101300
            GO TO       IF-4                                         .00101400
008660      IF          AA    %SUB< EQUAL @ALPHAB@                 00101500
008670      ADD         40    TO    RTN01                          00101600
008680      GO TO       IF-4                                         .00101700
008690 IF-2                                                           .00101800
008700      ADD         1     TO    SUB                            .00101900
008710      IF          AA    %SUB< EQUAL @TO     @                00102000
008720      ADD         1     TO    SUB                            00102100
008730      GO TO       IF-3                                         .00102200
008740      IF          AA    %SUB< EQUAL @THAN   @                00102300
008750      ADD         1     TO    SUB                            .00102400
008760 IF-3                                                           .00102500
008770      MOVE        AA    %SUB< TO    SECD1                      .00102600
008780 IF-4                                                           .00102700
            IF          RELF  %1<   EQUAL 1                          00102800
            GO TO       IFXIT                                        .00102900
008790      PERFORM     RDCOB THRU  CORDX                            .00103000
008800      IF          INSTR EQUAL @GO           @                  00103100
008810      MOVE        DATA1 TO    LAST1                            00103200
008820      GO TO       IFXIT                                        .00103300
008830      IF          INSTR EQUAL @GO TO        @                  00103400
008840      MOVE        DATA1 TO    LAST1                            00103500
008920      GO TO       IFXIT                                        .00103600
008860      MOVE        PPNAM TO    LAST1                            .00103700
```

```
008880      MOVE        1      TO    RDFLG                                  .00103800
008890      MOVE        RTN01  TO    FLGNX                                  .00103900
            MOVE        1      TO    BAK                                    .00104000
008900      IF          FLGNX  EQUAL 1                                      .00104100
008910      SUBTRACT    1      FROM  RTN01                                  .00104200
008850      GO TO       IFXIT                                               .00104300
008930      ADD         1      TO    RTN01                                  .00104400
008940 IFXIT                                                                .00104500
008950      EXIT                                                            .00104600
      *****************************************************************************.00104700
008280*         ADD, SUBTRACT, MULTIPLY, AND DIVIDE VERBS                   .00104800
008260*****************************************************************************.00104900
008980 RT-0                                                                 .00105000
009020      PERFORM     BRKDW  THRU  BRKXT                                  .00105100
009030      MOVE        4      TO    SUB                                    .00105200
009040 RT-1                                                                 .00105300
009050      IF          AA     %SUB< EQUAL SPACE                           .00105400
009060      GO TO       RT-2                                                .00105500
009070      IF          AA     %SUB< NOT   EQUAL @ROUNDE@                  .00105600
009080      ADD         1      TO    SUB                                    .00105700
009090      GO TO       RT-1                                                .00105800
009100      MOVE        180    TO    RTN01                                  .00105900
            MOVE        PWORD  TO    POUTS                                  .00106000
            PERFORM     WMS    THRU  WMXT                                   .00106100
009130      ADD         1      TO    SEQ                                    .00106200
009140 RT-2                                                                 .00106300
009150      MOVE        BB     %1<   TO    FRST1                          .00106400
009160      MOVE        BB     %2<   TO    SECD1                          .00106500
            MOVE        BB     %4<   TO    LAST1                          .00106600
            IF          BB     %5<   NOT   EQUAL SPACE                    .00106700
            MOVE        BB     %5<   TO    LAST1                          .00106800
            IF          BB     %3<   NOT   EQUAL SPACE                    .00106900
            MOVE        BB     %3<   TO    SECD1                          .00107000
009180      IF          PBDY   EQUAL SPACE                                 .00107100
009190      GO TO       RT-3                                               .00107200
            IF          SECD1  EQUAL SPACE                                 .00107300
            GO TO       RT-2A                                              .00107400
            IF          LAST1  NOT   EQUAL SPACE                           .00107500
            GO TO       RT-2A                                              .00107600
            IF          AA     %4<   NOT   EQUAL SPACE                     .00107700
            GO TO       RT-2A                                              .00107800
            MOVE        SECD1  TO    LAST1                                 .00107900
       RT-2A                                                               .00108000
009200      MOVE        250    TO    RTN01                                  .00108100
            MOVE        PWORD  TO    POUTS                                  .00108200
            PERFORM     WMS    THRU  WMXT                                   .00108300
009230      ADD         1      TO    SEQ                                    .00108400
009240 RT-3                                                                 .00108500
            MOVE        ZERO   TO    PBDY                                   .00108600
009250      MOVE        AA     %1<   TO    FRST1                          .00108700
            IF          T2     %1<   EQUAL @ADD                           .00108800
009270      MOVE        020    TO    RTN01                                  .00108900
009280      GO TO       ADR-1                                              .00109000
            IF          T2     %1<   EQUAL @SUBTRA@                       .00109100
009300      MOVE        160    TO    RTN01                                  .00109200
009310      GO TO       SUB-1                                              .00109300
            IF          T2     %1<   EQUAL @MULTIP@                       .00109400
009330      MOVE        100    TO    RTN01                                  .00109500
            IF          T2     %1<   EQUAL @DIVIDE@                       .00109600
009350      MOVE        050    TO    RTN01                                  .00109700
009360      MOVE        AA     %3<   TO    SECD1                          .00109800
```

```
           IF          AA    %4<   EQUAL @GIVING@                    00109900
           MOVE        AA    %5<   TO    LAST1                       00110000
009400     GO TO       SECL                                          .00110100
           MOVE        SECD1 TO    LAST1                           .00110200
           GO TO       SECL                                          .00110300
009410 ADR-1                                                         .00110400
009420     IF          AA    %2<   EQUAL @TO    @                    00110500
009430     MOVE        AA    %3<   TO    SECD1                       00110600
           MOVE        AA    %3<   TO    LAST1                       00110700
009440     GO TO       SECL                                          .00110800
009450     IF          AA    %2<   NOT   EQUAL @TO    @              00110900
009470     MOVE        AA    %2<   TO    SECD1                     .00111000
009480     IF          AA    %3<   EQUAL @TO    @                    00111100
009490     MOVE        AA    %4<   TO    LAST1                       00111200
           MOVE        260   TO    RTNO1                           00111300
009500     GO TO       SECL                                          .00111400
009510     IF          AA    %3<   EQUAL @GIVING@                    00111500
           MOVE        AA    %4<   TO    LAST1                       00111600
009540     GO TO       SECL                                          .00111700
009550 SUB-1                                                         .00111800
009560     IF          AA    %2<   NOT   EQUAL @FROM  @              00111900
009570     MOVE        AA    %2<   TO    SECD1                       00112000
009580     MOVE        AA    %4<   TO    LAST1                       00112100
           MOVE        270   TO    RTNO1                           00112200
009600     GO TO       SECL                                          .00112300
009610     MOVE        AA    %3<   TO    SECD1                     .00112400
009630     MOVE        AA    %5<   TO    LAST1                     .00112500
           IF          LAST1 EQUAL SPACE                           00112600
           MOVE        SECD1 TO    LAST1                           .00112700
009650 SECL                                                          .00112800
009660     MOVE        4     TO    SUB                             .00112900
009670 SECL1                                                         .00113000
009680     IF          AA    %SUB< EQUAL SPACE                       00113100
009690     GO TO       OPTXT                                         .00113200
009700     IF          AA    %SUB< EQUAL @ROUNDED@                   00113300
009710     ADD         2     TO    SUB                             00113400
009720     GO TO       SECL1                                         .00113500
009730     IF          AA    %SUB< NOT   EQUAL @SIZE  @              00113600
009740     ADD         1     TO    SUB                             00113700
009750     GO TO       SECL1                                         .00113800
009760     ADD         3     TO    SUB                             .00113900
           MOVE        PWORD TO    POUTS                           .00114000
           PERFORM     WMS   THRU  WMXT                            00114100
009790     ADD         1     TO    SEQ                             .00114200
009800     MOVE        ZERO  TO    PBDY                            .00114300
009810     MOVE        190   TO    RTNO1                           .00114400
009820     MOVE        AA    %SUB< TO    LAST1                     .00114500
009830 OPTXT                                                         .00114600
009840     EXIT                                                      .00114700
      ***************************************************************00114800
      *              CALL VERB                                      *00114900
      ***************************************************************00115000
       CALL1                                                       .00115100
           PERFORM   BRKDN THRU  BRKXT                             .00115200
           MOVE      290   TO    RTNO1                           .00115300
           MOVE      AA    %1<   TO    FRST1                     .00115400
           MOVE      AA    %3<   TO    SECD1                     .00115500
           MOVE      AA    %4<   TO    LAST1                     .00115600
           IF        AA    %5<   EQUAL SPACE                       00115700
           GO TO     CALLX                                       .00115800
           MOVE      PWORD TO    POUTS                           .00115900
```

```
           PERFORM     WMS    THRU  WMXT                                    .00116000
           ADD         1      TO    SE0                                     .00116100
           ADD         1      TO    RTN01                                   .00116200
           MOVE        AA     %5<   TO    FRST1                             .00116300
           MOVE        AA     %6<   TO    SECD1                             .00116400
           MOVE        AA     %7<   TO    LAST1                             .00116500
       CALLX                                                                .00116600
           EXIT                                                             .00116700
      *****************************************************************************00116800
      *                      DELETE VERB                                      00116900
      *****************************************************************************00117000
       DELT1                                                                .00117100
           MOVE        300    TO    RTN01                                   .00117200
           MOVE        DATA1  TO    FRST1                                   .00117300
           IF          IGN-1  EQUAL SPACE                                    00117400
           GO TO       DELTX                                                .00117500
           ADD         1      TO    RTN01                                   .00117600
           PERFORM     RDCOB  THPU  CORDX                                   .00117700
           IF          INSTR  EQUAL @GO          @                          00117800
           MOVE        DATA1  TO    LAST1                                   00117900
           GO TO       DELTX                                                .00118000
           IF          INSTR  EQUAL @GO TO       @                          00118100
           MOVE        DATA1  TO    LAST1                                   00118200
           GO TO       DELTX                                                .00118300
           MOVE        PPNAM  TO    LAST1                                   .00118400
           MOVE        1      TO    RDFLG                                   .00118500
           MOVE        1      TO    BAK                                     .00118600
       DELTX                                                                .00118700
           EXIT                                                             .00118800
007930*****************************************************************************00118900
007950*                      EXIT VERB                                        00119000
007960*****************************************************************************00119100
007970 EXEX                                                                 .00119200
007980     MOVE        170    TO    RTN01                                   .00119300
           IF          DATA1  NOT   EQUAL SPACE                              00119400
           ADD         1      TO    RTN01                                   .00119500
007990 EXTIT                                                                .00119600
008000     EXIT                                                             .00119700
007380*****************************************************************************00119800
007400*                      PERFORM  VERB                                   *00119900
007410*****************************************************************************00120000
007420 PERF1                                                                .00120100
           MOVE        120    TO    RTN01                                   00120200
           MOVE        DATA1  TO    FRST1                                   00120300
           MOVE        DATA2  TO    SECD1                                   00120400
           MOVE        @000010       TO    LAST1                           .00120500
           IF          DATA3  NOT   EQUAL SPACE                              00120600
           MOVE        DATA3  TO    IGN-2                                   00120700
           PERFORM     OOCR   THRU  OCRXT                                   00120800
           MOVE        SUC    TO    VLR
           MOVE        VLR    TO    ADR
           MOVE        ADR    TO    LAST1                                   .00121000
007490 PFXIT                                                                .00121100
007500     EXIT                                                             .00121200
007510*****************************************************************************00121300
007530*                      READ     VERB                                   *00121400
007540*****************************************************************************00121500
007550 READ1                                                                .00121600
007560     MOVE        130    TO    RTN01                                   .00121700
007570     MOVE        DATA1  TO    FRST1                                   .00121800
           MOVE        DATA3  TO    LAST1                                   .00121900
```

```
007610 RDXIT                                                            .00122000
007620      EXIT                                                        .00122100
007630*****************************************************************00122200
007650*                          REWRITE  VERB                         *00122300
007660*****************************************************************00122400
007670 RTPAR                                                            .00122500
007680      MOVE        140   TO    RTNO1                             .00122600
007690      MOVE        DATA1 TO    FRST1                             .00122700
            IF          IGN-1 EQUAL SPACE                              00122800
            GO TO       RTXIT                                         .00122900
            ADD         1     TO    RTNO1                             .00123000
            PERFORM     RDCOB THRU  CORDX                             .00123100
            IF          INSTR EQUAL @GO          @                     00123200
            MOVE        DATA1 TO    LAST1                              00123300
            GO TO       RIXIT                                         .00123400
            IF          INSTR EQUAL @GO TO       @                     00123500
            MOVE        DATA1 TO    LAST1                              00123600
            GO TO       RIXIT                                         .00123700
            MOVE        PPNAM TO    LAST1                             .00123800
            MOVE        1     TO    RDFLG                             .00123900
            MOVE        1     TO    BAK                               .00124000
007700 RTXIT                                                           .00124100
007710      EXIT                                                       .00124200
006970*****************************************************************00124300
006990*                          WRITE    VERB                        *00124400
007000*****************************************************************00124500
007010 RIT1                                                            .00124600
007020      MOVE        DATA1 TO    FRST1                             .00124700
            IF          IGN-1 EQUAL @INVALI@                           00124800
            GO TO       INVW                                          .00124900
007030      IF          IGN-1 EQUAL @BEFORE@                           00125000
007040      GO TO       BEFOR                                         .00125100
007050      IF          DATA2 EQUAL @PAGE  @                           00125200
007060      MOVE        083   TO    RTNO1                              00125300
007070      GO TO       RIXIT                                         .00125400
007080      MOVE        081   TO    RTNO1                             .00125500
            MOVE        @00001@     TO    SECD1                       .00125600
007100      IF          DATA2 NOT   EQUAL SPACE                        00125700
            MOVE        DATA2 TO    SECD1                             .00125800
007120      GO TO       RIXIT                                         .00125900
007130 BEFOR                                                           .00126000
007140      IF          DATA3 EQUAL @PAGE  @                           00126100
            MOVE        082   TO    RTNO1                              00126200
007160      GO TO       RIXIT                                         .00126300
007170      MOVE        080   TO    RTNO1                             .00126400
            MOVE        DATA3 TO    SECD1                             .00126500
            GO TO       RIXIT                                         .00126600
        INVW                                                          .00126700
            MOVE        84    TO    RTNO1                             .00126800
            PERFORM     RDCOB THRU  CORDX                             .00126900
            IF          INSTR EQUAL @GO          @                     00127000
            MOVE        DATA1 TO    LAST1                              00127100
            GO TO       RIXIT                                         .00127200
            IF          INSTR EQUAL @GO TO       @                     00127300
            MOVE        DATA1 TO    LAST1                              00127400
            GO TO       RIXIT                                         .00127500
            MOVE        PPNAM TO    LAST1                             .00127600
            MOVE        1     TO    RDFLG                             .00127700
            MOVE        1     TO    BAK                               .00127800
007190 RIXIT                                                           .00127900
007200      EXIT                                                       .00128000
```

```
009850*****************************************************************************00128100
009860 BRKDW                                                                     .00128200
           MOVE        SPACE TO    QSUB                                        .00128300
           MOVE        1     TO    SUFLG                                       .00128400
           MOVE        1     TO    QDFLG                                       .00128500
008060     MOVE        SOR   TO    TAB2                                        .00128600
           MOVE        SPACE TO    SOR                                         .00128700
           MOVE        0     TO    SUBZ                                        .00128800
           MOVE        2     TO    SBS                                         .00128900
       BRKD1                                                                    .00129000
009870     ADD         1     TO    SBS                                         .00129100
           IF          SBS   EQUAL 11                                          00129200
           PERFORM     RDCOB THRU  CORDX                                       00129300
           MOVE        DATA1 TO    T2    %11<                                  00129400
           MOVE        IGN-1 TO    T2    %12<                                  00129500
           MOVE        DATA2 TO    T2    %13<                                  00129600
           MOVE        DATA3 TO    T2    %14<                                  00129700
           MOVE        DATA4 TO    T2    %15<                                  00129800
           MOVE        DATA5 TO    T2    %16<                                  00129900
           MOVE        DATA6 TO    T2    %17<                                  00130000
           MOVE        DATA7 TO    T2    %18<                                  .00130100
           IF          PARAG NOT   EQUAL SPACE                                 00130200
           MOVE        1     TO    RDFLG                                       00130300
           MOVE        ZERO  TO    SUFLG                                       00130400
           MOVE        ZERO  TO    QDFLG                                       00130500
           PERFORM     QDUMP THRU  QDMPX                                       00130600
           GO TO       BRKXT                                                   .00130700
009880     IF          T2    %SBS< EQUAL SPACE                                 00130800
009890     ADD         1     TO    SBS                                         .00130900
009900     IF          T2    %SBS< EQUAL SPACE                                 00131000
           MOVE        ZERO  TO    SUFLG                                       00131100
           MOVE        ZERO  TO    QDFLG                                       00131200
           PERFORM     QDUMP THRU  QDMPX                                       00131300
009910     GO TO       BRKXT                                                   .00131400
009920     MOVE        T2    %SBS< TO    IGN-2                                 .00131500
009930     IF          COL-1 EQUAL @%@                                         00131600
009940     PERFORM     SUBED THRU  SUBEX                                       00131700
009960     MOVE        TAB1  TO    BB    %SUBZ<                                00131800
           GO TO       BRKD1                                                   .00131900
009990     IF          COL-1 EQUAL QUOTE                                       00132000
010000     PERFORM     Q21   THRU  Q6X                                         00132100
           MOVE        LITCN TO    TNU                                         00132200
010030     ADD         1     TO    SUBZ                                        00132300
010020     MOVE        VNAM  TO    AA    %SUBZ<                                00132400
           MOVE        TAB1  TO    XYZS  %QDFLG<                               00132500
           ADD         1     TO    QDFLG                                       00132600
           GO TO       BRKD1                                                   .00132700
010050     IF          COL-1 NUMERIC                                           00132800
010060     PERFORM     Q21   THRU  Q6X                                         00132900
           MOVE        LITCN TO    TNU                                         00133000
010090     ADD         1     TO    SUBZ                                        00133100
010080     MOVE        VNAM  TO    AA    %SUBZ<                                00133200
           MOVE        TAB1  TO    XYZS  %QDFLG<                               00133300
           ADD         1     TO    QDFLG                                       00133400
           GO TO       BRKD1                                                   .00133500
010120     ADD         1     TO    SUBZ                                        .00133600
010110     MOVE        T2    %SBS< TO    AA    %SUBZ<                          .00133700
           GO TO       BRKD1                                                   .00133800
010140 BRKXT                                                                    .00133900
010150     EXIT                                                                .00134000
       QDUMP                                                                    .00134100
```

```
           ADD          1      TO   QDFLG                                    .00134200
           IF           XYZS   %QDFLG<      EQUAL SPACE                      00134300
           GO TO        QDMPX                                                .00134400
           MOVE         XYZS   %QDFLG<      TO    LITR                       .00134500
           WRITE        LITR                                                 .00134600
           GO TO        QDUMP                                                .00134700
       QDMPX                                                                 .00134800
           EXIT                                                              .00134900
010160*******************************************************************************00135000
      *          EXTRACTS SUBSCRIPTS USED IN SOURCE COBOL                   *00135100
      *******************************************************************************00135200
010170 SUBED                                                                 .00135300
010180     MOVE         SPACE  TO    TAB1                                    .00135400
010190     MOVE         1      TO    SUB                                     .00135500
010200     MOVE         2      TO    SUA                                     .00135600
010210     MOVE         T2     %SBS< TO    TAB3                              .00135700
010220 SUBXT                                                                 .00135800
010230     MOVE         T3     %SUA< TO    T1    %SUB<                       .00135900
010240     ADD          1      TO    SUA                                     .00136000
010250     ADD          1      TO    SUB                                     .00136100
010260     IF           SUA    EQUAL 7                                       00136200
010270     ADD          1      TO    SBS                                     00136300
010280     MOVE         1      TO    SUA                                     00136400
010290     MOVE         T2     %SBS< TO    TAB3                            .00136500
010300     IF           T3     %SUA< NOT   EQUAL @<@                       00136600
010310     GO TO        SUBXT                                                .00136700
           MOVE         TAB1   TO    IGN-2                                   .00136800
           IF           COL-1  NOT   NUMERIC                                 00136900
010340     GO TO        SUBEX                                                .00137000
           MOVE         ZERO   TO    SUFLG                                   .00137100
010350     PERFORM      VALK   THRU  Q6X                                     .00137200
           MOVE         1      TO    SUFLG                                   .00137300
           MOVE         LITCN  TO    TNU                                     .00137400
010370     MOVE         VNAM   TO    TAB1                                    .00137500
010380 SUBEX                                                                 .00137600
010390     EXIT                                                              .00137700
010400*******************************************************************************00137800
010410*    EXTRACTS THE VALUES FOR BOTH NUMERIC AN ALPHANUMERIC AND           00137900
010420*        BUILDS THE VALUE TABLE WITH THE ACTUAL VALUE                   *00138000
010440*******************************************************************************00138100
010450 Q21                                                                   .00138200
           MOVE         ZERO   TO    FLONG                                   .00138300
           MOVE         SPACE  TO    TAB1                                    .00138400
010460     MOVE         1      TO    OCR                                     .00138500
010470     MOVE         T2     %SBS< TO    TAB3                            .00138600
010480     IF           T3     %OCR< EQUAL QUOTE                            00138700
           MOVE         3      TO    FLONG                                   .00138800
010510     MOVE         1      TO    VLR                                     .00138900
010520 VALB                                                                  .00139000
010530     MOVE         T3     %OCR< TO    T1    %VLR<                       .00139100
010540     ADD          1      TO    VLR                                     .00139200
           ADD          1      TO    OCR                                     .00139300
           IF           OCR    EQUAL 7                                       00139400
           GO TO        VALA                                                 .00139500
           IF           T3     %OCR< EQUAL QUOTE                            00139600
           GO TO        VALDD                                                .00139700
010580 VALD                                                                  .00139800
010590     IF           FLONG  EQUAL 3                                       00139900
010600     GO TO        VALDD                                                .00140000
010610     IF           T3     %OCR< EQUAL SPACE                             00140100
010620     GO TO        VALK                                                 .00140200
```

```
010630      GO TO       VALB                                                .00140300
010640 VALDD                                                                .00140400
010650      IF          T3    %OCR< EQUAL QUOTE                             00140500
            MOVE        T3    %OCR< TO    T1    %VLR<                       00140600
010660      MOVE        ZERO  TO    FLONG                                   00140700
010670      GO TO       VALK                                                .00140800
010680      GO TO       VALB                                                .00140900
010690 VALA                                                                 .00141000
010700      IF          SBS   EQUAL 10                                      00141100
010710      GO TO       VALK                                                .00141200
010720      MOVE        1     TO    OCR                                     .00141300
010730      ADD         1     TO    SBS                                     .00141400
010740      MOVE        T2    %SBS< TO    TAB3                            .00141500
010750      GO TO       VALD                                                .00141600
010760 VALK                                                                 .00141700
            IF          SUFLG EQUAL 1                                       00141800
            GO TO       06X                                                 .00141900
010770      MOVE        ZERO  TO    OCR                                     .00142000
            ADD         1     TO    LITCN                                   .00142100
            MOVE        TAB1  TO    LITR                                    .00142200
            WRITE       LITR                                                .00142300
010900 06X                                                                  .00142400
010910      EXIT                                                            .00142500
012120*****************************************************************************00142600
012130*         APPLIES GROUP OPTIONS TO ITS SUBORDINATE ITEMS                  00142700
012140*****************************************************************************00142800
012150 ONEGF                                                                .00142900
012160      IF          GF    LESS  THAN  LEVL                              00143000
012170      GO TO       ONEGA                                               .00143100
012180      MOVE        ZERO  TO    GF                                      .00143200
012190      MOVE        ZERO  TO    UF                                      .00143300
012200      MOVE        ZERO  TO    SF                                      .00143400
012210      GO TO       ONGFX                                               .00143500
012220 ONEGA                                                                .00143600
012230      IF          UF    EQUAL 1                                       00143700
012240      MOVE        1     TO    USGE                                    00143800
012250      GO TO       ONGFX                                               .00143900
012260      MOVE        SF    TO    SYGN                                    .00144000
012270      IF          SF    EQUAL 2                                       00144100
012280      ADD         1     TO    IDLNG                                   00144200
012290      GO TO       ONGFX                                               .00144300
012300      IF          SF    EQUAL 4                                       00144400
            ADD         1     TO    IDLNG
012320 ONGFX                                                                .00144600
012330      EXIT                                                            .00144700
012340*****************************************************************************00144800
012350*         GROUP ITEM PICK-UP                                              00144900
      *****************************************************************************00145000
       PIS                                                                  .00145100
            IF          INSA  EQUAL @02@                                    00145200
            PERFORM     PIS2  THRU  PIS9X                                   00145300
012400      MOVE        SUB   TO    G2                                      00145400
012410      GO TO       PISRO                                               .00145500
012420      IF          INSA  EQUAL @03@                                    00145600
012430      PERFORM     PIS3  THRU  PIS9X                                   00145700
012440      MOVE        SUB   TO    G3                                      00145800
012450      GO TO       PISRO                                               .00145900
012460      IF          INSA  EQUAL @04@                                    00146000
012470      PERFORM     PIS4  THRU  PIS9X                                   00146100
012480      MOVE        SUB   TO    G4                                      .00146200
            GO TO       PISRO                                               .00146300
```

```
                IF          INSA  EQUAL 05                          00146400
                PERFORM     PIS5  THRU  PIS9X                        00146500
                MOVE        SUB   TO    G5                          00146600
                GO TO       PISRO                                   00146700
                IF          INSA  EQUAL 06                          00146800
                PERFORM     PIS6  THRU  PIS9X                        00146900
                MOVE        SUB   TO    G6                          00147000
                GO TO       PISRO                                   00147100
                IF          INSA  EQUAL 07                          00147200
                PERFORM     PIS7  THRU  PIS9X                        00147300
                MOVE        SUB   TO    G7                          00147400
                GO TO       PISRO                                   00147500
                IF          INSA  EQUAL 08                          00147600
                PERFORM     PIS8  THRU  PIS9X                        00147700
                MOVE        SUB   TO    G8                          00147800
                GO TO       PISRO                                   00147900
                IF          INSA  EQUAL 09                          00148000
                PERFORM     PIS9  THRU  PIS9X                        00148100
                MOVE        SUB   TO    G9                          00148200
012490 PISRO                                                        00148300
012500          MOVE        INSA  TO    LEVL                        00148400
012510          IF          GF    NOT   EQUAL ZERO                  00148500
012520          PERFORM     ONEGF THRU  ONGFX                       00148600
012530          IF          FOCC  EQUAL ZERO                        00148700
012540          GO TO       PISR2                                   00148800
012550          IF          FOCC  LESS  THAN  LEVL                  00148900
012560          MOVE        1     TO    OCR                         00149000
                MOVE        OCLN  TO    VLR                         00149100
012570          GO TO       PISR2                                   00149200
                PERFORM     OCGR  THRU  OCGRX                       00149300
                MOVE        ZERO  TO    OCLN                        00149400
012580          MOVE        ZERO  TO    FOCC                        00149500
012590 PISR2                                                        00149600
012600          IF          DATA1 EQUAL @OCCURS@                    00149700
012610          MOVE        DATA2 TO    IGN-2                       00149800
                PERFORM     OOCR  THRU  OCRXT                       00149900
012630          MOVE        SUC   TO    VLR                         00150000
012640          MOVE        SUC   TO    PPNO                        00150100
012650          MOVE        INSA  TO    FOCC                        00150200
                MOVE        2     TO    IOPIC                       00150300
                MOVE        2     TO    OCR                         00150400
012670          GO TO       PISR1                                   00150500
012680          IF          DATA1 EQUAL @REDEFI@                    00150600
012690          PERFORM     REDE  THRU  REDEX                       00150700
012700          GO TO       PISR1                                   00150800
012710          IF          CODF  NOT   EQUAL SPACE                 00150900
012720          PERFORM     CODG  THRU  CODGX                       00151000
012730 PISR1                                                        00151100
012740          MOVE        CORLC TO    ADR                         00151200
012750 PIEXT                                                        00151300
012760          EXIT                                                00151400
        OCGR                                                        00151500
                MOVE        SUB   TO    OCLN                        00151600
        OCGRA                                                       00151700
                SUBTRACT    1     FROM  SUB                         00151800
                IF          IPICT %SUB< EQUAL 2                     00151900
                MOVE        ZERO  TO    IPICT %SUB<                 00152000
                MOVE        OCLN  TO    SUB                         00152100
                GO TO       OCGRX                                   00152200
                IF          SUB   EQUAL 1                           00152300
                GO TO       OCGRX                                   00152400
```

```
           DIVIDE      PPNO  INTO  IDLN  XSUB(                          .00152500
           GO TO       OCGRA                                            .00152600
       OCGRX                                                            .00152700
           EXIT                                                         .00152800
012770*************************************************************************00152900
012780*         TERMINATES PROCESSING FOR A GROUP                        00153000
012790*************************************************************************00153100
012800 PIS2                                                            .00153200
012810     IF            G2    NOT   EQUAL ZERO                         00153300
012820     MOVE          SUBR  TO    IDLN  XG2(                       00153400
012830     MOVE          ZERO  TO    SUBR                             .00153500
012840 PIS3                                                            .00153600
012850     IF            G3    NOT   EQUAL ZERO                         00153700
012860     MOVE          SUB   TO    IDLN  XG3(                       00153800
012870     MOVE          ZERO  TO    G3                               00153900
012880     MOVE          ZERO  TO    SUB                              .00154000
012890 PIS4                                                            .00154100
012900     IF            G4    NOT   EQUAL ZERO                         00154200
012910     MOVE          TNU   TO    IDLN  XG4(                       00154300
012920     MOVE          ZERO  TO    G4                               00154400
012930     MOVE          ZERO  TO    TNU                              .00154500
       PIS5                                                             .00154600
           IF            G5    NOT   EQUAL ZERO                         00154700
           MOVE          CG5   TO    IDLN  XG5(                       00154800
           MOVE          ZERO  TO    G5                               00154900
           MOVE          ZERO  TO    CG5                              .00155000
       PIS6                                                             .00155100
           IF            G6    NOT   EQUAL ZERO                         00155200
           MOVE          CG6   TO    IDLN  XG6(                       00155300
           MOVE          ZERO  TO    G6                               00155400
           MOVE          ZERO  TO    CG6                              .00155500
       PIS7                                                             .00155600
           IF            G7    NOT   EQUAL ZERO                         00155700
           MOVE          CG7   TO    IDLN  XG7(                       00155800
           MOVE          ZERO  TO    G7                               00155900
           MOVE          ZERO  TO    CG7                              .00156000
       PIS8                                                             .00156100
           IF            G8    NOT   EQUAL ZERO                         00156200
           MOVE          CG8   TO    IDLN  XG8(                       00156300
           MOVE          ZERO  TO    G8                               00156400
           MOVE          ZERO  TO    CG8                              .00156500
       PIS9                                                             .00156600
           IF            G9    NOT   EQUAL ZERO                         00156700
           MOVE          CG9   TO    IDLN  XG9(                       00156800
           MOVE          ZERO  TO    G9                               00156900
           MOVE          ZERO  TO    CG9                              .00157000
012940 PIS9X                                                           .00157100
012950     EXIT                                                        .00157200
012960*************************************************************************00157300
012970*         PROCESSES THE PICTURE CLAUSE                             00157400
012980*************************************************************************00157500
012990 PI                                                              .00157600
013000     MOVE          SOP   TO    TAR1                             .00157700
013010     MOVE          30    TO    ADR                              .00157800
013020     IF            T1    XADR( EQUAL 8X8                        00157900
013030     MOVE          8X8   TO    IDPIC                            00158000
013040     GO TO         PIA                                          .00158100
013050     IF            T1    XADR( EQUAL 898                        00158200
013060     MOVE          9     TO    IDPIC                            00158300
013070     GO TO         PIA                                          .00158400
013080     IF            T1    XADR( EQUAL 858                        00158500
```

```
013090      ADD         1      TO    ADR                                  00158600
013100      MOVE        9      TO    IDPIC                                00158700
013110      MOVE        1      TO    SYGN                                 00158800
013120      GO TO       PIA                                               .00158900
013130      IF          T1     %ADR< EQUAL @.@                            00159000
            MOVE        1      TO    IDDEC                                00159100
013140      MOVE        2      TO    FLONG                                00159200
013150      GO TO       PID                                               .00159300
013160      IF          T1     %ADR< EQUAL @V@                            00159400
013170      MOVE        9      TO    IDPIC                                00159500
013180      GO TO       PIV                                               .00159600
013190      GO TO       PID                                               .00159700
013200 PIA                                                                 .00159800
013210      ADD         1      TO    ADR                                  .00159900
013220      IF          T1     %ADR< EQUAL SPACE                          00160000
013230      MOVE        1      TO    IDLNG                                00160100
013240      GO TO       PIX                                               .00160200
013250      IF          T1     %ADR< NOT   EQUAL @%@                      00160300
013260      ADD         1      TO    IDLNG                                00160400
013270      GO TO       PIV                                               .00160500
013280      MOVE        ZERO   TO    SUC                                  .00160600
013290      PERFORM     PIBBB  THRU  PIBX                                 .00160700
            MOVE        SUC    TO    RTNO1                                .00160800
            MOVE        RTNO1  TO    IDLNG                                .00160900
013310      IF          T1     %ADR< EQUAL SPACE                          00161000
013320      GO TO       PIX                                               .00161100
013330      IF          IDPIC  EQUAL 9                                    00161200
013340      GO TO       PIV                                               .00161300
013350 PID                                                                 .00161400
013360      ADD         1      TO    ADR                                  .00161500
013370      IF          T1     %ADR< EQUAL @%@                            00161600
013380      MOVE        ZERO   TO    SUC                                  00161700
013390      PERFORM     PIBBB  THRU  PIBX                                 00161800
013400      MOVE        SUC    TO    RTNO1                                00161900
013410      ADD         RTNO1  TO    IDLNG                                00162000
013420      GO TO       PIDD                                              .00162100
013430      ADD         1      TO    IDLNG                                .00162200
013440 PIDD                                                               .00162300
013450      IF          T1     %ADR< EQUAL @V@                            00162400
013460      MOVE        IDLNG  TO    IDDEC                                00162500
013470      MOVE        2      TO    FLONG                                00162600
013480      ADD         1      TO    ADR                                  00162700
013490      GO TO       PID                                               .00162800
013500 PIDDD                                                              .00162900
013510      IF          T1     %ADR< EQUAL @.@                            00163000
            ADD         1      TO    IDLNG                                00163100
013520      MOVE        IDLNG  TO    IDDEC                                00163200
013530      MOVE        2      TO    FLONG                                00163300
            ADD         1      TO    ADR                                  00163400
013540      GO TO       PID                                               .00163500
013550      IF          T1     %ADR< EQUAL SPACE                          00163600
013560      GO TO       PIP                                               .00163700
013570      IF          ADR    NOT   EQUAL 60                             00163800
013580      GO TO       PID                                               .00163900
013590      MOVE        @TOBIGEDITPIC@       TO     MIASM                 .00164000
013600      PERFORM     WRIT   THRU  WRXIT                                .00164100
013610      GO TO       PIX                                               .00164200
013620 PIV                                                                 .00164300
013630      IF          T1     %ADR< NOT   EQUAL @V@                      00164400
013640      GO TO       PIDDD                                             .00164500
013650      MOVE        2      TO    FLONG                                .00164600
```

```
013660      MOVE        IDLNG TO    IDDEC                                  .00164700
013670      ADD         1     TO    ADR                                    .00164800
013680      IF          T1    %ADR< NOT   EQUAL 9                          00164900
013690      GO TO       PID                                                .00165000
013700      ADD         1     TO    ADR                                    .00165100
013710      IF          T1    %ADR< EQUAL @%@                              00165200
013720      MOVE        ZERO  TO    SUC                                    00165300
013730      PERFORM     PIBBB THRU  PIBX                                   00165400
013740      MOVE        SUC   TO    RTN01                                  00165500
013750      ADD         RTN01 TO    IDLNG                                  00165600
013760      GO TO       PIVA                                               .00165700
013770      IF          T1    %ADR< EQUAL SPACE                            00165800
013780      ADD         1     TO    IDLNG                                  00165900
013790      MOVE        1     TO    IDDEC                                  00166000
013800      MOVE        ZERO  TO    FLONG                                  00166100
013810      GO TO       PIX                                                .00166200
013820      ADD         1     TO    IDLNG                                  00166300
013830      GO TO       PID                                                .00166400
013840 PIVA                                                                 .00166500
013850      IF          T1    %ADR< EQUAL SPACE                            00166600
013860      MOVE        ZERO  TO    FLONG                                  00166700
013870      SUBTRACT    IDDEC FROM  IDLNG GIVING      IDDEC                00166800
013880      GO TO       PIX                                                .00166900
013890      GO TO       PID                                                .00167000
013900 PIP                                                                  .00167100
013910      IF          FLONG EQUAL 2                                      00167200
013920      SUBTRACT    IDDEC FROM  IDLNG GIVING      IDDEC                00167300
013930      MOVE        ZERO  TO    FLONG                                  00167400
013940      MOVE        @N@   TO    IDPIC                                  00167500
013950      GO TO       PIPA                                               .00167600
013960      MOVE        30    TO    EDR                                    .00167700
013970 PIPB                                                                 .00167800
013980      IF          T1    %EDR< EQUAL @%@                              00167900
013990      MOVE        @E@   TO    IDPIC                                  00168000
014000      GO TO       PIPA                                               .00168100
014010      IF          EDR   EQUAL 59                                     00168200
014020      MOVE        @N@   TO    IDPIC                                  00168300
014030      GO TO       PIPA                                               .00168400
014040      ADD         1     TO    EDR                                    00168500
014050      GO TO       PIPB                                               .00168600
014060 PIPA                                                                 .00168700
            MOVE        1     TO    EDR                                    .00168800
014080      MOVE        SPACE TO    TA                                     .00168900
014090      MOVE        IGN-1 TO    T2    %EDR<                            .00169000
014100      IF          ADR   LESS  THAN  36                               00169100
014110      GO TO       PIK                                                .00169200
014120      ADD         1     TO    EDR                                    .00169300
014130      MOVE        DATA2 TO    T2    %EDR<                            .00169400
014140      IF          ADR   LESS  THAN  42                               00169500
014150      GO TO       PIK                                                .00169600
014160      ADD         1     TO    EDR                                    .00169700
            MOVE        DATA3 TO    T2    %EDR<                            .00169800
014180      IF          ADR   LESS  THAN  48                               00169900
014190      GO TO       PIK                                                .00170000
014200      ADD         1     TO    EDR                                    .00170100
014210      MOVE        DATA4 TO    T2    %EDR<                            .00170200
014220      IF          ADR   LESS  THAN  54                               00170300
014230      GO TO       PIK                                                .00170400
014240      ADD         1     TO    EDR                                    .00170500
014250      MOVE        DATA5 TO    T2    %EDR<                            .00170600
       PIK                                                                 .00170700
```

```
014270      MOVE        1      TO    EDR                                    .00170800
014280 PIL                                                                  .00170900
014290      IF          TA     EQUAL EDT    %EDR<                          00171000
014300      GO TO       PIX                                                .00171100
014310      IF          EDT    %EDR< EQUAL SPACE                           00171200
014320      MOVE        TA     TO    EDT    %EDR<                          00171300
014330      GO TO       PIX                                                .00171400
014340      ADD         1      TO    EDR                                    .00171500
014350      IF          EDR    NOT   EQUAL 100                             00171600
014360      GO TO       PIL                                                .00171700
014370      MOVE        @EDITTAPERROR@     TO    MIASM                     .00171800
014380      PERFORM     WRIT   THRU  WRXIT                                 .00171900
014390 PIX                                                                  .00172000
014400      EXIT                                                           .00172100
014410*****************************************************************    00172200
014420*         EXTRACTS THE PICTURE SIZE                                   00172300
014430*****************************************************************    00172400
014440 PIBBB                                                                .00172500
014450      ADD         2      TO    ADR                                    .00172600
014460      IF          T1     %ADR< EQUAL @<@                             00172700
014470      SUBTRACT    1      FROM  ADR                                    00172800
014480      MOVE        T1     %ADR< TO    S3                             00172900
014490      ADD         2      TO    ADR                                    00173000
014500      GO TO       PIBX                                               .00173100
014510      ADD         1      TO    ADR                                    .00173200
014520      IF          T1     %ADR< EQUAL @<@                             00173300
014530      SUBTRACT    2      FROM  ADR                                    00173400
014540      MOVE        T1     %ADR< TO    S2                             00173500
014550      ADD         1      TO    ADR                                    00173600
014560      MOVE        T1     %ADR< TO    S3                             00173700
014570      ADD         2      TO    ADR                                    00173800
014580      GO TO       PIBX                                               .00173900
014590      SUBTRACT    2      FROM  ADR                                    .00174000
014600      MOVE        T1     %ADR< TO    S1                             00174100
014610      ADD         1      TO    ADR                                    00174200
014620      MOVE        T1     %ADR< TO    S2                             00174300
014630      ADD         1      TO    ADR                                    00174400
014640      MOVE        T1     %ADR< TO    S3                             .00174500
014650      ADD         2      TO    ADR                                    .00174600
014660 PIBX                                                                 .00174700
014670      EXIT                                                           .00174800
014680*****************************************************************    00174900
014690*         GROUP OPTIONS CONTROL                                       00175000
014700*****************************************************************    00175100
014710 CODG                                                                 .00175200
014720      MOVE        SOR    TO    TAB2                                   .00175300
014730      MOVE        5      TO    SEQ                                    .00175400
014740 CODGB                                                                .00175500
014750      IF          T2     %SEQ< EQUAL @VALUE @                        00175600
014760      PERFORM     VAL    THRU  VALX                                   00175700
014770      GO TO       CODGA                                              .00175800
014780 CODGK                                                                .00175900
014790      IF          T2     %SEQ< EQUAL @USAGE @                        00176000
014800      MOVE        INSA   TO    GF                                     00176100
014810      PERFORM     USA    THRU  USAX                                   00176200
014820      MOVE        USGE   TO    UF                                     00176300
014830      GO TO       CODGA                                              .00176400
014840      IF          T2     %SEQ< EQUAL @SIGN  @                        00176500
014850      MOVE        INSA   TO    GF                                     00176600
014860      PERFORM     SIG    THRU  SIGX                                   00176700
014870      MOVE        SYGN   TO    SF                                     00176800
```

```
014880      MOVE        ZERO  TO    SYGN                                  .00176900
014890 CODGA                                                              .00177000
014900      IF          SEQ   NOT   GREATER     9                         00177100
014910      ADD         1     TO    SEQ                                   00177200
014920      GO TO       CODGB                                             .00177300
014930      IF          STEND NOT   EQUAL @.@                             00177400
014940      PERFORM     RDCOB THRU  CORDX                                 00177500
014950      MOVE        SOR   TO    TAB2                                  00177600
014960      MOVE        3     TO    SEQ                                   00177700
014970      GO TO       CODGK                                             .00177800
014980      IF          GF    EQUAL ZERO                                  00177900
            GO TO       CODGX                                             .00178000
015000      IF          PARA  EQUAL @01   @                               00178100
015010      MOVE        01  TO      GF                                     .00178200
015020 CODGX                                                              .00178300
015030      EXIT                                                          .00178400
015040*****************************************************************00178500
015050*         PROCESSES THE VALUE CLAUSE                                00178600
015060*****************************************************************00178700
015070 VAL                                                                .00178800
015080      ADD         1     TO    SEQ                                   .00178900
015090      IF          T2    %SEQ< EQUAL @SPACE @                        00179000
015100      MOVE        999   TO    VLR                                   00179100
015110      GO TO       VALX                                              .00179200
015120      IF          T2    %SEQ< EQUAL @ZERO  @                        00179300
015130      MOVE        998   TO    VLR                                   00179400
015140      GO TO       VALX                                              .00179500
015150      IF          T2    %SEQ< EQUAL @QUOTE @                        00179600
015160      MOVE        997   TO    VLR                                   00179700
015170      GO TO       VALX                                              .00179800
015180      MOVE        SPACE TO    TAB1                                  .00179900
015190      IF          T2    %SEQ< EQUAL SPACE                           00180000
015200      GO TO       VALN                                              .00180100
015210      IF          SEQ   EQUAL 11                                    00180200
015220      GO TO       VALN                                              .00180300
015230      MOVE        SEQ   TO    SBS                                   .00180400
015240 VALH                                                               .00180500
015250      PERFORM     021   THRU  06X                                   .00180600
            MOVE        LITCN TO    VLR                                   .00180700
015260      MOVE        SBS   TO    SEQ                                   .00180800
015270      GO TO       VALX                                              .00180900
015280 VALN                                                               .00181000
015290      PERFORM     RDCOB THRU  CORDX                                 00181100
015300      MOVE        SOR   TO    TAB2                                  .00181200
015310      MOVE        1     TO    SBS                                   .00181300
015320      GO TO       VALH                                              .00181400
015330 VALX                                                               .00181500
015340      EXIT                                                          .00181600
015350*****************************************************************00181700
015360*         ELEMENTARY ITEM OPTIONS CONTROL                           00181800
015370*****************************************************************00181900
015380 COD36                                                              .00182000
015390      MOVE        SOR   TO    TAB2                                  .00182100
015400      MOVE        5     TO    SEQ                                   .00182200
015410 CODA                                                               .00182300
015420      IF          T2    %SEQ< EQUAL @VALUE @                        00182400
015430      PERFORM     VAL   THRU  VALX                                  00182500
015440      GO TO       CODC                                              .00182600
015450      IF          T2    %SEQ< EQUAL @OCCURS@                        00182700
            MOVE        2     TO    OCR                                   00182800
015470      ADD         2     TO    SEQ                                   00182900
```

```
015480      MOVE        T2     %SEQ< TO    IGN-2                          00183000
            PERFORM     OOCR   THRU  OCRXT                                00183100
015500      MOVE        SUC    TO    VLR                                    00183200
            MULTIPLY    IDLNG  BY    VLR   GIVING      IDLNG               00183300
015520      ADD         1      TO    SEQ                                    00183400
015530      GO TO       CODC                                                00183500
015540 CODB                                                                 00183600
015550      IF          T2     %SEQ< EQUAL @BLANK @                        00183700
015560      MOVE        1      TO    BAK                                    00183800
015570      ADD         1      TO    SEQ                                    00183900
015580      GO TO       CODC                                                00184000
015590      IF          T2     %SEQ< EQUAL @SYNC  @                        00184100
015600      PERFORM     SYN    THRU  SYNX                                   00184200
015610      GO TO       CODC                                                00184300
015620      IF          T2     %SEQ< EQUAL @USAGE @                        00184400
015630      PERFORM     USA    THRU  USAX                                   00184500
015640      GO TO       CODC                                                00184600
015650      IF          T2     %SEQ< EQUAL @SIGN  @                        00184700
015660      PERFORM     SIG    THRU  SIGX                                   00184800
015670 CODC                                                                 00184900
015680      IF          SEQ    NOT   GREATER     9                         00185000
015690      ADD         1      TO    SEQ                                    00185100
015700      GO TO       CODA                                                00185200
015710      IF          STEND  EQUAL @.@                                   00185300
015720      GO TO       CODX                                                00185400
015730      PERFORM     RDCOB  THRU  CORDX                                  00185500
015740      MOVE        SOR    TO    TAB2                                   00185600
015750      MOVE        3      TO    SEQ                                    00185700
015760      GO TO       CODB                                                00185800
015770 CODX                                                                 00185900
015780      EXIT                                                            00186000
015790*****************************************************************00186100
015800*        RIGHT JUSTIFIES NUMERIC FIELDS SUCH AS BLOCK CONTAINS    00186200
015810*****************************************************************00186300
       OOCR                                                                00186400
015830      MOVE        ZERO   TO    SUC                                    00186500
015840      IF          COL-2  EQUAL TO    SPACE                          00186600
015850      MOVE        COL-1  TO    S3                                     00186700
015860      GO TO       OCRXT                                               00186800
015870      IF          COL-3  EQUAL TO    SPACE                          00186900
015880      MOVE        COL-1  TO    S2                                     00187000
015890      MOVE        COL-2  TO    S3                                     00187100
015900      GO TO       OCRXT                                               00187200
015910      MOVE        COL-1  TO    S1                                     00187300
015920      MOVE        COL-2  TO    S2                                     00187400
015930      MOVE        COL-3  TO    S3                                     00187500
015940 OCRXT                                                                00187600
015950      EXIT                                                            00187700
015960*****************************************************************00187800
015970*****          PROCESSES THE REDEFINES CLAUSE              *****00187900
015980*****************************************************************00188000
015990 REDE                                                                 00188100
016000      MOVE        SUB    TO    SEQ                                    00188200
016010 REDER                                                                00188300
016020      SUBTRACT    1      FROM  SEQ                                    00188400
016030      IF          NA     %SEQ< EQUAL DATA2                           00188500
016040      SUBTRACT    IDLN   %SEQ< FROM  CORLC                           00188600
016050      GO TO       REDER                                               00188700
016060      IF          SEQ    NOT   EQUAL 1                                00188800
016070      GO TO       REDER                                               00188900
016080      MOVE        @NONAMEREDEFINES@ TO    MIASM                      00189000
```

```
016090      PERFORM     WRIT  THRU  WRXIT                                    .00189100
016100      GO TO       REDEX                                                .00189200
016110 REDEB                                                                 .00189300
            IF          FOCC  NOT   EQUAL ZERO                               00189400
            DIVIDE      PPNO  INTO  IDLN  %SEQ< GIVING      IDLNG            00189500
            SUBTRACT    IDLNG FROM  OCLN                                     00189600
            MOVE        OCLN  TO    VLR                                      .00189700
016120      SUBTRACT    IDLN  %SEQ< FROM  SUBZ                               .00189800
016130      IF          INSA  EQUAL @02@                                     00189900
016140      GO TO       REDEX                                                .00190000
016150      IF          INSA  EQUAL @03@                                     00190100
016160      SUBTRACT    IDLN  %SEQ< FROM  SUBR                               00190200
016170      GO TO       REDEX                                                .00190300
016180      IF          INSA  EQUAL @04@                                     00190400
016190      SUBTRACT    IDLN  %SEQ< FROM  SUBR                               00190500
016200      SUBTRACT    IDLN  %SEQ< FROM  SUA                                .00190600
016210 REDEX                                                                 .00190700
016220      EXIT                                                             .00190800
016230*****************************************************************************00190900
016240*****              PROCESSES SIGN CLAUSE                          ***** 00191000
016250*****************************************************************************00191100
016260 SIG                                                                   .00191200
016270      ADD         1     TO    SEQ                                      .00191300
016280      IF          SEQ   GREATER     THAN  9                            00191400
016290      PERFORM     RDCOB THRU  CORDX                                    00191500
016300      MOVE        SOR   TO    TAB2                                     00191600
016310      MOVE        3     TO    SEQ                                      .00191700
016320      IF          T2    %SEQ< EQUAL @LEADIN@                           00191800
016330      MOVE        1     TO    SYGN                                     00191900
016340      GO TO       SIGA                                                 .00192000
016350      MOVE        3     TO    SYGN                                     .00192100
016360 SIGA                                                                  .00192200
016370      ADD         2     TO    SEQ                                      .00192300
016380      IF          SEQ   LESS  THAN  10                                 00192400
016390      GO TO       SIGB                                                 .00192500
016400      IF          STEND EQUAL @ @                                      00192600
016410      GO TO       SIGX                                                 .00192700
016420      PERFORM     RDCOB THRU  CORDX                                    .00192800
016430      MOVE        SOR   TO    TAB2                                     .00192900
016440      MOVE        3     TO    SEQ                                      .00193000
016450 SIGB                                                                  .00193100
016460      IF          T2    %SEQ< NOT   EQUAL @SEPARA@                     00193200
016470      SUBTRACT    1     FROM  SEQ                                      00193300
016480      GO TO       SIGX                                                 .00193400
            IF          OCR   EQUAL 2                                        00193500
016500      ADD         VLR   TO    IDLNG                                    00193600
016510      GO TO       SIGC                                                 .00193700
016520      ADD         1     TO    IDLNG                                    .00193800
016530 SIGC                                                                  .00193900
016540      ADD         1     TO    SYGN                                     .00194000
016550 SIGX                                                                  .00194100
016560      EXIT                                                             .00194200
016570*****************************************************************************00194300
016580*****              PROCESSES THE USAGE CLAUSE                     *****00194400
016590*****************************************************************************00194500
016600 USA                                                                   .00194600
016610      ADD         1     TO    SEQ                                      .00194700
016620      IF          SEQ   EQUAL 11                                       00194800
016630      PERFORM     RDCOB THRU  CORDX                                    00194900
016640      MOVE        SOR   TO    TAB2                                     00195000
016650      MOVE        3     TO    SEQ                                      .00195100
```

```
016660      IF          T2     %SEQ< EQUAL @COMP  @                          00195200
016670      MOVE        1      TO    USGE                                    .00195300
016680 USAX                                                                  .00195400
016690      EXIT                                                            .00195500
016700*****************************************************************************00195600
016710*****              PROCESSES THE SYNC CLAUSE                        *****00195700
016720*****************************************************************************00195800
016730 SYN                                                                   .00195900
016740      ADD         1      TO    SEQ                                     .00196000
016750      IF          SEQ    EQUAL 11                                      00196100
016760      PERFORM     RDCOB  THRU  CORDX                                   00196200
016770      MOVE        SOR    TO    TAB2                                    00196300
016780      MOVE        3      TO    SEQ                                     .00196400
016790      IF          T2     %SEQ< EQUAL @LEFT  @                          00196500
016800      MOVE        1      TO    SINC                                    00196600
016810      GO TO       SYNX                                                 .00196700
016820      MOVE        2      TO    SINC                                    .00196800
016830 SYNX                                                                  .00196900
016840      EXIT                                                            .00197000
016850*****************************************************************************00197100
016860**********     PHYSICAL END OF PROGRAM                          **********00197200
016870*****************************************************************************00197300
```

```
000000 IDENTIFICATION DIVISION.                                         PREPRO
000001 PROGRAM-ID       NPP                                            .PREPRO
000002 ENVIRONMENT DIVISION                                            .PREPRO
000003 CONFIGURATION SECTION                                           .PREPRO
000004 SOURCE-COMPUTER                                                 .PREPRO
000005     UNIVAC-1108                                                 .PREPRO
000006 OBJECT-COMPUTER                                                 .PREPRO
000007     UNIVAC-1108                                                 .PREPRO
000008 INPUT-OUTPUT SECTION                                            .PREPRO
000009 FILE-CONTROL.                                                    PREPRO
           SELECT      COBIN ASSIGN      COBIN                        .
           SELECT      LITF  ASSIGN      X2                           .
           SELECT      COBF  ASSIGN      X3                           .
           SELECT      VAOF  ASSIGN      X5                           .
           SELECT      DNOF  ASSIGN      X6                           .
           SELECT      PRF   ASSIGN      X4                           .
000016 DATA DIVISION.                                                  PREPRO
000017 FILE SECTION.                                                   PREPRO
000018 FD  LITF                                                        PREPRO
000019     RECORD      60                                              PREPRO
000020     LABEL       RECORD      OMITTED                            .PREPRO
000021 01  LITR        PIC   X(60).                                   .PREPRO
000022 FD  COBIN                                                       PREPRO
000023     RECORD      30                                              PREPRO
000024     LABEL       RECORDS     OMITTED                            .PREPRO
000025 01  COBR                                                       .PREPRO
000026     02  IDL     PIC   X(5).                                    .PREPRO
000027     02  FILLER  PIC   X(25).                                   .PREPRO
000028 01  COBPN                                                      .PREPRO
000029     02  SEQ     PIC   9(5).                                    .PREPRO
000030     02  RTNO1   PIC   9(3).                                    .PREPRO
000031     02  FRST1   PIC   X(5).                                    .PREPRO
000032     02  SECD1   PIC   X(5).                                    .PREPRO
000033     02  LAST1   PIC   X(5).                                    .PREPRO
000034     02  FILLER  PIC   X(7).                                    .PREPRO
000035 01  IOTB1                                                      .PREPRO
000036     02  RELF    PIC   X(3).                                    .PREPRO
000037     02  KEYN    PIC   X(5).                                    .PREPRO
000038     02  FNAM    PIC   X(5).                                    .PREPRO
000039     02  DTYPE   PIC   X(5).                                    .PREPRO
000040     02  DFNAM   PIC   X(5).                                    .PREPRO
000041     02  FILLER  PIC   X(7).                                    .PREPRO
000042 FD  COBF                                                        PREPRO
000043     RECORD      27                                              PREPRO
000044     LABEL       RECORDS     OMITTED                            .PREPRO
000045 01  COBDN                                                      .PREPRO
000046     02  IDLBL   PIC   X(5).                                    .PREPRO
000047     02  ADR     PIC   9(5).                                    .PREPRO
000048     02  IDPIC   PIC   X.                                       .PREPRO
000049     02  IDLNG   PIC   9(4).                                    .PREPRO
000050     02  IDDEC   PIC   9(2).                                    .PREPPO
000051     02  BAK     PIC   9.                                       .PREPRO
000052     02  SYGN    PIC   9.                                       .PREPPO
000053     02  USGE    PIC   9.                                       .PREPRO
000054     02  SINC    PIC   9.                                       .PREPRO
000055     02  OCR     PIC   9.                                       .PREPRO
000056     02  EDR     PIC   9(2).                                    .PREPRO
000057     02  VLR     PIC   9(3).                                    .PREPRO
000058 FD  PRF                                                         PREPRO
000059     LABEL       RECORD      OMITTED                            .PREPRO
000060 01  PR                                                         .PREPRO
```

```
000061      02  PRA      PIC   X%10<  OCCURS      8     TIMES          . PREPRO
000062 FD   VADF                                                       PREPRO
000063      LABEL         RECORD       OMITTED                       . PREPRO
000064 01   VAL                                                        . PREPRO
000065      02  VU       PIC   X      OCCURS      80    TIMES          . PREPRO
000066 FD   DNDF                                                       PREPRO
000067      RECORD        22 -                                         PREPRO
000068      LABEL         RECORD       OMITTED                       . PREPRO
000069 01   DNR                                                        . PREPRO
000070      02  DNLOC    PIC   9%5<                                   . PREPRO
000071      02  DNTYA    PIC   9                                      . PREPRO
000072      02  DNTYP    PIC   9                                      . PREPRO
000073      02  DNLEN    PIC   9%4<                                   . PREPRO
000074      02  OCLV     PIC   9                                      . PREPRO
000075      02  SYN      PIC   9                                      . PREPRO
000076      02  BNK      PIC   9                                      . PREPRO
000077      02  OCCR     PIC   9%3<                                   . PREPRO
000078      02  DNEPT    PIC   9%3<                                   . PREPRO
000079      02  DECP     PIC   9%2<                                   . PREPRO
000080 WORKING-STORAGE SECTION                                        . PREPRO
000081 77   SAVLG        PIC   9%3<                                   . PREPRO
000082 77   SUBR         PIC   9      VALUE 1                         . PREPRO
000083 01   WKTAB                                                     . PREPRO
000084      02  WAREA    PIC   X%5<                                   . PREPRO
000085      02  FILLER   PIC   9%4<                                   . PREPRO
000086 01   QUINT                                                     . PREPRO
000087      02  SQN      PIC   9%5<                                   . PREPRO
000088      02  COED     PIC   9%3<                                   . PREPRO
000089      02  OP-1     PIC   9%4<                                   . PREPRO
000090      02  OP-2     PIC   9%4<                                   . PREPRO
000091      02  OP-3     PIC   9%4<                                   . PREPRO
000092 01   VALP                                                      . PREPRO
000093      02  HYP      PIC   X                                      . PREPRO
000094      02  NOV      PIC   9%4<                                   . PREPRO
000095 01   WN                                                        . PREPRO
000096      02  VCNT     PIC   9%5<   VALUE ZERO                      . PREPRO
000097      02  VRDF     PIC   9      VALUE ZERO                      . PREPRO
000098      02  SUB      PIC   9%5<                                   . PREPRO
000099      02  SUA      PIC   9%5<                                   . PREPRO
000100      02  DNP      PIC   9%4<   VALUE 1                         . PREPRO
000101      02  ECNT     PIC   9%5<   VALUE ZERO                      . PREPRO
000102      02  SUC      PIC   9%3<                                   . PREPRO
000103      02  VLP      PIC   9%3<   VALUE 1                         . PREPRO
000104      02  HDNP     PIC   9%4<                                   . PREPRO
000105      02  H        PIC   X                                      . PREPRO
000106      02  EDP      PIC   9%3<   VALUE 1                         . PREPRO
000107      02  SVLOC    PIC   9%5<   VALUE ZERO                      . PREPRO
000108      02  FNP      PIC   9%2<   VALUE 1                         . PREPRO
000109 01   EDTA                                                      . PREPRO
000110      02  EDI      PIC   X%30<  OCCURS      30    TIMES          . PREPRO
000111 01   DNTAB                                                     . PREPRO
000112      02  DNNS     OCCURS       400                            . PREPRO
000113      03  DNN      PIC   X%5<                                   . PREPRO
000114      03  DNPTZ    PIC   9%4<                                   . PREPRO
000115 01   IOTAB                                                     . PREPRO
000116      02  IOTBA    OCCURS       10    TIMES                    . PREPRO
000117      03  ION      PIC   X%5<                                   . PREPRO
000118      03  PELFA    PIC   X%3<                                   . PREPRO
000119      03  KEYNA    PIC   X%5<                                   . PREPRO
000120 01   FDTAB                                                     . PREPRO
000121      02  FDTAA    OCCURS       10    TIMES                    . PREPRO
```

```
000122      03  FDN      PIC   X(5)                                          .PREPRO
000123 01   FD01T                                                           .PREPRO
000124      02  FDNTA    OCCURS      30    TIMES                           .PREPRO
000125      03  FDDN     PIC   X(5)                                        .PREPRO
000126      03  FDPT     PIC   9(2)                                        .PREPRO
000127 01   TABE                                                           .PREPRO
000128      02  TE       PIC   X     OCCURS      61    TIMES               .PREPRO
000129 01   WKF                                                            .PREPRO
000130      02  WKA      PIC   9                                           .PREPRO
000131      02  WKB      PIC   9                                           .PREPRO
000132      02  WKC      PIC   9                                           .PREPRO
000133 01   PR2                                                            .PREPRO
000134      02  PRB      PIC   X(23) OCCURS      3     TIMES               .PREPRO
000135      02  FILLER   PIC   X(11)                                       .PREPRO
000136 01   PR3          REDEFINES   PR2                                   .PREPRO
000137      02  PRC      PIC   X     OCCURS      80    TIMES               .PREPRO
000138 01   PR4          REDEFINES   PR2                                   .PREPRO
000139      02  PRD      PIC   X(22) OCCURS      3     TIMES               .PREPRO
000140      02  FILLER   PIC   X(14)                                       .PREPRO
000141 01   PQOUT        REDEFINES   PR2                                   .PREPRO
000142      02  PQTS     PIC   X(20) OCCURS      4     TIMES               .PREPRO
000143 01   EDT                                                            .PREPRO
000144      02  ED       PIC   X     OCCURS      999   TIMES               .PREPRO
000145 01   FDT                                                            .PREPRO
000146      02  FDTT     OCCURS      10    TIMES                           .PREPRO
000147      03  FDSTL    PIC   9(5)                                        .PREPRO
000148      03  DNPTR    PIC   9(4)                                        .PREPRO
000149      03  IOPTR    PIC   9(2)                                        .PREPRO
000150      03  BKLEN    PIC   9(4)                                        .PREPRO
000151      03  LBL      PIC   9                                           .PREPRO
000152      03  RCOD     PIC   9(3)                                        .PREPRO
000153      03  FKEYN    PIC   9(4)                                        .PREPRO
000154 01   IOT                                                            .PREPRO
000155      02  IOD      OCCURS      10    TIMES                           .PREPRO
000156      03  IODD     PIC   X(5)                                        .PREPRO
000157      03  IOF      PIC   X(5)                                        .PREPRO
000158*****************************************************************PREPRO
000159 PROCEDURE DIVISION                                                 .PREPRO
000160*****************************************************************PREPRO
000161 ST                                                                 .PREPRO
000162      OPEN         OUTPUT      PRF                                  .PREPRO
000163      OPEN         INPUT COBIN                                      .PREPRO
000164      OPEN         INPUT LITF                                       .PREPRO
000165      MOVE         SPACE TO    IOTAB                                .PREPRO
000166      MOVE         SPACE TO    EDTA                                 .PREPRO
000167      MOVE         SPACE TO    IOT                                  .PREPRO
000168      MOVE         ZERO  TO    QUINT                                .PREPRO
000169      PERFORM      RED   THRU  REDX  2     TIMES                    PREPRO
000170*****************************************************************PREPRO
000171*****************************************************************PREPRO
000172*****************************************************************PREPRO
000173 ST1                                                                .PREPRO
000174      PERFORM      RED   THRU  REDX                                 .PREPRO
000175      IF           IDL   EQUAL "2"                                  PREPRO
000176      MOVE         1     TO    SUB                                  PREPRO
000177      PERFORM      IOTRD THRU  IOTX                                 .PREPRO
000178      IF           IDL   EQUAL "3"                                  PREPRO
000179      PERFORM      DBLD  THRU  DBLDX                                .PREPRO
000180      IF           IDL   EQUAL "5"                                  PREPRO
000181      PERFORM      EDBLD THRU  EDBDX                                .PREPRO
000182      IF           IDL   EQUAL "8"                                  PREPRO
```

```
000183      PERFORM     ST6    THRU  FDXT                                  PREPRO
000184      PERFORM     VB     THRU  VBX                                   PREPRO
000185      MOVE        1      TO    OP-1                                  PREPRO
000186      PERFORM     SRT-1  THRU  SRT-X                                 PREPRO
000187      MOVE        EDP    TO    SAVLG                                 PREPRO
000188      MOVE        ZERO   TO    OP-1                                  PREPRO
000189      PERFORM     PDIV   THRU  PDBXT                                 .PREPRO
000190      GO TO       ST1                                                .PREPRO
000191*****************************************************************************PREPRO
000192*****************************************************************************PREPRO
000193*****************************************************************************PREPRO
000194 ST6                                                                 .PREPRO
000195      CLOSE       COBF                                               .PREPRO
000196      OPEN        INPUT COBF                                         .PREPRO
000197      OPEN        OUTPUT       VAOF                                  .PREPRO
000198      OPEN        OUTPUT       DNOF                                  .PREPRO
000199      MOVE        SPACE TO     EDT                                   .PREPRO
000200      MOVE        ZERO  TO     FDT                                   .PREPRO
000201      MOVE        SPACE TO     DNTAB                                 .PREPRO
000202      MOVE        SPACE TO     FDTAB                                 .PREPRO
000203      PERFORM     RDD   THRU   RDDX                                  .PREPRO
000204      MOVE        1     TO     SUB                                   .PREPRO
000205      IF          IDLBL EQUAL  4                                     PREPRO
000206      GO TO       CNTSW                                              .PREPRO
000207      MOVE        ZERO  TO     SUA                                   .PREPRO
000208      PERFORM     RDD   THRU   RDDX                                  .PREPRO
000209 CNTSF                                                              .PREPRO
000210      PERFORM     FDPAR THRU   FDPAX                                 .PREPRO
000211      PERFORM     RDD   THRU   RDDX                                  .PREPRO
000212 CNTSD                                                              .PREPRO
000213      IF          OCR   EQUAL  9                                     PREPRO
000214      MOVE        IDLBL TO     FDDN  %FNP<                           PREPRO
000215      MOVE        SUA   TO     FDPT  %FNP<                           PREPRO
000216      ADD         1     TO     FNP                                   .PREPRO
000217      PERFORM     DNPAR THRU   DNPAX                                 .PREPRO
000218      IF          IDLBL EQUAL  4                                     PREPRO
000219      GO TO       CNTSW                                              .PREPRO
000220      IF          IDLBL EQUAL  5                                     PREPRO
000221      GO TO       FDXT                                               .PREPRO
000222      IF          IDPIC EQUAL  8                                     PREPRO
000223      GO TO       CNTSF                                              .PREPRO
000224      GO TO       CNTSD                                              .PREPRO
000225 CNTSW                                                              .PREPRO
000226      PERFORM     RDD   THRU   RDDX                                  .PREPRO
000227 CNTWS                                                              .PREPRO
000228      PERFORM     DNPAR THRU   DNPAX                                 .PREPRO
000229      IF          IDLBL NOT    EQUAL 5                               PREPRO
000230      GO TO       CNTWS                                              .PREPRO
000231 FDXT                                                               .PREPRO
000232      EXIT                                                           .PREPRO
000233*****************************************************************************PREPRO
000234 FDPAR                                                              .PREPRO
000235      ADD         1     TO     SUA                                   .PREPRO
000236      MOVE        IDLBL TO     FDN   %SUA<                           .PREPRO
000237      MOVE        ADR   TO     FDSTL %SUA<                           .PREPRO
000238      MOVE        DNP   TO     DNPTR %SUA<                           .PREPRO
000239      MOVE        1     TO     SUB                                   .PREPRO
000240 FDPAT                                                              .PREPRO
000241      IF          IDLBL EQUAL  ION   %SUB<                           PREPRO
000242      MOVE        SUB   TO     IOPTR %SUA<                           PREPRO
000243      MOVE        RELFA %SUB< TO     RCOD  %SUA<                     PREPRO
```

```
000244      GO TO       FDPAS                                          . PREPRO
000245      IF          SUB   EQUAL 1@                                 PREPRO
000246      MOVE        @ERRORINFDENTRY@  TO    PR                     PREPRO
000247      PERFORM     WPR   THRU  WPRXT                              PREPRO
000248      GO TO       FDPAX                                          . PREPRO
000249      ADD         1     TO    SUB                                . PREPRO
000250      GO TO       FDPAT                                          . PREPRO
000251 FDPAS                                                           . PREPRO
000252      MOVE        IDLNG TO    BKLEN %SUB<                        . PREPRO
000253      MOVE        IDDEC TO    LBL   %SUB<                        . PREPRO
000254 FDPAX                                                           . PREPRO
000255      EXIT                                                       . PREPRO
000256*****************************************************************PREPRO
000257*****************************************************************PREPRO
000258 DBLD                                                            . PREPRO
000259      OPEN        OUTPUT      COBF                               . PREPRO
000260      MOVE        3     TO    IDLBL                              . PREPRO
000261      WRITE       COBDW                                          . PREPRO
000262 DB1                                                             . PREPRO
000263      PERFORM     RED   THRU  REDX                               . PREPRO
000264      IF          IDL   EQUAL @ 5@                               PREPRO
000265      MOVE        COBR  TO    COBDW                              PREPRO
000266      MOVE        ADR   TO    SVLOC                              PREPRO
000267      MOVE        5     TO    IDLBL                              PREPRO
000268      WRITE       COBDW                                          PREPRO
000269      GO TO       DBLDX                                          . PREPRO
000270      IF          IDL   EQUAL @ 4@                               PREPPO
000271      MOVE        COBR  TO    COBDW                              PREPRO
000272      MOVE        ADR   TO    S@N                                PREPRO
000273      MOVE        4     TO    IDLBL                              PREPRO
000274      WRITE       COBDW                                          PREPRO
000275      GO TO       DB1                                            . PREPRO
000276      MOVE        COBR  TO    COBDW                              . PREPRO
000277      WRITE       COBDW                                          . PREPRO
000278      GO TO       DB1                                            . PREPRO
000279 DBLDX                                                           . PREPRO
000280      EXIT                                                       . PREPPO
000281*****************$***********************************************PREPPO
000282 EDBLD                                                           . PREPPO
000283      MOVE        1     TO    SUB                                . PREPRO
000284 EB1                                                             . PREPRO
000285      PERFORM     RED   THRU  REDX                               . PREPRO
000286      IF          IDL   EQUAL @ 8@                               PREPPO
000287      GO TO       EDBDX                                          . PREPPO
000288      MOVE        COBR  TO    EDI   %SUB<                        . PREPPO
000289      ADD         1     TO    SUB                                . PREPRO
000290      IF          SUB   LESS  THAN  31                           PREPRO
000291      GO TO       EB1                                            . PREPRO
000292      MOVE        @30&EDITS@  TO    PR                           . PREPRO
000293      PERFORM     WPR   THRU  WPRXT                              . PREPRO
000294 EDBDX                                                           . PREPRO
000295      EXIT                                                       . PREPRO
000296*****************************************************************PREPRO
000297*****************************************************************PREPRO
000298 IOTBD                                                           . PREPRO
000299      PERFORM     RED   THRU  REDX                               . PREPRO
000300      IF          IDL   EQUAL @ 3@                               PREPPO
000301      GO TO       IOTX                                           . PREPRO
000302      MOVE        DFNAM TO    IOF   %SUB<                        PREPPO
000303      MOVE        DTYPE TO    IODD  %SUB<                        PREPRO
000304      MOVE        FNAM  TO    ION   %SUB<                        PREPRO
```

```
000305      MOVE        RELF  TO    RELFA %SUB<                          PREPRO
000306      MOVE        KEYN  TO    KEYNA %SUB<                         .PREPRO
000307      ADD         1     TO    SUB                                 .PREPRO
000308      GO TO       IOTBD                                           .PREPRO
000309 IOTX                                                             .PREPRO
000310      EXIT                                                        .PREPRO
000311*************************************************************************PREPRO
000312 DNPAR                                                            .PREPRO
000313      MOVE        ZERO  TO    DNR                                 .PREPRO
000314      MOVE        IDLBL TO    DNN   %DNP<                         .PREPRO
000315      MOVE        DNP   TO    DNPTZ %DNP<                         .PREPRO
000316      MOVE        ADR   TO    DNLOC                               .PREPRO
000317      MOVE        IDLNG TO    DNLEN                               .PREPRO
000318      MOVE        BAK   TO    BNK                                 .PREPRO
000319      MOVE        SINC  TO    SYN                                 .PREPRO
000320      IF          IDPIC EQUAL @X@                                 PREPRO
000321      MOVE        2     TO    DNTYP                               PREPRO
000322      GO TO       DNPBR                                           .PREPRO
000323      IF          IDPIC EQUAL @9@                                 PREPRO
000324      MOVE        1     TO    DNTYP                               PREPRO
000325      MOVE        SYGN  TO    DNTYA                               PREPRO
000326      MOVE        IDDEC TO    DECP                                PREPRO
000327      GO TO       DNPBR                                           .PREPRO
000328      IF          IDPIC EQUAL 0                                   PREPRO
000329      MOVE        3     TO    DNTYP                               PREPRO
000330      GO TO       DNPBR                                           .PREPRO
000331      IF          IDPIC EQUAL @N@                                 PREPRO
000332      MOVE        IDDEC TO    DECP                                PREPRO
000333      MOVE        4     TO    DNTYP                               PREPRO
000334      PERFORM     DNPED THRU  DNPEX                               PREPRO
000335      GO TO       DNPBR                                           .PREPRO
000336      IF          IDPIC EQUAL @E@                                 PREPRO
000337      MOVE        5     TO    DNTYP                               PREPRO
000338      PERFORM     DNPED THRU  DNPEX                               .PREPRO
000339 DNPBR                                                            .PREPRO
000340      IF          OCR   EQUAL 9                                   PREPRO
000341      MOVE        2     TO    OCLV                                PREPRO
000342      GO TO       DNPV                                            .PREPRO
000343      IF          OCR   EQUAL ZERO                                PREPRO
000344      GO TO       DNPV                                            .PREPRO
000345      IF          OCR   EQUAL 1                                   PREPRO
000346      MOVE        HDNP  TO    DNLOC                               PREPRO
000347      MOVE        1     TO    OCLV                                PREPRO
000348      MOVE        VLR   TO    OCCR                                PREPRO
000349      GO TO       DNPP                                            .PREPRO
000350      DIVIDE      VLR   INTO  IDLNG GIVING      DNLEN             .PREPRO
000351      MOVE        VLR   TO    OCCR                                .PREPRO
000352      MOVE        DNP   TO    HDNP                                .PREPRO
000353      GO TO       DNPP                                            .PREPRO
000354 DNPV                                                             .PREPRO
000355      IF          VLR   EQUAL ZERO                                PREPRO
000356      GO TO       DNPP                                            .PREPRO
000357      PERFORM     DNPVZ THRU  DNPVX                               .PREPRO
000358      MOVE        IDLNG TO    SUC                                 .PREPRO
000359      MOVE        SUC   TO    WKF                                 .PREPRO
000360      PERFORM     DNPZZ THRU  DNPZX                               .PREPRO
000361      IF          VLR   EQUAL 999                                 PREPRO
000362      MOVE        SPACE TO    TABE                                PREPRO
000363      GO TO       DNPVG                                           .PREPRO
000364      IF          VLR   EQUAL 998                                 PREPRO
000365      MOVE        ZERO  TO    TABE                                PREPRO
```

```
000366      GO TO       DNPVG                                          . PREPRO
000367      IF          VLR   EQUAL 997                                  PREPRO
000368      MOVE        QUOTE TO    TABE                                 PREPRO
000369      GO TO       DNPVG                                          . PREPRO
000370      READ        LITF  END . GO TO EOF                          . PREPRO
000371      MOVE        LITR  TO    TABE                               . PREPRO
000372      IF          TE    %1<   EQUAL QUOTE                          PREPRO
000373      GO TO       DNPVG                                          . PREPRO
000374      MOVE        ZERO  TO    SUC                                . PREPRO
000375      MOVE        1     TO    SUB                                . PREPRO
000376 DNPV9                                                           . PREPRO
000377      IF          TE    %SUB< EQUAL @.@                            PREPRO
000378      ADD         1     TO    SUB                                . PREPRO
000379      IF          TE    %SUB< EQUAL SPACE                          PREPRO
000380      SUBTRACT    SUC   FROM  IDLNG GIVING       SUB               PREPRO
000381      GO TO       DNPVP                                          . PREPRO
000382      ADD         1     TO    SUB                                . PREPRO
000383      ADD         1     TO    SUC                                . PREPRO
000384      GO TO       DNPV9                                          . PREPRO
000385 DNPVP                                                           . PREPRO
000386      IF          SUB   EQUAL ZERO                                 PREPRO
000387      GO TO       DNPV7                                          . PREPRO
000388 DNPV8                                                           . PREPRO
000389      MOVE        0     TO    VU    %VLP<                        . PREPRO
000390      IF          VLP   EQUAL 80                                   PREPRO
000391      PERFORM     WVAL  THRU  WVALX                              . PREPRO
000392      ADD         1     TO    VLP                                . PREPRO
000393      SUBTRACT    1     FROM  SUB                                . PREPRO
000394      IF          SUB   NOT   EQUAL 0                              PREPRO
000395      GO TO       DNPV8                                          . PREPRO
000396 DNPV7                                                           . PREPRO
000397      MOVE        1     TO    SUB                                . PREPRO
000398 DNPV6                                                           . PREPRO
000399      IF          TE    %SUB< EQUAL @.@                            PREPRO
000400      ADD         1     TO    SUB                                . PREPRO
000401      MOVE        TE    %SUB< TO    VU    %VLP<                  . PREPRO
000402      IF          VLP   EQUAL 80                                   PREPRO
000403      PERFORM     WVAL  THRU  WVALX                              . PREPRO
000404      ADD         1     TO    VLP                                . PREPRO
000405      ADD         1     TO    SUB                                . PREPRO
000406      IF          TE    %SUB< NOT   EQUAL SPACE                    PREPRO
000407      GO TO       DNPV6                                          . PREPRO
000408      GO TO       DNPP                                           . PREPRO
000409 DNPV0                                                           . PREPRO
000410      MOVE        2     TO    SUB                                . PREPRO
000411 DNPVM                                                           . PREPRO
000412      MOVE        TE    %SUB< TO    VU    %VLP<                  . PREPRO
000413      IF          VLP   EQUAL 80                                   PREPRO
000414      PERFORM     WVAL  THRU  WVALX                              . PREPRO
000415      ADD         1     TO    VLP                                . PREPRO
000416      ADD         1     TO    SUB                                . PREPRO
000417      IF          TE    %SUB< EQUAL QUOTE                          PREPRO
000418      MOVE        SPACE TO    TE    %SUB<                        . PREPRO
000419      SUBTRACT    1     FROM  IDLNG                                PREPRO
000420      IF          IDLNG NOT   EQUAL 0                              PREPRO
000421      GO TO       DNPVM                                          . PREPRO
000422 DNPP                                                            . PREPRO
000423      ADD         1     TO    DNP                                . PREPRO
000424      WRITE       DNR                                            . PREPRO
000425      PERFORM     RDD   THRU  RDDX                               . PREPRO
000426 DNPAX                                                           . PREPRO
```

```
000427      EXIT                                                                . PREPRO
000428*************************************************************************PREPRO
000429 DNPVZ                                                                    . PREPRO
000430      ADD         1       TO      VCNT                                    . PREPRO
000431      MOVE        1       TO      SUB                                     . PREPRO
000432      MOVE        DNP     TO      TABE                                    . PREPRO
000433 DNPVB                                                                    . PREPRO
000434      MOVE        TE      %SUB<   TO      VU      %VLP<                   . PREPRO
000435      IF          VLP     EQUAL   80                                      PREPRO
000436      PERFORM     WVAL    THRU    WVALX                                   . PREPRO
000437      ADD         1       TO      VLP                                     PREPRO
000438      IF          SUB     NOT     EQUAL   4                               PREPRO
000439      ADD         1       TO      SUB                                     PREPRO
000440      GO TO       DNPVB                                                   . PREPRO
000441 DNPVX                                                                    . PREPRO
000442      EXIT                                                                . PREPRO
000443*************************************************************************PREPRO
000444 DNPZZ                                                                    . PREPRO
000445      MOVE        WKA     TO      VU      %VLP<                           . PREPRO
000446      IF          VLP     EQUAL   80                                      PREPRO
000447      PERFORM     WVAL    THRU    WVALX                                   . PREPRO
000448      ADD         1       TO      VLP                                     . PREPRO
000449      MOVE        WKB     TO      VU      %VLP<                           . PREPRO
000450      IF          VLP     EQUAL   80                                      PREPRO
000451      PERFORM     WVAL    THRU    WVALX                                   . PREPRO
000452      ADD         1       TO      VLP                                     . PREPRO
000453      MOVE        WKC     TO      VU      %VLP<                           . PREPRO
000454      IF          VLP     EQUAL   80                                      PREPRO
000455      PERFORM     WVAL    THRU    WVALX                                   . PREPRO
000456      ADD         1       TO      VLP                                     . PREPRO
000457 DNPZX                                                                    . PREPRO
000458      EXIT                                                                . PREPRO
000459*************************************************************************PREPRO
000460*************************************************************************PREPRO
000461*************************************************************************PREPRO
000462 VB                                                                       . PREPRO
000463      IF          EDP     GREATER         THAN    1                       PREPRO
000464      MOVE        @;@     TO      ED      %EDP<                           . PREPRO
000465      MOVE        ZERO    TO      DNR                                     . PREPRO
000466      MOVE        @SPACE@         TO      DNN     %DNP<                   . PREPRO
000467      MOVE        DNP     TO      DNPTZ   %DNP<                           . PREPRO
000468      PERFORM     DNPVZ   THRU    DNPVX                                   . PREPRO
000469      MOVE        001     TO      SUC                                     . PREPRO
000470      MOVE        SUC     TO      WKF                                     . PREPRO
000471      PERFORM     DNPZZ   THRU    DNPZX                                   . PREPRO
000472      MOVE        @ @     TO      VU      %VLP<                           . PREPRO
000473      IF          VLP     EQUAL   80                                      PREPRO
000474      PERFORM     WVAL    THRU    WVALX                                   . PREPRO
000475      ADD         1       TO      VLP                                     . PREPRO
000476      MOVE        1       TO      DNLEN                                   . PREPRO
000477      MOVE        7       TO      DNTYP                                   . PREPRO
000478      MOVE        SVLOC   TO      DNLOC                                   . PREPRO
000479      ADD         1       TO      SVLOC                                   . PREPRO
000480      MOVE        2       TO      OCLV                                    . PREPRO
000481      WRITE       DNR                                                     . PREPRO
000482      MOVE        ZERO    TO      DNR                                     . PREPRO
000483      ADD         1       TO      DNP                                     . PREPRO
000484      MOVE        @QUOTE@         TO      DNN     %DNP<                   . PREPRO
000485      MOVE        DNP     TO      DNPTZ   %DNP<                           . PREPRO
000486      PERFORM     DNPVZ   THRU    DNPVX                                   . PREPRO
000487      PERFORM     DNPZZ   THRU    DNPZX                                   . PREPRO
```

```
000488      MOVE        QUOTE TO    VU    %VLP<                    . PREPRO
000489      IF          VLP   EQUAL 80                               PREPRO
000490      PERFORM     WVAL  THRU  WVALX                          . PREPRO
000491      ADD         1     TO    VLP                            . PREPRO
000492      MOVE        SVLOC TO    DNLOC                          . PREPRO
000493      ADD         1     TO    SVLOC                          . PREPRO
000494      MOVE        1     TO    DNLEN                          . PREPRO
000495      MOVE        2     TO    DNTYP                          . PREPRO
000496      MOVE        2     TO    OCLV                           . PREPRO
000497      WRITE       DNR                                        . PREPRO
000498      ADD         1     TO    DNP                            . PREPRO
000499      MOVE        @ZERO@      TO    DNN   %DNP<              . PREPRO
000500      MOVE        DNP   TO    DNPTZ %DNP<                    . PREPRO
000501      PERFORM     DNPVZ THRU  DNPVX                          . PREPRO
000502      PERFORM     DNPZZ THRU  DNPZX                          . PREPRO
000503      MOVE        0     TO    VU    %VLP<                    . PREPRO
000504      IF          VLP   EQUAL 80                               PREPRO
000505      PERFORM     WVAL  THRU  WVALX                          . PREPRO
000506      ADD         1     TO    VLP                            . PREPRO
000507      MOVE        6     TO    DNTYP                          . PREPRO
000508      MOVE        SVLOC TO    DNLOC                          . PREPRO
000509      ADD         1     TO    SVLOC                          . PREPRO
000510      WRITE       DNR                                        . PREPRO
000511      ADD         1     TO    DNP                            . PREPRO
000512      MOVE        DNP   TO    EDP                            . PREPRO
000513 VBR                                                         . PREPRO
000514      READ        LITF  END   GO TO EOFVL                    . PREPRO
000515      GO TO       VBA                                        . PREPRO
000516 EOFVL                                                       . PREPRO
000517      PERFORM     EOFZ  THRU  EOFX                             PREPRO
000518      GO TO       VBX                                        . PREPRO
000519 VBA                                                         . PREPRO
000520      MOVE        ZERO  TO    DNR                            . PREPRO
000521      MOVE        SVLOC TO    DNLOC                          . PREPRO
000522      MOVE        2     TO    OCLV                           . PREPRO
000523      PERFORM     DNPVZ THRU  DNPVX                            PREPRO
000524      MOVE        LITR  TO    TABE                           . PREPRO
000525      MOVE        1     TO    SUC                            . PREPRO
000526      MOVE        ZERO  TO    SUB                            PPEPRO
000527      IF          TE    %1<   EQUAL QUOTE                      PREPRO
000528      MOVE        2     TO    DNTYP                          PREPRO
000529      MOVE        2     TO    SUA                            PREPRO
000530      GO TO       VBB                                        . PREPRO
000531      MOVE        1     TO    DNTYP                          . PREPRO
000532      MOVE        1     TO    SUA                            . PREPRO
000533      GO TO       VB9                                        . PREPRO
000534 VBB                                                         . PPEPPO
000535      MOVE        TE    %SUA< TO    PRC   %SUC<              . PREPRO
000536      ADD         1     TO    SUA                            PREPRO
000537      IF          TE    %SUA< NOT   EQUAL QUOTE              PREPRO
000538      ADD         1     TO    SUC                            PREPRO
000539      GO TO       VBB                                        . PREPRO
000540 VBD                                                         . PPEPPO
000541      MOVE        1     TO    SUA                            . PREPRO
000542      MOVE        SUC   TO    WKF                            . PREPRO
000543      PERFORM     DNPZZ THRU  DNPZX                          . PREPRO
000544 VBC                                                         . PREPRO
000545      MOVE        PRC   %SUA< TO    VU    %VLP<              . PREPRO
000546      IF          VLP   EQUAL 80                               PREPRO
000547      PERFORM     WVAL  THRU  WVALX                          . PREPRO
000548      ADD         1     TO    VLP                            . PREPRO
```

```
000549      IF          SUA    NOT   EQUAL SUC                          PREPRO
000550      ADD         1      TO    SUA                                PREPRO
000551      GO TO       VBC                                             .PREPRO
000552      MOVE        SUA    TO    DNLEN                              .PREPRO
000553      ADD         SUA    TO    SVLOC                              .PREPRO
000554      ADD         1      TO    DNP                                .PREPRO
000555      WRITE       DNR                                             .PREPRO
000556      GO TO       VBR                                             .PREPRO
000557 VB9                                                              .PREPRO
000558      IF          TE     %SUA< EQUAL @.@                          PREPRO
000559      MOVE        SUA    TO    SUB                                PREPRO
000560      ADD         1      TO    SUA                                .PREPRO
000561      MOVE        TE     %SUA< TO    PRC   %SUC<                  .PREPRO
000562      ADD         1      TO    SUA                                .PREPRO
000563      IF          TE     %SUA< NOT   EQUAL SPACE                  PREPRO
000564      ADD         1      TO    SUC                                PREPRO
000565      GO TO       VB9                                             .PREPRO
000566      IF          SUB    EQUAL 0                                  PREPRO
000567      GO TO       VBD                                             .PREPRO
000568      SUBTRACT    1      FROM  SUA                                .PREPRO
000569      SUBTRACT    SUB    FROM  SUA   GIVING      DECP             .PREPRO
000570      GO TO       VBD                                             .PREPRO
000571 VBX                                                              .PREPRO
000572      EXIT                                                        .PREPRO
000573*****************************************************************PREPRO
000574*****************************************************************PREPRO
000575 DNPED                                                            .PREPRO
000576      MOVE        EDR    TO    DNEPT                              .PREPRO
000577      IF          EDR    NOT   GREATER     THAN  ECNT             PREPRO
000578      GO TO       DNPEX                                           .PREPRO
000579      MOVE        EDR    TO    ECNT                               .PREPRO
000580      MOVE        EDI    %EDR< TO    TABE                         .PREPRO
000581      MOVE        1      TO    SUB                                .PREPRO
000582 DNPEA                                                           .PREPRO
000583      MOVE        TE     %SUB< TO    ED    %EDP<                  PREPRO
000584      IF          TE     %SUB< EQUAL @D@                          PREPRO
000585      ADD         1      TO    SUB                                .PREPRO
000586      ADD         1      TO    SUB                                .PREPRO
000587      IF          TE     %SUB< EQUAL @R@                          PREPRO
000588      ADD         1      TO    SUB                                .PREPRO
000589      IF          TE     %SUB< EQUAL @%@                          PREPRO
000590      ADD         2      TO    SUB                                PREPRO
000591      MOVE        ZERO   TO    WKF                                PREPRO
000592      GO TO       DNPET                                           .PREPRO
000593 DNPER                                                           .PREPRO
000594      ADD         1      TO    EDP                                .PREPRO
000595      IF          TE     %SUB< EQUAL SPACE                        PREPRO
000596      MOVE        @0@    TO    ED    %EDP<                        PREPRO
000597      ADD         1      TO    EDP                                PREPRO
000598      GO TO       DNPEX                                           .PREPRO
000599      GO TO       DNPEA                                           .PREPRO
000600 DNPET                                                           .PREPRO
000601      MOVE        ED     %EDP< TO    H                          .PREPRO
000602      IF          TE     %SUB< EQUAL @<@                          PREPRO
000603      SUBTRACT    1      FROM  SUB                                PREPRO
000604      MOVE        TE     %SUB< TO    WKC                          PREPRO
000605      ADD         2      TO    SUB                                PREPRO
000606      GO TO       DNPE0                                           .PREPRO
000607      ADD         1      TO    SUB                                .PREPRO
000608      IF          TE     %SUB< EQUAL @<@                          PREPRO
000609      SUBTRACT    1      FROM  SUB                                PREPRO
```

```
000610      MOVE        TE    %SUB< TO    WKC                          PREPRO
000611      SUBTRACT    1     FROM  SUB                                  PREPRO
000612      MOVE        TE    %SUB< TO    WKB                          PREPRO
000613      ADD         3     TO    SUB                                  PREPRO
000614      GO TO       DNPEQ                                            .PREPRO
000615      MOVE        TE    %SUB< TO    WKC                          .PREPRO
000616      SUBTRACT    1     FROM  SUB                                  .PREPRO
000617      MOVE        TE    %SUB< TO    WKB                          .PREPRO
000618      SUBTRACT    1     FROM  SUB                                  .PREPRO
000619      MOVE        TE    %SUB< TO    WKA                          .PREPRO
000620      ADD         4     TO    SUB                                  .PREPRO
000621 DNPEQ                                                           .PREPRO
000622      MOVE        WKF   TO    SUC                                  .PREPRO
000623 DNPES                                                           .PREPRO
000624      SUBTRACT    1     FROM  SUC                                  .PREPRO
000625      IF          SUC   EQUAL ZERO                                 PREPRO
000626      GO TO       DNPER                                            .PREPRO
000627      ADD         1     TO    EDP                                  .PREPRO
000628      MOVE        H     TO    ED    %EDP<                          PREPRO
000629      GO TO       DNPES                                            .PREPRO
000630 DNPEX                                                           .PREPRO
            EXIT                                                       .
000631*****************************************************************PREPRO
000632 RDD                                                             .PREPRO
000633      READ        COBF  END   GO TO EOFF                           .PREPRO
000634      GO TO       RDDX                                             .PREPRO
000635 EOFF                                                            .PREPRO
000636      MOVE        @ERRORINCOBF@     TO    PR                     PREPRO
000637      PERFORM     WPR   THRU  WPRXT                                PREPRO
000638      GO TO       EOF                                              .PREPRO
000639 RDDX                                                            .PREPRO
000640      EXIT                                                       .PREPRO
000641*****************************************************************PREPRO
000642*****************************************************************PREPRO
000643 RED                                                             .PREPRO
000644      READ        COBIN END   GO TO EOF                            .PREPRO
000645      GO TO       REDX                                             .PREPRO
000646 EOF                                                             .PREPRO
000647      CLOSE       LITF                                             .PREPRO
000648      CLOSE       COBIN                                            .PREPRO
000649      CLOSE       COBF                                             .PREPRO
000650      CLOSE       PRF                                              .PREPRO
000651      CLOSE       DNOF                                             .PREPRO
000652      STOP        RUN                                              .PREPRO
000653 REDX                                                            .PREPRO
000654      EXIT                                                       .PREPRO
000655*****************************************************************PREPRO
000656*****************************************************************PREPRO
000657 EOFZ                                                            .PREPRO
000658      MOVE        SPACE TO    PR                                   .PREPRO
000659      MOVE        SPACE TO    PR2                                  .PREPRO
000660      IF          IOD   %1<   EQUAL SPACE                          PREPRO
000661      MOVE        ZERO  TO    PRA   %2<                            PREPRO
000662      WRITE       PR                                               PREPRO
000663      MOVE        SPACE TO    PRA   %2<                            PREPRO
000664      GO TO       EOFB                                             .PREPRO
000665      MOVE        1     TO    SUB                                  .PREPRO
000666 EFA                                                             .PREPRO
000667      IF          RCOD  %SUB< NOT   EQUAL ZERO                     PREPRO
000668      GO TO       EFB                                              .PREPRO
000669 EFC                                                             .PREPRO
```

```
000670      IF          SUB   EQUAL 10                          PREPRO
000671      MOVE        1     TO    SUB                         PREPRO
000672      MOVE        1     TO    SUA                         PREPRO
000673      GO TO       EOFA                                    .PREPRO
000674      ADD         1     TO    SUB                         .PREPRO
000675      GO TO       EFA                                     PREPRO
000676 EFB                                                      .PREPRO
            MOVE        IOPTR %SUB< TO    SUA                   .
000677      MOVE        1     TO    HDNP                        .PREPRO
000678      IF          KEYNA %SUA< EQUAL ZERO                  PREPRO
000679      GO TO       EFC                                     .PREPRO
000680 EFD                                                      .PREPRO
000681      IF          KEYNA %SUA< EQUAL DNN   %HDNP<          PREPRO
000682      MOVE        HDNP  TO    FKEYN %SUB<                 PREPRO
000683      GO TO       EFC                                     .PREPRO
000684      IF          HDNP  EQUAL 500                         PREPRO
000685      GO TO       EFC                                     .PREPRO
000686      ADD         1     TO    HDNP                        PREPRO
000687      GO TO       EFD                                     .PREPRO
000688 EOFA                                                     .PREPRO
000689      MOVE        IOD   %SUB< TO    PRA   %SUA<           .PREPRO
000690      ADD         1     TO    SUB                         .PREPRO
000691      IF          IOD   %SUB< EQUAL SPACE                 PREPRO
000692      SUBTRACT    1     FROM  SUB                         PREPRO
000693      PERFORM     WPR   THRU  WPRXT                       PREPRO
000694      MOVE        SUB   TO    PRA   %2<                   PREPRO
000695      WRITE       PR                                      PREPRO
000696      MOVE        SPACE TO    PRA   %2<                   PREPRO
000697      GO TO       EOFB                                    .PREPRO
000698      IF          SUA   EQUAL 8                           PREPRO
000699      PERFORM     WPR   THRU  WPRXT                       PREPRO
000700      MOVE        1     TO    SUA                         PREPRO
000701      GO TO       EOFA                                    .PREPRO
000702      ADD         1     TO    SUA                         .PREPRO
000703      GO TO       EOFA                                    .PREPRO
000704 EOFB                                                     .PREPRO
000705      IF          FDTT  %1<   EQUAL ZERO                  PREPRO
000706      MOVE        ZERO  TO    PRB   %2<                   PREPRO
000707      MOVE        PR2   TO    PR                          PREPRO
000708      WRITE       PR                                      PREPRO
000709      MOVE        SPACE TO    PRB   %2<                   PREPRO
000710      GO TO       EOFD                                    .PREPRO
000711      MOVE        1     TO    SUB                         .PREPRO
000712      MOVE        1     TO    SUA                         .PREPRO
000713 EOFC                                                     .PREPRO
000714      MOVE        FDTT  %SUB< TO    PRB   %SUA<           .PREPRO
000715      ADD         1     TO    SUB                         .PREPRO
000716      IF          FDTT  %SUB< EQUAL ZERO                  PREPRO
000717      GO TO       EOFCC                                   .PREPRO
000718      IF          SUB   EQUAL 10                          PREPRO
000719      GO TO       EOFCC                                   .PREPRO
000720      IF          SUA   EQUAL 3                           PREPRO
000721      MOVE        PR2   TO    PR                          PREPRO
000722      MOVE        SPACE TO    PR2                         PREPRO
000723      PERFORM     WPR   THRU  WPRXT                       PREPRO
000724      MOVE        1     TO    SUA                         PREPRO
000725      GO TO       EOFC                                    .PREPRO
000726      ADD         1     TO    SUA                         .PREPRO
000727      GO TO       EOFC                                    .PREPRO
000728 EOFCC                                                    .PREPRO
000729      SUBTRACT    1     FROM  SUB                         .PREPRO
```

```
000730      MOVE        PR2   TO    PR                          . PREPRO
000731      MOVE        SPACE TO    PR2                         . PREPRO
000732      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000733      MOVE        SUB   TO    PRB   %2<                   . PREPRO
000734      IF          SQN   NOT   EQUAL ZERO                    PREPRO
000735      SUBTRACT    1     FROM  SQN                         . PREPRO
000736      MOVE        SQN   TO    PRB   %3<                   . PREPRO
000737      MOVE        PR2   TO    PR                          . PREPRO
000738      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000739      MOVE        SPACE TO    PRB   %2<                   . PREPRO
000740      MOVE        SPACE TO    PRB   %3<                   . PREPRO
000741 EOFD                                                     . PREPRO
000742      IF          ED    %1<   EQUAL SPACE                   PREPRO
000743      MOVE        ZERO  TO    PRA   %3<                   . PREPRO
000744      MOVE        ZERO  TO    PRA   %5<                   . PREPRO
000745      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000746      GO TO       EOFG                                    . PREPRO
000747      MOVE        1     TO    SUB                         . PREPRO
000748      MOVE        1     TO    SUA                         . PREPRO
000749 EOFE                                                     . PREPRO
000750      MOVE        ED    %SUB< TO    PRC   %SUA<           . PREPRO
000751      ADD         1     TO    SUB                         . PREPRO
000752      IF          ED    %SUB< EQUAL SPACE                   PREPRO
000753      SUBTRACT    1     FROM  SUB                           PREPRO
000754      MOVE        PR3   TO    PR                          . PREPRO
000755      MOVE        SPACE TO    PR3                         . PREPRO
000756      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000757      MOVE        ECNT  TO    PRA   %3<                   . PREPRO
000758      MOVE        SUB   TO    PRA   %5<                   . PREPRO
000759      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000760      GO TO       EOFG                                    . PREPRO
000761      IF          SUA   EQUAL 80                          . PREPRO
000762      MOVE        PR3   TO    PR                          . PREPRO
000763      MOVE        SPACE TO    PR3                         . PREPRO
000764      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000765      MOVE        ZERO  TO    SUA                         . PREPRO
000766      ADD         1     TO    SUA                         . PREPRO
000767      GO TO       EOFE                                    . PREPRO
000768 EOFG                                                     . PREPRO
000769      MOVE        0     TO    SUB                         . PREPRO
000770      MOVE        1     TO    SUA                         . PREPRO
000771      CLOSE       DNOF                                    . PREPRO
000772      OPEN        INPUT DNOF                              . PREPRO
000773 EOFH                                                     . PREPRO
000774      READ        DNOF  END   GO TO EOFI                  . PREPRO
000775      MOVE        DNR   TO    PRD   %SUA<                 . PREPRO
000776      ADD         1     TO    SUB                         . PREPRO
000777      IF          SUA   EQUAL 3                           . PREPRO
000778      MOVE        PR4   TO    PR                          . PREPRO
000779      MOVE        SPACE TO    PR4                         . PREPRO
000780      PERFORM     WPR   THRU  WPRXT                       . PREPRO
000781      MOVE        1     TO    SUA                         . PREPRO
000782      GO TO       EOFH                                    . PREPRO
000783      ADD         1     TO    SUA                         . PREPRO
000784      GO TO       EOFH                                    . PREPRO
000785 EOFI                                                     . PREPRO
000786      IF          PRD   %1<   EQUAL SPACE                 . PREPRO
000787      GO TO       EOFII                                   . PREPRO
000788      MOVE        99999 TO    PRD   %SUA<                 . PREPRO
000789      MOVE        PR4   TO    PR                          . PPEPRO
000790      MOVE        SPACE TO    PR4                         . PREPRO
```

```
000791      PERFORM     WPR    THRU   WPRXT                          . PREPRO
000792 EOFII                                                         . PREPRO
000793      SUBTRACT    1      FROM   SVLOC                          . PREPRO
000794      MOVE        99999  TO     PRD    %1<                     . PREPRO
000795      MOVE        SVLOC  TO     PRD    %3<                     . PREPRO
000796      MOVE        SUB    TO     PRD    %2<                     . PREPRO
000797      MOVE        PR4    TO     PR                             . PREPRO
000798      MOVE        SPACE  TO     PR4                            . PREPRO
000799      PERFORM     WPR    THRU   WPRXT                          . PREPRO
000800      MOVE        0      TO     SUB                            . PREPRO
000801      IF          VLP    EQUAL  1                               PREPRO
000802      MOVE        SUB    TO     PRA    %5<                     PREPRO
000803      MOVE        SUB    TO     PRA    %3<                     PREPRO
000804      PERFORM     WPR    THRU   WPRXT                          PREPRO
000805      GO TO       EOFX                                         . PREPRO
000806      MOVE        VLP    TO     SUA                            . PREPRO
000807      PERFORM     WVAL   THRU   WVALX                          PREPRO
000808      CLOSE       VAOF                                         . PREPRO
000809      OPEN        INPUT  VAOF                                  . PREPRO
000810      MOVE        VCNT   TO     PRA    %2<                     . PREPRO
000811      PERFORM     WPR    THRU   WPRXT                          . PREPRO
000812      MOVE        SPACE  TO     PRA    %2<                     . PREPRO
000813 EOFJ                                                          . PREPRO
000814      READ        VAOF   END    GO TO EOFQ                     . PREPRO
000815      ADD         1      TO     SUB                            . PREPRO
000816      MOVE        VAL    TO     PR                             . PREPRO
000817      PERFORM     WPR    THRU   WPRXT                          . PREPRO
            GO TO       EOFJ                                        .
000818 EOFQ                                                          . PREPRO
000819      MOVE        VCNT   TO     PRA    %3<                     . PREPRO
000820      SUBTRACT    1      FROM   SUB                            . PREPRO
000821      MULTIPLY    80     BY     SUB                            . PREPRO
000822      ADD         SUA    TO     SUB                            . PREPRO
000823      MOVE        SUB    TO     PRA    %5<                     . PREPRO
000824      CLOSE       VAOF                                         . PREPRO
000825      PERFORM     WPR    THRU   WPRXT                          . PREPRO
000826 EOFX                                                          . PREPRO
000827      EXIT                                                     . PREPRO
000828*************************************************************************PREPRO
000829*************************************************************************PREPRO
000830 PDIV                                                          . PREPRO
000831      PERFORM     RED    THRU   REDX                           . PREPRO
000832      IF          SEQ    EQUAL  99999                           PREPRO
000833      PERFORM     BCKC   THRU   BCKCD                           PREPRO
000834      MOVE        LAST1  TO     SQN                            PREPRO
000835      MOVE        QUINT  TO     PQTS   %2<                     PREPRO
000836      MOVE        PQOUT  TO     PR                             PREPRO
000837      PERFORM     WPR    THRU   WPRXT                          PREPRO
000838      GO TO       PDIV                                         . PREPRO
000839      MOVE        SEQ    TO     SQN                            . PREPRO
000840      MOVE        RTN01  TO     COED                           . PREPRO
000841      IF          FRST1  NUMERIC                               PREPRO
000842      MOVE        FRST1  TO     OP-1                           . PREPRO
000843      IF          SECD1  NUMERIC                               PREPRO
000844      MOVE        SECD1  TO     OP-2                           . PREPRO
000845      IF          LAST1  NUMERIC                               PREPRO
000846      MOVE        LAST1  TO     OP-3                           . PREPRO
000847      IF          RTN01  EQUAL  030                            PREPRO
000848      PERFORM     WPTR   THRU   WPTRX                          PREPRO
000849      GO TO       POUT                                         . PREPRO
000850      IF          RTN01  EQUAL  001                            PREPRO
```

```
000851      PERFORM     WPTR  THRU  WPTRX                                  PREPRO
000852      GO TO       POUT                                               .PREPRO
000853      IF          RTN01 EQUAL 030                                    PREPRO
000854      PERFORM     FDPTR THRU  FDPTX                                  PREPRO
000855      GO TO       POUT                                               .PREPRO
000856      IF          RTN01 EQUAL 110                                    PREPRO
000857      PERFORM     FDPTR THRU  FDPTX                                  PREPRO
000858      GO TO       POUT                                               .PREPRO
000859      IF          RTN01 EQUAL 111                                    PREPRO
000860      PERFORM     FDPTR THRU  FDPTX                                  PREPRO
000861      GO TO       POUT                                               .PREPRO
000862      IF          RTN01 EQUAL 112                                    PREPRO
000863      PERFORM     FDPTR THRU  FDPTX                                  PREPRO
000864      GO TO       POUT                                               .PREPRO
000865      IF          RTN01 EQUAL 130                                    PREPRO
000866      PERFORM     FDPTR THRU  FDPTX                                  PREPRO
000867      GO TO       POUT                                               .PREPRO
000868      IF          RTN01 EQUAL 070                                    PREPRO
000869      GO TO       POUT                                               .PREPRO
000870      IF          RTN01 EQUAL 190                                    PREPRO
000871      GO TO       POUT                                               .PREPRO
000872      IF          RTN01 EQUAL 180                                    PREPRO
000873      GO TO       POUT                                               .PREPRO
000874      IF          RTN01 EQUAL 150                                    PREPRO
000875      GO TO       POUT                                               .PREPRO
000876      IF          RTN01 EQUAL 170                                    PREPRO
000877      GO TO       POUT                                               .PREPRO
000878      IF          RTN01 NOT   EQUAL 250                              PREPRO
000879      PERFORM     D123  THRU  D01XT                                  PREPRO
000880      GO TO       POUT                                               .PREPRO
000881      PERFORM     D123  THRU  D01XT                                  .PREPRO
000882      MOVE        SECD1 TO    H                                      .PREPRO
000883      IF          H     NOT   EQUAL @-@                              PREPRO
000884      GO TO       POUT                                               .PREPRO
000885      IF          SECD1 EQUAL LAST1                                  PREPRO
000886      SUBTRACT    1     FROM  EDP                                    PREPRO
000887      SUBTRACT    1     FROM  OP-3                                   .PREPRO
000888 POUT                                                                .PREPRO
000889      MOVE        QUINT TO    PQTS  %SUBR<                           .PREPRO
000890      ADD         1     TO    SUBR                                   .PREPRO
000891      MOVE        ZERO  TO    QUINT                                  .PREPRO
000892      IF          SUBR  EQUAL 5                                      PREPRO
000893      MOVE        1     TO    SUBR                                   PREPRO
000894      MOVE        PQOUT TO    PR                                     PREPRO
000895      PERFORM     WPR   THRU  WPRXT                                  PREPRO
000896      MOVE        SPACE TO    PQOUT                                  .PREPRO
000897      GO TO       PDIV                                               .PREPRO
000898 PDBXT                                                               .PREPRO
000899      EXIT                                                           .PREPRO
000900*****************************************************************PREPRO
000901 D123                                                                .PREPRO
000902      IF          LAST1 NUMERIC                                      PREPRO
000903      GO TO       D012                                               .PREPRO
000904      IF          LAST1 EQUAL SPACE                                  PREPRO
000905      MOVE        ZERO  TO    OP-3                                   PREPRO
000906      GO TO       D012                                               .PREPRO
000907      MOVE        LAST1 TO    VALP                                   .PREPRO
000908      IF          HYP   EQUAL @-@                                    PREPRO
000909      MOVE        EDP   TO    OP-3                                   PREPRO
000910      ADD         1     TO    EDP                                    PREPRO
000911      GO TO       D012                                               .PREPRO
```

```
000912      MOVE        LAST1 TO    IDLBL                                  . PREPRO
000913      PERFORM     BS-1  THRU  BS-EX                                  . PREPRO
000914      MOVE        DNPTZ %SVLOC< TO    OP-3                           . PREPRO
            IF          SECD1 EQUAL LAST1
            MOVE        OP-3  TO    OP-2
            GO TO       DO1                                                .
000915 DO12                                                                . PREPRO
000916      IF          SECD1 NUMERIC                                       PREPRO
000917      GO TO       DO1                                                . PREPRO
000918      IF          SECD1 EQUAL SPACE                                   PREPRO
000919      MOVE        ZERO  TO    OP-2                                    PREPRO
000920      GO TO       DO1                                                . PREPRO
000921      MOVE        SECD1 TO    VALP                                   . PREPRO
000922      IF          HYP   EQUAL @-@                                     PREPRO
000923      MOVE        EDP   TO    OP-2                                    PREPRO
000924      ADD         1     TO    EDP                                     PREPRO
000925      GO TO       DO1                                                . PREPRO
000926      MOVE        SECD1 TO    IDLBL                                  . PREPRO
000927      PERFORM     BS-1  THRU  BS-EX                                  . PREPRO
000928      MOVE        DNPTZ %SVLOC< TO    OP-2                           . PREPRO
            IF          FRST1 EQUAL SECD1
            MOVE        OP-2  TO    OP-1
            GO TO       DO1XT                                              .
000929 DO1                                                                 . PREPRO
000930      IF          FRST1 NUMERIC                                       PREPRO
000931      GO TO       DO1XT                                              . PREPRO
000932      IF          FRST1 EQUAL SPACE                                   PREPRO
000933      MOVE        ZERO  TO    OP-1                                    PREPRO
000934      GO TO       DO1XT                                              . PREPRO
000935      MOVE        FRST1 TO    VALP                                   . PREPRO
000936      IF          HYP   EQUAL @-@                                     PREPRO
000937      MOVE        EDP   TO    OP-1                                    PREPRO
000938      ADD         1     TO    EDP                                     PREPRO
000939      GO TO       DO1XT                                              . PREPRO
000940      MOVE        FRST1 TO    IDLBL                                  . PREPRO
000941      PERFORM     BS-1  THRU  BS-EX                                  . PREPRO
000942      MOVE        DNPTZ %SVLOC< TO    OP-1                           . PREPRO
000943 DO1XT                                                               . PREPRO
000944      EXIT                                                           . PREPRO
000945*****************************************************************************PREPRO
000946 WPTR                                                                . PREPRO
000947      MOVE        0     TO    SUB                                    . PREPRO
000948 WPTR1                                                               . PREPRO
000949      ADD         1     TO    SUB                                    . PREPRO
000950      IF          FDDN  %SUB< NOT   EQUAL FRST1                       PREPRO
000951      GO TO       WPTR1                                              . PREPRO
000952      MOVE        FDPT  %SUB< TO    OP-1                             . PREPRO
000953 WPTRX                                                               . PREPRO
000954      EXIT                                                           . PREPRO
000955*****************************************************************************PREPRO
000956 FDPTR                                                               . PREPRO
000957      MOVE        0     TO    SUB                                    . PREPRO
000958 FDPT1                                                               . PREPRO
000959      ADD         1     TO    SUB                                    . PREPRO
000960      IF          FDN   %SUB< NOT   EQUAL FRST1                       PREPRO
000961      GO TO       FDPT1                                              . PREPRO
000962      MOVE        SUB   TO    OP-1                                   . PREPRO
000963 FDPTX                                                               . PREPRO
000964      EXIT                                                           . PREPRO
000965*****************************************************************************PREPRO
000966 WPR                                                                 . PREPRO
```

```
000967      WRITE       PR                                                    . PREPRO
000968      MOVE        SPACE TO    PR                                        . PREPRO
000969 WPRXT                                                                  . PREPRO
000970      EXIT                                                              . PREPRO
000971*****************************************************************************PREPRO
000972 WVAL                                                                   . PREPRO
000973      WRITE       VAL                                                   . PREPRO
000974      MOVE        SPACE TO    VAL                                       . PREPRO
000975      MOVE        ZERO  TO    VLP                                       . PREPRO
000976 WVALX                                                                  . PREPRO
000977      EXIT                                                              . PREPRO
000978*****************************************************************************PREPRO
000979 BCKC                                                                   . PREPRO
000980      IF          PQOUT NOT   EQUAL SPACE                               PREPRO
000981      MOVE        PQOUT TO    PR                                        PREPRO
000982      PERFORM     WPR   THRU  WPRXT                                     . PREPRO
000983      MOVE        SPACE TO    PQOUT                                     . PREPRO
000984 BCKCD                                                                  . PREPRO
000985      EXIT                                                              . PREPRO
000986*****************************************************************************PREPRO
000987 SRT-1                                                                  . PREPRO
000988      PERFORM     SRT-2 THRU  SRT2X                                     . PREPRO
000989      ADD         1     TO    OP-1                                      . PREPRO
000990      IF          OP-1  NOT   EQUAL EDP                                 PREPRO
000991      GO TO       SRT-1                                                 . PREPRO
000992 SRT-X                                                                  . PREPRO
000993      EXIT                                                              . PREPRO
000994 SRT-2                                                                  . PREPRO
000995      MOVE        DNNS  %OP-1<      TO    WKTAB                       . PREPRO
000996      ADD         1     OP-1  GIVING      OP-2                        . PREPRO
000997      MOVE        OP-1  TO    OP-3                                      . PREPRO
000998 SRT21                                                                  . PREPRO
000999      PERFORM     SRT-3 THRU  SRT3X                                     . PREPRO
001000      ADD         1     TO    OP-2                                      . PREPRO
001001      IF          OP-2  NOT   GREATER     EDP                           PREPRO
001002      GO TO       SRT21                                                 . PREPRO
001003      MOVE        DNNS  %OP-1<      TO    DNNS  %OP-3<                . PREPRO
001004      MOVE        WKTAB TO    DNNS  %OP-1<                            . PREPRO
001005 SRT2X                                                                  . PREPRO
001006      EXIT                                                              . PREPRO
001007 SRT-3                                                                  . PREPRO
001008      IF          DNN   %OP-2<      LESS  THAN  WAREA                 PREPRO
001009      MOVE        DNNS  %OP-2<      TO    WKTAB                       PREPRO
001010      MOVE        OP-2  TO    OP-3                                      . PREPRO
001011 SRT3X                                                                  . PREPRO
001012      EXIT                                                              . PREPRO
001013*****************************************************************************PREPRO
001014 BS-1                                                                   . PREPRO
001015      ADD         SAVLG 1     GIVING      HDNP                          . PREPRO
001016      MOVE        ZERO  TO    VLP                                       . PREPRO
001017 BS-2                                                                   . PREPRO
001018      ADD         VLP   HDNP  GIVING      SVLOC                         . PREPRO
001019      DIVIDE      2     INTO  SVLOC GIVING      SVLOC                 . PREPRO
001020      IF          IDLBL EQUAL DNN   %SVLOC<                           PREPRO
001021      GO TO       BS-EX                                                 . PREPRO
001022      IF          IDLBL LESS  THAN  DNN   %SVLOC<                     PREPRO
001023      MOVE        SVLOC TO    HDNP                                      . PREPRO
001024      IF          IDLBL GREATER     THAN  DNN   %SVLOC<               PREPRO
001025      MOVE        SVLOC TO    VLP                                       . PREPRO
001026      ADD         1     VLP   GIVING      VCNT                          . PREPRO
001027      IF          VCNT  NOT   EQUAL HDNP                                PREPRO
```

```
001028      GO TO       BS-2                                                 . PREPRO
001029      DISPLAY     IDLBL                                                . PREPRO
001030 BS-EX                                                                 . PREPRO
001031      EXIT                                                             . PREPRO
001032*******************************************************************************  PREPRO
001033*******************************************************************************PREPRO
```

```
        .TITL   MULTIPLY
        .ENT    MULT
        .EXTN   HERE,SHIFT
        .NREL
MULT:   ISZ     RTST,2  ;GET NEXT STORE POSITION
         STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,PACK1,2       ;GET BEGINNING ADDRESS OF
        STA     1,PACT1,2       ;PACK1,PACK2,PACK3, AND
        LDA     1,PACK2,2       ;INITILIZE TEMPORARY ADDRESS
        STA     1,PACT2,2       ;AREAS
        LDA     1,PACK3,2
        STA     1,PACT3,2
        LDA     1,TEN,2 ;GET AN EIGHT FOR EIGHT WORD MULTIPLY
        STA     1,N,2   ;STORE IN N
        LDA     1,FOUR,2        ;GET A FOUR
        STA     1,X4T,2         ;INITILIZE A COUNTER
        LDA     1,N,2           ;GET NUMBER OF WORDS LONG
        MOVZL   1,1             ;MULTIPLY THE WORD
        DSZ     X4T,2           ;BY 16 TO GET PROPER
        JMP     .-2             ;VALUE FOR MULTIPLICATION FACTOR
        STA     1,M,2           ;STORE FINAL VALUE IN @M@
        LDA     1,PACK3,2       ;GET RECEIVING FIELD FOR
        STA     1,PACZ,2        ;ZEROING OUT ROUTINE
        JSR     @ZERA,2           ;GO TO ZERO OUT ROUTINE
        JSR     @RTER,2
START:  LDA     1,PACK2,2       ;GET SECOND ARGUMENT
        STA     1,PAC1,2        ;INITILIZE ADDRESS FOR SHIFTRIGHT
        STA     1,PAC2,2        ;ROUTINE
        JSR     @SHIFR          ;GO TO SHIFTRIGHT ROUTINE
        JSR     @RTER,2
        MOV     1,1,SNC         ;CHECK TO SEE IF CARRY BIT IS SET
        JMP     .+2             ;NO
        JMP     AA              ;YES
AD:     LDA     1,PACK3,2       ;INITILIZE ADDRESS FOR
        STA     1,PAC1,2        ;SHIFTRIGHT ROUTINE
        STA     1,PAC2,2
        JSR     @SHIFR          ;GO TO SHIFTRIGHT ROUTINE
        JSR     @RTER,2
        DSZ     M,2             ;DECREMENT @M"
        JMP     START           ;NO ZERO DO AGAIN
        JMP     AB              ;ZERO
AA:     LDA     1,PACK1,2       ;INITILIZE ADDRESS FOR
        STA     1,PAC1,2        ;ADDPACK ROUTINE
        LDA     1,PACK3,2
        STA     1,PAC2,2
        LDA     1,PACT,2
        STA     1,PAC3,2
        JSR     @ADDT,2           ;GO TO ADDPACK ROUTINE
        JSR     @RTER,2
        JSR     XX              ;GO TO INITILIZE PACK3
        JSR     @RTER,2
        JMP     AD
AB:     LDA     1,PACK2,2         ;GET SECOND ARGUMENT
        STA     1,PAC1,2          ;INITILIZE ADDRESS FOR SHIFTRIGHT
        STA     1,PAC2,2          ;ROUTINE
        JSR     @SHIFR  ;GO TO SHIFTRIGHT ROUTINE
        JSR     @RTER,2
       LDA     1,FOUR,2       ;GET A FOUR
        STA     1,X4T,2         ;STORE IT
        LDA     0,PACT3,2       ;GET RECEIVING FIELD
        ADD     1,0             ;MODIFY ADDRESS
```

```
        STA     0,PACT3,2       ;STORE NEW ADDRESS
        DSZ     PACT3,2
AG:     LDA     1,@PACT3,2      ;GET CONTENTS OF RECEIVING FIELD
        MOV#    1,1,SNR         ;CHECK TO SEE IF EQUAL TO ZERO
        JMP     .+2             ;EQUAL TO ZERO
        JMP     AX              ;NOT EQUAL
        DSZ     X4T,2           ;DECREMENT COUNTER
        JMP     .+2             ;NOT ZERO
        JMP     AX+2            ;ZERO
        DSZ     PACT3,2         ;DECREMENT ADDRESS
        JMP     AG              ;DO AGAIN
AX:     LDA   1,ONE,2             ;GENERATE A ONE
        STA     1,SFLAG,2       ;SET SIZE ERROR FLAG
        LDA     1,N,2           ;GET NUMBER OF WORDS TO BE MOVED
        MOVZR    1,1,SZR              ;DIVIDE BY TWO
        JMP     .+2
        INC     1,1       ;IF AC1 IS ZERO INITILIZE IT TO ONE
        STA     1,NT,2    ;STORE IN NT
        LDA     0,PACK3,2       ;GET RECEIVING FIELD
        STA     0,PAC2,2        ;STORE IN SECOND ARGUMENT
        ADD     1,0             ;FOR MATHMOVE ROUTINE
        STA     0,PAC1,2        ;STORE MODIFIED ADDRESS FOR MATHMOVE
        JSR     @HERE1          ;GO TO MOVE ROUTINE
        JSR     @RTER,2
        LDA     1,N,2           ;GET NUMBER OF WORDS TO BE MOVED
        MOVZR    1,1,SZR              ;DIVIDE BY TWO
        JMP     .+2
        INC     1,1       ;IF AC1 IS ZERO INITILIZE IF ONE
        STA     1,NT,2    ;STORE IN NT
        LDA     0,PACT2,2       ;GET SECOND ARGUMENT
        STA     0,PAC1,2        ;SET ADDRESS FOR MATHMOVE ROUTINE
        LDA     0,PACK3,2       ;MODIFY ADDRESS
        ADD     1,0             ;FOR PROPER ALIGNMENT
        STA     0,PAC2,2        ;SECOND ARGUMENT FOR MATHMOVE ROUTINE
        JSR     @HERE1          ;GO TO MATHMOVE ROUTINE
        JSR     @RTER,2
        LDA     1,N,2           ;GET NUMBER OF WORDS
        MOVR    1,1             ;DIVIDE BY TWO
        INC     1,1             ;INCREMENT BY ONE
        LDA     0,PACT2,2       ;GET SECOND ARGUMENT
        ADD     1,0             ;GET NEW ADDRESS
        STA     0,PAC2,2        ;STORE IT
        LDA     1,@PAC2,2       ;GET CONTENTS OF THAT POSITION
        MOVL#   1,1,SNC         ;CHECK TO SEE IF BIT ZERO IS ON
        JMP     @RTN,2        ;NO , RETURN
        LDA     1,PACK2,2       ;GET BEGINNING ADDRESS
        STA     1,PAC2,2        ;OF PACK2 FOR ZERO OUT ROUTINE
        JSR     @ZERA,2           ;GO TO ZERO-OUT ROUTINE
        JSR     @RTER,2
        LDA   1,ONE,2             ;GENERATE A ONE
        STA     1,@PAC2,2       ;STORE IT
        LDA     1,PACK2,2       ;GET ADDRESS@S FOR
        STA     1,PAC1,2        ;ADD PACK ROUTINE
        LDA     1,PACK3,2
        STA     1,PAC2,2
        LDA     1,PACT,2
        STA     1,PAC3,2
        JSR     @ADDT,2           ;GO TO ADDPACK ROUTINE
        JMP     .+1
        JSR     XX              ;INITILIZE PACK3
        JSR     @RTER,2
```

```
        JMP     @RTN,2          ;RETURN
XX:     ISZ     RTST,2
        STA     3,@RTST,2         ;SAVE RETURN ADDRESS
        LDA     1,PAC1,2        ;INITILIZE ADDRESS@S
        STA     1,PAC1,2        ;FOR MATHMOVE
        LDA     1,PAC3,2        ;ROUTINE
        STA     1,PAC2,2
        JSR     @MMOV,2          ;GO TO MATHMOVE ROUTINE
        JSR     @RTER,2
        JSR     @RTN,2          ;RETURN
HERE1:  HERE
SHIFR:  SHIFT
```

```
        .TITLE  MULTIPLY10
        .ENT    MUL10
        .NREL
;THIS ROUTINE MULTIPLIES ONE PACK BY 10 AND STORES BACK IN SAME PACK
;THE PACK FIELD TO BE USED IS PACTL
MUL10:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     1,PACTL,2
        LDA     0,N,2
        NEG     0,3
        COM     3,3
        ADD     3,1
        STA     1,X4T,2         ;NTH WORD OF PAC
        STA     0,M,2           ;N
        SUB     0,0
        STA     0,X17T,2        ;SET PREV HIGH ORDER VALUE TO ZERO
AA:     SUB     0,0
        LDA     3,@X4T,2        ;CURRENT PACK WORD CONTENTS
        MOVZL   3,1
        MOVL    0,0
        MOVL    1,1             ;LOW ORDER * 4
        MOVL    0,0             ;HIGH ORDER * 4
        ADDZ    3,1,SZC         ;LOW & HIGH ORDER * 5
        INC     0,0             ;ADD 1 TO HIGH ORDER IF OVERFLOW
        MOVZL   1,1             ;LOW ORDER * 10
        MOVL    0,0             ;HIGH ORDER * 10
        LDA     3,X17T,2        ;PREV HIGH ORDER * 10
        ADDZ    3,1,SZC         ;COMBINE WITH LOW ORDER
        INC     0,0             ;INC HI ORDER IF OVERFLOW
        STA     0,X17T,2
        STA     1,@X4T,2        ; * 10]
        DSZ     X4T,2
        DSZ     M,2
        JMP     AA              ;NEXT WORD
        MOVZ    0,0,SZR         ;ZERO CARRY, CHECK IF OVERFLOW
        SUB     1,1             ;YES, SET CARRY
        JSR     @RTN,2
```

```
        .TITL   NEGATIVENUMBER
        .ENT    SNG
        .EXTD   GET
        .NREL
SNG:    ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        SUB     0,0     ;CLEAR AC0
        STA     0,NEGNO,2       ;CLEAR NEGATIVE NUMBER FLAG
        LDA     3,PNT1,2        ;GET BYTE POINTER TO DATA
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET    ;GET A BYTE
        JMP     @RTER,2
        MOV     1,3     ;SAVE AC1
        LDA     0,C60,2 ;GET A MASK OF 60
        AND     0,3     ;TEST BYTES TO SEE IF ASCII NUMBER
        SUB#    0,3,SZR ;TEST
        JMP     .+11    ;NO NOT A NUMBER
        MOV     1,3     ;SAVE AC1
        LDA     0,EMSK5,2       ;GET MASK OF 200
        AND     0,3     ;TEST TO SEE IF BIT EIGHT IS SET
        SUB     0,3,SNR ;TEST
        JMP     @RTN,2  ;NOT SET NUMBER IS POSITIVE
        SUBZL   0,0     ;NUMBER IS NEGATIVE GENERATE A ONE
        STA     0,NEGNO,2       ;STORE IN NEGATIVE NUMBER FLAG
        JMP     @RTN,2  ;RETURN
        MOV     1,3     ;SAVE AC1
        LDA     0,C100,2        ;BRING IN MASK OF 100
        AND     0,3     ;TEST TO SEE IF BIT SEVEN IS SET
        SUB#    0,3,SNR ;TEST
        JMP     LETT    ;YES IT IS A LETTER
        LDA     1,C60,2 ;NO IT IS A MINUS SIGN
        SUBZL   0,0     ;GENERATE A ONE
        STA     0,NEGNO,2       ;STORE IN NEGATIVE NUMBER FLAG
        JMP     @RTN,2  ;RETURN
LETT:   LDA     0,C147,2        ;CHANGE LETTER TO NUMBER
        ADD     0,1     ;DO THE CHANGE
        JMP     .-5
```

```
;
;       NEW COMMAND
;
        .TITLE  NEW
        .NREL
        .TXTM   1
        .ENT    NEW
NEW:    ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GCHAP,2
        JSR     @RTER,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,2
        LDA     0,LF,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,2
        ISZ     NEWF,2
        JSR     @ALCRP,2
        JMP     .+4
        DSZ     NEWF,2
        JSR     @RTER,2
        JSR     @RTN,2
        DSZ     NEWF,2
        JSR     @RTER,2
        JSR     @RTER,2
```

```
        .TITL   PERFORM
        .ENT    PFRM1
        .NREL
PFRM1:  ISZ     RTST,2  ;GET NEXT       STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,QP,2          ;GET QUINT POINTER
        LDA     0,2,3           ;STORE OPERAND 1 IN SAVE.
        ISZ     SAVE,2
        STA     0,@SAVE,2       ;THIS IS THE FIRST STATEMENT
                                ;TO EXECUTE FROM THE PERFORM
        STA     0,TQC,2 ;STORE IN TEMPORARY QUINT COUNTER
        LDA     0,3,3           ;STORE OPERAND 2 IN @NDPT.
        ISZ     NDPT,2
        STA     0,@NDPT,2       ;THIS IS THE LAST STATEMENT
                                ;TO EXECUTE FROM THE PERFORM
        LDA     0,4,3   ;STORE OPERAND 3 IN CNT.
        ISZ     CNT,2
        STA     0,@CNT,2        ;THIS IS THE NUMBER OF TIMES
                                ;TO PERFORM THIS SERIES OF QUINTS
        ISZ     RTPT,2
        LDA     3,QC,2
        INC     3,3
        STA     3,@RTPT,2
        JMP     @RTN,2          ;PROCESSING COMPLETE.  RETURN
```

```
        .TITL   OPEN
        .ENT    OPEN
        .NREL
OPEN:   ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADRESS
        LDA     1,OPFLG,2       ;LOAD OPCODE FLAG
        MOVZR#  1,1,SEZ         ;FIND OUT CONTENTS (0,1,2, OR 3)
        JMP     RTER,2          ;3
        MOV     1,1,SNR
        JMP     RPEN            ;0, OPEN FOR READ ONLY
        MOVZR#  1,1,SZR
        JMP     EPEN            ;2, OPEN FOR WRITE ONLY
        LDA     0,ONP           ;1, OPEN FOR INPUT AND OUTPUT
        JMP     STAR            ;DO THE OPEN
RPEN:   LDA     0,ROPN
        JMP     STAR            ;DO THE OPEN
EPEN:   LDA     0,EOPN
STAR:   STA     0,BLNK          ;STORE THE TYPE OF OPEN
        LDA     0,OP1DP,2       ;LOAD THE BYTE POINTER
        .SYSTEM
        .GCHN                   ;GET A FREE CHANNEL
        JMP     .-2             ;NO FREE CHANNEL AVAILABLE
        .SYSTEM
BLNK:   0
        JMP     ERR             ;BAD OPEN
        STA     2,CHANL,3       ;STORE AC1 IN DATA BASE
        MOV     3,2     ;RESTORE AC2
        JSR     @RTN,2          ;PROCESSING COMPLETE, RETURN
ERR:    STA     2,ERCDE,3           ;STORE ERROR CODE
        JSR     @RTER,2             ;ERROR CONDITION, RETURN
ROPN:   .ROPEN  77
EOPN:   .EOPEN  77
ONP:    .OPEN   77
```

```
;
;
;       SUBROUTINE OPENS AN AVAILABLE CHANNEL
;
;       INPUT:  AC0             BYTE PTR TO NAME OF FILE
;               AC2             DATA PTR
;
;       OUTPUT: AC1; CHNUM, 2   CHANNEL #
;
;
        .TITLE  OPAVC
        .NREL
        .TXTM   1
        .ENT    OPAVC
OPAVC:  ISZ     RTST,2
        STA     3,@RTST,2
OP1:    .SYSTM
        .GCHN                   ;GET FREE CHANNEL
        JSR     @RTER,3         ;NONE
        STA     2,CHNUM,3
        SUB     1,1
        .SYSTM
        .OPEN   77              ;OPEN IT
        JMP     .+2
        JMP     OP2
        LDA     1,TWNY1
        SUB     2,1,SNR
        JMP     OP1             ;CANT, STOLEN
        JSR     @RTER,3
OP2:    MOV     2,1
        MOV     3,2
        JSR     @RTN,2
TWNY1:  21
```

```
;
;       THIS ROUTINE PRINTS SIX OCTAL DIGITS FOR THE CONTENTS OF AC0
;       ON THE MASTER CONSOLE, PRECEDED BY ONE BLANK
;
        .TITLE  PROCT
        .NREL
        .TXTM   1
        .ENT    PROCT,PRCRL,RDAEC
PROCT:  ISZ     RTST,2
        STA     3,@RTST,2
        STA     0,TEMP,2
        LDA     0,FIVE,2
        STA     0,TEMP2,2
        LDA     0,BLK,2         ;BLANK
        .SYSTM
        .PCHAR                  ;PRINT IT
        JSR     @RTER,3
        SUB     0,0
        LDA     1,TEMP,2
        MOVZL   1,1             ; * 2
        MOVL    0,0             ;1ST CHAR
        STA     1,TEMP,2
        LDA     1,C60,2
        ADD     1,0             ;CONVERT TO ASCII
        .SYSTM
        .PCHAR                  ;PRINT IT
        JSR     @RTER,2
PRO1:   LDA     1,TEMP,2
        SUB     0,0
        MOVZL   1,1
        MOVL    0,0
        MOVL    1,1
        MOVL    0,0
        MOVL    1,1             ; * 8 AND MOVE
        MOVL    0,0             ; CHAR INTO AC0
        STA     1,TEMP,2
        LDA     1,C60,2
        ADD     1,0             ;CONVERT TO ASCII
        .SYSTM
        .PCHAR                  ;PRINT IT
        JSR     @RTER,3
        DSZ     TEMP2,2         ; ALL SEX PRINTED?
        JMP     PRO1            ;NO
        JSR     @RTN,2
;
;       PRINT CARRIAGE RETURN LINE FEED
;
PRCRL:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,C1,2
        .SYSTM
        .PCHAR                  ;PRINT CARRIAGE RETURN
        JSR     @RTER,3
        LDA     0,LF,2
        .SYSTM
        .PCHAR                  ;&RINT LINE FEED
        JSR     @RTER,3
        JSR     @RTN,2
;
;       READ CHAR AND ECHO, IF CARRIAGE RETURN PRINT LF AND RETURN +3
;
```

```
RDAEC:  ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GCHAP,2        ;READ
        JSR     @RTER,2
        .SYSTM
        .PCHAR                  ;ECHO
        JSR     @RTER,2
        LDA     1,C1,2
        SUB     0,1,SZR         ;CARRIAGE RETURN?
        JSR     @RTN,2          ;NO
        ISZ     @RTST,2         ;RETURN +3
        LDA     0,LF,2
        .SYSTM
        .PCHAR
        JSR     @RTER,2
        JSR     @RTN,2
```

```
        .TITL   PUTEDIT
        .ENT    EDITT
        .EXTN   B1,PUTTT,PTT,DOLAR,ASTK,ZEE,COMMA,MYNUS,PLUSS,PUTCH
        .EXTD   PUT,GET
        .NREL
EDITT:  LDA     3,ECHAR,2       ;GET EDIT TABLE DISPLACEMENT
        LDA     0,TABLE ;GET BEGINNING OF TABLE
        ADD     0,3     ;GET EDIT TABLE DISPLACEMENT
        LDA     3,0,3   ;GET  THE ROUTINE TO GO TO
        JMP     0,3     ;GO T ROUTINE
TABLE:  TABL
TABL:           NULL    ;NULL CHARACTER FILL ROUTINE
                BLANK   ;BLANK-FILL CHARACTER ROUTINE
                SLASH   ;SLASH FILL ROUTINE
                ZEROT   ;ZERO FILL CHARACTER ROUTINE
                COMMA   ;COMMA FILL ROUTINE
                DEBIT   ;DEBIT SIGN FILL ROUTINE
                CREDT   ;CREDIT SIGN ROUTINE
                PEROD   ;PERIOD FILL CHARACTER ROUTINE
                DOLAR   ;DOLLAR SIGN FILL CHARACTER ROUTINE
                MYNUS   ;MINUS SIGN FILL CHARACTER ROUTINE
                PLUSS   ;PLUS SIGN FILL CHARACTER ROUTINE
                ASTK    ;ASTERISK FILL ROUTINE
                ZEE     ;ZERO FILL ZERO SUPPRESS ROUTINE
                TEST1   ;CHARACTER FROM DATA STRING FILL ROUTINE
                PUTCH   ;"V" FILL CHARACTER ROUTINE
                RTRN    ;RETURN SHOULD NOT HAVE COME INTO THIS ROUTINE
NULL:   LDA     1,NULT,2        ;GET A NULL CHARACTER
        STA     1,ECHAR,2       ;STORE EDIT CHARACTER
        JMP     PUTIT   ;GO TO STORE ROUTINE
SLASH:  LDA     1,SLSH,2        ;GET A SLASH
        STA     1,ECHAR,2       ;STORE EDIT CHARACTER
        JMP     PUTIT   ;GO TO STORE ROUTINE
BLANK:  LDA     1,BLK,2 ;GET A BLANK
        STA     1,ECHAR,2       ;STORE EDIT CHARACTER
        JMP     PUTIT   ;STORE ROUTINE
ZEROT:  LDA     1,C60,2 ;GET AN ASCII ZERO
        STA     1,ECHAR,2       ;STORE EDIT CHARACTER
PUTIT:  DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     .+2
        JSR     @RTN,2  ;WENT TO ZERO SOMETHING WRONG
        LDA     1,ECHAR,2       ;LOAD EDIT CHARACTER
        LDA     3,PNT2,2        ;GET BYTE POINTER TO RECEIVING FIELD
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;GO TO PUT BYTE ROUTINE
        JSR     @RTER,2
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        DSZ     RCNT,2  ;DECREMENT REPITION COUNT
        JMP     .+2     ;NOT ZERO
        JMP     @.B1    ;ZERO GET ANOTHER EDIT CHARACTER
        DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     PUTIT+1 ;DO AGAIN
        JMP     @.B1    ;BETTER BE END OF EDIT FORMAT
PEROD:  DSZ     CNT2,2
        JMP     .+2
        JSR     @RTN,2  ;ERROR RETURN
        LDA     1,PERD,2        ;GET A PERIOD
        LDA     3,PNT2,2        ;GET BYTE POINTER TO RECEIVING FIELD
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;GO TO PUT BYTE ROUTINE
        JSR     @RTER,2
```

```
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        LDA     1,CNT1,2        ;GET COUNT ONE
        MOV#    1,1,SZR ;TEST TO SEE IF ZERO
        JMP     @.B1    ;GET ANOTHER EDIT STRING
        LDA     1,CNT2,2
        STA     1,RCNT,2
        JMP     ZEROT
DEBIT:  LDA     0,NEGNO,2       ;GET NEGATIVE NUMBER FLAG
        MOVR    0,0,SNC ;CHECK TO SEE IF SET
        JMP     BLANK   ;FIELD IS POSITIVE
        LDA     1,DB,2  ;GET DEBIT SIGN
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        LDA     3,PNT2,2        ;GET BYTE POINTER TO STORE POSITION
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE IT
        JSR     @RTER,2
        DSZ     RCNT,2
        JMP     .+2     ;NOT ZERO
        JMP     @.B1    ;GET ANOTHER EDIT CHARACTER
        ISZ     ECHAR,2 ;GET LAST HALF OF WORK
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        JMP     .-10    ;DO AGAIN
CREDT:  LDA     0,NEGNO,2       ;GET NEGATIVE NUMBER FLAG
        MOVR    0,0,SNC ;CHECK TO SEE IF SET
        JMP     BLANK   ;FIELD IS POSTIVE
        LDA     1,CR,2  ;GET CREDIT SIGN
        JMP     DEBIT+4
TEST1:  LDA     0,OP2DT,2       ;GET TYPE OF MOVE
        LDA     1,FOUR,2        ;NUMERIC TYPE MOVE
        SUB     1,0,SZR ;TEST TO SEE IF EQUAL
        JMP     .+11    ;NO ALPHNUMERIC EDIT MOVE
        LDA     1,PNT0,2        ;YES, SEE IF BYTE POINTER IS SET
        MOV     1,1,SNR
        JMP     @PUCH   ;YES CHARACTER FILL
        LDA     1,C60,2 ;NO GET A ZERO
        STA     1,ECHAR,2       ;STORE IN ECHAR
        LDA     1,C1,2
        STA     1,ECHT,2
        JMP     @PUTT1
        LDA     1,PNT0,2        ;TEST TO SEE IF SET
        MOV     1,1,SNR
        JMP     @PUCH   ;YES PUTCHARACTER OUT
        LDA     1,BLK,2 ;GET A BLANK
        STA     1,ECHAR,2       ;STORE IN EDITCHARACTER
        STA     1,ECHT,2        ;STORE IN TEMPORARY EDIT CHARACTER
        JMP     @PUTT1
.B1:    B1
PUTT1:  PUTTT
PUCH:   PTT
DOLR:   DOLAR
ASKT:   ASTK
ZER:    ZEE
COM1:   COMMA
POST:   PLUSS
NEGI:   MYNUS
```

```
;
;       READ SAVE FILE TO BE EXECUTED
;
        .TITLE  RDSVF
        .NREL
        .TXTN   1
        .ENT    RDSVF
;
RDSVF:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,TTOP,2
        JSR     @OPAVP,2        ;OPEN TTO, ALSO SETS DATAP IN USP FOR .SYSTM
        JSR     @RTEP,2
        MOV     1,2
        LDA     0,MSG
        .SYSTM
        .WRL    77
        JSR     @RTER,2           ;ERROR, DATA PTR IS IN AC3 AFTER .SYSTM
        LDA     2,CHNUM,3
        .SYSTM
        .CLOSE  77
        JSR     @RTER,2
        MOV     3,2
        LDA     0,TTIN
        JSR     @OPAVP,2        ;OPEN TTI
        JSR     @RTER,2
        LDA     0,SVFLN,2
        MOV     1,2
        .SYSTM
        .RDL    77              ;READ SAVE FILE NAME
        JSR     @RTER,2
        LDA     2,CHNUM,3
        .SYSTM
        .CLOSE  77
        JSR     @RTER,2
        MOV     3,2             ;RESET DATAP IN AC2
        JSR     @RTN,2
MSG:    .+1*2
        .TXT    /ENTER SAVE FILE NAME<15> /
TTIN:   .+1*2
        .TXT    /$TTI/
```

```
        .TITL   READ
        .ENT    READ
        .EXTD   GET.PUT
        .NREL
READ:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     0,OP1DP,2       ;GET BYTE POINTER
        LDA     1,OP1DL,2          ;GET LENGTH OF INPUT
        LDA     2,CHANL,2       ;GET THE CHANNEL NUMBER
        .SYSTEM
        .RDL    77
        JMP     EROUT           ;BAD READ?
        JMP     FNISH
EROUT:  LDA     0,EOFM,3           ;LOAD END OF FILE CODE
        SUB#    2,0,SNR            ;COMPARE
        JMP     .+3       ;END OF FILE
        STA     2,ERCDE,3
        JSR     @RTER,3
        MOV     3,2
        JSR     @RTN,2
FNISH:  MOV     3,2
        LDA     0,OP1DL,2
        STA     0,CNT0,2
        LDA     1,OP1DP,2          ;GET BYTE POINTER
        STA     1,OPDP,2           ;STORE IN TEMP
AA:     LDA     1,OPDP,2           ;GET BYTE POINTER
        STA     1,BYPTR,2          ;STORE IN BYTE POINTER
        JSR     @GET     ;GET A BYTE
        JSR     @RTER,2 ;ERROR
        LDA     0,C1,2  ;GET A CARRAGE RETURN
        SUB     0,1,SZR ;TEST TO SEE IF EQUAL
        JMP     .+2       ;NO NOT A CARRAGE RETURN
        JMP     .+5       ;CARRAGE RETURN
        ISZ     OPDP,2  ;GET NEXT POSITION
        DSZ     CNT0,2  ;DECREMENT LENGTH
        JMP     AA
        JMP     .+11
        LDA     1,BLK,2 ;STORE OUT A BLANK
        LDA     3,OPDP,2           ;GET BYTEPOINTER
        STA     3,BYPTR,2          ;STORE IN BYTE POINTER
        JSR     @PUT     ;STORE OUT BLANK
        JSR     @RTER,2
        ISZ     OPDP,2
        DSZ     CNT0,2  ;DECREMENT COUNTER
        JMP     .-7
        SUB     1,1                ;CLEAR AC1
        STA     1,TOC,2            ;STORE IN TEMP.  QUINT CTR
        JSR     @RTN,2             ;PROCESSING COMPLETE, RETURN
```

```
        .TITL   RELATIONAL
        .ENT    RLAT1
        .EXTN   MATHP
        .EXTD   GET
        .NREL
MPACK:  MATHP
RLAT1:  LDA     1,OP1DT,2       ;LOAD OPER1 TYPE
        LDA     0,OP2DT,2       ;LOAD OPER2 TYPE
        ADD     0,1             ;ADD THE TWO TOGETHER
        MOVZR#  1,1,SNR         ;CHECK RESULT FOR 2 OR GREATER
        JMP     @MPACK          ;< 2, GO TO THE MATH-PACKAGE
        LDA     0,TWO,2         ;LOAD A 2
        SUB#    1,0,SNR         ;CHECK IF 2 OR > 2
        JMP     @MPACK          ;EQUAL TO 2, GO TO THE MATH-PACKAGE
        LDA     1,OP1DT,2       ;LOAD ARG1 TYPE
        LDA     0,ONE,2 ;LOAD A ONE
        SUB     1,0,SZR         ;CHECK IF TYPV = 1
        JMP     CARG2           ;NOT 1, CHECK ARG2
        LDA     1,DEC1,2        ;LOAD AC1 WITH ARG DECIMAL PTR.
        JMP     CNTGR           ;JUMP TO CHECK IF INTEGER
CARG2:  LDA     1,OP2DT,2       ;LOAD ARG2 TYPE
        LDA     0,ONE,2         ;LOAD A 0
        SUB     1,0,SZR         ;CHECK IF TYPE EQUAL 1
        JMP     HSKPG           ;JUMP TO RELATIONAL TEST
        LDA     1,DEC2,2        ;LOAD AC1 WITH DECIMAL PTR.
CNTGR:  MOV     1,1,SZR         ;CHECK TO SEE IF = 0
        JMP     @RTER,2 ;NOT ZERO, RUNTIME ERROR
HSKPG:  LDA     1,OP1DL,2       ;LOAD OPER1 LENGTH
        STA     1,CNT1,2        ;STORE IN CNT1
        LDA     1,OP2DL,2       ;LOAD OPER2 LENGTH
        STA     1,CNT2,2        ;STORE IN CNT2
        LDA     1,OP1DP,2       ;LOAD OPER1 BYTE PTR.
        STA     1,PNT1,2        ;STORE IN PNT1
        LDA     1,OP2DP,2       ;LOAD OPER2 BYTE PTR.
        STA     1,PNT2,2        ;STORE IN PNT2
A:      SUB     0,0             ;CLEAR AC0
        STA     0,CHKWD,2       ;STORE IN CHKWD
        LDA     1,CNT2,2        ;LOAD CNT2
        MOV     1,1,SNR         ;CHECK TO SEE IF = 0
        JMP     .+3             ;NOT ZERO
        MOVL# 1,1,SNC   ;NEG ?
        JMP     A1              ;EQUAL TO ZERO
        LDA     1,CNT1,2
        MOV     1,1,SNR ;EQUAL TO ZERO
        JMP     NTEST           ;GO TO NEGATIVE TEST
        MOVL#   1,1,SZC         ;NEGATIVE?
        JMP     NTEST
        LDA     1,OP2DT,2
        LDA     0,SIX,2
        SUB     1,0,SNR         ;CHECK TO SEE IF ZERO FILL
        JMP     .+4
        LDA     1,BLK,2
        STA     1,CHAR2,2
        JMP     .+3
        LDA     1,C60,2
        STA     1,CHAR2,2
        JSR     ARG1
        JSR     @RTER,2
        JMP     CHECK
NTEST:  LDA     1,NFLAG,2
        MOV     1,1,SNR         ;TEST T SEE IF SET
```

```
        .TITL   RETURNPOINT
        .ENT    ERTRN,RETRN,GCHAR
        .NREL
ERTRN:  LDA     2,USP    ;RESTORE AC2
        STA     0,X17T,2
        ISZ     RTST,2  ;GET NEXT STORE POSITION
        LDA     0,@RTST,2.       ;BRING INTO AC0
        MOVL    0,0,SZC ;CHECK TO SEE IF BIT ZERO IS SET
        JMP     ERTRI
        MOVL    3,3      ;SHIFTLEFT RETURN POINT
        MOVOR   3,3      ;SET CARRY PUT INTO BIT ZERO
        STA     3,@RTST,2
ERTRI:  DSZ     RTST,2
        LDA     3,@RTST,2
        MOVL    3,0      ;DECREMENT RETURN POINTER. MOVE IT INTO AC3 AND SHIFTLEF
        MOVOR   0,0      ;BRING IN CARRY
        STA     0,@RTST,2        ;STORE IN RTST POINTER
        DSZ     RTST,2
        LDA     0,X17T,2
        JMP     0,3
RETRN:  LDA     3,@RTST,2        ;LOAD IT INTO AC2
        STA     3,X17T,2
        SUBC    3,3
        STA     3,@RTST,2
        LDA     3,X17T,2
        DSZ     RTST,2  ;GET THE RETURRN POINT
        INC     3,3      ;INCREMENT AC3
        JMP     0,3      ;RETURN
GCHAR:  ISZ     RTST,2           ;GET NEXT STORE POSITION
        STA     3,@RTST,2        ;SAVE RETURN ADDRESS
        .SYSTM
        .GCHAR
        JSR     @RTER,2
        LDA     3,DEL,2
        SUB#    0,3,SZR ;CHECK FOR RUBOUT
        JSR     @RTN,2  ;NOT A RUBOUT
        SUBZL   3,3
        STA     3,CERFL,2        ;SET RUBOUT FLAG
        JSR     @RTER,2
```

```
        JSR     @RTN,2  ;NOT SET, EQUAL COMPARE
        JMP     C11             ;SET EQUAL COMPARE ZERO OUT TEMPORARY QUINT COUN
A1:     LDA     1,CNT1,2
        MOV     1,1,SNR ;EQUAL TO ZERO
        JMP     .+3     ;YES
        MOVL#   1,1,SNC ;NEGITIVE
        JMP     A2
        LDA     1,OP1DT,2
        LDA     0,SIX,2
        SUB     1,0,SNR         ;CHECK TO SEE IF ZERO FILL
        JMP     .+4
        LDA     1,BLK,2
        STA     1,CHAR1,2
        JMP     .+3
        LDA     1,C60,2
        STA     1,CHAR1,2
        JSR     ARG2
        JSR     @RTER,2
        JMP     CHECK
A2:     JSR     ARG1
        JSR     @RTER,2
        JSR     ARG2
        JSR     @RTER,2
CHECK:  LDA     0,CHAR1,2       ;LOAD CHAR1
        LDA     1,CHAR2,2       ;LOAD CHAR2
        SUB     0,1,SNR ;CHAR2 - CHAR1
        JMP     INC3            ;RESULT = 0 (ARG1 = ARG2)
        MOVL    1,1,SZC         ;MOVE BIT ZERO TO CARRY
        JMP     INC2            ;BIT0 = 1 (ARG1 >ARG2)
        JMP     INC1            ;BIT0 = 0 (ARG1 < ARG2)
INC3:   ISZ     CHKWD,2         ;INCREMENT CHKWD (WILL END UP A 3)
INC2:   ISZ     CHKWD,2         ;INCREMENT CHKWD(WILL END UP A 2)
INC1:   ISZ     CHKWD,2         ;INCREMENT CHKWD (WILL END UP A 1)
        LDA     3,RQCHK,2       ;LOAD REQUEST CHECK
        LDA     1,THREE,2
        SUB#    3,1,SZR
        JMP     OTHER
        LDA     1,CHKWD,2       ;LOAD CHECK WORD
        SUB     3,1,SZR         ;CHKWD - RQCHK
        JMP     A0              ;RESULT = ZERO
        JMP     B1              ;FLAG IS SET
C11:    SUB     1,1
        STA     1,TQC,2
        JMP     @RTN,2          ;RETURN
A0:     LDA     1,NFLAG,2       ;LOAD NOT FLAG
        MOV     1,1,SNR         ;C11HECK IF FLAG IS SET
        JMP     C11             ;FLAG IS SET
        JSR     @RTN,2
B1:     DSZ     CNT1,2          ;FLAG NOT SET, DECREMENT CNT1
        JMP     .+1             ;CNT1 NOT EQUAL ZERO
        DSZ     CNT2,2          ;DECREMENT CNT2
        JMP     .+1             ;CNT2 NOT ZERO
        ISZ     PNT1,2          ;INCREMENT PNT1
        ISZ     PNT2,2          ;INCREMENT PNT2
        JMP     A               ;GO DO NEXT CHARACTER
ARG1:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,PNT1,2        ;LOAD OPER1 BYTE PTR
        STA     3,BYPTR,2       ;PASS TO GET BYTE ROUTINE
        JSR     @GET
        JSR     @RTER,2
```

```
        LDA     0,MSK,2 ;LOAD MASK 177
        AND     0,1     ;SAVE RIGHT 7 BITS
        STA     1,CHAR1,2       ;STORE THE CHARACTER
        JSR     @RTN,2
ARG2:   ISZ     RTST,2
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,PNT2,2        ;LOAD OPER2 BYTE POINTER
        STA     3,BYPTR,2       ;PASS TO GET BYTE ROUTINE
        JSR     @GET    ;GET THE BYTE
        JSR     @RTER,2
        LDA     0,MSK,2 ;LOAD MASK 177
        AND     0,1     ;SAVE RIGHT 7 BITS
        STA     1,CHAR2,2       ;STORE THE CHARACTER
        JSR     @RTN,2
OTHER:  LDA     1,TWO,2
        SUB#    3,1,SZR
        JMP     ONEA    ;IT IS A ONE
        LDA     1,CHKWD,2
        LDA     0,THREE,2
        SUB#    1,0,SNR         ;CHECK TO SEE IF CHARACTERS ARE EQUAL
        JMP     B1              ;YES, GET ANOTHER CHARACTER
        SUB#    3,1,SZR
        JMP     A0      ;GREATER THAN CONDITION
        JMP     NTEST
ONEA:   LDA     1,CHKWD,2
        LDA     0,THREE,2
        SUB#    1,0,SNR         ;CHECK TO SEE IF CHARACTERS ARE EQUAL
        JMP     B1              ;YES, GET ANOTHER CHARACTER
        SUB#    3,1,SZR
        JMP     A0      ;LESS THAN CONDITION
        JMP     NTEST
```

```
;
;       THESE ROUTINES RUN AN ENTERED PROGRAM
;
        .TITLE  RUN
        .NREL
        .ENT    RUN,CONT,STEP,GTONT
        .TXTM   1
RUN:    ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GCHAP,2
        JSR     @RTER,2
        .SYSTM
        .PCHAR
        JSR     @RTER,3
        LDA     1,C1,2
        SUB     0,1,SNR         ;CARRIAGE RETURN?
        JMP     RUN1            ;YES
        JSR     @NMNCP,2        ;GET #
        JMP     .+2             ;NOT #
        JMP     RUN1+1
        LDA     1,S
        SUB     0,1,SNR         ;S?
        JMP     .+3
        ISZ     CERFL,2
        JSR     @RTER,2
        JSR     @GCHAP,2
        JSR     @RTER,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,2
        LDA     0,LF,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,2
        SUBZL   1,1
        STA     1,QC,2
        JSR     @PRQNP,2
        JSR     @RTER,2
        JSR     @RTN,2
RUN1:   LDA     0,LF,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        SUBZL   1,1
        STA     1,QC,2          ;QUINT # TO RUN AT
        JMP     RUN2
CONT:   ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @RDAEP,2        ;GET CHAR
        JSR     @RTER,2
        JMP     .+1             ;NOT C/R
        SUB     0,0             ;C/R
        STA     0,STPFL,2       ;CLEAR STEP MODE
        JMP     RUN2
STEP:   ISZ     RTST,2
        STA     3,@RTST,2
        JSR     @GCHAP,2
        JSR     @RTER,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
```

```
        LDA     1,C1,2
        SUB     0,1,SZR          ;CARRAGE RETURN
        JMP     .+5
        LDA     0,LF,2
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        LDA     1,TWO,2
        STA     1,STPFL,2        ;SET STOP FLAG
RUN2:   JSR     GTQNT            ;GET QUINT POINTER
        JSR     @RTER,2
        LDA     3,QP,2
        LDA     0,1,3
        LDA     1,BELL,2
        AND     0,1
        STA     1,QPFLG,2
        MOVZR   0,0
        MOVZR   0,0
        MOVZR   0,0
        LDA     3,QPDSP,2
        ADD     0,3
        LDA     0,@0,3
        STA     0,SP,2
        JSR     @EXQNP,2         ;EXECUTE QUINT
        JSR     @RTER,2
        LDA     0,STPFL,2
        MOV     0,0,SNR          ;STOP OR STEP?
        JMP     RUN2
        MOVR    0,0,SZC
        JMP     .+3
        JSR     @PRQNP,2
        JSR     @RTER,2
        SUB     0,0
        STA     0,STPFL,2
        JSR     @RTN,2
GTQNT:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,QC,2   ;QUNIT NUMBER
        ADC     1,1
        ADD     1,0      ;-1
        MOVZL   0,1
        MOVZL   1,1
        ADD     1,0      ;*5
        LDA     1,QNST,2         ;PLUS QUINT TABLE STARTING ADDRESS
        ADD     1,0
        STA     0,QP,2
        JSR     @RTN,2
        .TXTM   0
S:      .TXT    /S/
```

```
;
;       THESE ROUTINES PROCESS THE SET SUBSCRIPT AUXILLIARY STATE TABLES
;
        .TITLE  SETSB
        .NREL
        .TXTM   1
        .ENT    SETSB,SSB11,SSB21,SSB12,SSB22,SSB13,SSB23
SETSB:  MOV     3,1
        LDA     3,QP,2
        LDA     0,2,3
        STA     0,SBF1,2
        LDA     0,3,3
        STA     0,SBF2,2
        LDA     0,4,3
        STA     0,SBF3,2
        MOV     1,3
        JMP     1,3
SSB11:  SUB     0,0
        STA     0,SBF1,2        ;CLEAR SUBFLAG FOR 1ST OPERAND
        JMP     1,3
SSB12:  SUB     0,0
        STA     0,SBF2,2        ;CLEAR SUBFLAG FOR 2ND OPERAND
        JMP     1,3
SSB13:  SUB     0,0
        STA     0,SBF3,2        ;CLEAR SUBFLAG FOR 3RD OPERAND
        JMP     1,3
SSB21:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,OP1DT,2
        MOVS    0,0
        MOVZR#  0,0,SZC         ;LEVEL =1?
        JMP     S211            ;YES
        LDA     3,PACK1,2
        LDA     1,3,3           ;SUBSCRIPT IN 4TH WORD OF PACK
        STA     1,TEMP2,2
        LDA     0,OP1DX,2       ;# OCCURS
        SUBZ    1,0,SNC
        JSR     @RTER,2         ;ARRAY EXCEEDED
        LDA     0,OP1DP,2
        STA     0,TEMP,2        ;ARRAY BYTE PTR
        LDA     0,OP1DL,2       ;LENGTH OF ARRAY
        SUB     1,1             ;ZERO CHAR TO FRONT
        JMP     S212
S211:   LDA     0,SAV1,2       ;GROUP DATA NAME ENTRY
        JSR     @DNSTP,2        ;GET DATA NAME POINTER
        JSR     @RTER,2
        LDA     3,PACK1,2
        LDA     0,3,3           ;SUBSCRIPT
        STA     0,TEMP2,2
        MOV     1,3
        STA     1,SAV1,2
        LDA     1,3,3           ;GET NUMBER OF OCCURS
        LDA     3,MSK4,2
        AND     3,1
        SUBZ    0,1,SNC
        JSR     @RTER,2         ;BOUNDS EXCEEDED
        LDA     3,SAV1,2
        LDA     0,2,3           ;GET LENGTH
        STA     0,SAV1,2        ;SAVE IT
        LDA     0,0,3
        JSR     @RSSTP,2
```

```
        JSR     @RTER,2
        STA     0,TEMP,2
        LDA     0,SAV1,2
        LDA     1,OP1DX,2       ;# CHAR FROM ARRAY
S212:   JSR     SSB00           ;STORE ACTUAL BYTE PTR
        JSR     @RTER,2
        STA     1,OP1DP,2
        JSR     @RTN,2
SSB22:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,OP2DT,2
        MOVS    0,0
        MOVZR#  0,0,SZC         ;LEVEL =1?
        JMP     S221            ;YES
        LDA     3,PACK2,2
        LDA     1,3,3           ;SUBSCRIPT IN 4TH WORD OF PACK
        STA     1,TEMP2,2
        LDA     0,OP2DX,2       ;# OCCURS
        SUBZ    1,0,SNC
        JSR     @RTER,2         ;ARRAY EXCEEDED
        LDA     0,OP2DP,2
        STA     0,TEMP,2        ;ARRAY BYTE PTR
        LDA     0,OP2DL,2       ;LENGTH OF ARRAY
        SUB     1,1             ;ZERO CHAR TO FRONT
        JMP     S222
S221:   LDA     0,SAV2,2       ;GROUP DATA NAME ENTRY
        JSR     @DNSTP,2        ;GET DATA NAME POINTER
        JSR     @RTER,2
        LDA     3,PACK2,2
        LDA     0,3,3           ;SUBSCRIPT
        STA     0,TEMP2,2
        MOV     1,3
        STA     1,SAV1,2
        LDA     1,3,3    ;GET NUMBER OF OCCURS
        LDA     3,MSK4,2        ;GET MASK
        AND     3,1      ;SAVE RIGHT BYTE
        SUBZ    0,1,SNC
        JSR     @RTER,2         ;BOUND EXCEEDED
        LDA     3,SAV1,2
        LDA     0,2,3
        STA     0,SAV1,2
        LDA     0,0,3
        JSR     @RSSTP,2
        JSR     @RTER,2
        STA     0,TEMP,2
        LDA     0,SAV1,2
        LDA     1,OP2DX,2       ;# CHAR FROM ARRAY
S222:   JSR     SSB00           ;STORE ACTUAL BYTE PTR
        JSR     @RTER,2
        STA     1,OP2DP,2
        JSR     @RTN,2
SSB23:  ISZ     RTST,2
        STA     3,@RTST,2
        LDA     0,OP3DT,2
        MOVS    0,0
        MOVZR#  0,0,SZC         ;LEVEL =1?
        JMP     S231            ;YES
        LDA     3,PACK2,2
        LDA     1,3,3           ;SUBSCRIPT IN 4TH WORD OF PACK
        STA     1,TEMP2,2
        LDA     0,OP3DX,2       ;# OCCURS
```

```
        SUBZ    1,0,SNC
        JSR     @RTER,2         ;ARRAY EXCEEDED
        LDA     0,OP3DP,2
        STA     0,TEMP,2        ;ARRAY BYTE PTR
        LDA     0,OP3DL,2       ;LENGTH OF ARRAY
        SUB     1,1             ;ZERO CHAR TO FRONT
        JMP     S232
S231:   LDA     0,SAV3,2       ;GROUP DATA NAME ENTRY
        JSR     @DNSTP,2        ;GET DATA NAME POINTER
        JSR     @RTER,2
        LDA     3,PACK2,2
        LDA     0,3,3           ;SUBSCRIPT
        STA     0,TEMP2,2
        MOV     1,3
        STA     1,SAV1,2
        LDA     1,3,3   ;GET NUMBER OF OCCURS
        LDA     3,MSK4,2        ;GET MASK
        AND     3,1
        SUBZ    0,1,SNC
        JSR     @RTER,2         ;BOUNDS EXCEEDED
        LDA     3,SAV1,2
        LDA     0,2,3
        STA     0,SAV1,2        ;GROUP LENGTH
        LDA     0,0,3
        JSR     @RSSTP,2
        JSR     @RTER,2
        STA     0,TEMP,2
        LDA     0,SAV1,2
        LDA     1,OP3DX,2       ;# CHAR FROM ARRAY
S232:   JSR     SSB00           ;STORE ACTUAL BYTE PTR
        JSR     @RTER,2
        STA     1,OP3DP,2
        JSR     @RTN,2
;
;       SUBROUTINE CALCULATES BYTE POINTER FOR SUBSCRIPT
;
;       INPUT IS:
;               AC0             TOTAL LENGTH OF ONE ARRAY ELEMENT
;               AC1             NUMBER OF CHARACTERS FROM START OF (GROUP)
;               TEMP,2          (GROUP) ARRAY STARTING POINTER
;               TEMP2,2         SUBSCRIPT NUMBER (INTO ARRAY)
SSB00:  ISZ     RTST,2
        STA     3,@RTST,2
        SUB     3,3
        DSZ     TEMP2,2
        JMP     .+2
        JMP     S02
S01:    ADD     0,3             ;MULTIPLY LENGTH * SUBSCRIPT-1
        DSZ     TEMP2,2
        JMP     S01
S02:    ADD     3,1             ;ADD # CHAR TO START OF ARRAY
        LDA     3,TEMP,2
        ADD     3,1             ;ADD # CHAR FROM START OF CORE
        JSR     @RTN,2
```

```
        .TITL   SHIFTLEFT
        .ENT    MOLF,MOZLF
        .NREL
;CALL TO THIS ROUTINE IS (N,PAC1,PAC2)
MOZLF:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        JSR     SETZ
        JSR     @RTER,2
        LDA     1,@PAC1,2       ;GET FIRST ARGUMENT
        MOVZ    1,1
        JMP     MOLF+12         ;CLEAR CARRY DO A SHIFT-LEFT MOVE
MOLF:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        JSR     SETZ
        JSR     @RTER,2
        JMP     .+6
        DSZ     I,2     ;DECREMENT LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @RTN,2  ;RETURN
        DSZ     PAC1,2  ;GET NEXT ADDRESS
        DSZ     PAC2,2  ;GET NEXT STORE POSITION
        LDA     1,@PAC1,2       ;GET FIRST ARGUMENT
        MOVL    1,1             ;SHIFT LEFT
        STA     1,@PAC2,2       ;STORE IT
        JMP     MOLF+5          ;DO AGAIN
SETZ:   ISZ     RTST,2
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,N,2           ;GET NUMBER OF WORDS LONG
        STA     1,I,2           ;STORE IT IN I
        LDA     3,PAC1,2        ;GET ARGUMENT TO BE SHIFTED
        ADD     1,3             ;ADD NUMBER OF WORDS TO BE USED
        STA     3,PAC1,2        ;STORE BACK IN PAC1
        LDA     3,PAC2,2        ;GET PLACE TO STORE AFTER MOVE
        ADD     1,3             ;TO GET PROPER PLACEMENT
        STA     3,PAC2,2        ;STORE IT BACK OUT IN PAC2
        DSZ     PAC1,2          ;TO GET PROPER ALIGNMENT
        DSZ     PAC2,2          ;FOR STORAGE
        JMP     @RTN,2         ;RETURN
```

```
        .TITL   SHIFTRIGHT
        .ENT    SHIFT
        .EXTN   RETRN
        .NREL
;CALL TO THIS ROUTINE IS (N,PAC1,PAC2).
SHIFT:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,N,2           ;GET NUMBER OF WORDS TO BE SHIFTED
        STA     3,NT,2          ;STORE IN NO
AA:     LDA     1,@PAC1,2       ;GET FIRST WORD TO BE SHIFTED
        MOVR    1,1             ;SHIFT
        STA     1,@PAC2,2       ;STORE IN PAC2
        ISZ     PAC1,2  ;GET NEXT WORD TO BE SHIFTED
        ISZ     PAC2,2          ;GET NEXT STORE POSITION
        DSZ     NT,2            ;DECREMENT NUMBER OF WORDS
        JMP     AA             ;SKIP NEXT STATEMENT IF NOT ZERO
        JMP     @RTN,2          ;RETURN
```

```
          .TITL   SIGN
          .ENT    PLUSS,MYNUS
          .EXTN   GETBT,PUTTT,B1,RETRN,ERTRN
          .EXTD   PUT,GET
          .NREL
PLUSS:    LDA     1,PLUS,2        ;GET A PLUS SIGN
          STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
          JMP     TEST
MYNUS:    LDA     1,MINUS,2       ;GET THE EDIT SIGN
          STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
          JMP     TEST    ;FIND OUT WHAT TYPE OF SIGN
TEST:     LDA     1,SIGN,2        ;GET THE SIGN
          MOVZR#  1,1,SEZ
          JMP     SITS    ;SIGN TRAILING SEPARATE TYPE THREE
          MOV     1,1,SNR
          JMP     SILI    ;SIGN LEADING INCLUDED TYPE ZERO
          MOVZR#  1,1,SZR
          JMP     SITI    ;SIGN TRAILING INCLUDED TYPE TWO
          JMP     SILS    ;SIGN LEADING SEPARATE TYPE ONE
SILS:     LDA     1,PNT0,2        ;GET BYTE POINTER TO DATA
          MOV     1,1,SZR ;TEST TO SEE IF BYTE POINTER IS SET
          JSR     BLNK    ;JMP TO BLANK FILL
          LDA     3,PNT1,2        ;GET DATA
          STA     3,BYPTR,2       ;STORE IN BYTE POINTER
          JSR     @GET    ;GO TO GET BYTE ROUTINE
          JMP     @RTER,2
          SUB     0,0     ;CLEAR AC0
          SUB#    0,1,SZR ;TEST TO SEE IF SIGN
          JMP     .+2     ;FIELD IS NEGATIVE
          JMP     POSIT   ;FIELD IS POSITIVE
          LDA     1,RCNT,2        ;GET REPITITION COUNT
          LDA     1,ONE,2 ;GET A ONE
          SUB#    0,1,SNR ;TEST TO SEE IF EQUAL TO ONE
          JMP     ONMIN   ;ONE MINUS SIGN
NEGA:     LDA     1,MINUS,2       ;GETA MINUS SIGN
          STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
          JMP     @GTBY   ;GO TO GET BYTE ROUTINE
ONMIN:    LDA     1,MINUS,2       ;GET A MINUS SGN
          STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
          JMP     @PUTT1
POSIT:    LDA     1,PLUS,2        ;GET A PLUS SIGN
          LDA     0,ECHAR,2       ;GET EDIT CHARACTER
          SUB#    0,1,SNR ;TEST TO SEE IF EQUAL
          JMP     .+4
          LDA     0,BLK,2 ;GET A BLANK FIELD IS POSITIVE
          STA     0,ECHAR,2       ;STORE IN EDIT CHARACTER, EDIT CHARACTER IS NEGA
          JMP     @GTBY   ;GET BYTE ROUTINE
          LDA     1,RCNT,2        ;GET REPITITION COUNT
          LDA     1,ONE,2         ;GET A ONE
          SUB#    0,1,SNR ;TEST TO SEE IF EQUAL TO ONE
          JMP     ONPLS   ;ONE PLUS SIGN
          LDA     0,PNT0,2        ;GET BYTE POINTER
          MOV#    0,0,SNR ;TEST TO SEE IF EQUAL TO ZERO
          JMP     .+2
          JMP     .+4
          LDA     1,PLUS,2        ;GET A PLUS SIGN
          STA     1,ECHAR,2       ;STORE IN EDITCHARACTER
          JMP     @GTBY   ;GO TO GET BYTE ROUTINE
          LDA     1,BLK,2 ;STORE IN EDIT CHARACTER
          STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
          STA     1,ECHT,2        ;STORE IN EDIT CHARACTER TEMPOSRARUY
```

```
        JMP     @PUTT1  ;GO TO STORE ROUTINE
ONPLS:  LDA     1,PLUS,2        ;GET A MINUS SIGN
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        JMP     @PUTT1
BLNK:   ISZ     RTST,2
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,BLK,2 ;GET A BLANK
        LDA     3,PNT2,2        ;GET STORE POINTER
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE THE BLANK
        JMP     @RTER,2
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        DSZ     RCNT,2  ;DECREMENT REPITITION COUNT
        JMP     .+2     ;NOT ZERO
        JMP     @.B1    ;GET ANOTHER EDIT CODE
        DSZ     CNT0,2  ;GET NEXT BLANK AREA
        JMP     .+4     ;NOT ZERO
        SUB     1,1     ;CLEAR AC1
        STA     1,PNT0,2        ;CLEAR ZERO FLAG
        JMP     AAA     ;RESTORE POINTERS
        DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     BLNK+1  ;DO AGAIN
        JMP     @RTN,2  ;RECEIVING FIELD TO SHORT
AAA:    LDA     1,OP1DP,2       ;GET BYTE POINTER TO SENDING FIELD
        STA     1,PNT1,2        ;RESTORE BYTE POINTER
        LDA     1,OP1DL,2       ;GET THE LENGTH TO SENDING FIELD
        STA     1,CNT1,2        ;RSTORE COUNTER
        JSR     @RTN,2  ;RETURN
SITS:   DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     @RTER,2 ;NOT ZERO SOMETHING IS WRONG
        LDA     3,PNT1,2        ;GET THE SIGN
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET    ;GO TO GET BYTE ROUTINE
        JMP     @RTER,2
        MOV#    1,1,SNR ;CHECK TO SEE IF FIELD IS POSITIVE
        JMP     POT     ;FIELD IS BLANK PLUS RESULT
        LDA     3,PNT2,2        ;GET BYTE POINTER TO RECEIVING FIELD
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;GO TO STORE CHARACTER ROUTINE
        JMP     @RTER,2
        JMP     @RTN,2  ;RETURN TO INTERPPETER
POT:    LDA     1,MINUS,2       ;TEST TO SEE IF MINUS SIGN
        LDA     0,ECHAR,2       ;;BRING IN CURRENT EDIT CHARACTER
        SUB#    0,1,SZR ;TEST TO SEE IF EQUAL
        JMP     .+5     ;FIELD IS POSITIVE
        LDA     1,BLK,2 ;BRING IN A BLANK
        LDA     3,PNT2,2        ;GET POINTER TO RECEIVING FIELD
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE THE CHARACTER
        JMP     @RTER,2
        JMP     @RTN,2  ;RETURN TO INTERPPETER
        LDA     1,ECHAR,2       ;;GET THE EDIT CHARACTER
        JMP     .-6     ;PUT THE CHARACTER OUT
SILI:   LDA     0,NEGNO,2       ;GET NEGATIVE NUMBER FLAG
        MOVR    0,0,SNC ;TEST TO SEE IF SET
        JMP     POSIT   ;NOT SET NUMBER IS POSTIVE
        LDA     1,RCNT,2        ;GET REPITITION COUNT
        LDA     1,ONE,2 ;GET A ONE
        SUB#    0,1,SNR ;TEST T SEE IF EQUAL OTO ONE
        JMP     ONMIN   ;ONE MINUS SIGN
        JMP     NEGA    ;ZERO SUPPRESS PUT OUT SIGN
```

```
SITI:   LDA     0,NEGNO,2       ;GET NEGATIVE NUMBER FLAG
        MOVR    0,0,SNC ;TEST TO SEE IF SET
        JMP     POSIT   ;NOT SET NUMBER IS POSITIVE
        JMP     .-10    ;NUMBER IS NEGATIVE
PUTT1:  PUTTT
.B1:    B1
GTBY:   GETBT
```

```
        .TITL   SIZEROUND
        .ENT    SIZER
        .NREL
SIZER:  ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        SUBZL   1,1             ;GENERATE A ONE
        STA     1,MPFLG,2.      ;SET SIZE ERROR FLAG
        JSR     @RTN,2
```

```
        .TITLE  START
        .ENT    START
        .EXTN   COBOL
        .NREL
START:  JMP     @COBLP
COBLP:  COBOL
        .END    START
```

```
        .TITL   STOP
        .ENT    STOP1,STOP2,KILL
        .NREL
KILL:   JSR     @RDAEP,2
        JSR     @RTER,2
        JMP     .+1
        .SYSTEM                 ;THIS IS THE ENTRY POINT TO
        .RESET                  ;TERMINATE THE PROGRAM  ALL
        JSR     @RTER,3         ;PRESENTLY OPENED FILES ARE
        .SYSTEM                 ;CLOSED, AND CONTROL IS
        .RTN                    ;PASSED BACK TO THE
        JSR     @RTER,3         ;OPERATING SYSTEM
STOP1:  ISZ     RTST,2              ;GET NEXT STORE POSITION
        STA     3,@RTST,2           ;SAVE RETURN ADDRESS
        ISZ     STPFL,2             ;SET STOP FLAG
        JSR     @RTN,2
STOP2:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        .SYSTEM
        .GCHAR                  ;GET A CHARACTER FROM THE TTI
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        LDA     1,RBMSK,2       ;LOAD AC1 WITH BYTE MASK
        AND     1,0             ;CLEAR THE LEFT BYTE
        LDA     3,C1,2          ;LOAD AC3 WITH CARRIAGE RETURN
        SUB     0,3,SZR         ;COMPARE
        JMP     .+2             ;NOT EQUAL, ECHO THE CHARACTER
        JMP     STOP3           ;EQUAL
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        JMP     STOP2+2
STOP3:  .SYSTEM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        LDA     0,LF,2          ;LOAD LINE FEED CHARACTER
        .SYSTEM
        .PCHAR
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        JMP     @RTN,2      ;EQUAL, PROCESSING COMPLETE, RETURN
```

```
        .TITL   STORECHANNEL
        .ENT    STRIT
        .NREL
STRIT:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADRESS
        LDA     3,OP1DP,2       ;LOAD I/O BYTE PTR.
        LDA     1,CHANL,2.      ;LOAD CHANNEL NO.
        MOVZR   3,3             ;CONVERT BYTE PTR. TO WORD PTR.
        STA     1,5,3           ;STORE CHANNEL NO. IN WORD SIX
                                ;OF I/O ENTRY
        JMP     @RTN,2          ;RETURN. CHANNEL NO. STORED
```

```
;
;       STATE TABLES
;
          .TITLE STTBL
         .NREL
         .ENT     OP1DS, OP2DS, OP3DS, OP1DR, OP2DR, OP3DR
         .ENT     OPDST
         .EXTN    SSB11, SSB12, SSB13, SSB21, SSB22, SSB23, SETSB
         .EXTN OP1D, OP2D, OP3D, OP4D, OP5D, OP6D, CLOS, GOTO1, WRIT, MOVE, OPEN, ALPHA
         .EXTN    STRIT, PERM1, READ, STOP1, STOP2, EXIT1, EQUAL, LESS, GREAT, NUMRC
         .EXTN    LDPK1, LDPK2, LDPK3, ADDPK, M3TO1, UNPK1, UNPK2, DVDPK, M2TO3, MULT, SUBPK
         .EXTN    APT1, DSPLY, AG1, AG2, SIZER, FLDEC, SFLDP
OPDST:   .
         001
         002
         003
         004
         005
         006
         007
         008
         009
         010
         011
         012
         013
         014
         015
         016
         017
         018
         019
        020
         021
         022
         023
         024
         025
         026
         027
         028
         0
         RTTOI=100000
019:     .+1
         SIZER
         RTTOI
003:     .+1
         OP6D
         CLOS
         RTTOI
007:     .+1
         OP4D
         GOTO1
         RTTOI
008:     .+1
         OP5D
         WRIT
         RTTOI
009:     .+1
         OP1D
         AG1
```

```
        OP2D
        AG2
        MOVE
        RTTOI
011:    .+1
        OP6D
        OPEN
        STRIT
        RTTOI
012:    .+1
        PFRM1
        RTTOI
013:    .+1
        OP4D
        OP5D
        READ
        RTTOI
015:    .+1
        STOP1
        RTTOI
028:    .+1
        OP1D
        DSPLY
        STOP2
        RTTOI
017:    .+1
        EXIT1
        RTTOI
020:    .+1
        OP1D
        AG1
        OP2D
        AG2
        OP4D
        EQUAL
        RTTOI
021:    .+1
        OP1D
        AG1
        OP2D
        AG2
        OP4D
        LESS
        RTTOI
022:    .+1
        OP1D
        AG1
        OP2D
        AG2
        OP4D
        GREAT
        RTTOI
023:    .+1
        OP1D
        AG1
        OP4D
        NUMRC
        RTTOI
024:    .+1
        OP1D
        AG1
```

```
        OP4D
        ALPHA
        RTTOI
026:    . +1
        OP1D
        OP2D
        OP3D
        LDPK1
        LDPK2
        ADDPK
        M3TO1
        LDPK3
        ADDPK
        UNPK2
        RTTOI
002:    . +1
        OP1D
        OP2D
        OP3D
        LDPK1
        LDPK2
        ADDPK
        UNPK2
        RTTOI
005:    . +1
        OP1D
        OP2D
        OP3D
        SFLDP
        LDPK2
        DVDPK
        FLDEC
        M2TO3
        UNPK2
        RTTOI
010:    . +1
        OP1D
        OP2D
        OP3D
        LDPK1
        LDPK2
        MULT
        UNPK2
        RTTOI
027:    . +1
        OP1D
        OP2D
        OP3D
        LDPK1
        LDPK2
        ADDPK
        M3TO1
        LDPK3
        SUBPK
        UNPK2
        RTTOI
016:    . +1
        OP1D
        OP2D
        OP3D
        LDPK1
```

```
          LDPK2
          SUBPK
          UNPK2
          RTTOI
001:      .+1
          OP1D
          APT1
          RTTOI
004:      .+1
          OP1D
          DSPLY
          RTTOI
OP1DS:    LDPK1                    ;STATE TABLE FOR SUBSCRIPT ON OPERAND 1
          SSB11
          OP1D
          SSB21
OP1DR:    .BLK     1               ;RETURN BRANCH (@SP+1 FOR OPCODE)
OP2DS:    LDPK2
          SSB12
          OP2D
          SSB22
OP2DR:    .BLK     1
OP3DS:    LDPK3
          SSB13
          OP3D
          SSB23
OP3DR:    .BLK     1
025:      .+1
          OP1D
          OP2D
          OP3D
          SETSB
          RTTOI
          .END
```

```
        .TITL   SUBTRACT
        .ENT    SUBT,SUBPK
        .NREL
;THE CALL TO THIS ROUTINE IS (N,PAC1,PAC2,PAC3)
SUBPK:  LDA     1,TEN,2 ;GET AN EIGHT FOR EIGHT WORD SUBTRACT
        STA     1,N,2   ;STORE IN N
        LDA     1,PACK1,2       ;INITILIZE ADDRESS
        STA     1,PAC1,2        ;FOR SUBTRACT
        LDA     1,PACK2,2       ;ROUTINE
        STA     1,PAC2,2
        LDA     1,PACK3,2
        STA     1,PAC3,2
SUBT:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,N,2           ;GET NUMBER OF WORDS TO BE SUBTRACTED
        STA     3,I,2           ;STORE IN I
        LDA     1,PAC1,2        ;GET FIRST FIELD TO BE SUBTRACT
        ADD     3,1             ;GET FIRST WORD
        STA     1,PAC1,2        ;STORE BACK OUT
        LDA     1,PAC2,2        ;DO THE SAME FOR PAC2
        ADD     3,1
        STA     1,PAC2,2
        LDA     1,PAC3,2        ;DO THE SAME FOR PAC3
        ADD     3,1
        STA     1,PAC3,2
        DSZ     PAC1,2          ;TO GET PROPER ALIGNMENT
        DSZ     PAC2,2          ;FOR SUBTRACTION AND
        DSZ     PAC3,2          ;FOR STORING
START:  LDA     1,@PAC1,2       ;GET FIRST ARGUMENT
        LDA     0,@PAC2,2       ;GET SECOND ARGUMENT
        SUBZ    1,0,SZC             ;SUBTRACT THE TWO
        JMP     SUBB
SUBA:   STA     0,@PAC3,2
        DSZ     I,2                 ;DECREMENT NUMBER OF WORDS
        JMP     .+2             ;NOT ZERO
        JMP     @RTN,2            ;RETURN
        DSZ     PAC1,2          ;TO GET NEXT WORD FOR
        DSZ     PAC2,2          ;SUBTRACTION AND
        DSZ     PAC3,2          ;FOR STORING
        LDA     1,@PAC1,2       ;GET FIRST ARGUMENT
        LDA     0,@PAC2,2       ;GET SECOND ARGUMENT
        ADCZ    1,0,SNC             ;ADD COMPLEMENT
        JMP     SUBA
SUBB:   STA     0,@PAC3,2       ;STORE ANSWER
        DSZ     I,2
        JMP     .+2
        JMP     @RTN,2
        DSZ     PAC1,2
        DSZ     PAC2,2
        DSZ     PAC3,2
        LDA     1,@PAC1,2           ;GET FIRST ARGUMENT
        LDA     0,@PAC2,2       ;GET SECOND ARGUMENT
        SUBZ    1,0,SZC             ;SUBTRACT
        JMP     SUBB
        JMP     SUBA
```

```
        .TITL   TYPE
        .ENT    TYPE1
        .EXTN   LDPK1,LDPK2,UNPK1
        .NREL
TYPE1:  ISZ     RTST,2  ;SAVE RETURN ADDRESS
        STA     3,@RTST,2
        LDA     3,INDIC,2       ;GET INDICATOR
        LDA     0,TABEL         ;GET BEGINNING OF TABLE
        ADD     0,3     ;GET ROUTINE
        LDA     3,0,3   ;GET NAME OF ROUTINE
        JMP     0,3
TABEL:  TAB
TAB:    XXX     ;ERROR RETURN NO INDICATOR ZERO
        TYPE0   ;TYPE ONE
        TYPE2
        TYPE3
        TYPE4
        TYPE5
        TYPE6
        TYPE7
TYPE0:   LDA     0,DEC1,2     ;GET ASSUMMED DECIMAL PT
        MOV#    0,0,SZR         ;CHECK TO SEE IF EQUAL
XXX:    JMP     @RTER,2         ;NO, RUN TIME ERROR
        JSR     @RTN,2
TYPE2:  SUBZL   1,1     ;GENERATE A ONE
        STA     1,EFLAG,2       ;STORE IN EDIT FLAG
        JSR     @RTN,2
TYPE3:  JSR     @LDP1           ;YES COMP TO COMP MOVE
        JMP     @RTER,2
        LDA     1,PACK1,2       ;GET SENDING FIELD
        STA     1,PAC1,2        ;STORE IN PAC1
        LDA     1,PACK2,2       ;GET RECEIVING FIELD
        STA     1,PAC2,2        ;STORE IN PAC2
        JSR     @MMOV,2 ;GO TO MATHMOVE ROUTINE
        JMP     @RTER,2
        JSR     @UNP1   ;UNPACK COMP
        JMP     @RTER,2
        JMP     @RTN,2
TYPE4:  JSR     @LDP1           ;YES BINARY TO ASCII MOVE
        JMP     @RTER,2
        LDA     1,PACK1,2       ;SET ADDRESS FOR MATHMOVE ROUTINE
        STA     1,PAC1,2
        LDA     1,PACK2,2
        STA     1,PAC2,2
        JSR     @MMOV,2
        JMP     @RTEP,2
        JSR     @UNP1
        JMP     @RTER,2
        JMP     @RTN,2
TYPE5:  JSR     @LDP1   ;YES ASCII TO BINARY MOVE
        JMP     @RTER,2
        LDA     1,PACK1,2       ;INITILIZE ADDRESS'S FOR MATHMOVE
        STA     1,PAC1,2
        LDA     1,PACK2,2
        STA     1,PAC2,2
        JSR     @MMOV,2
        JMP     @RTER,2
        JSR     @UNP1   ;THE UNPACK ROUTINE
        JMP     @RTER,2
        JMP     @RTN,2
TYPE6:  LDA     0,OP1DL,2       ;GET LENGTH OF SENDING FIELD
```

```
        LDA     1,DEC1,2    ;GET DECIMAL POINT POINTER
        SUB     1,0          ;COMPUTER # OF DIGITS LEFT OF DEC PT.
        STA     0,DIGIT,2    ;STORE IT
        LDA     0,OP2DL,2    ;YES COMPUTE RECEIVING LENGTH LEFT
        LDA     1,DEC2,2    ;D.P. POSITION
        SUB     1,0          ;COMPUTE # OF DIGITS LEFT OF D.P.
        STA     0,RDGIT,2    ;STORE REMAINING DIGIT
END:    LDA     1,DIGIT,2    ;GET SENDING DIGIT
        LDA     0,RDGIT,2    ;GET RECEIVING DIGIT
        SUBZ     1,0,SNC      ;WHAT IS LEFT
        JMP     BAD          ;NEG RESULT WONT FIT
        MOV#    0,0,SZR
        JMP     .+4
        STA     0,CNT0,2
        STA     0,PNT0,2
        JSR     @RTN,2
        SUBZL     3,3          ;GENERATE A ZERO
        STA     3,PNT0,2     ;SET BYTE PTR. TO ZERO
        STA     0,CNT0,2     ;STORE RESULT IN CNT1
        JMP     @RTN,2
BAD:    LDA     3,CNT1,2     ;GET CNT1
        ADD     0,3          ;ADD RESULT TO CNT1,2
        STA     3,CNT1,2
        LDA     3,PNT1,2     ;GET BYTE PTR.
        SUB     0,3          ;SUBTRACT RESULT, RUNTIME ERROR
        STA     3,PNT1,2
        JSR     @RTN,2
TYPE7:  SUBZL   1,1
        STA     1,EFLAG,2
        LDA     0,OP1DL,2
        LDA     1,DEC1,2
        SUB     1,0
        STA     0,DIGIT,2
        LDA     1,DEC2,2
        LDA     3,OP2DE,2       ;GET BYTE POINTER TO EDIT STRING
        MOVR    3,3     ;CONVERT TO WORD POINTER
        LDA     3,0,3   ;GET LENGTH OF FILL CHARACTER
        SUB     1,3     ;TO GET DECIMAL POSITION TO LEFT OF DECIMAL POINT
        SUB     1,1     ;TO GET DECIMAL POSITION TO LEFT OF DECIMAL POINT
        STA     3,RDGIT,2
        ISZ     OP2DE,2
        ISZ     OP2DE,2
        JMP     END
LDP1:   LDPK1
LDP2:   LDPK2
UNP1:   UNPK1
```

```
        .TITL   UNPACK
        .ENT    UNPK1,UNPK2
        .EXTD   PUT,GET
        .NREL
UNPK1:  LDA     1,PACK2,2       ;INITILIZE UNPACK ROUTINE
        STA     1,PACTL,2       ;FOR PACK2
        LDA     1,OP2DT,2
        STA     1,OPDT,2
        LDA     1,OP2DL,2
        STA     1,OPDL,2
        LDA     1,DEC2,2
        STA     1,OPDE,2
        LDA     1,OP2DP,2
        STA     1,OPDP,2
        JMP     UNPK
UNPK2:  LDA     1,PACK3,2       ;INITILIZE UNPACK ROUTINE
        STA     1,PACTL,2       ;FOR PACK3
        LDA     1,OP3DT,2
        STA     1,OPDT,2
        LDA     1,OP3DL,2
        STA     1,OPDL,2
        LDA     1,OP3DP,2
        STA     1,OPDP,2
        LDA     1,DEC3,2
        STA     1,OPDE,2
UNPK:   ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        SUBC    1,1
        STA     1,EFLAG,2
        LDA     1,@PACTL,2      ;GET FIRST FIELD TO SEE IF PLUS OR MINUS
        MOVL    1,1,SZC ;TEST TO SEE IF ZERO BIT IS SET
        JMP     CHANG   ;YES JUMP TO NEGATIVE NUMBER ROUTINE
        LDA     3,OPDT,2    ;GET TYPE
        SUBZ    0,0         ;CLEAR AC0
        SUB#    3,0,SNR     ;CHECK TO SEE IF EQUAL TO ZERO
        JMP     COMP        ;YES A COMP TO COMP MOVE
        LDA     0,TEN,2     ;NO, GET A EIGHT
        STA     0,N,2       ;STORE IN N
        LDA     0,OPDE,2    ;GET DECIMAL POSITION
        STA     0,MM,2      ;STORE IN M
        ISZ     MM,2        ;BUMP M TO TEST IF
AB:     DSZ     MM,2        ;EQUAL TO ZERO
        JMP     .+2         ;NOT EQUAL TO ZERO
        JMP     AC          ;M IS EQUAL TO ZERO
        JSR     @MTEN,2     ;GO TO MULTIPLY BY 10 ROUTINE
        JSR     @RTER,2
        JMP     AB          ;DO AGAIN
AC:     LDA     1,OPDL,2    ;GET RECEIVING FIELD LENGTH
        LDA     0,OPDP,2    ;GET BYTE POINTER
        ADD     1,0         ;ADD LENGTH TO BYTEPOINTER TO GET END
        STA     0,OPDP,2    ;OF FIELD STORE BACK OUT
        DSZ     OPDP,2
AA:     LDA     1,FIVE,2
        STA     1,N,2
      JSR     @DV10,2       ;GO TO DIV10 ROUTINE
        JSR     @RTEP,2
        LDA     1,FOUR,2
        STA     1,N,2
        LDA     0,PACTL,2       ;GET THEE REMAINDER
        LDA     1,N,2   ;GET THE NUMBER OF WORDS LONG
        ADD     0,1
```

```
        STA     1,PACT3,2       ;STORE POSITION OUT
        SUB     0,0     ;CLEAR AC0
        LDA     1,@PACT3,2      ;GET THE REMAINDER
        MOVS    1,1     ;SWAP BYTES
        STA     0,@PACT3,2      ;STORE A ZERO IN THAT POSITION
        MOVZL#  1,1,SZC         ;SHIFT LEFT CHA:ECK CARRY
        INC     0,0     ;CARRY IS SET
        LDA     3,RBMSK,2       ;GET BYTE MASK 377
        AND     3,1     ;SAVE RIGHT BYTE
        ADD     0,1     ;ADD OVER FLOW IF ANY
        MOVZL   1,0     ;MULTIPLY BY TEN
        MOVZL   0,0
        ADDZL   0,1
        MOVS    1,1     ;SWAP BYTES AGAIN
        MOVZL#  1,1,SZC ;CHECK TO SEE IF CARRY IS SET
        INC     1,1     ;YES
        AND     3,1     ;SAVE RIGHT BYTE
        LDA     0,C60,2 ;GET AN ASCII 60
        ADD     0,1     ;CHANGE TO A NUMBER
        STA     1,CHAR1,2       ;STORE IT
        LDA     3,X3T,2 ;TEST TO SEE IF FLAG IS SET
        MOV#    3,3,SNR
        JMP     TEST    ;NOT SET GO TO TEST FOR SIGN
SET:    LDA     3,OPDP,2        ;GET TEMP BYTE POINTER
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        LDA     1,CHAR1,2       ;GET THE CHARACTER
        JSR     @PUT            ;STORE IT
        JSR     @RTER,2
        DSZ     OPDP,2          ;DECREMENT BYTE POINTER
        DSZ     OPDL,2          ;DECREMENT LENGTH
        JMP     .+2
        JMP     @RTN,2      ;RETURN
        SUBZL   3,3     ;GENERATE A ONE
        LDA     1,OPDL,2        ;GET LENGTH
        SUB#    3,1,SZR ;TEST TO SEE IF EQUAL TO ONE
        JMP     AA
        LDA     1,EFLAG,2       ;GET FLAG
        MOV     1,1,SNR ;TEST TO SEE IF SET
        JMP     AA      ;ZERO NOT SET
        JMP     SILI+2
CHANG:  SUBZL   1,1     ;GENERATE A ONE
        STA     1,NEGNO,2       ;STORE IN NEGATIVE NUMBER FLAG
        LDA     1,PACTL,2       ;GET FIELD TO BE UNPACKED
        STA     1,PAC1,2        ;STORE ADDRESS IN TEMPORARY
        LDA     1,PACT,2        ;ZEROOUT TEMPORARY PACK FIELD
        STA     1,PACZ,2
        JSR     @ZERA,2 ;ZERO OUT ROUTINE
        JSR     @RTER,2 ;ERROR RETURN NOT POSSIBLE ERROR
        LDA     1,PACT,2        ;INITILIZE ADDRESS FOR SUBTRACT ROUTINE
        STA     1,PAC2,2
        LDA     1,PACTM,2
        STA     1,PAC3,2
        JSR     @SUBT1,2        ;GO TO SUBTRACT ROUTINE
        JSR     @RTER,2 ;ERROR RETURN NO POSSIBLE ERROR
        LDA     1,PACTM,2       ;GET ANSWER
        STA     1,PAC1,2        ;AND MOVE BACK TO
        LDA     1,PACTL,2       ;ORGINA PACK FIELD
        STA     1,PAC2,2
        JSR     @MMOV,2 ;MATHMOVE ROUTINE
        JSR     @RTER,2 ;ERROR RETURN NO POSSIBLE ERROR
        JMP     UNPK+5  ;CONTINUE ON
```

```
COMP:   LDA     1,FIVE,2        ;GET A FIVE
        STA     1,N,2           ;STORE IN N
        SUBZ    1,1             ;GENERATE A ZERO
        STA     1,MM,2           ;STORE IN M
        SUBZL   1,1             ;GENERATE A ONE
        STA     1,I,2           ;STORE IN I
        LDA     1,PACTL,2       ;GET FIRST ADDRESS
        STA     1,PAC1,2        ;STORE IT
AE:     LDA     1,@PAC1,2       ;GET FIRST ANSWER
        MOV#    1,1,SNR ;TEST TO SEE IF EQUAL TO ZERO
        JMP     AG              ;YES
        ISZ     I,2             ;NO
        ISZ     MM,2
        ISZ     PAC1,2
        LDA     1,FOUR,2        ;GET A FOUR
        LDA     0,I,2           ;GET I
        SUB#    1,0,SNR         ;TEST TO SEE IF EQUAL TO FOUR
        JMP     AG              ;YES
        JMP     AE              ;NO
AG:     LDA     1,PACTL,2       ;INITILIZE FOR
        STA     1,PAC1,2        ;MATHMOVE ROUTINE
        LDA     1,OPDP,2      ;GET ADDRESS OF DATA
        MOVZR   1,1       ;CONVERT TO WORD ADDRESS
        STA     1,PAC2,2
        JSR     @MMOV,2
        JSR     @RTER,2
        LDA     1,MM,2           ;GET M
        STA     1,OPDE,2      ;STORE IN DECIMAL POINTER
        JMP     @RTN,2        ;RETURN
TEST:   SUBZL   3,3       ;CLEAR AC3
        STA     3,X3T,2 ;STORE THE ONE IN FLAG
        LDA     3,SIGN,2          ;TO TEST TO SEE IF SIGN
        MOV     3,3,SNR ;TEST TO SEE IF ZERO
        JMP     SET       ;ZERO NO SIGN
        NEG     3,3       ;DECREMENT AC1
        COM     3,3
        MOVZR#  3,3,SEZ
        JMP     SITS      ;SIGN TRAILING SEPARATE TYPE FOUR
        MOV     3,3,SNR
        JMP     SILI      ;SIGN LEADING INCLUDED TYPE ONE
        MOVZR#  3,3,SZR
        JMP     SITI      ;SIGN TRAILING INCLUDED TYPE THREE
        JMP     SILS      ;SIGN LEADING SEPARATE TYPE TWO
SITS:   LDA     3,NEGNO,2         ;TO TEST TO SEE IF NEGATIVE NUMBER
        MOV     3,3,SNR
        JMP     .+7       ;ZERO PLUS NUMBER
        LDA     1,MINUS,2         ;GET A MINUS SIGN
        LDA     3,OPDP,2          ;GET BYTE POINTER
        STA     3,BYPTR,2         ;STORE IN BYTE  POINTER
        JSR     @PUT      ;STORE IT
        JSR     @RTER,2 ;ERROR RETURN
        DSZ     OPDP,2
        DSZ     CNT2,2
        JMP     SET
        JSR     @RTER,2
SILI:   SUBZL   3,3       ;GENERATE A ONE
        STA     3,EFLAG,2         ;STORE IN EFLAG
        LDA     3,NEGNO,2         ;TEST TO SEE IF NEGATIVE NUMBER
        MOV     3,3,SNR
        JMP     SET
        LDA     0,C31,2 ;GET AN EDIT CONVERSION CHARACTER
```

```
        ADD     0,1         ;CONVERT TO A LETTER
        STA     1,CHAR1,2           ;STORE IT
        JMP     SET
SITI:   LDA     3,NEGNO,2           ;TEST TO SEE IF NEGATIVE NUMBER
        MOV     3,3,SNR
        JMP     SET
        LDA     0,C31,2 ;GET A CONVERSION CHARACTER
        ADD     0,1         ;CONVERT TO LETTER
        STA     1,CHAR1,2           ;STORE IT
        JMP     SET
SILS:   LDA     3,NEGNO,2           ;TEST TO SEE IF NEGATIVE NUMBER
        MOV     3,3,SNR
        JMP     .+6
        LDA     0,OP2DP,2           ;GET BYTE POINTER
        STA     0,BYPTR,2           ;STORE IT IN BYTE POINTER
        LDA     1,MINUS,2           ;GET A MINUS SIGN
        JSR     @PUT        ;STORE IT
        JSR     @RTER,2
        DSZ     OPDL,2  ;DECREMENT TOTAL LENGTH
        JMP     SET
        JSR     @RTER,2
```

```
        .TITL   VALID
        .ENT    VALID
        .NREL
;F  TO
;  R
;   O    C0      91      X2      G3      9E4     XE5     Z6      B7
;    M
;C0:     3       4       0       0       0       0       0       0
;91:     5       6       1       1       7       2       0       0
;X2:     0       6       1       1       1       2       0       0
;G3:     0       1       1       1       1       1       0       0
;9E4:    0       0       1       0       0       1       0       0
;XE5:    0       0       1       0       0       1       0       0
;Z6:     0       6       1       1       0       0       0       0
;B7:     0       0       1       1       0       0       0       0
;
;
VALID:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     3,OP1DT,2       ;GET TYPE OF SENDING FIELD
        LDA     1,EMSK3,2
        AND     1,3
        LDA     0,THREE,2
        STA     0,X3T,2
        MOVZR   3,3
        DSZ     X3T,2
        JMP     .-2
        STA     3,SIGN,2        ;STORE SIGN
        LDA     3,OP1DT,2       ;GET TYPE
        LDA     1,EMSK3,2         ;GET MASK 7
        AND     1,3             ;GET TYPE
        LDA     1,TABLE         ;GET TABLE DISPLACEMENT
        ADD     1,3
        LDA     3,0,3
        STA     3,TAD,2
        LDA     3,OP2DT,2       ;GET TYPE OF RECEIVING FIELD
        LDA     0,TAD,2         ;GET DISPLACEMENT
        ADD     0,3             ;GET POSITION IN TABLE
        LDA     1,TABEL         ;GET BEGINNING OF TABLE
        ADD     1,3
        LDA     3,0,3
        STA     3,INDIC,2
        MOV     3,3,SZR         ;SEE IF INDICATOR IS ZERO
        JMP     @RTN,2        ;NO IT IS NOT. RETURN
        JMP     @RTER,2         ;YES RUN TIME ERROR
TABLE:  TAB
TAB:    0
        10
        20
        30
        40
        50
        60
        70
TABEL:  TAC
TAC:    3
        4
        0
        0
        0
        0
```

0
0
5
6
1
1
7
2
0
0
0
6
1
1
1
1
0
0
0
1
1
1
1
1
0
0
0
0
0
1
0
0
1
0
0
0
0
0
1
0
0
1
0
0
0
0
6
1
1
0
0
0
0
0
0
0
1
1
0
0
0
0

```
        .TITL   WRITE
        .ENT    WRIT
        .EXTD   PUT
        .NREL
WRIT:   ISZ     RTST,2          ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,OPFLG,2       ;LOAD OPCODE FLAG
        MOVZR#  1,1,SEZ         ;FIND OUT CONTENTS (0,1,2, OR 3)
        JMP     WRIT7           ;3, WRITE AFTER PAGE
        MOV     1,1,SNR
        JMP     WRIT3           ;0, WRITE BEFORE LINES
        MOVZR#  1,1,SZR
        JMP     WRIT4           ;2, WRITE BEFORE PAGE
        JMP     WRIT6           ;1, WRITE AFTER LINES
WRIT2:  LDA     0,OP1DP,2
        LDA     1,OP1DL,2
        LDA     2,CHANL,2
        .SYSTEM
        .WRS    77
        JSR     @RTER,3
        MOV     3,2
        JSR     @RTN,2
WRIT3:  SUBZL   1,1     ;GENERATE A ONE
        STA     1,BEFRL,2       ;SET BEFORE LINE FLAG
        STA     1,SKPIT,2       ;SET SKIP IT FLAG
        LDA     3,QP,2
        LDA     0,3,3
        MOV     0,0,SNR
        JMP     WRIT2
        STA     0,CNT1,2
        JMP     WRIT5
WRIT4:  SUBZL   1,1             ;GENERATE A ONE
        STA     1,BEFRP,2       ;SET BEFORE PAGE FLAG
        STA     1,SKPIT,2       ;SET SKIP IT FLAG
WRIT5:  LDA     0,OP1DP,2    ;LOAD THE BYTE POINTER
        LDA     1,OP1DL,2
        LDA     2,CHANL,2    ;LOAD THE CHANNEL NUMBER
        .SYSTEM
        .WRS    77
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        LDA     1,BEFRP,2    ;LOAD BEFORE PAGE FLAG
        MOVR     1,1,SZC      ;CHECK STATUS
        JMP     WRIT7        ;ON, JUMP TO WRIT6
        SUBZL   1,1
        STA     1,SKPIT,2
        JMP     WRIT6+10
WRIT6:  LDA     3,QP,2
        LDA     0,3,3
        STA     0,CNT1,2
        MOV#    0,0,SNR ;CHECK TO SEE IF ZERO
        JMP     WRIT2   ;ZERO WRITE WITHOUR ADVANCING
        SUBZL   1,1     ;GENERATE A ONE
        SUB#    1,0,SNR ;CHECK TO SEE IF EQUAL
        JMP     WRIT5   ;EQUAL TO ONE
        LDA     0,LF1           ;LOAD LINE FEED CHAR.
        LDA     2,CHANL,2    ;LOAD THE CHANNEL NUMBER
        .SYSTEM
        .WRL    77
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
```

```
        DSZ     CNT1,2          ;DECREMENT # LINES TO ADVANCE
        JMP     .-6              ;NOT ZERO, ADVANCE AGAIN
        LDA     1,SKPIT,2       ;LOAD SKIPIT FLAG
        MOV     1,1,SZR         ;CHECK STATUS
        JMP     @RTN,2          ;ON, PROCESSING COMPLETE, RETURN
        JMP     WRIT5            ;OFF, JUMP TO WRITE DATA LINE
WRIT7:      LDA     0,FF1           ;LOAD FORM FEED CHAR.
        LDA     2,CHANL,2       ;LOAD THE CHANNEL NUMBER
        .SYSTEM
        .WRL    77
        JSR     @RTER,3
        MOV     3,2     ;RESTORE AC2
        LDA     1,SKPIT,2       ;LOAD SKIPIT FLAG
        MOV     1,1,SZR         ;CHECK STATUS
        JMP     @RTN,2          ;ON,PROCESSING COMPLETE, RETURN
        JMP     WRIT5            ;OFF, JUMP TO WRITE DATA LINE
        .TXTM   0
LF1:    .+1*2
        .TXT    /<12><15>/
FF1:    .+1*2
        .TXT    /<14><15>/
```

```
        .TITL   ZEROOUT
        .ENT    ZEROA
        .NREL
;CALL TO THIS ROUTINE IS (N,PACZ)
ZEROA:  ISZ     RTST,2  ;GET NEXT STORE POSITION
        STA     3,@RTST,2       ;SAVE RETURN ADDRESS
        LDA     1,N,2           ;GET NUMBER OF WORDS TO
        STA     1,NUM,2         ;TO ZERO-OUT STORE IN NUM
AA:     SUBZ    0,0             ;CLEAR AC0
        STA     0,@PACZ,2       ;STORE ZERO-OUT
        DSZ     NUM,2           ;DECREMENT COUNTER
        JMP     .+2             ;COUNTER NOT ZERO
        JMP     @RTN,2        ;RETURN
        ISZ     PACZ,2          ;GET NEXT ZERO-OUT POSITION
        JMP     AA              ;DO AGAIN
```

```
        .TITL   ZEROSUPPRESSION
        .ENT    GETBT,COMMA,ZEE,PUCAR,DOLAR,AA,PUTCH,GETBY,PUTT,ASTK,PTT,PUTTT

        .EXTD   PUT,GET
        .EXTN   B1,SNG,AAB
        .NREL
ASTK:   SUB     1,1
        STA     1,NFLAG,2       ;CLEAR NFLAG
        LDA     1,RCNT,2        ;GET REPITITION COUNT
        LDA     1,ONE,2 ;GET A ONE
        SUB#    0,1,SNR ;TEST TO SEE IF EQUAL TO ONE
        JMP     ONAST   ;EQUAL TO ONE
        LDA     0,PNT0,2        ;GET COUNT FOR ASTERISK FILL
        MOV#    0,0,SNR ;TEST TO SEE IF ZERO
        JMP     .+2
        JMP     .+4
        LDA     1,AST,2 ;GET AN ASTERISK
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        JMP     PUTT+3  ;ZERO GO TO GET BYTE
        LDA     1,AST,2 ;GET AN ASTERISK
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        STA     1,ECHT,2        ;STORE IN EDIT CHARACTER TEMPORARY
        JMP     PUTT
ONAST:  LDA     1,AST,2 ;GET A ASTERISK
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     PUTTT   ;GO TO PUT CHARACTER ROUTINE
        JMP     @RTN,2  ;RUNTIME ERROR
PUTT:   DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @RTER,2 ;ZERO SOMETHING WRONG
        DSZ     RCNT,2
        JMP     .+2
        JMP     @.B1
PUTTT:  LDA     1,ECHAR,2       ;GET EDIT CHARACTER
        LDA     3,PNT2,2        ;GET BYTE POINTER TO RECEIVING
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE BYTE ROUTINE
        JMP     @RTER,2
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        DSZ     CNT0,2  ;DECREMENT FILL COUNTER
        JMP     .+6
        LDA     1,ECHT,2        ;GET EDIT CHARACTER TEMPORARY
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        SUB     1,1     ;CLEAR AC1
        STA     1,PNT0,2
        JMP     AA
        LDA     1,CNT0,2
        MOVL#   1,1,SZC ;CHECK TO SEE IF NEGATIVE
        JMP     .-3     ;CARRY IS SET
        JMP     PUTT    ;GET ANOTHER ASTERISK
GETBY:  DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @RTN,2
        JSR     @SNG1   ;GET A BYTE POINTER
        JMP     @RTER,2
        LDA     0,C60,2 ;GET AN ASCII ZERO
        SUB#    0,1,SZR ;TEST TO SEE IF A ZERO
        JMP     PTT     ;GO TO A PUTCHARACTER ROUTINE
        LDA     1,ECHAR,2       ;GET EDIT CHARACTER
        LDA     3,C1,2          ;TEST TO SEE IF FILL CHARACTER
```

```
        SUB#    3,1,SNR
        JMP     PUTCH+3
        LDA     3,PNT2,2        ;GET STORE POSITON
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE IT
        JMP     @RTER,2
        ISZ     PNT1,2  ;GET NEXT PIECE OF DATA
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        DSZ     RCNT,2  ;DECREMENT REPITITION COUNT
        JMP     .+2     ;NOT ZERO
        JMP     @.B1    ;GET ANOTHER EDIT FORMAT
        DSZ     CNT1,2  ;DECREMENT BEGIN LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @.AAB   ;GET ANOTHER EDIT CODE
        JMP     GETBY   ;GET ANOTHER BYTE
.AAB:   AAB
.B1:    B1
SNG1:   SNG
PUTCH:  DSZ     CNT2,2  ;DECREMENT TOTAL LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @RTN,2  ;ZERO SOMETHING WRONG
        DSZ     RCNT,2  ;DECREMENT REPITITION COUNT
        JMP     .+2
        JMP     @.B1    ;GET ANOTHER EDIT CHARACTER
PTT:    LDA     3,PNT1,2        ;GET DATA
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET    ;GO TO GET A BYTE ROUTINE
        JMP     @RTER,2
        ISZ     PNT1,2
        LDA     3,PNT2,2        ;GET STORE POSTION
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;GET NEXT STORE POSTION
        JMP     @RTER,2
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        DSZ     CNT1,2  ;DECREMENT SENDING LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @.AAB   ;GET ANOTHER EDIT CODE
        JMP     PUTCH   ;GET ANOTHER CHARACTER
AA:     LDA     1,NFLAG,2       ;TEST TO SEE IF DOLLAR EDIT
        MOVR    1,1,SNC ;SEE IF SET
        JMP     GETBY   ;GO TO PUTCHARACTER OUT ROUTINE
        JMP     GETBT   ;DOLLAR EDIT
DOLAR:  LDA     1,RCNT,2        ;GET REPITITION COUNT
        LDA     0,ONE,2 ;TEST TO SEE IF EQUAL TO ONE
        SUB#    0,1,SNR
        JMP     ONDOL   ;EQUAL TO ONE
        LDA     0,PNT0,2        ;GET BYTE POINTER
        MOV#    0,0,SNR ;TEST TO SEE IF ZERO
        JMP     .+2
        JMP     .+4
        LDA     1,DOL,2 ;GET A DOLLAR SIGN
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        JMP     GETBT   ;GO TO GET BYTE ROUTINE
        LDA     1,BLK,2 ;GET A BLANK
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        LDA     1,DOL,2 ;GET A DOLLAR SIGN
        STA     1,ECHT,2        ;STORE IN EDIT CHARACTER TEMPORARY
        SUBZL   1,1     ;GENERATE A ONE
        STA     1,NFLAG,2       ;STORE IN NFLAG FOR DOLLAR EDIT FLAG
        JMP     PUTT    ;GO TO STORE ROUTINE
ONDOL:  LDA     1,DOL,2 ;GET A DOLLAR SIGN
```

```
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        STA     1,ECHT,2        ;STORE IN EDIT CHARACTER TEMPORARY
        JMP     PUTTT
PUCAR:  DSZ     PNT2,2  ;GET LAST STORE POSITION
        LDA     1,ECHAR,2       ;GET A DOLLAR SIGN
        LDA     3,PNT2,2        ;GET STORE POSITION
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE IT
        JMP     @RTER,2
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        JMP     PUTCH+1 ;GO TO PUTCHARACTER ROUTINE
ZEE:    LDA     0,PNT0,2        ;GET BYTE POINTER TO ARGUMENT ONE
        MOV#    0,0,SNR ;TEST TO SEE IF ZERO
        JMP     .+2
        JMP     .+4
        LDA     1,BLK,2 ;GET THE EDIT CODE
        STA     1,ECHAR,2       ;STORE THE CHARACTER
        JMP     GETBY+1
        LDA     1,BLK,2 ;GET A BLANK
        STA     1,ECHAR,2       ;STORE IN EDIT CHACTER
        STA     1,ECHT,2        ;STORE IN EDIT CHARACTER TEMPORARY
        JMP     PUTT    ;GO TO A PUT CHARACTER ROUTINE
COMMA:  DSZ     PNT2,2  ;DECREMENT STORE POSITION
        LDA     3,PNT2,2        ;GET BYTE POINTER TO LAST STORE POSITION
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @GET    ;GET LAST BYTE STORED
        JMP     @RTER,2
        ISZ     PNT2,2  ;PUT BACK TO PROPER STORE POSITION
        LDA     0,C60,2 ;GET A MASK OF 60
        AND     0,1     ;TEST TO SEE IF NUMERIC
        SUB#    0,1,SNR
        JMP     .+10    ;A NUMBER
        LDA     3,PNT2,2        ;GET STORE POSITION
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE LAST CODE OUT
        JMP     @RTER,2
        DSZ     CNT2,2  ;DECREMENT LENGTH
        JMP     @.B1    ;RETURN TO GET ANOTHER EDIT CODE
        JMP     @RTN,2  ;RUNTIME ERROR
        LDA     1,COL,2 ;GET A COMMA
        STA     1,ECHAR,2       ;STORE IN EDIT CHARACTER
        JMP     PUTT1   ;PUT THE CHARACTER OUT
GETBT:  DSZ     CNT2,2  ;DECREMET TOTAL LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @RTN,2  ;ZERO SOMETHING WRONG
        JSR     @SNG1   ;GET A BYTE POINTER
        JMP     @RTER,2
        LDA     0,C60,2 ;GET AN ASCII ZERO
        SUB#    0,1,SZR ;TEST TO SEE IF A ZERO
        JMP     PUCAR   ;GO TO A PUTCHARACTER ROUTINE
        LDA     1,BLK,2 ;GET A BLANK
        LDA     3,PNT2,2        ;GET STORE POSITION
        STA     3,BYPTR,2       ;STORE IN BYTE POINTER
        JSR     @PUT    ;STORE IT
        JMP     @RTER,2
        ISZ     PNT1,2  ;GET NEXT PIECE OF DATA
        ISZ     PNT2,2  ;GET NEXT STORE POSITION
        DSZ     RCNT,2  ;DECREMENT REPITITION COUNT
        JMP     .+3     ;NOT ZERO
        ISZ     RCNT,2  ;PUT A ONE INTO REPITITION COUNT
        JMP     @PUTT1  ;GO TO STORE ROUTINE
```

```
        DSZ     CNT1,2  ;DECREMENT BEFIN LENGTH
        JMP     .+2     ;NOT ZERO
        JMP     @.AAB   ;GET ANOTHER EDIT CODE
        JMP     GETBT   ;GET ANOTHER BYTE
PUTT1:  PUTTT
```

APPENDIX D

INSTRUCTION REGISTER AND MICRO-PROCESSOR

Introduction

The Model 80 Micro-processor expects all of the instructions it will emulate to be in the format of the Instruction Register. The micro-decoder will then inspect the Instruction Register (IR) and generate micro-code based upon the operands found to be present.

Instruction Register

The format of the Instruction Register is presented below:

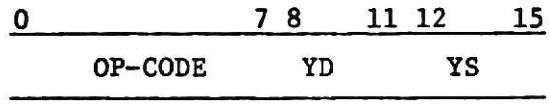


Figure D.1

Instruction Register

With respect to figure D.1, specifying YD as an A operand causes the YD field (IR 8:11) to be placed on bits 12:15 of the A bus. Bits 0:11 of the A bus will be set to zeroes. (See Figure D.2 - Model 80 Block Processor).[6]

Specifying YSI in the B field will cause the YS field (IR 12:15) to be placed on bits 12:15 of the B bus. Bits 0:11 of the B bus will be set to zeroes.

Specifying the keyword "NULL" in the A or B field will cause the corresponding operand bus to be set to zeroes. If NULL is specified in the S field, the S bus data will be loaded into MR7. Specifying YDP1 will cause the odd member of the even-odd pair of General Registers, one of whose number is in the YD field, to be selected.

Specifying YDI in the S field will cause bits 12:15 of the B bus to be loaded into the YD field of the IR.

The Micro-processor

When the INTERDATA emulates the instruction of a given machine, the micro-program will read the next user instruction which is to be performed from main memory. The instruction decoding circuit will then direct the processor to a uniquely identifiable micro-subroutine that is designed to perform the instruction. This subroutine may consist of from one to several micro-instructions which may be necessary to accomplish the given task. When emulation of the user instruction is completed, the micro-program will then read the next user instruction from main memory, closing the emulation loop. Because of the rather elementary level of the micro-code, the actual instructions will be directed to various parts of the hardware and registers (see figure D.2). The names for these specific areas tend to be long and unweildly and a series of abbreviations will be used in their place as defined below:

LOC - Location Counter. This contains the address of the Users' Instruction.

MAR - Memory Address Register. Contains the contents of LOC, i. e., the address of the Users' Instruction.

IR - Instruction Register. Contains the contents of memory specified by MAR.

MDR - Memory Data Register. The next sequential halfword (16 bits) in main memory.

Table D.1

Operand Abbreviations

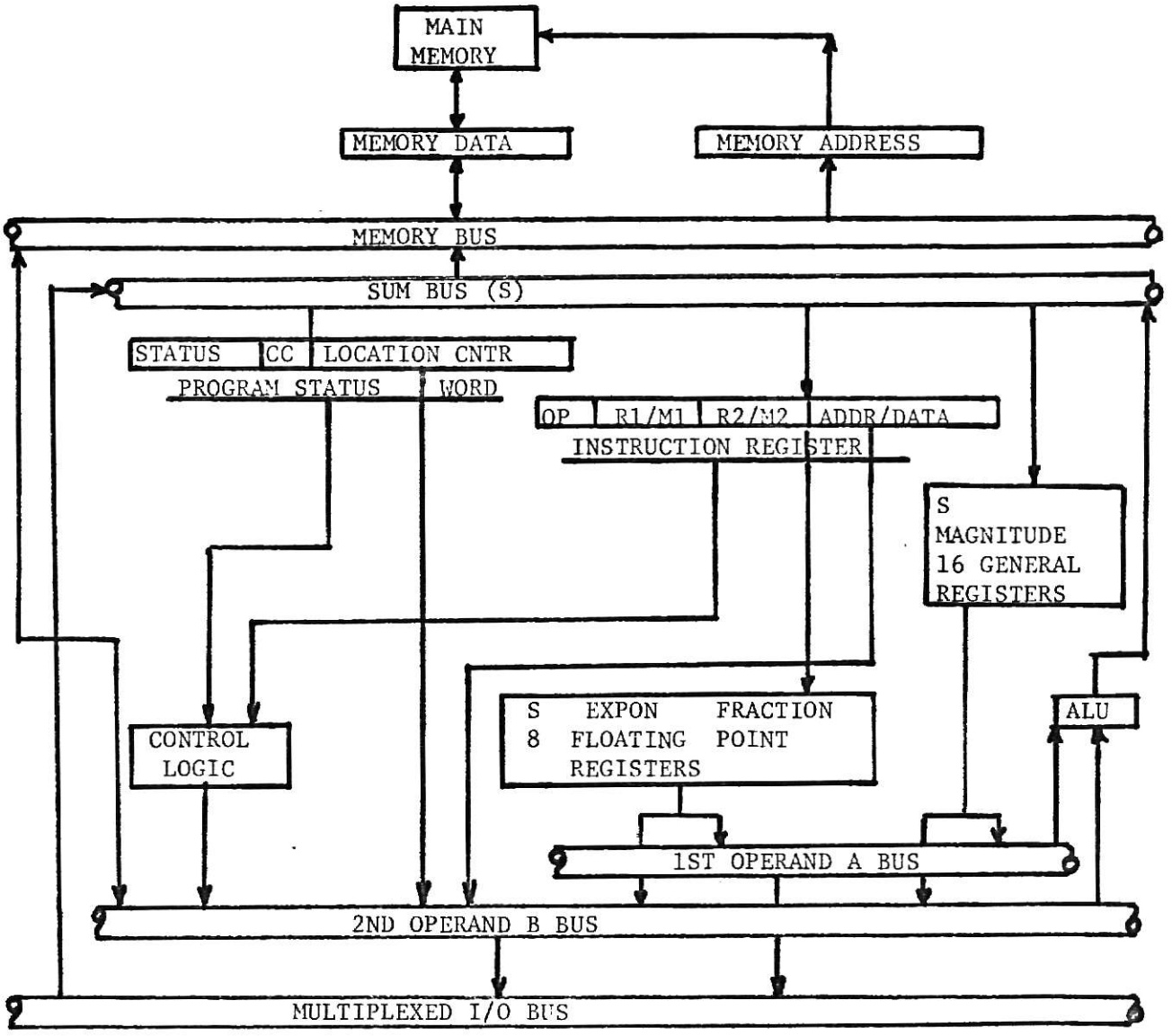


Figure D.2

Model 80 Processor Block Diagram

YD - IR (8:11)

YS - IR (12:15)

YX - Constant used the same as YS when YS is equal to zeroes.

Table D.1 (cont.)

Operand Abbreviations

When referencing the instruction word fields the following abbreviations are used:

A - Selects register to be used as first operand.

B - Selects register to be used as second operand.

S - Selects register to receive the result.

F - Specifies function of addressed module.

E - Enables the setting of the Condition Code

C - If set, transfer is conditional.

X - Execute.

I - Second operand is address of data.

D - Decode next user instruction.

K - F field extension.

Table D.2

Abbreviated Instruction Word Fields

There are in addition to the sixteen general registers abbreviated by the letters "GRn" where n is the digit 0-15, eight micro-registers which are abbreviated by the letters "MRn" where n is the digit 0-7. Both general registers and micro-registers are addressable by the micro-program.

When referencing the Memory Control field, the following abbreviations are used:

IR - Instruction Read

DR - Data Read

DW - Data Write

I2 - Increment MAR by two

I4 - Increment MAR and LOC by four

JAM - Copy LOC to MAR

Table D.3

Memory Control Abbreviations

Micro-instruction Formats

Model 80 INTERDATA micro-instructions can be presented in one of six formats designated as Address Link, Register Link, Register to Register Transfer, Register to Register Control, Register to Register Immediate, and Register Write. The formats are shown below.

ADDRESS LINK

0	2	3	4	5	6	7	10	11	15	16	19	20	30	31
000	I	X	E	D	MC		LINK REGISTER		F		ADDRESS			*

REGISTER LINK

0	2	3	4	5	6	7	10	11	15	16	19	20	24	25	31
000	0	X	E	D	MC		LINK		F		B		ADDRESS MASK		

REGISTER TO REGISTER TRANSFER

0	2	3	4	5	6	10	11	15	16	19	20	24	25	30	31
MOD	00	I		S		A		F		B		PAGE ADDRESS		C	

REGISTER TO REGISTER CONTROL

0	2	3	4	5	6	10	11	15	16	19	20	24	25	26	27	28	31
MOD	01	I		S		A		F		B		K	E	D		MC	

REGISTER TO REGISTER IMMEDIATE

0	2	3	4	5	6	10	11	15	16	19	20	31
MOD	10	I		S		A		F		DATA		

REGISTER WRITE

0	2	3	5	6	10	11	15	16	19	20	24	25	26	27	28	31
111	111		NULL		A		F		B		K	E	D		MC	

Figure D.3

Micro-instruction Formats

If the C bit on the Register to Register Transfer is set, then transfer only occurs if no predefined signal is returned from the addressed module. For the ALU, the signal is Carry, which means no transfer will occur if a carry is generated.

The X bit in the Address Link and Register Link instructions distinguishes Execute and Link instructions from the Branch and Link instructions.

If the I bit is set, then the operand developed on the B bus is taken as a Control Store address. The halfword contents of the addressed Control Store location are placed on the B bus and the instruction continues with the indirect data.

The D bit enables the Privilege/Illegal ROM and the Op-code-to-Address Translator. Unless a branch is taken or an interrupt occurs, the next instruction is taken from the address dictated by the Op-code-to-Address Translator.

The K bit is used as an extension of the F field allowing more than 16 functions to be performed by the addressed module.

The Memory Control (MC) field controls memory access and MAR and LOC activities. The Read Only Location Center (RLC) contains the address of the next micro-instruction to be fetched and executed.

An explanation of each of the basic instruction used in the micro-coding of this project is presented below in a diagrammatic form.

ADD

A, AI: (S)←---(A) + Be

then: (RLC 10:15)←---PAGE ADDRESS if C = 0 or Carry = 0

(RLC 4:15)←---(RLC 4:15) + 1 if C = 1 or Carry = 1

Figure D.4

Diagrammatic Instruction Flow

SUBTRACT

S, SI: (S)←---(A) - Be

then: (RLC 10:15)←---PAGE ADDRESS if C = 0 or Carry = 0

(RLC 4:15)←---(RLC 4:15) + 1 if C = 1 or Carry = 1

AND

N, NI: (S)←---(A) AND Be

then: (RLC 10:15)←---PAGE ADDRESS

EXCLUSIVE OR

X, XI: (S)←---(A) XOR Be

then: (RLC 10:15)←---PAGE ADDRESS

BRANCH and LINK

Tested Condition True:

(LINK)←---(RLC 4:15) + 1

(RLC 5:15)←---ADDRESS 20:30 (RLC 4)←---ADDRESS 31 (ADDRESS LINK)

(RLC) 4:15)←---(B) and MASK (REGISTER LINK)

Tested Condition False:

(LINK)←---(RLC 4:15) + 1

(RLC 4:15)←---(RLC 4:15) + 1

OR

O, OI: (S)←---(A) OR Be

then: (RLC 10:15)←---PAGE ADDRESS

LOAD

L, LI: (S)←---BE

LX: (S)←---Be

then: (RLC 10:15)←---PAGE ADDRESS

Figure D.4 (Cont.)

Diagrammatic Instruction Flow

EXECUTE and LINK

Tested Condition True:

(LINK)←(RLC) 4:15) + 1

do instruction at ADDRESS (ADDRESS LINK)

do instruction at (B) and MASK (REGISTER LINK)

(RLC 4:15)←---(RLC 4:15) + 1

Tested Condition False:

(LINK)←---(RLC 4:15) + 1

(RLC 4:15)←---(RLC 4:15) + 1

Figure D.4 (Cont.)

Diagrammatic Instruction Flow

APPENDIX E

ASSEMBLER OP-CODES AND MICRO-CODED EMULATION

Shift Left Logical

Assembler: SLLS R1,N

Description: The content of the first operand is shifted left the number of positions specified by the second operand. High order bits shifted out of position 0 are shifted through the Carry Bit of the PSW (PSW12) and then lost. Zeroes are shifted into the low order bit position.

Micro-code: SLLS SL YD,YD,YS1,IR2,E,D

Description: See figure D.4

Shift Right Logical

Assembler: SRLS R1,N

Description: The content of the first operand is shifted right the number of bit positions specified by the second operand. Low order bits shifted out of position 15 are shifted through the Carry Bit of the PSW (PSW12) and then lost. Zeroes are shifted into position 0.

Micro-code: SRLS SR YD,YD,YS1,IR2,E,D

Description: See figure D.4

Load Halfword

Assembler: LHR R1,R2

LHI R1,Be

LH R1,Be

Description: The second operand is loaded into the General Register specified by R1.

Micro-code: LH A MAR,MDR,YX,DR4

O YD,NULL,MDR,IRJ,E,D

LHI A MRO,MDR,YX,IR4

O YD,NULL,MRO,D,E

LHR 0 YD,NULL,YS,IR2,E,D

Description: See figure D.4

Store Halfword

Assembler: STH R1,Be

Description: The 16-bit first operand is stored in the memory location specified by the second operand. The first operand is unchanged.

Micro-code: STH A MAR,MDR,YX,I4

A YD,LOC

L LOC,MRO,IRJ,D

Description: See figure D.4

Add Halfword

Assembler: AHR R1,R2

AHI R1,Be

AIS R1,N

Description: The second operand is added algebraically to the contents of the General Register specified by R1. The Add Immediate Short (AIS) instruction, causes the four-bit second operand N to be added to the contents of the General Register specified by R1. The result replaces the contents of R1. The Add Halfword Immediate (AHI) adds the one-byte immediate second operand to the contents of the General Register as specified by the R1 first operand.

Micro-code: AHR A YD,YD,YS,IR2,E,D

AHI A MRO,MDR,YX,I4

A YD,YD,MRO,E,D

AIS A YD,YD,YSI,IR2,E,D

Description: See figure D.4

Subtract Halfword

Assembler: SHR R1,R2

SHI R1,Be

SIS R1,N

Description: The second operand is subtracted from the General Register specified by R1. The difference is contained in R1. The second operand is unchanged. The Subtract Immediate Short (SIS) instruction causes the fou-bit second operand N to be subtracted from the contents of the General Register specified by R1. The Subtract Halfword Immediate (SHI) instruction subtracts the immediate one-byte second operand from the General Register specified by R1.

Micro-code: SHR S YD,YD,YS,IR2,E,D

SHI A MRO,MDR,YX,IR4

S YD,YD,MRO,E,D

SIS S YD,YD,YS1,IR2,E,D

Description: See figure D.4

Compare Logical Halfword

Assembler: CLHR R1,R2

CLH R1,Be

CLHI R1,Be

Description: The first operand specified by R1 is compared logically to the 16-bit second operand. The result is indicated by the setting of the Condition Codes (PSW 12:15). Both operands remain unchanged.

Micro-code: CLHR S NUL,YD,YS,IR2,E,D

CLH A MAR,MDR,YX,DR4

CLH1 S NUL,YD,MDR,IRJ,E,D

```
CLHI   A MRO,MDR,YX,IR4
       S MRO,YD,MRO,D,E
NUL    EQU  '17'
```

Description: See figure D.4

Compare Halfword

Assembler: CHR R1,R2

Description: The first operand specified by R1 is compared to the 16-bit second operand. The comparison is algebraic, taking into account the sign and magnitude of each number. The result is indicated by the setting of the Condition Code (PSW 12:15). Both operands remain unchanged.

Micro-code:

```
CHR    BAL  CHR1(MR7),IR2
CHR1   A    MDR,NULL,YS
       X    MRO,YD,MDR
       BALNL CLH1(MR7)
       ASRI MRO,YD,1
       OI   MRO,MRO,MRO,IRJ,E,D
```

Description: See figure D.4

Exclusive OR Halfword

Assembler:

```
XHR    R1,R2
XHI    R1,Be
```

Description: The logical difference of the 16-bit second operand and the General Register specified by R1, replaces the contents of R1. The 16-bit difference is formed on a bit-by-bit basis.

Micro-code:

```
XHR   X  YD,YD,YS,IR2,E,D
XHI   A  MRO,MDR,YX,IR4
      X  YD,YD,MRO,E,D
```

Description: See figure D.4

AND Halfword

Assembler: NHR R1,R2

Description: The logical product of the 16-bit second operand and the content of the General Register specified by R1, replaces the contents of R1. The 16-bit product is formed on a bit-by-bit basis.

Micro-code:

```
NHR   N  YD,YD,YS,IR2,E,D
```

Description: See figure D.4

Branch and Link

Assembler:

```
BALR  R1,R2
BAL   R1,Be
```

Description: The address of the next sequential instruction is saved in the General Register specified by R1, and an unconditional branch is executed to the 16-bit address specified by the second operand.

Micro-code:

```
BALR  L  ADR,YS
      AI YD,LOC,2
      L  LOC,ADR
      A  MRO,MRO,NULL,IRJ,D
BAL   A  MRO,MDR,YX,I4
      L  YD,LOC
      L  LOC,MRO,IRJ,D
```

Description: See figure D.4

Extended Mnemonics

Extended mnemonics used in the assembly language in previous chapters will be translated into either a BTC or a BFC with the appropriate mask being substituted, i. e.,

```
BZ  Be   assembles BFC 3,Be
B   Be   assembles BTC 0,Be
BNC Be   assembles BFC 8,Be
BC  Be   assembles BTC 8,Be
BNE Be   assembles BFC 3,Be
BE  Be   assembles BFC 3,Be
BNZ Be   assembles BFC 3,Be
```

Description: The Condition code of the PSW is tested for the condition specified by the mask field. For the BFC, if all conditions tested for are found false, then the branch is executed to the 16-bit address specified (Be). For the BTC, he conditions are tested and if any of the conditions are found to be true, then the branch is executed to the 16-bit address specified (Be).[7]

```
Micro-code:    BTC   A  MRO,MDR,YX,I4
                     BALNF BRX(MR7),D
               BFC   A  MRO,MDR,YX,IR4
                     BALF BRX(MR7),D
               BRX   L  LOC,MRO,IRJ,D
```

Description: See figure D.4

THE DESIGN OF AN INTERDATA IMPLEMENTATION OF THE U. S. NAVY MINI-COBOL

by

TERRY WAYNE ANDERSON-ROVIA

B. S., LOYOLA UNIVERSITY (CHICAGO), 1964

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1976

This report describes the transfer of the U. S. Navy MINI-COBOL compiler from a NOVA to an INTERDATA mini-computer. The principles utilized in this report may be used to transfer the MINI-COBOL compiler to any mini-computer which uses a micro-coded instruction set with minor modification to the FORTRAN interpreter as designed by the Navy. Also presented is a description of the MODEL 80 INTERDATA microprocessor and the micro-instruction formats which should enable the reader to better understand the micro-code presented as emulated NOVA instructions. A description of the mnemonics used in the micro-code is given to facilitate the reading of the emulation code. Formats of the NOVA instructions which will be emulated are presented. Specifications of the Fetch and Decode modules found in the emulator are given along with the micro-code generated for each NOVA instruction. A sample algorithm is given including a step-by-step trace table of two typical NOVA instructions. Also included are suggestions for modifications of the MINI-COBOL interpreter, specifications of the MINI-COBOL language, a description of the compiler system, and listings of the source code for all modules used in the system.