SNAP SYMBOLIC LANGUAGE
FOR THE BURROUGHS 220
VERSION 1

BY

MICHAEL J. MAHON

5/10/64
TABLE OF CONTENTS

INTRODUCTION..........................II

I. GENERAL DESCRIPTION.................. 1

ii. BASIC FORMAT...........................3

iii. PSEUDO-OPERATIONS
   A. ASSEMBLER CONTROL..................... 6
   B. LOCATION COUNTER CONTROL............ 8
   C. SYMBOL DEFINING.......................12
   D. OPERATION-CODE DEFINING.............13
   E. DATA DEFINING.................15
   F. SUBROUTINE RELATED.................18
   G. LISTING CONTROL....................20

IV. DIAGNOSTIC MESSAGES
   A. SYMBOL RELATED.......................21
   B. OPERATION-CODE RELATED..............22
   C. MISCELLANEOUS.....................23

v. PROGRAM EXAMPLE.......................25

APPENDIX A (OP-TABLE).............28

INFORMATION REGARDING ERRORS, EITHER
IN THIS MANUAL OR IN 'SNAP', MAY BE SENT
TO THE AUTHOR DIRECTLY OR BY WAY OF THE
COMPUTING CENTER.
INTRODUCTION

COMPUTER PROGRAMS MAY, IN GENERAL, BE WRITTEN IN THREE DIFFERENT WAYS.

THE MOST DIRECT METHOD OF PROGRAMMING IS IN BASIC 'MACHINE LANGUAGE,' AND PROGRAMS WRITTEN IN THIS FORM ARE SUITABLE FOR IMMEDIATE EXECUTION BY THE COMPUTER.

IN THE SECOND METHOD OF PROGRAMMING, ONE WRITES IN AN 'ASSEMBLY LANGUAGE' IN WHICH MACHINE OPERATION CODES ARE REPRESENTED BY MNEMONIC NAMES, AND ADDRESSES IN INTERNAL STORAGE ARE REFERRED TO BY PROGRAMMER-ASSIGNED NAMES. ASSEMBLY LANGUAGE 'SOURCE' PROGRAMS ARE NOT SUITABLE FOR DIRECT EXECUTION, BUT MUST BE 'ASSEMBLED' INTO A MACHINE LANGUAGE 'OBJECT' PROGRAM PRIOR TO EXECUTION.

FINALLY, THE PROGRAMMER MAY WRITE IN A 'COMPILER LANGUAGE,' SUCH AS 'ALGOL' OR 'FORTRAN' IN WHICH THE PROGRAM TAKES THE FORM OF A MORE-OR-LESS 'READABLE' PROBLEM-SOLVING ALGORITHM. A COMPILER LANGUAGE PROGRAM MUST BE CONVERTED INTO MACHINE LANGUAGE BEFORE IT CAN BE EXECUTED, AND THIS MAY BE DONE IN EITHER OF TWO WAYS--THE 'COMPILER' MAY TRANSLATE DIRECTLY INTO MACHINE LANGUAGE, OR IT MAY PRODUCE AN ASSEMBLY LANGUAGE PROGRAM WHICH IS THEN ASSEMBLED INTO MACHINE LANGUAGE TEXT BY AN ASSEMBLER. THIS LATTER CASE IS THE MORE USUAL FOR SOPHISTICATED COMPILERS.

COMPILER LANGUAGE, BEING THE MOST PROBLEM-ORIENTED LANGUAGE, IS GENERALLY EASIEST TO LEARN, BUT ORDINARILY HAS MUCH LESS VERSATILITY THAN BASIC MACHINE LANGUAGE. MACHINE LANGUAGE, ON THE OTHER HAND, TENDS TO BE SOMEWHAT TEDIOUS AND ERROR-PRONE BECAUSE OF THE BASIC SIMILARITY OF THE MANY NUMERIC ADDRESSES AND OPERATION CODES WHICH ENTER INTO A PIECE OF CODING. SYMBOLIC ASSEMBLER LANGUAGE PROVIDES FULL ACCESS TO THE VERSATILITY OF MACHINE LANGUAGE, WHILE ALLOWING THE USE OF MEANINGFUL SYMBOLS TO DESIGNATE LOCATIONS IN THE INTERNAL STORAGE OF THE COMPUTER. A LONG PROGRAM MAY BE WRITTEN IN SECTIONS WITHOUT ANY SACRIFICE OF EFFICIENCY IN THE OBJECT PROGRAM, AND ERRORS OF MISKEPTEN ADDRESSES AND OPERATION CODES ARE DRastically REDUCED. IN ADDITION, PROVISION IS MADE FOR INCLUSION OF EXPLANATORY REMARKS WHICH BECOME AN INTEGRAL PART OF THE SYMBOLIC SOURCE PROGRAM, AND 'PSEUDO-OPERATION' CODES ARE PROVIDED TO HANDLE COMMON PROGRAMMING NEEDS, SUCH AS ARRAY DEFINITION, SUBROUTINE CALLING, AND LISTING CONTROL.

BECAUSE OF THE COMBINATION OF SIMPLICITY AND VERSATILITY OFFERED BY SYMBOLIC ASSEMBLER LANGUAGE, IT IS ALMOST UNIVERSALLY THE STANDARD TOOL OF THE SERIOUS PROGRAMMER.
I. GENERAL DESCRIPTION

'SNAP' ('SYMBOLIC-NUMERIC ASSEMBLY PROGRAM') IS A TWO-PASS SYMBOLIC ASSEMBLER FOR THE BURROUGHS 220 PAPER-TAPE DATA PROCESSING SYSTEM. OPERATION CODES AND ADDRESSES ARE SPECIFIED SYMBOLICALLY, WHILE SIGN AND VARIANT DIGITS (EXCEPT WHERE SPECIFIED BY THE OP-CODE) ARE WRITTEN IN NUMERIC FORM. SOURCE PROGRAMS IN 'SNAP' LANGUAGE ARE PREPARED ON CODING SHEETS, KEYPUNCHED ONTO IBM CARDS, AND FINALLY CONVERTED TO PAPER TAPE. THIS 'SOURCE TAPE' IS THE ORDINARY INPUT TO 'SNAP'.


THE OBJECT PROGRAM PRODUCED IS SUITABLE EITHER FOR IMMEDIATE EXECUTION OR FOR PHASE 0 INPUT TO THE NUMERIC ASSEMBLER IF THE PROGRAMMER HAS SO WRITTEN IT. OBJECT PROGRAMS MAY BE PREPARED EITHER IN ABSOLUTE MODE ('ABSMOD') OR RELATIVE MODE ('RELMOD') SUITABLE FOR LOADING AS A RELOCATABLE SUBPROGRAM.

A COMPLETE DIAGNOSTIC ROUTINE IS A STANDARD PART OF PASS I AND PASS II, THOUGH SENSE SWITCH OPTIONS MAY BE USED TO LIMIT ERROR TYPEOUT. PASS II ERRORS ARE 'FLAGGED' BRIEFLY ON THE LISTING, AND ALL ERRORS ARE NORMALLY TYPED OUT ON THE TELETYPE PRINTER, TOGETHER WITH THE STATEMENT WHICH CAUSED THE ERROR.

STANDARD UNIT DESIGNATIONS FOR OUTPUT DEVICES ARE AS FOLLOWS...

SPO = HIGH-SPEED PRINTER
PWR 0 = TELETYPE PRINTER
PWR 1 = HIGH-SPEED PUNCH

THE FOLLOWING SENSE SWITCH OPTIONS ARE PROVIDED...

1. SWITCH 5 CAUSES 'SNAP' TO TAKE ITS PASS I INPUT FROM MAGNETIC TAPE,
2. SWITCH 6 ON SUPPRESSES PUNCHING OF AN OBJECT TAPE,
3. SWITCH 7 ON ALLOWS MANUAL DEFINITION OF UNDEFINED SYMROLIC EXPRESSIONS,
4. SWITCH 8 ON SUPPRESSES ERROR TYPEOUT,
5. SWITCH 9 ON SUPPRESSES PRINTING OF THE SYMBOL DICTIONARY,
6. SWITCH 0 ON SUPPRESSES ALL USE OF MAGNETIC TAPE (PAPER-TAPE-ONLY MODE),
7. SWITCH 2 ON (OFF) RESTARTS PASS II (PASS I).
DURING BOTH PASSES, 'SNAP' MAINTAINS A CELL IN STORAGE REFERRED TO AS THE
'LOCATION COUNTER'. THIS CELL CONTAINS THE EXECUTION-TIME STORAGE ADDRESS OF
THE INSTRUCTION CURRENTLY BEING ASSEMBLED (RELATIVE TO THE LOAD ADDRESS IF
'RELMOD').

THE APPEARANCE OF A SYMBOL IN THE LABEL FIELD OF ANY MACHINE OPERATION AND
MOST PSEUDO-OPERATIONS CAUSES THAT SYMBOL TO BE DEFINED TO HAVE THE CURRENT
VALUE OF THE LOCATION COUNTER.

ALTHOUGH THE MOST COMMON '6' OR '7' SIGN DIGIT COMMANDS ARE GENERATED BY
THE TWO PSEUDO-OPERATIONS 'LOAD' AND 'TCD', THE PROGRAMMER MAY OCCASIONALLY
WISH TO SPECIFY THESE SIGN DIGITS IN OTHER INSTRUCTIONS. (OF COURSE, SUCH AN
INSTRUCTION INTERRUPTS NORMAL LOADING, AND IF FURTHER LOADING IS DESIRED IT MUST BE
RESTARTED MANUALLY OR BY PROGRAMMING.) 'SNAP' DOES NOT SENSE THAT A '6' OR
'7' SIGN INSTRUCTION HAS BEEN PRODUCED BY A MACHINE OP-CODE STATEMENT, AND SO
PROCEEDS TO DEFINE THE LABEL (IF ANY) AND INCREMENT THE LOCATION COUNTER. SINCE
SUCH A WORD DOES NOT ENTER STORAGE, THE CORRESPONDENCE BETWEEN LOAD ADDRESS AND
LOCATION COUNTER IS DISTURBED, AND MUST BE RESTORED OR TAKEN INTO ACCOUNT IF
LOADING IS TO BE RESTARTED.

NUMERIC ASSEMBLER ('ASSEMBLER II') PSEUDO-OPS (SEE APPENDIX A) ARE, IN MOST
CASES, HANDLED LIKE MACHINE OPERATIONS, EXCEPT THAT A SYMBOL IN THE LABEL FIELD
OF AN 'ASSEMBLER II' PSEUDO-OP WILL NOT BE DEFINED, AND THE LOCATION COUNTER
WILL NOT BE INCREMENTED. FOR THIS REASON, SUCH PSEUDO-OPS SHOULD NOT BE INTER-
SPERSED WITH MACHINE OPERATIONS.

'SNAP' IS CALLED INTO STORAGE (0000-1799) FROM MAGNETIC TAPE UNDER THE SPAC
CONTROL PROGRAM. IF CALL-IN OF 'SNAP' IS TO BE AUTOMATIC, APPROPRIATE MACHINE-
CODE WORDS MUST BE PUNCHED ONTO PAPER TAPE BEFORE THE SOURCE PROGRAM. IF MAG-
TAPE IS TO BE USED (THE NORMAL SITUATION), TAPE UNIT 1 MUST BE EDITED BEFORE
'SNAP' IS CALLED IN. THE TAPE MUST NOT BE BLOCKED, BUT MUST ALWAYS BE EDITED
WHEN 'SNAP' IS LOADED OR RELOADED (NOT BETWEEN 'BATCHED' ASSEMBLIES). WHEN
CONTROL IS FIRST PASSED TO 'SNAP', INITIALIZATION IS ACCOMPLISHED AND THE
TITLE AND DATE OF CREATION OF THE ASSEMBLER IS PRINTED OUT, FOLLOWED BY THE LIST
OF SENSE SWITCH OPTIONS. IF MAGNETIC TAPE UNIT 1 IS INOPERATIVE, OR IT IS NOT
DESIREABLE TO USE MAGNETIC TAPE FOR SOME REASON, SWITCH 0 SHOULD BE TURNED ON
EVEN BEFORE 'SNAP' IS CALLED INTO STORAGE. IF THIS IS NOT DONE, 'SNAP' WILL
WRITE THE OP-CODE TABLE ON MAG-TAPE UNIT 1 FOR RESETTING PURPOSES.

IF IT SHOULD BECOME NECESSARY TO RESTART 'SNAP' MANUALLY, THE FOLLOWING
PROCEDURE SHOULD BE USED...

1. TO RESTART PASS I, PUT SWITCH 2 UP, PRESS 'CLEAR', AND PRESS 'START'.
2. TO RESTART PASS II, PUT SWITCH 2 DOWN, PRESS 'CLEAR', AND PRESS 'START'.

1. TO RESTART PASS I, PUT SWITCH 2 UP, PRESS 'CLEAR', AND PRESS 'START'.
2. TO RESTART PASS II, PUT SWITCH 2 DOWN, PRESS 'CLEAR', AND PRESS 'START'.
II. BASIC FORMAT

'SNAP' LANGUAGE SYMBOLIC CARDS ARE PUNCHED IN THE FOLLOWING FORMAT--

COLUMN   .6       .11      .16     20.     .26                           55.
LABEL    VARIANT  OP-CODE     ADDRESS    COMMENTS

1) 'LABEL' (COL. 6-10) IS A LEFT-ADJUSTED STRING OF FROM ONE TO FIVE
NON-BLANK ALPHAMERIC CHARACTERS, AT LEAST ONE OF WHICH IS
NON-NUMERIC AND NONE OF WHICH IS AMONG THE FOLLOWING SEVEN....
+ (PLUS SIGN) - (MINUS SIGN) * (ASTERISK)
/ (SLASH)     $ (DOLLAR SIGN) , (COMMA)
' (APOSTROPHE)

FOR EXAMPLE-- ALPHA
           BETA
           X
           K46
           3M41
           .READ
           .NOP.
           3.14   (PERMISSIBLE, BUT NOT RECOMMENDED)

ARE ALL LEGAL SYMBOLS. ON THE OTHER HAND,

A*B
364
$XBAR
X AVG

ARE ILLEGAL SYMBOLS.

2) 'VARIANT' (COL. 11-15) IS A FIVE CHARACTER NUMERIC FIELD IN WHICH THE
NUMBER IN COL. 11 IS INTERPRETED AS THE SIGN DIGIT OF THE
INSTRUCTION AND THE DIGITS FROM COL. 12-15 ARE INTERPRETED AS
THE NUMERIC VARIANT OF THE INSTRUCTION. IN ALL CASES BLANKS
ARE INTERPRETED AS ZEROS. IF A NON-NUMERIC CHARACTER APPEARS
IN THIS FIELD, THE LEAST SIGNIFICANT DIGIT OF THE TWO-DIGIT
INTERNAL REPRESENTATION IS USED. NO ERROR IS DETECTED.

EXCEPTIONS--

IF THE OPERATION CODE UNIQUELY SPECIFIES THE LEAST
SIGNIFICANT DIGIT (4,1 DIGIT) OF THE VARIANT, THE ASSEMBLER
IGNORES THE CONTENTS OF COL. 15.

IF A SIGN OF 1 IS DESIRED, IT MAY BE PRODUCED BY A MINUS
SIGN (-) IN COL. 11. (ANY OTHER NON-NUMERIC CHARACTER IN THE
SIGN COLUMN WILL RESULT IN AN ASSUMED SIGN OF ZERO.)
FOR EXAMPLE-- VARIANT FIELD... ASSEMBLES AS...

-0410 1 0410XXAAAA
- 1 0000XXAAAA
3 3 0000XXAAAA
1000 0 1000XXAAAA
4423 0 4423XXAAAA
ABCDE 0 2345XXAAAA

(EACH OF THE ABOVE EXAMPLES ASSUMES THAT THE OP CODE DOES NOT SPECIFY THE LEAST SIGNIFICANT VARIANT DIGIT.)

3) 'OP-CODE' (COL. 16-20) IS A LEFT-ADJUSTED OPERATION-CODE MNEMONIC, AS DEFINED IN APPENDIX A, OR TWO LEFT-ADJUSTED DIGITS. OPERATION CODES MAY CONTAIN THE FOUR ARITHMETIC OPERATORS (+-*/) IF SO DEFINED BY THE PROGRAMMER.

FOR EXAMPLE-- CAD
09
HLT
37
LBC
EJECT
.ORG. (DEFINED BY PROGRAMMER)
STA-V (DEFINED BY PROGRAMMER)
IBB/2 (DEFINED BY PROGRAMMER)
GO TO (DEFINED BY PROGRAMMER)

ARE ALL LEGAL OPERATION CODES. NOTE THAT IN NUMERIC OP-CODES LESS THAN 10, THE LEADING ZERO IS SIGNIFICANT AND MUST BE PRESENT.

4) 'ADDRESS' (COL. 26 TO FIRST BLANK OR COL. 55) IS ANY EXPRESSION FORMED FROM LEGAL SYMBOLS, ACTUAL INTEGER NUMBERS, AND ARITHMETIC OPERATORS. EXPRESSIONS ARE EVALUATED FROM LEFT TO RIGHT USING WHOLE-WORD (10-DIGIT) INTEGER ARITHMETIC. MULTIPLICATIONS AND DIVISIONS ARE DONE FROM LEFT TO RIGHT BEFORE ADDITIONS AND SUBTRACTIONS, NO PARENTHESES ARE ALLOWED. FRACTIONAL RESULTS ARE TRUNCATED TO INTEGERS AND THE FINAL RESULT IS CONVERTED MODULO 10000 AND PLACED IN THE ADDRESS FIELD OF THE INSTRUCTION. THE * (ASTERISK) MEANS BOTH 'MULTIPLY' AND 'PRESENT VALUE OF LOCATION COUNTER' DEPENDING ON CONTEXT. ACTUAL INTEGERS APPEARING IN EXPRESSIONS WILL BE USED MODULO 10000. DIVISION BY ZERO WITHIN A TERM CAUSES THE VALUE OF THAT TERM TO BE SET TO ZERO. OTHER TERMS IN THE SAME EXPRESSION ARE NOT AFFECTED.

(IN SOME PSEUDO-OPERATIONS THE ADDRESS FIELD IS USED DIFFERENTLY. REFER TO INDIVIDUAL PSEUDO-OP DESCRIPTIONS.)
FOR EXAMPLE-- (ASSUME... ALPHA=2046, BETA=5, LOCATION COUNTER=25)

THE EXPRESSION... HAS THE VALUE...

\[
\begin{align*}
\text{ALPHA} & \quad 2046 \\
37 & \quad 0037 \\
-1 & \quad 9999 \\
-\text{ALPHA} & \quad 7954 \\
++2 & \quad 0027 \\
\text{ALPHA-BETA} & \quad 2041 \\
\text{BETA} \times 8-30 & \quad 0010 \\
2 \times \text{ALPHA} - 2 \times \text{BETA} & \quad 4082 \\
\text{ALPHA/BETA} & \quad 0409 \\
++\text{ALPHA/BETA} & \quad 0434 \\
***/2 & \quad 0312 \\
** & \quad 0000 \\
512000/1000 & \quad 0012 \quad \text{(NOT RECOMMENDED)} \\
51200 \times 10/1000 & \quad 0512 \\
\end{align*}
\]

ARE ALL LEGAL ADDRESS FIELD EXPRESSIONS,

5) 'COMMENTS' (FIRST BLANK AFTER COL. 26 TO COL. 55) IS ANY REMARK THE PROGRAMMER WISHES TO INSERT. AT LEAST ONE BLANK MUST OCCUR BETWEEN THE START OF THE VARIABLE FIELD (COL. 26) AND THE FIRST CHARACTER OF THE REMARK. REMARKS SHOULD NOT USE SPECIAL CHARACTERS OR MULTIPLE PUNCHES WHICH WILL INTERFERE WITH LISTING, NAMELY-- DOLLAR SIGN ($), COMMA (,), AND APOSTROPHE (*).

LIBERAL USE OF MEANINGFUL COMMENTS IS STRONGLY RECOMMENDED. IF MODIFICATIONS TO A PREVIOUSLY WRITTEN PROGRAM BECOME NECESSARY, COMMENTS ARE INVALUABLE.

CARD COLUMNS 1-5, 21-25, AND 55-80 ARE COMPLETELY IGNORED BY THE ASSEMBLER. COLUMNS 55-80 ARE A SUITABLE PLACE FOR DECK IDENTIFICATION AND SEQUENCING IF THE PROGRAMMER DESIRES.
III. PSEUDO-OPERATIONS

A. ASSEMBLER CONTROL PSEUDO-OPERATIONS

1) 'ABS' AND 'REL'

The pseudo-operations 'ABS' and 'REL' specify whether the source program is to be assembled in absolute mode ('ABSMOD') or relative mode ('RELMOD'). The object program produced by an 'ABSMOD' assembly is loaded into the absolute origin(s) specified in the source program. The object program produced by a 'RELMOD' assembly may be 'floated' into core storage with b-register address modification, in which case the addresses specified in the source program are relative to the loading origin (modulo 10000). The operations 'ABS' and 'REL' consist of...

1. Blanks in the label and variant fields,
2. 'ABS' or 'REL' in the op-code field,
3. Blanks or comments in the variable field.

'ABS' and 'REL' may occur any number of times in the source program, the mode of assembly being determined by the last preceding mode control card. If no mode control card appears, 'ABSMOD' is assumed.

It should be noted that some magnetic tape commands and several pseudo-ops are not allowed in a 'RELMOD' assembly. A list of these operation codes are provided below.

<table>
<thead>
<tr>
<th>MFC</th>
<th>MFS</th>
<th>SPAC</th>
<th>AUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRO</td>
<td>BRU</td>
<td>BRZ</td>
<td>CHA</td>
</tr>
<tr>
<td>DEL</td>
<td>INS</td>
<td>MSR</td>
<td>PSR</td>
</tr>
<tr>
<td>REF</td>
<td>SEC</td>
<td>ZZZ</td>
<td></td>
</tr>
</tbody>
</table>

2) 'END'

The 'END' pseudo-operation indicates to the assembler the physical end of the source program. The 'END' operation consists of...

1. Blanks in the label and variant fields,
2. 'END' in the op-code field,
3. Any alphabetic comment in the variable field. (No blank is necessary preceding comments.)

When the assembler encounters the 'END' card image in pass I, control is transferred to the interlude pass and then to pass II. When pass II encounters the 'END' statement, the symbol table is scanned for unreferenced symbols, and the assembly is terminated. If 'SNAP' is in the paper-tape-only mode (switch 0 on) in pass I, the 'END' statement will cause the computer to halt with P-T (0766 00 4740) in the C-register (to remind the operator to rewind the source tape) before proceeding to pass II.
3) 'PAUSE'

THE 'PAUSE' PSEUDO-OPERATION IS PROVIDED FOR CONVENIENCE IN CHANGING PAPER TAPE REELS IN A MULTI-REEL SOURCE PROGRAM. THE 'PAUSE' OPERATION CONSISTS OF...

1. BLANKS IN THE LABEL AND VARIANT FIELDS,
2. 'PAUSE' IN THE OP-CODE FIELD,
3. BLANKS IN THE VARIABLE FIELD.

WHEN A 'PAUSE' STATEMENT IS ENCOUNTERED DURING PASS I, THE ASSEMBLER HALTS WITH T-T (0474 00 4740) IN THE C-REGISTER. ASSEMBLY PROCEEDS NORMALLY UPON PRESSING 'START'. IF 'SNAP' IS IN THE PAPER-TAPE-ONLY MODE (SWITCH 0 ON), 'PAUSE' WILL CAUSE A SIMILAR HALT IN PASS II. THE 'PAUSE' STATEMENT DOES NOT APPEAR IN THE LISTING.

4) 'DUP'

THE 'DUP' PSEUDO-OPERATION IS NOT OPERATIVE INDEPENDENT OF THE MACRO-EXPANDER. AT PRESENT THE ONLY EFFECT OF THE 'DUP' IS TO DEFINE THE SYMBOL IN THE LABEL FIELD TO HAVE THE PRESENT VALUE OF THE LOCATION COUNTER.

5) 'TCD'

IN ACCORDANCE WITH COMMON USAGE ON CARD-Oriented MACHINES, THE PSEUDO-OPERATION 'TCD' (TRANSFER CARD) GENERATES A BRANCH-TO-PROGRAM INSTRUCTION. THE OPERATION 'TCD' CONSISTS OF...

1. BLANKS IN THE LABEL AND VARIANT FIELDS,
2. 'TCD' IN THE OP-CODE FIELD,
3. AN EXPRESSION REFERRING TO THE DESIRED ENTRY POINT IN THE PROGRAM.

THE CODING...

6 0000 30 AAAAA

IS GENERATED IN THE OBJECT PROGRAM, WHICH WILL TERMINATE LOADING AND TRANSFER CONTROL TO AAAAA, THE VALUE OF THE ADDRESS FIELD EXPRESSION.
B. LOCATION COUNTER CONTROL PSEUDO-OPERATIONS

1) 'LOAD'

   THE 'LOAD' PSEUDO-OPERATION ORIGINS THE LOCATION COUNTER AND GENERATES A '6 PRB' LOADING INSTRUCTION INTO THAT ORIGIN. THE 'LOAD' OPERATION CONSISTS OF...

   1. A SYMBOL OR BLANKS IN THE LABEL FIELD,
   2. BLANKS IN THE VARIANT FIELD,
   3. 'LOAD' IN THE OP-CODE FIELD,
   4. AN EXPRESSION (CONTAINING SYMBOLS DEFINED PHYSICALLY EARLIER IN THE SOURCE PROGRAM) IN THE ADDRESS FIELD.

   THE 'LOAD' OPERATION GENERATES THE INSTRUCTION...

   6 0000 04 AAAA

   IN AN 'ABSMOD' ASSEMBLY, OR...

   7 0008 04 AAAA

   IN A 'RELMOD' ASSEMBLY, WHERE AAAA IS THE VALUE OF THE ADDRESS FIELD EXPRESSION AND THE NEW VALUE OF THE LOCATION COUNTER. THE SYMBOL IN THE LABEL FIELD, IF ANY, IS DEFINED TO HAVE THE VALUE AAAA. 'LOAD' IS TYPICALLY THE FIRST STATEMENT OF ANY PROGRAM.

2) 'ORG'

   THE 'ORG' PSEUDO-OPERATION 'ORIGINS' THE VALUE OF THE LOCATION COUNTER WITHOUT GENERATING THE 'LOAD' (6 0000 04 AAAA) INSTRUCTION. THE 'ORG' OPERATION CONSISTS OF...

   1. A SYMBOL OR BLANKS IN THE LABEL FIELD,
   2. BLANKS IN THE VARIANT FIELD,
   3. 'ORG' IN THE OP-CODE FIELD,
   4. AN EXPRESSION (CONTAINING SYMBOLS DEFINED PHYSICALLY EARLIER IN THE SOURCE PROGRAM) IN THE ADDRESS FIELD.

   THE LOCATION COUNTER IS SET TO THE VALUE OF THE ADDRESS FIELD EXPRESSION, AND THE LABEL, IF ANY, IS DEFINED TO HAVE THAT VALUE.

   'ORG' IS ESPECIALLY USEFUL FOR SETTING UP REMOTE STORAGE BLOCKS AND FOR CODING INSTRUCTIONS WHICH ARE TO BE MOVED IN STORAGE PRIOR TO THEIR EXECUTION. FOR EXAMPLE, SUPPOSE THREE ARRAYS ARE TO BE DEFINED AT A REMOTE LOCATION IN STORAGE. THIS MAY BE ACCOMPLISHED EVEN IN THE MIDDLE OF A STRING OF IN-LINE CODING BY THE FOLLOWING SEQUENCE OF INSTRUCTIONS...
Suppose a program is desired which will dump a selected area of storage, generate a check sum, and punch a sum checking routine onto the tape to verify correct loading of the program when the dump is loaded. Say the sum checking routine will be loaded into 4985 when it is executed, but will reside at some other address in the dumping program. This may be coded as follows...

```
.SAVE    SET * SAVE LOCATION COUNTER
.BEGIN   ORG 4985 SET ORIGIN
  * SUM CHECKING LOOP TO BE PUNCHED OUT WITH DUMP
CKSUM   CSU .SUM. A-REG = -.SUM.
LENGT   SOR ** FILLED IN BY DUMP PROG
  0401DLB  +%1 B = LENGTH
HIADR-  ADD ** FILLED IN BY DUMP PROG
  1DBB  +%1 GO TIL FINISHED...
BOF     0+1 RESET OVFLOW
SOH     SET OVERFLOW HALT
START   BFA ** FILLED IN BY DUMP PROG
  9669Hlt 9669 NO GOOD. STOP
BUN    +%1 DEAD.....
.SUM.   DEC FILLED IN BY DUMP PROG
  * END OF SUM CHECKING ROUTINE...
.LGTH   SET  %-.BEGIN (LENGTH OF ROUTINE)
ORG .SAVE+.LGTH (RESTORE COUNTER)
```

3) 'BSS'

The 'BSS' ('BLOCK STARTED BY SYMBOL') pseudo-operation defines a block of storage. The name of the block refers to its first location. The 'BSS' operation consists of...

1. A SYMBOL OR BLANKS IN THE LABEL FIELD,
2. BLANKS IN THE VARIANT FIELD,
3. 'BSS' IN THE OP-CODE FIELD,
4. AN EXPRESSION (CONTAINING SYMBOLS DEFINED PHYSICALLY EARLIER IN THE SOURCE PROGRAM) IN THE ADDRESS FIELD.

```
LABEL  EQU       *
LOAD    *+LENGT
```

THE 'BSS' IS MOST USEFUL IN DEFINING ARRAYS TO BE REFERENCED BY B-REGISTER ADDRESS MODIFICATION WITHIN 'DBB' LOOPS, SINCE THE VALUE OF B-REGISTER MODIFIED ADDRESSES WITHIN SUCH LOOPS GOES FROM ACTUAL VALUE PLUS SOME POSITIVE MODIFICATION TO THE ACTUAL VALUE PLUS ZERO. THUS THE ARRAY IS REFERENCED SERIALLY FROM ITS HIGHEST LOCATION (=LABEL +LENGTH-1) TO ITS LOWEST LOCATION (=LABEL).

4) 'BES'

THE 'BES' ('BLOCK ENDED BY SYMBOL') PSEUDO-OPERATION DEFINES A BLOCK OF STORAGE. THE NAME OF THE BLOCK REFERENCES TO THE LAST LOCATION OF THE BLOCK PLUS ONE. THE 'BES' OPERATION CONSISTS OF...

1. A SYMBOL OR BLANKS IN THE LABEL FIELD,
2. BLANKS IN THE VARIANT FIELD,
3. 'BES' IN THE OP-CODE FIELD,
4. AN EXPRESSION (CONTAINING SYMBOLS DEFINED PHYSICALLY EARLIER IN THE SOURCE PROGRAM) IN THE ADDRESS FIELD.


```
LABEL  LOAD    *+LENGT
```

OR...

```
BSS   LENGT
LABEL  EQU    *
```

THE 'BES' IS MOST USEFUL IN DEFINING ARRAYS TO BE REFERENCED BY B-REGISTER ADDRESS MODIFICATION WITHIN 'IBB' LOOPS, SINCE THE VALUE OF B-REGISTER MODIFIED ADDRESSES WITHIN SUCH LOOPS GOES FROM ACTUAL VALUE PLUS SOME NEGATIVE (MODULO 10000) MODIFICATION TO ACTUAL VALUE PLUS 9999 (MINUS 1). THUS THE ARRAY IS REFERENCED SERIALLY FROM ITS LOWEST LOCATION (=LABEL-LENGTH) TO ITS HIGHEST LOCATION (=LABEL-1).
5) 'SEX'

THE 'SEX' PSEUDO-OPERATION IS INCORPORATED FOR USE WITH THE
NUMERIC ASSEMBLER. IT GENERATES A SECTION DEFINING WORD AND ORIGINS
THE LOCATION COUNTER AT AN EVEN HUNDRED LOCATION. THE 'SEX' OPERATION
CONSISTS OF...

1. BLANKS IN THE LABEL AND VARIANT FIELDS,
2. 'SEX' IN THE OP-CODE FIELD,
3. AN EXPRESSION (CONTAINING SYMBOLS DEFINED PHYSICALLY EARLIER
   IN THE SOURCE PROGRAM) IN THE ADDRESS FIELD.

THE LOCATION COUNTER IS ORIGINED TO 100 TIMES THE VALUE OF THE
ADDRESS FIELD EXPRESSION, MODULO 10000, AND THE WORD...

9 9999 99 9AAA

IS GENERATED, WHERE AAA IS THE VALUE OF THE ADDRESS FIELD EXPRESSION
TRUNCATED TO THREE DIGITS.
C. SYMBOL DEFINING PSEUDO-OPERATIONS

1) 'EQU'

   The 'EQU' pseudo-operation defines the symbol in the label field to be equal in value to the address field expression. The 'EQU' operation consists of...

   1. A symbol in the label field,
   2. blanks in the variant field,
   3. 'EQU' in the op-code field,
   4. an expression (containing symbols defined physically earlier in the source program) in the address field.

   A symbol defined by an 'EQU' is assumed to have a unique value, and any attempt at redefinition will be diagnosed as an error.

2) 'SET'

   The 'SET' pseudo-operation defines a 'set-symbol' to have the value of the address field expression. The 'SET' operation consists of...

   1. a symbol in the label field,
   2. blanks in the variant field,
   3. 'SET' in the op-code field,
   4. an expression (containing symbols defined physically earlier in the source program) in the address field.

   A symbol originally defined by a 'SET' operation is referred to as a 'set-symbol' and has the special property that it may be redefined as often as desired by subsequent 'SET' operations. The 'SET' operation may not be used to redefine any symbol which is already defined as an ordinary symbol, nor may any other operation be used to redefine a set-symbol. Any attempted violation of these rules will be diagnosed as an error and ignored. All 'SET' statements are evaluated in both passes of the assembler, and may therefore be used for setting assembly parameters, keeping tallies, saving values of the location counter, etc. Set-symbols are not flagged as multiply-defined in the symbol dictionary. A set-symbol is listed in the unreferenced symbol list if it is not referenced after its final definition.
C. OPERATION CODE DEFINING PSEUDO-OPERATIONS

1) 'OPSYN'

The 'OPSYN' pseudo-operation provides a means of renaming previously defined op-codes. The 'OPSYN' operation consists of...

1. An op-code mnemonic in the label field,
2. Blanks in the variant field,
3. 'OPSYN' in the op-code field,
4. A previously defined op-code mnemonic in the address field.

(COMMENTS MUST NOT OCCUR BEFORE COLUMN 31.)

The mnemonic appearing in the label field (which may also be the blank mnemonic, ordinarily synonymous with 'HLT') is defined to be synonymous with the mnemonic appearing in the address field. A mnemonic may consist of from one to five alphameric characters, excluding only the dollar sign ($), apostrophe ('), and comma (,).

It should be observed that once a 'standard' mnemonic has been redefined, the operation which it originally represented cannot be referenced by the remainder of the source program unless it has been 'saved' under some other name. For example, suppose 'ORG' is to be defined to be synonymous with 'LOAD'. This, in effect, destroys the 'ORG' operation unless 'ORG' is saved...

.ORG. OPSYN ORG
ORG OPSYN LOAD

Subsequent to these statements, the operation of 'ORG-ING' can be obtained under the new name '.ORG.'

2) 'OPDEF'

The 'OPDEF' pseudo-operation provides a means of defining new op-codes for use within an assembly. The 'OPDEF' operation consists of...

1. An op-code mnemonic in the label field,
2. Blanks in the variant field,
3. 'OPDEF' in the op-code field,
4. A number of up to ten digits in the address field.

The 'OPDEF' operation defines the mnemonic (including the blank mnemonic) in the label field to be the operation specified in the address field. For the format of this specification, see Appendix A.
NOTE...

'OPDEF' AND 'OPSYN' BOTH USE THE OPDEF TABLE FOR ENTRY OF NEW
OP-CODE MNEMONICS. SINCE THE CAPACITY OF THIS TABLE IS 100 UNIQUE
MNEMONICS, THE MAXIMUM NUMBER OF DIFFERENT MNEMONICS ADDED IS LIMITED
TO THIS FIGURE. THE OPDEF TABLE IS RESET ONLY WHEN 'SNAP' IS RELOADED,
BUT NOT BETWEEN PROGRAMS IN A BATCH ASSEMBLY. THIS ALLOWS A STANDARD
'PACKAGE' OF OP-CODES TO BE ADDED WHICH WILL APPLY TO ALL THE PROGRAMS
OF THE BATCH. (THE 'STANDARD' OP-CODES ARE RESET BETWEEN PROGRAMS AND
BETWEEN PASSES UNLESS 'SNAP' IS IN THE PAPER-TAPE-ONLY MODE.)

3) 'MACRO'

THE 'MACRO' PSEUDO-OPERATION IS NOT OPERATIVE INDEPENDENT OF THE
MACRO-EXPANDER.
E. DATA DEFINING PSEUDO-OPERATIONS

1) 'ALPH'

The 'ALPH' PSEUDO-OPERATION GENERATES FROM ONE TO SIX ALPHAMERIC WORDS. THE 'ALPH' OPERATION CONSISTS OF...

1. A SYMBOL OR BLANKS IN THE LABEL FIELD,
2. A SINGLE DIGIT BETWEEN 1 AND 6 OR A BLANK IN THE LEAST SIGNIFICANT POSITION OF THE VARIANT FIELD,
3. 'ALPH' IN THE OP-CODE FIELD,
4. ANY ALPHAMERIC CHARACTERS IN THE VARIABLE FIELD (INCLUDING CARRIAGE CONTROL CHARACTERS). COMMENTS NEED NOT BE PRECEDED BY A BLANK.

IF THE VARIANT FIELD OF AN 'ALPH' STATEMENT IS LEFT BLANK, A LENGTH OF ONE WILL BE ASSUMED. IF A LENGTH GREATER THAN SIX WORDS IS SPECIFIED, SIX WILL BE ASSUMED AND AN ERROR DETECTED. ANY CHARACTER IS LEGAL IN THE VARIABLE FIELD OF AN 'ALPH' EXCEPT DOLLAR SIGN ($), APOSTROPHE ('), AND COMMA (,). THESE CHARACTERS MAY BE GENERATED BY A 'DEC' WITH A SIGN OF 2 IF NEEDED.)

TO CLARIFY THE LISTING, ONLY THE FIRST WORD GENERATED BY EACH 'ALPH' STATEMENT IS SHOWN IN THE 'INSTRUCTION' COLUMN, AND CARRIAGE CONTROL CHARACTERS ARE TRANSLATED INTO ORDINARY PRINTABLE CHARACTERS FOR VERIFICATION OF THE CONTENTS OF THE VARIABLE FIELD. THE TRANSLATION IS AS FOLLOWS...

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>MULTIPLE PUNCH</th>
<th>LISTED AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARRIAGE RETURN</td>
<td>11-8-6</td>
<td>PHI</td>
</tr>
<tr>
<td>TABULATE</td>
<td>0-8-6</td>
<td>THETA</td>
</tr>
<tr>
<td>EJECT CODE</td>
<td>11-8-5</td>
<td>COMMA</td>
</tr>
</tbody>
</table>

2) 'DEC'

The 'DEC' PSEUDO-OPERATION DEFINES A SINGLE FIXED OR FLOATING POINT DECIMAL CONSTANT HAVING ANY SPECIFIED SIGN. THE 'DEC' OPERATION CONSISTS OF...

1. A SYMBOL OR BLANKS IN THE LABEL FIELD,
2. A SIGN OR BLANKS IN THE VARIANT FIELD,
3. 'DEC' IN THE OP-CODE FIELD,
4. A FIXED OR FLOATING POINT NUMBER (SEE BELOW) IN THE ADDRESS FIELD.
THE NUMBER IN THE ADDRESS FIELD MUST HAVE ITS LITERAL VALUE AND
MAY NOT BE AN EXPRESSION. IF THE NUMBER CONTAINS A DECIMAL POINT IT
IS CONSIDERED TO BE A FLOATING POINT NUMBER, OTHERWISE IT IS ASSUMED TO
BE FIXED POINT. THE FLOATING POINT CONVERSION ROUTINE STRIPS THE
EIGHT MOST SIGNIFICANT DIGITS (PRECEDING THE FIRST BLANK) FROM THE
ADDRESS FIELD, IGNORING NON-NUMERIC CHARACTERS. THE EXPONENT IS THEN
CALCULATED FROM THE RELATIVE POSITIONS OF THE DECIMAL POINT AND THE
FIRST SIGNIFICANT DIGIT. THE FIXED POINT CONVERSION ROUTINE STRIPS THE
TEN MOST SIGNIFICANT DIGITS AND JUSTIFIES THE NUMBER SO THAT THE LEAST
SIGNIFICANT DIGIT STRIPPED BECOMES THE \((0,1)\) DIGIT OF THE GENERATED
WORD. A PRECEDING PLUS OR MINUS SIGN IS INTERPRETED BY EITHER
CONVERTER AS A SIGN OF ZERO OR ONE, RESPECTIVELY. IF A SIGN DIGIT
OTHER THAN ZERO OR ONE IS DESIRED, IT MAY BE SPECIFIED IN THE VARIANT
FIELD SIGN POSITION.

NOTE...

A FLOATING POINT 'ZERO' DOES NOT RESULT IN THE GENERATION OF A
TRUE ZERO WORD, SINCE A NON-ZERO EXPONENT IS GENERATED.

SOME EXAMPLES OF THE USE OF 'DEC' ARE GIVEN BELOW...

<table>
<thead>
<tr>
<th>THE STATEMENT...</th>
<th>ASSEMBLES AS...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC 3.14159</td>
<td>0 5131415900</td>
</tr>
<tr>
<td>DEC -3.14159</td>
<td>1 5131415900</td>
</tr>
<tr>
<td>DEC 3.14159</td>
<td>3 5131415900</td>
</tr>
<tr>
<td>DEC -50</td>
<td>1 0000000050</td>
</tr>
<tr>
<td>DEC -3331</td>
<td>1 0000003331</td>
</tr>
<tr>
<td>DEC 0</td>
<td>0 0000000000</td>
</tr>
<tr>
<td>DEC -</td>
<td>1 0000000000</td>
</tr>
<tr>
<td>DEC 5</td>
<td>5 0000000000</td>
</tr>
<tr>
<td>DEC 0.0</td>
<td>0 5200000000</td>
</tr>
<tr>
<td>DEC 000000.</td>
<td>0 5800000000</td>
</tr>
<tr>
<td>DEC .000057689</td>
<td>0 4657689000</td>
</tr>
<tr>
<td>DEC 0202020216</td>
<td>2 0202020216 (ILLEGAL)</td>
</tr>
</tbody>
</table>

WARNING-- IT IS POSSIBLE TO GENERATE WORDS WITH A '2' IN THE SIGN
POSITION AND ARBITRARY DIGITS. WHEN ALPHAMERIC TRANSLATION
OF SUCH A WORD IS ATTEMPTED DURING PUNCHING, IT MAY RESULT IN
THE PRODUCTION OF ILLEGAL ALPHAMERIC CHARACTERS OR IN NON-
PUNCHABLE CHARACTERS (SUCH AS IGNORE CODES (02)) WHICH WILL
CAUSE INCORRECT READ-IN OF THE WORDS INVOLVED DURING OBJECT
PROGRAM LOADING.
3) 'ZZZ'

THE 'ZZZ' PSEUDO-OPERATION GENERATES THE 'ZERO CREATOR' WORD IN NUMERIC ASSEMBLER LANGUAGE. THE 'ZZZ' OPERATION CONSISTS OF...

1. BLANKS IN THE LABEL FIELD,
2. THE LENGTH OF THE NULL SECTION RIGHT-ADJUSTED IN THE VARIANT FIELD,
3. 'ZZZ' IN THE OP-CODE FIELD,
4. AN EXPRESSION (CONTAINING SYMBOLS DEFINED PHYSICALLY EARLIER IN THE SOURCE PROGRAM) IN THE ADDRESS FIELD.

THE ADDRESS FIELD EXPRESSION IS EVALUATED MODULO 10000 AND THE LEAST SIGNIFICANT TWO DIGITS (AA) ARE PLACED IN THE (0,2) FIELD OF THE GENERATED WORD...

9 LLLL 90 90AA
F. SUBROUTINE RELATED PSEUDO-OPERATIONS

1) 'CALL'

The 'CALL' pseudo-operation is a single-statement generator of the standard two-instruction linkage to a SPAC subroutine. The 'CALL' operation consists of...

1. A symbol or blanks in the label field,
2. A sign and variant or blanks in the variant field,
3. 'CALL' in the op-code field,
4. An expression in the address field.

The 'CALL' statement...

LABEL-1234CALL    SUBR

is exactly equivalent to the 'ABSMOD' sequence...

LABEL-1234STP    SUBR
-1234BUN    SUBR+1

Which is the standard linkage for most SPAC subroutines. As the example implies, the signs and variants of the generated instructions are identical and specified by the variant field of the 'CALL'. The symbol in the label field, if any, refers to the first ('STP') instruction of the calling sequence. If different variants are required in the 'STP' and 'BUN' instructions, 'CALL' cannot be used, but the individual instructions must be written out. Other calling parameters required by the subroutine may be written as 'DEC' or 'HLT' statements after the 'CALL' statement.

The 'CALL' pseudo-operation may be used in a 'RELMOD' program, but it will refer to absolute addresses. This is consistent with the necessity of assigning absolute locations to SPAC subroutines. If it is necessary to link to a subroutine written into the 'RELMOD' program, 'CALL' should not be used, but instead the 'STP'-'BUN' sequence should be written out.

Note...

When the 'SELF' symbol, asterisk (*), is used in the address field of a 'CALL' statement, it refers to the first instruction ('STP') generated by the 'CALL'.

2) 'SPAC'

THE 'SPAC' PSEUDO-OPERATION CAUSES A SPAC SUBROUTINE TO BE CALLED INTO STORAGE DURING LOADING OF THE OBJECT TAPE, AND ALSO ASSIGN AN INTERNAL NAME TO THE SUBROUTINE SO THAT IT MAY BE REFERENCED SYMBOLICALLY. THE 'SPAC' OPERATION CONSISTS OF...

1. A Symbol or Blanks in the Label Field,
2. The Number of Locations to Be Reserved for the Subroutine Right-Adjusted in the Variant Field,
3. 'SPAC' in the Op-Code Field,
4. An Expression (Containing Symbols Defined Physically Earlier in the Source Program) in the Address Field.


THE 'SPAC' PSEUDO-OP IS NOT ALLOWED IN A 'RELMOD' ASSEMBLY. THAT IS, A RELOCATABLE SUBROUTINE IS NOT ALLOWED TO CALL-IN RELOCATABLE SUBROUTINES. OF COURSE, A DUMMY 'ABSMOD' SECTION MAY PRECEDE A 'RELMOD' PROGRAM IF IT IS NECESSARY TO GENERATE AN ABSOLUTE SPAC CALL-IN SEQUENCE.

NOTE...

SINCE THE CALL-IN SEQUENCE PUTS THE SPAC CONTROL PROGRAM INTO LOCATIONS 0000 TO 0400, SUBROUTINES MAY NOT BE CALLED INTO THIS REGION. A GOOD PROCEDURE IN SHORT PROGRAMS IS TO USE THE 'SPAC' OPERATION UNDER A REMOTE LOCATION COUNTER ORIGIN. FOR EXAMPLE...

```
.SAVE     SET       *       SAVE LOCATION COUNTER
.ORG 2000 REMOTE ORIGIN..
SQRT 25SPAC 1   SQRT = SPAC 01
PLOT 230SPAC 28  PLOT = SPAC 28
LOAD  .SAVE  RESTORE LOAD ADDRESS.
```

THE SPAC SUBROUTINE CALL-IN GENERATED BY 'SNAP' DOES NOT REQUIRE ANY CONTROL SWITCH SETTINGS.
G. LISTING CONTROL PSEUDO-OPERATIONS

1) 'REM'
   THIS OPERATION IS USED TO INSERT REMARKS IN THE LISTING. THE 'REM' PSEUDO-OPERATION CONSISTS OF...
   
   1. ANY ALPHAMERIC CHARACTERS IN THE LABEL FIELD AND THE VARIANT FIELD,
   2. 'REM' IN THE OP-CODE FIELD,
   3. ANY ALPHAMERIC CHARACTERS IN COL. 21-25 AND THE VARIABLE FIELD.

   THE CARD IMAGE, WITH THE EXCEPTION OF THE OP-CODE FIELD, WILL BE LISTED.

   NOTE...
   REMARKS MAY ALSO BE INSERTED BY PUTTING AN ASTERISK (*) IN COLUMN 6. THE REST OF THE CARD (COL. 7-55) IS THEN AVAILABLE FOR COMMENTS.

2) 'SPACE'
   THE 'SPACE' PSEUDO-OPERATION CAUSES BLANK LINES TO APPEAR IN THE LISTING. THE 'SPACE' OPERATION CONSISTS OF...
   
   1. BLANKS IN THE LABEL AND VARIANT FIELDS,
   2. 'SPACE' IN THE OP-CODE FIELD,
   3. AN EXPRESSION HAVING VALUE LESS THAN 50 IN THE ADDRESS FIELD.

   THE NUMBER OF BLANK LINES INSERTED IS EQUAL TO THE VALUE OF THE EXPRESSION, WITH THE FOLLOWING EXCEPTIONS...
   
   1) IF THE NUMBER OF BLANK LINES SPECIFIED IS GREATER THAN THE NUMBER OF LINES REMAINING ON THE PAGE, 'EJECT' IS SIMULATED.
   2) IF THE VALUE OF THE ADDRESS FIELD EXPRESSION IS ZERO, ONE BLANK LINE IS INSERTED.

3) 'EJECT'
   THE 'EJECT' PSEUDO-OPERATION CAUSES THE PRESENT PAGE OF LISTING TO BE FILLED OUT WITH BLANK LINES SO THAT THE NEXT STATEMENT APPEARS AT THE TOP OF THE NEXT PAGE. THE 'EJECT' OPERATION CONSISTS OF...
   
   1. BLANKS IN THE LABEL AND VARIANT FIELDS,
   2. 'EJECT' IN THE OP-CODE FIELD,
   3. BLANKS IN THE VARIABLE FIELD.

   THE 'EJECT' STATEMENT DOES NOT APPEAR ON THE LISTING.
IV. DIAGNOSTIC MESSAGES

GENERAL--

WHEN AN ERROR IN THE SOURCE PROGRAM IS DETECTED DURING ASSEMBLY, THE TYPE OF ERROR WILL BE PRINTED IN CODED FORM ON THE TELETYPE PRINTER (E.G. ‘ER S1’) FOLLOWED BY THE CARD IMAGE CONTAINING THE ERROR. IN PASS II THE LISTING IS ALSO FLAGGED WITH THE ERROR CODE, IMMEDIATELY TO THE LEFT OF THE LABEL FIELD. ERROR PRINTING ON THE TELETYPE PRINTER MAY BE SUPPRESSED BY TURNING SWITCH 8 ON. (THIS IS NOT RECOMMENDED UNLESS A CATASTROPHIC SNOW-BALLING OF ERROR TYPEOUT OCCURS.)

A. SYMBOL RELATED ERRORS

1) S1 --UNDEFINED SYMBOL

   THIS ERROR IS DETECTED WHENEVER A REFERENCE IS MADE TO A SYMBOL WHICH IS NOT DEFINED AT THE TIME OF THE REFERENCE.

   DURING PASS I IT IS CAUSED BY PSEUDO-OPS WHICH MUST CALCULATE THE VALUE OF AN EXPRESSION CONTAINING SYMBOLS WHICH HAVE NOT AS YET ENTERED THE SYMBOL TABLE. SUCH PSEUDO-OPS ARE...

<table>
<thead>
<tr>
<th>BES</th>
<th>BSS</th>
<th>EQU</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>OPDEF</td>
<td>ORG</td>
</tr>
<tr>
<td>SET</td>
<td>SEX</td>
<td>SPAC</td>
</tr>
</tbody>
</table>

   SYMBOLS USED IN THE ADDRESS FIELD EXPRESSIONS OF THESE PSEUDO-OPS MUST HAVE BEEN DEFINED PHYSICALLY EARLIER IN THE SOURCE PROGRAM. AN S1 ERROR IN PASS I CAUSES PASS II TO BE ABANDONED (UNLESS A VALUE FOR THE EXPRESSION IS MANUALLY ASSIGNED AS DESCRIBED BELOW).

   ALL SYMBOL TABLE ENTRIES HAVE BEEN MADE BEFORE THE START OF PASS II, THEREFORE, AN S1 ERROR IN THE SECOND PASS INDICATES THAT A REFERENCE HAS BEEN MADE TO A SYMBOL NOT DEFINED ANYWHERE IN THE SOURCE PROGRAM.

   THE VALUE OF AN EXPRESSION IN WHICH AN UNDEFINED SYMBOL APPEARS WILL NORMALLY BE SET EQUAL TO ZERO. HOWEVER, A SENSE SWITCH OPTION (SWITCH 7) IS PROVIDED WHICH WILL CAUSE THE COMPUTER TO COME TO A HALT WITH THE C-REGISTER = 7777 00 7777 WHEN AN S1 ERROR OCCURS. AT THIS TIME THE OPERATOR MAY CLEAR THE ACCUMULATOR (WHICH CONTAINS THE NUMERIC REPRESENTATION OF THE UNDEFINED SYMBOL) AND INSERT IN THE 0,4 FIELD THE PROPER VALUE FOR THE ENTIRE EXPRESSION, AS DETERMINED FROM THE DICTIONARY LISTING AND THE PRESENT VALUE OF THE LOCATION COUNTER.
2) S2 --MULTIPLY-DEFINED SYMBOL

IF THE PROGRAMMER ATTEMPTS TO DEFINE THE SAME SYMBOL MORE THAN ONCE IN ANY OPERATION OTHER THAN 'SET', ERROR CODE S2 IS PRINTED. THE SYMBOL RETAINS ITS ORIGINAL VALUE BUT IS FLAGGED WITH '*M*' IN THE SYMBOL DICTIONARY. ONLY THE 'SET' PSEUDO-OPERATION IS ABLE TO ASSIGN A NEW VALUE TO A DEFINED SYMBOL, AND THEN ONLY IF THE SYMBOL WAS ORIGINALLY DEFINED BY A 'SET' OPERATION.

3) S3 --SYMBOL TABLE FULL

THE CAPACITY OF THE 'SNAP' SYMBOL TABLE IS 1535 SYMBOLS. IF THE PROGRAMMER ATTEMPTS TO USE MORE SYMBOLS, 'ER S3' WILL BE PRINTED, ALONG WITH THE CARD IMAGE CONTAINING THE 1536TH LABEL, AND THE ASSEMBLY WILL BE TERMINATED IMMEDIATELY. THE ONLY REMEDY IS TO RE-WRITE THE PROGRAM USING FEWER SYMBOLS.

4) S4 --SYMBOL MISSING FROM LABEL FIELD OF 'EQU'

SINCE THE 'EQU' PSEUDO-OP EXISTS FOR THE PURPOSE OF DEFINING SYMBOLS, AN 'EQU' WITH A BLANK LABEL FIELD IS DIAGNOSED AS AN ERRDR. SUCH A STATEMENT WILL BE TREATED AS A REMARK.

5) S5 --NON-'SET' SYMBOL IN LABEL FIELD OF 'SET'

IF AN ATTEMPT IS MADE TO REDEFINE A SYMBOL WHICH FIRST APPEARED IN THE LABEL FIELD OF ANY LABEL-DEFINING OPERATION EXCEPT 'SET' THE USE OF A 'SET' OPERATION, ERROR S5 WILL BE DETECTED AND THE VALUE OF THE SYMBOL IS LEFT UNCHANGED. THE SYMBOL DICTIONARY ENTRY IS NOT FLAGGED. A BLANK LABEL FIELD IN A 'SET' STATEMENT WILL ALSO RESULT IN ERROR S5.

B. OPERATION CODE RELATED ERRORS

1) O1 --UNDEFINED OPERATION CODE

IN EITHER PASS, AN UNDEFINED OPERATION CODE RESULTS IN THE ASSUMPTION OF A 'HLT' COMMAND. OPERATION CODES DEFINED BY 'OPDEF' OR 'OPSYN' (OR BY 'MACRO' DEFINITIONS) MUST BE DEFINED PHYSICALLY PRIOR TO THEIR APPEARANCE IN THE OP-CODE FIELD. A FREQUENT CAUSE OF O1 ERRORS IS MISSPELLING OF OP-CODES.

2) O2 --OPDEF TABLE FULL

THE PSEUDO-OPERATIONS 'OPDEF', 'OPSYN', AND 'MACRO' WILL INSERT A NEW ENTRY INTO THE OPDEF TABLE IF THE MNEMONIC BEING DEFINED IS NOT ALREADY DEFINED TO BE AN OP-CODE MNEMONIC. SINCE THE MAXIMUM CAPACITY OF THE OPDEF TABLE IN 'SNAP' IS 100 ENTRIES, ANY ATTEMPT TO ENTER A 101ST OP-CODE NAME WILL RESULT IN AN O2 ERROR. THIS ERROR WILL NOT TERMINATE ASSEMBLY, BUT IT MAY, OF COURSE, RESULT IN SUBSEQUENT O1 ERRORS.
3) O3 --OPERATION CODE ILLEGAL IN 'RELMOD' ASSEMBLY

   THE STATEMENT CAUSING THE ERROR IS ASSEMBLED AS IF IT WERE IN AN 'ABSMOD' ASSEMBLY. ASSEMBLY IS NOT TERMINATED.

   CERTAIN OPERATION CODES MAY NOT BE USED AT ALL IN 'RELMOD' ASSEMBLIES. NUMERIC ASSEMBLER PSEUDO-OPS ('AUX','PSR','SEX', ETC.) ARE EXAMPLES OF THIS TYPE.

   TWO OTHER OPERATION CODES, SPECIFYING MACHINE OPERATIONS, ARE NOT USEABLE IN THE NORMAL WAY IN 'RELMOD' ASSEMBLIES. THESE CODES ARE 'MFC' AND 'MFS', WHICH REQUIRE THE 4-BIT OF THE SIGN TO BE ON AS A PART OF THE OP-CODE. SINCE 'FLOATING ADDRESS' LOADING CAN ONLY MODIFY THE ADDRESSES OF INSTRUCTIONS WITH 8-BITS ON, AND SUBSEQUENT TO MODIFICATION REDUCES THE SIGN TO 0 OR 1, THE INCOMPATIBILITY OF 'MFC' AND 'MFS' IS CLEARLY SEEN. THE PROGRAMMER MUST OBTAIN THE RESULT HE SEeks BY PROGRAM MODIFICATION AT EXECUTION TIME. OF COURSE THERE IS NO SUCH INCOMPATIBILITY IN AN 'ABSMOD' ASSEMBLY.

4) O4 --'SPAC' USED IN 'RELMOD' ASSEMBLY

   THIS ERROR CAN ONLY OCCUR IN PASS I. SPAC SUBROUTINES CANNOT BE CALLED INTO MEMORY BY A RELOCATABLE PROGRAM. THE 'SPAC' STATEMENT CAUSING THE ERROR WILL BE TREATED AS A REMARK AND PASS II WILL BE ABANDONED. THE ERROR SCAN IN PASS I WILL CONTINUE. IF NECESSARY, A DUMMY 'ABSMOD' SECTION MAY BE INTRODUCED AHEAD OF A 'RELMOD' PROGRAM FOR THE PURPOSE OF GENERATING AN (ABSOLUTE) SPAC CALL-IN SEQUENCE.

C. MISCELLANEOUS ERRORS

1) V1 --ILLEGAL SIGN OR VARIANT


   \[ \text{STO} \quad \text{OPDEF} \quad 0621040100 \]

   NOW SUPPOSE IT IS USED IN THE FOLLOWING WAY...

   \[ 96210 \text{STO} \quad \text{ALPHA}+3 \]

   WHEN THE SIGN-VARIANT COMBINATION IS NOW ADDED TO THE OP-CODE Prototype, OVERFLOW RESULTS...

   \[ 0 \ 0621040100 + 0 \ 9621000000 = 0 \ 0242040100 \quad \text{(OVERFLOW)} \]

   THE INSTRUCTION WOULD THEN BE FLAGGED 'V1' AND ASSEMBLED AS...

   \[ 0 \ 242040AAAA \quad \text{(WHERE AAAA = ALPHA}+3) \]
IT SHOULD BE NOTICED THAT THE RESULTANT SIGN AFTER SUCH AN ERROR CAN BE 2. WHEN SUCH AN 'INSTRUCTION' IS PUNCHED OUT, ALPHAMERIC TRANSLATION WILL BE ATTEMPTED, RESULTING IN SOME UNIQUE AND PROBABLY ILLEGAL ALPHAMERIC CHARACTERS.

2) V2 --ILLEGAL SIGN
   THIS ERROR, LIKE V1, OCCURS ONLY DURING PASS II, WHEN OVERFLOW IS SENSED AS A RESULT OF ADDING IN THE SIGN OF 8 ASSOCIATED WITH 'RELMOD' ASSEMBLY. OBVIOUSLY, AN INSTRUCTION TO BE ASSEMBLED IN RELOCATABLE FORM CANNOT HAVE A SOURCE LANGUAGE SIGN OTHER THAN 0 OR 1.

3) V3 --'ALPH' LENGTH GREATER THAN SIX SPECIFIED
   A SINGLE 'ALPH' CAN DEFINE FROM ONE TO SIX ALPHAMERIC WORDS. IF A VARIANT DIGIT (LENGTH) GREATER THAN 6 IS GIVEN, 6 WILL BE ASSUMED AND ERROR V3 DETECTED. THIS ERROR WILL BE DETECTED IN BOTH PASS I AND PASS II. IT IS A WARNING ONLY AND DOES NOT TERMINATE ASSEMBLY.

4) T1 --INSUFFICIENT MAGNETIC TAPE
   THIS ERROR CAN OCCUR ONLY DURING PASS I WHEN MAGNETIC TAPE IS BEING WRITTEN. ONLY AN EXTREMELY LONG PROGRAM (GREATER THAN 5000 CARDS) CAN EVER CAUSE THIS ERROR UNDER NORMAL CIRCUMSTANCES. SINCE THE AMOUNT OF MAG-TAPE ON THE REEL IS USUALLY BEYOND THE CONTROL OF THE PROGRAMMER, THE ONLY SOLUTION IS TO RE-ASSEMBLE USING THE PAPER-TAPE-ONLY OPTION (SWITCH 0). (BEAR IN MIND THAT IN THIS MODE THE OPERATION CODE TABLE IS NOT RESET, EITHER BETWEEN PASSES OR BETWEEN BATCH ASSEMBLIES.)

5) (NO CODE) --OVERFLOW HALT
   A COMPUTER OVERFLOW HALT CAN OCCUR IF THE LOCATION COUNTER IS INCREMENTED PAST 9999. ASSEMBLY MAY BE CONTINUED (IF SUCH A CONDITION IS ALLOWABLE) BY RESETTING THE OVERFLOW INDICATOR AND PESSING START. THE LOCATION COUNTER WILL CONTINUE TO COUNT, MODULO 10000.

6) L1 --SPAC SUBROUTINE CALLED INTO LOCATION BELOW 0400
   SINCE THE SPAC CONTROL PROGRAM OCCUPIES 0000-0399 AT LOAD TIME, SPAC SUBROUTINES MAY NOT OCCUPY THIS REGION. PASS I WILL CONTINUE, BUT PASS II WILL BE ABANDONED. SEE SECTION III, F, PART 2, FOR A SUGGESTION AND EXAMPLE REGARDING SPAC SUBROUTINES IN SHORT PROGRAMS.
V. PROGRAM EXAMPLE

AN EXAMPLE OF A 'SNAP' LANGUAGE PROGRAM IS GIVEN BELOW. THE LISTING IS IN
EXACTLY THE SAME FORM AS THE LISTING PRODUCED BY 'SNAP'. THE EXAMPLE IS A
RELOCATABLE SUBROUTINE SUITABLE FOR WRITING TEN-WORD RECORDS ON MAGNETIC-TAPE.
(THIS SUBROUTINE IS OF PARTICULAR INTEREST SINCE IT IS VERY SIMILAR TO THE
'WRITE' SUBROUTINE IN 'SNAP' ITSELF.)

NOTE THAT THE VALUE OF THE LOCATION COUNTER IS PRINTED IN THE 'ADRES'
COLUMN AND THE ASSEMBLED TEXT IN THE 'INSTRUCTION' COLUMN. THE SPACES BETWEEN
THE 'INSTRUCTION' COLUMN AND THE 'LABEL' COLUMN ARE USED FOR ERROR FLAGS. THE
COMPLETE CARD IMAGE (COL. 6-55) APPEARS TO THE RIGHT OF THE ERROR FLAG FIELD.

S N A P VER. 1E 5/10/64

SWITCH OPTIONS MUST BE SET DURING O-O HALT.

SW 5 ON IF INPUT IS ON MAG-TAPE
SW 6 ON SUPPRESSES PUNCHING
SW 7 ON TO INSERT VALUE IN AC FOR UNDEFINED ADRES
SW 8 ON SUPPRESSES ERROR TYPING
SW 9 ON SUPPRESSES DICTIONARY PRINTING
SW 0 ON INDICATES PAPER-TAPE-ONLY MODE

DICTIONARY

<table>
<thead>
<tr>
<th>LABEL VALUE</th>
<th>LABEL VALUE</th>
<th>LABEL VALUE</th>
<th>LABEL VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>.WRIT 0009</td>
<td>BTGO 0017</td>
<td>BTOTL 0022</td>
<td>CLOSE 0015</td>
</tr>
<tr>
<td>EOT 0030</td>
<td>EOTMS 0033</td>
<td>BUF 0036</td>
<td>OPEN 0003</td>
</tr>
<tr>
<td>WRITE 0000</td>
<td>WRTRC 0026</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* WRITE --MAGNETIC TAPE WRITE SUBROUTINE
* WRITES 10 WRD RECORDS IN 100 WRD BLKS
* BLOCKS TAPE AS NEEDED
* (TAPE MUST BE EDITED UPON LOADING.)
*
* ENTRY AT WRITE+1 FOR NORMAL WRITE.
* ENTRY AT WRITE+2 TO CLOSE OUT THE FILE.
* INITIAL ENTRY AT WRITE+3 TO OPEN FILE.
* ENTER WITH ADDRESS OF FIRST WORD OF RECORD IN
* B-REGISTER. UPON EXIT NEXT AVAILABLE ADDRESS
* IN BUFFER IS LEFT IN B-REG.

0000 0 0000000000 REL SET RELMOD
0000 7 0008040000 LOAD 0
0000 8 0009300000 WRITE 9BUN ** RETURN
0001 8 0000300009 BUN .WRIT WRITE A RECORD
0002 8 0000300015 BUN CLOSE CLOSE FILE...
0003 8 1008500036 OPEN 1000MRW BUF PREPARE TAPE 1
0004 8 0000420022 LDB BTOTL SET BTOTL
0005 8 0412400017 0410STB BTOTO = BTOTL
0006 8 0000100000 CAD WRITE GET =9
0007 8 4410400014 4410STA CLOSE-1 SET TALY=9...
0008 8 0000300027 BUN WRTRC+1 AND GO ON....
0009 8 0412400011 .WRIT 0410STB +2 SET RECRD ADDR.
0010 8 0000420003 LDB OPEN LOAD BUF ADDR..
0011 8 0100290000 100RTF ** MOVE 10 WORDS
0012 8 0412400003 0410STB OPEN HOLD BUF ADDR..
0013 8 4101270014 4101DFL +1 BUMP TALLY.
0014 8 0009320000 9BRP WRITE IS IT TENTH....
0015 8 0410270017 CLOSE 0410DFL BTOTO ANY BLOCKS LEFT
0016 8 0000320026 BRP WRTRC YES. WRITE...
0000 0 0000000000 ABS (WANT ABSOLUTE ADDRESS)
0017 0 1002580000 BTOTO 1000MPE NO. SET AND
0000 0 0000000000 REL
0000 8 1005400036 1000MIW BUF WRITE 10 BLOCKS
0000 8 1000590021 1000MIB +2 DELAY UNTIL
0020 8 0000300019 BUN *-1 TAPE STOPS...
0021 8 1001590030 1000MIE EOT END OF TAPE ERR
0000 0 0000000000 ABS (WANT ABSOLUTE ADDRESS)
0000 8 1001580000 BTOTL 1000MPB 0 BACKSPACE 10...
0000 0 0000000000 REL
0000 8 0410260017 0410IFL BTOTO BUMP BTOTO 10..
0024 8 0000310025 BOF +1 RESET OVFLOW...
0025 8 0410260022 0410IFL BTOTL BUMP BTOTL 10.
SINCE THE COMMENTS MAKE THE PROGRAM MORE OR LESS SELF EXPLANATORY, WE WILL LOOK INTO THE DETAILS OF ASSEMBLER CONTROL WITHIN THE PROGRAM. IN THE FIRST PLACE, NOTE THAT THE ASSEMBLY IS 'RELMOD', SINCE A RELOCATABLE OBJECT PROGRAM IS DESIRED. THIS MEANS THAT 'SNAP', AFTER FINDING A PARTICULAR OP-CODE PROTOTYPE IN THE OP-TABLE AND ASCERTAINING THAT IT IS INDEED LEGAL IN 'RELMOD', WILL PROCEED TO ADD ON THE SIGN AND VARIANT SPECIFIED IN THE SOURCE PROGRAM. THEN, IF THE ADDRESS OF THE INSTRUCTION IS NOT SPECIFIED BY THE PROTOTYPE TO BE UNCONDITIONALLY ABSOLUTE (AS IN A SHIFT COMMAND, FOR EXAMPLE), 'SNAP' WILL ADD 8 TO THE SIGN DIGIT. THUS IT IS NECESSARY TO MOMENTARILY ENTER 'ABSMOD' FOUR TIMES DURING ASSEMBLY, THREE TIMES TO PRODUCE ABSOLUTE ADDRESSES AND ONCE TO ALLOW SPECIFICATION OF A LITERAL SIGN NOT EQUAL TO 8 OR 9. (OF COURSE, AN ALTERNATE METHOD WHICH THE PROGRAMMER MAY FIND USEFUL IN SOME CASES IS TO REDEFINE, SAY, THE 'HLT' OPERATION TO HAVE AN ABSOLUTE ADDRESS, AND DEFINING ANOTHER OP-CODE TO MEAN 'ZERO OP-CODE WITH RELATIVE ADDRESS'.)

NOTE ALSO THE USE OF ** (LITERALLY 'SELF' TIMES ZERO) IN THE CONVENTIONAL WAY TO INDICATE AN ADDRESS TO BE FILLED IN AT EXECUTION TIME. OF COURSE A ZERO ADDRESS MAY BE OBTAINED IN MANY WAYS, THE MOST COMMON BEING AN EXPLICIT ZERO IN THE ADDRESS FIELD OR SIMPLY A 'NULL' (BLANK) ADDRESS FIELD, OBTAINED BY LEAVING (AT LEAST) COLUMN 26 BLANK. EXAMPLES OF BOTH METHODS ARE SHOWN.

THE INITIAL CHARACTER IN THE ADDRESS FIELD OF THE 'ALPH' STATEMENT AT LOCATION 0033 WAS AN 11-8-6 PUNCH (CARRIAGE RETURN) WHICH WAS TRANSLATED INTO 'PHI' (REPRESENTED BY A $, SINCE 'PHI' CANNOT BE LISTED IN THIS MANUAL).
APPENDIX A

NOTE ON FORMAT OF ENTRIES IN OP-TABLE

THERE ARE TWO FORMATS, THE MACHINE-OPERATION FORMAT (INDICATED BY A SIGN OF 0) AND THE PSEUDO-OPERATION FORMAT (INDICATED BY A SIGN OF 1).

MACHINE-OPERATION FORMAT...

0 SVVVVXXABC

S = SIGN OF UNMODIFIED INSTRUCTION
VVVV = VARIANT OF UNMODIFIED INSTRUCTION
XX = NUMERIC OP-CODE OF INSTRUCTION
A = '(4,1) DIGIT FIXED' FLAG
   A=1 IMPLIES (4,1) DIGIT MAY NOT BE ALTERED
   A=0 IMPLIES (4,1) DIGIT MAY BE ALTERED
B = 'ILLEGAL IN RELMOD' FLAG
   B=1 IMPLIES ILLEGAL IN 'RELMOD'
   B=0 IMPLIES O.K. IN 'RELMOD'
C = 'ABSOLUTE ADDRESS' FLAG
   C=1 IMPLIES ADDRESS IS ABSOLUTE EVEN IN 'RELMOD'
   C=0 IMPLIES ADDRESS IS RELATIVE IN 'RELMOD'

PSEUDO-OPERATION FORMAT...

1 00AA00BBBB

BBBB = PASS II ENTRY POINT OF PSEUDO-OP SUBROUTINE
AA = PASS I ENTRY POINT (RELATIVE TO BBBB) OF PSEUDO-OP SUBROUTINE

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>OPERATION</th>
<th>TYPE</th>
<th>ENTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BLANK)</td>
<td>HALT</td>
<td>M</td>
<td>0 0000000000</td>
</tr>
<tr>
<td>ABS</td>
<td>SET 'ABSMOD'</td>
<td>P</td>
<td>1 0002000655</td>
</tr>
<tr>
<td>ADA</td>
<td>ADD ABSOLUTE</td>
<td>M</td>
<td>0 0000112100</td>
</tr>
<tr>
<td>ADD</td>
<td>ADD</td>
<td>M</td>
<td>0 0000012100</td>
</tr>
<tr>
<td>ADL</td>
<td>ADD TO LOCATION</td>
<td>M</td>
<td>0 0000019000</td>
</tr>
<tr>
<td>ALPH</td>
<td>ALPHAMERIC INFORMATION</td>
<td>P</td>
<td>1 0024000659</td>
</tr>
<tr>
<td>AUX</td>
<td>AUXILIARY STORAGE</td>
<td>N</td>
<td>0 9000081010</td>
</tr>
<tr>
<td>BCE</td>
<td>BRANCH, COMPARE EQUAL</td>
<td>M</td>
<td>0 0000035100</td>
</tr>
<tr>
<td>BCH</td>
<td>BRANCH, COMPARE HIGH</td>
<td>M</td>
<td>0 0000034100</td>
</tr>
<tr>
<td>BCL</td>
<td>BRANCH, COMPARE LOW</td>
<td>M</td>
<td>0 0000134100</td>
</tr>
<tr>
<td>BCS</td>
<td>BRANCH, CONTROL SWITCH</td>
<td>M</td>
<td>0 0000038000</td>
</tr>
<tr>
<td>BCU</td>
<td>BRANCH, COMPARE UNEQUAL</td>
<td>M</td>
<td>0 0000135100</td>
</tr>
<tr>
<td>BES</td>
<td>BLOCK ENDED BY SYMBOL</td>
<td>P</td>
<td>1 0004000703</td>
</tr>
<tr>
<td>BFA</td>
<td>BRANCH, FIELD A</td>
<td>M</td>
<td>0 0000036000</td>
</tr>
<tr>
<td>BFR</td>
<td>BRANCH, FIELD R</td>
<td>M</td>
<td>0 0000037000</td>
</tr>
<tr>
<td>BOF</td>
<td>BRANCH, OVERFLOW</td>
<td>M</td>
<td>0 0000031000</td>
</tr>
<tr>
<td>BRO</td>
<td>BRUTE-FORCE BY OPERATOR</td>
<td>N</td>
<td>0 9000096010</td>
</tr>
</tbody>
</table>
BRP  BRANCH, REPEAT
BRU  BRUTE-FORCE IN ASSEMBLY
BRZ  BRUTE-FORCE IN EXECUTION
BSA  BRANCH, SIGN A
BSS  BLOCK STARTED BY SYMBOL
BUN  BRANCH, UNCONDITIONALLY
CAA  CLEAR, ADD ABSOLUTE
CAB  CLEAR A, B
CAD  CLEAR, ADD
CALL CALL (SUBROUTINE)
CAR  CLEAR A, R
CFA  COMPARE FIELD A
CFR  COMPARE FIELD R
CHA  CHANGE
CLA  CLEAR A
CLB  CLEAR B
CLL  CLEAR LOCATION
CLR  CLEAR R
CLT  CLEAR A, R, B (TOTAL)
CRB  CLEAR R, B
CSA  CLEAR, SUBTRACT ABSOLUTE
CSU  CLEAR, SUBTRACT
DBB  DECREASE B, BRANCH
DBL  DECREASE FIELD LOCATION
DIV  DIVIDE
DLB  DECREASE FIELD LOCATION, LOAD B
DUP  DUPLICATE
EJECT EJECT PAGE
END  END (OF PROGRAM)
EQU  EQUALS
EXT  EXTRACT
FAA  FLOATING ADD ABSOLUTE
FAD  FLOATING ADD
FDV  FLOATING DIVIDE
FMU  FLOATING MULTIPLY
FSA  FLOATING SUBTRACT ABSOLUTE
FSU  FLOATING SUBTRACT
HLT  HALT
IBB  INCREASE B, BRANCH
IFL  INCREASE FIELD LOCATION
INS  INSERT
IOM  INTERROGATE OVERFLOW MODE
KAD  KEYBOARD ADD
LBC  LOAD B COMPLEMENT
LDB  LOAD B
LDR  LOAD R
LOAD LOAD (OBJECT TAPE)
LSA  LOAD SIGN A
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Mode</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRA</td>
<td>SHIFT RIGHT A</td>
<td>M</td>
<td>0 000048101</td>
</tr>
<tr>
<td>SRS</td>
<td>SHIFT RIGHT A WITH SIGN</td>
<td>M</td>
<td>0 0000248101</td>
</tr>
<tr>
<td>SRT</td>
<td>SHIFT RIGHT A AND R (TOTAL)</td>
<td>M</td>
<td>0 0000148101</td>
</tr>
<tr>
<td>STA</td>
<td>STORE A</td>
<td>M</td>
<td>0 0000040100</td>
</tr>
<tr>
<td>STB</td>
<td>STORE B</td>
<td>M</td>
<td>0 0000240100</td>
</tr>
<tr>
<td>STP</td>
<td>STORE P</td>
<td>M</td>
<td>0 0000044000</td>
</tr>
<tr>
<td>STR</td>
<td>STORE R</td>
<td>M</td>
<td>0 0000140100</td>
</tr>
<tr>
<td>SUA</td>
<td>SUBTRACT ABSOLUTE</td>
<td>M</td>
<td>0 0000113100</td>
</tr>
<tr>
<td>SUB</td>
<td>SUBTRACT</td>
<td>M</td>
<td>0 0000013100</td>
</tr>
<tr>
<td>TCD</td>
<td>'TRANSFER CARD'</td>
<td>P</td>
<td>1 0011001030</td>
</tr>
<tr>
<td>ZZZ</td>
<td>ZERO CREATOR</td>
<td>N</td>
<td>1 0009001044</td>
</tr>
</tbody>
</table>